ALTO Protocol
draft-ietf-alto-protocol-23.txt

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

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

Requirements Language

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

Status of this Memo

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

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

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

This Internet-Draft will expire on June 12, 2014.

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 modern
   network applications can be adaptive, and hence become more network-
   efficient (e.g., reduce network resource consumption) and achieve
   better application performance (e.g., accelerated download rate), by
   leveraging network-provided information.

   At a high level, the ALTO Protocol specified in this document is a
   unidirectional 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. When a network provides information through the ALTO
   Protocol, we say that the network provides the ALTO Service.

   To better understand the goal of the ALTO Protocol, we provide a
   short, non-normative overview of the benefits of ALTO to both
   networks and applications:

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

   o An application that uses an ALTO Service can benefit from better
      knowledge of the network to avoid network bottlenecks. For
      example, an overlay application can use information provided by
      the ALTO Service 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.

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

2. Terminology

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

We also use the following additional terms: Endpoint Address, Network
Location, ALTO Information, ALTO Information Base, and ALTO Service.

2.1. Endpoint

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

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

2.2. Endpoint Address

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

Each Endpoint Address has an associated Address Type, which indicates
both its syntax and semantics.

2.3. Network Location

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

2.4. ALTO Information

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

2.5. ALTO Information Base

We use 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.

2.6. ALTO Service

A network that provides ALTO Information through the ALTO Protocol is
said to provide the ALTO Service.

3. Architecture

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

3.1. ALTO Service and Protocol Scope

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

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

note that the "my-Internet view" defined in this document does not
specify the internal topology of a network, and hence, we say that it
provides a "single-switch" abstraction. Extensions to this document
may provide topology details in "my-Internet view".

To better understand the ALTO Service and the role of the ALTO
Protocol, we show in Figure 1 the overall ALTO system architecture.
In this architecture, an ALTO Server prepares ALTO Information; an
ALTO Client uses ALTO Service Discovery to identify an appropriate
ALTO Server; and the ALTO Client requests available ALTO Information
from the ALTO Server using the ALTO Protocol.

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

```
+----------------------------------------------------------------------+
|                         Network Region                               |
|                                                                      |
|                  +-----------+                                       |
|                  | Routing   |                                       |
|   +--------------+ | Protocols |                                     |
|   | Provisioning |  +-----------+                                    |
|   | Policy       |        |                                          |
|   +--------------+\       |                                          |
|                    \      |                                          |
|                     \     |                                          |
|   +-----------+      \+---------+              +--------+  |
|   |Dynamic    |       | ALTO    | ALTO Protocol | ALTO   |  |
|   |Network    |.......| Server  | ================== | Client |  |
|   |Information|       +---------+              +--------+  |
|   +-----------+      /                              /       |
|                    /         ALTO SD Query/Response /       |
|                   /                              /          |
|          +---------+                 +----------------+     |
|          | External |                | ALTO Service   |     |
|          | Interface|                | Discovery (SD) |     |
|          +----------+                +----------------+     |
|               |                                             |
+----------------------------------------------------------------------+
            |
      +------------------+
      | Third Parties    |
      |                  |
      | Content Providers|
      +------------------+
```
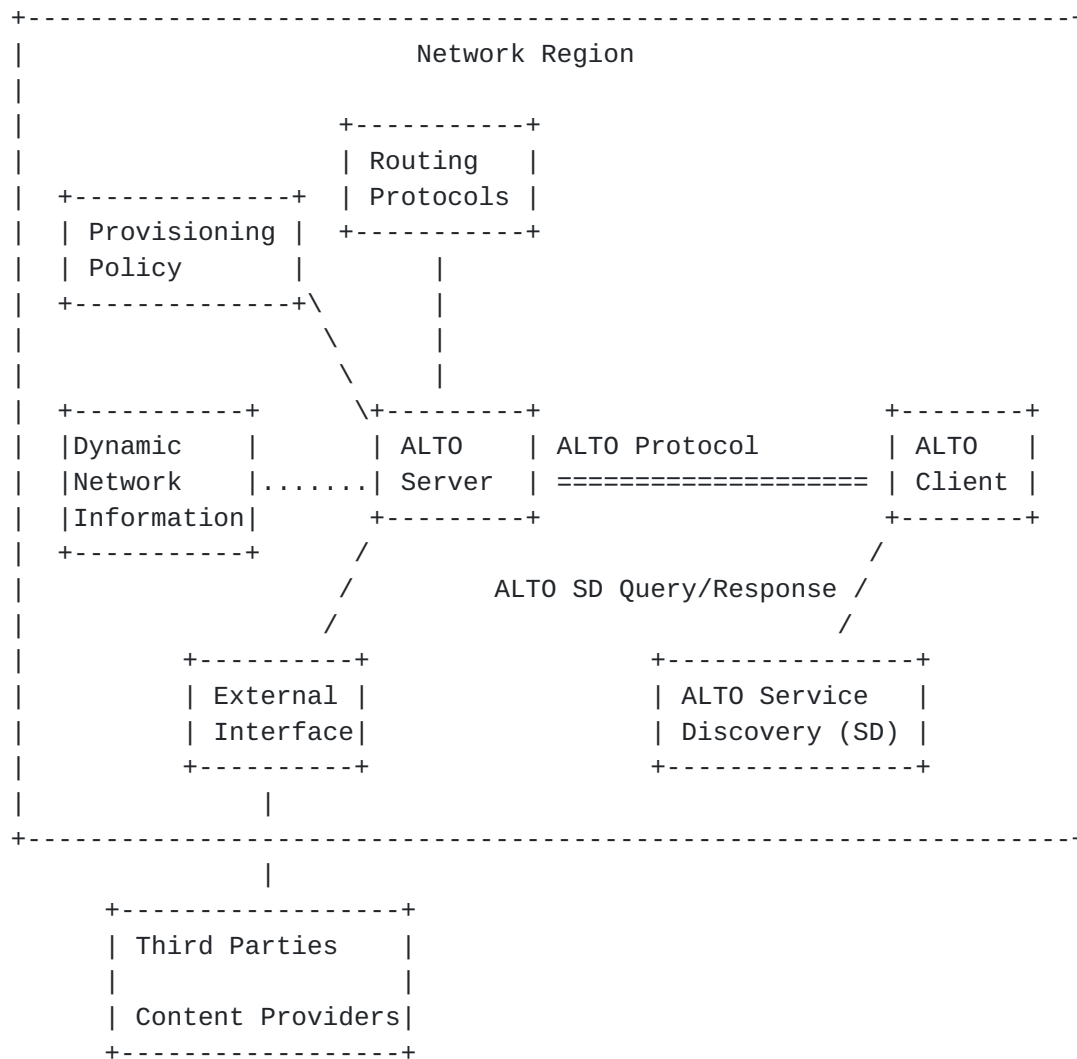
                    Figure 1: Basic ALTO Architecture.

