Internet Engineering Task Force (IETF)
Request for Comments: 7285
Category: Standards Track
ISSN: 2070-1721

R. Alimi, Ed.
Google
R. Penno, Ed.
Cisco Systems, Inc.
Y. Yang, Ed.
Yale University
S. Kiesel
University of Stuttgart
S. Previdi
Cisco Systems, Inc.
W. Roome
Alcatel-Lucent
S. Shalunov
Open Garden
R. Woundy
Comcast

September 2014

Application-Layer Traffic Optimization (ALTO) Protocol

Abstract

Applications using the Internet already have access to some 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 network application clients. What is missing is knowledge of the underlying network topologies from the point of view of ISPs. In other words, what an ISP prefers in terms of traffic optimization -- and a way to distribute it.

The Application-Layer Traffic Optimization (ALTO) services defined in this document provide 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 services. Although the ALTO services would primarily be provided by ISPs, other entities, such as content service providers, could also provide ALTO services. Applications that could use the ALTO services 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.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7285.

Copyright Notice

Table of Contents

1. Introduction

1.1. Problem Statement

This document defines the ALTO Protocol, which provides a solution for the problem stated in [RFC5693]. Specifically, in today's networks, network information such as network topologies, link availability, routing policies, and path costs are hidden from the application layer, and many applications benefited from such hiding of network complexity. However, new applications, such as application-layer overlays, can benefit from information about the underlying network infrastructure. In particular, these new network applications can be adaptive; hence, they can become more network efficient (e.g., reduce network resource consumption) and achieve better application performance (e.g., accelerated download rate), by leveraging network-provided information.

At a high level, the ALTO Protocol specified in this document is an information-publishing interface that allows a network to publish its network information such as network locations, costs between them at configurable granularities, and endhost properties to network applications. The information published by the ALTO Protocol should benefit both the network and the applications (i.e., the consumers of the information). Either the operator of the network or a third party (e.g., an information aggregator) can retrieve or derive related information of the network and publish it using the ALTO Protocol.

To allow better understanding of the goal of the ALTO Protocol, this document provides a short, non-normative overview of the benefits of ALTO to both networks and applications:

- A network that provides ALTO information can achieve better utilization of its networking infrastructure. For example, by using ALTO as a tool to interact with applications, a network 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. During the interaction, the network can choose to protect its sensitive and confidential network state information, by abstracting real metric values into non-real numerical scores or ordinal ranking.

- An application that uses ALTO information can benefit from better knowledge of the network to avoid network bottlenecks. For example, an overlay application can use information provided by the ALTO services to avoid selecting peers connected via high-delay links (e.g., some intercontinental links). Using ALTO to

initialize each node with promising ("better-than-random") peers, an adaptive peer-to-peer overlay may achieve faster, better convergence.

1.1.1. 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].

1.2. Design Overview

The ALTO Protocol specified in this document meets the ALTO requirements specified in [RFC5693], 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 (Representational State Transfer (REST)) design [Fielding-Thesis], and encodes its requests and responses using JSON [RFC7159]. 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 [RFC7230] implementations, infrastructures and deployment experience.

The ALTO Protocol uses a modular design by dividing ALTO information publication into multiple ALTO services (e.g., the Map service, the Map-Filtering Service, the Endpoint Property Service, and the Endpoint Cost Service). Each ALTO service provides a given set of functionalities and is realized by a set of information resources, which are announced by information resource directories, to guide ALTO clients.

2. Terminology

This document uses the following terms defined in [RFC5693]: Application, Overlay Network, Peer, Resource, Resource Identifier, Resource Provider, Resource Consumer, Resource Directory, Transport Address, ALTO Server, ALTO Client, ALTO Query, ALTO Response, ALTO Transaction, Local Traffic, Peering Traffic, and Transit Traffic.

This document extends the term "ALTO Service" defined in [RFC5693]. In particular, by adopting a modular design, this document allows the ALTO Protocol to provide multiple ALTO services.

This document also uses the following additional terms: Endpoint Address, 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 a resource consumer.

2.2. Endpoint Address

An endpoint address represents the communication address of an endpoint. Common forms of endpoint addresses include IP addresses, Media Access Control (MAC) addresses, and overlay IDs. 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. Network Location

This document uses network location as a generic term to denote 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.

2.4. ALTO Information

This document uses ALTO information as a generic term to refer to the network information provided by an ALTO server.

2.5. ALTO Information Base

This document uses the term ALTO information base to refer to the internal representation of ALTO information maintained by an ALTO server. Note that the structure of this internal representation is not defined by this document.

3. Architecture

This section defines the ALTO architecture and the ALTO Protocol's place in the overall architecture.

3.1. ALTO Services and Protocol Scope

Each network region in the global Internet can provide its ALTO services, which convey 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.

The ALTO services specified in this document define 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. We say that the network information provided by the ALTO services of a network region represents the "my-Internet view" of the network region.

The "my-Internet view" defined in this document does not specify the internal topology of a network, and hence, it is said to provide a "single-node" abstract topology. Extensions to this document may provide topology details in "my-Internet view".

Figure 1 provides an overall picture of ALTO's system architecture, so that one can better understand the ALTO services and the role of the ALTO Protocol. 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 they can be seen as a policy that is updated on a longer time scale.

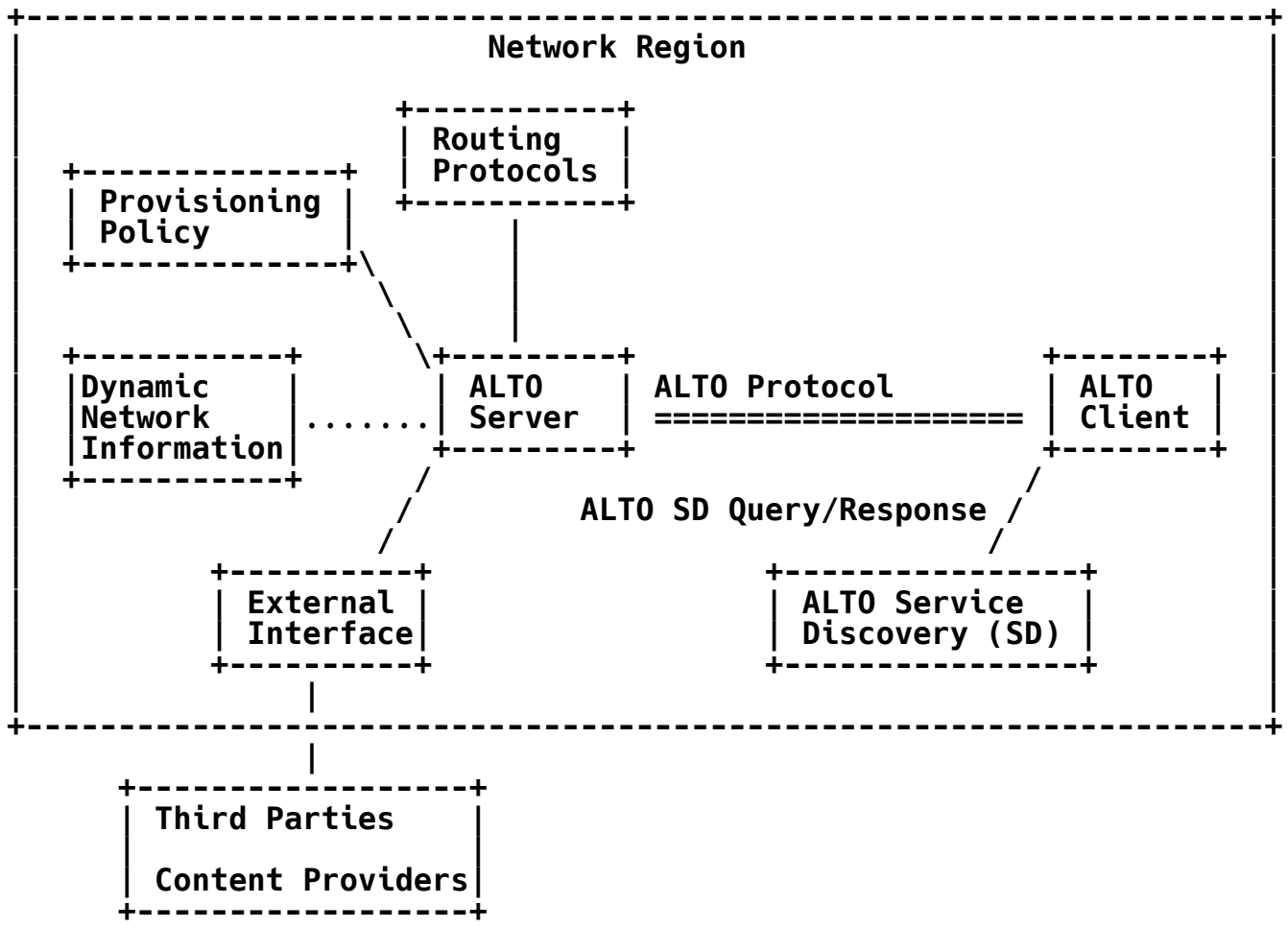

Figure 1: Basic ALTO Architecture

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 the ALTO Protocol 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 Distributed Hash Tables (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 ALTO information services (services for short), where each service defines a set of related functionalities. An ALTO client can request 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.

The goals of the ALTO information services defined in this document are to convey (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 endpoints. The aforementioned goals are achieved by defining the Map Service, which provides the core ALTO information to clients, and three additional information services: the Map-Filtering Service, the Endpoint Property Service (EPS), and the Endpoint Cost Service (ECS). Additional information services can be defined in companion documents. Figure 2 gives an overview of the information services. Details of the services are presented in subsequent sections.

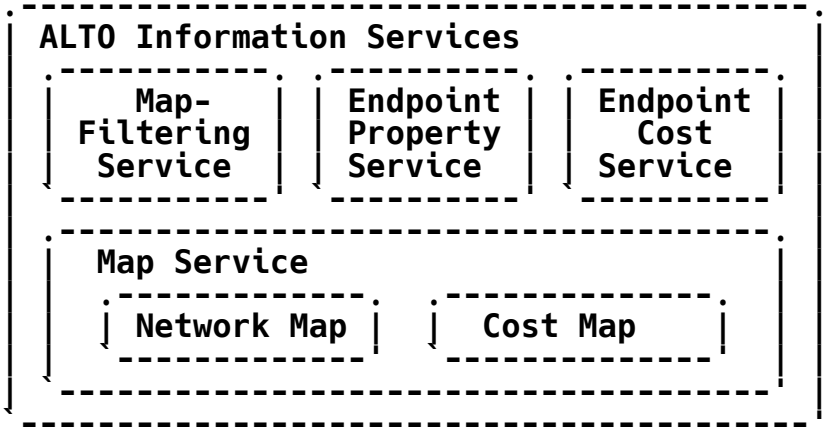

Figure 2: ALTO Information Service Framework

4.1. ALTO Information Services

4.1.1. Map Service

The Map Service provides batch information to ALTO clients in the forms of ALTO network maps (network maps for short) and ALTO cost maps (cost maps for short). An ALTO network map (See Section 5) provides a full set of network location groupings defined by the ALTO server and the endpoints contained within each grouping. An ALTO cost map (see Section 6) provides costs between defined groupings.

