

ALTO WG
Internet-Draft
Intended status: Standards Track
Expires: August 29, 2013

R. Alimi, Ed.
Google
R. Penno, Ed.
Cisco Systems
Y. Yang, Ed.
Yale University
Feb 25, 2013

ALTO Protocol
draft-ietf-alto-protocol-14.txt

Abstract

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

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

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

Requirements Language

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

Status of this Memo

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

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

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

This Internet-Draft will expire on August 29, 2013.

Copyright Notice

Copyright (c) 2013 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.	ALTO Information 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.	Provider-defined Identifier (PID)	13
4.2.	Endpoint Addresses	13
4.2.1.	IP Addresses	14
4.3.	Example Network Map	14
5.	Cost Map	15
5.1.	Cost Attributes	15
5.1.1.	Cost Type	15
5.1.2.	Cost Mode	16
5.2.	Cost Map Structure	17
5.3.	Network Map and Cost Map Dependency	17
6.	Endpoint Properties	18
6.1.	Endpoint Property Type	18
6.1.1.	Endpoint Property Type: pid	18
7.	Protocol Specification: General Processing	18
7.1.	Overall Design	18
7.2.	Notation	19
7.3.	Basic Operation	19
7.3.1.	Discovering Information Resources	19
7.3.2.	Requesting Information Resources	19
7.3.3.	Response	20
7.3.4.	Client Behavior	21
7.3.5.	Authentication and Encryption	21
7.3.6.	HTTP Cookies	21

7.3.7. Parsing	21
7.4. Information Resource Attributes	21
7.4.1. Capability Advertisement	22
7.4.2. Accept Input Parameters	22
7.4.3. Media Type	22
7.5. Information Resource Media Type Encoding	22
7.5.1. Meta Information	22
7.5.2. ALTO Information	23
7.5.3. Example	23
7.6. Information Resource Directory	23
7.6.1. Media Type	24
7.6.2. Encoding	24
7.6.3. Example	25
7.6.4. Usage Considerations	28
7.7. Protocol Errors	29
7.7.1. Media Type	29
7.7.2. Resource Format	30
7.7.3. Error Codes	30
7.7.4. Overload Conditions and Server Unavailability	31
8. Protocol Specification: Basic ALTO Data Types	31
8.1. PID Name	31
8.2. Version Tag	31
8.3. Endpoints	32
8.3.1. Address Type	32
8.3.2. Endpoint Address	32
8.3.3. Endpoint Prefixes	33
8.3.4. Endpoint Address Group	33
8.4. Cost Mode	34
8.5. Cost Type	34
8.6. Endpoint Property	35
9. Protocol Specification: Service Information Resources	35
9.1. Map Service	35
9.1.1. Network Map	35
9.1.2. Cost Map	37
9.2. Map Filtering Service	40
9.2.1. Filtered Network Map	40
9.2.2. Filtered Cost Map	42
9.3. Endpoint Property Service	46
9.3.1. Endpoint Property	46
9.4. Endpoint Cost Service	49
9.4.1. Endpoint Cost	49
10. Use Cases	53
10.1. ALTO Client Embedded in P2P Tracker	54
10.2. ALTO Client Embedded in P2P Client: Numerical Costs	55
10.3. ALTO Client Embedded in P2P Client: Ranking	56
11. Discussions	57
11.1. Discovery	57
11.2. Hosts with Multiple Endpoint Addresses	58

11.3. Network Address Translation Considerations	58
11.4. Endpoint and Path Properties	59
12. IANA Considerations	59
12.1. application/alto-* Media Types	59
12.2. ALTO Cost Type Registry	60
12.3. ALTO Endpoint Property Type Registry	62
12.4. ALTO Address Type Registry	62
12.5. ALTO Error Code Registry	63
13. Security Considerations	64
13.1. Privacy Considerations for ISPs	64
13.2. ALTO Clients	64
13.3. Authentication, Integrity Protection, and Encryption	65
13.4. ALTO Information Redistribution	65
13.5. Denial of Service	66
13.6. ALTO Server Access Control	66
14. Manageability Considerations	66
14.1. Operations	67
14.1.1. Installation and Initial Setup	67
14.1.2. Migration Path	67
14.1.3. Requirements on Other Protocols and Functional Components	67
14.1.4. Impact and Observation on Network Operation	68
14.2. Management	68
14.2.1. Management Interoperability	68
14.2.2. Management Information	69
14.2.3. Fault Management	69
14.2.4. Configuration Management	69
14.2.5. Performance Management	69
14.2.6. Security Management	70
15. References	70
15.1. Normative References	70
15.2. Informative References	71
Appendix A. Acknowledgments	73
Appendix B. Authors	74
Authors' Addresses	75

1. Introduction

1.1. Background and Problem Statement

Today, network information available to applications is mostly from the view of endhosts. There is no clear mechanism for a network to convey to network applications its point of view on its network topological structures and path preferences, forcing applications to make approximations using data sources such as BGP Looking Glass and/or applications' own measurements, which can be misleading or inaccurate. On the other hand, modern network applications can be adaptive, 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.

This document defines the ALTO protocol to implement the ALTO Service, which provides a simple mechanism to convey useful network topological and path preference information to applications from the underlying network's Provider's point of view. The ALTO protocol meets the ALTO requirements [I-D.ietf-alto-reqs], and unifies multiple protocols previously designed with similar intentions (see Section 1.2).

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

1.2. Design History and Merged Proposals

The ALTO Protocol specified in this document consists of contributions from

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

See Appendix A for a list of people who have made significant contributions to this effort as well as the aforementioned projects and proposals.

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.

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

1.3.1. Service Providers

Service Providers that use the ALTO Service can benefit in achieving better traffic management. For example, by using ALTO as a tool to interact with applications, a Service Provider gives network information to applications to manage traffic on more expensive or difficult to provision links such as long distance, transit or backup links. This improves the efficiency of provisioning the networking infrastructure of the Service Provider.

1.3.2. Applications

Applications that use the ALTO Service can benefit in achieving better network cooperation and reducing overhead. Specifically, applications taking advantage of the ISP's knowledge can both avoid network bottlenecks and boost application performance. By using ALTO information, applications can reduce the reliance on obtaining network information through third-party databases. Applications relying on measuring path performance metrics themselves can reduce the measurement overhead by conducting only fine-tuning or fault-tolerant measurements on top of ALTO information. A specific example application that can use ALTO information is peer-to-peer overlay applications who can use ALTO information in peer selection.

2. Architecture

We start by introducing the terminology. Then we define 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), Network Location, ALTO Information, and ALTO Information Base.

2.1.1. Endpoint

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

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

2.1.2. 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 a group of Endpoints. For instance, it can be a single IPv4 or IPv6 address, an IPv4 or IPv6 prefix, or a set of prefixes.

2.1.5. 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

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

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

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

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

Figure 1 illustrates that the ALTO Information provided by an ALTO Server may be influenced (at the operator's discretion) by other systems. In particular, the ALTO Server can aggregate 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 control protocols.

It may also be possible for an ALTO Server to exchange network information with other ALTO Servers (either within the same

administrative domain or another administrative domain with the consent of both parties) in order to adjust exported ALTO Information. Such a protocol is also outside the scope of this specification.

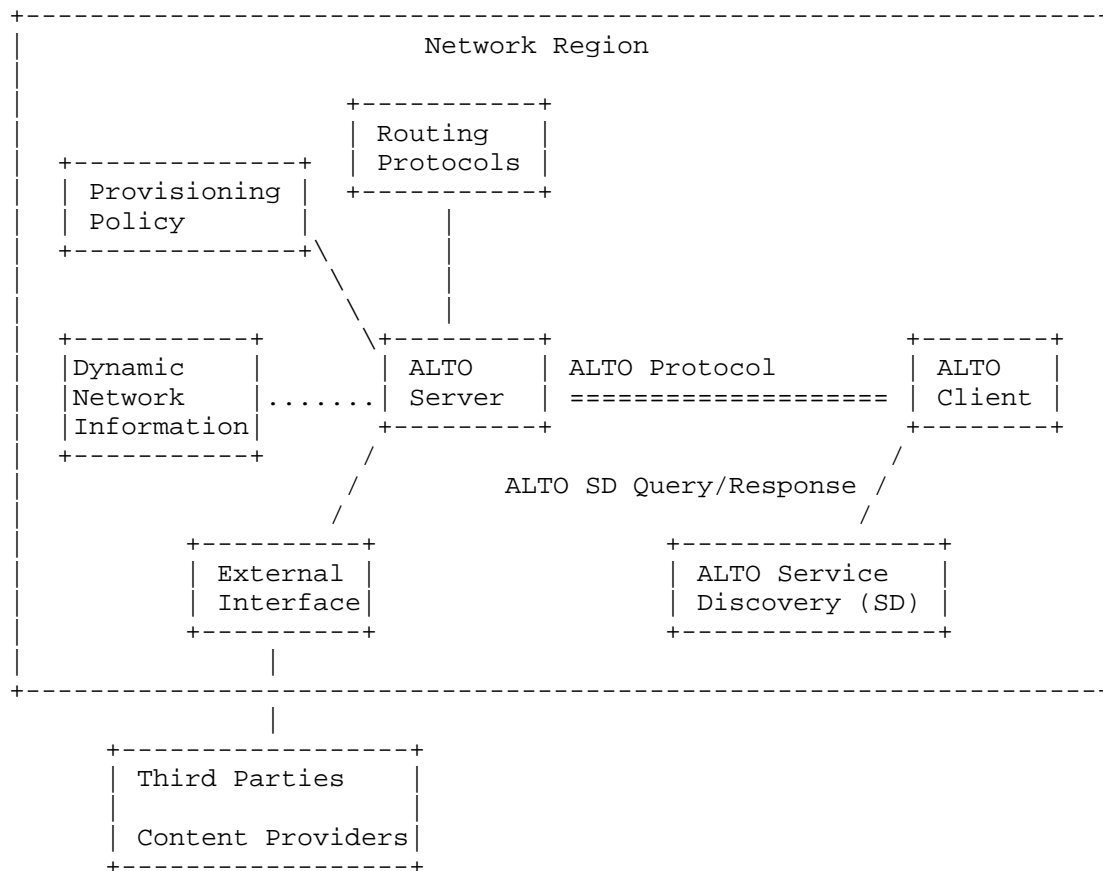


Figure 1: Basic ALTO Architecture.

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.

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

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

3. ALTO Information Service Framework

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

In this document, we focus on achieving the goal of conveying Network Locations, which denote the locations of Endpoints at a network, and provider-defined costs for paths between pairs of Network Locations. We achieve the goal by defining the Map Service, which provides the core ALTO information to clients, and three additional services: the Map Filtering Service, Endpoint Property Service, and Endpoint Cost Service. Additional services can be defined in companion documents. Below we give an overview of the services. Details of the services will be presented in the following sections.

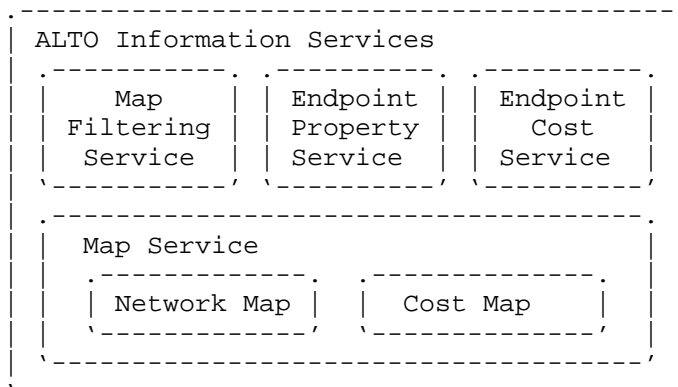


Figure 2: ALTO Service Framework.

3.1. ALTO Information Services

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 (i.e., its grouping defined by the ALTO Server). Another endpoint property is its connectivity type such as ADSL (Asymmetric Digital Subscriber Line), Cable, or FTTH (Fiber To The Home).

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

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

The definition of Network Map is based on the observation that in reality, many endpoints are close by to one another in terms of

network connectivity. By treating a group of close-by endpoints together as a single entity, an ALTO Server indicates aggregation of these endpoints due to their proximity. This aggregation can also lead to greater scalability without losing critical information when conveying other network information (e.g., when defining Cost Map).

4.1. Provider-defined Identifier (PID)

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

ALTO introduces provider-defined Network Location identifiers called Provider-defined Identifiers (PIDs) to provide an indirect and network-agnostic way to specify an aggregation of network endpoints that may be treated similarly, based on network topology, type, or other properties. Specifically, a PID is a US-ASCII string of type PIDName (see Section 8.1) and its associated set of Endpoint Addresses. As we discussed above, there can be many different ways of grouping the endpoints and assigning PIDs. For example, a PID may denote a subnet, a set of subnets, a metropolitan area, a PoP, an autonomous system, or a set of autonomous systems.

A key use case of PIDs is to specify network preferences (costs) between PIDs instead of individual endpoints. This allows cost information to be more compactly represented and updated at a faster time scale than the network aggregations themselves. For example, an ISP may prefer that endpoints associated with the same PoP (Point-of-Presence) in a P2P application communicate locally instead of communicating with endpoints in other PoPs. The ISP may aggregate endhosts within a PoP into a single PID in the Network Map. The cost may be encoded to indicate that Network Locations within the same PID are preferred; for example, $\text{cost}(\text{PID}_i, \text{PID}_i) == c^*$ and $\text{cost}(\text{PID}_i, \text{PID}_j) > c^*$ for $i \neq j$. Section 5 provides further details on using PIDs to represent costs in an ALTO Cost Map.

4.2. Endpoint Addresses

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

4.2.1. IP Addresses

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

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

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

4.3. Example Network Map

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

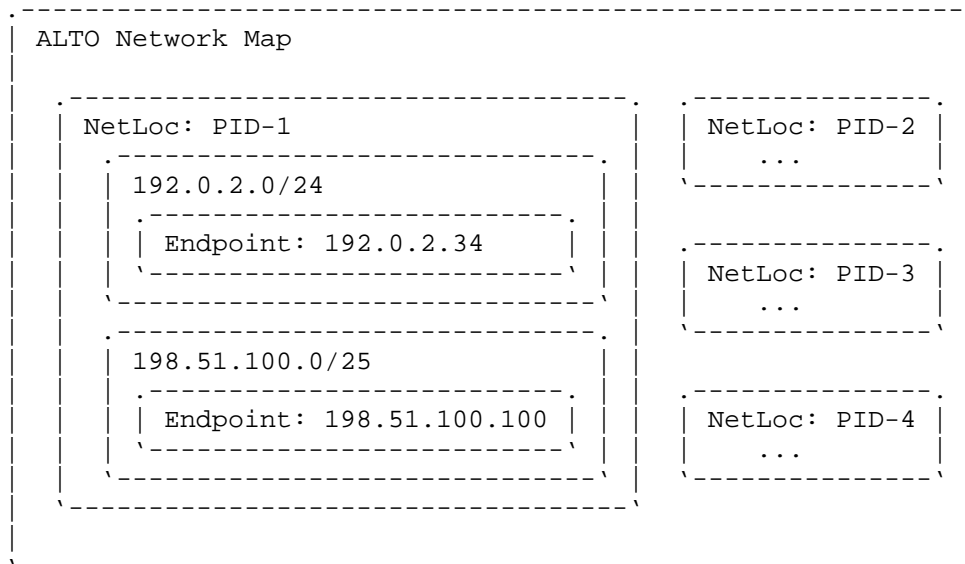


Figure 3: Example Network Map.

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 defined by PIDs. Each Path Cost is the end-to-end cost from the source to the destination.

As cost directional from the source to the destination, an application, when using ALTO, 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: routingcost

An ALTO Server MUST define the 'routingcost' 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. An ALTO Client SHOULD be cognizant of operations when a desired cost mode is not supported. For example, an ALTO Client desiring numerical costs may adjust 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 of the $m \times n$ communicating pairs (i.e., Src_1 -> Dst_1, ..., Src_1 -> Dst_n, ..., Src_m -> Dst_1, ..., Src_m -> Dst_n). If the ALTO Server does not define a Path Cost for a particular pair, it may be omitted. We refer to this structure as a Cost Map.

If the Cost Mode is 'ordinal', the Path Cost of each communicating pair is relative to the $m \times n$ 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. Two Version Tags match only if their strings are the same. Whenever the content of the Network Map maintained by the ALTO Server changes, the Version Tag MUST also be changed. Possibilities for generating a Version Tag include the

last-modified timestamp for the Network Map, or a hash of its contents, where the collision probability is considered zero in practical deployment scenarios.

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

6. Endpoint Properties

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

6.1. Endpoint Property Type

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

6.1.1. Endpoint Property Type: pid

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

7. Protocol Specification: General Processing

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

7.1. 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.

7.2. 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.

7.3. 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].

7.3.1. 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 7.6 for a detailed specification.

7.3.2. 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 ALTO Client MUST use the HTTP POST method.

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.

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.

7.3.3. 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 7.4.

Errors are indicated via either ALTO-level error codes, or via HTTP status codes; see Section 7.7.

7.3.4. Client Behavior

7.3.4.1. 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.

7.3.4.2. 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, when used in a peer-to-peer system). An ALTO Client may also retry the request at a later time.

7.3.5. Authentication and Encryption

An ALTO Server **MUST** support SSL/TLS [RFC5246] to implement server and/or client authentication, encryption, and/or integrity protection. See [RFC6125] for considerations regarding verification of server identity.

7.3.6. HTTP Cookies

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

7.3.7. 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.

7.4. Information Resource Attributes

An Information Resource encodes the ALTO Information desired by an ALTO Client. This document specifies multiple Information Resources that can be provided by an ALTO Server. Each Information Resource has certain attributes associated with it, including its capabilities, the accepted input parameters, and output data format.

7.4.1. Capability Advertisement

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.

7.4.2. Accept Input Parameters

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].

7.4.3. Media Type

An ALTO Server uses Media Type [RFC2046] to uniquely indicate the data format of the Information Resource that it returns in the HTTP entity body.

7.5. Information Resource Media Type 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.

Specifically, each 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

7.5.1. 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;
```

7.5.2. ALTO Information

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

7.5.3. 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" : {  
    ...  
  }  
}
```

7.6. 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]. For recommendations on how the URI may look like, see [I-D.ietf-alto-server-discovery].

7.6.1. Media Type

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

7.6.2. 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. URIs can be relative and MUST be resolved according to section 5 of [RFC3986].

media-types The list of all media types of Information Resources (see Section 7.4.3) 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 7.4.2) 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.

7.6.3. 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" ]
      }
    }
  ]
}
```

7.6.4. Usage Considerations

7.6.4.1. 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 7.6.3, 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.

7.6.4.2. 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".

7.7. Protocol Errors

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

An ALTO Client MUST interpret both HTTP Status Code and ALTO Error Code. If the ALTO Error Code indicates an error, the ALTO Client should consider that the request has failed.

7.7.1. Media Type

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

7.7.2. Resource Format

An ALTO Error Resource has the format:

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

where:

code An ALTO Error Code defined in Table 1

7.7.3. 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.

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

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.

7.7.4. 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.

8. Protocol Specification: Basic ALTO Data Types

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

8.1. 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.

8.2. 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.

8.3. 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.

8.3.1. 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 12.4.

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

8.3.2. 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.

8.3.2.1. IPv4

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

8.3.2.2. IPv6

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

8.3.2.3. 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.

8.3.3. 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.

8.3.3.1. IPv4

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

8.3.3.2. IPv6

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

8.3.4. 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"
  ]
}
```

8.4. 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.

8.5. 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. For an identifier with the 'priv:' or 'exp:' prefix, an additional string (e.g., company identifier or random string) **MUST** follow to reduce potential collisions. For example, a short string after 'exp:' to indicate the starting time of a specific experiment is recommended. All other identifiers appearing in an HTTP request or response with an 'application/alto-*' media type **MUST** be registered in the ALTO Cost Types registry Section 12.2.

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

8.6. 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 12.3.

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

9. Protocol Specification: Service Information Resources

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

9.1. 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.

9.1.1. 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.

9.1.1.1. Media Type

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

9.1.1.2. HTTP Method

This resource is requested using the HTTP GET method.

9.1.1.3. Accept Input Parameters

None.

9.1.1.4. Capabilities

None.

9.1.1.5. 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.

9.1.1.6. 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"
        ]
      }
    }
  }
}
```

9.1.2. 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 9.1.2.4) 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.

9.1.2.1. Media Type

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

9.1.2.2. HTTP Method

This resource is requested using the HTTP GET method.

9.1.2.3. Accept Input Parameters

None.

9.1.2.4. 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.

9.1.2.5. 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).

9.1.2.6. 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 }
    }
  }
}
```

9.2. 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.

9.2.1. Filtered Network Map

A Filtered Network Map is a Network Map Information Resource (Section 9.1.1) 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.

9.2.1.1. Media Type

See Section 9.1.1.1.

9.2.1.2. HTTP Method

This resource is requested using the HTTP POST method.

9.2.1.3. Accept Input Parameters

An ALTO Client supplies filtering parameters by specifying media type "application/alto-networkmapfilter+json" with HTTP POST body containing a JSON Object of type ReqFilteredNetworkMap, where:

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

with members:

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

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

9.2.1.4. Capabilities

None.

9.2.1.5. Response

See Section 9.1.1.5 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.

9.2.1.6. 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"
        ]
      }
    }
  }
}
```

9.2.2. Filtered Cost Map

A Filtered Cost Map is a Cost Map Information Resource (Section 9.1.2) 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.

9.2.2.1. Media Type

See Section 9.1.2.1.

9.2.2.2. HTTP Method

This resource is requested using the HTTP POST method.

9.2.2.3. Accept 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 9.2.2.4).

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 9.2.2.4).

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 9.2.2.4) 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.

9.2.2.4. Capabilities

The URI providing this resource supports all capabilities documented in Section 9.1.2.4 (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 9.1.2.4.

cost-types See Section 9.1.2.4.

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.

9.2.2.5. Response

See Section 9.1.2.5 for the format.

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

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

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

9.2.2.6. 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 }
    }
  }
}
```

9.3. Endpoint Property Service

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

9.3.1. 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.

9.3.1.1. Media Type

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

9.3.1.2. HTTP Method

This resource is requested using the HTTP POST method.

9.3.1.3. Accept Input Parameters

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

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

with members:

properties List of endpoint properties to be returned for each endpoint. Each specified property MUST be included in the list of supported properties indicated by this resource's capabilities (Section 9.3.1.4). 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.

9.3.1.4. 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 {  
  EndpointPropertyType prop-types<0..*>;  
} EndpointPropertyCapability;
```

with members:

prop-types The Endpoint Properties (see Section 8.6) supported by the corresponding URI. If not present, this member MUST be interpreted as an empty array.

9.3.1.5. Response

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

```
object {
  JSONValue [EndpointPropertyType];
  ...
} EndpointProps;

object {
  EndpointProps [TypedEndpointAddr]<0..*>;
  ...
} EndpointPropertyMapData;

object {
  VersionTag          map-vtag;
  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. Otherwise, it is OPTIONAL.

9.3.1.6. Example

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

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

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

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

9.4. 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.

9.4.1. 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 13.1 for additional details.

9.4.1.1. Media Type

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

9.4.1.2. HTTP Method

This resource is requested using the HTTP POST method.

9.4.1.3. Accept Input Parameters

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

```
object {  
  TypedEndpointAddr srcs<0..*>;           [OPTIONAL]  
  TypedEndpointAddr dsts<1..*>;  
} EndpointFilter;  
  
object {  
  CostMode          cost-mode;  
  CostType          cost-type;  
  JSONString        constraints<0..*>;    [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 9.4.1.4).

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 9.4.1.4).

constraints Defined equivalently to the "constraints" input parameter of a Filtered Cost Map (see Section 9.2.2).

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 11.3 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.

9.4.1.4. Capabilities

See Section 9.2.2.4.

9.4.1.5. 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 extension to this document that indicates when and how costs of other data types are signaled. If the ALTO Server does not define a cost value from a Source Endpoint to a particular Destination Endpoint, it MAY be omitted from the response.

9.4.1.6. 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
      }
    }
  }
}
```

10. Use Cases

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

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

10.1. ALTO Client Embedded in P2P Tracker

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

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

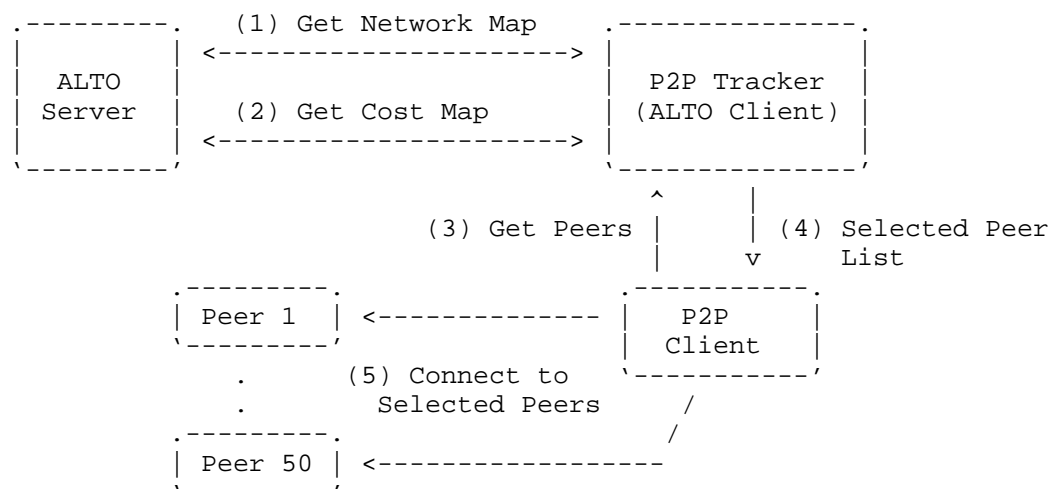


Figure 4: ALTO Client Embedded in P2P Tracker

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

1. The P2P Tracker requests 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.

10.2. 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.

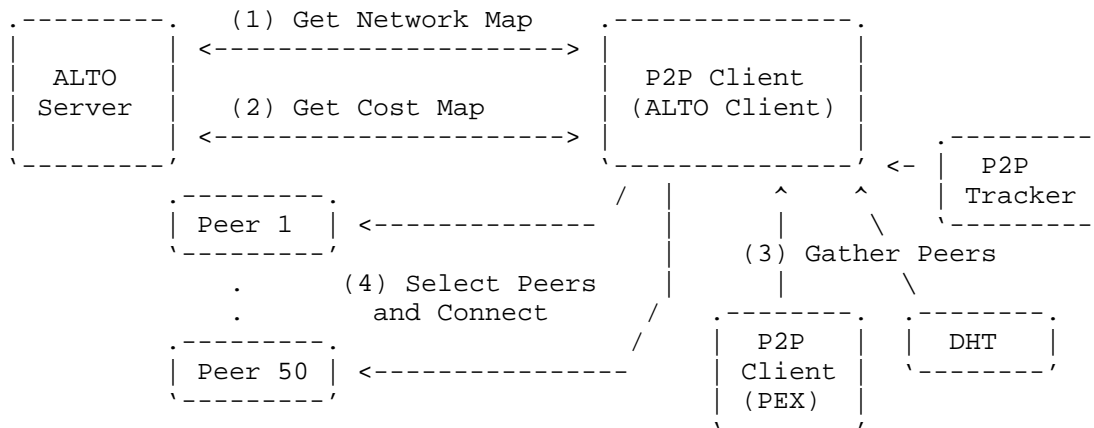


Figure 5: ALTO Client Embedded in P2P Client

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.

10.3. 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.

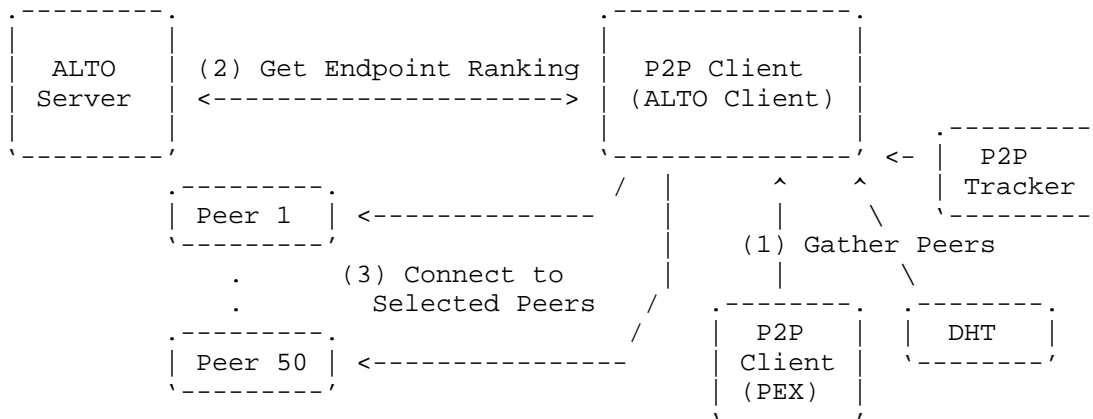


Figure 6: ALTO Client Embedded in P2P Client: Ranking

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.