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

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

4. ALTO Information Service Framework

The ALTO Protocol conveys network information through services, where
each service defines a set of related functionalities. An ALTO
Client can 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 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 endhosts.
The aforementioned goals are achieved by defining the Map Service,
which provides the core ALTO information to clients, and three
additional services: the Map Filtering Service, Endpoint Property
Service, and Endpoint Cost Service. Additional services can be
defined in companion documents. Below we give an overview of the
services. Details of the services will be presented in the following
sections.

```
.-------------------------------------------.
| ALTO Information Services                 |
| .-----------. .----------. .----------.   |
| |    Map    | | Endpoint | | Endpoint |   |
| | Filtering | | Property | |   Cost   |   |
| |  Service  | | Service  | | Service  |   |
| `-----------' `----------' `----------'   |
| .-------------------------------------.   |
| |   Map Service                       |   |
| |   .-------------.  .--------------.  |   |
| |   | Network Map |  |   Cost Map   |  |   |
| |   `-------------'  `--------------'  |   |
| `-------------------------------------'   |
`-------------------------------------------'
```
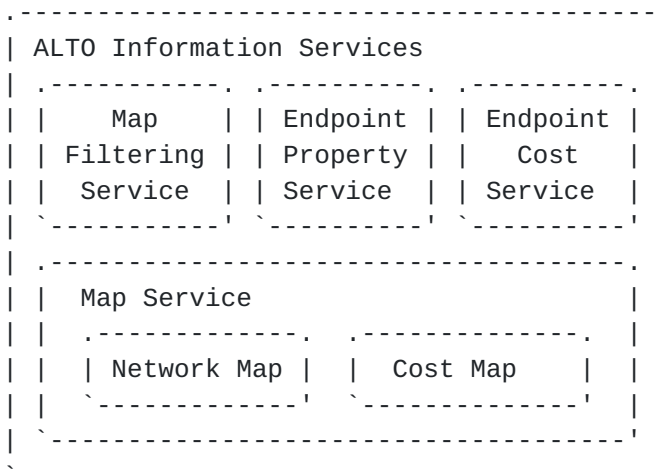
Figure 2: ALTO Service Framework.

4.1. ALTO Information Services

4.1.1. Map Service

The Map Service provides batch information to ALTO Clients in the
form of Network Map and Cost Map. A 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. A Cost Map
(see Section 6) provides costs between a defined groupings.

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

4.1.2. Map Filtering Service

   Resource constrained ALTO Clients may benefit from filtering of query
   results at the ALTO Server. This avoids that an ALTO Client 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 Network Map and Cost Map
   based on additional parameters.

4.1.3. Endpoint Property Service

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

4.1.4. Endpoint Cost Service

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

5. Network Map

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

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

5.1. Provider-defined Identifier (PID)

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

ALTO introduces provider-defined Network Location identifiers called Provider-defined Identifiers (PIDs) to provide an indirect and network-agnostic way to specify an aggregation of network endpoints that may be treated similarly, based on network topology, type, or other properties. Specifically, a PID is a US-ASCII string of type PIDName (see Section 10.1) and its associated set of Endpoint Addresses. As we discussed above, there can be many different ways of grouping the endpoints and assigning PIDs. For example, a PID may denote a subnet, a set of subnets, a metropolitan area, a PoP, an autonomous system, or a set of autonomous systems. Interpreting the PIDs defined in a Network Map using the "single-switch" 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 (Point-of-Presence) in a P2P application communicate locally instead of communicating with endpoints in other PoPs. The ISP may aggregate endhosts within a PoP into a single PID in the Network Map. The cost may be encoded to indicate that Network Locations within the same PID are preferred; for example, cost(PID_i, PID_i) == c and cost(PID_i, PID_j) > c for i != j. Section 6 provides further details on using PIDs to represent costs in an ALTO Cost Map.

5.2. Endpoint Addresses

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

We use the Network Map shown in Figure 3 in most examples of this document.

```
.--------------------------------------------------------------------.
| An ALTO Network Map                                                |
|                                                                    |
|   .-----------------------------------.  .-----------------.  |
|   | NetLoc: PID-1                     |  | NetLoc: PID-3   | |
|   |   .------------------------------. |  |                 | |
|   |   | 192.0.2.0/24                 | |  |  .-----------.  | |
|   |   | .---------------------------. | |  |  | 0.0.0.0/0 |  | |
|   |   | | Endpoint: 192.0.2.34      | | |  |  `-----------`  | |
|   |   | `---------------------------` | |  |                 | |
|   |   `------------------------------` |  |                 | |
|   |   .------------------------------. |  |                 | |
|   |   | 198.51.100.0/25              | |  |                 | |
|   |   | .---------------------------. | |  |                 | |
|   |   | | Endpoint: 198.51.100.100 | | |  |                 | |
|   |   | `---------------------------` | |  |                 | |
|   |   `------------------------------` |  |                 | |
|   `-----------------------------------` |  |                 | |
|                                         |  |                 | |
|   .-----------------------------------. |  |                 | |
|   | NetLoc: PID-2                     | |  |                 | |
|   |   .------------------------------. |  |                 | |
|   |   | 198.51.100.128/25            | |  |                 | |
|   |   `------------------------------` |  |                 | |
|   `-----------------------------------` |  `-----------------` |
|                                                                    |
`--------------------------------------------------------------------`
```
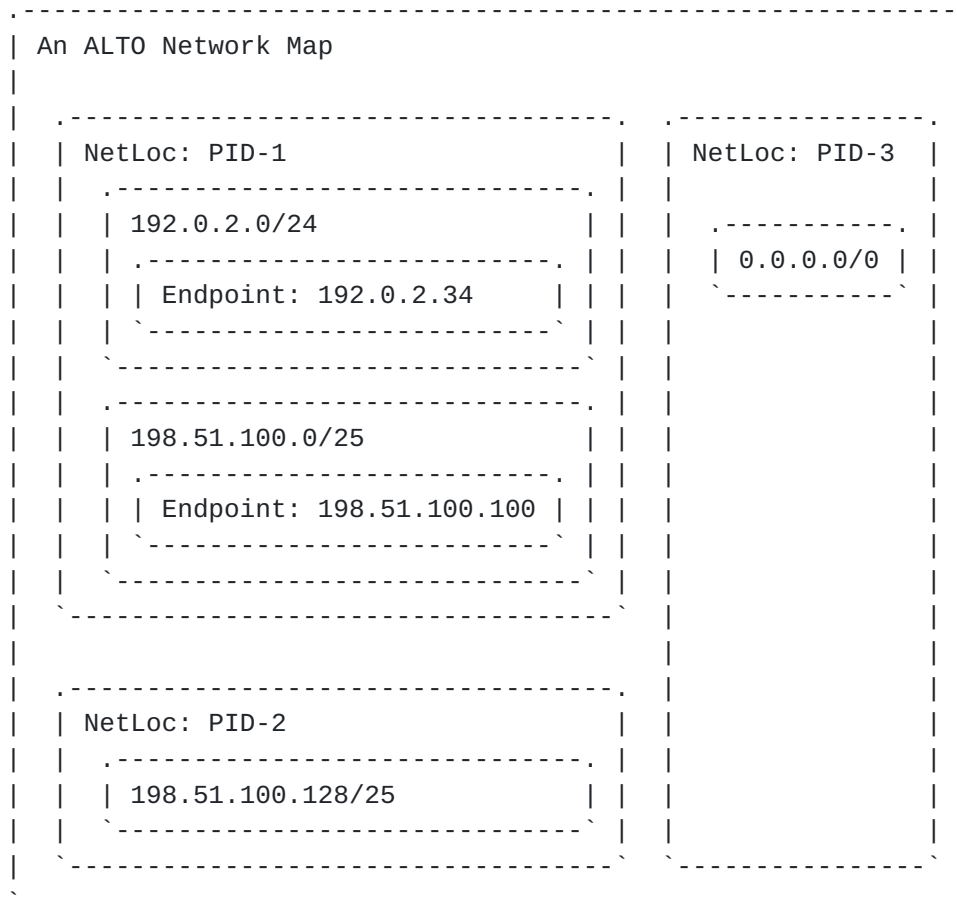
Figure 3: Example Network Map.

