ALTO Working Group
INTERNET-DRAFT
Intended Status: Standard Track
Expires: Oct 27, 2015

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

October 26, 2014

Extended Endpoint Properties for Application Layer Traffic Optimization
draft-deng-alto-p2p-ext-04

Abstract

The purpose of the ALTO protocol is to provide better-than-random peer selection for P2P networks. The base ALTO protocol focuses, however, 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 endpoint 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 (EP) 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 Overview

It is expected that EP properties from the following aspects can be useful for an ALTO client to provide better user experience or avoid performance degradation due to the ignorance of these EP specific information:

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

2.1 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:

Firstly, (usefulness) there should be a clear motivation and application scenarios that justify the necessity and value for providing such information via EP property enquiry.

Secondly, (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.

Thirdly, (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.

2.2 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.

2.2.1 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:

- the string value of its "name" attribute;
- the value set of its "precision" attribute; and
- 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.

2.2.2 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.

2.3 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.

2.4 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 endpoint.

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, and 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 the subscription information etc.

3 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].

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 Endpoint Property Type: participating_role

Different types of endpoints 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 endpoint property to support this indication.

In general, the endpoints 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 endpoint’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 Endpoint Property Type: battery_limited

Another important endpoint 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 endpoint, or "false" for an electricity supplied endpoint or for an endpoint with an unknown power supply type.

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

4.2.3 Endpoint 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 endpoint 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 Endpoint Property Type: network_access

One important endpoint 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 as long as its implications on the endpoint’s provisioned bandwidth etc. 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 endpoint property. The value for this property can be enumerated as "adsl", "ftth", "fttb", "dc", and etc.

In case that the endpoint 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 endpoints, gives the network operator a chance to say, which endpoint'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 endpoint in question, in terms of its access network. The endpoint with a higher number is more preferable to another endpoint 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 Endpoint Property Type: forwarding_class

As suggested for the NFV use-case, the endpoint property "forwarding_class" is meant to indicate the type of forwarding class the endpoint 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 Endpoint 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 endpoint 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 wrt 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 endpoint 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 Endpoint Property Type: provisioned_bandwidth

For applications seeking for a candidate peer for uploading services, the endpoint's uploading bandwidth is essential for the selection.

While it is straightforward for one to expose the accurate information over an endpoint's bandwidth capability, the subscriber of the endpoint 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 endpoint's provisioned bandwidth, in order to do better load balancing within the network by avoiding undesirable hot spots caused by competition from applications for the handful most provisioned endpoints.

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 endpoint 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 endpoint is "ranking", the following "content" is filled with the relative ranking of the endpoint'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 endpoint property types to the existing registry created by ALTO protocol [RFC7285].

TBA.

7 Acknowledgements

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

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.

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

8.2 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.

Authors' Addresses

Lingli Deng
China Mobile
China

Email: denglingli@chinamobile.com

Haibin Song
Huawei
China

Email: haibin.song@huawei.com

Sebastian Kiesel
University of Stuttgart, Computing Center
Germany

Email: ietf-alto@skiesel.de

Richard Yang
Y. Richard Yang
Yale University

Email: yry@cs.yale.edu

Qin Wu
Huawei
China

Email: sunseawq@huawei.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 30, 2015

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

October 27, 2014

ALTO Cost Calendar
draft-randriamasy-alto-cost-calendar-02

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 specifies a new Cost Mode, "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. 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.

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.

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.

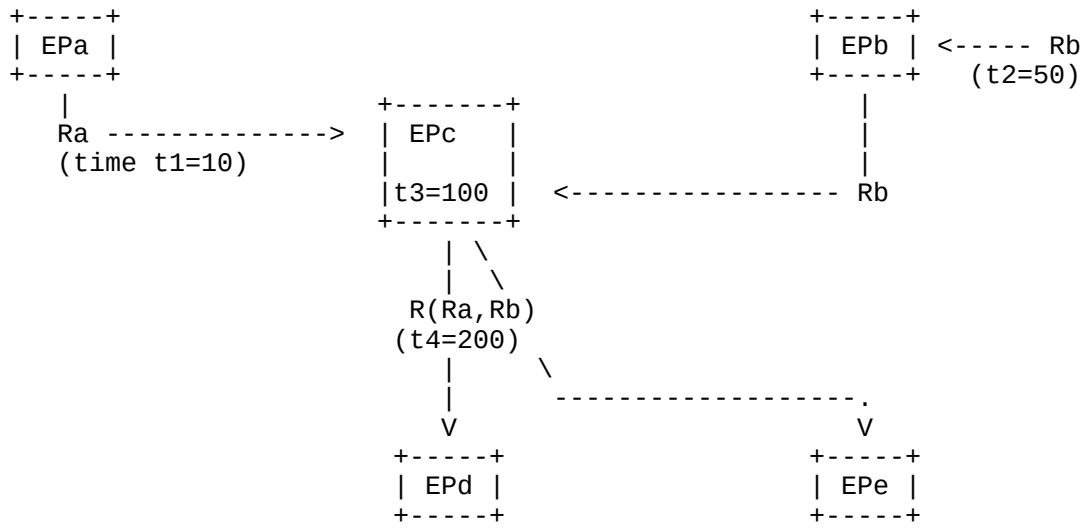
As a result, it would be very helpful to let these ICPs to have a good knowledge about the link utilization patterns between the different datacenters from the networks before making a more intelligent scheduling decision. While this usage data today already is gathered and also used for the scheduling 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.



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:

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

3.1. Purpose of an ALTO calendar

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

- the Calendar values are assumed to be stationary on each time interval,
- the ALTO Server may provide values on past time periods that can be interpreted as historical experience and used to anticipate future cost values,
- 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.

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.

- 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.
- numb-intervals:
  - the integer number of values of the cost calendar array, at least equal to 1.
- calendar-update-frequency:
  - the frequency at which ALTO Calendar values are updated in the ALTO Server. Must be not sooner than ’next-calendar-start-time’. The value is expressed in the same format as for ’time-interval-size’.
- numb-calendars: (OPTIONAL)
  - the number of calendars with deffering values, of duration ’time-interval-size’ multiplied by ’numb-intervals’ available in the ALTO Server during the period equal to ’calendar-update-frequency’.

- 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 'calendar-update-frequency' indicates the frequency at which the calendar values are computed and made available to the ALTO Client. The value is not necessarily constant and may change in given periods when particular events and related usage patterns occur. The ALTO Client should just consider that the ALTO Server does not find it necessary to update the Calendar values more often than indicated by the current value. The ALTO Client may thus assume that the ALTO Server considers the values as valid or stationary during this period.

- Attribute 'max-numb-calendars': this optional attribute stresses the fact that an ALTO Server may for example offer daily cost patterns conveying a number 'num-intervals' of 12 values valid on a time interval of size 'time-interval-size' equal to 2 hours. The calendars are updated at an 'calendar-update-frequency' of 7 days. However for example the ALTO Server may identify in some period a common daily pattern C1 for Monday, Tuesday, and Friday, another one C2 for Saturday and Sunday and a specific one C3 for Wednesday for instance due to maintenance or due to an important holiday. In this case, the number of different daily patterns thus calendars is equal to 3.

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

- The Calendar for 'TEpktloss'': is an hourly pattern that consists of 12 values provided each on a time interval of 5 minutes, provided for each hour, and the 24 calendars are updated every day at 0:00 GMT.

- 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. It is computed every day and updated at 0:00 GMT. 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 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. The daily patterns are updated every week on sunday at 23:59 GMT. An ALTO Client can thus store and use the needed routingcost calendars for maximum 1 week. 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.

4.3.1. Example IRD with ALTO cost Calendars

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

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

- "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,
                  "calendar-update-frequency" : "1 day"
              }
           },
           "calendar-bw": {
              "cost-mode"  : "calendar",
              "cost-metric": "Availbandwidth",
              "description": {
                  "time-interval-size" : "2 hour",
                  "numb-intervals" : 12,
                  "calendar-update-frequency" : "1 day"
              }
```

```
          },
          "calendar-routing": {
             "cost-mode"  : "calendar",
             "cost-metric": "routingcost",
             "description": {
                 "time-interval-size" : "1 hour",
                 "numb-intervals" : 24,
                 "calendar-update-frequency" : "1 week",
                 "num-calendars" : 3
        }
        ... 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

- calendar-start-time:
  - the date corresponding to the first value in the calendar values array
- next-calendar-start-time:
  - the starting date of the next calendar. To limit the number of provided calendars and schedule the next ALTO Calendar query. For example, a daily calendar may have the same values for the next 3 days. So the ALTO Client does not need to get one every day.
- last-calendar-update:
  - the last date at which ALTO Calendar values were updated in the ALTO Server. Must be no later than 'calendar-start-time'.
- 'time-interval-size': as specified for the IRD
- 'numb-intervals': as specified 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.

- Attribute 'next-calendar-start-time' is useful for clients to schedule their requests for calendars. For example: an ALTO Server may provide calendars lasting 24 hours and decide to estimate and update these daily patterns every week on Sunday. Suppose that in the coming week, the patterns are the same for Monday through Tursday. Then the calendar values will only change on Friday. It the ALTO Client gets a calendar on Tuesday, it may keep it until Thursday included and get another one the on the next Friday if needed. An example illustrating the usefulness of 'next-calendar-start-time' is provided in following sections.

- Attributes ’last-calendar-update’ indicates when the calendar values are last updated and uploaded in the ALTO Server. This attribute reflects the age of the ALTO Calendar information. Given the update frequency indicated in the IRD by the ’calendar-update-frequency’ attribute, the ALTO Client can figure when the next value update will occur.

DISCUSSION: can the ALTO responses omit repeating the values of ’time-interval-size’ and ’numb-intervals’, as they are already in the IRD?