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

4.1.2. Map-Filtering Service

Resource-constrained ALTO clients may benefit from the filtering of query results at the ALTO server. This avoids the situation in which an ALTO client first spends network bandwidth and CPU cycles to collect results and then performs client-side filtering. The Map-Filtering Service allows ALTO clients to query an ALTO server on ALTO network maps and/or cost maps 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 costs directly amongst endpoints.

5. Network Map

An ALTO network map defines a grouping of network endpoints. This document uses ALTO network map to refer to the syntax and semantics of how an ALTO server defines the grouping. This document does not discuss the internal representation of this data structure within an ALTO server.

The definition of ALTO network maps is based on the observation that, in reality, many endpoints are near by to one another in terms of network connectivity. By treating a group of nearby 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 maps).

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 string of type PIDName (see Section 10.1) and its associated set of endpoint addresses. As 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. Interpreting the PIDs defined in an ALTO network map using the "single-node" abstraction, one can consider that each PID represents an abstract port (POP) that connects a set of endpoints.

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 in a P2P application communicate locally instead of communicating with endpoints in other POPs. The ISP may aggregate endpoints within a POP into a single PID in a 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 document specifies (in Section 10.4) how to specify IPv4/IPv6 addresses or prefixes. Extension documents may define further address types; Section 14.4 of this document provides an IANA registry for endpoint address types.

5.3. Example Network Map

This document uses the ALTO network map shown in Figure 3 in most examples.

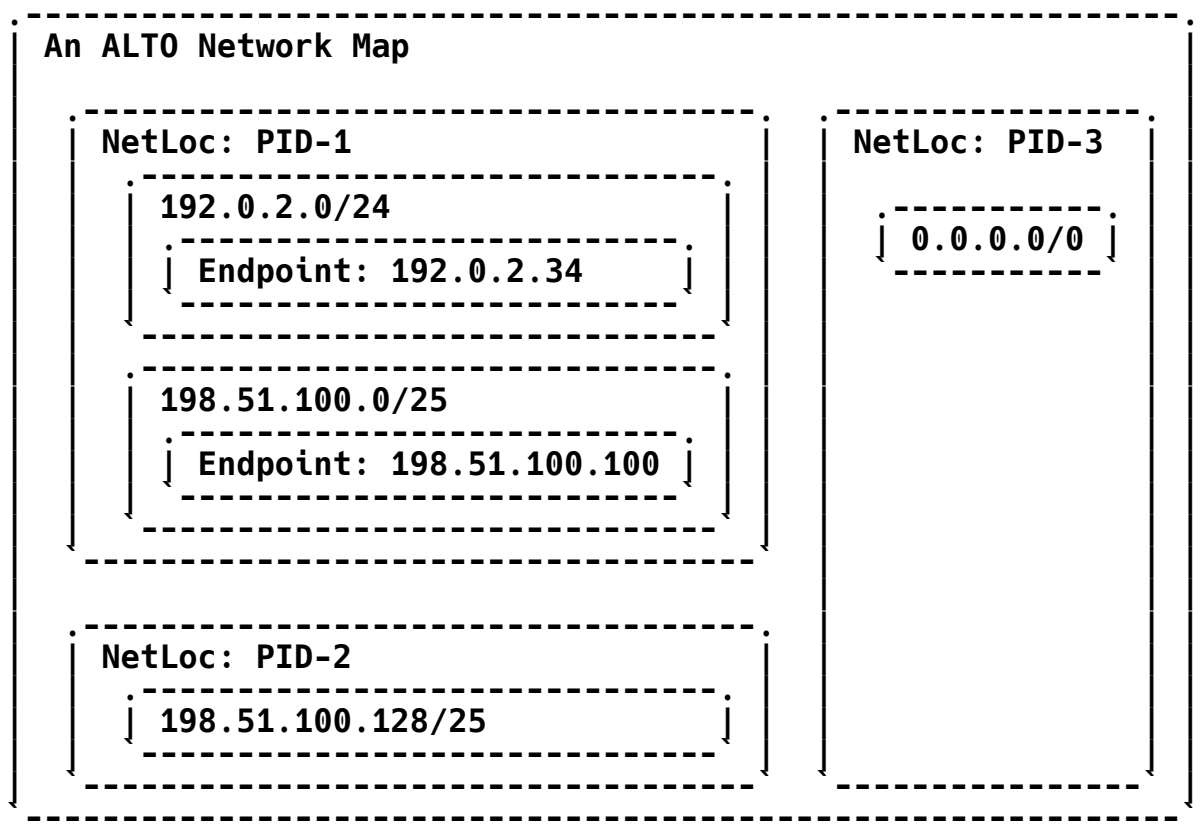

Figure 3: Example Network Map

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 policy.

For a given ALTO network map, an ALTO cost map defines path costs pairwise amongst the set of source and destination network locations defined by the PIDs contained in the network map. Each path cost is the end-to-end cost when a unit of traffic goes from the source to the destination.

Since 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 the resource consumer as the destination.

One advantage of separating ALTO information into network maps and cost maps is that the two types of maps 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 more dynamic network conditions.

As used in this document, an ALTO 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:

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

The combination of a cost metric and a cost mode defines an ALTO cost type. Certain queries for ALTO cost maps allow the ALTO client to indicate the desired cost type. For a given ALTO server, the combination of cost type and network map defines a key. In other words, an ALTO server MUST NOT define two ALTO cost maps with the same cost type \ network map pair.

6.1.1. Cost Metric

The cost 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. A lower value indicates 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 that it chooses (e.g., air miles or hop-count) as long as it conforms to the semantics.

6.1.2. Cost Mode

The cost mode attribute indicates how costs should be interpreted. Specifically, the cost 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 the following modes: numerical and ordinal. An ALTO client needs to be cognizant of operations when its desired cost mode is not supported. Specifically, 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 from retrieved 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 cost values in a cost map represent ranking (relative to all other values in a cost map), not actual costs. The values are non-negative integers, with a lower value indicating a higher preference. Ordinal cost values in a cost map need not be unique or contiguous. In particular, it is possible that two entries in a cost 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.

6.2. Cost Map Structure

A request for an ALTO cost map will either explicitly or implicitly include 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 request specifies a list of source network locations, say [Src_1, Src_2, ..., Src_m], and a list of 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 the path cost for a particular pair, that cost may be omitted. This document refers 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

An ALTO cost map gives path costs between the PIDs defined in an ALTO network map. An ALTO server may modify an ALTO network map at any time, say by adding or deleting PIDs, or even redefining them. Hence, to effectively use an instance of an ALTO cost map, an ALTO client must know which version of the network map defined the PIDs in that cost map. Version tags allow an ALTO client to correlate cost map instances with the corresponding versions of the network maps.

Specifically, a version tag is a tuple of (1) an ID for the resource (e.g., an ALTO network map) and (2) a tag (an opaque string) associated with the version of that resource. An ALTO network map distributed by an ALTO server includes its version tag. An ALTO cost map referring to PIDs also includes the version tag for the network map on which it is based.

Two ALTO network maps are the same if they have the same version tag. Whenever the content of an ALTO network map maintained by an ALTO server changes, the tag MUST also be changed. Possibilities of setting the tag component 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.

6.4. Cost Map Update

An ALTO server can update an ALTO cost map at any time. Hence, the same cost map retrieved from the same ALTO server but from different requests can be inconsistent.

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. See Section 10.8 for more details on specifying endpoint properties.

7.1.1. Endpoint Property Type: pid

An ALTO server MUST define the "pid" endpoint property type for each ALTO network map that it provides. Specifically, each ALTO network map defines multiple PIDs. For an "ipv4"/"ipv6" network map, given an endpoint's IP address, the ALTO server uses the algorithm specified in Section 11.2.2 to look up the PID of the endpoint. This PID is the "pid" property of the endpoint for the network map. See Section 11.4.1.7 for an example.

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 ALTO service is realized by a set of network information resources. Each information resource has a media type [RFC2046]. An ALTO client may construct an HTTP request for a particular information resource (including any parameters, if necessary), and the ALTO server returns the requested information resource in an HTTP response.

- Information Resource Directory (IRD): An ALTO server uses an IRD to inform an ALTO client about a list of available information resources and the URI at which each can be accessed. ALTO clients consult the IRDs to determine the services provided by ALTO servers.

8.2. Notation

This document uses JSONString, JSONNumber, and JSONBool to indicate the JSON string, number, and boolean types, respectively. The type JSONValue indicates a JSON value, as specified in Section 3 of [RFC7159].

This document uses an adaptation of the C-style struct notation to define JSON objects. A JSON object consists of name/value pairs. This document refers to each pair as a field. In some context, this document also refers to a field as an attribute. The name of a field/attribute may be referred to as the key. An optional field is enclosed by [ ]. In the definitions, the JSON names of the fields are case sensitive. An array is indicated by two numbers in angle brackets, <m..n>, where m indicates the minimal number of values and n is the maximum. When this document uses * for n, it means no upper bound.

For example, the definition below defines a new type Type4, with three fields named "name1", "name2", and "name3", respectively. The field named "name3" is optional, and the field named "name2" is an array of at least one value.

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

This document also defines 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 defined as any type.

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

This document uses subtyping to denote that one type is derived from another type. The example below denotes that TypeDerived is derived from TypeBase. TypeDerived includes all fields defined in TypeBase. If TypeBase does not have a field named "name1", TypeDerived will have a new field named "name1". If TypeBase already has a field named "name1" but with a different type, TypeDerived will have a field named "name1" with the type defined in TypeDerived (i.e., Type1 in the example).

```
object { Type1 name1; } TypeDerived : TypeBase;
```

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

8.3. Basic Operations

The ALTO Protocol employs standard HTTP [RFC7230]. 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 they MUST be compliant with [RFC7230].

