ALTO Working Group
INTERNET-DRAFT
Intended Status: Standard Track
Expires: December 11, 2015

L. Deng
China Mobile
H. Song
Huawei
S. Kiesel
University of Stuttgart
R. Yang
Yale
Q. Wu
Huawei

June 10, 2015

Extended End Point Properties for Application-Layer Traffic Optimization
draft-deng-alto-p2p-ext-06

Abstract

The purpose of the Application-Layer Traffic Optimization (ALTO) protocol is to provide better-than-random peer selection for P2P networks. The base ALTO protocol focuses, only on providing network topological location information (i.e., network maps and cost maps). However, the peer selection method of an endpoint may also use other properties, such as geographic location. This document defines a framework and an extended set of End Point properties (EP properties) to extend the base ALTO protocol.

Status of this Memo

Table of Contents

1. Introduction

The initial purpose for Application Layer Traffic Optimization (ALTO) protocol [RFC7285] is to provide better than random peer selection for Peer-to-Peer (P2P) networks. It is expected that ALTO can be used in serving a variety of applications and therefore it should be able to provide richer information in terms of End Point properties.

In this document, more EP property extensions are defined to provide guidance for both P2P and other applications in terms of end point selection.

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 RFC 2119 [RFC2119].

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

TBA.

3. Overview

It is expected that EP properties reflecting the following list of information can be useful for an ALTO client to provide better user experience or avoid performance degradation:

o location-related information, the information about the geographic location of the end point.

o node-related information, the information about the end point’s local features, such as software/hardware configuration and the participating role of the end point (e.g. as a end user, or a CDN server, or a P2P cache, etc.).

o network-related information, the information about the attached network of the end point, such as the type or configuration of the access network (e.g. 2G/3G/4G, WLAN, DSL, etc.) and the information about the network topology (e.g. ASN, Rack-id, etc.).

o subscription-related information, the information about the service provision agreement between the end point’s owner (i.e. the subscriber) and the network provider.

3.1. Guidelines and Methodology

The most basic principle would be to maintain the EP property set to a minimum, which in turn implies two guidelines: non-redundancy and generality.

o Non-redundancy, refers to the guideline that there is no complete coverage between any two properties.

o Generality, refers to the guideline that each property should be generally applicable to a group of settings. It is not economic to define a property which is bounded to a single type of application or a single deployment scenario.

In order to make sure that the properties as defined in this document fulfill the above principle and guidelines, we intend to justify each property's definition using the following methodology:

o Usefulness: there should be a clear motivation and application scenarios that justify the necessity and value for providing such information via EP property enquiry.

o Non-redundancy: avoid adding a property whose value can be implied by an already defined property or any combination of them. It may be of interest to keep the discussion and suggestions on how to acquire such information via from other already defined EP properties in the document.

o Case-independency: when designing the concrete information model for the properties, it is suggested to group application/deployment specific information into more general property definitions (with different value for different applications/scenarios) whenever possible.

3.2. Information flow

On the one hand, the same piece of information about a group of candidate endpoints may be acquired by an application in two ways: directly through one-to-many communication of application-specific message exchange with each candidate for flexibility, or indirectly via one-to-one transaction with the ALTO server for efficiency.

On the other hand, EP properties as defined in this document may as well be retrieved and aggregated into the ALTO server in two ways. One is from the endpoint itself, and the other is from the service provider which provides network service to the end point.

Note: There is currently no standardized mechanism by which a peer could publish information about itself into an ALTO server.

Therefore, it is to be decided whether or not if we should include EP properties in this document if their acquisition requires an extension to the base protocol for an endpoint to publish its information directly to the ALTO server.

An endpoint can discover the ALTO server with ALTO discovery mechanisms, and then setup a communication channel with its ALTO server. After that the endpoint property from the endpoint itself can be reported.

The ALTO server can also be configured to access the Network Management System server or other similar servers provided by the network service provider for information about end points, such as subscription related information.

3.3. Privacy considerations

Privacy considerations is a general concern for almost all EP properties, as they are by definition more stationary information regarding a specific end point.

However, each end point may have different concerns or sensitive preference over a specific EP property. For example, endpoint property regarding the service role of the endpoint, serving nodes deployed by the ISP or third party service provider, such like P2P caching server, or CDN node, may have different considerations over whether a piece of information is private or not. Therefore, it may be necessary to provide a mechanism to accommodate this type of individual customization by providing a channel for an end point to explicitly indicate this information based on its own preference.

More general, it is expected that the privacy level of a specific EP property is dependent on the nature of the information (i.e. the EP property), the type of the subscriber (i.e. the user who owns the end point in question), the type of the application (i.e. the ALTO client who is requesting the EP property) and the policy of the ISP (i.e. the owner of the ALTO server who is able to do information collection from the end points and determine how the the information is exposed to the requesting application).

Fortunately, there are generally applicable schemes to be used to address the privacy protection concerns, which may be applicable to a group of EP properties and can be configured by the ISP or the EP subscriber. In this section, several general schemes are introduced, whose application to each EP property is elaborated later in following sections.

3.3.1. Privacy-Preserving Information Mapping

On the one hand, the privacy concern is unnecessary if the specific endpoint property can also be measured/disclosed in another way. The privacy concern regarding to the accurate information of the endpoint would be alleviated if using relative numbers to rank them. For deployment considerations, it is also possible for each endpoint to make the choice whether to disclose the relative information or not, but an incentive could be used to encourage the disclosure when it is beneficial to the application.

In other words, in order to preserve the privacy of a piece of information, different data types can be defined via information mapping. In particular, in this document, each property is defined as a JSON object [RFC4627], which contains a dynamic typing attribute "content" as well as two deterministic attributes, "name" and "precision".

The "name" attribute is a string, whose value is the name of the property. The "precision" attribute is also a string, whose value comes from an attribute-dependent set. Depending on the value of the property’s "precision" attribute, its "content" attribute can be a string, number, boolean or another object.

In this document, in order to define an EP property as a JSON object, we specify:

- o the string value of its "name" attribute;
- o the value set of its "precision" attribute; and
- o the definitions of its "content" attribute for each "precision".

A special string value "" for "precision" attribute is used to indicate that an EP property, which is not privacy sensitive or using information mapping, has no precision-dynamic "content" definition.

3.2.2. Access Control

On the other hand, access control to sensitive property information may also be used to mitigate the privacy concern of a defined property. Even greater flexibility can be delivered by access control at the discretion of both the network operator and the individual subscriber, which is deployment specific and out of scope for the general discussion within this document.

3.4. Relation with other properties

Endpoint information can be extremely dynamic or relatively static. Currently, this specification does not intend to provide any real-

time properties such as the available bandwidth from the endpoint [I-D.draft-wu-alto-te-metrics], whose value is subject to frequent changes and hence requires a measurement-based exposure scheme.

The basic end point properties as defined in this document, serves as a basis for the property namespace to be used to derive PID properties [I-D.draft-roome-alto-pid-properties] for the corresponding peer group, when the direct enquiry for the information per end point is not efficient or economic for the ALTO client.

4. Endpoint Extensions

This document defines new endpoint property types for the ALTO protocol [RFC 7285].

4.1. Location-Related Properties

4.1.1. Endpoint Property Type: geolocation

It is believed that the information about an individual endpoint's geo-location is of value to a variety of applications. However, it is also well accepted that geolocation of an endpoint is likely to be considered as a private piece of information to the subscriber, and therefore should be protected against undesirable privacy intrusion.

Moreover, in a data-center, the relative location of a serving node may be of interest to an ALTO client, where much finer-grained information (e.g. the hosting physical server or rack number) are relevant and can be dynamically updated by either a live migration of a serving node contained in a virtualization container or a traffic handover between active and standby instances during an HA/LB switch-off.

To this end, an EP property is defined as a JSON object, with the name "geolocation", whose "content" definition is actually dependent on the "precision" attribute, which in turn is a JSON string whose value belongs to the following JSON array:

geolocation_precision_set = ["countrycode", "boundingbox", "circle", "dc"]

If the "precision" attribute of the "geolocation" property of an endpoint is "countrycode", the following "content" attribute is defined as the ISO 3166 two-letter country codes of the region the endpoint resides in, as a JSON string.

If the "precision" attribute of the "geolocation" property of an endpoint is "boundingbox", the following "content" attribute is defined as a four-element JSON object "bounding_box":

```
bounding_box = {
             "latul" : number,
             "longul" : number,
             "latbr": number,
             "longbr" : number
}
```

If the "precision" attribute of the "geolocation" property of an endpoint is "circle", the following "content" attribute is defined as a three-element JSON object "circle_location":

```
circle_location = {
             "latc" : number,
             "longc" : number,
             "radius": number
}
```

If the "precision" attribute of the "geolocation" property of an endpoint is "dc-location", the following "content" attribute is defined as a four-element JSON object "dc-location":

```
   dc-location = {                "rack-id" : number,
"server-id" : number    }
```

4.2. Node-related properties

4.2.1. End Point Property Type: participating_role

Different types of end points have different roles or participating policies for a given application, which can be explored in making a better decision when choosing a serving node. For example, as described in [I-D.draft-deng-alto-p2pcache], P2P caching node can also act as p2p peers in a p2p network. If a p2p caching peer is located near the edge of the network, it will reduce the backbone traffic, as well as the uploading traffic. [RFC7069] provides one example of such caching nodes. P2P caching peers are usually expected to be given higher priority than the ordinary peers for serving a content request so as to optimize the network traffic. So it’s necessary for the End Point property to support this indication.

In general, the end points which belong to different participating

parties (subscriber, ISP, or ICP) within an application's service transaction demonstrate different role/policies.

It is straightforward for an ISP to acquire the information of an end point's participation role from its local record for its subscribers, its local or third party infrastructure for a given application.

To this end, an EP property is defined as a JSON object, with the name "participating_role", whose "precision" attribute is set to "" and its "content" attribute is defined as a JSON string, whose value belongs to the following array:

```
participating_role_set=["user", "cache", "super_node"]
```

In other words, the "participating_role" property is defined as follows:

```
participating_role : {
     "precision": "",
     "content": ["user", "cache", "super_node"]
}
```

4.2.2. End Point Property Type: battery_limited

Another important End Point property that will impact peer selection is what kind of power supply the peer has. It can be either the electric power or the battery supply.

And for most of the time, it is safe to bet that electric power supplied nodes would stay online longer than those battery supplied nodes, while battery powered devices are usually less willing to act as super peer, relay, etc.

And most of the nowadays intelligent equipments are aware of their power supply type. But it is necessary that the power supply of a peer can be queried through some method no matter whether or not it is limited by its battery.

To this end, an EP property is defined as a JSON object, with the name "battery_limited", whose "precision" attribute is set to "" and its "content" attribute is defined as a boolean, is either "true" or "false".

If the peer in question is actually battery-limited, the value of this property with respect to the peer is set to "true".

In other words, the "content" attribute of the "battery_limited" property is defined as a JSON boolean, "true" for a battery supplied

end point, or "false" for an electricity supplied end point or for an end point with an unknown power supply type.

```
"battery_limited": {
     "precision": "",
     "content": true/false
}
```

4.2.3. End Point Property Type: local_capacity

For resource-consuming applications, it would be helpful to know the local capacity (e.g., in terms of computing, storage, and networking) of an end point before it is selected.

In other words, the "local_capacity" property is defined as a JSON object, as follows:

```
"local_capacity": {
     "precision": "",
     "content":{
             "CPU":          {
                     "volume": integer,
                     "meter": string
             },
             "memory":{
                     "volume": integer,
                     "meter": string
             },
             "storage": {
                     "volume": integer,
                     "meter": string
             }
     }
}
```

4.3. Network-Related Properties

4.3.1. End Point Property Type: network_access

One important End Point property that will impact peer selection is the type of the node’s access network.

Note: There is remaining doubt on whether or not this property is needed, since at least part of the information it reflects, for instance, the end point’s provisioned bandwidth, is defined and exposed by other properties.

For instance, a mobile subscriber’s access network can be cellular (2G, 3G, or 4G). Take another example of a node owned by a home subscriber, the type of its access network can be DSL, FTTB, or FTTH.

Different type of access network gives a clear indication on both the amount and the technology of the provisioned resources (e.g. the shared/guaranteed bandwidth, the interval for physical channel scheduling, etc.)

Moreover, one may prefer to specify a special access type for a node deployed in a data center too, because it is likely to be more robust, and have more network resources than either mobile or home users.

Hence application may have its own algorithm for peer selection or traffic rendering if the node access type information can be provided via an End Point property. The value for this property can be enumerated as "adsl", "ftth", "fttb", "dc", and etc.

In case that the end point has its own privacy concerns in revealing its access network type directly to potentially distrusted applications through ALTO, another indirect way of exposing the similar information can be used by "access_preference" as per ISP’s judgement.

In essence, an ISP assigned "access_preference" property for the end points gives the network operator a chance to say which end point’s link is "better" without having to tell what the actual criterion is.

The value for this property (defined as integer) can be set by the ISP of the ALTO server, based on its own relative preference to different network access types. A peer with the higher value is more preferable than another peer with the lower value.

For example, an ISP could use the following setting for now:

1 = DSL; 10 = FTTB; 12 = FTTH; 50 = DC;

and add "100=new_technology", when some new technology better than FTTH appears later.

To this end, an EP property is defined as a JSON object with the name "network_access", with two different values for "precision"

network_precision_set=["technology", "rank"]

In other words, the "content" of the "network_access" property is dependent on the value of its "precision" attribute.

If the value of "precision" is "technology", the following "content" attribute is defined as a JSON string, whose value belongs to the following array:

```
network_access_set = ["adsl", "ftth", "fttb", "dc", "cellular"]
```

If the value of "precision" is "rank", the following "content" attribute is defined as a JSON number, whose value indicates the relative preference over the end point in question, in terms of its access network. The end point with a higher number is more preferable to another end point with a lower number.

In summary, the "network_access" property is defined as a JSON object, as follows:

```
"network_access": {
     "precision": "technology",
     "content":["adsl", "ftth", "fttb", "dc", "cellular"]
}

"network_access": {
     "precision": "ranking",
     "content": number
}
```

Note: There is concern about undesirable privacy leakage via network_access properties to distrusted ALTO clients. In such cases, according to the definitions above, either the endpoint itself or the ISP who is running the ALTO server can either specify an access control policy to prevent undesirable exposure to specific ALTO clients or use a privacy preserving mapping from the raw description of access technologies to a number of abstract relative ranking information instead. Moreover, the endpoint or the ISP might choose to use another subscription related property "provisioned_bandwidth" (defined later in Section 4.4.2) instead of "network_access".

4.3.2. End Point Property Type: forwarding_class

As suggested for the NFV use-case, the End Point property "forwarding_class" is meant to indicate the type of forwarding class the end point or network supports.

Forwarding classes can be thought of as output queues. For a

classifier to assign an output queue to a packet, it must associate the packet with one of the following forwarding classes:

- o Expedited forwarding (EF), provides a low-loss, low-latency, low- jitter, assured bandwidth, end-to-end service.

- o Assured forwarding (AF), provides a group of values you can define and includes four subclasses: AF1, AF2, AF3, and AF4, each with three drop probabilities: low, medium, and high.

- o Best effort (BE), provides no service profile. For the best effort forwarding class, loss priority is typically not carried in a class-of-service (CoS) value.

- o Network control (NC), is typically high priority because it supports protocol control.

Hence, the "content" of the "forwarding_class" property is defined as a JSON string, whose value belongs to the following array:

forwarding_class_set = ["expedited", "assured", "network control", "best effort"]

In summary, the "forwarding_class" property is defined as a JSON object, as follows:

```
"forwarding_class": {
     "precision": "",
     "content": ["expedited", "assured", "network control", "best effort"]
}
```

4.4. Subscription-Related Properties

4.4.1. End Point Property Type: volume_limited

Many wireless operators offer low-cost plans, which limit the amount of data to be transmitted within a month to some gigabytes. After that they will throttle the subscriber’s bandwidth or charge extra money. Hosts with such a tariff, could be tagged by another End Point property "volume_limited" and should be avoided for peer selection to serve other peers.

The "content" value for this property (defined as a boolean) is either "true" or "false". If a peer is constrained by such a subscription plan, the value of this property with respect to the peer is set to "true".

In other words, the "volume_limited" property is defined as a JSON object with a boolean "content", "true"for an end point with such a limited data plan, or "false" for an point with unlimited or unknown data plan.

```
"volume_limited": {
     "precision": "",
     "content": true/false
}
```

4.4.2. End Point Property Type: provisioned_bandwidth

For applications seeking for a candidate peer for uploading services, the end point’s uploading bandwidth is essential for the selection.

While it is straightforward for one to expose the accurate information over an end point’s bandwidth capability, the subscriber of the end point might consider it a piece of private information.

On the other hand, it is suggested that the ISP can also choose to expose its relative preference in terms of the end point’s provisioned bandwidth; this ensures better load balancing within the network by avoiding undesirable hot spots caused by competition from applications for the handful most provisioned end points.

Therefore, the "provisioned_bandwidth" property is defined as a JSON object, whose "content" definition is actually dependent on the "precision" attribute, which in turn is a JSON string whose values belong to the following JSON array:

```
provisioned_bandwidth_precision_set = ["raw", "ranking"]
```

If the "precision" attribute of the "provisioned_bandwidth" property of an end point is "raw", the following "content" is filled with the accurate value of the provisioned bandwidth, as a JSON object "provisioned_bandwidth_value" with two elements:

```
provisioned_bandwidth_value = {
             "value" : number,
             "metric" : ["GB", "MB", "KB", "Gb", "Mb", "Kb"]
}
```

If the "precision" attributed of the "provisioned_bandwidth" property of an end point is "ranking", the following "content" is filled with the relative ranking of the end point’s provisioned bandwidth assigned by the ISP, which in turn is a JSON number where higher number indicating more preference.

In summary, the "provisioned_bandwidth" property is defined as a JSON object as follows:

```
"provisioned_bandwidth": {
     "precision": "raw",
     "content": {
             "value": number,
             "metric": ["GB", "MB", "KB", "Gb", "Mb", "Kb"]
     }
}

"provisioned_bandwidth": {
     "precision": "ranking",
     "content": number,
}
```

5. Security Considerations

TBA.

6. IANA Considerations

This document adds the following new End Point property types to the existing registry created by ALTO protocol [RFC7285].

TBA.

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.

[RFC7285] Alimi, R., Penno, R., and Y. Yang, "ALTO Protocol", RFC7285, March 2014.

7.2. Informative References

[I-D.draft-deng-alto-p2pcache] Deng, L., Chen, W., and Q. Yi, "Considerations for ALTO with network-deployed P2P caches", draft-deng-alto-p2pcache-03 (work in progress), February 2014.

[RFC7069] Alimi, R., Rahman, A., Kutscher, D., Yang, Y., Song, H., and K. Pentikousis, "DECoupled Application Data Enroute (DECADE)", RFC 7069, November 2013.

[I-D.draft-roome-alto-pid-properties] Roome, W. and Yang, R., "PID Property Extension for ALTO Protocol", draft-roome-alto-pid-properties-01 (work in progress), February 2014.

[I-D.draft-wu-alto-te-metrics] Wu, Q., Yang, R., Lee, Y., and Randriamasy, S., "ALTO Traffic Engineering Cost Metrics", draft-wu-alto-te-metrics-03 (work-in-progress), June 2014.

Acknowledgements

The authors would like to thank, Michael Scarf, Vijay Gurbani, Reinaldo Penno, Sabine Randriamsy and Qiao Fu for their review and valuable comments.

Authors' Addresses

Lingli Deng
China Mobile
China

Email: denglingli@chinamobile.com

Haibin Song
Huawei
China

Email: haibin.song@huawei.com

Sebastian Kiesel
University of Stuttgart, Computing Center
Germany

Email: ietf-alto@skiesel.de

Richard Yang
Y. Richard Yang
Yale University

Email: yry@cs.yale.edu

Qin Wu
Huawei
China

Email: sunseawq@huawei.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 7, 2016

Z. Dulinski
Jagiellonian University
P. Wydrych
R. Stankiewicz
AGH University

July 6, 2015

Inter-ALTO Communication Problem Statement
draft-dulinski-alto-inter-problem-statement-02

Abstract

The Application-Layer Traffic Optimization (ALTO) protocol has been standardized in RFC7285 to ease better-than-random overlay connection management. Due to various reasons, optimization capabilities of ALTO servers are limited by the fact that it may not be possible for an ALTO server to compute costs for source/destination pairs correctly if a source and/or a destination is outside the administrative domain it belongs to. In other words, an ALTO server may be generally unable to optimize the traffic with only locally available topology information.

Therefore, it is proposed that operators of distinct administrative domains may agree on topology- and policy-related information exchange through a standardized inter-ALTO communication framework. This document provides a rationale for design, standardization, and implementation of the inter-ALTO communication framework.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

As stated in [RFC5693], information on network topologies and routing policies in today Internet is not generally available to the application layer. At the same time, a lot of new overlay applications creating their own topologies on top of the physical one emerge. As both network operators and users of the overlay applications may benefit from better-than-random overlay connection management, the ALTO protocol has been standardized in [RFC7285].

Topology- and policy-related information may be supplied through ALTO in a proactive or in a reactive way. In the former case, an ALTO server distributes network and cost maps through which a client may compute costs of sending the traffic, while in the latter case, a client may request a server to provide rating of explicitly specified source and destination pairs using the endpoint cost service. As a server provides information composed only from information that can

be gathered by it and applies polices defined by operator of the administrative domain it belongs to, the returned information has always local meaning to the server - regardless of the method used by a client to request guidance from it. Moreover, an ALTO client may not be able to receive ALTO information from outside of its administrative domain.

Due to various reasons, it may be generally not possible to optimize the traffic with only locally available topology information. Therefore, in this document, it is proposed that operators of distinct administrative domains may agree on topology- and policy-related information exchange through a standardized inter-ALTO communication framework.

Among others, the inter-ALTO communication framework may allow an operator of one administrative domain to apply its policies to topology information gathered from other administrative domains. Moreover, a server may have separate security contexts for processes responsible for server-to-client (i.e., ALTO) and server-to-server (i.e., inter-ALTO) communication to ensure that no sensitive topology- and policy-related information is distributed in an uncontrolled way.

The requirements for the inter-ALTO communication framework, its detailed specification and issues related to server discovery [RFC7286] are out of scope of this document.

2. Definitions

This document uses the following terms defined in [RFC5693] and [RFC7285]: ALTO Server, ALTO Client, Endpoint, Peering Traffic, Transit Traffic. Moreover, the following terms have the special meaning in the definition of the inter-ALTO communication problem.

Local Administrative Domain: The administrative domain which the ALTO client belongs to.

Remote Administrative Domain: An administrative domain other than the one which the ALTO client belongs to.

Local ALTO Server: An ALTO server belonging to the local administrative domain.

Remote ALTO Server: An ALTO server belonging to a remote administrative domain.

Local Endpoint: An endpoint belonging to the local administrative domain.

Remote Endpoint: An endpoint belonging to a remote administrative domain.

3. Motivation for Inter-ALTO Communication

Optimization capabilities of ALTO servers are limited by the fact that they use information available locally only. It can be shown that without additional information on remote endpoints, routing paths, or remote administrative domains' preferences, rating provided by an ALTO server may be sub-optimal for both sides. Data from remote administrative domains obtained from an ALTO server holding authoritative information about those domains' topologies may have a substantial significance for the traffic management.

The ALTO Protocol specification [RFC7285] states that:

> 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.

This document provides a rationale for design, standardization, and implementation of the inter-ALTO communication framework, described as "External Interface" in Figure 1 of [RFC7285]. The requirements and detailed specification are out of scope of this document.

```
   ,--------------------------.            ,--------------------------.
   |                          |            |                          |.
   | ,-------------.          |            |         ,-------------.  ||
   | | Local       |.         |            |         | Remote      |. ||
   | | Information ||         |            |         | Information || ||
   | | Sources     ||         |            |         | Sources     || ||
   | '-------------'|         |            |         '-------------'| ||
   |  '-------------'         |            |          '-------------' ||
   |          \               |            |               /          ||
   |           \              |            |              /           ||
   |            ,--------.    |            |    ,--------.            ||
   |            | Local  |    | inter-ALTO |    | Remote |            ||
   |            | ALTO   |<-------------------->| ALTO   |            ||
   |            | Server |    |            |    | Server |            ||
   |            '--------'    |            |    '--------'            ||
   |           /              |            |              \           ||
   |          /               |            |               \          ||
   | ,---------.              |            |             ,---------.  ||
   | | Local   |.             |            |             | Remote  |. ||
   | | ALTO    ||             |            |             | ALTO    || ||
   | | Clients ||             |            |             | Clients || ||
   | '---------'|             |            |             '---------'| ||
   |  '---------'             |            |              '---------' ||
   |                          |            |                          ||
   '--------------------------'            '--------------------------'|
                                            '--------------------------'
   Local Administrative Domain             Remote Administrative Domains
```

Figure 1: Inter-ALTO communication architecture.

The architecture of the inter-ALTO communication framework is shown in Figure 1. Both ALTO servers gather the information from their information sources like routing protocols, provisioning policies, or dynamic network information sources using respective provisioning protocols. To provide (using the ALTO Protocol) better guidance for its clients, the local ALTO server needs to communicate with remote ALTO servers to obtain information which is available only at the entities in the remote administrative domains.

In particular, two general use-cases of inter-ALTO communication may be distinguished:

o providing the local ALTO server with information on remote view of the multi-domain topology;

- o providing the local ALTO server with information on remote network topology, more accurate than available from the local information sources.

Moreover, for partitioned networks run by one organization or a consortium, both use-cases are valid.

3.1. Remote View of Topology

Due to the complex topology of the current Internet and independent application of routing policies in autonomous systems, the communication between two endpoints does not need to follow the same path in the opposite directions. Moreover, there is a significant disproportion between availability of information on upstream and downstream paths. While the current local information sources can easily export data on the former, the latter are generally unknown to ISPs. Depending on the deployment-specific use-case and cost metric applied, this fact may indeed affect the ALTO server’s ability to calculate the costs.

For a set of reasons (e.g., application performance or quality perceived by end-users) an ALTO server may suggest its clients to connect to endpoints located in their proximity. One of the simplest measures of proximity is the number of AS hops, as announced by BGP. As indicated above, due to the route asymmetry, the number of AS hops between two communicating endpoints may significantly differ between the upstream and downstream paths.

Under other circumstances, an ISP may prefer to reduce the transit traffic by increasing the peering one and it may configure its ALTO server to differentiate endpoints into classes taking into account through which links the traffic is exchanged and what are its business relationships with its neighbors. Still, as the Internet routes are asymmetrical, a reply for request sent to an endpoint through a peering link may return via a transit one (or vice versa).

Therefore, an ALTO server may easily calculate costs of sending the traffic from the local administrative domain to remote ones, while calculation of correct costs of receiving the content from the remote administrative domains may be not possible at all. To mitigate this situation, the inter-ALTO communication framework may be used to exchange information on downstream paths between two interested parties. If a local ALTO server would be able to gather such information, a risk of suboptimal endpoint rating may be greatly reduced.

3.2. Detailed Information on Remote Topology

The second use-case stems from the fact that a lot of topology information is lost when a prefix is being announced via BGP by one AS to others. Moreover, prefix aggregation often used to reduce the size of routing tables causes that a number of networks of different characteristics are announced as one network. Therefore, a local ALTO server may consider - regardless of the cost metric used - two remote endpoints as equal, while they should be in fact differentiated.

For instance, a remote administrative domain may comprise (among others) of a set of networks connected to its core by expensive links or containing endpoints of worse capabilities than those in the rest of its network. Then, inter-ALTO communication may be used to denote such a fact and to characterize endpoints properly. Similarly, when some remote endpoints stand out above the rest, they may be promoted by the local ALTO server.