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 ’calendar-start-time’ and ’next-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,
      "next-start-time" : Fri, 4 Jul 2014 00:00:00 GMT,
      "last-calendar-update" : Sun, 29 Jun 2014 00:00:00 GMT,
      "time-interval-size" : "1 hour",
      "numb-intervals" : 24
    },
    "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]
        }
    }
  }
```

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,
      "next-start-time" : Wed, 2 Jul 2014 00:00:00 GMT ,
      "last-calendar-update" : Tue, 1 Jul 2014 00:00:00 GMT,
      "time-interval-size" : "2 hour",
      "numb-intervals" : 12
    },
    "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. 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],

5.1. 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.

5.2. 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.

6. Acknowledgements

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

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.

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

7.2. 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: ietf@nico-schwan.de

ALTO Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 30, 2015

U. Rauschenbach
Nokia Networks

October 27, 2014

ALTO in wireless access networks
draft-rauschenbach-alto-wireless-access-00

Abstract

The Application-Layer Traffic Optimization (ALTO) specification defines the concept of Provider-defined Identifiers (PIDs) as the aggregation of network endpoints for network nodes. Each PID can be associated with a set of endpoint addresses, e.g. aggregating endpoints that use the same network access point.

This document focuses on mobile networks and introduces proposes the toidea of using use cells of cellular access networks as an aggregation points. This allows applications to make decisions based on the path cost of using the current cell as a network attachment point, or to even choose which network access points network attachment point to select. Use cases are described which can benefit from this. The draft elaborates on possible ALTO modifications enabling such use cases. The intent of the draft is to start discussion on the topic.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on April 30, 2015.

Copyright Notice

described in the Simplified BSD License.

Table of Contents

1. Introduction

The Application-Layer Traffic Optimization (ALTO) [RFC7285] specification allows providing information about a network to support applications in making decisions regarding the most optimal use of the network. Originally geared towards supporting peer-to-peer applications in IP networks, the ALTO WG has recently broadened its scope.

ALTO defines the concept of Provider-defined Identifiers (PIDs) to aggregate network endpoints. Each PID can be associated with a set of endpoint addresses. The concept of PIDs is generic and potentially allows many types of endpoint addresses - the ALTO specification mentions IP addresses, overlay IDs and MAC addresses. However, RFC 7285 only defines address types for IPv4 and IPv6 addresses and leaves it to additional documents to define additional types.

This document introduces a number of use cases occurring in wireless networks which benefit from associating a cell or access point with a PID. In such scenarios, the cell or access point through which network attachment occurs or may occur in the future needs to be identifiable by the ALTO client. Wireless access points provide information such as cell ID to identify them. However, currently ALTO provides only IP addresses for endpoints, and arbitrary strings as PID values. Allowing an ALTO map to provide information about an actual cell or access point can benefit use cases when the terminal needs metrics describing the path cost of using a particular cell or access point before the terminal attaches to the network via that cell or access point, as well as for using a cell ID to aggregate information about all endpoints attached to that cell or access point.

2. Problem statement

ALTO defines the concept of PIDs which can be used to aggregate network nodes. A PID can aggregate e.g. cover all nodes connected to the network via a particular POP. It is assumed that in fixed networks, the actual POP via which an endpoint is connected to the network rarely changes. In cellular wireless networks, the situation is different. Network access happens through cells. Nodes frequently change cells when they move. Traffic conditions and other costs may vary widely from cell to cell. Therefore, it makes sense to include information about actual cells through which network attachment happens in the ALTO maps. However, currently this is not possible in the ALTO data entities. Similarly, access to WiFi networks happens through wireless access points which are currently also not considered in ALTO.

3. Use cases

3.1. Cell as aggregation point

Wireless terminals connect to a cellular network through different cells, and to Wi-Fi networks through different wireless access points. The quality of network access may vary greatly from connection point to connection point, e.g. due to congestion or the differing capacity of backhaul (connection point is used to refer to both cells and access points from now on). In particular in cellular networks, the connection point actually used frequently changes.

If the ALTO information model would be able to identify the connection point with a PID, then different costs could be provided for different map paths to the connection points. Also, there may potentially be a large number of cellular subscribers connected to a typical cell. It may not matter which endpoint addresses they use, but only via which connection point they are attached to the network. Exploiting this may greatly reduce map size as numerous endpoints under a connection point can be aggregated into a PID, and also reduce the number of map updates as endpoints move so that they can be attached via different connection points due to endpoint movement or handovers.

3.2. Passive reaction to handovers

Applications can benefit from knowledge or assumption about the cells a terminal is connected to or will with some probability be connected to in the near future.

Applications may know which cells a terminal typically connects throughout a day e.g. from observing daily commute patterns. They may also be provided candidate cells with favorable conditions by including in the ALTO response a set of non-congested cells in the vicinity of a queried cell. Once the terminal connects to such a cell while moving, the application can execute tasks which require e.g. higher throughput.

Also, if non-congested and congested cells overlap (such as e.g. a non-congested small cell or WiFi hotspot and a congested macro cell) and the terminal is in motion, an application can make assumptions about future congestion and prepare / react accordingly.

Finally, applications may obtain candidate cells by accessing information about neighboring cells through the APIs provided by modern smartphones.

Based on such knowledge, the following two classes of use cases can be distinguished.

3.2.1. Cost calendar to extend battery life for background tasks

Large fractions of Internet traffic today occur over wireless systems. Not all traffic is real-time and many tasks can be performed in the background. In particular, clients that download large volumes of data (e.g., mail applications or social network clients) can choose the time when they will actually download large items such as attachments or video clips.

If a cost calendar does include information that a client can use to compute the achievable throughput of a service via different cells (e.g. based on information about available bandwidth or congestion information per cell), it can initiate or schedule large downloads to take place when it can connect to a cell that currently allows high throughput.

If a high throughput cell is used for background downloads, the battery life can be extended as the terminal spends less time in

battery-consuming states with the processor busy and the radio on and instead spends more time in battery-conserving idle modes.

3.2.2. Cost calendar to optimize application sessions

Cost and other metrics usually fluctuate over time e.g. based on the day of week. Such knowledge can be used to provide hints to applications how to adapt proactively. For instance, during (or prior to) user mobility (e.g. a commute), an application may make use of such heuristic information to prefetch farther in advance (beyond just in time) more items within an ongoing streaming session at times of low congestion, prior to it entering time intervals or cells with higher congestion etc. Such knowledge can also contribute to picking a suitable video rate in progressive streaming. In a video telephony session, applications may proactively switch to audio-only calls in congested areas. This use case is somewhat related to the previous one; however, it focuses on optimizing the user experience of foreground tasks, rather than on scheduling background tasks.

3.3. Connection management and traffic offload

The previouis use cases have focussed on the application reacting on a change in network access point. However, often various connection points of different wireless networks are available to which an endpoint may actively attach. As mobile endpoints get more intelligent, functions for connection management and traffic offload can consider not only the quality of the radio connection when doing access selection and traffic offload, but also other properties such as cost parameters and additional metrics provided by ALTO. Also, when multiple different wireless access options are available, traffic may be distributed between these options based on ALTO cost maps so the network can be used more optimally. However, this requires that the wireless terminal can associate ALTO map entries with actual cell identifiers in the wireless networks.

4. Requirements and solution ideas

This section elaborates solution ideas as a starting point for the technical discussion.

Aggregation in ALTO is modeled by a PID which represents a group of nodes. A PID is identified by a provider-defined string with no general meaning, and an endpoint is identified by its IP address.

Cellular networks use a cell identifier (cell ID) to identify the cells through which network attachment happens. In Public Land Mobile Networks (PLMNs) based on E-UTRAN [TS36.311], the Cell ID is composed of the PLMN ID and the ECI, which in turn are composite identifiers. The PLMN ID (Public Land Mobile Network IDentifier) identifies a wireless communications system. It consists of the 3 digit Mobile Country Code (MCC) and the 3 digit Mobile Network Code (MNC). ECI is the E-UTRAN Cell Identifier which identifies a Cell within a PLMN. An ECI is composed of eNB ID and Cell ID. The eNB ID (20 bits) is the eNodeB IDentifier which identifies an eNB within a PLMN. The Cell ID (8 bits) identifies a (sub)cell within a particular eNodeB.

WiFi-based wireless networks provide data such as SSID, access point MAC address and other values that allow to identify the network access point. Note that the Hotspot 2.0 specification [HS2.0] allows a terminal querying a number of parameters prior to connecting to an access point (such as Venue Name, Roaming Consortium, IP Address Type Availability, 3GPP Cellular Network, Domain Name, Hotspot Query List, Hotspot Capability List and others). The terminal searching for a particular network access can retrieve the information elements through the IEEE 802.11 ANQP from the access point prior to connection establishment. However, once connected to an access

point, the terminal has to retrieve the information of other access points in the area by virtually detaching from the access point and running access network discovery over the air to the other access points. It would be beneficial when the terminal would be able to retrieve the information of adjacent access points and access networks by way of e.g. an ALTO query over the existing connection.

In order to enable the use cases defined above in ALTO, the IDs which identify the wireless connection point would have to be added to the ALTO data model.

To support the use cases above, two things are required:

1. To enable querying properties of a certain cell such as bandwidth or degree of congestion. Such queries could be supported through the mechanism of endpoint property queries (RFC7285 section 11.4). Possible solutions are:

   * Solution A: The cellular access point would need to be defined as an additional endpoint inside the group aggregated by a PID, i.e. a new address type that represents a cell ID would be required.

   * Solution B: A property query would be needed for the PID, plus a property that represents the cell identifier.

2. To enable cells as identifiable aggregation points, the notion of a PID (which is now just a string) would need to be extended by including cell identification. Assigning different PIDs to different cells would then allow to create different cost map sections for paths of access through different cells (e.g., costs would be higher for access through congested cells than for access through non-congested cells).

   * Solution: A PID would need to be extended by a cell identifier (related to Solution B above)

5. References

[HS2.0] The WiFi Alliance, "Hotspot 2.0 (Release 2) Technical Specification, Version 1.0.0", August 2014.

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

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

[TS36.311] Third Generation Partnership Project, "3GPP TS 36.331 V12: Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol specification (Release 12)", September 2014.

Author’s Address

Uwe Rauschenbach
Nokia Networks

Email: uwe.rauschenbach@nsn.com

ALTO WG
Internet-Draft
Intended status: Standards Track
Expires: April 30, 2015

W. Roome
Alcatel-Lucent
X. Shi
Y. Yang
Yale University
October 27, 2014

ALTO Incremental Updates Using Server-Sent Events (SSE)
draft-roome-alto-incr-update-sse-00

Abstract

The goal of Application-Layer Traffic Optimization (ALTO) [RFC7285] is to bridge the gap between network and applications by providing network related information to non-priviledged, application-level clients. This allows applications to make informed decisions, for example when selecting a target host from a set of candidates.

Therefore an ALTO Server provides network and cost maps to its clients. However, those maps can be very large, and portions of those maps may change frequently (cost maps in particular).

This draft presents a method to provide incremental updates for these maps. The goal is to reduce the load on the ALTO Client and Server by transmitting just the updated portions of those maps.

Requirements Language

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

Status of this Memo

This Internet-Draft will expire on April 30, 2015.

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. An ALTO Server provides network and cost maps, which may be very large and change very frequently.

Instead of having the clients request for a new complete network map or cost map every time, an incremental update from the server is much more efficient. The goals are to reduce the load on the ALTO Client and Server by efficiently transmitting only the updated portions of those maps, and to provide timely updates to clients.

This draft uses the JSON Merge Patch message format [RFC7386] to encode the incremental update messages for network maps and cost maps, and uses Server-Sent Events (SSE) as the transport mechanism to deliver those updates to clients.

2. Incremental Update Message Format

2.1. JSON Merge Patch

[RFC7386] defines JSON Merge Patch format and transport, which enables applications to update the server resources via the PATCH method [RFC5789] of HTTP. This draft adopts the format of the Merge Patch messages to encode our incremental updates objects, but uses a different transport mechanism.

The process of applying a Merge Patch is defined by the following 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 remove the pair with "name" in the original JSON document.

2.2. 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-default-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-default-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" ]
    }
  }
}
```

