ALTO Working Group
INTERNET-DRAFT
Intended Status: Standard Track
Expires: June 18, 2015

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

November 17, 2014

Extended End Point Properties for Application-Layer Traffic Optimization
draft-deng-alto-p2p-ext-05

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.

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"]
```

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
   }
```

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 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", "2G", "3G", "4G"]
```

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", "2G", "3G", "4G"]
   }

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

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 and Sabine Randriamsy 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

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: September 9, 2015

Q. Fu, Ed.
China Mobile
Z. Cao

H. Song
Huawei

March 8, 2015

What's the Impact of Virtualization on Application-Layer Traffic Optimization (ALTO)?
draft-fu-alto-nfv-usecase-04

Abstract

This documentation presents a use case of Application-Layer Traffic Optimization (ALTO) with the emergence of Network Function Virtualization (NFV). The Application-Layer Traffic Optimization (ALTO) Service provides network information (e.g., basic network location structure and preferences of network paths) with the goal of modifying network resource consumption patterns while maintaining or improving application performance. The emerging NFV, which is currently being in progress in ETSI NFV, leverages standard IT virtualisation technology to consolidate many network equipment types onto industry standard high-volume servers, switches, and storage. The use case presented in this document discusses the impact of virtualization on the ALTO protocol. An architecture is proposed for the interface between NFV MANO and ALTO server. And possible end point property extention is also discussed for such usecase.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document presents a use case of Application Layer Traffic Optimization (ALTO) with the emergence of Network Function Virtualization (NFV). The Application-Layer Traffic Optimization (ALTO) Service provides network information (e.g., basic network location structure and preferences of network paths) with the goal of modifying network resource consumption patterns while maintaining or improving application performance. Typical deployment scenarios of ALTO include P2P networks and Content Distribution Networks (CDNs), in which the P2P tracker or CDN request router queries the ALTO server for network map and cost map, in order to make decisions on which peer to select for content sharing.

The emerging Network Functions Virtualisation (NFV), as currently being in progress in ETSI NFV, leverages standard IT virtualisation technology to consolidate many network equipment types onto industry standard high volume servers, switches and storage. The NFV architecture in ETSI ongoing work includes the NFV MANO(Management and Orchestration), the OSS/BSS, the E/NMS (Element/Network Management System), the VNF (Virtualized Network Function) and the NFVI(Network Function Virtualization Infrastructure), as is shown in

Figure 1. The NFV MANO is composed of the VIM (Virtualized Infrastructure Manager), the NFV Orchestrator, and the VNF Manager. The VIM is responsible for controlling and managing the NFVI compute, storage and network resources, usually within one Operator’s infrastructure sub-domain. The NFV Orchestrator is responsible for the lifecycle management of Network Services across the entire Operator’s domain. The VNF manager is responsible for the lifecycle management of VNF instances.Interactions between NFV MANO, E/NMS, VNF and VNFI are beyond scope of this draft.

With the trend of various network functions being virtualized, there will be impacts on cost and network characteristics of the service endpoints. Under the ALTO architecture, we analyze the problems and the necessity of extending the ALTO protocols to faithfully reveal the network to the clients. The central problem this draft would like to investigate is: what’s the impact of virtualization to ALTO.

```
                                          +---------------------+
                                          |       NFV-MANO      |
                  +-----------+           |    +------------+   |
                  |  OSS/BSS  +-----------+----|Orchestrator|   |
                  +-----+-----+           |    +------+-----+   |
                        |                 |           |         |
                        |                 |    +------+-----+   |
                  +-----------+           |    |            |   |
                  |   E/NMS   |-----------+----+            |   |
                  +-----------+           |    |            |   |
                        |                 |    |    VNFM    |   |
                        |                 |    |            |   |
                  +-----+-----+           |    |            |   |
                  |    VNF    |-----------+----+            |   |
                  +-----------+           |    |            |   |
                        |                 |    +------+-----+   |
                        |                 |           |         |
                  +-----+-----+           |    +------+-----+   |
                  |   NFVI    |-----------+----|    VIM     |   |
                  +-----------+           |    +------------+   |
                                          +---------------------+
```

Figure 1: NFV Architecture in Brief

This document analyzes the impacts of virtualized endpoints to application-layer traffic optimization and presents a use case of ALTO in the CDN and P2P network with the peers as a VNF.

2. Terminology

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

We use the following terms defined in [RFC5693]. Application, Peer, ALTO service, ALTO server, ALTO client, ALTO query, ALTO Reply.

And the following terms used in this document have their definitions from the NFV end to end architecture [NFVE2E].

NFV: network function virtualization. NFV technology uses the commodity servers to replace the dedicated hardware boxes for the network functions, for example, home gateway, enterprise access router, carrier grade NAT and etc. So as to improve the re-usability, allow more vendors into the market, and reduce time to market. NFV architecture includes a NFV MANO to manage the virtual network functions and the infrastructure resources.

NF: A functional building block within an Operator’s network infrastructure, which has well-defined external interfaces and a well-defined functional behavior. Note that the totality of all network functions constitutes the entire network and services infrastructure of an Operator/service provider. In practical terms, a Network Function nowadays is often a network node or physical appliance.

VNF: virtual network function, an implementation of an executable software program that constitutes the whole or a part of an NF that can be deployed on a virtualization infrastructure.

VM: virtual machines, a program and configuration of part of a host computer server. Note that the Virtual Machine inherits the properties of its host computer server e.g. location, network interfaces.

SLA: Service Layer Agreement.

3. Impact of Virtualized Endpoints

This section analyzes the impact of virtualization when application or service endpoints are deployed on virtualized infrastructure.

It is generally believed that generic computing equipment is difficult to accomplish the same capability of specilized and dedicated equipment. Operator network normally consists of many dedicated equipment, and the services running on them are not vitualized. NFV initiatives investigate the use cases, architecture and requirements of moving network functions to the virtualized infrastructure.

We analyze the impacts of virtualized endpoints to application layer traffic optimization for the following aspects.

1. Performance. The NFV framework is claimed to be able to instantiate and configure any given VNF over the underlying infrastructure so that the resulting VNF instance performance is conforming to the expressed requirement. Using appropriate VNF configuration schemes [I-D.song-opsawg-virtual-network-function-config], the Operator or service provider can express their performance requirement. From this point, it is the same as physical and non-virtualized service endpoints. The difference is that the service assurance of virtualized endpoints is more difficult to ensure.

2. Portability. Different from physical equipment, NFV framework is able to provide the capability to load, execute and move VNFs across different but standard mutlivendor environments, and have to support an interface to decouple VNF associated software instances from the underlying infrastructure. Portability has impacts on the mobility and network location of the service points, which in return will impact the service point selection process and service continuity.

3. Elasticity. The NFV framework is able to allow VNFs to be scaled with SLA requirements, on-demand scaling or automatically scaling. With the elasticity capability, VNF endpoints capability with respect to computing and networking are dynamic. The ALTO discovery and selection process will be impact to reflect such dynamic information.

4. Resilience. NFV framework provides the necessary mechanisms to allow VNF to be recreated after a failure. In addition to OAM in traditional non-virtualized environment, the NFV MANO will manage the metrics such as packet loss rate, latency, delay variation of flows, maximum time to detect and recover from faults. All of these information will be valuable to ALTO client.

5. Energy efficient. Studies have indicated that NFV could potentially deliver up to 50% energy saving compared with traditional appliance based network infrastructure. In service point selection, this could be a criteria when the service provider is interested in saving power.

6. Service assurance. Dedicated carrier-grade devices normally have requirements like 99.999%, but the such high availability is still challenging for VNFs. The ALTO server should be aware of the assurance level of these virtualized endpoints.

7. Network infrastructure maintenance. The VNFs may be bridged and linked using the virtualized switches on the computing node. The network layer performance and availability metrics are only possible to collect when the OAM have established the tunnels to these virtual network infrastructure. For example, normal PING can only reflect the physical computing node availability, but cannot reflect the VMs bridged using virual switchs and hidden with tunnel encapsulations.

4. ALTO use case with NFV

The emergence of NFV means that some legacy devices, which used to work on a physical server, now can be moved to a VM and work as a VNF. Under such a circumstance, the NFV MANO can act as a dynamic network information provider for ALTO.

The following paragraph will present a use case of ALTO in CDN with NFV. In the CDN network, the user agent makes an initial request to the Request Router. The Request Router will first query the ALTO server for network and cost map to select an appropriate surrogate. The Request Router then responds to the UA with a redirection to the selected surrogate. The UA then connects directly to the suggested surrogate to obtain the content.

When a certain surrogate changes to a VNF and is managed by a NFV MANO, The NFV MANO can dynamically update the network and cost info of the surrogate to the ALTO server. In this, the NFV MANO should also inform the ALTO server about the virtualized nature of the VNF surrogate. In the migration stage of NFV, in which VNF and physical devices coexist in the network, ALTO server may consider the virtualized nature of VNF and should inform the clients.

In the P2P scenario, similar situations can also happen when peers become VNFs. In this case, NFV MANO should also inform ALTO server about the virtualize nature of the VNF peers.

5. Interaction Architecture of ALTO and NFV

A vertical architecture is proposed in this draft for ALTO and NFV interaction, in which NFV MANO is in responsible of info update to the ALTO server, as is shown in Figure 2.

```
                            +---------------------+
                            |       NFV-MANO      |
         +-----------+      |    +------------+   |       +-------------+
         |  OSS/BSS  +------+----|Orchestrator|   +-------+ ALTO server |
         +-----+-----+      |    +------+-----+   |       +-------------+
               |            |           |         |
               |            |    +------+-----+   |
         +-----------+      |    |            |   |
         |   E/NMS   |------+----+            |   |
         +-----------+      |    |            |   |
               |            |    |    VNFM    |   |
               |            |    |            |   |
         +-----+-----+      |    |            |   |
         |    VNF    |------+----+            |   |
         +-----------+      |    |            |   |
               |            |    +------+-----+   |
               |            |           |         |
         +-----+-----+      |    +------+-----+   |
         |   NFVI    |------+----|     VIM    |   |
         +-----------+      |    +------------+   |
                            +---------------------+
```

Figure 2 ALTO and NFV interaction architecture

In this architecture, NFV MANO can automatically update fine or coarse grained VNF info to the ALTO server timely. The virtualized nature of the VNFs should be informed to the ALTO server by NFV MANO as a rating criteria. In the meantime, details of VNF can be updated to the ALTO server by NFV MANO according to privacy privilege configured by the user.

6. End Point Property Extention

The information NFVO updates to the ALTO server for alto service discovery includes, but not limited to, the topology of the VNFs, the detail info of the network resource allocated to the VNF instances(such as the capacity, the available memory, the available CPU, and etc.), the lifecycle management info of VNFs, and etc.. There is another draft talking about details of the these properties [I-D.deng-alto-p2p-ext].

Endpoint property should also be extended when introducing virtualized end points into ALTO.

1) Endpoint geolocation extention. In traditional physical networks, geolocation for a certain endpoint is determined and should not change frequently. While in the NFV scenarios, endpoints, which should be VNFs, can be composed of several Virtual Mechines (VMs)

located on several physical servers at different geolocatoins. In the meantime, this geolocation may also change due to migration and restoration of the VNF. Such characteristics of VNFs require property extention of endpoint geolocation. How to discribe the multiple geolocations of a certain virtual endpoint, and how frequent to update the geolocation of these virtual endpoints may need careful discussion. Optimization algrithms may also need to change due to the virtualization nature of these endpoints.

2) Node related property extention. One of the benefit NFV brings to us is we can easily scale up and scale down services with the help of the virtualization technology. Such capability should be noticed and taken into consideration by the ALTO server. For instance, in the CDN network, when the request router reach out to the ALTO server asking for proper surrogate, the ALTO server may look out for the network and cost map of all the possible surrogates. At this point, a certain virtual surrogate may not have enough bandwidth or processing capability to handle this request. But it may scale up atomatically when request increase. Therefore, the ALTO server should have knowledge to such capability of the virtual surrogates, and even should be able to inform the NFV MANO to scale up certain service at a proper point. Such usecase requires extention of the node related property.

7. Informative References

[I-D.deng-alto-p2p-ext]
Lingli, D., Song, H., Kiesel, S., Yang, Y., and W. Wu, "Extended End Point Properties for Application-Layer Traffic Optimization", draft-deng-alto-p2p-ext-05 (work in progress), November 2014.

[I-D.song-opsawg-virtual-network-function-config]
Song, H. and Z. Cao, "The Problems of Virtual Network Function Configuration", draft-song-opsawg-virtual-network-function-config-01 (work in progress), October 2013.

[NFVE2E] "Network Functions Virtualisation: End to End Architecture, http://docbox.etsi.org/ISG/NFV/70-DRAFT/0010/NFV-0010v016.zip", .

[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.

Authors' Addresses

Qiao Fu (editor)
China Mobile
China
China

Email: fuqiao1@outlook.com

Zhen Cao

Email: zehn.cao@gmail.com

Haibin Song
Huawei

Email: haibin.song@huawei.com

ALTO WG
Internet-Draft
Intended status: Informational
Expires: September 10, 2015

R. Huang
Huawei
H. Guo
CAICT
Y. Yang
Yale University

March 9, 2015

Network Topology Service (NTS) Framework
draft-huang-alto-topocomp-framework-00

Abstract

This document introduces a network topology service (NTS) framework to collect network topologies from the physical heterogeneous network, NTS analyses and stores the topology information, and provides the path computing and topology information inquiring ability to applications (including network applications like OSS, and third-party applications).

Status of this Memo

Copyright and License Notice

Table of Contents

1. Introduction

Network topology is a prerequisite for performing many critical network management tasks, including resource managements, path computation, event correlation, fault monitoring and analysis. Current networks are continually being refined and upgraded as needs change and technology evolves. Many technologies have developed protocol-specific ways to obtain network topologies for their own usages. For example, a router supporting OSPF maintains an identical area-topology database to determine the shortest path to any neighboring router; BGP maintains a consistent view of network topology to optimize routing and scale the network. However, when network topologies or route paths are required by applications,

applications usually wish to be shielded from protocol-specifics information, even if network state information is collected in protocol-specific ways. It is obvious that none of these methods offer a general-purpose tool that can efficiently manage the network topology for a heterogeneous network with multiple technologies including BGP/OSPF/ISIS, and even SDN Open Flow, etc.

This document introduces a network topology service (NTS) framework to collect network topologies from the physical heterogeneous network, NTS analyses and stores the topology information, and provides flexible path computing and topology information inquiring ability to applications (including network applications like OSS, and third-party applications).

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 uses the following terms:

NTS: Network Topology Service

BGP: Border Gateway Protocol

OSPF: Open Shortest Path First

IS-IS: Intermediate System to Intermediate System

SDN: Software Defined Network

OSS: Operational Support Systems

3. Network Topology Service (NTS) Framework

This section describes the NTS framework as shown in Figure 1:

```
                                xxxxxxxxx
                                x       x
                                x  APP  x
                                xxxxxxxxx


                      ---Topology Service Interface---

                            xxxxxxxxxxxxxxxxx
                            x               x
                            x   Aggregator  x
                            x               x
                            xxxxxxxxxxxxxxxxx
                                   /|\
                   +----------------+---------------+
                   |                |               |
                   |                |               |
                   |                |               |
                  \|/              \|/             \|/
               xxxxxxxxx       xxxxxxxxxx       xxxxxxxxx
               x       x       x        x       x       x
               x  TS   x       x   TS   x       x  TS   x
               xxxxxxxxx       xxxxxxxxxx ....  xxxxxxxxx
                  /|\              /|\             /|\
                   |                |               |
                   |                |          +----+---+
              +----+----+           |          |        |
              |         |           |          |        |
              |         |           |          |        |
         xxxxx|xxxxxxxxx|xxxxxxxxxxx|xxxxxxxxxx|xxxxxxxx|xxxxxx
         x    |         |           |          |        |     x
         x xxx|xxx    xx|xxxx    xxx|xxx    xxx|xxx  xxxxxxx  x
         x x TA  x    x TA  x    x TA  x    x TA  x  x TA  x  x
         x x  R  x    x  R  x    x  R  x    x  R  x  x  R  x  x
         x xxxxxxx....xxxxxxx....xxxxxxx....xxxxxxx..xxxxxxx  x
         x                                                    x
         x                                Physical Network    x
         xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx


        APP  ---- Application
        TS   ---- Topology Server
        TA   ---- Topology Agent
```

Figure 1: Framework of NTS

The entities used in this framework are:

Topology Agent: A logical entity located in network devices like switches, routers, etc. It is responsible for reporting the topology information produced in some protocol-specific ways and network changes, event, or states to its topology server. One topology agent can only be controlled by one topology server to avoid global network topology duplicating.

Topology Server: A server that collects topology information from physical network for a subset of devices, analyses, abstracts, and stores the subset topology information in a protocol independent way. Usually, a large network is too hard for a single topology server to handle. Thus, multiple topology servers are considered in this framework. Each of them is only responsible for a part of the global network. Topology server has the ability to calculate the special topology or the most optimized path based on specific algorithms from applications. Different algorithms may lead to different results.

Aggregator: A server that maintains a sketch topology information among all topology servers. It does not perceive any detailed subset topology information as individual topology servers. This entity is only responsible for generating and maintaining relationship among different topology servers, and calculating the final topology or optimized path based on the results calculated from some or all of the topology servers.

Application: It represents network applications like OSS, and third-party applications that require to use network topology service.

The interfaces needed in this framework:

Interface between topology agent and topology server: An interface that can be used by TA to report different protocol-specific topology information, e.g., BGP/OSPF/IS-IS,or SDN OpenFlow, to TS. Besides, TA can use it to notify TS the changes, states, and events.

Interface between topology server and aggregator: Communication between topology servers and aggregator. It includes topology servers reporting their ingress and egress information to the aggregator, aggregator conveying applications’ local topology algorithms information to topology servers, and topology servers returning their calculation results based on the local topology algorithms from applications to the aggregator.

Topology Service Interface: This interface is used by applications to communicate with the aggregator on path computation requesting and topology information obtaining. Applications use the interface to insert their own algorithms and requirements for the network topology service system to do some application specific calculations. Two kinds of algorithms are considered here: One for a topology server to calculate the special topology or the most optimized path based on its subset topology information (local topology algorithm); The other for the aggregator to calculate the global and the special topology or the most optimized path based on the results from all topology servers and the sketch topology information generated by the aggregator (global topology algorithm).

4. Topology Managing of NTS Framework

In NTS Framework, there are two level topologis: One is the generic network topology that is managed by TS; Other is managed by Aggregator.

TA discovers network topology and states/events, then reports the protocol-specific network topology to TS. TS analyses the network topology information, and constructs a generic network topology. TS stores the generic network topology in a protocol independent way. TA also can generate an abstract topology basing the generic network topology by partition or aggregation to avoid giving the detail network information to Application. TS also reports the ingress and egress information of its managing subnet network to the Aggregator.

Aggregator generates a sketch topology information reflecting the relationship among all the TSes. In a sketch topology, each subset network managed by a TS is condensed into a node. Aggregation may be give the sketch topology to Application to assist the path computing

5. Topology Inquiring of NTS Framework

Any Application can inquiry a special network topology, for example Network Map or Cost Map in ALTO service, from the NTS framework. The detailed steps of topology inquiring is listed as following:

* Application inputs local topology algorithm and global topology algorithm the aggregator to request a special topology.

* Aggregator instructs all (or some relevant) TSes to calculate their own special topology in their subset topologies based on the local topology algorithm of the application.

* TS independently calculates the special topology in its subset

topology, and reports its result to the aggregator.

* Aggregator calculates the final global topology based on the results of all the TSes, the sketch topology information, and the global topology algorithm, then Aggregator returns the final topology to Application.

6. Path Computing of NTS Framework

In SDN network, applications without knowledge of physical network can be benefit from the NTS framework to obtain the most suitable and efficient network path based on which they can then do some programming.

The detailed steps of path computing is listed as following:

* Application inputs local topology algorithm, global topology algorithm, source information and destination information to the aggregator to request an optimized path.

* (Optional) Aggregator gives the sketch topology to Application.

* (Optional) With the sketch topology, Application decides whether or not to filter some irrelevant-like TSes that maybe not appear in the end to end network path. If so, Application inputs the filtering algorithm to Aggregator.

* (Optional) Aggregator uses the filtering algorithm to filter some irrelevant-like TSes.

* Aggregator instructs all (or some relevant) TSes to calculate their own optimized path in their subset topologies based on the local topology algorithm of the application.

* TS independently calculates the optimized path in its subset topology, and reports its result to the aggregator.

* Aggregator calculates the final global optimized path based on the results of all the TSes, the sketch topology information, and the global topology algorithm, then Aggregator returns the final global optimized path to Application.

When the number of TSes increasing, the performance of this framework maybe reduced as all of the TSes need to do calculation. This can be solved by applications inputting another algorithm or requirement to allow the aggregator filtering some irrelevant-like TSes before sending any instructions to TSes. Thus, only those TSes which are responsible for the network topology between the end-to-end network path are required to do the calculation. However, this function is optional here since some applications may need to all of the TSes to take part in the calculation.

7. Relationship with Other Existing IETF work

7.1. I2RS

I2RS is discussing a generic topology data model. However, current I2RS charter says it is not responsible to develop protocols, encoding languages, or data models. The topology work in I2RS can be considered to use in the interface between TA and TS. However, I2RS will not discuss a detailed topology service. The protocols and data models produced in I2RS can be considered in this work.

7.2. PCE

PCE is discussing to specify the required protocols so as to compute of paths for MPLS and GMPLS Point to Point and Point to Multi-point Traffic Engineered LSPs. PCE does not consider to manage the network topology from multiple protocols. NTS framework can provides more flexible path computing algorithm and topology information inquiring.

8. Security Considerations

TBD.

9. IANA Considerations

This document does not require any IANA creations or modifications.

10. Acknowledgments

TBD.

11. References

11.1. Normative References

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

11.2. Informative References

Authors' Addresses

Rachel Huang
Huawei
101 Software Avenue, Yuhua District
Nanjing 210012
China

EMail: rachel.huang@huawei.com

Huaming Guo
China Academy of Information and Communications Technology
36 A Nanlishi Road, Xicheng District
Beijing 10037
China

EMail: guohuaming@caict.ac

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

EMail: yry@cs.yale.edu

ALTO
Internet-Draft
Intended status: Informational
Expires: September 3, 2015

M. Stiemerling
NEC Europe Ltd.
S. Kiesel
University of Stuttgart
S. Previdi
Cisco
M. Scharf
Alcatel-Lucent Bell Labs
March 2, 2015

ALTO Deployment Considerations
draft-ietf-alto-deployments-11

Abstract

Many Internet applications are used to access resources such as pieces of information or server processes that are available in several equivalent replicas on different hosts. This includes, but is not limited to, peer-to-peer file sharing applications. 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, which are able to provide a desired resource. This memo discusses deployment related issues of ALTO. It addresses different use cases of ALTO such as peer-to-peer file sharing and CDNs and presents corresponding examples. The document also includes recommendations for network administrators and application designers planning to deploy ALTO.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Many Internet applications are used to access resources such as pieces of information or server processes that are available in several equivalent replicas on different hosts. This includes, but is not limited to, peer-to-peer (P2P) file sharing applications and Content Delivery Networks (CDNs). 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, which are able to provide a desired resource. The basic ideas and problem space of ALTO is described in [RFC5693] and the set of requirements is discussed in [RFC6708]. The ALTO protocol is specified in [RFC7285]. An ALTO server discovery procedure is defined in [RFC7286].

This document discusses use cases and operational issues that can be expected when ALTO gets deployed. This includes, but is not limited to, location of the ALTO server, imposed load to the ALTO server, or from whom the queries are performed. The document also provides guidance which ALTO services to use, and it summarizes known challenges. It thereby complements the management considerations in the protocol specification [RFC7285], which are independent of any specific use of ALTO.

2. General Considerations

2.1. ALTO Entities

2.1.1. Baseline Scenario

The ALTO protocol [RFC7285] is a client/server protocol, operating between a number of ALTO clients and an ALTO server, as sketched in Figure 1.

```
                 +----------+
                 |  ALTO    |
                 |  Server  |
                 +----------+
                       ^
                _.-----|------.
            ,-''       |       `--.
          ,'           |           `.
         (     Network |             )
          `.           |           ,'
            `--.       |       _.-'
                `------|-----''
                       v
    +----------+  +----------+   +----------+
    |  ALTO    |  |  ALTO    |...|  ALTO    |
    |  Client  |  |  Client  |   |  Client  |
    +----------+  +----------+   +----------+
```

Figure 1: Baseline deployment scenario of the ALTO protocol

This document uses the terminology introduced in [RFC5693]. In particular, the following terms are defined by [RFC5693]:

o ALTO Service: Several resource providers may be able to provide the same resource. The ALTO service gives guidance to a resource consumer and/or resource directory about which resource provider(s) to select in order to optimize the client’s performance or quality of experience, while improving resource consumption in the underlying network infrastructure.

- o ALTO Server: A logical entity that provides interfaces to the queries to the ALTO service.

- o ALTO Client: The logical entity that sends ALTO queries. Depending on the architecture of the application, one may embed it in the resource consumer and/or in the resource directory.

According to that definition, both an ALTO server and an ALTO client are logical entities. An ALTO service may be offered by more than one ALTO servers. In ALTO deployments, the functionality of an ALTO server can therefore be realized by several server instances, e.g., by using load balancing between different physical servers. The term ALTO server should not be confused with use of a single physical server.

2.1.2. Placement of ALTO Entities

The ALTO server and ALTO clients can be situated at various entities in a network deployment. The first differentiation is whether the ALTO client is located on the actual host that runs the application, as shown in Figure 2, or if the ALTO client is located on a resource directory, as shown in Figure 3.

```
                                                  +-----+
                                             =====|     |**
                                         ====     +-----+  *
                                     ====            *     *
                                 ====                *     *
        +-----+     +------+=====                 +-----+  *
        |     |.....|      |======================|     |  *
        +-----+     +------+=====                 +-----+  *
      Source of      ALTO        ====                *     *
      topological    service         ====            *     *
      information                        ====     +-----+  *
                                             =====|     |**
                                                  +-----+

      Legend:
      === ALTO protocol
      *** Application protocol
      ... Provisioning protocol
```

Figure 2: Overview of protocol interaction between ALTO elements without a resource directory

Figure 2 shows the operational model for an ALTO client running at endpoints. An example would be a peer-to-peer file sharing application that does not use a tracker, such as edonkey. In addition, ALTO clients at peers could also be used in a similar way even if there is a tracker, as further discussed in Section 4.1.2.

```
                                                  +-----+
                                                **|     |**
                                              **  +-----+  *
                                            **       *     *
                                          **         *     *
        +-----+     +------+     +-----+**        +-----+  *
        |     |.....|      |=====|     |**********|     |  *
        +-----+     +------+     +-----+**        +-----+  *
      Source of      ALTO        Resource **         *     *
      topological    service     directory  **       *     *
      information                             **  +-----+  *
                                                **|     |**
                                                  +-----+

      Legend:
      === ALTO protocol
      *** Application protocol
      ... Provisioning protocol
```

Figure 3: Overview of protocol interaction between ALTO elements with a resource directory

In Figure 3, a use case with a resource directory is illustrated, e.g., a tracker in peer-to-peer file-sharing. Both deployment scenarios may differ in the number of ALTO clients that access an ALTO service: If ALTO clients are implemented in a resource directory, ALTO servers may be accessed by a limited and less dynamic set of clients, whereas in the general case any host could be an ALTO client. This use case is further detailed in Section 4.

Using ALTO in CDNs may be similar to a resource directory [I-D.jenkins-alto-cdn-use-cases]. The ALTO server can also be queried by CDN entities to get guidance about where the a particular client accessing data in the CDN is exactly located in the Internet Service Provider’s network, as discussed in Section 5.

2.2. Classification of Deployment Scenarios

2.2.1. Roles in ALTO Deployments

ALTO is a general-purpose protocol and it is intended to be used by a wide range of applications. This implies that there are different possibilities where the ALTO entities are actually located, i.e., if the ALTO clients and the ALTO server are in the same Internet Service Provider (ISP) domain, or if the clients and the ALTO server are managed/owned/located in different domains.

An ALTO service includes four types of entities:

1. Source of topological information

2. ALTO server

3. ALTO client

4. Resource consumer (using the ALTO guidance)

Each of these entities corresponds to a certain role, which results in requirements and constraints on the interaction between the entities.

A key design objective of the ALTO service is that each these four roles can be separated, i.e., they can be realized by different organizations or disjoint system components. ALTO is inherently designed for use in multi-domain environments. Most importantly, ALTO is designed to enable deployments in which the ALTO server and the ALTO client are not located within the same administrative domain.

As explained in [RFC5693], from this follows that at least three different kinds of entities can operate an ALTO server:

1. Network operators. Network Service Providers (NSPs) such as Internet Service Providers (ISPs) may have detailed knowledge of their network topology and policies. In this case, the source of the topology information and the provider of the ALTO server may be part of the same organization.

2. Third parties. Topology information could also be collected by entities separate from network operators but that may either have collected network information or have arrangements with network operators to learn the network information. Examples of such entities could be Content Delivery Network (CDN) operators or companies specialized on offering ALTO services on behalf of ISPs.

3. User communities. User communities could run distributed measurements for estimating the topology of the Internet. In this case the topology information may not originate from ISP data.

Regarding the interaction between ALTO server and client, ALTO deployments can be differentiated e.g. according to the following aspects:

1. Applicable trust model: The deployment of ALTO can differ depending on whether ALTO client and ALTO server are operated within the same organization and/or network, or not. This affects a lot of constraints, because the trust model is very different. For instance, as discussed later in this memo, the level-of-detail of maps can depend on who the involved parties actually are.

2. Size of user group: The main use case of ALTO is to provide guidance to any Internet application. However, an operator of an ALTO server could also decide to only offer guidance to a set of well-known ALTO clients, e. g., after authentication and authorization. In the peer-to-peer application use case, this could imply that only selected trackers are allowed to access the ALTO server. The security implications of using ALTO in closed groups differ from the public Internet.

3. Covered destinations: In general, an ALTO server has to be able to provide guidance for all potential destinations. Yet, in practice a given ALTO client may only be interested in a subset of destinations, e.g., only in the network cost between a limited set of resource providers. For instance, CDN optimization may not need the full ALTO cost maps, because traffic between individual residential users is not in scope. This may imply that an ALTO server only has to provide the costs that matter for a given user, e. g., by customized maps.

The following sections enumerate different classes of use cases for ALTO, and they discuss deployment implications of each of them. An ALTO server can in principle be operated by any organization, and there is no requirement that an ALTO server is deployed and operated by ISPs. Yet, since the ALTO solution is designed for ISPs, most examples in this document assume that the operator of an ALTO server is a network operator (e.g., an ISP or the network department in a large enterprise) that offers ALTO guidance in particular to users if this network.

It must be emphasized that any application using ALTO must also work if no ALTO servers can be found or if no responses to ALTO queries are received, e.g., due to connectivity problems or overload situations (see also [RFC6708]).

2.2.2. Information Exposure

An ALTO server stores information about preferences (e.g., for IP address ranges) and ALTO clients can retrieve these preferences. There are basically two different approaches on where the preferences are actually processed:

1. The ALTO server has a list of preferences and clients can retrieve this list via the ALTO protocol. This preference list can partially be updated by the server. The actual processing of the data is done on the client and thus there is no data of the client's operation revealed to the ALTO server.

2. The ALTO server has a list of preferences or preferences calculated during runtime and the ALTO client is sending information of its operation (e.g., a list of IP addresses) to the server. The server is using this operational information to determine its preferences and returns these preferences (e.g., a sorted list of the IP addresses) back to the ALTO client.

Approach 1 has the advantage (seen from the client) that all operational information stays within the client and is not revealed to the provider of the server. On the other hand, approach 1 requires that the provider of the ALTO server, i.e., the network operator, reveals information about its network structure (e.g., IP ranges or topology information in general) to the ALTO client. The ALTO protocol supports this scheme by the Network and Cost Map Service.

Approach 2 has the advantage (seen from the operator) that all operational information stays with the ALTO server and is not revealed to the ALTO client. On the other hand, approach 2 requires that the clients send their operational information to the server. This approach is realized by the ALTO Endpoint Cost Service (ECS).

Both approaches have their pros and cons, as further detailed in Section 3.3.

2.2.3. More Advanced Deployments

From an ALTO client's perspective, there are different ways to use ALTO:

1. Single service instance with single metric guidance: An ALTO client only obtains guidance regarding a single metric from a

single ALTO service, e.g., an ALTO server that is offered by the network service provider of the corresponding access network. Corresponding ALTO server instances can be discovered e.g. by ALTO server discovery [RFC7286] [I-D.kiesel-alto-xdom-disc]. Being a REST-ful protocol, an ALTO service can use known methods to balance the load between different server instances or between clusters of servers, i.e., an ALTO server can be realized by many instances with a load balancing scheme. The ALTO protocol also supports the use of different URIs for different ALTO features.

2. Single service instance with multiple metric guidance: An ALTO client could also query an ALTO service for different kinds of information, e.g., cost maps with different metrics. The ALTO protocol is extensible and permits such operation. However, ALTO does not define how a client shall deal with different forms of guidance, and it is up to the client to determine what provided information may indeed be useful.

3. Multiple service offers: An ALTO client can also decide to access multiple ALTO servers providing guidance, possibly from different operators or organizations. Each of these services may only offer partial guidance, e.g., for a certain network partition. In that case, it may be difficult for an ALTO client to compare the guidance from different services. Different organization may use different methods to determine maps, and they may also have different (possibly even contradicting or competing) guidance objectives. How to discover multiple ALTO servers and how to deal with conflicting guidance is an open issue.

There are also different options regarding the guidance offered by an ALTO service:

1. Authoritative servers: An ALTO server instance can provide guidance for all destinations for all kinds of ALTO clients.

2. Cascaded servers: An ALTO server may itself include an ALTO client and query other ALTO servers, e.g., for certain destinations. This results is a cascaded deployment of ALTO servers, as further explained below.

3. Inter-server synchronization: Different ALTO servers my communicate by other means. This approach is not further discussed in this document.

An assumption of the ALTO design is that ISP operate ALTO servers independently, irrespectively of other ISPs. This may true for most envisioned deployments of ALTO but there may be certain deployments that may have different settings. Figure 4 shows such setting with a university network that is connected to two upstream providers. NREN is a National Research and Education Network and ISP is a commercial upstream provider to this university network. The university, as well as ISP, are operating their own ALTO server. The ALTO clients, located on the peers will contact the ALTO server located at the university.

```
         +-----------+
         |    ISP    |
         |   ALTO    |
         |  Server   |
         +----------=+
            ,-------=            ,------.
         ,-'        =`-.      ,-'         `-.
        /   Upstream=   \    /   Upstream    \
       (       ISP  =    )  (       NREN      )
        \           =   /    \               /
         `-.        =,-'      `-.         ,-'
            `---+---=            `+------'
                |   =             |
                |   =======================
                |,-------------.  |       =
              ,-+               `-+    +-----------+
            ,'      University     `.  |University |
           (        Network          ) |   ALTO    |
            `.  =======================|  Server   |
              `-=               +-'    +-----------+
                =`+------------'|
                = |             |
         +--------+-+         +-+--------+
         |   Peer1  |         |   PeerN  |
         +----------+         +----------+
```

Figure 4: Example of a cascaded ALTO server

In this setting all "destinations" useful for the peers within NREN are free-of-charge for the peers located in the university network (i.e., they are preferred in the rating of the ALTO server). However, all traffic that is not towards NREN will be handled by the ISP upstream provider. Therefore, the ALTO server at the university may also include the guidance given by the ISP ALTO server in its replies to the ALTO clients. This is an example for cascaded ALTO servers.

3. Deployment Considerations by ISPs

3.1. Objectives for the Guidance to Applications

3.1.1. General Objectives for Traffic Optimization

The Internet consists of many networks. The networks are operated by Network Service Providers (NSP) or Internet Service Providers (ISP), which also include e.g. universities, enterprises, or other organizations. The Internet provides network connectivity e.g. by access networks, such as cable networks, xDSL networks, 3G/4G mobile networks, etc. Network operators need to manage, to control and to audit the traffic. Therefore, it is important to understand how to deploy an ALTO service and its expected impact.

The general objective of ALTO is to give guidance to applications on what endpoints (e.g., IP addresses or IP prefixes) are to be preferred according to the operator of the ALTO server. The ALTO protocol gives means to let the ALTO server operator express its preference, whatever this preference is.

ALTO enables ISPs to support application-level traffic engineering by influencing application resource selections. This traffic engineering for overlay formed by the application can have different objectives:

1. Inter-network traffic localization: ALTO can help to reduce inter-domain traffic. The networks of ISPs are connected through peering points. From a business view, the inter-network settlement is needed for exchanging traffic between these networks. These peering agreements can be costly. To reduce these costs, a simple objective is to decrease the traffic exchange across the peering points and thus keep the traffic in the own network or Autonomous System (AS) as far as possible.

2. Intra-network traffic localization: In case of large ISPs, the network may be grouped into several networks, domains, or Autonomous Systems (ASs). The core network includes one or several backbone networks, which are connected to multiple aggregation, metro, and access networks. If traffic can be limited to certain areas such as access networks, this decreases the usage of backbone and thus helps to save resources and costs.

3. Network off-loading: Compared to fixed networks, mobile networks have some special characteristics, including smaller link bandwidth, high cost, limited radio frequency resource, and limited terminal battery. In mobile networks, wireless links should be used efficiently. For example, in the case of a P2P

service, it is likely that hosts in fixed networks should avoid retrieving data from hosts in mobile networks, and hosts in mobile networks should prefer retrieval of data from hosts in fixed networks.

4. Application tuning: ALTO is also a tool to optimize the performance of applications that depend on the network and perform resource selection decisions among network endpoints. And example is the network-aware selection of Content Delivery Network (CDN) caches.

In the following, these objectives are explained in more detail with examples.

3.1.2. Inter-Network Traffic Localization

ALTO guidance can be used to keep traffic local in a network. An ALTO server can let applications prefer other hosts within the same network operator’s network instead of randomly connecting to other hosts that are located in another operator’s network. Here, a network operator would always express its preference for hosts in its own network, while hosts located outside its own network are to be avoided (i.e., they are undesired to be considered by the applications). Figure 5 shows such a scenario where hosts prefer hosts in the same network (e.g., Host 1 and Host 2 in ISP1 and Host 3 and Host 4 in ISP2).