11. Discussions

11.1. 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 7.6) to locate an Information Resource with the desired ALTO Information.

11.2. Hosts with Multiple Endpoint Addresses

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

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

11.3. Network Address Translation Considerations

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

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

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

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

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

11.4. 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.

12. IANA Considerations

12.1. 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 7.6
application	alto-networkmap+json	Section 9.1.1
application	alto-networkmapfilter+json	Section 9.2.1
application	alto-costmap+json	Section 9.1.2
application	alto-costmapfilter+json	Section 9.2.2
application	alto-endpointprop+json	Section 9.3.1
application	alto-endpointpropparams+json	Section 9.3.1
application	alto-endpointcost+json	Section 9.4.1
application	alto-endpointcostparams+json	Section 9.4.1
application	alto-error+json	Section 7.7

Table 2: ALTO Protocol Media Types

Type name: application

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

Required parameters: n/a

Optional parameters: n/a

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

Security considerations: Security considerations relating to the generation and consumption of ALTO protocol messages are discussed in Section 13.

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).

12.2. ALTO Cost Type Registry

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

Identifier	Intended Semantics
routingcost	See Section 5.1.1.1
priv:	Private use
exp:	Experimental use

Table 3: ALTO Cost Types.

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 8.5. 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 13.1.

12.3. ALTO Endpoint Property Type Registry

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

Identifier	Intended Semantics
pid	See Section 6.1.1
priv:	Private use
exp:	Experimental use

Table 4: ALTO Endpoint Property Types.

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

12.4. ALTO Address Type Registry

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

Identifier	Address Encoding	Prefix Encoding	Mapping to/from IPv4/v6
ipv4	See Section 8.3.2	See Section 8.3.3	Direct mapping to IPv4
ipv6	See Section 8.3.2	See Section 8.3.3	Direct mapping to IPv6

Table 5: ALTO Address Types.

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

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

Registered ALTO Address Type identifiers MUST conform to the syntactical requirements specified in Section 8.3.1. 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 8.3.2).
- o Endpoint Prefix Encoding: The procedure for encoding a set of addresses of the registered type as an EndpointPrefix (see Section 8.3.3). 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', as shown in Table 5.

12.5. ALTO Error Code Registry

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

13. Security Considerations

13.1. 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.

It is possible that one or multiple ALTO Clients issue queries in an effort to reverse-engineer specific details (e.g., network topology) that was used to produce ALTO information. Operators should have security policies in place such that confidential information or information that could be reverse-engineered to reveal confidential information is not sent to unauthorized ALTO Clients.

ISPs must also be cognizant that ALTO may reveal additional information about IP addresses and associated information about it. For example, when adding the line bitrate as one endpoint property, such information may be potentially linked to the income of the inhabitants at the network location of an endpoint.

13.2. 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. As an ALTO Server may collect information from multiple client queries, the server may deduce additional application/content information through correlation. 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.

13.3. Authentication, Integrity Protection, and Encryption

SSL/TLS [RFC5246] 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 13.2.

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.

13.4. 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.

13.5. 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. The ALTO server can also indicate overload. Yet 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.

13.6. 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.ietf-alto-deployments] for a more detailed discussion on this issue).

14. Manageability Considerations

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

14.1. Operations

14.1.1. Installation and Initial Setup

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

Additionally, an operator of an ALTO Server will need to configure the ALTO information to be provided by the ALTO Server. The granularity of the topological map and the cost map is left to the specific policies of the operator of the ALTO Server. However, a reasonable default may include two PIDs, one to hold the endpoints in the operator's network and the second PID to represent full IPv4 and IPv4 reachability (see Section 4.2.1), with the cost between each source/destination PID set to 1. Another operational issue that the operator of an ALTO Server needs to consider is that the filtering service can degenerate into a full map service when the filtering input is empty. Although this choice as the degeneration behavior provides continuity, the operational impact should be considered.

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

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

14.1.2. Migration Path

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

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

14.1.3. Requirements on Other Protocols and Functional Components

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

exist.

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

14.1.4. Impact and Observation on Network Operation

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

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

Operators providing an ALTO Server should ensure that appropriate information is being exposed. Privacy implications for ISPs are discussed in Section 13.1. Both operators and ALTO Servers and those using ALTO Clients should be aware of the impact of incorrect or faked guidance (see Section 10.3 of [I-D.ietf-alto-deployments] and future versions of that document).

14.2. Management

14.2.1. Management Interoperability

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

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

14.2.2. Management Information

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

14.2.3. Fault Management

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

14.2.4. Configuration Management

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

An ALTO Server requires at least the following logical inputs:

- o Data sources from which ALTO Information is derived. This can either be raw network information (e.g., from routing elements) or pre-processed ALTO-level information in the form of a Network Map, Cost Map, etc.
- o Algorithms for computing the ALTO information returned to clients. These could either return information from a database, or information customized for each client.
- o Security policies mapping potential clients to the information that they have privilege to access.

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

14.2.5. Performance Management

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

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

14.2.6. Security Management

Section 13 documents ALTO-specific security considerations. Operators should configure security policies with those in mind. Readers should refer to HTTP [RFC2616] and SSL/TLS [RFC5246] and related documents for mechanisms available for configuring security policies. Other appropriate security mechanisms (e.g., physical security, firewalls, etc) should also be considered.

15. References

15.1. Normative References

- [IEEE.754.2008]
Institute of Electrical and Electronics Engineers,
"Standard for Binary Floating-Point Arithmetic", IEEE
Standard 754, August 2008.
- [RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail
Extensions (MIME) Part Two: Media Types", RFC 2046,
November 1996.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H.,
Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext
Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform
Resource Identifier (URI): Generic Syntax", STD 66,
RFC 3986, January 2005.
- [RFC4627] Crockford, D., "The application/json Media Type for
JavaScript Object Notation (JSON)", RFC 4627, July 2006.

- [RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", BCP 122, RFC 4632, August 2006.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.
- [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.
- [RFC5424] Gerhards, R., "The Syslog Protocol", RFC 5424, March 2009.
- [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.

15.2. Informative References

- [BitTorrent]
"Bittorrent Protocol Specification v1.0",
<<http://wiki.theory.org/BitTorrentSpecification>>.
- [Fielding-Thesis]
Fielding, R., "Architectural Styles and the Design of Network-based Software Architectures", University of California, Irvine, Dissertation 2000, 2000.
- [I-D.akonjang-alto-proxidior]
Akonjang, O., Feldmann, A., Previdi, S., Davie, B., and D. Saucez, "The PROXIDOR Service",
draft-akonjang-alto-proxidior-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-deployments]
Stiemerling, M., Kiesel, S., and S. Previdi, "ALTO Deployment Considerations", draft-ietf-alto-deployments-06 (work in progress), February 2013.
- [I-D.ietf-alto-reqs]
Previdi, S., Stiemerling, M., Woundy, R., and Y. Yang, "Application-Layer Traffic Optimization (ALTO) Requirements", draft-ietf-alto-reqs-08 (work in progress), March 2011.
- [I-D.ietf-alto-server-discovery]
Kiesel, S., Stiemerling, M., Schwan, N., Scharf, M., and S. Yongchao, "ALTO Server Discovery", draft-ietf-alto-server-discovery-07 (work in progress), January 2013.
- [I-D.jenkins-alto-cdn-use-cases]
Niven-Jenkins, B., Watson, G., Bitar, N., Medved, J., and S. Previdi, "Use Cases for ALTO within CDNs", draft-jenkins-alto-cdn-use-cases-03 (work in progress), June 2012.
- [I-D.jennings-sip-hashcash]
Jennings, C., "Computational Puzzles for SPAM Reduction in SIP", draft-jennings-sip-hashcash-06 (work in progress), July 2007.
- [I-D.medved-alto-svr-apis]
Medved, J., Ward, D., Peterson, J., Woundy, R., and D. McDysan, "ALTO Network-Server and Server-Server APIs", draft-medved-alto-svr-apis-00 (work in progress), March 2011.
- [I-D.mrw-nat66]
Wasserman, M. and F. Baker, "IPv6-to-IPv6 Network Prefix Translation", draft-mrw-nat66-16 (work in progress), April 2011.
- [I-D.p4p-framework]
Alimi, R., Pasko, D., Popkin, L., Wang, Y., and Y. Yang, "P4P: Provider Portal for P2P Applications", draft-p4p-framework-00 (work in progress), November 2008.
- [I-D.saumitra-alto-multi-ps]
Das, S., Narayanan, V., and L. Dondeti, "ALTO: A Multi Dimensional Peer Selection Problem", draft-saumitra-alto-multi-ps-00 (work in progress),

October 2008.

[I-D.saumitra-alto-queryresponse]

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

[I-D.shalunov-alto-infoexport]

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

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

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

[P4P-SIGCOMM08]

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

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

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

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

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: Experimental
Expires: August 16, 2013

S. Kiesel
University of Stuttgart
February 12, 2013

Third-Party ALTO Server Discovery Prototype Implementation and Testbed
draft-kiesel-alto-3pdisc-impl-00

Abstract

This document contains a prototype implementation for the third-party ALTO server discovery (3pdisc) procedure specified in draft-kist-alto-3pdisc-02.txt [I-D.kist-alto-3pdisc]. The prototype implementation has been written in Perl and tested with the GNU/Linux operating system.

A set of BIND 9 configuration files and DNS zone files is also included. They can be used for setting up a small, self-contained testbed.

Terminology and Requirements Language

This document makes use of the ALTO terminology defined in RFC 5693 [RFC5693].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "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 August 16, 2013.

Copyright Notice

Copyright (c) 2013 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. Prerequisites	5
2.1. Testbed Host	5
2.2. Operating System	5
2.3. Extracting the Source Code	5
3. Source Code Package Contents	7
3.1. Source Code	7
3.2. BIND 9 Configuration and DNS Zone Files	7
3.3. Sample Program Output	7
4. Security Considerations	8
5. IANA Considerations	9
6. References	10
6.1. Normative References	10
6.2. Informative References	10
Appendix A. Base64-Encoded Source Code	11
Author's Address	14

1. Introduction

The Application-Layer Traffic Optimization (ALTO) problem statement is documented in RFC 5693 [RFC5693] and ALTO requirements are enumerated in RFC 6708 [RFC6708]. draft-kist-alto-3pdisc-02.txt [I-D.kist-alto-3pdisc] specifies a procedure for third-party ALTO server discovery (3pdisc), which tries to satisfy requirement AR-33. This document assumes that the reader is familiar with these documents and the terminology used therein.

This document contains a prototype implementation for the third-party ALTO server discovery procedure specified in draft-kist-alto-3pdisc-02.txt [I-D.kist-alto-3pdisc]. The prototype implementation has been written in Perl and tested with the GNU/Linux operating system.

A set of BIND 9 configuration files and DNS zone files is also included. They can be used for setting up a small, self-contained testbed.

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

2. Prerequisites

2.1. Testbed Host

The testbed has been designed in a way that it is possible to run both the ALTO 3pdisc client and the test name servers on a single host (operating system requirements: see below).

The name server will be configured in a way that the testbed is completely self-contained, i.e., it does not rely on an external DNS infrastructure. On the other hand, this name server will not be able to give reasonable answers to clients other than the ALTO 3pdisc prototype. Therefore, configure and use this only on a dedicated lab host (possibly a virtual machine) completely separated from any production system!

2.2. Operating System

The ALTO 3pdisc prototype implementation has been written in Perl. It has been developed and tested using the Debian GNU/Linux operating system [<http://www.debian.org>], Version 6.0.6 "squeeze". We assume that this code can easily be executed on similar operating system platforms as well (possibly with slight modifications). However, for the sake of completeness we document the versions of the relevant software packages our code has been verified to work with:

ii	perl	5.10.1-17squeeze4
ii	libnet-dns-perl	0.66-2
ii	bind9	1:9.7.3.dfsg-1~squeeze8
ii	dnsutils	1:9.7.3.dfsg-1~squeeze8
ii	coreutils	8.5-1
ii	tar	1.26-4

2.3. Extracting the Source Code

A .tar.gz archive containing the source code as well as the BIND 9 configuration files for the testbed can be extracted from this document by running the following command line in an operating system environment as described above:

```
cat draft-kiesel-alto-3pdisc-impl-00.txt | \
sed -n '/^ \+###/{s/^ \+###//;p;}' | \
base64 --decode > draft-kiesel-alto-3pdisc-impl-00.tar.gz
```

The correct extraction can be verified by calculating a checksum:

```
shasum draft-kiesel-alto-3pdisc-impl-00.tar.gz  
c6f4408c85d12bec215414317c4eac6a2b884729
```

The tar archive can be unpacked:

```
tar -xvzf draft-kiesel-alto-3pdisc-impl-00.tar.gz  
  
draft-kist-alto-3pdisc-02.pl  
bind9/named.conf  
bind9/db.100.51.198.in-addr.arpa  
bind9/db.2.0.192.in-addr.arpa  
bind9/db.8.B.D.0.1.0.0.2.ip6.arpa  
bind9/db.example  
bind9/db.in-addr.arpa  
bind9/db.ip6.arpa  
bind9/db.isp.example  
bind9/db.root  
draft-kist-alto-3pdisc-02.log
```

3. Source Code Package Contents

3.1. Source Code

The ALTO 3pdisc prototype implementation is contained in a single Perl file: draft-kist-alto-3pdisc-02.pl

3.2. BIND 9 Configuration and DNS Zone Files

The bind9/ subdirectory contains several BIND9 configuration and DNS zone files. They will setup a complete self-contained DNS hierarchy with root server, .example TLD [RFC2606], an exemplary network operator's second level domain "isp.example", as well as reverse zones (in-addr.arpa / ip6.arpa) for IP prefixes reserved for documentation [RFC5737][RFC3849]. The zone files mimic delegation between several name servers, but all name servers' A records point to the IPv4 loopback address 127.0.0.1, i.e., the whole hierarchy can run on a single lab host.

This name server configuration will not give reasonable results in a production environment. Carefully review and understand these configuration files before installing them on a dedicated lab host only, in order to avoid negative impacts on your or other people's ability to communicate with Internet and/or LAN hosts!

3.3. Sample Program Output

If you do not have the resources for setting up a testbed according to Section 2 please have a look at the file draft-kist-alto-3pdisc-02.log, which contains the output of the 3pdisc prototype code (Section 3.1) executed in an environment as described in Section 2 and Section 3.2. This log file illustrates the algorithm's behavior.

4. Security Considerations

This draft contains a prototype implementation of the ALTO 3pdisc algorithm, in order to demonstrate the principal feasibility of implementing the algorithm specified in draft-kist-alto-3pdisc-02.txt [I-D.kist-alto-3pdisc]. No special care has been taken to ensure correct behavior in every possible error, DNS forgery or attack situation, or to protect the implementation against different kinds of attacks. Software engineers trying to write code for production use must carefully analyze and consider such issues.

Threats for ALTO and the ALTO server discovery are discussed on an architectural level in [RFC6708] and [I-D.kist-alto-3pdisc].

5. IANA Considerations

None.

6. References

6.1. Normative References

- [I-D.kist-alto-3pdisc]
Kiesel, S., Krause, K., and M. Stiemerling, "Third-Party ALTO Server Discovery (3pdisc)", draft-kist-alto-3pdisc-02 (work in progress), February 2013.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

6.2. Informative References

- [RFC2606] Eastlake, D. and A. Panitz, "Reserved Top Level DNS Names", BCP 32, RFC 2606, June 1999.
- [RFC3849] Huston, G., Lord, A., and P. Smith, "IPv6 Address Prefix Reserved for Documentation", RFC 3849, July 2004.
- [RFC5693] Seedorf, J. and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement", RFC 5693, October 2009.
- [RFC5737] Arkko, J., Cotton, M., and L. Vegoda, "IPv4 Address Blocks Reserved for Documentation", RFC 5737, January 2010.
- [RFC6708] Kiesel, S., Previdi, S., Stiemerling, M., Woundy, R., and Y. Yang, "Application-Layer Traffic Optimization (ALTO) Requirements", RFC 6708, September 2012.

Appendix A. Base64-Encoded Source Code

```
####H4sIAPiGGLLEAA+ld/3vaONLvz/wVCs3zBPbAgCFJSy67pQlteS6FvEC2l2fTzTlY
####JL4am7VN0tze3t/+zki2McbGhkCaNFk/hGBpRjMafWZGEKl1KFT/KrZTlHRHbNY
####HauaPSiWZWmsvlhTKUPZ393Fn5X93XLWjXrZ3t2VXlTkSnlvtyzvvy/svyhUZq5Py
####ujqwqExsR7EIEQhWj7G83CpNbKt0qRm1MbV0UrzNZF4S0r/WLLU4ViznjJRO+hli
####U+uGWgSNw4QXdyTHLSWPtceWaq6L8HdgGgM6dog2Gut0RA1HcTTTIEMTWsYamvPN
####QSKKoZiccrSJ0cy79TXqDh193oGi34FieZ+3dLtrk6GmUwI/6TfHUGyYOVcnQMkc+
####Y2pTfYYlDrJYLIMViSCVU50qNgVRKXGuFYeo5mCCQrD+K6qqoTCKTjQD3hgX0SRs
####9w4eT8aqqiyhaza8jzIAETqloRkDfaJqXhW8DZ20zYkloGRgqpwH0r12nLfdL5WA
####koIifKWWhAQTOuqBIRKC0UpoSq40o7M8Z2lXV07RC5XqqTV7L8jfQssnSm5Ry8V
####29EUg/yDUZJlQ9cJq28TizJFq0ww/NelwMGxtMsJG0okMAElAYYnAr4D5qOATaBa
####7AK51ZxrAjrBn+aEDe7IVLWhNmA6KxDFogSsbaQ5qDKwnhtNhRdM6ai0oanr5i3q
####CgyKq91mJgLTtRtSpe30rhnRtB7U7AhltqiJQHeRsHIJtgOPXPUGDcN0tAET8FHR
####gRjSmLJl4s32CTWuK9oIRie+I8AwoBSvIyCqOhnQTfSFcBGRjGdzjdiJRgMEX5a
####BKyyWwPqi2l0ts9FiphoQICBZm2qsKVYx1BHFPrUMIGNQh/TMAVjoXYHbGLCZ2l0B
####xLG8LmFDPjpjOkBLQLE4N9MCKxkpd+SSomWBhCahhgpvUyQI/RyZDnVhBrQHRqQC
####DDfe/Han09C5RfNwbW7KZ2xpaIoW2prBqKDh2XZQyP6HVo/0Ou/6nxrdJoHXp930
####r63j5jF5+xkeNsLR5/Rzt/X+Q5986JwcN7s90mgfw7vtfrf19qzf6faQzL/+1ehB
####650d9rTR/kya/zztNns90umSlsfTkxZQBBDrrvfavZAAe2jk7PjVvt9gQAV0u70
####kcxJ62OrDzX7nQJjPt+SdN6Rj83u0Qf4tfG2ddLqf2Ys37X6bWT3rtNFQg1y2uj2
####W0dnJ40uOT3rnnZ6TYISHrd6RyeNlsfmsQR9AL6k+Wuz3Se9D42Tk5DAnU/tJqMG
####MgQFJm+b0NPG25MmsmPSHre6zaM+iJv9dQRqHE6eFEjvtHnUghdIq/nPJgJv6H4u
####uGR7zf87g3rwnBw3Pjbeg4y5ee0EVCMAu9s5Ous2P2LfQSW9s7e9fqt/1m+S9530
####MVN7r9n9tXXU7B2Qk06PKe6slywAk34DebtUQHfQA6q/Peu1mAph7X6z2z077bc6
####7TyM+ifQEPS0Aa2Pma47bSYzKKvT/Qx0+ci5o1Egnz404VEX1cu01kB19EB7R/lg
####NWAJyuwHhEU67eb7k9b7ZvuoiRU6SOhTq9fMw+CloH/vkSoy/9QAzmdMdhW06Bt/
####2WLM6Bl0gY0uab0jjeNfW9h/tz7YQ6/lmg9T39EHV/tSJoMgj3gwcA7Y6zbgbv24
####3TvIZEZ3ZBv8BDn036zXu9Q2dXTQPpxv09iCDz+ElznrmT+3cDgRlUhn+VHbyB4S8
####BAAYMceIsxWmPPLo9HZwcg+Vie64EYedod80B/BBM3JvGt33v+ahAy/XWjIze3LJ
####OJA/MxkCBf09Cu1Q2xlgSJBjb2P5bafyWgYpZJCjQLKt05sauYVqHEy1gGpA1QkE
####Caf9LsNr9pZEzortBryV/VKIIFVLJgVACXiaitrreWqo44j6r6TdilQplwOyQAS0
####qiw+tVoytVTi+ARfzxOMLEiGxKJEPn77ql4HibDF3uqDEyRWTqSWQqAAQZUqav2S
####0qFPdyoSyYEzBF8IjutGGdxhOCvls1/yrmligUeGk8sub+rnXrmRJZJPiJcs2n4d
####JvM3mIkoINo8eIZTDHphIttlAo3IG38+/Fb+Ar9nF9DqezNHpfbA0sYU8xRqXhU
####cE57Bly5d0NypFKQCcgdpO+Kzrko+pUJjv16RG4UiGoMh7OAtoxuiLjXgPobkAow
####5pAF1BeYlFxfG0H3Bw+mLspybEbawfQOEwnTcrpwbvDOcJAtyYsnyHoUJgchUGVwz
####sS2SczsXEjxSAaYFoRC0lMIRgUd0yH4BxZKzbou9lr9EM/lrXjG+QOH6f0UNwbQa
####f3wPo5wloyzL5uAvRpwQCUqc01+bwf3YsSJc/y9hTo8nYFna2JsOhXnDI7kKBqty
####niiDAYwKy/NMH1YjA5cUTHFGyzYtZhwUy4K4FwwmR35jQ1nAIaQWhSy6MLG0LwUi
####SRLJ86EC88hta2MwRs4zDw74zcWB98w1m4PgYMQKV/c77lGDSa/rGFNjEH9KgFFg
####+qASXGMDqSxaRG8NjhnSSBB0ODEGPEG5hbrGv1XuIBmnzoClFW77XqcxYtkDwq4K
####sTlkZzomI+T2mrL8AtoafDqgQTbuWYCTy2qk37lEIIIdk2nG8yJ+yOJ+BL/7CNeCZ
####jg3spKnqyDaAK4zdNu8Hak83za+T8QUEMBfw7AK6duH1EzXtSR+cnwyg3zBSeTKd
####oeOJfelN3EKA7BnzAzmsX9j5HvdS6pj15+z8L9s7+aJ5s2DirkyH4ZsNQ6VY3oDP
####zh2c9zZXD2PtygiwZDDn4qJUyGy1YY5s+cSCQsET3zoOD8G4I6F4QYfnp4kMYZlO
####cXFGIvPOg0nwsd342IRkELV07GucKQZYmQvtfPB8txFsyZ8d/s8u/S799Mu5VCod
```

```
###Ln1flxsYIwCdhsSJzjKbwiHVbXp/pVTSKkXjSwC2AzNssT4C6J5kmoxCvHGuzTQT
###zTNC0/G2ma5b0+XKsWUOqDoBkxoqGuBc1Ii6SrOoM7GMKZ5uyPEsAB8vD3lJHOUr
###JFoY+pYgSpXGZO5j3lUbwC2Hs7tSwP9l12uQAuFmw9QNTeOOJOW/xo5l4cKD/+a2
###bSoXrE8XnjpccliPFiEj/pOp4BMAL5ulW6zNYRpdInLsxh/+Brtg2+BsDl3xMvgp4
###gJB95yd/4GUarG6rLRHSGnrq8NagUHm4RPYVKQb0Aok0pgnMh93s5XEpjeISGFHG
###YAFgMOByXFpofX9M0DZuNV3HroyphYtluPjJq9gmeiXVNHyc7pugW5d8IW0ALoe5
###P6AN2taMIvZAUqyx4lLF5Nm4ygSxfgGQlwm8G+gTumHfB3uDoRj2LfA+JDy9tiE4
###YhEB2YGGLEdqe9VnYkoLvR9vXPzZJRICeQxV/JkNEf2D04sJvRcKgTiAQmQlRg9M
###SsUHUgRcc1wKGF2BzLQhEfOTD4tKLyfgD+YJZCdJx7w1J0OrXdSVS2liaEXbmTjO
###lWI5kkEd3+m9b/ah03VUU47ryJwY6mGFHLgv/34o+6//9regKtAr8tqHlRU0pBiK
###fvcfMNsJZHgzB8BQKIueFvmYaOBbyId+G+vaQHPYxI+xKa8sMCxoPwTYAaCmsrOp
###mXnyxad2ws4PUb/cwflmib6x253v/LzxYp/DxpuWM8df13xHcbbLlBaGxsNpm/na
###Oi5ccKuLy6lmmFCavnL0RxBCTQWoe144MDeY+ngAiGDMA4lQsxygrwbRT25OtPwv
###WSVbzxpmNs8IoUlxTXFYUw25zhMSmiB3SGTmaJLN5XRzUc7UofJHmHmH3rrrNBgv
###2yZuzPjaCWZx3XdHpPaq9mo2H1TB77LQCRpdTVDfMOWQEG66+FtZ3e6OzZYyNaB+
###rdhIb6RAbrS2jJD3YdvlAcq9gn7E5YYxtsWlVCEzRFj80/vWdKDjiaBiuVoDsDMb
###20U4M1bXk2aHtZ8FHg2UsuW2mkm9YjtimG4PuPR1Nq2RF7Us0+L+OTy7w8FfYEou
###6lKnoGRQT6JEh+GPxFdtDAIYvoGiPdY9YEKqUbhk0G9O1A+J54MOM7M4z2Zn0ADL
###v03Ny0lkSXanwMXhNuk66q1l8jeGunIFXjxq2Yo9d03Jrc0Nyy9lrCsDdmQgnwFJ
###wquCoM6tXJDI4f9KIdMs5ZdSboEo04npexrvjcgkDCJBCMYN0+EzODxbGJHQe5GJ
###atwooSebDHJTxeBRcrJn2eRIbE4y1h4nMHb3Pht2vkQ/XvqIVyA90iA1aQ/yM/00
###UlfdvLVJdmDi+Q4H4dIfytiyL7wM0AMTmgbfANiLK/nL3V0rHCLKAEJ4xYcw8JeKd
###B2FYyoAXsmZ9AiRwQYqvGyFgFzBQ//eErV0bLGsKEOFLQ35PGVHUEcsm6C2n6NfH
###6Q1IO5OEbpFcDt/MHwZMF0zw9/Otc+n8p/OtnPRT/nxru5RfbqhACmCuqcSbDHgU
###BJTPdQ96X9cUx9Vglga6Hq/OZJylPrNQ8FtguodnOzb9kp/BxpfMgZHL0+IiBKA1
###/QM3Ry7vyLQppq+u184e8zZ+Ao8WfcQH774c/k+1L/vq//+VvVwJvw+u/Ipc5I4Se
###jY3CyxLuQkQAZDaf9H/vk17R5VIz1Nc19LqqNDCN4SZ44Cm/vVot5vxfubJbrfHz
###f7Xybrm6+6JcKe9Xxfm/BynmK/cTlFL1SDYcEwIm7KlG8UqDSDmoXhAUM0GHDKe
###TqJGEUJdKH8Sf3P/gPx1MF+peLMHlep1/hj+Zv5jomuTsgHGLFQaQZ5ErSkJdggv
###W6L0gHWhpf5Klmk6WUaFE6HfFHRHK9Fy2wbJafZYug/JQhtONuNJG1wkWo10gECw
###y7jv7v/92YQRSjICdfjcU/+9av784qhFdbZeO8e+nIbB0V4Jb2Vj1FMZq+ydC8G
###ccQ4w+89t9MUjv/xw7EOHon4X9kL4b+8V6kJ/H+Ist3vn6Bx75Vrr8rlzBt42WoT
###tsCjGnYRz9sUK0FMk6DGtWk7fHbMpsnNRKrzpeK/OoDEAs+hxjfgHQo06NIhBGfX
###Cli8AgOabeFYd7H15Rog3Uz95rcx+L7ELuX9Bml6pTjaDSVH6CMJqDID0HXA0qfd
###/eo+6Td7/WK72S/KJHCcqVySax1fle0eYaqOUOchTtX+FBqO6Dry4joRwxpdJ0Qn
###U+GvCL2AsMAgkpQcXStMrBqoptp6EVRU3K0UQUXF6mzVWnzN2mzN3fiaU7M19+Jr
###7s3W3I+vuT9b83vP5uWLj/9Rrn9NPBLxv1YJ439ld1/g/0OUBPyPABwi8N9rkB7/
###K8Q7nftDgX9U1+bBP6pTEeAvF8tF0Rn5ebVE2OfVEjGfV0sEfF5tP6yQvUC9iaID
###lAy+Vp6US/DxPy6VWQOPJPzHxZ4w/u+L9Z8HKQL/p/UF/ifgP40G9H9illpIKj+y
###WW4h4bx26+QW5xKYGayZ180e7tAt73t8/HcbbQJjEvF/bv2nvF/bE/j/EGUB/rsW
###UTTSKcIHKd/qsXAA6AAcnnKueCABob1M1EaD7Rrw099oyWRCTjmSTZxvSds2CpiX
###aiuv1jbKRy3VNuy7YhQRO9v40GdhbTlt7ehllYW1U9Fefv77+L+BdR+vJMf/1RD+
###V/BKGIH/D1BS4L9rGQhLP0oIhv/uM3NMLcUx3e1A4QeYH3hDmDIR12KvmclEbt66
###lJNgPG3buPg6dVt5tbaRmwPLtJ1LQWI2oNMOK23bKGUtlVZerW2UspZqK6+w8jTF
###//Wt94RLEv5XK+H1//K+LPD/QUoa/B/vubdvReI/PvbhX2QCyR4gpM5MZ1F+z0oS
###silFIArelicg34NAFNAtT2AVtBMlXKb4PlXmunkk4P9+tVKei/+r4vzPg5SE9f/o
###lFms//MGEaCfOUCUwg/XXekT//NkYrFmYdvnuVgDpsKuVgl+wsvmd6Wa7DYy91l5
###0wKrH+MdkMYVEAhWY5ecuLekJSmcM4F0Av6xjxRl/fsebPhlS/ppy7t+FC9RCpa/
```



```
#####5H4ov6RZ6hbeT5GalZyGlxX/XnKUWAFWaxNKjhJqntFikTLhvkizRNBK/O1C4LJj
#####UNj9Djl+bSYDDKK295h9/APNg5LcTopY8kOrPMEj0XbSdFtgOgk65kR9wRMm6up8
#####ONF5PtHT8J58ZJ/PwYF3iSGo373G0L42b8klvVZuNHYrMN5+rGj6BPzKwUECmqze
#####MU400LEEJLomp6kK0DZRK9Qy+MfqwOzMiUUM6tyaltc6Sd4LP2BXddrZxeqr2mtC
#####AhfllaoySJg6bvO9WzsTu2HsTx7/gsbEulCCzGYazClulricWDdEXAYx8P083g0n
#####MTMaMH2U108FAJvCgAcMDiHXpj+OCcJsgCOoQ7V1/KA9XtuijcnYNHXv+hZDv8tr
#####4k+FhQejmok/GhauW8vEnw8L193NxB8SC9fdy8SfFAvX3YcQJ/psQMAH+Ga811g5
#####bmf1LTNRxgKCLCRsnczJUUW44ixz1qJ8X2JZwmd9TjXj4Q3j52r0ZgVnnrLzqTYHqxu
#####10hywUyK1WMDHDMslz84HDCofrgqf4TXjORLSphduE505Em9lwm/ACimizJ24TnksR
#####bfbf8S+KxV//wU9VbohH4vn/3bn1f3m/KtZ/HqIsWP9Bi3AXqqPO/0wfi+X/mOX/
#####yBNAbvEWQSK0nInSfLBdaB3hgc4bzSzHLyFN1EZvXPfmsdjdYfE93Fu22AR/BG
#####TOyGVzQv3RwoOk6ICGbrOJrzICX+a3l082pNPBbj/36lUt33vv+pWq1VX7AdYXH+
#####80HKKlesJV8XPw3rF10Hv+yXJ2Rib32H2ZjiqutK6I7raS+XvJgzZUPvMsy4dh5x
#####69iLiHc341cnQpe73adfMeYSy5T652ywaRiIfpdh9Prk4pbTe4Yqc9crKnP3JGZS
#####3bkXuoo5q1lHUHXtpzGz+2je80wMyTxn/W8PCTjYbx27+CqplSEdRDeoer3NahAHY
#####gOrKy9E9hEvzjQ6Z4BcxYFrPv4UBknsm8FiixuObnArfZIFvT6yiwDeBb88I31a4
#####Um+ZaLN2j2gz8vubNhVwllYF5ISGHiDH7oYIRL4nIicMwDND5FR2Fnsbc7vT7HY/
#####3SRUK6dFtZW/cyXqylWSR2b92kzNcs67VVbxbv6BgXX5NE5w456MmfcaFglhQ9z
#####RdpUZC4cwaOtKbzB83AE8pwjiPjqsZjhWe6rxBWKj9zfeSx5InRZ/5FEcsWEbtnd
#####pat2O5ZmFr/11Vz1/Tp9P1XLqXPndSpfT1PwbCKWksCPXSEkwy0yY12sgEGacip
#####A51kChHpfFnwnNp3mv14izXe/U31TfzFrVeO3hIb3CsAeKXoop4k6YRLInkrtmntwJ
#####cHiU4PCcA/6nE8+JLEVkKSLGf+4Bv3/AP8VBMvx8xHc7SBbo6NJOP23b2eMW859p
#####EWu6K6/pph2CVWKOZqfQES3rprG09Z24iPmA0zp8SRTpH+zERaSI609tBNA9hYoC
#####6ATQPS+g23wImuJ0WWwI+kCnywJ9XR2clzpaEP0BYghQ9wf0DRwweMIAndbaHucZ
#####gzT7Eemcz1r3I9Kx3Igg+YLLAgr4wrXBtmPLHFB1YlF2kQ5Vl3IxGwyABcY+8ooC
#####Y58TxqbelokYoR9jW+fXr9mLnSixEyW2dcS2ju+AUxz18nPqjR3lCvRm9Yju8ZzZ
#####WD0CebxnNkQC9QMmUGK6Pdbp9pxj6ScVLYkcQQQAIPz+xrH0zO3VCbH03ve8a2um
#####o0v6/fRtxY00G1o0TT8Ez+yTr+JSmqdzcuBBLt0SQPC0KgqgE0D3vIDuAULQ8j1i
#####0Ac5IzXb2XvAc0Jjfwc/8ps7BDqvAZ0TRuCWxWOyoT3ObXvxUW3xUe3v91Ft4Q6e
#####RkXhDp6JO3jOH85/QjsLYr9M7JeJnSex88R9r0oVtX5J6TBF9j89yEVyI+XukoJI
#####2o0yuAMf70j5Deb7016uHue1pCFu6BE5ocgJn35OKCDjqUDGc84bnk5YKJIdkeyI
#####vOEHzhv61wWh8bvXCTVU6bF+66kooogiiaiiiiiCKKKKKIIooooogiyyMp/w/Y2ZWt
#####APAAAA==
```