A cost metric may also take into account congestions on intra- and inter-domain links or another exhaustive consumption of some resources. When a link becomes congested for a longer period of time, it may be desirable to promote endpoints reachable through lightly loaded links. Likewise, a set of endpoints providing a content or a service may be overloaded and clients should be discouraged from using them. Regardless of the reason for endpoint differentiation, a local ALTO server may be informed by a remote one about remote domain’s preference in endpoint selection.

3.3. Partitioned Networks

Means for exchange of detailed information on view of network topology may be also important for partitioned networks, run by one organization or a consortium of organizations fully trusting each other. To optimize the traffic flowing within partitions and between them, ALTO servers located in each partition may exchange detailed network topology information. In principle, ALTO servers may be deployed hierarchically or in a mesh. When a hierarchical architecture is used, a central ALTO server may gather a view of topology information from the rest of ALTO servers, merge the information, calculate the costs for all source/destination pairs, and distribute the merged network and cost maps to the ALTO servers and/or serve ALTO clients from all partitions. ALTO servers may be as well set up in each partition independently, connected to each other, gathering the network topology information from other partitions, and serving their own clients.

In both deployment schemes, both remote view of the inter-partition topology (see Section 3.1) and detailed view of remote partition topology (see Section 3.2) may bring a lot of benefit to the organization/consortium. The former can be used to optimize the traffic flowing between the partitions, while the latter may allow an ALTO server to differentiate endpoints within one partition.

4. IANA Considerations

This document does not define any new media types nor does it introduce any new IANA considerations.

5. Security Considerations

In general, communication between ALTO servers run by distinct parties and exchange of information on their topologies may require a formal agreement between them, mutual authentication, and authorization. Since sensitive data may be exchanged, a secure deployment of inter-ALTO communication framework may require setting up encrypted tunnels or using SSL between the ALTO servers.

The inter-ALTO communication may allow ALTO servers to exchange any parameters which allow them to manage traffic in an optimal way for mutual benefit. In order to achieve these results a set of administrative domains may exchange sensitive data which should be kept confidential. They should be used to calculate the cost maps, but should not be revealed directly to a third party, e.g., an ALTO client. To implement such a differentiation, an ALTO server may need to use separate security contexts for client-to-server and server-to-server communication.

Moreover, to ease secure environment deployment, administrative domains may form ALTO server communities, i.e., groups of ALTO servers trusting each other and working for common benefit.

6. References

6.1. Normative References

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

[RFC7285] Alimi, R., Penno, R., Yang, Y., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, September 2014.

6.2. Informative References

[RFC7286] Kiesel, S., Stiemerling, M., Schwan, N., Scharf, M., and H. Song, "Application-Layer Traffic Optimization (ALTO) Server Discovery", RFC 7286, November 2014.

Appendix A. Acknowledgments

P. Wydrych was supported by the grant no. 15.11.230.190.

Z. Dulinski and R. Stankiewicz were supported by the SmartenIT project, funded by the EU FP7 Program under contract no. FP7-2012-ICT-317846.

The authors would like to thank Y. Richard Yang (Tongji/Yale University), Wendy Roome (Alcatel-Lucent/Bell Labs), and Chen Guohai (Huawei) for their valuable comments.

Authors' Addresses

Zbigniew Dulinski
Jagiellonian University
ul. Lojasiewicza 11
Krakow 30-348
Poland

Email: dulinski@th.if.uj.edu.pl

Piotr Wydrych
AGH University of Science and Technology
al. Mickiewicza 30
Krakow 30-059
Poland

Phone: +48 12 617 48 17
Email: piotr.wydrych@agh.edu.pl
URI: http://wydrych.net/

Rafal Stankiewicz
AGH University of Science and Technology
al. Mickiewicza 30
Krakow 30-059
Poland

Email: rstankie@agh.edu.pl

ALTO WG
Internet-Draft
Intended status: Standards Track
Expires: January 7, 2016

K. Gao
Tsinghua University
X. Wang
Tongji University
Y. Yang
Yale University
G. Chen
Huawei

July 6, 2015

ALTO Extension: A Routing State Abstraction Service Using Declarative Equivalence

draft-gao-alto-routing-state-abstraction-00.txt

Abstract

The Application-Layer Traffic Optimization (ALTO) protocol has defined multiple services (e.g., network maps, cost maps, filtered maps, the endpoint cost service, and the endpoint property service) to provide network state information to network applications. In a higher-level view, both the cost maps and the endpoint cost service can be considered as providing views into the routing state of a network (i.e., the path properties). A drawback of these existing services, however, is that they are static, application-oblivious views, without guidance from network applications. This document designs a new ALTO service named Routing State Abstraction using Declarative Equivalence (RSADE). Allowing applications to provide declarative guidance on the intended use of the network routing state, RSADE allows a network to compute compact, customized routing state abstraction beyond the existing services.

Requirements Language

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The key services of the ALTO protocol [RFC7285] can be considered as information query services about the routing state of a network. Specifically, a cost map of an ALTO metric allows a network application to look up the end-to-end value of the given metric, for the routing path(s) from a given source to a given destination. The endpoint cost service provides a similar service.

The recent advance of newer network architectures such as SDN, however, reveals that the existing services may have limitations. First, the existing services distinguish routing state at the host

level. This is reasonable in a traditional network such as a network using destination IP based routing. The emergence of new techniques such as SDN using OpenFlow may convert more networks to use more fine-grained routing, such as the 5-tuple (source and destination IPs, source and destination ports, and protocol) routing. In such a setting, revealing routing state (e.g., cost) at the granularity of endhosts may be too coarse. For example, for a network where port 80 HTTP traffic is routed differently from port 22 traffic, the existing services cannot provide the differentiation.

Second, the existing (routing state query) ALTO services are designed for relatively simple network applications. More complex network applications, such as the multi-flow scheduling application [I-D.yang-alto-path-vector], may need more complex routing state information for better application-level coordination. Let f be the network application (or network component) and let view() be the function that constructs an abstract routing state view for f. One can see that view() may compute an on-demand, instead of static, view that will depend on f. The existing ALTO services do not provide this customization capability.

A possibility to address the customization problem is that the network provides raw, complete routing state view. However, providing abstract views on top of raw network state, as ALTO does, can provide substantial benefits to both the network, which manages the network state, and the network applications, which consume the network state. First, a more compact abstract network state view can reduce the requirement on client scaling. The raw network state of a large network may consist of a large number of network devices. A consumer of such a large amount of information must be scalable. Second, an abstract network state view can better protect the privacy of the provider of the network. Third, an abstract network state view may substantially reduce the load of information updates.

The objective of this document is to design an ALTO extension service named Routing State Abstraction using Declarative Equivalence (RSADE) to address the preceding two issues. Specifically, RSADE provides a simple, declarative API for a network application to specify its need (i.e., requirements) of routing and topology state, and the network computes a minimal, but equivalent routing state to the network application. For simplicity, this document focuses on extending the endpoint cost service, leaving the aggregation aspects of using network aggregation maps as future work.

The organization of this document is organized as follows. Section 2 replicates the multi-flow scheduling example from [I-D.yang-alto-path-vector]. Section 3 gives an overview of the service, and Section 4 gives more details on specifying state

equivalence. Sections 5 and 6 discuss security and IANA considerations.

2. The Multi-flow Scheduling Use Case

A foundation of the ALTO services is the routing cost value (for a given metric) for each pair of source and destination. Although simple, this foundation may not convey enough information to some applications. This document uses a simple use case in [I-D.yang-alto-path-vector] to illustrate the issue. See [I-D.lee-alto-app-net-info-exchange] for earlier, more comprehensive discussions.

Consider a network as shown in Figure 1. The network has 7 switches (sw1 to sw7) forming a dumb-bell topology. Switches sw1/sw3 provide access on one side, s2/s4 provide access on the other side, and sw5-sw7 form the backbone. Endhosts eh1 to eh4 are connected to access switches sw1 to sw4 respectively. Assume that the bandwidth of each link is 100 Mbps. Assume that the network is abstracted with 4 PIDs, with each representing the hosts at one access switch.

```
                              +------+
                              |      |
                            --+ sw6  +--
                          /   |      |  \
  PID1 +-----+           /    +------+   \          +-----+  PID2
  eh1__|     |_         /                  \     ____|     |__eh2
       | sw1 | \   +--+---+         +---+--+ /    | sw2 |
       +-----+  \  |      |         |      |/     +-----+
                 \_| sw5  +---------+ sw7  |
  PID3 +-----+   / |      |         |      |\     +-----+  PID4
  eh3__|     |__/  +------+         +------+ \____|     |__eh4
       | sw3 |                                   | sw4 |
       +-----+                                   +-----+
```

Figure 1: Raw Network Topology.

Consider an application overlay (e.g., a large data transfer system) which needs to schedule the traffic among a set of endhost source-destination pairs, say eh1 -> eh2, and eh3 -> eh4. The application can request a cost map (or endpoint cost service) providing end-to-end available bandwidth, using 'available bw' as cost-metric and 'numerical' as cost-mode, where the 'available bw' between two endhosts represents their available bandwidth, if no other applications use shared resources.

Assume that the application receives from the cost map that both eh1 -> eh2 and eh3 -> eh44 have bandwidth 100 Mbps. It cannot determine that if it schedules the two flows together, whether it will obtain a total of 100 Mbps or 200 Mbps. This depends on whether the routing of the two flows shares a bottleneck in the underlying topology:

- o Case 1: If the two flows use different paths in the current routing state, for example, when the first uses sw1 -> sw5 -> sw7 -> sw2, and the second uses sw3 -> sw5 -> sw6 -> sw7 -> sw4. Then the application will obtain 200 Mbps.

- o Case 2: If the two flows share a bottleneck in the current routing state, for example, when both use the direct link sw5 -> sw7, then the application will obtain only 100 Mbps.

To allow applications to distinguish the two aforementioned cases, the network needs to provide more details on the routing state. A naive solution to this problem, then, is to return the two complete, detailed routes and the available bandwidth of each link on the routes. But this may not be desirable, as the application may not need the details and/or may not have the permission to see networks details.

Now consider what route abstraction can achieve. Assuming case 2 (shared bottleneck), it is sufficient for the network to return a single abstract link for each flow: ane1(100Mbps), where ane stands for abstract network element, and the number in the number 100Mbps denotes its capacity.

Consider a variation of the preceding case. Assume that the capacity of the link from sw1 to sw5 is 70 Mbps, while the rest are still at 100 Mbps. Then the abstract route from eh2 to eh4 becomes ane1(100Mbps) and ane2(70Mbps).

3. The RSADE Service

The more the network knows about what a network application f needs regarding a routing state query, the more concise the network response can be. Hence, an extreme API is that the complete network application f (i.e., the code and related state) is sent to the network. This, however, can create substantial complexity in the routing-state query component, as even some simple program properties (e.g., halting) are already difficult to analyze. Also, in settings such as interdomain, the owner of the function f may not want to provide the complete f to the network.

Another extreme API is that each routing state query provides only the most basic information (i.e., the source and the destination).

This, however, does not provide enough information for the routing-state service to compute efficient route abstraction/compression. Hence, the returned routes will be independent of individual functions, missing out opportunities on abstraction or compression.

The RSADE service tries to strike a balance between the two extremes. Figure 1 gives the grammar to specify the query information that a network application sends to the network:

```
rs-query := flow-list equiv-cond
flow-list := flow [flow-list]
flow := generic-match-condition
```

Specifically, the first component of a RSADE query is a list of flows (flow-list). Each flow in the list is specified by a match condition, as in OpenFlow.

The second component of the query input is the declared equivalence condition. A particular type of equivalence condition, in the context of routing-state query, is the equal range condition. We give the detailed specification of the condition in Section 4.

After receiving an RSADE request, the network retrieves the route for each flow, and then computes the result after compression (abstraction). RSADE may allow a network application to specify an indicator, on whether it wants to receive incremental updates to the query results, achieving push notification. The push notification is implemented using HTTP SSE [Roome-SSE].

4. The RSADE Equivalence Condition

Let attr (e.g., delay) be a vector for a given link attribute. Let vector R[i] represent the result of route lookup for flow i, where R[i][e] is the fraction of traffic of flow i on link e, according to the current routing state. For example, the result of route lookup for the use case in Section 2 can be represented as the following:

```
       R[0] R[1] avabw delay bg-tr
link1  1    0       1G   2ms  ...
link2  1    0     100M   5ms  ...
link3  1    1     100M   5ms
Link4  1    1     100M   5ms
link5  0    1     100M   7ms
link6  0    1       1G   4ms
...
linkM
```

Although a routing-state query without abstraction/compression will return all of the data shown above, route abstraction/compression will select only a subset link attributes (columns) and some links (rows). Elimination of links from the complete result achieves compression but may result in loss of information to the application. Hence, a specification on conditions whether the elimination of a set of links from the complete result leads to information loss or not is the key to the problem definition. Such a specification, however, can be provided only by the application itself.

Specifically, in the general case, the result from the routing-state query will become the input parameters for the algorithms in the network application, to help the application to make decisions. Let x be the vector of the decision variables in the application. Then, one can identify that a generic structure of the application is to solve/optimize obj(x), subject to two types of constraints on x: (1) those do not involve the results from the routing state query; and (2) those do. Let the first type limit x in X0. Consider the second type. The state of art in algorithmic design typically handles only linear constraints, and hence the set S of constraints of this type will be of the format a_k x <= b_k, where a_k is a vector, and b_k a constant. Hence, it is in a_k or b_k where the result from the routing-state query appears. Let A x <= b as a matrix format to represent the whole set of constraints.

Now, consider the case that a link appears in the complete result of a RSADE query, but its parameters do not appear in a boundary constraint among the aforementioned constraints, then the link may not need to appear in a compressed RSADE query.

[Equivalence]: Two constraint sets S_1 and S_2 of a network function are equivalent if and only if they limit the decision variables in the same way: X_0 ^ {x: A_1 x <= b_1 } = X_0 ^ {x: A_2 x <= b_2 }.

[Redundant]: A constraint s is redundant to a constraint set S if and only if s in S and the two sets S and S-{s} are equivalent.

[Minimal Constraint Set]: A constraint set S is minimal if and only if for any s in S, s is not redundant.

[Equivalent Routing-State Query]

A declarative equivalence based routing-state query is one where the querier (application) declares X_0 and a set of constraints S = {a_k x <= b_k}. If the attribute of a link does not appear in a minimal constraint set, the link can be eliminated from the routing-state result.

A concern one may have is that the preceding definition may be limited. Consider the case of hierarchical networks, where the upper-layer network (i.e., the network application) conducts routing (traffic engineering) in its layer and uses RSADE to obtain the state of the lower layer. Let flows be the n(n-1) source-destination pairs in the upper layer network with n nodes. Let x be the set of decision variables controlling the routing in the upper-layer, where each element is the routing on each of the preceding flows. Let X_0 encode the constraints on traffic demand. We have the following result:

[UTE Completeness]

Any upper-layer routing (traffic engineering) algorithm where the goal of RSADE in the lower-layer network is to avoid congestion of shared links or shared risk groups can be implemented using the declarative equivalence based routing-state query. We refer to this as the upper-layer traffic engineering (UTE). Let A = R and b = cap. Then the RSADE query returns a link only if the link may become a bottleneck in the upper layer network.

5. Security Considerations

This document has not conducted its security analysis.

6. IANA Considerations

This document requires the definition of a new cost-mode named path-vector.

7. Acknowledgments

The author thanks discussions with Jun Bi and Andreas Voellmy.

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.

8.2. Informative References

[I-D.lee-alto-app-net-info-exchange]
Lee, Y., Bernstein, G., Choi, T., and D. Dhody, "ALTO Extensions to Support Application and Network Resource Information Exchange for High Bandwidth Applications", draft-lee-alto-app-net-info-exchange-02 (work in progress), July 2013.

[I-D.yang-alto-path-vector]
Bernstein, G., Lee, Y., Roome, W., Scharf, M., and Y. Yang, "ALTO Topology Extension: Path Vector as a Cost Mode", draft-yang-alto-path-vector-00 (work in progress), March 2015.

[RFC7285] Alimi, R., Penno, R., Yang, Y., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, September 2014.

Authors' Addresses

Kai Gao
Tsinghua University
30 Shuangqinglu Street
Beijing 100084
China

Xing (Tony) Wang
Tongji University
4800 CaoAn Road
Shanghai 210000

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

Email: yry@cs.yale.edu

G. Robert Chen
Huawei
Nanjing
China

Email: chenguohai@huawei.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 29, 2017

S. Randriamasy
W. Roome
Nokia Bell Labs
N. Schwan
Thales Deutschland

April 27, 2017

Multi-Cost ALTO
draft-ietf-alto-multi-cost-10

Abstract

The ALTO (Application Layer-Traffic Optimization) Protocol ([RFC7285]) defines several services that return various metrics describing the costs between network endpoints.

This document defines a new service that allows an ALTO Client to retrieve several cost metrics in a single request for an ALTO Filtered Cost Map and Endpoint Cost Map. In addition, it extends the constraints to further filter those maps by allowing a client to specify a logical combination of tests on several cost metrics.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

IETF has defined ALTO services in [RFC7285] to provide 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 such as the topological distance, that affect performance of the data transmission between the hosts. 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 a view of the Internet called a Network Map and composed of Provider defined locations spanning from subnets to several Autonomous Systems (AS). ALTO may also convey the Provider determined Costs between Network Map locations or between groups of individual endpoints.

Current ALTO Cost Types provide values 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 (CDN), Data Centers and applications that need additional information to select their endpoints or network locations. Thus a multitude of new Cost Types that better reflect the requirements of these applications are expected to be specified.

The ALTO protocol [RFC7285], which this document refers to as the base protocol, restricts ALTO Cost Maps and Endpoint Cost Services to only one Cost Type per ALTO request. To retrieve information for several Cost Types, an ALTO Client must send several separate requests to the Server.

It is far more efficient, in terms of Round Trip Time (RTT), traffic, and processing load on the ALTO Client and Server, to get all costs with a single query/response transaction. 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. Another advantage is consistency: providing values for several Cost Types in one single batch is useful for ALTO Clients needing synchronized ALTO information updates. This document defines how to retrieve multiple cost metrics in a single request for ALTO Filtered Cost Maps and Endpoint Cost Maps. To ensure compatibility with legacy ALTO Clients, only the Filtered Cost Map and Endpoint Cost Map services are extended to return Multi-Cost values.

Along with multi-cost values queries, the filtering capabilities need to be extended to allow constraints on multiple metrics. The base protocol allows an ALTO Client to provide optional constraint tests for a Filtered Cost Map or the Endpoint Cost Service, where the

constraint tests are limited to the AND-combination of comparison tests on the value of the (single) requested Cost Type. However, applications that are sensitive to several metrics and struggle with complicated network conditions may need to arbitrate between conflicting objectives such as routing cost and network performance. To this end, this document extends the base protocol with constraints that may test multiple metrics and may be combined with logical 'ORs' as well as logical 'ANDs'. This allows an application to make requests such as: "select solutions with either (moderate "hopcount" AND high "routingcost") OR (higher "hopcount" AND moderate "routingcost")".

This document is organized as follows: Section 2 defines terminology used in this document. Section 3 gives a non-normative overview of the multi-cost extensions, and Section 4 gives their formal definitions. Section 5 gives several complete examples. The remaining sections describe the IANA and privacy considerations.

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].

When the words appear in lower case, their natural language meaning is used.

2. Terminology

- o ALTO transaction: A request/response exchange between an ALTO Client and an ALTO Server.

- o Client: This term refers to an ALTO client, when used with a capital "C".

- o Endpoint (EP): An endpoint is defined as in {2.1} of [RFC7285]. It can be for example 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 Server: This term refers to an ALTO server, when used with a capital "S".

References with curly brackets such as '{1.2.3}' are to sections in the ALTO protocol specification [RFC7285], to avoid overloading the document with citations of [RFC7285].

3. Overview Of Approach

The following is a non-normative overview of the multi-cost extensions defined in this document. It assumes the reader is familiar with Cost Map resources in the ALTO Protocol ([RFC7285]).

3.1. Multi-Cost Data Format

Formally, the cost entries in an ALTO Cost Map can be any type of JSON value (see the DstCosts object in {11.2.3.6}). However, that section also says that an implementation may assume costs are JSON numbers, unless the implementation is using an extension which signals a different data type.

Therefore this document extends the definition of a Cost Map to allow a cost to be an array of costs, one per metric, instead of just one number. For example, here is a Cost Map with the "routingcost" and "hopcount" metrics. Note that this is identical to a regular ALTO Cost Map, except that the values are arrays instead of numbers. The multiple metrics are listed in member "multi-cost-types", indicating to the Client how to map values in the array to cost metrics.

```
{
 "meta" : {
   "dependent-vtags" : [ ... ],
   "cost-type" : {},
   "multi-cost-types" : [
     {"cost-mode": "numerical", "cost-metric": "routingcost"},
     {"cost-mode": "numerical", "cost-metric": "hopcount"}
   ]
 }
 "cost-map" : {
   "PID1": { "PID1":[1,0],  "PID2":[5,23],  "PID3":[10,5] },
   ...
 }
}
```

Note also the presence of member ’"cost-type" : {}’ to maintain backwards compatibility with [RFC7285].

3.2. Compatibility With Legacy ALTO Clients

This document does not define any new media types. Instead, as described below, it extends the specifications in the ALTO Server’s Information Resource Directory (IRD) so that legacy Clients will not request array-valued Multi Cost Map resources. This relies on the requirement that ALTO Clients MUST ignore unknown fields ({8.3.7}).

3.3. Filtered Multi Cost Map Resources

This document extends the Filtered Cost Map service to allow the same resource to return either a single-valued Cost Map, as defined in [RFC7285], or an array-valued Multi Cost Map, as defined in this document. An extended Filtered Cost Map resource has a new capability, "max-cost-types". The value is the maximum number of cost types this resource can return for one request. The existence of this capability means the resource understands the extensions in this document.

For example, the following fragment from an IRD defines an extended Filtered Cost Map resource:

```
"filtered-multicost-map" : {
  "uri" : "http://alto.example.com/multi/costmap/filtered",
  "media-types" : [ "application/alto-costmap+json" ],
  "accepts" : [ "application/alto-costmapfilter+json" ],
  "uses" : [ "my-default-network-map" ],
  "capabilities" : {
    "max-cost-types" : 2,
    "cost-type-names" : [ "num-routingcost",
                          "num-hopcount" ],
    ...
  }
```

A legacy ALTO Client will ignore the "max-cost-types" capability, and will send a request with the input parameter "cost-type" describing the desired cost metric, as defined in [RFC7285]. The ALTO Server will return a single-valued legacy Cost Map.

However, a multi-cost-aware ALTO Client will realize that this resource supports the multi-cost extensions, and can send a POST request with the new input parameter "multi-cost-types", whose value is an array of cost types. Because the request has the "multi-cost-types" parameter (rather than the "cost-type" parameter defined in the base protocol), the Server realizes that the ALTO Client also supports the extensions in this document, and hence responds with a Multi Cost Map, with the costs in the order listed in "multi-cost-types".

3.4. Endpoint Cost Service Resources

Section {4.1.4} of [RFC7285] specifies that "The Endpoint Cost Service allows an ALTO Server to return costs directly amongst endpoints.", whereas the Filtered Cost Map Service returns costs amongst PIDs. This document uses the technique described in

Section 3.3 to extend the Endpoint Cost Service to return array-valued costs to ALTO Clients who also are aware of these extensions.

3.5. Full Cost Map Resources

Section {11.3.2.3} of [RFC7285] requires a Filtered Cost Map to return the entire Cost Map if the ALTO Client omits the source and destination PIDs. Hence a Multi-Cost aware ALTO Client can use an extended Filtered Cost Map resource to get a full Multi Cost Map.

Full Cost Map resources are GET-mode requests. The response for a Full Cost Map conveying multiple cost types would include a "meta" field that would itself include a "cost-type" field, that would list several values corresponding to the cost types of the cost map. A legacy ALTO Client would not be able to understand this list. Neither would it be able to interpret the cost values array provided by a Multi-Cost full maps.

3.6. Extended Constraint Tests

[RFC7285] defines a simple constraint test capability for Filtered Cost Maps and Endpoint Cost Services. If a resource supports constraints, the Server restricts the response to costs that satisfy a list of simple predicates provided by the ALTO Client. For example, if the ALTO Client gives the constraints

```
"constraints": ["ge 10", "le 20"]
```

Then the Server only returns costs in the range [10,20].

To be useful with multi-cost requests, the constraint tests require several extensions.

3.6.1. Extended constraint predicates

First, because a multi-cost request involves more than one cost metric, the simple predicates must be extended to specify the metric to test. Therefore we extend the predicate syntax to "[##] op value", where "##" is the index of a cost metric in this multi-cost request.

3.6.2. Extended logical combination of predicates

Second, once multiple cost metrics are involved, the "AND" of simple predicates is no longer sufficient. To be useful, Clients must be able to express "OR" tests. Hence we add a new field, "or-constraints", to the Client request. The value is an array of arrays

of simple predicates, and represents the OR of ANDs of those predicates.

Thus, the following request tells the Server to limit its response to cost points with "routingcost" <= 100 AND "hopcount" <= 2, OR else "routingcost" <= 10 AND "hopcount" <= 6:

```
{
  "multi-cost-types": [
      {"cost-metric": "routingcost", "cost-mode": "numerical"},
      {"cost-metric": "hopcount",    "cost-mode": "numerical"}
  ],
  "or-constraints": [
      ["[0] le 100", "[1] le 2"],
      ["[0] le 10",  "[1] le 6"]
  ],
  "pids": {...}
}
```

Note that a "constraints" parameter with the array of predicates [P1, P2, ...] is equivalent to an "or-constraints" parameter with one array of value [[P1, P2, ...]]. A Client is therefore allowed to express either "constraints" or "or-constraints" but not both.

3.6.3. Testable Cost Types in constraints

Finally, a Client may want to test a cost type whose actual value is irrelevant, as long as it satisfies the tests. For example, a Client may want the value of the cost metric "routingcost" for all PID pairs that satisfy constraints on the metric "hopcount", without needing the actual value of "hopcount".

To this end, we add a specific parameter named "testable-cost-types", that does not contain the same cost types as parameter "multi-cost-types". The Client can express constraints only on cost types listed in "testable-cost-types".

For example, the following request tells the Server to return just "routingcost" for those source and destination pairs for which "hopcount" is <= 6:

```
{
  "multi-cost-types": [
      {"cost-metric": "routingcost", "cost-mode": "numerical"},
  ],
  "testable-cost-types": [
      {"cost-metric": "hopcount", "cost-mode": "numerical"},
  ],
  "constraints": ["[0] le 6"],
  "pids": {...}
}
```

3.6.4. Testable Cost Type Names in IRD capabilities

In [RFC7285], when a resource’s capability "constraints" is true, the Server accepts constraints on all the cost types listed in the "cost-type-names" capability. However, some ALTO Servers may not be willing to allow constraint tests on all available cost metrics. Therefore the Multi-Cost ALTO protocol extension defines the capability field "testable-cost-type-names". Like "cost-type-names", it is an array of cost type names. If present, that resource only allows constraint tests on the cost types in that list. "testable-cost-type-names" must be a subset of "cost-type-names".

3.6.5. Legacy ALTO Client issues

While a multi-cost-aware Client will recognize the "testable-cost-type-names" field, and will honor those restrictions, a legacy Client will not. Hence, when "constraints" has the value ’true’, a legacy client may send a request with a constraint test on any of the cost types listed in "cost-type-names".

To avoid that problem, the "testable-cost-type-names" and "cost-constraints" fields are mutually exclusive: a resource may define one or the other capability, but MUST NOT define both. Thus a resource that does not allow constraint tests on all cost metrics will set "testable-cost-type-names" to the testable metrics, and will set "cost-constraints" to "false". A multi-cost-aware Client will recognize the "testable-cost-type-names" field, and will realize that its existence means the resource does allow (limited) constraint tests, while a legacy Client will think that resource does not allow constraint tests at all. To allow legacy Clients to use constraint tests, the ALTO Server can define an additional resource with "cost-constraints" set to "true" and "cost-type-names" set to the metrics which can be tested.

