

ALTO WG
Internet-Draft
Intended status: Standards Track
Expires: January 12, 2013

R. Alimi, Ed.
Google
R. Penno, Ed.
Cisco Systems
Y. Yang, Ed.
Yale University
July 11, 2012

ALTO Protocol
draft-ietf-alto-protocol-12.txt

Abstract

Networking applications today already have access to a great amount of Inter-Provider network topology information. For example, views of the Internet routing table are easily available at looking glass servers and entirely practical to be downloaded by clients. What is missing is knowledge of the underlying network topology from the ISP or Content Provider (henceforth referred as Provider) point of view. In other words, what a Provider prefers in terms of traffic optimization -- and a way to distribute it.

The ALTO Service provides network information (e.g., basic network location structure, 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 the maps.

This document describes a protocol implementing the ALTO Service. Although the ALTO service would primarily be provided by the network operator (e.g., an ISP), content providers and third parties could also operate this service. Applications that could use this service are those that have a choice in connection endpoints. 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 January 12, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	6
1.1. Background and Problem Statement	6
1.2. Design History and Merged Proposals	6
1.3. Solution Benefits	7
1.3.1. Service Providers	7
1.3.2. Applications	7
2. Architecture	7
2.1. Terminology	8
2.1.1. Endpoint	8
2.1.2. Endpoint Address	8
2.1.3. ASN	8
2.1.4. Network Location	8
2.1.5. ALTO Information	8
2.1.6. ALTO Information Base	9
2.2. ALTO Service and Protocol Scope	9
2.3. ALTO Information Reuse and Redistribution	10
3. Service Framework	11
3.1. ALTO Information Services	12
3.1.1. Map Service	12
3.1.2. Map Filtering Service	12
3.1.3. Endpoint Property Service	12
3.1.4. Endpoint Cost Service	12
4. Network Map	12
4.1. PID	13
4.2. Endpoint Addresses	14
4.2.1. IP Addresses	14
4.3. Example Network Map	14
5. Cost Map	15
5.1. Cost Attributes	16
5.1.1. Cost Type	16
5.1.2. Cost Mode	16
5.2. Cost Map Structure	17
5.3. Network Map and Cost Map Dependency	18
6. Protocol Specification	18
6.1. Overall Design	18
6.2. Notation	19
6.3. Basic Operation	19
6.3.1. Discovering Information Resources	19
6.3.2. Requesting Information Resources	19
6.3.3. Response	20
6.3.4. Client Behavior	21
6.3.5. Authentication and Encryption	21
6.3.6. HTTP Cookies	21
6.3.7. Parsing	21
6.4. Information Resource	21
6.4.1. Capabilities	22

6.4.2.	Input Parameters Media Type	22
6.4.3.	Media Type	22
6.4.4.	Encoding	22
6.5.	ALTO Errors	23
6.5.1.	Media Type	24
6.5.2.	Resource Format	24
6.5.3.	Error Codes	24
6.5.4.	Overload Conditions and Server Unavailability	25
6.6.	ALTO Types	25
6.6.1.	PID Name	26
6.6.2.	Version Tag	26
6.6.3.	Endpoints	26
6.6.4.	Cost Mode	28
6.6.5.	Cost Type	29
6.6.6.	Endpoint Property	29
6.7.	Information Resource Directory	29
6.7.1.	Media Type	30
6.7.2.	Encoding	30
6.7.3.	Example	31
6.7.4.	Usage Considerations	34
6.8.	Information Resources	35
6.8.1.	Map Service	35
6.8.2.	Map Filtering Service	40
6.8.3.	Endpoint Property Service	46
6.8.4.	Endpoint Cost Service	49
7.	Use Cases	53
7.1.	ALTO Client Embedded in P2P Tracker	54
7.2.	ALTO Client Embedded in P2P Client: Numerical Costs	55
7.3.	ALTO Client Embedded in P2P Client: Ranking	56
8.	Discussions	57
8.1.	Discovery	57
8.2.	Hosts with Multiple Endpoint Addresses	58
8.3.	Network Address Translation Considerations	58
8.4.	Endpoint and Path Properties	59
9.	IANA Considerations	59
9.1.	application/alto-* Media Types	59
9.2.	ALTO Cost Type Registry	61
9.3.	ALTO Endpoint Property Registry	62
9.4.	ALTO Address Type Registry	63
10.	Security Considerations	64
10.1.	Privacy Considerations for ISPs	64
10.2.	ALTO Clients	65
10.3.	Authentication, Integrity Protection, and Encryption	65
10.4.	ALTO Information Redistribution	66
10.5.	Denial of Service	66
10.6.	ALTO Server Access Control	67
11.	References	67
11.1.	Normative References	67

11.2. Informative References	68
Appendix A. Acknowledgments	70
Appendix B. Authors	71
Authors' Addresses	71

1. Introduction

1.1. Background and Problem Statement

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

The ALTO Service provides a simple mechanism to convey network information to applications. Its objective is to provide basic, abstract but useful network information to applications. The mechanism includes abstractions to achieve concise, flexible network information expression.

The goal of this document is to specify a simple and unified protocol that meets the ALTO requirements [I-D.ietf-alto-reqs] while providing a migration path for Internet Service Providers (ISP), Content Providers, and clients that have deployed protocols with similar intentions (see Section 1.2).

The ALTO Protocol design uses a REST-ful design with the goal of leveraging current HTTP [RFC2616] implementations and infrastructure. The REST-ful design supports flexible deployment strategies and provides extensibility. ALTO requests and responses are encoded with JSON [RFC4627].

1.2. Design History and Merged Proposals

The protocol specified here 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];
- o ATTP [ATTP];

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

See Appendix A for a list of people that have contributed significantly to this effort and the projects and proposals listed above.

1.3. Solution Benefits

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

The ALTO Service offers many benefits to both end-users (consumers of the service) and Internet Service Providers (providers of the service).

1.3.1. Service Providers

The ALTO Service enables Service Providers to influence the peer or resource selection process in distributed applications in order to increase locality of traffic, improve user-experience, amongst others. It also helps ISPs to efficiently manage traffic that traverses more expensive links such as transit and backup links, thus allowing a better provisioning of the networking infrastructure.

1.3.2. Applications

Applications that use the ALTO Service can benefit in multiple ways. For example, they may no longer need to infer topology information, and some applications can reduce reliance on measuring path performance metrics themselves. They can take advantage of the ISP's knowledge to avoid bottlenecks and boost performance.

An example type of application is a Peer-to-Peer overlay where peer selection can be improved by including ALTO information in the selection process.

2. Architecture

Two key design objectives of the ALTO Protocol are simplicity and extensibility. At the same time, it introduces additional techniques to address potential scalability and privacy issues. This section first introduces the terminology, and then defines the ALTO architecture and the ALTO Protocol's place in the overall architecture.

2.1. Terminology

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

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

2.1.1. Endpoint

An endpoint is an entity capable of communicating (sending and/or receiving messages) on a network.

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

2.1.2. Endpoint Address

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

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

2.1.3. ASN

An Autonomous System Number.

2.1.4. Network Location

Network Location is a generic term denoting a single endpoint or group of endpoints.

2.1.5. ALTO Information

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

2.1.6. ALTO Information Base

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

2.2. ALTO Service and Protocol Scope

An ALTO Server conveys the network information from the perspective of a network region; the ALTO Server presents its "my-Internet View" of the network region. In particular, an ALTO Server defines network Endpoints (and aggregations thereof) and generic costs amongst them from the network region's own perspective. A network region in this context can be an Autonomous System, an ISP, or perhaps a smaller region or set of ISPs; the details depend on the ALTO deployment scenario and ALTO service discovery mechanism.

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.

More specifically, the ALTO Information provided by an ALTO Server may be influenced (at the operator's discretion) by other systems. The ALTO Server aggregates information from multiple systems to provide an abstract, unified, useful network view 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 ALTO may convey dynamic network information, it is not intended to replace near-real-time congestion protocols.

It may also be possible for ALTO Servers 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.

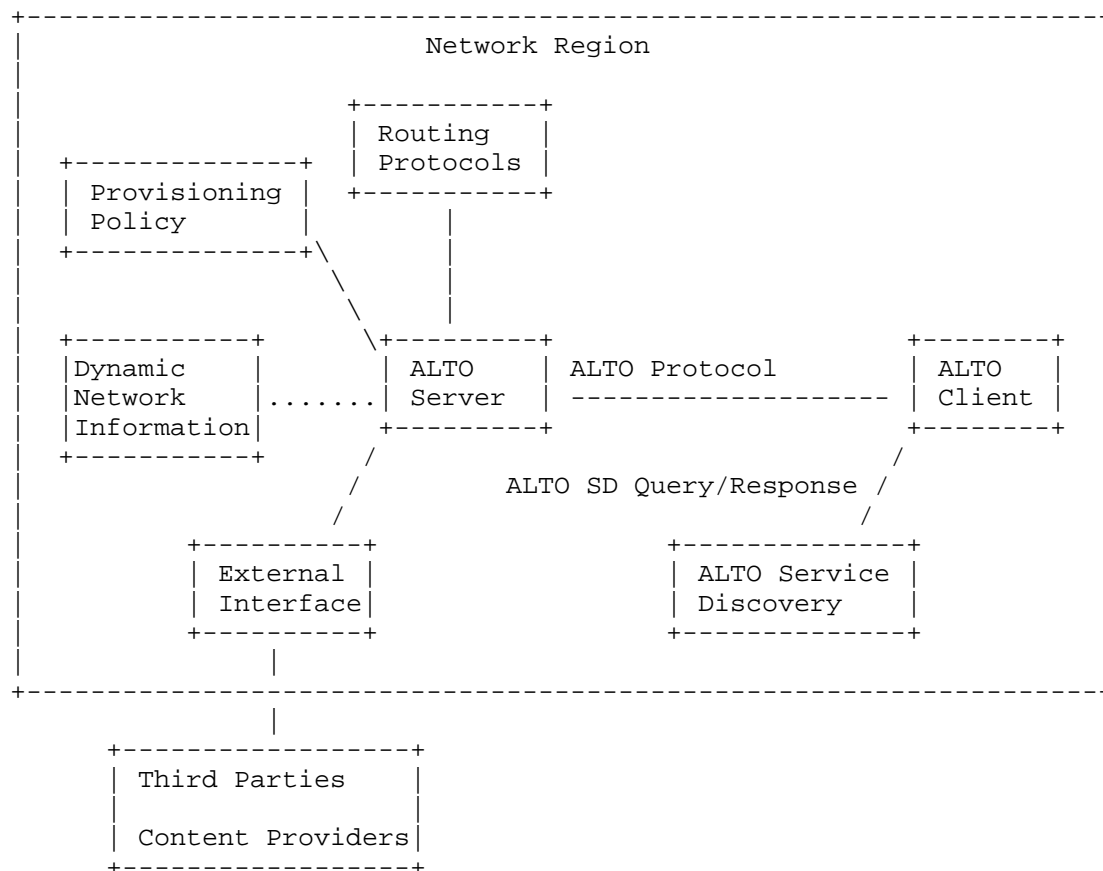


Figure 1: Basic ALTO Architecture.

2.3. 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.

Beyond integration with existing HTTP caching infrastructure, ALTO information may also be cached or redistributed using application-dependent mechanisms, such as P2P DHTs or P2P file-sharing. This document does not define particular mechanisms for such redistribution. See [I-D.gu-alto-redistribution] for further discussion.

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

documents.

If caching or redistribution is used, the response message may be returned from another (possibly third-party) entity.

3. Service Framework

The ALTO Protocol uses a simple extensible framework to convey network information. In the general framework, the ALTO protocol will convey properties on both Network Locations and the paths between Network Locations.

In this document, we focus on a particular Endpoint property to denote the location of an endpoint, and provider-defined costs for paths between pairs of Network Locations.

The ALTO Protocol is built on a common transport protocol, messaging structure and encoding, and transaction model. The protocol is subdivided into services of related functionality. The Map Service provides the core ALTO information to clients. Other ALTO Information services provide additional functionalities. There are three such services defined in this document: the Map Filtering Service, Endpoint Property Service, and Endpoint Cost Service. Additional services may be defined in companion documents. Functionalities offered in different services may overlap (e.g., the Map Service and Map Filtering Service).

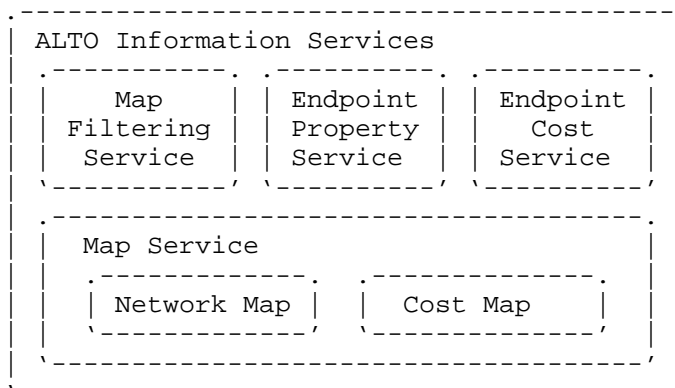


Figure 2: ALTO Service Framework

3.1. ALTO Information Services

Multiple, distinct services are defined to allow ALTO Clients to query ALTO Information from an ALTO Server. The ALTO Server internally maintains an ALTO Information Base that encodes the network provider's preferences. The ALTO Information Base encodes the Network Locations defined by the ALTO Server (and their corresponding properties), as well as the provider-defined costs between pairs of Network Locations.

3.1.1. Map Service

The Map Service provides batch information to ALTO Clients in the form of Network Map and Cost Map. The Network Map (See Section 4) provides the full set of Network Location groupings defined by the ALTO Server and the Endpoints contained with each grouping. The Cost Map (see Section 5) provides costs between the defined groupings.

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

3.1.2. Map Filtering Service

Resource constrained ALTO Clients may benefit from query results being filtered at the ALTO Server. This avoids an ALTO Client spending network bandwidth or CPU collecting results and performing client-side filtering. The Map Filtering Service allows ALTO Clients to query for the ALTO Server Network Map and Cost Map based on additional parameters.

3.1.3. Endpoint Property Service

This service allows ALTO Clients to look up properties for individual Endpoints. An example endpoint property is its Network Location (its grouping defined by the ALTO Server) or connectivity type (e.g., ADSL, Cable, or FTTH).

3.1.4. 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.

4. Network Map

The Network Location Endpoint Property allows an ALTO Server to group

endpoints together to indicate their proximity. The resulting set of groupings is called the ALTO Network Map.

In reality, many endpoints are very close to one another in terms of network connectivity, for example, endpoints on the same site of an enterprise. By treating a group of endpoints together as a single entity in ALTO, we can achieve much greater scalability without losing critical information.

The definition of 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 PoP may be considered close.

As used in this document, the Network 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.

4.1. PID

Each group of Endpoints is identified by a provider-defined Network Location identifier called a PID. A PID is a US-ASCII string of type PIDName (see Section 6.6.1) and its associated set of Endpoint Addresses. There can be many different ways of grouping the endpoints and assigning PIDs.

A PID is an identifier that provides 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. For example, a PID may be defined by the ALTO service provider to denote a subnet, a set of subnets, a metropolitan area, a PoP, an autonomous system, or a set of autonomous systems. Aggregation of endpoints into PIDs can indicate proximity and can improve scalability. In particular, network preferences (costs) may be specified between PIDs, allowing cost information to be more compactly represented and updated at a faster time scale than the network aggregations themselves.

Using PIDs, the Network Map may also be used to communicate simple preferences with only minimal information from the Cost Map. 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 Map may be encoded to indicate that Network Locations within the same PID are preferred; for example, `cost(PID_i, PID_i) == c*` and

$\text{cost}(\text{PID}_i, \text{PID}_j) > c^*$ for $i \neq j$. Section 5 provides further details about Cost Map structure.

4.2. Endpoint Addresses

Communicating endpoints may have many types of addresses, such as IP addresses, MAC addresses, or overlay IDs. The current specification only considers IP addresses.

4.2.1. IP Addresses

The endpoints aggregated into a PID are denoted by a list of IP prefixes. When either an ALTO Client or ALTO Server needs to determine which PID in a Network Map contains a particular IP address, longest-prefix matching **MUST** be used.

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

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

4.3. Example Network Map

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

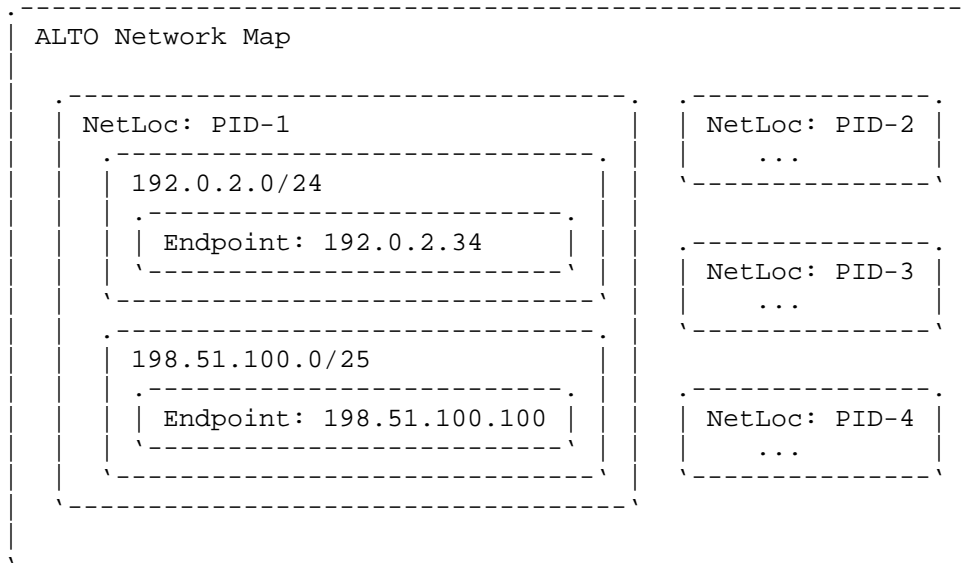


Figure 3: Example Network Map

5. Cost Map

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

An ALTO Cost Map defines Path Costs pairwise amongst sets of source and destination Network Locations. Each Path Cost is the end-to-end cost from the source to the destination.

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

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

As used in this document, the Cost Map refers to the syntax and

semantics of the information distributed by the ALTO Server. This document does not discuss the internal representation of this data structure within the ALTO Server.

5.1. Cost Attributes

Path Costs have attributes:

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

Certain queries for Cost Maps allow the ALTO Client to indicate the desired Type and Mode.

5.1.1. Cost Type

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

Cost types are indicated in protocol messages as strings.

5.1.1.1. Cost Type: routinycost

An ALTO Server MUST define the 'routinycost' Cost Type.

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

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

5.1.2. 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' costs. ALTO Clients **SHOULD** be cognizant of operations when a desired cost mode is not supported. For example, an ALTO Client desiring numerical costs may adjust behavior if only the ordinal Cost Mode is available. Alternatively, an ALTO Client desiring ordinal costs may construct ordinal costs given numerical values if only the numerical Cost Mode is available.

5.1.2.1. 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.

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

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

5.2. Cost Map Structure

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

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

The ALTO Server will return the Path Cost for each communicating pair (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.

5.3. Network Map and Cost Map Dependency

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

A Version Tag is an opaque string associated with a Network Map maintained by the ALTO Server. When the Network Map changes, the Version Tag MUST also be changed. (Thus, the Version Tag is defined similarly to HTTP's Entity Tags; see Section 3.11 of [RFC2616].) Possibilities for generating a Version Tag include the last-modified timestamp for the Network Map, or a hash of its contents.

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

6. Protocol Specification

This section first specifies general client and server processing, followed by a detailed specification for each ALTO Information Resource.

6.1. Overall Design

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

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

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

6.2. Notation

This document uses an adaptation of the C-style struct notation to define the required and optional members of JSON objects. Unless explicitly noted, each member of a struct is REQUIRED.

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

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

6.3. Basic Operation

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

6.3.1. Discovering Information Resources

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

Informally, an Information Resource Directory enumerates URIs at which an ALTO Server offers Information Resources. Each entry in the directory indicates a URI at which an ALTO Server accepts requests, and returns either the requested Information Resource or an Information Resource Directory that references additional Information Resources. See Section 6.7 for a detailed specification.

6.3.2. Requesting Information Resources

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

available.

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

It is possible for an ALTO Server to leverage caching HTTP intermediaries for responses 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.

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.

6.3.3. Response

Upon receiving a request, an ALTO server either returns the requested resource, provides the ALTO Client an Information Resource Directory indicating how to reach the desired resource, or returns an error.

The type of response MUST be indicated by the media type attached to the response (the Content-Type HTTP header). If an ALTO Client receives an Information Resource Directory, it can consult the received directory to determine if any of the offered URIs contain the desired Information Resource.

The generic encoding for an Information Resource is specified in Section 6.4.

Errors are indicated via either ALTO-level error codes, or via HTTP status codes; see Section 6.5.

6.3.4. Client Behavior

6.3.4.1. 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.

6.3.4.2. Error Conditions

If an ALTO Client does not successfully receive a desired Information Resource from a particular ALTO Server, it 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). An ALTO Client may also retry the request at a later time.

6.3.5. Authentication and Encryption

An ALTO Server MAY support SSL/TLS to implement server and/or client authentication, encryption, and/or integrity protection. See [RFC6125] for considerations regarding verification of server identity.

6.3.6. HTTP Cookies

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

6.3.7. Parsing

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

6.4. Information Resource

An Information Resource is an HTTP entity body received by an ALTO Server that encodes the ALTO Information desired by an ALTO Client.

This document specifies multiple Information Resources that can be provided by an ALTO Server. Each Information Resource has certain attributes associated with it, indicating its data format, the input parameters it supports, and format of the input parameters.

6.4.1. Capabilities

An ALTO Server may advertise to an ALTO Client that it supports certain capabilities in requests for an Information Resource. For example, if an ALTO Server allows requests for a Cost Map to include constraints, it may advertise that it supports this capability.

6.4.2. Input Parameters Media Type

An ALTO Server may allow an ALTO Client to supply input parameters when requesting certain Information Resources. The format of the input parameters (i.e., as contained in the entity body of the HTTP POST request) is indicated by the media type [RFC2046].

6.4.3. Media Type

The media type [RFC2046] uniquely indicates the data format of the Information Resource as returned by an ALTO Server in the HTTP entity body.

6.4.4. Encoding

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

An Information Resource has a single top-level JSON object of type `InfoResourceEntity`:

```
object {  
  InfoResourceMetaData  meta;    [OPTIONAL]  
  [InfoResourceDataType] data;  
} InfoResourceEntity;
```

with members:

meta meta-information pertaining to the Information Resource

data the data contained in the Information Resource

6.4.4.1. Meta Information

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

```
object {  
  } InfoResourceMetaData;
```

6.4.4.2. ALTO Information

The "data" member of the InfoResourceEntity encodes the resource-specific data; the structure of this member is detailed later in this section for each particular Information Resource.

6.4.4.3. Example

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

```
HTTP/1.1 200 OK  
Content-Length: 40  
Content-Type: application/alto-costmap+json  
  
{  
  "meta" : {},  
  "data" : {  
    ...  
  }  
}
```

6.5. ALTO Errors

If there is an error processing a request, an ALTO Server SHOULD return additional ALTO-layer information, if it is available, in the form of an ALTO Error Resource encoded in the HTTP response's entity body.

If no ALTO-layer information is available, an ALTO Server may omit an ALTO Error resource from the response. An appropriate HTTP status code MUST be set.

It is important to note that the HTTP Status Code and ALTO Error Code have distinct roles. An ALTO Error Code 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.

6.5.1. Media Type

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

6.5.2. Resource Format

An ALTO Error Resource has the format:

```
object {  
  JSONString code;  
} ErrorResponseEntity;
```

where:

code An ALTO Error Code defined in Table 1

6.5.3. Error Codes

This document defines ALTO Error Codes to support the error conditions needed for purposes of this document. Additional status codes may be defined in companion or extension documents.

The HTTP status codes corresponding to each ALTO Error Code are defined to provide correct behavior with HTTP intermediaries and clients. When an ALTO Server returns a particular ALTO Error Code, it MUST indicate one of the corresponding HTTP status codes in Table 1 in the HTTP response.

If multiple errors are present in a single request (e.g., a request uses a JSONString when a JSONInteger is expected and a required field is missing), then the ALTO Server MUST return exactly one of the detected errors. However, the reported error is implementation defined, since specifying a particular order for message processing encroaches needlessly on implementation technique.

ALTO Error Code	HTTP Status Code(s)	Description
E_SYNTAX	400	Parsing error in request (including identifiers)
E_JSON_FIELD_MISSING	400	Required field missing
E_JSON_VALUE_TYPE	400	JSON Value of unexpected type
E_INVALID_COST_MODE	400	Invalid cost mode
E_INVALID_COST_TYPE	400	Invalid cost type
E_INVALID_PROPERTY_TYPE	400	Invalid property type

Table 1: Defined ALTO Error Codes

6.5.4. 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.