```
                            ,-------.         +-----------+
          ,---.          ,-’         ‘-.      |   Host 1  |
       ,-’     ‘-.      /     ISP 1   ########|ALTO Client|
      /           \    /              #  \    +-----------+
     /    ISP X    \   |              #  |    +-----------+
    /               \  \              ########|   Host 2  |
   ;             +----------------------------|ALTO Client|
   |             |   |   ‘-.         ,-’      +-----------+
   |             |   |      ‘-------’
   |             |   |      ,-------.         +-----------+
   :             |   ;   ,-’         ‘########|   Host 3  |
    \            |  /   /     ISP 2   # \     |ALTO Client|
     \           | /   /              #  \    +-----------+
      \          +---------+          #  |    +-----------+
       ‘-.     ,-’     \   |          ########|   Host 4  |
          ‘---’         \  +------------------|ALTO Client|
                         ‘-.         ,-’      +-----------+
                            ‘-------’

       Legend:
       ### preferred "connections"
       --- non-preferred "connections"
```

Figure 5: Inter-network traffic localization

Examples for corresponding ALTO maps can be found in Section 3.5. Depending on the application characteristics, it may not be possible or even not be desirable to completely localize all traffic.

3.1.3. Intra-Network Traffic Localization

The above sections described the results of the ALTO guidance on an inter-network level. However, ALTO can also be used for intra-network localization. In this case, ALTO provides guidance which internal hosts are to be preferred inside a single network or, e.g., one AS. Figure 6 shows such a scenario where Host 1 and Host 2 are located in Net 2 of ISP1 and connect via a low capacity link to the core (Net 1) of the same ISP1. If Host 1 and Host 2 exchange their data with remote hosts, they would probably congest the bottleneck link.

```
                               ,-------.         +-----------+
          ,---.             ,-'         `-.      |   Host 1  |
       ,-'     `-.         /     ISP 1  #########|ALTO Client|
      /           \       /      Net 2  #   \    +-----------+
     /    ISP 1    \      |     #########   |    +-----------+
    /     Net 1     \     \     #           /    |   Host 2  |
   ;             ###;      \    #      ##########|ALTO Client|
   |               X~~~~~~~~~~~~X#######,-'      +-----------+
   |             ### |  ^      `-------'
   |                 |  |
   :                 ;  |
    \               /  Bottleneck
     \             /
      \           /
       `-.     ,-'
          `---'
       Legend:
       ### peer "connections"
       ~~~ bottleneck link
```

         Figure 6: Without intra-network ALTO traffic localization

The operator can guide the hosts in such a situation to try first local hosts in the same network islands, avoiding or at least lowering the effect on the bottleneck link, as shown in Figure 7.

```
                               ,-------.         +-----------+
          ,---.             ,-'         `-.      |   Peer 1  |
       ,-'     `-.         /     ISP 1  #########|ALTO Client|
      /           \       /      Net 2  #   \    +-----------+
     /    ISP 1    \      |             #   |    +-----------+
    /     Net 1     \     \             #########|   Peer 2  |
   ;                ;      \           ##########|ALTO Client|
   |                #~~~~~~~~~~~########,-'      +-----------+
   |             ### |  ^      `-------'
   |                 |  |
   :                 ;  |
    \               /  Bottleneck
     \             /
      \           /
       `-.     ,-'
          `---'
       Legend:
       ### peer "connections"
       ~~~ bottleneck link
```

          Figure 7: With intra-network ALTO traffic localization

The objective here is to avoid bottlenecks by optimized endpoint selection at application level. ALTO is not a method to deal with the congestion at the bottleneck.

3.1.4. Network Off-Loading

Another scenario is off-loading traffic from networks. This use of ALTO can be beneficial in particular in mobile networks. The network operator may have the desire to guide hosts in its own network to use hosts in remote networks. One reason can be that the wireless network is not made for the load cause by, e.g., peer-to-peer applications, and the operator has the need that peers fetch their data from remote peers in other parts of the Internet.

```
                            ,-------.         +-----------+
          ,---.          ,-'         `-.      |   Host 1  |
       ,-'     `-.      /     ISP 1   +-------|ALTO Client|
      /           \    /              |  \    +-----------+
     /    ISP X    \   |              |  |    +-----------+
    /               \  \              +-------|   Host 2  |
   ;             #-###########################|ALTO Client|
   |             #   |   `-.         ,-'      +-----------+
   |             #   |      `-------'
   |             #   |      ,-------.         +-----------+
   :             #   ;   ,-'         `+-------|   Host 3  |
    \            #  /   /     ISP 2   | \     |ALTO Client|
     \           # /   /              |  \    +-----------+
      \          ###########          |  |    +-----------+
       `-.     ,-'     \   #          +-------|   Host 4  |
          `---'         \  ###################|ALTO Client|
                         `-.         ,-'      +-----------+
                            `-------'

       Legend:
       === preferred "connections"
       --- non-preferred "connections"
```

Figure 8: ALTO traffic network de-localization

Figure 8 shows the result of such a guidance process where Host 2 prefers a connection with Host 4 instead of Host 1, as shown in Figure 5.

A realization of this scenario may have certain limitations and may not be possible in all cases. For instance, it may require that the ALTO server can distinguish mobile and non-mobile hosts, e.g., based on their IP address. This may depend on mobility solutions and may not be possible or accurate. In general, ALTO is not intended as a fine-grained traffic engineering solution for individual hosts. Instead, it typically works on aggregates (e.g., if it is known that certain IP prefixes are often assigned to mobile users).

3.1.5. Application Tuning

ALTO can also provide guidance to optimize the application-level topology of networked applications, e.g., by exposing network performance information. Applications can often run own measurements to determine network performance, e.g., by active delay measurements or bandwidth probing, but such measurements result in overhead and complexity. Accessing an ALTO server can be a simpler alternative. In addition, an ALTO server may also expose network information that applications cannot easily measure or reverse-engineer.

3.2. Provisioning of ALTO Topology Data

3.2.1. Data Sources

An ALTO server can collect topological information from a variety of sources in the network and provides a cohesive, abstracted view of the network topology to applications using an ALTO client. Sources that may include routing protocols, network policies, state and performance information, geo-location, etc. Based on the input, the ALTO server builds an ALTO-specific network topology that represents the network as it should be understood and utilized by applications (resource consumers) at endpoints using ALTO services (e.g., Network/ Cost Map Service or ECS).

The ALTO protocol does not assume a specific network topology. In principle, ALTO can be used with various types of addresses (Endpoint Addresses). [RFC7285] defines the use of IPv4/IPv6 addresses or prefixes in ALTO, but further address types could be added by extensions. In this document, only the use of IPv4/IPv6 addresses is considered.

The exposure of network topology information is controlled and managed by the ALTO server. ALTO abstract network topologies can be automatically generated from the physical or logical topology of the network. The generation would typically be based on policies and rules set by the network operator. The maps and the guidance can significantly differ depending on the use case, the network architecture, and the trust relationship between ALTO server and ALTO client, etc. Besides the security requirements that consist of not delivering any confidential or critical information about the infrastructure, there are efficiency requirements in terms of what aspects of the network are visible and required by the given use case and/or application.

The ALTO server operator has to ensure that the ALTO topology does not contain any details that would endanger the network integrity and security. For instance, ALTO is not intended to leak raw Interior Gateway Protocol (IGP) or Border gateway Protocol (BGP) databases to ALTO clients.

```
        +--------+   +--------+
        |  ALTO  |   |  ALTO  |
        | Client |   | Client |
        +--------+   +--------+
               /\     /\
               ||     || ALTO protocol
               ||     ||
              +---------+
              |  ALTO   |
              | Server  |
              +---------+
               ^   ^   ^
               :   :   :
      +........+   :   +........+ Provisioning
      :            :            : protocol
      :            :            :
 +---------+  +---------+  +---------+
 |   BGP   |  |   I2RS  |  |   NMS   | Potential
 | Speaker |  |  Client |  |   OSS   | data sources
 +---------+  +---------+  +---------+
      ^            ^            ^
      |            |            |
 Link-State      I2RS      SNMP/NETCONF,
  NLRI for       data      traffic statistics,
  IGP/BGP                  IPFIX, etc.
```

Figure 9: Potential data sources for ALTO

As illustrated in Figure 9, the topology data used by an ALTO server can originate from different data sources:

o The document [I-D.ietf-idr-ls-distribution] describes a mechanism by which links state and traffic engineering information can be collected from networks and shared with external components using the BGP routing protocol. This is achieved using a new BGP Network Layer Reachability Information (NLRI) encoding format. The mechanism is applicable to physical and virtual IGP links and can also include Traffic Engineering (TE) data. For instance, prefix data can be carried and originated in BGP, while TE data is originated and carried in an IGP. The mechanism described is subject to policy control. An ALTO Server can also use other mechanisms to get network data, for example, peering with multiple IGP and BGP speakers.

- The Interface to the Routing System (I2RS) is a solution for state transfer in and out of the Internet’s routing system [I-D.ietf-i2rs-architecture]. An ALTO server could use an I2RS client to observe routing-related information.

- An ALTO server can also leverage a Network Management System (NMS) or an Operations Support System (OSS) as data sources. NMS or OSS solutions are used to control, operate, and manage a network, e.g., using the Simple Network Management Protocol (SNMP) or NETCONF. As explained for instance in [I-D.farrkingel-pce-abno-architecture], the NMS and OSS can be consumers of network events reported and can act on these reports as well as displaying them to users and raising alarms. The NMS and OSS can also access the Traffic Engineering Database (TED) and Label Switched Path Database (LSP-DB) to show the users the current state of the network. In addition, NMS and OSS systems may have access to IGP/BGP routing information, network inventory data (e.g., links, nodes, or link properties not visible to routing protocols, such as Shared Risk Link Groups), statistics collection system that provides traffic information, such as traffic demands or link utilization obtained from IP Flow Information Export (IPFIX), as well as other Operations, Administration, and Maintenance (OAM) information (e.g., syslog). NMS or OSS systems also may have functions to correlate and orchestrate information originating from other data sources. For instance, it could be required to correlate IP prefixes with routers (Provider, Provider Edge, Customer Edge, etc.), IGP areas, VLAN IDs, or policies.

3.2.2. Privacy Requirements

Providing ALTO guidance can result in a win-win situation both for network providers and users of the ALTO information. Applications possibly get a better performance, while the network provider has means to optimize the traffic engineering and thus its costs. Yet, there can be security concerns with exposing topology data. Corresponding limitations are discussed in Section 7.2.

ISPs may have important privacy requirements when deploying ALTO. In particular, an ISP may not be willing to expose sensitive operational details of its network. The topology abstraction of ALTO enables an ISP to expose the network topology at a desired granularity only, determined by security policies.

With the Endpoint Cost Service (ECS), the ALTO client does not to have to implement any specific algorithm or mechanism in order to retrieve, maintain and process network topology information (of any kind). The complexity of the network topology (computation, maintenance and distribution) is kept in the ALTO server and ECS is delivered on demand. This allows the ALTO server to enhance and modify the way the topology information sources are used and combined. This simplifies the enforcement of privacy policies of the ISP.

The ALTO Network Map and Cost Map service expose an abstracted view on the ISP network topology. Therefore, in this case care is needed when constructing those maps in order to take into account privacy policies, as further discussed in Section 3.2.3. The ALTO protocol also supports further features such as endpoint properties, which could also be used to expose topology guidance. The privacy considerations for ALTO maps also apply to such ALTO extensions.

3.2.3. Partitioning and Grouping of IP Address Ranges

ALTO introduces provider-defined network location identifiers called Provider-defined Identifiers (PIDs) to aggregate network endpoints in the Map Services. Endpoints within one PID may be treated as single entity, assuming proximity based on network topology or other similarity. A key use case of PIDs is to specify network preferences (costs) between PIDs instead of individual endpoints. It is up to the operator of the ALTO server how to group endpoints and how to assign PIDs. For example, a PID may denote a subnet, a set of subnets, a metropolitan area, a POP, an autonomous system, or a set of autonomous systems.

This document only considers deployment scenarios in which PIDs expand to a set of IP address ranges (CIDR). A PID is characterized by a string identifier and its associated set of endpoint addresses [RFC7285]. If an ALTO server offers the Map Service, corresponding identifiers have to be configured.

An automated ALTO implementation may use dynamic algorithms to aggregate network topology. However, it is often desirable to have a mechanism through which the network operator can control the level and details of network aggregation based on a set of requirements and constraints. This will typically be governed by policies that enforce a certain level of abstraction and prevent leakage of sensitive operational data.

For instance, an ALTO server may leverage BGP information that is available in a networks service provider network layer and compute the group of prefix. An example are BGP communities, which are used in MPLS/IP networks as a common mechanism to aggregate and group prefixes. A BGP community is an attribute used to tag a prefix to group prefixes based on mostly any criteria (as an example, most ISP networks originate BGP prefixes with communities identifying the Point of Presence (PoP) where the prefix has been originated). These BGP communities could be used to map IP address ranges to PIDs. By an additional policy, the ALTO server operator may decide an arbitrary cost defined between groups. Alternatively, there are algorithms that allow a dynamic computation of cost between groups. The ALTO protocol itself is independent of such algorithms and policies.

3.2.4. Rating Criteria and/or Cost Calculation

An ALTO server indicates preferences amongst network locations in the form of path costs. Path costs are generic costs and can be internally computed by the operator of the ALTO server according to its own policy. For a given ALTO network map, an ALTO cost map defines directional path costs pairwise amongst the set of source and destination network locations defined by the PIDs.

The ALTO protocol permits the use of different cost types. An ALTO cost type is defined by the combination of a cost metric and a cost mode. The cost metric identifies what the costs represent. The cost mode identifies how the costs should be interpreted, e.g., whether returned costs should be interpreted as numerical values or ordinal rankings. The ALTO protocol also allows the definition of additional constraints defining which elements of a cost map shall be returned.

The ALTO protocol specification [RFC7285] defines the "routingcost" cost metric as basic set of rating criteria, which has to be supported by all implementations. This cost metric conveys a generic measure for the cost of routing traffic from a source to a destination. A lower value indicates a higher preference for traffic to be sent from a source to a destination. It is up to the ALTO server how that metric is calculated.

There is also an extension procedure for adding new ALTO cost types. The following list gives an overview on further rating criteria that have been proposed or which are in use by ALTO-related prototype implementations. This list is not intended as normative text; a definition of further metrics can be found for instance in [I-D.wu-alto-te-metrics]. Instead, the only purpose of the following list is to document and discuss rating criteria that have been proposed so far. It can also depend on the use case of ALTO whether such rating criteria are useful, and whether the corresponding information would indeed be made available by ISPs.

Distance-related rating criteria:

- Relative topological distance: The term relative means that a larger numerical value means greater distance, but it is up to the ALTO service how to compute the values, and the ALTO client will not be informed about the nature of the information. One way of generating this kind of information may be counting AS hops, but when querying this parameter, the ALTO client must not assume that the numbers actually are AS hops. In addition to the AS path, a relative cost value could also be calculated taking into account other routing protocol parameters, such as BGP local preference or multi-exit discriminator (MED) attributes.

- Absolute topological distance, expressed in the number of traversed autonomous systems (AS).

- Absolute topological distance, expressed in the number of router hops (i.e., how much the TTL value of an IP packet will be decreased during transit).

- Absolute physical distance, based on knowledge of the approximate geo-location (e.g., continent, country) of an IP address.

Performance-related rating criteria:

- The minimum achievable throughput between the resource consumer and the candidate resource provider, which is considered useful by the application (only in ALTO queries).

- An arbitrary upper bound for the throughput from/to the candidate resource provider (only in ALTO responses). This may be, but is not necessarily the provisioned access bandwidth of the candidate resource provider.

- The maximum round-trip time (RTT) between resource consumer and the candidate resource provider, which is acceptable for the application for useful communication with the candidate resource provider (only in ALTO queries).

- An arbitrary lower bound for the RTT between resource consumer and the candidate resource provider (only in ALTO responses). This may be, for example, based on measurements of the propagation delay in a completely unloaded network.

Charging-related rating criteria:

- Traffic volume caps, in case the Internet access of the resource consumer is not charged by "flat rate". For each candidate resource provider, the ALTO service could indicate the amount of data that may be transferred from/to this resource provider until a given point in time, and how much of this amount has already been consumed. Furthermore, it would have to be indicated how excess traffic would be handled (e.g., blocked, throttled, or charged separately at an indicated price). The interaction of several applications running on a host, out of which some use this criterion while others don’t, as well as the evaluation of this criterion in resource directories, which issue ALTO queries on behalf of other endpoints, are for further study.

- o Other metrics representing an abstract cost, e.g., determined by policies that distinguish "cheap" from "expensive" IP subnet ranges, e.g., without detailing the cost function.

These rating criteria are subject to the remarks below:

The ALTO client must be aware that with high probability the actual performance values differs from whatever an ALTO server exposes. In particular, an ALTO client must not consider a throughput parameter as a permission to send data at the indicated rate without using congestion control mechanisms.

The discrepancies are due to various reasons, including, but not limited to the facts that

- o the ALTO service is not an admission control system
- o the ALTO service may not know the instantaneous congestion status of the network
- o the ALTO service may not know all link bandwidths, i.e., where the bottleneck really is, and there may be shared bottlenecks
- o the ALTO service may not have all information about the actual routing
- o the ALTO service may not know whether the candidate endpoints itself is overloaded
- o the ALTO service may not know whether the candidate endpoints throttles the bandwidth it devotes for the considered application
- o the ALTO service may not know whether the candidate endpoints will throttle the data it sends to us (e.g., because of some fairness algorithm, such as tit-for-tat).

Because of these inaccuracies and the lack of complete, instantaneous state information, which are inherent to the ALTO service, the application must use other mechanisms (such as passive measurements on actual data transmissions) to assess the currently achievable throughput, and it must use appropriate congestion control mechanisms in order to avoid a congestion collapse. Nevertheless, these rating criteria may provide a useful shortcut for quickly excluding candidate resource providers from such probing, if it is known in advance that connectivity is in any case worse than what is considered the minimum useful value by the respective application.

Rating criteria that should not be defined for and used by the ALTO service include:

- o Performance metrics that are closely related to the instantaneous congestion status. The definition of alternate approaches for congestion control is explicitly out of the scope of ALTO. Instead, other appropriate means, such as using TCP based transport, have to be used to avoid congestion.

- o Performance metrics that raise privacy concerns. For instance, it has been questioned whether an ALTO service could publicly expose the provisioned access bandwidth, e.g. of cable / DSL customers, because this could enables identification of "premium" customers.

3.3. Known Limitations of ALTO

3.3.1. Limitations of Map-based Approaches

The specification of the Map Service in the ALTO protocol [RFC7285] is based on the concept of network maps. A network map partitions the network into Provider-defined Identifier (PID) that group one or multiple endpoints (e.g., subnetworks) to a single aggregate. The "costs" between the various PIDs is stored in a cost map. Map-based approaches lower the signaling load on the server as maps have to be retrieved only if they change.

One main assumption for map-based approaches is that the information provided in these maps is static for a longer period of time. This assumption is fine as long as the network operator does not change any parameter, e.g., routing within the network and to the upstream peers, IP address assignment stays stable (and thus the mapping to the partitions). However, there are several cases where this assumption is not valid:

1. ISPs reallocate IP subnets from time to time;

2. ISPs reallocate IP subnets on short notice;

3. IP prefix blocks may be assigned to a router that serves a variety of access networks;

4. Network costs between IP prefixes may change depending on the ISP’s routing and traffic engineering.

These effects can be explained as follows:

Case 1: ISPs may reallocate IP subnets within their infrastructure from time to time, partly to ensure the efficient usage of IPv4 addresses (a scarce resource), and partly to enable efficient route tables within their network routers. The frequency of these "renumbering events" depend on the growth in number of subscribers and the availability of address space within the ISP. As a result, a subscriber’s household device could retain an IP address for as short as a few minutes, or for months at a time or even longer.

It has been suggested that ISPs providing ALTO services could sub-divide their subscribers’ devices into different IP subnets (or certain IP address ranges) based on the purchased service tier, as well as based on the location in the network topology. The problem is that this sub-allocation of IP subnets tends to decrease the efficiency of IP address allocation, in particular for IPv4. A growing ISP that needs to maintain high efficiency of IP address utilization may be reluctant to jeopardize their future acquisition of IP address space.

However, this is not an issue for map-based approaches if changes are applied in the order of days.

Case 2: ISPs can use techniques that allow the reallocation of IP prefixes on very short notice, i.e., within minutes. An IP prefix that has no IP address assignment to a host anymore can be reallocated to areas where there is currently a high demand for IP addresses.

Case 3: In residential access networks (e.g., DSL, cable), IP prefixes are assigned to broadband gateways, which are the first IP-hop in the access-network between the Customer Premises Equipment (CPE) and the Internet. The access-network between CPE and broadband gateway (called aggregation network) can have varying characteristics (and thus associated costs), but still using the same IP prefix. For instance one IP addresses IP11 out of a IP prefix IP1 can be assigned to a VDSL (e.g., 2 MBit/s uplink) access line while the subsequent IP address IP12 is assigned to a slow ADSL line (e.g., 128 kbit/s uplink). These IP addresses are assigned on a first come first served basis, i.e., a single IP address out of the same IP prefix can change its associated costs quite fast. This may not be an issue with respect to the used upstream provider (thus the cross ISP traffic) but depending on the capacity of the aggregation-network this may raise to an issue.

Case 4: The routing and traffic engineering inside an ISP network, as well as the peering with other autonomous systems, can change dynamically and affect the information exposed by an ALTO server. As a result, cost map and possibly also network maps can change.

3.3.2. Limitations of Non-Map-based Approaches

The specification of the ALTO protocol [RFC7285] also includes the Endpoint Cost Service (ECS) mechanism. ALTO clients can ask guidance for specific IP addresses to the ALTO server, thereby avoiding the need of processing maps. This can mitigate some of the problems mentioned in the previous section.

However, asking for IP addresses, asking with long lists of IP addresses, and asking quite frequently may overload the ALTO server. The server has to rank each received IP address, which causes load at the server. This may be amplified by the fact that not only a single ALTO client is asking for guidance, but a larger number of them. The results of the ECS are also more difficult to cache than ALTO maps. Therefore, the ALTO client may have to await the server response before starting a communication, which results in an additional delay.

Caching of IP addresses at the ALTO client or the usage of the H12 approach [I-D.kiesel-alto-h12] in conjunction with caching may lower the query load on the ALTO server.

When ALTO server receives an ECS request, it may not have the most appropriate topology information in order to accurately determine the ranking. [RFC7285] generally assumes that a server can always offer some guidance. In such a case the ALTO server could adopt one of the following strategies:

o Reply with available information (best effort).

o Query another ALTO server presumed to have better topology information and return that response (cascaded servers).

o Redirect the request to another ALTO server presumed to have better topology information (redirection).

The protocol mechanisms and decision processes that would be used to determine if redirection is necessary and which mode to use is out of the scope of this document, since protocol extensions could be required.

3.3.3. General Limitations

ALTO is designed as a protocol between clients integrated in applications and servers that provide network information and guidance (e.g., basic network location structure and preferences of network paths). The objective is to modify network resource consumption patterns at application level while maintaining or improving application performance. This design focus results in a number of characteristics of ALTO:

- o Endpoint focus: In typical ALTO use cases, neither the consumer of the topology information (i.e., the ALTO client) nor the considered resources (e.g., files at endpoints) are part of the network. The ALTO server presents an abstract network topology containing only information relevant to an application overlay for better-than-random resource selections among its endpoints. The ALTO protocol specification [RFC7285] is not designed to expose network internals such as routing tables or configuration data that are not relevant for application-level resource selection decisions in network endpoints.

- o Abstraction: The ALTO services such as the Network/Cost Map Service or the ECS provide an abstract view of the network only. The operator of the ALTO server has full control over the granularity (e.g., by defining policies how to aggregate subnets into PIDs) and the level-of-detail of the abstract network representation (e.g., by deciding what cost types to support).

- o Multiple administrative domains: The ALTO protocol is designed for use cases where the ALTO server and client can be located in different organizations or trust domains. ALTO assumes a loose coupling between server and client. In addition, ALTO does not assume that an ALTO client has any a priori knowledge about the ALTO server and its supported features. An ALTO server can be discovered automatically.

- o Read-only: ALTO is a query/response protocol to retrieve guidance information. Neither network/cost map queries nor queries to the endpoint cost service are designed to affect state in the network.

If ALTO shall be deployed for use cases violating these assumptions, the protocol design may result in limitations.

For instance, in an Application-Based Network Operation (ABNO) environment the application could issue explicit service request to the network [I-D.farrkingel-pce-abno-architecture]. In this case, the application would require detailed knowledge about the internal network topology and the actual state. A network configuration would also require a corresponding security solution for authentication and authorization. ALTO is not designed for operations to control, operate, and manage a network.

Such deployments could be addressed by network management solutions, e.g., based on SNMP [RFC3411] or NETCONF [RFC6241] and YANG [RFC6020] that are typically designed to manipulate configuration state. Reference [I-D.farrkingel-pce-abno-architecture] contains a more detailed discussion of interfaces between components such as Element Management System (EMS), Network Management System (NMS), Operations Support System (OSS), Traffic Engineering Database (TED), Label Switched Path Database (LSP-DB), Path Computation Element (PCE), and other Operations, Administration, and Maintenance (OAM) components.

3.4. Monitoring ALTO

3.4.1. Impact and Observation on Network Operation

ALTO presents a new opportunity for managing network traffic by providing additional information to clients. In particular, the deployment of an ALTO Server may shift network traffic patterns, and the potential impact to network operation can be large. An ISP providing ALTO may want to assess the benefits of ALTO as part of the management and operations (cf. [RFC7285]). For instance, the ISP might be interested in understanding whether the provided ALTO maps are effective, and in order to decide whether an adjustment of the ALTO configuration would be useful. Such insight can be obtained from a monitoring infrastructure. An ISP offering ALTO could consider the impact on (or integration with) traffic engineering and the deployment of a monitoring service to observe the effects of ALTO operations. The measurement of impacts can be challenging because ALTO-enabled applications may not provide related information back to the ALTO Service Provider.

To construct an effective monitoring infrastructure, the ALTO Service Provider should decide how to monitor the performance of ALTO and identify and deploy data sources to collect data to compute the performance metrics. In certain trusted deployment environments, it may be possible to collect information directly from ALTO clients. It may also be possible to vary or selectively disable ALTO guidance for a portion of ALTO clients either by time, geographical region, or some other criteria to compare the network traffic characteristics with and without ALTO. Monitoring an ALTO service could also be realized by third parties. In this case, insight into ALTO data may require a trust relationship between the monitoring system operator and the network service provider offering an ALTO service.

The required monitoring depends on the network infrastructure and the use of ALTO, and an exhaustive description is outside the scope of this document.

3.4.2. Measurement of the Impact

ALTO realizes an interface between the network and applications. This implies that an effective monitoring infrastructure may have to deal with both network and application performance metrics. This document does not comprehensively list all performance metrics that could be relevant, nor does it formally specify metrics.

The impact of ALTO can be classified regarding a number of different criteria:

- Total amount and distribution of traffic: ALTO enables ISPs to influence and localize traffic of applications that use the ALTO service. An ISP may therefore be interested in analyzing the impact on the traffic, i.e., whether network traffic patterns are shifted. For instance, if ALTO shall be used to reduce the inter-domain P2P traffic, it makes sense to evaluate the total amount of inter-domain traffic of an ISP. Then, one possibility is to study how the introduction of ALTO reduces the total inter-domain traffic (inbound and/our outbound). If the ISPs intention is to localize the traffic inside his network, the network-internal traffic distribution will be of interest. Effectiveness of localization can be quantified in different ways, e.g., by the load on core routers and backbone links, or by considering more advanced effects, such as the average number of hops that traffic traverses inside a domain.

- Application performance: The objective of ALTO is improve application performance. ALTO can be used by very different types applications, with different communication characteristics and requirements. For instance, if ALTO guidance achieves traffic localization, one would expect that applications achieve a higher throughput and/or smaller delays to retrieve data. If application-specific performance characteristics (e.g., video or audio quality) can be monitored, such metrics related to user experience could also help to analyze the benefit of an ALTO deployment. If available, selected statistics from the TCP/IP stack in hosts could be leveraged, too.

Of potential interest can also be the share of applications or customers that actually use an offered ALTO service, i.e., the adoption of the service.

Monitoring statistics can be aggregated, averaged, and normalized in different ways. This document does not mandate specific ways how to calculate metrics.

3.4.3. System and Service Performance

A number of interesting parameters can be measured at the ALTO server. [RFC7285] suggests certain ALTO-specific metrics to be monitored:

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

This data characterizes the workload, the system performance as well as the map data. Obviously, such data will depend on the implementation and the actual deployment of the ALTO service. Logging is also recommended in [RFC7285].

3.4.4. Monitoring Infrastructures

Understanding the impact of ALTO may require interaction between different systems, operating at different layers. Some information discussed in the preceding sections is only visible to an ISP, while application-level performance can hardly be measured inside the network. It is possible that not all information of potential interest can directly be measured, either because no corresponding monitoring infrastructure or measurement method exists, or because it is not easily accessible.

One way to quantify the benefit of deploying ALTO is to measure before and after enabling the ALTO service. In addition to passive monitoring, some data could also be obtained by active measurements, but due to the resulting overhead, the latter should be used with care. Yet, in all monitoring activities an ALTO service provider has to take into account that ALTO clients are not bound to ALTO server guidance as ALTO is only one source of information, and any measurement result may thus be biased.

Potential sources for monitoring the use of ALTO include:

- o Network Operations, Administration, and Maintenance (OAM) systems: Many ISPs deploy OAM systems to monitor the network traffic, which may have insight into traffic volumes, network topology, and bandwidth information inside the management area. Data can be obtained by SNMP, NETCONF, IP Flow Information Export (IPFIX), syslog, etc.

- o Applications/clients: Relevant data could be obtained by instrumentation of applications.

- o ALTO server: If available, log files or other statistics data could be analyzed.

- o Other application entities: In several use cases, there are other application entities that could provide data as well. For instance, there may be centralized log servers that collect data.

In many ALTO use cases some data sources are located within an ISP network while some other data is gathered at application level. Correlation of data could require a collaboration agreement between the ISP and an application owner, including agreements of data interchange formats, methods of delivery, etc. In practice, such a collaboration may not be possible in all use cases of ALTO, because the monitoring data can be sensitive, and because the interacting entities may have different priorities. Details of how to build an over-arching monitoring system for evaluating the benefits of ALTO are outside the scope of this memo.

3.5. Map Examples for Different Types of ISPs

3.5.1. Small ISP with Single Internet Uplink

The ALTO protocol does not mandate how to determine costs between endpoints and/or determine map data. In complex usage scenarios this can be a non-trivial problem. In order to show the basic principle, this and the following sections explain for different deployment scenarios how ALTO maps could be structured.

For a small ISP, the inter-domain traffic optimizing problem is how to decrease the traffic exchanged with other ISPs, because of high settlement costs. By using the ALTO service to optimize traffic, a small ISP can define two "optimization areas": one is its own network; the other one consists of all other network destinations.

The cost map can be defined as follows: the cost of link between clients of inner ISP’s networks is lower than between clients of outer ISP’s networks and clients of inner ISP’s network. As a result, a host with ALTO client inside the network of this ISP will prefer retrieving data from hosts connected to the same ISP.

An example is given in Figure 10. It is assumed that ISP A is a small ISP only having one access network. As operator of the ALTO service, ISP A can define its network to be one optimization area, named as PID1, and define other networks to be the other optimization area, named as PID2. C1 is denoted as the cost inside the network of ISP A. C2 is denoted as the cost from PID2 to PID1, and C3 from PID1 to PID2. For the sake of simplicity, in the following C2=C3 is assumed. In order to keep traffic local inside ISP A, it makes sense to define: C1<C2

```
              -----------
          ////           \\\\
        //                   \\
      //                       \\                  /-----------\
     | +---------+               |             ////             \\\\
     | | ALTO    |  ISP A        |    C2      |    Other Networks   |
    |  | Service |  PID 1         <-----------     PID 2
     | +---------+  C1           |----------->|                     |
     |                           |  C3 (=C2)   \\\\             ////
      \\                       //                  \-----------/
        \\                   //
          \\\\           ////
              -----------
```

Figure 10: Example ALTO deployment in small ISPs

A simplified extract of the corresponding ALTO network and cost maps is listed in Figure 11 and Figure 12, assuming that the network of ISP A has the IPv4 address ranges 192.0.2.0/24 and 198.51.100.0/25. In this example, the cost values C1 and C2 can be set to any number C1<C2.

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

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

Figure 11: Example ALTO network map

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

{
    ...
    "cost-type" : {"cost-mode"  : "numerical",
                   "cost-metric": "routingcost"
    }
  },
  "cost-map" : {
    "PID1": { "PID1": C1,  "PID2": C2 },
    "PID2": { "PID1": C2,  "PID2": 0 },
  }
}
```

Figure 12: Example ALTO cost map

3.5.2. ISP with Several Fixed Access Networks

This example discusses a P2P application traffic optimization use case for a larger ISP with a fixed network comprising several access networks and a core network. The traffic optimizing objectives include (1) using the backbone network efficiently, (2) adjusting the traffic balance in different access networks according to traffic conditions and management policies, and (3) achieving a reduction of settlement costs with other ISPs.

Such a large ISP deploying an ALTO service may want to optimize its traffic according to the network topology of its access networks. For example, each access network could be defined to be one optimization area, i.e., traffic should be kept local withing that area if possible. This can be achieved by mapping each area to a PID. Then the costs between those access networks can be defined according to a corresponding traffic optimizing requirement by this ISP. One example setup is further described below and also shown in Figure 13.

In this example, ISP A has one backbone network and three access networks, named as AN A, AN B, and AN C. A P2P application is used in this example. For a reasonable application-level traffic optimization, the first requirement could be a decrease of the P2P traffic on the backbone network inside the Autonomous System of ISP A and the second requirement could be a decrease of the P2P traffic to other ISPs, i.e., other Autonomous Systems. The second requirement can be assumed to have priority over the first one. Also, we assume that the settlement rate with ISP B is lower than with other ISPs. ISP A can deploy an ALTO service to meet these traffic distribution requirements. In the following, we will give an example of an ALTO setting and configuration according to these requirements.

In the network of ISP A, the operator of the ALTO server can define each access network to be one optimization area, and assign one PID to each access network, such as PID 1, PID 2, and PID 3. Because of different peerings with different outer ISPs, one can define ISP B to be one additional optimization area and assign PID 4 to it. All other networks can be added to a PID to be one further optimization area (PID 5).

In the setup, costs (C1, C2, C3, C4, C5, C6, C7, C8) can be assigned as shown in Figure 13. Cost C1 is denoted as the link cost in inner AN A (PID 1), and C2 and C3 are defined accordingly. C4 is denoted as the link cost from PID 1 to PID 2, and C5 is the corresponding cost from PID 3, which is assumed to have a similar value. C6 is the cost between PID 1 and PID 3. For simplicity, this scenario assumes symmetrical costs between the AN this example. C7 is denoted as the link cost from the ISP B to ISP A. C8 is the link cost from other networks to ISP A.