Author's Address

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

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

ALTO
Internet-Draft
Intended status: Standards Track
Expires: June 30, 2013

S. Kiesel
University of Stuttgart
R. Penno
Cisco Systems
December 27, 2012

ALTO Server Discovery based on well-known IP Address
draft-kiesel-alto-ip-based-srv-disc-01

Abstract

The goal of Application-Layer Traffic Optimization (ALTO) is to provide guidance to applications that have to select one or several hosts from a set of candidates capable of providing a desired resource. ALTO is realized by a client-server protocol.

This document establishes a well-known IP address for the ALTO service and specifies how ALTO clients embedded in the resource consumer can use it to access the ALTO service.

Requirements Language

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

Status of this Memo

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

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

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

This Internet-Draft will expire on June 30, 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. ALTO Server Discovery based on well-known IP Address	5
2.1. Well-Known ALTO Server Discovery IP Address (WkAsdIPa)	5
2.2. Well-Known ALTO Server Discovery URIs (WkAsdURI)	5
2.3. ALTO Discovery Client behavior	5
2.4. ALTO Discovery Server behavior	6
3. Deployment Considerations	7
4. IANA Considerations	8
4.1. Registration of IPv4 Special Purpose Address	8
4.2. Registration of IPv6 Special Purpose Address	9
5. Security Considerations	11
6. References	12
6.1. Normative References	12
6.2. Informative References	12
Authors' Addresses	14

1. Introduction

The goal of Application-Layer Traffic Optimization (ALTO) is to provide guidance to applications that have to select one or several hosts from a set of candidates capable of providing a desired resource [RFC5693]. ALTO is realized by a client-server protocol, see requirement AR-1 in [RFC6708]. An HTTP based ALTO client protocol is specified in [I-D.ietf-alto-protocol].

Before an ALTO client can ask for guidance it needs to discover one or more ALTO servers that can provide suitable guidance. Several algorithms have been specified that produce a suitable HTTP URI for a given ALTO client (i.e., the URI may vary for different clients or different points of network attachment, etc.). These approaches are based on user input or DHCP [I-D.ietf-alto-server-discovery], a "reverse DNS" (PTR) lookup [I-D.kist-alto-3pdisc], or redirection within the application protocol [I-D.kiesel-alto-alto4alto]. However, each of these approaches has technical or operational issues that will hinder the fast deployment of ALTO.

This document follows a different approach: it establishes a well-known address for the ALTO service (TBD: this approach could easily be generalized in order to discover other services as well. But this is for further study). All ALTO clients seeking ALTO guidance are expected to send requests to this address. It is then the duty of "the network" to direct the query to a suitable server. This (re-)directing could be done on several layers, e.g., by resolving a well-known DNS domain name to different IP addresses (DNS split horizon), or by routing IP packets with the well-known IP address to different servers. This document follows the second option, as ALTO is closely related to IP routing and routing costs.

This document specifies a procedure that can be used if the ALTO client is embedded in the resource consumer. In other words, this document tries to meet requirement AR-32 in [RFC6708] while AR-33 is out of scope. Note that AR-20 mandates that "an ALTO client protocol must be designed in a way that the ALTO service can be provided by an entity that is not the operator of the underlying IP network." Though not violating said requirement, the procedure specified here is not helpful to fulfill it.

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].

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

2. ALTO Server Discovery based on well-known IP Address

2.1. Well-Known ALTO Server Discovery IP Address (WkAsdIPa)

IANA is requested to register (see Section 4) a single IPv4 address 192.0.0.X (TBD) and a single IPv6 address 2001:YYYY::ZZZZ (TBD) within the respective Special Purpose Address Registries as the well-known IP anycast addresses for the ALTO service. These addresses are called WkAsdIPa (well-known ALTO server discovery IP address(es)) in this document.

2.2. Well-Known ALTO Server Discovery URIs (WkAsdURI)

The Well-Known ALTO Server Discovery URIs (WkAsdURI) are formed using the HTTP or HTTPS protocol identifier, the WkAsdIPa in their literal forms (for literal IPv6 addresses in URIs see [RFC2732]), and a constant suffix. That is, there are four WkAsdURIs (TBD: replace X, Y, Z with real values assigned by IANA):

`http://192.0.0.X/alto`

`https://192.0.0.X/alto`

`http://[2001:YYYY::ZZZZ]/alto`

`https://[2001:YYYY::ZZZZ]/alto`

2.3. ALTO Discovery Client behavior

ALTO Clients that need to discover an ALTO server use the HTTP GET method [RFC2616] to access one WkAsdURI, e.g. GET `http://192.0.0.X/alto` . They MUST be prepared to receive an HTTP 307 temporary redirect to the ALTO server's Information Resource Directory URI (Sec. 6.7 of [I-D.ietf-alto-protocol]).

For hosts equipped with multiple interfaces and/or using IPv4/v6 dual stack, this discovery method might yield different Information Resource Directory URIs for each interface and address family (i.e., IPv4/v6). In general, if a client wishes to communicate using one of its interfaces and using a specific IP address family, it SHOULD use this interface and the IP address associated with this interface to access the WkAsdURI of the corresponding IP address family. Selecting an interface and IP address family, as well as comparing results returned from different ALTO servers, is out of the scope of this document.

TBD: rules for retrying (timers, etc.) in case of failure.

TBD: rules for caching discovery results.

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 re-run the ALTO server discovery procedure without relying on earlier gained information.

2.4. ALTO Discovery Server behavior

ALTO discovery servers MUST listen on the WkAsdIPa on the HTTP and HTTPS ports for incoming HTTP(S) requests. They MUST answer GET requests to WkAsdURI using the 307 (Temporary Redirect) status code and redirect to an ALTO server's Information Resource Directory URI.

The ALTO discovery server MAY consider the client's address and other information when generating the reply, in order to redirect to different ALTO servers depending on the client's identity or location within the network topology.

The Information Resource Directory itself MUST NOT reside on a WkAsdIPa, and it MUST not reside on an URI that resolves via DNS to a WkAsdIPa. After issuing the 307 status code ALTO discovery serves MUST close the HTTP(S) connection.

Rationale for the requirements in the previous paragraph: The goal is to keep the TCP connection to the WkAsdIPa as short as possible. When using anycast routing, IP packets belonging to an established TCP connection could be diverted to another ALTO discovery server due to state changes in the routing protocol or due to scheduled maintenance. Keeping the connection duration as short as possible reduces the risk of stalled or aborted connections. A UDP based lookup using one query packet and one reply packet (e.g., based on httpu) would eliminate that risk. However, there seems not to be a well-standardized candidate protocol and studies [Levine2006] suggest that short-lived TCP connections work well enough with anycast routing.

TBD: do we need some URI such as `http://192.0.0.X/discovery-server-identity` in order to be able to identify the (misbehaving) discovery server that currently serves us?

TBD: how should the ALTO discovery server handle GET requests to other URIs or other HTTP methods?

TBD: should the discovery server always redirect http requests to the http URI of the information resource Directory and redirect https always to https? Or are there other reasonable scenarios?

3. Deployment Considerations

Network operators have to install one or more ALTO discovery servers as specified above. Depending on the the network deployment scenario they may use IP routing tables, HTTP proxies with URI rewriting, or other suitable mechanisms to direct GET-requests for a WkAsdURI to one of these servers.

[TBD: explain in more detail] This works fine even with cascaded access routers with NATs. After each router hop the operator may decide whether to handle the discovery requests, e.g., using a static routing table entry, or whether let them flow "automatically" towards the internet backbones using the default routing table entry.

TBD: what happens if an operator does not deploy these scheme? Requests could be dropped at administrative borders, or there could be one or several "public" default discovery servers.

[TBD: explain in more detail] The advantage of this scheme is that it does not need support in home gateways, which would harm quick deployment. This scheme also doesn't need new interfaces between the operating system and applications, e.g., for passing DHCP options from the operating system to the application.

4. IANA Considerations

4.1. Registration of IPv4 Special Purpose Address

IANA is requested to register a single IPv4 address in the IANA IPv4 Special Purpose Address Registry [RFC5736].

[RFC5736] itemizes some information to be recorded for all designations:

1. The designated address prefix.

Prefix: TBD by IANA. Prefix length: /32

2. The RFC that called for the IANA address designation.

This document.

3. The date the designation was made.

TBD.

4. The date the use designation is to be terminated (if specified as a limited-use designation).

Unlimited. No termination date.

5. The nature of the purpose of the designated address (e.g., unicast experiment or protocol service anycast).

protocol service anycast.

6. For experimental unicast applications and otherwise as appropriate, the registry will also identify the entity and related contact details to whom the address designation has been made.

N/A.

7. The registry will also note, for each designation, the intended routing scope of the address, indicating whether the address is intended to be routable only in scoped, local, or private contexts, or whether the address prefix is intended to be routed globally.

Typically used within a network operator's network domain, but in principle globally routable.

8. The date in the IANA registry is the date of the IANA action, i.e., the day IANA records the allocation.

TBD.

4.2. Registration of IPv6 Special Purpose Address

IANA is requested to register a single IPv6 address in the IANA IPv6 Special Purpose Address Block [RFC4773].

[RFC4773] itemizes some information to be recorded for all designations:

1. The designated address prefix.

Prefix: TBD by IANA. Prefix length: /128

2. The RFC that called for the IANA address designation.

This document.

3. The date the designation was made.

TBD.

4. The date the use designation is to be terminated (if specified as a limited-use designation).

Unlimited. No termination date.

5. The nature of the purpose of the designated address (e.g., unicast experiment or protocol service anycast).

protocol service anycast.

6. For experimental unicast applications and otherwise as appropriate, the registry will also identify the entity and related contact details to whom the address designation has been made.

N/A.

7. The registry will also note, for each designation, the intended routing scope of the address, indicating whether the address is intended to be routable only in scoped, local, or private contexts, or whether the address prefix is intended to be routed globally.

Typically used within a network operator's network domain, but in principle globally routable.

8. The date in the IANA registry is the date of the IANA action, i.e., the day IANA records the allocation.

TBD.

5. Security Considerations

TBD

Issue: how to deal with TLS certificates for HTTPS?

TBD: rules for filtering route at administrative boundaries

6. References

6.1. Normative References

- [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.
- [RFC2732] Hinden, R., Carpenter, B., and L. Masinter, "Format for Literal IPv6 Addresses in URL's", RFC 2732, December 1999.
- [RFC4773] Huston, G., "Administration of the IANA Special Purpose IPv6 Address Block", RFC 4773, December 2006.
- [RFC5736] Huston, G., Cotton, M., and L. Vegoda, "IANA IPv4 Special Purpose Address Registry", RFC 5736, January 2010.

6.2. Informative References

- [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]
Alimi, R., Penno, R., and Y. Yang, "ALTO Protocol", draft-ietf-alto-protocol-13 (work in progress), September 2012.
- [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-06 (work in progress), November 2012.
- [I-D.kiesel-alto-alto4alto]
Kiesel, S., "Using ALTO for ALTO server selection", draft-kiesel-alto-alto4alto-00 (work in progress), July 2010.
- [I-D.kist-alto-3pdisc]
Kiesel, S. and M. Stiemerling, "Third-Party ALTO Server Discovery (3pdisc)", draft-kist-alto-3pdisc-01 (work in progress), October 2012.

[Levine2006]

Levine, M., Lyon, B., and T. Underwood, "TCP Anycast - Don't believe the FUD. Operational experience with TCP and Anycast.", Presentation at NANOG37 <http://www.nanog.org/meetings/nanog37/presentations/matt.levine.pdf>, June 2006.

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

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

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/>

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

Email: repenno@cisco.com

ALTO
Internet-Draft
Intended status: Experimental
Expires: August 15, 2013

S. Kiesel
K. Krause
University of Stuttgart
M. Stiemerling
NEC Europe Ltd.
February 11, 2013

Third-Party ALTO Server Discovery (3pdisc)
draft-kist-alto-3pdisc-02

Abstract

The goal of Application-Layer Traffic Optimization (ALTO) is to provide guidance to applications that have to select one or several hosts from a set of candidates capable of providing a desired resource. ALTO is realized by a client-server protocol. Before an ALTO client can ask for guidance it needs to discover one or more ALTO servers that can provide suitable guidance.

This document specifies a procedure for third-party ALTO server discovery, which can be used if the ALTO client is not co-located with the actual resource consumer, but instead embedded in a third party such as a peer-to-peer tracker.

This algorithm takes a resource consumer's IP address as argument, performs several DNS lookups (for PTR, SOA, and U-NAPTR resource records), and produces URIs of ALTO servers that are able to give reasonable ALTO guidance to a resource consumer willing to communicate using this IP address.

Starting with draft-kist-alto-3pdisc-02 the algorithm has significantly changed compared to previous versions of this document, including draft-kiesel-alto-3pdisc-* and draft-ietf-alto-server-discovery-*. The new algorithm does not try "DNS tree climbing" and it does not necessarily rely on PTR records, i.e., it can also produce results if no PTR records are populated in the DNS, for example when IPv6 privacy extensions are in use.

Terminology and Requirements Language

This document makes use of the ALTO terminology defined in RFC 5693 [RFC5693].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "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 August 15, 2013.

Copyright Notice

Copyright (c) 2013 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. Third-Party ALTO Server Discovery Procedure Overview	5
2.1. Variant 1	6
2.2. Variant 2	7
3. Third-party ALTO Server Discovery Procedure Specification	8
4. Deployment Considerations and Operational Considerations	9
5. Security Considerations	10
6. IANA Considerations	11
7. References	12
7.1. Normative References	12
7.2. Informative References	12
Appendix A. Contributors List and Acknowledgments	13
Authors' Addresses	14

1. Introduction

The goal of Application-Layer Traffic Optimization (ALTO) is to provide guidance to applications that have to select one or several hosts from a set of candidates capable of providing a desired resource [RFC5693]. ALTO is realized by a client-server protocol; see requirement AR-1 in [RFC6708]. Before an ALTO client can ask for guidance it needs to discover one or more ALTO servers that can provide suitable guidance. For applications that use a centralized resource directory, such as tracker-based P2P applications, the efficiency of ALTO is significantly improved if the ALTO client is embedded in said resource directory instead of the resource consumer (see Section 4.1 of [I-D.ietf-alto-deployments]). The ALTO client embedded into the resource directory asks for guidance on behalf of the resource consumers. To that end, it needs to discover ALTO servers that can give guidance suitable for these resource consumers, respectively. This is called third-party party ALTO server discovery.

This document specifies a procedure for third-party ALTO server discovery. In other words, this document tries to meet requirement AR-33 in [RFC6708].

The ALTO protocol specification [I-D.ietf-alto-protocol] is based on HTTP and expects the discovery procedure to yield the HTTP(S) URI of an ALTO server's information resource directory. Therefore, this document specifies an algorithm that takes a resource consumer's IP address as argument, performs several DNS lookups (for PTR, SOA, and U-NAPTR [RFC4848] resource records), and produces URIs of ALTO servers that are able to give reasonable ALTO guidance to a resource consumer willing to communicate using this IP address.

To some extent, AR-32, i.e., resource consumer initiated ALTO server discovery, can be seen as a special case of third-party ALTO server discovery. For that matter, an ALTO client embedded in a resource consumer would have to figure out its own "public" IP address (e.g., using STUN [RFC5389]), and then perform the procedures described in this document. However, note that a less flexible yet simpler approach for resource consumer initiated ALTO server discovery is specified in [I-D.ietf-alto-server-discovery].

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].

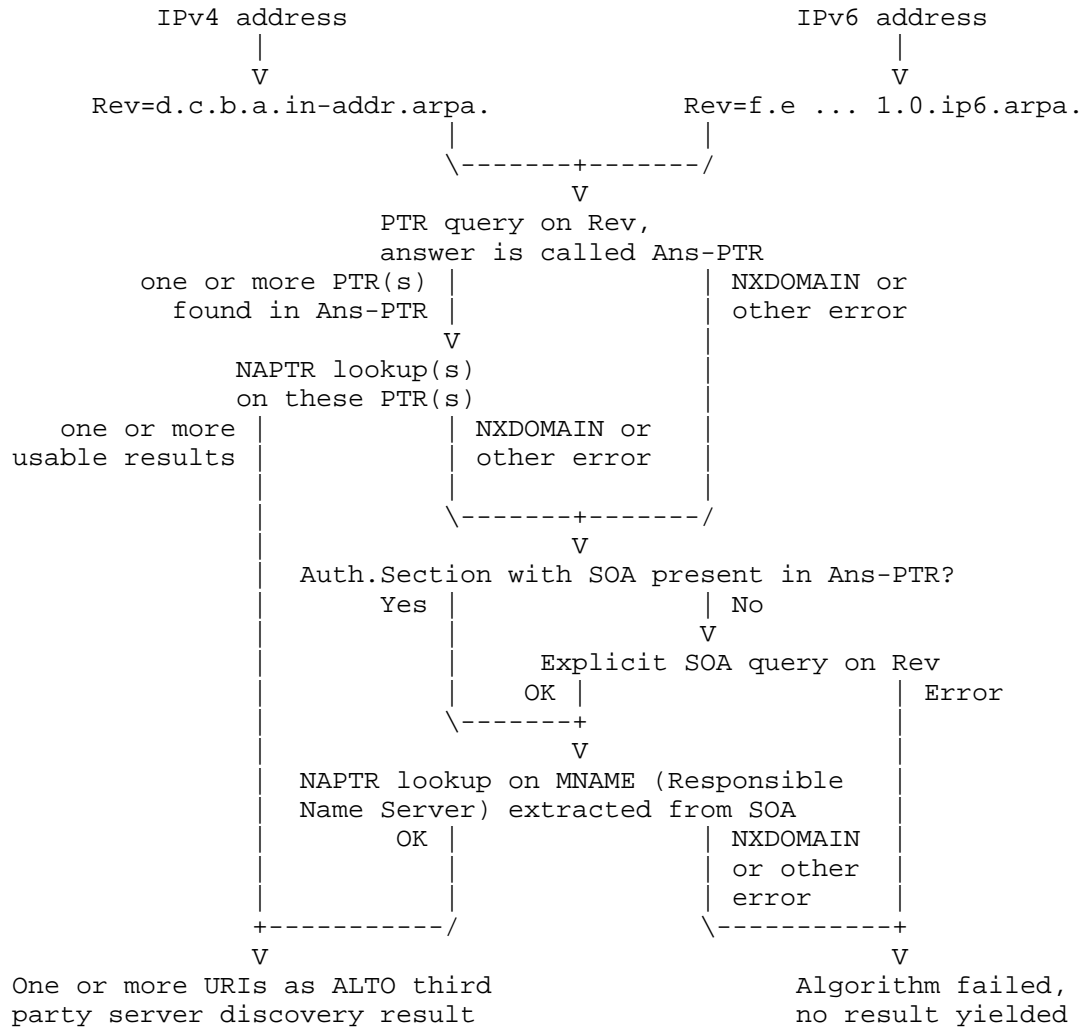
Comments and discussions about this memo should be directed to the ALTO working group: alto@ietf.org.

2. Third-Party ALTO Server Discovery Procedure Overview

There are currently two different algorithms for ALTO third-party server discovery, which only differ slightly in respect to handling of the MNAME. The following two figures give an overview on these algorithms. For a detailed specification of the U-NAPTR lookups, see Step 2 in [I-D.ietf-alto-server-discovery].

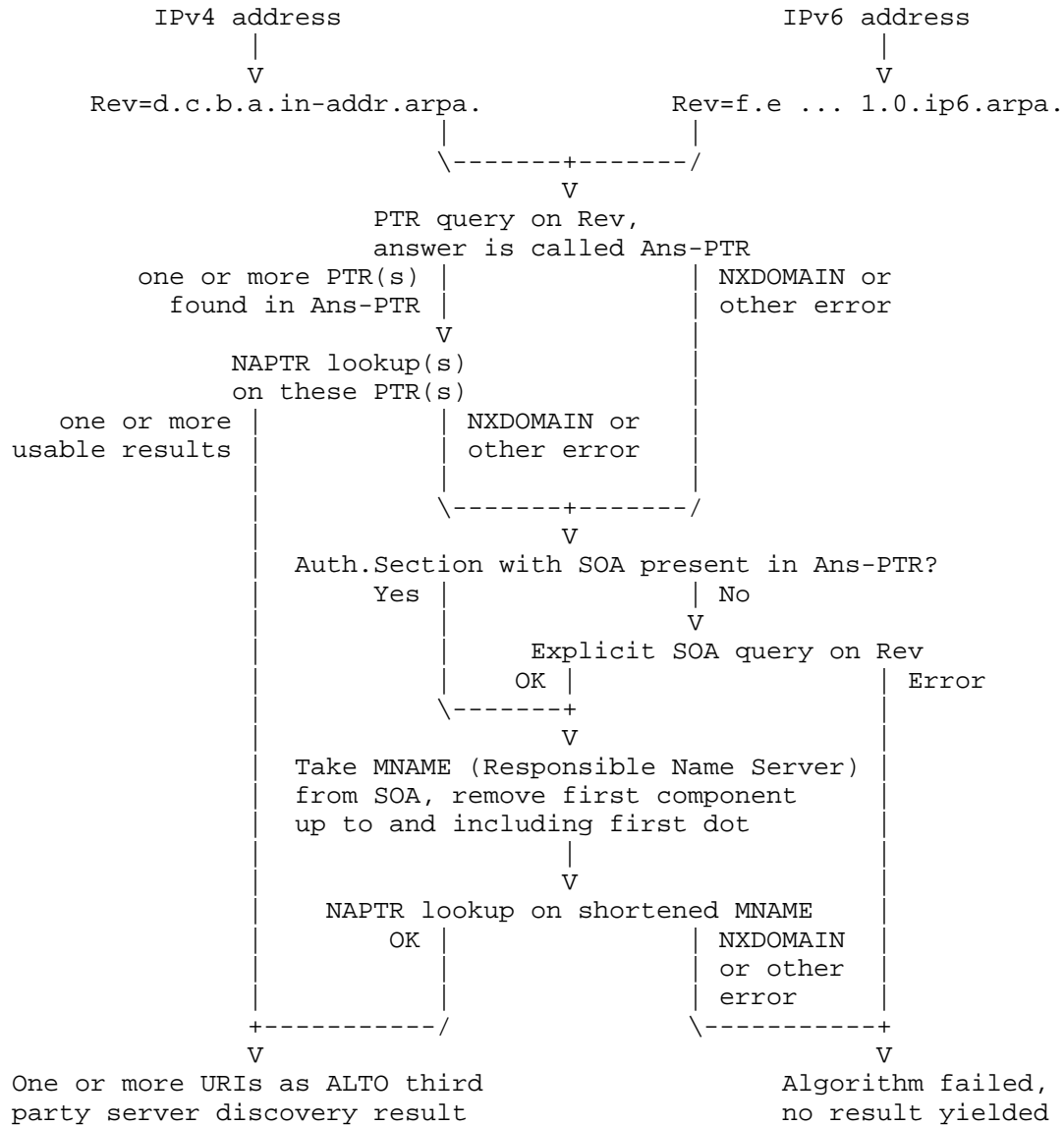
NOTE: only one of these algorithms will eventually be specified as the official ALTO third-party discovery algorithm. An analysis of their advantages and drawbacks, respectively, and a decision is for further study.