In the IRD example below, the resource "filtered-cost-map-extended" provides values for three metrics: "num-routingcost", "num-hopcount" and "num-bwscore". The capability "testable-cost-type-names"

indicates that the Server only allows constraints on "routingcost" and "hopcount". A multi-cost capable Client will see this capability, and will limit its constraint tests to those metrics. Because capability "cost-constraints" is false (by default), a legacy Client will not use constraint tests on this resource at all.

The second resource, "filtered-multicost-map", is similar to the first, except that all the metrics it returns are testable. Therefore it sets "cost-constraints" to "true", and does not set the "testable-cost-type-names" field. A legacy Client that needs a constraint test will use this resource rather than the first. A multi-cost-aware Client that does not need to retrieve the "num-bwscore" metric may use either resource.

Note that if a multi-cost Server specifies a "filtered-cost-map-extended", it will most likely not specify an "filtered-multicost-map" if the capabilities of the latter are covered by the capabilities of the former or unless the "filtered-multicost-map" ressource is also intended for legacy Clients.

```
"filtered-cost-map-extended" : {
   "uri" : "http://alto.example.com/multi/extn/costmap/filtered",
   "media-types" : [ "application/alto-costmap+json" ],
   "accepts" : [ "application/alto-costmapfilter+json" ],
   "uses" : [ "my-default-network-map" ],
   "capabilities" : {
      "max-cost-types" : 3,
      "cost-type-names" : [ "num-routingcost",
                            "num-hopcount",
                            "num-bwscore"],
      "testable-cost-type-names" : [ "num-routingcost",
                                     "num-hopcount" ]
   }
},

"filtered-multicost-map" : {
   "uri" : "http://alto.example.com/multi/costmap/filtered",
   "media-types" : [ "application/alto-costmap+json" ],
   "accepts" : [ "application/alto-costmapfilter+json" ],
   "uses" : [ "my-default-network-map" ],
   "capabilities" : {
     "cost-constraints" : true,
     "max-cost-types" : 2,
     "cost-type-names" : [ "num-routingcost",
                           "num-hopcount"],
   }
}
```

4. Protocol Extensions for Multi-Cost ALTO Transactions

This section formally specifies the extensions to [RFC7285] to support Multi-Cost ALTO transactions.

This document uses the notation rules specified in {8.2}. In particular, an optional field is enclosed by [ ]. In the definitions, the JSON names of the fields are case sensitive. An array is indicated by two numbers in angle brackets, <m..n>, where m indicates the minimal number of values and n is the maximum. When this document uses * for n, it means no upper bound.

4.1. Filtered Cost Map Extensions

This document extends Filtered Cost Maps, as defined in {11.3.2} of [RFC7285], by adding new input parameters and capabilities, and by returning JSONArrays instead of JSONNumbers as the cost values.

The media type (11.3.2.1}, HTTP method (11.3.2.2} and "uses" specifications (11.3.2.5} are unchanged.

4.1.1. Capabilities

The filtered cost map capabilities are extended with two new members:

- o max-cost-types,

- o testable-cost-type-names

The capability "max-cost-types" indicates whether this resource supports the Multi-Cost ALTO extensions, and the capability "testable-cost-type-names" allows the resource to restrict constraint tests to a subset of the available cost types. With these two additional members, the FilteredCostMapCapabilities object in {11.3.2.4} is structured as follows:

```
object {
   JSONString cost-type-names<1..*>;
   [JSONBool cost-constraints;]
   [JSONNumber max-cost-types;]
   [JSONString testable-cost-type-names<1..*>;]
} FilteredCostMapCapabilities;
```

cost-type-names: As defined in {11.3.2.4} of [RFC7285].

cost-constraints: As defined in {11.3.2.4} of [RFC7285]. Thus if "cost-constraints" is true, the resource MUST accept constraint tests on any cost type in "cost-type-names". Note in addition

that if "cost-constraints" is "true", the "testable-cost-type-names" capability MUST NOT be present

max-cost-types: If present with value N greater than 0, this resource understands the multi-cost extensions in this document, and can return a Multi Cost Map with any combination of N or fewer cost types in the "cost-type-names" list. If omitted, the default value is 0.

testable-cost-type-names: If present, the resource allows constraint tests, but only on the cost type names in this array. Each name in "testable-cost-type-names" MUST also be in "cost-type-names". If "testable-cost-type-names" is present, the "cost-constraints" capability MUST NOT be true.

As discussed in Section 3.6.4, this capability is useful when a Server is unable or unwilling to implement constraint tests on all cost types. As discussed in Section 3.6.5, "testable-cost-type-names" and "cost-constraints" are mutually exclusive to prevent legacy Clients from issuing constraint tests on untestable cost types.

4.1.2. Accept Input Parameters

The ReqFilteredCostMap object in {11.3.2.3} of [RFC7285] is extended as follows:

```
object {
   [CostType cost-type;]
   [CostType multi-cost-types<1..*>;]
   [CostType testable-cost-types<1..*>;]
   [JSONString constraints<0..*>;]
   [JSONString or-constraints<1..*><1..*>;]
   [PIDFilter pids];
} ReqFilteredCostMap;
```

cost-type: As defined in {11.3.2.3} of [RFC7285], with the additional requirement that the Client MUST specify either "cost-type" or "multi-cost-types", but MUST NOT specify both. Therefore this field is made optional. When placing a single cost request as specified in [RFC7285], a Client MUST use "cost-type".

multi-cost-types: If present, the ALTO Server MUST return array-valued costs for the cost types in this list. For each entry, the "cost-metric" and "cost-mode" fields MUST match one of the supported cost types indicated in member "cost-type-names" of this resource’s "capabilities" field (Section 4.1.1). The Client MUST

NOT use this field unless this resource's "max-cost-types" capability exists and has a value greater than 0. This field MUST NOT have more than "max-cost-types" cost types. The Client MUST specify either "cost-type" or "multi-cost-types", but MUST NOT specify both.

Note that if "multi-cost-types" has one cost type, the values in the cost map will be arrays with one value.

testable-cost-types: A list of cost types used for extended constraint tests, as described for the "constraints" and "or-constraints" parameters. These cost types must either be a subset of the cost types in the resource's "testable-cost-type-names" capability (Section 4.1.1), or else, if the resource's capability "cost-constraints" is true, a subset of the cost types in the resource's "cost-type-names" capability.

If "testable-cost-types" is omitted, it is assumed to have the cost types in "multi-cost-types" or "cost-type".

This feature is useful when a Client wants to test a cost type whose actual value is irrelevant, as long as it satisfies the tests. For example, a Client may want the cost metric "routingcost" for those PID pairs whose "hopcount" is less than 10. The exact hopcount does not matter.

constraints: If this resource's "max-cost-types" capability (Section 4.1.1) has the value 0 (or is not defined), this parameter is as defined in {11.3.2.3} of [RFC7285]: an array of constraint tests related to each other by a logical AND. In this case it MUST NOT be specified unless the resource's "cost-constraints" capability is "true".

If this resource's "max-cost-types" capability has a value greater than 0, then this parameter is an array of extended constraint predicates as defined below and related to each other by a logical AND. In this case, it MAY be specified if the resource allows constraint tests (the resource's "cost-constraints" capability is "true" or its "testable-cost-type-names" capability is not empty).

This parameter MUST NOT be specified if the "or-constraints" parameter is specified.

An extended constraint predicate consists of two or three entities separated by white space: (1) an optional cost type index, of the form "[#]", with default value "[0]", (2) a required operator, and (3) a required target value. The operator and target value are as defined in {11.3.2.3} of [RFC7285]. The cost type index, i,

specifies the cost type to test. If the "testable-cost-type" parameter is present, the test applies to the i’th cost type in "testable-cost-types", starting with index 0. Otherwise if the "multi-cost-types" parameter is present, the test applies to the i’th cost type in that array. If neither parameters are present, the test applies to the cost type in the "cost-type" parameter, in which case the index MUST be 0. Regardless of how the tested cost type is selected, it MUST be in the resource’s "testable-cost-type-names" capability, or, if not present, in the "cost-type-names" capability.

As an example, suppose "multi-cost-types" has the single element "routingcost", "testable-cost-types" has the single element "hopcount", and "constraints" has the single element "[0] le 5". This is equivalent to the database query "SELECT and provide routingcost WHERE hopcount <= 5".

Note that the index is optional, so a constraint test as defined in {11.3.2.3}, such as "le 10", is equivalent to "[0] le 10". Thus legacy constraint tests are also legal extended constraint tests.

Note that a "constraints" parameter with the array of extended predicates [P1, P2, ...] is equivalent to an "or-constraints" parameter as defined below, with the value [[P1, P2, ...]].

or-constraints: A JSONArray of JSONArrays of JSONStrings, where each string is an extended constraint predicate as defined above. The "or-constraint" tests are interpreted as the logical OR of ANDs of predicates. That is, the ALTO Server should return a cost point only if it satisfies all constraints in any one of the sub-arrays.

This parameter MAY be specified if this resource’s "max-cost-types" capability is defined with a value greater than 0 (Section 4.1.1), and if the resource allows constraint tests (the resource’s "cost-constraints" capability is "true" or its "testable-cost-type-names" capability is not empty). Otherwise this parameter MUST NOT be specified.

This parameter MUST NOT be specified if the "constraints" parameter is specified.

This parameter MUST NOT contain any empty array of AND predicates. An empty array would be equivalent to a constraint that is always "true". An OR combination including such a constraint would be always "true" and thus useless.

As an example, suppose "multi-cost-types" has the two elements "routingcost" and "bandwidthscore", and "testable-cost-types" has the two elements "routingcost" and "hopcount", and "or-constraints" has the two elements ["[0] le 100", "[1] le 2"] and ["[0] le 10", "[1] le 6"]. This is equivalent to the words: "SELECT and provide routingcost and bandwidthscore WHERE ("routingcost" <= 100 AND "hopcount" <= 2) OR ("routingcost" <= 10 AND "hopcount" <= 6)".

Note that if the "max-cost-types" capability has a value greater than 0, a Client MAY use the "or-constraints" parameter together with the "cost-type" parameter. That is, if the Client and Server are both aware of the extensions in this document, a Client MAY use an "OR" test for a single-valued cost request.

pids: As defined in {11.3.2.3} of [RFC7285].

4.1.3. Response

If the Client specifies the "cost-type" input parameter, the response is exactly as defined in {11.2.3.6} of [RFC7285]. If the Client provides the "multi-cost-types" instead, then the response is changed as follows:

- o In "meta", the value of field "cost-type" will be ignored by the receiver and set to {}. Instead, the field "multi-cost-types" is added with the same value as the "multi-cost-types" input parameter.

- o The costs are JSONArrays, instead of JSONNumbers. All arrays have the same cardinality as the "multi-cost-types" input parameter, and contain the cost type values in that order. If a cost type is not available for a particular source and destination, the ALTO Server MUST use the JSON "null" value for that array element. If none of the cost types are available for a particular source and destination, the ALTO Server MAY omit the entry for that source and destination.

4.2. Endpoint Cost Service Extensions

This document extends the Endpoint Cost Service, as defined in {11.5.1} of [RFC7285], by adding new input parameters and capabilities, and by returning JSONArrays instead of JSONNumbers as the cost values.

The media type {11.5.1.1}, HTTP method {11.5.1.2} and "uses" specifications {11.5.1.5} are unchanged.

4.2.1. Capabilities

The extensions to the Endpoint Cost Service capabilities are identical to the extensions to the Filtered Cost Map (see Section 4.1.1).

4.2.2. Accept Input Parameters

The ReqEndpointCostMap object in {11.5.1.3} of [RFC7285] is extended as follows:

```
object {
   [CostType cost-type;]
   [CostType multi-cost-types<1..*>;]
   [CostType testable-cost-types<1..*>;]
   [JSONString constraints<0..*>;]
   [JSONString or-constraints<1..*><1..*>;]
   EndpointFilter endpoints;
} ReqEndpointCostMap;
```

cost-type: As defined in {11.5.1.3} of [RFC7285], with the additional requirement that the Client MUST specify either "cost-type" or "multi-cost-types", but MUST NOT specify both.

multi-cost-types: If present, the ALTO Server MUST return array-valued costs for the cost types in this list. For each entry, the "cost-metric" and "cost-mode" fields MUST match one of the supported cost types indicated in this resource’s "capabilities" field (Section 4.2.1). The Client MUST NOT use this field unless this resource’s "max-cost-types" capability exists and has a value greater than 0. This field MUST NOT have more than "max-cost-types" cost types. The Client MUST specify either "cost-type" or "multi-cost-types", but MUST NOT specify both.

Note that if "multi-cost-types" has one cost type, the values in the cost map will be arrays with one value.

testable-cost-types, constraints, or-constraints: Defined equivalently to the corresponding input parameters for an extended Filtered Cost Map (Section 4.1.2).

endpoints: As defined in {11.5.1.3} of [RFC7285].

4.2.3. Response

The extensions to the Endpoint Cost Service response are similar to the extensions to the Filtered Cost Map response (Section 4.1.3). Specifically, if the Client specifies the "cost-type" input parameter, the response is exactly as defined in {11.5.1.6} of [RFC7285]. If the Client provides the "multi-cost-types" instead, then the response is changed as follows:

- In "meta", the value of field "cost-type" will be ignored by the receiver and set to {}. Instead, the field "multi-cost-types" is added with the same value as the "multi-cost-types" input parameter.

- The costs are JSONArrays, instead of JSONNumbers. All arrays have the same cardinality as the "multi-cost-types" input parameter, and contain the cost type values in that order. If a cost type is not available for a particular source and destination, the ALTO Server MUST use the JSON "null" value for that array element. If none of the cost types are available for a particular source and destination, the ALTO Server MAY omit the entry for that source and destination.

5. Examples

This section provides examples of Multi-Cost ALTO transactions. It uses cost metrics, in addition to the mandatory legacy ’routingcost’, that are deliberately irrelevant and not registered at the IANA.

5.1. Information Resource Directory

The following is an example of an ALTO Server’s Information Resource Directory. In addition to Network and Cost Map resources, it defines two Filtered Cost Map and an Endpoint Cost Service, which all understand the multi-cost extensions.

```
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: 2704
Content-Type: application/alto-directory+json

{
  "meta" : {
    "default-alto-network-map" : "my-default-network-map",
```

```
      "cost-types" : {
        "num-routing" : {
          "cost-mode" : "numerical",
          "cost-metric" : "routingcost"
        },
        "num-shoesize" : {
          "cost-mode" : "numerical",
          "cost-metric" : "shoesize"
        },
        "num-scenery" : {
          "cost-mode" : "numerical",
          "cost-metric" : "sceneryrate"
        }
      }
    },
    "resources" : {
      "my-default-network-map" : {
        "uri" : "http://alto.example.com/networkmap",
        "media-type" : "application/alto-networkmap+json"
      },
      "numerical-routing-cost-map" : {
        "uri" : "http://alto.example.com/costmap/num-routing",
        "media-types" : [ "application/alto-costmap+json" ],
        "uses" : [ "my-default-network-map" ],
        "capabilities" : {
          "cost-type-names" : [ "num-routing" ]
        }
      },
      "numerical-shoesize-cost-map" : {
        "uri" : "http://alto.example.com/costmap/num-shoesize",
        "media-types" : [ "application/alto-costmap+json" ],
        "uses" : [ "my-default-network-map" ],
        "capabilities" : {
          "cost-type-names" : [ "num-shoesize" ]
        }
      },
      "filtered-multicost-map" : {
        "uri" : "http://alto.example.com/multi/costmap/filtered",
        "media-types" : [ "application/alto-costmap+json" ],
        "accepts" : [ "application/alto-costmapfilter+json" ],
        "uses" : [ "my-default-network-map" ],
        "capabilities" : {
          "cost-constraints" : true,
          "max-cost-types" : 2,
          "cost-type-names" : [ "num-routingcost",
                                "num-shoesize" ]
        }
      },
```

```
      "filtered-cost-map-extended" : {
        "uri" : "http://alto.example.com/multi/extn/costmap/filtered",
        "media-types" : [ "application/alto-costmap+json" ],
        "accepts" : [ "application/alto-costmapfilter+json" ],
        "uses" : [ "my-default-network-map" ],
        "capabilities" : {
          "max-cost-types" : 3,
          "cost-type-names" : [ "num-routingcost",
                                "num-shoesize",
                                "num-scenery"],
          "testable-cost-type-names" : [ "num-routingcost",
                                         "num-shoesize" ]
        }
      },
      "endpoint-multicost-map" : {
        "uri" : "http://alto.example.com/multi/endpointcost/lookup",
        "media-types" : [ "application/alto-endpointcost+json" ],
        "accepts" : [ "application/alto-endpointcostparams+json" ],
        "uses" : [ "my-default-network-map" ],
        "capabilities" : {
          "cost-constraints" : true,
          "max-cost-types" : 2,
          "cost-type-names" : [ "num-routingcost",
                                "num-shoesize" ]
        }
      }
    }
  }
```

5.2. Multi-Cost Filtered Cost Map: Example #1

This example illustrates a simple multi-cost ALTO transaction. The ALTO Server provides two Cost Types, "routingcost" and "shoesize", both in "numerical" mode. The Client wants the entire Multi-Cost Map. The Server does not know the value of "routingcost" between PID2 and PID3, and hence returns the value ’null’ for "routingcost" between PID2 and PID3.

```
POST /multi/costmap/filtered" HTTP/1.1
Host: alto.example.com
Accept: application/alto-costmap+json,application/alto-error+json
Content-Type: application/alto-costmapfilter+json
Content-Length: 206

{
  "multi-cost-types": [
    {"cost-mode": "numerical", "cost-metric": "routingcost"},
    {"cost-mode": "numerical", "cost-metric": "shoesize"}
  ],
  "pids" : {
    "srcs" : [ ],
    "dsts" : [ ]
  }
}

HTTP/1.1 200 OK
Content-Type: application/alto-costmap+json
Content-Length: 549

{
 "meta" : {
   "dependent-vtags" : [
     {"resource-id": "my-default-network-map",
      "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"
     }
   ],
   "cost-type" : {},
   "multi-cost-types" : [
     {"cost-mode": "numerical", "cost-metric": "routingcost"},
     {"cost-mode": "numerical", "cost-metric": "shoesize"}
   ]
 }
 "cost-map" : {
   "PID1": { "PID1":[1,0],   "PID2":[4,3],    "PID3":[10,2]   },
   "PID2": { "PID1":[15,5],  "PID2":[1,0],    "PID3":[null,9] },
   "PID3": { "PID1":[20,12], "PID2":[null,1], "PID3":[1,0]    }
 }
}
```

5.3. Multi-Cost Filtered Cost Map: Example #2

This example uses constraints to restrict the returned source/destination PID pairs to those with "routingcost" between 5 and 10, or "shoesize" equal to 0.

```
POST /multi/costmap/filtered HTTP/1.1
Host: alto.example.com
Accept: application/alto-costmap+json,application/alto-error+json
Content-Type: application/alto-costmapfilter+json
Content-Length: 333

{
  "multi-cost-types" : [
    {"cost-mode": "numerical", "cost-metric": "routingcost"},
    {"cost-mode": "numerical", "cost-metric": "shoesize"}
  ],
  "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-Type: application/alto-costmap+json
Content-Length: 461

{
  "meta" : {
    "dependent-vtags" : [
      {"resource-id": "my-default-network-map",
       "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"
      }
    ],
    "cost-type" : {},
    "multi-cost-types" : [
      {"cost-mode": "numerical", "cost-metric": "routingcost"},
      {"cost-mode": "numerical", "cost-metric": "shoesize"}
    ]
  }
  "cost-map" : {
    "PID1": { "PID1": [1,0], "PID3": [10,5] },
    "PID2": { "PID2": [1,0]                 }
  }
}
```

5.4. Multi-Cost Filtered Cost Map: Example #3

This example uses extended constraints to limit the response to cost points with ("routingcost" <= 10 and "shoesize" <= 2), or else ("routingcost" <= 3 and "shoesize" <= 6). Unlike the previous example, the Client is only interested in the "routingcost" cost type, and uses the "cost-type" parameter instead of "multi-cost-types" to tell the Server to return scalar costs instead of array costs.

In this example, "[0]" means the constraint applies to "routingcost" because that is the first cost type in the "testable-cost-types" parameter. (If "testable-cost-types" is omitted, it is assumed to be the same as "multi-cost-types".) The choice of using an index to refer to cost types aims at minimizing the length of the expression of constraints, especially for those combining several OR and AND expressions. It was also the shortest path from the constraints design in [RFC7285].

```
POST /multi/multicostmap/filtered HTTP/1.1
Host: alto.example.com
Accept: application/alto-costmap+json,application/alto-error+json
Content-Type: application/alto-costmapfilter+json
Content-Length: 390

{
  "cost-type" : {
    "cost-mode": "numerical", "cost-metric": "routingcost"
  },
  "testable-cost-types" : [
    {"cost-mode": "numerical", "cost-metric": "routingcost"},
    {"cost-mode": "numerical", "cost-metric": "shoesize"}
  ],
  "or-constraints": [
         ["[0] le 10", "[1] le 2"],
         ["[0] le 3",  "[1] le 6"]
  ],
  "pids" : {
    "srcs" : [ ],
    "dsts" : [ ]
  }
}
```

```
HTTP/1.1 200 OK
Content-Type: application/alto-costmap+json
Content-Length: 368

{
  "meta" : {
    "dependent-vtags" : [
      {"resource-id": "my-default-network-map",
       "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"
      }
    ],
    "cost-type" : {
      "cost-mode": "numerical", "cost-metric": "routingcost"
    }
  }
  "cost-map" : {
    "PID1": { "PID1": 1, "PID3": 10 },
    "PID2": { "PID2": 1 },
    "PID3": { "PID3": 1 }
  }
}
```

5.5. Multi-Cost Filtered Cost Map: Example #4

This example uses extended constraints to limit the response to cost points with ("routingcost" <= 10 and "shoesize" <= 2), or else ("routingcost" <= 3 and "shoesize" <= 6). In this example, the Client is interested in the "routingcost" and "sceneryrate" cost metrics, but not in the "shoesize" metric:

```
POST /multi/extn/costmap/filtered HTTP/1.1
Host: alto.example.com
Accept: application/alto-costmap+json,application/alto-error+json
Content-Type: application/alto-costmapfilter+json
Content-Length: 461

{
  "multi-cost-types" : [
    {"cost-mode": "numerical", "cost-metric": "routingcost"},
    {"cost-mode": "numerical", "cost-metric": "sceneryrate"}
  ],
  "testable-cost-types" : [
    {"cost-mode": "numerical", "cost-metric": "routingcost"},
    {"cost-mode": "numerical", "cost-metric": "shoesize"}
  ],
  "or-constraints": [
         ["[0] le 10", "[1] le 2"],
```

```
            ["[0] le 3",  "[1] le 6"]
    ],
    "pids" : {
      "srcs" : [ ],
      "dsts" : [ ]
    }
  }


  HTTP/1.1 200 OK
  Content-Type: application/alto-costmap+json
  Content-Length: 481

  {
    "meta" : {
      "dependent-vtags" : [
        {"resource-id": "my-default-network-map",
         "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"
        }
      ],
      "cost-type" : {},
      "multi-cost-types" : [
        {"cost-mode": "numerical", "cost-metric": "routingcost"},
        {"cost-mode": "numerical", "cost-metric": "sceneryrate"}
      ]
    }
    "cost-map" : {
      "PID1": { "PID1": [1,16] "PID3": [10,19] },
      "PID2": { "PID2": [1,8] },
      "PID3": { "PID3": [1,19] }
    }
  }
```

5.6. Endpoint Cost Service

This example uses the Endpoint Cost Service to retrieve the "routingcost" and "shoesize" for selected endpoints, limiting the response to costs with either low shoesize and reasonable routingcost (shoesize <= 2 and routingcost <= 10), or else low routingcost and reasonable shoesize (routingcost <= 3 and shoesize <= 6).

```
  POST /multi/endpointcost/lookup HTTP/1.1
  Host: alto.example.com
  Accept: application/alto-endpointcost+json,
          application/alto-error+json
  Content-Type: application/alto-endpoincostparams+json
  Content-Length: 455
```

```
{
  "multi-cost-types" : [
    {"cost-mode": "numerical", "cost-metric": "routingcost"},
    {"cost-mode": "numerical", "cost-metric": "shoesize"}
  ],
  "or-constraints": [
         ["[0] le 10", "[1] le 2"],
         ["[0] le 3",  "[1] le 6"]
  ],
  "endpoints" : {
    "srcs": [ "ipv4:192.0.2.2", "ipv6:2001:db8::1:0 ],
    "dsts": [
      "ipv4:192.0.2.89",
      "ipv4:198.51.100.34",
      "ipv4:203.0.113.45",
      "ipv6:2001:db8::10"
    ]
  }
}
```

```
HTTP/1.1 200 OK
Content-Length: 419
Content-Type: application/alto-endpointcost+json
```

```
{
  "meta" : {
    "multi-cost-types" : [
      {"cost-mode": "numerical", "cost-metric": "routingcost"},
      {"cost-mode": "numerical", "cost-metric": "shoesize"}
    ]
  }
  "endpoint-cost-map" : {
    "ipv4:192.0.2.2": {
      "ipv4:192.0.2.89":    [15, 5],
      "ipv4:203.0.113.45":  [4, 23]
    }
    "ipv6:2001:db8::1:0": {
      "ipv4:198.51.100.34": [16, 5],
      "ipv6:2001:db8::10":  [10, 2]
    }
  }
}
```

6. IANA Considerations

This document does not define any new media types or introduce any new IANA considerations.

7. Privacy And Security Considerations

This document does not introduce any privacy or security issues not already present in the ALTO protocol.

The Multi-Cost optimization even tends to reduce the on the wire data exchange volume, compared to multiple single cost ALTO transactions. Likewise, the risk related to massive Multi-Cost requests is moderated by the fact that Multi-Cost constraints additionally filter ALTO Server responses and thus reduce their volume.

Note that, because queries for multiple metrics represent a stronger fingerprinting signal than queries for a single metric, implementations of this protocol may leak more information about the ALTO client than would occur with a succession of individual queries. Though, in many cases it would already be possible to link those queries by using the source IP address or other existing information.

8. Acknowledgements

The authors would like to thank Richard Alimi, Fred Baker, Dhruv Dhodi, Vijay Gurbani, Dave Mac Dysan, Young Lee, Richard Yang, for fruitful discussions and feedback on this document and previous versions. Gao Kai, Hans Seidel, Richard Yang, Qiao Xiang and Wang Xin provided substantial review feedback and suggestions to the protocol design.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC7285] Almi, R., Penno, R., Yang, Y., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, September 2014.

9.2. Informative References

[RFC5693] "Application Layer Traffic Optimization (ALTO) Problem Statement", October 2009.

[RFC6708] "Application-Layer Traffic Optimization (ALTO) Requirements", February 2012.

Authors' Addresses

Sabine Randriamasy
Nokia Bell Labs
Route de Villejust
NOZAY 91460
FRANCE

Email: Sabine.Randriamasy@nokia-bell-labs.com

Wendy Roome
Nokia Bell Labs
124 Burlington Rd
Murray Hill, NJ 07974
USA

Email: ietf@wdroome.com

Nico Schwan
Thales Deutschland
Lorenzstrasse 10
Stuttgart 70435
Germany

Email: nico.schwan@thalesgroup.com

ALTO
Internet-Draft
Intended status: Informational
Expires: January 8, 2017

S. Kiesel
University of Stuttgart
M. Stiemerling
H-DA

July 7, 2016

Application Layer Traffic Optimization (ALTO) Cross-Domain Server Discovery
draft-kiesel-alto-xdom-disc-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.