Instead of specifying the generic application/json media type for all ALTO request parameters (if any) and responses, ALTO clients and servers use multiple, specific JSON-based media types (e.g., application/alto-networkmap+json, application/alto-costmap+json) to indicate content types; see Table 2 for a list of media types defined in this document. This allows easy extensibility while maintaining clear semantics and versioning. For example, a new version of a component of the ALTO Protocol (e.g., a new version of ALTO network maps) can be defined by simply introducing a new media type (e.g., application/alto-networkmap-v2+json).

8.3.1. Client Discovering Information Resources

To discover available information resources provided by an ALTO server, an ALTO client requests its IRD(s).

Specifically, using an ALTO service discovery protocol, an ALTO client obtains a URI through which it can request an information resource directory (IRD). This document refers to this IRD as the Root IRD of the ALTO client. Each entry in an IRD indicates a URI at which an ALTO server accepts requests, and returns either an information resource or an information resource directory that references additional information resources. Beginning with its Root IRD and following links to IRDs recursively, an ALTO client can discover all information resources available to it. This set of information resources is referred to as the information resource closure of the ALTO client. By inspecting its information resource closure, 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.

See Section 9.2 for a detailed specification of IRDs.

8.3.2. Client Requesting Information Resources

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.

An ALTO client MUST NOT assume that the HTTP GET and POST methods are interchangeable. In particular, for an information resource that uses the HTTP GET method, an ALTO client MUST NOT assume that the information resource will accept a POST request as equivalent to a GET request.

8.3.3. Server Responding to Information Resource Request

Upon receiving a request for an information resource that the ALTO server can provide, the ALTO server normally returns the requested information resource. In other cases, to be more informative ([RFC7231]), the ALTO server either provides the ALTO client with an information resource directory indicating how to reach the desired information resource, or it returns an ALTO error object; see Section 8.5 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 [RFC7230]). 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 Server Response and IRD

After receiving an information resource directory, the client can consult it to determine if any of the offered URIs contain the desired information resource. However, an ALTO client MUST NOT assume that the media type returned by the ALTO server for a request to a URI is the media type advertised in the IRD or specified in its request (i.e., the client must still check the Content-Type header). The expectation is that the media type returned should normally be the media type advertised and requested, but, in some cases, it may legitimately not be so.

In particular, 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 9.2.1 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

ALTO server implementations as well as ALTO client implementations MUST support the "https" URI scheme [RFC2818] and Transport Layer Security (TLS) [RFC5246]. See Section 15.1.2 for security considerations and Section 16 for manageability considerations regarding the usage of HTTPS/TLS.

For deployment scenarios where client authentication is desired, HTTP Digest Authentication MUST be supported. TLS Client Authentication is the preferred mechanism if it is available.

8.3.6. Information Refreshing

An ALTO client can determine the frequency at which ALTO information is refreshed based on information made available via HTTP.

8.3.7. Parsing of Unknown Fields

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

8.4. Server Response Encoding

Though each type of ALTO server response (i.e., an information resource directory, an individual information resource, or an error message) has its distinct syntax and, hence, its unique media type, they are designed to have a similar structure: a field named "meta" to provide meta definitions, and another field named "data" to contain the data, if needed.

Specifically, this document defines the base type of each ALTO server response as ResponseEntityBase:

```
object { ResponseMeta meta; } ResponseEntityBase;
```

with field:

meta: meta information pertaining to the response.

8.4.1. Meta Information

Meta information is encoded as a map object for flexibility. Specifically, ResponseMeta is defined as:

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

8.4.2. Data Information

The data component of the response encodes the response-specific data. This document derives five types from ResponseEntityBase to add different types of data component: InfoResourceDirectory (Section 9.2.2), InfoResourceNetworkMap (Section 11.2.1.6), InfoResourceCostMap (Section 11.2.3.6), InfoResourceEndpointProperties (Section 11.4.1.6), and InfoResourceEndpointCostMap (Section 11.5.1.6).

8.5. Protocol Errors

If an ALTO server encounters an error while processing a request, the 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 the 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 information resource was not successful. The HTTP status code, on the other hand, indicates to HTTP processing elements (e.g., intermediaries and clients) how the response should be treated.

8.5.1. Media Type

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

8.5.2. Response Format and Error Codes

An ALTO error response MUST include a field named "code" in the "meta" field of the response. The value MUST be an ALTO error code, encoded in string, 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_MISSING_FIELD	A required JSON field is missing
E_INVALID_FIELD_TYPE	The type of the value of a JSON field is invalid
E_INVALID_FIELD_VALUE	The value of a JSON field is invalid

Table 1: Defined ALTO Error Codes

After an ALTO server receives a request, it needs to verify the syntactic and semantic validity of the request. The following paragraphs in this section are intended to illustrate the usage of the error codes defined above during the verification. An individual implementation may define its message processing in a different order.

In the first step after an ALTO server receives a request, it checks the syntax of the request body (i.e., whether the JSON structure can be parsed), and indicates a syntax error using the error code E_SYNTAX. For an E_SYNTAX error, the ALTO server MAY provide an optional field named "syntax-error" in the "meta" field of the error response. The objective of providing "syntax-error" is to provide technical debugging information to developers, not end users. Hence, it should be a human-readable, free-form text describing the syntax error. If possible, the text should include position information about the syntax error, such as line number and offset within the line. If nothing else, the value of the field named "syntax-error" could include just the position. If a syntax error occurs in a production environment, the ALTO client could inform the end user that there was an error communicating with the ALTO server, and suggest that the user submit the error information, which includes "syntax-error", to the developers.

A request without syntax errors may still be invalid. An error case is that the request misses a required field. The server indicates such an error using the error code E_MISSING_FIELD. This document defines required fields for Filtered Network Map (Section 11.3.1.3),

Filtered Cost Map (Section 11.3.2.3), Endpoint Properties (Section 11.4.1.3), and Endpoint Cost (Section 11.5.1.3) services. For an E_MISSING_FIELD error, the server may include an optional field named "field" in the "meta" field of the error response, to indicate the missing field. "field" should be a JSONString indicating the full path of the missing field. For example, assume that a Filtered Cost Map request (see Section 11.3.2.3) omits the "cost-metric" field. The error response from the ALTO server may specify the value of "field" as "cost-type/cost-metric".

A request with the correct fields might use a wrong type for the value of a field. For example, the value of a field could be a JSONString when a JSONNumber is expected. The server indicates such an error using the error code E_INVALID_FIELD_TYPE. The server may include an optional field named "field" in the "meta" field of the response, to indicate the field that contains the wrong type.

A request with the correct fields and types of values for the fields may specify a wrong value for a field. For example, a Filtered Cost Map request may specify a wrong value for CostMode in the "cost-type" field (Section 11.3.2.3). The server indicates such an error with the error code E_INVALID_FIELD_VALUE. For an E_INVALID_FIELD_VALUE error, the server may include an optional field named "field" in the "meta" field of the response, to indicate the field that contains the wrong value. The server may also include an optional field named "value" in the "meta" field of the response to indicate the wrong value that triggered the error. If the "value" field is specified, the "field" field MUST be specified. The "value" field MUST have a JSONString value. If the invalid value is not a string, the ALTO server MUST convert it to a string. Below are the rules to specify the "value" key:

- If the invalid value is a string, "value" is that string;
- If the invalid value is a number, "value" must be the invalid number as a string;
- If the invalid value is a subfield, the server must set the "field" key to the full path of the field name and "value" to the invalid subfield value, converting it to a string if needed. For example, if the "cost-mode" subfield of the "cost-type" field is an invalid mode "foo", the server should set "value" to "foo", and "field" to "cost-mode/cost-type";
- If an element of a JSON array has an invalid value, the server sets "value" to the value of the invalid element, as a string, and "field" to the name of the array. An array element of the wrong type (e.g., a number in what is supposed to be an array of

strings) is an invalid value error, not an invalid type error. The server sets "value" to the string version of the incorrect element, and "field" to the name of the array.

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 techniques.

8.5.3. 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 [RFC7230], 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. Using Temporary Redirect may generate infinite redirection loops. Although [RFC7231] Section 6.4 specifies that an HTTP client SHOULD detect infinite redirection loops, it is more desirable that multiple ALTO servers be configured not to form redirection loops.

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: Information Resource Directory

As already discussed, an ALTO client starts by retrieving an information resource directory, which specifies the attributes of individual information resources that an ALTO server provides.

9.1. Information Resource Attributes

In this document, each information resource has up to five attributes associated with it, including its assigned ID, its response format, its capabilities, its accepted input parameters, and other resources on which it may depend. The function of an information resource directory is to publishes these attributes.

9.1.1. Resource ID

Each information resource that an ALTO client can request MUST be assigned a resource ID attribute that is unique amongst all information resources in the information resource closure of the client. The resource ID SHOULD remain stable even when the data provided by that resource changes. For example, even though the number of PIDs in an ALTO network map may be adjusted, its resource ID should remain the same. Similarly, if the entries in an ALTO cost map are updated, its resource ID should remain the same. IDs SHOULD NOT be reused for different resources over time.

9.1.2. Media Type

ALTO uses media types [RFC2046] to uniquely indicate the data format used to encode the content to be transmitted between an ALTO server and an ALTO client in the HTTP entity body.

9.1.3. Capabilities

The Capabilities attribute of an information resource 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, then the server advertises the "cost-constraints" capability of the cost map information resource.

9.1.4. 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 such an information resource specifies a media type, which indicates how the client specifies the input parameters as contained in the entity body of the HTTP POST request.

9.1.5. Dependent Resources

The information provided in an information resource may use information provided in some other resources (e.g., a cost map uses the PIDs defined in a network map). The "uses" attribute conveys such information.

9.2. Information Resource Directory (IRD)

An ALTO server uses the 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 one 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 it could be accomplished by manual configuration or by returning the URI of an information resource directory from the ALTO Discovery Protocol [ALTO-SERVER-DISC]. For recommendations on what the URI may look like, see [ALTO-SERVER-DISC].

9.2.1. Media Type

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

9.2.2. Encoding

An information resource directory response may include in the "meta" field the "cost-types" field, whose value is of type IRDMetaCostTypes defined below, where CostType is defined in Section 10.7:

```
object-map {
  JSONString -> CostType;
} IRDMetaCostTypes;
```

The function of "cost-types" is to assign names to a set of CostTypes that can be used in one or more "resources" entries in the IRD to simplify specification. The names defined in "cost-types" in an IRD are local to the IRD.

For a Root IRD, "meta" MUST include a field named "default-alto-network-map", which value specifies the resource ID of an ALTO network map. When there are multiple network maps defined in an IRD (e.g., with different levels of granularity), the "default-alto-network-map" field provides a guideline to simple clients that use only one network map.