2.1. Variant 1



Note: for a detailed specification of the NAPTR lookups see Step 2 (Section 3.2) of draft-ietf-alto-server-discovery-07.txt

2.2. Variant 2



3. Third-party ALTO Server Discovery Procedure Specification

TBD.

The third-party ALTO server discovery procedure is performed in two steps:

1. A DNS domain name is yielded, by means of a DNS PTR lookup on the resource consumer's IP address (or the "public" IP address of the outermost NAT in front of the resource consumer). This IP address is the source address of application protocol messages arriving at the resource directory. If no PTR can be found, a SOA lookup is performed instead, and the MNAME (Responsible Name Server) entry is extracted from it.

The two proposed algorithms differ slightly: Variant 2 chops off the first component from the MNAME, i.e., nameserver.domain.tld becomes domain.tld before the U-NAPTR lookup is performed.

2. This DNS domain name is used for an U-NAPTR lookup yielding the URI. Further DNS lookups may be necessary to determine the ALTO server's IP address(es).

Note: Step 2 is identical to Step 2 specified in [I-D.ietf-alto-server-discovery], which already specifies two alternatives for its Step 1. Therefore, it is possible to merge both documents to one specification with three alternatives for Step 1 and a common Step 2.

4. Deployment Considerations and Operational Considerations

Individual PTR records per IP address and corresponding individual ALTO U-NAPTR records for these names allow very fine-grained discovery of ALTO "entry point" URIs on a per-IP-address basis. If an operator considers this procedure too cumbersome, or if IPv6 privacy extensions is to be used without dynamic DNS updates, setting up SOA records in the in-addr.arpa. or ip6.arpa. subdomains plus setting up corresponding ALTO U-NAPTR records will also give reasonable, yet less fine-grained results.

Note that the ALTO server discovery procedure is supposed to produce only a first URI of an ALTO server that can give reasonable guidance to the client. An ALTO server can still return different results based on the client's address (or other identifying properties) and/or redirect the client to another ALTO server using mechanisms of the ALTO protocol (see Sect. 6.7 of [I-D.ietf-alto-protocol]).

We assume that if two organizations share parts of their DNS infrastructure, i.e., have a common SOA record in their in-addr.arpa. or ip6.arpa. subdomain(s), they will also be able to operate a common ALTO server, which may do redirections if desired or required by policies.

5. Security Considerations

TBD. See also [I-D.ietf-alto-server-discovery].

6. IANA Considerations

None.

7. References

7.1. Normative References

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

7.2. Informative References

- [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]
Alimi, R., Penno, R., and Y. Yang, "ALTO Protocol", draft-ietf-alto-protocol-13 (work in progress), September 2012.
- [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-04 (work in progress), July 2012.
- [RFC4848] Daigle, L., "Domain-Based Application Service Location Using URIs and the Dynamic Delegation Discovery Service (DDDS)", RFC 4848, April 2007.
- [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.
- [RFC5693] Seedorf, J. and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement", RFC 5693, October 2009.
- [RFC6708] Kiesel, S., Previdi, S., Stiemerling, M., Woundy, R., and Y. Yang, "Application-Layer Traffic Optimization (ALTO) Requirements", RFC 6708, September 2012.

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.

This memo borrows some text from [I-D.ietf-alto-server-discovery], as the 3pdisc was historically part of that memo. Special thanks to Michael Scharf and Nico Schwan.

Martin Stiernerling is partially supported by the CHANGE project (<http://www.change-project.eu>), a research project supported by the European Commission under its 7th Framework Program (contract no. 257422). 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 CHANGE project or the European Commission.

Authors' Addresses

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

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

Kilian Krause
University of Stuttgart Information Center
Allmandring 30
Stuttgart 70550
Germany

Email: schreibt@normalerweise.net
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>

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

January 8, 2013

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

draft-lee-alto-app-net-info-exchange-01.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 July 8, 2011.

Copyright Notice

Copyright (c) 2013 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.....	13
8.1. Informative References.....	13
Author's Addresses.....	14
Intellectual Property Statement.....	14
Disclaimer of Validity.....	15

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.

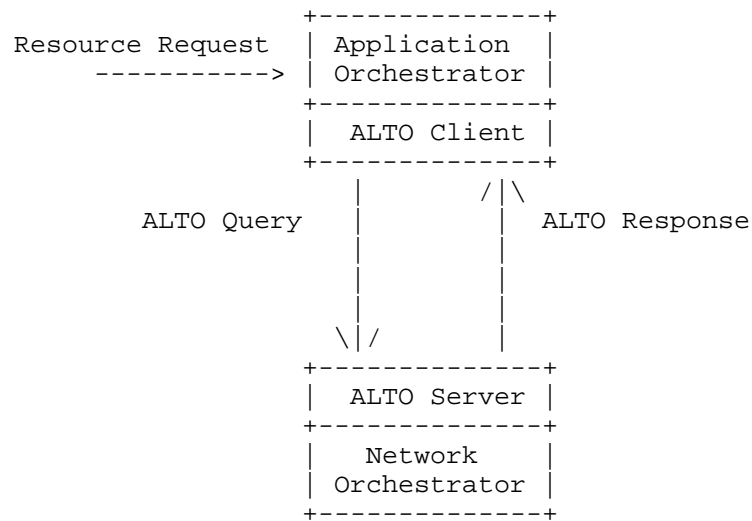


Figure 1 ALTO Client-Server Architecture for Application-Network information Exchange

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.

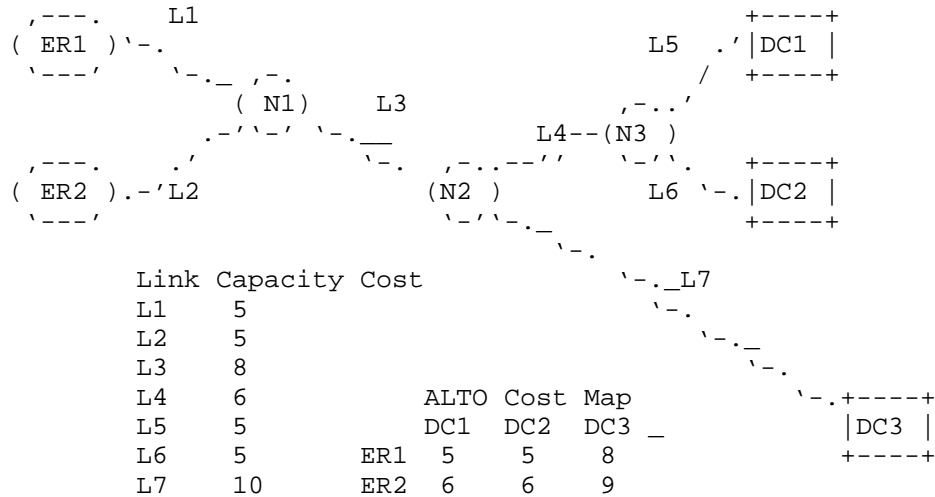


Figure 2. Example network illustrating bottlenecks

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:= 'routingcost' as defined by base specification. Additional cost (ex. latency, hopcount) are defined in [MultiCost].
- . Cost Mode := {summary, graph} /* the cost map can be either a summary form or a graph form */
 - o Cost Mode: summary

This cost mode is indicated by string 'summary'. This mode indicates that the returned costs contain constraints values which can be used by application stratum for better selection of resources.
 - o Cost Mode: graph

This cost mode is indicated by string 'graph'. [TBD]
- . Constraints /* a set of constraints that apply to the requested path summary or graph for filtering. For instance, constraints can be like bandwidth greater than 'x', latency less than 'y', hopcount less than 'z', packetloss less than 'a' etc. */
- . Objective-function: The summary or the graph should be computed based on optimizing which parameter - IGP cost, latency, residual bandwidth, etc. This is for future use.

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 */

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

Alto query:

```
object {
  CostMode          cost-mode;
  CostType          cost-type;
  JSONString        constr<0..*>;          [OPTIONAL]
  JSONString        ObjectiveFunction <0..*>; [OPTIONAL]
  EndpointFilter    endpoints;
} CsoReqEndpointCostMap;
```

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

Alto response:

```
object {
  JSONNumber hopcount;
  JSONNumber latency;
  JSONNumber pktloss;
  JSONNumber cost;      /* the cost/rank is determined based on
the filtering done using Objective function parameter and mode*/
} DstCostsConstraints;

object CsoEndpointDstCosts {
  DstCostsConstraints[TypedEndpointAddr];    ...
};
object {
  CsoEndpointDstCosts [TypedEndpointAddr]<0..*>;
  ...
}
```



```
    } CsoEndpointCostMapData;  
  
    object {  
        CostMode            cost-mode;  
        CostType            cost-type;  
        CsoEndpointCostMapData map;  
    } CsoInfoResourceEndpointCostMap;
```

For each source endpoint, a CsoEndpointDstCosts object denotes the associated "cost + constraints" 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. This can be viewed as a two dimensional array which holds cost + constraints value.

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" : "summary",  
    "cost-type" : "routingcost",  
    "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" : "summary",
    "cost-type" : "routingcost",
    "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
Sreekanthm
Huawei
Dhruv Dhody
Huawei
Taesang Choi
ETRI

January 9, 2013

ALTO Extensions for Collecting Data Center Resource Information

draft-lee-alto-ext-dc-resource-01.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

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.

Table of Contents

1. Introduction.....	3
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 (Srikant, please review this section thoroughly).....	7
4.1. Pull Based Query/Response.....	7
4.2. Push Based Query/Response.....	8
5. Security Considerations.....	9
6. IANA Considerations.....	9
7. References.....	9
7.1. Informative References.....	9
Author's Addresses.....	10
Intellectual Property Statement.....	10
Disclaimer of Validity.....	11

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.

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

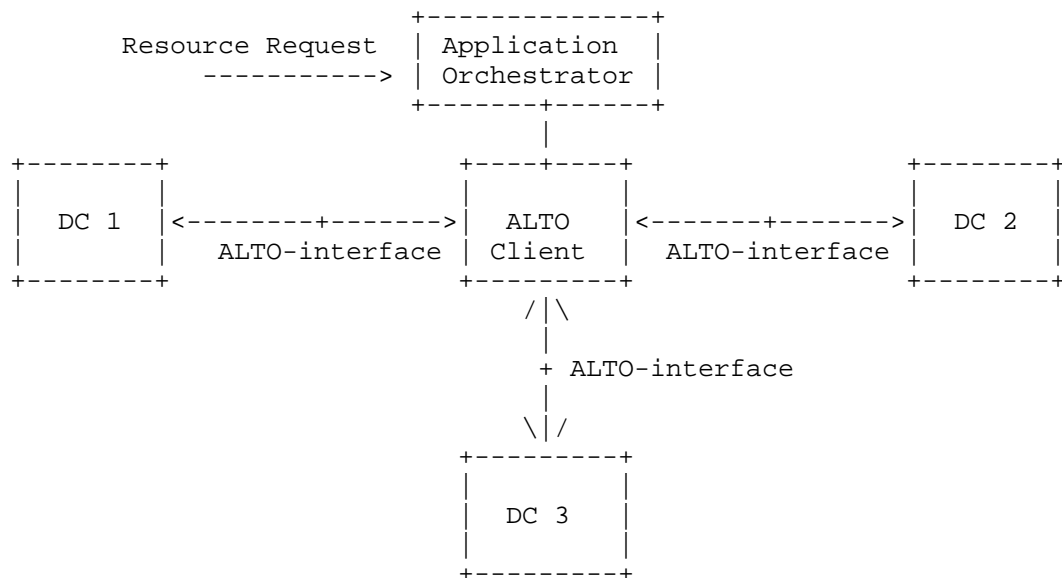


Figure 1 ALTO Architecture in Distributed Data Center Networks

Figure 1 shows that ALTO Client can establish an ALTO-interface to each data center 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.

[TO DO: 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.

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.

[To do: 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. 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 (Srikant, please review this section thoroughly)

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;
  JSONNumber  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;  
  JSONNumber  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

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

Young Lee
Huawei Technologies
5360 Legacy Drive, Building 3
Plano, TX 75024 USA
Email: leeyoung@huawei.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.

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>

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: April 22, 2013

S. Randriamasy, Ed.
Alcatel-Lucent Bell Labs
N. Schwan
October 19, 2012

ALTO Cost Schedule
draft-randriamasy-alto-cost-schedule-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. The ALTO problem statement [RFC5693] considers typical applications as file sharing, real-time communication and live streaming peer-to-peer networks. Recently other use cases focused on Content Distribution Networks and Data Centers have emerged.

The present draft proposes to extend the cost information provided by the ALTO protocol. The purpose is to broaden the decision possibilities of applications to not only decide 'where' to connect to, but also 'when'. This is useful to applications that have a degree of freedom on when to schedule data transfers, such as non-instantaneous data replication between data centers or service provisioning to end systems with irregular connectivity. The draft therefore specifies a new cost mode, called the "schedule" mode. In this mode the ALTO server offers cost maps that contain path ratings that are valid for a given timeframe (e.g. hourly) for a period of time (e.g. a day). Besides the functional time-shift enhancement providing multi-timeframe cost values, the ALTO Cost Schedule also allows to save a number of ALTO transactions and thus resources on the ALTO server and clients. Last, guidance to schedule application traffic can also efficiently help for load balancing and resources efficiency.

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 April 22, 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. Use cases for ALTO Cost Schedule	5
2.1. Bulk Data Transfer scheduling	5
2.2. Endsistemas with limited connectivity or access to datacenters	6
2.3. SDN Controller guided access to application endpoints . . .	8
3. ALTO Cost Schedule extension	9
3.1. Cost Schedule Attributes	10
3.1.1. ALTO Cost-Mode: Schedule	10
3.2. ALTO Capability: Cost-Scope	10
3.2.1. Example of time scope for a cost schedule	11
3.3. Example of scheduled information resources in the IRD . . .	11
3.3.1. Example scenario and ALTO transaction with a Cost Schedule	14
4. IANA Considerations	15
4.1. Information for IANA on proposed Cost Types	16
4.2. Information for IANA on proposed Endpoint Properties . . .	16
5. Acknowledgements	16
6. References	16
6.1. Normative References	16
6.2. Informative References	17
Authors' Addresses	17

1. Introduction

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

The ALTO protocol therefore [ID-alto-protocol] specifies a Network Map, which defines groupings of endpoints in a network region (called a PID) as seen by the ALTO server. The Endpoint Cost Service and the Endpoint (EP) Ranking Service then provide rankings for connections between the specified network regions and thus incentives for application clients to connect to ISP preferred endpoints, e.g. to reduce costs imposed to the network provider. Thereby ALTO intentionally avoids the provisioning of realtime information (cmp. ALTO Problem Statement [RFC5693] and ALTO Requirements [RFC5693]), as "Such information is better suited to be transferred through an in-band technique at the transport layer instead". Thus the current Cost Map and Endpoint Cost Service are providing, for a given Cost Type, exactly one rating per link between two PIDs or to an Endpoint. Applications are expected to query one of these two services in order to retrieve the currently valid cost values. They therefore need to plan their ALTO information requests according to the estimated frequency of cost value change. In case these value changes are predictable over a certain period of time and the application does not require immediate data transfer, it would save time to get the whole set of cost values over the period in one ALTO response and using these values to schedule data transfers would allow to optimise the network resources usage and QoE.

In this draft we introduce use cases that describe applications that have a degree of freedom on scheduling data transfers over a period of time, thus they do not need to start a transfer instantaneously on a retrieved request. For this kind of applications we propose to extend the Cost Map and Endpoint Cost Services by adding a schedule on the cost values, allowing applications to time-shift data transfers.

In addition to this functional ALTO enhancement, we expect to further gain by gathering multiple Cost Values for one cost type as one Cost Map reporting on N Cost Values is less bulky than N Cost Maps containing one Cost value each, in addition to reducing N ALTO transactions to a single one. This is valuable for both the storage

of these maps and their transfer. Similar gains can be obtained for the ALTO Endpoint Cost Service.

The remainder of this draft first provides use cases that motivate the need for a 'schedule' cost mode. It then specifies the needed extensions to the ALTO protocol and details some example messages.

2. Use cases for ALTO Cost Schedule

This section introduces use cases showing the benefits of providing ALTO Cost values in 'schedule' mode. Most likely, the ALTO Cost Schedule would be used for the Endpoint Cost Service where a limited set of feasible non real time application Endpoints is already identified, they need to be accessed neither simultaneously nor immediately and their access can be scheduled within a given time period. The Filtered Cost Map service is also applicable as long as the size of the Map is manageable. An ALTO Cost schedule can be used in several ways:

- o the ALTO Server may provide values on past time periods that can be interpreted as historical experience and used to anticipate future cost values in order to schedule transfers of application data or services,
- o the ALTO Server may provide values on present or future time periods that can be interpreted as predictions on cost values and used to schedule transfers of application data or services,
- o the ALTO Server may provide values on time periods covering the past, present and future and logically be all interpreted as predictions and used to schedule transfers of application data or services.

2.1. Bulk Data Transfer scheduling

Some CDNs are prepopulating caches with content before it actually gets available for the user and thus there is a degree of freedom on when the content is transmitted from the origin server to the caching node. Other applications like Facebook or YouTube rely on data replication across multiple sites for several reasons, such as offloading the core network or increasing user experience through short latency. Typically the usage pattern of these data centers or caches follows a location dependent diurnal pattern.

In the examples above data needs to be replicated across the various locations of a CDN provider, leading to bulk data transfers between datacenters. Scheduling these data transfers is a non-trivial task

as the transfer should not infer with the user peak demand to avoid degradation of user experience and to decrease billing costs for the datacenter operator by leveraging off-peak hours for the transfer. This peak demand typically follows a diurnal pattern according to the geographic region of the datacenter. One precondition to schedule transfers however is to have a good knowledge about the demand and link utilization patterns between the different datacenters and networks.

While this usage data today already is gathered and also used for the scheduling of data transfer, provisioning this data gets increasingly complex with the number of CDN nodes and in particular the number of datacenter operators that are involved. For example, privacy concerns prevent that this kind of data is shared across administrative domains. The ALTO Cost Schedule specified later in this document avoids this problem by presenting an abstracted view of time sensitive utilization maps through a dedicated ALTO service to allow CDN operators a mutual scheduling of such data transfers across administrative domains.

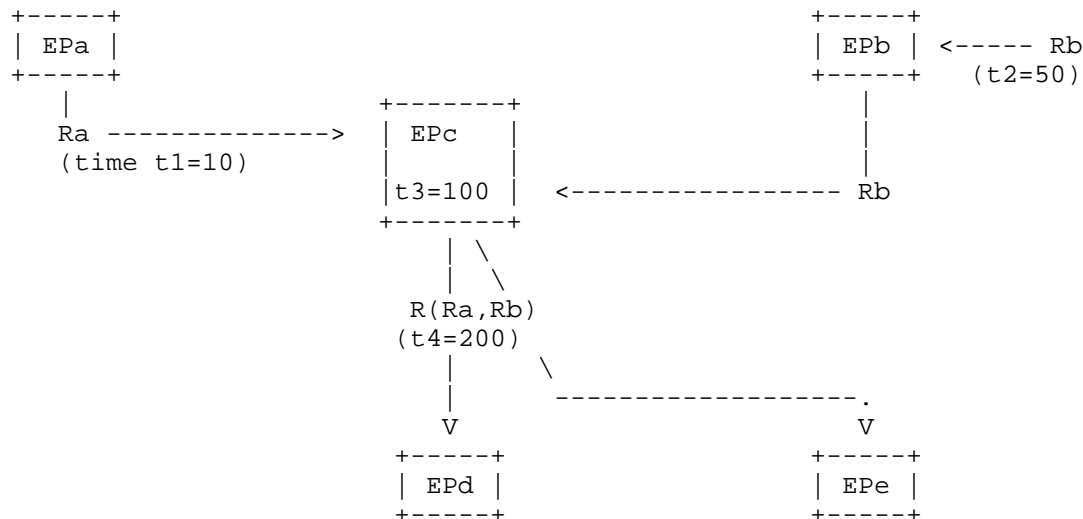
2.2. Endsistemas with limited connectivity or access to datacenters

Another use case that benefits from the availability of multi-timeframe cost information is based on applications that are limited by their connectivity either in time or resources or both. For example applications running on devices in remote locations or in developing countries that need to synchronize their state with a data center periodically, in particular if sometimes there is no connection at all. Example applications is enterprise database update, remote learning, remote computation distributed on several data center endpoints.

Wireless connectivity has a variable quality or may even be intermittent. On the other hand, the connectivity conditions are often predicable. For non real time applications, it is thus desirable to provide ALTO clients with routing costs to connection nodes (i.e. Application Endpoints) over different time periods. This would allow end systems using ALTO aware application clients to schedule their connections to application endpoints.

Another challenge arises with end systems using resources located in datacenters and trading content and resources scattered around the world. For non-real time applications, the interaction with Endpoints can be scheduled at the time slots corresponding to the best possible QoE. For instance, resource Ra downloaded from Endpoint EPa at time t1, Resource Rb uploaded to EPb at time t2, some batch computation involving Ra and Rb done on EPc at time t3 and results R(A,B) downloaded to EPd and EPe at time t4. Example

applications are similar to the ones cited in the previous paragraph.



These examples describe situations where a client has the choice of trading content or resources with several Endpoints and needs to decide with which Endpoint it will trade and at what time. For instance, one may assume that the Endpoints are spread over different time-zones, or have intermittent access. The ALTO Schedule mode specified below allows these clients to retrieve Endpoint cost maps valid for a certain timeframe (e.g. 24 hours), and get a set of values, each applicable on a (e.g. hourly) slot. Thus the application can optimize the needed data transfer according to this information.

Last the ALTO Cost schedule is beneficial to optimizing ALTO transactions themselves. Indeed, let us assume that an Application Client is located in an end system with limited resources and/or has an access to the network that is either intermittent or provides an acceptable QoE in limited but predictable time periods. In that case, it needs to both schedule its resources demanding networking activities and its ALTO requests. Instead of having to figure out when the cost values may change and having to carefully schedule multiple ALTO requests, it could avoid this by relying on Cost Schedule attributes that indicate the time granularity, the validity and time scope of the cost information, together with the time related cost values themselves.

Suppose that for some Cost Types, the ALTO cost values are available

in the "schedule" mode. If the values of Cost type 'routingcost' and/or another time-sensitive Cost Type named for example 'pathoccupationcost' are available in the "schedule" mode for the 24 following the last update, the ALTO Client embedded in the Application Client may query ALTO information on 'routingcost' or 'pathoccupationcost' for these 24 hours, and get a set of values, each applicable to an hour slot. If appropriate Cost Attributes are provided together with the cost values, the Application client also knows the date of their last update. An example ALTO transaction is provided later in this draft.

2.3. SDN Controller guided access to application endpoints

The Software Defined Networking (SDN), see [sdnrg], is a model that attempts to manage and reconfigure networks in a more flexible way in order to better cope with the traffic challenges posed by nowadays resources greedy applications. To this end, one option is "moving the control plane out of the network elements into "controllers", see [SDN charter, <http://www.1-4-5.net/~dmm/sdnrg/sdnrg.html>], that implements the network control and management. The SDN Controllers are deemed to gather the network state information and provide it in an abstracted form to SDN aware applications while gathering their requirements in QoE and exchanging other application "management" information and commands.

The relevance of ALTO to perform a number of SDN functions has been recently highlighted. An ALTO Server can assist an SDN Controller by hosting abstracted network information that can be provided to SDN aware applications via an ALTO Client. It can also assist other SDN Control operations using information in and outside the ALTO scope.

In particular, [article-gslh-alto-sdn] identifies SDN Controller functions that ALTO is well suited to perform: the primitives of Abstraction, Get network topology, Get network resources and Event notification. Additionally, the interaction between ALTO and SDN has been investigated in [draft-xie-alto-sdn] to provide applications with a path selection meeting QoS requirements on bandwidth and delay.

Currently, the base ALTO protocol allows to perform the following SDN services, see [article-gslh-alto-sdn]:

1. Abstraction: through aggregation into PIDs, ranking and a generic cost type.
2. Get network topology: through the Map and the Cost Map Services

3. Get device capabilities: through the Endpoint Property Service.

Another SDN primitive "Get network resources" provides applications with informations allowing them to evaluate the expected QoE. QoE related information includes delay and bandwidth at the application endpoints as well as on the network paths. Such information may be provided via the ALTO Service by proposed extensions of the ALTO protocol that define new ALTO Cost Types allowing to abstract and report QoE to applications.

One key objective of an SDN controller is the ability to balance the application traffic whenever possible. For non real time applications, data and resources transfer can be time shifted, resources availability may often be predicable and last, strong incentives for applications to time shift their traffic may be given by network operators appropriately setting routing cost values at different time values, according to their policy to cope with network occupation over time.

To achieve this objective, the SDN controller can:

1. get the network state history from its controlled network elements through its southbound API
2. possibly derive an estimation or a prediction of these values over given time frames
3. store their abstraction in an ALTO Server in the form of ALTO Cost Schedule values defined for different time periods
4. deliver these values to the SDN applications via the ALTO Endpoint Cost Service, either as history or prediction or as estimations covering both the past and the future.

This way:

- o One one hand, the applications get the best possible QoE, as they can pick the best time for them to access one or more Endpoints,
- o One the other hand the SDN controller achieves load balancing as it may guide the application traffic so as to better distribute the traffic over time, and thus optimize its resources usage.

3. ALTO Cost Schedule extension

One example of non-realtime information that can be provisioned in a 'schedule' is the expected path bandwidth. While the transmission

rate can be measured in real time by end systems, the operator of a data center is in the position of formulating preferences for given paths, at given time periods of given time scales, for example to avoid hotspots due to diurnal usage patterns. The entity managing the ALTO Server values can decide to integrate path bandwidth in the ALTO 'routingcost' metric. However to better highlight the purpose of the cost schedule the remainder of this document will use a Cost Type named 'pathoccupationcost' and assumed to report an abstracted form of available bandwidth. A definition and usage of such a Cost-Type is proposed in [draft-randriamasy-multi-cost-alto].

The usage of a time related cost is rather proactive in that it can be used like a "time table" to figure out the best time to schedule data transfer and also anticipate predictable events including predictable flash crowds. An ALTO Cost Schedule should be viewed as a synthetic abstraction of real measurements that can be historic or be a prediction for upcoming time periods.

3.1. Cost Schedule Attributes

Specifications on the cost "schedule" are proposed here and will be completed in further versions of this draft.

3.1.1. ALTO Cost-Mode: Schedule

The "schedule" mode applies to Costs that are eligible for a single-valued Cost Mode and can also be expressed as such. In that sense, when the "numerical" mode is available for a Cost-Type, the cost expressed in the "schedule" mode is an extension of its expression from one value in the "numerical" mode to an array of several values varying over time.