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 Network Map, an ALTO Cost Map defines Path Costs pairwise
amongst sets of source and destination Network Locations defined by
PIDs defined 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 Resource Consumer as the destination.

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

As used in this document, a 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:

o Metric: identifies what the costs represent;

o Mode: identifies how the costs should be interpreted.

The combination of a metric and a mode defines a Cost Type. Certain
queries for Cost Maps allow the ALTO Client to indicate the desired
Cost Type. 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 Cost Maps with the same Cost Type, Network Map pair.

6.1.1. Cost Metric

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

Cost metrics are indicated in protocol messages as strings.

6.1.1.1. Cost Metric: routingcost

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

This Cost Metric conveys a generic measure for the cost of routing
traffic from a source to a destination. 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 these semantics.

[6.1.2](#). **Cost Mode**

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

   It is important to communicate such information to ALTO Clients, as
   certain operations may not be valid on certain costs returned by an
   ALTO Server. For example, it is possible for an ALTO Server to
   return a set of IP addresses with costs indicating a ranking of the
   IP addresses. Arithmetic operations that would make sense for
   numerical values, do not make sense for ordinal rankings. ALTO
   Clients may handle such costs differently.

   Cost Modes are indicated in protocol messages as strings.

   An ALTO Server MUST support at least one of 'numerical' and 'ordinal'
   modes. An ALTO Client SHOULD be cognizant of operations when a
   desired Cost Mode is not supported. For example, an ALTO Client
   desiring numerical costs may adjust its behaviors if only the ordinal
   Cost Mode is available. Alternatively, an ALTO Client desiring
   ordinal costs may construct ordinal costs 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 costs values in a Cost Map are a ranking (relative
   to all other values in a Cost Map), with a lower value indicating a
   higher preference. The values are non-negative integers. Ordinal
   cost values in a Cost Map need not be unique nor contiguous. In
   particular, it is possible that two entries in a map have an
   identical rank (ordinal cost value). This document does not specify
   any behavior by an ALTO Client in this case; an ALTO Client may
   decide to break ties by random selection, other application
   knowledge, or some other means.

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

6.2. Cost Map Structure

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

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

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

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

6.3. Network Map and Cost Map Dependency

   A Cost Map gives Path Costs between the PIDs defined in a Network
   Map. An ALTO Server may modify a Network Map at any time, say by
   adding or deleting PIDs, or even redefining them. Hence to
   effectively use an instance of a Cost Map, an ALTO Client must know
   which version of the Network Map defined the PIDs in that Cost Map.
   Version Tags allow ALTO Clients to correlate Cost Map instances with
   the corresponding versions of the Network Map.

   A Version Tag is a tuple of (1) an ID for the resource (e.g., a
   Network Map), and (2) a tag (an opaque string) associated with the
   version of that resource. A Network Map distributed by an ALTO
   Server includes its Version Tag. A Cost Map referring to PIDs also
   includes Version Tag for the Network Map on which it is based.

   Two Network Maps are the same if they have the same Version Tag.
   Whenever the content of the Network Map maintained by an ALTO Server
   changes, 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 a 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.

7.1.1. Endpoint Property Type: pid

An ALTO Server MUST define the 'pid' Endpoint Property Type for each
Network Map that it provides.

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:

o Information Resources: An ALTO Server provides 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.

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

8.2. Notation

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

   We use an adaptation of the C-style struct notation to define the
   fields (names/values) of JSON objects. An optional field is enclosed
   by [ ], and 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 we write * for n, it means no upper bound. In the
   definitions, the JSON names of the fields are case sensitive.

   For example, the definition below defines a new type Type4, with
   three field members (or fields for short) 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;

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

   We use 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 document these as necessary.

8.3. Basic Operations

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

[RFC2616].

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 the Network
Map) 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, an ALTO Client requests
Information Resource Directories. Informally, an Information
Resource Directory enumerates URIs at which an ALTO Server offers
Information Resources.

Specifically, using the ALTO Discovery protocol, an ALTO Client
obtains a URI through which it can request an Information Resource
Directory (IRD). We refer 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. We refer to this set of Information Resources 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 on 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.

8.3.3. Server Responding to IR Request

   Upon receiving a request for an Information Resource that the ALTO
   Server can provide, the ALTO Server MUST return the requested
   Information Resource. In other cases, to be more informative
   ([I-D.ietf-httpbis-p2-semantics]), the ALTO Server MAY provide the
   ALTO Client with an Information Resource Directory indicating how to
   reach the desired information resource, or return an ALTO error
   object; see Section 8.5 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 [RFC2616]).
   Caching of POST requests is not widely implemented by HTTP
   intermediaries, however an alternative approach is for an ALTO
   Server, in response to POST requests, to return an HTTP 303 status
   code ("See Other") indicating to the ALTO Client that the resulting
   Information Resource is available via a GET request to an alternate
   URL. HTTP intermediaries that do not support caching of POST
   requests could then cache the response to the GET request from the
   ALTO Client following the alternate URL in the 303 response if the
   response to the subsequent GET request contains explicit freshness
   information.

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

8.3.4. Client Handling Server Response

8.3.4.1. Using Information Resources

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

8.3.4.2. Handling 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 for the media type.

8.3.4.3. Handling Error Conditions

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

8.3.5. Authentication and Encryption

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

8.3.6. Information Refreshing

An ALTO Client MAY determine the frequency at which ALTO Information
is refreshed based on information made available via HTTP.

8.3.7. HTTP Cookies

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

8.3.8. 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 meta field providing
   meta definitions, and another field containing the data, if needed.

   Specifically, we define 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. In this document, we derive five types from ResponseEntityBase
   to add different types of data component: InforResourceDirectory
   (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 there an error processing a request, an ALTO Server SHOULD
   return additional ALTO-layer information, if it is available, in the
   form of an ALTO Error Resource encoded in the HTTP response' entity
   body. If no ALTO-layer information is available, an ALTO Server may
   omit an ALTO Error resource from the response.