According to previous discussion of the first requirement and the second requirement, the relationship of these costs will be defined as: (C1, C2, C3) < (C4, C5, C6) < (C7) < (C8)

```
    +------------------------------------+         +----------------+
    | ISP A   +---------------+          |         |                |
    |         |    Backbone   |          |   C7    |      ISP B     |
    |     +---+    Network    +----+     |<--------+      PID 4     |
    |     |   +-------+-------+    |     |         |                |
    |     |           |            |     |         |                |
    |     |           |            |     |         +----------------+
    | +---+--+     +--+---+     +--+---+ |
    | |AN A  |  C4 |AN B  |  C5 |AN C  | |
    | |PID 1 +<--->|PID 2 |<--->+PID 3 | |
    | |C1    |     |C2    |     |C3    | |         +----------------+
    | +---+--+     +------+     +--+---+ |         |                |
    |     ^                        ^     |   C8    | Other Networks |
    |     |                        |     |<--------+ PID 5          |
    |     +------------------------+     |         |                |
    |                 C6                 |         |                |
    +------------------------------------+         +----------------+
```

Figure 13: ALTO deployment in large ISPs with layered fixed network structures

3.5.3. ISP with Fixed and Mobile Network

An ISP with both mobile network and fixed network my focus on optimizing the mobile traffic by keeping traffic in the fixed network as far as possible, because wireless bandwidth is a scarce resource and traffic is costly in mobile network. In such a case, the main requirement of traffic optimization could be decreasing the usage of radio resources in the mobile network. An ALTO service can be deployed to meet these needs.

Figure 14 shows an example: ISP A operates one mobile network, which is connected to a backbone network. The ISP also runs two fixed access networks AN A and AN B, which are also connected to the backbone network. In this network structure, the mobile network can be defined as one optimization area, and PID 1 can be assigned to it. Access networks AN A and B can also be defined as optimization areas, and PID 2 and PID 3 can be assigned, respectively. The cost values are then defined as shown in Figure 14.

To decrease the usage of wireless link, the relationship of these costs can be defined as follows:

From view of mobile network: C4 < C1. This means that clients in mobile network requiring data resource from other clients will prefer clients in AN A to clients in the mobile network. This policy can decrease the usage of wireless link and power consumption in terminals.

From view of AN A: C2 < C6, C5 = maximum cost. This means that clients in other optimization area will avoid retrieving data from the mobile network.

```
    +-----------------------------------------------------------------+
    |                                                                 |
    |  ISP A                 +-------------+                          |
    |               +--------+   ALTO      +---------+                |
    |               |        |   Service   |         |                |
    |               |        +------+------+         |                |
    |               |               |                |                |
    |               |               |                |                |
    |               |               |                |                |
    |       +-------+-------+       | C6    +--------+------+         |
    |       |     AN A      |<--------------|      AN B     |         |
    |       |     PID 2     |   C7  |       |      PID 3    |         |
    |       |     C2        |-------------->|      C3       |         |
    |       +---------------+       |       +---------------+         |
    |             ^    |            |              |     ^            |
    |             |    |            |              |     |            |
    |             |    |C4          |              |     |            |
    |          C5 |    |            |              |     |            |
    |             |    |   +--------+---------+    |     |            |
    |             |    +-->|  Mobile Network  |<---+     |            |
    |             |        |  PID 1           |          |            |
    |             +------- |  C1              |----------+            |
    |                      +------------------+                       |
    +-----------------------------------------------------------------+
```

Figure 14: ALTO deployment in ISPs with mobile network

These examples show that for ALTO in particular the relations between different costs matter; the operator of the server has several degrees of freedom how to set the absolute values.

3.6. Deployment Experiences

The examples in the previous section are simple and do not consider specific requirements inside access networks, such as different link types. Deploying an ALTO service in real network may require dealing with further network conditions and requirements. One real example is described in greater detail in reference [I-D.lee-alto-chinatelecom-trial].

Also, experiments have been conducted with ALTO-like deployments in Internet Service Provider (ISP) networks. For instance, NTT performed tests with their HINT server implementation and dummy nodes to gain insight on how an ALTO-like service influence peer-to-peer systems [I-D.kamei-p2p-experiments-japan]. The results of an early experiment conducted in the Comcast network are documented in [RFC5632].

4. Using ALTO for P2P Traffic Optimization

4.1. Overview

4.1.1. Usage Scenario

Originally, peer-to-peer (P2P) applications have been the main driver for the development of ALTO. In this use case it is assumed that one party (usually the operator of a "managed" IP network domain) will disclose information about the network through ALTO. The application overlay will query this information and optimize its behavior in order to improve performance or Quality of Experience in the application while reducing the utilization of the underlying network infrastructure. The resulting win-win situation is assumed to be the incentive for both parties to provide or consume the ALTO information, respectively.

P2P systems can be build without or with use of a centralized resource directory ("tracker"). The scope of this section is the interaction of P2P applications with the ALTO service. In this scenario, the resource consumer ("peer") asks the resource directory for a list of candidate resource providers, which can provide the desired resource. There are different options how ALTO can be deployed in such use cases with a centralized resource directory.

For efficiency reasons (i.e., message size), usually only a subset of all resource providers known to the resource directory will be returned to the resource consumer. Some or all of these resource providers, plus further resource providers learned by other means such as direct communication between peers, will be contacted by the resource consumer for accessing the resource. The purpose of ALTO is giving guidance on this peer selection, which is supposed to yield better-than-random results. The tracker response as well as the ALTO guidance are most beneficial in the initial phase after the resource consumer has decided to access a resource, as long as only few resource providers are known. Later, when the resource consumer has already exchanged some data with other peers and measured the transmission speed, the relative importance of ALTO may dwindle.

4.1.2. Applicability of ALTO

A tracker-based P2P application can leverage ALTO in different ways. In the following, the different alternatives and their pros and cons are discussed.

```
                            ,-------.         +-----------+
          ,---.          ,-’         ‘-.  +==>|   Peer 1  |*****
       ,-’     ‘-.      /     ISP 1     \ =   |ALTO Client|    *
      /           \    / +-------------+<=+   +-----------+    *
     /    ISP X    \   | + ALTO Server |<=+   +-----------+    *
    /               \  \ +-------------+ /=   |   Peer 2  |    *
   ;   +---------+   :  \               / +==>|ALTO Client|*****
   |   | Global  |   |   ‘-.         ,-’      +-----------+   **
   |   | Tracker |   |      ‘-------’                         **
   |   +---------+   |      ,-------.         +-----------+   **
   :        *        ;   ,-’         ‘-.  +==>|   Peer 3  |   **
    \       *       /   /     ISP 2     \ =   |ALTO Client|*****
     \      *      /   / +-------------+<=+   +-----------+  ***
      \     *     /    | | ALTO Server |<=+   +-----------+  ***
       ‘-.  *  ,-’     \ +-------------+ /=   |   Peer 4  |*****
          ‘-*-’         \               / +==>|ALTO Client| ****
            *            ‘-.         ,-’      +-----------+ ****
            *               ‘-------’                       ****
            *                                               ****
            ***********************************************<****
       Legend:
       === ALTO protocol
       *** Application protocol
```

Figure 15: Global tracker and local ALTO servers

Figure 15 depicts a tracker-based P2P system with several peers. The peers (i.e., resource consumers) embed an ALTO client to improve the resource selection. The tracker (i.e., resource directory) itself may be hosted and operated by another entity. A tracker outside the networks of the ISPs of the peers may be a typical use case. For instance, a tracker like Pirate Bay can serve Bittorrent peers world-wide. The figure only shows one tracker instance, but deployments with several trackers could be possible, too.

In the scenario depicted in Figure 15 lets the peers directly communicate with their ISP's ALTO server (i.e., ALTO client embedded in the peers), giving thus the peers the most control on which information they query for, as they can integrate information received from one tracker or several trackers and through direct peer-to-peer knowledge exchange. For instance, the latter approach is called peer exchange (PEX) in bittorent. In this deployment scenarios, the peers have to discover a suitable ALTO server e.g. offered by their ISP, as described in [RFC7286].

There are also tracker-less P2P system architectures that do not rely on centralized resource directories, e.g., unstructured P2P networks. Regarding the use of ALTO, their deployment would be similar to Figure 15, since the ALTO client would be embedded in the peers as well. This option is not further considered in this memo.

```
                                 ,-------.
         ,---.                ,-'         `-.   +-----------+
       ,-'     `-.           /     ISP 1     \  |   Peer 1  |*****
      /           \         / +-------------+ \ |           |    *
     /    ISP X    \   +=====>| ALTO Server |  )+-----------+    *
    /               \  =    \ +-------------+ / +-----------+    *
   ; +-----------+   : =     \               /  |   Peer 2  |    *
   | |  Tracker  |<====+      `-.         ,-'   |           |*****
   | |ALTO Client|<====+         `-------'      +-----------+   **
   | +-----------+   | =         ,-------.                      **
   :        *        ; =      ,-'         `-.   +-----------+   **
    \       *       /  =     /     ISP 2     \  |   Peer 3  |   **
     \      *      /   =    / +-------------+ \ |           |*****
      \     *     /    +=====>| ALTO Server |  )+-----------+  ***
       `-.  *  ,-'          \ +-------------+ / +-----------+  ***
          `-*-'              \               /  |   Peer 4  |*****
            *                 `-.         ,-'   |           | ****
            *                    `-------'      +-----------+ ****
            *                                                 ****
            *                                                 ****
            ***********************************************<******
       Legend:
       === ALTO protocol
       *** Application protocol
```

Figure 16: Global tracker accessing ALTO server at various ISPs

An alternative deployment scenario for a tracker-based system is depicted in Figure 16. Here, the tracker embeds the ALTO client. As already explained, the tracker itself may be hosted and operated by an entity different than the ISP hosting and operating the ALTO server. The key difference to the previously discussed use case is that the ALTO client is different to the resource consumer. Initially, the tracker has to look-up the ALTO server in charge for each peer where it receives a ALTO query for. Therefore, the ALTO server has to discover the handling ALTO server for a pear [RFC7286] [I-D.kiesel-alto-xdom-disc]. The peers do not have any way to query the ALTO server themselves. This setting allows giving the peers a better selection of candidate peers for their operation at an initial time, but does not consider peers learned through direct peer-to-peer knowledge exchange.

```
                             ,-------.         +-----------+
           ,---.          ,-'  ISP 1  '-.  ***>|   Peer 1  |
        ,-'     '-.      /+-------------+\ *   |           |
       /         \     / +   Tracker   |<**   +-----------+
      /    ISP X    \    | +-----===-----+<**   +-----------+
     /               \  \ +-----===-----+ /*   |   Peer 2  |
    ;   +---------+   :  \+ ALTO Server |/ ***>|           |
    |   | Global  |   |   +-------------+      +-----------+
    |   | Tracker |   |      '-------'
    |   +---------+   |                         +-----------+
    :          ^      ;      ,-------.         |   Peer 3  |
     \         *     /    ,-'  ISP 2  '-.  ***>|           |
      \        *    /    /+-------------+\ *   +-----------+
       \       *   /    / +   Tracker   |<**   +-----------+
        '-.    *,-'     | +-----===-----+ |    |   Peer 4  |<*
           '---*        \ +-----===-----+ /    |           | *
               *         \+ ALTO Server |/     +-----------+ *
               *          +-------------+                    *
               *             '-------'                       *
               ***********************************************
        Legend:
        === ALTO protocol
        *** Application protocol
```

Figure 17: Local trackers and local ALTO servers (P4P approach)

There are some attempts to let ISP's to deploy their own trackers, as shown in Figure 17. In this case, the client has no chance to get guidance from the ALTO server, other than talking to the ISP's tracker. However, the peers would have still chance the contact other trackers, deployed by entities other than the peer's ISP.

4.2. Deployment Recommendations

4.2.1. ALTO Services

The ALTO protocol specification [RFC7285] details how an ALTO client can query an ALTO server for guiding information and receive the corresponding replies. In case of peer-to-peer networks, two different ALTO services can be used: The Cost Map Service is often preferred as solution by peer-to-peer software implementors and users, since it avoids disclosing peer IP addresses to a centralized entity. Different to that, network operators may have a preference for the Endpoint Cost Service (ECS), since it does not require exposure of the network topology.

For actual use of ALTO in P2P applications, both software vendors and network operators have to agree which ALTO services to use. The ALTO protocol is flexible and supports both services. Note that for other use cases of ALTO, in particular in more controlled environments, both the Cost Map Service as well as Endpoint Cost Service might be feasible and it is more an engineering trade-off whether to use a map-based or query-based ALTO service.

4.2.2. Guidance Considerations

As explained in Section 4.1.2, for 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")

Both approaches have advantages and drawbacks that have to be considered. If the ALTO client is in the resource consumer (Figure 15), a potentially very large number of clients has to be deployed. Instead, when using an ALTO client in the resource directory (Figure 16 and Figure 17), ostensibly peers do not have to directly query the ALTO server. In this case, an ALTO server could even not permit access to peers.

However, it seems to be beneficial for all participants to let the peers directly query the ALTO server. Considering the plethora of different applications that could use ALTO, e.g. multiple tracker or non-tracker based P2P systems or other applications searching for relays, this renders the ALTO service more useful. The peers are also the single point having all operational knowledge to decide whether to use the ALTO guidance and how to use the ALTO guidance. For a given peer one can also expect that an ALTO server of the corresponding ISP provides useful guidance and can be discovered.

Yet, ALTO clients in the resource consumer also have drawbacks compared to use in the resource directory. In the following, both scenarios are compared more in detail in order to explain the impact on ALTO guidance and the need for third-party ALTO queries.

In the first scenario (see Figure 18), the peer (resource consumer) queries the tracker (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.

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

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

Figure 18: Basic message sequence chart for resource consumer-initiated ALTO query

In the second scenario (see Figure 19), the resource directory has an embedded ALTO client, which we will refer to as Resource Directory ALTO Client (RDAC) in this document. After receiving a query for a given resource (F1) the resource directory invokes the RDAC 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).

```
         Peer             Tracker w. RDAC         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 protocol
```

Figure 19: Basic message sequence chart for third-party ALTO query

Note: The message sequences depicted in Figure 18 and Figure 19 may occur both in the target-aware and the target-independent query mode (cf. [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 first approach has the following problem: 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.

Much better traffic optimization could be achieved if the tracker would evaluate all known peers using ALTO. This list would then include a significantly higher fraction of "good" peers. 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 third-party discovery. 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 (see [I-D.kiesel-alto-xdom-disc]).

5. Using ALTO for CDNs

5.1. Overview

5.1.1. Usage Scenario

This section briefly introduces the usage of ALTO for Content Delivery Networks (CDNs), as explained e.g. in [I-D.jenkins-alto-cdn-use-cases]. CDNs are used in the delivery of some Internet services (e.g. delivery of websites, software updates and video delivery) from a location closer to the location of the user. A CDN typically consists of a network of servers often attached to Internet Service Provider (ISP) networks. The point of attachment is often as close to content consumers and peering points as economically or operationally feasible in order to decrease traffic load on the ISP backbone and to provide better user experience measured by reduced latency and higher throughput.

CDNs use several techniques to redirect a client to a server (surrogate). A request routing function within a CDN is responsible for receiving content requests from user agents, obtaining and maintaining necessary information about a set of candidate surrogates, and for selecting and redirecting the user agent to the appropriate surrogate. One common way is relying on the DNS system, but there are many other ways, see [RFC3568].

```
   +--------------------+
   | CDN Request Router |
   |  with ALTO Client  |
   +--------------------+
             /\
             || ALTO protocol
             ||
         +---------+
         |  ALTO   |
         | Server  |
         +---------+
              ^
              : Provisioning protocol
              :
        ,-----------.
     ,-'  Source of  `-.
    (    topological    )
     `-. information ,-'
        `-----------'
```

Figure 20: Use of ALTO information for CDN request routing

In order to derive the optimal benefit from a CDN it is preferable to deliver content from the servers (caches) that are "closest" to the end user requesting the content. "closest" may be as simple as geographical or IP topology distance, but it may also consider other combinations of metrics and CDN or Internet Service Provider (ISP) policies. As illustrated in Figure 20, ALTO could provide this information.

```
   User Agent                  Request Router                 Surrogate
        |                             |                           |
        |     F1 Initial Request      |                           |
        +---------------------------->|                           |
        |                             +--+                        |
        |                             |  | F2 Surrogate Selection |
        |                             |<-+       (using ALTO)     |
        |   F3 Redirection Response   |                           |
        |<----------------------------+                           |
        |                             |                           |
        |     F4 Content Request      |                           |
        +-------------------------------------------------------->|
        |                             |                           |
        |                             |          F5 Content       |
        |<--------------------------------------------------------+
        |                             |                           |
```

Figure 21: Example of CDN surrogate selection

Figure 21 illustrates the interaction between a user agent, a request router, and a surrogate for the delivery of content in a single CDN. As explained in [I-D.jenkins-alto-cdn-use-cases], the user agent makes an initial request to the CDN (F1). This may be an application-level request (e.g., HTTP) or a DNS request. In the second step (F2), the request router selects an appropriate surrogate (or set of surrogates) based on the user agent’s (or its proxy’s) IP address, the request router’s knowledge of the network topology (which can be obtained by ALTO) and reachability cost between CDN caches and end users, and any additional CDN policies. Then (F3), the request router responds to the initial request with an appropriate response containing a redirection to the selected cache, for example by returning an appropriate DNS A/AAAA record, a HTTP 302 redirect, etc. The user agent uses this information to connect directly to the surrogate and request the desired content (F4), which is then delivered (F5).

5.1.2. Applicability of ALTO

The most simple use case for ALTO in a CDN context is to improve the selection of a CDN surrogate or origin. In this case, the CDN makes use of an ALTO server to choose a better CDN surrogate or origin than would otherwise be the case. Although it is possible to obtain raw network map and cost information in other ways, for example passively listening to the ISP’s routing protocols or use of active probing, the use of an ALTO service to expose that information may provide additional control to the ISP over how their network map/cost is exposed. Additionally it may enable the ISP to maintain a functional separation between their routing plane and network map computation functions. This may be attractive for a number of reasons, for example:

o The ALTO service could provide a filtered view of the network and/or cost map that relates to CDN locations and their proximity to end users, for example to allow the ISP to control the level of topology detail they are willing to share with the CDN.

o The ALTO service could apply additional policies to the network map and cost information to provide a CDN-specific view of the network map/cost, for example to allow the ISP to encourage the CDN to use network links that would not ordinarily be preferred by a Shortest Path First routing calculation.

o The routing plane may be operated and controlled by a different operational entity (even within a single ISP) to the CDN. Therefore, the CDN may not be able to passively listen to routing protocols, nor may it have access to other network topology data (e.g., inventory databases).

When CDN servers are deployed outside of an ISP’s network or in a small number of central locations within an ISP’s network, a simplified view of the ISP’s topology or an approximation of proximity is typically sufficient to enable the CDN to serve end users from the optimal server/location. As CDN servers are deployed deeper within ISP networks it becomes necessary for the CDN to have more detailed knowledge of the underlying network topology and costs between network locations in order to enable the CDN to serve end users from the most optimal servers for the ISP.

The request router in a CDN will typically also take into account criteria and constraints that are not related to network topology, such as the current load of CDN surrogates, content owner policies, end user subscriptions, etc. This document only discusses use of ALTO for network information.

A general issue for CDNs is that the CDN logic has to match the client’s IP address with the closest CDN surrogate, both for DNS or HTTP redirect based approaches (see, for instance, [I-D.penno-alto-cdn]). This matching is not trivial, for instance, in DNS based approaches, where the IP address of the DNS original requester is unknown (see [I-D.vandergaast-edns-client-ip] for a discussion of this and a solution approach).

In addition to use by a single CDN, ALTO can also be used in scenarios that interconnect several CDNs. This use case is detailed in [I-D.seedorf-cdni-request-routing-alto].

5.2. Deployment Recommendations

5.2.1. ALTO Services

In its simplest form an ALTO server would provide an ISP with the capability to offer a service to a CDN that provides network map and cost information. The CDN can use that data to enhance its surrogate and/or origin selection. If an ISP offers an ALTO network and cost map service to expose a cost mapping/ranking between end user IP subnets (within that ISP’s network) and CDN surrogate IP subnets/ locations, periodic updates of the maps may be needed. As introduced in Section 3.3), it is common for broadband subscribers to obtain their IP addresses dynamically and in many deployments the IP subnets allocated to a particular network region can change relatively frequently, even if the network topology itself is reasonably static.

An alternative would be to use the ALTO Endpoint Cost Service (ECS): When an end user request a given content, the CDN request router issues an ECS request with the endpoint address (IPv4/IPv6) of the end user (content requester) and the set of endpoint addresses of the surrogate (content targets). The ALTO server receives the request and ranks the list of content targets addresses based on their distance from the content requester. Once the request router obtained from the ALTO Server the ranked list of locations (for the specific user), it can incorporate this information into its selection mechanisms in order to point the user to the most appropriate surrogate.

Since CDNs operate in a controlled environment, the ALTO network/cost map service and ECS have a similar level of security and confidentiality of network-internal information. However, the network/cost map service and ECS differ in the way the ALTO service is delivered and address a different set of requirements in terms of topology information and network operations.

If a CDN already has means to model connectivity policies, the map-based approaches could possibly be integrated into that. If the ECS service is preferred, a request router that uses ECS could cache the results of ECS queries for later usage in order to address the scalability limitations of ECS and to reduce the number of transactions between CDN and ALTO server. The ALTO server may indicate in the reply message how long the content of the message is to be considered reliable and insert a lifetime value that will be used by the CDN in order to cache (and then flush or refresh) the entry.

5.2.2. Guidance Considerations

In the following it is discussed how a CDN could make use of ALTO services.

In one deployment scenario, ALTO could expose ISP end user reachability to a CDN. The request router needs to have information which end user IP subnets are reachable via which networks or network locations. The network map services offered by ALTO could be used to expose this topology information while avoiding routing plane peering between the ISP and the CDN. For example, if CDN surrogates are deployed within the access or aggregation network, the ISP is likely to want to utilize the surrogates deployed in the same access/aggregation region in preference to surrogates deployed elsewhere, in order to alleviate the cost and/or improve the user experience.

In addition, CDN surrogates could also use ALTO guidance, e.g., if there is more than one upstream source of content or several origins. In this case, ALTO could help a surrogate with the decision which upstream source to use. This specific variant of using ALTO is not further detailed in this document.

If content can be provided by several CDNs, there may be a need to interconnect these CDNs. In this case, ALTO can be uses as interface [I-D.seedorf-cdni-request-routing-alto], in particular for footprint and capabilities advertisement interface.

Other and more advanced scenarios of deploying ALTO are also listed in [I-D.jenkins-alto-cdn-use-cases] and [I-D.penno-alto-cdn].

The granularity of ALTO information required depends on the specific deployment of the CDN. For example, an over-the-top CDN whose surrogates are deployed only within the Internet "backbone" may only require knowledge of which end user IP subnets are reachable via which ISPs’ networks, whereas a CDN deployed within a particular ISP’s network requires a finer granularity of knowledge.

ALTO server ranks addresses based on topology information it acquires from the network. By default, according to [RFC7285], distance in ALTO represents an abstract "routingcost" that can be computed for instance from routing protocol information. But an ALTO server may also take into consideration other criteria or other information sources for policy, state, and performance information (e.g., geo-location), as explained in Section 3.2.1.

The different methods and algorithms through which the ALTO server computes topology information and rankings is out of the scope of this document. If rankings are based on routing protocol information, it is obvious that network events may impact the ranking computation. Due to internal redundancy and resilience mechanisms inside current networks, most of the network events happening in the infrastructure will be handled internally in the network, and they should have limited impact on a CDN. However, catastrophic events such as main trunks failures or backbone partitioning will have to take into account by the ALTO server to redirect traffic away from the impacted area.

An ALTO server implementation may want to keep state about ALTO clients so to inform and signal to these clients when a major network event happened, e.g., by a notification mechanism. In a CDN/ALTO interworking architecture with few CDN components interacting with the ALTO server there are less scalability issues in maintaining state about clients in the ALTO server, compared to ALTO guidance to any Internet user.

6. Other Use Cases

This section briefly surveys and references other use cases that have been tested or suggested for ALTO deployments.

6.1. Application Guidance in Virtual Private Networks (VPNs)

Virtual Private Network (VPN) technology is widely used in public and private networks to create groups of users that are separated from other users of the network and allows these users to communicate among them as if they were on a private network. Network Service Providers (NSPs) offer different types of VPNs. [RFC4026] distinguishes between Layer 2 VPN (L2VPN) and Layer 3 VPN (L3VPN) using different sub-types. In the following, the term "VPN" is used to refer to provider supplied virtual private networking.

From the perspective of an application at an endpoint, a VPN may not be very different to any other IP connectivity solution, but there are a number of specific applications that could benefit from ALTO topology exposure and guidance in VPNs. Similar like in the general Internet, one advantage is that applications do not have to perform excessive measurements on their own. For instance, potential use cases for ALTO application guidance in VPNs environments are:

o Enterprise application optimization: Enterprise customers often run distributed applications that exchange large amounts of data, e.g., for synchronization of replicated data bases. Both for placement of replicas as well as for the scheduling of transfers insight into network topology information could be useful.

o Private cloud computing solution: An enterprise customer could run own data centers at the four sites. The cloud management system could want to understand the network costs between different sites for intelligent routing and placement decisions of Virtual Machines (VMs) among the VPN sites.

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

These examples focus on enterprises, which are typical users of VPNs. VPN customers typically have no insight into the network topology that transports the VPN. Similar like in other ALTO use cases, better-than-random application-level decisions would be enabled by an ALTO server offered by the NSP, as illustrated in Figure 22.

```
                       +---------------+
                       |  Customer’s   |
                       |   management  |
                       |  application  |.
                       | (ALTO client) |  .
                       +---------------+    .  VPN provisioning
                               ^              . (out-of-scope)
                               | ALTO           .
                               V                  .
                    +---------------------+       +----------------+
                    |     ALTO server     |       | VPN portal/OSS |
                    |   provided by NSP   |       | (out-of-scope) |
                    +---------------------+       +----------------+
                               ^ VPN network
                               * and cost maps
                               *
                     /---------*---------\ Network service provider
                     |         *         |
        +-------+   _______________________   +-------+
        | App a | ()_____. ._______. .____() | App d |
        +-------+    |   | |       | |  |    +-------+
                     \---| |-------| |--/
                         | |       | |
                         |^|       |^| Customer VPN
                          V         V
                      +-------+  +-------+
                      | App b |  | App c |
                      +-------+  +-------+
```

Figure 22: Using ALTO in VPNs

A common characteristic of these use cases is that applications will not necessarily run in the public Internet, and that the relationship between the provider and customer of the VPN is rather well-defined. Since VPNs run often in a managed environment, an ALTO server may have access to topology information (e.g., traffic engineering data) that would not be available for the public Internet, and it may expose it to the customer of the VPN only.

Also, a VPN will not necessarily be static. The customer could possibly modify the VPN and add new VPN sites by a Web portal, network management systems, or other Operation Support Systems (OSS) solutions. Prior to adding a new VPN site, an application will not be have connectivity to that site, i.e., an ALTO server could offer access to information that an application cannot measure on its own (e.g., expected delay to a new VPN site).

The VPN use cases, requirements, and solutions are further detailed in [I-D.scharf-alto-vpn-service].

6.2. In-Network Caching

Deployment of intra-domain P2P caches has been proposed for a cooperations between the network operator and the P2P service providers, e.g., to reduce the bandwidth consumption in access networks [I-D.deng-alto-p2pcache].

```
      +--------------+                +------+
      | ISP 1 network+----------------+Peer 1|
      +-----+--------+                +------+
      |
   +--------+------------------------------------------------------+
   |        |                                      ISP 2 network   |
   |  +---------+                                                  |
   |  |L1 Cache |                                                  |
   |  +-----+---+                                                  |
   |        +--------------------+----------------------+          |
   |        |                    |                      |          |
   | +------+------+      +------+-------+      +------+-------+  |
   | | AN1         |      | AN2          |      | AN3           |  |
   | | +---------+ |      | +----------+ |      |              |  |
   | | |L2 Cache | |      | |L2 Cache  | |      |              |  |
   | | +---------+ |      | +----------+ |      |              |  |
   | +------+------+      +------+-------+      +------+-------+  |
   |        |                                           |          |
   |        +--------------------+                      |          |
   |        |                    |                      |          |
   | +------+------+      +------+-------+      +------+-------+  |
   | | SUB-AN11    |      | SUB-AN12     |      | SUB-AN31      |  |
   | | +---------+ |      |              |      |              |  |
   | | |L3 Cache | |      |              |      |              |  |
   | | +---------+ |      |              |      |              |  |
   | +------+------+      +------+-------+      +------+-------+  |
   |        |                    |                      |          |
   +--------+--------------------+----------------------+----------+
            |                    |                      |
        +---+---+            +---+---+                  |
        |       |            |       |                  |
     +--+--+ +--+--+      +--+--+ +--+--+            +--+--+
     |Peer2| |Peer3|      |Peer4| |Peer5|            |Peer6|
     +-----+ +-----+      +-----+ +-----+            +-----+
```

Figure 23: General architecture of intra-ISP caches

Figure 23 depicts the overall architecture of a potential P2P cache deployments inside an ISP 2 with various access network types. As shown in the figure, P2P caches may be deployed at various levels, including the interworking gateway linking with other ISPs, internal access network gateways linking with different types of accessing networks (e.g. WLAN, cellular and wired), and even within an accessing network at the entries of individual WLAN sub-networks. Moreover, depending on the network context and the operator’s policy, each cache can be a Forwarding Cache or a Bidirectional Cache [I-D.deng-alto-p2pcache].

In such a cache architecture, the locations of caches could be used as dividers of different PIDs to guide intra-ISP network abstraction and mark costs among them according to the location and type of relevant caches.

Further details and deployment considerations can be found in [I-D.deng-alto-p2pcache].

6.3. Other Application-based Network Operations

An ALTO server can be part of an overall framework for Application-Based Network Operations (ABNO) [I-D.farrkingel-pce-abno-architecture] that brings together different technologies for gathering information about the resources available in a network, for consideration of topologies and how those topologies map to underlying network resources, for requesting path computation, and for provisioning or reserving network resources. Such an architecture may include additional components such as a Path Computation Element (PCE) for on-demand and application-specific reservation of network connectivity, reliability, and resources (such as bandwidth). Some use cases how to leverage ALTO for joint network and application-layer optimization are explained in [I-D.farrkingel-pce-abno-architecture].

7. Security Considerations

Security concerns were extensively discussed from the very beginning of the development of the ALTO protocol, and they have been considered in detail in the ALTO requirements document [RFC6708] as well as in the ALTO protocol specification document [RFC7285]. The two main security concerns are related to the unwanted disclosure of information through ALTO and the negative impact of specially crafted, wrong ("faked") guidance presented to an ALTO client. In addition to this, the usual concerns related to the operation of any networked application apply.

This section focuses on the peer-to-peer use case, which is - from a security perspective - probably the most difficult ALTO use case that has been considered. Special attention is given to the two main security concerns.

7.1. ALTO as a Protocol Crossing Trust Boundaries

The optimization of peer-to-peer applications was the first use case and the impetus for the development of the ALTO protocol, in particular file sharing applications such as BitTorrent [RFC5594].

As explained in Section 4.1.1, for the publisher of the ALTO information (i.e., the ALTO server operator) it is not always clear who is in charge of the P2P application overlay. Some P2P applications do not have any central control entity and the whole overlay consists only of the peers, which are under control of the individual users. Other P2P applications may have some control entities such as super peers or trackers, but these may be located in foreign countries and under the control of unknown organizations. As outlined in Section 4.2.2, in some scenarios it may be very beneficial to forward ALTO information to such trackers, super peers, etc. located in remote networks. This somewhat intransparent situation is aggravated by the vast number of different P2P applications which are evolving quickly and often without any coordination with the network operators.

In summary it can be said that in many instances of the P2P use case, the ALTO protocol bridges the border between the "managed" IP network infrastructure under strict administrative control and one or more "unmanaged" application overlays, i.e., overlays for which it is hard to tell who is in charge of them. This is different to more controlled environments (e.g., in the CDN use case), in which bilateral agreements between the producer and consumer of guidance are possible.

7.2. Information Leakage from the ALTO Server

An ALTO server will be provisioned with information about the ISP’s network and possibly also with information about neighboring ISPs. This information (e.g., network topology, business relations, etc.) is often considered to be confidential to the ISP and can include very sensitive information. ALTO does not require any particular level of details of information disclosure, and hence the provider should evaluate how much information is revealed and the associated risks.

Furthermore, if the ALTO information is very fine grained, it may also be considered sensitive with respect to user privacy. For example, consider a hypothetical endpoint property "provisioned access link bandwidth" or "access technology (ADSL, VDSL, FTTH, etc.)" and an ALTO service that publishes this property for individual IP addresses. This information could not only be used for traffic optimization but, for example, also for targeted advertising to residential users with exceptionally good (or bad) connectivity, such as special banner ads. For an advertisement system it would be more complex to obtain such information otherwise, e.g., by bandwidth probing.

Different scenarios related to the unwanted disclosure of an ALTO server’s information have been itemized and categorized in RFC 6708, Section 5.2.1., cases (1)-(3) [RFC6708].

In some use cases it is not possible to use access control (see Section 7.3) to limit the distribution of ALTO knowledge to a small set of trusted clients. In these scenarios it seems tempting not to use network maps and cost maps at all, and instead completely rely on endpoint cost service and endpoint ranking in the ALTO server. While this practice may indeed reduce the amount of information that is disclosed to an individual ALTO client, some issues should be considered: First, when using the map based apporach, it is trivial to analyze the maximum amount of information that could be disclosed to a client: the full maps. In contrast, when providing endpoint cost service only, the ALTO server operator could be prone to a false feeling of security, while clients use repeated queries and/or collaboration to gather more information than they are expected to get (see Section 5.2.1., case (3) in [RFC6708]). Second, the endpoint cost service reveals more information about the user or application behavior to the ALTO server, e.g., which other hosts are considered as peers for the exchange of a significant amount of data (see Section 5.2.1., cases (4)-(6) in [RFC6708]).

Consequently, users may be more reluctant to use the ALTO service at all if it is based on the endpoint property service instead of providing network and cost maps. Given that some popular P2P applications are sometimes used for purposes such as distribution of files without the explicit permission from the copyright owner, it may also be in the interest of the ALTO server operator that an ALTO server cannot infer the behavior of the application to be optimized. One possible conclusion could be to publish network and cost maps through ALTO that are so coarse-grained that they do not violate the network operator’s or the user’s interests.

In other use cases in more controlled environments (e.g., in the CDN use case) bilateral agreements, access control (see Section 7.3), and encryption could be used to reduce the risk of information leakage.

7.3. ALTO Server Access

Depending on the use case of ALTO, it may be desired to apply access restrictions to an ALTO server, i.e., by requiring client authentication. According to [RFC7285], ALTO requires that HTTP Digest Authentication is supported, in order to achieve client authentication and possibly to limit the number of parties with whom ALTO information is directly shared. TLS Client Authentication may also be supported.

In general, well-known security management techniques and best current practices [RFC4778] for operational ISP infrastructure also apply to an ALTO service, including functions to protect the system from unauthorized access, key management, reporting security-relevant events, and authorizing user access and privileges.

For peer-to-peer applications, a potential deployment scenario is that an ALTO server is solely accessible by peers from the ISP network (as shown in Figure 15). For instance, the source IP address can be used to grant only access from that ISP network to the server. This will "limit" the number of peers able to attack the server to the user’s of the ISP (however, including botnet computers).

If the ALTO server has to be accessible by parties not located in the ISP’s network (see Figure 16), e.g., by a third-party tracker or by a CDN system outside the ISP’s network, the access restrictions have to be looser. In the extreme case, i.e., no access restrictions, each and every host in the Internet can access the ALTO server. This might no be the intention of the ISP, as the server is not only subject to more possible attacks, but also the server load could increase, since possibly more ALTO clients have to be served.

There are also use cases where the access to the ALTO server has to be much more strictly controlled, i. e., where an authentication and authorization of the ALTO client to the server may be needed. For instance, in case of CDN optimization the provider of an ALTO service as well as potential users are possibly well-known. Only CDN entities may need ALTO access; access to the ALTO servers by residential users may neither be necessary nor be desired.

Access control can also help to prevent Denial-of-Service attacks by arbitrary hosts from the Internet. Denial-of-Service (DoS) can both affect an ALTO server and an ALTO client. A server can get overloaded if too many requests hit the server, or if the query load of the server surpasses the maximum computing capacity. An ALTO client can get overloaded if the responses from the sever are, either intentionally or due to an implementation mistake, too large to be handled by that particular client.