6.6. ALTO Types

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

6.6.1. PID Name

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

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

6.6.2. Version Tag

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

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

6.6.3. Endpoints

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

6.6.3.1. Address Type

Address Types are encoded as US-ASCII strings consisting of only alphanumeric characters (code points 0x30-0x39, 0x41-0x5A, and 0x61-0x7A). This document defines the address type 'ipv4' to refer to IPv4 addresses, and 'ipv6' to refer to IPv6 addresses. All Address Type identifiers appearing in an HTTP request or response with an 'application/alto-*' media type MUST be registered in the ALTO Address Type registry Section 9.4.

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

6.6.3.2. Endpoint Address

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

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

6.6.3.2.1. IPv4

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

6.6.3.2.2. IPv6

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

6.6.3.2.3. Typed Endpoint Addresses

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

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

6.6.3.3. Endpoint Prefixes

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

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

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

6.6.3.3.1. IPv4

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

6.6.3.3.2. IPv6

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

6.6.3.4. Endpoint Address Group

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

An Endpoint Address Group is defined as:

```
object {  
  EndpointPrefix [AddressType]<0..*>;  
  ...  
} EndpointAddrGroup;
```

In particular, an Endpoint Address Group is a JSON object with the name of each member being the string corresponding to the address type, and the member's corresponding value being a list of 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"  
  ]  
}
```

6.6.4. Cost Mode

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

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

6.6.5. Cost Type

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

Identifiers prefixed with 'priv:' are reserved for Private Use [RFC5226]. Identifiers prefixed with 'exp:' are reserved for Experimental use. All other identifiers appearing in an HTTP request or response with an 'application/alto-*' media type MUST be registered in the ALTO Cost Types registry Section 9.2.

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

6.6.6. Endpoint Property

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

Identifiers prefixed with 'priv:' are reserved for Private Use [RFC5226]. Identifiers prefixed with 'exp:' are reserved for Experimental use. All other identifiers appearing in an HTTP request or response with an 'application/alto-*' media type MUST be registered in the ALTO Endpoint Property registry Section 9.3.

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

6.7. Information Resource Directory

An Information Resource Directory indicates to ALTO Clients which Information Resources are made available by an ALTO Server.

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].

6.7.1. Media Type

The media type is "application/alto-directory+json".

6.7.2. Encoding

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

```
object {  
  ...  
} Capabilities;  
  
object {  
  JSONString    uri;  
  JSONString    media-types<1..*>;  
  JSONString    accepts<0..*>;           [OPTIONAL]  
  Capabilities  capabilities;             [OPTIONAL]  
} ResourceEntry;  
  
object {  
  ResourceEntry resources<0..*>;  
} InfoResourceDirectory;
```

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

Any URI endpoint indicated in an Information Resource Directory MAY provide a response to an OPTIONS request that is in the format of an Information Resource Directory response. This provides ALTO Clients a means to discover resources and capabilities offered by that URI endpoint. ALTO Servers that reply with an HTTP 300 status code ("Multiple Choices") SHOULD use the Information Resource Directory format in the reply.

Each entry in the directory specifies:

uri A URI at which the ALTO Server provides one or more Information Resources, or an Information Resource Directory indicating additional Information Resources.

media-types The list of all media types of Information Resources (see Section 6.4.3) available via GET or POST requests to the corresponding URI or URIs discoverable via the URI.

accepts The list of all media types of input parameters (see Section 6.4.2) accepted by POST requests to the corresponding URI or URIs discoverable via the URI. If this member is not present, it MUST be assumed to be an empty array.

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

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

6.7.3. Example

The following is an example Information Resource Directory returned by an ALTO Server. In this example, the ALTO Server provides additional Network and Cost Maps via a separate subdomain, "custom.alto.example.com". The maps available via this subdomain are Filtered Network and Cost Maps as well as pre-generated maps for the "hopcount" and "routingcost" Cost Types in the "ordinal" Cost Mode.

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

In this example, the ALTO server provides the Endpoint Cost Service for Cost Types 'routingcost' and 'hopcount', each available for both 'numerical' and 'ordinal' mode".

```
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: 1472
Content-Type: application/alto-directory+json
```

```
{
  "resources" : [
    {
      "uri" : "http://alto.example.com/networkmap",
      "media-types" : [ "application/alto-networkmap+json" ]
    }, {
      "uri" : "http://alto.example.com/costmap/num/routingcost",
      "media-types" : [ "application/alto-costmap+json" ],
      "capabilities" : {
        "cost-modes" : [ "numerical" ],
        "cost-types" : [ "routingcost" ]
      }
    }, {
      "uri" : "http://alto.example.com/costmap/num/hopcount",
      "media-types" : [ "application/alto-costmap+json" ],
      "capabilities" : {
        "cost-modes" : [ "numerical" ],
        "cost-types" : [ "hopcount" ]
      }
    }, {
      "uri" : "http://custom.alto.example.com/maps",
      "media-types" : [
        "application/alto-networkmap+json",
        "application/alto-costmap+json"
      ],
      "accepts" : [
        "application/alto-networkmapfilter+json",
        "application/alto-costmapfilter+json"
      ]
    }, {
      "uri" : "http://alto.example.com/endpointprop/lookup",
      "media-types" : [ "application/alto-endpointprop+json" ],
      "accepts" : [ "application/alto-endpointpropparams+json" ],
      "capabilities" : {
        "prop-types" : [ "pid" ]
      }
    }, {

```



```
    "uri" : "http://alto.example.com/endpointcost/lookup",
    "media-types" : [ "application/alto-endpointcost+json" ],
    "accepts" : [ "application/alto-endpointcostparams+json" ],
    "capabilities" : {
      "cost-constraints" : true,
      "cost-modes" : [ "ordinal", "numerical" ],
      "cost-types" : [ "routingcost", "hopcount" ]
    }
  ]
}
```

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

HTTP/1.1 200 OK
Content-Length: 1001
Content-Type: application/alto-directory+json

```
{
  "resources" : [
    {
      "uri" : "http://custom.alto.example.com/networkmap/filtered",
      "media-types" : [ "application/alto-networkmap+json" ],
      "accepts" : [ "application/alto-networkmapfilter+json" ]
    }, {
      "uri" : "http://custom.alto.example.com/costmap/filtered",
      "media-types" : [ "application/alto-costmap+json" ],
      "accepts" : [ "application/alto-costmapfilter+json" ],
      "capabilities" : {
        "cost-constraints" : true,
        "cost-modes" : [ "ordinal", "numerical" ],
        "cost-types" : [ "routingcost", "hopcount" ]
      }
    }, {
      "uri" : "http://custom.alto.example.com/ord/routingcost",
      "media-types" : [ "application/alto-costmap+json" ],
      "capabilities" : {
        "cost-modes" : [ "ordinal" ],
        "cost-types" : [ "routingcost" ]
      }
    }, {
      "uri" : "http://custom.alto.example.com/ord/hopcount",
      "media-types" : [ "application/alto-costmap+json" ],
      "capabilities" : {
        "cost-modes" : [ "ordinal" ],
        "cost-types" : [ "hopcount" ]
      }
    }
  ]
}
```

6.7.4. Usage Considerations

6.7.4.1. 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 6.7.3, a full Cost Map with "numerical" Cost Mode and "routingcost" Cost Type could be retrieved via a GET request to "http://alto.example.com/costmap/num/routingcost", or via a POST request to "http://custom.alto.example.com/costmap/filtered".

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

6.7.4.2. 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".

6.8. Information Resources

This section documents the individual Information Resources defined in the ALTO Protocol.

6.8.1. Map Service

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

6.8.1.1. Network Map

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

6.8.1.1.1. Media Type

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

6.8.1.1.2. HTTP Method

This resource is requested using the HTTP GET method.

6.8.1.1.3. Input Parameters

None.

6.8.1.1.4. Capabilities

None.

6.8.1.1.5. Response

The returned InfoResourceEntity object "data" member of type InfoResourceNetworkMap:

```
object {  
  EndpointAddrGroup [pidname]<0..*>;  
  ...  
} NetworkMapData;  
  
object {  
  VersionTag      map-vtag;  
  NetworkMapData map;  
} InfoResourceNetworkMap;
```

with members:

map-vtag The Version Tag (Section 5.3) of the Network Map.

map The Network Map data itself.

NetworkMapData is a JSON object with each member representing a single PID and its associated set of endpoint addresses. A member's name is a string of type PIDName.

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

6.8.1.1.6. Example

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

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

```
{
  "meta" : {},
  "data" : {
    "map-vtag" : "1266506139",
    "map" : {
      "PID1" : {
        "ipv4" : [
          "192.0.2.0/24",
          "198.51.100.0/25"
        ]
      },
      "PID2" : {
        "ipv4" : [
          "198.51.100.128/25"
        ]
      },
      "PID3" : {
        "ipv4" : [
          "0.0.0.0/0"
        ],
        "ipv6" : [
          "::/0"
        ]
      }
    }
  }
}
```

6.8.1.2. Cost Map

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

Note that since this resource, an unfiltered Cost Map requested by an HTTP GET, does not indicate the desired Cost Mode or Cost Type as input parameters, an ALTO Server MUST indicate in an Information Resource Directory a unfiltered Cost Map Information Resource by specifying the capabilities (Section 6.8.1.2.4) with "cost-types" and "cost-modes" members each having a single element. This technique will allow an ALTO Client to determine a URI for an unfiltered Cost Map of the desired Cost Mode and Cost Type.

6.8.1.2.1. Media Type

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

6.8.1.2.2. HTTP Method

This resource is requested using the HTTP GET method.

6.8.1.2.3. Input Parameters

None.

6.8.1.2.4. Capabilities

This resource may be defined for across multiple Cost Types and Cost Modes. The capabilities of an ALTO Server URI providing this resource are defined by a JSON Object of type CostMapCapability:

```
object {  
  CostMode cost-modes<0..*>;  
  CostType cost-types<0..*>;  
} CostMapCapability;
```

with members:

cost-modes The Cost Modes (Section 5.1.2) supported by the corresponding URI. If not present, this member MUST be interpreted as an empty array.

cost-types The Cost Types (Section 5.1.1) supported by the corresponding URI. If not present, this member MUST be interpreted as an empty array.

An ALTO Server MUST support all of the Cost Types listed here for each of the listed Cost Modes. Note that an ALTO Server may provide multiple Cost Map Information Resources, each with different capabilities.

6.8.1.2.5. Response

The returned InfoResourceEntity object has "data" member of type InfoResourceCostMap:

```
object DstCosts {
  JSONValue [PIDName];
  ...
};

object {
  DstCosts [PIDName]<0..*>;
  ...
} CostMapData;

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

with members:

cost-mode Cost Mode (Section 5.1.2) used in the Cost Map.

cost-type Cost Type (Section 5.1.1) used in the Cost Map.

map-vtag The Version Tag (Section 5.3) of the Network Map used to generate the Cost Map.

map The Cost Map data itself.

CostMapData is a JSON object with each member representing a single Source PID; the name for a member is the PIDName string identifying the corresponding Source PID. For each Source PID, a DstCosts object denotes the associated cost to a set of destination PIDs (Section 5.2); the name for each member in the object is the PIDName string identifying the corresponding Destination PID. 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).

6.8.1.2.6. Example

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

```
HTTP/1.1 200 OK
Content-Length: 262
Content-Type: application/alto-costmap+json
```

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

6.8.2. Map Filtering Service

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

6.8.2.1. Filtered Network Map

A Filtered Network Map is a Network Map Information Resource (Section 6.8.1.1) for which an ALTO Client may supply a list of PIDs to be included. A Filtered Network Map MAY be provided by an ALTO Server.

6.8.2.1.1. Media Type

See Section 6.8.1.1.1.

6.8.2.1.2. HTTP Method

This resource is requested using the HTTP POST method.

6.8.2.1.3. Input Parameters

Input parameters 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-networkmapfilter+json", which is a JSON Object of type ReqFilteredNetworkMap, where:

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

with members:

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

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

6.8.2.1.4. Capabilities

None.

6.8.2.1.5. Response

See Section 6.8.1.1.5 for the format.

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

parameters contain an address type that is not currently known to the ALTO Server, the ALTO Server MUST behave as if the address type did not appear in the input parameters.

6.8.2.1.6. Example

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

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

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

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

6.8.2.2. Filtered Cost Map

A Filtered Cost Map is a Cost Map Information Resource (Section 6.8.1.2) for which an ALTO Client may supply additional parameters limiting the scope of the resulting Cost Map. A Filtered

Cost Map MAY be provided by an ALTO Server.

6.8.2.2.1. Media Type

See Section 6.8.1.2.1.

6.8.2.2.2. HTTP Method

This resource is requested using the HTTP POST method.

6.8.2.2.3. Input Parameters

Input parameters 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 {  
  PIDName srcs<0..*>;  
  PIDName dsts<0..*>;  
} PIDFilter;  
  
object {  
  CostMode    cost-mode;  
  CostType    cost-type;  
  JSONString  constraints<0..*>;  [OPTIONAL]  
  PIDFilter   pids;                [OPTIONAL]  
} ReqFilteredCostMap;
```

with members:

cost-type The Cost Type (Section 5.1.1) for the returned costs. This MUST be one of the supported Cost Types indicated in this resource's capabilities (Section 6.8.2.2.4).

cost-mode The Cost Mode (Section 5.1.2) for the returned costs. This MUST be one of the supported Cost Modes indicated in this resource's capabilities (Section 6.8.2.2.4).

constraints Defines a list of additional constraints on which elements of the Cost Map are returned. This parameter MUST NOT be specified if this resource's capabilities (Section 6.8.2.2.4) indicate that constraint support is not available. A constraint contains two entities separated by whitespace: (1) an operator, 'gt' for greater than, 'lt' for less than, 'ge' for greater than or equal to, 'le' for less than or equal to, or 'eq' for equal to

(2) a target cost value. The cost value is a number that MUST be defined in the same units as the Cost Type indicated by the cost-type parameter. ALTO Servers SHOULD use at least IEEE 754 double-precision floating point [IEEE.754.2008] to store the cost value, and SHOULD perform internal computations using double-precision floating-point arithmetic. If multiple 'constraint' parameters are specified, they are interpreted as being related to each other with a logical AND.

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

6.8.2.2.4. Capabilities

The URI providing this resource supports all capabilities documented in Section 6.8.1.2.4 (with identical semantics), plus additional capabilities. In particular, the capabilities are defined by a JSON object of type `FilteredCostMapCapability`:

```
object {  
  CostMode cost-modes<0..*>;  
  CostType cost-types<0..*>;  
  JSONBool cost-constraints;  
} FilteredCostMapCapability;
```

with members:

cost-modes See Section 6.8.1.2.4.

cost-types See Section 6.8.1.2.4.

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

6.8.2.2.5. Response

See Section 6.8.1.2.5 for the format.

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

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

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

6.8.2.2.6. Example

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

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

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

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

6.8.3. Endpoint Property Service

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

6.8.3.1. Endpoint Property

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

6.8.3.1.1. Media Type

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

6.8.3.1.2. HTTP Method

This resource is requested using the HTTP POST method.

6.8.3.1.3. Input Parameters

Input parameters are supplied in the entity body of the POST request. This document specifies the data format of input parameters with the media type "application/alto-endpointpropparams+json", which is a JSON Object of type ReqEndpointProp:

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

with members:

properties List of endpoint properties to be returned for each endpoint. Each specified property MUST be included in the list of supported properties indicated by this resource's capabilities (Section 6.8.3.1.4). The ALTO Server MUST interpret entries appearing multiple times as if they appeared only once.

endpoints List of endpoint addresses for which the specified properties are to be returned. The ALTO Server MUST interpret entries appearing multiple times as if they appeared only once.

6.8.3.1.4. Capabilities

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

```
object {  
  EndpointProperty prop-types<0..*>;  
} EndpointPropertyCapability;
```

with members:

prop-types The Endpoint Property Types (see Section 6.6.6) supported by the corresponding URI. If not present, this member MUST be interpreted as an empty array.

6.8.3.1.5. Response

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

```
object {  
  JSONValue [EndpointProperty];  
  ...  
} EndpointProps;  
  
object {  
  EndpointProps [TypedEndpointAddr]<0..*>;  
  ...  
} EndpointPropertyMapData;  
  
object {  
  VersionTag          map-vtag;          [OPTIONAL]  
  EndpointPropertyMapData map;  
} InfoResourceEndpointProperty;
```

EndpointPropertyMapData has one member for each endpoint indicated in the input parameters (with the name being the endpoint encoded as a TypedEndpointAddr). The requested properties for each endpoint are encoded in a corresponding EndpointProps object, which encodes one name/value pair for each requested property, where the property names are encoded as strings of type EndpointProperty. An implementation of the protocol in this document SHOULD assume that the property value is a JSONString and fail to parse if it is not, unless the implementation is using an extension to this document that indicates when and how property values of other data types are signaled.

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

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

The ALTO Server MAY include the Version Tag (Section 5.3) of the Network Map used to generate the response (if desired and applicable) as the 'map-vtag' member in the response. If the 'pid' property is returned for any endpoints in the response, the 'map-vtag' member is

REQUIRED instead of OPTIONAL.

6.8.3.1.6. Example

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

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

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

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

6.8.4. Endpoint Cost Service

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

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

6.8.4.1. Endpoint Cost

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

It is important to note that although this resource allows an ALTO Server to reveal costs between individual endpoints, an ALTO Server

is not required to do so. A simple alternative would be to compute the cost between two endpoints as the cost between the PIDs corresponding to the endpoints. See Section 10.1 for additional details.

6.8.4.1.1. Media Type

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

6.8.4.1.2. HTTP Method

This resource is requested using the HTTP POST method.

6.8.4.1.3. Input Parameters

Input parameters are supplied in the entity body of the POST request. This document specifies input parameters with a data format indicated by media type "application/alto-endpointcostparams+json", which is a JSON Object of type ReqEndpointCostMap:

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

object {
  CostMode          cost-mode;
  CostType          cost-type;
  JSONString        constraints;    [OPTIONAL]
  EndpointFilter    endpoints;
} ReqEndpointCostMap;
```

with members:

cost-mode The Cost Mode (Section 5.1.2) to use for returned costs.
This MUST be one of the Cost Modes indicated in this resource's capabilities (Section 6.8.4.1.4).

cost-type The Cost Type (Section 5.1.1) to use for returned costs.
This MUST be one of the Cost Types indicated in this resource's capabilities (Section 6.8.4.1.4).

constraints Defined equivalently to the "constraints" input parameter of a Filtered Cost Map (see Section 6.8.2.2).

endpoints A list of Source Endpoints and Destination Endpoints for which Path Costs are to be returned. If the list of Source Endpoints is empty (or not included), the ALTO Server MUST interpret it as if it contained the Endpoint Address corresponding to the client IP address from the incoming connection (see Section 8.3 for discussion and considerations regarding this mode). The list of destination Endpoints MUST NOT be empty. The ALTO Server MUST interpret entries appearing multiple times in a list as if they appeared only once.

6.8.4.1.4. Capabilities

See Section 6.8.2.2.4.

6.8.4.1.5. Response

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

```
object EndpointDstCosts {  
  JSONValue [TypedEndpointAddr];  
  ...  
};  
  
object {  
  EndpointDstCosts [TypedEndpointAddr]<0..*>;  
  ...  
} EndpointCostMapData;  
  
object {  
  CostMode          cost-mode;  
  CostType          cost-type;  
  EndpointCostMapData map;  
} InfoResourceEndpointCostMap;
```

InfoResourceEndpointCostMap has members:

cost-mode The Cost Mode used in the returned Cost Map.

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

map The Endpoint Cost Map data itself.

EndpointCostMapData is a JSON object with each member representing a single Source Endpoint specified in the input parameters; the name for a member is the TypedEndpointAddr string identifying the

corresponding Source Endpoint. For each Source Endpoint, a `EndpointDstCosts` object denotes the associated cost to each Destination Endpoint specified in the input parameters; the name for each member in the object is the `TypedEndpointAddr` string identifying the corresponding Destination Endpoint. 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 extensions 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.

6.8.4.1.6. Example

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

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

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

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

7. Use Cases

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

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

7.1. ALTO Client Embedded in P2P Tracker

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

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

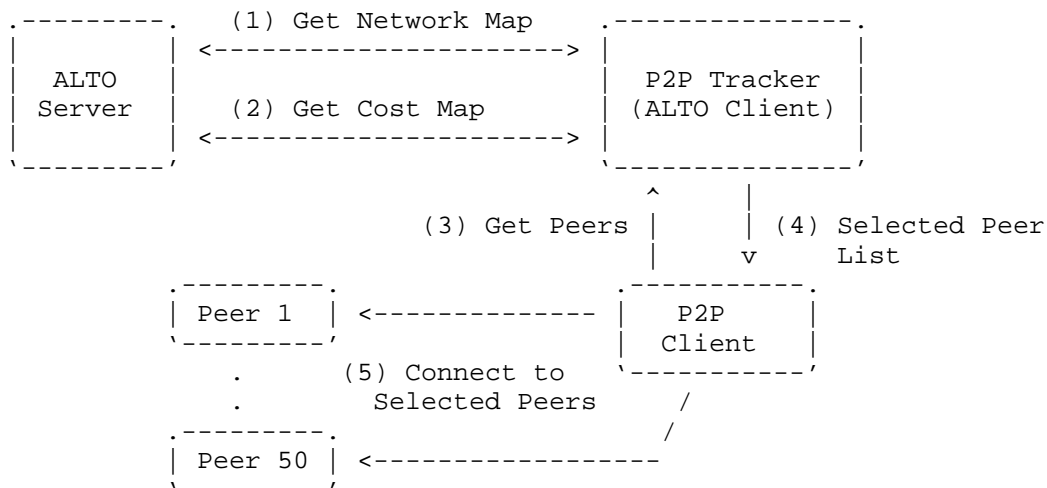


Figure 4: ALTO Client Embedded in P2P Tracker

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

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

2. The P2P Tracker requests the Cost Map amongst all PIDs from the ALTO Server.
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.

7.2. ALTO Client Embedded in P2P Client: Numerical Costs

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

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

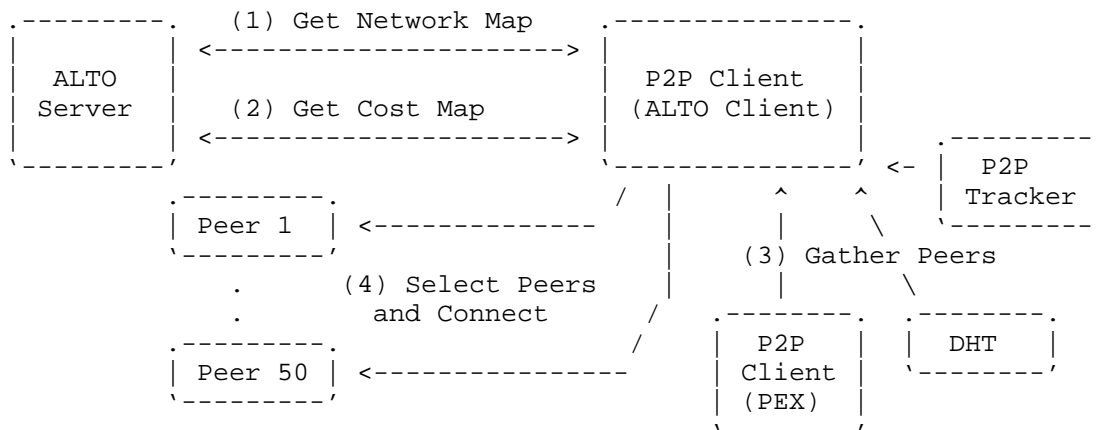


Figure 5: ALTO Client Embedded in P2P Client

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

1. The P2P Client requests the Network Map covering all PIDs from the ALTO Server servicing its own ISP.
2. The P2P Client requests the Cost Map amongst all PIDs from the ALTO Server. The Cost Map by default specifies numerical costs.
3. The P2P Client discovers peers from sources such as Peer Exchange (PEX) from other P2P Clients, Distributed Hash Tables (DHT), and P2P Trackers.
4. The P2P Client uses ALTO information as part of the algorithm for selecting new peers, and connects to the selected peers.

7.3. ALTO Client Embedded in P2P Client: Ranking

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

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

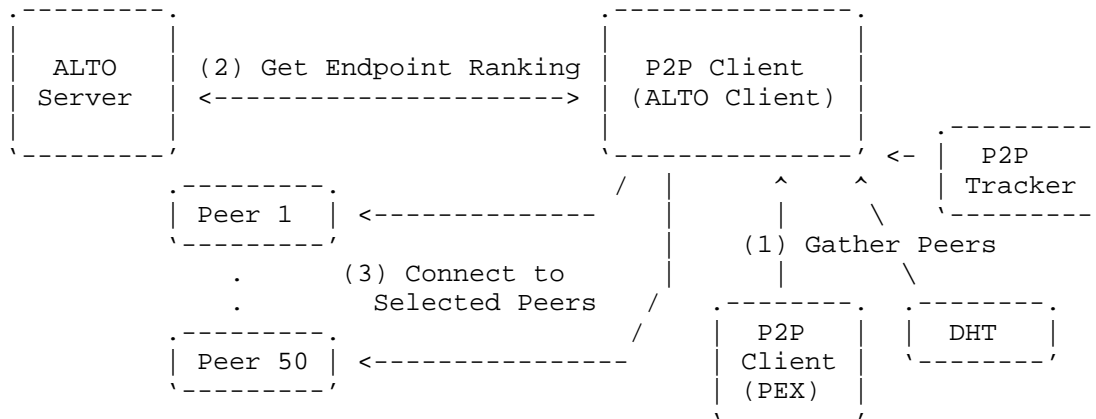


Figure 6: ALTO Client Embedded in P2P Client: Ranking

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

1. The P2P Client discovers peers from sources such as Peer Exchange (PEX) from other P2P Clients, Distributed Hash Tables (DHT), and P2P Trackers.
2. The P2P Client queries the ALTO Server's Ranking Service, including discovered peers as the set of Destination Endpoints, and indicates the 'ordinal' Cost Mode. The response indicates the ranking of the candidate peers.
3. The P2P Client connects to the peers in the order specified in the ranking.