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

8.5.1. 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 the "code" key in the "meta"
field of the response. The value of "code" MUST be an ALTO Error
Code defined in Table 1. Note that the ALTO Error Codes defined in
Table 1 are limited to support the error conditions needed for
purposes of this document. Additional status codes may be defined in
companion or extension documents.

```
+-----------------------+-------------------------------------------+
| ALTO Error Code       | Description                               |
+-----------------------+-------------------------------------------+
| E_SYNTAX              | Parsing error in request (including       |
|                       | identifiers)                              |
| E_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 key "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
such as line number and offset within line about the syntax error.
If nothing else, "syntax-error" could have 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 Network Map Filtering (Section 11.3.1.3),
Cost Map Filtering (Section 11.3.2.3), Endpoint Properties
(Section 11.4.1.3), and Endpoint Cost (Section 11.5.1.3). For an
E_MISSING_FIELD error, the server may include an optional "field" key
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 CostMap request
(see Section 11.3.2.3) misses the "cost-metric" field in the request.
The error response from the ALTO Server may specify the "field" key
as "cost-type/cost-mode".

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" key 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 cost map
filtering request may specify a wrong value of 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" key in the "meta" field of the response, to indicate the
field that contains the wrong value. The server may also include an
optional "value" key in the "meta" field of the response to indicate
the wrong value that triggered the error.

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" key in the "meta"
field of the response, to indicate the field that contains the wrong
value. The server may also include an optional "value" key in the
"meta" field of the response to indicate the wrong value that
triggered the error. If the "value" key is specified, the "field"
key MUST be specified. The "value" key MUST have a JSONString value.
If the invalid value is not a string, the ALTO Server MUST convert it
to a string. See Appendix C for the rules to specify the "value"
key.

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:

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

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

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

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

9. Protocol Specification: Information Resource Directory

As we 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 five attributes
associated with it, including its assigned ID, its response format,
its capabilities, its accepted input parameters, and other resources
that it may depend on. 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 an ID that is unique amongst all Information Resources in
the Information Resource Closure of the client. The ID SHOULD remain
stable even when the data provided by that resource changes. For

example, even though the number of PIDs in a Network Map may be
adjusted, its Resource ID should remain the same. Similarly, if the
entries in a Cost Map are updated, its Resource ID should remain the
same. IDs SHOULD NOT be re-used for different resources over time.

9.1.2. Media Type

ALTO uses Media Type [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 an Information Resource is a Media Type, which
indicates how the Client specifies the input parameters as contained
in the entity body of the HTTP POST request.

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 Information Resource Directory to publish
available Information Resources and their aforementioned attributes.
Since resource selection happens after consumption of the Information
Resource Directory, the format of the Information Resource Directory
is designed to be simple with the intention of future ALTO Protocol
versions maintaining backwards compatibility. Future extensions or
versions of the ALTO Protocol SHOULD be accomplished by extending
existing media types or adding new media types, but retaining the
same format for the Information Resource Directory.

An ALTO Server MUST make an Information Resource Directory available

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

9.2.1. Media Type

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

9.2.2. Encoding

An Information Resource Directory response may include in "meta" the
"cost-types" key, 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 the "default-alto-network-map"
key, which specifies the Resource ID of a Network Map. When there are
multiple Network Maps defined in an IRD (e.g., with different levels
of granularity), the "default-alto-network-map" key 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 Information Resource (see
      Section 9.1.2) available via GET or POST requests to the
      corresponding URI or "application/alto-directory+json", which
      indicates that the response for a request to the URI will be an
      Information Resource Directory for URIs discoverable via the URI.

   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, a 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: TBA
    Content-Type: application/alto-directory+json

    {
      "meta" : {
        "cost-types": {
          "num-routing": {
              "cost-mode" : "numerical",
              "cost-metric": "routingcost",
              "description": "My default"
          },

```
      "num-hop":      {
         "cost-mode"  : "numerical",
         "cost-metric": "hopcount"
      },
      "ord-routing": {
         "cost-mode"  : "ordinal",
         "cost-metric": "routingcost"
      },
      "ord-hop":      {
         "cost-mode"  : "ordinal",
         "cost-metric": "hopcount"
      }
   },
   "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" key 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 key 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 a 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" key of "meta" of the IRD. Hence, one can verify that the
Endpoint Cost Information Resource supports both Cost Metrics

'routingcost' and 'hopcount', each available for both 'numerical' and
'ordinal'. When requesting the Information Resource, an ALTO Client
can specify cost constraints, as indicated by the "cost-constraints"
field of the "capabilities" attribute.

9.2.4. Delegation using IRD

ALTO Information Resource Directory provides flexibility to provide
ALTO Service (e.g., delegation to another domain). 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: TBA
Content-Type: application/alto-directory+json

{
  "meta" : {
    "cost-types": {
      "num-routing": {
        "cost-mode"  : "numerical",
        "cost-metric": "routingcost",
        "description": "My default"
      },
      "num-hop":      {
        "cost-mode"  : "numerical",
        "cost-metric": "hopcount"
      },
      "ord-routing": {
        "cost-mode"  : "ordinal",
        "cost-metric": "routingcost"
```

```
            },
            "ord-hop":       {
               "cost-mode"  : "ordinal",
               "cost-metric": "hopcount"
            }
         }
      },
      "resources" : {
         "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 Network Map, 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 regards to how they process an Information Resource
   Directory to identify the URI corresponding to a desired Information
   Resource. However, some advice is provided for implementors.

   It is possible that multiple entries in the directory match a desired
   Information Resource. For instance, in the example in Section 9.2.3,
   a full Cost Map with "numerical" Cost Mode and "routingcost" Cost
   Metric could be retrieved via a GET request to
   "http://alto.example.com/costmap/num/routingcost", or via a POST
   request to "http://custom.alto.example.com/costmap/filtered".

   In general, it is preferred for ALTO Clients to use GET requests
   where appropriate, since it is more likely for responses to be
   cachable. 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 US-ASCII string. The string MUST be no
   more than 64 characters, and MUST NOT contain characters other than
   alphanumeric characters (code points 0x30-0x39, 0x41-0x5A, and 0x61-
   0x7A), the hyphen ('-', code point 0x2D), the colon (':', code point
   0x3A), the at ('@', code point 0x40), the underline ('_', code point
   0x5F), or the '.' separator (code point 0x2E). 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 an particular resource (e.g., a
   Network Map) within an ALTO Server (see Section 9.2).

   A Resource ID is encoded as a US-ASCII string with the same format as
   that of 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;

   The 'resource-id' attribute is the Resource ID of a resource (e.g., a
   Network Map) defined in the Information Resource Directory, and 'tag'
   is a case-sensitive US-ASCII string. The 'tag' string MUST be no
   more than 64 characters, and MUST NOT contain any ASCII character
   below 0x21 or above 0x7E.