2.3. 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-default-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 that cost map so that (1) PID1->PID2 is 9 instead of 5; (2) PID3->PID1 is no longer available; and (3) PID3->PID3 is now 1:

```
{
  "cost-map" : {
    "PID1" : { "PID2" : 9 },
    "PID3" : { "PID1" : null, "PID3" : 1 }
  }
}
```

Here is the updated Cost Map:

```
{
  "meta" : {
    "dependent-vtags" : [
      {"resource-id": "my-default-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": { "PID1": 20,             "PID3": 1  }
  }
}
```

3. Server-Sent Events

3.1. Overview of SSEs

Server-Sent Events [SSE] enable a server to send new data to a client by pushing messages to the client. To summarize the protocol, the client establishes an HTTP 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 of the full specification), and contain three fields: an event type, an id, and data. All fields are strings. The data field may contain new-lines; the other fields cannot. The event type and id fields are optional.

Here is a sample SSE stream, starting with the client request. The server sends three events and then closes the stream.

```
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
data: hello there

event: middle
data: let's chat some more ... and more ...

event: end
data: good bye
```

3.2. ALTO SSE Update Messages

In our events, the data field is a JSON object. There two types of data objects. One is a message describing an ALTO resource, such as a Network Maps or Cost Map, as defined in [RFC7285]. We will refer to these as full-map messages. The other type is a Merge Patch message to apply to an ALTO resource.

Our event types have two sub-fields: the media-type of the JSON message in the data field, and the resource-id of the ALTO resource. The media-types for ALTO resource 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 Merge Patch message is "application/merge-patch+json", and is defined by [RFC7285].

We do not use the SSE id field.

Because commas (character code 0x2c) are not allowed in media-type names, we encode the event type sub-fields as

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

Here examples of ALTO update events:

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

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

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

3.3. Keep-Alive Messages

An SSE event with an empty event type 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. Update Stream Service

An Update Stream Service returns a stream of SSE messages, as defined in Section 3.2.

4.1. Media Type

The media type of an ALTO Update Stream resource is "text/event-stream".

4.2. HTTP Method

An ALTO Update Stream resource is requested using the HTTP GET method.

4.3. Accept Input Parameters

None.

4.4. Capabilities

The capabilities are defined by an object of type UpdateStreamCapabilities:

```
object {
  JSONString events<1..*>;
} UpdateEventStreamCapabilities;
```

The strings in the array are the event types (see Section 3.2) sent by this Update Stream.

If an Update Event Service’s event capability list has an event with

a media-type of "text/merge-patch+json" for a resource-id, then the event capability list MUST also have an full-map event for that resource-id. For example, suppose "my-costmap" is the resource-id of a Cost Map. Then if the event list has "text/merge-patch+json,my-costmap", it MUST also have the event "application/alto-costmap+json,my-costmap".

4.5. Uses

An array with the resource-ids of the resources for which this stream sends updates. This array MUST contain the resource-ids of every event type in the "events" capability.

4.6. Event Order Requirements

There are several requirements on the order in which an ALTO Server sends SSE Update messages on the event stream:

- For any given resource-id, the ALTO Server MUST send a full-map update event (media-type "application/alto-networkmap+json" or "application/alto-costmap+json") before the first Merge Patch event (media-type "application/merge-patch+json") for that resource-id.

- The ALTO Server SHOULD send full-map update events for all resource-ids covered by this Update Stream resource as soon as possible after the client initiates the connection.

- If the event list contains a resource-id R0 on which resource-id R1 depends, when R0 changes, the ALTO Server MUST send the update for R0 before sending the update for R1. For example, suppose the event list includes a Network Map resource and its dependent Cost Map resources. When the Network Map changes, the ALTO Server MUST send an update event for that Network Map before sending the update events for the dependent Cost Maps.

- If the event list contains a resource-id R0 on which resource-id R1 depends, 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 all dependent Cost Maps as soon as possible after the update event for the Network Map.

4.7. Response

Here is an example of a client’s request and the server’s immediate response, using the Update Stream resource "my-routingcost-update-stream" defined in the IRD in Section 6. This assumes the Update

Stream service sends updates for a Network Map with resource-id "my-network-map" and an associated Cost Map with resource-id "my-routingcost-map":

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

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

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

event: application/alto-costmap+json,my-routingcost-map
data: { ... full 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: {"resource-id":"my-routingcost-map",
    "media-type":"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: application/alto-networkmap+json,my-network-map
data: { ... full Network Map message ... }

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

4.8. Client Actions When Receiving Update Messages

In general, when a client receives a full-map 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 when the resource on which it depends changes. There are at least two ways a client may 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 to 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. If the client uses the Filtered Update Stream service, 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.

5. Filtered Update Stream Service

The Filtered Update Stream service is similar to the Update Stream service, except that the client can select the types of update events. Specifically, except as noted below, the Filtered Update Stream service is identical to the Update Stream service (Section 4).

5.1. HTTP Method

A Filtered ALTO Update Stream resource is requested using the HTTP POST method.

5.2. Accept Input Parameters

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

```
object {
  [UpdateEventType  events<1..*>;]
  [VersionTag       vtags<1..*>;]
  [ResourceInputs   inputs<1..*>;]
} ReqFilteredUpdateStream;

object-map {
  ResourceID -> JSONObject;
} ResourceInputs;
```

The "events" field gives the types of the events the ALTO Client wishes to receive. These events MUST be a subset of the "events" capability of this resource. If the "events" list is omitted, the ALTO Server MUST send all event types in the "events" capability of this resource.

The "vtags" field gives the version tags, as defined in Section 10.3 of [RFC7285], for any resources which the client already has. If those versions are still current, the server SHOULD NOT send the full version of that resource at startup.

The "inputs" field gives the client input needed for any POST-mode resources requested by the client. The value is a JSON object; the key is the resource-id of the POST-mode resource, and the value is the JSON object that it requires as "accepts" input.

If a client requests Merge Patch update events for a given resource-id, the client MUST also request the corresponding full map update events for that resource-id.

If a client requests the full-map update event for given resource-id, but does not request the Merge Patch update event for that resource-id, then the ALTO Server MUST send full-map update events whenever the map changes. For Network Map resources, the ALTO Server SHOULD send the full map as soon as it would have sent the Merge Patch event. For Cost Map and other resources, the ALTO Server MAY delay sending the full-map until more changes are available.

5.3. Response

Here is an example of a client’s request and the server’s immediate response, using the Filtered Update Stream resource "my-allresources-update-stream" defined in the IRD in Section 6. The client requests updates for the Network Map and the "routingcost" Cost Map, but does not want updates for the "hopcount" Cost Map. The "vtags" field gives the client’s version of the Network Map. Because that version is still current, the server does not send the full Network Map update event at the beginning of the stream:

```
POST /updates/allresources HTTP/1.1
Host: alto.example.com
Accept: text/event-stream
Content-Type: application/alto-updatestreamfilter+json
Content-Length: ###

{ "events": [
    "application/alto-networkmap+json,my-network-map",
    "application/alto-costmap+json,my-routingcost-map",
    "application/merge-patch+json,my-routingcost-map"
    ],
  "vtags": [
    "resource-id": "my-network-map", "tag": "314159265359"}
  ]
}


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

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

After that, the ALTO Server sends updates for the Network Map and "routingcost" Cost Map as they become available.

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-map 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/allresources HTTP/1.1
Host: alto.example.com
Accept: text/event-stream
Content-Type: application/alto-updatestreamfilter+json
Content-Length: ###

{ "events": [
    "application/alto-endpointprop+json,my-properties",
    "application/merge-patch+json,my-properties"
    ],
  "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: application/alto-endpointprop+json,my-properties
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:  } }

event: text/merge-patch+json,my-properties
data: { "endpoint-properties":
data:   {"ipv4:1.0.0.1" : {"priv:ietf-bandwidth": "3"}}
data: }

event: text/merge-patch+json,my-properties
data: { "endpoint-properties":
data:   {"ipv4:1.0.0.3" : {"priv:ietf-bandwidth": "38"}}
data: }
```

6. IRD Example

Here is an example of an IRD that offers both regular and Filtered Update Stream services. The unfiltered Update Stream provides updates for the Network Map and "routingcost" Cost Map. The Filtered Update Stream provides update to both those maps, plus the "hopcount" Cost Map and 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-routingcost-update-stream": {
    "uri": "http://alto.example.com/updates/routingcost",
    "media-type": "text/event-stream",
    "uses": ["my-network-map", "my-routingcost-map"],
    "capabilities": {
      "events": [
        "application/alto-networkmap+json,my-network-map",
        "application/alto-costmap+json,my-routingcost-map",
        "application/merge-patch+json,my-routingcost-map"
      ]
    }
  },
```