8. Discussions

8.1. 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 6.7) to locate an Information Resource with the desired ALTO Information.

8.2. Hosts with Multiple Endpoint Addresses

In practical deployments, especially during the transition from IPv4 to IPv6, a particular host may be reachable using multiple addresses. Furthermore, the particular network path followed when sending packets to the host may differ based on the address that is used. Network providers may prefer one path over another (e.g., one path may have a NAT64 middlebox). 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 types of endpoint addresses. In supporting multiple address types, the ALTO Protocol also allows 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. Note that in general, the path through the network may differ dependent on the types of addresses that are used.

Note that there are limitations as to what information ALTO can provide in this regard. In particular, a particular ALTO Service provider may not be able to determine if connectivity with a particular endhost will succeed over IPv4 or IPv6, as this may depend upon information unknown to the ISP such as particular application implementations.

8.3. Network Address Translation Considerations

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

The protocol specified in this document provides a mode of operation where the source network location is computed by the ALTO Server (i.e., the 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 be dependent on the network interfaces, 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.

8.4. 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.

9. IANA Considerations

9.1. 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 6.7
application	alto-networkmap+json	Section 6.8.1.1
application	alto-networkmapfilter+json	Section 6.8.2.1
application	alto-costmap+json	Section 6.8.1.2
application	alto-costmapfilter+json	Section 6.8.2.2
application	alto-endpointprop+json	Section 6.8.3.1
application	alto-endpointpropparams+json	Section 6.8.3.1
application	alto-endpointcost+json	Section 6.8.4.1
application	alto-endpointcostparams+json	Section 6.8.4.1
application	alto-error+json	Section 6.5

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 10.

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).

9.2. ALTO Cost Type Registry

This document requests the creation of an ALTO Cost Type registry to be maintained by IANA.

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

New ALTO Cost 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 ALTO Cost Type 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 Type should be interpreted. Updates and deletions of ALTO Cost Types follow the same procedure.

Registered ALTO Cost Type identifiers MUST conform to the syntactical requirements specified in Section 6.6.5. Identifiers are to be recorded and displayed as ASCII strings.

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

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

- o Identifier: The name of the desired ALTO Cost Type.
- 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 Type 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 Type.

This specification requests registration of the identifier 'routingcost'. Semantics for the this Cost Type are documented in Section 5.1.1.1, and security considerations are documented in Section 10.1.

9.3. ALTO Endpoint Property Registry

This document requests the creation of an ALTO Endpoint Property registry to be maintained by IANA.

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

New ALTO Endpoint Properties 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 Endpoint Property 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 Endpoint Properties should be interpreted. Updates and deletions of ALTO Endpoint Properties follow the same procedure.

Registered ALTO Endpoint Property identifiers MUST conform to the syntactical requirements specified in Section 6.6.6. 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 Endpoint Property.
- o Intended Semantics: ALTO Endpoint Properties 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.
- o Security Considerations: ALTO Endpoint Properties expose information to ALTO Clients. As such, proper usage of a particular Endpoint Properties 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 an Endpoint Property.

This specification requests registration of the identifier 'pid'. Semantics for the this Endpoint Property are documented in Section 4.1, and security considerations are documented in Section 10.1.

9.4. ALTO Address Type Registry

This document requests the creation of an ALTO Address Type registry to be maintained by IANA.

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 6.6.3.1. Identifiers

are to be recorded and displayed as ASCII strings.

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

- 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 6.6.3.2).
- o Endpoint Prefix Encoding: The procedure for encoding a set of addresses of the registered type as an EndpointPrefix (see Section 6.6.3.3). If no such compact encoding is available, the same encoding used for a singular address may be used. In such a case, it must be documented that sets of addresses of this type always have exactly one element.
- 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'. Endpoint Address Encoding is documented in Section 6.6.3.2.1 and Section 6.6.3.2.2. Endpoint Prefix Encoding is documented in Section 6.6.3.3.1 and Section 6.6.3.3.2. Since these are registrations for IPv4 and IPv6 address types, no mapping is required. Security considerations are documented in Section 10.1 and Section 10.2.

10. Security Considerations

10.1. Privacy Considerations for ISPs

ISPs must be cognizant of the network topology and provisioning information provided through ALTO Interfaces. ISPs should evaluate how much information is revealed and the associated risks. On the one hand, providing overly fine-grained information may make it easier for attackers to infer network topology. In particular, attackers may try to infer details regarding ISPs' operational policies or inter-ISP business relationships by intentionally posting

a multitude of selective queries to an ALTO server and analyzing the responses. Such sophisticated attacks may reveal more information than an ISP hosting an ALTO server intends to disclose. On the other hand, revealing overly coarse-grained information may not provide benefits to network efficiency or performance improvements to ALTO Clients.

10.2. ALTO Clients

Applications using the information must be cognizant of the possibility that the information is malformed or incorrect. Even if an ALTO Server has been properly authenticated by the ALTO Client, the information provided may be malicious because the ALTO Server and its credentials have been compromised (e.g., through malware). Other considerations (e.g., relating to application performance) can be found in Section 6 of [RFC5693].

ALTO Clients should also be cognizant of revealing Network Location Identifiers (IP addresses or fine-grained PIDs) to the ALTO Server, as doing so may allow the ALTO Server to infer communication patterns. One possibility is for the ALTO Client to only rely 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.

In addition, ALTO clients should be cautious not to unintentionally or indirectly disclose the resource identifier (of which they try to improve the retrieval through ALTO-guidance), e.g., the name/identifier of a certain video stream in P2P live streaming, to the ALTO server. Note that the ALTO Protocol specified in this document does not explicitly reveal any resource identifier to the ALTO Server. However, for instance, depending on the popularity or other specifics (such as language) of the resource, an ALTO server could potentially deduce information about the desired resource from information such as the Network Locations the client sends as part of its request to the server.

10.3. Authentication, Integrity Protection, and Encryption

SSL/TLS can provide encryption and integrity protection of transmitted messages as well as authentication of the ALTO Client and Server. HTTP Basic or Digest authentication can provide authentication of the client (combined with SSL/TLS, it can additionally provide encryption, integrity protection and server authentication).

Issues resulting from an attacker controlling the data received by an ALTO Client are discussed in Section 10.2.

An ALTO Server may optionally use authentication (and potentially encryption) to limit the parties with whom ALTO information is directly shared. There may be special use cases where encryption of ALTO information is desirable. In many cases, however, information sent out by an ALTO Server may be regarded as non-confidential information.

ISPs 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.

10.4. ALTO Information Redistribution

It is possible for applications to redistribute ALTO information to improve scalability. Even with such a distribution scheme, ALTO Clients obtaining ALTO information must be able to validate the received ALTO information to ensure that it was generated by an appropriate ALTO Server. Support for this validation is not provided in this document, but may be provided by extension documents.

10.5. Denial of Service

ISPs should be cognizant of the workload at the ALTO Server generated by certain ALTO Queries, such as certain queries to the Map Filtering Service and Ranking Service. 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). One way to limit Denial-of-Service attacks is to employ access control to the ALTO server. Another possible mechanism for an ALTO Server to protect itself against a multitude of computationally expensive bogus requests is to demand that each ALTO Client to solve a computational puzzle first before allocating resources for answering a request (see, e.g., [I-D.jennings-sip-hashcash]). The current specification does not use such computational puzzles, and discussion regarding tradeoffs of such an approach would be needed before including such a technique in the ALTO Protocol.

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

10.6. ALTO Server Access Control

In order to limit access to an ALTO server (e.g., for an ISP to only allow its users to access its ALTO server, or to prevent Denial-of-Service attacks by arbitrary hosts from the Internet), an ALTO server may employ access control policies. Depending on the use-case and scenario, an ALTO server may restrict access to its services more strictly or rather openly (see [I-D.stiemerling-alto-deployments] for a more detailed discussion on this issue).

11. References

11.1. Normative References

- [IEEE.754.2008]
Institute of Electrical and Electronics Engineers,
"Standard for Binary Floating-Point Arithmetic", IEEE
Standard 754, August 2008.
- [RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail
Extensions (MIME) Part Two: Media Types", RFC 2046,
November 1996.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H.,
Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext
Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform
Resource Identifier (URI): Generic Syntax", STD 66,
RFC 3986, January 2005.
- [RFC4627] Crockford, D., "The application/json Media Type for
JavaScript Object Notation (JSON)", RFC 4627, July 2006.
- [RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing
(CIDR): The Internet Address Assignment and Aggregation
Plan", BCP 122, RFC 4632, August 2006.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an
IANA Considerations Section in RFCs", BCP 26, RFC 5226,
May 2008.
- [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing,
"Session Traversal Utilities for NAT (STUN)", RFC 5389,

October 2008.

- [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.

11.2. Informative References

- [BitTorrent]
"Bittorrent Protocol Specification v1.0",
<<http://wiki.theory.org/BitTorrentSpecification>>.
- [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.gu-alto-redistribution]
Yingjie, G., Alimi, R., and R. Even, "ALTO Information Redistribution", draft-gu-alto-redistribution-03 (work in progress), July 2010.
- [I-D.ietf-alto-reqs]
Previdi, S., Stiemerling, M., Woundy, R., and Y. Yang, "Application-Layer Traffic Optimization (ALTO) Requirements", draft-ietf-alto-reqs-08 (work in progress), March 2011.
- [I-D.ietf-alto-server-discovery]
Kiesel, S., Stiemerling, M., Schwan, N., Scharf, M., and S. Yongchao, "ALTO Server Discovery",
draft-ietf-alto-server-discovery-03 (work in progress),
March 2012.
- [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.mrw-nat66]

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

[I-D.p4p-framework]

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

[I-D.saumitra-alto-multi-ps]

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

[I-D.saumitra-alto-queryresponse]

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

[I-D.shalunov-alto-infoexport]

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

[I-D.stiemerling-alto-deployments]

Stiemerling, M. and S. Kiesel, "ALTO Deployment Considerations", draft-stiemerling-alto-deployments-06 (work in progress), January 2011.

[I-D.wang-alto-p4p-specification]

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

[P4P-SIGCOMM08]

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

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

October 2009.

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

Appendix A. Acknowledgments

Thank you to Jan Seedorf for contributions to the Security Considerations section.

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

Obi Akonjang (DT Labs/TU Berlin),
Saumitra M. Das (Qualcomm Inc.),
Syon Ding (China Telecom),
Doug Pasko (Verizon),
Laird Popkin (Pando Networks),
Satish Raghunath (Juniper Networks),
Albert Tian (Ericsson/Redback),
Yu-Shun Wang (Microsoft),
David Zhang (PPLive),
Yunfei Zhang (China Mobile).

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

Appendix B. Authors

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

Stefano Previdi
Cisco

Email: sprevidi@cisco.com

Stanislav Shalunov
BitTorrent

Email: shalunov@bittorrent.com

Richard Woundy
Comcast

Richard_Woundy@cable.comcast.com

Authors' Addresses

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

Email: ralimi@google.com

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

Email: repenno@cisco.com

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

Email: yry@cs.yale.edu

ALTO
Internet-Draft
Intended status: Standards Track
Expires: January 17, 2013

S. Kiesel
University of Stuttgart
M. Stiemerling
NEC Europe Ltd.
N. Schwan
M. Scharf
Alcatel-Lucent Bell Labs
H. Song
Huawei
July 16, 2012

ALTO Server Discovery
draft-ietf-alto-server-discovery-04

Abstract

The goal of Application-Layer Traffic Optimization (ALTO) is to provide guidance to applications, which have to select one or several hosts from a set of candidates that are able to provide a desired resource.

Entities seeking guidance need to discover and possibly select an ALTO server to ask. This is called ALTO server discovery. This memo describes an ALTO server discovery mechanism based on several alternative mechanisms that are applicable when the resource consumer is co-located with the ALTO client.

Requirements Language

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

Status of this Memo

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

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

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

This Internet-Draft will expire on January 17, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	4
1.1. Discovery Scenarios	5
1.1.1. ALTO Server Discovery by Resource Consumers	6
1.1.2. ALTO Server Discovery by a Third Party	7
1.2. Pre-Conditions	8
2. Protocol Overview	10
3. Retrieving the URI by U-NAPTR	12
3.1. Retrieving the Domain Name	12
3.1.1. Option 1: User input	12
3.1.2. Option 2: DHCP	13
3.1.3. Option 3: PPP	13
3.2. U-NAPTR Resolution	14
4. Applicability	16
4.1. Applicability for Resource Consumer Server Discovery	16
4.2. Applicability for Third Party Server Discovery	16
5. Deployment Considerations	18
5.1. DHCP option for DNS Suffix	18
5.2. PPP option for DNS Suffix	18
6. IANA Considerations	19
6.1. Registration of PPP IPCP Configuration Option Type	19
6.2. Registration of U-NAPTR application service tag	19
7. Security Considerations	20
7.1. General	20
7.2. For U-NAPTR	20
8. Conclusion	22
9. References	23
9.1. Normative References	23
9.2. Informative References	23
Appendix A. Contributors List and Acknowledgments	25
Authors' Addresses	26

1. Introduction

The goal of Application-Layer Traffic Optimization (ALTO) is to provide guidance to applications, which have to select one or several hosts from a set of candidates, that are able to provide a desired resource [RFC5693]. The requirements for ALTO are itemized in [I-D.ietf-alto-reqs].

ALTO is realized by a client-server protocol. ALTO clients send queries to ALTO servers, in order to solicit guidance. Hence, ALTO clients need to know the contact information of ALTO servers, which can provide appropriate guidance for a given resource consumer. Typically the closer an ALTO server is to a resource consumer the more accurate guidance it can provide. Thus a design objective is to automatically discover an ALTO server topologically close to the network attachment point of the resource consumer, if available. The contact information of the ALTO server is retrieved by invoking the ALTO discovery procedure defined in this document.

The ALTO protocol specification [I-D.ietf-alto-protocol] is based on HTTP. Therefore, it expects that the ALTO discovery procedure yields the HTTP(S) URI of the ALTO server's Information Resource Directory, which gives further information about the capabilities and services provided by that ALTO server. Further (DNS) lookups may be necessary in order to find out the ALTO server's IP address.

The goal of this memo is to propose a discovery mechanism for ALTO client deployments in uncontrolled environments that is implementable and deployable at a fast pace, i.e., without creating other deployment dependencies for ALTO. We propose a schema which employs the U-NAPTR mechanism [RFC4848] to determine the URI of the ALTO server and where multiple input methods to the U-NAPTR process can be used. U-NAPTR is used because the discovery mechanism must return an URI, and thus other discovery mechanisms are not applicable (e. g., DNS SRV records).

There are various architectural options where to place the ALTO client and the ALTO server discovery procedure:

- o One option is that the ALTO client and the ALTO server discovery procedure are embedded directly in the resource consumer, i.e., the application protocol entity that will eventually initiate data transmission to/from the selected resource provider(s). In this case, the ALTO server discovery procedure might be able to interact with the user (i.e., prompt for a host name). Furthermore, it may use services such as DHCP, which are only available within the access network to which the resource consumer is connected. This option is described in this memo.

- o A similar architectural setup exists in controlled environments, e.g. a CDN that uses ALTO as information base for request redirection. However in such a controlled environment it is unlikely that a dynamic server discovery is necessary. Instead it is expected that the CDN administrator will manually configure the contact information of the ALTO server. Thus this draft is focused on allowing dynamic discovery in uncontrolled environments where provisioning contact information is impossible or undesirable.
- o Another option is to integrate the ALTO client and the ALTO server discovery procedure into a third party such as a resource directory ("peer-to-peer tracker"), which issues ALTO queries on behalf of various resource consumers. This third party may reside in a different part of the network (administrative domain) than the resource consumer. It may occur that said third party wishes to issue ALTO queries on behalf of a resource consumer, but all it knows about the resource consumer is the source IP address of messages originating from it (i.e., the resource consumer's IP address or the "public" IP address of the outermost NAT in front of the resource consumer). A general server discovery procedure based solely on this IP address raises several issues, and are out of scope of this document. This option is further discussed in [I-D.kist-alto-3pdisc].

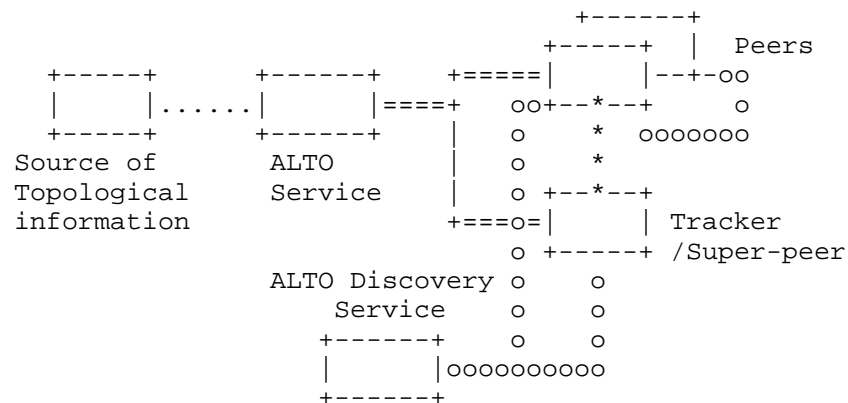
A more detailed discussion of various options where to place the functional entities comprising the overall ALTO architecture can be found in [I-D.ietf-alto-deployments].

A general mechanism that redirects ALTO clients from one ALTO server to another, potentially closer, ALTO server raises several issues. First ALTO servers by definition provide Network Maps for the whole IP address space and thus can provide each client with a potentially useful answer. Second ALTO servers might be deployed independently by separate administrative entities and are thus not necessarily aware of each other. The use of a redirection mechanism thus depends on the deployment scenario and is for this reason out of the scope of this document.

Comments and discussions about this memo should be directed to the ALTO working group: alto@ietf.org.

1.1. Discovery Scenarios

Figure 1 below shows an overview on the different entities of a generic ALTO framework. The ALTO Server discovery mechanism is used by the peer-to-peer (P2P) application in order to retrieve the point of contact of the ALTO Service.



Legend:

=== ALTO query protocol
 ooo ALTO service discovery protocol
 *** Application protocol (out of scope)
 ... Provisioning or initialization (out of scope)

Figure 1: ALTO Discovery Overview

Hereby the ALTO service discovery scenarios are classified into two types: one is the ALTO server discovery by the resource consumer, and the other is the ALTO server discovery by a third party, such as application trackers. Before the specification of the discovery mechanism the following section illustrates and discusses both scenarios.

1.1.1. ALTO Server Discovery by Resource Consumers

The ALTO service discovery in some scenarios needs to be performed by the resource consumer itself. In particular in P2P applications without a tracker like DHTs and other conventional client/server applications. Also P2P application which are tracker based may embed the ALTO client into the resource consumer to allow peers a selection of peers after retrieving the peer list from the application tracker.

The following figure illustrates this scenario, showing the relationship between the different entities as discussed before.

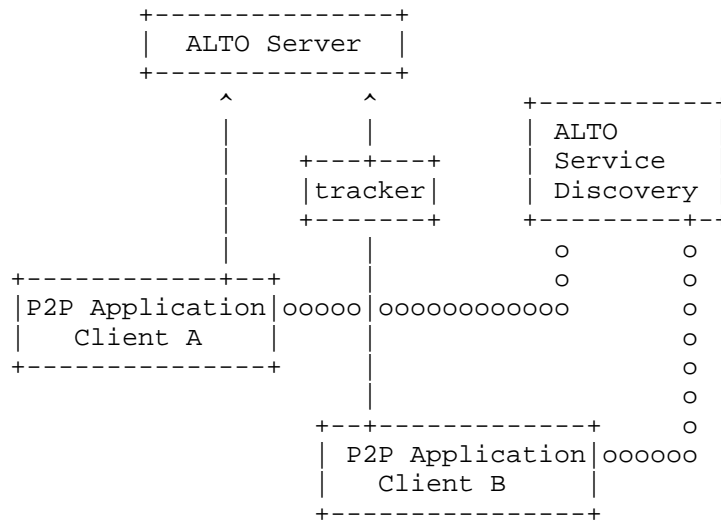


Figure 2: Resource Consumer ALTO Server Discovery (Example)

1.1.2. ALTO Server Discovery by a Third Party

Some P2P applications have trackers, and these applications might not need to have their clients looking for the ALTO server guidance. In these scenarios trackers query the ALTO servers for guidance themselves, and then return the final ranked result to the application clients. However, application clients are distributed among different network operators and autonomous systems. Trackers thus need to find different ALTO servers for the clients located in different operator networks or autonomous systems. In such scenarios the discovery is thus not performed by the resource consumer, but a third party entity on behalf of the resource consumer.

This third party ALTO server discovery raises several issues. Typically only the IP address of the ALTO client is known that needs to suffice for finding the corresponding ALTO server. This requires determining a FQDN as input for the U-NAPTR process. One option to do this is to use a reverse DNS lookup. However, reverse DNS has several limitations:

First, there is no established unique way of maintaining the DNS tree, and there are different practices in different networks. Furthermore, it is possible that a lookup fails or that the returned value is not valid. For instance, it can point to a different domain. As a result, any potential use of reverse DNS lookup for service discovery must be able to deal with failures of lookup and react accordingly.

Second, determining a domain name from IP addresses by tree climbing is problematic, in particular for IPv6. [RFC4472] discusses the issues for IPv6.

Third, populating a DNS name space what looks like a reverse tree is a significant administrative DNS overhead.

Finally it must be emphasized that any tree walking procedure raises several issues. There is no single best way tree, and heuristics are needed.

Given these problems, this memo does not specify a reverse DNS mechanism to determine a FQDN.

Thus, this draft is limited to scenarios where the discovery procedure is done by the resource consumer. In case a third party needs to know the contact information of the ALTO server it is RECOMMENDED that the resource consumer discovers the ALTO server and then sends the contact information directly to the third party. However, this requires a modification of the protocol used between the resource consumer and the third party, e.g., the tracker protocol in the peer-to-peer case. As a potential alternative, the client could provide a valid FQDN, so that the third party can use this as input for the U-NAPTR process, but this variant has no significant advantages. The options on how to transmit the contact information from the resource consumer to the third party are out of scope of this document.

1.2. Pre-Conditions

In general, the ALTO server discovery SHOULD be based on the IP address that is used to communicate with other peers, i. e., it should return a server that can provide guidance for that address.

In order to achieve this, the whole document assumes certain pre-conditions, in particular:

- o The ALTO server discovery procedure is executed on a per IP family base, i.e., separate for IPv4 and IPv6. It is up to the ALTO client to decide which of the possible multiple results of different IP address families to use. The choice of whether to use IPv4 or IPv6 is out of scope of this document.
- o A change of the IP address at an interface invalidates the result of the ALTO server discovery procedure. For instance, if the IP address assigned to a mobile host changes due to host mobility, it is required to run the ALTO server discovery procedure for the new IP address without relying on earlier gained information.

- o The ALTO server discovery procedure is executed on a per IP address base. Multiple IP addresses per interface or multiple IP addresses assigned to different IP interfaces require to repeat the procedure for every IP address. It may be fine to group IP addresses according their domain suffixes and to perform the procedure for such a group. However, this is out of scope of this document.
- o There are several challenges with DNS on hosts with multiple interfaces [RFC6418], which can affect the ALTO server discovery. If the DNS resolution is performed on the wrong interface, it can return an ALTO server that could provide sub-optimal or wrong guidance. Finding the best ALTO server for multi-interfaced hosts is outside the scope of this document.
- o The discovery procedure may need information about the public IP address and thus have to discover NATs. Details of NAT discovery are not discussed in this memo.
- o Similarly the discovery procedure may need information about the IP address of a proxy or agent if this address is used for communication. Examples include Virtual Private Network (VPN) or mobile IP triangular routing. Details on how to obtain the IP address that is used for communication in such setups are not discussed in this memo, since they can be specific to the network in use.

2. Protocol Overview

We define multiple alternatives to discover the IP address of the ALTO server, as there are a number of ways possible how such information can be provided to the ALTO client. The choice of method is up to the local network deployment. For instance, there can be deployments where the ALTO server in charge for ALTO client is provisioned by the network operator and communicated to the ALTO client's host via a DHCP option, while in other deployments no such means may exist.

It should be noted that there is no silver bullet solution to the ALTO server discovery, as there too many deployment scenarios in the server discovery space.

The following figure illustrates the different protocols that SHOULD be used to find the URI of a suitable ALTO server.