   Two values of the VersionTag are equal if and only if both the the
   'resource-id' attributes are byte-for-byte equal and the 'tag'
   attributes are byte-for-byte equal.

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. We refer
   to such addresses as Typed Endpoint Addresses.

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

10.4.2. Address Type

   The AddressType component of TypedEndPointAddr is defined as an US-
   ASCII string consisting of only alphanumeric characters (code points
   0x30-0x39, 0x41-0x5A, and 0x61-0x7A). 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 an
   US-ASCII 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 US-ASCII strings. The exact
characters and format depend on the type of endpoint address.

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

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 US-ASCII string. The string MUST either
   have the value 'numerical' or 'ordinal'.

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

10.6. Cost Metric

   A Cost Metric is encoded as a US-ASCII string. The string MUST be no
   more than 32 characters, and MUST NOT contain characters other than
   alphanumeric characters (code points 0x30-0x39, 0x41-0x5A, and 0x61-
   0x7A), the hyphen ('-', code point 0x2D), the colon (':', code point
   0x3A), the underline ('_', code point 0x5F), or the '.' separator
   (0x2E). The '.' separator is reserved for future use and MUST NOT be
   used unless specifically indicated by a companion or extension
   document.

   Identifiers prefixed with 'priv:' are reserved for Private Use
   [RFC5226]. Identifiers prefixed with 'exp:' are reserved for
   Experimental use. For an identifier with the 'priv:' or 'exp:'
   prefix, an additional string (e.g., company identifier or random
   string) MUST follow to reduce potential collisions. For example, a
   short string after 'exp:' to indicate the starting time of a specific
   experiment is recommended. All other identifiers 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 Metrics
   registry Section 14.2.

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

10.7. Cost Type

   The combination of a CostMetric and a CostMode defines a CostType:

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

'description', if present, MUST contain a US-ASCII string with a
human-readable description of the cost-metric and cost-mode. An ALTO
Client MAY present this string to a developer, as part of a discovery
process. But the field SHOULD NOT be interpreted by an ALTO Client.

10.8. Endpoint Property

We distinguish two types of Endpoint Properties: Resource Specific
Endpoint Properties and Global Endpoint Properties. The type
'EndpointPropertyType' is used in this document to indicate a US-
ASCII string denoting either a Resource Specific Endpoint Property or
a Global Endpoint Property.

10.8.1. Resource Specific Endpoint Properties

We define only one Resource Specific Endpoint Property in this
document: pid. It has the following format: a Resource ID, followed
by the '.' separator (0x2E), followed by "pid". An example is "my-
default-networkmap.pid".

10.8.2. Global Endpoint Properties

An Global Endpoint Property is encoded as a US-ASCII string. The
string MUST be no more than 32 characters, and MUST NOT contain
characters other than alphanumeric characters (code points 0x30-0x39,
0x41-0x5A, and 0x61-0x7A), the hyphen ('-', code point 0x2D), the
colon (':', code point 0x3A), or the underline ('_', code point
0x5F). 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]. Identifiers prefixed with 'exp:' are reserved for
Experimental use. For an identifier with the 'priv:' or 'exp:'
prefix, an additional string (e.g., company identifier or random
string) MUST follow to reduce potential collisions. For example, a
short string after 'exp:' to indicate the starting time of a specific
experiment is recommended. All other identifiers for Endpoint
Properties appearing in an HTTP request or response with an
'application/alto-*' media type MUST be registered in the ALTO
Endpoint Property registry Section 14.3.

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, we define two generic keys: "vtag", which is the Version
Tag (see Section 10.3) of the current Information Resource; and
"dependent-vtags", which is an array of Version Tags, to indicate the
Version Tags of the resources that this resource depends on.

11.2. Map Service

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

11.2.1. Network Map

A 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 Network Map is "application/alto-networkmap+json".

11.2.1.2. HTTP Method

A 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 a Network Map response MUST include "vtag", which
is the Version Tag of the retrieved Network Map.

The data component of a 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, and each value representing the associated set of endpoint
addresses of a 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: TBA
Content-Type: application/alto-networkmap+json

{
  "meta" : {
    "vtag": {
      "resource-id": "my-default-network-map",
       "tag": "1266506139"
    }
  },
  "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 a 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 a 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 the 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 a 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 [RFC1812] MUST be used.

   To ensure that the longest-prefix matching algorithm yields one and
   only one PID, Network Maps 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. We refer to this as the completeness property of such a 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. We refer to this as the non-overlapping property of such a
   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 the IP address belongs to the PID
   containing the identified longest prefix.

   The following example shows a complete and non-overlapping 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

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

11.2.3.1. Media Type

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

11.2.3.2. HTTP Method

A 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 key "cost-types" in "meta" 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 CostType 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" key, 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 "cost-
   type", to indicate 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 a Path Cost is not defined (e.g.,
both the Source and Destination PIDs contain addresses outside of the
Network Provider's administrative domain).

Similar to Network Map, the encoding of Cost Map 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: TBA
Content-Type: application/alto-costmap+json

{
  "meta" : {
    "dependent-vtags" : [
      {"resource-id": "my-default-network-map",
       "tag": "1266506139"
      }
    ],
    "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, we considered array-based encoding
for "map", but chose the current encoding for clarity.

11.3. **Map Filtering Service**

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

11.3.1. **Filtered Network Map**

   A Filtered Network Map is a Network Map Information Resource
   (Section 11.2.1) for which an ALTO Client may supply a list of PIDs
   to be included. A Filtered Network Map MAY be provided by an ALTO
   Server.

11.3.1.1. **Media Type**

   Since a Filtered Network Map is still a Network Map, it uses the
   media type defined for Network Map at Section 11.2.1.1.

11.3.1.2. **HTTP Method**

   A Filtered 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 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 Map. 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 which were
specified (implicitly or explicitly) in the request. If the input
parameters contain an address type that is not currently known to the
ALTO Server, the ALTO Server MUST behave as if the address type did
not appear in the input parameters.

The Version Tag included in the "vtag" 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: TBA
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: TBA
Content-Type: application/alto-networkmap+json

{
  "meta" : {
    "vtag" : {
       "resource-id": "my-default-network-map",
       "tag": "1266506139"
    }
  },
  "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 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
Cost Map MAY be provided by an ALTO Server.

11.3.2.1. **Media Type**

   Since a Filtered Cost Map is still a Cost Map, it uses the media type
   defined for Cost Map at Section 11.2.3.1.

11.3.2.2. **HTTP Method**

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

11.3.2.3. **Accept Input Parameters**

   The input parameters for a Filtered Map are supplied in the entity
   body of the POST request. This document specifies the input
   parameters with a data format indicated by the media type
   "application/alto-costmapfilter+json", which is a JSON Object of type
   ReqFilteredCostMap, where:

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

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

   with 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
      (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 (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;
      (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 1 to many cost types.

cost-constraints If true, then the ALTO Server allows cost
   constraints to be included in requests to the corresponding URI.
   If not present, this 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 Cost Map. See
Section 11.2.3.6 for the format.