The data component of an information resource directory response is named "resources", which is a JSON object of type IRDResourceEntries:

```
object {
  IRDResourceEntries resources;
} InfoResourceDirectory : ResponseEntityBase;


object-map {
  ResourceID  -> IRDResourceEntry;
} IRDResourceEntries;

object {
  JSONString      uri;
  JSONString      media-type;
  [JSONString     accepts;]
  [Capabilities   capabilities;]
  [ResourceID     uses<0..*>;]
} IRDResourceEntry;



object {
  ...
} Capabilities;
```

An IRDResourceEntries object is a dictionary map keyed by ResourceIDs, where ResourceID is defined in Section 10.2. The value of 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 to the URI of the IRD and MUST be resolved according to Section 5 of [RFC3986].

media-type: The media type of the information resource (see Section 9.1.2) available via GET or POST requests to the corresponding URI. A value of "application/alto-directory+json" indicates that the response for a

request to the URI will be an information resource directory defining additional information resources in the information resource closure.

accepts: The media type of input parameters (see Section 9.1.4) accepted by POST requests to the corresponding URI. If this field is not present, it MUST be assumed to be empty.

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 field 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.

uses: A list of resource IDs, defined in the same IRD, that define the resources on which this resource directly depends. An ALTO server SHOULD include in this list any resources that the ALTO client would need to retrieve in order to interpret the contents of this resource. For example, an ALTO cost map resource should include in this list the network map on which it depends. ALTO clients may wish to consult this list in order to pre-fetch necessary resources.

If an entry has an empty list for "accepts", then the corresponding URI MUST support GET requests. If an entry has a non-empty "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.

9.2.3. Example

The following is an example information resource directory returned by an ALTO server to an ALTO client. Assume it is the Root IRD of the client.

```
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: 2333
 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"
         }
      },
      "default-alto-network-map" : "my-default-network-map"
    },
    "resources" : {
      "my-default-network-map" : {
         "uri" : "http://alto.example.com/networkmap",
         "media-type" : "application/alto-networkmap+json"
      },
      "numerical-routing-cost-map" : {
         "uri" : "http://alto.example.com/costmap/num/routingcost",
         "media-type" : "application/alto-costmap+json",
         "capabilities" : {
            "cost-type-names" : [ "num-routing" ]
         },
         "uses": [ "my-default-network-map" ]
      },
      "numerical-hopcount-cost-map" : {
         "uri" : "http://alto.example.com/costmap/num/hopcount",
         "media-type" : "application/alto-costmap+json",
         "capabilities" : {
```

```
              "cost-type-names" : [ "num-hop" ]
           },
           "uses": [ "my-default-network-map" ]
        },
        "custom-maps-resources" : {
           "uri" : "http://custom.alto.example.com/maps",
           "media-type" : "application/alto-directory+json"
        },
        "endpoint-property" : {
           "uri" : "http://alto.example.com/endpointprop/lookup",
           "media-type" : "application/alto-endpointprop+json",
           "accepts" : "application/alto-endpointpropparams+json",
           "capabilities" : {
             "prop-types" : [ "my-default-network-map.pid",
                              "priv:ietf-example-prop" ]
           },
        },
        "endpoint-cost" : {
           "uri" : "http://alto.example.com/endpointcost/lookup",
           "media-type" : "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 "cost-types" field of "meta" of the example IRD defines names for four cost types in this IRD. For example, "num-routing" in the example is the name that refers to a cost type with cost mode being "numerical" and cost metric being "routingcost". This name is used in the second entry of "resources", which defines a cost map. In particular, the "cost-type-names" of its "capabilities" specifies that this resource supports a cost type named as "num-routing". The ALTO client looks up the name "num-routing" in "cost-types" of the IRD to obtain the cost type named as "num-routing". The last entry of "resources" uses all four names defined in "cost-types".

Another field defined in "meta" of the example IRD is "default-alto-network-map", which has value "my-default-network-map", which is the resource ID of an ALTO network map that will be defined in "resources".

The "resources" field of the example IRD defines six information resources. For example, the second entry, which is assigned a resource ID "numerical-routing-cost-map", provides a cost map, as indicated by the media-type "application/alto-costmap+json". The cost map is based on the network map defined with resource ID "my-default-network-map". As another example, the last entry, which is assigned resource ID "endpoint-cost", provides 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" field of the "capabilities" attribute of the information resource includes four defined cost types specified in the "cost-types" field of "meta" of the IRD. Hence, an ALTO client can verify that the Endpoint Cost information resource supports both cost metrics "routingcost" and "hopcount", each available for both "numerical" and "ordinal" cost modes. When requesting the information resource, an ALTO client can specify cost constraints, as indicated by the "cost-constraints" field of the "capabilities" attribute.

9.2.4. Delegation Using IRD

ALTO IRDs provide the flexibility to define a set of information resources that are provided by ALTO servers running in multiple domains. Consider the preceding example. Assume that the ALTO server running at alto.example.com wants to delegate some information resources to a separate subdomain: "custom.alto.example.com". In particular, assume that the maps available via this subdomain are filtered network maps, filtered cost maps, and some pre-generated maps for the "hopcount" and "routingcost" cost metrics in the "ordinal" cost mode. The fourth entry of "resources" in the preceding example IRD implements the delegation. The entry has a media-type of "application/alto-directory+json", and an ALTO client can discover the information resources available at "custom.alto.example.com" if its request to "http://custom.alto.example.com/maps" is successful:

```
     GET /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: 1900
   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" : {
        "filtered-network-map" : {
           "uri" : "http://custom.alto.example.com/networkmap/filtered",
           "media-type" : "application/alto-networkmap+json",
           "accepts" : "application/alto-networkmapfilter+json",
           "uses": [ "my-default-network-map" ]
        },
        "filtered-cost-map" : {
           "uri" : "http://custom.alto.example.com/costmap/filtered",
           "media-type" : "application/alto-costmap+json",
           "accepts" : "application/alto-costmapfilter+json",
           "capabilities" : {
              "cost-constraints" : true,
              "cost-type-names"  : [ "num-routing", "num-hop",
                                     "ord-routing", "ord-hop" ]
           },
           "uses": [ "my-default-network-map" ]
        },
```

```
        "ordinal-routing-cost-map" : {
           "uri" : "http://custom.alto.example.com/ord/routingcost",
           "media-type" : "application/alto-costmap+json",
           "capabilities" : {
              "cost-type-names" : [ "ord-routing" ]
           },
           "uses": [ "my-default-network-map" ]
        },
        "ordinal-hopcount-cost-map" : {
           "uri" : "http://custom.alto.example.com/ord/hopcount",
           "media-type" : "application/alto-costmap+json",
           "capabilities" : {
              "cost-type-names" : [ "ord-hop" ]
           },
           "uses": [ "my-default-network-map" ]
        }
     }
   }
```

Note that the subdomain does not define any network maps, and uses the network map with resource ID "my-default-network-map" defined in the Root IRD.

9.2.5. Considerations of Using IRD

9.2.5.1. ALTO client

This document specifies no requirements or constraints on ALTO clients with regard to how they process an information resource directory to identify the URI corresponding to a desired information resource. However, some advice is provided for implementers.

It is possible that multiple entries in the directory match a desired information resource. For instance, in the example in Section 9.2.3, a full cost map with the "numerical" cost mode and the "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 cacheable. However, an ALTO client may need to use POST, for example, to get ALTO costs or properties that are for a restricted set of PIDs or endpoints or to update cached information previously acquired via GET requests.

9.2.5.2. 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".

10. Protocol Specification: Basic Data Types

This section details the format of basic data types.

10.1. PID Name

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

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

10.2. Resource ID

A resource ID uniquely identifies a particular resource (e.g., an ALTO network map) within an ALTO server (see Section 9.2).

A resource ID is encoded as a JSON string with the same format as that of the type PIDName.

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

10.3. Version Tag

A version tag is defined as:

```
object {
  ResourceID resource-id;
  JSONString tag;
} VersionTag;
```

As described in Section 6.3, the "resource-id" field provides the resource ID of a resource (e.g., a network map) defined in the information resource directory, and "tag" provides an identifier string.

Two version tags are equal if and only if both the "resource-id" fields are byte-for-byte equal and the "tag" fields are byte-for-byte equal.

A string representing the "tag" field MUST be no more than 64 characters, and it MUST NOT contain any character below U+0021 or above U+007E. It is RECOMMENDED that the "tag" string have a low collision probability with other tags. One suggested mechanism is to compute it using a hash of the data contents of the resource.

10.4. 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.

10.4.1. 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 types. This document refers to such addresses as "Typed Endpoint Addresses".

Typed endpoint addresses are encoded as strings of the format AddressType:EndpointAddr, with the ':' character as a separator. The type TypedEndpointAddr is used to indicate a string of this format.

10.4.2. Address Type

The AddressType component of TypedEndPointAddr is defined as a string consisting of only US-ASCII alphanumeric characters (U+0030-U+0039, U+0041-U+005A, and U+0061-U+007A). The type AddressType is used in this document to indicate a string of this format.

This document defines two values for AddressType: "ipv4" to refer to IPv4 addresses and "ipv6" to refer to IPv6 addresses. All AddressType 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 14.4).

10.4.3. Endpoint Address

The EndpointAddr component of TypedEndPointAddr is also encoded as a string. The exact characters and format depend on AddressType. This document defines EndpointAddr when AddressType is "ipv4" or "ipv6".

10.4.3.1. IPv4

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

10.4.3.2. IPv6

IPv6 endpoint addresses are encoded as specified in Section 4 of [RFC5952].

10.4.4. 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 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.

10.4.4.1. IPv4

IPv4 endpoint prefixes are encoded as specified in Section 3.1 of [RFC4632].

10.4.4.2. IPv6

IPv6 endpoint prefixes are encoded as specified in Section 7 of [RFC5952].

10.4.5. 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"
  ]
}
```

10.5. Cost Mode

A cost mode is encoded as a string. The string MUST have a value of either "numerical" or "ordinal".

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

10.6. Cost Metric

A cost metric is encoded as a string. The string MUST be no more than 32 characters, and it MUST NOT contain characters other than US-ASCII alphanumeric characters (U+0030-U+0039, U+0041-U+005A, and U+0061-U+007A), the hyphen ('-', U+002D), the colon (':', U+003A), the low line ('_', U+005F), or the '.' separator (U+002E). The '.' separator is reserved for future use and MUST NOT be used unless specifically indicated by a companion or extension document.