7.4. Faking ALTO Guidance

The ALTO services enables an ALTO service provider to influence the behavior of network applications. An attacker who is able to generate false replies, or e.g. an attacker who can intercept the ALTO server discovery procedure, can provide faked ALTO guidance.

Here is a list of examples how the ALTO guidance could be faked and what possible consequences may arise:

Sorting: An attacker could change to sorting order of the ALTO guidance (given that the order is of importance, otherwise the ranking mechanism is of interest), i.e., declaring peers located outside the ISP as peers to be preferred. This will not pose a big risk to the network or peers, as it would mimic the "regular" peer operation without traffic localization, apart from the communication/processing overhead for ALTO. However, it could mean that ALTO is reaching the opposite goal of shuffling more data across ISP boundaries, incurring more costs for the ISP.

Preference of a single peer: A single IP address (thus a peer) could be marked as to be preferred all over other peers. This peer can be located within the local ISP or also in other parts of the Internet (e.g., a web server). This could lead to the case that quite a number of peers to trying to contact this IP address, possibly causing a Denial-of-Service (DoS) attack.

It has not yet been investigated how a faked or wrong ALTO guidance by an ALTO server can impact the operation of the network and also the applications, e.g., peer-to-peer applications.

8. IANA Considerations

This document makes no specific request to IANA.

9. Conclusion

This document discusses how the ALTO protocol can be deployed in different use cases and provides corresponding guidance and recommendations to network administrators and application developers.

10. Acknowledgments

This memo is the result of contributions made by several people:

o Xianghue Sun, Lee Kai, and Richard Yang contributed text on ISP deployment requirements and monitoring.

- o Stefano Previdi contributed parts of the Section 5 on "Using ALTO for CDNs".
- o Rich Woundy contributed text to Section 3.3.
- o Lingli Deng, Wei Chen, Qiuchao Yi, and Yan Zhang contributed Section 6.2.

Thomas-Rolf Banniza, Vinayak Hegde, and Qin Wu provided very useful comments and reviewed the document.

Martin Stiemerling is partially supported by the CHANGE project (http://www.change-project.eu), a research project supported by the European Commission under its 7th Framework Program (contract no. 257422). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the CHANGE project or the European Commission.

11. References

11.1. Normative References

[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.

[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, November 2014.

11.2. Informative References

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

[I-D.farrkingel-pce-abno-architecture]
King, D. and A. Farrel, "A PCE-based Architecture for Application-based Network Operations", draft-farrkingel-pce-abno-architecture-16 (work in progress), January 2015.

[I-D.ietf-i2rs-architecture]
Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", draft-ietf-i2rs-architecture-08 (work in progress), January 2015.

[I-D.ietf-idr-ls-distribution]
Gredler, H., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and TE Information using BGP", draft-ietf-idr-ls-distribution-10 (work in progress), January 2015.

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

[I-D.kamei-p2p-experiments-japan]
Kamei, S., Momose, T., Inoue, T., and T. Nishitani, "ALTO-Like Activities and Experiments in P2P Network Experiment Council", draft-kamei-p2p-experiments-japan-09 (work in progress), October 2012.

[I-D.kiesel-alto-h12]
Kiesel, S. and M. Stiemerling, "ALTO H12", draft-kiesel-alto-h12-02 (work in progress), March 2010.

[I-D.kiesel-alto-xdom-disc]
Kiesel, S. and M. Stiemerling, "Application Layer Traffic Optimization (ALTO) Cross-Domain Server Discovery", draft-kiesel-alto-xdom-disc-00 (work in progress), July 2014.

[I-D.lee-alto-chinatelecom-trial]
Li, K. and G. Jian, "ALTO and DECADE service trial within China Telecom", draft-lee-alto-chinatelecom-trial-04 (work in progress), March 2012.

[I-D.penno-alto-cdn]
Penno, R., Medved, J., Alimi, R., Yang, R., and S. Previdi, "ALTO and Content Delivery Networks", draft-penno-alto-cdn-03 (work in progress), March 2011.

[I-D.scharf-alto-vpn-service]
Scharf, M., Gurbani, V., Soprovich, G., and V. Hilt, "The Virtual Private Network (VPN) Service in ALTO: Use Cases, Requirements and Extensions", draft-scharf-alto-vpn-service-02 (work in progress), February 2014.

[I-D.seedorf-cdni-request-routing-alto]
Seedorf, J., Yang, Y., and J. Peterson, "CDNI Footprint and Capabilities Advertisement using ALTO", draft-seedorf-cdni-request-routing-alto-07 (work in progress), June 2014.

[I-D.vandergaast-edns-client-ip]
Contavalli, C., Gaast, W., Leach, S., and D. Rodden, "Client IP information in DNS requests", draft-vandergaast-edns-client-ip-01 (work in progress), May 2010.

[I-D.wu-alto-te-metrics]
Wu, W., Yang, Y., Lee, Y., Dhody, D., and S. Randriamasy, "ALTO Traffic Engineering Cost Metrics", draft-wu-alto-te-metrics-05 (work in progress), October 2014.

[RFC3411] Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", STD 62, RFC 3411, December 2002.

[RFC3568] Barbir, A., Cain, B., Nair, R., and O. Spatscheck, "Known Content Network (CN) Request-Routing Mechanisms", RFC 3568, July 2003.

[RFC4026] Andersson, L. and T. Madsen, "Provider Provisioned Virtual Private Network (VPN) Terminology", RFC 4026, March 2005.

[RFC4778] Kaeo, M., "Operational Security Current Practices in Internet Service Provider Environments", RFC 4778, January 2007.

[RFC5594] Peterson, J. and A. Cooper, "Report from the IETF Workshop on Peer-to-Peer (P2P) Infrastructure, May 28, 2008", RFC 5594, July 2009.

[RFC5632] Griffiths, C., Livingood, J., Popkin, L., Woundy, R., and Y. Yang, "Comcast’s ISP Experiences in a Proactive Network Provider Participation for P2P (P4P) Technical Trial", RFC 5632, September 2009.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

Authors' Addresses

Martin Stiemerling
NEC Laboratories Europe
Kurfuerstenanlage 36
Heidelberg 69115
Germany

Phone: +49 6221 4342 113
Fax: +49 6221 4342 155
Email: martin.stiemerling@neclab.eu
URI: http://ietf.stiemerling.org

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

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

Stefano Previdi
Cisco Systems, Inc.
Via Del Serafico 200
Rome 00191
Italy

Email: sprevidi@cisco.com

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

Email: michael.scharf@alcatel-lucent.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 10, 2015

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

March 9, 2015

ALTO Cost Calendar
draft-randriamasy-alto-cost-calendar-03

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 proposes to extend the cost information provided by the ALTO protocol. The purpose is to broaden the decision possibilities of applications to not only decide ’where’ to connect to, but also ’when’. This is useful to applications that have a degree of freedom on when to schedule data transfers, such as non- instantaneous data replication between data centers or service provisioning to end systems with irregular connectivity. ALTO guidance to schedule application traffic can also efficiently help for load balancing and resources efficiency.

The draft proposes a new cost Mode called "Calendar" Mode, that is applicable to time-sensitive ALTO metrics and allows Applications to carefully schedule their connections or data transfers. In the Calendar Mode, 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. Besides the functional time-shift enhancement the ALTO Cost Calendar also allows to schedule the ALTO requests themselves and thus save a number of ALTO transactions.

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 Endpoint Cost Service and the Endpoint (EP) Ranking Service then provide rankings for connections between the specified network regions and thus incentives for application clients to connect to ISP preferred endpoints, e.g. to reduce costs imposed to the network provider. Thereby ALTO intentionally avoids the provisioning of realtime information as explained in the ALTO Problem Statement [RFC5693] and ALTO Requirements [RFC5693]) drafts that write "Such information is better suited to be transferred through an in-band technique at the transport layer instead". Thus the current Cost Map and Endpoint Cost Service are providing, for a given Cost Type, exactly one rating per link between two PIDs or to an Endpoint. Applications are expected to query one of these two services in order to retrieve the currently valid cost values. They therefore need to plan their ALTO information requests according to the estimated frequency of cost value change.

Since network costs can fluctuate, due to diurnal patterns of traffic demand andor network maintenance, an ALTO client should interpret the returned costs as those at the query moment. Providing network costs for only the current time, however, 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.

In case these value changes are predicable over a certain period of time and the application does not require immediate data transfer, it would save time to get the whole set of cost values over the period in one ALTO response and using these values to schedule data transfers would allow to optimise the network resources usage and QoE.

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

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 for the pair of network locations. In addition to this functional ALTO enhancement, we expect to further gain by gathering multiple Cost Values for one cost type as firstly one Cost Map reporting on N Cost Values is less bulky than N Cost Maps containing one Cost value each and secondly, this reduces N ALTO transactions to a single one. This is valuable for both the storage of these ALTO maps and their transfer. Similar gains can be obtained for the ALTO Endpoint Cost Service.

In this draft an "ALTO Calendar" is presented as a Cost Mode that is applicable to time-sensitive ALTO metrics and allows applications using such metrics to carefully schedule their connections or data transfers. In the Calendar Mode, 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 (the ones suggested by Richard) are specified in the IRD and allow the ALTO Client to interpret the received ALTO values. This draft proposes a set of Calendar attributes to be added to the IRD, for discussion in the ALTO WG. In order to support the calendaring of Cost values represented in Modes such as ’string’ this draft also proposes one option, which is to extend the Cost Mode to a combination of several indicators, such as ’string’ and ’calendar’.

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

2. Motivating Use cases for ALTO Cost Schedule

This section introduces use cases showing the benefits of providing ALTO Cost values in ’calendar’ mode. 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 Cost Map service, filtered or not, is also applicable as long as the size of the Map is manageable.

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

2.1. Bulk Data Transfer scheduling

Large Internet Content Providers (ICPs) like Facebook or YouTube, as well as CDNs rely on data replication across multiple sites 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 pattern.

In the examples above, data needs to be replicated across the various locations of a Internet Content Provider (ICP), leading to bulk data transfers between datacenters on a diurnal pattern.

In the mean time, 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 the transfer should not infer with the user peak demand to avoid degradation of user experience and to decrease billing costs for the datacenter operator by leveraging off-peak hours for the transfer. This peak demand typically follows a diurnal pattern according to the geographic region of the datacenter.

As a result, it would be very helpful to let these ICPs to have a good knowledge about the link utilization patterns between the different datacenters from the networks before making a more intelligent scheduling decision. While this usage data today already is gathered and also used for the scheduling of data transfer, provisioning this data gets increasingly complex with the number of CDN nodes and in particular the number of datacenter operators that are involved. For example, privacy concerns prevent that this kind of data is shared across administrative domains. The ALTO Cost Calendar specified later in this document avoids this problem by presenting an abstracted view of time sensitive utilization maps through a dedicated ALTO service to allow ICPs a coherent scheduling of such data transfers across administrative domains.

2.2. Endsystems with limited connectivity or access to datacenters

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

Wireless 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 end systems using resources located in datacenters and trading content and resources scattered around the world. For non-real time applications, the interaction with Endpoints can be scheduled at the time slots corresponding to the best possible 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. Example applications are similar to the ones cited in the previous paragraph.

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

2.3. SDN Controller guided access to application endpoints

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

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

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

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

To achieve this objective, the SDN controller can:

1. get the network state history from its controlled network elements through its southbound API

2. possibly derive an estimation or a prediction of these values over given time frames

3. compute estimates and/or network provider preferences on end to end paths and store their abstraction in an ALTO Server in the form of ALTO Cost Calendar values defined for different time periods

4. deliver these values to the SDN applications via the ALTO Endpoint Cost Service, as estimations covering the past and/or the future and/or preferences.

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,

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

2.4. Large flow scheduling on extended ALTO topologies

[draft-yang-alto-topology-00] presents initial thinking on extending ALTO for topology exposure services, that would provide flexible abstractions based on the raw network topology. Among other features, an ALTO topology may expose several paths between a source (src) and destination (dst), or topology details may be provided on restricted parts. This work was presented to the ALTO WG at IETF88.

The presentation slides [slides-88-alto-5-topology] on [draft-yang-alto-topology-00] expose a use case entitled "Large Flow Scheduling". This case includes a "daylife example" where a Google Map service proposes multiple routes between 2 points A and B, each calculated w.r.t. length and estimated time. For each of these selected paths, the map service exposes a time-sensitive qualitative value taking 4 values between Slow and Fast. A user of this application may thus organize its transfer w.r.t. metrics, paths and time, provided s/he does not have to commute immediately.

The use case on Large flow scheduling on extended ALTO topologies in the present section illustrates one modality of ALTO topology service, that would expose several paths between end to end (src, dst) pairs, computed w.r.t. one of more metrics, possibly under given constraints. On top of this enriched topology service, non real-time applications may also choose the time of data/resources transfer, taking thus advantage of a richer set of decision variables.

The use case "Large Flow Scheduling" of presentation [slides-88-alto-5-topology]can thus be adapted as follows:

- o Step1 - obtain the set T transfer tasks {(src, dest, data)}

- o Step2 - identify one or more paths for each (src, dst): several information sources exist among which:

  - * (a) ALTO CostMap with a "path" metric, // not specified here

  - * (b) an ALTO Topology Service providing a path computation hint (e.g. w.r.t. routingcost and/or other metrics)

- o Step 3 - while T not empty:

  - * 1 - query for example values for some metric ’available bandwidth’ on paths:

    - + to this end, query the values in the ALTO ’calendar’ Mode: on the selected (src, dst) for a set of time intervals. With this mode, the ALTO client will receive an array of values, each applicable to a time slot .

  - * 2 - schedule data transfer at the time slots corresponding to the preferred value.

2.5. Time-sensitve TE metrics Calendaring

Draft [draft-wu-alto-te-metrics] , proposes to extend the set of ALTO metrics with 11 ALTO traffic engineering (TE) metrics to reflect measurement on network delay, jitter, packet loss, hop count, and bandwidth. ALTO TE metrics that are time-sensitive, either by nature such as bandwidth and delay related metrics, or due to "normally" changing network conditions or both.

The values of ALTO TE metrics are typically collected from routing protocols and provided in a non-real time manner. In "normally" changing network conditions, TE metric values remain uniformly distributed over given time intervals and can be aggregated over bigger time intervals of periodic patterns. For example, an ALTO Server may collect values for e.g. delay from a routing protocol produced by measurements done every second over a measurement period of 30 seconds. The ALTO Server may then aggregate these values over two measurement periods (i.e. 60 seconds) and repeat the operation as it wishes. Then every hour, the ALTO Server provides these delay values in ’calendar’ mode, encoded as an array of 60 values, assumed to estimate network performance statistics on each minute of this hour.

Another example is Bandwidth Calendaring. 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.

A Cost calendar provided by an ALTO server can support the scheduled bulk data replication application with better efficiency since it can alleviate the burden of processing on network elements. This requires the ALTO server to maintain the calendared TE cost metrics on the end to end paths associated to data transfer.

To support cost calendaring for these time-sensitive ALTO TE metrics, the network topology and the dynamicity of the traffic need to be considered. 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 request at a pace that would be overload both the ALTO Client and the Server.

3. Design considerations for an ALTO calendar

This section enumerates a set of challenges in designing the calendaring specifications, and will be updated upon discussions in the ALTO WG.

An ALTO Cost calendar provided by the ALTO Server is an array of values for a given metric, where each value corresponds to a time interval which length is specified for this metric in the IRD, together with other attributes describing the time scope of the calendar. 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 Cost Map service, filtered or not, is also applicable as long as the size of the Map is manageable.

Given a cost calendar (i.e., a sequence of cost values such as [1, 2, 3, 4]), an ALTO client needs meta information to interpret the values (Q1) Which network metric (e.g., routingcost, or latency) do the numbers represent (Q2) Are the values absolute values or relative ranking order or strings ? (i.e., numerical or ordinal or strings) and (Q3) What is the time associated with each value ? RFC7285 defines CostType, which consists of two fields named cost-metric and cost-mode, to provide information for Q1 and Q2. A design issue in providing ALTO cost calendar is to provide information for Q3.

3.1. Purpose of an ALTO calendar

A calendar is used to schedule transfers of application data or services and has several characteristics:

- o the Calendar values are assumed to be stationary on each time interval,

- o the ALTO Server may provide values on past time periods that can be interpreted as historical experience and used to anticipate future cost values,

- o the ALTO Server may provide stationary values on present or future time periods that can be interpreted as predictions on cost values,

- the ALTO Server may provide stationary values on time intervals covering the past, and/or present and/or future.

- for metrics provided with units and claiming to be aggregated from network measurements, the values can be interpreted as estimations.

- For abstracted metrics provided with no units such as the ’routingcost’ defined in the base ALTO protocol or abstracted unitless scores on network performances such as some potential ’bandwidth score’ or ’unreliability cost’, the values can be interpreted as network provider preferences.

Note that we distinguish between "estimates" that we see as value aggregations represented with units such as bytes, seconds, percentage and "preferences" that we see as abstracted costs or scores w.r.t. a metric or state such as ’routingcost’, ’bandwidthscore’, ’link quality’.

The method used to generate the estimation and aggregation of measured values is currently outside the scope of this draft and expected to be documented in the applicable metric definition document.

3.2. Design requirements for an ALTO calendar

TO BE COMPLETED IN FURTHER DRAFT VERSIONS

An ALTO Calendar can be seen as a cyclic value array pattern that is valid for a certain time period with specified beginning date, duration and number of time intervals.

- needs to convey cyclic network provider preferences expressed w.r.t. given ALTO metric values (e.g., hourly, daily, weekly measurement/prediction)

- needs to convey cyclic network status if the ALTO Server claims to provide aggregated information on network status (e.g., hourly, daily, weekly measurement/prediction)

- needs to be able to convey the result of a particular instance of time (e.g., to convey predicted network status during a maintenance outage on July 4, 2014 from 5-7pm)

- needs at least the following attributes to report on cyclic patterns:

  * generic time zone,

  - applicable time interval for each calendar value (measurement estimation with units or unitless preference value) : combining <nb-int-unit> and <interval-unit> to reflect for example: 1hour, 2minutes, 1week, 1month

  - date range of the Calendar, e.g. number of intervals allowing to derive the calendar time range in terms of: year, month, week, day, hour, min, secs

- needs to expose validity period of the calendar: indicating when the next ALTO Calendar for this date range should be fetched if needed,

- needs to provide time stamps:

  - last-update-time: specifying when the metric values were last computed ,

  - next-update-time: specifying when the calendar values will be re-computed, indicating thus when an ALTO client should fetch an update if it uses a Calendar.

  - calendar-start-date: specifying when the current already computed calendar starts,

  - next-calendar-start-date: specifying when the already computed calendar will have different values, indicating thus that the ALTO client should fetch the next pre-computed calendar

It may be useful to keep a cyclic network status with date, in case of exceptional predicted events such as New Year evening on a Tuesday or any worldwide event generating a lot of traffic.Traffic calendars may be particularly useful in such cases.

4. ALTO extensions for a Cost Calendar

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

Specifications on the cost "calendar" attributes are proposed here and will be completed in further versions of this draft, upon discussion with the ALTO WG.

The format of ALTO requests and responses will be specified in further versions of this draft, as in particular it may be necessary that the ALTO response indicates the computation and validity dates of the provided ALTO Calendar.

4.1. ALTO Cost-Mode: Calendar

This draft introduces a new ALTO Cost Mode called "calendar". This mode applies preferably to Costs that can be expressed in a single-valued Cost Mode. In that sense, when the "numerical" mode is available for a Cost-Type, the cost expressed in the "calendar" mode is an extension of its expression from one value in the "numerical" mode to an array of several values varying over time.

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

DISCUSSION: The current draft focuses on Calendars representing values encoded in the 'numerical' mode. However, Calendars should also be able to represent time sensitive values represented by strings, such as "medium", "high", "low". To support this, one option is to extend the Cost Mode to be a combination (attribute) indicators, for example cost-mode : "ordinal;calendar"

cost-mode : "ordinal;calendar"

We chose ";" as separator to be consistent with existing formats such as the HTTP list of multiple options (e.g., Accept).

4.2. ALTO Calendar attributes in the IRD

To ensure that the application client understands the provided information in the cost calendar in an unambiguous way, we specify the Calendar attributes in the ALTO IRD "meta" information, that defines the time scope of the "calendared" cost values. The Calendar attributes in the IRD are meant to carry constant dateless values.

- o time-interval-size:

  - * expresses the unit in which the duration of an ALTO calendar time interval duration is expressed 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". These vales mean that each calendar value is provided on a time interval that lasts respectively 5 minutes and 2 hours.

- o numb-intervals:

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

- o repeat:

  * an integer number representing the number of times that the Calendar pattern repeats. That is : if for example a given daily pattern is represented Calendar is made of "num-intervals" = 24 cost values each applicable to a time slot lasting "time-interval-size" = 1 hour and the value of ’repeat’ is 5, the client can interpret that 5 daily patterns starting from the date indicated in the Server response have identical values.

- Attributes ’time-interval-size’ and ’numb-intervals’, when mutlipled, reflect 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 ’numb-intervals’ has the value 12, then the duration of the provided calendar is "1 hour". Note also that in this example, a 5 minutes interval may cover the aggregation of real TE measurements done every 30 seconds, but this latter aspect is outside the scope of this draft as it is to be specified in the definition of the ALTO metric.

- Attribute ’repeat’ reflects the frequancy at which the patterns represented by a Calendar may change. This information is completed by the ’start’ attribute provided in Server responses to calendar requests.

NOTE that: To cope with existing representation fomats, further versions will re-name these attributes. The current proposal for renaming in further versions is to replace "time-interval-size" by "interval" and "num-interval" by "count" :

4.3. Example of calendared information resources in the IRD

This section describes an example IRD and related ALTO calendar transaction in a scenario where an ALTO Server offers the Calendar Mode for several Cost Types that are either specified in the base ALTO protocol or proposed in other drafts see [draft-wu-alto-te-metrics] or suggested here as examples, like a cost metric reporting on measured packet loss and called ’TEpktloss. The provided example transactions are based on the use cases of section 2.

These examples describe situations where a client has the choice of trading content or resources with several Endpoints and needs to decide with which Endpoint it will trade and at what time. For instance, one may assume that the Endpoints are spread over different time-zones, or have intermittent access. The ALTO Calendar mode specified below allows these clients to retrieve Endpoint Cost Maps valid for a certain timeframe (e.g. 24 hours), and get a set of values, each applicable on a specified time interval (e.g. 1 hour). Thus the application can optimize the needed data transfer according to this information.

In the example IRD of the present draft, the available Endpoint Costs metrics are: "routingcost", "AShopcount", ’TEpktloss’ and ’Availbandwidth’. "routingcost" and "AShopcount" are available in the "numerical" Cost Mode. ’TEpktloss’ , ’Availbandwidth’ and "routingcost" as well are available in the "calendar" Cost Mode.

We suppose that the ALTO Client GETs the IRD on Tuesday July 1st 2014 at 13:00

o The Calendar for ’TEpktloss’’: is an hourly pattern that consists of 12 values provided each on a time interval of 5 minutes, and the values are the same for the next 2 hours.

o The Calendar for ’Availbandwidth’: is a daily pattern that consists of 12 values provided each on time intervals of 2 hours, with the first interval starting at 0h00. This information is typically used to enable applications to see which time intervals in a day are the most favorable to operate, and which "busy " time intervals should be avoided. The pattern is the same for the next "repeat" = 7days.

o The Calendar for ’routingcost’: is a daily pattern that consists of an array of 24 time intervals lasting each 1 hour. The routingcost calendar covers a 1 day period, starting at midnight. This may be applicable for networks with poor or intermittent connectivity where the operator may integrate monetary as well as network performance metrics in the provided ’routingcost’ values.The pattern is the same for the nex "repeat" = 4 days.

4.3.1. Example IRD with ALTO cost Calendars

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

o "http://alto.example.com/endpointcost/lookup", in which the ALTO Server offers the numerical mode for metrics "routingcost" and "AShopcount".

o "http://alto.example.com/endpointcost/calendar/lookup", in which the ALTO Server provides "calendar" mode for metrics ’TEpktloss’ and ’Availbandwidth’ and ’routingcost’.

For Cost Type ’calendar-routing’, this example assumes that the 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 example ALTO response shown in a further section also illustrates how specific calendar attributes allow an ALTO client to fetch 3 Calendars instead of 7 and thus to reduce the volume of on-the-wire data exchange. For Cost Type ’calendar-routing’ , the IRD provides a value for attribute ’num-calendars’ which is equal to 3.

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


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

   {
     "meta" : {
        "cost-types": {
           "num-routingcost": {
              "cost-mode" : "numerical",
              "cost-metric" : "routingcost"
              },
           "num-AShopcount": {
              "cost-mode" : "numerical",
              "cost-metric" : "hopcount"
              },
           "calendar-TEpktloss": {
              "cost-mode"  : "calendar",
              "cost-metric": "TEpktloss",
              "description": {
                  "time-interval-size" : "5 minute",
                  "numb-intervals" : 12,
                  "repeat" : 2
              }
           },
           "calendar-bw": {
              "cost-mode"  : "calendar",
              "cost-metric": "Availbandwidth",
              "description": {
                  "time-interval-size" : "2 hour",
                  "numb-intervals" : 12,
                  "repeat" : 7
              }
          },
          "calendar-routing": {
             "cost-mode"  : "calendar",
             "cost-metric": "routingcost",
             "description": {
                 "time-interval-size" : "1 hour",
                 "numb-intervals" : 24,
                 "repeat: 4"

         }
         ... other meta ...
       },

    "resources" : {

       ... usual ALTO resources such as Network Map, Cost Maps ...

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

4.4. ALTO Calendar information in ALTO responses

ALTO responses convey additional attributes with usually non constant values that inform the ALTO Client about the next date at which the calendar values stored in the ALTO Server will change and at which time updates calendar values will be uploaded in the ALTO Server. A number of Calendar attributes in ALTO responses are dates. The reference time zone for the provided values is GMT. Indeed, the option chosen to express the time format is the HTTP header fields formats such as:

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

o calendar-start-time:

   * the date corresponding to the first value in the calendar values array

o time-interval-size: as defined for the IRD

o numb-intervals: as defined for the IRD

o repeat: as defined for the IRD

- Attribute 'calendar-start-time' indicates when the calendar provided to the ALTO client starts. 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.

Discussion: like for the attributes of the IRD, "calendar-start-time" can be renames "start" to comply with existing formats.

4.4.1. Example transaction for a routingcost Calendar to face intermittent connectivity

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

The present example also illustrates how attributes "repeat" and "calendar-start-time" allow an ALTO client to fetch 3 Calendars instead of 7 and thus to reduce the volume of on-the-wire data exchange, because the 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 5-7pm, or holiday such as New Year evening)

```
POST endpointcost/calendar/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" : "calendar", "cost-metric" : "routingcost"},
    "endpoints" : {
      "srcs": [ "ipv4:192.0.2.2" ],
      "dsts": [
        "ipv4:192.0.2.89",
        "ipv4:198.51.100.34",
        "ipv4:203.0.113.45"
      ]
    }
  }


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

  {
    "meta" : {
      "calendar-start-time" : Tue, 1 Jul 2014 00:00:00 GMT,
      "time-interval-size" : "1 hour",
      "numb-intervals" : 24,
      "repeat": 4
    },
    "cost-type" : {"cost-mode" : "calendar", "cost-metric" : "routingcost"},
    "endpoint-cost-calendar-map" : {
        "ipv4:192.0.2.2": {
          "ipv4:192.0.2.89"    : [7, ... 24 values],
          "ipv4:198.51.100.34" : [4, ... 24 values],
          "ipv4:203.0.113.45"  : [2, ... 24 values]
        }
    }
  }
```

NOTE also: that the Calendar could enumerate 24*7 hourly values to represent the pattern for such a week.

4.4.2. Example transaction for a 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 of sections 2.1 and 2.5.

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

```
POST endpointcost/calendar/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" : "calendar", "cost-metric" : "Availbandwidth"},
    "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" : {
      "calendar-start-time" : Tue, 1 Jul 2014 00:00:00 GMT,
      "time-interval-size" : "2 hour",
      "numb-intervals" : 12,
      "repeat" : 7
    },
    "cost-type" : {"cost-mode" : "calendar", "cost-metric" : "Availbandwidth"},
    "endpoint-cost-calendar-map" : {
        "ipv4:192.0.2.2": {
          "ipv4:192.0.2.89"    : [7, ... 12 values],
          "ipv4:198.51.100.34" : [4, ... 12 values],
          "ipv4:203.0.113.45"  : [2, ... 12 values]
        }
    }
  }
```

5. Use cases for ALTO Cost Schedule

This section introduces use cases showing the benefits of providing ALTO Cost values in ’calendar’ mode. 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 Cost Map service, filtered or not, is also applicable as long as the size of the Map is manageable.

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

5.1. Bulk Data Transfer scheduling

Large Internet Content Providers (ICPs) like Facebook or YouTube, as well as CDNs rely on data replication across multiple sites 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 pattern.

In the examples above, data needs to be replicated across the various locations of a Internet Content Provider (ICP), leading to bulk data transfers between datacenters on a diurnal pattern.

In the mean time, 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 the transfer should not infer with the user peak demand to avoid degradation of user experience and to decrease billing costs for the datacenter operator by leveraging off-peak hours for the transfer. This peak demand typically follows a diurnal pattern according to the geographic region of the datacenter.

As a result, it would be very helpful to let these ICPs to have a good knowledge about the link utilization patterns between the different datacenters from the networks before making a more intelligent scheduling decision. While this usage data today already is gathered and also used for the scheduling of data transfer, provisioning this data gets increasingly complex with the number of CDN nodes and in particular the number of datacenter operators that are involved. For example, privacy concerns prevent that this kind of data is shared across administrative domains. The ALTO Cost

Calendar specified later in this document avoids this problem by presenting an abstracted view of time sensitive utilization maps through a dedicated ALTO service to allow ICPs a coherent scheduling of such data transfers across administrative domains.

5.2. Endsystems with limited connectivity or access to datacenters

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

Wireless 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 end systems using resources located in datacenters and trading content and resources scattered around the world. For non-real time applications, the interaction with Endpoints can be scheduled at the time slots corresponding to the best possible 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. Example applications are similar to the ones cited in the previous paragraph.

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

5.3. SDN Controller guided access to application endpoints

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

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

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

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

To achieve this objective, the SDN controller can:

1. get the network state history from its controlled network elements through its southbound API

2. possibly derive an estimation or a prediction of these values over given time frames

3. compute estimates and/or network provider preferences on end to end paths and store their abstraction in an ALTO Server in the form of ALTO Cost Calendar values defined for different time periods

4. deliver these values to the SDN applications via the ALTO Endpoint Cost Service, as estimations covering the past and/or the future and/or preferences.

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,

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

5.4. Large flow scheduling on extended ALTO topologies

[draft-yang-alto-topology-00] presents initial thinking on extending ALTO for topology exposure services, that would provide flexible abstractions based on the raw network topology. Among other features, an ALTO topology may expose several paths between a source (src) and destination (dst), or topology details may be provided on restricted parts. This work was presented to the ALTO WG at IETF88.

The presentation slides [slides-88-alto-5-topology] on [draft-yang-alto-topology-00] expose a use case entitled "Large Flow Scheduling". This case includes a "daylife example" where a Google Map service proposes multiple routes between 2 points A and B, each calculated w.r.t. length and estimated time. For each of these selected paths, the map service exposes a time-sensitive qualitative value taking 4 values between Slow and Fast. A user of this application may thus organize its transfer w.r.t. metrics, paths and time, provided s/he does not have to commute immediately.

The use case on Large flow scheduling on extended ALTO topologies in the present section illustrates one modality of ALTO topology service, that would expose several paths between end to end (src, dst) pairs, computed w.r.t. one of more metrics, possibly under given constraints. On top of this enriched topology service, non real-time applications may also choose the time of data/resources transfer, taking thus advantage of a richer set of decision variables.

The use case "Large Flow Scheduling" of presentation [slides-88-alto-5-topology]can thus be adapted as follows:

- o Step1 - obtain the set T transfer tasks {(src, dest, data)}
- o Step2 - identify one or more paths for each (src, dst): several information sources exist among which:
  - * (a) ALTO CostMap with a "path" metric, // not specified here
  - * (b) an ALTO Topology Service providing a path computation hint (e.g. w.r.t. routingcost and/or other metrics)
- o Step 3 - while T not empty:
  - * 1 - query for example values for some metric ’available bandwidth’ on paths:
    - + to this end, query the values in the ALTO ’calendar’ Mode: on the selected (src, dst) for a set of time intervals. With this mode, the ALTO client will receive an array of values, each applicable to a time slot .
  - * 2 - schedule data transfer at the time slots corresponding to the preferred value.

5.5. Time-sensitve TE metrics Calendaring

Draft [draft-wu-alto-te-metrics] , proposes to extend the set of ALTO metrics with 11 ALTO traffic engineering (TE) metrics to reflect measurement on network delay, jitter, packet loss, hop count, and bandwidth. ALTO TE metrics that are time-sensitive, either by nature such as bandwidth and delay related metrics, or due to "normally" changing network conditions or both.

The values of ALTO TE metrics are typically collected from routing protocols and provided in a non-real time manner. In "normally" changing network conditions, TE metric values remain uniformly distributed over given time intervals and can be aggregated over bigger time intervals of periodic patterns. For example, an ALTO Server may collect values for e.g. delay from a routing protocol produced by measurements done every second over a measurement period of 30 seconds. The ALTO Server may then aggregate these values over two measurement periods (i.e. 60 seconds) and repeat the operation as it wishes. Then every hour, the ALTO Server provides these delay values in ’calendar’ mode, encoded as an array of 60 values, assumed to estimate network performance statistics on each minute of this hour.

Another example is Bandwidth Calendaring. 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.

A Cost calendar provided by an ALTO server can support the scheduled bulk data replication application with better efficiency since it can alleviate the burden of processing on network elements. This requires the ALTO server to maintain the calendared TE cost metrics on the end to end paths associated to data transfer.

To support cost calendaring for these time-sensitive ALTO TE metrics, the network topology and the dynamicity of the traffic need to be considered. 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 request at a pace that would be overload both the ALTO Client and the Server.