The "dependent-vtags" key in the "meta" field is 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 consistent with the Version Tag of the
Network Map used to generate the request (if applicable). If it is
not, the ALTO Client may wish to request an updated Network Map,
identify changes, and consider requesting a new Filtered Cost Map.

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
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: TBA
Content-Type: application/alto-costmap+json

{
  "meta" : {
    "dependent-vtags" : [
      {"resource-id": "my-default-network-map",
       "tag": "1266506139"
      }
    ],
    "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. It MAY be provided by an ALTO Server.

11.4.1.1. Media Type

   The media type of Endpoint Property 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

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

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

   with 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
      (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

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

```
   object {
     EndpointPropertyType prop-types<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 look
   up of pid for every Network Map defined.

[11.4.1.5](#). Uses

   None.

[11.4.1.6](#). Response

   The "dependent-vtags" key in the "meta" field of the response MUST be
   an array that includes the Version Tags of all 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: TBA
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: TBA
Content-Type: application/alto-endpointprop+json

{
  "meta" : {
    "dependent-vtags" : [
      {"resource-id": "my-default-network-map",
       "tag": "1266506139"
      }
    ]
  },
  "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.

   It is important to note that although this resource allows an ALTO
   Server to reveal costs between individual endpoints, an ALTO Server
   is not required to do so. A simple alternative would be to compute
   the cost between two endpoints as the cost between the PIDs
   corresponding to the endpoints. See Section 15.3 for additional
   details.

11.5.1.1. Media Type

   The media type of Endpoint Cost 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 (
   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

In this document, we define EndpointCostCapabilities 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, an ALTO Server
is not required to do so. A simple implementation of an ECS resource
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. However, to preserve
flexibility, the ECS resource MAY omit declaring in the "uses"
attribute the network map and/or cost map on which it depends.

11.5.1.6. Response

The "meta" field of an Endpoint Cost response MUST include the "cost-
type" key, 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, a EndpointDstCosts dictionary map object denotes the
associated cost to each Destination Endpoint specified in input
parameters. An implementation of the protocol in this document
SHOULD assume that the cost value is a JSONNumber and fail to parse
if it is not, unless the implementation is using an extension to this
document that indicates when and how costs of other data types are
signaled. If the ALTO Server does not define a cost value from a
Source Endpoint to a particular Destination Endpoint, it MAY be
omitted from the response.

11.5.1.7. **Example**


```
POST /endpointcost/lookup HTTP/1.1
Host: alto.example.com
Content-Length: TBA
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: TBA
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 CDNs [I-D.jenkins-alto-cdn-use-cases].

12.1. ALTO Client Embedded in P2P Tracker

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

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

```
.---------.   (1) Get Network Map    .---------------.
|         | <----------------------> |               |
|  ALTO   |                          |  P2P Tracker  |
| Server  |   (2) Get Cost Map       | (ALTO Client) |
|         | <----------------------> |               |
`---------'                          `---------------'
                                         ^     |
                        (3) Get Peers |      | (4) Selected Peer
                                      |      v     List
           .---------.                .-----------.
           | Peer 1  | <------------- |    P2P    |
           `---------'                |  Client   |
              .        (5) Connect to  `-----------'
              .           Selected Peers    /
           .---------.                      /
           | Peer 50 | <------------------
           `---------'
```
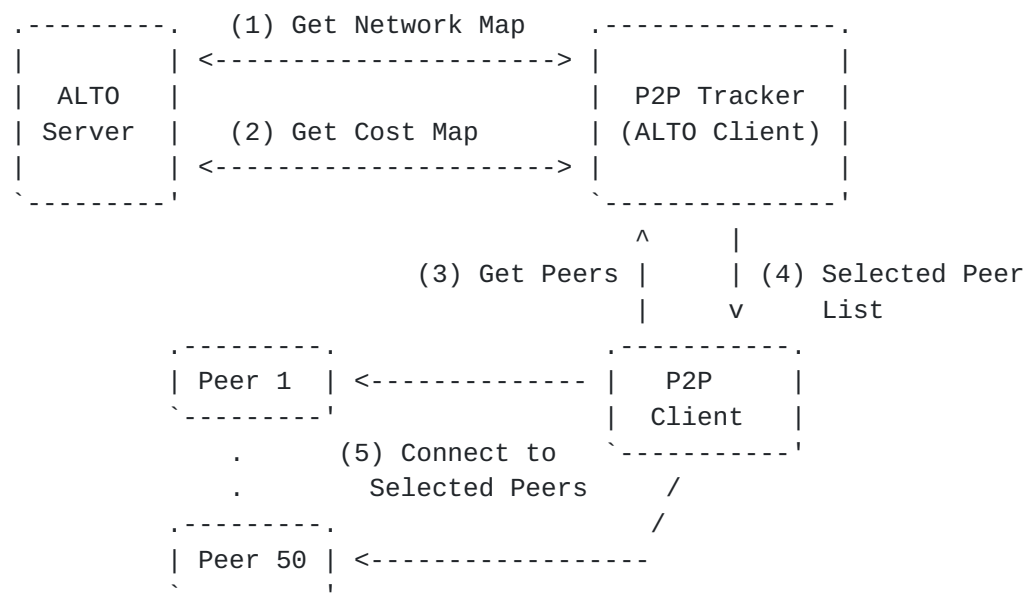
Figure 4: ALTO Client Embedded in P2P Tracker

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

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

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

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

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

5. The P2P Client connects to the selected peers.

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

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

```
.---------.   (1) Get Network Map     .---------------.
|         | <----------------------> |               |
|  ALTO   |                          |  P2P Client   |
| Server  |   (2) Get Cost Map       | (ALTO Client) |
|         | <----------------------> |               |   .---------.
`---------'                          `---------------' <- |  P2P    |
      .---------.                   / |     ^    ^   | Tracker |
      | Peer 1  | <--------------     |     |     \  `---------'
      `---------'                     |   (3) Gather Peers
          .        (4) Select Peers   |     |       \
          .           and Connect    /  .---------.  .---------.
      .---------.                   /   |  P2P    |  |  DHT    |
      | Peer 50 | <---------------      | Client  |  `--------'
      `---------'                       | (PEX)   |
                                        `--------'
```
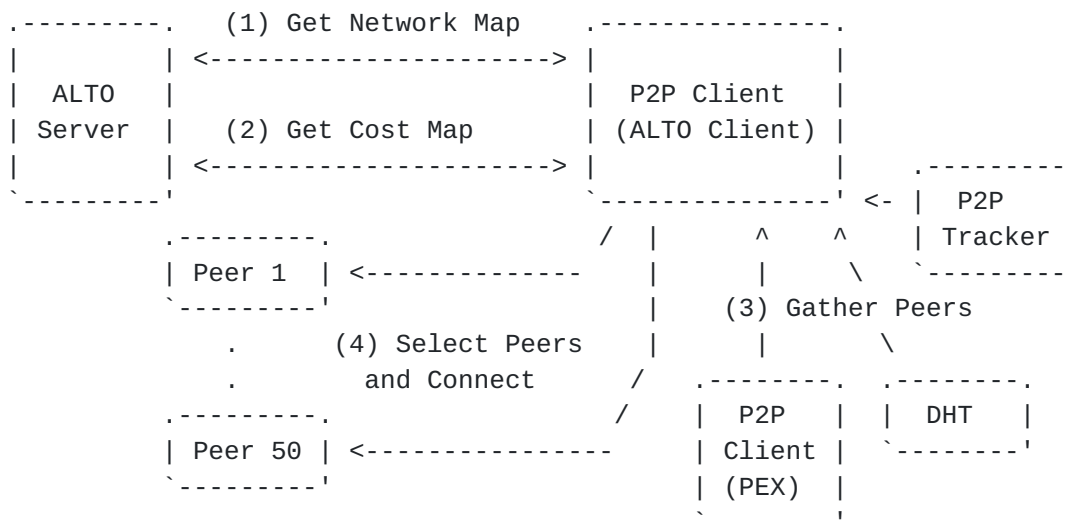
Figure 5: ALTO Client Embedded in P2P Client