Identifiers prefixed with "priv:" are reserved for Private Use [RFC5226] without a need to register with IANA. All other identifiers that appear in an HTTP request or response with an "application/alto-*" media type and indicate cost metrics MUST be registered in the "ALTO Cost Metric Registry" Section 14.2. For an identifier with the "priv:" prefix, an additional string (e.g., company identifier or random string) MUST follow (i.e., "priv:" only is not a valid identifier) to reduce potential collisions.

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

10.7. Cost Type

The combination of CostMetric and CostMode defines the type CostType:

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

The "description" field, if present, MUST provide a string value with a human-readable description of the cost-metric and cost-mode. An ALTO client MAY present this string to a developer, as part of a discovery process; however, the field is not intended to be interpreted by an ALTO client.

10.8. Endpoint Property

This document distinguishes two types of endpoint properties: resource-specific endpoint properties and global endpoint properties. The type EndpointPropertyType is used in this document to indicate a string denoting either a resource-specific endpoint property or a global endpoint property.

10.8.1. Resource-Specific Endpoint Properties

The name of resource-specific endpoint property MUST follow this format: a resource ID, followed by the '.' separator (U+002E), followed by a name obeying the same rules as for global endpoint property names (Section 10.8.2).

This document defines only one resource-specific endpoint property: pid. An example is "my-default-networkmap.pid".

10.8.2. Global Endpoint Properties

A global endpoint property is encoded as a string. The string MUST be no more than 32 characters, and it MUST NOT contain characters other than US-ASCII alphanumeric characters (U+0030-U+0039, U+0041-U+005A, and U+0061-U+007A), the hyphen ('-', U+002D), the colon (':', U+003A), or the low line ('_', U+005F). Note that the '.' separator is not allowed so that there is no ambiguity on whether an endpoint property is global or resource specific.

Identifiers prefixed with "priv:" are reserved for Private Use [RFC5226] without a need to register with IANA. All other identifiers for endpoint properties appearing in an HTTP request or response with an "application/alto-*" media type MUST be registered in the "ALTO Endpoint Property Type Registry" Section 14.3. For an endpoint property identifier with the "priv:" prefix, an additional string (e.g., company identifier or random string) MUST follow (i.e., "priv:" only is not a valid endpoint property identifier) to reduce potential collisions.

11. Protocol Specification: Service Information Resources

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

11.1. Meta Information

For the "meta" field of the response to an individual information resource, this document defines two generic fields: the "vtag" field, which provides the version tag (see Section 10.3) of the current information resource, and the "dependent-vtags" field, which is an array of version tags, to indicate the version tags of the resources on which this resource depends.

11.2. Map Service

The Map Service provides batch information to ALTO clients in the form of two types of maps: ALTO network maps and ALTO cost maps.

11.2.1. Network Map

An ALTO network map information resource defines a set of PIDs, and for each PID, lists the network locations (endpoints) within the PID. An ALTO server MUST provide at least one network map.

11.2.1.1. Media Type

The media type of ALTO network maps is "application/alto-networkmap+json".

11.2.1.2. HTTP Method

An ALTO network map resource is requested using the HTTP GET method.

11.2.1.3. Accept Input Parameters

None.

11.2.1.4. Capabilities

None.

11.2.1.5. Uses

None.

11.2.1.6. Response

The "meta" field of an ALTO network map response MUST include the "vtag" field, which provides the version tag of the retrieved network map.

The data component of an ALTO network map response is named "network-map", which is a JSON object of type NetworkMapData:

```
object {
  NetworkMapData network-map;
} InfoResourceNetworkMap : ResponseEntityBase;

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

Specifically, a NetworkMapData object is a dictionary map keyed by PIDs. The value of each PID is the associated set of endpoint addresses for the PID.

The returned network map MUST include all PIDs known to the ALTO server.

11.2.1.7. 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: 449
   Content-Type: application/alto-networkmap+json

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

When parsing an ALTO network map, an ALTO client MUST ignore any EndpointAddressGroup whose address type it does not recognize. If as a result a PID does not have any address types known to the client, the client still MUST recognize that PID name as valid, even though the PID then contains no endpoints.

Note that the encoding of an ALTO network map response was chosen for readability and compactness. If lookup efficiency at runtime is crucial, then the returned network map can be transformed into data structures offering more efficient lookup. For example, one may store an ALTO network map as a trie-based data structure, which may allow efficient longest-prefix matching of IP addresses.

11.2.2. Mapping IP Addresses to PIDs for 'ipv4'/'ipv6' Network Maps

A key usage of an ALTO network map is to map endpoint addresses to PIDs. For network maps containing the "ipv4" and "ipv6" address types defined in this document, when either an ALTO client or an ALTO server needs to compute the mapping from IP addresses to PIDs, the longest-prefix matching algorithm (Longest Match in Section 5.2.4.3 of [RFC1812]) MUST be used.

To ensure that the longest-prefix matching algorithm yields one and only one PID, an ALTO network map containing the "ipv4"/"ipv6" address types MUST satisfy the following two requirements.

First, such 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. This is defined as the completeness property of an ALTO network 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.

Second, such a network map MUST NOT define two or more PIDs that contain an identical IP prefix, in order to ensure that the longest-prefix matching algorithm maps each IP addresses into exactly one PID. This is defined as the non-overlapping property of an ALTO network map. Specifically, to map an IP address to its PID in a non-overlapping network map, one considers the set S, which consists of all prefixes defined in the network map, applies the longest-prefix mapping algorithm to S to identify the longest prefix containing the IP address and assigns that prefix the IP address belonging to the PID containing the identified longest prefix.

The following example shows a complete and non-overlapping ALTO network map:

```
    "network-map" : {
      "PID0" : { "ipv6" : [ "::/0" ] },
      "PID1" : { "ipv4" : [ "0.0.0.0/0" ] },
      "PID2" : { "ipv4" : [ "192.0.2.0/24", "198.51.100.0/24" ] },
      "PID3" : { "ipv4" : [ "192.0.2.0/25", "192.0.2.128/25" ] }
    }
```

The IP address 192.0.2.1 should be mapped to PID3.

If, however, the two adjacent prefixes in PID3 were combined as a single prefix, then PID3 was changed to:

```
      "PID3" : { "ipv4" : [ "192.0.2.0/24" ] }
```

The new map is no longer non-overlapping, and 192.0.2.1 could no longer be mapped unambiguously to a PID by means of longest-prefix matching.

Extension documents may define techniques to allow a single IP address being mapped to multiple PIDs, when a need is identified.

11.2.3. Cost Map

An ALTO cost map resource lists the path cost for each pair of source/destination PIDs 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.

11.2.3.1. Media Type

The media type of ALTO cost maps is "application/alto-costmap+json".

11.2.3.2. HTTP Method

An ALTO cost map resource is requested using the HTTP GET method.

11.2.3.3. Accept Input Parameters

None.

11.2.3.4. 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..1>;
} CostMapCapabilities;
```

with field:

cost-type-names: Note that the array MUST include a single CostType name defined by the "cost-types" field in the "meta" field of the IRD. This is because an unfiltered cost map (accept == "") is requested via an HTTP GET that accepts no input parameters. As a contrast, for filtered cost maps (see Section 11.3.2), the array can have multiple elements.

11.2.3.5. Uses

The resource ID of the network map based on which the cost map will be defined. Recall (Section 6) that the combination of a network map and a cost type defines a key. In other words, an ALTO server MUST NOT define two cost maps with the same cost type / network map pair.

11.2.3.6. Response

The "meta" field of a cost map response MUST include the "dependent-vtags" field, whose value is a single-element array to indicate the version tag of the network map used, where the network map is specified in "uses" of the IRD. The "meta" MUST also include the "cost-type" field, whose value indicates the cost type (Section 10.7) of the cost map.

The data component of a cost map response is named "cost-map", which is a JSON object of type CostMapData:

```
object {
  CostMapData cost-map;
} InfoResourceCostMap : ResponseEntityBase;

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

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

Specifically, a CostMapData object 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 path costs are not defined (e.g., either the source or the destination PIDs contain addresses outside of the network provider's administrative domain).

Similar to the encoding of ALTO network maps, the encoding of ALTO cost maps was chosen for readability and compactness. If lookup efficiency at runtime is crucial, then the returned cost map can be transformed into data structures offering more efficient lookup. For example, one may store a cost map as a matrix.

11.2.3.7. 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: 435
   Content-Type: application/alto-costmap+json

   {
     "meta" : {
       "dependent-vtags" : [
         {"resource-id": "my-default-network-map",
          "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"
         }
       ],
       "cost-type" : {"cost-mode"  : "numerical",
                      "cost-metric": "routingcost"
       }
     },
     "cost-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, array-based encoding for "map" was considered, but the current encoding was chosen for clarity.

11.3. Map-Filtering Service

The Map-Filtering Service allows ALTO clients to specify filtering criteria to return a subset of a full map available in the Map Service.

11.3.1. Filtered Network Map

A filtered ALTO network map is an ALTO network map information resource (Section 11.2.1) for which an ALTO client may supply a list of PIDs to be included. A filtered ALTO network map MAY be provided by an ALTO server.

11.3.1.1. Media Type

Since a filtered ALTO network map is still an ALTO network map, it uses the media type defined for ALTO network maps at Section 11.2.1.1.

11.3.1.2. HTTP Method

A filtered ALTO network map is requested using the HTTP POST method.

11.3.1.3. 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 fields:

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 a list of address types to be included in the returned filtered network map. If the "address-types" field is not specified, or 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.

11.3.1.4. Capabilities

None.

11.3.1.5. Uses

The resource ID of the network map based on which the filtering is performed.

11.3.1.6. Response

The format is the same as unfiltered network maps. See Section 11.2.1.6 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 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.

The version tag included in the "vtag" field of the response MUST correspond to the full (unfiltered) network map information resource from which the filtered information is provided. This ensures that a single, canonical version tag is used independent of any filtering that is requested by an ALTO client.

11.3.1.7. Example

```
POST /networkmap/filtered HTTP/1.1
Host: custom.alto.example.com
Content-Length: 33
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: 342
Content-Type: application/alto-networkmap+json