Types of Cost values such as JSONBool can also be expressed in the "schedule" mode, as states may be "true" or "false" depending on given time periods. It may be expressed as a single value which is either "true" or "false" following a decision rule outside the ALTO protocol.

3.2. ALTO Capability: Cost-Scope

To ensure that the application client uses the NP provided information in the cost schedule in an unambiguous way we define the Cost Scope capability, which defines the validity of the "scheduled" cost values.

For Cost Types whose values are provided in a mode different than 'schedule', the Cost Scope capability is specified by the string "permanent". The Cost Scope attributes provided for the 'schedule'

mode are listed below. The reference time zone for the provided values is UTC.

- o Unit: expresses the time interval applicable to each value. A two element array where the first element is the time unit, ranging from "second" to "year", and the second one the number of units of this duration. For example: '['minute', 5] means that each value is provided on a time interval lasting 5 minutes.
- o Size: the number of values of the cost schedule array,
- o Begin: the index of the first unit in the array,
- o Reference time zone: set to "UTC",
- o Next update: the date at which the sample will be re-computed,
- o Last update: the last re-computation date.

The reference time zone is UTC.

Attributes 'Last update 'and 'Next update' report on the update frequency and age of the information.

3.2.1. Example of time scope for a cost schedule

Let us assume that the metric 'pathoccupationcost' (POC for short) is computed for 24 hours, on time intervals lasting 2 hours, with the first interval starting at 0h00. The ALTO Server thus provides an array 12 values. This information is then used to enable applications to see which time intervals in a day are the most favorable to operate, and which "busy " time intervals should be avoided. If the "Begin" date is past, the application can also use the information to compute statistics or infer a some customized prediction.

3.3. Example of scheduled information resources in the IRD

The example IRD given in this Section includes 2 particular URIs:

- o "http://alto.example.com/endpointcost/lookup", in which the ALTO Server offers several Endpoint Cost Types, including a Cost called "pathoccupationcost" for which the "schedule" Cost Mode is available. The Endpoint Costs available are the "hopcount", "routingcost" and "pathoccupationcost" Cost Types, with the two first ones in the "numerical" Cost Mode and "pathoccupationcost" in the "schedule" Cost Mode.

- o "http://custom.alto.example.com/endpointcost/schedule/lookup", in which the ALTO Server provides the 'routingcost' in both "numerical" and "schedule" modes. This resource is accessible via a separate subdomain called "custom.alto.example.com". The ALTO Client may either get the last update of the 'routingcost' value or request for a previsual sample of 24 values established each for 1 hour. An ALTO Client can discover the services available at "custom.alto.example.com" by successfully performing an OPTIONS request to "http://custom.alto.example.com/endpointcost".

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

```
{
  ... usual ALTO resources ...

  "resources" : [
    {
      "uri" : "http://alto.example.com/endpointcost/lookup",
      "media-types" : [ "application/alto-endpointcost+json" ],
      "accepts" : [ "application/alto-endpointcostparams+json" ],
      "capabilities" : {
        "cost-constraints" : true,
        "cost-modes" : [ "numerical", "numerical", "schedule" ],
        "cost-types" : [ "routingcost", "hopcount", "pathoccupationcost" ],
        "cost-scope": [ "permanent", "permanent",
          { "unit": [ "hour", 1 ], "size": 24, "begin": 0,
            "time zone": "UTC",
            "lastupdate": mm/hh/dd/mm/yyyy,
            "nextupdate": mm/hh/dd/mm/yyyy }
        ]
      },
    },
    {
      "uri" : "http://custom.alto.example.com/endpointcost/schedule/lookup",
      "media-types" : [ "application/alto-endpointcost+json" ],
      "accepts" : [ "application/alto-endpointcostparams+json" ],
      "capabilities" : {
        "cost-constraints" : true,
        "cost-modes" : [ "numerical", "schedule" ],
        "cost-types" : [ "routingcost", "routingcost" ],
        "cost-scope": [ "permanent",
          { "unit": [ "hour", 1 ], "size": 24, "begin": 0,
            "time zone": "UTC",
            "lastupdate": mm/hh/dd/mm/yyyy,
            "nextupdate": mm/hh/dd/mm/yyyy }
        ]
      },
    }
  ]
}
```

3.3.1. Example scenario and ALTO transaction with a Cost Schedule

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

The Application Client has the choice to trade content or resources with a set of Endpoints of moderate 'routingcost', and needs to decide with which Endpoint it will trade at what time. For instance, one may assume that the Endpoints are spread on different time-zones, or have intermittent access. In this example, the 'routingcost' is assumed constant for the scheduling period and the time sensitive decision metric is the path bandwidth reflected by a Cost type called 'pathoccupationcost'.

The ALTO Client embedded in the Application Client queries ALTO information on 'pathoccupationcost' for the 24 hours following (implicitly) the date of "lastupdate", as this resource is listed in the IRD.

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

```
{
  "cost-type" : ["pathoccupationcost"],
  "cost-mode" : ["schedule"],
  "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-endpointcost+json
```

```
{
  "meta" : {},
  "data" : {
    "cost-type" : ["pathoccupationcost"],
    "cost-mode" : ["schedule"],
    "map" : {
      "ipv4:192.0.2.2": {
        "ipv4:192.0.2.89" : [7, ... 24 values],
        "ipv4:198.51.100.34" : [4, ... 24 values],
        "ipv4:203.0.113.45" : [2, ... 24 values]
      }
    }
  }
}
```

4. IANA Considerations

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

[ID-alto-protocol],

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

4.1. Information for IANA on proposed Cost Types

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

4.2. Information for IANA on proposed Endpoint Properties

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

5. Acknowledgements

Thank you to the ALTO WG for fruitful discussions.

Sabine Randriamasy is partially supported by the MEVICO project (<http://www.celtic-initiative.org/Projects/Celtic-projects/Call7/MEVICO/mevico-default.asp>), a research project supported by the European Commission under its 7th Framework Program CELTIC initiative (project no. CP 07-011). 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 MEVICO project or the European Commission.

6. References

6.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC5693] Seedorf, J. and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement", RFC 5693, October 2009.

6.2. Informative References

- [ID-alto-protocol]
R. Alimi, R. Penno, Y. Yang, Eds., "ALTO Protocol, draft-ietf-alto-protocol-13.txt", September 2012.
- [article-gslh-alto-sdn]
V. Gurbani, M. Scharf, T. Lakshman, 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.
- [draft-jenkins-alto-cdn-use-cases-01]
B. Niven-Jenkins (Ed.), G. Watson, N. Bitar, J. Medved, S. Previdi, "Use Cases for ALTO within CDNs, draft-jenkins-alto-cdn-use-cases-01", June 2011.
- [draft-randriamasy-multi-cost-alto]
S. Randriamasy, Ed., B. Roome, N. Schwan, "Multi-Cost ALTO, draft-randriamasy-alto-multi-cost-07", October 2012.
- [draft-xie-alto-sdn]
H. Xie, T. Tsou, D. Lopez, H. Yin, "Use Cases for ALTO with Software Defined Networks, draft-xie-alto-sdn-extension-use-cases-00", June 2012.
- [sdnrg] "Software Defined Network Research Group, <http://trac.tools.ietf.org/group/irtf/trac/wiki/sdnrg>".

Authors' Addresses

Sabine Randriamasy (editor)
Alcatel-Lucent Bell Labs
Route de Villejust
NOZAY 91460
FRANCE

Email: Sabine.Randriamasy@alcatel-lucent.com

Nico Schwan

Email: ietf@nico-schwan.de

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: April 22, 2013

S. Randriamasy, Ed.
W. Roome
Alcatel-Lucent Bell Labs
N. Schwan
October 19, 2012

Multi-Cost ALTO
draft-randriamasy-alto-multi-cost-07

Abstract

IETF is designing a new service called ALTO (Application Layer traffic Optimization) that includes a "Network Map Service", an "Endpoint Cost Service" and an "Endpoint (EP) Ranking Service" and thus incentives for application clients to connect to ISP preferred Endpoints. These services provide a view of the Network Provider (NP) topology to overlay clients.

The present draft proposes a light way to extend the information provided by the current ALTO protocol in two ways. First, including information on multiple Cost Types in a single ALTO transaction provides a better mapping of the Selected Endpoints to needs of the growing diversity of Content and Resources Networking Applications and to the network conditions. Second, one ALTO query and response exchange on N Cost Types is faster and lighter than N single cost transactions. All this also helps producing a faster and more robust choice when multiple Endpoints need to be selected.

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 April 22, 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	5
2. Application scope and terminology	6
3. Uses cases for using multiple costs	7
3.1. Use cases for using additional costs	7
3.1.1. Delay Sensitive Overlay Applications	8
3.1.2. Selection of physical servers involved in virtualized applications	8
3.1.3. CDN Surrogate Selection	9
3.1.4. Some proposed additional properties and costs	9
3.2. Use cases for Multi-Cost ALTO transactions	10
3.2.1. Optimized Endpoint Cost Service	11
3.2.2. Optimized Filtered Cost Map Service	11
3.2.3. Cases of unpredictable Endpoint cost value changes	12
3.2.3.1. Case of a Multi-Cost ALTO query upon a route change	12
3.2.3.2. Case of a Multi-Cost ALTO query upon a cost value change	13
4. ALTO Protocol updates needed to support multi-cost transactions	14
4.1. List of ALTO protocol updates required and recommended	15
4.2. Updates required in the member format of objects	16
4.2.1. Cost value encoded in JSONArray	16
4.2.2. Format update on CostMode and CostType	16
4.3. Rules required on object member description	17
4.3.1. Rule on cost value order in ALTO reponses	17
4.3.2. Rule on mapping for cost-type and cost-mode array members	17
4.4. Updates recommended in the object structure	17
4.5. Rule recommended on the cost value mode	17
4.6. Extended constraints on multi-cost values	18
4.6.1. Use cases for mutli-cost multi-operator constraints	18
4.6.2. Extended constraints in Multi-Cost ALTO	19
5. Protocol extensions for multi-cost ALTO transactions	19
5.1. Information Resources Directory	20
5.1.1. Example of Multi-Cost specific resources in the IRD	20
5.2. Multi-Cost Map Service	22
5.2.1. Media Type	22
5.2.2. HTTP Method	22
5.2.3. Input Parameters	22
5.2.4. Capabilities	22
5.2.5. Response	23
5.2.6. Example	25
5.3. Filtered Multi-Cost Map	25
5.3.1. Media Type	26
5.3.2. HTTP Method	26
5.3.3. Input Parameters	26

5.3.4.	Capabilities	28
5.3.5.	Response	28
5.3.6.	Example 1	29
5.3.7.	Example 2	29
5.4.	Endpoint Multi-Cost Service	30
5.4.1.	Media Type	31
5.4.2.	HTTP Method	31
5.4.3.	Input Parameters	31
5.4.4.	Capabilities	32
5.4.5.	Response	32
5.4.6.	Example	33
6.	IANA Considerations	34
6.1.	Information for IANA on proposed Cost Types	35
6.2.	Information for IANA on proposed Endpoint Properties	35
7.	Acknowledgements	35
8.	References	35
8.1.	Normative References	35
8.2.	Informative References	36
	Authors' Addresses	36

1. Introduction

IETF is designing a new service called ALTO that provides guidance to overlay applications, which have to select one or several hosts from a set of candidates that are able to provide a desired resource. This guidance is based on parameters that affect performance and efficiency of the data transmission between the hosts, e.g., the topological distance. The purpose of ALTO is to improve Quality of Experience (QoE) in the application while reducing resource consumption in the underlying network infrastructure. The ALTO protocol conveys the Internet View from the perspective of a Provider Network region that spans from a region to one or more Autonomous System (AS). Together with this Network Map, it provides the Provider determined Cost Map between locations of the Network Map. Last, it provides the Ranking of Endpoints w.r.t. their routing cost.

Current ALTO Costs and their modes provide values that are seen to be stable over a longer period of time, such as hopcount and administrative routing cost to reflect ISP routing preferences. Recently, new use cases have extended the usage scope of ALTO to Content Delivery Networks, Data centers and applications that need additional information to select their Endpoints or handle their PIDs.

Thus a multitude of new Cost Types that better reflect the requirements of these applications are expected to be specified, in particular cost values that change more frequently than previously assumed.

The current ALTO protocol draft [ID-alto-protocol-11] restricts ALTO Cost Maps and Endpoint Cost services to only one Cost Type and Cost Mode per ALTO request. To retrieve information for several Cost Types, an ALTO client must send several separate requests to the server.

It would be far more efficient, in terms of RTT, traffic, and processing load on the ALTO client and server, to get all costs with a single query/response transaction. Vector costs provide a robust and natural input to multi-variate path computation as well as robust multi-variate selection of multiple Endpoints. In particular, one Cost Map reporting on N Cost Types is less bulky than N Cost Maps containing one Cost Type each. This is valuable for both the storage of these maps and their transmission. Additionally, for many emerging applications that need information on several Cost Types, having them gathered in one map will save time.

There are three parts in this draft. The first part exposes use cases motivating the introduction of new Cost Types and why multi-

cost transactions are useful. The second part specifies the core ALTO protocol extensions that are required or recommended to support requests and responses on multiple Cost Types in one single transaction. These extensions also integrate the discussions within the ALTO Working Group. The third part lists the Multi-Cost ALTO services that can be supported by these extensions.

2. Application scope and terminology

This draft generalizes the case of a P2P client to include the case of a CDN client, a client of an application running on a virtual server, a GRID application client and any Client having the choice in several connection points for data or resource exchange. To do so, it uses the term "Application Client" (AC).

This draft focuses on the use case where the ALTO client is embedded in the Application Client or in some Application Endpoint tracker in the network, such as a P2P tracker, a CDN request router or a cloud computing orchestration system implemented in a logically centralized management system.

It is assumed that Applications likely to use the ALTO service have a choice in connection endpoints as it is the case for most of them. The ALTO service is managed by the Network Provider (NP) and reflects its preferences for the choice of endpoints. The NP defines in particular the network map, the routing cost among Network Locations, the cost types used to reflect it, and which ALTO services are available at a given ALTO server.

The draft uses terms defined as follows:

- o Endpoint (EP): can be a Peer, a CDN storage location, a physical server involved in a virtual server-supported application, a Party in a resource sharing swarm such as a computation Grid or an online multi-party game.
- o Endpoint Discovery (EP Discovery) : this term covers the different types of processes used to discover the eligible endpoints.
- o Network Service Provider (NSP): includes both ISPs, who provide means to transport the data, and Content Delivery Networks (CDNs) who care for the dissemination, persistent storage and possibly identification of the best/closest content copy.
- o ALTO transaction: a request/response exchange between an ALTO Client and an ALTO Server.

- o Application Client (AC): this term generalizes the case of a P2P client to include the case of a CDN client, a client of an application running on a virtual server, a GRID application client and any Client having the choice in several connection points for data or resource exchange.

3. Uses cases for using multiple costs

The ALTO protocol specification in [ID-alto-protocol-11] focuses on the basic use case of optimizing routing costs in NSP networks. Upcoming use cases however will require both new Cost Types and new Endpoint Properties. Recent ALTO use cases now extend to CDNs, Data centers and other applications that need additional information to select their Endpoints or handle their PIDs. The needed Cost Types depend on the QoE requirements that are specific to the applications. Moreover, the cost values that they may use may change more rapidly than assumed up to now.

The goal of this section is to describe forward looking use case scenarios that are likely to benefit from ALTO, in order to motivate the introduction of new Cost Types and Endpoint Properties as well as the ALTO Multi-Cost extension.

3.1. Use cases for using additional costs

ALTO Cost Types and Endpoint Properties are registered in two registries maintained by IANA. The ALTO Cost Type registry ensures that the Cost Types that are represented by an ALTO Cost Map are unique identifiers, and it further contains references to the semantics of the Cost Type. The current specification registers 'routingcost' as a generic measure for routing traffic from a source to a destination. In a similar way the ALTO Endpoint Property Registry ensures uniqueness of ALTO Endpoint Property identifiers and provides references to particular semantics of the allocated Endpoint Properties. Currently the 'pid' identifier is registered, which serves as an identifier that allows aggregation of network endpoints into network regions. Both registries accept new entries after Expert Review. New entries should conform to the respective syntactical requirements, and must include information about the new identifier, the intended semantics, and the security considerations. One basic example advocating for multiple Cost Type transactions is an Application Client looking for destination Endpoints or Source/Destination PID pairs yielding jointly the lowest 'routingcost' and path delay. We hereby assume that 'routingcost' values report some monetary cost and that the Application Client chooses to rely on the hopcount to reflect the path delay.

3.1.1. Delay Sensitive Overlay Applications

The ALTO working group has been created to allow P2P applications and NSPs a mutual cooperation, in particular because P2P bulk file-transfer applications have created a huge amount of intra-domain and congestion on low-speed uplink traffic. By aligning overlay topologies according to the 'routingcost' of the underlying network, both layers are expected to benefit in terms of reduced costs and improved Quality-of-Experience.

Other types of overlay applications might benefit from a different set of path metrics. In particular for real-time sensitive applications, such as gaming, interactive video conferencing or medical services, creating an overlay topology with respect to a minimized delay is preferable. However it is very hard for an NSP to give accurate guidance for this kind of realtime information, instead probing through end-to-end measurements on the application layer has proven to be the superior mechanism. Still, a NSP might give some guidance to the overlay application, for example by providing statistically preferable paths, possibly with respect to the time of day. Also static information like hopcount can serve as an indicator for the delay that can be expected. Thus a Cost Type that can indicate latency, without the need for end-to-end measurements between endpoints, is likely to be useful.

3.1.2. Selection of physical servers involved in virtualized applications

Virtualized applications in large Datacenters are supported by virtualized servers that actually gather resources distributed on several physical servers. The federation of these resources is often orchestrated by a centralized entity that needs to select the physical servers from or to which it will take resources. This entity can be co-located with an ALTO Client that will request and get the ALTO information on the network formed by the physical servers. The physical servers can be assimilated to endpoints with which the orchestration entity trades application resources or content. These resources include computation resources, storage capacity and path bandwidth between the physical servers.

Here too, the applications that are ran are diverse and may have different and specific QoE requirements. The Endpoint selection typically needs to consider both the computational resources at the Endpoints and the resources e.g. in bandwidth on the transmission paths to or among Endpoints. Thus the application QoE requirements drive the Endpoint selection with more or less weight on QoE specific metrics such as hopcount/delay, bandwidth and other resources, that are typically combined with the routing cost and need to jointly

integrate the Endpoint and transmission path perspective in the decision process, which is difficult to do with one single Cost Type.

3.1.3. CDN Surrogate Selection

Another use case is motivated through draft [draft-jenkins-alto-cdn-use-cases-01]. The request router in today's CDNs makes a decision about the surrogate or cache node to which a content request should be forwarded. Typically this decision is based on locality aspects, i.e. the request router tries to select the surrogate node closest to the client. By using the 'routingcost' Cost Type, an ALTO server allows an NSP to guide the CDN in selecting the best cache node. This is particularly important as CDNs place cache nodes deeper into the network (i.e., closer to the end user), which requires finer grained information. Finally the provisioning of abstracted network topology information across administrative boundaries gains importance for cache federations.

While distance today is the predominant metric used for routing decisions, other metrics might allow sophisticated request routing strategies. For example the load a cache node sees in terms of CPU utilization, memory usage or bandwidth utilization might influence routing decisions for load-balancing reasons. There exist numerous ways of gathering and feeding this kind of information into the request routing mechanism.

For example, information reporting on the occupation level of a cache could be based on a cost reflecting: its remaining computation resources, its remaining storage capacity w.r.t its capacity in storage or computation resources.

As ALTO is likely to become a standardized interface to provide network topology information, the ALTO server could also provide other information that a request router needs. In the next iterations of this draft we will analyse which of these metrics is suitable as a Cost Type or Endpoint Property for CDN Surrogate Selection, and propose to register them in the respective registries.

3.1.4. Some proposed additional properties and costs

In addition to CDN caches, Endpoint Properties and Costs can be useful to report an Endpoint's load, given that an Endpoint can as well be a physical server in a datacenter or any entity as defined in Section 2 of this draft.

Proposed new Endpoint properties and costs include:

- o an Endpoint Property called "EPCapacity", reflecting the nominal capacity of this endpoint. This capacity could be split into:
 - * EP Nominal Memory: the storage capacity of the Endpoint.
 - * EP Nominal Bandwidth: the capacity of the computation resources of the Endpoint.
- o an Endpoint Cost called "EP occupied Capacity", reflecting the currently available resources w.r.t. their nominal capacity. As with EP Capacity, this can be split into:
 - * EP Occupied Memory: the remaining storage capacity,
 - * EP Occupied Bandwidth: the remaining computation resources.

Likewise, new Cost Types are needed to describe the resources of the network paths needed for content transport, in particular the utilized network path bandwidth.

- o A Cost Type named 'pathoccupationcost' (POC) can be used to reflect the NP view of the utilized path bandwidth. Such an ALTO Cost Type is likely to have values that change frequently. By no means, as stated in the ALTO requirements, are ALTO Cost types expected to reflect real-time values, as these can be gathered by other mechanisms. Instead, a Cost Type such as 'pathoccupationcost' should be used as an abstraction that may be represented by a statistical value, or be updated regularly at a frequency lower than 'real-time', or be provided according to different time periods or other parameters. A provision mode for time dependent cost values is proposed in [draft-randriamasy-alto-cost-schedule-01]

3.2. Use cases for Multi-Cost ALTO transactions

Different Cost Types are suitable for different applications. For example, delay sensitive applications look for both low routing cost and low delay, where as other applications, such as non real time content download, look for moderate delay and minimal losses. On the other hand, applications or entities managing application input information may want, for various reasons to update their ALTO information on several Cost Types. So an ALTO Client may want to mix Cost Types in either 'numerical' and 'ordinal' mode, for Cost Types values that can be represented by numerical values.

The Multi-Cost ALTO Services propose to:

- o include several Cost Types (and/or Cost Modes) in an ALTO client's Cost Map and Endpoint Cost request,
- o provide several Cost Type values (and/or Cost Mode) in an ALTO server's response, instead of one.

The primary reasons to use Multi-Cost ALTO are:

- o Optimizing time and bandwidth: a single ALTO response with a Multi-Cost cost map with three separate Cost Type values takes much less network bandwidth, and fewer CPU cycles, than three separate ALTO requests for three complete single-cost cost maps. The motivation also holds for the Endpoint Cost Service. Multi-Cost ALTO services can straightforwardly provide a more complete set of cost information.
- o Facing unpredictable and/or rapid value changes: an ALTO client can get a consistent snapshot of several different rapidly-varying Cost Type values.

3.2.1. Optimized Endpoint Cost Service

The Endpoint Cost Service (ECS) provides cost information about both the application Endpoint resources and the networking resources used to access those Endpoints. In addition, the ECS may be invoked in "short term" situations, that is for frequent requests and/or requests requiring fast responses. For the ECS, the server's response is restricted to the requested Endpoints, and so is much smaller than a complete Cost Map. Therefore the ECS can be invoked for 'nearly-instant' information requests, and is particularly well suited for multi-cost ALTO transactions, supporting requests and responses on several Cost Type values simultaneously.

3.2.2. Optimized Filtered Cost Map Service

The set of ALTO Cost Types is not restricted to 'routingcost': ALTO Servers may provide a broader set of metrics. One thing to consider is that the frequency of updates can vary from a Cost Type to another one. Additionally the volume of an entire cost map with values of all available Cost Types, may get rapidly prohibitive for frequent downloads. Given these considerations the Application Client may take better advantage when:

- o requesting multi-cost maps filtered w.r.t. Cost Types of compatible update frequencies or dates, which is the responsibility of the Application Client,

- o requesting multi-cost maps filtered w.r.t. a restricted set of PID pairs.

In such a case, as with the Endpoint Cost Service, the purpose of a Multi-Cost transaction is to gain time with whatever future use of the received ALTO information. In this case, the Client may mix Cost Types in either 'numerical' and 'ordinal' mode, for Cost Type values that can be represented by numerical values.

3.2.3. Cases of unpredictable Endpoint cost value changes

Querying all Endpoint cost values simultaneously is always more time and resources efficient than doing it sequentially.

It becomes a necessity in case of unpredictable and/or rapid value changes on at least one of the ALTO Cost Types. The term 'rapid' here means "Typical update intervals [that] may be several orders of magnitude longer than the typical network-layer packet round-trip time (RTT)", as described in [ID-ALTO-Requirements13], up to a couple of minutes.

This section provides two examples of a delay sensitive application using 'routingcost' and 'hopcount' to select an Endpoint. The application can choose between two candidate Endpoints, EP1 and EP2. The initial choice at T=1 is EP1. It is assumed that at T=2 events in the network occur that impact both 'routingcost' and 'hopcount'.

These examples illustrate the need to query 'hopcount' and 'routingcost' values at the same time in order to re-evaluate the EP costs w.r.t. the QoE needs of the application. It is assumed that the application triggers regular ALTO requests to get the latest cost values for a list of candidate Endpoints.

In some cases the Application client wants to use the ALTO information to perform multi-variate optimization on several Cost Type values. In order for the optimization to be reliable, it is recommended that the Cost Type values are provided in 'numerical' Cost Mode. Therefore the requested Cost Mode for the applicable Cost Types SHOULD be 'numerical'.

3.2.3.1. Case of a Multi-Cost ALTO query upon a route change

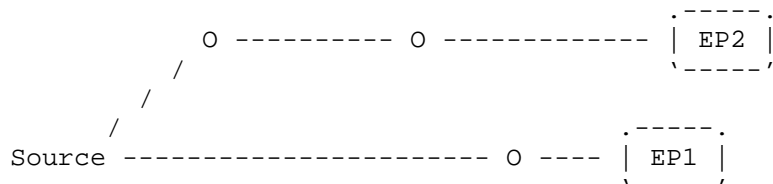
In Figure 1, initially at time T=1, the application has chosen EP1 rather than EP2, despite the higher routing cost, because EP1 has a "better" (lower) 'hopcount' value and despite the higher routing cost and possibly because the application has set a higher weight to 'hopcount'.

At a time T=2, the route to EP1 changes. The ALTO Server information is accordingly updated. The ALTO client makes its next request to update the cost values for 'routingcost' and 'hopcount' on EP1 and EP2. It appears that EP1 has now a hopcount value of 3, the same than for EP2 while its routing cost is higher.

The application realizes that there is no more benefit in keeping interacting with EP1 and therefore switches to EP2, that now has the same hopcount but a lower routing cost.

T = 1 : EP1: routingcost = 40, hopcount = 2
 EP2: routingcost = 30, hopcount = 3

EP1 is selected because application is time-sensitive and metric 'hopcount' has a higher weight



T = 2 : EP1: routingcost = 40, hopcount = 3
 EP2: routingcost = 30, hopcount = 3

- Route to EP1 has changed. Hopcount is now 3

=> EP2 is selected because routingcost is lower than for EP1, with the same hopcount value

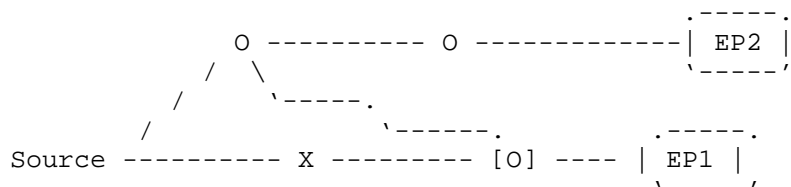
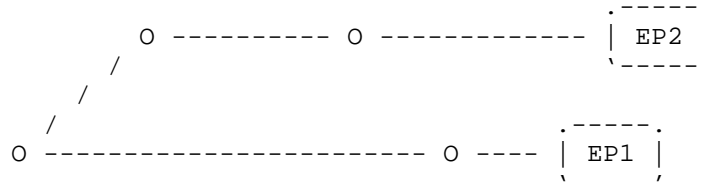


Figure 1: Endpoint re-selection using Multi-Cost ALTO request on updated cost values, upon a change in the route.

3.2.3.2. Case of a Multi-Cost ALTO query upon a cost value change

T = 1 : EP1: routingcost = 30, hopcount = 2
 EP2: routingcost = 30, hopcount = 3
 ==> EP1 is selected because application is time-sensitive and
 hopcount metrics has higher weight



T = 2 : EP1: routingcost = 40, hopcount = 2
 EP2: routingcost = 30, hopcount = 3
 Routingcost to EP1 has increased. Hopcount is the same.
 ==> Delay sensitive applications willing to minimize hopcount
 remain with EP1 while other applications may remain
 with EP2, that now has a lower routingcost.