In some deployment scenarios, in particular if the information about the network topology is partitioned and distributed over several ALTO servers, it may be needed to discover an ALTO server outside of the own network domain, in order to get appropriate guidance. This document details applicable scenarios, itemizes requirements, and specifies a procedure for ALTO cross-domain server discovery.

Technically, the algorithm specified in this document takes one IP address and a U-NAPTR Service Parameter (i.e., "ALTO:http" or "ALTO:https") as parameters. It performs DNS lookups (for NAPTR resource records in the in-addr.arpa. or ip6.arpa. tree) and returns one or more URI(s) of information resources related to that IP address.

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

Copyright Notice

Table of Contents

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 an HTTP-based client-server protocol [RFC7285], which can be used in various deployment scenarios [I-D.ietf-alto-deployments].

1.1. Multiple Information Sources and Partitioned Knowledge

The ALTO base protocol document [RFC7285] specifies the communication between an ALTO client and a single ALTO server. It is implicitly assumed that this server can answer any query, possibly with some kind of default value if no exact data is known. No special provisions were made for the case that the ALTO information originates from multiple sources, which are possibly under the control of different administrative entities (e.g., different ISPs) or that the overall ALTO information is partitioned and stored on several ALTO servers.

1.1.1. Classification of Solution Approaches

Various protocol extensions and other solutions have been proposed to deal with multiple information sources and partitioned knowledge. They can be classified as follows:

1 Ensure that all ALTO servers have the same knowlegde

1.1 Ensure data replication and synchronization within the provisioning protocol (cf. RFC 5693, Fig 1 [RFC5693]).

1.2 Use an Inter-ALTO-server data replication protocol. Possibly, the ALTO protocol itself - maybe with some extensions - could be used for that purpose; however, this has not been studied in detail so far.

2 Accept that different ALTO servers (possibly operated by different organizations, e.g., ISPs) do not have the same knowledge

2.1 Allow ALTO clients to send arbitrary queries to any ALTO server (e.g. the one discovered using [RFC7286]). If this server cannot answer the query itself, it will fetch the data on behalf of the client, using the ALTO protocol or a to-be-defined inter-ALTO-server request forwarding protocol.

2.2 Allow ALTO clients to send arbitrary queries to any ALTO server (e.g. the one discovered using [RFC7286]). If this server cannot answer the query itself, it will redirect the client to the "right" ALTO server that has the desired information, using a small to-be-defined extension of the ALTO protocol.

2.3 ALTO clients need to use some kind of "search engine" that indexes ALTO servers and redirects and/or gives cached results.

2.4 ALTO clients need to use a new discovery mechanism to discover the ALTO server that has the desired information and contact it directly.

1.1.2. Discussion of Solution Approaches

The provisioning or initialization protocol for ALTO servers (cf. RFC 5693, Fig 1 [RFC5693]) is currently not standardized. It was a conscious decision not to include this in the scope of the IETF ALTO working group. The reason is that there are many different kinds of information sources. This implementation specific protocol will adapt them to the ALTO server, which offers a standardized protocol to the ALTO clients. However, adding the task of synchronization between ALTO servers to this protocol (i.e., approach 1.1) would overload this protocol with a second functionality that requires standardization for seamless multi-domain operation.

For the 1.? solution approaches, in addition to general technical feasibility and issues like overhead and caching efficiency, another aspect to consider is legal liability. Operator "A" might prefer not to publish information about nodes in or paths between the networks of operators "B" and "C" through A’s ALTO server, even if A knew that information. This is not only a question of map size and processing load on A’s ALTO server. Operator A could also face legal liability issues if that information had a bad impact on the traffic engineering between B’s and C’s networks, or on their business models.

No specific actions to build a "search engine" based solution (approach 2.3) are currently known and it is unclear what could be the incentives to operate such an engine. Therefore, this approach is not considered in the remainder of this document.

1.2. The Need for Cross-Domain ALTO Server Discovery

Approaches 1.1, 1.2, 2.1, and 2.2 do not only require the specification of an ALTO protocol extension or a new protocol that runs between ALTO servers. A large-scale, maybe Internet-wide, multi-domain deployment would also need mechanisms by which an ALTO

server could discover other ALTO servers, learn which information is available where, and ideally also who is authorized to publish information related to a given part of the network. Approach 2.4 needs the same mechanisms, except that they are used on the client-side instead of the server-side.

It is sometimes questioned whether there is a need for a solution that allows clients to ask arbitrary queries, even if the ALTO information is partitioned and stored on many ALTO servers. The main argument is, that clients are supposed to optimize the traffic from and to themselves, and that the information needed for that is most likely stored on a "nearby" ALTO server, i.e., the one that can be discovered using [RFC7286]. However, there are scenarios where the ALTO client is not co-located with an endpoint of the to-be-optimized data transmission. Instead, the ALTO client is located at a third party, which takes part in the application signaling, e.g., a so-called "tracker" in a peer-to-peer application. One such scenario, where it is advantageous to place the ALTO client not at an endpoint of the user data transmission, is analyzed in Appendix B.

1.3. Solution Approach

Several solution approaches for cross-domain ALTO server discovery have been evaluated, using the criteria documented in Appendix A. One of them was to use the ALTO protocol itself for the exchange of information availability [I-D.kiesel-alto-alto4alto]. However, the drawback of that approach is that a new registration administration authority would have to be established.

This document specifies a DNS-based procedure for cross-domain ALTO server discovery, which was inspired by "Location Information Server (LIS) Discovery Using IP Addresses and Reverse DNS" [RFC7216]. The primary goal is that this procedure can be used on the client-side (i.e., approach 2.4), but together with new protocols or protocol extensions it could also be used to implement the other solution approaches itemized above.

1.4. ALTO Requirements

During the design phase of the overall ALTO solution, two different server discovery scenarios have been identified and documented in the ALTO requirements document [RFC6708]. The first scenario, documented in Req. AR-32, can be supported using the discovery mechanisms specified in [RFC7286]. An alternative approach, based on IP anycast [I-D.kiesel-alto-ip-based-srv-disc], has also been studied. This document, in contrast, tries to address Req. AR-33.

1.5. Document History

This document is a direct successor of [I-D.kiesel-alto-3pdisc] and [I-D.kist-alto-3pdisc]. The scenario and mechanisms described here and in these documents have been referred to as "third-party server discovery" in the past. However, to avoid naming ambiguities with a completely different scenario, it has been renamed to "ALTO Cross-Domain Server Discovery".

1.6. Feedback

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

2. ALTO Cross-Domain Server Discovery Procedure Specification

2.1. Interface

The algorithm specified in this document takes one IP address and a U-NAPTR [RFC4848] Service Parameter (i.e., "ALTO:http" or "ALTO:https") as parameters. It performs DNS lookups (for NAPTR resource records) and returns one or more URI(s) of information resources related to that IP address.

2.2. Basic Principle

This algorithm closely follows [RFC7216] and re-uses parts of [RFC7286].

The algorithm sequentially tries two different lookup strategies. First, an ALTO-specific U-NAPTR record is searched in the "reverse tree", i.e., in subdomains of in-addr.arpa. or ip6.arpa. corresponding to the given IP address. If this lookup does not yield a usable result, further lookups with truncated domain names may be tried. The goal is to allow deployment scenarios that require fine-grained discovery on a per-IP basis, as well as large-scale scenarios where discovery is to be enabled for a large number of IP addresses with a small number of additional DNS resource records.

2.3. Step 1: Prepare Domain Name for Reverse DNS Lookup

This task takes the IP address parameter the procedure was called with and constructs a domain name, which is used for DNS lookups in subsequent tasks.

If the IP address given as a parameter to the procedure is an IPv4 address, the domain name is constructed according to the rules specified in Section 3.5 of [RFC1035] and it is rooted in the in the special domain "IN-ADDR.ARPA.". For IPv6 addresses, the construction rules in Section 2.5 of [RFC3596] apply and the special domain "IP6.ARPA." is used.

Example values for IPv4 and IPv6 addresses could be (Note: a line break was added in the IPv6 example):

```
R:="3.100.51.198.in-addr.arpa."

R:="0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.B.D.0.
1.0.0.2.ip6.arpa."
```

2.4. Step 2: Add Shortened Domain Names

This task creates a list of several additional domain names, based on the domain name yielded in Step 1.

- For IP version 4, the domain name from Step 1 SHOULD be shortened successively by one and two labels (i.e., purge the first or second dot from the left and everything left of it, respectively), and the results being added to the list. This corresponds to a search on a /24 or /16 network prefix.

- For IP version 6, the domain name from Step 1 SHOULD be shortened successively by 16, 18, 20, and 24 labels, and the results being added to the list. This corresponds to a search on a /64, /56, /48, or /32 network prefix.

This list is intended to provide network operators with a degree of flexibility in where discovery-related resource records can be placed without significantly increasing the number of DNS names that are searched. This does not attach any other significance to these specific zone cuts or create a classful addressing hierarchy based on the reverse DNS tree.

For example, the IPv4 address "192.0.2.75" could result in a list of domain names (with the result from Step 1 put in the first position):

- 75.2.0.192.in-addr.arpa.

- 2.0.192.in-addr.arpa.

- 0.192.in-addr.arpa.

Similarly, the IPv6 address "2001:DB8::28e4:3a93:4429:dfb5" could result in a list:

- 5.b.f.d.9.2.4.4.3.9.a.3.4.e.8.2.0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa.

- 0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa.

- 0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa.

- 0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa.

- 8.b.d.0.1.0.0.2.ip6.arpa.

The limited number of labels by which each name is shortened is intended to limit the maximum number of DNS queries produced by a

single invocation of the cross-domain ALTO server discovery procedure. No more than five U-NAPTR resolutions are invoked for each IP address.

2.5. Step 3: DNS lookups

The list of domain names which was created in the previous step is sequentially (from longest to shortest name) processed, as described in Section 3.2 of RFC 7286 [RFC7286].

3. Using ALTO Cross-Domain Server Discovery with the ALTO Protocol

TBD: expand

3.1. Endpoint Property Service

If an ALTO client wants to query the Endpoint Property Service (see Section 11.4 of RFC 7285 [RFC7285]) for an endpoint with IP address X, it has to invoke the cross-domain ALTO server discovery procedure with parameter X. The result will be the IRD URI of the ALTO server to query.

3.2. Endpoint Cost Service

If an ALTO client wants to query the Endpoint Cost Service (see Section 11.5 of RFC 7285 [RFC7285]) for the costs from source address X to destination address(es) Y (and Z), it has to invoke the cross-domain ALTO server discovery procedure with parameter X. The result will be the IRD URI of the ALTO server to query for the costs from X to Y (and Z)..

3.3. Other ALTO services

TBD. In particular, how to assemble a NxN network map from individual snippets (1xN vectors?) retrieved from different ALTO servers?

4. Implementation, Deployment, and Operational Considerations

4.1. Considerations for ALTO Clients

4.1.1. Resource Consumer Initiated Discovery

To some extent, ALTO requirement AR-32 [RFC6708], i.e., resource consumer initiated ALTO server discovery, can be seen as a special case of cross-domain ALTO server discovery. To that end, an ALTO client embedded in a resouce consumer would have to figure out its own "public" IP address and perform the procedures described in this document on that address. However, due to the widespread deployment of Network Address Translators (NAT), additional protocols and mechanisms such as STUN [RFC5389] would be needed and considerations for UNSAF [RFC3424] apply. Therefore, using the procedures specified in this document for resource consumer based ALTO server discovery is generally NOT RECOMMENDED. Note that a less versatile yet simpler approach for resource consumer initiated ALTO server discovery is specified in [RFC7286].

4.1.2. IPv4/v6 Dual Stack, Multihoming, NAT, and Host Mobility

The algortihm specified in this document can discover ALTO server URIs for a given IP address. The intention is, that a third party (e.g., a resource directory) that receives query messages from a resource consumer can use the source address in these messages to discover suitable ALTO servers for this specific resource consumer.

However, resource consumers (as defined in Section 2 of [RFC5693]) may reside on hosts with more than one IP address, e.g., due to IPv4/v6 dual stack operation and/or multihoming. IP packets sent with different source addresses may be subject to different routing policies and path costs. In some deployment scenarios, it may even be required to ask different sets of ALTO servers for guidance. Furthermore, source addresses in IP packets may be modified en-route by Network Address Translators (NAT).

If a resource consumer queries a resource directory for candidate resource providers, the locally selected (and possibly en-route translated) source address of the query message - as observed by the resource directory - will become the basis for the ALTO server discovery and the subsequent optimization of the resource directory’s reply. If, however, the resource consumer then selects different source addresses to contact returned resource providers, the desired better-than-random "ALTO effect" may not occur.

Therefore, a dual stack or multihomed resource consumer SHOULD either always use the same address for contacting the resource directory and

the resource providers, i.e., overriding the operating system's automatic source IP address selection, or use resource consumer based ALTO server discovery [RFC7286] to discover suitable ALTO servers for every local address and then locally perform ALTO-influenced resource consumer selection and source address selection. Similarly, resource consumers on mobile hosts SHOULD query the resource directory again after a change of IP address, in order to get a list of candidate resource providers that is optimized for the new IP address.

4.2. Deployment Considerations for Network Operators

4.2.1. Separation of Interests

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

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) or redirect the client to another ALTO server using mechanisms of the ALTO protocol (see Sect. 9 of [RFC7285]).

5. Security Considerations

A high-level discussion of security issues related to ALTO is part of the ALTO problem statement [RFC5693]. A classification of unwanted information disclosure risks, as well as specific security-related requirements can be found in the ALTO requirements document [RFC6708].

The remainder of this section focuses on security threats and protection mechanisms for the cross-domain ALTO server discovery procedure as such. Once the ALTO server’s URI has been discovered and the communication between the ALTO client and the ALTO server starts, the security threats and protection mechanisms discussed in the ALTO protocol specification [RFC7285] apply.

5.1. Integrity of the ALTO Server’s URI

Scenario Description

An attacker could compromise the ALTO server discovery procedure or infrastructure in a way that ALTO clients would discover a "wrong" ALTO server URI.

Threat Discussion

This is probably the most serious security concern related to ALTO server discovery. The discovered "wrong" ALTO server might not be able to give guidance to a given ALTO client at all, or it might give suboptimal or forged information. In the latter case, an attacker could try to use ALTO to affect the traffic distribution in the network or the performance of applications (see also Section 15.1. of [RFC7285]). Furthermore, a hostile ALTO server could threaten user privacy (see also Section 5.2.1, case (5a) in [RFC6708]).

However, it should also be noted that, if an attacker was able to compromise the DNS infrastructure used for cross-domain ALTO server discovery, (s)he could also launch significantly more serious other attacks (e.g., redirecting various application protocols).

Protection Strategies and Mechanisms

The cross-domain ALTO server discovery procedure relies on a series of DNS lookups. If an attacker was able to modify or spoof any of the DNS records, the resulting URI could be replaced by a forged URI. The application of DNS security (DNSSEC) [RFC4033] provides a means to limit attacks that rely on modification of the DNS records while in transit. Additional operational precautions for safely operating the DNS infrastructure are required in order to ensure that name servers do not sign forged (or otherwise

"wrong") resource records. Security considerations specific to U-NAPTR are described in more detail in [RFC4848].

A related risk is the impersonation of the ALTO server (i.e., attacks after the correct URI has been discovered). This threat and protection strategies are discussed in Section 15.1 of [RFC7285]. Note that if TLS is used to protect ALTO, the server certificate will contain the host name (CN). Consequently, only the host part of the HTTPS URI will be authenticated, i.e., the result of the ALTO server discovery procedure. The DNS/U-NAPTR based mapping within the cross-domain ALTO server discovery procedure needs to be secured as described above, e.g., by using DNSSEC.

In addition to active protection mechanisms, users and network operators can monitor application performance and network traffic patterns for poor performance or abnormalities. If it turns out that relying on the guidance of a specific ALTO server does not result in better-than-random results, the usage of the ALTO server may be discontinued (see also Section 15.2 of [RFC7285]).

5.2. Availability of the ALTO Server Discovery Procedure

Scenario Description

An attacker could compromise the cross-domain ALTO server discovery procedure or infrastructure in a way that ALTO clients would not be able to discover any ALTO server.

Threat Discussion

If no ALTO server can be discovered (although a suitable one exists) applications have to make their decisions without ALTO guidance. As ALTO could be temporarily unavailable for many reasons, applications must be prepared to do so. However, The resulting application performance and traffic distribution will correspond to a deployment scenario without ALTO.

Protection Strategies and Mechanisms

Operators should follow best current practices to secure their DNS and ALTO (see Section 15.5 of [RFC7285]) servers against Denial-of-Service (DoS) attacks.

5.3. Confidentiality of the ALTO Server's URI

Scenario Description

An unauthorized party could invoke the cross-domain ALTO server discovery procedure, or intercept discovery messages between an authorized ALTO client and the DNS servers, in order to acquire knowledge of the ALTO server URI for a specific IP address.

Threat Discussion

In the ALTO use cases that have been described in the ALTO problem statement [RFC5693] and/or discussed in the ALTO working group, the ALTO server's URI as such has always been considered as public information that does not need protection of confidentiality.

Protection Strategies and Mechanisms

No protection mechanisms for this scenario have been provided, as it has not been identified as a relevant threat. However, if a new use case is identified that requires this kind of protection, the suitability of this ALTO server discovery procedure as well as possible security extensions have to be re-evaluated thoroughly.

5.4. Privacy for ALTO Clients

Scenario Description

An unauthorized party could intercept messages between an ALTO client and the DNS servers, and thereby find out the fact that said ALTO client uses (or at least tries to use) the ALTO service in order to optimize traffic from/to a specific IP address.

Threat Discussion

In the ALTO use cases that have been described in the ALTO problem statement [RFC5693] and/or discussed in the ALTO working group, this scenario has not been identified as a relevant threat.

Protection Strategies and Mechanisms

No protection mechanisms for this scenario have been provided, as it has not been identified as a relevant threat. However, if a new use case is identified that requires this kind of protection, the suitability of this ALTO server discovery procedure as well as possible security extensions have to be re-evaluated thoroughly.

6. IANA Considerations

This document does not require any IANA action.

7. References

7.1. Normative References

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

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

[RFC3596] Thomson, S., Huitema, C., Ksinant, V., and M. Souissi, "DNS Extensions to Support IP Version 6", RFC 3596, October 2003.

[RFC4848] Daigle, L., "Domain-Based Application Service Location Using URIs and the Dynamic Delegation Discovery Service (DDDS)", RFC 4848, April 2007.

7.2. Informative References

[I-D.ietf-alto-deployments]
Stiemerling, M., Kiesel, S., Scharf, M., Seidel, H., and S. Previdi, "ALTO Deployment Considerations", draft-ietf-alto-deployments-15 (work in progress), May 2016.

[I-D.kiesel-alto-3pdisc]
Kiesel, S., Stiemerling, M., Schwan, N., Scharf, M., Tomsu, M., and H. Song, "ALTO Server Discovery Protocol", draft-kiesel-alto-3pdisc-05 (work in progress), March 2011.

[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.kiesel-alto-ip-based-srv-disc]
Kiesel, S. and R. Penno, "Application-Layer Traffic Optimization (ALTO) Anycast Address", draft-kiesel-alto-ip-based-srv-disc-03 (work in progress), July 2014.

[I-D.kist-alto-3pdisc]
Kiesel, S., Krause, K., and M. Stiemerling, "Third-Party ALTO Server Discovery (3pdisc)", draft-kist-alto-3pdisc-05 (work in progress), January 2014.

[RFC3424] Daigle, L. and IAB, "IAB Considerations for UNilateral Self-Address Fixing (UNSAF) Across Network Address Translation", RFC 3424, November 2002.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.

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

[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.

[RFC7216] Thomson, M. and R. Bellis, "Location Information Server (LIS) Discovery Using IP Addresses and Reverse DNS", RFC 7216, April 2014.

[RFC7285] Alimi, R., Penno, R., Yang, Y., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, September 2014.

[RFC7286] Kiesel, S., Stiemerling, M., Schwan, N., Scharf, M., and H. Song, "Application-Layer Traffic Optimization (ALTO) Server Discovery", RFC 7286, June 2014.

Appendix A. Requirements for ALTO Cross-Domain Server Discovery

A solution for the problem described in the previous section would be an ALTO Cross-Domain Server Discovery system. This section itemizes requirements.

A.1. Discovery Client Application Programming Interface

The discovery client will be called through some kind of application programming interface (API) and the parameters will be an IP address and, for purposes of extensibility, a service identifier such as "ALTO". It will return one or more URI(s) that offers the requested service ("ALTO") for the given IP address.

In other words, the client would be used to retrieve a mapping:

(IP address, "ALTO") -> IRD-URI(s)

where IRD-URI(s) is one or more URI(s) of Information Resource Directories (IRD, see Section 9 of [RFC7285]) of ALTO server(s) that can give reasonable guidance to a resource consumer with the indicated IP address.

A.2. Data Storage and Authority Requirements

The information for mapping IP addresses and service parameters to URIs should be stored in a - preferably distributed - database. It must be possible to delegate administration of parts of this database. Usually, the mapping from a specific IP address to an URI is defined by the authority that has administrative control over this IP address, e.g., the ISP in residential access networks or the IT department in enterprise, university, or similar networks.

A.3. Cross-Domain Operations Requirements

The cross-domain server discovery mechanism should be designed in such a way that it works across the public Internet and also in other IP-based networks. This in turn means that such mechanisms cannot rely on protocols that are not widely deployed across the Internet or protocols that require special handling within participating networks. An example is multicast, which is not generally available across the Internet.

The ALTO Cross-Domain Server Discovery protocol must support gradual deployment without a network-wide flag day. If the mechanism needs some kind of well-known "rendezvous point", re-using an existing infrastructure (such as the DNS root servers or the WHOIS database) should be preferred over establishing a new one.

A.4. Protocol Requirements

The protocol must be able to operate across middleboxes, especially across NATs and firewalls.

The protocol shall not require any pre-knowledge from the client other than any information that is known to a regular IP host on the Internet.

A.5. Further Requirements

The ALTO cross domain server discovery cannot assume that the server discovery client and the server discovery responding entity are under the same administrative control.

Appendix B. ALTO and Tracker-based Peer-to-Peer Applications

The ALTO protocol specification [RFC7285] details how an ALTO client can query an ALTO server for guiding information and receive the corresponding replies. However, in the considered scenario of a tracker-based P2P application, there are two fundamentally different possibilities where to place the ALTO client:

1. ALTO client in the resource consumer ("peer")

2. ALTO client in the resource directory ("tracker")

In the following, both scenarios are compared in order to explain the need for ALTO queries on behalf of remote resource consumers.

In the first scenario (see Figure 2), the resource consumer queries the resource directory for the desired resource (F1). The resource directory returns a list of potential resource providers without considering ALTO (F2). It is then the duty of the resource consumer to invoke ALTO (F3/F4), in order to solicit guidance regarding this list.

In the second scenario (see Figure 4), the resource directory has an embedded ALTO client. After receiving a query for a given resource (F1) the resource directory invokes this ALTO client to evaluate all resource providers it knows (F2/F3). Then it returns a, possibly shortened, list containing the "best" resource providers to the resource consumer (F4).

```
    ............................          ............................
    : Tracker                    :          : Peer                       :
    :   ______                   :          :                            :
    : +-______-+                 :          :            k good          :
    : |        |     +--------+ : P2P App. : +--------+ peers +------+ :
    : |   N    |     | random | : Protocol : | ALTO-  |------>| data | :
    : | known  |====>| pre-   |*************>| biased |       | ex-  | :
    : | peers, |     | selec- | : transmit : | peer   |------>| cha- | :
    : | M good |     | tion   | : n peer   : | select | n-k   | nge  | :
    : +-______-+     +--------+ : IDs      : +--------+ bad p.+------+ :
    :............................:          :.....^......................:
                                                  |
                                                  | ALTO
                                                  | client protocol
                                                __|___
                                              +-______-+
                                              |        |
                                              | ALTO   |
                                              | server |
                                              +-______-+
```

Figure 1: Tracker-based P2P Application with random peer preselection

```
Peer w. ALTO cli.            Tracker               ALTO Server
--------+--------       --------+--------       --------+--------
        | F1 Tracker query      |                       |
        |======================>|                       |
        | F2 Tracker reply      |                       |
        |<======================|                       |
        | F3 ALTO client protocol query                 |
        |----------------------------------------------->|
        | F4 ALTO client protocol reply                 |
        |<-----------------------------------------------|
        |                       |                       |

====  Application protocol (i.e., tracker-based P2P app protocol)
----  ALTO client protocol
```

Figure 2: Basic message sequence chart for resource consumer-initiated ALTO query

```
    ............................          ............................
    : Tracker                    :          : Peer                       :
    :   ______                   :          :                            :
    : +-______-+                 :          :                            :
    : |        |     +--------+ : P2P App. :  k good peers &  +------+ :
    : |   N    |     | ALTO-  | : Protocol :  n-k bad peers   | data | :
    : | known  |====>| biased |*****************************>| ex-  | :
    : | peers, |     | peer   | : transmit :                  | cha- | :
    : | M good |     | select | : n peer   :                  | nge  | :
    : +-______-+     +--------+ : IDs      :                  +------+ :
    :....................^.....:          :............................:
                         |
                         | ALTO
                         | client protocol
                       __|___
                     +-______-+
                     |        |
                     | ALTO   |
                     | server |
                     +-______-+
```

Figure 3: Tracker-based P2P Application with ALTO client in tracker

```
          Peer             Tracker w. ALTO cli.       ALTO Server
    --------+--------       --------+--------       --------+--------
            | F1 Tracker query      |                       |
            |======================>|                       |
            |                       | F2 ALTO cli. p. query |
            |                       |---------------------->|
            |                       | F3 ALTO cli. p. reply |
            |                       |<----------------------|
            | F4 Tracker reply      |                       |
            |<======================|                       |
            |                       |                       |

    ====  Application protocol (i.e., tracker-based P2P app protocol)
    ----  ALTO client protocol
```

Figure 4: Basic message sequence chart for ALTO query on behalf of remote resource consumer

Note: the message sequences depicted in Figure 2 and Figure 4 may occur both in the target-aware and the target-independent query mode (c.f. [RFC6708]). In the target-independent query mode no message exchange with the ALTO server might be needed after the tracker query, because the candidate resource providers could be evaluated using a locally cached "map", which has been retrieved from the ALTO

server some time ago.

The problem with the first approach is, that while the resource directory might know thousands of peers taking part in a swarm, the list returned to the resource consumer is usually shortened for efficiency reasons. Therefore, the "best" (in the sense of ALTO) potential resource providers might not be contained in that list anymore, even before ALTO can consider them.

For illustration, consider a simple model of a swarm, in which all peers fall into one of only two categories: assume that there are "good" ("good" in the sense of ALTO's better-than-random peer selection, based on an arbitrary desired rating criterion) and "bad' peers only. Having more different categories makes the maths more complex but does not change anything to the basic outcome of this analysis. Assume that the swarm has a total number of N peers, out of which are M "good" and N-M "bad" peers, which are all known to the tracker. A new peer wants to join the swarm and therefore asks the tracker for a list of peers.

If, according to the first approach, the tracker randomly picks n peers from the N known peers, the result can be described with the hypergeometric distribution. The probability that the tracker reply contains exactly k "good" peers (and n-k "bad" peers) is:

```
            / m \   / N - m \
            \ k /   \ n - k /
P(X=k) =  ---------------------
                  / N \
                  \ n /


        / n \        n!
with    \ k /  = -----------    and   n! = n * (n-1) * (n-2) * .. * 1
                  k! (n-k)!
```

The probability that the reply contains at most k "good" peers is: P(X<=k)=P(X=0)+P(X=1)+..+P(X=k).