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

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

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

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

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

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
gathers a list of known peers in the swarm, and asks the ALTO Server
to rank them.

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

```
    .---------.                      .--------------.
    |         |                      |              |
    |  ALTO   | (2) Get Endpoint Ranking | P2P Client  |
    | Server  | <---------------------> | (ALTO Client) |
    |         |                      |              |
    `---------'                      `--------------' <- |  P2P   |
          .---------.                  / |     ^    ^    | Tracker |
          | Peer 1  | <--------------    |     |     \   `---------'
          `---------'                    |   (1) Gather Peers
             .       (3) Connect to     |     |      \
             .          Selected Peers   /   .---------.  .---------.
          .---------.                  /    |  P2P   |  |  DHT    |
          | Peer 50 | <----------------     | Client |  `--------'
          `---------'                       | (PEX)  |
                                            `--------'
```
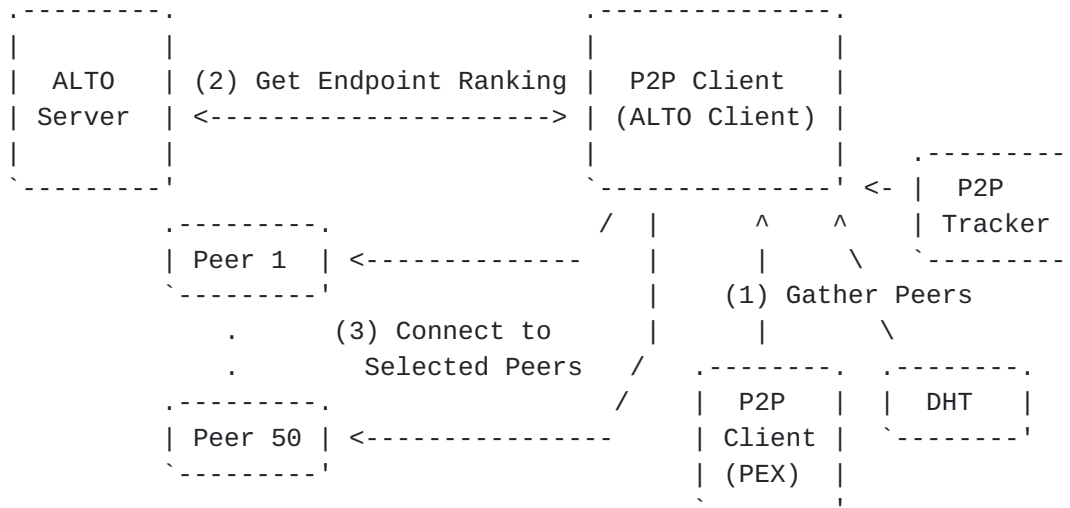
                Figure 6: ALTO Client Embedded in P2P Client: Ranking

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

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

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

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

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

At 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 provide the information
necessary to the server.

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

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

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

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

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 Metric, ALTO Endpoint Property Types, ALTO Address Types,
and ALTO Error Codes. Initial values for the registries are given
below; future assignments are to be made through the IANA Policy of
Standards Action, defined in [RFC5226].

14.1. application/alto-* Media Types

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

Type	Subtype	Specification
application	alto-directory+json	Section 9.2
application	alto-networkmap+json	Section 11.2.1
application	alto-networkmapfilter+json	Section 11.3.1
application	alto-costmap+json	Section 11.2.3
application	alto-costmapfilter+json	Section 11.3.2
application	alto-endpointprop+json	Section 11.4.1
application	alto-endpointpropparams+json	Section 11.4.1
application	alto-endpointcost+json	Section 11.5.1
application	alto-endpointcostparams+json	Section 11.5.1
application	alto-error+json	Section 8.5

Table 2: ALTO Protocol Media Types.

Type name: application

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

Required parameters: n/a

Optional parameters: n/a

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

Security considerations: Security considerations relating to the
   generation and consumption of ALTO Protocol messages are discussed
   in Section 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 standalone or embedded within other applications.

Additional information:

   Magic number(s): n/a

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

   Macintosh file type code(s): n/a

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

Intended usage: COMMON

Restrictions on usage: n/a

Author: See "Authors' Addresses" section.

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

14.2. ALTO Cost Metric Registry

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

```
+-------------+--------------------+
| Identifier  | Intended Semantics |
+-------------+--------------------+
| routingcost | See Section 6.1.1.1 |
| priv:       | Private use        |
| exp:        | Experimental use   |
+-------------+--------------------+
```

Table 3: ALTO Cost Metrics.

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

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

Registered ALTO Cost Metric identifiers MUST conform to the
syntactical requirements specified in Section 10.6. Identifiers are
to be recorded and displayed as ASCII strings.

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

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

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

o   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.

o 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

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

```
             +------------+--------------------+
             | Identifier | Intended Semantics |
             +------------+--------------------+
             | pid        | See Section 7.1.1  |
             | priv:      | Private use        |
             | exp:       | Experimental use   |
             +------------+--------------------+
```

Table 4: ALTO Endpoint Property Types.

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

14.4. ALTO Address Type Registry

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

Identifier	Address Encoding	Prefix Encoding	Mapping to/from IPv4/v6
ipv4	See Section 10.4.3	See Section 10.4.4	Direct mapping to IPv4

```
| ipv6       | See            | See            | Direct mapping   |
|            | Section 10.4.3 | Section 10.4.4 | to IPv6          |
+------------+----------------+----------------+------------------+
```

                   Table 5: ALTO Address Types.

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

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

   Registered ALTO Address Type identifiers MUST conform to the
   syntactical requirements specified in Section 10.4.2. Identifiers
   are to be recorded and displayed as ASCII strings.