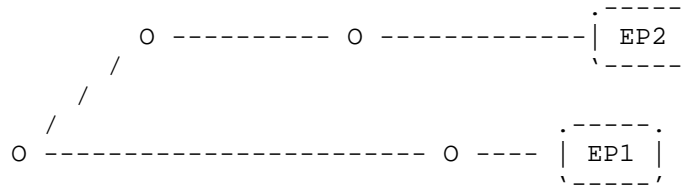


Figure 2: Endpoint selection using 2 Cost Types with joint request on updated cost values and for delay sensitive applications.

4. ALTO Protocol updates needed to support multi-cost transactions

To allow running Multi-Cost ALTO Services some minor changes in the base protocol are needed. A set of multi-cost specific media-types is introduced and the main updates consist into changing the JSON type of the value taken by a few members of the objects describing the information resources.

As written in the introduction, this section relies on the previous version of the ALTO protocol draft, see [ID-alto-protocol]. It partially integrates an update of the current version issued recently, see [ID-alto-protocol-11], that proposes a generic encoding of cost values in the 'JSONValue' data type. The proposed Multi-Cost specifications will be updated according to the outcome of WG discussions.

This section lists and details the proposed changes according to the previous ALTO protocol draft, [ID-alto-protocol] .

If members 'cost-type' and 'cost-mode' of objects InfoResourceCostMap, InfoResourceEndpointCostMap, ReqFilteredCostMap, ReqEndpointCostMap remain specified as single values in the base ALTO protocol, then Multi-Cost specific media types need to be used similarly to those specified in the previous version of this draft, see [draft-randriamasy-alto-multi-cost-05].

4.1. List of ALTO protocol updates required and recommended

The following updates to the current ALTO protocol, see [ID-alto-protocol], are required or recommended to support multi-cost ALTO transactions. The new resulting JSON formats are specified in the next sections.

- o Updates required in the format of objects member(s):
 - * Objects DstCosts (to destination PIDs) and EndpointDstCosts (to destination Endpoints): JSON type of cost value member evolves from JSONNumber to JSONArray.
 - * Objects InfoResourceCostMap, ReqFilteredCostMap, ReqEndpointCostMap, InfoResourceEndpointCostMap: members 'cost-mode' and 'cost-type' have now an array of values rather than a single value.
- o Updates recommended in the object structure:
 - * Objects CostMapCapability and FilteredCostMapCapability: new member giving the maximum number of Cost Types in a response.
- o Rules required on object member description:
 - * Order in which the multiple cost values are provided in the responses,
 - * Number of values in member 'cost-types' of objects InfoResourceCostMap, InfoResourceEndpointCostMap, ReqFilteredCostMap, ReqEndpointCostMap.
- o Rule recommended on the cost value mode:
 - * when the mode 'numerical' is available or applicable.

4.2. Updates required in the member format of objects

This section specifies the changes in the object member format that are required to enable multi-cost ALTO transactions.

The term Single Cost qualifies the items as they are specified in the current ALTO protocol draft, up to version 10

4.2.1. Cost value encoded in JSONArray

The fundamental change to support multi-cost is to encode the cost values with the type JSONArray. This way, the cost between 2 PIDs or to an Endpoint can be represented in a generic way:

- o with several Cost Types,
- o with Cost Types whose value can each be encoded with any type of JSON value.

For example, a multi-cost value represented with Cost Types (assuming they are supported by the ALTO Server):

```
["routingcost", "hopcount", "quarterlyvaluexxx", "statustring"]
```

will be encoded in the following JSON Array in a Multi Cost ALTO response:

```
[23, 6, [2, 5, 4, 1], "medium"]
```

The objects impacted by the encoding of ALTO Multi-Cost values in a JSONArray are: DstCosts and EndpointDstCosts. Full specification will be provided in later sections of this draft.

4.2.2. Format update on CostMode and CostType

In the base protocol, Objects InfoResourceCostMap, ReqFilteredCostMap, ReqEndpointCostMap, InfoResourceEndpointCostMap have members 'cost-mode' and 'cost-type' that list which Cost Type is reported and in which mode this Cost Type is represented.

In Multi-Cost ALTO several Cost Types are used per destination PID or Endpoint, so the member 'cost-type' of these objects must now be an array of values rather than a single value. Likewise, the member 'cost-mode' must now be an array, where each value reports the representation mode of the corresponding index in the 'cost-type' list.

The change on members 'cost-mode' and 'cost-type' from a single value to an array of values are specified in later sections.

4.3. Rules required on object member description

When several cost values are provided, it is necessary to unambiguously specify to which Cost Type each value corresponds and in which mode each value is provided.

4.3.1. Rule on cost value order in ALTO responses

The cost values each Source/Destination pair **MUST** be provided in the same order as in the array of Cost Types. This way, the cost type values are provided without any ambiguity on the Cost Type they report on.

4.3.2. Rule on mapping for cost-type and cost-mode array members

The cost-mode array **MUST** be of the same size as the cost-type array. Each value of this array maps to the Cost Type ID at the same place in the Cost Type array and this value specifies the mode in which the value for this Cost Type is provided.

4.4. Updates recommended in the object structure

Objects MultiCostMapCapability and FilteredMultiCostMapCapability: new member giving the maximum number of Cost Types in a response.

4.5. Rule recommended on the cost value mode

In multi-cost transactions: when the mode 'numerical' is available for a Cost Type, it **MUST** be the one used to represent the cost values. In any case, the Cost Mode used for each Cost Type **MUST** be exactly specified.

The following example illustrates how this rule can be applied:

Assume the Cost Types array:

```
["routingcost", "hopcount", "quarterlyvaluexxx", "statuststring"]
```

The corresponding Cost Mode array should be (assuming that these modes are supported):

```
["numerical", "numerical", "dynamic", "string"]
```

An example of values is:

```
[23, 6, [21, 9, 4, 12], "medium"]
```

In this example, it is assumed that when the value of a Cost Type is expressed by an array of numbers such as [21, 9, 4, 12], the values in this array are expressed in the 'numerical' mode.

4.6. Extended constraints on multi-cost values

This draft proposes to extend the constraint tests in the base protocol to allow tests on the various costs in a request, and to allow more general predicates.

The base ALTO protocol allows optional constraints in the input parameters to a request for a Filtered Cost Map or the Endpoint Cost Service. The 'constraints' member is an array of expressions that all apply to the (single) requested Cost Type. The encoding of 'constraints' member, is fully specified in Section 6.8.2.2.3 "Input parameters" of the base protocol as follows:

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 costtype parameter

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

Such a specification covers multiple predicates on one metric such as:

'routingcost' values belong to [6, 20)

However, an application

4.6.1. Use cases for mutli-cost multi-operator constraints

Suppose that an application uses information on the ALTO Cost Types 'hopcount' and 'routingcost'. This application may want to select paths or Endpoints with bounds on values for both 'hopcount' and 'routingcost'. For instance solutions meeting a constraint like:

'hopcount' values in [6,20) OR 'routingcost' values in [100,200]

Moreover, this application may be ready to make compromises and to select paths or Endpoints by bounding their cost values according to

two options:

1. either solutions with moderate 'hopcount' and high 'routingcost', for instance: 'hopcount' values in [6,20] AND 'routingcost' values in [100,200],
2. or solutions with higher 'hopcount' and moderate 'routingcost', for instance: 'hopcount' values in [20,50] AND 'routingcost' values in [30,100].

4.6.2. Extended constraints in Multi-Cost ALTO

This draft proposes to support the two above mentioned use cases by extending the scope of constraints in two ways:

- o allow the 'constraint' member to be applicable to multiple Cost Types,
- o allow the multiple constraints to be related to each other by both logical AND and logical OR.

The two options would be covered by a logical expression like:

```
[('hopcount' ge 6) AND ('hopcount' lt 20) AND
 ('routingcost' ge 100) AND ('routingcost' le 200)]
OR
[('hopcount' ge 20) AND ('hopcount' le 50) AND
 ('routingcost' ge 30) AND ('routingcost' le 100)]
```

A simple encoding of multi-cost constraints for such expressions is specified in Section 5.3.3 of this draft, describing the input parameters to request for Filtered Cost Map. This specification is applicable to the EP Cost service as well.

5. Protocol extensions for multi-cost ALTO transactions

This section proposes extensions of the ALTO protocol to support Multi Cost ALTO Services or provide additional ALTO information. It integrates discussions on the ALTO mailing list.

If an ALTO client desires information on several Cost Types, then instead of placing as many requests as costs, it may request and receive all the desired Cost Types in one single transaction.

The ALTO server then, provided it supports the requested Cost Types, and provided it supports multi-cost ALTO transactions, sends one

single response where for each {source, destination} pair, the cost values are arranged in an array, where each component corresponds to a specified Cost Type. The correspondence between the components and the Cost Types is implicitly indicated in the ALTO response. Indeed, the values in the Cost values MUST be provided in the same order as in the array of cost types indicated in the response.

The following ALTO services have corresponding Multi-Cost extensions:

- o Information Resources Directory: extended with multi-cost related URIs and associated capabilities.
- o Cost Map Service: extended with the Multi-Cost Map Service,
- o Cost Map Filtering Service: extended with the Multi-Cost Map Filtering Service,
- o Endpoint Cost Lookup Service: extended with the Endpoint Multi-Cost Lookup Service.

5.1. Information Resources Directory

When the ALTO server supports the provision of information on multiple costs in a single transaction, the Information Resources Directory will list the corresponding resources. The media type remains the same as in the current ALTO protocol.

5.1.1. Example of Multi-Cost specific resources in the IRD

The following is an example Information Resource Directory returned by an ALTO Server and containing Multi-Cost specific services: the Multi-Cost Map Service, Filtered Multi-Cost Map and the Endpoint Multi-Cost Service. It is assumed that the IRD contains usual ALTO Services as described in the example IRD of the current ALTO protocol. In this example, the ALTO Server additionally provides Multi-Cost Services in a specific folder of "alto.example.com" called "multi". This folder contains the Multi-Cost Maps, Filtered Multi-Cost Maps as well as the Endpoint Multi-Cost Service.

In this example, the ALTO IRD exposes Multi-Cost capabilities on cosy types "routingcost", "hopcount", "pathoccupationcost", that can be combined in a request. The values on these metrics are provided in numerical mode. Values provided for cost-type string are in "string" 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: [TODO]
Content-Type: application/alto-directory+json

{
  "resources" : [
    {
      .....
      Usual ALTO "single-cost" Services as described in current ALTO Protocol
      .....
    }, {
      "uri" : "http://alto.example.com/multi/costmap",
      "media-types" : ["application/alto-multicostmap+json"],
      "capabilities" : {
        "cost-types" : [ "routingcost", "hopcount" ],
        "cost-modes" : [ "numerical", "numerical" ]
      }
    }, {
      "uri" : "http://alto.example.com/multi/costmap/filtered",
      "media-types" : ["application/alto-multicostmap+json" ],
      "accepts" : ["application/alto-multicostmapfilter+json" ],
      "capabilities" : {
        "cost-constraints" : true,
        "max-cost-types" : 3,
        "cost-types" : [ "routingcost", "hopcount", "pathoccupationcost" ],
        "cost-modes" : [ "numerical", "numerical", "numerical" ]
      }
    }, {
      "uri" : "http://alto.example.com/multi/endpointmulticost/lookup",
      "media-types" : [ "application/alto-endpointmulticost+json" ],
      "accepts" : [ "application/alto-endpointmulticostparams+json" ],
      "capabilities" : {
        "cost-constraints" : true,
        "max-cost-types" : 2,
        "cost-types" : [ "routingcost", "hopcount", "status" ],
        "cost-modes" : [ "numerical", "numerical", "string" ]
      }
    }
  ]
}
```

5.2. Multi-Cost Map Service

This section introduces a new media-type for the Multi-Cost map. For each source/destination pair of PIDs, it provides the values of the different Cost Types supported for the Multi-Cost map, in the same order as in the list of Cost Types specified in the capabilities.

A Multi-Cost Map MAY be provided by an ALTO Server.

Note that the capabilities specify implicitly the order in which the different Cost Type values will be listed in the Cost Map.

The Cost Type values in the responses are encoded as a JSONArray of cost values for the different Cost Types.

Note that values in a Multi-Cost map are arrays of values of the various Cost Types. If the ALTO server does not have the value for a particular Cost Type for a source/destination PID pair, the server MUST use 'null' (a reserved JSON symbol) for that location in the array. If the ALTO server does not have a value for any of the Cost Types for a given source/destination pair -- that is, the array is a list of nulls -- then the ALTO server MAY omit the entry for that source/destination pair.

5.2.1. Media Type

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

5.2.2. HTTP Method

This resource is requested using the HTTP GET method.

5.2.3. Input Parameters

None.

5.2.4. Capabilities

This resource may be defined for 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 {  
  CostType cost-types<0..*>;  
  CostMode cost-modes<0..*>;  
} MultiCostMapCapability;
```

with members

cost-types The Cost Types (Section 5.XX) supported by the corresponding URI. If not present, this member MUST be interpreted as an empty array.

cost-modes The Cost Modes (Section 5.XX) supported for each of the supported Cost Types listed in the array 'cost-types'. This array MUST have the same size as the array 'cost-types', and each member of this array MUST give the mode of the Cost Type at that index.

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.

An ALTO Server supporting the Multi-Cost Map service, MUST support the Cost mode 'numerical' for all supported Cost Types encoded with the 'JSONNumber' type.

5.2.5. Response

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

```
object DstMultiCosts {
  JSONArray [PIDName];
  ...
};

object {
  DstMultiCosts [PIDName]<0..*>;
  ...
} MultiCostMapData;

object {
  CostType      cost-type<0..*>;
  CostMode      cost-mode<0..*>;
  JSONString    map-vtag;
  MultiCostMapData map;
} InfoResourceMultiCostMap;
```

with members:

cost-mode Array of Cost Modes (Section xxx) used in the Cost Map. This array MUST have the same size as the array 'cost-types'. where each member of the cost-mode array is the Cost Mode used for the Cost Type at the same place in the array.

cost-type The array of Cost Types (Section xxx) used in the Cost Map.

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

map The Cost Map data itself.

MultiCostMapData 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 DstMultiCosts object denotes the associated costs to a set of destination PIDs each identified by a string indexed by PIDName. For each destination PID, object DstMultiCost[PIDName] provides an array of one or several values, each corresponding to the Cost Type listed at the same place in the 'cost-type' array. This array MUST have the same size as the 'cost-type' array. The values in the DstMultiCosts[PIDName] array MUST be listed in the same order as in the 'cost-type' array.

The returned Cost Map MUST include the required Path Costs for each pair of Source and Destination PID for which this information is available. If a cost value is not defined, it MUST be replaced by

the reserved JSON symbol 'null'.

The members 'cost-mode' and 'cost-type' MUST be arrays with the same number of elements.

5.2.6. Example

This example illustrates a 'static' multi-cost' ALTO transaction, where the utilized Cost Types all have 'static' values. We assume here that the Cost Types available at the ALTO Server are "routingcost" and "hopcount" and the 'numerical' mode is available for both of them. The "routingcost" may be based on monetary considerations where as the "hopcount" is used to report on the path delay. We also assume that ALTO server does not know the value of the "routingcost" between PID2 and PID3, and hence uses null for those costs.

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

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

```
{
  "meta" : {},
  "data" : {
    "cost-mode" : ["numerical", "numerical"],
    "cost-type" : ["routingcost", "hopcount"],
    "map-vtag" : "1266506139",
    "map" : {
      "PID1": { "PID1":[1,0],    "PID2":[5,23],    "PID3":[10,5] },
      "PID2": { "PID1":[null,5], "PID2":[1,0],    "PID3":[15,9] },
      "PID3": { "PID1":[20,12],  "PID2":[null,1],  "PID3":[1,0] }
    }
  }
}
```

5.3. Filtered Multi-Cost Map

A Multi-Cost Map may be very large. In addition, an Application Client assisted by the ALTO Client does not necessarily need the Cost Types for all the source/destination PID pairs.

Therefore applications may more likely use Cost Map information filtered w.r.t. the Cost types as well as the source/destination

pairs of PIDs. This section specifies Filtered Multi-Cost Maps.

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

5.3.1. Media Type

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

5.3.2. HTTP Method

This resource is requested using the HTTP POST method.

5.3.3. 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-multicostmapfilter+json", which is a JSON Object of type ReqFilteredMultiCostMap, where:

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

object {
  CostType    cost-type<0..*>;
  CostMode    cost-mode<0..*>;
  JSONString  constraints<0..*>;      [OPTIONAL]
  JSONArray   or-constraints<0..*>;   [OPTIONAL]
  PIDFilter   pids;                   [OPTIONAL]
} ReqFilteredMultiCostMap;
```

with members:

cost-type The Cost Types for the returned costs.
Each listed Cost Type MUST be one of the supported Cost Types indicated in this resource's capabilities.

cost-mode The Cost Mode for each of the returned Cost Types.
As for the choice of Cost Modes, the ALTO Clients SHOULD be cognizant of operations applicable to different Cost Modes.
For Cost types values encoded with the 'JSONNumber' type, the Cost Mode SHOULD be numerical when the purpose is to perform

multi-variate optimization.

constraints

Defines an array of additional constraints. The ALTO server MUST return the costs for all source/destination pair that satisfy all constraints in this list, and no other costs. This parameter MUST NOT be specified if this resource's capabilities (Section XXXX?) indicate that constraint support is not available. Each string in the 'constraint' array MUST contain three entities separated by whitespace, in the following format:

[index] op value

'Index' is a number between 0 and the number of Cost Types minus 1, and indicates the Cost Type to which this constraint applies. (The square brackets ([]) surrounding 'index' are required syntactic sugar. They serve as a reminder that 'index' is an array index, not a value to test, and they avoid unusual-looking constraints such as "1 ge 5".) 'Op' is an operator: 'gt' for greater than, 'lt' for less than, 'ge' for greater than or equal to, 'le' for less than or equal to, 'eq' for equal to, or 'ne' for not equal to. 'Value' is a target cost value to compare against the indicated Cost Type. For numeric Cost Types, 'value' MUST be a number defined in the same units as the Cost Type indicated by 'index'. ALTO servers SHOULD use at least IEEE 754 doubleprecision floating point [IEEE.754.2008] to store the cost value, and SHOULD perform internal computations using double-precision floating-point arithmetic. For string Cost Types, 'value' MUST be a string enclosed in single quotes ('). For array-valued Cost Types, 'eq' is true iff one of the Cost Type values is equal to 'value', and 'ne' is true iff none of the Cost Type values are equal to 'value'. The other operators are not defined for array-valued Cost Types.

or-constraints

Defines an array of arrays of constraint strings. The individual constraint strings MUST be in the same format as strings in the 'constraints' parameter. The ALTO server MUST return costs that satisfy all constraints in one or more of the inner lists, and no other costs. That is, 'or-constraints' is the logical OR of ANDs. The client MUST NOT specify this parameter if this resource's capabilities (Section XXXX?) indicate that constraint support is not available. The client MUST NOT specify both a 'or-constraints' and a 'constraints' parameter.

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.

5.3.4. Capabilities

The URI providing this resource supports all capabilities documented in Section 7.7.2.2.4 (with identical semantics), plus additional capabilities. In particular, the capabilities are defined by a JSON object of type `FilteredMultiCostMapCapability`:

```
object {  
  CostMode    cost-modes<0..*>;  
  CostType    cost-types<0..*>;  
  JSONBool    cost-constraints;  
  JSONNumber  max-cost-types; [OPTIONAL]  
} FilteredMultiCostMapCapability;
```

with members:

`cost-modes` See Section 4.2.5 of this MC draft

`cost-types` See Section 4.2.5 of this MC draft

`max-cost-types` Indicates the maximum number of cost values the ALTO Server can provide in a multi-cost array of a Multi-Cost Map.

`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.

5.3.5. Response

See Section on Multi Cost Map Service of this draft 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.

5.3.6. Example 1

```
POST multi/multicostmap/filtered HTTP/1.1
Host: alto.example.com
Content-Type: application/alto-multicostmapfilter+json
Accept: application/alto-multicostmap+json,application/alto-error+json
```

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

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

```
{
  "meta" : {},
  "data" : {
    "cost-mode" : ["numerical", "numerical"],
    "cost-type" : ["routingcost", "hopcount"],
    "map-vtag" : "1266506139",
    "map" : {
      "PID1": { "PID1": [1,6], "PID2": [5,23], "PID3": [10,5] }
    }
  }
}
```

5.3.7. Example 2

This is an example of using constraints to restrict returned source/destination PID pairs to those with 'routingcost' between 5 and 10, or 'hopcount' equal to 0.

```

POST multi/multicostmap/filtered HTTP/1.1
Host: alto.example.com
Content-Type: application/alto-multicostmapfilter+json
Accept: application/alto-multicostmap+json,application/alto-error+json

```

```

{
  "cost-mode" : ["numerical", "numerical"],
  "cost-type" : ["routingcost", "hopcount"],
  "or-constraints" : [ ["[0] ge 5", "[0] le 10"],
                        ["[1] eq 0"] ]
  "pids" : {
    "srcs" : [ "PID1", "PID2" ],
    "dsts" : [ "PID1", "PID2", "PID3" ]
  }
}

```

```

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

```

```

{
  "meta" : {},
  "data" : {
    "cost-mode" : ["numerical", "numerical"],
    "cost-type" : ["routingcost", "hopcount"],
    "map-vtag" : "1266506139",
    "map" : {
      "PID1": { "PID2": [5,23], "PID3": [10,5] },
      "PID2": { "PID2": [1,0] },
    }
  }
}

```

5.4. Endpoint Multi-Cost Service

The Endpoint Multi-Cost Service provides information on several Cost Types between individual Endpoints.

This service 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 costs between the PIDs corresponding to the endpoints if these values are available for the requested Cost Types.

When the cost values are requested to perform multi-variate numerical

optimization and are each available in the 'numerical' mode, then the ALTO Client SHOULD request the 'numerical' mode in order to get a reliable result. Note that this consideration is outside the scope of the ALTO protocol as it relates to the responsibility of the ALTO Client and related entries. However common sense lead to warn that a necessary condition for vector ranking method to be reliable is that the components of the processed vectors are numerical and not ordinal values.

5.4.1. Media Type

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

5.4.2. HTTP Method

This resource is requested using the HTTP POST method

5.4.3. 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-endpointmulticostparams+json", which is a JSON Object of type ReqEndpointMultiCostMap:

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

object{
    CostType      cost-type<0..*>;
    CostMode      cost-mode<0..*>;
    JSONString    constraints<0..*>;    [OPTIONAL]
    JSONArray     or-constraints<0..*>; [OPTIONAL]
    EndpointFilter endpoints;
} ReqEndpointMultiCostMap;
```

with members:

`cost-mode` Defined equivalently to the "cost-mode" input parameter of a Filtered Multi Cost Map.

`cost-type` Defined equivalently to the "cost-type" input parameter of a Filtered Multi Cost Map.

`constraints` Defined equivalently to the "constraints" input parameter of a Filtered Multi Cost Map.

`or-constraints` Defined equivalently to the "or-constraints" input parameter of a Filtered Multi Cost Map.

`endpoints` A list of Source Endpoints and Destination Endpoints for which Path multiple 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 10.3 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.

5.4.4. Capabilities

See section on Filtered Multi Cost Map capabilities in this draft.

5.4.5. Response

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

```
object EndpointDstMultiCosts {
  JSONArray [TypedEndpointAddr];
  ...
};

object {
  EndpointDstMultiCosts [TypedEndpointAddr]<0..*>;
  ...
} EndpointMultiCostMapData;

object {
  CostMode          cost-mode<0..*>;
  CostType          cost-type<0..*>;
  EndpointMultiCostMapData map;
} InfoResourceEndpointMultiCostMap;
```

InfoResourceEndpointMultiCostMap has members:

cost-type<0..*> The Cost Types used in the returned Cost Map.

cost-mode<0..*> The Cost Mode for each of the Cost Types used in the returned Cost Map.

map The Endpoint Multi-Cost Map data itself.

EndpointMultiCostMapData 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 EndpointDstMultiCosts object denotes the cost vector associated 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.

5.4.6. Example

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

```
{
  "cost-type" : ["routingcost", "hopcount"],
  "cost-mode" : ["numerical", "numerical"],
  "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-endpointmulticost+json
```

```
{
  "meta" : {},
  "data" : {
    "cost-type" : ["routingcost", "hopcount"],
    "cost-mode" : ["numerical", "numerical"],
    "map" : {
      "ipv4:192.0.2.2": {
        "ipv4:192.0.2.89" : [1, 7],
        "ipv4:198.51.100.34" : [2, 4],
        "ipv4:203.0.113.45" : [3, 2]
      }
    }
  }
}
```

6. IANA Considerations

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

[ID-alto-protocol],

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

6.1. Information for IANA on proposed Cost Types

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

6.2. Information for IANA on proposed Endpoint Properties

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

7. Acknowledgements

The authors would like to thank Dave Mac Dusan and Vijay Gurbani for fruitful discussions and comments on this draft and previous versions.

Sabine Randriamasy is partially supported by the MEDIEVAL project (<http://www.ict-medieval.eu/>), 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 MEDIEVAL project or the European Commission.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC5693] "Application Layer Traffic Optimization (ALTO) Problem Statement", October 2009.

8.2. Informative References

- [ID-ALTO-Requirements13]
"draft-ietf-alto-reqs-13.txt", January 2012.
- [ID-alto-protocol]
, Eds., "ALTO Protocol" draft-ietf-alto-protocol-10.txt",
October 2011.
- [ID-alto-protocol-11]
, Eds., "ALTO Protocol" draft-ietf-alto-protocol-11.txt",
March 2012.
- [draft-jenkins-alto-cdn-use-cases-01]
"Use Cases for ALTO within CDNs"
draft-jenkins-alto-cdn-use-cases-01", June 2011.
- [draft-randriamasy-alto-cost-schedule-01]
"ALTO Cost Schedule", July 2012.
- [draft-randriamasy-alto-multi-cost-05]
"Multi-Cost ALTO", October 2011.

Authors' Addresses

Sabine Randriamasy (editor)
Alcatel-Lucent Bell Labs
Route de Villejust
NOZAY 91460
FRANCE

Email: Sabine.Randriamasy@alcatel-lucent.com

W. D. Roome
Alcatel-Lucent Bell Labs
600 Mountain Ave.
Murray Hill, NJ 07974
USA

Phone:
Fax:
Email: W.Roome@alcatel-lucent.com
URI:

Nico Schwan

Email: ietf@nico-schwan.de

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: August 18, 2013

M. Scharf, Ed.
V. Gurbani, Ed.
G. Soprovich
V. Hilt
Alcatel-Lucent
February 14, 2013

The Virtual Private Network (VPN) Service in ALTO: Use Cases,
Requirements and Extensions
draft-scharf-alto-vpn-service-00

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 resources from a candidate set. This document provides motivation for using ALTO in a Virtual Private Network (VPN) environment. We discuss use cases, requirements, and possible extensions to the base ALTO protocol that will be needed to support VPN services.

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 August 18, 2013.

Copyright Notice

Copyright (c) 2013 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. Overview	3
2. Terminology	4
3. Encompassing example	4
3.1. A VPN scenario	4
3.2. Exemplary use of ALTO	6
4. Use cases	9
4.1. Use case 1: Application guidance in an L3VPN	10
4.2. Use case 2: Application guidance in an L2VPN	11
4.3. Use case 3: VPN guidance without addresses	12
4.4. Use case 4: Extending the VPN	12
4.5. Use case 5: Shrinking the VPN	13
4.6. Use case 6: VPN selection	14
5. Requirements and gap analysis	14
5.1. Requirements	14
5.2. Gap analysis	15
5.3. Differences from other proposed ALTO extensions	16
6. Security considerations	18
7. IANA considerations	18
8. References	18
8.1. Normative References	18
8.2. Informative References	18
Appendix A. Acknowledgements	19
Authors' Addresses	19

1. Overview

Virtual Private Network (VPN) technology is widely used in public and private networks to create groups of users that are separated from other users of the network and allows these users to communicate among them as if they were on a private network. According to [RFC4364], the generic term "Virtual Private Network" is used to refer to a specific set of sites as either an intranet or an extranet that have been configured to allow communication. A site is a member of at least one VPN and may be a member of many.

Service providers offer different types of VPNs. [RFC4026] distinguishes between Layer 2 VPN (L2VPN) and Layer 3 VPN (L3VPN) using different sub-types. Virtual Private LAN Service (VPLS) is an L2VPN provider service that emulates the full functionality of a traditional Local Area Network (LAN) [RFC4762]. A VPLS makes it possible to interconnect several LAN segments over a packet switched network.