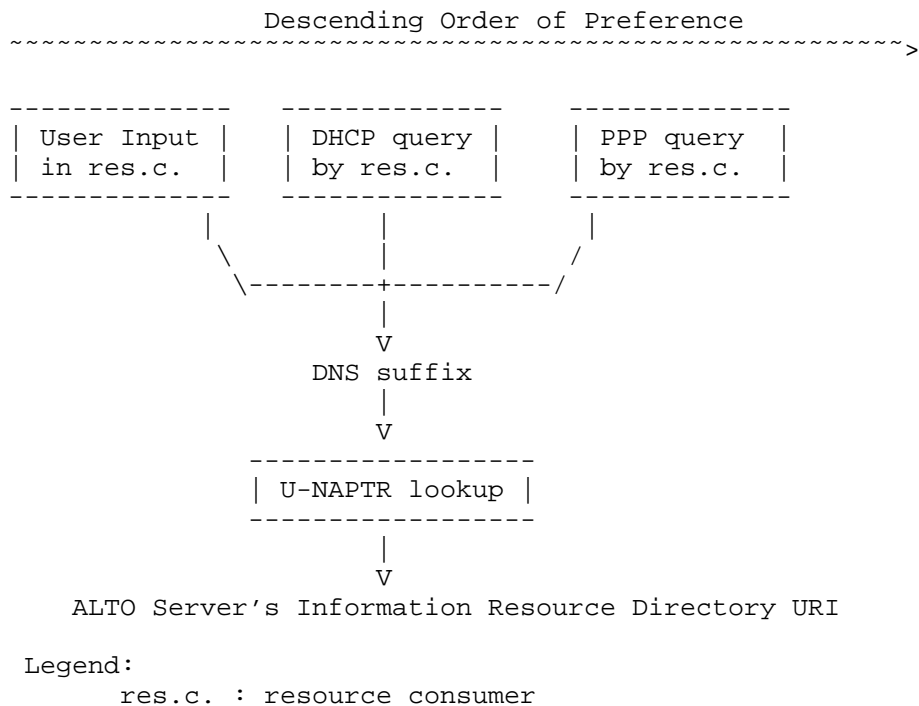


Figure 3: Protocol Overview

Figure 3 illustrates the U-NAPTR based resolution process to retrieve the ALTO Server URI. As a precondition for resolution, the U-NAPTR

process needs the right DNS suffix as input. This domain name is typically the domain name of the client's IP address. In order to retrieve the DNS suffix we specify three options, which are listed in descending order of preference. A client SHOULD use the first method that determines a DNS suffix. It MAY try the other methods in case the U-NAPTR lookup failed.

User input: A user MAY manually specify the DNS suffix on its own, either to access a 3rd party ALTO service provider or as it does know such information. This input MAY also origin from a web page where the user downloads the configuration, which is loaded as user input, or obtained by other discovery methods.

DHCP: A network provider MAY provide the DNS suffix through a DHCP option.

PPP: A network provider MAY provide the DNS suffix through a Point-to-Point (PPP) Internet Protocol Control Protocol (IPCP) extension.

This discovery method SHOULD be repeated if the resource consumer moves, i. e., if its IP address changes.

Given that DHCP and PPP are widely used for IP address configuration, this discovery method is applicable to many networks. The mechanism described in this document can also be applied to further networks if they provide an equivalent mechanism to retrieve the DNS suffix. Such straightforward extensions to other technologies are not specified in this memo.

Instead of using the standard ALTO server discovery method, applications MAY also use own methods to discover an ALTO server. This variant is outside the scope of this document.

3. Retrieving the URI by U-NAPTR

This section specifies the U-NAPTR based resolution process. To start the U-NAPTR resolution process a domain name is required as input. Thus the section is divided into two parts: Section 3.2 describes the U-NAPTR resolution process itself. How the client identifies this DNS suffix of the access network where the resource consumer is registered in is described in Section 3.1.

3.1. Retrieving the Domain Name

The U-NAPTR resolution process requires a domain name as input. The algorithm that SHOULD be applied to determine this domain name is described in this section. We specify three different options. In option 1 the user manually configures a specific ALTO service instance that he wants to use. Option 2 defines a DHCP and option 3 defines a PPP IPCP option to allow the network service provider a remote configuration of the client.

In general, the ALTO server discovery SHOULD be based on the IP address that is used to communicate with other peers. The resource consumer may have private IP addresses and public IP addresses and depending on the deployment it might be necessary to determine for all IP addresses the ALTO server in charge of. To determine its public IP address the resource consumer may need to use STUN[RFC5389] or BEP24[bep24]. Determining the correct IP address out of multiple options strongly depends on the deployment scenario but is out of scope for this document, although we discuss it to some extent in Section 4. For the following examples we assume that the IP address of the resource consumer is a.b.c.d.

3.1.1. Option 1: User input

A user may want to use a third party ALTO service instance. Therefore we allow the user to specify a DNS suffix on its own, for example in a config file option. The DNS suffix given by the user is combined with the IP address of the resource consumer to allow the third party ALTO service to direct the client to a suitable ALTO server based on the location of the client. A possible DNS suffix entered by the user may be:

myaltoprovider.org

In case not ALTO NAPTR records are found, we consider the discovery process based on user input as failed. A client MAY try one of the other options.

3.1.2. Option 2: DHCP

As a second option network operators MAY configure the domain name to be used for service discovery within an access network using DHCP. RFC 5986[RFC5986] defines DHCP IPv4 and IPv6 access network domain name options that identify a domain name that is suitable for service discovery within the access network. The ALTO server discovery procedure uses these DHCP options to retrieve the domain name as an input for the U-NAPTR resolution. One example could be:

example.com

3.1.3. Option 3: PPP

In some network deployments the PPP [RFC1661] IPCP [RFC1332] is used to do network configuration of residential user equipment, such as assigning an IP address or the name of a DNS server[RFC1877]. Thus as a third option a network operator MAY configure the domain name to be used for the server discovery using a PPP IPCP extension. The next section specifies the extension for configuration of the access network domain name, which can be used as input for the U-NAPTR process. One possible example yielded by this extension could be:

example.com

3.1.3.1. Access Network Name Encoding

This section describes the encoding of the domain name used in PPP IPCP extension Section 3.1.3.2.

The domain name is encoded according to Section 3.1 of [RFC1035] whereby each label is represented as a one-octet length field followed by that number of octets. This is the same encoding as in DHCP according to RFC 5986[RFC5986]. Since every domain name ends with the null label of the root, a domain name is terminated by a length byte of zero. The high-order two bits of every length octet MUST be zero, and the remaining six bits of the length field limit the label to 63 octets or less. To simplify implementations, the total length of a domain name (i.e., label octets and label length octets) is restricted to 255 octets or less.

For example, the domain "example.com." is encoded in 13 octets as:

```
+---+---+---+---+---+---+---+---+---+---+---+---+---+
| 7 | e | x | a | m | p | l | e | 3 | c | o | m | 0 |
+---+---+---+---+---+---+---+---+---+---+---+---+---
```

3.1.3.2. Access Network Domain Name IPCP Configuration Option

This Configuration Option defines a method for negotiating with the remote peer the name of the Access Network Domain Name to be used on the local end of the link.

A summary of the Access Network Domain Name Configuration Option format is shown below. The fields are transmitted from left to right.

Type	Len	Access Network Domain Name				
-----	-----	-----	-----	-----	-----	-----
TBD	n	s1	s2	s3	s4	s5 ...
-----	-----	-----	-----	-----	-----	-----

The values s1, s2, s3, etc. represent the domain name labels in the domain name encoding. Note that the length field in the IPCP option represents the length of the entire domain name encoding, whereas the length fields in the domain name encoding (see Section 3.1.3.1) is the length of a single domain name label.

Type: to be assigned by IANA

Len: Length of the 'Access Network Domain Name' field in octets.

Access Network Domain Name: The domain name of the Access Network for the client to use.

3.2. U-NAPTR Resolution

The ALTO protocol specification [I-D.ietf-alto-protocol] expects that the ALTO discovery procedure yields the HTTP(S) URI of the ALTO server's Information Resource Directory, which gives further information about the capabilities and services provided by that ALTO server. The first step of the ALTO server discovery procedure (see Section 3.1) yielded an U-NAPTR/DDDS (URI-Enabled NAPTR/Dynamic Delegation Discovery Service) [RFC4848] application unique strings, in the form of a DNS name. An example is "example.com".

In the second step, the ALTO Server discovery procedure needs to use the U-NAPTR [RFC4848] specification described below to obtain a URI (indicating host and protocol) for the ALTO server's Information Resource Directory. In this document, only the HTTP and HTTPS URL schemes are defined, as the ALTO protocol specification defines the access over both protocols, but no other [I-D.ietf-alto-protocol]. Note that the HTTP URL can be any valid HTTP(s) URL, including those

containing path elements.

The following two DNS entries show the U-NAPTR resolution for "example.com" to the HTTPS URL `https://altoserver.example.com/secure/directory` or the HTTP URL `http://altoserver.example.com/directory`, with the former being preferred.

example.com.

```
IN NAPTR 100 10 "u" "ALTO:https"  
"!.*!https://altoserver.example.com/secure/directory!" ""  
  
IN NAPTR 200 10 "u" "ALTO:http"  
"!.*!http://altoserver.example.com/directory!" ""
```

There is a potential that retrieveing the domain name or the U-NAPTR lookup itself does not yield to a result, i.e. no ALTO NAPTR record is found. In this case the discovery procedure failed for this IP address. It is RECOMMENDED that clients give up the discovery process and wait a period of time before repeating the procedure. Clients MAY repeat the discovery procedure for a different IP address instantaneously.

4. Applicability

This section discusses the applicability of the proposed solution with respect to the resource consumer server discovery and the third party deployment scenarios. Each section discusses the proposed steps that are needed to determine the ALTO Server URI.

4.1. Applicability for Resource Consumer Server Discovery

In this scenario the ALTO server discovery procedure is performed by the resource consumer, for example a peer in a P2P system. After the discovery the peer does the ALTO query on its own, or it might share the ALTO server contact information with a third party, for example a tracker, which then executes the ALTO query on behalf of the peer.

To complete the ALTO server discovery process the resource consumer first SHOULD check whether the user has provided the domain name through manual configuration. If this is not the case the next step SHOULD be to check for the access network domain name DHCP option (Section 3.1.2). Finally the client SHOULD try to retrieve the domain name by the PPP option Section 3.1.3.

A client can have several candidate IP addresses that it may use for the discovery process. For example if it is located behind a NAT, a private and a public IP address may be used for the discovery process. It depends on the deployment scenario which of the IP addresses is the correct one. Thus it is out-of-scope of this document to specify how exactly the client finds the right IP address. However in the following we list methods that may be used in order to determine these candidate IP addresses. Generally in P2P environments peers already have implemented mechanisms for NAT-traversal. This includes proprietary solutions to determine a peer's public IP address, for example by asking a neighbour peer about its record of the own IP address. Non-proprietary solutions that are favorable include the Session Traversal Utilities for NAT (STUN) [RFC5986] protocol to determine the public address. If the client is behind a residential gateway another option may be to use Universal Plug and Play (UPnP) [UPnP-IGD-WANIPConnection1] or the NAT Port Mapping Protocol (NAT-PMP) [I-D.cheshire-nat-pmp].

In case the ALTO discovery client has determined the domain name through one of the described options it proceeds with the U-NAPTR lookup as described in Section 3.2.

4.2. Applicability for Third Party Server Discovery

In case of the third party server discovery deployment scenario the entity performing the ALTO server discovery process is different from

the resource consumer. Typically the resource consumer is a peer whereas the ALTO client is a resource directory which seeks for ALTO guidance on behalf of the peer. Another use case for the third party discovery is an application that looks for ALTO guidance transparently for the resource consumer, for example a CDN.

Here the ALTO server discovery process can also retrieve guidance through the PPP/DHCP options or manual user configuration, but only if the provided discovery information is forwarded by the resource consumer to the third party entity. In this case, additional mechanisms for the forwarding of this discovery information need to be specified. However these mechanisms are out of scope of this document.

In case the resource consumer needs guidance for a different IP address, for example one from a private network, we recommend that the resource consumer discovers the server itself and forwards the ALTO server contact information directly to the third party entity, which in turn can then do the third party ALTO query. Again, forwarding the contact information from the resource consumer to the third party entity is out of scope of this document.

5. Deployment Considerations

The mechanism specified in this document needs some configuration effort in order to work properly.

5.1. DHCP option for DNS Suffix

Section 3.1.2 describes the usage of a DHCP option. It enables the network operator of the network, in which the ALTO client is located, to provide a DNS suffix. However, this assumes that this particular DHCP option is correctly passed from the DHCP server to the actual host with the ALTO client, and that the particular host understands this DHCP option. This memo assumes the client to be able to understand the proposed DHCP option, otherwise there is no further use of the DHCP option, but the client has to use the other proposed mechanisms.

There are well-known issues with the handling of DHCP options in home gateways. One issue is that unknown DHCP options are not passed through some home gateways, effectively eliminating the DHCP option.

Another well-known issue is the usage of home gateway specific DNS suffixes which "override" the DNS suffix provided by the network operator. For instance, a host behind a home gateway may receive a DNS suffix ".local" instead of "example.com". This suffix is not usable for the server discovery procedure.

5.2. PPP option for DNS Suffix

Section 3.1.3 describes the usage of a PPP option. It enables the network operator of the network, in which the ALTO client is located, to provide a DNS suffix. In residential networks, PPP is often terminated in the residential gateway. The ALTO client may run on hosts behind that gateway. As a result, the information may have to be passed to the client. The residential gateway could for instance use the DHCP option for that.

6. IANA Considerations

6.1. Registration of PPP IPCP Configuration Option Type

The IANA is requested to assign an Type code for the PPP IPCP Configuration Option Types for an Access Network Domain Name, as described in Section 3.1.3.2 of this document.

[TO BE REMOVED: This registration should take place at the following location: <http://www.iana.org/assignments/ppp-numbers>]

6.2. Registration of U-NAPTR application service tag

The IANA is requested to register the following U-NAPTR application service tag:

Application Service Tag: ALTO

Intended usage: Identifies a service that provides a Device with its location information.

Defining Publication: The specification contained within this document.

Contact information: The authors of this document

Author/Change controller: The IESG

7. Security Considerations

7.1. General

There are two different failures for the ALTO server discovery, which can both be caused by malicious attacks or by configuration problems, e. g., in case of DNS configuration errors or multi-homed hosts.

First, the discovery might not be able to discover an ALTO server, even if a suitable ALTO server exists. In that case, ALTO guidance will not be used. The resulting application performance and traffic distribution will correspond to a deployment scenario without ALTO guidance. But given that users cannot rely on the availability of an ALTO server, this results in no significant additional security risk.

Second, the discovery procedure may discover a sub-optimal or wrong ALTO server. Such an ALTO server may either not be able to provide information for a given resource consumer (e. g., behind a NAT), thus rendering the ALTO service useless. Alternatively, it may provide sub-optimal or forged information. In the latter case, attackers could try to use ALTO to affect the traffic distribution or the performance of applications. Users may then observe performance problems, and network operators could detect traffic anomalies. A potential counter-measure is to disable the use of the ALTO service.

Security issues of ALTO in general and potential solutions are also discussed in [I-D.ietf-alto-protocol].

7.2. For U-NAPTR

The address of an ALTO server is usually well-known within an access network; therefore, interception of messages does not introduce any specific concerns.

The primary attack against the methods described in this document is one that would lead to impersonation of an ALTO server since a device does not necessarily have a prior relationship with an ALTO server.

An attacker could attempt to compromise ALTO discovery at any of three stages:

1. providing a falsified domain name to be used as input to U-NAPTR
2. altering the DNS records used in U-NAPTR resolution
3. impersonation of the ALTO server

This document focuses on the U-NAPTR resolution process and hence

this section discusses the security considerations related to the DNS handling. The security aspects of obtaining the domain name that is used for input to the U-NAPTR process is described in respective documents, such as [RFC5986].

The domain name that is used to authenticated the ALTO server is the domain name in the URI that is the result of the U-NAPTR resolution. Therefore, if an attacker was able to modify or spoof any of the DNS records used in the DDDS resolution, this URI could be replaced by an invalid URI. The application of DNS security (DNSSEC) [RFC4033] provides a means to limit attacks that rely on modification of the DNS records used in U-NAPTR resolution. Security considerations specific to U-NAPTR are described in more detail in [RFC4848].

An "https:" URI is authenticated using the method described in Section 3.1 of [RFC2818]. The domain name used for this authentication is the domain name in the URI resulting from U-NAPTR resolution, not the input domain name as in [RFC3958]. Using the domain name in the URI is more compatible with existing HTTP client software, which authenticate servers based on the domain name in the URI.

An ALTO server that is identified by an "http:" URI cannot be authenticated. If an "http:" URI is the product of the ALTO discovery, this leaves devices vulnerable to several attacks. Lower layer protections, such as layer 2 traffic separation might be used to provide some guarantees.

8. Conclusion

This document describes a general ALTO server discovery process and discusses how the process can be applied in different deployment scenarios. The discovery process uses U-NAPTR resolution based on input information obtained either manually, by DHCP, or by PPP.

9. References

9.1. Normative References

- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.
- [RFC1332] McGregor, G., "The PPP Internet Protocol Control Protocol (IPCP)", RFC 1332, May 1992.
- [RFC1661] Simpson, W., "The Point-to-Point Protocol (PPP)", STD 51, RFC 1661, July 1994.
- [RFC1877] Cobb, S. and F. Baker, "PPP Internet Protocol Control Protocol Extensions for Name Server Addresses", RFC 1877, December 1995.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.
- [RFC3958] Daigle, L. and A. Newton, "Domain-Based Application Service Location Using SRV RRs and the Dynamic Delegation Discovery Service (DDDS)", RFC 3958, January 2005.
- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.
- [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.
- [RFC6418] Blanchet, M. and P. Seite, "Multiple Interfaces and Provisioning Domains Problem Statement", RFC 6418, November 2011.

9.2. Informative References

- [I-D.cheshire-nat-pmp]
Cheshire, S., "NAT Port Mapping Protocol (NAT-PMP)",
draft-cheshire-nat-pmp-03 (work in progress), April 2008.
- [I-D.ietf-alto-deployments]
Stiemerling, M. and S. Kiesel, "ALTO Deployment
Considerations", draft-ietf-alto-deployments-03 (work in
progress), November 2011.

- [I-D.ietf-alto-protocol]
Penno, R., Alimi, R., and Y. Yang, "ALTO Protocol",
draft-ietf-alto-protocol-10 (work in progress),
October 2011.
- [I-D.ietf-alto-reqs]
Kiesel, S., Previdi, S., Stiemerling, M., Woundy, R., and
Y. Yang, "Application-Layer Traffic Optimization (ALTO)
Requirements", draft-ietf-alto-reqs-11 (work in progress),
July 2011.
- [I-D.kist-alto-3pdisc]
Kiesel, S. and M. Stiemerling, "3rd Party ALTO Server
Discovery (3pdisc)", draft-kist-alto-3pdisc-00 (work in
progress), July 2012.
- [RFC4472] Durand, A., Ihren, J., and P. Savola, "Operational
Considerations and Issues with IPv6 DNS", RFC 4472,
April 2006.
- [RFC4848] Daigle, L., "Domain-Based Application Service Location
Using URIs and the Dynamic Delegation Discovery Service
(DDDS)", RFC 4848, April 2007.
- [RFC5693] Seedorf, J. and E. Burger, "Application-Layer Traffic
Optimization (ALTO) Problem Statement", RFC 5693,
October 2009.
- [RFC5986] Thomson, M. and J. Winterbottom, "Discovering the Local
Location Information Server (LIS)", RFC 5986,
September 2010.
- [UPnP-IGD-WANIPConnection1]
UPnP Forum, "Internet Gateway Device (IGD) Standardized
Device Control Protocol V 1.0: WANIPConnection:1 Service
Template Version 1.01 For UPnP Version 1.0", DCP 05-001,
November 2001.
- [bep24] Harrison, D., "Tracker Returns External IP",
BEP http://bittorrent.org/beps/bep_0024.html.

Appendix A. Contributors List and Acknowledgments

The initial version of this document was co-authored by Marco Tomsu <marco.tomsu@alcatel-lucent.com>.

Hannes Tschofenig provided the initial input to the U-NAPTR solution part. Hannes and Martin Thomson provided excellent feedback and input to the server discovery.

Olafur Gudmundsson provided an excellent DNS expert review on an earlier version of this document.

The authors would also like to thank the following persons for their contribution to this document or its predecessors: Richard Alimi, David Bryan, Roni Even, Gustavo Garcia, Jay Gu, Xingfeng Jiang, Enrico Marocco, Victor Pascual, Y. Richard Yang, Yu-Shun Wang, Yunfei Zhang, Ning Zong.

Nico Schwan is partially supported by the ENVISION project (<http://www.envision-project.org>), a research project supported by the European Commission under its 7th Framework Program (contract no. 248565). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the ENVISION project or the European Commission.

Michael Scharf is supported by the German-Lab project (<http://www.german-lab.de>) funded by the German Federal Ministry of Education and Research (BMBF).

Martin Stiemerling is partially supported by the COAST project (Content Aware Searching, retrieval and sTreaming, <http://www.coast-fp7.eu>), a research project supported by the European Commission under its 7th Framework Program (contract no. 248036). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the COAST project or the European Commission.

Authors' Addresses

Sebastian Kiesel
University of Stuttgart Computing Center
Allmandring 30
Stuttgart 70550
Germany

Email: ietf-alto@skiesel.de
URI: <http://www.rus.uni-stuttgart.de/nks/>

Martin Stiernerling
NEC Laboratories Europe
Kurfuerstenanlage 36
Heidelberg 69115
Germany

Phone: +49 6221 4342 113
Email: martin.stiernerling@neclab.eu
URI: <http://ietf.stiernerling.org>

Nico Schwan
Alcatel-Lucent Bell Labs
Lorenzstrasse 10
Stuttgart 70435
Germany

Email: nico.schwan@alcatel-lucent.com
URI: www.alcatel-lucent.com/bell-labs

Michael Scharf
Alcatel-Lucent Bell Labs
Lorenzstrasse 10
Stuttgart 70435
Germany

Email: michael.scharf@alcatel-lucent.com
URI: www.alcatel-lucent.com/bell-labs

Haibin Song
Huawei

Email: melodysong@huawei.com

Network Working Group
Internet Draft
Intended status: standard

Young Lee
Huawei
Greg Bernstein
Grotto Networking
Tae Sang Choi
ETRI
Sreekanth Madhavan
Huawei
Dhruv Dhody
Huawei

July 9, 2012

ALTO Extensions to Support Application and Network Resource
Information Exchange for High Bandwidth Applications

draft-lee-alto-app-net-info-exchange-00.txt

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on January 9, 2011.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Abstract

This draft proposes ALTO information model and protocol extensions to support application and network resource information exchange for high bandwidth applications in partially controlled and controlled environments as part of the infrastructure to application information exposure (i2aex) initiative.

Table of Contents

1. Introduction.....	3
2. Problem Statement.....	5
3. ALTO Constraints Filtering Extension Model.....	7
3.1. ALTO Query from Application Stratum to Network Stratum....	7
3.2. ALTO Response from Network Stratum to Application Stratum.	8
3.3. Information Model of ALTO Query from Application Stratum to Network Stratum.....	9
3.4. Information Model of ALTO Response from Network Stratum to Application Stratum.....	9
3.5. ALTO Protocol Extension for Constraints Filtering Mechanism	10
4. ALTO Protocol Extension for Graph Representation Mechanism....	11
4.1. Representing bandwidth constraints.....	11
5. Summary and Conclusion.....	12
6. Security Considerations.....	12
7. IANA Considerations.....	12
8. References.....	12
8.1. Informative References.....	12
Author's Addresses.....	13
Intellectual Property Statement.....	13
Disclaimer of Validity.....	14

1. Introduction

This draft proposes ALTO information model and protocol extensions to support application and network resource information exchange for high bandwidth applications in partially controlled and controlled environments as part of the infrastructure to application information exposure (i2aex) initiative. The Controlled and partially controlled ALTO environments referred to here are those where general access to a specific ALTO server may be restricted to a qualified list of clients.

This draft is build upon the previously introduced High Bandwidth Use Cases [HighBW]. In [HighBW], we have discussed two generic use cases that motivate the usefulness of general interfaces for cross stratum optimization in the network core. In our first use case, network resource usage became significant due to the aggregation of many individually unique client demands. In the second use case where data centers are sending large amount of data with each other, bandwidth usage was already significant enough to warrant the use of traffic engineered "express lanes" (e.g., private line service). We introduce third use case where inter-CDN providers may want to expose controlled network resource usage information so that CDN applications (e.g., request routing server) can utilize such information when appropriate decisions (e.g., request routing redirection) are needed.

These use cases result in optimization problems that trade off computational versus network costs and constraints. Both featured use cases show the usefulness of an ALTO interface between the application and network strata in optimizing the networked applications.

In particular, this draft introduces: (i) enhanced constraints filtering extensions to the ALTO protocol to reduce extraneous information transfer and enhance information hiding from the network's perspective; (ii) constrained cost graph mechanism encoding that enables enhanced application traffic optimization that was introduced by [HighBW].

In controlled and partially controlled environments in which operators are willing to share additional network stratum resource information such as bandwidth constraints or additional cost types of topology (e.g., graph or summary), it can be useful to reduce the amount of information transferred from the ALTO server to the ALTO client.