For example, consider a swarm with N=10,000 peers known to the tracker, out of which M=100 are "good" peers. If the tracker randomly selects n=100 peers, the formula yields for the reply: P(X=0)=36%, P(X<=4)=99%. That is, with a probability of approx. 36% this list does not contain a single "good" peer, and with 99% probability there are only four or less of the "good" peers on the

list. Processing this list with the guiding ALTO information will ensure that the few favorable peers are ranked to the top of the list; however, the benefit is rather limited as the number of favorable peers in the list is just too small.

Much better traffic optimization could be achieved if the tracker would evaluate all known peers using ALTO, and return a list of 100 peers afterwards. This list would then include a significantly higher fraction of "good" peers. (Note, that if the tracker returned "good" peers only, there might be a risk that the swarm might disconnect and split into several disjunct partitions. However, finding the right mix of ALTO-biased and random peer selection is out of the scope of this document.)

Therefore, from an overall optimization perspective, the second scenario with the ALTO client embedded in the resource directory is advantageous, because it is ensured that the addresses of the "best" resource providers are actually delivered to the resource consumer. An architectural implication of this insight is that the ALTO server discovery procedures must support ALTO queries on behalf of remote resource consumers. That is, as the tracker issues ALTO queries on behalf of the peer which contacted the tracker, the tracker must be able to discover an ALTO server that can give guidance suitable for that respective peer.

Appendix C. Contributors List and Acknowledgments

The initial version of this document was co-authored by Marco Tomsu (Alcatel-Lucent).

This document borrows some text from [RFC7286], as it was historically part of that memo. Special thanks to Michael Scharf and Nico Schwan.

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/

Martin Stiemerling
University of Applied Sciences Darmstadt, Computer Science Dept.
Haardtring 100
Darmstadt 64295
Germany

Phone: +49 6151 16 7938
Email: mls.ietf@gmail.com
URI: http://ietf.stiemerling.org

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 7, 2016

S. Randriamasy
Alcatel-Lucent Bell Labs
R. Yang
Yale University
Q. Wu
Huawei
L. Deng
China Mobile
N. Schwan
Thales Deutschland
July 6, 2015

ALTO Cost Calendar
draft-randriamasy-alto-cost-calendar-04

Abstract

The goal of Application-Layer Traffic Optimization (ALTO) is to bridge the gap between network and applications by provisioning network related information in order to allow applications to make informed decisions. The present draft extends the ALTO cost information so as to broaden the decision possibilities of applications to not only decide 'where' to connect to, but also 'when'. This is useful to applications that need to schedule their data transfers and connections and have a degree of freedom to do so. ALTO guidance to schedule application traffic can also efficiently help for load balancing and resources efficiency. Besides, the ALTO Cost Calendar also allows to schedule the ALTO requests themselves and thus save a number of ALTO transactions.

The draft proposes new capabilities and attributes on filtered cost maps and endpoint costs enabling an ALTO Server to provide "Cost Calendars". These capabilities are applicable to time-sensitive ALTO metrics With ALTO Cost Calendars, an ALTO Server exposes ALTO Cost Values in JSON arrays where each value corresponds to a given time interval. The time intervals as well as other Calendar attributes are specified in the IRD and ALTO Server responses.

Requirements Language

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

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 [RFC7285] specifies a Network Map, which defines groupings of endpoints in a network region (called a PID) as seen by the ALTO server. The Cost Maps Service, Endpoint Cost

Service (ECS) and Endpoint Ranking Service then provide rankings for connections between the specified endpoints and 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 as explained in the ALTO Problem Statement [RFC5693] and ALTO Requirements [RFC5693]) .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 Endpoints. 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 their own estimation of frequency of cost value change.

With the base protocol, an ALTO client should interpret the returned costs as those at the query moment. However, Network costs can fluctuate, e.g. due to diurnal patterns of traffic demand or planned events such as network maintenance or holidays or highly publicized events. Providing network costs for only the current time thus may not be sufficient, in particular, for applications that can schedule their traffic in a span of time, for example, by deferring backup to night during traffic trough. Besides, other applications would like to anticipate their connections and transfers to favorable times.

In case these value changes are predicable over a certain period of time and the application does not require immediate data transfer, it can save time to get the whole set of cost values over this period in one ALTO response and using these values to schedule data transfers would allow to optimise the network resources usage and QoE. Morerover a Client can minimize its requests for calendars by scheduling them at appropriate times.

This document extends RFC7285 to allow an ALTO server to provide network costs for a given duration of time. A sequence of network costs across a time span for a given pair of network locations is referred to as an ALTO Cost Calendar. The Filtered Cost Map Service and Endpoint Cost Service are extended to provide Cost Calendars. In addition to this functional ALTO enhancement, we expect to further gain on storage and on the wire data exchange by gathering multiple Cost Values for one Cost Type into one single ALTO Server response.

In this draft an "ALTO Cost Calendar" is specified by information resources capabilities that are applicable to time-sensitive ALTO metrics. An ALTO Cost Calendar exposes ALTO Cost Values in JSON arrays where each value corresponds to a given time interval. The time intervals as well as other Calendar attributes are specified in the IRD and in the Server response and allow the ALTO Client to interpret the received ALTO values. This draft proposes a set of

Calendar attributes to be added to the resources capabilities in the IRD. Last, the proposed extensions for ALTO Calendars are applicable to any Cost Mode and they ensure backwards compatibility with legacy clients.

The rest of this document is organized as follows. Section 2 provides the design characteristics. Section 3 and 4 define the formal specification for the IRD and the information resources. Section 5 provides non-normative use cases to illustrate the usage of cost calendars. IANA considerations and security considerations will be completed in further versions.

2. Overview of ALTO Cost Calendars

An ALTO Cost calendar provided by the ALTO Server provides 2 information items:

- o an array of values for a given metric, where each value corresponds to a time interval, wher the value array can sometimes be a cyclic pattern that repeats a certain number of times.

- o attributes describing the time scope of the calendar, allowing an ALTO Client to properly interpret the values, such as the size and number of the intervals and the date of the starting point of the calendar.

An ALTO Cost Calendar can be used like a "time table" to figure out the best time to schedule data transfers and also anticipate predictable events such as flash crowds, traffic intensive holidays and network maintenance. An ALTO Cost Calendar may be viewed as a synthetic abstraction of real measurements that can be historic or be a prediction for upcoming time periods.

Most likely, the ALTO Cost Calendar would be used for the Endpoint Cost Service, assuming that a limited set of feasible Endpoints for a non-real time application is already identified, that they do not need to be accessed immediately and that 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 allows it.

2.1. ALTO Cost Calendar information features

The Calendar attributes are provided in the IRD and in ALTO Server responses. The IRD announces attributes with dateless values in its information resources capabilities, where as attributes with time dependent values are provided in the "meta" of Server responses. The ALTO Cost Calendar attributes provide the following information:

- o attributes to report on Calendar value array:
  - * generic time zone,
  - * applicable time interval for each calendar value: combining numbers and time units to reflect for example: 1 hour, 2 minutes, 10 seconds, 1 week, 1 month,
  - * duration of the Calendar: e.g. the number of intervals provided in the calendar.
- o attributes on time stamps for Calendars: the ALTO Servers chooses and specifies the date at which it starts its calendars, either at the client request date or periodically:
  - * "calendar-start-date": specifying when the calendar starts, that is to which date the first value of the cost calendar is applicable.
- o possible periodicity of the calendar start date: allowing to predict when the next ALTO Calendar should be requested if needed,
  - * "repeat-indication": may be provided when the "calendar-start-date" is periodic, to indicate whether the Server will provide the number of repetitions.
  - * "repeated": is an attribute indicating for how many iterations the provided calendar will have the same values, to allow the client to schedule its next request.

2.2. ALTO Calendar design characteristics

This draft introduces new capabilities and attributes that specify an ALTO Cost Calendar. The protocol extension placeholders are: the IRD, the ALTO requests and responses for Cost calendars.

Extensions are designed to be light and ensure backwards compatibility with base protocol ALTO Clients and with other extensions. It uses section 8.3.7 "Parsing of Unknown Fields" of RFC7285 that writes: "Extensions may include additional fields within JSON objects defined in this document. ALTO implementations MUST ignore unknown fields when processing ALTO messages."

The calendar specific capabilities are integrated in the information resources of the IRD and in the "meta" member of ALTO responses to Cost Calendars requests. A calendar and its capabilities are associated to a given information resource and within this

information resource to a given cost type. This design has several advantages:

- o it does not introduce a new mode,
- o it does not introduce new media types,
- o it allows an ALTO Server to offer calendar capabilities on a cost type, with attributes values adapted to each information resource.

The Applicable Calendared information resources are:

- o the Filtered Cost Map,
- o the Endpoint Cost Map.

The ALTO Server can choose in which frequency it provides cost Calendars to ALTO Clients. It may either provide calendar updates starting at the request date, or carefully schedule its updates so as to take profit from a potential repetition/periodicity of calendar values.

2.2.1. ALTO Cost Calendar for all cost modes

Calendars are well-suited for values encoded in the 'numerical' mode. However, Calendars can also represent any metric considered as time-sensitive by an ALTO Server. For example, types of Cost values such as JSONBool can also be expressed as calendars, as states may be "true" or "false" depending on given time periods or likewise, values represented by strings, such as "medium", "high", "low", "blue", "open" .

Note also that a Calendar is applicable as well to time-sensitive metrics provided in the 'ordinal' mode, if these values are time-sentitive and their update is carefully managed by the ALTO Server.

2.2.2. Compatibility with legacy ALTO Clients

The ALTO protocol extensions for Cost Calendars have been defined so as to ensure that Calendar capable ALTO Servers can provide legacy ALTO Clients with legacy information resources as well. That is a legacy ALTO Client can request resources and receive responses as specified in RFC7285.

For compatibility with legacy ALTO Clients specified in RFC7285, calendared information resources are not applicable for Cost Maps for the following reason: a legacy ALTO client would receive a Calendared Cost Map via an HTTP 'GET' command. As specified in section 8.3.7 of

RFC7285, it will ignore the Calendar Attributes indicated in the "meta" of the responses. Therefore, lacking information on calendar attributes, it will not be able to correctly interpret and process the values of the received array of calendar cost values.

3. ALTO Calendar specification: IRD extensions

The Calendar attributes in the IRD information resources capabilities carry constant dateless values. A calendars is associated to an information resource rather than a cost type. For example, a Server can provide a "routingcost" values calendar for the Filtered Cost Map Service at a granularity of one day and a "routingcost" values calendar for the Endpoint Cost service at a finer granularity but for a limited number of endpoints.

3.1. Calendar attributes in the IRD resources capabilities

When for an applicable resource , an ALTO Server provides a Cost Calendar for a given Cost Type, it MUST indicate this in the IRD capabilities of this resource, by an object of type ’CalendarAttributes’, associated to this Cost Type and specified below.

The capabilities of a Calendar aware information resource entry have a member named "calendar-attributes" which is an array of objects of type CalendarAttributes. The array has as many values as cost-type-names announced for the resource. It is necessary to use an array because of resources such as Filtered Cost Map and Endpoint Cost Map, for which the member "cost-type-names" is an array of 1 or more values. If for a given cost-type-name of this ressource no Calendar attributes are defined, the ALTO Server MUST replace that value in the array by the symbol ’null’.

RULE: a member "calendar-attributes" MUST appear only once for each applicable cost type name of a resource entry. If "calendar-attributes" are specified several times for a same "cost-type-name" in the capabilities of a resource entry, the ALTO client SHOULD ignore any calendar capabilities on this "cost-type-name" for this entry.

```
CalendarAttributes calendar-attributes <1..*>;

object{
  [JSONString   cost-type-name;]
  JSONString    calendar-start-mode;
  JSONString    time-interval-size;
  JSONNumber    number-of-intervals;
  [JSONBoolean   repeat-indication;]
} CalendarAttributes;
```

o "cost-type-name":

  * an optional member indicating the cost-type-name in the IRD entry to which the capabilities apply. If this not present, it MUST be assumed to correspond to its index in the "cost-type-names" list of the IRD entry.

o "calendar-start-mode":

  * takes values in {"request date", "periodic"} . Indicates whether the ALTO Server provides this Calendar with values starting at the date of the client request or at periodical dates.

o "time-interval-size":

  * is the duration of an ALTO calendar time interval, expressed as a time unit appended to the number of these units. The time unit, ranges from "second" to "year". The number is encoded with an integer. Example values are: "5 minute" , "2 hour", meaning that each calendar value applies on a time interval that lasts respectively 5 minutes and 2 hours.

o "number-of-intervals":

  * the integer number of values of the cost calendar array, at least equal to 1.

o "repeat-indication":

  * a boolean value that indicates whether or not the ALTO Server indicates how many times the provided calendar values repeat. If this member is not present, it MUST be assumed to have a value equal to "false".

- Attribute "cost-type-name" , if used, provides a better readability to the calendar attributes specified in the IRD and avoids confusion with calendar attributes of other cost-types.

- Multiplying Attributes ’time-interval-size’ and ’number-of-intervals’ provides the duration of the provided calendar. For example an ALTO Server may provide a calendar for ALTO values changing every ’time-interval-size’ equal to 5 minutes. If ’number-of-intervals’ has the value 12, then the duration of the provided calendar is "1 hour".

- Attribute "repeat-indication": indicates whether or not the ALTO Server informs the client if the provided ALTO Calendar is a cyclic value pattern that will be repeated for a number of times.

NOTE : to cope with existing representation fomats and proposed unified ALTO naming schemes proposed in the WG, the names given in the current proposal may be revised in further versions.

3.2. Calendars in a delegate IRD

An option to clarify IRD resources is that a "root" ALTO Server implementing base protocol resources delegates "specialized" information resources such as the ones providing Cost Calendars to another ALTO Server running in a subdomain specified with its URI in the "root" ALTO Server. This option is described in Section 9.2.4 "Delegation using IRDs" of RFC7285.

This document provides an example, where a "root" ALTO Server runs in a domain called "alto.example.com". It delegates the announcement of Calendars capabilities to an ALTO Server running in a subdomain called "custom.alto.example.com". The location of the "delegate Calendar IRD" is assumed to be indicated in the "root" IRD by the resource entry: "custom-calendared-resources".

Another advantage is that some Cost Types for some resources may be more advantageous as Cost Calendars and it makes few sense to get them as a single value. For example, Cost Types with predicatble and frequently changing values, calendared in short time intervals such as a minute.

3.3. Example IRD with ALTO Cost Calendars

The cost types in this example are either specified in the base ALTO protocol or proposed in other drafts see [draft-wu-alto-te-metrics]. In this example the available cost metrics are indicated in the "meta" field by cost type names "num-routingcost", "num-AShopcount", ’num-TEpktloss", ’num-pathbandwidth’ and "string-quality-status". Metrics "routingcost" , "hopcount" , ’TEpktloss’ and ’Availbandwidth’ are available in the "numerical" Cost Mode. Metric "quality-status" is available in the "string" Cost Mode.

This ALTO server does not provide a calendar for cost type name num-AShopcount.

The example IRD includes 2 particular URIs providing calendars:

- "http://custom.alto.example.com/calendar/costmap/filtered": a filtered cost map in which calendar capabilities are indicated for cost type names: "num-routingcost", "num-pathbandwidth" and "string-service-status",

- "http://custom.alto.example.com/endpointcost/calendar/lookup": an endpoint cost map in which in which calendar capabilities are indicated for cost type names: "num-routingcost", "num-TEpktloss", "num-pathbandwidth", "string-service-status".

The design of the Calendar capabilities allows that some calendars on a cost type name are available in several information resources with different Calendar Attributes. This is the case for calendars on "num-routingcost", "num-pathbandwidth" and "string-service-status" , available in both the Filtered Cost map and Endpoint Cost map service, as detailed afterwards.

```
GET /calendars-directory HTTP/1.1
Host: custom.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

   {
     "meta" : {
        "cost-types": {
           "num-routingcost": {
              "cost-mode" : "numerical",
              "cost-metric" : "routingcost"
           },
           "num-AShopcount": {
              "cost-mode" : "numerical",
              "cost-metric" : "hopcount"
           },
           "num-TEpktloss": {
              "cost-mode"  : "numerical",
              "cost-metric": "TEpktloss"
           },
           "num-pathbandwidth": {
              "cost-mode"  : "numerical",
```

```
              "cost-metric": "Availbandwidth",
           },
           "string-qual-status": {
              "cost-mode"  : "string",
              "cost-metric": "quality-status",
           }
         ... other meta ...
       },

    "resources" : {
       "filtered-cost-map-calendar" : {
           "uri" : "http://custom.alto.example.com/calendar/costmap/filtered",
           "media-type" : "application/alto-costmap+json",
           "accepts" : "application/alto-costmapfilter+json",
           "capabilities" : {
              "cost-constraints" : true,
              "cost-type-names"  : [ "num-routingcost", "num-pathbandwidth",
                                     "string-service-status", "num-AShopcount" ]
,
              "calendar-attributes" : [
                 {"cost-type-names" : "num-routingcost",
                  "calendar-start-mode" : "request-date",
                  "time-interval-size" : "1 hour",
                  "number-of-intervals" : 24
                 },
                 {"cost-type-names" : "num-pathbandwidth",
                  "calendar-start-mode" : "request-date",
                  "time-interval-size" : "1 hour",
                  "number-of-intervals" : 24
                 },
                 {"cost-type-names" : "string-service-status",
                  "calendar-start-mode" : "request-date",
                  "time-interval-size" : "30 minute",
                  "number-of-intervals" : 48
                 },
                 null
                 ]
              "uses": [ "my-default-network-map" ]
              } // FCM capab
           },

       "endpoint-cost-calendar-map" : {
         "uri" : "http://custom.alto.example.com/calendar/endpointcost/calendar/
lookup",
         "media-types" : [ "application/alto-endpointcost+json" ],
         "accepts" : [ "application/alto-endpointcostparams+json" ],
         "capabilities" : {
           "cost-constraints" : true,
           "cost-type-names" : [ "num-AShopcount", "num-routingcost",
                                 "num-TEpktloss", "num-pathbandwidth",
```

```
                                "string-service-status"  ],
           "calendar-attributes" : [
              null,
              {"cost-type-names" : "num-routingcost",
               "calendar-start-mode" : "periodic",
               "time-interval-size" : "1 hour",
               "number-of-intervals" : 24,
               "repeat-indication" : true
              },
              {"cost-type-names" : "latency",
               "calendar-start-mode" : "periodic",
               "time-interval-size" : "5 minute",
               "number-of-intervals" : 12,
               "repeat-indication" : true
              },
              {"cost-type-names" : "num-pathbandwidth",
               "calendar-start-mode" : "periodic",
               "time-interval-size" : "1 minute",
               "number-of-intervals" : 60,
               "repeat-indication" : true
              },
              {"cost-type-names" : "string-service-status",
               "calendar-start-mode" : "periodic",
               "time-interval-size" : "2 minute",
               "number-of-intervals" : 30,
               "repeat-indication" : true
              },
             ]
           "uses": [ "my-default-network-map" ]
         } // ECM capab
      } //info resource N
   } // ressources
```

In this example IRD for the filtered cost map service, all calendars have a duration of 1 day and start in the "request-date" mode, that is the "date" of first value of the array belongs to the time interval "containing" the date of the request.

- o the Calendar for ’num-routingcost’: is an array of 24 values each provided on a time interval lasting 1 hour.

- o the Calendar for ’num-pathbandwidth’: is an array of 24 values each provided on a time interval lasting 1 hour.

- o the Calendar for "string-service-status": "is an array of 48 values each provided on a time interval lasting 30 minutes.

For the endpoint cost map service, the cost calendars have a duration of 1 day for "num-routingcost" and 1 hour for the 3 other cost type names. They start in the "periodic" mode, that is the "date" of first value of the array is chosen by the ALTO Server, that is in this case assumed to update the calendars periodically. The value of member "repeat-indication" is set to ’true’, which means that the ALTO Server informs the ALTO Client if the values of the current calendar will be the same in the next periods and for how many periods.

- o the Calendar for ’num-routingcost’: is an array of 24 values each provided on a time interval lasting 1 hour.

- o the Calendar for ’TEpktloss’’: is an array of 12 values each provided on a time interval lasting 5 minutes.

- o the Calendar for ’num-pathbandwidth’: is an array of 60 values each provided on a time interval lasting 1 minute.

- o the Calendar for "string-service-status": "is an array of 30 values each provided on a time interval lasting 2 minutes.

4. ALTO Calendar specification: Service Information Resources

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

The reference time zone for the provided time values is GMT because the option chosen to express the time format is the HTTP header fields format:

Date: Tue, 15 Nov 1994 08:12:31 GMT

Note that if the ’calendar-start-time’ date is past, the application can also use the information to compute statistics on values provided by ALTO over time to guide applications. Besides estimating some customized prediction the ALTO Client may use these values to assess their reliability w.r.t. some real measures of QoE.

4.1. Calendar extensions for Filtered Cost Maps

A legacy ALTO client requests and gets filtered cost map responses as specified in RFC7285.

4.1.1. Calendar extensions in Filtered cost map requests

The input parameters of a "legacy" request for a filtered cost map, defined by object ReqFilteredCostMap in section 11.3.2 of RFC7285, are augmented with one additional member.

A Calendar aware ALTO client requesting a Calendar on a given Cost Type for a Filtered Cost Map resource having Calendar capabilities MUST add the following field to its input parameters:

JSONBoolean calendared<1..*>;

This field is an array of 1 to several boolean values indicating whether or not the ALTO Server should provide the values for this Cost Type as a calendar.

This field MUST NOT be specified if the calendar capability is not present or equal to false for this information resource.

A Calendar-aware ALTO client supporting single cost type values, as specified in RFC7285, MUST provide an array of 1 element: "calendared" : [true],

A Calendar-aware ALTO client that is also Multi-Cost aware MUST provide an array of N values set to "true" or "false", depending whether it wants the applicable Cost Type values as a single or calendared value.

If this field is not present, it MUST be assumed to have only values equal to "false".

4.1.2. Calendar extensions in Filtered Cost map responses

The calendared costs are JSONArrays instead of JSONNumbers for the legacy ALTO implementation. All arrays have a number of values equal to ’number-of-intervals’.

The "meta" field of a Calendared Filtered Cost map response MUST include at least:

- o the "meta" fields specified for these information service responses, as specified in RFC 7285 if the ALTO Client supports costs for one Cost Type at a time only,
  - * "dependent-vtags ",
  - * "cost-type" field.

- o the "meta" fields specified for these information service responses, as specified in RRRR [draft-ietf-multi-cost-alto] if the ALTO Client supports Multi-Cost capabilities, that is:

  - * "dependent-vtags ",

  - * "multi-cost-types" field.

The "meta" field of a Calendared Filtered Cost map response MUST include in addition the member "calendar-response-attributes" for the requested information resource, together with the values provided by the ALTO Server for these attributes. This member is an array of objects of type "CalendarResponseAttributes", defined as follows:

```
CalendarResponseAttributes calendar-response-attributes <1..*>;

object{
  JSONString    calendar-start-time;
  JSONString    time-interval-size;
  JSONNumber    number-of-intervals;
  [JSONNumber   repeated;]            [OPTIONAL]
} CalendarResponseAttributes;
```

- o "calendar-start-time": indicates the date at which the first value of the calendar applies. The value provided for the "calendar-start-time" attribute MUST be no later than the request date.

- o "time-interval-size": as specified in section "Calendar attributes in the IRD resources capabilities",

- o "number-of-intervals": as specified in section "Calendar attributes in the IRD resources capabilities",

- o "repeated": is an optional field provided for Calendars available in the "periodic" ’calendar-start-mode’. It is an integer greater or equal to ’0’ that indicates how many times the calendar starting at the date indicated by "calendar-start-time" will have the same value array. An ALTO Server indicating a "calendar-start-mode" set to "periodic" for this Calendar SHOULD provide a value for this member. If omitted this member MUST be interpreted as having a value equal to ’0’.

For example:

- o if a calendar has a "calendar-start-time" member with value "Mon, 30 Jun 2014 at 00:00:00 GMT" and if the calendar values are the same from Monday through Thursday included, then the value of member "repeated" will be equal to 4. The ALTO Client thus may

use the same calendar for the 4 duration periods following "calendar-start-time".

- o If the calendars for Friday, Saturday and Sunday all have different values, the value of their member "repeated" will be equal to 1.

- o If in a next week, the values are identical for Monday, Tuesday, Thursday and different for Wednesday (holiday, world wide event), the calendar update provided on Monday will have a member "repeated" with value 2.

4.1.2.1. Calendar Start time value for "request-date" calendar start mode

When the ALTO Server IRD announces a Calendar for which attribute "calendar-start-mode" is set to "request-date", the value provided in the ALTO response for attribute "calendar-start-time" MUST correspond to the start of the time interval "including" the date of the request. Figure FFFF1 uses the example IRD of section 3.3: in resource "filtered-cost-calendar-map" a calendar is offered for cost-type-name "num- pathbandwidth", as an array on 24 slots of 1 hour and is available in the calendar-start-mode "request".

```
Ts-----------Ts---X--------Ts-----------Ts-----------
                  |
11:00        12:00|        13:00        14:00
                  |
                  Request on Tue, 1 Jul 2014 at 12:15:00
```

The ALTO Client knows from IRD that:
- the "time-interval-size" Ts is equal to 1 hour
- the "calendar-start-time" is the beginning of the time interval to which the "request-date" belongs

So the ALTO Client can expect that the "calendar-start-time" of the requested calendar is: Tue, 1 Jul 2014 at 12:00:00

Figure FFFF1: "calendar-start-time" when the IRD indicates a "request date" set to "calendar-start-mode".

4.1.2.2. Calendar Start time value for "periodic" calendar start mode

When the ALTO Server IRD offers a Calendar for which attribute ’calendar-start-mode’ is set to "periodic", it means that it chooses to update the values at periodic dates, for example every week or

hour or day. In this case, the value in the server response for attribute "calendar-start-time" MUST be the one of the last Calendar update present in the ALTO Server.

Figure FFFF2 uses the example IRD of section 3.3: in resource "endpoint-cost-calendar-map", a calendar is offered for cost- type-name "num- routingcost" as an array of array of 24 slots of 1 hour and is available in the "periodic" calendar-start-mode. The ALTO Client can expect that the value of "calendar-start-time" is aged between 0 and 23 hours plus 0 to N days. Suppose the calendar starts everyday at 00.00.00. GMT and that its values are the same from Monday through Thursday included. If the request is done at date: Tue, 1 Jul 2014 at 12:15:00 GMT, the "calendar- start-time" field takes value "Mon, 30 Jun 2014 at 00:00:00 GMT".

```
<---- D ----><---- D ----><---- D ----><---- D ---->

C1-----------C2------X----C3-----------C4-----------
                     |
Mon          Tue     |    Wed          Thu
                     |
                     Request on Tue, 1 Jul 2014 at 12:15:00
```

The ALTO Client knows from IRD that:
- the "calendar-start-time" is "periodic"
- the "time-interval-size" Ts is equal to 1 hour
- each period has a duration D of 24 Ts
- Calendars Ci may have the same values for 1 or more periods D

So the ALTO Client can expect that the "calendar-start-time" of the requested calendar is aged n*D = k*Ts, where k lies in [0,23] and n >= 0

Figure FFFF2: estimation by the ALTO Client of the "calendar-start-time" when the IRD indicates a "periodic" "calendar-start-mode".

4.1.3. Example transaction for a FCM with a "request-date" bandwidth Calendar

An example of non-real time information that can be provisioned in a ’calendar’ 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 for example to avoid traffic peaks due to diurnal usage patterns. In this example, we assume that an ALTO Client requests a bandwidth calendar as specified in the IRD to shedule its bulk data transfers as described in the use cases.

In the example IRD, calendars for cost type name "num-pathbandwidth" are available for the information resources: "filtered-cost-calendar-map" starting at the request date and "endpoint-cost-calendar-map" starting in a periodic date. The ALTO Client requests a calendar for "num-pathbandwidth" via a POST request for a filtered cost map.