{
  "meta" : {
    "vtag" : {
       "resource-id": "my-default-network-map",
       "tag": "c0ce023b8678a7b9ec00324673b98e54656d1f6d"
    }
  },
  "network-map" : {
    "PID1" : {
      "ipv4" : [
        "192.0.2.0/24",
        "198.51.100.0/24"
      ]
    },
    "PID2" : {
      "ipv4": [
        "198.51.100.128/24"
      ]
    }
  }
}
```

11.3.2. Filtered Cost Map

A filtered ALTO cost map is a cost map information resource (Section 11.2.3) for which an ALTO client may supply additional parameters limiting the scope of the resulting cost map. A filtered ALTO cost map MAY be provided by an ALTO server.

11.3.2.1. Media Type

Since a filtered ALTO cost map is still an ALTO cost map, it uses the media type defined for ALTO cost maps at Section 11.2.3.1.

11.3.2.2. HTTP Method

A filtered ALTO cost map is requested using the HTTP POST method.

11.3.2.3. Accept Input Parameters

The input parameters for a filtered cost 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 fields:

cost-type: The CostType (Section 10.7) for the returned costs. The "cost-metric" and "cost-mode" fields MUST match one of the supported cost types indicated in this resource's "capabilities" field (Section 11.3.2.4). The ALTO client SHOULD omit the "description" field, and if present, the ALTO server MUST ignore the "description" field.

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" field (Section 11.3.2.4) 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 and (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" field is not present, both lists MUST be interpreted by the ALTO server as containing the full set of currently defined PIDs.

11.3.2.4. Capabilities

The URI providing this resource supports all capabilities documented in Section 11.2.3.4 (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 fields:

cost-type-names: See Section 11.2.3.4 and note that the array can have one 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 field 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.

11.3.2.5. Uses

The resource ID of the network map based on which the cost map will be filtered.

11.3.2.6. Response

The format is the same as an unfiltered ALTO cost map. See Section 11.2.3.6 for the format.

The "dependent-vtags" field in the "meta" field provides an array consisting of a single element, which is the version tag of the network map used in filtering. ALTO clients should verify that the version tag included in the response is equal to 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.

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.

11.3.2.7. Example

```
POST /costmap/filtered HTTP/1.1
Host: custom.alto.example.com
Content-Type: application/alto-costmapfilter+json
Content-Length: 181
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: 341
Content-Type: application/alto-costmap+json

{
  "meta" : {
    "dependent-vtags" : [
      {"resource-id": "my-default-network-map",
       "tag": "75ed013b3cb58f896e839582504f622838ce670f"
      }
    ],
    "cost-type": {"cost-mode" : "numerical",
                  "cost-metric" : "routingcost"
    }
  },
  "cost-map" : {
      "PID1": { "PID1": 0,  "PID2": 1,  "PID3": 2 }
  }
}
```

11.4. Endpoint Property Service

The Endpoint Property Service provides information about endpoint properties to ALTO clients.

11.4.1. Endpoint Property

An endpoint property resource provides information about properties for individual endpoints. In addition to the required "pid" endpoint property (see Sections 7.1.1 and 11.4.1.4), further endpoint properties MAY be provided by an ALTO server.

11.4.1.1. Media Type

The media type of an endpoint property resource is "application/alto-endpointprop+json".

11.4.1.2. HTTP Method

The endpoint property resource is requested using the HTTP POST method.

11.4.1.3. Accept Input Parameters

The input parameters for an endpoint property request are supplied in the entity body of the POST request. This document specifies the input parameters with a data format indicated by the media type "application/alto-endpointpropparams+json", which is a JSON object of type ReqEndpointProp:

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

with fields:

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" field (Section 11.4.1.4). 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.

11.4.1.4. Capabilities

The capabilities of an ALTO server URI providing endpoint properties are defined by a JSON object of type EndpointPropertyCapabilities:

```
    object {
      EndpointPropertyType prop-types<1..*>;
    } EndpointPropertyCapabilities;
```

with field:

prop-types: The endpoint properties (see Section 10.8) supported by the corresponding URI.

In particular, the information resource closure MUST provide the lookup of pid for every ALTO network map defined.

11.4.1.5. Uses

None.

11.4.1.6. Response

The "dependent-vtags" field in the "meta" field of the response MUST be an array that includes the version tags of all ALTO network maps whose "pid" is queried.

The data component of an endpoint properties response is named "endpoint-properties", which is a JSON object of type EndpointPropertyMapData, where:

```
    object {
      EndpointPropertyMapData endpoint-properties;
    } InfoResourceEndpointProperties : ResponseEntityBase;

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

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

Specifically, an EndpointPropertyMapData object 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.

11.4.1.7. Example

```
POST /endpointprop/lookup HTTP/1.1
Host: alto.example.com
Content-Length: 181
Content-Type: application/alto-endpointpropparams+json
Accept: application/alto-endpointprop+json,application/alto-error+json

{
  "properties" : [ "my-default-networkmap.pid",
                   "priv:ietf-example-prop" ],
  "endpoints"  : [ "ipv4:192.0.2.34",
                   "ipv4:203.0.113.129" ]
}


HTTP/1.1 200 OK
Content-Length: 396
Content-Type: application/alto-endpointprop+json

{
  "meta" : {
    "dependent-vtags" : [
      {"resource-id": "my-default-network-map",
       "tag": "7915dc0290c2705481c491a2b4ffbec482b3cf62"
      }
    ]
  },
  "endpoint-properties": {
    "ipv4:192.0.2.34"    : { "my-default-network-map.pid": "PID1",
                             "priv:ietf-example-prop": "1" },
    "ipv4:203.0.113.129" : { "my-default-network-map.pid": "PID3" }
  }
}
```

11.5. 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.

11.5.1. Endpoint Cost

An endpoint cost resource provides information about costs between individual endpoints. It MAY be provided by an ALTO server.

How an ALTO server provides the endpoint cost resource is implementation dependent. An ALTO server may use either fine-grained costs among individual endpoints or coarse-grained costs based on the costs between the PIDs corresponding to the endpoints. See Section 15.3 for additional details.

11.5.1.1. Media Type

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

11.5.1.2. HTTP Method

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

11.5.1.3. 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<0..*>;]
} EndpointFilter;
```

with fields:

cost-type: The cost type (Section 10.7) to use for returned costs. The "cost-metric" and "cost-mode" fields MUST match one of the supported cost types indicated in this resource's "capabilities" fields (Section 11.5.1.4). The ALTO client SHOULD omit the "description" field, and if present, the ALTO server MUST ignore the "description" field.

constraints: Defined equivalently to the "constraints" input parameter of a filtered cost map (see Section 11.3.2).

endpoints: A list of source endpoints and destination endpoints for which path costs are to be returned. If the list of source or destination 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 13.3 for discussion and considerations regarding this mode). The source and destination endpoint lists MUST NOT be both empty. The ALTO server MUST interpret entries appearing multiple times in a list as if they appeared only once.

11.5.1.4. Capabilities

This document defines EndpointCostCapabilities as the same as FilteredCostMapCapabilities. See Section 11.3.2.4.

11.5.1.5. Uses

It is important to note that although this resource allows an ALTO server to reveal costs between individual endpoints, the ALTO server is not required to do so. A simple implementation of ECS may compute the cost between two endpoints as the cost between the PIDs corresponding to the endpoints, using one of the exposed network and cost maps defined by the server. ECS MUST NOT specify the "use" field to indicate a network or cost map. Hence, the ECS cost is the cost from the source endpoint to the destination endpoint. A future extension may allow ECS to state that it "uses" a network map. The extension then will need to define the semantics.

11.5.1.6. Response

The "meta" field of an endpoint cost response MUST include the "cost-type" field, to indicate the cost type used.

The data component of an endpoint cost response is named "endpoint-cost-map", which is a JSON object of type EndpointCostMapData:

```
object {
  EndpointCostMapData endpoint-cost-map;
} InfoResourceEndpointCostMap : ResponseEntityBase;

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

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

Specifically, an EndpointCostMapData object is a dictionary map with each key representing a TypedEndpointAddr string identifying the source endpoint specified in the input parameters. For each source endpoint, an 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.