In considering information exchange between the application stratum and the network stratum, especially from the network stratum to the

application stratum, the degree of information details is one of the key concerns from the network providers' standpoint. On the one hand, the network information has to be useful to the application; on the other hand, the provided network information should hide details about the network. In order to achieve these desired goals, a simple enhancement to ALTO protocol would help significantly both in reducing/filtering the amount of information and in increasing the usefulness of the information from network to application.

Figure 1 shows ALTO Client-Server Architecture for Application-Network information Exchange. Figure 1 shows that ALTO Client in the application stratum can interface with ALTO Server in the network stratum. With this architecture, a simple ALTO query mechanism from application (via ALTO client) to network (via ALTO server) can be implemented. According to this architecture, ALTO Client is assumed to interact with the Application Orchestrator that has the knowledge of the end-user (i.e., source) application requirement, Data Center locations (i.e., destinations) and their resource information.

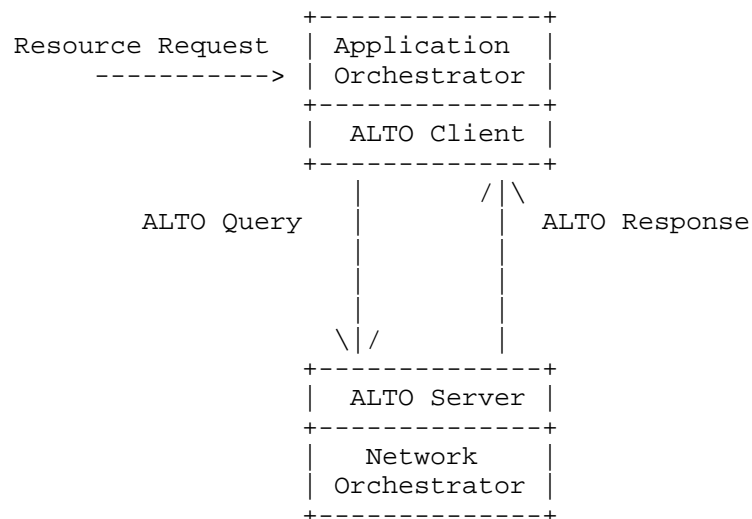


Figure 1 ALTO Client-Server Architecture for Application-Network information Exchange

The Application Orchestration functions depicted in Figure 1 interfacing data centers and end-users are out of the scope of this document. For simplicity purpose, Figure 1 doesn't depict the detailed relationship between ALTO client and server. In fact, both client and server don't need to be in the same administration boundary. It can be inter-operator and one to many relationships. For example, in the cases of inter-CDN environment or generic multi-domain environment, ALTO client represents a request routing server of upstream CDN operator and it interacts with multiple downstream CDN operators for their network resource information to make efficient decisions for desired quality CDN services. Interaction methods can either be iterative or recursive. That is, ALTO client can interact with multiple ALTO servers directly or ALTO client can interact with one representative ALTO server and subsequent interaction is done by the representative one to rest of ALTO servers.

The organization of this document is as follows. Section 2 discusses the ALTO architecture in the context of the application and network strata interaction. Section 3 provides ALTO Information model and protocol extension to support ALTO Constraints Filtering Mechanism. Section 4 provides ALTO information model and the protocol extension to support ALTO Constrained Cost Graph Mechanism.

2. Problem Statement

One critical issue in Application-Network information exchange in ALTO is the amount of information exchanged between the application and network strata. The information provided by network providers can be not so useful to the application stratum unless such information is abstracted into an appropriate level that the application stratum can understand.

In partially controlled and controlled environments, network providers can furnish appropriately abstracted and pruned information to the application stratum with the cooperation of the application stratum that can indicate some level of filtering and pruning in its query.

To reduce extraneous information this draft allows for "filtering" (or "pruning") of the following information in query/response of the ALTO pull model:

- . Topology Filtering - reduction of the results to only those specified set of source(s) and destination(s) instead of the entire network cost map. Note that this mechanism is not new in the current ALTO protocol. In the context of application-network information exchange, this topology filtering can be

of a tremendous help in reducing the amount of data exchanged between application and network.

- . Constraint Filtering on paths or graphs (e.g., bandwidth, latency, hop count, packet loss, etc.) - reduction of results to only those that meet ALTO client specified cost bounds.

As discussed in [HighBW], in a controlled environment optimization is significantly enhanced by sharing data related to bandwidth constraints and additional cost measures [MultiCost]. Such information may be considered sensitive to the network provider just as application data, e.g., usage, demand, etc., may be considered sensitive to an application provider. Section 3 provides ALTO information model and protocol extensions to support topology/constraints filtering mechanism.

While it is important to reduce and filter the information amount from network to application, for some applications that require stringent QoS objectives (e.g., bandwidth and latency), simple summary source-destination network resource information (i.e., summary form of topology) may not provide sufficient details to the application stratum. For example, suppose that a multiple number of large concurrent flows need to be scheduled from application to network. In such a case, a summary form of network topology that only shows source-destination bandwidth availability may not show the bottleneck links over which more than one flow may compete for the link bandwidth resource. This problem was indicated by [HighBW]. The following are the excerpts from [HighBW].

Consider the network shown in Figure 2, where DC indicates a datacenter, ER an end user region (as in the end user aggregation use case), N a switching node of some sort, and L a link. The link capacities and costs are also shown on the figure as well as a cost map between [ER1, ER2] and [DC1, DC2, DC3]. Since the network has a tree structure (very unusual but easier to draw in ASCII art), the cost map is unique.

As an illustration, assume that the maximum available capacity between any individual end region and a data center is 5 units (i.e., $L1=L2=L5=L6=5$). However, link L3 (capacity 8 units) represents a bottle neck to all the data centers (L3 is on all the paths to DC1, DC2, or DC3 from all end regions, ER1 and ER2).

ALTO Cost Map is shown in the lower right corner of Figure 2. This summary cost map does not provide enough details on the bottle necks. The lower left corner shows Link Capacity Cost, from which the bottle necks can be shown such that multi-flow commodity scheduling can be made possible to avoid such bottle necks.

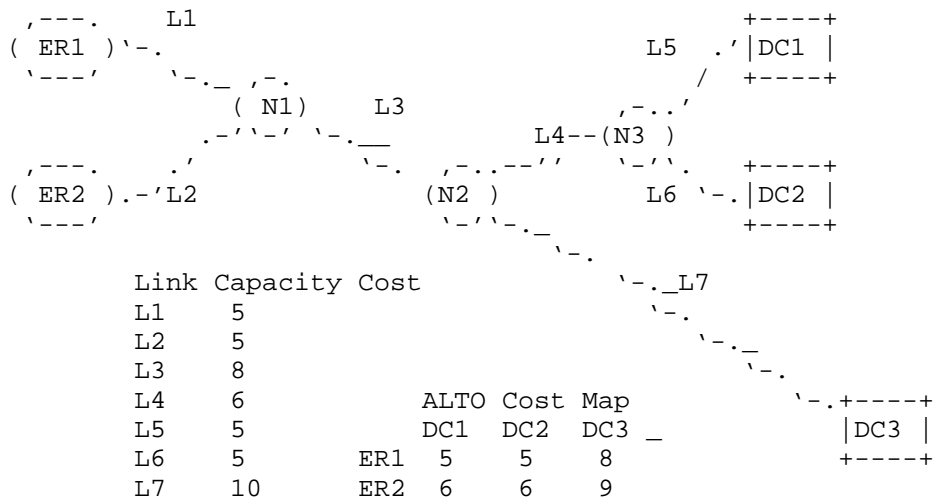


Figure 2. Example network illustrating bottlenecks

With the current ALTO cost map structure, the least cost path from ER1 would be either to DC1 or DC2. However, with the proposed capacitated cost map, the connection from ER1 to DC3 could be a better choice than the rest depending on the relative cost of network resources to data center resources.

A more general and relatively efficient alternative is to provide the requestor with a capacitated and multiply weighted graph that approximates and abstracts the capabilities of the network as seen by the source and destination location sets. This document provides ALTO information model and protocol extensions to support the graph model in Section 4.

3. ALTO Constraints Filtering Extension Model

3.1. ALTO Query from Application Stratum to Network Stratum

In order for the network stratum to provide its resource information, the application stratum needs to provide the End Point Cost Map to the network stratum. The End Point Cost Map should include the following information at a minimum:

- . End Point Source Address(es) /* these are the respective addresses of the nearest PE's to the user/client location */
- . End Point Destination Address(es) /* these are the respective addresses of the nearest PE's to a set of the candidate Data Center locations that can provide service to the user request. */

Note that how ALTO client derives the End Point Source/Destination addresses in terms of the nearest PE's is beyond the scope of this document.

- . Cost Type := {summary, graph} /* the cost map can be either a summary form or a graph form */
- . Constraints /* a set of constraints that apply to the requested path summary or graph for filtering. For instance, constraints can be the minimum bandwidth, maximum latency, maximum hop counts, maximum packet loss, etc. */
- . Parameters: /* a set of result parameters that each result (summary or a link in graph) should have. For instance, latency, cost, etc.)
- . Objective-function: The summary or the graph should be computed based on optimizing which parameter - IGP cost, latency, residual bandwidth, etc.

3.2. ALTO Response from Network Stratum to Application Stratum

In response to the ALTO Query from the Application Stratum, the Network Stratum needs to provide the filtered Cost Map Data of the feasible path found. The Filtered End Cost Map Data should include the following information at a minimum:

- . The list of feasible Source-Destination pair and its Cost Type
- . For each feasible S-D pair, indicate the following:
 - . Constraints Values /* indicate the actual values of each constraint requested */
- . Administration Domain ID /* For each network administration domain, the domain ID needs to be conveyed */

3.3. Information Model of ALTO Query from Application Stratum to Network Stratum

Alto query:

```
object {
  TypedEndpointAddr src;
  TypedEndpointAddr dsts<1..*>;
} EndpointFilterExt;
object {
  CostMode          cost-mode;
  CostType          cost-type;
  JSONString        constraints<0..*>;  [OPTIONAL]
  EndpointFilterExt endpoints;
} CsoReqEndpointCostMap;
```

3.4. Information Model of ALTO Response from Network Stratum to Application Stratum

Alto response:

```
object {
  JSONNumber hopcount;
  JSONNumber latency;
  JSONNumber pktloss;
} DstCostsConstraints;

object EndpointDstCosts {
  DstCostsConstraints[TypedEndpointAddr];  ...
};
object {
  EndpointDstCosts [TypedEndpointAddr]<0..*>;
  ...
} EndpointCostMapData;
object {
  CostMode          cost-mode;
  CostType          cost-type;
  EndpointCostMapData map;
} CsoInfoResourceEndpointCostMap;
```

3.5. ALTO Protocol Extension for Constraints Filtering Mechanism

This section provides the ALTO protocol extensions based on the information model discussed in Sections 3.3. and 3.4. The scenario provided in this section is that the ALTO client in the Application Stratum requests the summary cost map from the network with one source and three destinations.

In this particular example, the ALTO client requests the filtered summary of the network path subject to: bandwidth ≥ 20 , latency < 10 , hop count < 5 and packet loss < 0.03 .

The ALTO server provides the resulted network paths in summary.

```
POST /endpointcost/lookup HTTP/1.1
Host: alto.example.com
Content-Length: [TODO]
Content-Type: application/alto-csoendpointcostparams+json
Accept: application/alto-csoendpointsummary+json,application/alto-
error+json
{
  "cost-mode" : "ordinal",
  "cost-type" : "summary",
  "constraints": ["bw gt 20", "latency lt 10", "hopcount lt 5",
"pktloss lt 0.03"],
  "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: [TODO]
Content-Type: application/alto-csoendpointsummary+json
{
  "meta" : {},
  "data" : {
    "cost-mode" : "ordinal",
    "cost-type" : "summary",
    "map" : {
```

```

    "ipv4:192.0.2.2": {
    "ipv4:192.0.2.89" : [ "latency eq 5",
                        "hopcount eq 8", "pktloss eq 0.01" ],
    "ipv4:18.51.100.34" : [ "latency eq 9",
                        "hopcount eq 10", "pktloss eq 0.02" ],
    "ipv4:203.0.113.45" : [ "latency eq 40",
                        "hopcount eq 12", "pktloss eq 0.02" ]
    }
  }
}

```

4. ALTO Protocol Extension for Graph Representation Mechanism

4.1. Representing bandwidth constraints

```

object {
  LinkEntry [LinkName]<0..*>;
} CostConstraintGraphData;

object {
  PIDName:      a-end; // Node name at one side of the link
  PIDName:      z-end; // Node name at the other side of the link
  Weight:       wt;
  JSONNumber:   latency;
  Capacity:     r-cap; // Reservable capacity
} LinkEntry;

```

Where a link name is formatted like a PIDName (but names a link), and PID names are used for both provider defined location and provider defined internal model node identification. A graph representation of the network of Figure 2 might look like:

```

{
  "meta" : {},
  "data" : {
    "graph": {
      "L1": { "a-end": "ER1", "z-end": "N1", "wt": 1, "r-cap": 5 },
      "L2": { "a-end": "ER2", "z-end": "N1", "wt": 2, "r-cap": 5 },
      "L3": { "a-end": "N1", "z-end": "N2", "wt": 1, "r-cap": 8 },
      "L4": { "a-end": "N2", "z-end": "N3", "wt": 2, "r-cap": 6 },
    }
  }
}

```

```
"L5": {"a-end": "N3", "z-end": "DC1", "wt": 1, "r-cap": 5},
"L6": {"a-end": "N3", "z-end": "DC2", "wt": 1, "r-cap": 5},
"L7": {"a-end": "N2", "z-end": "DC3", "wt": 6, "r-cap": 10}
  }
}
```

5. Summary and Conclusion

TBD

6. Security Considerations

TBD

7. IANA Considerations

TBD

8. References

8.1. Informative References

- [HighBW] G. Bernstein and Y. Lee, "Use Cases for High Bandwidth Query and Control of Core Networks," draft-bernstein-alto-large-bandwidth-cases, work in progress.
- [MultiCost] S. Randriamasy, Ed., "Multi-Cost ALTO," draft-randriamasy-alto-multi-cost, work in progress.

Author's Addresses

Young Lee
Huawei Technologies
1700 Alma Drive, Suite 500
Plano, TX 75075
USA
Phone: (972) 509-5599
Email: ylee@huawei.com

Greg M. Bernstein
Grotto Networking
Fremont California, USA
Phone: (510) 573-2237
Email: gregb@grotto-networking.com

Sreekanth Madhavan
Huawei Technologies, India
Email: sreekanth.madhavan@huawei.com

Dhruv Dhody
Huawei Technologies, India
Email: dhruv.dhody@huawei.com

Tae-Sang Choi
ETRI
161 Gajong-Dong, Yusong-Gu
Daejeon, Republic of Korea
Phone: (8242) 860-5628
Email: choits@etri.re.kr

Intellectual Property Statement

The IETF Trust takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in any IETF Document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights.

Copies of Intellectual Property disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or

the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement any standard or specification contained in an IETF Document. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

All IETF Documents and the information contained therein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION THEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.

Network Working Group
Internet Draft
Intended status: standard

Young Lee
Huawei
Greg Bernstein
Grotto Networking
Sreekanth Madhavan
Huawei
Dhruv Dhody
Huawei
Taesang Choi
ETRI

July 8, 2012

ALTO Extensions for Collecting Data Center Resource Information

draft-lee-alto-ext-dc-resource-00.txt

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on January 8, 2011.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Abstract

In a networked application environment where resources (e.g., computing, storage, etc.) are geographically distributed in Data Centers, a key decision is to allocate the application request to an "optimal" Data Center location in which to host the application request. Key constraints in this decision include resource availability, network cost, infrastructure constraints (e.g., power) and others. This draft proposes an ALTO extension to support data center resource information model and its related protocol extensions as part of the infrastructure to application information exposure (i2aex) initiative.

Table of Contents

1. Introduction	2
2. Data Center Compute Resource Models	4
2.1. Current IaaS User Resource Models	4
2.1.1. Cost or Pricing Models	5
2.2. Internal Data Center Resource Models	6
3. ALTO-Interface Data Center Resource Information Model.....	6
4. ALTO Protocol Extension	7
4.1. Pull Based Query/Response	7
4.2. Push Based Query/Response	7
5. Security Considerations	8
6. IANA Considerations	8
7. References	8
7.1. Informative References	8
Author's Addresses	9
Intellectual Property Statement	9
Disclaimer of Validity	10

1. Introduction

In a networked application environment where resources (e.g., computing, storage, etc.) are geographically distributed in Data

Centers, a key decision is to allocate the application request to an "optimal" Data Center location in which to host the application request. Key constraints in this decision include resource availability (e.g., memory, storage, CPU, etc.), DC network cost, DC network resource constraints (e.g., bandwidth), structure constraints (e.g., Data Center power consumption) and others.

This draft describes data center resource information model in the context of i2aex (infrastructure to application exchange) and proposes its related ALTO protocol extensions as part of the infrastructure to application information exposure (i2aex) initiative.

The following figure depicts the key components in a networked application environment where Data Centers provide resources for an application.

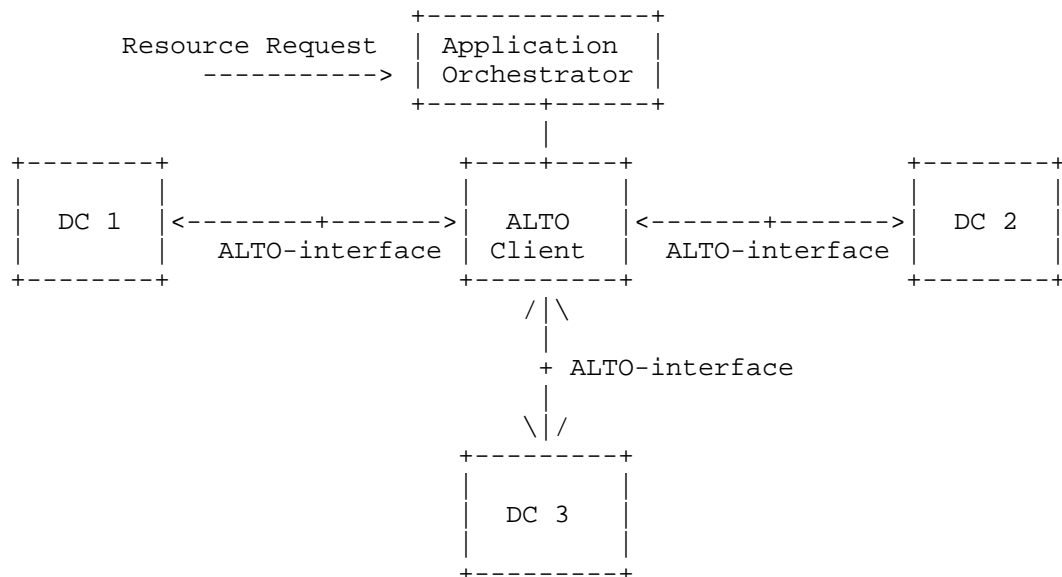


Figure 1 ALTO Architecture in Distributed Data Center Networks

Figure 1 shows that ALTO Client can establish an ALTO-interface to each data center (running ALTO server) to collect the abstracted

data center resource information and then select an "optimal" data center location based on the collected resource information for the resource request.

Resource request arrives to the "application orchestrator" which is a separate functional entity from the ALTO Client. The collected Data Center resource information by the ALTO Client needs to be sent to the "application orchestrator" where the optimal data center selection is made. How the application orchestrator interfaces with the ALTO Client is out of the scope of this document.

The potential information to be shared concerning capacity, performance, structure, and/or network costs associated with a data center may be considered sensitive to the data center owner/operator. It is assumed that ALTO client interfaces with data centers in a trusted relationship.

Combined compute and network resource optimization is of value to both application owners and data center operators. For example a data center operator with multiple buildings in a metropolitan area may also want to balance compute and network costs. When looking to model compute resources we consider both application owner and data center owner perspectives.

2. Data Center Compute Resource Models

2.1. Current IaaS User Resource Models

In a typical infrastructure as a service (IaaS) data center model, application software runs within one or more virtual machines (VMs). These VMs are allocated to physical hardware by IaaS scheduling software where they are run under the supervision of a virtualization hypervisor.

To achieve a given level of performance a particular VM within an application needs a specified amount of compute resources. The raw compute resources are (fast dynamic) memory, virtual CPUs (VCPUs), and dedicated local disk storage. One can think of a virtual CPU as an execution thread on a processor core, however, the notion maybe hypervisor specific. [Editor's note: we have currently only looked at details on the Xen hypervisor.]

Currently IaaS data center operators offer a fixed number of combinations of resources (memory, VCPUs, local disk) on which an application may run a VM. These are referred to as instance types.

[TBD: give examples of some instance types from Amazon or OpenStack.]

A scalable application is typically implemented so that it can run on a number of VMs with the number varying depending on load or other conditions. Different VMs may assume different roles in the application, e.g., a controller/dispatcher, request processor, data base engine, etc... These different roles may dictate the use of different instance types for the different VMs.

We are led to a model where an application will utilize a variable number of VMs over time. These VMs may play different roles within the application and hence require different combinations of processing resources.

The data center can furnish a limited number of instance types (combination of resources) for an application to use. In addition there is a finite amount of resources that the data center may have to offer a particular application.

Currently two different approaches appear to be used by IaaS providers: (a) Provide no information about available compute capacity and respond positively or negatively to requests for resources, i.e., a specified number of VM instances of a given type. (b) Provide overall quotas (limits) on memory, number of VCPUs, and local storage [OpenStackQuotas].

In this document, we consider the latter model. In such case, ALTO client will collect the overall data center resource information: memory, numbers of VCPUs, local storage, etc. This point will be elaborated in details in Section 2.2. This model allows application orchestrator to make better decision in optimizing geographically dispersed data center resources.

2.1.1. Cost or Pricing Models

IaaS service providers have introduced a number of pricing models. One provider [EC2Price] currently offers three different models based on the notions of (a) reserved instances, (b) on demand instances, and (c) spot instances.

For the more complicated models http based interfaces are given to facilitate queries and purchases. [TBD: Are there generic or parameterized pricing models that could represent a significant fraction of important cases?]

The pricing models discussed in this section are envisioned to be implemented by application orchestrator shown in Figure 1. As the user interacts with the application orchestrator and sends its

request, the application orchestrator can make decisions whether it should honor the request or not based on the collected information via the ALTO client interface. The information collection to the ALTO client from various data centers is the critical piece that facilitates this process of selecting the optimal data center.

2.2. Internal Data Center Resource Models

When previously discussing data center resources from an application perspective, the IaaS provider abstracts away the specifics of the hardware to a large degree. However when an IaaS provider is seeking to minimize its costs to provide services then the particulars of hardware resources are important: in particular, the cost of power at one site versus another site, the efficiency of physical hosts in delivering a given number of VCPUs and/or memory. Such information along with actual hardware capacity and usage can be used to weigh data center resource costs relative to bandwidth costs.

[TBD: compare Amazon's ECU (elastic compute unit) with VCPU notion or other hypervisor computation unit notions.]

3. ALTO-Interface Data Center Resource Information Model

ALTO Client collects a set of the abstracted resource information from each participating Data Centers. The following information is the list of Data Center resource abstract information that will give the ALTO Client a good level of abstracted view of the status of each participating Data Center.

This collected information will be used for the ALTO Client to find the data center where the requested resource can be provided (via an interface with the application orchestrator).

- . Data Center Identifier (DCI)
- . Data Center Location Identifier (e.g., IP address of the gateway node)
- . Time Stamp
- . Abstracted Memory Usage
- . Abstracted CPU Level
- . Abstracted Power Consumption Level
- . DC Network Cost
- . DC Network Resource Constraints

The list above is not an exhaustive list and can be expanded. How to represent DC Network Cost and DC Network Resource Constraints is yet to be determined. Note also that how to represent physical resources

information to abstract information is out of the scope of this document and is subject to further research.