```
    "my-allresources-update-stream": {
      "uri": "http://alto.example.com/updates/allresources",
      "media-type": "text/event-stream",
      "uses": [
         "my-network-map",
         "my-routingcost-map",
         "my-hopcount-map",
         "my-properties"
      ],
      "accepts": "application/alto-updatestreamfilter+json",
      "capabilities": {
        "events": [
          "application/alto-networkmap+json,my-network-map",
          "application/alto-costmap+json,my-routingcost-map",
          "application/merge-patch+json,my-routingcost-map"
          "application/alto-costmap+json,my-hopcount-map",
          "application/merge-patch+json,my-hopcount-map"
          "application/alto-endpointprops+json,my-properties",
          "application/merge-patch+json,my-properties"
        ]
      }
    }
```

7. Design Decisions and Discussion

7.1. 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 a lot of complication. The server would have to 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's last id is bogus, then the server will have to resend the complete maps anyway.

In short, using event ids to avoid resending the full map adds a lot of complication to avoid a situation which is hopefully very rare. Hence we decided to keep it simple.

The Filtered Update Stream service does allow the client to specify the vtag of the last received Network Map, and if that is still current, the server can avoid retransmitting the Network Map.

7.2. Is Incremental Update Useful for Network Maps?

It is not clear whether incremental update (that is, Merge Patch update) is useful for Network Maps. For minor changes, such as moving a prefix from one PID to another, it might 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.

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. But we expect most Network Map updates will be full updates with full Network Map message, rather than incremental Merge Patch updates.

Note that while we allow a server to use Merge Patch on Network Maps, we do not require the server to do so.

8. Security Considerations

Allowing persistent update stream connections does enable 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.

9. IANA Considerations

This draft defines a new media-type, "application/ alto-updatestreamfilter+json", as described in Section 5.2. That type must be registered with IANA.

All other media-types used in this document have already been registered, either for ALTO or JSON Merge Patch.

10. References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate

Requirement Levels", RFC 2119, BCP 14, March 1997.

[RFC5789] Dusseault, L. and J. Snell, "PATCH Method for HTTP", RFC 5789, March 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.

[RFC7386] Hoffman, P. and J. Snell, "JSON Merge Patch", RFC 7386, October 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
Yale University
51 Prospect St
New Haven CT
USA

Email: yang.r.yang@gmail.com

ALTO WG
Internet-Draft
Intended status: Standards Track
Expires: January 4, 2015

W. Roome
Alcatel-Lucent
Y. Yang
Yale

July 3, 2014

PID Property Extension for ALTO Protocol
draft-roome-alto-pid-properties-02

Abstract

This document extends the Application-Layer Traffic Optimization (ALTO) Protocol [I-D.ietf-alto-protocol] by defining PID-based properties in much the same way that the original ALTO Protocol defines endpoint-based properties.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 4, 2015.

Copyright Notice

Table of Contents

1. Introduction

A key abstraction introduced by the ALTO Protocol [I-D.ietf-alto-protocol] is PIDs (Provider-defined Identifiers), where each PID is defined as a name and a set of associated endpoint addresses. For IPv4/IPv6 networks, a PID's address set is defined by one or more endpoint address prefixes called CIDRs [RFC.4632]. This extension focuses on IPv4/IPv6 networks.

An ALTO Server uses PIDs when defining one or more Network Maps, each of which is defined by a set of PIDs. Each Network Map defines a logical partition of a network address space, where similar endpoints are grouped in the same PID, specified by the addresses contained in the definition of the PID. An ALTO Server may publish multiple Network Maps when there are multiple ways to partition networks. For example, one Network Map may partition endpoints according to geographical locations, and hence each PID defined in the Network Map represents the set of endpoints at a given location. Another Network Map may partition endpoints according to the capabilities (e.g., CDN delivery protocols such as HTTP or HTTPS) that the network can provide. In this case, each PID defined in the Network Map represents the endpoints with similar capabilities.

A major missing component of the base ALTO Protocol is that the common properties are not specified. In particular, in the base ALTO Protocol, each PID has only a name and a set of endpoint addresses. The objective of this document is to allow PIDs to have properties. Example PID properties include "country code", "continent code", "ISP", "lat/long bounding box", "endpoint type" (server farm, end users, cell data connections, etc). We identify use cases (e.g., VPN

selection and CDN Capability Advertisement) where PID properties can provide value.

2. The Consistency and Inheritance Design Views

When we define PID properties, we follow a key consistency design guideline that PID properties should be consistent with and generalize the endpoint properties already defined in the base ALTO Protocol. Specifically, in the base ALTO Protocol, for each selected endpoint address, there can be a set of (prop-type, value) pairs associated with the endpoint address. These are called the endpoint properties of the selected endpoint. The ALTO Protocol allows an ALTO Client to obtain defined endpoint properties.

Consider a given endpoint property p and all endpoints defined in a PID named pid1. If all of the endpoints have the same value v for p, then it is natural and consistent that when we define the value for p, as a PID property, the value should be v. For the more general case, let ip1.p denote the value of property p for endpoint ip1. Assume that pid1 consists of a set of n IP addresses, ip1, ip2, ..., ipn. Let pid1.p denote the value of property p for pid1. Then we can consider that pid1.p is from an aggregation function of ip1.p, ip2.p, ..., ipn.p. Example aggregation functions include average/ mean, mode, geo-center, union, bounding box, where meaningful aggregations depend on the specific property p.

Complementing the bottom-up aggregation view, we also adopt a top-down inheritance view, by considering that when ip1 is in pid1, ip1.p inherits the value of pid1.p, if the value of ip1.p is not defined; otherwise, ip1.p overrides the value of pid1.p. The concept of inheritance is a simple, but powerful concept to reduce information redundancy.

3. A Hierarchical View of a Network Map

3.1. Default Containment Hierarchy

A Network Map defined in the base ALTO Protocol can be considered as a default three-level hierarchy: with the highest (1st) level being a root, the next (2nd) level being the PIDs, and the lowest (3rd or leaf) level being the individual endpoint addresses. An issue that the base ALTO Protocol needs to resolve is that PID definitions can overlap, and hence we must determine the PID to which an endpoint address belongs. For example, consider a Network Map with two PIDs: PID1 is 10.0.0.0/8, and PID2 is 10.0.1.0/24. Then all addresses in PID2 are also in PID1. The base ALTO Protocol requires that an endpoint address be in one, and only one, PID, among the set of PIDs defined in the same Network Map. ALTO achieves this by specifying

that if an address matches several CIDR, the address is in the PID with the CIDR with the longest prefix. We refer to this PID as the home PID of the endpoint. Thus, for the example, 10.0.1.5 is in PID2, and 10.0.2.6 in in PID1.

3.2. Extension: Implicit Inheritance Via Nested PIDs

We would like to use the PID hierarchy to inherit property values. That is, if all endpoints in px, py and pz are also in pa, then unless otherwise overridden, PIDs px, py and pz should inherit all properties defined in PID pa.

Unfortunately overlapping PID definitions result in the usual issues with multiple inheritance. Consider the following example:

```
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]
```

All endpoints in p2a and p2b are also in p1, so those two PIDs should inherit any properties defined in p1. However, the endpoints in p3 are split between p2a and p2b, so p3 cannot simply inherit values from p2a or p2b. On the other hand, all endpoints in p3 are in p1, so we would expect p3 to inherit any properties defined in p1 that are not overridden in p2a or p2b.

Hence we will define inheritance as follows.

Definition: The immediate parent of CIDR C is the CIDR C’ with the longest prefix of all CIDRs, in the set of all CIDRs in all PIDs in the Network Map, which contain all endpoints in C. 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 C’ defines the value V for property PR, or if its immediate parent CIDR C’ inherits the value V for property P.

Definition: A PID P has the value V for property PR if that value is explicitly defined for PR in P, or if all CIDRs C in P inherit the same value V for property PR.

Suppose the following properties are defined for PIDs described above:

```
PID p1:  ISP="Verizon" country-code="us"
PID p2a: state-code="NJ"
PID p2b: state-code="NY"
```

Then p2a, p2b, and p3 would all inherit the ISP and country-code properties from p1. However, p3 would not inherit the state-code property, because it has different values in p2a and p2b.

4. Services

In the interests of simplicity, we will give an overview of the proposed services, rather than detailed descriptions.

4.1. PID Properties Announcement

Given the consistency and inheritance design guideline, we require that PID Properties and Endpoint Properties use the same property name space. Such property names must be registered with IANA.

To allow an ALTO Client to know the set of PID Properties associated with a PID Property Resource, we use the same approach as that of endpoint properties: announcement in IRD. An example is shown below.