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

[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.

```
   [article-gslh-alto-sdn]
              V. Gurbani, M. Scharf, T.Lakshman, and V. Hilt, ,
              "Abstracting network state in Software Defined Networks
              (SDN) for rendezvous services, IEEE International
              Conference on Communications (ICC) Workshop on Software
              Defined Networks (SDN)", June 2012.

   [draft-jenkins-alto-cdn-use-cases-01]
              B. Niven-Jenkins (Ed.), G. Watson, N. Bitar, J. Medved, S.
              Previdi, , "Use Cases for ALTO within CDNs, draft-jenkins-
              alto-cdn-use-cases-01", June 2011.

   [draft-randriamasy-multi-cost-alto]
              S. Randriamasy, Ed., W. Roome, N. Schwan, , "Multi-Cost
              ALTO (work in progress), draft-randriamasy-alto-multi-
              cost-07", October 2012.

   [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-xie-alto-sdn]
              H. Xie, T. Tsou, D. Lopez, H. Yin, , "Use Cases for ALTO
              with Software Defined Networks (work in progress), draft-
              xie-alto-sdn-extension-use-cases-01", January 2013.

   [draft-yang-alto-topology-00]
              Y. Yang, , "ALTO Topology Considerations (work in
              progress)", July 2013.

   [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 (editor)
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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 10, 2015

S. Randriamasy
W. Roome
Alcatel-Lucent
N. Schwan
Thales Deutschland

March 9, 2015

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

Abstract

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

The present draft proposes a simple way to extend the information provided by the current ALTO protocol in two ways. First, including information on multiple Cost Types in a single ALTO transaction provides a better mapping of the Selected Endpoints to needs of the growing diversity of Content and Resources Networking Applications and to the network conditions. Second, one ALTO query and response exchange on N Cost Types is faster and more efficient than N single cost transactions. All this also helps producing a faster and more robust choice when multiple Endpoints need to be selected. Last, the draft proposes to enrich the filtering capabilities by allowing constraints involving several metrics combined by several types of logical operators. This allows the applications to set finer requirements and above all to include compromises on those requirements.

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

Table of Contents

1. Introduction

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

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

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

The ALTO protocol [RFC7285] restricts ALTO Cost Maps and Endpoint Cost services to only one Cost Type and Cost Mode per ALTO request. To retrieve information for several Cost Types, an ALTO client must send several separate requests to the server.

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

Along with multi-cost values queries, the filtering capabilities need to be extended to allow constraints on multiple metrics. The base protocol allows optional constraints in the input parameters to a request for a Filtered Cost Map or the Endpoint Cost Service. The ’constraints’ member is an AND-combination of expressions that all apply to the (single) requested Cost Type. It is therefore necessary to allow constraints on multiple metrics. Beyond that, 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 address this issue, this draft proposes to extend the base protocol by both allowing to combine constraints on multiple metrics and relating these constraints with a logical ’AND’ and a logical ’OR’. This allows an application to make compromises such as: "select solutions with either (moderate ’hopcount’ AND high ’routingcost’) OR (higher ’hopcount’ AND moderate ’routingcost’)".

This draft is organized as follows: section 3 exposes use cases motivating the introduction of new Cost Types and why multi-cost transactions are useful. Section 4 identifies the core ALTO protocol extensions that are required or recommended to support requests and responses on multiple Cost Types in one single transaction. Section 5 specifies the extended constraints on mutli-cost values. Section 6 specifies the protocol extensions for Multi-Cost ALTO transactions and provides examples.

2. Application Scope And Terminology

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

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

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

This draft uses terms defined as follows:

- Endpoint (EP): can be a Peer, a CDN storage location, a physical server involved in a virtual server-supported application, a Party in a resource sharing swarm such as a computation Grid or an online multi-party game.

- Endpoint Discovery (EP Discovery) : this term covers the different types of processes used to discover the eligible endpoints.

- Network Service Provider (NSP): includes both ISPs, who provide means to transport the data, and Content Delivery Networks (CDNs) who care for the dissemination, persistent storage and possibly identification of the best/closest content copy.

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

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

3. Uses Cases For Using Multiple Costs

The ALTO protocol specification in [RFC7285] focuses on the basic use case of optimizing routing costs in NSP networks. Upcoming use cases however will require both new Cost Types and new Endpoint Properties. Recent ALTO use cases now extend to CDNs, Data centers and other applications that need additional information to select their Endpoints or handle their PIDs. The needed Cost Types depend on the QoE requirements that are specific to the applications. Moreover, the cost values that they may use may change more rapidly than assumed up to now.

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

3.1. Use Cases For Using Additional Costs

ALTO Cost Types and Endpoint Properties are registered in two registries maintained by IANA. The ALTO Cost Type registry ensures that the Cost Types that are represented by an ALTO Cost Map are unique identifiers, and it further contains references to the semantics of the Cost Type. The ALTO specification registers 'routingcost' as a generic measure for routing traffic from a source to a destination. In a similar way the ALTO Endpoint Property

Registry ensures uniqueness of ALTO Endpoint Property identifiers and provides references to particular semantics of the allocated Endpoint Properties. Currently the 'pid' identifier is registered, which serves as an identifier that allows aggregation of network endpoints into network regions. Both registries accept new entries after Expert Review. New entries should conform to the respective syntactical requirements, and must include information about the new identifier, the intended semantics, and the security considerations. One basic example advocating for multiple Cost Type transactions is an Application Client looking for destination Endpoints or Source/ Destination PID pairs yielding jointly the lowest 'routingcost' and path delay. We hereby assume that 'routingcost' values report some monetary cost and that the Application Client chooses to rely on the hopcount to reflect the path delay.

3.1.1. Delay Sensitive Overlay Applications

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

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

3.1.2. Selection Of Physical Servers Involved In Virtualized Applications

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

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

3.1.3. CDN Surrogate Selection

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

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

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

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

3.1.4. Some Proposed Additional Properties And Costs

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

Proposed new Endpoint properties and costs include:

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

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

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

3.2. Use Cases For Multi-Cost ALTO Transactions

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

The Multi-Cost ALTO Services propose to:

o include several Cost Types (and/or Cost Modes) in an ALTO client's Cost Map and Endpoint Cost request,

o provide several Cost Type values (and/or Cost Mode) in an ALTO server's response, instead of one.

The primary reasons to use Multi-Cost ALTO are:

o Optimizing time and bandwidth: a single ALTO response with a Multi-Cost cost map with three separate Cost Type values takes much less network bandwidth, and fewer CPU cycles, than three separate ALTO requests for three complete single-cost cost maps. The motivation also holds for the Endpoint Cost Service. Multi-Cost ALTO services can straightforwardly provide a more complete set of cost information.

o Facing unpredictable and/or rapid value changes: an ALTO client can get a consistent snapshot of several different rapidly-varying Cost Type values.

3.2.1. Optimized Endpoint Cost Service

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

3.2.2. Optimized Filtered Cost Map Service

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

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

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

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

3.2.3. Cases Of Unpredicable Endpoint Cost Value Changes

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

It becomes a necessity in case of unpredictable and/or rapid value changes on at least one of the ALTO Cost Types. The term 'rapid' here means "Typical update intervals [that] may be several orders of magnitude longer than the typical network-layer packet round-trip time (RTT)", as described in [RFC6708], up to a couple of minutes.

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

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

In some cases the Application client wants to use the ALTO information to perform multi-variate optimization on several Cost

Type values. In order for the optimization to be reliable, it is recommended that the Cost Type values are provided in ’numerical’ Cost Mode. Therefore the requested Cost Mode for the applicable Cost Types SHOULD be ’numerical’.

3.2.3.1. Case Of A Multi-Cost ALTO Query Upon A Route Change

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

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

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

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

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

                                                  .-----.
                 O ---------- O ------------- | EP2 |
               /                                  '-----'
             /
           /                                   .-----.
     Source ----------------------- O ---- | EP1 |
                                               '-----'


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

           - Route to EP1 has changed. Hopcount is now 3

           ==> EP2 is selected because routingcost is lower than for
           EP1, with the same hopcount value
                                                 .-----.
                 O ---------- O -------------| EP2 |
               /  \                              '-----'
             /     '-----.
           /             '------.          .-----.
   Source ---------- X --------- [O] ---- | EP1 |
                                               '-----'
```

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

3.2.3.2. Case Of A Multi-Cost ALTO Query Upon A Cost Value Change

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

                                             .-----.
                O ---------- O ------------- | EP2 |
              /                              '-----'
            /
          /                               .-----.
        O ------------------------ O ---- | EP1 |
                                          '-----'


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

                                            .-----.
                O ---------- O -------------| EP2 |
              /                             '-----'
            /
          /                               .-----.
        O ------------------------ O ---- | EP1 |
                                          '-----'
```

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

4. ALTO Protocol Updates Needed To Support Multi-Cost Transactions

To allow running Multi-Cost ALTO Services some minor changes in the base protocol are needed. The main updates consist of changing the JSON type of the value taken by the costs and add a few members to the objects describing the information resources, client requests and server responses to Multi-Cost information services.

As written in the introduction, this section relies on Section {11.2.3.6} of the ALTO protocol draft, see [RFC7285] , which allows protocol extensions to encode cost values as the 'JSONValue' data type.

4.1. List Of ALTO Protocol Updates Required And Recommended

The following updates to the ALTO protocol ([RFC7285]) are required or recommended to support multi-cost ALTO transactions. The new resulting JSON formats are specified in the next sections. Section references ({##}) are to the ALTO protocol document.

The applicable ALTO information resources are: Cost Map, Filtrered Cost Map and Endpoint Cost Map, becoming respectively Multi-Cost Map, Filtered Multi-Cost Map and Endpoint Multicost Map provided with the same media-type.

- o Updates required in the format of objects member(s):

    * Objects DstCosts (to destination PIDs, {11.2.3.6}) and EndpointDstCosts (to destination Endpoints, {11.5.1.6}): the cost value member evolves to an array of JSONValues.

    * Object ReqFilteredCostMap {11.3.2.3} and ReqEndpointCostMap {11.5.1.3}: a new member named "multi-cost-types" is introduced. This member is an array of 1 or more cost types for which a Multi-Cost ALTO Client requests values. Each cost type of the array is encoded as specified in {10.7}.

- o Updates recommended in the object structure:

    * The capabilities for the Filtered Cost Map Service {11.3.2.4} and the Endpoint Cost Map Service {11.5.1.4} need to be extended with a new member entitled "max-cost-types" giving the maximum number of Cost Types allowed in a Multi-Cost request and response.

    * The capabilities for the Multi Cost Map Service need to include a new member named "multi-cost-type-names" and giving the list of Cost Types that are provided in a Multi-Cost Map requested via a GET method.

    * The capabilities for the Cost Map Service, the Filtered Cost Map Service {11.3.2.4} and the Endpoint Cost Map Service {11.5.1.4} need to be extended with a new member named "multi-cost-type-names" and giving the list of Cost Types that may be included in the constraints member of a request.

    * In a Server response to {11.3.2.6} filtered cost map request and {11.5.1.6} and to filtered endpoint cost service request: a new member named "multi-cost-types" and described above is added to the "meta" field of the response.

- o Rules required on object member description:

  * Order in which the multiple cost values are provided in the responses.

4.2. Updates Required In The Member Format Of Objects

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

The term Single Cost qualifies the items as they are specified in the current ALTO protocol.

4.2.1. Cost Value Encoded In array of JSON values

The fundamental change to support multi-cost is to encode the cost values as an array of JSONValues. This way, the cost between two PIDs or two Endpoints can be represented in a generic way:

- o with several Cost Types,

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

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

```
["num-routingcost", "num-hopcount", "string-status"]
```

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

```
[23, 6, "medium"]
```

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

4.2.2. Scalar ’cost-type’ Member Replaced By Array ’multi-cost-types’ Member

In the base protocol, the various single-cost-map services use a scalar "cost-type" member in the "meta" section to indicate the cost metric and cost mode of the returned values.

In Multi-Cost ALTO, the multi-cost-map services use an array member named "cost-types" instead. The array elements are in the same format as the "cost-type" member in single cost maps, and the order corresponds to the order of values in the array values in the multi-cost map.

Alternatively, we could use the same member name, but define it as an array for multi-cost services. This would simplify some things for a client, but complicate others. Overall, we believe it is easier for a client to use a new member name than to overload the type of an existing member name.

4.2.3. Rule On Cost Value Order In ALTO Reponses

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

4.3. Updates Recommended In The Object Structure

Objects MultiCostMapCapability {11.2.3.4} and FilteredMultiCostMapCapability {11.3.2.4}: are extended with:

- o a new member a new member entitled "max-cost-types" giving the maximum number of Cost Types allowed in a Multi-Cost request and response and giving the maximum number of Cost Types in a response. The default value is set to 1 to avoid a multi-cost aware client requesting a multi-cost map from a server that does not support them.

- o a new member named "multi-cost-type-names" and giving the list of Cost Types that are provided in a Multi-Cost Map requested via a GET method.

5. Extended Constraints On Multi-Cost Values

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

NOTE: Constraint tests on multiple cost metrics are useful even when retrieving single costs, and we expect there will be proposals to add multi-cost constraint tests to the ALTO protocol, relating to the extensions proposed in this draft. Draft [draft-lee-alto-app-net-info-exchange] proposes in particular extensions to query values on a metric M1 with constraints on other metrics {M2, ... Mk}, that adds an interesting feature to extend ALTO constraints. This motivates the need to augment the capabilities in the IRD of the Filteredt Multi-Cost Map and Endpoint Multi-Cost Map with the extensive list of Cost-Types that may be included in the constraints of requests.

The base ALTO protocol allows optional contraints in the input parameters to a request for a Filtered Cost Map or the Endpoint Cost Service. The 'constraints' member is an array of expressions that all apply to the (single) requested Cost Type. The encoding of 'constraints' member, is fully specified in Section 11.3.2.3 of the base protocol as follows:

```
 A constraint contains two entities separated by whitespace:
(1) an operator,'gt' for greater than, 'lt' for less than,
'ge' for greater than or equal to, 'le' for less than or equal to,
or 'eq' for equal to
(2) a target cost value. The cost value is a number that MUST be
defined in the same units as the Cost Type indicated by the costtype
parameter
    ...
If multiple 'constraint' parameters are specified, they are
interpreted as being related to each other with a logical AND.
```

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

'routingcost' values belong to [6, 20)

5.1. Use Cases For Multi-Cost Multi-Operator Constraints

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

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

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

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

2. or solutions with higher ’hopcount’ and moderate ’routingcost’, for instance: ’hopcount’ values in [20,50] AND ’routingcost’ values in [30,100].

5.2. Extended constraints in Multi-Cost ALTO

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

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

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

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

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

6. Protocol Extensions for Multi-Cost ALTO Transactions

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

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

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

The following ALTO services have corresponding Multi-Cost extensions:

o Information Resources Directory: extended with multi-cost related URIs and associated capabilities.

o Cost Map Service: extended with the Multi-Cost Map Service,

o Cost Map Filtering Service: extended with the Multi-Cost Map Filtering Service,

o Endpoint Cost Lookup Service: extended with the Endpoint Multi-Cost Lookup Service.

6.1. Information Resources Directory

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

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

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

In this example, the ALTO IRD exposes Multi-Cost capabilities on cost types "num-routingcost", "num-hopcount", "num-pathoccupationcost", that can be combined in a request. The values on these metrics are provided in numerical mode. Values provided for cost-type string are in "string" mode.

For the "filtered-multicost-map" resource and the "endpoint-multicost-map" resource, the IRD exposes in its capabilities a member noted "testable-cost-types" that is the list of cost-types that are allowed to be included in the constraints of a request. Note that this set may be different than the set "multi-cost-type-names". The "endpoint-multicost-map" resource provides cost-values for Cost Types "num-routingcost", "num-hopcount" and "str-status" and supports constraints on "num-routingcost", "num-hopcount", "num-pathoccupationcost" where as it does not provide values on "num-

   pathoccupationcost" and does not supports constraints on "str-
   status".

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


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

   {
     "meta" : {
       "cost-types" : {
          "num-pathoccupationcost" : {
            "cost-mode" : "numerical",
            "cost-metric" : "pathoccupationcost"
          },
          "str-status" : {
            "cost-mode" : "string",
            "cost-metric" : "status"
          },
          "num-routing" : {
            "cost-mode" : "numerical",
            "cost-metric" : "routingcost"
          },
          "num-hopcount" : {
            "cost-mode" : "numerical",
            "cost-metric" : "hopcount"
          },
           .....
           Other ALTO cost types as described
           in current ALTO Protocol
           .....
         },
         "default-alto-network-map" : "my-default-network-map"
       },
     "resources" : {
         "my-default-network-map" : {
           "uri" : "http://alto.example.com/networkmap",
           "media-type" : "application/alto-networkmap+json"
         },
         "numerical-routing-cost-map" : {
            .....
            Single-cost Services as described
            in current ALTO Protocol
            .....
         },
         "rc-hc-multicost-map" : {
           "uri" : "http://alto.example.com/multi/costmap",
           "media-types" : ["application/alto-costmap+json"],
           "uses" : [ "my-default-network-map" ],
           "capabilities" : {
             "multi-cost-type-names" : [ "num-routing", "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" : 3,
             "cost-type-names" : [ "num-routingcost",
                                   "num-hopcount",
                                   "num-pathoccupationcost" ],
             "testable-cost-types": ["num-routingcost",
                                     "num-hopcount",
                                     "num-pathoccupationcost" ]
           }
         },
         "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" : 3,
             "cost-type-names" : [ "num-routingcost",
                                   "num-hopcount",
                                   "str-status" ],
             "multi-cost-type-names" : [ "num-routingcost",
                                         "num-hopcount",
                                         "str-status"],
             "testable-cost-types": ["num-routingcost",
                                     "num-hopcount",
                                     "num-pathoccupationcost" ]
         }
       }
     }
   }
```

6.2. Multi-Cost Map Service

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

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

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

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

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

6.2.1. Media Type

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

6.2.2. HTTP Method

This resource is requested using the HTTP GET method.

6.2.3. Input Parameters

None.

6.2.4. Capabilities

The capabilities of the URI providing this resource are defined by a JSON object of type FilteredCostMapCapabilities::

```
   object {
        JSONString multi-cost-type-names<1..*>;
      } MultiCostMapCapabilities;
```

with members

multi-cost-type-names The Cost Type names returned by this map.

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

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

A full cost map resource capabilities has either "cost-type-names" or "multi-cost-type-names", but not both. The former means it returns a Single Cost Map, the latter means it returns a multi-cost Map. Since this resource is requested via the GET method, the Server returns what it returns and the client has no choice.

6.2.5. Uses

The Resource ID of the Network Map which defines the PIDs used in this Multi Cost Map. An ALTO Server MUST NOT define two Multi Cost Maps with the same Network Map and set of Cost Types.

6.2.6. Response

The "meta" field of a Cost Map response MUST include the "dependent-vtags" key, whose value is a single-element array to indicate the Version Tag of the Network Map used, where the Network Map is specified in "uses" of the IRD.

The "meta" MUST also include the member "multi-cost-types", which is a JSONArray of the CostTypes in this Multi Cost Map.

The data component of a Multi Cost Map response is named "cost-map", which is a JSON object of type CostMapData, as defined in {11.2.3.6} of the ALTO protocol. This is identical to the format of the ALTO Cost Map response, except that the JSONValues are arrays rather than numbers. The values in the arrays correspond to the Cost Type listed at the same place in the 'multi-cost-types' array. This array MUST have the same size as the 'multi-cost-types' array, and the provided values MUST be in the same order as in the 'multi-cost-types' array.

The returned Multi Cost Map MUST include the required Path Costs for each pair of Source and Destination PID for which this information is available. If a cost value is not defined, the ALTO Server MUST replace that value in the array with the reserved JSON symbol 'null'. If no costs are defined for a pair of Source and Destination PIDs, so the Path Cost would be an array of nulls, the ALTO Server MAY omit the array for that pair.

6.2.7. Example

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

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

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

{
 "meta" : {
   "dependent-vtags" : [
     {"resource-id": "my-default-network-map",
      "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"
     }
   ],
   "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] },
   "PID2": { "PID1":[null,5], "PID2":[1,0],    "PID3":[15,9] },
   "PID3": { "PID1":[20,12],  "PID2":[null,1], "PID3":[1,0]  }
 }
}
```

6.3. Filtered Multi-Cost Map

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

Therefore applications may more likely use Cost Map information filtered w.r.t. the Cost types as well as the source/destination pairs of PIDs. This section specifies Filtered Multi-Cost Maps.

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

6.3.1. Media Type

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

6.3.2. HTTP Method

This resource is requested using the HTTP POST method.

6.3.3. Input Parameters

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

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

object {
  CostType    multi-cost-types<1..*>;
  JSONString  constraints<0..*>; [OPTIONAL] - TO BE UPDATED
  JSONArray   or-constraints<0..*>;   [OPTIONAL]
  PIDFilter   pids;                   [OPTIONAL]
} ReqFilteredMultiCostMap;
```

with members:

multi-cost-type-names The array of requested Cost Types for the returned cos
ts.
      Each listed Cost Type MUST be one of the supported Cost Types

```
      indicated in this resource's capabilities.

   constraints
      As specified in section {11.3.2.3} of RFC7285.
      The Client MUST specify this member in its requests for
      single cost services as specified in RFC7285.
      The Client MUST NOT specify this member in requests for
      multi-cost services.
      The Client MUST NOT specify both a 'constraint' and
      an 'or-constraints' parameter.
      NB: THIS TEXT ON SUPPORT OF BASE PROTOCOL SINGLE COST CONSTRAINTS
      WILL BE UPDATED IN NEXT VERSIONS

   or-constraints
      Defines an array of arrays of constraint strings.
      This parameter MUST NOT be specified if this resource's capabilities
      indicate that constraint support is not available.
      A constraint string is an array of additional constraints.
      That is the constraint strings of the array are related by
      logical ANDs. Each string in the
      constraint array MUST contain three entities separated by
      whitespace, in the following format:
             [index] op value
      'Index' is a number between 0 and the number of Testable Cost
      Types minus 1, and indicates the Cost Type to which this
      constraint applies.  (The square brackets ([]) surrounding
      'index' are required syntactic sugar. They serve as a
      reminder that 'index' is an array index, not a value to test,
      and they avoid unusual-looking constraints such as "1 ge 5".)
      'Op' is an operator: 'gt' for greater than, 'lt' for less
      than, 'ge' for greater than or equal to, 'le' for less than
      or equal to, 'eq' for equal to, or 'ne' for not equal to.
      'Value' is a target cost value to compare against the
      indicated Cost Type. For numeric Cost Types, 'value' MUST be
      a number defined in the same units as the Cost Type indicated
      by 'index'.  ALTO servers SHOULD use at least IEEE 754
      doubleprecision floating point [IEEE.754.2008] to store the
      cost value, and SHOULD perform internal computations using
      double-precision floating-point arithmetic.  For string Cost
      Types, 'value' MUST be a string enclosed in single quotes (').
      For array-valued Cost Types, 'eq' is true iff one of the
      Cost Type values is equal to 'value', and 'ne' is true iff
      none of the Cost Type values are equal to 'value'.  The other
      operators are not defined for array-valued Cost Types.

      The "or-constraints" member defines an array of arrays of
      constraint strings in the format : [index] op value
      The ALSO server MUST return costs that satisfy all constraints
```

in one or more of the inner lists, and no other costs. That is, 'or-constraints' is the logical OR of ANDs.

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

6.3.4. Capabilities

The URI providing this resource supports all capabilities documented in Section 6.2.4 (with identical semantics), plus additional capabilities. In particular, the capabilities are defined by a JSON object of type FilteredMultiCostMapCapability:

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

with members:

cost-type-names
    The array of cost types available from this service.

multi-cost-type-names
    The array of cost types available from this service.
    Its resence means that this resource can return
    a multi-cost map. A filtered cost map resource can have
    either "cost-type-names" or "multi-cost-type-names" or both
    in its capabilities. The former means it can return a single
    cost map, the latter a multi cost. The Client selects which.

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

cost-constraints If true, then the ALTO Server allows cost
    constraints to be included in requests to the corresponding URI.
    If not present, this member MUST be interpreted as if it specified
    false.

Note that a filtered cost map resource can have either "cost-type-names" or "multi-cost-type-names" or both in its capabilities. The former means it can return a single cost map, the latter a multi cost. The Client selects which one its wants.

6.3.5. Uses

The Resource ID of the Network Map which defines the PIDs used in this Filtered Multi Cost Map.

6.3.6. Response

The response is the same format as for the Multi Cost Map Service (Section 6.2.6). The returned Cost Map MUST NOT contain any source/destination pair that was not indicated (implicitly or explicitly) in the input parameters. If the input parameters contain a PID name that is not currently defined by the ALTO Server, the ALTO Server

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

6.3.7. Example 1

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

{
  "multi-cost-types" : [
    {"cost-mode": "numerical", "cost-metric": "routingcost"},
    {"cost-mode": "numerical", "cost-metric": "hopcount"}
  ],
  "pids" : {
    "srcs" : [ "PID1" ],
    "dsts" : [ "PID1", "PID2", "PID3" ]
  }
}


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

{
  "meta" : {
    "dependent-vtags" : [
      {"resource-id": "my-default-network-map",
       "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"
      }
    ],
    "multi-cost-types" : [
      {"cost-mode": "numerical", "cost-metric": "routingcost"},
      {"cost-mode": "numerical", "cost-metric": "hopcount"}
    ]
  }
  "cost-map" : {
    "PID1": { "PID1": [1,6], "PID2": [5,23], "PID3": [10,5] }
  }
}
```

6.3.8. Example 2

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

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

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


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

{
  "meta" : {
    "dependent-vtags" : [
      {"resource-id": "my-default-network-map",
       "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"
      }
    ],
    "multi-cost-types" : [
      {"cost-mode": "numerical", "cost-metric": "routingcost"},
      {"cost-mode": "numerical", "cost-metric": "hopcount"}
    ]
  }
  "cost-map" : {
    "PID1": { "PID2": [5,23], "PID3": [10,5] }
    "PID2": { "PID2": [1,0]                  },
  }
}
```

6.4. Endpoint Multi-Cost Service

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

This service MAY be provided by an ALTO Server. It is important to note that although this resource allows an ALTO Server to reveal costs between individual endpoints, an ALTO Server is not required to do so. A simple alternative would be to compute the cost between two endpoints as the costs between the PIDs corresponding to the endpoints if these values are available for the requested Cost Types.

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

6.4.1. Media Type

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

6.4.2. HTTP Method

This resource is requested using the HTTP POST method

6.4.3. Input Parameters

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

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

     object{
         CostType    multi-cost-types<1..*>;
         JSONString  constraints<0..*>;      [OPTIONAL] // TO BE UPDATED
         JSONArray   or-constraints<0..*>;   [OPTIONAL]
         EndpointFilter endpoints;
     } ReqEndpointMultiCostMap;
```

with members:

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

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

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

endpoints A list of Source Endpoints and Destination Endpoints for which Path multiple Costs are to be returned. If the list of Source Endpoints is empty (or not included), the ALTO Server MUST interpret it as if it contained the Endpoint Address corresponding to the client IP address from the incoming connection (see Section 10.3 for discussion and considerations regarding this mode). The list of destination Endpoints MUST NOT be empty. The ALTO Server MUST interpret entries appearing multiple times in a list as if they appeared only once.

6.4.4. Capabilities

The capabilities are the same as described in Section 6.3.4.

6.4.5. Uses

As with the ALTO Endpoint Cost Service, the Endpoint Multi Cost Service MUST NOT use a Network Map.

6.4.6. Response

The "meta" field of an Endpoint Multi Cost response MUST include the "multi-cost-types" key, to indicate the Cost Types used.

The data component of an Endpoint Multi Cost response is named "endpoint-cost-map", which is a JSON object of type EndpointCostMapData, as defined in Section 11.5.1.6 of the ALTO protocol. This is identical to the format of the ALTO Cost Map response, except that the JSONValues are arrays rather than numbers. The values in the arrays correspond to the Cost Type listed at the same place in the 'multi-cost-types' array. This array MUST have the same size as the 'multi-cost-types' array, and the values in the MUST be in the same order as in the 'multi-cost-types' array.

6.4.7. Example

This is an example of requesting jointly cost values for "routingcost" and "hopcount" while using constraints to restrict the returned source/destination endpoints to those with 'pathoccupationcost' between 5 and 10, or 'hopcount' equal to 0, where 'pathoccupationcost' and 'hopcount' respectively have index 2 and 1 in the "testable-cost-types" member of the IRD capabilities of the "endpoint-multicost-map" resource. Only 2 of the 3 requested source/destination pairs meet the constraints.

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

{
  "multi-cost-types" : [
    {"cost-mode": "numerical", "cost-metric": "routingcost"},
    {"cost-mode": "numerical", "cost-metric": "hopcount"}
  ],
  "or-constraints" : [ ["[2] ge 5", "[2] le 10"],
                       ["[1] eq 0"] ],
  "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" : {
      "dependent-vtags" : [
        {"resource-id": "my-default-network-map",
         "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"
        }
      ],
      "multi-cost-types" : [
        {"cost-mode": "numerical", "cost-metric": "routingcost"},
        {"cost-mode": "numerical", "cost-metric": "hopcount"}
      ]
    }
    "endpoint-cost-map" : {
      "ipv4:192.0.2.2": {
        "ipv4:192.0.2.89"    : [1, 7],
        "ipv4:203.0.113.45"  : [3, 2]
      }
    }
  }
```

7. 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 [RFC7285],

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],

7.1. 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.

7.2. Information for IANA on proposed Endpoint Properties

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

8. Acknowledgements

The authors would like to thank Vijay Gurbani, Dave Mac Dysan, Dhruv Dhodi and Young Lee for fruitful discussions and comments on this draft and previous versions.

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.

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

9.2. Informative References

[ID-alto-protocol]
""ALTO Protocol" RFC 7285", September 2014.

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

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

[draft-jenkins-alto-cdn-use-cases-01]
""Use Cases for ALTO within CDNs" draft-jenkins-alto-cdn-use-cases-01", June 2011.

[draft-lee-alto-app-net-info-exchange]
"Information Exchange for High Bandwidth Applications in TE networks", October 2013.

[draft-randriamasy-alto-cost-schedule-01]
"ALTO Cost Schedule", July 2012.

[draft-randriamasy-alto-multi-cost-05]
"Multi-Cost ALTO", October 2011.

Authors’ Addresses

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

Email: Sabine.Randriamasy@alcatel-lucent.com

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

Nico Schwan
Thales Deutschland
Lorenzstrasse 10
Stuttgart 70435
Germany

Email: nico.schwan@thalesgroup.com

ALTO WG
Internet-Draft
Intended status: Standards Track
Expires: September 9, 2015

W. Roome
Alcatel-Lucent
X. Shi
Yale University
Y. Yang
Tongji/Yale University

March 8, 2015

ALTO Incremental Updates Using Server-Sent Events (SSE)
draft-roome-alto-incr-update-sse-02

Abstract

The Application-Layer Traffic Optimization (ALTO) [RFC7285] protocol provides network related information to client applications so that clients may make informed decisions. To that end, an ALTO Server provides Network and Cost Maps. Using those maps, an ALTO Client can determine the costs between endpoints.

However, the ALTO protocol does not define a mechanism to allow a client to obtain updates to those maps, other than by periodically re-fetching them. Because the maps may be large (potentially tens of megabytes), and because only parts of the maps may change frequently (especially Cost Maps), that can be extremely inefficient.

Therefore this document presents a mechanism to allow an ALTO Server to provide updates to ALTO Clients. Updates can be both immediate, in that the server can send updates as soon as they are available, and incremental, in that if only a small section of a map changes, the server can send just the changes.

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

Table of Contents

1. Introduction

The Application-Layer Traffic Optimization (ALTO) [RFC7285] protocol provides network related information to client applications so that clients may make informed decisions. To that end, an ALTO Server provides Network and Cost Maps, where a Network Map partitions the set of endpoints into a manageable number of Provider-Defined Identifiers (PIDs), and a Cost Map provides directed costs between PIDs. Given Network and Cost Maps, an ALTO Client can obtain costs between endpoints by using the Network Map to get the PID for each endpoint, and then using the Cost Map to get the costs between those PIDs.

However, the ALTO protocol does not define a mechanism to allow a client to obtain updates to those maps, other than by periodically re-fetching them. Because the maps may be large (potentially tens of megabytes), and because parts of the maps may change frequently (especially Cost Maps), that can be extremely inefficient.

Therefore this document presents a mechanism to allow an ALTO Server to provide updates to ALTO Clients. Updates can be both immediate, in that the server can send updates as soon as they are available, and incremental, in that if only a small section of a map changes, the server can send just the changes.

While primarily intended to provide updates to Network and Cost Maps, the mechanism defined in this document can provide updates to any ALTO resource, including POST-mode services such as Endpoint Property and Endpoint Cost Services, as well as new ALTO services to be defined by future extensions.

The rest of this document is organized as follows. Section 2 gives an overview of the incremental update approach, which is based on Server-Sent Events (SSEs). Section 3 defines the update events, and Section 4 defines the format of the incremental update messages. Section 5 defines the new Update Stream Service, Section 6 describes how a client should handle incoming updates, and Section 7 gives an example of the Information Resource Directory (IRD) for an ALTO

Server that offers a comprehensive set of Update Services. Section 8 discusses the design decisions behind this update mechanism. The remaining sections review the security and IANA considerations.

2. Overview of Approach

This section presents a non-normative overview of the update mechanism.

An ALTO Server can offer one or more Update Stream resources. An Update Stream is a POST-mode service that returns a continuous sequence of update messages for one or more ALTO resources. An Update Stream can provide updates to both GET-mode resources, such as Network and Cost Maps, and POST-mode resources, such as Endpoint Property Services.

Each update message updates one resource, and is sent as a Server-Sent Event (SSE), as defined by [SSE]. An update message is either a full replacement or else an incremental change. Full replacement updates use the JSON message formats defined by the ALTO protocol. Incremental updates use JSON Merge Patch ([RFC7386]) to describe the changes to the resource. The ALTO Server decides when to send update messages, and whether to send full replacements or incremental updates. These decisions can vary from resource to resource and from update to update.

An ALTO Server may offer any number of Update Stream resources, for a collection of the server’s resources. An ALTO Server’s Information Resource Directory (IRD) defines the Update Stream resources, and declares the set of resources for which each Update Stream provides updates. The server selects the resource set for each stream, although the set should be closed under the ALTO resource dependency relationship (i.e., the "uses" relationship). Thus the Update Stream for a Cost Map should also provide updates for the Network Map upon which that Cost Map depends.

When an ALTO Client requests an Update Stream resource, the client establishes a new persistent connection to the server. The connection remains open, and the server continues to send updates, until either the client or the server closes it. A client may request any number of Update Streams simultaneously. Because each stream consumes resources on the server, a server may limit the number of open Update Streams, may close inactive streams, may provide Update Streams via other processors, or may require client authorization/authentication.

3. Update Events

3.1. Overview of SSEs

The following is a non-normative summary of Server-Sent Events (SSEs). See [SSE] for the normative definition.

Server-Sent Events enable a server to send new data to a client by "server-push". The client establishes an HTTP ([RFC2616]) connection to the server, and keeps the connection open. The server continually sends messages. Messages are delimited by two new-lines (this is a slight simplification; see [SSE] for details). Messages may contain three fields: event, id, and data. All fields are strings as values. The data field may contain new-lines; the other two fields cannot. The event and id fields are optional.

Figure 1 is a sample SSE stream, starting with the client request. The server sends three events and then closes the stream. Note that the server may "chunk" the returned data (see [RFC2616]); for simplicity, we have omitted those details.

```
GET /stream HTTP/1.1
Host: example.com
Accept: text/event-stream


HTTP/1.1 200 OK
Connection: keep-alive
Content-Type: text/event-stream

event: start
id: 1
data: hello there

event: middle
id: 2
data: let’s chat some more ...
data: and more and more and ...

event: end
id: 3
data: good bye
```

Figure 1: A Sample SSE stream.

3.2. ALTO Update Events

In the events defined in this document, the data field is a JSON object. That object is either a complete specification of an ALTO resource, or else a JSON Merge Patch object describing changes to apply to an ALTO resource. We will refer to these as full-replacement and Merge Patch messages, respectively. The data objects in full-replacement messages are defined by [RFC7285]; examples are Network and Cost Map messages. The data objects in Merge Patch messages are defined by [RFC7386].

To indicate whether the data is a full-replacement or a Merge Patch object, in our update messages, the SSE "event" field has two sub-fields: the resource-id of an ALTO resource, and the media-type of the JSON message in the data field. The media-types for full-replacement messages are defined by [RFC7285], and include "application/alto-networkmap+json" for Network Map messages and "application/alto-costmap+json" for Cost Map messages. The media-type for a JSON Merge Patch message is "application/merge-patch+json", and is defined by [RFC7386]. An extension document may introduce other media-types to indicate new types of update messages.

Specifically, the two sub-fields of the event field are encoded as:

```
      resource-id , media-type
```

Note that commas (character code 0x2c) are allowed in ALTO resource-ids, but not in media-type names. Hence when parsing the SSE event field to obtain the two sub-fields, a client MUST split the string on the last comma.

This document does not use the SSE "id" field.

Figure 2 shows some examples of ALTO update events:

```
  event: my-network-map,application/alto-networkmap+json
  data: { ... full Network Map message ... }

  event: my-routingcost-map,application/alto-costmap+json
  data: { ... full Cost Map message ... }

  event: my-routingcost-map,application/merge-patch+json
  data: { ... Merge Patch update for the Cost Map ... }
```

Figure 2: Examples of ALTO update events.

3.3. Keep-Alive Messages

An SSE event with an empty "event" field is a keep-alive message. An ALTO Server MAY send keep-alive messages as needed. An ALTO Client MUST ignore any keep-alive messages.

4. Incremental Update Message Format

4.1. Overview of JSON Merge Patch

The following is a non-normative summary of JSON Merge Patch. See [RFC7386] for the normative definition.

JSON Merge Patch is intended to allow applications to update server resources via the HTTP PATCH method [RFC5789]. This document adopts the JSON Merge Patch message format to encode the changes, but uses a different transport mechanism.

Informally, a Merge Patch object is a JSON data structure that defines how to transform one JSON value into another. Merge Patch treats the two JSON values as trees of nested JSON Objects (dictionaries of name-value pairs), where the leaves are values other than JSON Objects (e.g., JSON Arrays, Strings, Numbers, etc.), and the path for each leaf is the sequence of keys leading to that leaf. When the second tree has a different value for a leaf at a path, or adds a new leaf, the Merge Patch tree has a leaf, at that path, with the new value. When a leaf in the first tree does not exist in the seond tree, the Merge Patch tree has a leaf with a JSON "null" value. The Merge Patch tree does not have an entry for any leaf that has the same value in both versions.

As a result, if all leaf values are simple scalars, JSON Merge Patch is a very efficient representation of the change. It is less efficient when leaf values are arrays, because JSON Merge Patch replaces arrays in their entirety, even if only one entry changes.