We suppose in this example that the ALTO Client sends its request on Tuesday July 1st 2014 at 13:15

```
POST /calendar/costmap/filtered HTTP/1.1
  Host: alto.example.com
  Content-Length: [TODO]
  Content-Type: application/alto-costmapfilter+json
  Accept: application/alto-costmap+json,application/alto-error+json

  {
    "cost-type" : {"cost-mode" : "calendar", "cost-metric" : "Availbandwidth"},
    "calendared" : [true],

    "pids" : {
       "srcs" : [ "PID1", "PID2" ],
       "dsts" : [ "PID1", "PID2", "PID3" ]
    }
  }


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

  {
    "meta" : {
      "dependent-vtags" : [...],
      "cost-type" : {"cost-mode" : "numerical", "cost-metric" : "Availbandwidth"
},
      "calendar-response-attributes" : [
         "calendar-start-time" : Tue, 1 Jul 2014 13:00:00 GMT,
         "time-interval-size" : "2 hour",
         "numb-intervals" : 12
       ]
     },

    "cost-map" : {
       "PID1": { "PID1": [v1,v2,v3,v4,v5,v6,v7,v8,v9,v10,v11,v12],
                 "PID2": [v1,v2,v3,v4,v5,v6,v7,v8,v9,v10,v11,v12],
                 "PID3": [v1,v2,v3,v4,v5,v6,v7,v8,v9,v10,v11,v12] },
       "PID2": { "PID1": [v1,v2,v3,v4,v5,v6,v7,v8,v9,v10,v11,v12],
                 "PID2": [v1,v2,v3,v4,v5,v6,v7,v8,v9,v10,v11,v12],
                 "PID3": [v1,v2,v3,v4,v5,v6,v7,v8,v9,v10,v11,v12] }
    }
  }
```

4.2. Calendar extensions in the Endpoint Cost Map Service

This document extends the Endpoint Cost Service, as defined in {11.5.1} of [RFC7285], by adding new input parameters and capabilities, and by returning JSONArrays instead of JSONNumbers as the cost values. The media type (11.5.1.1} and HTTP method (11.5.1.2} are unchanged.

4.2.1. Calendar specific input in Endpoint cost map requests

The extensions to the requests for calendared Endpoint Cost Maps are the same as for the Filtered Cost Map Service, previously specified in section XXXX.

The ReqEndpointCostMap object for a Calendared ECM request will have the following format:

```
object {
  CostType       cost-type;
  [JSONBoolean    calendared<1..*>;]
  EndpointFilter endpoints;
} ReqEndpointCostMap;

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

4.2.2. Calendar attributes in the Endpoint Cost Map responses

The "meta" field of a Calendared Endpoint Cost map response MUST include at least:

- o the "meta" fields specified for these Endpoint Cost service responses, as specified in RFC 7285 if the ALTO Client supports costs for one Cost Type at a time only,

  - * "cost-type" field.

- o the "meta" fields specified for these information service responses, as specified in RRRR [draft-ietf-multi-cost-alto] if the ALTO Client supports Multi-Cost capabilities, that is:

  - * "multi-cost-types" field.

The "meta" member of a Calendared Endpoint Cost Map response MUST include the same addifional member "calendar-response-attributes" as

specified for the Filtered Cost Map Service. If the client request does not provide member "calendared" or if it provides it with a value equal to ’false’, then the ALTO Server response is exactly as specified in {11.5.1.6} of [RFC7285]. If the client provides member "calendared" with a value equal to ’true’ in the input parameters, the Server response is changed as follows:

- o the "meta" member has one additional field "CalendarResponseAttributes", as specified for the Filtered Cost Map Service,

- o the calendared costs are JSONArrays instead of JSONNumbers for the legacy ALTO implementation. All arrays have a number of values equal to ’number-of-intervals’.

4.2.3. Example transaction for the ECS with a "periodic" routingcost Calendar

Let us assume an Application Client located in an end sytem 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 to be the time sentitive decision metric, with values provided in the ALTO Calendar Mode.

The ALTO Client embedded in the Application Client queries an ALTO Calendar on ’routingcost’ and will get the Calendar covering the 24 hours time period "containing" the date and time of the ALTO client request.

For Cost Type ’calendar-routing’, this example ALTO Server has defined 3 different daily patterns each represented by a Calendar, to cover the week of Monday June 30th at 00:00 to Sunday July 6th 23:59:

- C1 for Monday, Tuesday, Wednesday, Thursday, (week days)

- C2 for Saturday, Sunday, (week end)

- C3 for Friday (maintenance outage on July 4, 2014 from 02:00:00 GMT to 04:00:00 GMT, or big holiday such as New Year evening).

The presence of attributes "repeated" and "calendar-start-time" allows an ALTO client to fetch 3 Calendars instead of 7 and thus to reduce the volume of on-the-wire data exchange,

In the following example transaction, the ALTO Client sends its request on Tuesday July 1st 2014 at 13:15. It will this get the value pattern "C1" , valid from Monday at 00:00:00 until Thursday at 23:59:59, thus repeated 4 times.

```
POST /calendar/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" : {"cost-mode" : "numerical", "cost-metric" : "routingcost"},
    "calendared" : [true],
    "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" : {
      "cost-type" : {"cost-mode" : "numerical", "cost-metric" : "routingcost"},
      "calendar-response-attributes" : [
         { "calendar-start-time" : Mon, 30 Jun 2014 00:00:00 GMT,
           "time-interval-size" : "1 hour",
           "numb-intervals" : 24,
           "repeated": 4 }
        ],
      } // end meta

      "endpoint-cost-map" : {
        "ipv4:192.0.2.2": {
          "ipv4:192.0.2.89"    : [v1, v2, ... v24],
          "ipv4:198.51.100.34" : [v1, v2, ... v24],
          "ipv4:203.0.113.45"  : [v1, v2, ... v24]
        }
      }
  }
```

4.2.4. Example transaction for the ECS with a calendar on both routingcost and latency

In this example, it is assumed that the ALTO Server implements multi-cost capabilities, as specified in [draft-ietf-alto-multi-cost] . That is, an ALTO client can request and receive values for several cost types in one single transaction. An illustrating use case is a path selection done on the basis of 2 metrics: routing cost and latency.

The assumptions on the routing cost calendars are the same as in the previous example.

For metric "latency", the calendar attributes in the IRD indicate that the values are updated every hour and each one applies to an interval of 5 minutes. The Server response provides value 1 for attribute ’repeated’ it is assumed that the current calendar values are repeated 1 time, that is, in the next hour the calendar will have different values.

In the following example transaction, the ALTO Client sends its request on Tuesday July 1st 2014 at 13:15. It is assumed that it does this multi-cost request for the first time.

When receiving the response, the client sees that the calendar for latency will change in the next hour where as the calendar values for routing cost will be the same for the next 3 days. Therefore, in its next requests until the routing cost calendar is expected to change, the client will only need to request a calendar for 1 single metric which is latency.

Without the ALTO Calendar extensions, the ALTO client would have no clue on the dynamicity of the metric value change and would spend needless time requesting values at an inappropriate pace. In addition, without the Multi-Cost ALTO capabilities, the ALTO client would duplicate this waste of time as it would need to send one request per cost metric.

```
POST calendar/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

  {
    "multi-cost-types" : [
        {"cost-mode" : "numerical", "cost-metric" : "routingcost"},
        {"cost-mode" : "numerical", "cost-metric" : "latency"}
```

```
    ],
    "calendared" : [true, true],
    "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" : {
      "multi-cost-types" : [
        {"cost-mode" : "numerical", "cost-metric" : "routingcost"},
        {"cost-mode" : "numerical", "cost-metric" : "latency"}
       ],
      "calendar-response-attributes" : [
         { "cost-type-name : num-routingcost"
           "calendar-start-time" : Mon, 30 Jun 2014 00:00:00 GMT,
           "time-interval-size" : "1 hour",
           "numb-intervals" : 24,
           "repeated": 4 },
         { "cost-type-name : num-latency"
           "calendar-start-time" : Tue, 1 Jul 2014 13:00:00 GMT,
           "time-interval-size" : "5 minute",
           "numb-intervals" : 12,
           "repeated": 1 }
       ],
      } // end meta

      "endpoint-cost-map" : {
        "ipv4:192.0.2.2": {
          "ipv4:192.0.2.89"    : [[r1, r2, ... r24], [l1, l2, ... l12]],
          "ipv4:198.51.100.34" : [[r1, r2, ... r24], [l1, l2, ... l12]],
          "ipv4:203.0.113.45"  : [[r1, r2, ... r24], [l1, l2, ... l12]]
        }
      }
  }
```

4.3. Recap of rules related to ALTO Cost Calendars

XXXXX TO BE COMPLETED + MOVED AT THE END OF THE SPECS

A Calendar-aware ALTO Server MUST implement the base protocol specified in RFC7285.

If no Calendar attributes are defined for a given Cost Type, in a given resource entry, the ALTO Server MUST set the value in the 'calendar-attributes' array to the symbol 'null' .

When a metric is available as a calendar, it MUST be available as a single value. An ALTO Server aquiring cost values in limited time intervals only can construct a single value from the value array.

Calendared information resources MUST be requested via a POST method.

If this member "repeat-indication" is not present in the calendar attributes indicated in the IRD, it MUST be assumed to have a value equal to "false".

5. Use cases for ALTO Cost Schedule

This section presents use cases showing the benefits of ALTO Cost calendars for applications needing to decide both "where" to connect and "when".

5.1. Bulk Data Transfer scheduling upon bandwidth calendars

Large Internet Content Providers (ICPs) like Facebook or YouTube, as well as CDNs rely on data replication across multiple sites and time zones to offload the core site and increase user experience through shorter latency from a local site. Typically the usage pattern of these data centers or caches follows a location dependent diurnal demand pattern. In these examples, data replication across the various locations of an ICP, leads to bulk data transfers between datacenters on a diurnal pattern.

In the meantime, there is a degree of freedom on when the content is transmitted from the origin server to the caching node, or from the core site to a local site. However, scheduling these data transfers is a non-trivial task as they 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.

As a result, these ICPs need to have a good knowledge on the link utilization patterns between the different datacenters before making

an efficient scheduling decision. While usage data today is already gathered and used to schedule data transfers, provisioning these data gets increasingly complex with the number of CDN nodes and datacenter operators that are involved. In particular, privacy concerns prevent that this kind of data is shared across administrative domains. The ALTO Cost Calendar avoids these problems by presenting an abstracted view of time sensitive utilization maps through a dedicated ALTO service to allow ICPs a coherent scheduling of data transfers across administrative domains and time zones.

Likewise, bandwidth Calendaring allows network operators to reserve resources in advance according to agreements with their customers, enabling them to transmit data with specified starting time and duration, for example, for a scheduled bulk data replication between data centers. Traditionally, this can be supported by a Network Management System operation such as path pre-establishment and activation on the agreed starting time. However, this does not provide efficient network usage since the established paths exclude the possibility of being used by other services even when they are not used for undertaking any service.

An ALTO Cost calendar for TE metrics on transfer paths can support the scheduled bulk data replication with better efficiency since it can alleviate the processing burden on network elements.

Cost calendars for these time-sensitive ALTO TE metrics need to consider the network topology and the dynamicity of the traffic. For example, a small topology with low density and low capacity that carries inpredictable, heavy and bursty traffic has few chances to exhibit stationary TE metric value patterns over large periods and would benefit to use the ALTO Calendar over smaller time slots. Some ALTO TE metric values, even aggregated over time may need to be updated at a frequency that would require doing ALTO requests at a pace that would be overload both the ALTO Client and the Server. Large high capacity topologies would benefit from Cost Calendars with a coarse time granularity for the filtered cost map service where as Calendars of finer time granularity for the Endpoint Cost Service would be better suited for small low density and capacity topologies.

5.1.1. Applicable example transaction

Assuming a Large high capacity topology, an applicable example transaction for this us case is provided by section 4.1.3. "Example transaction for a FCM with a "request-date" bandwidth Calendar".

5.2. Applications with limited connectivity or access to datacenters

Some applications are limited in 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 are enterprise database update, remote learning, remote computation distributed on several data center endpoints.

Wireless connections have a variable quality and may even be intermittent. On the other hand, the wireless network conditions are often predicable and have a rapid impact on applications. Non real time applications and time-insensitive data transfers such as client patching, archive syncing, etc. can benefit from careful scheduling. 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 applications using data and physical 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 network conditions in order to improve the 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.

```
     +-----+                                          +-----+
     | EPa |                                          | EPb | <----- Rb
     +-----+                                          +-----+   (t2=50)
        |                   +-------+                   |
        Ra --------------->  | EPc   |                   |
        (time t1=10)        |       |                   |
                            |t3=100 |  <----------------- Rb
                            +-------+
                                | \
                                |  \
                              R(Ra,Rb)
                             (t4=200)
                                |     \
                                |      -------------------.
                                V                         V
                             +-----+                   +-----+
                             | EPd |                   | EPe |
                             +-----+                   +-----+
```

5.2.1. Applicable example transaction

An applicable example transaction for this us case is provided by section 4.2.3. "Example transaction for the ECS with a "periodic" routingcost Calendar".

5.3. SDN Controller guided traffic scheduling with Calendars

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.

Via the Northbound interface (NBI), applications may get QoE impacting information such as network provider preferences w.r.t. delay and bandwidth on the network paths. Such information may be provided via the ALTO Service if the latter supports the requested metrics.

One key objective of an SDN controller is the ability to balance the application traffic whenever possible. Resources availability may often be predicted and 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 on network utilization over time.

To achieve this objective, the SDN controller can:

1. get the network state information from its controlled network elements through its southbound API and derive an estimation of these values over given time frames

2. abstract the network topology and costs on end to end paths and store this in an ALTO Server in the form of Network Maps and Cost Calendars

3. deliver these values to ALTO Clients linked to SDN applications, through the NBI.

This way:

- On one hand, the applications get the best possible QoE, as they can pick the best time for them to access one or more Endpoints or PIDs,

- One the other hand the SDN controller achieves load balancing and optimizes application traffic as it may guide the application traffic so as to better distribute the traffic over time, and thus optimize its resources usage.

5.3.1. Applicable example transaction

An applicable example transaction for this us case is provided by section 4.2.4. "Example transaction for the ECS with a calendar on both routingcost and latency".

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 [RFC7285],

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 Propeeries

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

Thank you to Diego Lopez, He Peng and Haibin Song and the ALTO WG for fruitful discussions.

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

8.2. Informative References

[draft-ietf-alto-multi-cost]
S. Randriamasy, W. Roome, N. Schwan, , "Multi-Cost ALTO (work in progress), draft-ietf-alto-multi-cost", May 2015.

[draft-wu-alto-te-metrics]
Q. Wu, Y. Yang, Y. Lee, D. Dhody, S. Randriamasy, , "ALTO Traffic Engineering Cost Metrics (work in progress)", October 2014.

[draft-yang-alto-topology]
Y. Yang, , "ALTO Topology Considerations (work in progress)", July 2013.

[ID-alto-protocol]
R.Alimi, R. Penno, Y. Yang, Eds., "ALTO Protocol, RFC 7285", September 2014.

[RFC7285] R. Alimi, R. Yang, R. Penno, Eds., "ALTO Protocol", September 2014.

[sdnrg] "Software Defined Network Research Group, http://trac.tools.ietf.org/group/irtf/trac/wiki/sdnrg".

[slides-88-alto-5-topology]
G. Bernstein, Y. Lee, Y. Yang, , , "ALTO Topology Service: Use Cases, Requirements and Framework (presentation slides IETF88 ALTO WG session), http://tools.ietf.org/agenda/88/slides/slides-88-alto-5.pdf", November 2013.

Authors' Addresses

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

Email: Sabine.Randriamasy@alcatel-lucent.com

Richard Yang
Yale University
51 Prospect st
New Haven, CT 06520
USA

Email: yry@cs.yale.edu

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: sunseawq@huawei.com

Lingli Deng
China Mobile
China

Email: denglingli@chinamobile.com

Nico Schwan
Thales Deutschland

Email: nico.schwan@thalesgroup.com

ALTO WG
Internet-Draft
Intended status: Standards Track
Expires: January 6, 2016

W. Roome
Alcatel-Lucent

July 5, 2015

Extensible Property Maps for the ALTO Protocol
draft-roome-alto-unified-props-00

Abstract

This document extends the Application-Layer Traffic Optimization (ALTO) Protocol [RFC7285] by generalizing the concept of "endpoint properties" to other entity domains, and by presenting those properties as maps, similar to the network and cost maps in ALTO.

Requirements Language

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

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

The ALTO protocol [RFC7285] introduced the concept of "properties" attached to "endpoint addresses." While useful, this concept has at least two limitations.

First, it only allowed properties to be associated with a particular domain of entities, namely individual IP addresses. It is reasonable to think that collections of endpoints, as defined by CIDRs or PIDs, may also have properties. Furthermore, recent proposals ([ID-draft-yang-alto-topology-06] and [ID-draft-yang-alto-path-vector-00]) have suggested new classes of entities with properties. The Endpoint Property Service as defined in RFC7285 is limited to properties associated with individual endpoints, and cannot be extended to new entity domains. Instead, new services, with new request and response messages, would have to be defined for each new entity domain.

Second, the ALTO Endpoint Property Service is only defined as a POST-mode service. Clients must request the properties for an explicit set of addresses. While [RFC7285] defines a GET-mode Cost Map resource which returns all available costs, so a client can get the full set of costs once, and then lookup costs without querying the ALTO server, ALTO does not define an equivalent service for endpoint properties. Granted, it is not be practical to enumerate the properties for every possible internet address. But it is unlikely a property will be defined for every possible address. It is very likely that properties will only be defined for a subset of addresses, and that subset would be small enough to enumerate. This is particularly true if blocks of addresses with a common prefix (e.g., a CIDR) have the same value for a property. Furthermore, entities in other domains may very well be enumerable.

This document proposes a new approach to ALTO properties. Specifically, it defines two new resource types, namely Property Maps and Filtered Property Maps. The former are GET-mode resources which return the property values for all entities in a domain, and are analogous the ALTO’s Network Map and Cost Map resources. The latter are POST-mode resources which return the values for a set of properties and entities requested by the client, and are analogous to ALTO’s Filtered Network Maps and Filtered Cost Maps.

Entity domains and property names are extensible, so that new domains can be defined without revising the messages defined in this document, in the same way that new cost metrics and new endpoint properties can be defined without revising the messages defined by the ALTO protocol.

This proposal would subsume the Endpoint Property Service defined in RFC7285, although that service may be retained for legacy clients (see Section 6).

2. Definitions and Concepts

2.1. Entities

An entity is an object with a (possibly empty) set of properties. Every entity is in a domain, such as the IPv4 and IPv6 domains.

2.2. Domains

A domain is a family of entities. Examples are the internet address and PID domains (see Section 3.1 and Section 3.2). Another example is the proposed domain of Abstract Network Elements associated with topology and routing, as suggested by [ID-draft-yang-alto-path-vector-00].

2.3. Entity Addresses

Every entity has a name of the form:

```
domain-name : domain-specific-entity-address
```

Examples include "ipv4:1.2.3.4" and "ipv6:1234::".

The type EntityAddr denotes a JSON string with an entity address in this format.

The format of the second part of the entity address depends on the domain, and must be specified when registering a new domain. Addresses may be hierarchical, and properties may be inherited based on that hierarchy. Again, the rules defining any hierarchy or inheritance must be defined when the domain is registered.

Note that entity addresses do NOT have a unique textual representation. For example, the strings "ipv6::1" and "ipv6:0:0:0:0:0:0:0:1" refer to the same entity.

2.4. Domain Names

Each domain has a unique name. The name MUST be no more than 32 characters, and it MUST NOT contain characters other than US-ASCII alphanumeric characters (U+0030-U+0039, U+0041-U+005A, and U+0061-

U+007A), the hyphen ('-', U+002D), or the low line ('_', U+005F). For example, the names "ipv4" and "ipv6" identify objects in the internet address domain (Section 3.1).

The type DomainName denotes a JSON string with a domain name in this format.

Domain names must be registered with the IANA, and the format of the entity addresses in that domain, as well as any hierarchical or inheritance rules for those entities, must be specified at the same time.

2.5. Property Names

The space of property names associated with entities defined by this document is the same as, and is shared with, the endpoint property names defined by [RFC7285]. Thus entity property names are as defined in Section 10.8.2 of that document, and must be registered with the "ALTO Endpoint Property Type Registry" defined in Section 14.3 of that document.

The type PropertyName denotes a JSON string with a property name in this format.

Property names are not specific to a domain, although some properties may only be applicable for particular domains, and the interpretation of the value may depend on the domain. For example, suppose the "geo-location" property is defined as the coordinates of a point, encoded as (say) "latitude longitude [altitude]." When applied to an entity that represents a specific host computer, such as an internet address, the property defines the host's location. When applied to an entity that represents a set of computers, such as a CIDR, the property would be the location of the center of that set. If it is necessary to represent the bounding box of a set of hosts, another property, such as "geo-region", should be defined.

2.6. Relationship to Network Maps

[RFC7285] recognized that some properties may be specific to another ALTO resource, such as a network map. Accordingly [RFC7285] defined the concept of "resource-specific endpoint properties" (Section 10.8.1), and indicated that dependency by prefixing the property name with the ID of the resource on which it depends. That document defined one resource-specific property, namely the "pid" property, whose value was the name of the name of the PID containing that endpoint in the associated network map.

This document also recognizes that some properties may be specific to

resources such as network maps, but takes a different syntactic approach. Instead of associating the resource dependency with a property, this document takes the position that the dependency is determined by the entity domain, not the property, and is shared by all entries in that domain. For example, the Abstract Network Elements suggested by [ID-draft-yang-alto-path-vector-00] are defined in the context of a network map. If an ALTO server offers two separate network maps, there would be two separate spaces of Abstract Network Elements, one for each network map.

Therefore instead of qualifying a property name with the ID of the resource on which it depends, this document associates the dependent resource(s) with the property map as a whole, via the "uses" mechanism defined in Section 9.1.5 of [RFC7285]. Thus all entities and properties in any given property map depend on those resources(s).

According to [RFC7285], an ALTO server with two network maps, with resource IDs "net1" and "net2", could offer a single Endpoint Property Service for the two properties "net1.pid" and "net2.pid". Instead, an ALTO server which supports the extensions in this document would offer two different property maps for the "pid" property, one depending on "net1", the other on "net2".

3. Entity Domains

This document defines the following entity domains.

3.1. Internet Address Domains

The domain of internet addresses consists of two domains (IPv4 and IPv6). Both domains include individual addresses and blocks of addresses.

3.1.1. IPV4 Domain

3.1.1.1. Domain Name

ipv4

3.1.1.2. Domain-Specific Entity Addresses

Individual addresses are strings as specified by the IPv4Addresses rule of Section 3.2.2 of [RFC3986]. Blocks of addresses are prefix-match strings as specified in Section 3.1 of [RFC4632].

For the purpose of defining properties, an individual internet

address and the corresponding 32-bit prefix are considered aliases for the same entity. That is, "ipv4:10.0.0.0" and "ipv4:10.0.0.0/32" are equivalent, and have the same set of properties.

3.1.2. IPV6 Domain

3.1.2.1. Domain Name

ipv6

3.1.2.2. Domain-Specific Entity Addresses

Individual addresses are strings as specified by Section 4 of [RFC5952]. Blocks of addresses are prefix-match strings as specified in Section 7 of [RFC5952].

For the purpose of defining properties, an individual internet address and the corresponding 128-bit prefix are considered aliases for the same entity. That is, "ipv6:::1" and "ipv:::1/128" are equivalent, and have the same set of properties.

3.1.3. Heirarchy And Inheritance

Both domains allow property values to be inherited. Specifically, if a property P is not defined for a specific internet address IP, but P is defined for some block C which prefix-matches IP, then the address IP inherits the value of P defined for block C. If more than one such block defines a value for P, IP inherits the value of P in the block with the longest prefix.

Address blocks can also inherit properties: if property P is not defined for a block C, but is defined for some block C’ prefix-matches C, and C’ has a shorter mask than C, then block C inherits the property from C’. If there are several such blocks C’, C inherits from the block with the longest prefix.

As an example, suppose that a server defines the property P for the following entities:

```
ipv4:10.0.0.0/8:  P=v1
ipv4:10.0.0.0/16: P=v2
ipv4:10.0.0.0/24: P=v3
ipv4:10.0.0.0:    P=v4
```

Defined Property Values

Then the following entities have the indicated values:

```
ipv4:10.0.0.0:    P=v4
ipv4:10.0.0.1:    P=v3
ipv4:10.0.1.0:    P=v2
ipv4:10.1.0.0:    P=v2
ipv4:11.0.0.0:    (not defined)
ipv4:10.0.0.0/28: P=v3
ipv4:10.0.1.0/28: P=v2
ipv4:10.0.0.0/12: P=v1
ipv4:10.0.0.0/6:  (not defined)
```

Inherited Property Values

3.1.4. Relationship To Network Maps

An internet address domain may or may not be associated with an ALTO network map resource. Logically, there is a map of internet address entities to property values for each network map defined by the ALTO server, plus an additional property map for internet address entities which are not associated with a network map. These maps are separate from each other. The prefixes in the property map do not have to correspond to the prefixes defining the network map’s PIDs. For example, the property map for a network map may assign properties to "ipv4:10.0.0.0/8" even if that prefix is not associated with any PID in the network map.

3.2. PID Domain

The PID domain associates property values with the PIDs in a network map. Accordingly, this domain always depends on a network map.

3.2.1. Domain Name

pid

3.2.2. Domain-Specific Entity Addresses

The entity addresses are the PID names of the associated network map.

3.2.3. Heirarchy And Inheritance

There is no hierarchy or inheritance for properties associated with PIDs.

3.2.4. Relationship To Internet Addresses Domains

The PID domain and the internet address domains are completely independent; the properties associated with a PID have no relation to the properties associated with the prefixes or endpoint addresses in

that PID. An ALTO server MAY choose to assign some or all properties of a PID to the prefixes in that PID, but is not required to do so.

For example, suppose "PID1" consists of the prefix "ipv4:10.0.0.0/8", and has the property P" with value "v1". In internet address entities "ipv4:10.0.0.0" and "ipv4:10.0.0.0/8" may or may not have a value for the property "P", and if they do, it is not necessarily "v1".

3.3. Internet Address Properties vs. PID Properties

Because the internet address and PID domains are completely separate, the question may arise as to which domain is best for a property. In general, the internet address domain is best for properties that are closely related to the internet address, or which are associated with, and inherited through, blocks of addresses.

The PID domain is best for properties that arise from the definition of the PID, rather than from the internet address prefixes in that PID.

For example, because internet addresses are allocated to server providers by blocks of prefixes, an "ISP" property would be best associated with the internet address domain. On the other hand, a property that explains why a PID was formed, or how it relates the a provider’s network, would best be associated with the PID domain.

4. Property Map Resource

A Property Map returns the properties defined for all entities in one of more domains.

Section 7.4 gives an example of a property map request and response.

4.1. Media Type

The media type of an ALTO Property Map resource is "application/alto-propmap+json".

4.2. HTTP Method

An ALTO Property Map resource is requested using the HTTP GET method.

4.3. Accept Input Parameters

None.

4.4. Capabilities

The capabilities are defined by an object of type PropertyMapCapabilities:

```
object {
  DomainName domain-types<1..*>;
  PropertyName prop-types<1..*>;
} PropertyMapCapabilities;
```

where "domain-types" is an array with the domains of the entities in this property map, and "prop-types" is an array with the names of the properties returned for entities in those domains.

4.5. Uses

An array with the resource ID(s) of resource(s) with which the domains in this map are associated. In most cases, this array will have at most one ID, and it will be for a network map resource.

4.6. Response

If the domains in this property map depend on other resources, the "dependent-vtags" field in the "meta" field of the response MUST be an array that includes the version tags of those resources.

The data component of a Property Map response is named "property-map", which is a JSON object of type PropertyMapData, where:

```
object {
  PropertyMapData property-map;
} InfoResourceProperties : ResponseEntityBase;

object-map {
  EntityAddr -> EntityProps;
} PropertyMapData;

object {
  PropertyName -> JSONValue;
} EntityProps;
```

The ResponseEntityBase type is defined in Section 8.4 of [RFC7285].

Specifically, a PropertyMapData object has one member for each entity in the Property Map. The entity’s properties are encoded in the corresponding EntityProps object. EntityProps encodes one name/value pair for each property, where the property names are encoded as strings of type PropertyName. A protocol implementation SHOULD

assume that the property value is either a JSONString or a JSON "null" value, 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.

For each entity in the Property Map, the ALTO Server returns the value defined for each of the properties specified in this resource's "capabilities" list. For efficiency, the ALTO Server SHOULD omit property values that are inherited rather than explicitly defined; if a client needs inherited values, the client SHOULD use the domain's inheritance rules to deduce those values.

An ALTO Server MAY explicitly define a property as not having a value for a particular entity. That is, a server may say that a property is "defined to have no value", as opposed to the property being "undefined". If that entity would inherit a value for that property, then the ALTO server MUST return a "null" value for that property, and an ALTO client MUST recognize a "null" value means "do not apply the inheritance rules for this property." If the entity would not inherit a value, the ALTO server MAY return "null" or MAY just omit the property.

If the ALTO Server does not define any properties for an entity, then the server MAY omit that entity from the response.

5. Filtered Property Map Resource

A Filtered Property Map returns the values of a set of properties for a set of entities selected by the client.

Section 7.5, Section 7.6 and Section 7.7 give examples of filtered property map requests and responses.

5.1. Media Type

The media type of an ALTO Property Map resource is "application/alto-propmap+json".

5.2. HTTP Method

An ALTO Property Map resource is requested using the HTTP POST method.

5.3. Accept Input Parameters

The input parameters for a Filtered Property Map request are supplied in the entity body of the POST request. This document specifies the

input parameters with a data format indicated by the media type "application/alto-propmapparams+json", which is a JSON object of type ReqFilteredPropertyMap:

```
object {
  EntityAddr     entities<1..*>
  PropertyName   properties<1..*>;
} ReqFilteredPropertyMap;
```

with fields:

entities: List of entity 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. The domain of each entity MUST be included in the list of domains in this resource’s "capabilities" field (Section 5.4).

properties: List of properties to be returned for each entity. Each specified property MUST be included in the list of properties in this resource’s "capabilities" field (Section 5.4). The ALTO server MUST interpret entries appearing multiple times as if they appeared only once.

Note that the "entities" and "properties" fields MUST have at least one entry each.

5.4. Capabilities

The capabilities are defined by an object of type PropertyMapCapabilities, as defined in Section 4.4.

5.5. Uses

An array with the resource ID(s) of resource(s) with which the domains in this map are associated. In most cases, this array will have at most one ID, and it will be for a network map resource.

5.6. Response

The response is the same as for the property map (Section 4.6), except that it only includes the entities and properties requested by the client.

Also, the Filtered Property Map response MUST include all inherited property values for the specified entities (unlike the Full Property Map, the Filtered Property Map response does not include enough information for the client to calculate the inherited values).

6. Impact On Legacy Servers And Clients

6.1. Impact on Endpoint Property Service

The property maps defined in this document provide the same functionality as the Endpoint Property Service (EPS) defined in Section 11.4 of [RFC7285]. Accordingly, it is RECOMMENDED that the EPS be deprecated in favor of property maps. However, ALTO servers MAY provide an EPS for the benefit of legacy clients.

6.2. Impact on Resource-Specific Properties

Section 10.8 of [RFC7285] defines two categories of endpoint properties: "resource-specific" and "global". Resource-specific property names are prefixed with the ID of the resource they depended upon, while global property names have no such prefix. The property map resources defined in this document do not distinguish between those two types of properties. Instead, if there is a dependency, it is indicated by the "uses" capability of a property map, and is shared by all properties and entity domains in that map. Accordingly, it is RECOMMENDED that resource-specific endpoint properties be deprecated, and no new resource-specific endpoint properties be defined.

6.3. Impact on the "pid" Property

Section 7.1.1 of [RFC7285] defines the resource-specific endpoint property "pid", whose value is the name of the PID containing that endpoint. For compatibility with legacy clients, an ALTO server which provides the "pid" property via the Endpoint Property Service MUST use that definition, and that syntax, in the EPS resource.

However, when used with property maps, this document amends the definition of the "pid" property as follows.

First, the name of the property is simply "pid"; the name is not prefixed with the resource ID of a network map. The "uses" capability of the property map resource indicates the associated network map. This implies that a property map can only return the "pid" property for one network map; if an ALTO server provides several network maps, it must provide a property map resource for each one.

Second, a client MAY request the "pid" property for a block of addresses. An ALTO server determines the value of "pid" for an address block C as follows. Let CS be the set of all address blocks in the network map. If C is in CS, then the value of "pid" is the name of the PID associated with C. Otherwise, find the longest block

C’ in CS such that C’ prefix-matches C, but is shorter than C. If there is such a block C’, the value of "pid" is the name of the PID associated with C’. If not, then "pid" has no value for block C.

Note that although an ALTO server MAY provide a GET-mode property map resource which returns the entire map for the "pid" property, there is no need to do so, because that map is simply the inverse of the network map.

6.4. Impact on Other Properties

In general, there should be little or no impact on other previously defined properties. The only consideration is that properties can now be defined on blocks of addresses, rather than just individual addresses, which might change the semantics of a property.

7. Examples

7.1. Network Map

The examples in this section use a very simple default network map:

```
defaultpid:  ipv4:0.0.0.0/0  ipv6::0/0
pid1:        ipv4:10.0.0.0/8
pid2:        ipv4:10.0.0.0/16  ipv4:10.1.0.0/16
```

Figure 1: Example Network Map

7.2. Property Definitions

The examples in this section use four additional properties, "ISP", "ASN", "country" and "state", with the following values:

	ISP	ASN	country	state
ipv4:10.0.0.0/8:	MyISP	-	us	-
ipv4:10.0.0.0/16:	-	12345	-	NJ
ipv4:10.0.0.0:	-	-	-	PA
ipv4:10.1.0.0/16:	-	12345	-	CT

Figure 2: Example Property Values

7.3. Information Resource Directory (IRD)

The following IRD defines the relavent resources of the ALTO server. It provides two Property Map resources, one for the "ISP" and "ASN" properties, and another for the "country" and "state" properties. The server could have provided a Property Map resource for all four

properties, but did not, presumably because the organization that runs the ALTO server believes any given client is not interested in all four properties.

The server provides two Filtered Property Maps. The first returns all four properties, and the second just returns the "pid" property for the default network map.

The Property Maps for the "ISP", "ASN", "country" and "state" properties do not depend on the default network map (they do not have a "uses" capability), because the definitions of those properties do not depend on the default network map. The Filtered Property Map for the "pid" property does have a "uses" capability for the default network map, because that defines the values of the "pid" property.

Note that for legacy clients, the ALTO server provides an Endpoint Property Service for the "pid" property for the default network map.

```
  "meta": { ... },
  "resources" : {
     "default-network-map" : {
        "uri" : "http://alto.example.com/networkmap",
        "media-type" : "application/alto-networkmap+json"
     },
     .... cost map resources ....
     "country-state-property-map" : {
        "uri" : "http://alto.example.com/propmap/full/inet-cs",
        "media-type" : "application/alto-propmap+json",
        "capabilities" : {
          "domain-types": [ "ipv4", "ipv6" ],
          "prop-types" : [  "country", "state" ]
        }
     },
     "isp-asn-property-map" : {
        "uri" : "http://alto.example.com/propmap/full/inet-ia",
        "media-type" : "application/alto-propmap+json",
        "capabilities" : {
          "domain-types": [ "ipv4", "ipv6" ],
          "prop-types" : [ "ISP", "ASN" ]
        }
     },
     "iacs-property-map" : {
        "uri" : "http://alto.example.com/propmap/lookup/inet-iacs",
        "media-type" : "application/alto-propmap+json",
        "accepts" : "application/alto-propmapparams+json",
        "capabilities" : {
          "domain-types": [ "ipv4", "ipv6" ],
```

```
            "prop-types" : [ "ISP", "ASN", "country", "state" ]
         }
      },
      "pid-property-map" : {
         "uri" : "http://alto.example.com/propmap/lookup/pid",
         "media-type" : "application/alto-propmap+json",
         "accepts" : "application/alto-propmapparams+json",
         "uses" : [ "default-network-map" ]
         "capabilities" : {
           "domain-types": [ "ipv4", "ipv6" ],
           "prop-types" : [ "pid" ]
         }
      },
      "legacy-pid-property" : {
         "uri" : "http://alto.example.com/legacy/eps-pid",
         "media-type" : "application/alto-endpointprop+json",
         "accepts" : "application/alto-endpointpropparams+json",
         "capabilities" : {
           "prop-types" : [ "default-network-map.pid" ]
         }
      }
   }
```

Example IRD

7.4. Property Map Example

The following example uses the properties and IRD defined above to retrieve a property map for entities with the "ISP" and "ASN" properties. Note that the response does not include the entity "ipv4:10.0.0.0", because it does not have a value for either of those properties. Also note that the entities "ipv4:10.0.0.0/16" and "ipv4:10.1.0.0/16" are refinements of "ipv4:10.0.0.0/8", and hence inherit its value for "ISP" property. But because that value is inherited, it is not explicitly listed in the property map.

```
GET /propmap/full/inet-ia HTTP/1.1
Host: alto.example.com
Accept: application/alto-propmap+json,application/alto-error+json


HTTP/1.1 200 OK
Content-Length: ###
Content-Type: application/alto-propmap+json
{
  "property-map": {
    "ipv4:10.0.0.0/8":   {"ISP: "BitsRus"},
    "ipv4:10.0.0.0/16":  {"ASN": "12345"},
    "ipv4:10.1.0.0/16":  {"ASN": "12345"}
  }
}
```

7.5. Filtered Property Map Example #1

The following example uses the Filtered Property Map resource to request the "ISP", "ASN" and "state" properties for several IPv4 addresses. Note that the value of "state" for "ipv4:10.0.0.0" is the only explicitly defined property; the other values are all derived by the inheritance rules for internet address entities.

```
POST /propmap/lookup/inet-iacs HTTP/1.1
Host: alto.example.com
Accept: application/alto-propmap+json,application/alto-error+json
Content-Length: ###
Content-Type: application/alto-propmapparams+json

{
  "entities" : [ "ipv4:10.0.0.0", "ipv4:10.0.0.1", "ipv4:10.2.0.1" ],
  "properties" : [ "ISP", "ASN", "state" ]
}


HTTP/1.1 200 OK
Content-Length: ###
Content-Type: application/alto-propmap+json
{
  "property-map": {
    "ipv4:10.0.0.0": {"ISP": "MyISP", "ASN": "12345", "state": "PA"},
    "ipv4:10.0.0.1": {"ISP": "MyISP", "ASN": "12345", "state": "NJ"},
    "ipv4:10.1.0.0": {"ISP": "MyISP", "ASN": "12345", "state": "CT"}
  }
}
```

7.6. Filtered Property Map Example #2

The following example uses the Filtered Property Map resource to request the "ASN", "country" and "state" properties for several IPv4 prefixes. Note that none of the returned property values were explicitly defined; all values are derived by the inheritance rules for internet address entities.

Also note the "ASN" property has the value "12345" for both the blocks "ipv4:10.0.0.0/16" and "ipv4:10.1.0.0/16", so every address in the block "ipv4:10.0.0.0/15" has that property value. However the block "ipv4:10.0.0.0/15" itself does not have a value for "ASN": address blocks cannot inherit properties from blocks with longer prefixes, even if every such block has the same value.

```
POST /propmap/lookup/inet-iacs HTTP/1.1
Host: alto.example.com
Accept: application/alto-propmap+json,application/alto-error+json
Content-Length: ###
Content-Type: application/alto-propmapparams+json

{
  "entities" : [ "ipv4:10.0.0.0/10",
                 "ipv4:10.0.0.0/15",
                 "ipv4:10.0.0.0/17" ],
  "properties" : [ "ASN","country", "state" ]
}


HTTP/1.1 200 OK
Content-Length: ###
Content-Type: application/alto-propmap+json
{
  "meta" : {
    "dependent-vtags" : [
       {"resource-id": "default-network-map",
        "tag": "7915dc0290c2705481c491a2b4ffbec482b3cf62"}
    ]
  },
  "property-map": {
    "ipv4:10.0.0.0/10":  {"country": "us"},
    "ipv4:10.0.0.0/15":  {"country": "us"},
    "ipv4:10.0.0.0/17":  {"ASN": "12345",
                          "country": "us",
                          "state": "NJ"}
 }
}
```

7.7. Filtered Property Map Example #3

The following example uses the Filtered Property Map resource to request the "pid" property for several IPv4 addresses and prefixes.

Note that the value of "pid" for the prefix "ipv4:10.0.0.0/15" is "pid1", even though all addresses in that block are in "pid2", because "ipv4:10.0.0.0/8" is the longest prefix in the network map which prefix-matches "ipv4:10.0.0.0/15", and that prefix is in "pid1".

```
POST /propmap/lookup/pid HTTP/1.1
Host: alto.example.com
Accept: application/alto-propmap+json,application/alto-error+json
Content-Length: ###
Content-Type: application/alto-propmapparams+json

{
  "entities" : [
                "ipv4:10.0.0.0",
                "ipv4:10.1.0.0",
                "ipv4:10.3.0.0",
                "ipv4:11.0.0.0",
                "ipv4:10.0.0.0/15",
                "ipv4:10.0.0.0/17" ],
  "properties" : [ "pid" ]
}


HTTP/1.1 200 OK
Content-Length: ###
Content-Type: application/alto-propmap+json
{
  "property-map": {
    "ipv4:10.0.0.0":    {"pid": "pid2"},
    "ipv4:10.1.0.0":    {"pid": "pid2"},
    "ipv4:10.3.0.0":    {"pid": "pid1"},
    "ipv4:11.0.0.0":    {"pid": "defaultpid"},
    "ipv4:10.0.0.0/15": {"pid": "pid1"},
    "ipv4:10.0.0.0/17": {"pid": "pid2"}
  }
}
```

8. Security Considerations

As discussed in Section 15 of [RFC7285], properties may have sensitive customer-specific information. If this is the case, an

ALTO Server may limit access to those properties by providing several different property maps. For non-sensitive properties, the ALTO Server would provide a URI which accepts requests from any client. Sensitive properties, on the other hand, would only be available via a secure URI which would require client authentication.

Also, while technically this document does not introduce any security risks not inherent in the Endpoint Property Service defined by [RFC7285], the GET-mode property map resource defined in this document does make it easier for a client to download large numbers of property values. Accordingly, an ALTO Server should limit GET-mode property maps for to properties which do not contain sensitive data.

9. IANA Considerations

This document defines additional application/alto-* media types, and extends the ALTO endpoint property registry.

9.1. application/alto-* Media Types

This document registers two additional ALTO media types, listed in Table 1.

Type	Subtype	Specification
application	alto-propmap+json	Section 4.1
application	alto-propmapparams+json	Section 5.3

Table 1: Additional ALTO Media Types

Type name: application

Subtype name: This documents registers multiple subtypes, as listed in Table 1.

Required parameters: n/a

Optional parameters: n/a

Encoding considerations: Encoding considerations are identical to those specified for the "application/json" media type. See [RFC7159].

Security considerations: Security considerations relating to the generation and consumption of ALTO Protocol messages are discussed in Section 15 of [RFC7285].

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 1 for the section documenting each media type.

Applications that use this media type: ALTO servers and ALTO clients either stand alone or are embedded within other applications.

Additional information:

Magic number(s): n/a

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

Macintosh file type code(s): n/a

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

Intended usage: COMMON

Restrictions on usage: n/a

Author: See Authors' Addresses section.

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

9.2. ALTO Entity Domain Registry

This document requests IANA to create and maintain the "ALTO Entity Domain Registry", listed in Table 2.

Identifier	Entity Address Encoding	Hierarchy & Inheritance
ipv4	See Section 3.1.1	See Section 3.1.3
ipv6	See Section 3.1.2	See Section 3.1.3
pid	See Section 3.2	None

Table 2: ALTO Entity Domain Names

This registry serves two purposes. First, it ensures uniqueness of identifiers referring to ALTO entity domains. Second, it states the requirements for allocated domain names.

New ALTO entity domains are assigned after IETF Review [RFC5226] to ensure that proper documentation regarding the new ALTO entity domains and their security considerations has been provided. RFCs defining new entity domains should indicate how an entity in a registered domain is encoded as an EntityName, and, if applicable, the rules defining the entity hierarchy and property inheritance. Updates and deletions of ALTO entity domains follow the same procedure.

Registered ALTO entity domain identifiers MUST conform to the syntactical requirements specified in Section 2.4. Identifiers are to be recorded and displayed as strings.

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

- o Identifier: The name of the desired ALTO entity domain.
- o Entity Address Encoding: The procedure for encoding the address of an entity of the registered type as an EntityAddr (see Section 2.3).
- o Hierarchy: If the entities form a hierarchy, the procedure for determining that hierarchy.
- o Inheritance: If entities can inherit property values from other entities, the procedure for determining that inheritance.
- o Security Considerations: In some usage scenarios, entity 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", "ipv6" and "pid", as shown in Table 2.

9.3. ALTO Endpoint Property Type Registry

The ALTO Endpoint Property Type Registry was created by [RFC7285]. If possible, the name of that registry should be changed to "ALTO

Entity Property Type Registry", to indicate that it is not restricted to Endpoint Properties. If it is not feasible to change the name, the description must be amended to indicate that it registers properties in all domains, rather than just the internet address domain.

10. References

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

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

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

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

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

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

[RFC7285] Almi, R., Penno, R., Yang, Y., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, September 2014.

[ID-draft-yang-alto-path-vector-00] Bernstein, G., Lee, Y., Roome, W., Scharf, M., and Y. Yang, "ALTO Topology Extension: Path Vector as a Cost Mode", March 2015.

[ID-draft-yang-alto-topology-06] Bernstein, G., Lee, Y., Roome, W., Scharf, M., and Y. Yang, "ALTO Topology Extensions: Node-Link Graphs", March 2015.

Author’s Address

Wendy Roome
Alcatel-Lucent/Bell Labs
600 Mountain Ave, Rm 3B-324
Murray Hill, NJ 07974
USA

Phone: +1-908-582-7974
Email: w.roome@alcatel-lucent.com

ALTO WG
Internet-Draft
Intended status: Informational
Expires: January 7, 2016

X. Wang
S. Dong
Tongji University
G. Chen
Huawei Technologies

July 6, 2015

Design and Implementation of Large Data Transfer Coordinator
draft-wang-alto-large-data-framework-00.txt

Abstract

The Application-Layer Traffic Optimization (ALTO) protocol provides network information with the goal of improving both application performance and network resource utilization. As data transfers become larger (e.g., due to big data analysis), more data transfers are concurrent but with service requirements, and more network capabilities are emerging (e.g., SDN allowing a data transfer to request specific routes or Qos), the management of large data transfers has become an increasingly challenging issue. This document introduces Data Transfer Center (DTC), a centralized framework to coordinate and schedule large data transfers. DTC considers all three components: data transfer requirements, (ALTO) network information, and SDN control capabilities. This document specifies not only the basic framework of DTC, but also a key component, Data Transfer Set (DTS) to specify data transfers and their relations.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

There is substantial need to manage large data transfers. Considering limited network resources such as bandwidth, inappropriate handling large data transfer would reduce performance significantly. It could be easier to cause network congestion than low traffic. Congested network can result in higher rate of packet loss, then triggers retransmissions, which can cripple already

heavily loaded networks. It's necessary to manage large data transfer not only for high network resource utilization but also for users' experience aspect.

Scheduling data flows needs network information such as available bandwidth between two transfer nodes. ALTO defines cost maps providing cost between two pids and endpoint cost service for two endpoints. By utilizing these network information, application can determine how to allocate bandwidth for each data flow. However, to achive such scheduling, there needs a centralized coordinator that can be aware of every data flow requirements. Moreover, to get the customized requirements for each data transfer, a general interface is need to obtain the correlation among data flows besides single data flow requirements.

This document introduces the design and implementation of a framework to Data Transfer Center (DTC), a centralized framework to schedule data flows based on network information and correlation among data flows. Framework design details are described in Section 4.1.

This document is organized as follows: Section 4 gives our general architecture for large data transfer. Section 5 gives details on DTS, Data Transfer Task (DTT) and northbound API under our general architecture.

2. 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 [RFC2119].

3. Terminology and Notation

This document uses the following additional terms: DTT, DTS, Relation.

- o DTT

  Data Transfer Task. The data transfer request for a single file or directory. It's the minimum data transfer unit user can manipulate. See more detailed description in Section 5.1.

- o DTS

  Data Transfer Set. A group of DTTs which have relations with each other but have no relation with any other DTTs outside of the DTS it belonging to. See more detailed description in Section 5.2.

- o Relation

  A relation defines how two DTTs connect. Any DTTs having relations will be grouped into the same DTS. See more detailed description in Section 5.2.

4. Framework

4.1. Architecture

This section decribes the design details of four components of the framework, 1. Job Collector; 2. ALTO Client; 3. Task Scheduler; 4. DTN Controller. Among these four modules, task scheduler is the core of the framework. Job Collector provides interface to users for submitting data transfer requests, which will be tranformed to DTTs and DTSs and passed to task scheuler for further process. Task scheduler makes scheduling based on the network information generated by ALTO client as well as the requirements of each data transfer from DTS. After computing allocation of bandwidth for each DTT, task scheduler will send transfer commands to DTN controller to start data transmission. Figure 1 shows the whole process.

```
                              .-----------.
                              |   Users   |
                              '-----------'
                                    | submit jobs
  .- - - - - - - - - - - - - - - - - | - - - - - - - - - - - - - - - - .
  |                           .-----------.                          |
  |                           |    Job    |                          |
  |                           | Collector |                          |
  |    DTC                    '-----------'                          |
  |                                | transform data transfer          |
  |                                | requests to DTTs and DTSs        |
  | .-----------.             .-----------.           .---------.    |
  | |    DTN    |-----------|   Task    |-----------|   ALTO  |    |
  | | Controller|  send     | Scheduler |   get     |  Client |    |
  | '-----------' transfer   '-----------' network   '---------'    |
  |               commands                 state                     |
  ' - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -'
```

The benefits of DTC include:

- o 1. It can achieve better network resource (bandwidth) allocation since it manages all data transfer requirement in a centralized framework.

- o 2. It takes customized data transfer requirement into consideration by introducing DTS to capture correlation among data flows.

- o 3. It’s modular to support different scheduler algorithm implementations.

4.2. Job Collector

The job collector is responsible to manage data transfer requests from user and transforming these data transfer requests to DTTs for task scheduler to process. It is important that the requests are dynamic and hence the API of the job collector allows dynamic insertion and deletion of data transfers. Details of the data transfer description and APIs for users are described in Section 5.3: Northbound API.

4.3. ALTO Client

ALTO client will be responsible to get network state to task scheduler for further usage. Although different scheduling algorithms may request different ALTO services, cost map and endpoint cost map seems to be the most useful services for scheduling tasks.

4.3.1. PASSIVE and ACTIVE Mode

ALTO client should support two modes according to the way it perceives network state changes, PASSIVE and ACTIVE. In PASSIVE mode, ALTO client will query ALTO server periodically to get latest network states. If the network state changes after one query, the ALTO client will not be aware of the change until next query. In ACTIVE mode, ALTO client will only query ALTO server once to get the initial network state. If network state changes after that, the ALTO client will be notified by ALTO server so it does not have to query ALTO server again. Note that ACTIVE mode will only be supported by ALTO server with ALTO SSE implemented.

4.4. Task Scheduler

The duty of task scheduler is to assign tasks from job collector to proper data transfer nodes(DTNs), splitting a file to several partial files to different DTNs if necessary, and notify the DTN controller to initiate the transfer. We will not discuss specific algorithm in this document but we assume algorithms used by scheduler should take network states provided by ALTO client into consideration. Different schedulers may obey different principles, some schedulers aims to maximize the number of finished tasks while some try to transfer as much data as possible.

4.4.1. Priority Model

In this section, we proposed a schedule model based on priority. In this model, every DTT will be set a predefined priority value, e.g. LOW, MEDIUM and HIGH, to indicate how important it is. The principle of this model is that DTTs with higher priority have the privilege to occupy more resources such as available bandwidth. If the priority is not set, the DTT must be set a default one. Things become tricky when user does not specify priority but an expected finish time instead. However, in this model, it is easy to be solved by transforming expected finish time to priority by following steps:

- o 01. Assign the lowest priority to the task and schedule the task.

- o 02. Calculate the task’s estimated finish time. If the estimated finish time is longer than user specified finish time, increase the task priority by one and reschedule the task, else the schedule procedure completes.

- o 03. Keep doing step 2 until either the schedule procedure completes or the task is assigned as highest priority. If the task is still not able to be finished, we will keep it as highest priority and transfer as much data as possible.

The specific algorithm used to adjust the resources according to the priority is not described in this document.

4.5. DTN Controller

DTN controller is only responsbile for two following functions:

- o 01. Receive and process instructions from task scheduler, e.g. starting a new transfer, aborting a running transfer and adjusting transfer parameters such as transfer rate or number of connections.

- o 02. Monitor transfer status and update status changes to task scheduler. If a transfer failed or finished, it should notify task scheduler the details for further scheduling.

If we assume task scheduler is a manager, then DTN controller are workers who focusing on its own job without caring anything else. DTN controllers are not able to communicate with each other, which means it does not have a global view. Since the DTN controller has to utilize DTNs to transfer data, it should be deployed either in a server able to access DTNs or in the DTNs themselves.

5. Data Transfer Description

Introducing a systematic description of data transfer is challenging. Although it is easy to describe each individual data transfer, this simple description method is not sufficient for a centralized data transfer coordinator because it is not capable of representing relations, e.g. dependencies, between different data transfers. To solve this problem, this section introduces the concetps Data Transfer Task (DTT) first, then it enumerates common relations between different DTTs and introduces the concept of Data Transfer Set (DTS) to represent them.

5.1. Data Transfer Task (DTT) Description

The schema for DTT representation is decribed as following Backus-Naur Form:

```
dtt := dtt_id src_file_candidates dst_file [requirement]
src_file_candidates := src_file {src_file_candidates}
src_file := resource_path
dst_file := resource_path
resource_path := ss_id file_path
requirement := [start_time] [finish_time] [priority]
```

with fields:

- o src_file_candidates

  This field specifies the source file candidates. since there could be multiple sources for a file, the field is set as a list of source files while each source file is represented by a resource path decribed below.

- o dst_file

  This field specifies the destination file. Unlike multiple source file candidates, there could only be one destination file for a DTT. Destionation file is also represented by a resource path.

- o resource_path

  This field identifies a unique resource in multiple storege systems. Since a storge system could be connected by multiple data transfer nodes, it is not accurate to identify a resource by server host and file path anymore. To solve this problem, DTC will assgin every connected storge system a unique id. Thus, users can combine ss_id, which is the unique storge system id, and

file_path, which indicates location of the file in the corresponding storge system, to identify a unique resource.

- o requirements

  This field specifies the requirement of the task. Currently, the DTT defines three kinds of requirements, start_time, finish_time and priority.

5.2. Data Transfer Set (DTS) Description

The schema for DTS representation is decribed as following Backus-Naur Form:

```
dts := dts_id relation_list
relation_list := relation {relation_list}
relation := depend | sharing
depend := dtt_id "depend" dtt_id
share := dtt_id "share" dtt_id requirement
requirment := "start_time" | "finish_time"
```

- o relation

  This field indicates how two DTTs connect. This document specifies two kinds of relations, "depend" and "share". "depend" relation points out that one DTT depends on another DTT, which means the former DTT must not start until the latter completes. "share" relation points out that two DTTs share the same requirement. For example, if two DTTs share the same finish_time, they are expected to finish at the same time.

"depend" and "share" relation are both transitive. If DTT_1 depends on DTT_2, and DTT_2 depends on DTT_3, then DTT_1 depends on DTT_3. If DTT_4 shares requirement R with DTT_5, and DTT_5 shares requirements R with DTT_6, then DTT_4 shares requirements R with DTT_6.

5.3. Northbound API

Normally, users will submit a group of DTTs at the same time to submit a DTS. While DTS is running, the user should be able to add DTTs to or remove DTTs from DTS dynamically. To enable these features, a job collector should provide the following five functions for user:

- o submitDTS(dtt_list)

This function creates a new DTS. It accepts a list DTT as parameter and must return a DTS id for user to identify the DTS created. If the creation fails, it must throw an error.

- o abortDTS(dts_id)

  This function aborts a running DTS. It accepts a dts_id parameter and must abort all DTTs belonging to the DTS. The function return value should indicate if the abort action succeeds or not. If the DTS does not exist, it must throw an error.

- o addDTT2DTS(dts_id, dtt)

  This function adds a new DTT to a existing DTS. This function accepts a dtt_id and a list of DTS as parameters. The function return value should indicate if the add action succeeds or not.

- o removeDTTFromDTS(dtt_id, src_resource, dst_resource)

  This function removes a DTT from a existing DTS. This function accepts a dtt_id and a source and destination resource path pair. The source and destination resource pair will identify a unique DTT to be removed. The function return value should indicate if the remove action succeeds or not.

6. Security Considerations

This document has not conducted its security analysis.

7. IANA Considerations

This document does not specified its IANA considerations, yet.

8. Acknowledgments

The authors thank discussions with Yicheng Qian.

9. References

9.1. Normative References

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

9.2. Informative References

[RFC7285] Alimi, R., Penno, R., Yang, Y., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, September 2014.

Authors' Addresses

Xin Wang
Tongji University
4800 Cao'an Road, Jiading District
Shanghai
China

Email: xinwang2014@hotmail.com

Shu Dong
Tongji University
4800 Cao'an Road, Jiading District
Shanghai
China

Email: dongs2011@gmail.com

Guohai Chen
Huawei Technologies
101 Software Avenue, Yuhua District
Nanjing
China

Email: chenguohai@huawei.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 7, 2016

P. Wydrych
AGH University
Q. Fu
China Mobile
July 6, 2015

Paths Extension for the ALTO Protocol
draft-wydrych-alto-paths-00

Abstract

The Application-Layer Traffic Optimization (ALTO) protocol has been standardized in RFC7285 to ease better-than-random overlay connection management. However, through the base protocol it is not possible to differenciate paths from a given source to a given destination and provide an ALTO client with a detailed information of sending traffic through these paths.

This document describes an extension to the ALTO protocol allowing for representation of paths in the network maps, cost maps, and endpoint cost calculation. Morever, this document defines a new path computation service.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

As stated in [RFC5693], information on network topologies and routing policies in today Internet are not generally available to the application layer. At the same time, a lot of new overlay

applications creating their own topologies on top of the physical one emerge. As both network operators and users of the overlay applications may benefit from better-than-random overlay connection management, the ALTO protocol has been standardized in [RFC7285].

In the current networks, a number of unique paths from a given source to a given destination often exist. Even for the same pair of source and destination, the routing cost may be significantly different for each of the paths. Especially in case of transit traffic, cost may depend on ingress and egress link policies. Other factors that may influence the routing cost are, e.g., load of the links traversed throughout the network or capabilities of used routers.

Moreover, a number of tunneling techniques are used currently to handle parts of the traffic separately. To ensure that services of high quality are provided to customers, Internet Service Providers (ISPs) set up tunnels using specified paths. Then, network devices are configured to take into account the tunnel parameters and, e.g., prioritize traffic sent through a tunnel by using separate queues on routers.