```
...
"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" ]
      },
   },
   "my-pid-property" : {
      "uri" : "http://alto.example.com/pidprop/netmap1/pidp1",
      "media-type" : "application/alto-pidprop+json",
      "uses" : ["my-default-network-map" ]
      "capabilities" : {
        "prop-types" : [ "country-code",
                         "asn" ]
      },
   }
}
```

4.2. Full PID Property Map Service

Analogous to ALTO's Full Cost Map Service, a Full PID Map Service returns properties defined for all PIDs in a Network Map.

This is a GET request. The response message is similar to that of ALTO's Endpoint Property Service, but with PID names instead of endpoint addresses. The IRD entry for the service defines a "prop-types" capability with the names of the properties that this service returns, and specifies a "uses" attribute for the Network Map defining the PIDs.

In the interests of limiting the response message size, the Full PID Property Map Service would NOT enumerate inherited property values. Thus if PID1 defines PROP1, and if PID2 is contained within PID1 and does not override the value for PROP1, then the response message gives a value for PROP1 in PID1, but not in PID2. In this case the client is expected to deduce the inheritance. That is feasible because the client has all information needed to do that.

4.3. Filtered PID Property Map Service

Analogous to ALTO's Filtered Cost Map Service, a Filtered PID Map Service returns a subset of the Full PID Property Map. The client specifies the desired property and PID names.

This is a POST request. The response message is the same as for the Full PID Property Map Service. The request message is similar to the request message for ALTO's Endpoint Property Service, except with PID names instead of endpoint addresses. The IRD entry for the service defines a "prop-types" capability with the names of the properties this service returns, and specifies a "uses" attribute for the Network Map defining the PIDs.

Unlike the Full Filtered PID Property Service, the Filtered PID Property Service would explicitly enumerate inherited property values. Thus if PID1 defines PROP1, and if PID2 is contained within PID1 and does not override the value for PROP1, then the response message includes PID1's value for PROP1 in PID2's properties. This is necessary because the Filtered PID Property Map response does not give the client enough information to deduce the inherited properties. For consistency, the Filtered PID Property Service would enumerate inherited properties for a PID even if the client also requested properties for all PIDs that containing that PID.

4.4. Endpoint Property Service

As described in Section 10.8 of the ALTO protocol specification, 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 Cost Service MUST return the value that the Filtered PID Property Map Service would return for the PID containing that endpoint.

For properties 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.

5. Security Considerations

Some 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.

6. IANA Considerations

No actions are required from IANA as result of the publication of this document.

7. References

[I-D.ietf-alto-protocol]
Almi, R., Penno, R., and Y. Yang, "ALTO Protocol", draft-ietf-alto-protocol-20 (work in progress), October 2013.

[RFC.4632]
Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", RFC 4632, BCP 122, August 2006.

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 WG
Internet-Draft
Intended status: Standards Track
Expires: April 11, 2015

W. Roome
Alcatel-Lucent

October 8, 2014

Resource Attributes for ALTO Protocol
draft-roome-alto-resource-attr-00

Abstract

This document extends the Application-Layer Traffic Optimization (ALTO) Protocol [RFC7285] by defining additional descriptive attributes for the resources offered by an ALTO Server.

Status of this Memo

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

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

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

This Internet-Draft will expire on April 11, 2015.

Table of Contents

1. Introduction

In the ALTO Protocol [RFC7285], the Information Resource Directory (IRD) defines the resources, or services, offered by an ALTO server. An IRD contains one entry for each resource. Each entry provides the information a client needs to use that resource: the URI for the resource, the type of data the server returns, the type of data the client sends (if any), the identifiers of any resources on which this resource depends, and optional capabilities for the resource.

However, resources have many additional attributes. For example, a Network Map resource maps PID (Provider-defined Identifier) names to network address prefixes. A small Network Map resource might define 20 PIDs and a total of 100 prefixes, while a large Network Map might have 5,000 PIDs and 500,000 prefixes. A client might like to know the size of the Network Map before retrieving it, but the IRD gives no hint.

Also, an ALTO server provided by an Internet Service Provider (ISP) might have detailed information for network addresses managed by that ISP, but only coarse information (or no information at all) for other network addresses. If an ALTO Client needs cost information for a particular network address, and the client knows about several different ALTO Servers, the client would prefer to use the ALTO Server with the best data for that address. But the IRD entries give no hint as to the best server; the client would have to try each server and guess as to which was the most accurate.

This document defines a framework for declaring such attributes in IRD entries, and defines an initial set of attributes.

1.1. Attributes Versus Capabilities

The ALTO Protocol already defines a "capabilities" section for IRD entries, so one alternative is simply to define additional capabilities. We prefer not to do that for several reasons:

- Capabilities are always defined; the ALTO protocol either requires the IRD entry to specify a value, or else defines a default. Attributes are optional, and do not have defaults.

- Capability names are defined by the ALTO protocol; adding a new capability requires a new RFC. Attribute names are managed by IANA, so new attributes can be added without extending the protocol.

- A client can deduce attributes by using the resource. Capabilities cannot be deduced.

- Attribute values are approximate, and/or may change frequently. Capabilities are accurate, and should not change unless the resource is redefined. For example, if a Cost Map attribute says the map has 500 cost points, that means that was the size when the client retrieved the IRD. If the client retrieves the map a few minutes later, the size might be different. However, if a capability says a Cost Maps returns a "routingcost" cost metric, it is an error if the resource returns a "hopcount" instead.

2. Changes To ALTO Protocol

2.1. IRD Entries

Resources attributes are defined by a new field, named "attributes", at the same level as the existing "uri" and "media-type" fields in IRD entries. That is, in [RFC7285] Sec. 9.2.2, IRDResourceEntry is revised as:

```
object {
  JSONString      uri;
  JSONString      media-type;
  [JSONString     accepts;]
  [Capabilities   capabilities;]
  [ResourceID     uses<0..*>;]
  [Attributes     attributes;]      // Added
} IRDResourceEntry;

object {
   ...
} Attributes;    // Added
```

Here is an example of an IRD with attributes:

```
...
"resources" : {
   "my-default-network-map" : {
      "uri" : "http://alto.example.com/networkmap",
      "media-type" : "application/alto-networkmap+json",
      "attributes" : {
         "pid-count-current" : 50,
         "prefix-count-current" : 150,
         "authoritative-prefixes" : {
            "ipv4": [ "1.2.0.0/16", "3.4.0.0/16" ]
         },
         "authoritative-exclusions" : {
            "ipv4": [ "1.2.1.0/25" ]
         }
     }
   },
   "numerical-routing-cost-map" : {
      "uri" : "http://alto.example.com/costmap/num/routingcost",
      "media-type" : "application/alto-costmap+json",
      "capabilities" : {
         "cost-type-names" : [ "num-routing" ]
      },
      "uses" : [ "my-default-network-map" ],
      "attributes" : {
         "cost-count-current" : 500,
         "cost-count-range" : [ 250, 750 ]
      }
   },
   ...
```

3. Resource Attributes

3.1. Attribute Names

An attribute name is encoded as a string. The string MUST be no more than 32 characters, and it MUST NOT contain characters other than US-ASCII alphanumeric characters (U+0030-U+0039, U+0041-U+005A, and U+0061-U+007A), the hyphen ('-', U+002D), the colon (':', U+003A), the low line ('_', U+005F), or the '.' separator (U+002E). The '.' separator is reserved for future use and MUST NOT be used unless specifically indicated by a companion or extension document.

Identifiers prefixed with "priv:" are reserved for Private Use [RFC5226] without a need to register with IANA. All other identifiers MUST be registered in the "ALTO Resource Attribute Registry" (see Section 7). For an identifier with the "priv:"

prefix, an additional string (e.g., company identifier or random string) MUST follow (i.e., "priv:" only is not a valid identifier) to reduce potential collisions.

Section 3.3 defines an initial set of Resource Attributes.

3.2. Attribute Values

The type of an attribute value depends on the attribute. When registering an attribute with IANA, the JSON value type MUST be specfied.

3.3. Proposed Resource Attributes

We propose the following Resource Attributes:

pid-count-current:
: Resource type: Network Map
Value type: JSON Number
Semantics: The number of PIDs in the current Network Map.

prefix-count-current:
: Resource type: Network Map
Value type: JSON Number
Semantics: The total number of prefixes in the current Network Map.

pid-count-range:
: Resource type: Network Map
Value type: JSON Array with two JSON Numbers
Semantics: Low and high values for the number of PIDs in the Network Map. This range SHOULD cover the expected size of the map for the foreseeable future.

prefix-count-range:
: Resource type: Network Map
Value type: JSON Array with two JSON Numbers
Semantics: Low and high values for the total number of prefixes in the Network Map. This range SHOULD cover the expected size of the map for the foreseeable future.

authoritative-prefixes:
: Resource types: Network Map and Endpoint Cost Map
Value type: A JSON Object of type NetworkMapData (Section 11.2.1.6 of [RFC7285])
This resource is authoritative (see Section 3.4) for all endpoints which are covered by a prefix in this set, and which are not covered by a prefix in the "authoritative-exclusions"

set.

authoritative-exclusions:
Resource types: Network Map and Endpoint Cost Map
Value type: A JSON Object of type NetworkMapData (Section 11.2.1.6 of [RFC7285])
See the "authoritative-prefixes" resource, above.

cost-count-current:
Resource type: Cost Map
Value type: JSON Number
Semantics: The number of cost points in the current Cost Map. Note that although a Cost Map is an NxN matrix, where N is the number of PIDs, a Cost Map is not required to define a cost for every {source,destination} pair.

cost-count-range
Resource type: Cost Map
Value type: JSON Array with two JSON Numbers
Semantics: Low and high values for the total number of cost points in the Cost Map. This range SHOULD cover the expected size of the map for the foreseeable future.

3.4. Authoritative Endpoint Sets

The set of endpoints defined by the "authoritative-prefixes" and "authoritative-exclusions" attributes are the endpoints for which a resource provides authoritative data. For an ALTO Server provided by an ISP, this will usually be the endpoints of the ISP's customers.

While we do not rigorously define "authoritative", in general if a resource is authoritative for a set of endpoints, then:

- No other ALTO Server has more accurate data for those endpoints.
- The ALTO Server SHOULD fully specify the costs for those endpoints. That is, for every endpoint EA in the authoritative set and every other endpoint EX, the associated cost map service(s) SHOULD define the costs from EA to EX and from EX to EA.
- PIDs covering endpoints in the authoritative set SHOULD be more detailed (smaller, finer-grained, etc.) than PIDs for other endpoints.

We define the authoritative set with two sets of prefixes, one inclusive, the other exclusive, for the following reason. The most likely case is that an authoritative set represents the addresses of

an ISP's customers. In this case, the inclusive set would be the prefixes that have been allocated to the ISP by the appropriate authority. If customers have migrated to other ISPs and taken their addresses, those prefixes form the exclusionary set. While the exclusionary set is not necessary, it provides an efficient way to represent a large block of addresses with a few exceptions. This two-level model seems to be a good balance between efficiency and simplicity.

4. Use Cases

Here are brief descriptions of several ways in which clients can take advantage of Resource Attributes:

- Advance knowledge of the size of a network or cost map allows an ALTO client to allocate sufficient space to hold the map, or to select the appropriate way to store it. For example, a Cost Map is an NxN matrix, where N is the number of PIDs. But it may be sparse. If the number of costs in the Cost Map is considerably less than the square of the number of PIDs in the associated Network Map, then the client knows in advance that the Cost Map is likely to be sparse, and can use a sparse matrix object. On the other hand, if the number of costs is close to the square of the number of PIDs, the client knows that there is no advantage in using sparse matrix techniques.

- A client such as a P2P tracker can use the authoritative sets to select the appropriate ALTO Server. For example, suppose the tracker has obtained the IRDs for a number of ISP-maintained ALTO Servers. When a peer at address E asks the tracker for a set of peers, the tracker would locate the Network Map (or Endpoint Cost Service) whose authoritative set includes E. This will mostly be the ALTO Server for the requesting peer's ISP. The tracker uses that ALTO resource to evaluate costs between E and the other peers, and returns the peers with the lowest costs.

- If an ALTO Server offers several different Network Maps, a client can use the network map size and/or authoritative sets to select the appropriate Network Map.

5. Alternatives And Discussion (Delete if draft is published)

Here a few points for which I welcome discussion:

- Can anyone suggest a better name than "attributes"? I considered "properties", but that conflicts with Endpoint Properties.

Another possibility is "meta", but we have overloaded that name.

- I don’t like the term "authoritive" to describe the endpoints on which a Network Map is centered. However, I can’t think of anything better, and it does seem to accurately reflect the intent. Can anyone suggest an alternative?

6. Security Considerations

Because attributes should not provide any information which cannot be deduced from the data already provided by an ALTO Server, this extension does not introduce any security considerations not already present in the ALTO Protocol.

7. IANA Considerations

This document defines a registry for ALTO Resource Attributes. When a new ALTO Resource Attribute is defined and accepted by the ALTO working group, requests for IANA registration MUST include the identifier, applicable resource type(s), JSON type and intended semantics.

Section 3.3 defines the initial set of Resource Attributes.

8. References

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, BCP 26, May 2008.

[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.

Author's Address

Wendy Roome
Alcatel-Lucent/Bell Labs
600 Mountain Ave, Rm 3B-324
Murray Hill, NJ 07974
USA

Phone: +1-908-582-7974
Email: w.roome@alcatel-lucent.com

ALTO Working Group
Internet-Draft
Intended status: Informational
Expires: April 30, 2015

X. Shi
Y. Yang
Yale University
October 27, 2014

Modeling JSON Messages Using YANG
draft-shi-alto-yang-json-00

Abstract

JavaScript Object Notation (JSON) has been a popular choice as the message encoding for many network protocols. Meanwhile, there are broad interests in the networking community to use the YANG data modeling language [RFC6020] to define data store and protocol messages, so that one can use YANG related tools such as the OpenDayLight Controller. Although YANG itself is XML based, there have been efforts to model JSON content using YANG [draft-ietf-netmod-yang-json-01]

This document explores the conditions under which the messages of a JSON based protocol can have a syntactically equivalent and hence interoperable YANG model. In particular, this document shows that any JSON protocol message with stand-alone non-object JSON values, certain JSON arrays of elements of mixed types, or non-keyword keys in key-value pairs cannot have a syntactically equivalent YANG model. It also applies these conditions to the ALTO and CDNi protocol messages as examples.

Table of Contents

1. Introduction

JavaScript Object Notation (JSON) has been a popular choice as the message encoding for many network protocols such as the Application-Layer Traffic Optimization (ALTO) protocol [RFC7285], the Content Delivery Networks Interconnection (CDNi) protocol [RFC6707], the Port Control Protocol (PCP) [RFC6887], etc.

Meanwhile, there are broad interests in the networking community to use the YANG data modeling language [RFC6020] to define data store and protocol messages, so that one can use YANG related tools such as the OpenDayLight Controller. Although YANG itself is XML based, there have been efforts to model JSON content using YANG [draft-ietf-netmod-yang-json-01]

This document explores the conditions under which the messages of a JSON based protocol can have a syntactically equivalent hence interoperable YANG model, and provides a conversion process from the JSON message to the YANG model. In particular, this document shows that any JSON protocol message with stand-alone non-object JSON values, certain JSON arrays of elements of mixed types, or non-keyword keys in key-value pairs cannot have a syntactically equivalent YANG model. It also applies these conditions to the ALTO and CDNi protocol messages as examples. For protocols that do not have syntactically equivalent YANG models, we also explore the possibility of a semantically equivalent YANG model.

2. Claim

Assume that we follow the specifications of JSON as specified in [RFC7159], the YANG modeling languate as specified in [RFC6020]. and the JSON encoding of data modeled in YANG as specified in [draft-ietf-netmod-yang-json-01].

A JSON based protocol message can have a syntactically equivalent YANG model if and only if:

(1) the message encoding condition is met;

(2) the uri condition is met.

2.1. Message Encoding Condition

2.1.1. Conditions

The JSON message encoding condition is the following:

(1) The message MUST be a JSON object.

(2) The arrays MUST be either all objects, or all non-object non-array types, i.e. false, null, true, number, string;

(3) The JSON message MUST NOT contain a variable as a key or null as a value in a JSON object key-value pair; in other words, the keys in the JSON message must be an already defined constant keyword in the message format of the protocol.

2.1.2. Application of the Message Encoding Condition

2.1.2.1. ALTO Network Map Example