11.5.1.7. Example

```
  POST /endpointcost/lookup HTTP/1.1
  Host: alto.example.com
  Content-Length: 248
  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: 274
  Content-Type: application/alto-endpointcost+json

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

12. 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 Content Distribution Networks (CDNs) [ALTO-USE-CASES].

12.1. ALTO Client Embedded in P2P Tracker

Many 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 (e.g., ALTO network maps and cost maps) from the ISPs containing the P2P clients, and benefit from the same aggregation of network locations done by ALTO servers.

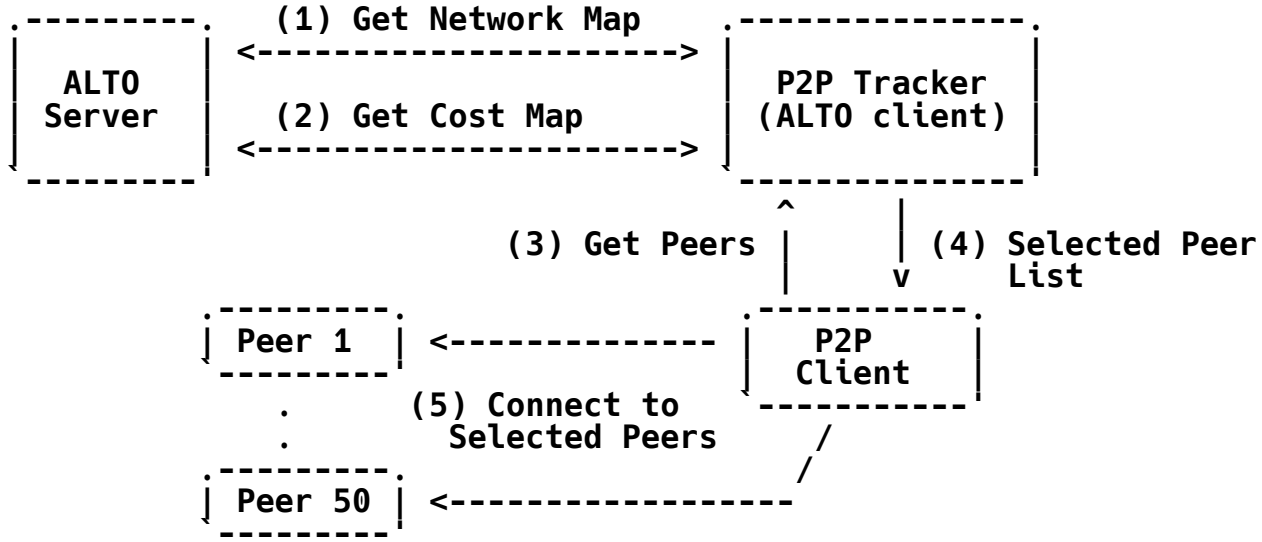

Figure 4: ALTO Client Embedded in P2P Tracker

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 a network map, so that it 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 the 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.

12.2. 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 a 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.

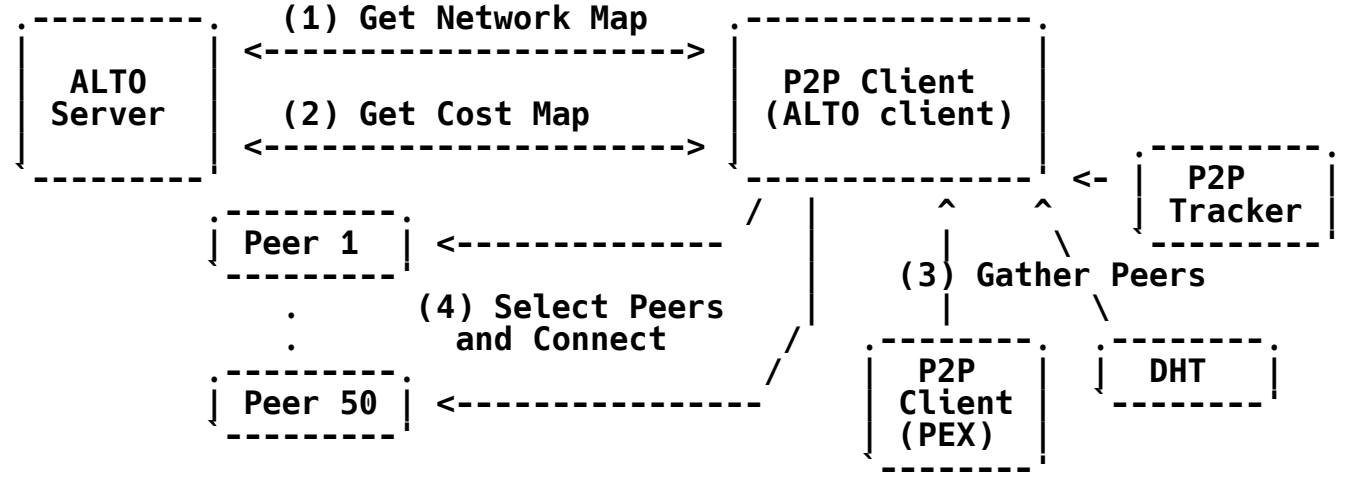

Figure 5: ALTO Client Embedded in P2P Client

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 providing path costs 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.

12.3. 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 embedded in the P2P client gathers a list of known peers in the swarm, and asks the ALTO server to rank them. This document limits the use case to when the P2P client and the ALTO server are deployed by the same entity; hence, the P2P client uses the ranking provided by the ALTO server directly.

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

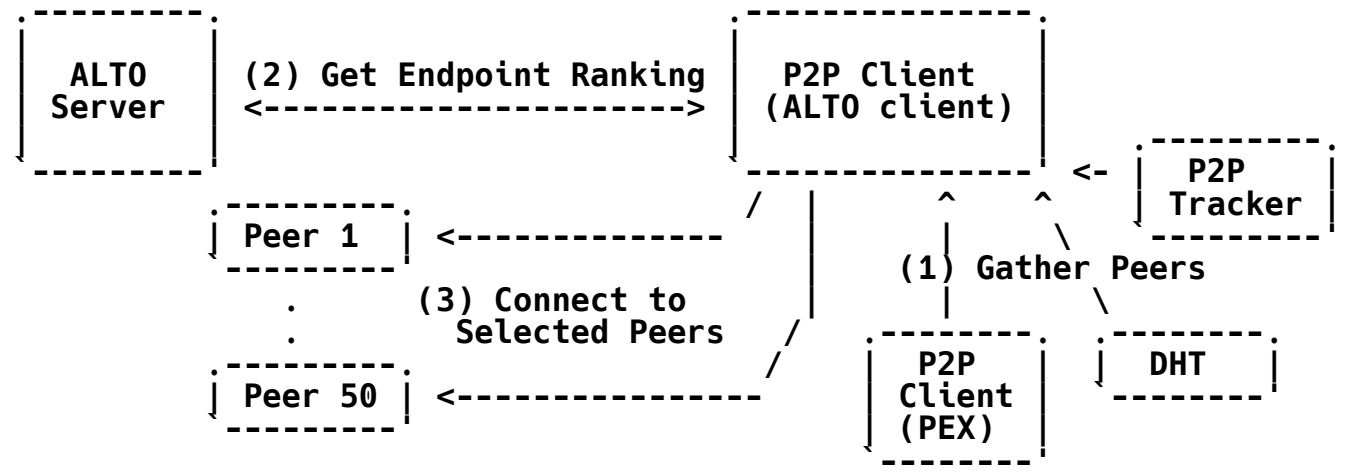

Figure 6: ALTO Client Embedded in P2P Client: Ranking

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 (i.e., the ECS Service), by including the discovered peers as the set of destination endpoints, and indicating 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.

13. Discussions

13.1. 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 9.2) to locate an information resource with the desired ALTO information.

13.2. 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.

13.3. Network Address Translation Considerations

In this day and age of NAT v4<->v4, v4<->v6 [RFC6144], and possibly v6<->v6 [RFC6296], 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 provides 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 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 in which 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.

13.4. 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.

14. IANA Considerations

This document defines registries for application/alto-* media types, ALTO cost metrics, ALTO endpoint property types, ALTO address types, and ALTO error codes. Initial values for the registries and the process of future assignments are given below.

14.1. application/alto-* Media Types

This document registers multiple media types, listed in Table 2.

Type	Subtype	Specification
application	alto-directory+json	Section 9.2.1
application	alto-networkmap+json	Section 11.2.1.1
application	alto-networkmapfilter+json	Section 11.3.1.1
application	alto-costmap+json	Section 11.2.3.1
application	alto-costmapfilter+json	Section 11.3.2.1
application	alto-endpointprop+json	Section 11.4.1.1
application	alto-endpointpropparams+json	Section 11.4.1.1
application	alto-endpointcost+json	Section 11.5.1.1
application	alto-endpointcostparams+json	Section 11.5.1.1
application	alto-error+json	Section 8.5.1

Table 2: ALTO Protocol Media Types

Type name: application

Subtype name: This documents registers 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 [RFC7159].

Security considerations: Security considerations relating to the generation and consumption of ALTO Protocol messages are discussed in Section 15.

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 stand alone or are embedded within other applications.

Additional information:

Magic number(s): n/a

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

Macintosh file type code(s): n/a

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

Intended usage: COMMON

Restrictions on usage: n/a

Author: See Authors' Addresses section.

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

14.2. ALTO Cost Metric Registry

IANA has created and now maintains the "ALTO Cost Metric Registry", listed in Table 3.

Identifier	Intended Semantics
routingcost	See Section 6.1.1.1
priv:	Private 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 IETF Review [RFC5226] to ensure that proper documentation regarding ALTO cost metric semantics and security considerations has been provided. The RFCs documenting the new metrics 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 10.6. Identifiers are to be recorded and displayed as strings.

As specified in Section 10.6, identifiers prefixed with "priv:" are reserved for Private 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 15.3.

14.3. ALTO Endpoint Property Type Registry

IANA has created and now maintains the "ALTO Endpoint Property Type Registry", listed in Table 4.

Identifier	Intended Semantics
pid	See Section 7.1.1
priv:	Private use

Table 4: ALTO Endpoint Property Types

The maintenance of this registry is similar to that of the preceding ALTO cost metrics. That is, the registry is maintained by IANA, subject to the description in Section 10.8.2.

New endpoint property types are assigned after IETF Review [RFC5226] to ensure that proper documentation regarding ALTO endpoint property type semantics and security considerations has been provided. Updates and deletions of ALTO endpoint property types follow the same procedure.

Registered ALTO endpoint property type identifiers MUST conform to the syntactical requirements specified in Section 10.8.1. Identifiers are to be recorded and displayed as strings.

As specified in Section 10.8.1, identifiers prefixed with "priv:" are reserved for Private Use.

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

- Identifier: The name of the desired ALTO endpoint property type.
- Intended Semantics: ALTO endpoint properties carry with them semantics to guide their usage by ALTO clients. Hence, a document defining a new type should provide guidance to both ALTO service providers and applications utilizing ALTO clients as to how values of the registered ALTO endpoint property should be interpreted. For example, if a value refers to a measurement, the measurement units must be documented.
- Security Considerations: ALTO endpoint properties expose information to ALTO clients. ALTO service providers should be made aware of the security ramifications related to the exposure of an endpoint property.

In particular, the request should discuss the sensitivity of the information, and why such sensitive information is required for ALTO-based operations. It may recommend that ISP provide mechanisms for users to grant or deny consent to such information sharing. Limitation to a trust domain being a type of consent bounding.

A request defining new endpoint properties should focus on exposing attributes of endpoints that are related to the goals of ALTO -- optimization of application-layer traffic -- as opposed to more general properties of endpoints. Maintaining this focus on technical, network-layer data will also help extension developers avoid the privacy concerns associated with publishing information about endpoints. For example:

- An extension to indicate the capacity of a server would likely be appropriate, since server capacities can be used by a client to choose between multiple equivalent servers. In addition, these properties are unlikely to be viewed as private information.

- An extension to indicate the geolocation of endpoints might be appropriate. In some cases, a certain level of geolocation (e.g., to the country level) can be useful for selecting content sources. More precise geolocation, however, is not relevant to content delivery, and is typically considered private.

- An extension indicating demographic attributes of the owner of an endpoint (e.g., age, sex, income) would not be appropriate, because these attributes are not related to delivery optimization, and because they are clearly private data.

This specification requests registration of the identifier "pid". Semantics for this property are documented in Section 7.1.1, and security considerations are documented in Section 15.4.

14.4. ALTO Address Type Registry

IANA has created and now maintains the "ALTO Address Type Registry", listed in Table 5.

Identifier	Address Encoding	Prefix Encoding	Mapping to/from IPv4/v6
ipv4	See Section 10.4.3	See Section 10.4.4	Direct mapping to IPv4
ipv6	See Section 10.4.3	See Section 10.4.4	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 IETF Review [RFC5226] to ensure that proper documentation regarding the new ALTO address types and their security considerations has been provided. RFCs defining new address types 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 10.4.2. Identifiers are to be recorded and displayed as 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 10.4.3).
- Endpoint Prefix Encoding: The procedure for encoding a set of addresses of the registered type as an EndpointPrefix (see Section 10.4.4). 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.

14.5. ALTO Error Code Registry

IANA has created and now maintains the "ALTO Error Code Registry". Initial values are listed in Table 1, and recommended usage of the error codes is specified in Section 8.5.2.

Although the error codes defined in Table 1 are already quite complete, future extensions may define new error codes. The "ALTO Error Code Registry" ensures the uniqueness of error codes when new error codes are added.

New ALTO error codes are assigned after IETF Review [RFC5226] to ensure that proper documentation regarding the new ALTO error codes and their usage has been provided.

A request to add a new ALTO error code to the registry MUST include the following information:

- Error Code: A string starting with E_ to indicate the error.

- Intended Usage: ALTO error codes carry with them semantics to guide their usage by ALTO servers and clients. In particular, if a new error code indicates conditions that overlap with those of an existing ALTO error code, recommended usage of the new error code should be specified.

15. 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 [RFC6708] outlines minimum-to-implement authentication and other security requirements. This document considers the following threats and protection strategies.

15.1. Authenticity and Integrity of ALTO Information

15.1.1. Risk Scenarios

An attacker may want to provide false or modified ALTO information resources or an 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 whether an 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 a 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.

15.1.2. 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 (see Section 8.3.5).

ALTO service providers who request server certificates and certification authorities who issue ALTO-specific certificates SHOULD consider the recommendations and guidelines defined in [RFC6125].

Software engineers developing and service providers deploying ALTO should make themselves familiar with possibly updated standards documents as well as up-to-date Best Current Practices on configuring HTTP over TLS.

15.1.3. 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 it may be provided by extension documents.

15.2. Potential Undesirable Guidance from Authenticated ALTO Information

15.2.1. Risk Scenarios