Another solution is an L3VPN, which interconnects sets of hosts and routers based on Layer 3 addresses. In this context, a virtual private network is defined in [RFC4364] as follows:

Consider a set of "sites" that are attached to a common network that we call "the backbone". Now apply some policy to create a number of subsets of that set, and impose the following rule: two sites may have IP interconnectivity over that backbone only if at least one of these subsets contains them both.

These subsets are Virtual Private Networks (VPNs). Two sites have IP connectivity over the common backbone only if there is some VPN that contains them both. Two sites that have no VPN in common have no connectivity over that backbone.

VPNs can also include "pseudo L1/L2" connectivity, such as pseudowire emulation (PWE) carrying legacy TDM or ATM circuits for point to point connectivity. Further examples are integrated optical solutions delivering light paths or integrated optical and Ethernet transport. It is instructive to note that point-to-point VPN services of this type rarely carry VPN edge addresses within the network; e.g., packets are encapsulated and transported without any kind of address facing the customer drop side of the network.

A VPN may also include mechanisms to enhance the level of separation (e.g., by end-to-end encryption), but the discussion of such mechanisms is outside the scope of this document. In the following, the term "VPN" is used to refer to provider supplied virtual private networking.

The ALTO protocol [I-D.ietf-alto-protocol] is designed to provide 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 most important use case is providing application guidance in the global Internet, so that applications do not have to perform excessive measurements on their own. For the very same reason, topology exposure is also very useful in VPNs. But the constraints for using ALTO in L3VPNs or L2VPNs differ from the public Internet. This document presents these use cases and discusses requirements and extensions to the base ALTO protocol that will be needed to realize the VPN Service in ALTO.

2. Terminology

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

3. Encompassing example

3.1. A VPN scenario

Below, we present an example for a VPN scenario that describes an environment for an ALTO VPN Service. This scenario is subsequently used to analyze specific use cases.

We consider the following: there are two distinct entities, one, the network service provider (NSP) who owns the network and offers a VPN to the second entity, the customer, who has premises in four different locations that shall be interconnected by that VPN. The sites could be office branches, data centers, etc. Throughout this document, we assume the following four sites:

- o Site 1

- Location name: SITE-CHICAGO

- Geography Degree: 41.85 N, 87.65 W

- o Site 2

- Location name: SITE-OTTAWA

- Geography Degree: 45.24 N, 75.43 W

- o Site 3

Location name: SITE-SANFRANCISCO

Geography Degree: 37.75 N, 122.28 W

- o Site 4

Location name: SITE-PARIS

Geography Degree: 48.86 N, 2.35 E

It is assumed that these sites are interconnected by a VPN that may be identified by the hypothetical name "vpn42". This document specifically considers two different VPN types for the interconnection:

- o L3VPN: The local area networks at each site will have a certain IP subnet ranges, for instance 10.0.1.0/24 at site 1, 10.0.2.0/24 at site 2, etc.
- o L2VPN: All sites form part for a flat sub-IP network, e.g. a logical Ethernet segment. Different to a local network, the network potentially interconnects geographically remote sites.

The VPN will not necessarily be static. The customer could possibly modify the VPN and add new VPN sites, e. g., to handle peak-load demand or to consolidate VPN sites to account for reduced traffic. The service provider could offer a Web portal or other Operation Support Systems (OSS) solutions that allow the customer to grow or consolidate the VPN. Details on how the customer can configure VPNs are outside the scope of this document.

Furthermore, we assume that the customer is running at least one application that can benefit from application-level traffic optimization, e.g., using application-internal routing mechanisms or placement functions. For instance, typical uses cases for VPN customers could be:

- o Enterprise application optimization: Enterprise customers often run distributed applications that exchange large amounts of data, e.g., for synchronization of replicated data bases. Both for placement of replicas as well as for the scheduling of transfers insight into network topology information could be useful.
- o Private cloud computing solution: An enterprise customer could run own data centers at the four sites. The cloud management system could want to understand the network costs between different sites

for intelligent routing and placement decisions of Virtual Machines (VMs) among the VPN sites.

- o Cloud-bursting: One or more VPN endpoints could be located in a public cloud. If an enterprise customer needs additional resources, they could be provided by a public cloud, which is accessed through the VPN. Network topology awareness would help to decide in which data center of the public cloud those resources should be allocated.

These examples focus on enterprise customers of NSPs, which are typical users of provider-supplied VPNs. Such VPN customers typically have no insight into the network topology that transports the VPN. For instance, the actual delay between two VPN sites may significantly depend on the routing in the NSP MPLS/IP network. If better-than-random decisions are required, applications have to rely on own measurements. An alternative would be guidance by an ALTO server offered by the NSP.

It is important to emphasize that other scenarios and use cases exist and the examples enunciated so far are merely used to illustrate how ALTO can be used in a VPN context. A common characteristic of these use cases is that applications will not necessarily run in the public Internet, and they will typically not be accessed by residential customers. The internal use of ALTO by a specific application is not considered in this document.

3.2. Exemplary use of ALTO

In the example VPN described in the previous section, it would be beneficial if an ALTO server would expose cost maps or provide a ranking service that represents the costs between different sites, e.g., endpoints of the VPN. Similar to existing use cases of ALTO, this enables an application integrating an ALTO client to use this information for application-level traffic optimization. This results in the following scenario:

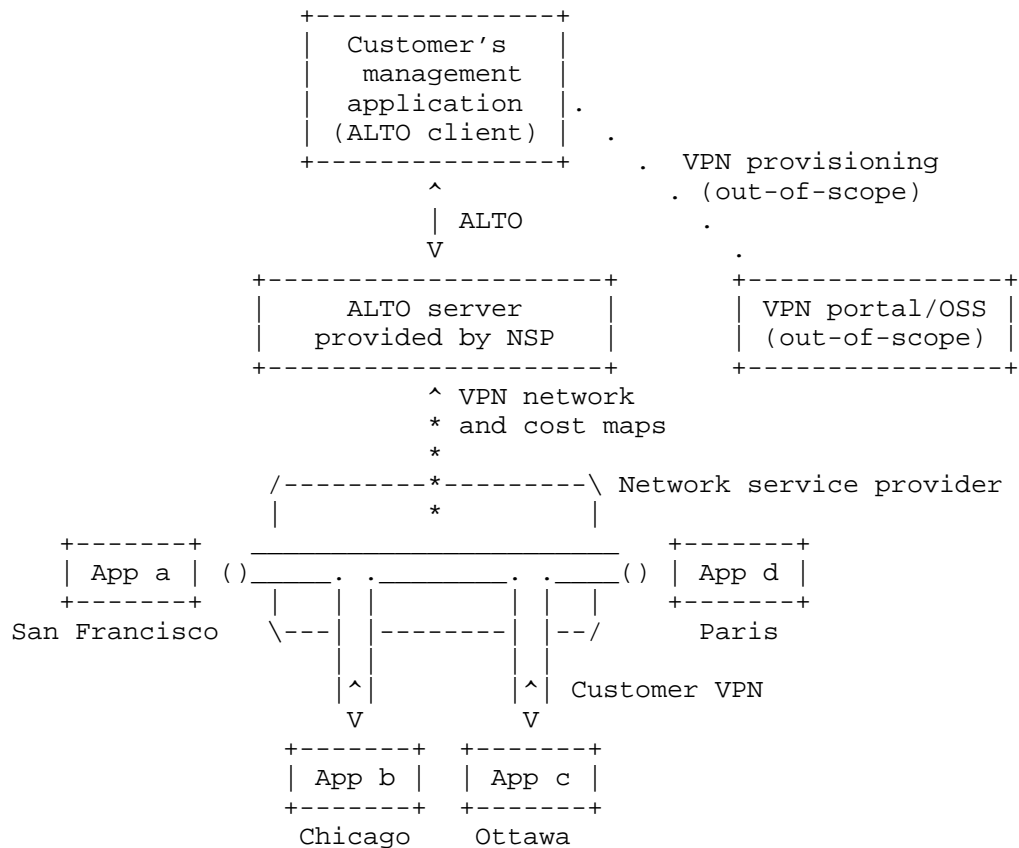


Figure 1: Overview of an ALTO usage scenario

The network service provider could operate an ALTO server. An ALTO client in an application could then retrieve an ALTO cost map by querying a corresponding URI, such as:

```
uri: http://alto.nsp.org/vpn42/costmap
```

The NSP can assign PIDs to each of the VPN endpoints; this renders computations at the ALTO server to fit in the current model of using the protocol. A corresponding example would be:

Site 1: PID "pid14"

Site 2: PID "pid21"

Site 3: PID "pid11"

Site 4: PID "pid27"

The example below further expands on the VPN by demonstrating the resulting network topology provided to an ALTO server. The picture corresponds to the VPN of the customer and also includes the Provider Edge (PE) routers and Customer Edge (CE) devices:

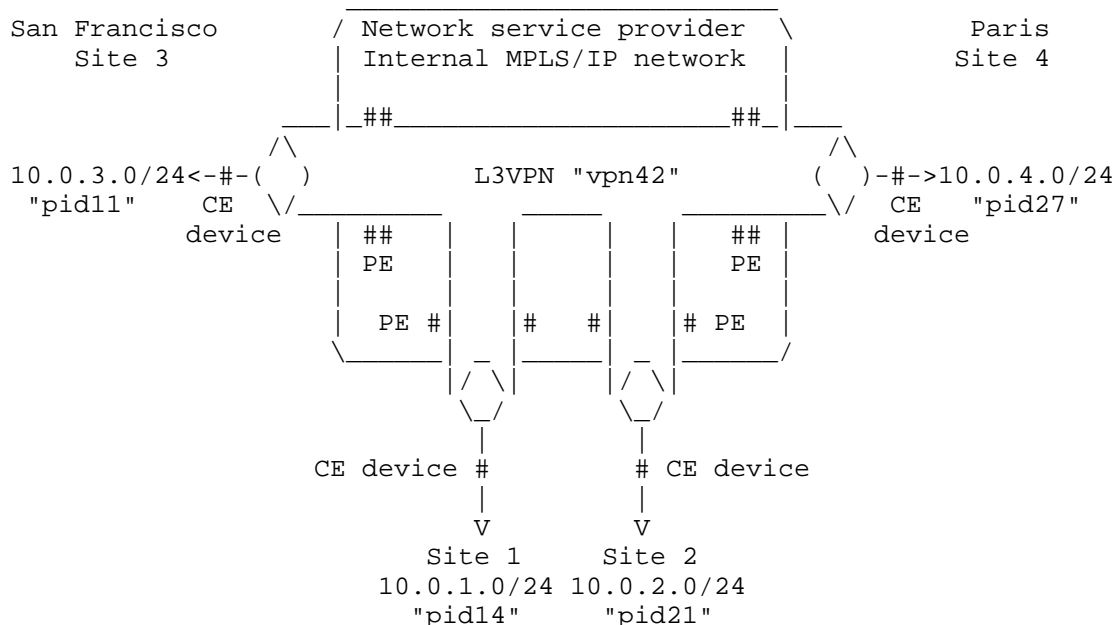


Figure 2: Example for mapping of VPN sites to ALTO PIDs

The costs exposed by the ALTO server can be based on routing costs inside the service provider network or other network topology information, such as delay measurements, traffic engineering (TE) data, etc. As with other use cases of ALTO, the costs can reflect the service provider's preference and policies regarding communication between the involved VPN endpoints.

Generally, two different types of applications can consume the information provided by the ALTO server. The first class can be composed of discrete application instances executing at the various sites that are interconnected by the VPN. ALTO is used to optimize the routing or resource consumption among those application instances. A typical examples is a distributed database, i.e., an enterprise backend system. In Figure 1, these application instances are referred to as "App a", "App b", "App c", and "App d". Generally speaking, this usage mirrors the canonical use of ALTO in

unstructured P2P networks or Content Delivery Networks (CDN) networks whereby a rendezvous is desired between a consumer and a plurality of producers. In this document, we label this class of applications by the term "user applications".

The second class represents management applications that typically work on VPN level. In addition to consulting an ALTO server provided by the NSP, this type of application possibly has its own understanding of what resources are available at different sites, and it could possibly even trigger more complex actions such as building out VPNs, e. g., by contacting a VPN portal of the NSP. In Figure 1, as well as the rest of this document, uses the term "management application" for this use case. An example would be an orchestration solution for cloud computing resources. It could use the topology and cost maps illustrated in Figure 2 to control VPN placement. In principle, management applications have some similarity to centralized resource directories in P2P networks (e. g., trackers), which are an important existing use case for ALTO. Yet, unlike resource directories in the Internet, a VPN typically interconnects mainly sites within one administrative domain.

There may also be an overlap between both types of applications. Furthermore, in particular for the first class of applications, the customer could run an own ALTO server, which could expose topology map and cost maps with further details only visible to the VPN customer (e.g., network segments behind NATs). Since such information is independent of the use of a VPN and typically not known to an NSP, these usage scenarios are not further detailed in this memo.

4. Use cases

Current VPNs provide no clear mechanism to convey information about the network infrastructure to management or user applications using that VPN, e.g. regarding preferences or topological properties of the network service provider network. Applications thus have to rely on other mechanisms such as local measurements to optimize their traffic. The ALTO protocol has been designed to overcome such limitations in the Internet. ALTO, being a well-established, generic, and flexible protocol, can be used in VPNs, too.

We now present various use cases that exhibit the utility of considering a VPN extension of ALTO. Through a series of use cases we demonstrate how a VPN customer and the NSP can use ALTO; we also highlight similarities and differences when using ALTO in the general Internet.

4.1. Use case 1: Application guidance in an L3VPN

The NSP providing the L3VPN service can offer an ALTO server that exposes network and cost information to applications running traffic over that VPN. Since an L3VPN is IP-based, this use case is in principle similar to the use cases already addressed by the ALTO base protocol.

Example 1: Consider the customer in Section 3.1 that has four VPN sites. A user application in one site (say Site 1) would now like to find out which of the other sites (Site 2 to Site 4) are topologically close to Site 1, perhaps to determine where to replicate a certain data set. A corresponding ALTO query would return the costs between those sites. The user application could then select a host in the corresponding subnetwork and connect to that endpoint.

Example 2: In addition to network proximity information, the user application could also be interested in guidance regarding network parameters that cannot be measured directly. For instance, a relevant parameter for a VPN site could be the level of redundancy for that VPN site, e.g., whether there is resilience by network protection schemes in the NSP network.

Example 3: It is quite common for VPN Customer Equipment (CE) to be multi-homed at the Provider Edge (PE). A CE may well home into to several PE routers and thus may have different network cost functions. For instance, assume that in Site 1 the CE will peer to a local PE1 and remote PE2. The cost to reach Site 2 in the VPN could be 1575 for PE1 and 2250 for PE2. The CE will thus choose to steer traffic from Site 1 to Site 2 toward PE1. While the realization of such traffic steering is outside the scope of this document, CE multi-homing places an explicit need to expose more than one set of network costs for a VPN endpoint.

In principle, the existing ALTO services such as network and cost map can provide such guidance. However, it is important to note that a VPN might not run in a public environment. The IP address ranges inside a VPN might not be globally unique or routable. Furthermore, a provider based VPN service normally maintains a strict separation between service provider addressing (such as addresses or Provider Edge routers) and customer addressing. As a result, an ALTO server will not expose the internal IP addressing of the network service provider, making it difficult to identify services using IP addresses in general. In a BGP L3VPN, the VPRN BGP Route Distinguisher could possibly be used as a service identifier, but it is unclear whether an application of a customer or the ALTO client will indeed know such network-internal information of the NSP and whether the NSP would

want to expose it. Also, it would make sense to define an ALTO VPN extension independent of a specific VPN technology.

The network costs in a VPN depends on VPN topology, which needs to be taken into consideration when calculating ALTO information. Given that VPNs are often offered by a single network service provider, ALTO cost information could include information that may be available for a single autonomous system, but difficult to gather in the Internet as a whole. Examples would be the provisioned bandwidth, network-internal latencies, or the path resilience. In a static VPN environment e.g. with a reserved resources in an MPLS/IP wide area network, these costs can be assumed to be rather stable and e. g. reflect the reserved bandwidth between VPN sites. For an application it is simpler and less intrusive to obtain such information about the VPN from the network instead of performing measurements, which would possibly require special probe instances at the different VPN sites (e. g., data centers). But as the encoding of such costs in ALTO is independent of the usage of a VPN, this document does not mandate any specific way how to build ALTO cost maps.

4.2. Use case 2: Application guidance in an L2VPN

The use case outlined in Example 1 also exists for L2VPNs, which are an important technology to transparently interconnect different LAN segments of enterprise users. Again, applications could benefit from getting insight into topological properties of the wide area network providing the L2VPN service, in order to avoid the overhead of own measurements.

Example 4: The user application described in Section 3.1 again wants to find out how well connected (topologically close) Site 1 is to Site 2, 3, or 4. Different from the previous example, all sites are now part of the same Layer 2 subnet. Another example for an application that would benefit from ALTO is a cloud management system. Such a management application could be interested in finding out whether migrating of a Virtual Machine from Site 1 to another site would improve performance, perhaps due to better connectivity or lower latency.

While this use case is in principle similar to the previous one, there is a major difference regarding addressing: Unlike the L3VPN, an L2VPN is not necessarily IP-based; it may use MAC addresses instead of IP addresses. While IP addresses can be aggregated easily and represented succinctly using CIDR notation, MAC addresses do not lend themselves to such aggregation and representation. Furthermore, MAC addresses are not useful to applications themselves. And finally, MAC addresses may not readily be known and available to an ALTO server of the network service provider. And even if they are,

an ALTO map using MAC addresses will be very large. In summary, use of MAC addresses is not scalable and nor does it denote any hierarchy that can be used for aggregation. Some other means of identifying services and hosts will be required when using ALTO in L2VPNs.

4.3. Use case 3: VPN guidance without addresses

The VPN interconnects different sites through the network service provider's network. An application might be interested in getting topology information among those sites without knowing actual addresses or identifiers used internally by the VPN. In fact, a VPN site may not even have an address known or visible to applications, e.g., a pseudo-wire VPN.

Example 5: A management application might ask for all VPN sites (i. e., corresponding PIDs) that have a delay less than 40ms or a routing cost less than 55, from VPN Site 1. A specific example for such an application might be cloud management system that uses application-level traffic optimization mechanisms. In the scenario introduced in Section 3.1, such an application may have a-priori information, learned from e.g. a VPN portal, about the VPN type and/or VPN identifiers ("vpn42") as well as some unique site identifier such as "SITE-CHICAGO" but no network addresses. The query could also be more complex or include constraints, e. g., limited to a particular TE class. Note that the ATLO protocol does not necessarily have to support the query constraint itself; if corresponding maps are available, the application can analyze the data itself.

Example 6: In absence of well-known existing network identifiers, a management application might want to query for VPN sites based on yet other attributes, such as geographical distance. For example, an application might want to find all the VPN sites (i. e., corresponding PIDs) within 50 KM of 135.07 W and 61.22 N. Such geographic queries would be typical of policies bounding delay by geographic distance or administrative and legal requirements.

Such application guidance is obviously similar to existing use of the ALTO cost map or ranking services except that the queries are not based on network addresses.

4.4. Use case 4: Extending the VPN

The customer can possibly grow the VPN to include new sites that are connected at a later time to the VPN. The actual mechanisms for VPN reconfiguration are outside the scope of this document.

Example 7: A management application could be interested in guidance for VPN sites that are currently not part of the VPN, but that would

be available e. g. to increase capacity or geographic coverage. Assume that two sites Site 1 and Site 2 are already connected to the VPN. Some time later, scale-out to a third site is required, and the application has to decide whether Site 3 or 4 is better suited for a new application instance. This is an realistic example for a cloud management system that is geographically distributed. Such a system would then have to decide whether Site 3 or Site 4 is topologically closer to the existing VPN endpoints, in order to determine the best location from the network point of view. An ALTO server could provide guidance on the offnet distance of Sites 3 and 4 to the existing VPN sites.

Apparently, the question whether to actually extend the VPN in a specific way may also include decisions outside the scope of ALTO, such as price information or other commercial or legal policies. The actual VPN re-configuration and attachment of a new site to the VPN topology requires back-office interaction and provisioning actions by separate, orthogonal mechanisms such as a Path Computation Element (PCE). Actual path setup by a PCE is independent from the selection of a suitable target site. But it makes sense to use the well-established ALTO methods in order to get at an early stage network proximity information as input information for the selection and configuration process. Applications typically cannot measure the network performance to destinations not already part of the VPN.

For a network service provider, customer guidance for VPN extension by ALTO offers a new possibility to optimize its internal traffic engineering. For instance, an operator could recommend to customers not to connect to a destination operating in protection mode, e.g., after a fiber cut, because in such a case the network may have less sparse resources. Note that a customer is not able to measure such constraints. ALTO is a simple interface to expose such information to applications.

From an ALTO perspective, growing VPN sites possibly results in different types of endpoints, some of which may exist a-priori but not be reachable within the VPN. They could possibly be understood as "shadow" PIDs that become active once the VPN is extended. Once the VPN is modified, new endpoints or PIDs may be created, i. e., the ALTO network and cost maps may have to be updated accordingly after the VPN is re-configured.

4.5. Use case 5: Shrinking the VPN

Much like a VPN may grow dynamically, it can also shrink when the resources in the VPN are underutilized. Instead of keeping the underutilized resources alive, the VPN operator may decide to consolidate the resources and remove sites from the VPN.

Example 8: Once again, consider the customer in Section 3.1 that has four VPN sites. Based on low resource demand, the management application may wonder whether Site 1 (Chicago) and Site 2 (Ottawa) can be consolidated, e. g., by moving resources between them. One important constraint for such a decision could network proximity information. After such a consolidation, the VPN network and cost maps will be updated to reflect the new topology.

From an ALTO server perspective, this use case is similar to a general application guidance. Yet, there could be a benefit for the service provider to provide special guidance regarding removal of VPN endpoints if there is a benefit for its internal traffic engineering (e. g., consolidation of network resources used by several VPN customers).

4.6. Use case 6: VPN selection

In a more advanced use case, ALTO could also be a selection function to choose VPNs based on network cost criteria.

Example 9: In a multi-homing environment, ALTO could be used to select one VPN out of several candidates to reach a certain destination, taking into account smaller costs, e. g., according to distance or to preferences of the network service provider network.

This use case differs from the previous examples since more than one VPN is involved, i. e., the ALTO guidance is not used to perform application-layer traffic optimization within one VPN, but instead across different VPNs.

5. Requirements and gap analysis

5.1. Requirements

Based on the scenarios listed in Section 4, several requirements can be derived for a VPN Service in ALTO:

REQ 1: The existing ALTO protocol and RESTful interface should be used as far as possible to enable an NSP to expose properties of a VPN.

REQ 2: A VPN Service must not require that network service provider expose internal addressing, such as internal addresses or loopback addresses of the Provider Edge (PE) routers.

REQ 3: A VPN Service must use the PID concept of the base ALTO protocol as far as possible, i. e., the VPNs and network entities in

the VPNs can be identified by PIDs. This permits use of the existing ALTO services such as the map service for VPNs, as well as the inherent topology abstraction provided by ALTO.

REQ 4: A VPN Service must be possible for different VPN types, i. e., it must not be limited to L3VPNs only.

REQ 5: The VPN Service must support use cases where IP addresses are not the only form of network identification.

REQ 6: If IP addresses are used, a VPN Service must not assume that IP address are globally routable or unique.

REQ 7: A VPN Service should include certain attributes that are unique to a VPN and that are not represented by the current set of attributes in the base ALTO protocol. Examples include location name of a site, geography coordinates (degree/digital), role, default policy, or geography restriction.

REQ 8: The PID must be selectable using standard ALTO filtering. A standard interface query should allow finding resources using, say, the location name attribute or the geography attributes.

REQ 9: The PID should be selectable using a filter that computes matching sites within a certain distance of a particular geographic coordinate based on latitude and longitude, in case that no other address information is known in advance.

REQ 10: Incremental build out (as well as the shrinking) of resources that are part of the VPN must be supported, i. e., the ALTO VPN service should also be able to expose information about new sites to be attached to the VPN, or provide guidance for removal of sites.

REQ 11: Information about a VPN must only be exposed to authorized users of that VPN.

5.2. Gap analysis

In the following we analyze to which extent the requirements of a VPN Service can be met by the existing ALTO protocol.

REQ 1: This is an inherent, general requirement for any new use or extension of ALTO.

REQ 2: This requirement can be supported in ALTO today, because it is left to the service provider which information to expose e.g. in ALTO cost maps.

REQ 3: The PID concept itself is generic and thus can fulfill this requirement.

REQ 4: L3VPNs are rather similar to existing use cases of ALTO in the Internet. Insofar as L2VPNs or pseudo-wire VPNs have the notion of some address, ALTO seems to be able to handle these through an extension that extends the definition of an address to include other identifiers besides IP addresses.

REQ 5: Use of ALTO with network identifiers that are not IP addresses requires work. There is a need to analyze how to name VPNs and endpoints and how to achieve a mapping to the information stored in the ALTO server.

REQ 6: ALTO can be used as of now with IP address ranges that are not globally routable. However, it must be emphasized that private VPN environments without uplink to the global Internet may only have connectivity to a limited number of IP subnets, i. e., the ALTO server will not be able to provide any reasonable guidance for most parts of the IP address space. Also, the ALTO server operator must take into account that IP address ranges in different VPNs may overlap, possibly also with the transport network infrastructure (e. g., PE routers).

REQ 7: Extensions to ALTO will be needed.

REQ 8: Assuming extensions in REQ 7, filtering should be fairly easy.

REQ 9: Extensions to ALTO will be needed, aligned with REQ 6.

REQ 10: This requirement will possibly require extensions to ALTO, e. g., to distinguish between endpoints that are already attached to the VPN and sites outside the VPN. Changes of the VPN topology are likely to change the ALTO maps, i.e., standard ALTO mechanism for incremental updates and push notifications would be of added value.

REQ 11: Existing authentication and access control mechanisms for ALTO could be sufficient to meet this requirement, subject to further analysis.

5.3. Differences from other proposed ALTO extensions

There have been various other proposals for ALTO extensions. In the following, we discuss why none of these extensions addresses the requirements of using ALTO in VPNs.

A use case of ALTO for traffic optimization in high bandwidth core networks is discussed in [I-D.bernstein-alto-large-bandwidth-cases].

It is proposed to enhance ALTO by bandwidth constraint representations, focusing on high-speed circuit switched optical networks that have a fixed capacity. However, L2VPNs or L3VPNs can be deployed in an MPLS/IP network without any bandwidth guarantees. An encoding of network parameters such as bandwidth in ALTO is therefore entirely orthogonal to the use of VPNs. The ALTO extensions suggested by [I-D.bernstein-alto-large-bandwidth-cases] are therefore not required by the use cases summarized in this document.

A related extension proposal [I-D.lee-alto-app-net-info-exchange] defines enhanced filtering constraints for ALTO, as well as a constrained cost graph encoding. The objective of the filtering is to retrieve paths or graphs for given constraints (e.g., bandwidth, latency, hop count, packet loss, etc.). This proposal basically enhances the way how ALTO can represent the costs in a network. However, the core challenge in VPNs is the addressing and lookup of VPN endpoints. With the VPN service, ALTO can be used in L2VPNs or L3VPNs with the existing encodings for cost maps, i. e., the extensions of [I-D.lee-alto-app-net-info-exchange] are orthogonal as well.

A general data center resource information model has been suggested in [I-D.lee-alto-ext-dc-resource]. According to that model, the ALTO server also includes data-center information not related at all to the network, such as compute resources, memory, power consumption, etc. This implies a significant extension of the scope of ALTO. While VPNs are an important technology to interconnect data centers, the ALTO VPN service solely focuses on networking cost, and ALTO extensions are limited to the minimum set of additional protocol features that are required in a VPN context. This memo does not argue that ALTO shall be used as a generic data center information exchange protocol.

[I-D.xie-alto-sdn-use-cases] presents an architecture how ALTO can be used if data-forwarding plane and control plane are separated. In such an architecture, ALTO could be used to exchange connectivity information between controllers in different domains. This proposal is entirely disjoint to the problem addressed by this document. Since the separation of data-forwarding and control plane is an internal network design issue, it does not matter for the ALTO VPN service how Network Service Provider control their infrastructure, and existing management solutions can be applied as well. Even though the realization of network control and management of VPNs is outside the scope of this document, we note that existing L2VPN and L3VPN solutions often integrate data-forwarding and control plane.

In summary, this document proposes a small and well-focused extension

to enable the use of ALTO in VPN environments, given that the current address types and information models of ALTO is not sufficient in some cases. This document does explicitly not suggest any non-networking or technology-specific ALTO extension.