```
{
  "meta" : {
    "vtag": {
      "resource-id": "my-default-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"
      ]
    }
  }
}
```

Some of the keys (e.g. "PID1", "PID2", "PID3") in this JSON object are not pre-defined keywords, which violated Condition (3). Hence it cannot have a syntactically equivalent YANG Model.

2.1.2.2. CDNi Metadata Example

```
{
  "hosts": [
    {
      "host": "video.example.com",
      "_links": {
        "host-metadata" : {
          "type": "application/cdni.HostMetadata.v1+json",
          "href": "http://metadata.ucdn.example/host1234"
        }
      }
    },
    {
      "host": "images.example.com",
      "_links": {
        "host-metadata" : {
          "type": "application/cdni.HostMetadata.v1+json",
          "href": "http://metadata.ucdn.example/host5678"
        }
      }
    }
  ]
}
```

This JSON object, as specified in Section 6.4.2 in [CDNiMetadata], satisfies all three message encoding conditions. Hence it is possible to model it using YANG.

2.2. The URI Condition

Some of the YANG related protocols might have URI constraints, e.g., RESTCONF. There might be also be future protocols that put on additional restraints. In addition, it is a restraint on the data (the uri in the JSON message), instead of on the data model.

In case these constraints are in place, to resolve the issue, the uri in the JSON message could be conformed to constraint-compliant uri; alternatively, the server could set up a gateway translation for uri. Hence, for now we focus only on the message encoding condition and leave this condition open.

3. Proof of syntactic equivalence conditions

3.1. The necessity of the message encoding condition

We prove the necessity of the message encoding condition by proving its contrapositive.

3.1.1. Condition (1)

Although there are no strict rules to convert from JSON to XML, a stand-alone non-object JSON value would not have an XML tag, which fails because of the same reason for Condition (3).

3.1.2. Condition (2)

YANG cannot model stand alone arrays as non of the YANG statements maps to an array according to [draft-ietf-netmod-yang-json-01].

There are two statements which generate a name/array pair in YANG, the leaf-list statement and the list statement. For a leaf-list, the element type must be a YANG datatype (Section. 6 of [draft-ietf-netmod-yang-json-01]); for a list, the array elements must be JSON objects. Hence, there cannot be a mix.

3.1.3. Condition (3)

If one of the keys in the key-value pair in the JSON document is not pre-defined, the corresponding XML tags will not be pre-defined keywords. Therefore, it would not possible to model it in YANG without using YANG's anyxml statement (which allows arbitrary XML content). However, using the anyxml statement defeats the purpose of modeling the data as it allows arbitrary XML content, and will not help the subsequent parsing process.

3.1.4. Conclusion

The above three conditions are necessary to model a JSON message in YANG.

3.2. The sufficiency of the message encoding condition

We prove this by providing a translation procedure from a JSON message that is compliant with the protocol we are trying to model, to a custom java class that can be used for Jackson data binding, then to a YANG model. Note that the middle step of translating to and from the custom parser class is not necessary, but it is useful. This also implies which data binding/JSON parser we use does not matter.

3.2.1. Motivation of using Jackson data binding

JSON data binding is the process of binding data structures (objects, arrays, etc.) in JSON to the appropriate data structures in the server (e.g. java classes, database tables, etc.) in parsing the JSON text. In order to process JSON messages in a meaningful manner, data binding is necessary. Even if the binding is not explicit, the server would need to do it eventually. For example, one can read JSON content in a stream without binding it to the java classes, but eventually in order to make sense of the data, the server would eventually have to organize it, which is analogous to data binding upfront. Popular choices for JSON parsing and data binding include jackson and gson.

We use Jackson full data binding as our approach. Full data binding binds JSON content into plain old java objects (POJOs), i.e. this custom parser class can neither extend nor implement any other class. Jackson uses ObjectMapper with the custom parser class to parse JSON content into this class.

The no inheritance assumption means that all custom parser classes are base classes. Nevertheless, this assumption is also not necessary--we can simply make any derived classes a new based class.

3.2.2. The message encoding condition in Jackson terms

For now, we make a stronger assumption than Condition (2): we assume all arrays contain elements that are homogeneous non-array JSON values. We handle the case of heterogeneous objects and heterogeneous non-object non-array types later.

The message encoding condition (3) is that all keys in each key-value pair in the JSON text must be pre-defined keywords. As the keys will become either class names and instance variable names, or be keys in the java maps, it is easy to see that this condition is equivalent to: there exists a full data binding in Jackson custom parser class without using any map structures (Map<String, ?>)."

3.2.3. Proof

3.2.3.1. Generate Jackson Parser from JSON Message

We provide a recursive binding process from a JSON object to the Jackson custom parser class to be used by Jackson ObjectMapper.

```
Type determine_type(value) {
  if (type(value) is string or number or boolean or null) {
    return the corresponding java primitive type;
  }
  if (value is a JSON object) {
    return build_parser_class(value).class;
  }
  if (value is an array) {
    return ArrayList<T> where T=determine_type(value[0]);
  }
  // should not reach here.
}

Class build_parser_class(JSONObject obj) {
  create custom class C;
  for each key/value pair in the obj {
    add instance variable v in C;
    the name of variable v <- key;
    the type of variable v <- determine_type(value);
  }
  return C.class;
}
```

To ensure the naming in the YANG model is consistent with the JSON message, We follow the naming process: change everything into CamelCase (i.e. remove dashes, etc.); for instance variables, use "my" prefix, (e.g. myVariable, myNetworkMap, etc.); for the custom class name, if the object is an element of the array, use "Element" suffix; if a class already exists, we add a number after it which we then remove at the next stage.

3.2.3.2. Generate YANG Model from Jackson Parser

Given a Jackson Parser Java Class, the following algorithm generates a YANG model:

```
YANGModel build_yang_model(Class C) {
  for each instance variable (Type, Name) {
    if (Type is primitive type: string, number, boolean, null) {
      add the following to the YANG module:
      "leaf Name { type <YANG equivalent of Type>; }"
    }
    if (Type is an ArrayList<TypeElement>) {
      if (TypeElement is primitive type) {
        add the following to the YANG module:
        "leaf-list Name { type <YANG equivalent of TypeElement>; }"
      } else {
        // TypeElement is a custom parser class
        add the following to the YANG module:
        "list Name { build_yang_model(TypeElement.class) }"
      }
    }
    if (Type is a custom parser class) {
      add the following to the YANG module:
      "container Name { build_yang_model(Type.class) }"
    }
  }
}
```

3.2.3.3. Handling heterogeneous arrays

Due to Java's strong typing, handling heterogeneous arrays is difficult, but not impossible. For simplicity, we provide the YANG model directly for heterogeneous arrays and provide guidelines for Jackson parsing.

3.2.3.3.1. Arrays of entirely primitive types

If the array is consisted of entirely non-object, non-value types type_1, type_2, ..., type_n, the corresponding YANG model is a leaf-list with a derived type.