The ALTO services make it possible for an ALTO service provider to influence the behavior of network applications. An ALTO service provider may be hostile to some applications and, hence, try to use ALTO information resources to achieve certain goals [RFC5693]:

> ...redirecting applications to corrupted mediators providing malicious content, or applying policies in computing cost maps based on criteria other than network efficiency.

See [ALTO-DEPLOYMENT] 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 the Endpoint Cost Service 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.

15.2.2. 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 an ALTO 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 [RFC6708]). 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.

15.3. Confidentiality of ALTO Information

15.3.1. Risk Scenarios

In many cases, although ALTO information resources may be regarded as non-confidential information, there are deployment cases in which ALTO information resources can be sensitive information that can pose risks if exposed to unauthorized parties. This document discusses 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 its ALTO network and cost maps. As a result, a sophisticated attacker may be able to infer more fine-grained 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 Section 5.2.1 of [RFC6708], 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 or endpoint addresses) to an unauthorized third party; and risk type (3) Excess retrieval of the ALTO service provider's data by collaborating ALTO clients. [ALTO-DEPLOYMENT] also discusses information leakage from ALTO.

15.3.2. Protection Strategies

To address risk types (1) and (3), 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; hence, the provider should evaluate how much information is revealed and the associated risks.

To address risk type (2), the ALTO Protocol needs 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 is supported to achieve ALTO client authentication to limit the number of parties with whom ALTO information is directly shared. TLS client authentication may also be supported. 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 [ALTO-DEPLOYMENT] for a more detailed discussion on this issue.

See Section 14.3 on guidelines when registering endpoint properties to protect endpoint privacy.

15.3.3. 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.

15.4. Privacy for ALTO Users

15.4.1. 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, the Endpoint Cost, and the 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 [RFC6708], Section 5.2.

15.4.2. Protection Strategies

To protect user privacy, an ALTO client should be cognizant about potential ALTO server tracking through client queries, e.g., by using HTTP cookies. The ALTO Protocol as defined by this document does not rely on HTTP cookies. ALTO clients MAY decide not to return cookies received from the server, in order to make tracking more difficult. However, this might break protocol extensions that are beyond the scope of this document.

An ALTO client may consider the possibility of relying only on ALTO network maps for PIDs and cost maps 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 SHOULD minimize the amount of information sent in IP addresses. For example, the 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.

15.5. Availability of ALTO Services

15.5.1. Risk Scenarios

An attacker may want to disable the ALTO services of a network as a way to disable network guidance to large scale applications. In particular, queries that 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).

15.5.2. Protection Strategies

The ALTO service 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 [SIP] may be considered in an extension document.

An ALTO service 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.

16. 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.

16.1. Operations

16.1.1. 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 maps 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 11.2.2), 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 computational and network load of serving full maps to a large number of ALTO clients 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 [ALTO-SERVER-DISC].

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

16.1.2. 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.

16.1.3. Dependencies 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., [ALTO-SVR-APIS]) and are expected to be provided in extension documents.

16.1.4. Impact and Observation on Network Operation

ALTO presents a new opportunity for managing network traffic by providing additional information to clients. In particular, the deployment of an ALTO server may shift network traffic patterns, and the potential impact to network operation can be large. An ALTO service provider should ensure that appropriate information is being exposed. Privacy implications for ISPs are discussed in Section 15.3.

An ALTO service provider should consider how to measure impacts on (or integration with) traffic engineering, in addition to monitoring correctness and responsiveness of ALTO servers. The measurement of impacts can be challenging because ALTO-enabled applications may not provide related information back to the ALTO service provider. Furthermore, the measurement of an ALTO service provider may show that ALTO clients are not bound to ALTO server guidance as ALTO is only one source of information.

While it can be challenging to measure the impact of ALTO guidance, there exist some possible techniques. In certain trusted deployment environments, it may be possible to collect information directly from ALTO clients. It may also be possible to vary or selectively disable ALTO guidance for a portion of ALTO clients either by time, geographical region, or some other criteria to compare the network traffic characteristics with and without ALTO.

Both ALTO service providers and those using ALTO clients should be aware of the impact of incorrect or faked guidance (see [ALTO-DEPLOYMENT]).

16.2. Management

16.2.1. 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].

16.2.2. 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.

16.2.3. Fault Management

An ALTO service provider should monitor whether any ALTO servers have failed. See Section 16.2.5 for related metrics that may indicate server failures.

16.2.4. 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 resources is derived. This can be either raw network information (e.g., from routing elements) or pre-processed ALTO-level information in the forms of network maps, cost maps, etc.

- Algorithms for computing the ALTO information returned to clients. These could return either 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, TLS client and server authentication, TLS encryption parameters) intended to serve the same information should be monitored for consistency.

16.2.5. Performance Management

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

- Requests and responses for each service listed in an 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).

16.2.6. Security Management

Section 15 documents ALTO-specific security considerations. Operators should configure security policies with those in mind. Readers should refer to HTTP [RFC7230] and 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.

17. References

17.1. Normative References

[RFC1812] Baker, F., "Requirements for IP Version 4 Routers", RFC 1812, June 1995.

[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.

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

[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.

[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.

[RFC7230] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, June 2014.

17.2. Informative References

[ALTO-DEPLOYMENT]
Stiemerling, M., Ed., Kiesel, S., Ed., Previdi, S., and M. Scharf, "ALTO Deployment Considerations", Work in Progress, February 2014.

[ALTO-INFOEXPORT]
Shalunov, S., Penno, R., and R. Woundy, "ALTO Information Export Service", Work in Progress, October 2008.

[ALTO-MULTI-PS]
Das, S., Narayanan, V., and L. Dondeti, "ALTO: A Multi Dimensional Peer Selection Problem", Work in Progress, October 2008.

[ALTO-QUERYRESPONSE]
Das, S. and V. Narayanan, "A Client to Service Query Response Protocol for ALTO", Work in Progress, March 2009.

[ALTO-SERVER-DISC]
Kiesel, S., Stiemerling, M., Schwan, N., Scharf, M., and H. Song, "ALTO Server Discovery", Work in Progress, September 2013.

[ALTO-SVR-APIS]
Medved, J., Ward, D., Peterson, J., Woundy, R., and D. McDysan, "ALTO Network-Server and Server-Server APIs", Work in Progress, March 2011.

[ALTO-USE-CASES]
Niven-Jenkins, B., Watson, G., Bitar, N., Medved, J., and S. Previdi, "Use Cases for ALTO within CDNs", Work in Progress, June 2012.

[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.

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

[P4P-FRAMEWORK]
Alimi, R., Pasko, D., Popkin, L., Wang, Y., and Y. Yang, "P4P: Provider Portal for P2P Applications", Work in Progress, November 2008.

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

[P4P-SPEC] Wang, Y., Alimi, R., Pasko, D., Popkin, L., and Y. Yang, "P4P Protocol Specification", Work in Progress, March 2009.

[PROXIDOR] Akonjang, O., Feldmann, A., Previdi, S., Davie, B., and D. Saucez, "The PROXIDOR Service", Work in Progress, March 2009.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

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

[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.

[RFC6296] Wasserman, M. and F. Baker, "IPv6-to-IPv6 Network Prefix Translation", RFC 6296, June 2011.

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

[RFC7159] Bray, T., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, March 2014.

[RFC7231] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, June 2014.

[SIP] Jennings, C., "Computational Puzzles for SPAM Reduction in SIP", Work in Progress, July 2007.

Appendix A. Acknowledgments

Thank you to Jan Seedorf (NEC) for substantial contributions to the Security Considerations section. Ben Niven-Jenkins (Velocix), Michael Scharf, and Sabine Randriamasy (Alcatel-Lucent) 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).

Stanislav Shalunov would like to thank BitTorrent, where he worked while contributing to ALTO development.

Appendix B. Design History and Merged Proposals

The ALTO Protocol specified in this document consists of contributions from

- P4P [P4P-FRAMEWORK], [P4P-SIGCOMM08], [P4P-SPEC];
- ALTO Info-Export [ALTO-INFOEXPORT];
- Query/Response [ALTO-QUERYRESPONSE], [ALTO-MULTI-PS]; and
- Proxidor [PROXIDOR].

Authors' Addresses

Richard Alimi (editor)
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
USA

EMail: ralimi@google.com

Reinaldo Penno (editor)
Cisco Systems, Inc.
170 West Tasman Dr
San Jose, CA 95134
USA

EMail: repenno@cisco.com

Y. Richard Yang (editor)
Yale University
51 Prospect St
New Haven, CT 06511
USA

EMail: yry@cs.yale.edu

Sebastian Kiesel
University of Stuttgart Information Center
Networks and Communication Systems Department
Allmandring 30
Stuttgart 70550
Germany

EMail: ietf-alto@skiesel.de

Stefano Previdi
Cisco Systems, Inc.
Via Del Serafico, 200
Rome 00142
Italy

EMail: sprevidi@cisco.com

Wendy Roome
Alcatel-Lucent
600 Mountain Ave.
Murray Hill, NJ 07974
USA

EMail: w.roome@alcatel-lucent.com

Stanislav Shalunov
Open Garden
751 13th St
San Francisco, CA 94130
USA

EMail: shalunov@shlang.com

Richard Woundy
Comcast Cable Communications
One Comcast Center
1701 John F. Kennedy Boulevard
Philadelphia, PA 19103
USA

EMail: Richard_Woundy@cable.comcast.com