6. Security considerations

TBD.

7. IANA considerations

TBD.

8. References

8.1. Normative References

- [I-D.ietf-alto-protocol]
Alimi, R., Penno, R., and Y. Yang, "ALTO Protocol",
draft-ietf-alto-protocol-13 (work in progress),
September 2012.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4026] Andersson, L. and T. Madsen, "Provider Provisioned Virtual
Private Network (VPN) Terminology", RFC 4026, March 2005.
- [RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private
Networks (VPNs)", RFC 4364, February 2006.
- [RFC4762] Lasserre, M. and V. Kompella, "Virtual Private LAN Service
(VPLS) Using Label Distribution Protocol (LDP) Signaling",
RFC 4762, January 2007.

8.2. Informative References

- [I-D.bernstein-alto-large-bandwidth-cases]
Bernstein, G. and Y. Lee, "Use Cases for High Bandwidth
Query and Control of Core Networks",
draft-bernstein-alto-large-bandwidth-cases-02 (work in
progress), July 2012.
- [I-D.lee-alto-app-net-info-exchange]
Lee, Y., Bernstein, G., Choi, T., Madhavan, S., and D.

Dhody, "ALTO Extensions to Support Application and Network Resource Information Exchange for High Bandwidth Applications", draft-lee-alto-app-net-info-exchange-01 (work in progress), January 2013.

[I-D.lee-alto-ext-dc-resource]

Lee, Y., Bernstein, G., and D. Dhody, "ALTO Extensions for Collecting Data Center Resource Information", draft-lee-alto-ext-dc-resource-01 (work in progress), January 2013.

[I-D.xie-alto-sdn-use-cases]

Xie, H., Tsou, T., Lopez, D., and H. Yin, "Use Cases for ALTO with Software Defined Networks", draft-xie-alto-sdn-use-cases-01 (work in progress), June 2012.

Appendix A. Acknowledgements

TBD.

Authors' Addresses

Michael Scharf (editor)
Alcatel-Lucent

Email: Michael.Scharf@alcatel-lucent.com

Vijay K. Gurbani (editor)
Alcatel-Lucent

Email: vkg@bell-labs.com

Greg Soprovich
Alcatel-Lucent

Email: Greg.Soprovich@alcatel-lucent.com

Volker Hilt
Alcatel-Lucent

Email: volker.hilt@bell-labs.com

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

LMAP
Internet-Draft
Intended status: Informational
Expires: August 22, 2013

J. Seedorf
NEC
V. Gurbani
Bell Labs, Alcatel-Lucent
E. Marocco
Telecom Italia
February 18, 2013

ALTO for LMAP
draft-seedorf-lmap-alto-00

Abstract

In the context of Large-Scale Measurement of Broadband Performance (LMAP), measurement results are currently made available to the public either at the finest granularity level (e.g. as a list of results of all individual tests), or in a very high level human-readable format (e.g. as PDF reports).

This document argues that there is a need for an intermediate way to provide access to large-scale network measurement results, flexible enough to enable querying of specific and possibly aggregated data. The Application-Layer Traffic Optimization (ALTO) Protocol, defined with the goal to provide applications with network information, seems a good candidate to fulfill such a role.

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 August 22, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Example Use Cases	5
3. Advantages of using ALTO	6
4. Examples	7
4.1. Download speeds	7
4.1.1. Network map	8
4.1.2. Cost map	9
5. References	10
5.1. Normative References	10
5.2. Informative References	10
Appendix A. Acknowledgment	11
Authors' Addresses	12

1. Introduction

Recently, there is a discussion on standardizing protocols that would allow measurements of broadband performance on a large scale (LMAP [I-D.schulzrinne-lmap-requirements]). In principle, the vision is that "user networks gather data, either on their own initiative or instructed by a measurement controller, and then upload the measurement results to a designated measurement server."

Apart from protocols that can be used to gather measurement data and to upload such data to dedicated servers, there is also a need for protocols to retrieve - potentially aggregated - measurement results for a certain network (or part of a network), possibly in an automated way. Currently, two extremes are being used to provide access to large-scale measurement results: On the one hand, highly aggregated results for certain networks may be made available in the form of PDFs or figures. Such presentations may be suitable for certain use cases, but certainly do not allow a user (or entity such as a service provider) to select specific criteria and then create corresponding results. On the other hand, complete and detailed results may be made available in the form of comma-separated-values (csv) files. Such data sets typically include the complete results being measured on a very fine-grained level and usually imply large file sizes (of result data sets). Such detailed result data sets are very useful e.g. for the scientific community because they enable to execute complex data analytics algorithms or queries to analyse results.

Considering the two extremes discussed above, this document argues that there is a need for an intermediate way to provide access to large-scale network measurement results: It must be possible to query for specific, possibly aggregated, results in a flexible way. Otherwise, entities interested in measurement results either cannot select what kind of result aggregation they desire, or must always fetch large amounts of detailed results and process these huge datasets themselves. The need for a flexible mechanism to query for dedicated, partial results becomes evident when considering use cases where a service provider or a process wants to use certain measurement results in an automated fashion. For instance, consider a video streaming service provider which wants to know for a given end-user request the average download speed by the end user's access provider in the end user's region (e.g. to optimize/parametrize its http adaptive streaming service). Or consider a website which is interested in retrieving average connectivity speeds for users depending on access provider, region, or type of contract (e.g. to be able to adapt web content on a per-request basis according to such statistics).

This document argues that use cases as described above may enhance the value of measurements of broadband performance on a large scale (LMAP), given that it is possible to query for selected results in an automated fashion. Therefore, in order to facilitate such use cases, a protocol is needed that enables to query LMAP measurements results while allowing to specify certain parameters that narrow down the particular data (i.e. measurement results) the issuer of the query is interested in. This document argues that ALTO [RFC5693] [I-D.ietf-alto-protocol] could be a suitable candidate for such a flexible LMAP result query protocol.

2. Example Use Cases

To motivate the usefulness of ALTO for querying LMAP results, consider some key use cases:

- o Video Streaming Service Provider: For HTTP adaptive streaming, it may be very useful to be able to query for average measurement values regarding a particular end user's access network provider. For instance, consider a video streaming service provider that queries LMAP measurement results to retrieve for a given end-user request the average download speed by the end user's access provider in the end user's region. Such data could help the service provider to optimize/parametrize its HTTP adaptive streaming service.
- o Website Front End Optimization: A website might be interested in statistics about average connectivity types or download speeds for a given end user request in order to dynamically adapt HTML/CSS/JavaScript content depending on such information (sometimes referred to as "Front End Optimization"). For instance, image compression may be employed depending on the average connectivity type of a user in a given region or with a given access network provider.
- o Troubleshooting: In general, any service on the Internet may be interested in LMAP data for troubleshooting. In case a service does not work as expected (e.g. low throughput, high packet loss, ...), it may be of value for the service provider to retrieve (fairly) recent measurement data regarding the host that is requesting the service.
- o TBD: add more use cases

3. Advantages of using ALTO

The ALTO protocol [I-D.ietf-alto-protocol] specifies a very lightweight JSON-based encoding for network information and can play an important role in querying the measurement results as we argue in Section 2.

ALTO is designed on two abstractions that are useful here. First is the abstraction of the physical network topology into an aggregated but logical topology. In this abstract topological view, referred to as "network map", individual hosts are aggregated into a well defined network location identifier called a PID. Hosts could be aggregated into the PID depending on certain identifying characteristics such as geographical location, serving ISP, network mask, nominal access speed, or any mix of them. The "network map" abstraction is essential for exporting network information in a scalable and privacy-preserving way.

The second abstraction that is useful for LMAP is the notion of a "cost map". Each PID identified in the network map can, in a sense, become a vertex in a cost map, and each edge joining adjacent vertices can have an associated cost. The cost can be defined by the measurement server and can indicate routing hops, the financial cost of sending data over the link, available bandwidth on the link with bottlenecked links increasing showing a smaller value, or a user-defined cost attribute that allows arbitrary reasoning.

The ALTO protocol defines several basic services based on such abstractions, but additional ones can be easily defined as extensions.

There are other advantages to using ALTO as well. The protocol is defined as a set of REST APIs on top of HTTP. The data carried by the protocol is encoded as JSON. Queries can be performed by clients locally after downloading the entire topological and cost maps or clients can send filtered requests to the ALTO server such that the ALTO server performs the required computation and returns the results. The protocol supports a set of atomic constraints related to equality that can be used to filter results and only obtain a set of interest to the query.

Additionally, protocol extensions that could also be useful for the LMAP usage scenario (e.g. extensions for incremental updates, for asynchronous change notifications and for encoding of multiple costs within the same cost map) have been proposed and are currently being discussed in the ALTO WG.

4. Examples

[NOTE: syntax most certainly wrong!]

4.1. Download speeds

This section shows, as an example, how average download speeds measured in a given time interval can be reported. The aggregation approach in this case is based on ISP and geographical location. Two types of data are reported in this example:

- o data collected from measurements against specific endpoints (e.g. active measurements);
- o data collected from all measurements (e.g. passive measurements).

4.1.1. Network map

```
{
  "meta" : {},
  "data" : {
    "map-vtag" : "1266506139",
    "map" : {
      "ISP1-GEO1" : {
        "ipv4" : [ "10.1.0.0/16", "172.20.0.0/16" ]
      },
      "ISP2-GEO1" : {
        "ipv4" : [ "10.2.0.0/17" ]
      },
      "ISP3-GEO1" : {
        "ipv4" : [ "10.3.0.0/16" ]
      },
      "ISP2-GEO2" : {
        "ipv4" : [ "10.2.128.0/17" ]
      },
      "ISP4-GEO2" : {
        "ipv4" : [ "10.4.0.0/16" ]
      },
      .
      .
      .

      "MSMNT-CL1" : {
        "ipv4" : [ "192.168.0.0/30" ]
      },
      "TOTAL" : {
        "ipv4" : [ "0.0.0.0/0" ]
      }
    }
  }
}
```

4.1.2. Cost map

```
{
  "meta" : {},
  "data" : {
    "cost-mode" : "numerical",
    "cost-type" : "avg-dl-speed",
    "map-vtag" : "1266506139",
    "time-interval" : "2629740",
    "map" : {
      "ISP1-GEO1": { "MSMNT-CL1" : 13.2,
                     "TOTAL" : 10.2},
      "ISP2-GEO1": { "MSMNT-CL1" : 11.4,
                     "TOTAL" : 12.3},
      "ISP3-GEO1": { "MSMNT-CL1" : 13.2,
                     "TOTAL" : 10.2},
      .
      .
      .
    }
  }
}
```

5. References

5.1. Normative References

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

5.2. Informative References

- [I-D.ietf-alto-protocol]
Alimi, R., Penno, R., and Y. Yang, "ALTO Protocol",
draft-ietf-alto-protocol-13 (work in progress),
September 2012.
- [I-D.schulzrinne-lmap-requirements]
Schulzrinne, H., Johnston, W., and J. Miller, "Large-Scale
Measurement of Broadband Performance: Use Cases,
Architecture and Protocol Requirements",
draft-schulzrinne-lmap-requirements-00 (work in progress),
September 2012.

Appendix A. Acknowledgment

Jan Seedorf is partially supported by the mPlane project (mPlane: an Intelligent Measurement Plane for Future Network and Application Management), a research project supported by the European Commission under its 7th Framework Program (contract no. 318627). 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 mPlane project or the European Commission.

Authors' Addresses

Jan Seedorf
NEC
Kurfuerstenanlage 36
Heidelberg 69115
Germany

Phone: +49 6221 4342 221
Fax: +49 6221 4342 155
Email: seedorf@neclab.eu

Vijay K. Gurbani
Bell Labs, Alcatel-Lucent

Email: vkg@bell-labs.com

Enrico Marocco
Telecom Italia
Via G. Reiss Romoli, 274
Turin 10148
Italy

Email: enrico.marocco@telecomitalia.it

ALTO
Internet-Draft
Intended status: Informational
Expires: August 26, 2013

H. Song
Y. Lee
Huawei
Feb 22, 2013

Infrastructure to Application Information Exposure
draft-song-alto-i2rs-01

Abstract

This document describes the scenarios that applications can use the network layer especially the network routing system exposed information, so as to optimize application layer traffic. The use cases in this document include the ISP broadband network (using P2P and CDN as examples) and the data center network. This document also describes what information should be collected for ALTO service for traffic optimization.

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 August 26, 2013.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

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

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. Use Cases	3
3.1. ISP network	3
3.2. Data Center Network	4
4. Open Discussion	6
5. Informative References	6
Authors' Addresses	7

1. Introduction

ALTO provides an interface to applications and appropriate information to guide an optimal node selection when there are more than one application node providing the same service, through aggregated network map and cost map. It usually aggregates network locations into PIDs, and assigns lower cost value for a PID pair that are topologically closer. So when application node follows the advice from ALTO server to choose one resource provider within a PID that has lower cost from its own PID, with higher probability the application node can keep the content request and response traffic flow intra domain. This can reduce interdomain traffic for ISPs, and avoid the congestion in the backbone network. More factors for node selection can be considered, such as pricing, congestion, and etc.

In order to assure optimality, the underlying infrastructure needs to expose its topology information, node/link status information, pricing information for path selection to ALTO server, which will either be abstracted as the network map or as the impacting input factor of cost map.

2. Terminology

I2AEX: Infrastructure to Application Exposure.

ALTO: application layer traffic optimization.

IaaS: Infrastructure as a Service.

3. Use Cases

3.1. ISP network

ISP broadband networks are consisted of interconnected autonomous systems. They run BGP protocols between the boarder gateway routers, and run IGP protocols intra autonomous systems. There are core routers, and access routers (BRAS) which fullfill the admission control through RADIUS servers, the access router connects to multiple aggregation switches. And an aggregation switch connects to multiple DSLAMs or OLTs. And the DSLAM or OLT connect to the home gateways or ONUs, which connect to the user devices. [It's better to give a figure here.]

The ISP network usually hide all its information to applications. But in the trend of big traffic use case, the main motivation is to reduce the backbone and interdomain traffic. CDN and P2P are the two

target applications. In a P2P application, the content requester requests contents from peers in the same swarm. It may choose a peer that is far away ignoring a peer that is topologically closer to it. Due to the topology ignorance, the application may create unnecessary backbone and interdomain traffic.

In another application of CDN, a popular resource usually is stored in multiple data centers. Without the knowledge of the network topology, the CDN's DNS server can redirect the content requester to a sub-optimal edge server, which is not always topologically close to the content requester. Even with measuring the round trip time between its edge servers and the requester's local DNS server, it may not get the accurate result because RTT is dynamic and user can specify its own DNS server.

For the above two broadband network use cases, the infrastructure information from the home to the access router does not help much. But more attention should be paid to the information from the access router to the routing system, which can provide important input for the ALTO maps. An ALTO server needs the following information as input:

- o The network segments information. Every access router manages one or more IP address pools, to be assigned to the users through DHCP or other ways.
- o The IP addresses of the interfaces of routers, and the routing table information (IGP or BGP). This information can help to construct the whole network topology.
- o The congestion status of the router interfaces, but this information could be reflected in the routing table change.
- o Policy information. For example, one multi-homing AS prefers to use which AS to transit its traffic, including the pricing information.

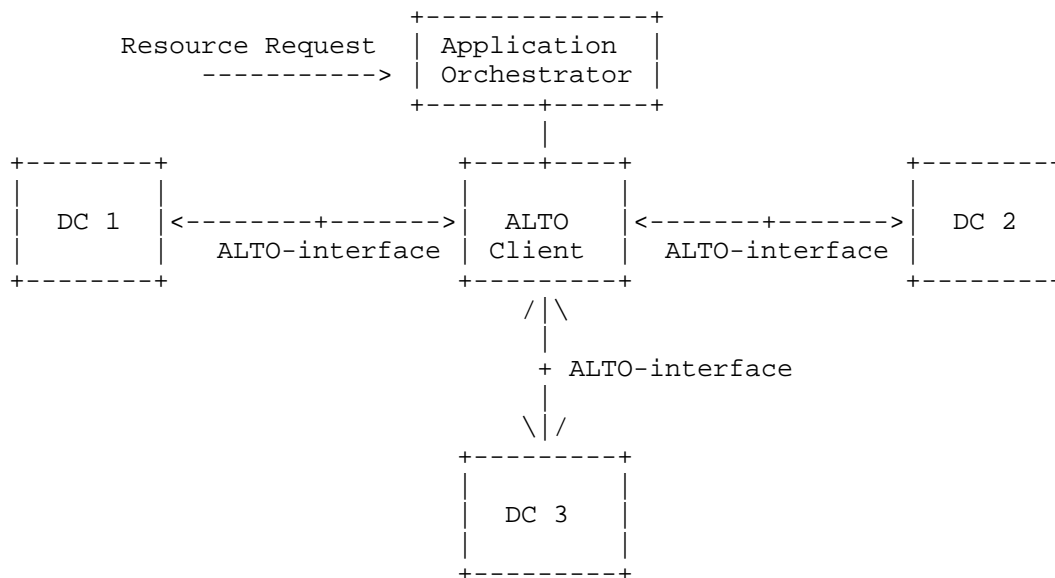
ALTO can use the collected information mentioned above to be able to select a node topologically closer, with lower transit price and less congested link.

3.2. Data Center Network

Infrastructure as a service (IaaS) is a way how the data center provides its services. There are different kinds of resources in a data center, physical machines, virtual machines, switches, firewalls, computing power, storage space, and electric power. [I-D.lee-alto-ext-dc-resource] proposes collecting data center resource information to make use of such information for a key

decision 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.

Combined computing 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.



The ALTO server needs to collect the following information so as to provide this kind of service.

- o All switches' network capacity information
- o All physical servers' information(CPU, memory) in the data center
- o The storage space information in the data center
- o The physical links information
- o The virtual machines' information(CPU, memory) and its affinity
- o Pricing models

Based on the aforementioned information, ALTO server can provide the load balancing/traffic optimization information to ALTO client for data centers, through map or ranking.

4. Open Discussion

In order to optimize the application traffic, network layer needs to provide necessary information to the according applications in the previous section, with a method (request/response, or subscription/notification) to make the underlying information changes be timely sent to the ALTO server. This requires more interactions between the network layer and the application layer. I2RS seems to consider more configuration use cases such like policy based routing, but what ALTO protocol needs is the part that discloses the network information. So there could be two ways to go forward.

(1) Defining a new protocol to cover all required functions such as configuration and information disclosure required by I2RS. This protocol covers ALTO's south bound information requirements on network topology, with its one component. But this protocol will be tremendously complicated. It will be related and overlapped with other existing protocols, such like NetConf and YANG.

(2) Defining the south bound interface for ALTO, to collect the infrastructure information, and bring the requirements discussion in I2RS, and identify the final scope, to avoid the overlap between different WGs(ALTO, NetConf, NetMod). In this case, the protocol development will belong to ALTO WG.

5. Informative References

[I-D.lee-alto-ext-dc-resource]

Lee, Y., Bernstein, G., and D. Dhody, "ALTO Extensions for Collecting Data Center Resource Information", draft-lee-alto-ext-dc-resource-01 (work in progress), January 2013.

[I-D.ietf-alto-protocol]

Alimi, R., Penno, R., and Y. Yang, "ALTO Protocol", draft-ietf-alto-protocol-13 (work in progress), September 2012.

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

Authors' Addresses

Haibin Song
Huawei

Email: haibin.song@huawei.com

Young Lee
Huawei

Email: leeyoung@huawei.com

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: July 13, 2013

H. Xie
Huawei & USTC
T. Tsou
Huawei (USA)
D. Lopez
Telefonica I+D
H. Yin
Huawei (USA)
January 9, 2013

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

Abstract

The introduction of SDN fundamentally changes the way that the application layer traffic optimization (ALTO) works. This draft describes two architectures, the Vertical Architecture and the Horizontal Architecture, allowing coherent coexistence of ALTO and 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 July 13, 2013.

Copyright Notice

Copyright (c) 2013 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 and Applications	5
2.2. Division of Network	6
2.3. SDN Domain	8
3. Architectures for Co-existing SDN and ALTO	9
3.1. ALTO Changes Due to SDN	9
3.1.1. ALTO Scopes	9
3.1.2. ALTO clients	10
3.2. The Vertical and Horizontal Architectures	11
3.3. Interactions between SDN and ALTO	13
3.3.1. Downward Interaction	13
3.3.2. Upward Interaction	14
3.4. Interactions between Legacy ALTO Clients and ALTO Servers	15
4. Information Flows	15
4.1. Information Flow of SDN Controller	15
4.2. Information Flow of Applications, SDN and ALTO	16
4.2.1. SDN-aware Applications	16
4.2.2. SDN-unaware Applications	17
4.2.3. Legacy Applications	17
4.3. Summary	18
5. Messaging	18
5.1. Service Negotiation	18
5.2. Status Report (Upward Information Flow)	18
5.3. ALTO Message Dissemination (Downward Information Flow)	19
6. Use Case for Co-existing SDN and ALTO	19
6.1. Use Case for Upward Flow	19
6.1.1. Unrestrictive Information Exporting	20
6.1.2. Restrictive Information Exporting	20
6.1.3. Information Aggregation	21
6.2. Use Case for Downward Flow	21
6.2.1. SDN-Aware QoS Metrics	22
6.2.2. Content Delivery Networks (CDN)	23
6.2.3. Information-Centric Content Delivery Networks (IC-CDN)	26
7. Conclusions	27
8. Contributors	27
9. Acknowledgements	27
10. Security Considerations	28
11. IANA Considerations	28
12. Informative References	28
Authors' Addresses	28

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 and Applications

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.

There are two types of applications in software defined networks:

- o SDN-aware applications: applications 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 SDN-unaware applications: applications indirectly communicate with SDN controllers by sending application protocol 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. Architectures for Co-existing SDN and ALTO

In this section, we first compare the ALTO scopes without and with the existence of SDN, and then describe two architectures for co-existing SDN and ALTO.

3.1. ALTO Changes Due to SDN

SDN incurs significant changes to ALTO scopes and clients. We describe the major differences below.

3.1.1. ALTO Scopes

The existence of SDN differentiates two scopes of ALTO, namely,

- o The current scope of ALTO without SDN (referred to as 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 The new scope of ALTO with coherent coexistence of SDN (referred to as the SDN-friendly Scope).

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.

3.1.2. ALTO clients

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

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

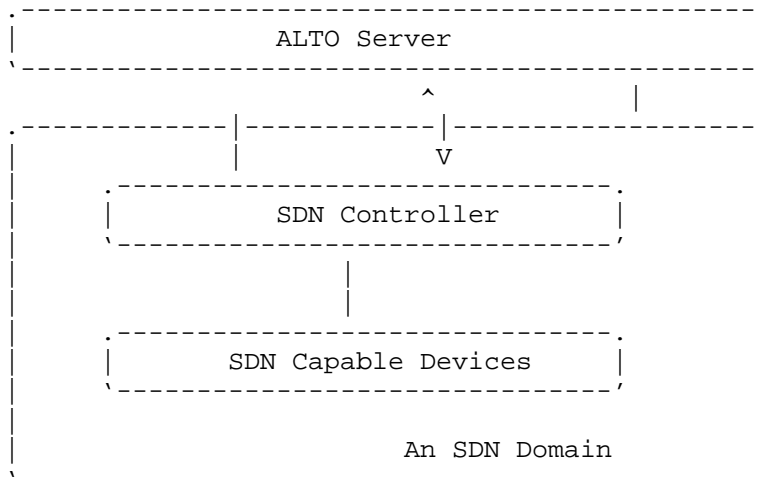
of the SDN domain is owned by these customers and so is the management. In this case, the SDN controllers as ALTO clients are slightly more like legacy ALTO clients. However, there still exist fundamental differences which we will illustrate later.

3.2. The Vertical and Horizontal Architectures

We now 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.

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.

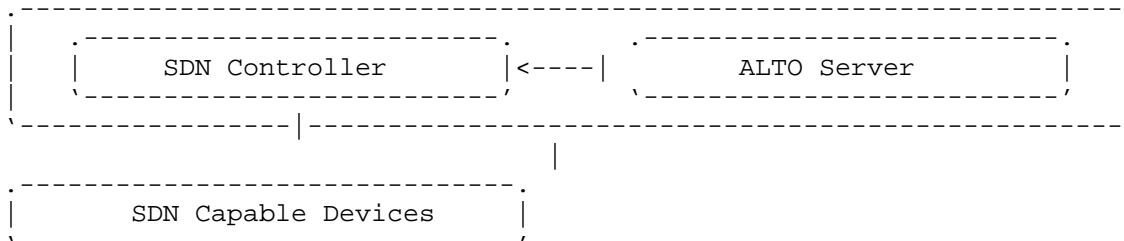


The Vertical Architecture is a hierarchical architecture consisting of three tiers. In the first tier, the only entity is the ALTO server. In the second tier, the only entities are the SDN domain controllers. In the third tier, the only entities are SDN domains (each domain consists of numerous networking 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.

- o the Horizontal Architecture (or the H Architecture for short) simplifies the implementation of ALTO extensions for SDN. 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.

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.

3.3. Interactions between SDN and ALTO

The interactions between SDN controllers (as ALTO clients) and ALTO servers are significantly different. 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.

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

3.3.1. Downward Interaction

The downward interaction is the information flow from ALTO servers to ALTO clients (i.e., 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).

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.

3.3.2. 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 inevitable requirement that SDN controllers collect the information of the networking devices in its domain. Each SDN controller may collect 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.

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.

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.

3.4. 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. 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.

4.1. 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.

4.2. 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.

4.2.1. 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.

4.2.2. 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.

4.2.3. 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.

4.3. 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.

5. Messaging

The information exchanged between the SDN domain controllers and the ALTO server is encoded and implemented by specific messaging mechanisms. Below we describe a preferred messaging mechanism where we focus mainly on the semantics of the messages.

Based on the ALTO services, there are two-way message exchanges between the SDN controller and the ALTO server. NBI is used and should be adapted to accommodate such message exchanges. The concept of SDN domain is enforced by the controller if its policy defines so; therefore, the controller can opt to export the relevant information at policy-specific granularities.

5.1. Service Negotiation

SDN Domain controllers can communicate with the ALTO server to negotiate any or all of the service information described in the next two subsections. After negotiation, such information can be pulled from or pushed to ALTO server depending further on the communication mechanism provided by NBI. Further, the detail mechanism of consuming the above information will depend on the types of ALTO services being offered and not be covered by this use case.

5.2. Status Report (Upward Information Flow)

- o network "node" information (its granularity is specified by controller's policy), mainly including network and/or geographical location, services, etc

- o network "topology" information (at a granularity specified by controller's policy), mainly including SDN-domain-level (interdomain) topology and an abstract SDN intradomain topology if any; if the policy allows, controller can also export detailed intradomain topology (the granularity should be specified by the policy).
- o network "link" information, similar to "node" and "topology", such information (e.g., link usage and state like congestion, delay, cost etc) is policed by the controller's policy and could be exported at different levels of granularity
- o network "routing" information, for flows defined in flow tables, at the policy-specified granularity
- o path information, about the path initiation and status policed by controller's policy.

5.3. ALTO Message Dissemination (Downward Information Flow)

It is important to note that the vanilla ALTO service (i.e., cost map or path cost information) is no longer directly applicable to the context of co-existing SDN and ALTO.

In vanilla ALTO service scenarios, paths (i.e., routing between any pair of routers) are deterministic a priori, regardless of ALTO recommendations. However, in the context of co-existing SDN and ALTO, routing is to be determined based on many factors including ALTO. For instance, the routing between any pair of two SDN capable routers may not be fully determined when the SDN domain controller(s) query ALTO service for recommendations.

- o network path-cost map at the granularity of SDN domains (keep in mind that the routing path may not be finalized when ALTO is consulted, as the flow table may not be propagated for the given flows).
- o selection or preferences of one or multiple paths among a set of paths at the granularity of SDN domains; selected/preferred paths can have defined priority and/or failover definitions;

6. Use Case for Co-existing SDN and ALTO

6.1. 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.

6.1.1. 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.

6.1.2. 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.

6.1.3. 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 determines 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.

6.2. 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.

6.2.1. 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.

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.

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.

6.2.2. 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)

6.2.3. 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).

7. 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.

8. 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)

9. Acknowledgements

The authors would like to thank Tom Taylor and Aditi Vira for editing the draft.

This memo is based upon work supported in part by the National Science Foundation of China (NSFC) under Grant No. 61073192 and the China 973 Program under Grant No. 2011CB302905. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF.

10. Security Considerations

TBD.

11. IANA Considerations

This document makes no specific request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

12. 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