4. ALTO Protocol Extension

4.1. Pull Based Query/Response

Pull based Query:

Based on GET URL /getdcinfo

Pull based response:

```
object
{
  VersionTag      vtag; [OPTIONAL]
  TypedEndpointAddrall addr;
  JSONNumber      srvload;
  JSONNUnber      ramusage;
  ...
} InfoDCProperty;
GET /dcinfo HTTP/1.1
Host: alto.example.com
Accept: application/alto-dcinfo+json
HTTP/1.1 200 OK
Content-Length: [TODO]
Content-Type: application/alto-dcinfo+json
{
  "meta" : {},
  "data" : {
    "vtag" : "1266506139",
    "addr" : "ipv4: 10.18.51.151:5060",
    "srvload": 25,
    "ramusage": 60
  }
}
```

4.2. Push Based Query/Response

Push based Query:

```
object
{
```

```
VersionTag      vtag; [OPTIONAL]
TypedEndpointAddrall addr;
JSONNumber      srvload;
JSONNNUmber      ramusage;
...
} InfoDCProperty;
Push based response:
200 OK with body NUL
POST /dcinfo HTTP/1.1
Host: alto.example.com
Content-Length: [TODO]
Content-Type: application/alto-dcinfo+json
{
  "meta" : {},
  "data" : {
    "vtag" : "1266506139",
    "addr" : "ipv4: 10.18.51.151:5060",
    "srvload": 25,
    "ramusage": 60
  }
}
HTTP/1.1 200 OK
Host: alto.example.com
```

5. Security Considerations

TBD

6. IANA Considerations

TBD

7. References

7.1. Informative References

Author's Addresses

Young Lee
Huawei Technologies
1700 Alma Drive, Suite 500
Plano, TX 75075
USA
Phone: (972) 509-5599
Email: ylee@huawei.com

Greg M. Bernstein
Grotto Networking
Fremont California, USA
Phone: (510) 573-2237
Email: gregb@grotto-networking.com

Sreekanth Madhavan
Huawei Technologies, India
Email: sreekanth.madhavan@huawei.com

Dhruv Dhody
Huawei Technologies, India
Email: dhruv.dhody@huawei.com

Tae-Sang Choi
ETRI
161 Gajong-Dong, Yusong-Gu
Daejeon, Republic of Korea
Phone: (8242) 860-5628
Email: choits@etri.re.kr

Contributor's Addresses

Nandiraju Ravi Shankar
Huawei Technologies, India
Email: nandiraju.shankar@huawei.com

Intellectual Property Statement

The IETF Trust takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be

claimed to pertain to the implementation or use of the technology described in any IETF Document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights.

Copies of Intellectual Property disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement any standard or specification contained in an IETF Document. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

All IETF Documents and the information contained therein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION THEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: January 17, 2013

E. Marocco
Telecom Italia
J. Seedorf
NEC
July 16, 2012

WebSocket-based server-to-client notifications for the Application-Layer
Traffic Optimization (ALTO) Protocol
draft-marocco-alto-ws-01

Abstract

The Application-Layer Traffic Optimization (ALTO) protocol is designed to allow entities with knowledge about the network infrastructure to export such information to applications that need to choose one or more endpoints to connect to among large sets of logically equivalent ones. The base protocol specification adopts a simple pull-based model, according to which the client retrieves the information encoded as JSON objects over HTTP directly from the server.

This document proposes (for discussion) a mechanism for providing server-initiated information update notifications through a WebSocket-based ALTO protocol extension that easily integrates in the basic protocol model.

Status of this Memo

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

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

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

This Internet-Draft will expire on January 17, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
1.1. Requirements Language	4
2. Overview of operations	4
3. Information Resource Directory (IRD) Extensions	5
3.1. Example	5
4. Client-to-server Version Indication	6
5. Partial Updates Encoding	7
6. Example	7
7. Security Considerations	7
8. Conclusion	7
9. References	7
9.1. Normative References	7
9.2. Informative References	7
Authors' Addresses	8

1. Introduction

The Application-Layer Traffic Optimization (ALTO) protocol [I-D.ietf-alto-protocol] is designed to allow entities with knowledge about the network infrastructure to export such information to applications that need to choose one or more endpoints to connect to among large sets of logically equivalent ones. The base protocol specification adopts a simple pull-based model, according to which the client retrieves the information encoded as JSON objects over HTTP directly from the server.

Such a pull-based model is well suited for use cases where the information does not change frequently, e.g. when it represents network and cost maps intended to provide a hint to peer-to-peer applications that have to perform initial peer selection (i.e. the primary use case that motivated the specification of the ALTO protocol). However, over the years several similar use cases have emerged, most of them with more stringent requirements in terms of information freshness. Those use cases could also simply and effectively be addressed by the ALTO protocol, provided it features a mechanism for clients to receive server-initiated information update notifications. This document proposes (for discussion) a mechanism for providing such notifications through a WebSocket-based [RFC6455] extension that easily integrates in the basic ALTO protocol model.

The WebSocket protocol is only one option such an extension could be based on. Many alternatives can of course be considered, based on virtually any bi-directional protocol that provides some sort of publish/subscribe framework. Among others, XMPP, BGP and SNMP have been proposed and to some extent discussed in different contexts as a basis for providing similar features. The strong points of the WebSocket protocol in this context -- and thus the reason why the extension proposed here is based on it -- include:

- o WebSocket is explicitly intended to provide bi-directionality to HTTP, the transport the ALTO protocol is based on. The main implication of this fact is that both the HTTP client and server libraries/frameworks that ALTO implementations are based on will natively support it (or, since the technology is very new, soon will);
- o a resource representing an update notification service related to a particular resource instance made available by an ALTO server can simply be identified by a WebSocket URI and advertized in the Information Resource Directory (IRD) just as any other regular resource;

- o a resource representing an update notification service can be unambiguously defined through a MIME type, just as any other regular resource;
- o reuse of HTTP authentication.
- o [TODO: Is Origin-based security of any use here?]

The most appropriate way for encoding partial updates of ALTO information is an open issue itself, at the time of writing, discussed in [I-D.schwan-alto-incr-updates].

1.1. Requirements Language

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

2. Overview of operations

When an ALTO client wants to retrieve a particular piece of information made available by an ALTO server and then receive notifications about each subsequent change, it achieves that in the following steps:

1. retrieve the IRD of the ALTO service it is going to access;
2. find in the IRD the URI of the resource it is interested in, identifying it through the associated content type (e.g. application/alto-networkmap+json);
3. retrieve a copy of the resource it is interested in;
4. find in the IRD the WebSocket URI of the update notification service associated to the specific resource just retrieved;
5. establish a WebSocket connection against the URI of the update notification service;
6. indicate the version tag of the retrieved resource to the server;
7. process each subsequent updates received on the WebSocket connection in order to keep the local representation of the resource up-to-date.

Steps 1 to 3 are regular ALTO operations, as defined in [I-D.ietf-alto-protocol]. The following section will discuss (and at

some point hopefully define) the missing pieces of specification needed for performing the remaining steps, namely:

1. a mechanism for identifying the WebSocket URI of the update notification service associated to a particular resource;
2. a mechanism for the client to tell the server the version of the resource stored locally;
3. a mechanism for encoding information updates.

The mechanism discussed here is intended to allow steps from 4 to 7 to be executed at an arbitrarily later stage in respect to steps 1-3 (i.e. the mechanism needs to be able to update arbitrarily stale resource representations).

3. Information Resource Directory (IRD) Extensions

[NOTE: strawman proposal.]

This document specifies the additional optional "updates" property for top-level IRD entries. The new property is specifically defined as:

updates A WebSocket URI as defined in [RFC6455] at which the ALTO server provides dynamic updates of the corresponding resource.

3.1. Example

The following is an example Information Resource Directory returned by an ALTO Server. In this example, the ALTO Server provides both a network map and a cost map with corresponding update notification services.

```
{
  "resources" : [
    .
    .
    .
    {
      "uri" : "http://alto.example.com/networkmap",
      "media-types" : [ "application/alto-networkmap+json" ],
      "updates" : "ws://alto.example.com/networkmap"
    }, {
      "uri" : "http://alto.example.com/costmap/num/routingcost",
      "media-types" : [ "application/alto-costmap+json" ],
      "capabilities" : {
        "cost-modes" : [ "numerical" ],
        "cost-types" : [ "routingcost" ]
      },
      "updates" : "ws://alto.example.com/costmap/num/routingcost"
    }
  ]
}
```

4. Client-to-server Version Indication

As discussed in [I-D.schwan-alto-incr-updates], indication of the version of the locally stored resource can happen in two ways:

- o after the WebSocket connection has been established, with an ad-hoc client-to-server signalling message such as:

```
{"reference-tag": "1266506140"}
```

The main drawback of such an approach consists with the added complexity both on the client side and on the server side (e.g. for handling error conditions, race conditions, etc.);

- o in the WebSocket connection initiating GET request, by means of a HTTP header field such as If-Modified-Since or If-None-Match. The advantage of such an approach consists of the fact that the data on the WebSocket connection flows on the server-to-client direction only, adding no additional complexity to a client already able to process partial updates. The drawback is that none of the existing HTTP headers seem to have the exact required semantic.

5. Partial Updates Encoding

Possible encoding options for partial updates are discussed in [I-D.schwan-alto-incr-updates], for the case of client-initiated transactions. The very same considerations also apply to the case of server-initiated notifications. It seems therefore straightforward to assume that the same encoding will be adopted here.

6. Example

[TODO: Illustrate a full example, from the IRD, the retrieval of the resource and the establishment of the WS connection.]

7. Security Considerations

[TODO: A lot to be said here.]

8. Conclusion

This document discusses an extension to the ALTO protocol to allow for server-initiated information update notifications. Specifically, a WebSocket-based ALTO protocol extension is proposed that easily integrates in the basic protocol model.

9. References

9.1. Normative References

- [I-D.ietf-alto-protocol]
Alimi, R., Penno, R., and Y. Yang, "ALTO Protocol",
draft-ietf-alto-protocol-12 (work in progress), July 2012.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC6455] Fette, I. and A. Melnikov, "The WebSocket Protocol",
RFC 6455, December 2011.

9.2. Informative References

- [I-D.schwan-alto-incr-updates]
Schwan, N. and B. Roome, "ALTO Incremental Updates",
draft-schwan-alto-incr-updates-01 (work in progress),
March 2012.

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

Authors' Addresses

Enrico Marocco
Telecom Italia
Via Reiss Romoli, 274
Torino, 10148
Italy

Phone:
Email: enrico.marocco@telecomitalia.it

Jan Seedorf
NEC Laboratories Europe, NEC Europe Ltd.
Kurfuersten-Anlage 36
Heidelberg, 69115
Germany

Phone: +49 (0) 6221 4342 221
Email: jan.seedorf@neclab.eu
URI: <http://www.neclab.eu>

ALTO
Internet-Draft
Intended status: Standards Track
Expires: January 17, 2013

N. Schwan
W. Roome
Alcatel-Lucent Bell Labs
July 16, 2012

ALTO Incremental Updates
draft-schwan-alto-incr-updates-02

Abstract

The goal of Application-Layer Traffic Optimization (ALTO) is to bridge the gap between network and applications by provisioning network related information. This allows applications to make informed decisions, for example when selecting a target host from a set of candidates.

Therefore an ALTO server provides network and cost maps to its clients. This draft discusses options on how to provide incremental updates for these maps, with the goal of reducing the amount of data needed for transmitting the maps and shortly evaluates the two most promising options.

Requirements Language

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

Status of this Memo

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

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

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

This Internet-Draft will expire on January 17, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	4
2. Problem Statement	6
3. Determine Client Map Version	7
3.1. HTTP	7
3.1.1. If-Modified-Since HTTP Header	7
3.1.2. If-None-Match HTTP Header	9
3.2. Version-based incremental updates as ALTO extension	10
3.2.1. CURRENT NETWORK MAP vtag	10
3.2.2. Extensions to full cost-map response:	11
4. Incremental Update Options	12
4.1. Send entire map	12
4.2. Patch map	12
4.3. Encode map	12
4.4. HTTP Range Retrieval	12
4.5. JSON patch	13
4.6. ALTO Incremental Update service	14
4.6.1. Incremental Update messages	14
4.6.2. Incremental Update Capability	16
5. Numerical Evaluation	18
5.1. Full Cost Map Size estimation	18
5.2. JSON Patch vs. ALTO extension	19
6. IANA Considerations	21
7. Security Considerations	22
8. Conclusion	23
9. References	24
Appendix A. Acknowledgments	25
Authors' Addresses	26

1. Introduction

The goal of Application-Layer Traffic Optimization (ALTO) is to bridge the gap between network and applications by provisioning network related information. This allows applications to make informed decisions, for example when selecting a target host from a set of candidates. Typical applications are file sharing, real-time communication and live streaming peer-to-peer networks [RFC5693] as well as Content Distribution Networks [I-D.jenkins-alto-cdn-use-cases].

The ALTO protocol [I-D.ietf-alto-protocol] is specified as a client-server protocol based on the HyperText Transfer Protocol (HTTP) and encoded in JavaScript Object Notation (JSON). An ALTO server provides services that guide ALTO clients in their decisions. The Endpoint Property Service allows ALTO clients to look up properties of endpoints, for example its Network Location. The Endpoint Cost Service allows ALTO server to rank endpoints amongst each other with respect to numerical or ordinal costs. The Map Service and the Map Filtering Service allows ALTO client to retrieve full or partial Network Maps and the associated Cost Maps that are provisioned by an ALTO server.

The ALTO Network Map contains groupings of endpoints as defined by the ALTO server. By aggregating multiple endpoints that are close to one another with respect to their network connectivity a greater scalability is achieved. Each group of endpoints is associated to a Network Location identifier called a PID, for example by a list of IP prefixes that belong to the PID. The ALTO Server then indicates preferences amongst the PIDs in the Cost Map by defining Path Costs amongst sets of Network Locations.

The size of the Network and Cost Maps depend on the granularity of the map an ALTO server provides for its clients. While some use cases allow operators to configure their servers to support only a small numbers of PIDs, other use cases are expected to require a much greater accuracy in terms of network locations. In order to avoid the transmission of the same information in each client request, a mechanism that allows a server to send incremental updates, in particular for large Network and Cost Maps, is needed.

The goal of this draft is to list and discuss the different options that allow such incremental updates of Network and Cost Maps. It is focused on options that are compatible with the ALTO base protocol and encoding, namely HTTP and JSON. The draft is structured as follows: First it lists options that allow a server to validate the cached version of a map a client has. Then it discusses several options a server has to send incremental updates, including JSON

Patch and an ALTO extension. Finally it shortly evaluates two promising options.

Comments and discussions about this memo should be directed to the ALTO working group: alto@ietf.org.

2. Problem Statement

The ALTO protocol uses Network and Cost Maps to allow ALTO servers the specification of its own aggregated network view. Essentially the Network Map contains information on how the endpoints are grouped together, which is typically done according to their proximity. The Cost Map contains Path Costs between the network regions defined in the Network Map. The size of these maps strongly depends on the scenario an ALTO server is configured for by its operator. While in some scenarios both maps might only comprise a small number of PIDs, others need much greater accuracy. For large maps partial updates might become necessary.

Both map types have slightly different characteristics. Network Maps in general are expected to be smaller than Cost Maps. As an example, a Network Map with 5,000 PIDs, each having 10 cidrs will result in a map with the size of roughly 1.25 megabytes. A Cost Map in contrast contains a $n \times n$ matrix for cost entries where n is the number of PIDs. Even for short PID names a full cost map for 5,000 PIDs takes up to 417 megabytes. Network Maps are also seen to be less dynamic than Cost Maps. This is due to the fact that the topology an ALTO server sees changes slower than the path costs of the network. Another characteristic is that changes to the Network Map will impact the Cost Map, whereas vice versa this is presumably not the case. A final discussion on whether partial updates are useful for both map types is out of the scope of this document.

The remainder of this document discusses options to allow partial updates of Network and Cost Maps. Therefore two sections focus on two separate problems that need a solution. The first part (Section 3) discusses how an ALTO client and an ALTO server can synchronize their map state without transmitting the whole map. This is needed to identify whether a partial update can be applied and also to calculate the partial update itself. The second part (Section 4) of the document discusses how partial updates can be encoded and sent to the client. The final section gives a numerical evaluation of proposed incremental update options.

3. Determine Client Map Version

To allow a server sending incremental updates to a client it first needs to check the version of a cached map a client already has. In this section we discuss options for this validation. We focus our discussion on approaches where the client polls the ALTO server for map changes and the server decides based on the client request. Therefore we first discuss HTTP built-in options and follow-up with a possible extension to ALTO network and cost maps themselves.

3.1. HTTP

HTTP [RFC2616] provides request-header fields to express conditional requests. Typically these conditional requests are used by caches to decide whether a copy of a resource they have can be served to a requesting client directly or not. Responses to conditional HTTP requests must be exactly the same as for normal HTTP GET requests. Thus sending a modified map version (i.e. a partial update) violates the HTTP standard. Conditional requests can still be used to avoid transmitting an unchanged Map multiple times. The options are discussed in the following.

3.1.1. If-Modified-Since HTTP Header

One possible option is to use the HTTP If-Modified-Since header in the request if the client has previously contacted the ALTO service and thus already has a version of the map. Therefore the server includes date and time when the map was last modified into the Last-Modified entity-header for a particular map, which is cached by the client together with the map and sends it in new requests for the same map in the If-Modified-Since header.

The following figure illustrates a GET request for a Network Map Information Resource. Additionally the client indicates the time when it retrieved the Network Map the last time in the If-Modified-Since header field.

```
GET /networkmap HTTP/1.1
Host: alto.example.com
Accept: application/alto-networkmap+json,application/alto-error+json
If-Modified-Since: Sat, 24 Dec 2011 19:43:31 GMT
```

Figure 1: If-Modified-Since HTTP Header

A server retrieving this request uses the timestamp provided by the client to decide whether to send a full map or no map at all in case there were no changes. In case the Network Map has not been modified since the time provided by the client in the request, the server

SHOULD reply with a 304 HTTP response. The client then caches the updated value of the Last-Modified entity-header for future requests.

```
304 Not Modified
Last-Modified: Sun, 25 Dec 2011 09:33:31 GMT
```

Figure 2: HTTP Response 304

In case the server determines that the timestamp provided by the client is out-of-date it must return the full Network Map, as defined in the ALTO core protocol specification. The following figure shows the Last-Modified entity-header in the HTTP response that is used as a timestamp for the map as well as the ETag header for a If-None-Match request, as explained in section Section 3.1.2.

```
HTTP/1.1 200 OK
Content-Length: [TODO]
Content-Type: application/alto-networkmap+json
Date: Sun, 25 Dec 2011 09:33:31 GMT
ETag: "nm000012"
```

```
{
  "meta" : {},
  "data" : {
    "map-vtag" : "1266506139",
    "map" : {
      "PID1" : {
        "ipv4" : [
          "192.0.2.0/24",
          "198.51.100.0/25"
        ]
      },
      "PID2" : {
        "ipv4" : [
          "198.51.100.128/25"
        ]
      },
      "PID3" : {
        "ipv4" : [
          "0.0.0.0/0"
        ],
        "ipv6" : [
          "::/0"
        ]
      }
    }
  }
}
```

Figure 3: Full Network Map HTTP Response

3.1.2. If-None-Match HTTP Header

A second HTTP based mechanism could employ ETags in combination with the If-None-Match header. Here the server creates entity tags that identify the version of a map. A client that caches a map includes this identifier in every future request. The server can use this ETag to identify whether a cached map version is up-to-date.

The following example illustrates a client GET request which includes an ETag, identifying a network map. The example assumes the client received the Network Map response in Figure 3.

```
GET /networkmap HTTP/1.1
Host: alto.example.com
Accept: application/alto-networkmap+json,application/alto-error+json
If-None-Match: "nm000012"
```

Figure 4: If-None-Match HTTP Header

3.2. Version-based incremental updates as ALTO extension

With this approach, clients poll the ALTO server for changes. The server provides hints as to the polling frequency. We propose two different mechanisms in the ALTO message format, one for network-map changes, the other for cost-map changes.

For network-map changes, add a new GET-mode request, "CURRENT NETWORK MAP VTAG." The response is short and simple: just the current map-vtag and a hint about how often the network-map might change. Once a client has the full network map, the client periodically sends that CURRENT VTAG request to the server. If the map-vtag changes, the client re-gets the network map. For cost-map changes, add two new fields to the full cost-map response: a "cost-map-vtag" and a hint about the how often the server updates the cost map.

Using these vtags both client and server can determine if it is necessary to request or to send an updated map, a full map, or if the current version is still up-to-date.

3.2.1. CURRENT NETWORK MAP vtag

This is a GET-mode request. The response is a simple json structure with

- o The current map-vtag for the network map.
- o The average number of seconds between changes to the network map.

It needs a new media type, say application/alto-currentmapvtag+json

For example,

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

{
  "meta" : {},
  "data" : {
    "map-vtag" : "123456",
    "update-interval" : 86400
  }
}
```

Figure 5: CURRENT NETWORK MAP vtag

3.2.2. Extensions to full cost-map response:

Add two new fields to the costmap response, as in:

```
object {
  CostMode      cost-mode;
  CostType      cost-type;
  VersionTag    map-vtag;
  VersionTag    cost-map-vtag;    // Optional
  JSONNumber    update-interval;  // Optional
  CostMapData   map;
} InfoResourceCostMap;
```

Figure 6: Extensions to full cost-map response

cost-map-vtag: A string that (together with the network map-vtag) uniquely identifies this version of the cost map.

update-interval: Average time between cost-map updates, in seconds. (A hint, not a guarantee). Perhaps required if cost-map-vtag is present

These fields would only be in the full cost map response, not in a filtered cost map response.

4. Incremental Update Options

Once a server has decided to send a partial update, there are several ways to do so. This section discusses these response options.

4.1. Send entire map

One trivial option is always to send the entire map anyways. The advantage of sending the whole map typically is that there is no computational effort needed on the server side. Thus this can always be a fallback in case the server is under load, or in case a partial update appears to be inefficient.

4.2. Patch map

A server that knows the version of the map a client currently has can use this information to calculate the contextual diff to the newest version of the map. This can also be done in a batch process for all previous versions once a new map is loaded on the server to avoid a per request calculation. The diff output can then be sent in a response to the client, which in turn can use it to patch its version of the map. By doing this the newest version of the map can be recreated.

4.3. Encode map

One major goal of applying partial updates is to reduce transmission time by reducing the amount of data which is to be transferred to the client. This goal can be achieved by applying compression techniques, such as gzip, to the message content, both for partial updates as well as for entire maps.

HTTP supports this by the Content-Encoding entity header field. The advantage of using compression is that there is no need to change the underlying media-type of the response. Typically not all ALTO clients will support this optimization from the beginning, thus the server will need to store two representations of the maps: One which is compressed and one uncompressed.

4.4. HTTP Range Retrieval

The HTTP Range header allows a client to request only a subset of a resource. This is useful for quick recovery of partially failed transfers. Using this option for incremental updates of ALTO Maps difficult. The Byte range is specified by the client, however the diff is calculated by the server. Thus an additional mechanism would be needed tell the client which byte sequences to request to. Additionally the Byte offsets might change between Map versions, thus

this option appears to be error-prone.

4.5. JSON patch

JSON Patch [I-D.pbryan-json-patch] defines a JSON document structure that allows partial modifications to a JSON document and defines the associated media type "application/json-patch". Therefore JSON Patch is a suitable option for incremental updates of the Network and Cost Maps. JSON patch supports add, remove and replace operations that can be used in combination with JSON Pointers [I-D.pbryan-zyp-json-pointer] to modify values and arrays of the JSON document members.

Typically JSON Patch is used in combination with the HTTP PATCH method [RFC5789] to partially modify existing resources on a server. As an ALTO client is not modifying a resource, but wants to be updated if the resource has changed it needs to signal to the server that it is able to receive and understand JSON Patch updates. This can be done by including the media type "application/json-patch" in the Accept header field of the HTTP request.

Although JSON Patch permits pointers to index individual array elements, that's potentially ambiguous. "Add" and "delete" change the array indexes; do subsequent updates to that array refer to the original indexes or the revised indexes? And even if that's well-specified, it's a potential source of error. Furthermore, some clients may store the CIDRs in a PID as a set, rather than an array, so they don't keep the original index numbers.

To avoid these problems, we recommend that when updating the CIDRs for a PID, the server replaces the full array value for that PID, rather than updating individual CIDRs by index.

This may also simplify the server, because it just has to flag the PID as having changed; it doesn't have to remember the previous sequence of CIDRs.

The following figure illustrates one example where the server decides to send a partial update to the client using JSON Patch. The server indicates this in the response Content-Type header. In the following example the Network Map from the example in Figure 3 above has changed. The map-vtag element has been incremented by 1, which results in a replace operation for the respective element containing the new value. Also two new subnets are added to the Network Map in PID1 and PID2 by two replace operations on the ipv4 arrays.

```
HTTP/1.1 200 OK
Content-Length: [TODO]
Content-Type: application/json-patch
Date: Sun, 25 Dec 2011 09:33:31 GMT

{ "replace": "/data/map-vtag", "value": "1266506140" },
{ "replace": "/data/map/PID1/ipv4", "value":
  ["192.0.2.0/24", "198.51.200.0/25", "198.51.100.0/25"] },
{ "replace": "/data/map/PID2/ipv4", "value":
  ["198.51.100.128/25", "198.51.200.128/25"] }
```

Figure 7: Partial update with JSON Patch

One benefit of using JSON Patch for partial updates would be if standard JSON parsers would implement JSON Patch already. This would allow ALTO protocol implementers to reuse existing functionality. However to the knowledge of the authors there are only few implementations supporting JSON Patch to date, and a widespread adoption is still outstanding and will presumably take some time.

4.6. ALTO Incremental Update service

Another option is to offer a dedicated ALTO service for partial updates. A client that determines that its current map is out-of-date, for example by comparing cost-map-vtag values (see Section 3.2) can then query this service to retrieve the partial update. This section defines this service in the next section and a new capability for the IRD in the following section

4.6.1. Incremental Update messages

This service can be implemented in new POST-mode requests, Get Network Map Updates and Get Cost Map Updates. These partial update messages use a new MIME type [RFC2046]:

```
"application/alto-update-param+json"
```

The client includes its current map version (i.e. the map-vtag or the cost-map-vtag) to the post data of the request in a new reference tag field. Based on this reference-tag a partial update response is created. Or if the current (cost-)map-vtag is the same as the reference-tag -- that is, if there are no changes -- the "map" entry in the response has no entries. Or if it's sufficiently old that the server no longer knows what changed since that version or if the tag is invalid, the server returns a complete cost map. Thus the response contains an optional field to allow clients to distinguish:

```
{"full-map": true}
```

Thus the response MUST have costs that changed since the specified version, but MAY have other costs as well.

The partial update response essentially is a Filtered Network Map or Filtered Cost Map message, where for a Network Map for each PID in the message, previous CIDRs are replaced with new CIDRs in case they have changed. To remove a PID from the Map the value "delete" or an empty array is used. PIDs that are not in the message remain unchanged. Similarly for Cost Maps, costs specified in the message replace previous costs for the respective source/destination PIDs. Again, to remove a cost the value "delete" (or "-1", or "NaN",...) is used whereas costs that are not in the message stay the same.

The following figures give one example of a request/response transaction of the proposed Partial Update ALTO extension service.