Formally, the process of applying a Merge Patch is defined by the following recursive algorithm, as specified in [RFC7386]:

```
define MergePatch(Target, Patch) {
  if Patch is an Object {
    if Target is not an Object {
      Target = {} # Ignore the contents and
                  # set it to an empty Object
    }
    for each Name/Value pair in Patch {
      if Value is null {
        if Name exists in Target {
          remove the Name/Value pair from Target
        }
      } else {
        Target[Name] = MergePatch(Target[Name], Value)
      }
    }
    return Target
  } else {
    return Patch
  }
}
```

Note that null as the value of a name/value pair will delete the element with "name" in the original JSON value.

4.2. JSON Merge Patch Applied to Network Map Messages

Section 11.2.1.6 of [RFC7285] defines the format of a Network Map message. Here is a simple example:

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

When applied to that message, the following Merge Patch update message adds the ipv6 prefix "2000::/3" to "PID1", deletes "PID2", and assigns a new "tag" to the Network Map:

```
     {
       "meta" : {
         "vtag" : {
           "tag" : "a10ce8b059740b0b2e3f8eb1d4785acd42231bfe"
         }
       },
       "network-map": {
         "PID1" : {
           "ipv6" : [ "2000::/3" ]
         },
         "PID2" : null
       }
     }
```

Here is the updated Network Map:

```
     {
       "meta" : {
         "vtag": {
           "resource-id" : "my-network-map",
           "tag" : "a10ce8b059740b0b2e3f8eb1d4785acd42231bfe"
         }
       },
       "network-map" : {
         "PID1" : {
           "ipv4" : [ "192.0.2.0/24", "198.51.100.0/25" ],
           "ipv6" : [ "2000::/3" ]
         },
         "PID3" : {
           "ipv4" : [ "0.0.0.0/0" ],
           "ipv6" : [ "::/0" ]
         }
       }
     }
```

4.3. JSON Merge Patch Applied to Cost Map Messages

Section 11.2.3.6 of [RFC7285] defines the format of a Cost Map message. Here is a simple example:

```
     {
       "meta" : {
         "dependent-vtags" : [
           {"resource-id": "my-network-map",
            "tag": "a10ce8b059740b0b2e3f8eb1d4785acd42231bfe"
           }
         ],
         "cost-type" : {
           "cost-mode"  : "numerical",
           "cost-metric": "routingcost"
         }
       },
       "cost-map" : {
         "PID1": { "PID1": 1,  "PID2": 5,  "PID3": 10 },
         "PID2": { "PID1": 5,  "PID2": 1,  "PID3": 15 },
         "PID3": { "PID1": 20, "PID2": 15  }
       }
     }
```

The following Merge Patch message updates the example cost map so that PID1->PID2 is 9 instead of 5, PID3->PID1 is no longer available, and PID3->PID3 is now defined as 1:

```
     {
       "cost-map" : {
         "PID1" : { "PID2" : 9 },
         "PID3" : { "PID1" : null, "PID3" : 1 }
       }
     }
```

Here is the updated cost map:

```
     {
       "meta" : {
         "dependent-vtags" : [
           {"resource-id": "my-network-map",
            "tag": "a10ce8b059740b0b2e3f8eb1d4785acd42231bfe"
           }
         ],
         "cost-type" : {
           "cost-mode"  : "numerical",
           "cost-metric": "routingcost"
         }
       },
       "cost-map" : {
         "PID1": { "PID1": 1,  "PID2": 9,  "PID3": 10 },
         "PID2": { "PID1": 5,  "PID2": 1,  "PID3": 15 },
         "PID3": {             "PID2": 15, "PID3": 1  }
       }
     }
```

5. Update Stream Service

An Update Stream Service returns a stream of SSE messages, as defined in Section 3.2.

5.1. Media Type

The media type of an ALTO Update Stream resource is "text/event-stream".

5.2. HTTP Method

An ALTO Update Stream resource is requested using the HTTP POST method.

5.3. Accept Input Parameters

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

```
     object-map {
        ResourceID -> ResourceUpdateReq;
     } UpdateStreamReq;

     object {
        [String      tag;]
        [Boolean     incremental-updates;]
        [Object      input;]
     } ResourceUpdateReq;
```

The keys are the resource-ids of the resources for which the client wants updates. Each resource-id MUST be one of those in the Update Streams's "uses" list (see Section 5.5). The ResourceUpdateReq values give additional parameters for the updates for each resource.

If any resource-id is invalid, or is not associated with this Update Stream, the server MUST return an E_INVALID_FIELD_VALUE error response (see Section 8.5.2 of [RFC7285]), and MUST close the stream without sending any update events.

If the client wants to receive updates for a resource, but does not need to set any of the sub-fields described below, the client MUST provide an entry for that resource-id whose value is an empty JSON Object (e.g., "{}").

If the "incremental-updates" field for a resource-id is "true", the server MAY send incremental update events for this resource-id (assuming the server supports incremental updates for that resource; see Section 5.4). If the "incremental-updates" field is "false", the ALTO Server MUST NOT send incremental update events for that resource. In this case, whenever a change occurs, the server MUST send a full-replacement update instead of an incremental update. The ALTO Server SHOULD send the full-replacement message soon after the change, although the server MAY wait until more changes are available. Thus an ALTO Client which declines to accept incremental updates may not get updates as quickly as a client which does.

The default for "incremental-updates" is "true", so to suppress incremental updates, the client MUST explicitly set "incremental-updates" to "false". Note that the client cannot suppress full-replacement update events.

If the resource-id is a GET-mode resource with a version tag (or "vtag"), as defined in Sections 6.3 and 10.3 of [RFC7285], and if the client has already retrieved that resource from the server, the client MAY set the "tag" field to "tag" part of the resource's version tag. If that version is still current, the ALTO Server MAY omit sending a full replacement update at the start of the stream (see Section 5.6.1).

If the resource-id is a POST-mode service that requires input, the client MUST set the "input" field to a JSON Object with the parameters that resource expects. If the "input" field is missing or invalid, the ALTO Server MUST return the same error response that that resource would return for missing or invalid input (see [RFC7285]). In this case, the server MUST close the Update Stream without sending any update events. If the inputs for several POST-mode resources are missing or invalid, the server MUST pick one error response and return it.

5.4. Capabilities

The capabilities are defined by an object of type UpdateStreamCapabilities:

```
  object {
    IncrementalUpdateMediaTypes incremental-update-media-types;
  } UpdateStreamCapabilities;

  object-map {
     ResourceID -> String;
  } IncrementalUpdateMediaTypes;
```

If this Update Stream can provide incremental update events for a resource, the "incremental-update-media-types" field has an entry for that resource-id, and the value is the media-type of the incremental update message. Normally this will be "application/merge-patch+json", because, as described in Section 3.2, JSON Merge Patch is the only incremental update event type supported by this document. However future extensions may allow other types of incremental updates.

5.5. Uses

The "uses" attribute MUST be an array with the resource-ids of every resource for which this stream can provide updates.

This set can include any subset of the resources proved by the ALTO Server, and may include resources defined in linked IRDs. However, it is RECOMMENDED that the ALTO Server select a set that is closed under the resource dependency relationship. That is, if an Update Stream’s "uses" set includes resource R1, and resource R1 depends on ("uses") resource R0, then the Update Stream’s "uses" set should include R0 as well as R1. For example, an Update Stream for a Cost

Map SHOULD also provide updates for the Network Map upon which that Cost Map depends.

5.6. Response

The response is a stream of SSE update events. Section 3.2 defines the events, and [SSE] defines how they are encoded into a stream.

There are additional requirements between events in the stream, as described below.

5.6.1. Event Sequence Requirements

- As soon as possible after the client initiates the connection, the ALTO Server MUST send a full-replacement update event for each resource-id requested by the client. The only exception is for a GET-mode resource with a version tag: the server MAY omit the initial full-replacement event for that resource if the "tag" field the client provided for that resource-id matches the tag of the server’s current version.

- If this stream provides updates for resource-ids R0 and R1, and if R1 depends on R0, then the ALTO Server MUST send the update for R0 before sending the related update for R1. For example, suppose a stream provides updates to a Network Map and its dependent Cost Maps. When the Network Map changes, the ALTO Server MUST send the Network Map update before sending the Cost Map updates.

- If this stream provides updates for resource-ids R0 and R1, and if R1 depends on R0, then the ALTO Server SHOULD send an update for R1 as soon as possible after sending the update for R0. For example, when a Network Map changes, the ALTO Server SHOULD send update events for the dependent Cost Maps as soon as possible after the update event for the Network Map.

5.6.2. Cross-Stream Consistency Requirements

If several distinct Update Stream resources offer updates for the same resource-id, the ALTO Server MUST send the same update data on all of those Update Streams. Similarly, the server MUST send the same updates to all clients connected to the that stream. However, the server MAY pack data items into different Merge Patch events, as long as the net result of applying those updates is the same.

For example, suppose two different clients open the same Cost Map Update Stream, and suppose the ALTO Server processes three separate cost point updates with a brief pause between each update. The server MUST send all three new cost points to both clients. But the server MAY send a single Merge Patch event (with all three cost points) to one client, while sending three separate Merge Patch events (with one cost point per event) to the other client.

5.7. Example: Simple Network and Cost Map Updates

Here is an example of a client's request and the server's immediate response, using the Update Stream resource "my-costs-update-stream" defined in the IRD in Section 7. The client requests updates for the Network Map and "routingcost" Cost Map, but not for the "hopcount" Cost Map. Because the client does not provide a "tag" for the Network Map, the server must send a full update for the Network Map as well as for the Cost Map. The client does not set "incremental-updates" to "false", so it defaults to "true". Thus server will send Merge Patch updates for the Cost Map, but not for the Network Map, because this Update Stream resource does not provide incremental updates for the Network Map.

Note that the server may "chunk" the returned data (see [RFC2616]); for simplicity, we have omitted those details.

```
  POST /updates/costmaps HTTP/1.1
  Host: alto.example.com
  Accept: text/event-stream,application/alto-error+json
  Content-Type: application/alto-updatestreamparams+json
  Content-Length: ###

  { "my-network-map": {},
    "my-routingcost-map": {}
  }


  HTTP/1.1 200 OK
  Connection: keep-alive
  Content-Type: text/event-stream

  event: my-network-map,application/alto-networkmap+json
  data: { ... full Network Map message ... }

  event: my-routingcost-map,application/alto-costmap+json
  data: { ... full routinccost Cost Map message ... }
```

After sending those two events immediately, the ALTO Server will send additional events as the maps change. For example, the following represents a small change to the Cost Map:

```
  event: my-routingcost-map,application/merge-patch+json
  data: {"cost-map": {"PID1" : {"PID2" : 9}}}
```

If a major change to the Network Map occurs, the ALTO Server MAY choose to send full Network and Cost Map messages rather than Merge Patch messages:

```
  event: my-network-map,application/alto-networkmap+json
  data: { ... full Network Map message ... }

  event: my-routingcost-map,application/alto-costmap+json
  data: { ... full Cost Map message ... }
```

5.8. Example: Advanced Network and Cost Map Updates

This example is similar to the previous one, except that the client requests updates for the "hopcount" as well as "routingcost" Cost Map, and provides the current version tag of the Network Map, so the server does not send the full Network Map update event at the beginning of the stream. After that, the ALTO Server sends updates for the Network Map and Cost Maps as they become available:

```
     POST /updates/costmaps HTTP/1.1
     Host: alto.example.com
     Accept: text/event-stream,application/alto-error+json
     Content-Type: application/alto-updatestreamparams+json
     Content-Length: ###

     { "my-network-map": {
           "tag": "a10ce8b059740b0b2e3f8eb1d4785acd42231bfe"
       },
       "my-routingcost-map": {}
       "my-hopcount-map": {}
     }


     HTTP/1.1 200 OK
     Connection: keep-alive
     Content-Type: text/event-stream

     event: my-routingcost-map,application/alto-costmap+json
     data: { ... full routingcost Cost Map message ... }

     event: my-hopcount-map,application/alto-costmap+json
     data: { ... full hopcount Cost Map message ... }

        (pause)

     event: my-routingcost-map,application/merge-patch+json
     data: {"cost-map": {"PID2" : {"PID3" : 31}}}

     event: my-hopcount-map,application/merge-patch+json
     data: {"cost-map": {"PID2" : {"PID3" : 4}}}
```

5.9. Example: Endpoint Property Updates

As another example, here is how a client can request updates for the property "priv:ietf-bandwidth" for a set of endpoints. The ALTO Server immediately sends a full-replacement message with the property values for all endpoints. After that, the server sends update events for the individual endpoints as their property values change.

```
     POST /updates/properties HTTP/1.1
     Host: alto.example.com
     Accept: text/event-stream
     Content-Type: application/alto-updatestreamparams+json
     Content-Length: ###

     { "my-properties": {
         "input": {
           "properties" : [ "priv:ietf-bandwidth" ],
           "endpoints" : [
              "ipv4:1.0.0.1",
              "ipv4:1.0.0.2",
              "ipv4:1.0.0.3"
           ]
         }
       }
     }

     { "update-events": [
         "my-properties,application/alto-endpointprops+json",
         "my-properties,application/merge-patch+json"
       ],
       "inputs": {
         "my-properties": {
           "properties" : [ "priv:ietf-bandwidth" ],
           "endpoints" : [
              "ipv4:1.0.0.1",
              "ipv4:1.0.0.2",
              "ipv4:1.0.0.3"
           ]
         }
       }
     }


     HTTP/1.1 200 OK
     Connection: keep-alive
     Content-Type: text/event-stream

     event: my-properties,application/alto-endpointprops+json
     data: { "endpoint-properties": {
     data:     "ipv4:1.0.0.1" : { "priv:ietf-bandwidth": "13" },
     data:     "ipv4:1.0.0.2" : { "priv:ietf-bandwidth": "42" },
     data:     "ipv4:1.0.0.3" : { "priv:ietf-bandwidth": "27" }
     data:  } }

        (pause)

     event: my-properties,application/merge-patch+json
     data: { "endpoint-properties":
     data:   {"ipv4:1.0.0.1" : {"priv:ietf-bandwidth": "3"}}
     data: }

        (pause)

     event: my-properties,application/merge-patch+json
```

```
     data: { "endpoint-properties":
     data:   {"ipv4:1.0.0.3" : {"priv:ietf-bandwidth": "38"}}
     data: }
```

6. Client Actions When Receiving Update Messages

In general, when a client receives a full-replacement update message for a resource, the client should replace the current version with the new version. When a client receives a Merge Patch update message for a resource, the client should apply those patches to the current version of the resource.

However, because resources can depend on other resources (e.g., Cost Maps depend on Network Maps), an ALTO Client MUST NOT use a dependent resource if the resource on which it depends has changed. There are at least two ways a client can do that. We will illustrate these techniques by referring to Network and Cost Map messages, although these techniques apply to any dependent resources.

One approach is for the ALTO Client to save the Network Map update message in a buffer, and continue to use the previous Network Map, and the associated Cost Maps, until the client receives the update messages for all dependent Cost Maps. The client then applies all Network and Cost Map updates atomically.

Alternatively, the client MAY update the Network Map immediately. In this case, the client MUST mark each dependent Cost Map as temporarily invalid, and MUST NOT use that map until the client receives a Cost Map update message with the new Network Map version tag. Note that the client MUST NOT delete the Cost Maps, because the server may send Merge Patch update messages.

The ALTO Server SHOULD send updates for dependent resources in a timely fashion. However, if the client does not receive the expected updates, the client MUST close the Update Stream connection, discard the dependent resources, and reestablish the Update Stream. The client MAY retain the version tag of the last version of any tagged resources, and give those version tags when requesting the new Update Stream. In this case, if a version is still current, the ALTO Server will not re-send that resource.

Although not as efficient as possible, this recovery method is simple and reliable.

7. IRD Example

Here is an example of an IRD that offers two Update Stream services. The first provides updates for the Network Map and "routingcost" and "hopcount" Cost Maps. The second provides updates to the Endpoint Properties service.

```
  "my-network-map": {
    "uri": "http://alto.example.com/networkmap",
    "media-type": "application/alto-networkmap+json",
  },
  "my-routingcost-map": {
    "uri": "http://alto.example.com/costmap",
    "media-type": "application/alto-costmap+json",
    "uses": ["my-networkmap+json"],
    "capabilities": {
      "cost-type-names": ["num-routingcost"]
    }
  },
  "my-hopcount-map": {
    "uri": "http://alto.example.com/costmap",
    "media-type": "application/alto-costmap+json",
    "uses": ["my-networkmap+json"],
    "capabilities": {
      "cost-type-names": ["num-hopcount"]
    }
  },
  "my-properties": {
    "uri": "http://alto.example.com/properties",
    "media-type": "application/alto-endpointprops+json",
    "accepts": "application/alto-endpointpropparams+json",
    "capabilities": {
      "prop-types": ["priv:ietf-bandwidth"]
    }
  },
  "my-costs-update-stream": {
    "uri": "http://alto.example.com/updates/costs",
    "media-type": "text/event-stream",
    "accepts": "application/alto-updatestreamparams+json",
    "uses": [
       "my-network-map",
       "my-routingcost-map",
       "my-hopcount-map"
    ],
    "capabilities": {
      "incremental-update-media-types": {
        "my-routingcost-map": application/merge-patch+json",
        "my-hopcount-map": "application/merge-patch+json"
```

```
         }
       }
     },
     "my-properties-update-stream": {
       "uri": "http://alto.example.com/updates/properties",
       "media-type": "text/event-stream",
       "uses": [ "my-properties" ],
       "accepts": "application/alto-updatestreamparams+json",
       "capabilities": {
         "incremental-update-media-types": {
           "my-properties": "application/merge-patch+json"
         }
       }
     }
```

8. Design Decisions and Discussions

8.1. HTTP2 Server-Push

An alternative would be to use HTTP 2 Server-Push [I-D-ietf-http2], instead of SSE over HTTP 1.1, as the transport mechanism for update messages. That would have several advantages: HTTP 2 Server-Push is designed to allow a server to send asynchronous messages to the client, and HTTP library packages should make it simple for servers to send those asynchronous messages, and for clients to receive them.

The disadvantage is HTTP 2 is a new protocol, and it is considerably more complicated than HTTP 1.1. While there is every reason to expect that HTTP library packages will eventually support HTTP 2, we do not want to delay deployment of an ALTO incremental update mechanism until that time.

Hence we have chosen to base ALTO updates on HTTP 1.1 and SSE. When HTTP 2 support becomes ubiquitous, a future extension of this document may define updates via HTTP 2 Server-Push.

8.2. Not Allowing Stream Restart

If an update stream is closed accidentally, when the client reconnects, the server must resend the full maps. This is clearly inefficient. To avoid that inefficiency, the SSE specification allows a server to assign an id to each event. When a client reconnects, the client can present the id of the last successfully received event, and the server restarts with the next event.

However, that mechanism adds additional complexity. The server must save SSE messages in a buffer, in case clients reconnect. But that mechanism will never be perfect: if the client waits too long to reconnect, or if the client sends an invalid id, then the server will have to resend the complete maps anyway.

Also, although this is a theoretical inefficiency, in practice it is unlikely to be a problem. Clients who want continuous updates for large resources, such as full Network and Cost Maps, are likely to be things like P2P trackers. These clients will be well connected to the network; they will rarely drop connections.

Mobile devices certainly can and do drop connections, and will have to reconnect. But mobile devices will not need continuous updates for multi-megabyte Cost Maps. If mobile devices need continuous updates at all, they will need them for small queries, such as the costs from a small set of media servers from which the device can stream the currently playing movie. If the mobile device drops the connection and reestablishes the Update Stream, the ALTO Server will have to retransmit only a small amount of redundant data.

In short, using event ids to avoid resending the full map adds a considerable amount of complexity to avoid a situation which is hopefully very rare. We believe that complexity is not worth the benefit.

The Update Stream service does allow the client to specify the tag of the last received version of any tagged resource, and if that is still current, the server need not retransmit the full resource. Hence clients can use this to avoid retransmitting full Network Maps. Cost Maps are not tagged, so this will not work for them. Of course, the ALTO protocol could be extended by adding version tags to Cost Maps, which would solve the retransmission-on-reconnect problem. However, adding tags to Cost Maps might add a new set of complications.

8.3. Is Incremental Update Useful for Network Maps?

It is not clear whether incremental updates (that is, Merge Patch updates) are useful for Network Maps. For minor changes, such as moving a prefix from one PID to another, they can be useful. But more involved changes to the Network Map are likely to be "flag days": they represent a completely new Network Map, rather than a simple, well-defined change.

At this point we do not have sufficient experience with ALTO deployments to know how frequently Network Maps will change, or how extensive those changes will be. For example, suppose a link goes down and the network uses an alternative route. This is a frequent occurrence. If an ALTO Server models that by moving prefixes from one PID to another, then Network Maps will change frequently.

However, an ALTO Server might model that as a change in costs between PIDs, rather than a change in the PID definitions. If a server takes that approach, simple routing changes will affect Cost Maps, but not Network Maps.

So while we allow a server to use Merge Patch on Network Maps, we do not require the server to do so. Each server may decide on its own whether to use Merge Patch for Network Maps.

This is not to say that Network Map updates are not useful. Clearly Network Maps will change, and update events are necessary to inform clients of the new map.

8.4. Other Incremental Update Message Types

Other JSON-based incremental update formats have been defined, in particular JSON Patch ([RFC6902]). The update events defined in this document have the media-type of the update data. JSON Patch has its own media type ("application/json-patch+json"), so this update mechanism could easily be extended to allow servers to use JSON Patch for incremental updates.

However, we think that JSON Merge Patch is clearly superior to JSON Patch for describing incremental updates to Cost Maps, Endpoint Costs, and Endpoint Properties. For these data structures, JSON Merge Patch is more space-efficient, as well as simpler to apply; we see no advantage to allowing a server to use JSON Patch for those resources.

The case is not as clear for incremental updates to Network Maps. For example, suppose a prefix moves from one PID to another. JSON Patch could encode that as a simple insertion and deletion, while Merge Patch would have to replace the entire array of prefixes for both PIDs. On the other hand, to process a JSON Patch update, the client would have to retain the indexes of the prefixes for each PID. Logically, the prefixes in a PID are an unordered set, not an array; aside from handling updates, a client has no need to retain the array indexes of the prefixes. Hence to take advantage of JSON Patch for Network Maps, clients would have to retain additional, otherwise unnecessary, data.

However, it is entirely possible that JSON Patch will be appropriate for describing incremental updates to new, as yet undefined ALTO resources. In this case, the extensions defining those new resources can use the update framework defined in this document, but recommend using JSON Patch, or some other method, to describe the incremental changes.

9. Security Considerations

Allowing persistent update stream connections enables a new class of Denial-of-Service attacks. An ALTO Server MAY choose to limit the number of active streams, and reject new requests when that threshold is reached. In this case the server should return the HTTP status "503 Service Unavailable".

Alternatively an ALTO Server MAY return the HTTP status "307 Temporary Redirect" to redirect the client to another ALTO Server which can better handle a large number of update streams.

This extension does not introduce any privacy issues not already present in the ALTO protocol.

10. IANA Considerations

This document defines a new media-type, "application/alto-updatestreamparams+json", as described in Section 5.3. All other media-types used in this document have already been registered, either for ALTO or JSON Merge Patch.

Type name: application

Subtype name: alto-updatestreamparams+json

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 9 of this document and Section 15 of [RFC7285].

Interoperability considerations: This document specifies format of conforming messages and the interpretation thereof.

Published specification: Section 5.3 of this document.

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).

11. References

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

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

[RFC5789] Dusseault, L. and J. Snell, "PATCH Method for HTTP", RFC 5789, March 2010.

[RFC6902] Bryan, P. and M. Nottingham, "JavaScript Object Notation (JSON) Patch", RFC 6902, April 2013.

[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.

[RFC7386] Hoffman, P. and J. Snell, "JSON Merge Patch", RFC 7386, October 2014.

[I-D-ietf-http2]
Belshe, M., Peon, R., and M. Thomson, "Hypertext Transfer Protocol version 2", draft-ietf-httpbis-http2-16 (work in progress), November 2014.

[SSE] Hickson, I., "Server-Sent Events (W3C)", December 2012.

Authors' Addresses

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

Xiao Shi
Yale University
51 Prospect Street
New Haven, CT 06511
USA

Email: xiao.shi@yale.edu

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

Email: yang.r.yang@gmail.com

ALTO WG
Internet-Draft
Intended status: Standards Track
Expires: July 27, 2015

W. Roome
Alcatel-Lucent
Y. Yang
Yale

January 23, 2015

PID Property Extension for ALTO Protocol
draft-roome-alto-pid-properties-03

Abstract

This document extends the Application-Layer Traffic Optimization (ALTO) Protocol [RFC7285] by defining PID-based properties in much the same way that the original ALTO Protocol defined endpoint-based properties.

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

A key abstraction introduced by the ALTO Protocol [RFC7285] is a PID (Provider-defined Identifier): a named set of endpoint addresses. For IPv4/IPv6 networks, this set is defined by one or more endpoint address prefixes, or CIDRs [RFC4632].

An ALTO Server defines one or more Network Maps. Each Network Map is defined as a set of uniquely named PIDs, with the constraint that any given CIDR may appear in only one PID. Thus a Network Map logically partitions the space of endpoint addresses into groups defined by the PIDs.

An ALTO Server may define several Network Maps, partitioning the endpoints in different ways. For example, one Network Map might partition endpoints according to geographical locations, with each PID representing the set of endpoints at a given location. Another Network Map might partition the endpoints according to their network capabilities (e.g., CDN delivery protocols such as HTTP or HTTPS). In this case, each PID defined in the Network Map represents endpoints with similar capabilities.

Another abstraction introduced by the ALTO protocol is an Endpoint Property, which is a named value associated with an endpoint. Properties can include data such as the endpoint’s ISP or ASN (Autonomous System Number), or the endpoint’s latitude and longitude, or the country or continent codes for the endpoint’s location. The ALTO protocol defines the mechanism for a client to query a server for these property values, but does not define the Endpoint Properties themselves; that is left to extensions.

Just as Endpoint Properties are useful, it is reasonable to expect that it will be useful for sets of endpoints -- that is, PIDs -- to have properties as well. A PID Property might be an Endpoint Property shared by all endpoints in that PID (e.g., their ISP or ASN). Alternatively, a PID Property could be the aggregation of individual properties of the endpoints in the PID (e.g., the latitude and longitude of their geographic bounding box). Or a PID Property might simply be inherent in the way in which the ALTO Server defined the PID.

However, the base ALTO protocol does not define the concept of PID Properties, or present a mechanism for a client to obtain PID Properties from an ALTO Server. This extension remedies that omission.

2. The Consistency and Inheritance Design Views

A key goal for defining PID Properties is that they should be consistent with, and generalize, Endpoint Properties. Specifically, in the base ALTO Protocol, the ALTO Server may associate a set of (name,value) pairs with any endpoint. These sets are the Endpoint Properties. The ALTO Protocol specifies the mechanism for a client to retrieve those Endpoint Properties from a server.

Consider the Endpoint Property PR and all endpoints defined in the PID P. If those endpoints all have the same value for PR, then the PID property PR for P should also have that same value.

For the more general case, assume that P consists of the set of endpoint addresses {EPi}, and EPi.PR is the value of property PR for endpoint EPi. Then we could obtain the value of PR for PID P by evaluating an aggregation function over the endpoint values {EPi.PR}. Typical aggregation functions include average/mean, mode, geo-center, union, bounding box, etc., although of course the actual aggregation function would depend on the semantics of the property.

Complementing the bottom-up aggregation view, we also adopt a top-down inheritance view. That is, if a property PR is not explicitly defined for an endpoint, but the endpoint’s PID does define a value for PR, then the endpoint inherits the value of PID property. The concept of inheritance is a simple, but powerful concept to reduce information redundancy.

3. PID Property Names

PID Property names MUST follow the same rules as Global Endpoint Property names (Section 10.8.2 of [RFC7285]), and non-private names MUST be registered with IANA.

Furthermore, PID Properties and Endpoint Properties share the same name space, and their definitions MUST be consistent. That is, if PR is used as both a PID Property and an Endpoint Property, then the semantics of PR as a PID Property must be clearly related to semantics of PR as an Endpoint Property. For example, when used as a PID Property, PR could be defined as the average (or some other representative value) of the values of PR for a set of endpoints.

This document will use the type PIDPropertyType to indicate a string denoting a PID Property Name. This type is identical to EndpointPropertyType as defined in Section 10.8 of [RFC7285].

4. A Hierarchical View of a Network Map

4.1. Nested And Overlapping PIDs

Each PID in a Network Map covers a set of endpoint addresses, and these sets can form a nested hierarchy. As an example, consider the following Network Map:

```
    PID p0:  [0.0.0.0/0, ::0/0]
    PID p1:  [1.0.0.0/8]
    PID p2a: [1.0.0.0/16]
    PID p2b: [1.1.0.0/16]
```

Every endpoint in p2a and p2b is also in the set covered by p1, and every possible IPV4 endpoint is also covered by PID p0. We would like to take advantage of this hierarchy to allow PIDs to inherit properties, just as a class in an object oriented programming language inherits attributes from its parent class. In the example, it is reasonable to expect that PIDs p2a and p2b would inherit any property of p1 that they do not override, and every PID would inherit the properties of PID p0.

Note that when calculating costs, the ALTO Protocol requires every endpoint to be in one, and only one, PID. The ALTO Protocol resolves this by using the "longest match" rule. That is, the ALTO Protocol says when an endpoint address is covered by more than one prefix, the address is in the PID with the prefix with the longest mask. Because any given prefix can be on only one PID, this ensures each endpoint is in only one PID. Thus in our example, 1.0.0.1 is in p2a, 1.1.0.1 is in p2b, and 1.2.0.1 in in p1.

However, while PIDs can nest within other PIDs, they do not necessarily form a simple hierarchy. Suppose we add PID p3 to the previous Network Map:

```
    PID p0:  [0.0.0.0/0, ::0/0]
    PID p1:  [1.0.0.0/8]
    PID p2a: [1.0.0.0/16]
    PID p2b: [1.1.0.0/16]
    PID p3:  [1.0.255.0/24, 1.1.0.0/24]
```

Some endpoints in p3 are in p2a, and others are in p2b. Thus p3 is partially contained in PIDs p2a and p2b, without being completely contained in either. So we would not expect p3 to inherit properties from p2a or p2b. But p3 is completely contained in p1 and p0, so it is reasonable to expect p3 to inherit properties from p1 and p0 that are not overridden in p2a or p2b. In short, "PID inheritance" raises issues similar to those of "multiple inheritance" in object oriented programming languages.

4.2. Definition Of PID Property Inheritence

While PID inheritance raises some complications, it is too useful to ignore. Fortunately CIDRs do form a simple "single parent" hierarchy, so we will use CIDR inheritance as the basis for defining PID inheritance. We believe this preserves the useful cases, and avoids the pathological ones.

Note that the ALTO Protocol did not define "PID inheritance" because it was not relevant to the base protocol.

Definition: A parent of CIDR C is any CIDR, in the set of CIDRs for all PIDs in the Network Map, which covers all endpoints covered by C, and whose mask is shorter than the mask of C.

Definition: The immediate parent of CIDR C is the parent of C with the longest prefix. The immediate parent CIDR might not exist, but if it does, it is unique.

Definition: A CIDR C inherits the value V for property PR if the PID containing its immediate parent CIDR defines the value V for property PR, or if its immediate parent CIDR inherits the value V for property PR.

Definition: A PID P has the value V for property PR if PR is explicitly defined with value V in P, or if every CIDR C in P inherits the same value V for PR.

Suppose the following properties are defined for PIDs described above:

```
    PID p1:  ISP="Verizon" country="us"
    PID p2a: ASN="12345" state="NJ"
    PID p2b: ASN="12345" state="CT"
```

Then p2a, p2b, and p3 all inherit the "ISP" and "country" properties from p1, because those properties are not defined for p2a or p2b. p3 would inherit "12345" for the "ASN" property, because although p2a and p2b both define that property, they give it the same value. However, p3 would not inherit the "state" property, because it has different values in p2a and p2b.

5. Protocol Specification: New Service Information Resources

This document defines two new ALTO resources. The PID Property Map returns the values of a set of properties for all PIDs in a Network Map. The Filtered PID Property Map returns PID Properties for a subset of PIDs and properties selected by the client. These are analogous to the Full and Filtered Network Maps in the base ALTO Protocol.

An ALTO Server may provide any number of resource of these types, provided that they have unique capabilities. For example, an ALTO Server may offer several Full PID Property Maps, for the same Network Map, as long as each one returns a different set of properties.

Both services are optional. In particular, an ALTO Server may provide a Full PID Property Map without providing a corresponding Filtered PID Property Map, and vice versa.

5.1. Full PID Property Map

A Full PID Property Map returns the properties defined for all PIDs in a Network Map.

5.1.1. Media Type

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

5.1.2. HTTP Method

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

5.1.3. Accept Input Parameters

None.

5.1.4. Capabilities

The capabilities are defined by an object of type PIDPropertyCapabilities:

```
object {
  PIDPropertyType prop-types<1..*>;
} PIDPropertyCapabilities;
```

where "prop-types" is an array with the names of the PID Properties returned by this resource.

5.1.5. Uses

An array with the resource-id of the Network Map which defines the PIDs whose properties are returned.

5.1.6. Response

The "dependent-vtags" field in the "meta" field of the response MUST be an array that includes the version tag of the ALTO Network Map defining these PIDs.

The data component of a PID Properties response is named "pid-properties", which is a JSON object of type PIDPropertyMapData, where:

```
object {
  PIDPropertyMapData pid-properties;
} InfoResourcePIDProperties : ResponseEntityBase;

object-map {
  PIDName -> PIDProps;
} PIDPropertyMapData;

object {
  PIDPropertyType -> JSONValue;
} PIDProps
```

The PIDName and ResponseEntityBase types are defined in Sections 10.1 and 8.4 of the ALTO Protocol [RFC7285].

Specifically, a PIDPropertyMapData object has one member for each PID in the Network Map. The PID’s properties are encoded in the corresponding PIDProps object. PIDProps encodes one name/value pair for each property, where the property names are encoded as strings of type PIDPropertyType. A protocol implementation SHOULD assume that the property value is a JSONString and fail to parse if it is not, unless the implementation is using an extension to this document that indicates when and how property values of other data types are signaled.

For each PID in the Network Map, the ALTO Server returns the value defined for each of the PID Properties specified in this resource’s "capabilities" list. For efficiency, the ALTO Server SHOULD omit property values that are inherited rather than explicitly defined. The client SHOULD use the algorithm defined in Section 4.2 to determine inherited property values.

An ALTO Server MAY explicitly define a property as not having a value for a particular PID. That is, a server may say that a property is "defined to have no value", as opposed to the property being "undefined". In this case, if the PID would not inherit a value for that property, then the ALTO Server MUST either omit the property, or else return a "null" value for it. However, if that PID would inherit a value for that property, then the ALTO server MUST return a "null" value. An ALTO Client MUST recognize a "null" value as meaning "do not apply the inheritance rules for this property in the PID."

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

5.1.7. Example

The following example uses the Network Map and PID Properties defined above. Note that the response does not include entries for PIDs p0 or p3. The former was omitted because no properties were defined for it, and the latter because its only properties are inherited.

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


HTTP/1.1 200 OK
Content-Length: ###
Content-Type: application/alto-pidprop+json
{
  "meta" : {
    "dependent-vtags" : [
      {"resource-id": "my-default-network-map",
       "tag": "7915dc0290c2705481c491a2b4ffbec482b3cf62"
      }
    ]
  },
  "pid-properties": {
    "p1" : { "ISP": "Verizon", "country": "us" },
    "p2a": { "ASN": "12345", "state": "NJ" },
    "p2b": { "ASN": "12345", "state": "CT" }
  }
}
```

5.2. Filtered PID Property Map

A Filtered PID Property Map returns the values for set of PIDs and properties selected by the client.

5.2.1. Media Type

The media type of an ALTO Filtered PID Property Map resource is the same as a Full PID Property Map, and is defined in Section 5.1.1.

5.2.2. HTTP Method

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

5.2.3. Accept Input Parameters

The input parameters for a Filtered PID 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-pidpropparams+json", which is a JSON object of type ReqPIDProp:

```
  object {
    PIDPropertyType properties<1..*>;
    PIDName         pids<1..*>
  } ReqPIDProp;
```

with fields:

properties: List of PID properties to be returned for each PID. Each specified property MUST be included in the list of supported properties indicated by this resource’s "capabilities" field (Section 5.2.4). The ALTO server MUST interpret entries appearing multiple times as if they appeared only once.

pids: List of PID names for which the specified properties are to be returned. The ALTO server MUST interpret entries appearing multiple times as if they appeared only once.

5.2.4. Capabilities

The capabilities are defined by an object of type PIDPropertyCapabilities, as defined above.

5.2.5. Uses

An array with the resource-id of the Network Map which defines the PIDs whose properties are returned.

5.2.6. Response

The response is the same as for the Full PID Property Map (Section 5.1.6), except that it only includes the properties and PIDs requested by the client.

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

5.2.7. Example

The following example uses the Network Map and PID Properties defined above. Note that the response includes the inherited property "ISP" for PID p2a, and "state" and "ISP" for p3.