```
typedef element-type {
  type union {
    type type_1;
    type type_2;
    ...
    type type_m;
  }
}

leaf-list key-name {
  type element-type;
}
```

In Jackson, one may parse such arrays as String[] and then convert to whichever types needed.

3.2.3.3.2. Example

```
foo : [
  12,
  "abc",
  true
]
```

```
leaf-list foo {
  type union {
    type int32;
    type string;
    type boolean;
  }
}
```

3.2.3.3.3. Arrays of entirely JSON objects

If the array is consisted of entirely JSON object, we create a YANG grouping for each object (by following the same process as build_parser_class and build_yang_object above). Then use list, choice, and case statements.

```
list key-name {
  choice elements {
    case element-object-1 {
      uses grouping-1;
    }
    case element-object-2 {
      uses grouping-2;
    }
    ...
    case element-object-n {
      uses grouping-n;
    }
  }
}
```

In Jackson, one may use annotations or custom deserializers to parse such structures.

3.2.3.4. Example

```
foo : [
  {
    "bar" : 12,
    "baz" : "abc"
  },
  {
    "name" : "Cyrus T. Elk",
    "year" : 1938
  }
]
```

```
grouping grouping-1 {
  leaf bar {
    type int32;
  }
  leaf baz {
    type string;
  }
}

grouping grouping-2 {
  leaf name {
    type string;
  }
  leaf year {
    type int32;
  }
}

list foo {
  choice element-types {
    case element-type-1 {
      uses grouping-1;
    }
    case element-type-2 {
      uses grouping-2;
    }
  }
}
```

3.3. Examples

3.3.1. ALTO cost map example

Take the cost map response from the ALTO protocol [RFC7285] as an example:

```
{
  "meta" : {
    "dependent-vtags" : [
      {"resource-id": "my-default-network-map",
       "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"
      }
    ],
    "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  }
  }
}
```

This JSON object does not have a syntactically equivalent model in YANG because some of its keys are variables, e.g., the "PID1", "PID2", etc.

If we change this cost map object to the following, we will be able to model it.

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

```
            },
            {
              "dst": "PID2",
              "cost": 5
            },
            {
              "dst": "PID3",
              "cost": 10
            }
          ]
        },
        {
          "src":"PID2",
          "dst-costs" : [
            {
              "dst": "PID1",
              "cost" : 5
            },
            {
              "dst": "PID2",
              "cost": 1
            },
            {
              "dst": "PID3",
              "cost": 15
            }
          ]
        },
        {
          "src":"PID3",
          "dst-costs" : [
            {
              "dst": "PID1",
              "cost" : 20
            },
            {
              "dst": "PID2",
              "cost": 15
            }
          ]
        }
      ]
    }
```

Result of build_parser_class(obj):

```
Class Message {
  Meta myMeta;
  ArrayList<CostMapElement> myCostMap;
}
Class Meta {
  ArrayList<DependentVtagsElement> myDependentVtags;
  CostType myCostType;
}
Class DependentVtagsElement {
  String myResourceId;
  String myTag;
}
Class CostType {
  String myCostMode;
  String myCostMetric;
}
Class CostMapElement {
  String mySrc;
  ArrayList<DstCostsElement> myDstCosts;
}
Class DstCostsElement {
  String myDst;
  int myCost;
}
```

Generated YANG model:

```
   container meta {
     list dependent-vtags {
       leaf resource-id {
         type string;
       }
       leaf tag {
         type string;
       }
     }
     container cost-type {
       leaf cost-mode {
         type string;
       }
       leaf cost-metric {
         type string;
       }
     }
   }
   list cost-map {
     leaf src {
       type string;
     }
     list dst-costs {
       leaf dst {
         type string;
       }
       leaf cost {
         type int64;
       }
     }
   }
```

This YANG model does validate the JSON cost map object with JSON encoding defined in [draft-ietf-netmod-yang-json-01].

3.3.2. CDNi metadata example

The original JSON message is the following:

```
{
  "metadata": [
    {
      "generic-metadata-type":"application/cdni.SourceMetadata.v1+json",
      "generic-metadata-value": {
        "sources": [
          {
            "_links": {
              "auth": {
```

```
                "auth-type": "application/cdni.Auth.v1+json",
                "href": "http://metadata.ucdn.example/auth1234"
              }
            },
            "endpoint": "acq1.ucdn.example",
            "protocol": "ftp"
          },
          {
            "_links": {
              "auth": {
                "auth-type": "application/cdni.Auth.v1+json",
                "href": "http://metadata.ucdn.example/auth1234"
              }
            },
            "endpoint": "acq2.ucdn.example",
            "protocol": "http"
          }
        ]
      }
    },
    {
      "generic-metadata-type": "application/cdni.LocationACL.v1+json",
      "generic-metadata-value": {
        "locations": [
          {
            "locations": [
              {
                "footprint-type": "IPv4CIDR",
                "footprint-value": "192.168.0.0/16"
              }
            ],
            "action": "deny"
          }
        ]
      }
    },
    {
      "generic-metadata-type": "application/cdni.ProtocolACL.v1+json",
      "generic-metadata-value": {
        "protocols": [
          {
            "protocols": [
              "ftp"
            ],
            "action": "deny"
          }
        ]
      }
```

```
    }
  ],
  "paths": [
    {
      "path-pattern": {
        "pattern": "/video/trailers/*"
      },
      "_links": {
        "path-metadata": {
          "type": "application/cdni.PathMetadata.v1+json",
          "href": "http://metadata.ucdn.example/host1234/pathABC"
        }
      }
    },
    {
      "path-pattern": {
        "pattern": "/video/movies/*"
      },
      "_links": {
        "path-metadata": {
          "type": "application/cdni.PathMetadata.v1+json",
          "href": "http://metadata.ucdn.example/host1234/pathDCE"
        }
      }
    }
  ]
}
```

Result of build_parser_class(obj):

```
Class Message {
  ArrayList<MetaDataElement> myMetaData;
  ArrayList<PathsElement> myPaths;
}
Class MetaDataElement {
  String myGenericMetadataType;
  GenericMetadataValue myGenericMetadataValue;
}
Class GenericMetadataValue {
  ArrayList<SourcesElement> mySources;
  ArrayList<LocationsElement> myLocations;
  ArrayList<ProtocolsElement> myProtocols;
}
Class SourcesElement {
  _Links my_Links;
  String myEndpoint;
  String myProtocol;
```

```
}
Class _Links {
  Auth myAuth;
}
Class Auth {
  String myAuthType;
  String myHref;
}
Class LocationsElement {
  ArrayList<LocationsElement2> myLocations;
  String myAction;
}
Class LocationsElement2 {
  String myFootprintType;
  String myFootprintValue;
}
Class ProtocolsElement {
  ArrayList<String> myProtocols;
  String myAction;
}
Class PathsElement {
  PathPattern myPathPattern;
  _Links2 my_Links;
}
Class PathPattern {
  String myPattern;
}
Class _Links2 {
  PathMetadata myPathMetadata;
}
Class PathMetadata {
  String myType;
  String myHref;
}
```

Generated YANG model:

```
list metadata {
  leaf generic-metadata-type {
    type string;
  }
  container generic-metadata-value {
    list sources {
      container _links {
        container auth {
          leaf auth-type {
            type string;
          }
```

```
            leaf href {
              type string;
            }
          }
        }
        leaf endpoint {
          type string;
        }
        leaf protocol {
          type string;
        }
      }
      list locations {
        list locations {
          leaf footprint-type {
            type string;
          }
          leaf footprint-value {
            type string;
          }
        }
        leaf action {
          type string;
        }
      }
      list protocols {
        leaf-list protocols {
          type string;
        }
        leaf action {
          type string;
        }
      }
    }
  }
  list paths {
    container path-pattern {
      leaf pattern {
        type string;
      }
    }
    container _links {
      container path-metadata {
        leaf type {
          type string;
        }
        leaf href {
          type string;
```

```
        }
      }
    }
  }
```

This YANG model does validate the JSON cost map object with JSON encoding defined in [draft-ietf-netmod-yang-json-01].

3.4. Concluding remarks

This process proves that the message encoding condition is a sufficient condition for the JSON object to have a YANG model.

Note the model generated is very crude and lose almost all constraints and all inheritance features (if any), because it focuses on the syntax and is essentially converted from an JSON object compliant with a protocol instead of from the protocol itself. Hence this result is more useful in determining which JSON based protocols cannot have a syntactically equivalent YANG model, than in generating a usable YANG model.

4. Semantic Equivalence

For JSON based protocols that don’t satisfy the message encoding condition, it is still possible to have a semantically equivalent YANG model. All that is required for the protocol compliant clients and the YANG model compliant server to interoperate is an adapter which does the following:

1) translate FROM YANG server compliant response msg TO alto compliant response msg;

2) translate FROM alto compliant request msg TO YANG server compliant request msg;

4.1. Claim

This adapter needs to be protocol-aware.

Ideally, given any YANG model, we would like to be able to automatically (or at least mechanically) generate this message adapter, which means not looking at the protocol or its compliant msgs. However, without knowing the specific protocol that we are working with (i.e. human intervention, i.e. looking at the protocol compliant msgs), such an adapter cannot be auto-generated.

4.2. Proof by Indistinguishability

Suppose both the YANG server compliant msg m_y and the actually protocol compliant msg m_p are in JSON (or have been encoded into JSON). Looking at the differences between the two messages, call these differences {d1, d2, ..., dn}. The goal for the auto-generated adapter would be to identify and eliminate these differences. Construct a new JSON msg m' where all but one difference di is the same as m_p and di is the same as the m_y. Without looking at the protocol (or m_p), the auto-generated adapter would not be able to distinguish between m' and m_p in its translation process, which means, it won't be able to tell whether it should change di or not. Hence, such an adapter must be protocol-aware.

A good example is the dependent-vtag in the ALTO protocol:

```
"dependent-vtag" : [
  {
    "resource-id" : "my-network-map",
    "tag" : "abcd1234"
  }
]
```

It was specified this way in the alto protocol. However, it could conceivably be the case that it was originally the following map structure, and was converted into the above encoding because of the map->list+key issue. (This case is actually one of the few differences in the m_y and m_p where the adapter does not need to convert it back to a map structure.)