```
POST /partialupdate/costmap/rcost/incrementalupdate/costmap
Content-Type: application/alto-update-param+json
Accept: application/alto-costmap+json
{ "reference-tag": "1266506140" }
```

Figure 8: Incremental update request

```
HTTP/1.1 200 OK
Content-Type: application/alto-costmap+json
{ "meta": {},
  "data": {
    "cost-mode": "numerical",
    "cost-type": "routingcost",
    "map-vtag": "314159",
    "cost-vtag": "1266506141",
    "full-map": false,
    "map": { "PID1": { "PID2": 1, "PID3": 2 } }
  }
}
```

Figure 9: Incremental update response

4.6.2. Incremental Update Capability

An ALTO server needs to advertise its ability of providing incremental updates to a client. One option of doing this is by including an ALTO information resource capability indicating the partial update option. This can be done per ALTO service, which allows a fine grained control over which URIs handle partial requests.

The following Information Resource Directory illustrates one example where network and costmaps are available with and without partial update option respectively. This is expressed by the JSONBool capability element "partial-update".

```
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: [TODO]
Content-Type: application/alto-directory+json

{
  "resources" : [
    {
      "uri" : "http://alto.example.com/serverinfo",
      "media-types" : [ "application/alto-serverinfo+json" ]
    }, {
      "uri" : "http://alto.example.com/networkmap",
      "media-types" : [ "application/alto-networkmap+json" ]
    }, {
      "uri" : "http://alto.example.com/costmap/rcost",
      "media-types" : [ "application/alto-costmap+json" ],
      "capabilities" : {
        "cost-modes" : [ "numerical" ],
        "cost-types" : [ "routingcost" ]
      }
    }, {
      "uri" : "http://alto.example.com/partialupdate/networkmap",
      "media-types" : [ "application/alto-networkmap+json" ],
      "accepts" : [ "application/alto-update-param+json" ],
      "capabilities" : {
        "incremental-update" : true,
      }
    }, {
      "uri" : "http://alto.example.com/partialupdate/costmap/rcost",
      "media-types" : [ "application/alto-costmap+json" ],
      "accepts" : [ "application/alto-update-param+json" ],
      "capabilities" : {
        "incremental-update" : true,
        "cost-modes" : [ "numerical" ],
        "cost-types" : [ "routingcost" ]
      }
    }
  ]
}
```

Figure 10: IRD With Partial Update Capability

5. Numerical Evaluation

In this section we provide a numerical evaluation of the efficiency of incremental updates. We focus on the two most promising approaches JSON patch (Section 4.5) and the ALTO incremental Updates service (Section 4.6). For our calculations we use cost map formats to conclude on the incremental update efficiency.

5.1. Full Cost Map Size estimation

In the following we estimate the size of full cost maps with respects to the number of PIDs that are contained. Neglecting some fixed overhead we find:

$$np^2 * (4 + pl + cl) + np * (6 + pl)$$

where

np: number of PIDS

pl: Average length of PID names

cl: Average number of digits in costs

If we assume PID names are 8 characters (pl = 8) and costs are 3 digits (cl = 3), we get:

np	cost-map size
10	1,640
25	9,725
50	38,200
100	151,400
250	941,000
500	3,757,000
1000	15,014,000
2500	93,785,000
5000	375,070,000

Figure 11: Full Cost Map Sizes

As a conclusion we do think that for less than 100 PIDs incremental updates are not needed. Instead the Full Cost Map can be sent. However for Maps that require a greater accuracy and thus a higher number of PIDs incremental updates are required.

5.2. JSON Patch vs. ALTO extension

In this section we estimate the performance of JSON Patch and the proposed ALTO extension. We limit our estimation on the worst-case scenario, which are replace operations.

For each changed cost a JSON Patch replace operation is represented by the following encoding:

```
{ "replace": "meta.data.map.SRC-PID.DEST-PID", "value": 123 },
```

Whereas an Incremental Update ALTO Cost Map message as defined in this document takes an encoding of (Note: This is actually the encoding of the existing ALTO Cost Map response - it's just a matter of interpretation):

```
"SRC-PID": { "DEST-PID": 123 },
```

Note that this is the worst case, it takes less space if several updates share the same source PID:

```
"SRC-PID": { "DEST-1": 123, "DEST-2": 321, .... },
```

The number of bytes per changed cost are:

JSON patch:

$$33 + 2 * \text{NameLength} + \text{CostLength}$$

ALTO Cost Map Message (worst case):

$$8 + 2 * \text{NameLength} + \text{CostLength}$$

We estimate the number of bytes for:

JSON patch:

$$cf * np * np * (33 + pl + cl)$$

ALTO Cost Map Message (worst case):

$$cf * np * np * (8 + pl + cl)$$

where

cf: Fraction of src-dest costs that changed (0 to 1)

np: Number of PIDS

pl: Average length of PID names

cl: Average number of digits in costs

If we assume PID names are 8 characters ($pl = 8$) and costs are 3 digits ($cl = 3$), and 5% of the costs change ($cf = .05$), we get:

	JSON	ALTO Cost
np	patch	Map message
10	220	95
50	5,500	2,375
100	22,000	9,500
250	137,500	59,375
500	550,000	237,500
1000	2,200,000	950,000
2500	13,750,000	5,937,500
5000	55,000,000	23,750,000

Figure 12: Results JSON Patch vs. ALTO extension

Clearly, the ALTO extension as proposed in this document has a higher efficiency in terms of encoded bytes needed compared to JSON Patch.

6. IANA Considerations

The Incremental Update service as proposed introduces a new MIME type "application/alto-update-param+json" which needs to be registered.

7. Security Considerations

To be done in later versions of this document.

8. Conclusion

This document describes different options that can be applied to support incremental updates of ALTO Network and Cost maps. In particular it comprises option for client and server to synchronize themselves about their current map state, and further includes options on how to encode partial updates. Finally it proposes an new incremental update service and evaluates different options numerically.

9. References

- [I-D.ietf-alto-protocol]
Alimi, R., Penno, R., and Y. Yang, "ALTO Protocol",
draft-ietf-alto-protocol-10 (work in progress),
October 2011.
- [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-01 (work in progress),
June 2011.
- [I-D.pbryan-json-patch]
Bryan, P., "JSON Patch", draft-pbryan-json-patch-02 (work
in progress), October 2011.
- [I-D.pbryan-zyp-json-pointer]
Bryan, P. and K. Zyp, "JSON Pointer",
draft-pbryan-zyp-json-pointer-02 (work in progress),
October 2011.
- [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.
- [RFC5693] Seedorf, J. and E. Burger, "Application-Layer Traffic
Optimization (ALTO) Problem Statement", RFC 5693,
October 2009.
- [RFC5789] Dusseault, L. and J. Snell, "PATCH Method for HTTP",
RFC 5789, March 2010.

Appendix A. Acknowledgments

The authors would like to thank Vijay Gurbani for his valuable input and excellent feedback to this document.

Nico Schwan is partially supported by the ENVISION project (<http://www.envision-project.org>), a research project supported by the European Commission under its 7th Framework Program (contract no. 248565). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the ENVISION project or the European Commission.

Authors' Addresses

Nico Schwan
Alcatel-Lucent Bell Labs
Lorenzstrasse 10
Stuttgart 70435
Germany

Email: nico.schwan@alcatel-lucent.com
URI: www.alcatel-lucent.com/bell-labs

Bill Roome
Alcatel-Lucent Bell Labs

Email: w.roome@alcatel-lucent.com
URI: www.alcatel-lucent.com/bell-labs

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: December 29, 2012

H. Xie
Huawei & USTC
T. Tsou
Huawei (USA)
D. Lopez
Telefonica I+D
H. Yin
Huawei (USA)
June 27, 2012

Use Cases for ALTO with Software Defined Networks
draft-xie-alto-sdn-use-cases-01.txt

Abstract

The introduction of SDN fundamentally changes the way that the ALTO works. This draft describes the Vertical Architecture and the Horizontal Architecture allowing coherent coexistence of application layer traffic optimization (ALTO) with software defined network (SDN). Unique requirements for design and operations are identified and summarized, suggesting that the Vertical Architecture allows better division, management, flexibility, privacy control and long-term evolution of the network. We also define the main interactions and information flows, and present a set of use cases to illustrate how we extend ALTO to support SDN, in the Vertical Architecture.

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 December 29, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	4
1.1. Terminology	5
2. An Overview of Software Defined Network	5
2.1. Software Defined Network	5
2.2. Division of Network	6
2.3. SDN Domain	8
3. Architectural Considerations for SDN and ALTO	9
3.1. The Vertical Architecture	9
3.2. The Horizontal Architecture	11
3.3. Summary	11
4. Interactions between SDN and ALTO	12
4.1. ALTO Scopes	12
4.2. ALTO clients	13
4.3. Interactions between SDN and ALTO	14
4.3.1. Downward Interaction	14
4.3.2. Upward Interaction	14
4.4. Interactions between Legacy ALTO Clients and ALTO Servers	15
4.5. Summary	15
5. Information Flows	15
5.1. Information Flow of SDN Controller	16
5.2. Information Flow of Applications, SDN and ALTO	16
5.2.1. SDN-aware Applications	16
5.2.2. SDN-unaware Applications	17
5.2.3. Legacy Applications	17
5.3. Summary	18
6. Use Case for Upward Flow	18
6.1. Unrestrictive Information Exporting	18
6.2. Restrictive Information Exporting	19
6.3. Information Aggregation	19
7. Use Case for Downward Flow	20
7.1. SDN-Aware QoS Metrics	20
7.1.1. Use Case: On-Demand Bandwidth	21
7.1.2. Delay	22
7.2. Content Delivery Networks (CDN)	22
7.3. Information-Centric Content Delivery Networks (IC-CDN)	24
8. Conclusions	26
9. Contributors	26
10. Acknowledgements	26
11. Security Considerations	26
12. IANA Considerations	27
13. Informative References	27
Authors' Addresses	27

1. Introduction

The concept of Software Defined Network (SDN) has emerged and become one of the fundamental, promising networking primitives that allow flexibility of control, separation of functional planes and continuous evolution in networks.

One of the key features of SDN is the full separation of two functional planes: the control plane and the data-forwarding plane. SDN requires that networking devices support such separation, implementing the data plane mechanisms, while the control plane is provided by an external entity called the "controller". The other key feature of SDN is the new interaction process between the separated control plane and data-forwarding plane. This interaction is mandated to be performed specific open protocols, allowing for a free combination of networking devices and controllers, as well as supporting the controller to take decisions on information additional to the networking device status.

There could be numerous benefits as a result of the above features in SDN, e.g., just to name a few, network virtualization, better flexible control and utilization of the network, networks customized for scenarios with specific requirements. For instance, some SDN technologies have started to find their ways into Data Center Networks (DCNs). Furthermore, in order to allow efficient and flexible implementation of the above separation and interactions, a significant portion of the SDN system could typically be implemented in software, as opposed to the hardware-based implementation adopted by most of today's networking devices.

Due to the great potentials of SDN and the unique requirements of DCNs, Data Centers are likely to become a first place where SDN could be deployed. We envision that SDN could be gradually adopted by enterprise networks and then by carrier networks due to its unique, attractive features. When deploying SDN in large-scale distributed networks, we expect to see a collection of deployments limited in relatively small segments of a bigger network. In other words, it is likely that the operator of a large-scale enterprise / carrier network prefers to divide the whole networks into multiple smaller segments and put each of there segments in an SDN domain. These smaller network segments (therefore their corresponding SDN domains) are connected and jointly form the larger whole network. Such a divide-and-conquer methodology not only allows gradual deployment and continuous evolution, but also enables flexible provisioning of the network.

With the deployment of SDN, application layer traffic optimization (ALTO) faces new challenges including, but not limited to, privacy

preservation, granularity of information collection and exchange, join optimization, etc. We shall elaborate these challenges and their impacts on the design of ALTO extensions for SDN in this draft.

1.1. Terminology

While the definition of software defined networks is still "nebulous" to some extent, we refer to as SDN the networks that implement the separation of the control and data-forwarding planes and software defined interactions between these two separated planes; such interactions are characterized by open interfaces which allow programming the network equipment's forwarding plane by external agents, e.g., SDN controllers. However, how the two separated planes interact is not a focus of this draft; instead, the ALTO extension for SDN recommended in this draft is independent of how such interactions would be.

An SDN domain is a portion of a network infrastructure, consisting of numerous connected networking devices that are SDN capable (i.e., SDN capable devices implement the control/forwarding plane separation and the open interfaces allowing external agents to program the devices) and a domain controller which implements the SDN control plane functionalities for these devices. An SDN domain can be as small as a sub-network of a dozen devices or as large as a medium/large data center network.

A software defined network is a network that has one or multiple SDN domains. Due to an SDN domain typically has limited coverage, an SDN may be comprised of networking devices under control of some SDN domains (i.e., SDN managed devices) and devices under no control of any SDN domain (i.e., normal devices). Note that such normal devices could still be SDN capable and their control/forwarding planes are managed as in normal networks today. This implies that a network with both normal devices and SDN capable devices (managed by SDN domains) needs both normal and SDN capable control/forwarding plane management.

2. An Overview of Software Defined Network

This section provides a high level and conceptual overview of software defined network in order to help illustrate its unique requirements for ALTO.

2.1. Software Defined Network

We refer to as an "SDN" a carrier's or an enterprise's network which deploys or implements software defined networking technologies.

Since SDN separates the control plane and data-forwarding plane, we refer to the entity that implements the control-plane functionalities as the "SDN controller".

In order for a network to be classified as an SDN, it is unnecessary that all networking devices have to be SDN capable. Some of devices in a network may be managed by an SDN controller while the remaining ones may not; such a network is still qualified as an SDN.

Applications in software defined networks are either SDN-aware or unaware of SDN.

- o If an application is SDN-aware, then the application may prefer direct communication with SDN controllers, which implies that there must exist mechanism(s) for SDN-aware applications to locate and communication with SDN controllers. If the application prefers indirect communication with SDN controllers, then it follows the case of SDN-unaware applications (see below). Applications that are SDN-aware may be able to better utilize the SDN capable network due to its knowledge about SDN and its capability of proactive, direct interaction with SDN.
- o If an application is SDN-unaware, then the application indirectly communicates with SDN controllers by sending application datagrams with specific formats, via which the application can specify its requirements for the network resources. Legacy applications (applications for the current IP networks) are largely SDN-unaware, and such applications may not be able to utilize the SDN capable networks as efficiently as SDN-aware applications.

Whether and how applications should/can be SDN-aware or SDN-unaware is beyond the scope of this draft. However, the framework we described in this draft is applicable to both SDN-aware and SDN-unaware cases.

2.2. Division of Network

A network operator may decide to divide the network into multiple sub-networks, some of which are SDN capable and thus are managed by corresponding SDN controllers.

There could be numerous reasons for such division of network. Below we summarize a few of them:

- o Scalability.

The number of devices an SDN controller can feasibly manage is likely to be limited. Therefore, in order to manage a many

devices that cannot be put under control of a single SDN controller, multiple controllers have to be deployed. Hence, the network is divided into multiple sub-networks; if a sub-network has SDN capable devices, it should be managed by an SDN controller.

- o Manageability.

At the network level, in order to reduce the complexity of management, a carrier may decide to divide the network into multiple sub-networks and put some of them under control of some SDN controllers (provided that the devices in such sub-networks are SDN capable); each of the sub-networks can be managed independently to some extent, thus reducing the overall complexity of managing the whole network. Even at the sub-network level, a carrier may still decide to further divide the sub-network in order to further reduce complexity of management. For instance, a sub-network under control of an SDN controller may span across a large geographical area or cover a large number of devices; in this case it is reasonable for the carrier to further divide it into two or even more sub-networks, such that the complexity of managing each individual sub-network plus the overall complexity of managing all divided sub-networks are reduced.

- o Privacy.

When a carrier divides its network into multiple sub-networks and put them under control of SDN, the carrier may choose to implement difference privacy policies in different sub-networks. For example, a carrier may dedicate a part of its infrastructure to some certain customers, dividing the whole network and dedicate some of the sub-networks is a convenient and scalable way to manage the network resources for these customers. Another example is that a sub-network in a carrier's network may be dedicated to some certain customers and such information as network topology may not be disclosed to any external entity.

- o Deployment

The deployment of network infrastructures, especially new network infrastructure and technologies, has to be incremental. This means that at any time, a carrier's network may consist of a portion of legacy and a portion of non-legacy infrastructure. When deploying new infrastructure or technologies, it is highly preferable to limit the scope of new deployment and do it in an incremental way. In such cases, it is very favorable to divide the network into multiple individually manageable sub-networks and choose some of them to deploy the new infrastructure /

technologies.

2.3. SDN Domain

With the introduction of SDN, we believe that it is inevitable for carriers to divide their networks for many reasons as illustrated in the preceding subsection. Therefore, we believe that to better suit this need, it should be recommended that SDN domains are defined and applied in division of the networks.

An SDN domain is a sub-network, resulted from division of a network which is determined by the network operator. Each domain typically consists of numerous connected networking devices, each of which is SDN capable. Each domain also has a domain controller which implements SDN control plane functionalities for the devices in this domain. Another important task such a domain controller is responsible for includes fine-grain network information collection (for devices in this domain). The collected information is necessary for the controller to make data-forwarding plane decisions. Note that such a domain controller may be integrated as a part of a so-called "network operating system" (NOS). An example of such a network operating system is illustrated in [1].

Any networking device, if under the control of certain SDN domains, should belong to only one SDN domain and should be under the control of the SDN domain's controller. In other words, the intersection of two domains is always empty.

Furthermore, SDN domains are connected (via the connectivity among underlying devices provided by the underlying network; such devices belong to different SDN domains) to form the whole network. For a large-scale distributed networks owned by a national/multi-national carrier or enterprise, it is natural to adopt the "divide-and-conquer" strategy and divide the whole network into multiple SDN domains. Even for small or medium networks, multiple SDN domains may be necessary in order to virtualize the network resources (e.g., set up multiple SDN domains for a large data center network to instantiate multiple virtual data centers for many content providers). Note that how multiple SDN domains inside a carrier's/enterprise' network work together is beyond the scope of this draft and is handled by other working groups.

Inside each SDN domain, its controller may define domain-specific policies on information importing from devices, aggregating, and exporting to external entities. Such policies may not be made public; therefore, other domain controllers or ALTO may not know the existence of such policies for any given SDN domain.

The natural network division implemented by SDN domains impose significantly new and challenging requirement for shaping the interactions between SDN and ALTO, and therefore designing the protocols to enable such interactions.

3. Architectural Considerations for SDN and ALTO

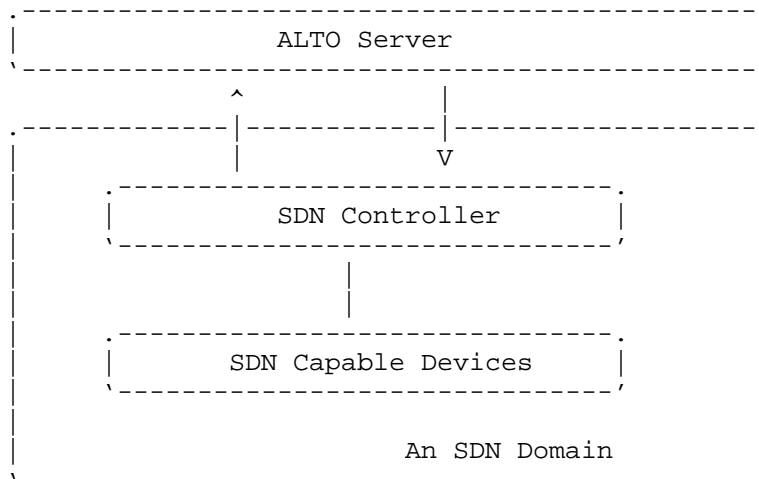
We introduce two architectures that allow coherent co-existence of SDN and ALTO in this section:

- o the Vertical Architecture (or the V Architecture for short) allows better division, management, flexibility, privacy control and long-term evolution of the network, and
- o the Horizontal Architecture (or the H Architecture for short) simplifies the implementation of ALTO extensions for SDN.

We next present each of these two architectures individually.

3.1. The Vertical Architecture

The Vertical Architecture is illustrated in the following figure. The network has one or multiple SDN domains and an ALTO server. The interactions between the SDN controllers and the SDN capable devices fall in the scope of SDN and therefore is out of the scope of this draft; instead, the interactions between the SDN domains (more specifically, SDN controllers) and the ALTO server is what this draft focuses on.



The Vertical Architecture is a hierarchical architecture consisting of three tiers. In the first tier, the only entity is the ALTO server. In the second tier, the only entities are the SDN domain controllers. In the third tier, the only entities are SDN domains (each domain consists of numerous networking devices). In this architecture, all entities are owned by the same carrier. However, some of the SDN domains (and the networking devices in them) may be dedicated to certain customers of the carrier (the carrier gives the customers privileges access). This is subject to a collaboration agreement between them.

The interactions between the SDN controllers and the ALTO server can be divided into two categories:

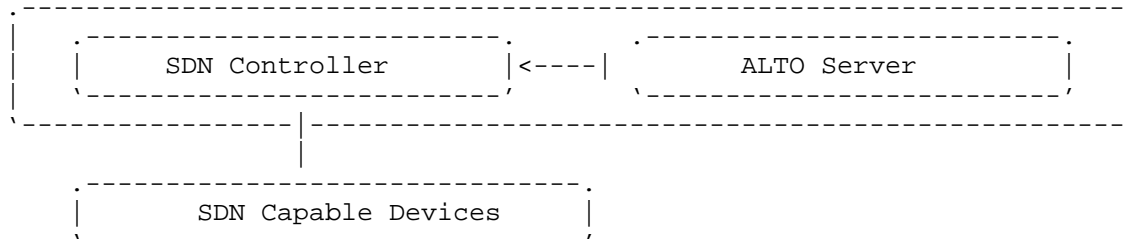
- o Upward interactions (i.e., from SDN controllers to ALTO servers): each SDN controller collects network information from the devices managed by it and information from other SDN controllers), and report such information to the ALTO server, subject to the information aggregation and privacy policies defined for the corresponding individual SDN domain. Such network information is referred to the inter-domain network information. The network information could include key information such as domain-level network cost, bandwidth, domain-specific connectivity, etc. The upward interactions could be implemented in either the push model or the pull model. It is worth noting that network information collection has not been explored, and that network information collection could introduce significant overhead and complexity, in the current scope of ALTO. However, automated network information collection is a key to the success of ALTO. With the help of SDN and the Vertical Architecture, such automated network information collection becomes feasible and appealing. Note that this does not exclude the possibility of network operators manually or automatically update the ALTO server with the network information (e.g., the network cost map). It is also worth noting that an SDN controller may choose to report its domain-specific network information only (referred to as the intra-domain information), with or without privacy policies. In this case, SDN controllers become an automated information collector for the ALTO server.
- o Downward interactions (i.e., from ALTO servers to SDN controllers): each SDN controller is also an ALTO client and retrieves relevant network information from the ALTO server. This is similar to the current scope of ALTO without the existence of SDN; however, the differences are that with the existence of SDN, the network information is generally specific to SDN and SDN domains; SDN controllers as ALTO clients could query the ALTO server for either inter-domain or intra-domain network information (provided that intra-domain information is reported and made

available).

We refer to as the "upward flow" the information flow from the second tier (SDN controllers as ALTO clients) to the first tier (ALTO server), and refer to as the "downward flow" the information flow in the reverse direction, i.e., from the first tier (ALTO server) to the second tier (SDN controllers as ALTO clients).

3.2. The Horizontal Architecture

The Horizontal Architecture is illustrated in the following figure. Each SDN controller is integrated with an ALTO server. The interactions between the SDN controllers and the ALTO server is represented by the horizontal line in the figure. An example of this architecture can be found in [2].



In the Horizontal Architecture, the SDN controller can act as an ALTO client and query the network information of the networking devices from the ALTO server. However, such network information may not meet the SDN controllers' granularity requirement (i.e., the information provided by the ALTO server may not be as fine-grained as needed by the SDN controllers). In addition, there may exist redundant information collection, as SDN controllers are inevitably collecting various fine-grain information from the devices they manage; the information collection by the ALTO server, either manually or automatically, is likely to be redundant. Furthermore, when the carrier decides to divide its network into multiple SDN domains, it can be difficult, if not impossible at all, for the Horizontal Architecture to adapt to the network division.

3.3. Summary

The ALTO server covers a carrier's network as a whole; however, if the carrier divide the network into multiple SDN domains, each SDN domain covers only a segment of the network. Additionally, the ALTO server has only relatively coarse-grained information, while SDN domain controllers could easily collect more fine-grained

information. More importantly, different SDN domains may implement different information aggregation and privacy policies (e.g., when such domains are dedicated to certain customers of the carrier).

These observations suggest that the Vertical Architecture is advantageous over the Horizontal Architecture. With the Vertical Architecture, SDN and ALTO are explicitly separated and as a result the logic is cleaner and this allows them to evolve independently. Furthermore, the Vertical Architecture makes automated information collect possible for ALTO, which could make ALTO deployment and management easier and more attractive. Therefore, in the remainder of this draft, we mainly focus on the Vertical Architecture. We will describe the interactions and the information flows in further details for the Vertical Architecture.

4. Interactions between SDN and ALTO

We now describe the interactions between SDN and ALTO in details. We first compare the ALTO scopes without and with the existence of SDN, and then describe the various interactions existing in the Vertical Architecture.

4.1. ALTO Scopes

The existence of SDN differentiates two scopes of ALTO, namely, the current scope of ALTO without SDN (referred to as the SDN-unfriendly Scope) and the new scope of ALTO with coherent coexistence of SDN (referred to as the SDN-friendly Scope):

- o the SDN-unfriendly Scope: in the current scope of ALTO, there exist interactions between ALTO clients and ALTO servers. Such interactions are one-way interaction, meaning that ALTO information flows in one direction, i.e., from the server to the clients. More specifically, ALTO clients submit ALTO requests to (and subsequently receive ALTO responses from) an ALTO server. Additionally, ALTO clients in the current scope of ALTO are network applications who would like to consume the network resources. ALTO clients are typically managed by network applications rather than by network carriers. However, ALTO servers are owned and managed by network carriers.
- o With the introduction of SDN, the interactions between ALTO clients and ALTO servers become more complex. A more careful examination as well as important ALTO extensions are necessary to make ALTO work in the context of SDN. It is important to note that if the network does not implement network division (i.e., does not implement SDN domains), the interactions are "compressed"

into a compact set of interactions; specifically, both the SDN controller (recall that there exists only one single controller for the whole network) and the ALTO server could be integrated in many equally efficient fashions. For instance, ALTO server could be put underneath the controller, i.e., ALTO server provides information to the controller, as suggested by [2]. However, if the network implements division via SDN domains (i.e., there exists multiple SDN domains), we essentially "unfold" the compressed interaction space and need more complex structures that allow efficient design and implementation, due to the facts that we listed in the preceding sections. Furthermore, the design originally for the compressed space could be instantiated for the unfolded space but could not be as efficient.

We next focus on the SDN-friendly Scope and highlight the complex structures and the important differences.

4.2. ALTO clients

With the existence of SDN and SDN controllers, ALTO clients could be not only legacy network applications (or proxies for these network applications), but also SDN controllers.

In SDN, SDN controllers have similar needs as the legacy ALTO clients which communicate with ALTO servers, because ALTO clients would like to better understand the network and thus improve the application performance.

There are multiple reasons for this similarity. The most important reason is that each SDN controller is only responsible for managing a sub-network in a carrier's network; therefore, it may not understand well other sub-networks in the same carrier network. However, in order to allocate the network resources to satisfy application requirements (note that the end points of such applications may well span across multiple SDN domains), an SDN controller may have to involve other SDN controllers because the network paths needed may traverse multiple SDN domains. Thus, obtaining global information from the ALTO server is a significantly more efficient and more preferable than otherwise from SDN interconnection protocols; plus, such protocols do not exist yet today.

Although SDN controllers have similar needs as legacy ALTO applications do, the fundamental properties of SDN controllers as ALTO clients are significantly different. One of the differences is the ownership and management, as most SDN controllers require additional (and more likely complex) access privileges. Specifically, SDN controllers are typically owned by the network carriers who also own their ALTO servers, while legacy ALTO clients

are network applications typically not owned by network carriers (there are cases where owned collaboratively amongst operators).

In terms of management, the entity managing SDN controller is the same as that managing the ALTO server. We emphasize that when an SDN domain is dedicated to some customers of a network carrier, the use of the SDN domain is owned by these customers and so is the management. In this case, the SDN controllers as ALTO clients are slightly more like legacy ALTO clients. However, there still exist fundamental differences which we will illustrate later.

4.3. Interactions between SDN and ALTO

Another difference is the interactions between SDN controllers as ALTO clients and ALTO servers. Legacy ALTO clients retrieve information from ALTO servers. However, SDN controllers may also need to push information to ALTO servers. In a carrier's network, SDN controllers and the ALTO server are owned by the same carrier. Interactions between them could be significantly more complex.

4.3.1. Downward Interaction

The downward interaction is the information flow from ALTO servers to ALTO clients (i.e., SDN controllers). SDN controllers request information from ALTO servers, similar to legacy ALTO clients. However, the requested information is significantly different. The fundamental difference is a result of SDN and SDN domain division, which do not exist in legacy network application scenarios. For instance, an SDN controller for a specific SDN domain may be interested in obtaining internal information of other SDN domains (provided that other domains allow to do so), or obtaining domain-level information such as inter-SDN-domain connectivity. None of these is applicable to legacy ALTO client scenarios. As a result, ALTO server and its protocol should be extended to support such scenarios.

4.3.2. Upward Interaction

The upward interaction is the information flow from ALTO clients (i.e., SDN controllers) to ALTO servers. SDN controllers open up the possibilities of conveniently collecting network information and exporting such information to ALTO servers. SDN controllers are at the best position to collect network information. More importantly, it is an evitable requirement that SDN controllers collect the information of the networking devices in its domain.

In some scenarios, it is a requirement that information flow from ALTO clients to ALTO servers. For instance, an SDN domain could be

dedicated to some of a carrier's certain customers; the usage of such a domain gives privileged client access. However, such a domain is an integral sub-network of the carrier's network. In such a case, the ALTO server for the carrier's network is not able to collect necessary information in a scalable, manageable way. Even if we assume that the ALTO server can automatically pull necessary information directly from networking devices, the dedicated domain may disallow the ALTO server to do so, because customers who own and manage this domain may enforce stringent privacy policies and disallow exporting information externally. The SDN controller is the best entity that can facilitate the automation of information collection while at the same time enforce the specific privacy policy.

4.4. Interactions between Legacy ALTO Clients and ALTO Servers

With the existence of SDN, the way that legacy network applications (i.e., as legacy ALTO clients) interact with ALTO servers is also different.

In legacy ALTO client/server scenarios, ALTO clients obtain cost maps from ALTO servers, with the implicit assumption that ALTO servers understand how the underlying network routes packets, which allows ALTO servers to define or compute a cost metric associated with a given route.

However, with the introduction of SDN, such assumption may no longer hold, as SDN controllers may dynamically negotiate and determine a route between two end points (which may belong to two different SDN domains), especially when applications have specific requirements for network resources (e.g. bandwidth, delay, etc). Thus, in order for applications to best utilize the network resources, the way that legacy ALTO clients communicate with ALTO servers should be adapted to SDN.

4.5. Summary

In the context of SDN, due to the specific and unique properties of SDN domains, SDN controllers as ALTO clients are significantly different from legacy ALTO clients, posing new requirements for the interactions between ALTO clients and ALTO servers.

5. Information Flows

We now further describe the two different information flows through two sets of use cases, one for the information flow from ALTO servers to ALTO clients, the other for the information flow from SDN

controllers to ALTO servers.

5.1. Information Flow of SDN Controller

A network may consist of multiple SDN domains. Note that due to operational or deployment concerns, there may exist networking devices that do not belong to any SDN domain. In each SDN domain, the SDN controller is responsible for the following tasks (only ALTO related tasks are included below):

- o Collect fine-grain information from the networking devices it manages. Such information could include, but not limited to, SDN domain topology, link capacity, available bandwidth on each link, links connected to external devices belonging to other SDN domains.
- o Implement pre-defined domain-specific policies. Such policies could include, but not limited to, how resources should be allocated, how the collected information should be combined and presented.
- o Optionally aggregate the collected information for external view purposes per its policies.
- o Obtain cost maps at the granularity of SDN domains or obtain internal cost maps for specific domains (if available), consult for cross-domain data-forwarding plane recommendations from ALTO.
- o Make (ALTO recommended) data/forwarding plane decisions based on the cost maps obtained from ALTO.

5.2. Information Flow of Applications, SDN and ALTO

We now give three examples to describe a complete work flow, which connects all key elements in an SDN.

5.2.1. SDN-aware Applications

- o An application's end point sends a request for network resources to the SDN controller it belongs to (i.e., the SDN controller for the SDN domain where this application's end point belongs to). The request should include the destination end point or the set of destination end points, and a set of requirements on network resources (e.g., bandwidth)
- o The SDN controller obtains an SDN-specific cost map from the ALTO server (this step may occur independent of remaining steps)

- o The SDN controller uses the cost map and negotiate one or many path(s) with other SDN controllers (since the path may span across multiple SDN domains, thus all SDN controllers of the involved domains should participate in setting up the paths)
- o The SDN controller responds to the requesting application's end point.
- o If the requested path(s) are successfully set up, the application's end point starts to communicate with the destination end points.

5.2.2. SDN-unaware Applications

SDN-unaware applications do not directly communicate with SDN controllers. Instead, they follow special packet formatting rules to encode the SDN-specific requests, and the SDN capable networking devices pick up these requests and forward them the SDN controllers.

The remaining work flow is similar to the work flow of SDN-aware applications, except that SDN controllers do not respond to the requesting applications. Thus, when the requests cannot be satisfied, SDN-unaware applications may suffer from packet losses, due to networking devices process these applications' packets in a best effort fashion.

5.2.3. Legacy Applications

Legacy applications can be greatly simplified, as it is unnecessary and is not helpful for them to directly communicate with ALTO servers any more:

- o An end point of a legacy application sends a packet to a known destination
- o A SDN capable networking device picks up the packet; however, if the path for the two end points has not been set up yet, the SDN controller will be consulted
- o The SDN controller obtains a cost map from the ALTO server (this step may occur independent of remaining steps).
- o The SDN controller negotiate with other SDN controllers to set up a best-effort path for the requesting end point.
- o The forwarding rules for this path are pushed to all networking devices that are on the chosen path

- o Communications between the two end points continue; the forwarding rules may expire if the communication is tore down

In this case, legacy applications are relieved from the complexity of dealing with the ALTO server using the ALTO protocol. ALTO-related intelligence, which fundamentally belongs to the network intelligence, is implemented in the network, rather than partly outside the network.

5.3. Summary

It is worth noting that this architecture is fundamentally different from common ALTO use cases such as ALTO in CDN or data center (DC). The differences lie in that in the latter cases the components in question (e.g., CDN or DC) are largely consumers of ALTO services, while in the former case SDN domains are not only making decisions that may affect ALTO and generating/aggregating information that ALTO needs, but also the consumers of ALTO services. Furthermore, in the former case, SDN domains are an integral part of the underlying network infrastructure where their decisions could be treated as constraints for ALTO; however, in the latter cases, the components in question (e.g., CDN or DC) are apparently not necessarily integral parts of the underlying network and their decisions could be treated as recommended outcomes suggested by ALTO.

6. Use Case for Upward Flow

The upward flow delivers SDN domains' network information by SDN controllers to the ALTO server. Each SDN controller is responsible for collecting, aggregating, and submitting its own domain's network information to the ALTO server. Due to the possibility of some SDN domain being dedicated to certain customers, we illustrate the upward flow in two use cases.

6.1. Unrestrictive Information Exporting

SDN domain controllers have to collect various network information from the networking devices they manage no matter if ALTO exists or not. The reason is that an SDN controller may have to make decisions for allocating resources within its domain, and making such decisions need various network information. Since such information is readily collected and available, an SDN controller could submit such information as is (or after simple processing) to the ALTO server. Take the available link bandwidth as an example (available link bandwidth could be used as a measure of "distance"). An SDN controller could periodically collect the available bandwidth on all links in its domain and submit it to the ALTO server. However, such

information should be annotated with the domain information (e.g., domain ID). By submitting such information, later other SDN controllers may request for this domain's available link bandwidth information.

6.2. Restrictive Information Exporting

An SDN domain belonging to a carrier may be dedicated to certain customers of that carrier. In this case, the dedicated users of an SDN domain manage not only how resources should be allocated but also what information should be exported.

A carrier may dedicate a set of small data centers (on multiple sites) to its certain customer. These data centers are put under a single SDN domain. The customer can manage the dedicated multi-site, small data centers via the SDN controller. Periodically the SDN controller collects network information from all data centers.

However, different than the former unrestrictive case, the customer may have stringent privacy policies and therefore decide to aggregate the collected information before submitting to the ALTO server.

For instance, the customer may aggregate the information for a data center network in the same site such that the data center network is shrunk into a single node; by doing so, the multi-site data center network is aggregated into a multi-node network topology, each node in the topology actually corresponds to a small data center in reality. The aggregated network topology could be annotated with available link bandwidth information or other information that is collected and allowed to be exported.

The customer's information aggregation policy defines how the information should be pre-processed before exporting to the ALTO server. The main purpose of aggregation is to protect privacy. As a result of information aggregation, the exported network information could be a logical topology (annotated with various network information, e.g., distance or cost) which is totally different from the physical topology.

6.3. Information Aggregation

Without SDN, ALTO defines cost maps for an aggregated view of the network topology, where the granularity of aggregation is determined by the network carrier and could be either coarse-grain or fine-grain.

However, with the introduction of SDN, such information aggregation could be greatly simplified and should be policed based on the

policies defined for each SDN domain. For instance, ALTO only needs to collect information from a pre-defined set of SDN domain controllers, where the controllers determine at what granularity they would like to aggregate the information and export them. In addition, such aggregation is governed by the domain-specific policies, which defines not only the granularity of aggregation but also to whom such aggregated information may be exposed.

More specifically, an illustrative use case is as follows. SDN controllers collect fine-grain information and aggregate it periodically per their policies. ALTO is configured to obtain the aggregated information from a set of SDN domain controllers and obtain possibly raw information from networking devices (or the network operation center). ALTO then constructs a complete view of the overall network (an aggregated view of the network). SDN controllers obtain cost maps from ALTO and apply such maps when making data/forwarding plane decisions.

Another illustrative use case is as follows. SDN controllers may choose to export fine-grain information to ALTO. After it obtains the cost maps from ALTO, it could leverage the cost maps with greater details about their own domains and make informed decisions. However, SDN controllers should not overload ALTO by exporting too much fine-grain information.

7. Use Case for Downward Flow

We illustrate the use of downward flow through several use cases as follows.

Note that when the originating SDN domain's controller make decisions for choosing path(s) and set up the path(s), each involved SDN domain controller should map the overall decision to scoped decisions specifically for their responsible domains.

7.1. SDN-Aware QoS Metrics

We use two use cases to describe SDN-aware QoS. When aggregating QoS information, SDN controllers or the information aggregation policies should understand the semantics of each QoS metrics. For instance, some metrics (e.g., delay) are additive, while some others are multiplicative (e.g., packet loss rate). The information aggregation policy should be flexible enough to specify such details.

7.1.1.1. Use Case: On-Demand Bandwidth

An SDN capable application / source end-point may request for a certain amount of end-to-end bandwidth to a destination end-point on the fly. The two end points in question should be in the same administrative domain, but they are not in the same SDN domain. The path(s) set up for such a request span across multiple SDN domains.

The SDN controller of the source domain (i.e., the SDN domain where the source end-point is located), referred to as the source SDN controller, should first obtain the cost maps from the ALTO server. Such cost maps are SDN-domain-specific, namely, the costs are defined for pairs of SDN domains, rather than for pairs of end points as in the legacy ALTO case.

The source SDN domain controller should then determine path(s) for the two end points based on the cost maps and associated information obtained from ALTO. More specifically, the controller should:

- o Compute a lowest-cost path at the SDN domain level using the obtained SDN-domain-specific cost map.
- o Contact the controllers of those SDN domains on the selected path, probing for the available bandwidth that could be dedicated to the requested session.
- o Check if all of the selected path have sufficient combined bandwidth that matches the required bandwidth
- o if the combined bandwidth of all selected paths cannot match the requirement, then go back to step 1 and select another lowest-cost path (different than the already selected ones)
 - * if no path can be selected and the combined bandwidth does not match the requirement, the request cannot be satisfied.
- o if the combined bandwidth of all selected paths match the requirement, then set up all selected paths by signaling all involved SDN domain controllers. Note that the signaling protocol and how to set up paths are beyond the scope of this document.

Data backup and migration among data centers, which typically require bulk data transfers, is an example of on-demand bandwidth use case. Data centers may be managed by one or multiple SDN domains; thus bulk data transfer could be thought of as bulk data transfer among multiple SDN domains.

7.1.2. Delay

Similar to the preceding use case, applications may request for paths satisfying some certain QoS metrics, e.g., VoIP applications may ask for paths with delay being lower than certain thresholds. This requires that ALTO cost maps embed such information, and that SDN controller should export such information to ALTO.

7.2. Content Delivery Networks (CDN)

Content Delivery Network (CDN) has been widely deployed to help dissemination of content at the Internet scale. Network carriers are also deploying CDNs inside their own networks to improve the user experiences of their customers. With the introduction of SDN, not only legacy CDN but also a new SDN-based CDN can be seamlessly implemented and integrated with the current network infrastructure.

Here is an example of the flow of SDN-enabled CDN. Suppose that there are a set of CDN servers deployed in a carrier's network and they are willing to be managed by SDN. An equivalent class for each of the CDN server is defined by either the CDN carrier or the network carrier (these two carriers can be the same). An equivalent class is a set of IP addresses, one for a CDN server, where if one can be used to fulfill requests for a specific content, then any server in this class can also be used to serve the same requests. In the extreme case, there is only one equivalent class for all CDN servers.

Then the pre-defined equivalent classes are pushed to the SDN controllers, which leverage such information to select CDN servers and set up paths for any end point to any such servers.

- o A network client (e.g., an HTTP-based Web client) obtains the IP address, referred to as A, of one of the CDN servers in the carrier's network (e.g., by DNS queries)
- o The client sends a first packet destined for A (for HTTP requests, this packet is a TCP SYN packet)
- o An SDN capable networking device picks up the packet
- o If there are forwarding rules already set up for the communication between the requesting client and the destination A, then follow the rules to forward the request packet
- o Otherwise, forward the request packet to the SDN controller of this domain

- o Once receiving a forwarded packet from a networking device, the SDN controller takes the following actions:
 - * Retrieves the equivalent class for the given destination A
 - * Obtains a cost map from the ALTO server (this step could take place asynchronously)
 - * Ranks all CDN servers in the equivalent class according to the cost map obtained from the ALTO server
 - * Selects the best CDN server, referred to as B, based on the above ranking
 - * Negotiates and sets up a best-effort path to the selected CDN server with other controllers
 - * Sets up forwarding rules for the path, and rewriting rules for replacing the IP address of A with the IP address of B (so that the client is actually communicating with B, although it may think that it is communicating with A; however, which server it communicates is not important)
- o The request packet is forwarded to the chosen CDN server B, subject to the forwarding rules and rewriting rules
- o The client communicates with the CDN server B
- o The path and associated forwarding/rewriting rules are expired when the communication is torn down (this step is irrelevant to the ALTO extension for SDN, therefore, it is out of scope)

However, the above use case has two limitations. First, it violates the TCP semantics; namely, the client intends to and believes that it is communicating with server A, but actually it is communicating with server B. Second, it has to rely on the capability of devices being able to rewrite forwarding rules (e.g., use one IP address to replace another one in a packet).

If the above two limitations become concerns, e.g., either TCP semantics should not be violated or rewriting is not available or both, the above SDN-enabled CDN use case can be implemented in similar way, with the help of a redirection server.

Below we describe the steps that are different:

- o A redirection server (or server farm), referred to as R, is set up for redirecting client requests

- o Each SDN controller sets up path(s) to the given redirection server R
- o Note that the redirection server could be an integral component of an SDN controller (either collocated or integrated), in which path(s) are not necessary
- o Once receiving a forwarded packet from a networking device, the SDN controller takes the following actions:
 - * Retrieves the equivalent class for the given destination A
 - * Obtains a cost map from the ALTO server (this step could take place asynchronously)
 - * Ranks all CDN servers in the equivalent class according to the cost map obtained from the ALTO server
 - * Selects the best CDN server, referred to as B, based on the above ranking
 - * Sends the information of the chosen CDN server, i.e., its IP address B, to the redirection server R
 - * Negotiates and sets up a best-effort path to the redirection server R (if R is not integrated with the SDN controller)
 - * Sets up forwarding rules for the path to R
 - * Negotiates and sets up a best-effort path to the CDN server B
 - * Sets up forwarding rules for the path to B
- o The client communicates with the redirection server R
- o R sends an HTTP redirection packet to the client, redirecting future requests to the CDN server B (which is notified by the SDN controller)
- o The client communicates with the chosen CDN server B (note that the path to B has been already set up)

7.3. Information-Centric Content Delivery Networks (IC-CDN)

Information-Centric Networking (ICN) is a "host-to-information" communication model, different from the legacy "host-to-host" model implemented by the Internet. Content Delivery Network (CDN) is more of a "host-to-information" model (i.e., CDNs can be treated as a

special instance of ICN), but implemented in the "host-to-host" model, due to the fact that the current semantics provided by the Internet only support the "host-to-host" model.

With the introduction of SDN, CDNs can be converted into an information-centric networking implementation:

- o A CDN client sends a request for a specific content
- o The request packet is formatted per the CDN in SDN specification (beyond the scope of this draft), containing
 - * the requested content name in the packet
 - * destination (a specific anycast IP address) which is reserved for legacy applications to invoke SDN capabilities
 - * (optional) QoS requirements (e.g., prefer fast/local servers vs. slow/remote servers, demands are elastic or not)
- o An SDN capable networking device picks up the request packet
- o If there are forwarding rules set up for this content request already, then follow the rules to forward the request packet, and terminate this.
- o Otherwise, forward the request packet to the SDN controller for this domain.
- o The SDN controller communicates with the CDN's name directory to look up possible CDN servers that can satisfy the request
- o The SDN controller obtains a cost map from the ALTO server
- o The SDN controller applies the cost map to select the best CDN server per the QoS requirements if specified in the request
- o The SDN controller negotiate the path to the selected CDN server with other controllers
- o The SDN controllers that along the chosen path set up the path, and push the forwarding rule(s) for this chosen path to all networking devices that are involved
- o The request packet is forwarded to the chosen CDN server
- o Data packets flow back to the CDN client

In this use case, the CDN clients could be modified to send the "information-centric" request. However, in a realistic implementation, neither the CDN clients nor the CDN servers have to be significantly modified (e.g., CDN redirection could be leveraged to implement the above work flow).

8. Conclusions

In this draft, we identify the fundamental differences between legacy ALTO client/server and ALTO client/server with the existence of SDN. The introduction of SDN fundamentally changes the way that the ALTO works. We present the Vertical Architecture and the Horizontal Architecture to allow coherent coexistence of SDN and ALTO. We believe that the Vertical Architecture allows better division, management, flexibility, privacy control and long-term evolution of the network. Therefore we mainly focus on the Vertical Architecture in this draft. We also define the main interactions and information flows, and present a set of use cases to illustrate how we extend ALTO to support SDN, in the Vertical Architecture.

9. Contributors

The authors would like to thank Vijay K. Gurbani for his many detailed reviews and helpful assistance on this draft.

Vijay K. Gurbani
Bell Laboratories, Alcatel-Lucent
1960 Lucent Lane, Rm. 9C-533
Naperville, IL 60566
USA

Email: vkg AT (acm.org,bell-labs.com)

10. Acknowledgements

The authors would like to thank Aditi Vira for editing the draft.

11. Security Considerations

TBD.

12. IANA Considerations

This document makes no specific request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

13. Informative References

- [1] Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., Ramanathan, R., Iwata, Y., Inoue, H., Hama, T., and S. Shenker, "Onix: A Distributed Control Platform for Large-scale Production Networks", Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI 10), Vancouver, Canada, pp. 351-364", October 2010.
- [2] Gurbani, V., Scharf, M., Lakshman, T., and V. Hilt, "Abstracting network state in Software Defined Networks (SDN) for rendezvous services, IEEE International Conference on Communications (ICC) Workshop on Software Defined Networks (SDN)", June 2012.

Authors' Addresses

Haiyong Xie
Huawei & USTC
2330 Central Expy
Santa Clara, CA 95050
USA

Phone:
Email: Haiyong.xie@huawei.com

Tina Tsou
Huawei (USA)
2330 Central Expy
Santa Clara, CA 95050
USA

Phone:
Email: Tina.Tsou.Zouting@huawei.com

Diego R. Lopez
Telefonica I+D
Don Ramon de la Cruz, 84
Madrid, 28006
Spain

Phone:
Email: diego@tid.es

Hongtao Yin
Huawei (USA)
2330 Central Expy
Santa Clara, CA 95050
USA

Phone:
Email: Hongtao.yin@huawei.com