```
     POST /pidprop/lookup HTTP/1.1
     Host: alto.example.com
     Content-Length: ###
     Content-Type: application/alto-pidpropparams+json
     Accept: application/alto-pidprop+json,application/alto-error+json

     {
       "properties" : [ "ISP", "ASN", "state" ]",
       "pids" : [ "p1", "p2a", "p3" ]
     }


     HTTP/1.1 200 OK
     Content-Length: ###
     Content-Type: application/alto-pidprop+json
     {
       "meta" : {
         "dependent-vtags" : [
           {"resource-id": "my-default-network-map",
            "tag": "7915dc0290c2705481c491a2b4ffbec482b3cf62"
           }
         ]
       },
       "pid-properties" : {
         "p1" : { "ISP": "Verizon" },
         "p2a": { "ISP": "Verizon", "ASN": "12345", "state": "NJ" },
         "p3" : { "ISP": "Verizon", "ASN": "12345" }
       }
     }
```

6. IRD Example

Here is an example of an ALTO Information Resource Directory (IRD) which defines several PID Property resources. Note that "full-pid-property-map" returns several PID Properties for all PIDs in the Network Map, while "asn-pid-property-map" returns just the ASN property. The resource "filtered-pid-properties" resource returns values for properties and PIDs selected by the client.

```
     ...
     "resources" : {
        "my-default-network-map" : {
           "uri" : "http://alto.example.com/networkmap",
           "media-type" : "application/alto-networkmap+json"
        },
        "endpoint-property" : {
           "uri" : "http://alto.example.com/endpointprop/lookup",
           "media-type" : "application/alto-endpointprop+json",
           "accepts" : "application/alto-endpointpropparams+json",
           "capabilities" : {
             "prop-types" : [ "my-default-network-map.pid",
                              "priv:ietf-example-prop" ]
           },
        },
        "full-pid-property-map" : {
           "uri" : "http://alto.example.com/pidprop/full",
           "media-type" : "application/alto-pidprop+json",
           "uses" : ["my-default-network-map" ]
           "capabilities" : {
             "prop-types" : [ "ISP", "country", "ASN", "state" ]
           },
        }
        "asn-pid-property-map" : {
           "uri" : "http://alto.example.com/pidprop/asn",
           "media-type" : "application/alto-pidprop+json",
           "uses" : ["my-default-network-map" ]
           "capabilities" : {
             "prop-types" : [ "ASN" ]
           },
        }
        "filtered-pid-properties" : {
           "uri" : "http://alto.example.com/pidprop/lookup",
           "media-type" : "application/alto-pidprop+json",
           "accepts" : "application/alto-pidpropparams+json",
           "uses" : ["my-default-network-map" ]
           "capabilities" : {
             "prop-types" : [ "ISP", "country", "ASN", "state" ]
           },
        }
     }
```

7. Extensions To Endpoint Property Service

As described in Section 10.8 of [RFC7285], Endpoint Property names may be prefixed with the Resource ID of a Network Map. For such resource-specific properties, if a value is not explicitly defined for an endpoint, the Endpoint Property Service MUST return the value that a PID Property Map would return for the PID containing that endpoint.

For property names that are not prefixed by a Network Map Resource ID, if a value is not defined for an endpoint, the Endpoint Property Service MAY return the value defined for that property in one of the ALTO Server’s PID Property Maps for the PID containing the endpoint.

8. Security Considerations

As with Endpoint Properties, PID 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 PID Property services. 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.

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-pidprop+json	Section 5.1.1
application	alto-pidpropparams+json	Section 5.2.3

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 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 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 PID Properties as well as Endpoint Properties.

10. References

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

[RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", RFC 4632, BCP 122, August 2006.

[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.

Authors' Addresses

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

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

Email: yry@cs.yale.edu

ALTO Working Group
Internet-Draft
Intended status: Informational
Expires: September 10, 2015

X. Shi
Yale University
Y. Yang
Tongji/Yale University
M. Scharf
Alcatel-Lucent Bell Labs

March 9, 2015

A YANG Data Model for Base ALTO Data
draft-shi-alto-yang-model-02

Abstract

This document defines a YANG model for the base ALTO information resources defined in [RFC7285]. In particular, the introduction of this model allows a standard approach to provision and update ALTO state stored at an ALTO server.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document defines a YANG model for the base ALTO information resources defined in [RFC7285]. Such a model can provide value in multiple aspects. For example, the tree diagram of the YANG model may provide a concise visualization of the information contained in ALTO base information resources.

A particular goal of our designing the model is to provide a standard way for a management entity or even an ALTO client to provision or update ALTO information resources stored at an ALTO server. We refer to an entity that interacts with the ALTO server through the YANG model as an ALTO/YANG client. ALTO clients and ALTO/YANG clients are different in that the former uses [RFC7285] to obtain ALTO information resources. An ALTO/YANG client, on the other hand, uses NETCONF [RFC6241] or RESTCONF [RESTCONF], which will be based on the

YANG model, to retrieve, and more importantly, update the ALTO information resources at the ALTO server. Figure 1 is a non-normative diagram illustrating the related entities. How an ALTO/YANG server stores and pushes updates to the ALTO server belongs to implementation.

```
   +--------------------------------------------------------------------+
   |                                                                    |
   |                          +-----------------+                       |
   |  +-----------+  NETCONF/ |  +-----------+  |                       |
   |  | ALTO/YANG |  RESTCONF |  | ALTO/YANG |  |                       |
   |  | Client    | ============ | Server    |  |                       |
   |  +-----------+           |  +-----------+  |                       |
   |                          |        |        |                       |
   |                          |        |        |                       |
   |                          |  +-----------+  |          +--------+   |
   |                          |  | ALTO      |  | RFC7285  | ALTO   |   |
   |                          |  | Server    | =========== | Client |   |
   |                          |  +-----------+  |          +--------+   |
   |                          +-----------------+                       |
   +--------------------------------------------------------------------+
```

Figure 1: ALTO/YANG Components.

The ALTO YANG model defined in this document contains only data instances. There are no definitions for RPCs or notifications. Specifically, the model defines data instances for ALTO information resource directory (IRD), network maps, cost maps, and the endpoint property map.

As a result of the preceding data-instances only design, the model will not directly provide other ALTO information services such as filtered maps and endpoint cost services. This is consistent with the design goal: using the YANG model as a basis for ALTO base information provisioning. For example, an ALTO/YANG client may use the NETCONF filtering features (See Section 6 and Section 8.9 of [RFC6241]) to approximate the ALTO filtering services. We provide examples and templates in the appendix.

The rest of the document is organized as follows. In Section 2, we give the details of the ALTO YANG model. In Section 3, we give a non-normative specification on how one may use NETCONF and RESTCONF to update ALTO YANG data. Section 4 and Section 5 give IANA and security considerations respectively.

2. ALTO YANG Model

Figure 2 shows the tree diagram of the structure of the ALTO/YANG data model. See Appendix A for the complete YANG model.

```
module: alto-service
   +--rw resources
      +--rw IRD
      |  +--rw meta
      |  |  +--rw cost-types* [cost-type-name]
      |  |  |  +--rw cost-type-name    cost-type-name
      |  |  |  +--rw cost-mode         cost-mode
      |  |  |  +--rw cost-metric       cost-metric
      |  |  |  +--rw description?      string
      |  |  +--rw default-alto-network-map    resource-id
      |  +--rw resources* [resource-id]
      |     +--rw resource-id     resource-id
      |     +--rw uri             inet:uri
      |     +--rw media-type      media-type
      |     +--rw accepts*        media-type
      |     +--rw capabilities
      |     |  +--rw cost-constraints?   boolean
      |     |  +--rw cost-type-names*    cost-type-name
      |     |  +--rw prop-types*         endpoint-property-type
      |     +--rw uses*           resource-id
      +--rw network-maps
      |  +--rw network-map* [resource-id]
      |     +--rw resource-id    alto:resource-id
      |     +--rw tag            alto:tag-string
      |     +--rw map* [pid]
      |        +--rw pid                          alto:pid-name
      |        +--rw endpoint-address-group* [address-type]
      |           +--rw address-type       endpoint-address-type
      |           +--rw endpoint-prefix*   endpoint-prefix
      +--rw cost-maps
      |  +--rw cost-map* [resource-id]
      |     +--rw resource-id    alto:resource-id
      |     +--rw tag            alto:tag-string
      |     +--rw meta
      |     |  +--rw dependent-vtags*
      |     |  |  +--rw resource-id    resource-id
      |     |  |  +--rw tag            tag-string
      |     |  +--rw cost-type
      |     |     +--rw cost-mode      cost-mode
      |     |     +--rw cost-metric    cost-metric
      |     |     +--rw description?   string
      |     +--rw map* [src]
      |        +--rw src          alto:pid-name
```

```
         |        +--rw dst-costs* [dst]
         |           +--rw dst     alto:pid-name
         |           +--rw cost
         +--rw endpoint-property-map
            +--rw meta
            |  +--rw dependent-vtags*
            |     +--rw resource-id    resource-id
            |     +--rw tag            tag-string
            +--rw endpoint-properties* [endpoint]
               +--rw endpoint      typed-endpoint-address
               +--rw properties* [property-type]
                  +--rw property-type    endpoint-property-type
                  +--rw property         endpoint-property-value
```

Figure 2: Tree Diagram of ALTO YANG Model.

This model has certain limitations concerning multiple versions of resources. In particular, network maps and cost maps are modeled as lists keyed by their resource ids only. Therefore, the datastore may store only the most updated copy of the resources with their appropriate tags. The advantages of this design include: 1) the server does not need to keep all version history of all resources; 2) the clients need not know the tag in order to request the most recent resource. The disadvantages include: 1) the server must keep all updated resources consistent as they are updated; 2) there is no way for a client to request an old version of a resource.

3. Examples: Update ALTO Information Resources using YANG Model

The Network Configuration (NETCONF) Protocol [RFC6241] provides mechanisms to manipulate configuration data and state data of network devices via some of its standard operations (e.g, <get-config>). The NETCONF protocol messages are encoded in YANG-modeled XML and transported via persistent connections such as SSH. Given the datastore, NETCONF will be able to handle simple retrieval and update (insertion, deletion, and replacement) of configuration or state data. Our focus is on the updates of configuration or state data.

The RESTCONF Protocol ([RESTCONF]) uses restful HTTP operations to provide CRUD operations on a NETCONF datastore containing YANG-defined data. RESTCONF is similar to NETCONF in terms of usage, hence we elaborate on NETCONF update operations in this section and provide more details in Appendix C.

3.1. Update Operations

The most relevant update operation that NETCONF provides is the <edit-config> operation, which allows a client to edit the configuration data in different ways (e.g., merge, replace, etc.). Section 7.2 of [RFC6241] contains the specification of <edit-config>. This section gives two non-normative examples on modifying network and cost maps to illustrate the usage.

3.1.1. <edit-config> Examples

3.1.1.1. Modifying Network Maps

Suppose we would like to modify a network map in the following manner: remove endpoint prefix "192.0.2.0/24" from PID1 and add it to PID 2. The <edit-config> operation would be the following:

```
   <rpc message-id=SEQ-NUM
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <edit-config>
       <target>
         <running/>
       </target>
       <config
         xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
         <resources
           xmlns="urn:ietf:params:xml:ns:yang:alto-service">
           <network-maps>
             <network-map>
               <resource-id>myNetMap1</resource-id>
               <tag>ANEWTAG</tag> <TODO NEED VALID>
               <map>
                 <pid>PID1</pid>
                 <endpoint-address-group>
                   <endpoint-prefix xc:operation="delete">
                     192.0.2.0/24
                   </endpoint-prefix>
                 </endpoint-address-group>
               </map>
               <map>
                 <pid>PID2</pid>
                 <endpoint-address-group>
                   <endpoint-prefix xc:operation="create">
                     192.0.2.0/24
                   </endpoint-prefix>
                 </endpoint-address-group>
               </map>
             </network-map>
           </network-maps>
         </resources>
       </config>
     </edit-config>
   </rpc>
```

Note that we specified different "suboperations" for the two endpoint prefixes, one to "delete" and the other to "create". The reply from the NETCONF server would be either <rpc-reply> with <OK> or an <rpc-error>.

3.1.1.2. Modifying Cost Maps

To change the cost from PID3 to PID2 from 15 to 10 in myCostMap1, the NETCONF RPC would be the following:

```
<rpc message-id=SEQ-NUM
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config
      xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
      <resources
        xmlns="urn:ietf:params:xml:ns:yang:alto-service">
        <cost-maps>
          <cost-map>
            <resource-id>myCostMap1</resource-id>
            <tag>ANEWTAG</tag> <TODO NEED VALID>
            <map>
              <src>PID3</src>
              <dst-costs>
                <dst>PID2</dst>
                <cost>10</cost>
              </dst-costs>
            </map>
          </cost-map>
        </cost-maps>
      </resources>
    </config>
  </edit-config>
</rpc>
```

3.1.2. Consistency Considerations

The ALTO/YANG server MUST validate update operations to maintain the consistency of the ALTO resources. Specifically, this document specifies the following consistency requirements:

o ALTO/YANG server MUST check that each update operation MUST include a new tag to indicate the update.

o TODO: Add more

3.2. Other Operations

3.2.1. <delete-config>

3.2.1.1. Examples

4. Security Considerations

This document depends on standard NETCONF/RESTCONF security mechanisms to achieve authentication and authorization of update operations.

5. IANA Considerations

This document does not have IANA considerations.

6. References

[RESTCONF]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-04 (work in progress), January 2015.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, June 2010.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6020, October 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.

[draft-ietf-netmod-yang-json]
Lhotka, L., "JSON Encoding of Data Modeled with YANG", draft-ietf-netmod-yang-json-01 (work in progress), October 2014.

[draft-shi-alto-yang-json]
Shi, X. and Y. Yang, "Modeling JSON Messages Using YANG", draft-shi-alto-yang-json-00 (work in progress), October 2014.

Appendix A. YANG Data Model for ALTO Protocol

A.1. ALTO/YANG: Common Data Types

```
module alto-service-types {
  yang-version 1;

  namespace "urn:ietf:params:xml:ns:yang:alto-service-types";
  // TODO: replace with IANA namespace when assigned

  prefix "alto";

  import ietf-inet-types {
    prefix inet;
  }

  organization "ALTO WG";
  contact "alto@ietf.org";

  description
    "This module defines the data types and groupings for a semantically
     equivalent data model for the ALTO services defined in RFC7285.";

  revision 2014-11-01 {
    description "Separate types module";
  }

  revision 2014-10-24 {
    description "Initial version.";
  }

  /********************
   * TYPE DEFINITIONS *
   ********************/

  /************************************************************
    Definitions for addresses

    ALTO RFC7285 uses the following addresses, as shown in the
    examples below:

      - Endpoint property service (Sec. 11.4.1.7):
        "endpoints"  : [ "ipv4:192.0.2.34",
                         "ipv4:203.0.113.129" ]
      - Endpoint cost service (Sec. 11.5.1.7):
        "endpoints" : {
        "srcs": [ "ipv4:192.0.2.2" ],
       "dsts": [
         "ipv4:192.0.2.89",
         "ipv4:198.51.100.34",
         "ipv4:203.0.113.45"
     - Network map (Sec. 11.2.1.7.):
         "ipv4": [
           "192.0.2.0/24",
           "198.51.100.0/25"
         ],
         "ipv6": [
           "2001:db8:0:1::/64",
           "2001:db8:0:2::/64"
         ]

    To handle the proceeding, we need the following definitions:
       ipv4-address (e.g., 192.0.2.0, already defined in rfc6991),
       ipv6-address (already defined in rfc6991),
       ipv4-prefix (e.g., 192.0.2.0/24, already defined in rfc6991),
       ipv6-prefix (defined in rfc6991),
       typed-ipv4-address (e.g., ipv4:192.0.2.1, to be defined below)
       typed-ipv6-address
       typed-ipv4-prefix-list (e.g., "ipv4": [
            "192.0.2.0/24",
            "198.51.100.0/25"
         ],

   *******************************************************************/

   /*
      First define typed-ipv4-address and typed-ipv6-address, as used
      by endpoint services.

      The ideal case is to define it as "ipv4:"+ipv4-address, but there
      is not such a type constructor (YANG EXTENSION).  Hence, the
      current definition cuts-and-pastes (i.e., repeats verbatim) the
      definition of ipv4-address and prepend "ipv4:". The downside is
      that if someone redefines ipv4-address, there could be
      inconsistency.
    */

   typedef typed-ipv4-address {
     type string {
         pattern
           'ipv4:(([0-9]|[1-9][0-9]|1[0-9][0-9]|'
         + '2[0-4][0-9]|25[0-5])\.){3}'
         + '([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])'
         + '(%[\p{N}\p{L}]+)?';
      }
```

```
  }


  typedef typed-ipv6-address {
    type string {
      pattern 'ipv6:((:|[0-9a-fA-F]{0,4}):)([0-9a-fA-F]{0,4}:){0,5}'
            + '((([0-9a-fA-F]{0,4}:)?(:|[0-9a-fA-F]{0,4}))|'
            + '(((25[0-5]|2[0-4][0-9]|[01]?[0-9]?[0-9])\.){3}'
            + '(25[0-5]|2[0-4][0-9]|[01]?[0-9]?[0-9])))'
            + '(%[\p{N}\p{L}]+)?';
      pattern 'ipv6:(([^:]+:){6}(([^:]+:[^:]+)|(.*\..*)))|'
            + '((([^:]+:)*[^:]+)?::(([^:]+:)*[^:]+)?)'
            + '(%.+)?';
    }
  }

  typedef typed-endpoint-address {
    type union {
      type typed-ipv4-address;
      type typed-ipv6-address;
      // EXTENSION: ADD NEW TYPE HERE.
    }
    description
      "Ref: RFC7285 Sec. 10.4.1 Typed Endpoint Addresses" +
      "= AddressType:EndpointAddr";
  }

  /* Next, we define endpoint address group, as used in the definition
     of ALTO network maps. Specifically, an endpoint address group in
     ALTO is defined as a key-value store, with address type as key,
     and an array of prefix as the value of each key:

     EndpointAddrGroup. RFC7285 Sec. 10.4.5." +
       object-map {
       AddressType -> endpoint-prefix<0..*>;
     } EndpointAddrGroup;

     There are two challenges:

     1) To specify that AddressType is key, we must use the list type,
     which is the only type that one can specify key. However, the
     current JSON-YANG encoding generates an array, instead of a
     key-value map;

     2) Ideally, we want to enforce address type and prefix
     consistency; for example, an ipv6 prefix in an ipv4 type should
     not be allowed. However, we encounter problems. We leave this as
     an OPEN ISSUE.
  */

  typedef endpoint-address-type {
    type union {
      type enumeration {
        enum ipv4;
        enum ipv6;
        // EXTENSION: ADD NEW TYPE HERE
      }
    }
    description
      "Ref: RFC7285 Sec 2.2.";
  }

  typedef endpoint-prefix {
    type inet:ip-prefix;
    description
      "endpoint prefix, identical to ip-prefix defined in RFC6991.";
  }

  grouping endpoint-address-group {
    list endpoint-address-group {
      key address-type;
      leaf address-type {
        type endpoint-address-type;
        mandatory true;
      }
      leaf-list endpoint-prefix {
        type endpoint-prefix;
      }
    }
    description
      "EndpointAddrGroup. RFC7285 Sec. 10.4.5." +
      " object-map {
          AddressType -> endpoint-prefix<0..*>;
        } EndpointAddrGroup;";
  }

  /***************************************************************
   * Definitions for IDs and names
   *
   * ALTO defines the following concepts that are names and IDs:
   *
   *   pid name (used in network map, cost map),
   *   resource IDs (used to identify alto network/cost maps),
   *   version tag (used to indicate uniqueness of resource),
   *   cost-type-name (used in IRD),
   *   cost-metric,
   *   cost-mode
   *
   * We group their definitions together below.
   **************************************************************/

  typedef valid-id-string {
    type string {
      length "1..64";
      pattern "[0-9a-zA-Z_\-:@\.]+";
    }
    description
      "Type for valid ID strings.";
  }

  typedef tag-string {
    type string {
      length "1..64";
      pattern "[!-~]+";
    }
    description
      "Tag. RFC7285 Sec. 10.3. U+0021-U+007E";
  }

  typedef pid-name {
    type valid-id-string;
    description
      "Name for the PID." +
      "RFC7285, Section 10.1. Note: the '.' separator MUST NOT be" +
      "used unless specifically indicated in RFC7285 or an" +
      " extension document.";
  }

  typedef resource-id {
    type valid-id-string;
    description
      "Resource-ID.";
  }

  grouping vtag {
    leaf resource-id {
      type resource-id;
      mandatory true;
    }
    leaf tag {
      type tag-string;
      mandatory true;
    }
    description
```

```
      "Version tag. Both resource-id and tag must be equal
       byte-for-byte. RFC7285 Sec. 10.3." +
      " object {
          ResourceID resource-id;
          JSONString tag;
        } VersionTag;";
  }

  grouping dependent-vtags {
    list dependent-vtags {
      key "resource-id tag";
      uses vtag;
      min-elements 1;
    }
  }

 /************************************
    Definitions for cost type and cost types

    In ALTO, a cost type consists of two required components:

      cost-metric,
      cost-mode
      and an optional description component.

    In the IRD, one can name each cost type. Such info is collected
    in a hash map called cost types.
  *************************************/

  typedef cost-metric {
    type union {
      type enumeration {
        enum routingcost {
          description
          "Default metric. MUST support. RFC7285 Sec. 6.1.1.1.";
        }
        enum hopcount {
          description
          "Hopcount metric.";
        }
        // EXTENSION: Additional cost-metric will be defined here.
      }
      type string {
        length 1..32;
        pattern "priv:[0-9a-zA-Z_\-:\.]+";
      }
    }
    description
```

```
      "Cost metric. for type string,
      ’priv:’ reserved for Private Use.";
  }

  typedef cost-mode {
    type enumeration {
      enum numerical {
        description
          "Numerical cost mode.";
      }
      enum ordinal {
        description
          "Ordinal cost mode.";
      }
      // EXTENSION: Additional cost-mode will be defined here.
    }
    description
      "Cost mode. MUST support at least one of numerical and ordinal";
  }

  grouping cost-type {
    leaf cost-mode {
      type cost-mode;
      mandatory true;
      description
        "Cost mode.";
    }
    leaf cost-metric {
      type cost-metric;
      mandatory true;
      description
        "Cost metric.";
    }
    leaf description {
      type string;
      description
        "Optional description field.";
    }
    description
      "Cost type. RFC7285 Sec. 10.7." +
      " object {
          CostMetric cost-metric;
          CostMode   cost-mode;
          [JSONString description;]
        } CostType;";
  }

  typedef cost-type-name {
    type valid-id-string;
    // NOTE: not fully specified in RFC7285, default as valid id
  }

  grouping cost-types {
    list cost-types {
      key cost-type-name;
      leaf cost-type-name {
        type cost-type-name;
      }
      uses cost-type;
    }
    description
      "RFC 7285 Sec. 9.2.2." +
      "object-map {
         JSONString -> CostType;
       } IRDMetaCostTypes;";
  }


  /**************************************
  * Definitions for endpoint properties *
  **************************************/
  typedef global-endpoint-property {
    type union {
      type enumeration {
        enum pid {
          description "PID property.";
        }
        // EXTENSION: other options here
      }
      type string {
        pattern "priv:[\w\-:@]+";
      }
    }
    description
      "Global endpoint property. RFC7285 Sec. 10.8.2." +
      "'priv:' for Private Use " +
      " length 1..32; '.' is not allowed";
  }

  /*
   * Ideally we would want to extend the typedef of resource-id and
   * global endpoint properties, however, YANG 1.0 does not allow
   * that, hence we simply copied the regex for resource-id over
   * verbatim.
   */
```

```
  typedef resource-specific-endpoint-property {
    type string {
      length "3..97"; //len(resource-id) + 1 + len(global-property)
      pattern "(priv:)?[\w\-:@\.]+\.[\w\-:_]+"; // resource-id.property
    }
    description
      "Resource-specific endpoint property.";
  }

  typedef endpoint-property-type {
    type union {
      type resource-specific-endpoint-property;
      type global-endpoint-property;
    }
    description
      "Endpoint property type. RFC7285 Sec. 10.8.";
  }

  typedef endpoint-property-value {
    type string;
    description
      "Endpoint property (value).";
  }

 /************************************
  * Definitions for response header
  ************************************/

  typedef media-type {
    type union {
      type string {
        pattern "application/alto\-.*";
      }
      type enumeration {
        enum alto-directory+json;
        enum alto-networkmap+json;
        enum alto-networkmapfilter+json;
        enum alto-costmap+json;
        enum alto-costmapfilter+json;
        enum alto-endpointprop+json;
        enum alto-endpointpropparams+json;
        enum alto-endpointcost+json;
        enum alto-endpointcostparams+json;
        enum alto-error+json;
      }
    }
  }
  grouping alto-cost {
    anyxml cost {
      mandatory true;
      description
        "ALTO cost is a JSONValue, which could be
        an object, array, string, etc. (Ref: RFC 7159 Sec.3.)";
    }
  }

  typedef constraint {
    type string {
      pattern "(gt|ge|lt|le|eq) [0-9]+";
    }
    description
      "RFC7285 Sec. 11.3.2.3. The second part must be in the" +
      "same unit as cost-metric, IEEE 754 2008 floating point.";
  }

  /*******************************************
    Groupings for ALTO information resource
  *******************************************/

  /* meta */
  grouping IRD-meta {
    uses cost-types;
    leaf default-alto-network-map {
      type resource-id;
      mandatory true;
    }
  }

  grouping network-map-meta {
    container vtag {
      uses vtag;
    }
  }

  grouping cost-map-meta {
    uses dependent-vtags {
      refine dependent-vtags {
        max-elements 1;
      }
    }
    container cost-type {
      uses cost-type;
    }
  }
  grouping endpoint-property-meta {
    uses dependent-vtags;
  }

  /* accepts (optional) */
  grouping accepts {
    leaf-list accepts {
      type media-type;
      min-elements 1;
    }
  }

  /* capabilities (capabilities) */
  grouping IRD-capabilities {
    container capabilities {
      leaf cost-constraints {
        type boolean;
      }
      leaf-list cost-type-names {
        type cost-type-name;
      }
      leaf-list prop-types {
        type endpoint-property-type;
      }
    }
  }

  /* uses (optional) */
  grouping uses {
    leaf-list uses {
      type resource-id;
      min-elements 1;
    }
  }

  /* Information Resource Directory Grouping */
  grouping IRD {
    container meta {
      uses IRD-meta;
    }
    uses IRD-data;
  }

  grouping IRD-data {
    list resources {
      key resource-id;
      leaf resource-id {
        type resource-id;
        mandatory true;
      }
      leaf uri {
        type inet:uri;
        mandatory true;
      }
      leaf media-type {
        type media-type;
        mandatory true;
      }
      uses accepts {
        when "current()";
      }
      uses IRD-capabilities {
        when "current()";
      }
      uses uses {
        when "current()";
      }
      description
        "IRDResourceEntry. RFC7285 9.2.2." +
        " object {
            JSONString      uri;
            JSONString      media-type;
            [JSONString     accepts;]
            [Capabilities   capabilities;]
            [ResourceID     uses<0..*>;]
          } IRDResourceEntry;" +
        "IRDResourceEntries. RFC7285 9.2.2." +
        " object-map {
            ResourceID  -> IRDResourceEntry;
          } IRDResourceEntries;" +
        "InformationResourceDirectory. RFC7285 9.2.2." +
        " object {
            IRDResourceEntries resources;
          } InfoResourceDirectory : ResponseEntityBase;";
    }
  }

  /* Network Map Grouping */
  grouping network-map {
    container meta {
      uses network-map-meta;
    }
    uses network-map-data;
  }

  grouping network-map-data {
    list network-map {
      key "pid";
      leaf pid {
        type pid-name;
      }
      uses endpoint-address-group;
      description
        "RFC7285 Sec. 11.2.1.6." +
        " object-map {
            PIDName -> EndpointAddrGroup;
          } NetworkMapData;";
    }
    description
      "Network map. RFC7285 Sec. 11.2.1.6." +
      "object {
         NetworkMapData network-map;
       } InfoResourceNetworkMap : ResponseEntityBase;";
  }

  /* Cost Map Grouping */
  grouping cost-map {
    container meta {
      uses cost-map-meta;
    }
    uses cost-map-data;
  }

  grouping cost-map-data {
    list cost-map {
      leaf src {
        type pid-name;
        description
          "Source PID.";
      }
      key "src";
      list dst-costs {
        leaf dst {
          type pid-name;
          description
            "Destination PID.";
        }
        key "dst";
        uses alto-cost {
          description
            "Cost from source to destination.";
        }
        description
          "The list represents the inner part of the cost matrix." +
          "DstCosts. RFC7285 Sec. 11.2.3.6." +
          " object-map {
              PIDName -> JSONValue;
            } DstCosts;";
      }
      description
        "The list represents the outer part of the cost matrix." +
        "CostMapData. RFC7285 Sec. 11.2.3.6." +
        " object-map {
            PIDName -> DstCosts;
          } CostMapData;";
    }
    description
      "Cost map. RFC7285 Sec. 11.2.3.6." +
      " object {
          CostMapData cost-map;
        } InfoResourceCostMap : ResponseEntityBase;";
  }

  /* Endpoint Property Map Grouping */
  grouping endpoint-property-map {
    container meta {
      uses endpoint-property-meta;
    }
    uses endpoint-property-map-data;
  }

  grouping endpoint-property-map-data {
    list endpoint-properties {
      key endpoint;
      leaf endpoint {
        type typed-endpoint-address;
        mandatory true;
      }
      list properties {
        key property-type;
        leaf property-type {
          type endpoint-property-type;
          mandatory true;
        }
        leaf property {
          type endpoint-property-value;
          mandatory true;
        }
        description
          "EndpointProps. RFC7285 Sec. 11.4.1.6." +
          " object {
              EndpointPropertyType -> JSONValue;
            } EndpointProps;";
      }
      description
        "EndpointPropertyMapData. Sec. 11.4.1.6." +
        " object-map {
            TypedEndpointAddr -> EndpointProps;
          } EndpointPropertyMapData;";
    }
    description
      "InfoResourceEndpointProperties. Sec. 11.4.1.6." +
      " object {
          EndpointPropertyMapData endpoint-properties;
        } InfoResourceEndpointProperties : ResponseEntityBase;";
  }
}
```

Figure 3: ALTO/YANG Common Types.

A.2. ALTO/YANG Model

```
   module alto-service {
     yang-version 1;

     namespace "urn:ietf:params:xml:ns:yang:alto-service";
     // TODO: replace with IANA namespace when assigned

     prefix "as";

     import alto-service-types {
       prefix alto;
     }

     organization "ALTO WG";
     contact "alto@ietf.org";

     description
       "This module defines a semantically equivalent data model
       for the ALTO services defined in RFC7285.";

     revision 2015-03-03 {
       description "Revise according to IETF91.";
     }

     revision 2014-11-01 {
       description "Inherit from alto-service-types.";
     }
     revision 2014-10-24 {
       description "Initial version.";
     }

     /******************************************
       Groupings for ALTO information resource
     *******************************************/

     grouping network-map-data {
       list map {
         key "pid";
         leaf pid {
           type alto:pid-name;
         }
         uses alto:endpoint-address-group;
       }
     }

     /* Network Map Grouping */
     grouping network-map {
       leaf resource-id {
         type alto:resource-id;
         mandatory true;
       }
       leaf tag {
         type alto:tag-string;
         mandatory true;
       }
       uses network-map-data;
     }

     grouping cost-map-data {
       list map {
         leaf src {
           type alto:pid-name;
         }
         key "src";
         list dst-costs {
           leaf dst {
             type alto:pid-name;
           }
           key "dst";
           uses alto:alto-cost;
         }
       }
     }

     /* Cost Map Grouping */
     grouping cost-map {
       leaf resource-id {
         type alto:resource-id;
         mandatory true;
       }
       leaf tag {
         type alto:tag-string;
         mandatory true; //TODO: sx: mandatory tag? tag manager?
       }
       container meta {
         must "current()";
         uses alto:cost-map-meta;
       }
       uses cost-map-data;
     }

     grouping alto-resources {
       container IRD {
         uses alto:IRD;
       }

       container network-maps {
         list network-map {
           key "resource-id";
           uses network-map;
         }
       }

       container cost-maps {
         list cost-map {
           key "resource-id";
           uses cost-map;
         }
       }

       container endpoint-property-map {
         uses alto:endpoint-property-map;
       }
     }

     /****************************************************
        DATA INSTANCES of all ALTO information resources

        unfiltered network-maps, unfiltered cost-maps are all instances
        of resources. IRD is also modeled as data.

        The design uses augment as the basic approach to implement
        inheritance.
```

```
      ****************************************************/

     container resources {
       uses alto-resources;
     }
   }
```

Figure 4: ALTO YANG Model.

Appendix B. Using NETCONF to Read ALTO/YANG Information

NETCONF has provided two RPCs to retrieve data: the <get> operation and the <get-config> operation. The <get> operation can be used to retrieve both configuration data and state data, whereas the <get-config> operation is used to retrieve only configuration data. Since state data is read-only, in the current design, we model ALTO resources as configuration data. The <get> operation retrieves only the configuration data in the running datastore where as the <get-config> operation can specify the target datastore. Hence we use <get-config> as an example, and <get> operation is analogous. The <get-config> operation defines two filter types in the NETCONF base protocol [RFC6241], subtree filtering and optional XPATH filtering capabilities. Hence we show examples for both types.

B.1. Full Network Map Service

B.1.1. Approach 1: Using Subtree Filtering

To retrieve a network map with resource-id INPUT-NETWORK-MAP-RESOURCE-ID, the client specifies it in the following netconf RPC template:

```
<rpc message-id=SEQ-NUM
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter type="subtree">
      <resources
        xmlns="urn:ietf:params:xml:ns:yang:alto-service">
        <network-maps>
          <network-map>
            <resource-id>INPUT-NETWORK-MAP-RESOURCE-ID</resource-id>
            <tag/>
            <map/>
          </network-map>
        </network-maps>
      </resources>
    </filter>
  </get-config>
</rpc>
```

One can observe that the query mapping specifies not only the resource-id as a content matching node, but also the tag and map nodes, as two selection nodes, to indicate that these two fields should be included in the filter output. A simpler mapping, using the default processing of filtering output (the last output rule of Section 6.2.5 of [RFC6241]), is to omit <tag/> and <map/>. This will give the same output. We suggest the more complete template for more explicit results.

An example of the query, for a network map with resource-id myNetMap1, is the following:

```
<rpc message-id=SEQ-NUM
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter type="subtree">
      <resources
        xmlns="urn:ietf:params:xml:ns:yang:alto-service">
        <network-maps>
          <network-map>
            <resource-id>myNetMap1</resource-id>
            <tag/>
            <map/>
          </network-map>
        </network-maps>
      </resources>
    </filter>
  </get-config>
</rpc>
```

An example reply from the server then will be:

```
   <rpc-reply message-id=SEQ-NUM
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <data>
       <resources
         xmlns="urn:ietf:params:xml:ns:yang:alto-service">
         <network-maps>
           <network-map>
             <resource-id>myNetMap1</resource-id>
             <tag>da65eca2tus10ce8b0740a1938e3f8eb1d4785</tag>
             <map>
               <pid>PID1</pid>
               <endpoint-address-group>
                 <address-type>ipv4</address-type>
                 <endpoint-prefix>192.0.2.0/24</endpoint-prefix>
                 <endpoint-prefix>198.51.100.0/25</endpoint-prefix>
               </endpoint-address-group>
             </map>
             <map>
               <pid>PID2</pid>
               <endpoint-address-group>
                 <address-type>ipv4</address-type>
                 <endpoint-prefix>198.51.100.128/25</endpoint-prefix>
               </endpoint-address-group>
             </map>
             <map>
               <pid>PID3</pid>
               <endpoint-address-group>
                 <address-type>ipv4</address-type>
                 <endpoint-prefix>0.0.0.0/0</endpoint-prefix>
               </endpoint-address-group>
               <endpoint-address-group>
                 <address-type>ipv6</address-type>
                 <endpoint-prefix>::/0</endpoint-prefix>
               </endpoint-address-group>
             </map>
           </network-map>
         </network-maps>
       </resources>
     </data>
   </rpc-reply>
```

B.1.2. Approach 2: Using XPATH Filtering

To retrieve a network map with resource-id INPUT-NETWORK-MAP-RESOURCE-ID, the client specifies it in the following netconf RPC template:

```
<rpc message-id=SEQ-NUM
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter
      xmlns:t="urn:ietf:params:xml:ns:yang:alto-service"
      type="xpath"
      select="/t:resources/t:network-maps/t:network-map
              [t:resource-id=INPUT-NETWORK-MAP-RESOURCE-ID]" />
  </get-config>
</rpc>
```

Note that [RFC6241] requires only that a NETCONF server MAY support xpath. Hence, this approach may or may not work on a given NETCONF server.

An example of the query, for a network map with resource-id myNetMap1, is the following. Note that the XPATH expression would select the appropriate <network-map> node including its subtree, but not the ancestor node <resources> or <network-maps>. According to the NETCONF modification rules defined in Section 8.9.5.1 of [RFC6241], the ancestor nodes of the XPATH result are also encoded, in particular, the tags <resources> and <network-maps>.