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

   o Identifier: The name of the desired ALTO Address Type.

   o Endpoint Address Encoding: The procedure for encoding an address
      of the registered type as an EndpointAddr (see Section 10.4.3).

   o 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.

   o 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.

   o 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

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

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

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

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. In

particular, the ALTO Protocol requires that HTTP over TLS [RFC2818]
MUST be supported, when protecting the authenticity and integrity of
ALTO Information is required. The rules in [RFC2818] for a client to
verify server identity using server certificates MUST be supported.
ALTO Providers who request server certificates and certification
authorities who issue ALTO-specific certificates SHOULD consider the
recommendations and guidelines defined in [RFC6125]

Software engineers developing and service providers deploying ALTO
should make themselves familiar with 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 may be provided by extension
documents.

15.2. Potential Undesirable Guidance from Authenticated ALTO
       Information

15.2.1. Risk Scenarios

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

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

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 the Cost Map to
   select resource providers, the application may want to disable ALTO
   usage or switch to an external ALTO Server provided by an
   "independent organization" (see AR-20 and AR-21 in [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

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

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

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

   In [RFC6708] Section 5.2.1, three types of risks associated with the
   confidentiality of ALTO Information Resources are identified: risk
   type (1) Excess disclosure of the ALTO service provider's data to an
   authorized ALTO Client; risk type (2) Disclosure of the ALTO service
   provider's data (e.g., network topology information) to an
   unauthorized third party; and risk type (3) Excess retrieval of the

ALTO service provider's data by collaborating ALTO Clients.
[I-D.ietf-alto-deployments] 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,
and hence the Provider should evaluate how much information is
revealed and the associated risks.

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

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

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, Endpoint Cost,
and fine-grained Filtered Map services. The ALTO Server or a third-
party who is able to intercept such messages can store and process
obtained information in order to analyze user behaviors and
communication patterns. The analysis may correlate information

collected from multiple clients to deduce additional application/
content information. Such analysis can lead to privacy risks. For a
more comprehensive classification of related risk scenarios, see
cases 4, 5, and 6 in [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. An ALTO
Client may consider the possibility of relying only on Network Map
for PIDs and Cost Map amongst PIDs to avoid passing IP addresses of
other endpoints (e.g., peers) to the ALTO Server. When specific IP
addresses are needed (e.g., when using the Endpoint Cost Service), an
ALTO Client may consider obfuscation techniques such as specifying a
broader address range (i.e., a shorter prefix length) or by zeroing
out or randomizing the last few bits of IP addresses. Note that
obfuscation may yield less accurate results.

15.5. Availability of ALTO Service

15.5.1. Risk Scenarios

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

15.5.2. Protection Strategies

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

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

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 map is left to the specific
   policies of the ALTO Service Provider. However, a reasonable default
   may include two PIDs, one to hold the endpoints in the provider's
   network and the second PID to represent full IPv4 and IPv6
   reachability (see Section 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 operational impact should be considered.

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

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

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. Requirements on Other Protocols and Functional Components

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

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

16.1.4. Impact and Observation on Network Operation

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

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

An ALTO Service Provider should ensure that appropriate information
is being exposed. Privacy implications for ISPs are discussed in
Section 15.3. Both ALTO Service Providers and those using ALTO
Clients should be aware of the impact of incorrect or faked guidance
(see [I-D.ietf-alto-deployments]).

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

Monitoring ALTO Servers and Clients is described in
[I-D.ietf-alto-deployments].

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:

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

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

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

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

16.2.5. Performance Management

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

   o Requests and responses for each service listed in a Information
      Directory (total counts and size in bytes).

   o CPU and memory utilization

   o ALTO map updates

   o Number of PIDs

   o 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 [RFC2616] and SSL/TLS [RFC5246] and
   related documents for mechanisms available for configuring security
   policies. Other appropriate security mechanisms (e.g., physical
   security, firewalls, etc) should also be considered.

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.

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

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

RFC 3986, January 2005.

[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.

17.2. Informative References

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

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

[I-D.akonjang-alto-proxidor]
           Akonjang, O., Feldmann, A., Previdi, S., Davie, B., and D.
           Saucez, "The PROXIDOR Service",
           draft-akonjang-alto-proxidor-00 (work in progress),
           March 2009.

[I-D.ietf-alto-deployments]
           Stiemerling, M., Kiesel, S., Previdi, S., and M. Scharf,
           "ALTO Deployment Considerations",

draft-ietf-alto-deployments-08 (work in progress),
October 2013.

[I-D.ietf-alto-server-discovery]
          Kiesel, S., Stiemerling, M., Schwan, N., Scharf, M., and
          S. Yongchao, "ALTO Server Discovery",
          draft-ietf-alto-server-discovery-10 (work in progress),
          September 2013.

[I-D.ietf-httpbis-p2-semantics]
          Fielding, R. and J. Reschke, "Hypertext Transfer Protocol
          (HTTP/1.1): Semantics and Content",
          draft-ietf-httpbis-p2-semantics-25 (work in progress),
          November 2013.

[I-D.jenkins-alto-cdn-use-cases]
          Niven-Jenkins, B., Watson, G., Bitar, N., Medved, J., and
          S. Previdi, "Use Cases for ALTO within CDNs",
          draft-jenkins-alto-cdn-use-cases-03 (work in progress),
          June 2012.

[I-D.jennings-sip-hashcash]
          Jennings, C., "Computational Puzzles for SPAM Reduction in
          SIP", draft-jennings-sip-hashcash-06 (work in progress),
          July 2007.

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

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

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

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

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

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

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

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

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

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

    [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.

Appendix A. Acknowledgments

   Thank you to Sebastian Kiesel (University of Stuttgart) and Jan

Appendix B. Design History and Merged Proposals

   The ALTO Protocol specified in this document consists of
   contributions from

o P4P [I-D.p4p-framework], [P4P-SIGCOMM08],
   [I-D.wang-alto-p4p-specification];

o ALTO Info-Export [I-D.shalunov-alto-infoexport];

o Query/Response [I-D.saumitra-alto-queryresponse],
   [I-D.saumitra-alto-multi-ps]; and

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

Appendix C. Specifying the "value" Key for E_INVALID_FIELD_VALUE

   In Section 8.5.2, an ALTO Server may specify an optional "value" key
   to indicate the invalid value that triggerred an
   E_INVALID_FIELD_VALUE error. The "value" key must specify a
   JSONString, but the invalid value may not be a simple JSONString.
   Below are the rule to specify the "value" key:

   o If the invalid value is a string, "value" is that string;

   o If the invalid value is a number, "value" must be the invalid
      number as a string;

   o 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";

   o 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.

Appendix D. Authors

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

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

      Email: ietf-alto@skiesel.de

      Stefano Previdi
      Cisco

      Email: sprevidi@cisco.com

      Wendy Roome
      Alcatel Lucent

      Email: w.roome@alcatel-lucent.com

      Stanislav Shalunov
      BitTorrent

      Email: shalunov@bittorrent.com

      Richard Woundy
      Comcast

      Richard_Woundy@cable.comcast.com

Authors' Addresses

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

      Email: ralimi@google.com

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

Email: repenno@cisco.com

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

Email: yry@cs.yale.edu