```
                                +------+
                                |      |
                             ,--+ PID3 +--.
                            /   |      |   \
+------+       +------+  /    +------+    \  +------+       +------+
|      +-------+      +-'    /        \    '-+      +-------+      |
| PIDA |       | PID1 |     /          \     | PID6 |       | PIDY |
|      +.     ,+      +---./            \,---+      +.     ,+      |
+------+ \   / +------+   /\  +------+  /\   +------+ \   / +------+
          \ /          \ /  '-+      +-'  \ /          \ /
           X            X     | PID4 |     X            X
          / \          / \  ,-+      +-.  / \          / \
+------+ /   \ +------+   \/  +------+  \/   +------+ /   \ +------+
|      +'     '+      +---'\            /'---+      +'     '+      |
| PIDB |       | PID2 |     \          /     | PID7 |       | PIDZ |
|      +-------+      +-.    \        /    ,-+      +-------+      |
+------+       +------+  \    +------+    /  +------+       +------+
                            \   |      |   /
                             '--+ PID5 +--'
                                |      |
                                +------+
```

Figure 1: A redundant network topology.

In Figure 1, a redundant network topology has been presented. Due to the abovementioned observations, the cost of sending data from PIDA via PID1, PID4, and PID7 to PIDZ (using a known path) does not have

to be equal to the sum of costs of sending independent flows from PIDA to PID1, from PID1 to PID4, from PID4 to PID7, and from PID7 to PIDZ in a best-effort way.

Therefore, information on the cost of data transfer from a source to a destination provided by the network and cost maps using a base ALTO Protocol may be not accurate.

1.1. Use-cases

There are a number of situations in which would be optimal if a network device selected a path to a destination based not only on the information from the routing information databases. In this document, the following are shortly described:

- o establishments tunnels in MPLS networks,
- o providing optimal EID-to-RLOC mapping in LISP sites,
- o storing content replicas in CDNs,
- o forwarding data between virtualized network functions.

1.1.1. MPLS and RSVP-TE

Currently, a lot of core networks use Multi-Protocol Label Switching (MPLS) [RFC3031] to forward IP packets and Resource Reservation Protocol (RSVP) to establish tunnels [RFC3209] between ingress and egress MPLS nodes. Due to the link redundancy, there may be a number of paths that packet may flow through for each ingress/egress nodes pair. The ALTO Protocol may be used during the optimal path establishment process, taking into account various factors like link policies or device and link load.

For instance, PIDA and PIDB (in Figure 1) may aggregate the traffic sources, while PIDY and PIDZ - the traffic destinations. If a MPLS network is established between nodes in PID1 to PID9, PID1 and PID2 are ingress MPLS nodes, and PID6 and PID7 are the egress MPLS nodes, the ALTO Protocol with a paths extension may be used by the control plane to assist tunnel establishment from the ingress to the egress nodes. Then, ISP may take into account source and destination of the packets to direct the traffic into a specific tunnel.

1.1.2. Locator/ID Separation

The recently standardized Locator/ID Separation Protocol (LISP) [RFC6830] separates the IP addresses into two numbering spaces: Endpoint Identifiers (EIDs) and Routing Locators (RLOCs). Due to the

multihoming, one EID may be possible to be mapped to more than one RLOC. The ALTO Protocol may be used by the mapping service and/or by the ingress nodes select the optimal EID-to-RLOC mapping and, thus, the path used by the encapsulated packets.

For instance, PIDA and PIDB (in Figure 1) may aggregate the traffic sources, while PIDY and PIDZ - the traffic destinations. If a LISP network is established between nodes in PID1 to PID9, PID1 and PID2 are Ingress Tunnel Routers, and PID6 and PID7 are the Egress Tunnel Routers, the ALTO Protocol with a paths extension may be used by the EID-to-RLOC mapping service to assist LISP tunnel establishment from the ingress to the egress nodes.

1.1.3. Content Delivery Networks

The content-delivery networks (CDNs) use replica servers to take benefit of caching in providing high quality service to the end-users [CDNs]. One of the crucial parts of each CDN is the algorithm deciding which data should be cached on which replica server and for how long. Especially, when an end-user requests for a relatively new content that is available only at the origin servers, it may be cached on the replica servers while it is being delivered to the end-user.

For instance, PIDA and PIDB (in Figure 1) may aggregate the origin servers, i.e., the authoritative content sources, while PIDY and PIDZ - the end-users’ devices. If a CDN network design is tiered, PID1 and PID2 may aggregate data centers (DCs) with global replica servers, PID3 to PID5 - country-wide caches, and PID6 and PID7 - regional ones. The ALTO protocol with a paths extension may assist the CDN controller in selecting through which DCs traffic from the content source to the end-users should be forwarded to both provide high-quality service and store content replicas in correct caches at the same time.

1.1.4. Service Function Chaining

As described in [I-D.fu-alto-nfv-usecase], the ALTO protocol can be used in the service function chaining (SFC) scenario, in which the SFC control plane may act as an ALTO client and ask ALTO server to provide a cost of a service function path (SFP). SFP is a sequence of network functions which the traffic flow should travel through. Utilizing the ALTO protocol, the SFC control plane can get the cost of each different paths and make the decision of which path to choose. In such a scenario, an extension of path is needed to define the SFP.

For instance, PIDA and PIDB (in Figure 1) may aggregate the service consumers, while PIDY and PIDZ are the service providers. If there is a requirement that the traffic from the service comsumer has to be processed by a sequence of service functions, e.g., a firewall (PID1 or PID2), a deep packet inspector (PID3, PID4, or PID5), and a load-balancer (PID6 or PID7) in the correct order before reaching a service provider, the ALTO protocol with a paths extension may be used to optimize forwarding of the data between the service functions.

1.2. 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. Extension Specification

This document specifies an extension to the ALTO Protocol, as defined in RFC 7285, by adding ability to provide path-specific information through map and endpoint cost services. Moreover, this extension defines a new service called "path computation service".

An ALTO server that is compliant with the paths extension MUST implement at least one of the services:

- o paths-enhanced map service (both path-enabled network map and path-enabled cost map MUST be provided),
- o paths-enhanced endpoint cost service,
- o path computation service.

2.1. Information Resource Directory

This extension defines a new Boolean ALTO server capability: "paths". The default value of the "paths" capability is false. Each resource that provides information using syntax defined in this document MUST be assigned "paths": true capability. For clarity, an ALTO server implementing this extension MAY explicitly assign "path": false capability to the resources that are not providing any path information and are not using the syntax defined in this document.

2.1.1. Example

```
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: 2477
Content-Type: application/alto-directory+json

{
  "meta": {
    "cost-types": {
      "num-routing": {
        "cost-mode": "numerical",
        "cost-metric": "routingcost"
      },
      "ord-routing": {
        "cost-mode": "ordinal",
        "cost-metric": "routingcost"
      }
    },
    "default-alto-network-map": "network-map"
  },
  "resources": {
    "my-default-network-map": {
      "uri": "http://alto.example.com/networkmap",
      "media-type": "application/alto-networkmap+json"
    },
    "my-default-paths-network-map": {
      "uri": "http://alto.example.com/paths/networkmap",
      "media-type": "application/alto-networkmap+json",
      "capabilities": {
        "paths": true
      }
    },
    "numerical-routing-cost-map": {
      "uri": "http://alto.example.com/costmap/num/routingcost",
      "media-type": "application/alto-costmap+json",
      "capabilities": {
        "cost-type-names": [
          "num-routing"
        ]
      },
      "uses": [
        "my-default-network-map"
      ]
    },
    "paths-numerical-routing-cost-map": {
      "uri": "http://alto.example.com/paths/costmap/num/routingcost",
      "media-type": "application/alto-costmap+json",
      "capabilities": {
        "paths": true,
        "cost-type-names": [
```

```
            "num-routing"
          ]
        },
        "uses": [
          "my-default-network-map"
        ]
      },
      "endpoint-cost": {
        "uri": "http://alto.example.com/endpointcost/lookup",
        "media-type": "application/alto-endpointcost+json",
        "accepts": "application/alto-endpointcostparams+json",
        "capabilities": {
          "cost-constraints": true,
          "cost-type-names": [
            "num-routing",
            "ord-routing"
          ]
        }
      },
      "paths-endpoint-cost": {
        "uri": "http://alto.example.com/paths/endpointcost/lookup",
        "media-type": "application/alto-endpointcost+json",
        "accepts": "application/alto-endpointcostparams+json",
        "capabilities": {
          "paths": true,
          "cost-constraints": true,
          "cost-type-names": [
            "num-routing",
            "ord-routing"
          ]
        }
      },
      "path-computation": {
        "uri": "http://alto.example.com/paths/compute",
        "media-type": "application/alto-pathcompute+json",
        "accepts": "application/alto-pathcomputeparams+json",
        "capabilities": {
          "paths": true,
          "cost-constraints": true,
          "cost-type-names": [
            "num-routing",
            "ord-routing"
          ]
        }
      }
    }
  }
```

2.2. Provider-Defined Path Identifier (PPID)

This extension introduces Provider-defined Path Identifiers (PPIDs) to provide a way to specify a sequence of endpoints traversed by the traffic. A PPID is a string of type PPIDName and its associated list of PIDs. The list of PIDs associated to a PPID MUST NOT be empty. The list of PIDs MUST NOT contain an undefined PID.

A PPID Name MUST conform to all PID Name requirements specified by Section 10.1 of RFC 7285. A PPID Name MUST begin with ’path.’, i.e., the word ’path’ encoded in lower-case US-ASCII followed by the dot separator (’.’, U+002E). A PPID Name MUST contain at least one character after the dot separator.

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

2.2.1. PPID Nesting

The Provider-defined Path Identifiers may be nested in case one path contains another. For instance, in case PIDs "PID1", "PID3", and "PID6" are defined, and a PPID "path.1-3" is defined as ["PID1", "PID3"], a PPID "path.1-3-6" consisting of "PID1", "PID3", and "PID6" may be defined both as ["PID1", "PID3", "PID6"] or ["path.1-3", "PID6"].

Nested PPIDs MUST NOT create circular dependencies. I.e., "path.A" MUST NOT contain "path.B" if "path.B" contains (directly or indirectly) "path.A".

2.3. Map Service: Network Map

Through this extension, a set of PPIDs is added to the network map. A network map MUST define all PIDs that PPIDs being defined comprise of. A network map SHOULD define all needed PIDs before defining a PPID. A network map SHOULD define a nested path before defining an outer one.

2.3.1. Example

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

HTTP/1.1 200 OK
Content-Length: 988
Content-Type: application/alto-networkmap+json

{
  "meta": {
    "vtag": {
      "resource-id": "paths-network-map",
      "tag": "e65e696925e7cc350b562b6a7d5f2540"
    }
  },
  "network-map": {
    "PIDA": { "ipv4": [ "192.0.2.0/25" ] },
    "PIDB": { "ipv4": [ "192.0.2.128/25" ] },
    "PID1": { "ipv4": [ "198.51.100.1/32" ] },
    "PID2": { "ipv4": [ "198.51.100.2/32" ] },
    "PID3": { "ipv4": [ "198.51.100.3/32" ] },
    "PID4": { "ipv4": [ "198.51.100.4/32" ] },
    "PID5": { "ipv4": [ "198.51.100.5/32" ] },
    "PID6": { "ipv4": [ "198.51.100.6/32" ] },
    "PID7": { "ipv4": [ "198.51.100.7/32" ] },
    "PIDY": { "ipv4": [ "203.0.113.0/25" ] },
    "PIDZ": { "ipv4": [ "203.0.113.128/25" ] },
    "path.1-3-6": [ "PID1", "PID3", "PID6" ],
    "path.1-4-7": [ "PID1", "PID4", "PID7" ],
    "path.2-5-7": [ "PID2", "PID5", "PID6" ],
    "path.1-3-6-Y": [ "path.1-3-6", "PIDY" ],
    "path.1-3-6-Z": [ "path.1-3-6", "PIDZ" ],
    "path.1-4-7-Z": [ "path.1-4-7", "PIDZ" ],
    "path.2-5-7-Z": [ "path.2-5-7", "PIDZ" ]
  }
}
```

2.4. Map Service: Cost Map

Through a cost map, an ALTO server implementing this extension lists the path costs from sources to destinations via paths defined in the network map.

For each source/destination pair defined by a cost map, where a destination is a PPID, the cost value corresponds to the traffic originated in the source, traversing all-but-last PIDs of the path, and directed to the endpoint belonging to the last PID in the path.

A cost map MUST NOT define costs source/destination pair where source is a PPID. In other words, a PPIDName MUST NOT be a key of a CostMapData dictionary map object.

2.4.1. Example

```
GET /paths/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: 572
Content-Type: application/alto-costmap+json

{
  "meta": {
    "dependent-vtags": [
      {
        "resource-id": "paths-network-map",
        "tag": "e65e696925e7cc350b562b6a7d5f2540"
      }
    ],
    "cost-type": {
      "cost-mode": "numerical",
      "cost-metric": "routingcost"
    }
  },
  "cost-map": {
    "PIDA": { "PIDY": 50, "PIDZ": 100,
              "path.1-3-6-Y": 10, "path.1-3-6-Z": 15,
              "path.1-4-7-Z": 10, "path.2-5-7-Z": 20 },
    "PIDB": { "PIDY": 75, "PIDZ": 125,
              "path.1-3-6-Y": 20, "path.1-3-6-Z": 30,
              "path.1-4-7-Z": 25, "path.2-5-7-Z": 20 }
  }
}
```

In the above example, the routing cost of sending data from 192.0.2.0/25 to 203.0.113.128/25 using a best-effort service is 125, while the routing cost of sending data from 192.0.2.0/25 via 198.51.100.1, 198.51.100.4, and 198.51.100.7 to 203.0.113.128/25 using a dedicated path (e.g., a tunnel) is 25.

2.5. Map-Filtering Service: Filtered Network Map

An ALTO client requesting for network map filtering may specify both PIDs and PPIDs in the "pids" field of the request. Through this extension, PIDs and PPIDs may be requested implicitly. The behavior is different for PIDs and PPIDs requested.

If a PID name was specified, an ALTO server MUST return the requested PID as well as:

- o all PPIDs that have the specified PID at the last entry (i.e., PPIDs for all paths ending in the specified PID);
- o all nested PPID, recursively;
- o all PIDs that are needed to define implicitly requested paths.

If a PPID name was specified, an ALTO server MUST return the requested PPID as well as:

- o all nested PPID, recursively;
- o all PIDs that are needed to define explicitly and implicitly requested paths.

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 and PPIDs.

2.5.1. Example

```
POST /paths/networkmap/filtered HTTP/1.1
Host: alto.example.com
Content-Length: 33
Content-Type: application/alto-networkmapfilter+json
Accept: application/alto-networkmap+json,application/alto-error+json

{
  "pids": [ "PIDA", "PIDY" ]
}

HTTP/1.1 200 OK
Content-Length: 477
Content-Type: application/alto-networkmap+json

{
  "meta": {
    "vtag": {
      "resource-id": "paths-network-map",
      "tag": "e65e696925e7cc350b562b6a7d5f2540"
    }
  },
  "network-map": {
    "PIDA": { "ipv4": [ "192.0.2.0/25" ] },
    "PID1": { "ipv4": [ "198.51.100.1/32" ] },
    "PID3": { "ipv4": [ "198.51.100.3/32" ] },
    "PID6": { "ipv4": [ "198.51.100.6/32" ] },
    "PIDY": { "ipv4": [ "203.0.113.0/25" ] },
    "path.1-3-6": [ "PID1", "PID3", "PID6" ],
    "path.1-3-6-Y": [ "path.1-3-6", "PIDY" ],
  }
}
```

2.6. Map-Filtering Service: Filtered Cost Map

An ALTO client requesting for cost map filtering may specify both PIDs and PPIDs in the "pids"."dsts" field of the request. Through this extension, PPIDs may be requested implicitly. If a PID name was specified as a destination, an ALTO server MUST return the cost map for all source/destination pairs in which the requested PID is a destination as well as all source/destination pairs in which a PPID that have the specified PID at the last entry is a destination.

If the list of destination PIDs is empty, the ALTO server MUST interpret the list as if it contained a list of all currently defined PIDs and PPIDs.

The source PID list MUST NOT contain any PPIDs.

2.6.1. Example

```
POST /paths/costmap/filtered HTTP/1.1
Host: alto.example.com
Content-Type: application/alto-costmapfilter+json
Content-Length: 145
Accept: application/alto-costmap+json,application/alto-error+json

{
  "cost-type": {
    "cost-mode": "numerical",
    "cost-metric": "routingcost"
  },
  "pids": {
    "srcs": [ ],
    "dsts": [ "PIDY" ]
  }
}

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

{
  "meta": {
    "dependent-vtags": [
      {
        "resource-id": "paths-network-map",
        "tag": "e65e696925e7cc350b562b6a7d5f2540"
      }
    ],
    "cost-type": {
      "cost-mode": "numerical",
      "cost-metric": "routingcost"
    }
  },
  "cost-map": {
    "PIDA": { "PIDY": 50, "path.1-3-6-Y": 10 },
    "PIDB": { "PIDY": 75, "path.1-3-6-Y": 20 }
  }
}
```

2.7. Enpoint Cost Service

An ALTO client requesting for cost map filtering may specify both desinations and paths in the "endpoints"."dsts" field of the request.

A cost for a path is requsted by specifying an array of typed endpoint addresses as a destination entry.

For each source endpoint, the "endpoint-cost-map" field of the response contains a tree denoting costs for requested paths. The subsequent endpoints a path comprise of are represented as internal nodes of a cost tree.

2.7.1. Example

```
POST /paths/endpointcost/lookup HTTP/1.1
Host: alto.example.com
Content-Length: 468
Content-Type: application/alto-endpointcostparams+json
Accept: application/alto-endpointcost+json,application/alto-error+json

{
  "cost-type": {
    "cost-mode": "ordinal",
    "cost-metric": "routingcost"
  },
  "endpoints": {
    "srcs": [ "ipv4:192.0.2.2" ],
    "dsts": [
      "ipv4:203.0.113.128.129",
      [
        "ipv4:198.51.100.1",
        "ipv4:198.51.100.3",
        "ipv4:198.51.100.6",
        "ipv4:203.0.113.128.129"
      ],
      [
        "ipv4:198.51.100.1",
        "ipv4:198.51.100.4",
        "ipv4:198.51.100.7",
        "ipv4:203.0.113.128.129"
      ]
    ]
  }
}

HTTP/1.1 200 OK
Content-Length: 495
Content-Type: application/alto-endpointcost+json

{
  "meta": {
    "cost-type": {
```

```
        "cost-mode": "ordinal",
        "cost-metric": "routingcost"
      }
    },
    "endpoint-cost-map": {
      "ipv4:192.0.2.2": {
        "ipv4:203.0.113.128.129": 3,
        "ipv4:198.51.100.1": {
          "ipv4:198.51.100.3": {
            "ipv4:198.51.100.6": {
              "ipv4:203.0.113.128.129": 1
            }
          },
          "ipv4:198.51.100.4": {
            "ipv4:198.51.100.7": {
              "ipv4:203.0.113.128.129": 2
            }
          }
        }
      }
    }
  }
```

2.8. Path Computation Service

This document defines a new service called path computation service. This service provides information on best paths composed of the endpoints specified by a client.

The path computation resource is requested using the HTTP POST method. The media type of the path computation resource is "application/alto-pathcompute+json". An ALTO client supplies the path computation parameters through a media type "application/alto-pathcomputeparams+json", with an HTTP POST entity body of a JSON object with fields:

cost-type: The cost type (Section 10.7 of <RFC7285>) to use for returned costs. The "cost-metric" and "cost-mode" fields MUST match one of the supported cost types indicated in this resource's "capabilities" fields. The ALTO client SHOULD omit the "description" field, and if present, the ALTO server MUST ignore the "description" field.

endpoints: A list of lists of endpoints from which paths may be composed. The ALTO server MUST compose a path taking exactly one element from each list, preserving the order.

The response comprises a list of suggested paths. Each path is an object containing a list of endpoints forming the path and the path's cost. The server MAY return more than one path. The server MAY return no paths if it cannot compute an optimal one.

2.8.1. Example

```
POST /paths/compute HTTP/1.1
Host: alto.example.com
Content-Length: 296
Content-Type: application/alto-pathcomputeparams+json
Accept: application/alto-pathcompute+json,application/alto-error+json

{
  "cost-type": {
    "cost-mode": "ordinal",
    "cost-metric": "routingcost"
  },
  "endpoints": [
    [ "ipv4:192.0.2.2" ],
    [ "ipv4:198.51.100.1" ],
    [ "ipv4:198.51.100.3", "ipv4:198.51.100.4" ],
    [ "ipv4:198.51.100.6", "ipv4:198.51.100.7" ],
    [ "ipv4:203.0.113.128.129" ]
  ]
}

HTTP/1.1 200 OK
Content-Length: 536
Content-Type: application/alto-pathcompute+json

{
  "meta": {
    "cost-type": {
      "cost-mode": "ordinal",
      "cost-metric": "routingcost"
    }
  },
  "computed-paths": [
    {
      "path": [
        "ipv4:192.0.2.2",
        "ipv4:198.51.100.1",
        "ipv4:198.51.100.3",
        "ipv4:198.51.100.6",
        "ipv4:203.0.113.128.129"
      ],
      "cost": 1
```

```
      },
      {
        "path": [
          "ipv4:192.0.2.2",
          "ipv4:198.51.100.1",
          "ipv4:198.51.100.4",
          "ipv4:198.51.100.7",
          "ipv4:203.0.113.128.129"
        ],
        "cost": 2
      }
    ]
  }
```

3. IANA Considerations

This document registers two application/alto-* media types: application/alto-pathcomputeparams+json and application/alto-pathcompute+json.

4. Security Considerations

This document does not introduce any privacy or security issues not already present in the ALTO protocol.

5. Compatibility Considerations

The extension defined in this document is compatible with the multi-cost extension [I-D.ietf-alto-multi-cost]. Whenever a cost value is considered in the server response, an array of multiple costs may be used if a server implements both paths and multi-cost extensions.

The extension defined in this document is compatible with the incremental updates using server-sent events [I-D.ietf-alto-incr-update-sse].

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.

[RFC7285] Alimi, R., Penno, R., Yang, Y., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, September 2014.

6.2. Informative References

[CDNs] Pathan, A. and R. Buyya, "A Taxonomy and Survey of Content Delivery Networks", 2007, <http://www.cloudbus.org/reports/CDN-Taxonomy.pdf>.

[I-D.fu-alto-nfv-usecase] Fu, Q., Cao, Z., and H. Song, "What's the Impact of Virtualization on Application-Layer Traffic Optimization (ALTO)?", draft-fu-alto-nfv-usecase-05 (work in progress), June 2015.

[I-D.ietf-alto-incr-update-sse] Roome, W. and Y. Yang, "ALTO Incremental Updates Using Server-Sent Events (SSE)", draft-ietf-alto-incr-update-sse-00 (work in progress), May 2015.

[I-D.ietf-alto-multi-cost] Randriamasy, S., Roome, W., and N. Schwan, "Multi-Cost ALTO", draft-ietf-alto-multi-cost-00 (work in progress), May 2015.

[RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, January 2001.

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, December 2001.

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

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, January 2013.

Appendix A. Acknowledgments

P. Wydrych's work on this draft were performed under contract 15.11.230.190.

Authors' Addresses

Piotr Wydrych
AGH University of Science and Technology
Mickiewicza 30
Krakow 30-059
Poland

Phone: +48 12 617 48 17
Email: piotr.wydrych@agh.edu.pl
URI: http://wydrych.net/

Fu Qiao
China Mobile
Xuanwumenxi Ave. No.32
Beijing
China

Email: fuqiao1@outlook.com