```
<rpc message-id=SEQ-NUM
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter
      xmlns:t="urn:ietf:params:xml:ns:yang:alto-service"
      type="xpath"
      select="/t:resources/t:network-maps/t:network-map
              [t:resource-id='myNetMap1']" />
  </get-config>
</rpc>
```

The output of this query is the same as the subtree based query.

B.2. Filtered Cost Map Service

Then we consider filtered cost map service. For consistency, we consider the same yang module as in Figure 2. For ease of reading, we duplicate the tree diagram here:

```
   +--ro cost-maps
   |  +--ro cost-map* [resource-id]
   |     +--ro resource-id    alto:resource-id
   |     +--ro tag            alto:tag-string
   |     +--ro meta
   |     |  +--ro dependent-vtags*
   |     |  |  +--ro resource-id    resource-id
   |     |  |  +--ro tag            tag-string
   |     |  +--ro cost-type
   |     |     +--ro cost-mode      cost-mode
   |     |     +--ro cost-metric    cost-metric
   |     |     +--ro description?   string
   |     +--ro map* [src]
   |        +--ro src          alto:pid-name
   |        +--ro dst-costs* [dst]
   |           +--ro dst     alto:pid-name
   |           +--ro cost
```

B.2.1. Approach 1: Using Subtree Filtering

To retrieve INPUT-SRC-PID-1, INPUT-SRC-PID-2, ..., INPUT-SRC-PID-p and INPUT-DST-PID-1, INPUT-DST-PID-2, ..., INPUT-DST-PID-q of a cost map with resource-id INPUT-COST-MAP-RESOURCE-ID, the client specifies it in the following NETCONF RPC template:

```
   <rpc message-id=SEQ-NUM
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <get-config>
       <source>
         <running/>
       </source>
       <filter type="subtree">
         <resources
           xmlns="urn:ietf:params:xml:ns:yang:alto-service">
           <cost-maps>
             <cost-map>
               <resource-id>
                 INPUT-COST-MAP-RESOURCE-ID
               </resource-id>
               <tag/>
               <meta/>
               <map>
                 <src>INPUT-SRC-PID-1</src>
                 <dst-costs>
                   <dst>INPUT-DST-PID-1</dst>
                   <cost/>
                 </dst-costs>
                 ...
                 <dst-costs>
                   <dst>INPUT-DST-PID-q</dst>
                   <cost/>
                 </dst-costs>
               </map>
               ...
               <map>
                 <src>INPUT-SRC-PID-p</src>
                 <dst-costs>
                   <dst>INPUT-DST-PID-1</dst>
                   <cost/>
                 </dst-costs>
                 ...
                 <dst-costs>
                   <dst>INPUT-DST-PID-q</dst>
                   <cost/>
                 </dst-costs>
               </map>
             </cost-map>
           </cost-maps>
         </resources>
       </filter>
     </get-config>
   </rpc>
```

One can observe that in the template, client must list all combinations of src and dst to retrieve the cost, which is very inefficient when the number of src and dst grows.

An example of the query, for src: PID1, PID3 and dst: PID2, PID3 of a cost map with resource-id myCostMap1, is the following:

```
   <rpc message-id=SEQ-NUM
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <get-config>
       <source>
         <running/>
       </source>
       <filter type="subtree">
         <resources
           xmlns="urn:ietf:params:xml:ns:yang:alto-service">
           <cost-maps>
             <cost-map>
               <resource-id>
                 myCostMap1
               </resource-id>
               <tag/>
               <meta/>
               <map>
                 <src>PID1</src>
                 <dst-costs>
                   <dst>PID2</dst>
                   <cost/>
                 </dst-costs>
                 <dst-costs>
                   <dst>PID3</dst>
                   <cost/>
                 </dst-costs>
               </map>
               <map>
                 <src>PID3</src>
                 <dst-costs>
                   <dst>PID2</dst>
                   <cost/>
                 </dst-costs>
                 <dst-costs>
                   <dst>PID3</dst>
                   <cost/>
                 </dst-costs>
               </map>
             </cost-map>
           </cost-maps>
         </resources>
       </filter>
     </get-config>
   </rpc>
```

An example reply from the server then will be:

```
   <rpc-reply message-id=SEQ-NUM
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <data>
       <resources
         xmlns="urn:ietf:params:xml:ns:yang:alto-service">
         <cost-maps>
           <cost-map>
             <resource-id>myCostMap1</resource-id>
             <tag>tus10ce8b0740a1938e3f8eb1d4785da65eca2</tag>
             <meta>
               <dependent-vtags>
                 <resource-id>myNetMap1</resource-id>
                 <tag>da65eca2tus10ce8b0740a1938e3f8eb1d4785</tag>
               </dependent-vtags>
               <cost-type>
                 <cost-mode>numerical</cost-mode>
                 <cost-metric>routingcost</cost-metric>
               </cost-type>
             </meta>
             <map>
               <src>PID1</src>
               <dst-costs>
                 <dst>PID2</dst>
                 <cost>5</cost>
               </dst-costs>
               <dst-costs>
                 <dst>PID3</dst>
                 <cost>10</cost>
               </dst-costs>
             </map>
             <map>
               <src>PID3</src>
               <dst-costs>
                 <dst>PID2</dst>
                 <cost>15</cost>
               </dst-costs>
             </map>
           </cost-map>
         </cost-maps>
       </resources>
     </data>
   </rpc-reply>
```

B.2.2. Approach 2 : Using XPath

To retrieve INPUT-SRC-PID-1, INPUT-SRC-PID-2, ..., INPUT-SRC-PID-p and INPUT-DST-PID-1, INPUT-DST-PID-2, ..., INPUT-DST-PID-q of a cost map with resource-id INPUT-COST-MAP-RESOURCE-ID, the client specifies it in the following netconf RPC template:

```
<rpc message-id=SEQ-NUM
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter
      xmlns:t="urn:ietf:params:xml:ns:yang:alto-service"
      type="xpath"
      select="/t:resources/t:cost-maps/t:cost-map
          [t:resource-id=INPUT-COST-MAP-RESOURCE-ID]/t:resource-id
        | /t:resources/t:cost-maps/t:cost-map
          [t:resource-id=INPUT-COST-MAP-RESOURCE-ID]/t:tag
        | /t:resources/t:cost-maps/t:cost-map
          [t:resource-id=INPUT-COST-MAP-RESOURCE-ID]/t:meta
        | /t:resources/t:cost-maps/t:cost-map
          [t:resource-id=INPUT-COST-MAP-RESOURCE-ID]/
          t:map[t:src=INPUT-SRC-PID-1 or t:src=INPUT-SRC-PID-2
            or ... or t:src=INPUT-SRC-PID-p]/t:src
        | /t:resources/t:cost-maps/t:cost-map
          [t:resource-id=INPUT-COST-MAP-RESOURCE-ID]/
          t:map[t:src=INPUT-SRC-PID-1 or t:src=INPUT-SRC-PID-2
            or ... or t:src=INPUT-SRC-PID-p]/t:dst-costs
          [t:dst=INPUT-DST-PID-1 or t:dst=INPUT-DST-PID-2
            or ... or t:dst=INPUT-DST-PID-q]" />
  </get-config>
</rpc>
```

One observe that similar to the xpath template for filtered network map, the xpath template for filtered cost map also used XPATH union in the select attribute of the filter, which is necessary because the XPATH operation will not select the resource-id and tag and src nodes if we only have the XPATH expression "/t:resources/t:cost-maps/ t:cost-map[t:resource-id=INPUT-COST-MAP-RESOURCE- ID]/t:map[t:src=INPUT-SRC-PID-1 or t:src=INPUT-SRC-PID-2 or ... or t:src=INPUT-SRC-PID-p]/t:dst-costs[t:dst=INPUT-DST-PID-1 or t:dst=INPUT-DST-PID-2 or ... or t:dst=INPUT-DST-PID-q]".

Despite that one needs to specify the src PIDs multiple times, there is no need to provide a cartesian product of src PIDs and dst PIDs, which is more efficient compared to the subtree filtering approach.

An example of the query, for srcs: PID1, PID3 and dsts: PID2, PID3 of a cost map with resource-id myCostMap1, is the following:

```
<rpc message-id=SEQ-NUM
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter
      xmlns:t="urn:ietf:params:xml:ns:yang:alto-service"
      type="xpath"
      select="/t:resources/t:cost-maps/t:cost-map
          [t:resource-id='myCostMap1']/t:resource-id
        | /t:resources/t:cost-maps/t:cost-map
          [t:resource-id='myCostMap1']/t:tag
        | /t:resources/t:cost-maps/t:cost-map
          [t:resource-id='myCostMap1']/t:meta
        | /t:resources/t:cost-maps/t:cost-map
          [t:resource-id='myCostMap1']/
          t:map[t:src='PID1' or t:src='PID3']/t:src
        | /t:resources/t:cost-maps/t:cost-map
          [t:resource-id='myCostMap1']/
          t:map[t:src='PID1' or t:src='PID3']/t:dst-costs
          [t:dst='PID2' or t:dst='PID3']" />
  </get-config>
</rpc>
```

An example reply from the server then will be:

```
   <rpc-reply message-id=SEQ-NUM
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <data>
       <resources
         xmlns="urn:ietf:params:xml:ns:yang:alto-service">
         <cost-maps>
           <cost-map>
             <resource-id>myCostMap1</resource-id>
             <tag>tus10ce8b0740a1938e3f8eb1d4785da65eca2</tag>
             <meta>
               <dependent-vtags>
                 <resource-id>myNetMap1</resource-id>
                 <tag>da65eca2tus10ce8b0740a1938e3f8eb1d4785</tag>
               </dependent-vtags>
               <cost-type>
                 <cost-mode>numerical</cost-mode>
                 <cost-metric>routingcost</cost-metric>
               </cost-type>
             </meta>
             <map>
               <src>PID1</src>
               <dst-costs>
                 <dst>PID2</dst>
                 <cost>5</cost>
               </dst-costs>
               <dst-costs>
                 <dst>PID3</dst>
                 <cost>10</cost>
               </dst-costs>
             </map>
             <map>
               <src>PID3</src>
               <dst-costs>
                 <dst>PID2</dst>
                 <cost>15</cost>
               </dst-costs>
             </map>
           </cost-map>
         </cost-maps>
       </resources>
     </data>
   </rpc-reply>
```

Appendix C. Using RESTCONF to Read ALTO/YANG Information

The RESTCONF Protocol [RESTCONF] provides a RESTful interface to data defined in YANG. RESTCONF must support XML and may support JSON encoding. We can retrieve information with HTTP GET method and update information with PATCH method.

RESTCONF uses the following HTTP request structure for clients to encode query parameters:

```
        <OP> /<restconf>/<path>?<query>#<fragment>

         ^       ^        ^       ^         ^
         |       |        |       |         |
       method  entry  resource  query    fragment

         M       M        O        O         I

       M=mandatory, O=optional, I=ignored

       <text> replaced by client with real values
```

The particular relevant components are the path component and the query component. The goal of the path component (Section 3.5.1 of [RESTCONF]) is to identify a single node (referred to as the target resource) in the data tree. This is different from xpath, whose basic Location Path concept is built on node set; that is, the goal of xpath is to identify a set of nodes (address parts of an XML document), not a single node. The query component (Section 3.8 of [RESTCONF]) consists of a set of "name=value" pairs, with a given set of names (query parameters) to control the query behavior. A particularly relevant parameter is select, which allows a client to request a subset of the target resource contents. The exact syntax of select is specified in Section 3.8.4 of [RESTCONF]. Note that all query parameters are optional to implement by the server and optional to use by the client.

C.1. Example: Full Network Map Service

To request a network map with resource-id INPUT-NETWORK-MAP-RESOURCE-ID, the RESTCONF client sends the following HTTP request:

```
GET /restconf/data/alto-service:resources/network-maps
  /network-map=INPUT-NETWORK-MAP-RESOURCE-ID?content=all HTTP/1.1
Host: alto.example.com
Accept: application/yang.data+json,application/yang.errors+json
```

An example query for a network map whose resource-id is myNetMap1 is:

```
   GET /restconf/data/alto-service:resources/network-maps
     /network-map=myNetMap1?content=all HTTP/1.1
   Host: alto.example.com
   Accept: application/yang.data+json,application/yang.errors+json
```

An example response from a RESTCONF server can be:

```
   HTTP/1.1 200 OK
   Date: Mon, 31 Oct 2011 23:59:00 GMT
   Server: example-alto-server
   Cache-Control: no-cache
   Pragma: no-cache
   Content-Type: application/yang.data+json

   {
     "alto-service:network-map" : {
       "resource-id" : "myNetMap1",
       "tag" : "da65eca2tus10ce8b0740a1938e3f8eb1d4785",
       "map": [
         {
           "pid": "PID1",
           "endpoint-address-group": {
             "address-type": "ipv4",
             "endpoint-prefix": [
               "192.0.2.0/24",
               "198.51.100.0/25"
             ]
           }
         },
         {
           "pid": "PID2",
           "endpoint-address-group": {
             "address-type": "ipv4",
             "endpoint-prefix": ["198.51.100.128/25"]
           }
         },
         {
           "pid": "PID3",
           "endpoint-address-group": [
             {
               "address-type": "ipv4",
               "endpoint-prefix": ["0.0.0.0/0"]
             },
             {
               "address-type": "ipv6",
               "endpoint-prefix": ["::/0"]
             }
           ]
         }
       ]
     }
   }
```

C.2. Impossibility to Encode Filtered Maps and Endpoint Properties using Standard RESTCONF Query

The preceding sections provided templates to implement full network map service using standard operations in RESTCONF; full cost map services are analogous. However, it is not possible to do so for the filtered map services or the endpoint property service.

Specifically, supporting the filtered map services requires the ability to select multiple nodes from the data tree based on the data content. Recall that the path component in the uri can return only a single node. Hence, the only component that allows one to select multiple nodes is the query component, specifically, the "fields" query parameter. However, the definition of the fields expression (Section 4.8.5 of [RESTCONF]) only includes terms defined in a schema, not any data content (i.e., select does not include any content match capabilities). Hence, it is impossible to implement filtered maps.

Authors' Addresses

Xiao Shi
Yale University
51 Prospect Street
New Haven, CT 06511
USA

Email: xiao.shi@yale.edu

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

Email: yang.r.yang@gmail.com

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

Email: michael.scharf@alcatel-lucent.com

ALTO WG
Internet-Draft
Intended status: Standards Track
Expires: September 9, 2015

G. Bernstein
Grotto Networking
Y. Lee
Huawei
W. Roome
M. Scharf
Alcatel-Lucent
Y. Yang
Yale University
March 8, 2015

ALTO Topology Extension: Path Vector as a Cost Mode
draft-yang-alto-path-vector-00.txt

Abstract

The Application-Layer Traffic Optimization (ALTO) Service has defined network and cost maps to provide basic network information, where the cost maps allow only scalar (numerical or ordinal) cost mode values. This document introduces a new cost mode called path vector to allow ALTO clients to better distinguish cost information. This document starts with a non-normative use case called multi-flow scheduling to illustrate that ALTO cost maps without path vectors cannot provide sufficient information. This document then defines path vector as a new cost mode.

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

Table of Contents

1. Introduction

The ALTO base protocol [RFC7285] is designed for a setting of exposing network topology using the extreme "my-Internet-view" representation, which abstracts a whole network as a single node that has a set of access ports, with each port connects to a set of endhosts. The base protocol refers to each access port as a PID. This "single-node" abstraction is simple and can support a wide range of applications already.

A problem of this abstraction, however, is that it does not provide sufficient information for use cases such as multi-flow scheduling (see Section 3), which essentially require exposure of topology information beyond the single-node abstraction, to detect sharing of resources in the underlying topology.

This document goes beyond single-node topology by introducing path vector as a new ALTO cost mode, where each path vector specifies the network elements on the routing path from a set of source endhosts to a set of destination endhosts. Since the network elements on a path vector are abstract network elements defined by ALTO servers, the new path-vector cost mode provides a mechanism to allow a network to control the level of topology exposure, and at the same time better support application traffic optimization. The design of path vector is based on the ALTO WG discussions at IETF 89, with summary slides at http://tools.ietf.org/agenda/89/slides/slides-89-alto-2.pdf.

The organization of this document is organized as follows. Section 2 gives a non-normative use case called multi-flow scheduling to illustrate the need to introduce path vectors. Section 3 formally specifies the path vector cost mode. Sections 4 and 5 discuss security and IANA considerations.

2. The Multi-flow Scheduling Use Case

ALTO uses a simple single-node network abstraction. Specifically, each network map in ALTO defines an abstract, single node network, where endhosts are partitioned to a set of access ports, with each access port called a PID. For a given network map, a cost map of a given cost metric provides a scalar (numerical or ranking) cost value for each pair of source and destination PIDs.

Although simple, the single-node, simple scalar cost maps may not convey enough information to the applications about pair-wise connection properties between one PID and another PID. See [I-D.bernstein-alto-topo] for a survey of use-cases where extended network topology information is needed.

This document uses a simple use case to illustrate the issue. 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.

The single-node ALTO topology abstraction of the network is shown in Figure 2.

```
                    +----------------------+
           {eh1}    |                      |     {eh2}
           PID1     |                      |     PID2
             +------+                      +------+
                    |                      |
                    |                      |
           {eh3}    |                      |     {eh4}
           PID3     |                      |     PID4
             +------+                      +------+
                    |                      |
                    +----------------------+
```

Figure 2: Base Single-Node Topology Abstraction.

Consider an application overlay (e.g., a large data analysis 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 providing end-to-end available bandwidth, using 'available bw' as cost-metric and 'numerical' as cost-mode, where the 'available bw' between two PIDs represents possible bandwidth for PIDi -> PIDj, if no other applications use shared resources.

Assume that the application receives from the cost map that both PID1 -> PID2 and PID3 -> PID4 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 PID1 -> PID2 and PID3 -> PID4 use different paths, 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 PID1 -> PID2 and PID3 -> PID4 share the bottleneck, 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 possible cases, the network needs to provide more details. This document introduces path vector to resolve the issue.

3. Path-Vector as a new Cost Mode

An extension supporting the path-vector cost-mode MUST support the following following extension of Section 11.2.3.6 of [RFC7285]:

```
    object {
      cost-map.DstCosts.JSONValue -> JSONString<0,*>;
      meta.cost-mode = "path-vector";
    } InfoResourcePVCostMap : InfoResourceCostMap;
```

Specifically, the preceding specifies that InfoResourcePVCostMap extends InfoResourceCostMap. The body specifies that the first extension is achieved by changing the type of JSONValue defined in DstCosts of cost-map to be an array of JSONString; the second extension is that the cost-mode of meta MUST be "path-vector".

An example cost map using path-vector is the following:

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

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

     {
       "meta" : {
         "dependent-vtags" : [
           { "resource-id": "my-default-network-map",
             "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"
           },
           {"resource-id": "my-topology-map", // See below
            "tag": "4xee2cb7e8d63d9fab71b9b34cbf76443631554de"
           }
         ],
         "cost-type" : {"cost-metric": "routingcost",
                        "cost-mode"  : "path-vector"
         }
       },

       "cost-map" : {
         "PID1": { "PID1":[],
                   "PID2":["ne56", "ne67"],
                   "PID3":[],
                   "PID4":["ne57"]
         },
         "PID2": { "PID1":["ne75"],
                   "PID2":[],
                   "PID3":["ne75"],
                   "PID4":[]
         },
         "PID3": { "PID1":[],
                   "PID2":["ne57"],
                   "PID3":[],
                   "PID4":["ne57"]
         },
         "PID4": { "PID1":["ne75"],
                   "PID2":[],
                   "PID3":["ne75"],
                   "PID4":[]}
       }
     }
```

To interpret the path-vector in a cost map providing path vectors, the client will need access to the properties of the network elements named in the path vectors. Such properties should be provided from an element property service. Hence, the "dependent-tags" of a cost map supporting path vectors MUST include two dependent resources: one for a network map, and the other for an element property service. This document does not define the property service. The appendix gives one definition, but it can be a different one.

4. Security Considerations

This document has not conducted its security analysis.

5. IANA Considerations

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

6. Acknowledgments

The author thanks discussions with Xiao Shi, Xin Wang, Erran Li, Tianyuan Liu, Andreas Voellmy, Haibin Song, and Yan Luo.

7. References

7.1. Normative References

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

7.2. Informative References

[I-D.amante-i2rs-topology-use-cases]
Medved, J., Previdi, S., Lopez, V., and S. Amante, "Topology API Use Cases", draft-amante-i2rs-topology-use-cases-01 (work in progress), October 2013.

[I-D.clemm-i2rs-yang-network-topo]
Clemm, A., Medved, J., Tkacik, T., Varga, R., Bahadur, N., and H. Ananthakrishnan, "A YANG Data Model for Network Topologies", draft-clemm-i2rs-yang-network-topo-01 (work in progress), October 2014.

[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.

[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.

Appendix A. Network Element Properties Map

A missing piece to complete the path-vector design to resolve the ambiguity in the use case is how to provide information on the elements of the path vectors. A minimal approach is to introduce network element properties (NEP) maps, where each NEP map provides a mapping from a network element to its properties such as bandwidth or shared risk link group (srlg).

A schema of an NEP map is:

```
object-map {
  JSONString -> NetworkElementProperties; // name to properties
} NetworkElementMapData;

object-map {
  JSONString bw;
  JSONString srlg<0,*>;
  [JSONString type;] // should be from an enumeration only
} NetworkElementProperties;
```

An example network element property map:

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

```
HTTP/1.1 200 OK
Content-Length: TBD
Content-Type: application/alto-nepmap+json

{
  "meta" : {
    "vtag": {
      "resource-id": "my-topology-map",
      "tag": "da65eca2eb7a10ce8b059740b0b2e3f8eb1d4785"
    }
  },
  "nep-map" : {
    "ne57" : {"bw" : 100, "srlg" : [1, 3]}, // link sw5->sw7
    "ne75" : {"bw" : 100, "srlg" : [1, 3]}, // link sw7->sw5
    "ne56" : {"bw" : 100, "srlg" : [1]},    // link sw5->sw6
    "ne65" : {"bw" : 100, "srlg" : [1]},    // link sw6->sw5
    "ne67" : {"bw" : 100, "srlg" : [3]},    // link sw6->sw7
    "ne76" : {"bw" : 100, "srlg" : [3]},    // link sw7->sw6
  }
}
```

Authors' Addresses

Greg Bernstein
Grotto Networking
Fremont, CA
USA

Email: gregb@grotto-networking.com

Young Lee
Huawei
TX
USA

Email: leeyoung@huawei.com

Wendy Roome
Alcatel-Lucent Technologies/Bell Labs
600 Mountain Ave, Rm 3B-324
Murray Hill, NJ 07974
USA

Phone: +1-908-582-7974
Email: w.roome@alcatel-lucent.com

Michael Scharf
Alcatel-Lucent Technologies
Germany

Email: michael.scharf@alcatel-lucent.com

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

Email: yry@cs.yale.edu

ALTO WG
Internet-Draft
Intended status: Standards Track
Expires: September 10, 2015

G. Bernstein
Grotto Networking
Y. Lee
Huawei
W. Roome
M. Scharf
Alcatel-Lucent
Y. Yang
Yale University

March 9, 2015

ALTO Topology Extensions: Node-Link Graphs
draft-yang-alto-topology-06.txt

Abstract

The Application-Layer Traffic Optimization (ALTO) Service has defined network and cost maps to provide basic network information. In this document, we discuss designs to provide abstracted (node-link) graph representations of network topology.

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

Table of Contents

1. Introduction

Topology is a basic information component that a network can provide to network management tools and applications. Example tools and applications that can utilize network topology include traffic engineering, network services (e.g., VPN) provisioning, PCE, application overlays, among others [RFC5693,I-D.amante-i2rs-topology-use-cases, I-D.lee-alto-app-net-info-exchange].

A basic challenge in exposing network topology is that there can be multiple representations of the topology of the same network infrastructure, and each representation may be better suited for its own set of deployment scenarios. For example, the current ALTO base protocol [RFC7285] is designed for a setting of exposing network topology using the extreme "my-Internet-view" representation, which abstracts a whole network as a single node that has a set of access ports, with each port connects to a set of endhosts called endpoints. The base protocol refers to each access port as a PID. This "single-node" abstraction achieves simplicity and provides flexibility. A problem of this abstraction, however, is that the base protocol as currently defined does not provide sufficient information for use cases such as the multi-flow scheduling use case (see [draft-yang-alto-path-vector]).

An opposite of the single-node representation is the complete raw topology, spanning across multiple layers, to include all details of network states such as endhosts attachment, physical links, physical switch equipment, and logical structures (e.g., LSPs) already built on top of the physical infrastructural devices. A problem of the raw topology representation, however, is that its exposure may violate privacy constraints. Also, a large raw topology may be overwhelming and unnecessary for specific applications. Since the target of ALTO is general applications which do not want or need to understand detailed routing protocols or raw topology collected in routing information bases (RIB), raw topology does not appear to be a good fit for ALTO.

A main objective of this document is to specify a new type of ALTO Information Resources, which provide abstracted graph (node-link) representations of a network to provide only enough information for applications. We call such Information Resources ALTO topology maps, or topology maps for short. Different from the base single-node abstraction, a topology map includes multiple network nodes. Different from the raw topology representation that uses real network nodes, a topology map may use abstract nodes, although they will be constructed from the real, raw topology, in order to provide grounded information. The design of this document is based on the ALTO WG discussions at IETF 89, with summary slides at http://tools.ietf.org/agenda/89/slides/slides-89-alto-2.pdf.

The organization of this document is organized as follows. We first review the ALTO base protocol in Section 2. In Section 3, we give a node-link representation.

2. Review: the Base Single-Node Representation

We distinguish between endhosts and the network infrastructure of a network. Endhosts are sources and destinations of data that the network infrastructure carries. The network itself is neither the source nor the destination of data.

For a given network, it provides "access ports" (interfaces, or access points) where data signal from endhosts enter and leave the network infrastructure. One should understand "access ports" in a generic sense. For example, an access port can be a physical Ethernet port connecting to a specific endhost, or it can be a port connecting to a CE which connects to a large number of endhosts. Let AP be the set of access ports (AP) that the network provides.

A high-level abstraction of a network topology is only the set AP, and one can visualize, as Figure 1, the network as a single, abstract node with the set AP of access ports attached. At each ap in AP, a set of endhosts are attached to send or receive information from the network. Let attach(ap) denote the set of endhosts attached to ap.

```
                    +----------------------+
           ap_1     |                      |
             +------+                      +------+
                    |                      |
                    |                      |
             +------+                      +------+
                    |                      |

                    |                      |
             +------+                      +------+
                    |                      |
                    |                      |
             +------+                      +------+
                    |                      |     ap_n
                    +----------------------+
```

Figure 1: Base Single-Node Topology Abstraction.

There can be multiple ways to partition the set AP. Each partition is called a network map. Given a complete partition of AP, the ALTO base protocol introduces PID to represent each partition subset. The ALTO base protocol then conveys the pair-wise connection properties between one PID and another PID through the "single-node". This is the cost map.

3. Topology using a Graph (Node-Link) Representation

3.1. Use Cases

[draft-yang-alto-path-vector] proposes path vectors to extend the preceding topology to expose network elements. A potential problem of the path vector representation, however, is its lacking of compactness. For example, suppose a network has N PIDs, then it will need to represent N * (N-1) paths, if each source-destination pair has one path computed using a shortest-path algorithm. On the other hand, the underlying graph may have only O(F * N) elements, where F is the average degree of the topology, and hence can be a much smaller value than N. For such settings, in particular, when privacy protection is not an issue (e.g., in the same-trust domain setting), a node-link representation can be more compact.

Another setting where a node-link graph approach is beneficial is application guided path selection. With a topology graph, an application can compute maximum flows to discover the desired paths and signal (out the scope of this document) to the network to set up the paths. The computation can be done by the application itself, or through a third entity such as a PCE server. The recent development of SDN makes this use case more possible. A requirement of realizing this use case is that the path computed by the application is realizable, in particular, when the topology is an abstract topology. By realizable, we mean that a path computed on the abstract topology can be converted to configurations on network devices to achieve the properties in the abstract topology.

3.2. A Node-Link Schema

A schema for the graph (node-link) representation, based on the types already defined in the base ALTO protocol, is the following:

```
     object {
       TopologyMapData topology-map;
     } InfoResourceTopologyMap : ResponseEntityBase;

     object {
       NodeMapData nodes;
       LinkMapData links;
     } TopologyMapData;

     object-map {
       JSONString -> NodeProperties; // node name to properties
     } NodeMapData;

     object {
       JSONString type;
       ...
     } NodeProperties;


     object-map {
       JSONString -> LinkProperties; // link name to properties
     } LinkMapData;

     object {
       JSONString src;
       JSONString dst;
       JSONString type;
       CostValue costs<0,*>;
     } LinkProperties;

     object {
       CostMetric metric;
       JSONValue value; // value type depends on metric type
     } CostValue;
```

In particular, the schema distinguishes two types of links: edge-attach, and core, where the former is for a link between a network element and a group of endhosts (PID), and the later is between two network elements.

An example using the schema is:

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




     HTTP/1.1 200 OK
     Content-Length: TBD
     Content-Type: application/alto-topologymap+json

     {
       "meta" : {
         "dependent-vtags" : [
           { "resource-id": "my-default-network-map",
             "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"
           }
         ],
         "vtag": {
           "resource-id": "my-topology-map",
           "tag": "da65eca2eb7a10ce8b059740b0b2e3f8eb1d4785"
         }
       },
       "topology-map" : {
         "nodes" : {
           "sw1" : {"type" : "switch"},
           "sw2" : {"type" : "switch"},
           "sw3" : {"type" : "switch"},
           "sw4" : {"type" : "switch"},
           "sw5" : {"type" : "switch"},
           "sw6" : {"type" : "switch"},
           "sw7" : {"type" : "switch"}
         },
         "links" : {
           "e1" : {"src" : "PID1",
                   "dst" : "sw1",
                   "type": "edge-attach",
                   "costs" : [
                     {"cost-metric" : "availbw", "value" : 100
                     },
                     {"cost-metric" : "srlg", value : [1, 3]}

                   ]
           },
           "e2" : {"src" : "PID2",
                   "dst" : "sw2",
                   "type": "edge-attach",
                   ...
           },
           "e3" : {"src" : "PID3",
                   "dst" : "sw3",
                   ...
           },
           "e4" : {"src" : "PID4",
                   "dst" : "sw4",
                   "type": "edge-attach",
                   ...
           },
           "e15" : {"src" : "sw1",
                    "dst" : "sw5",
                    "type": "core",
                    ...
           },
           "e35" : {"src" : "sw3",
                    "dst" : "sw5",
                    "type": "core",
                    ...
           },
           "e27" : {"src" : "sw2",
                    "dst" : "sw7",
                    "type": "core",
                    ...
           },
           "e47" : {"src" : "sw4",
                    "dst" : "sw7",
                    "type": "core",
                    ...
           },
           "e57" : {"src" : "sw5",
                    "dst" : "sw7",
                    "type": "core",
                    ...
           },
           "e56" : {"src" : "sw5",
                    "dst" : "sw6",
                    "type": "core",
                    ...
           },
           "e67" : {"src" : "sw6",
                    "dst" : "sw7",
                    "type": "core",
                    ...
           }
         }
       }
```

```
     }
```

3.3. Discussions

The node-link schema specified in the preceding section is still a standard graph representation of a network (graph). An alternative design, which may provide substantial benefit, is using a property graph design. In particular, in a property graph based design, it is unnecessary that a node in the property graph represents a network node, a link in the property graph represents a network link. Instead, network nodes, network links and network paths can all be represented as nodes in a property graph, and links represent their relationship. This design can be flexible in modeling settings such as topology abstraction (e.g., to denote, in the same graph, that a network link is composed of a path, through a aggregation label). Property-graph frameworks such as Gremlin can provide powerful and compact querying languages for application’s usage.

Using either the standard node-link graph in the preceding section or the property graph abstraction, one may not use a rigid hierarchical design. Consider a model that uses a strict hierarchy, and a higher layer node can specify a set of nodes in the lower layer as supporting nodes; a higher layer link can specify a set of links in the lower layer as supporting links [draft-clemm-i2rs-yang-network-topo-01]. To test the problem of that model, consider a simple topology. Assume that the network consists of 3 data centers (dc1, dc2, and dc3). dc1 has two routers dc11 and dc12; dc2 has dc21 and dc22; and dc3 has dc31 and dc32. The connections are that (1) two routers in the same data center are connected; (2) dc11, dc21 and dc31 are mutually connected; same for dc12, dc22, and dc32.

The network can provide different abstract topologies: for tenants in dc1, they see dc11, dc12, and dc2, dc3; same for tenants in dc2, and dc3. In other words, each tenant in a DC sees the detailed topology of its DC and the other data centers are abstracted to be single nodes.

This case turns out to be not doable for their pure hierarchical layer approach, where a top layer node/link has supporting nodes/links. Specifically, thee model cannot have cross-layer links such as dc11 -> dc2.

4. Security Considerations

This document has not conducted its security analysis.

5. IANA Considerations

This document does not specified its IANA considerations, yet.

6. Acknowledgments

The author thanks discussions with Xiao Shi, Xin Wang, Erran Li, Tianyuan Liu, Andreas Voellmy, Haibin Song, and Yan Luo.

7. References

7.1. Normative References

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

7.2. Informative References

[I-D.amante-i2rs-topology-use-cases]
Medved, J., Previdi, S., Lopez, V., and S. Amante, "Topology API Use Cases", draft-amante-i2rs-topology-use-cases-01 (work in progress), October 2013.

[I-D.clemm-i2rs-yang-network-topo]
Clemm, A., Medved, J., Tkacik, T., Varga, R., Bahadur, N., and H. Ananthakrishnan, "A YANG Data Model for Network Topologies", draft-clemm-i2rs-yang-network-topo-01 (work in progress), October 2014.

[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.

[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.

Appendix A. Graph Transformations and Operations to Build Topology Representation for Applications

In this appendix, we give a graph transformation framework to build the schema from a raw topology G(0). The network conducts transformations on G(0) to obtain other topologies, with the following objectives:

1. Simplification: G(0) may have too many details that are unnecessary for the receiving app (assume intradomain); and

2. Preservation of privacy: there are details that the receiving app should not be allowed to see; and

3. Conveying of logical structure (e.g., MPLS paths already computed); and

4. Conveying of capability constraints (the network can have limitations, e.g., it uses only shortest path routing); and

5. Allow modular composition: path from one point to another point is delegated to another app.

The transformation of G(0) is to achieve/encode the preceding. For conceptual clarity, we assume that the network uses a given set of operators. Hence, given a sequence of operations and starting from G(0), the network builds G(1), to G(2), ...

Below is a list of basic operators that the network may use to transform from G(n-1) to G(n):

o O1: Deletion of a switch/port/link from G(n-1);

o O2: Switch aggregation: a set Vs of switches are merged as one new (logical) switch, links/ports connected to switches in Vs are now connected to the new logical switch, and then all switches in Vs are deleted;

o O3: Path representation: For a given extra path from A to R1 to R2 ... to B in G(n-1), a new (logical) link A -> B is added; if the constraint is that A -> must use the path, it will be put into the Overlay;

o O4: Switch split: A switch s in G(n-1) becomes two (logical) switches s1 and s2. The links connected to s1 is a subset of the original links connected to s; so is s2.

Authors' Addresses

Greg Bernstein
Grotto Networking
Fremont, CA
USA

Email: gregb@grotto-networking.com

Young Lee
Huawei
TX
USA

Email: leeyoung@huawei.com

Wendy Roome
Alcatel-Lucent Technologies/Bell Labs
600 Mountain Ave, Rm 3B-324
Murray Hill, NJ 07974
USA

Phone: +1-908-582-7974
Email: w.roome@alcatel-lucent.com

Michael Scharf
Alcatel-Lucent Technologies
Germany

Email: michael.scharf@alcatel-lucent.com

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

Email: yry@cs.yale.edu