```
"dependent-vtag" : {
  "my-network-map" : {
    "tag" : "abcd1234"
  }
}
```

Without knowing the protocol or protocol messages, it is impossible to distinguish.

5. Ramifications

We now understand the basic condition for a JSON based protocol to have a YANG Model. For the protocols that don't meet this condition, there can be a semantic equivalent YANG model, but there won't be a generic process of generating the adapter for all protocols.

6. Security Considerations

This document does not introduce security or privacy concerns.

7. IANA Considerations

This document does not have IANA considerations.

8. References

[RFC6707] Niven-Jenkins, B., Le Faucheur, F., and N. Bitar, "Content Distribution Network Interconnection (CDNi) Problem Statement", RFC 6707, September 2012.

[CDNiMetadata] Niven-Jenkins, B., Murray, R., Caulfield, M., Leung, K., and K. Ma, "CDN Interconnection Metadata", July 2014.

[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.

[RFC6887] Wing, D., Ed., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol", RFC 6887, April 2013.

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

[draft-ietf-netmod-yang-json-01] Lhotka, L., "JSON Encoding of Data Modeled with YANG", October 2014.

Authors' Addresses

Xiao Shi
Yale University
51 Prospect Street
New Haven, CT 06511
USA

Email: xiao.shi@yale.edu

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

Email: yang.r.yang@gmail.com

ALTO Working Group
Internet-Draft
Intended status: Informational
Expires: April 30, 2015

X. Shi
Y. Yang
Yale University
M. Scharf
Alcatel-Lucent Bell Labs
October 27, 2014

A YANG Data Model for Base ALTO Data and Services
draft-shi-alto-yang-model-00

Abstract

The Application-Layer Traffic Optimization (ALTO) protocol [RFC7285] defines a set of network information services, including the network-map service, the cost-map service, the filtered map services, the endpoint property service, and the endpoint cost service. A meta service, called the information resource directory (IRD) service, allows an ALTO server to provide ALTO clients with meta information (e.g., the access URI) about each resource and service it provides. [RFC7285] uses a RESTful design and encodes client request parameters and server responses using JSON. One may consider that most of these services are based on data maintained at an ALTO server. Hence, in this document, we explore how one may use the data modeling language YANG [RFC6020] to specify the services defined in [RFC7285]. We first define two YANG models for RPC specification and data instance description of of ALTO services. We then discuss the "standard operations" defined in NETCONF/RESTCONF to evaluate potential integration.

Table of Contents

1. Introduction

This document explores how one may use the data modeling language YANG [RFC6020] to specify the network information services defined in the Application-Layer Traffic Optimization (ALTO) protocol [RFC7285]. In particular, this work is motivated by the recent substantial interest in the networking community to use YANG in networking protocol specification and development. This interest is highly justified, given that the potential benefits of using a formal specification can be multifold, including both precision and the potential of automation. However, at the same time, data modeling languages are generally domain specific languages (DSL) and hence are restricted by their specific domains. For example, YANG is developed in the specific domain of modeling configuration and state data stored at network devices. ALTO information does not fit in this domain but is highly related: ALTO information should be derived from network configuration and state data. Hence, modeling ALTO using YANG provides an interesting exercise on how one may apply YANG slightly outside its original domain.

The initial goal of this document was to produce an ALTO specification using YANG so that the YANG specification of ALTO, which we refer to as ALTO/YANG, and the original specification [RFC7285] can inter-operate. We refer to an interop specification as a syntactically equivalent specification. In other words, the ALTO/YANG specification produces the same behavior as that of the original specification. Unfortunately, as we have shown in [draft-shi-alto-yang-json-00], even without the need to consider other components such as NETCONF/RESTCONF to construct a complete YANG system, the basic encoding rule of YANG already makes syntactic equivalence unfeasible. In particular, [RFC7285] makes extensive use of the common key-value store abstraction in its JSON object encoding. But such encoding cannot be generated using the existing YANG/JSON encoding. As a result, the focus of this document is on using YANG to define semantically equivalent ALTO services. We still strive that the syntax generated by ALTO/YANG is close to that of [RFC7285].

The rest of this document is organized as follows. In Section Section 2, we provide an overview of our design approaches. We discuss three specification approaches, considering the three use cases of YANG. In Section 3, we discuss more detailed specification issues that we have encountered. The detailed specification of ALTO/YANG is in Appendix A.

2. Design and Structure Overview of ALTO/YANG

2.1. Overview

Our design tries to use YANG in three aspects: (1) RPC specification, (2) data instance description, and (3) standard operations. These three aspects are not mutually exclusive of each other, but assume different levels of matching and use of YANG.

The RPC specification aspect allows precise specification and automation of RPC input/output data serialization and deserialization. This aspect of using YANG can be considered the simplest. The protocol to be specified does not need to match the operational model of YANG. In Section Section 2.2, we specify a YANG model named alto-service-rpc that uses YANG only in this aspect.

The data instance description and standard operations aspects are analogous to databases. In relational database, one uses SQL to describe the data schema (e.g., in CREATE TABLE). We refer to this process of modeling the data as the data instance description aspect. Modeling the data in YANG does not guarantee full compatibility with the operational model of the existing protocol. In particular, the data instances specified in a YANG model may be virtual in a protocol design. In Section Section 2.3, we specify a YANG model named alto-service-did that uses YANG to specify both RPCs and potential ALTO base data instances to implement certain services.

SQL also defines statements such as SELECT, INSERT, UPDATE, and DELETE. With a DBMS implementing SQL, when a user has a new set of data to be managed, the user describes the new data schema. The existing statements are already available to retrieve and manipulate the data. This substantially reduce the efforts to introduce and manage a new set of data. We refer to this as the standard operations aspect. In YANG related efforts, NETCONF and RESTCONF each tries to provide some of these standard operations (e.g., RESTCONF query). To utilize this aspect, the YANG specification need to be "embedded" in the chosen specification (e.g., NETCONF). In Section Section 2.4, we discuss specification by considering that the YANG model has the support of "standard operations" defined in RESTCONF. This document does not provide a specific specification, and will do so in the next version.

2.2. ALTO/YANG using only RFC Specification

Figure 1 gives the tree diagram of a YANG model named alto-service-rpc to specify ALTO services. As it is clear from the specification, it has only RPCs, which are supported by typedef and grouping not

shown in the tree diagram. Such a model is still valuable in that it clarifies the input and output data format of ALTO services.

```
module: alto-service-rpc
rpcs:
   +---x IRD-service
   |  +--ro output
   |     +--ro IRD-service
   |        +--ro meta
   |        |  +--ro cost-types* [cost-type-name]
   |        |  |  +--ro cost-type-name    cost-type-name
   |        |  |  +--ro cost-mode         cost-mode
   |        |  |  +--ro cost-metric       cost-metric
   |        |  |  +--ro description?      string
   |        |  +--ro default-alto-network-map    resource-id
   |        +--ro resources* [resource-id]
   |           +--ro resource-id     resource-id
   |           +--ro uri             inet:uri
   |           +--ro media-type      media-type
   |           +--ro accepts*        media-type
   |           +--ro capabilities
   |           |  +--ro cost-constraints?   boolean
   |           |  +--ro cost-type-names*    cost-type-name
   |           |  +--ro prop-types*         endpoint-property-type
   |           +--ro uses*           resource-id
   +---x network-map-service
   |  +--ro input
   |  |  +--ro uri    inet:uri
   |  +--ro output
   |     +--ro network-map-service
   |        +--ro meta
   |        |  +--ro vtag
   |        |     +--ro resource-id    resource-id
   |        |     +--ro tag            string
   |        +--ro network-map* [pid]
   |           +--ro pid                        pid-name
   |           +--ro endpoint-address-group* [address-type]
   |              +--ro address-type       endpoint-address-type
   |              +--ro endpoint-prefix*   endpoint-prefix
   +---x cost-map-service
   |  +--ro input
   |  |  +--ro uri    inet:uri
   |  +--ro output
   |     +--ro cost-map-service
   |        +--ro meta
   |        |  +--ro dependent-vtags*
   |        |  |  +--ro resource-id    resource-id
   |        |  |  +--ro tag            string
```

```
|        |  +--ro cost-type
|        |     +--ro cost-mode      cost-mode
|        |     +--ro cost-metric    cost-metric
|        |     +--ro description?   string
|        +--ro cost-map* [src]
|           +--ro src          pid-name
|           +--ro dst-costs* [dst]
|              +--ro dst     pid-name
|              +--ro cost
+---x filtered-network-map-service
|  +--ro input
|  |  +--ro pids*            pid-name
|  |  +--ro address-types*   endpoint-address-type
|  +--ro output
|     +--ro filtered-network-map-service
|        +--ro meta
|        |  +--ro vtag
|        |     +--ro resource-id    resource-id
|        |     +--ro tag            string
|        +--ro network-map* [pid]
|           +--ro pid                        pid-name
|           +--ro endpoint-address-group* [address-type]
|              +--ro address-type       endpoint-address-type
|              +--ro endpoint-prefix*   endpoint-prefix
+---x filtered-cost-map-service
|  +--ro input
|  |  +--ro cost-type
|  |  |  +--ro cost-mode      cost-mode
|  |  |  +--ro cost-metric    cost-metric
|  |  |  +--ro description?   string
|  |  +--ro constraints*   constraint
|  |  +--ro pids
|  |     +--ro srcs*   pid-name
|  |     +--ro dsts*   pid-name
|  +--ro output
|     +--ro filtered-cost-map-service
|        +--ro meta
|        |  +--ro dependent-vtags*
|        |  |  +--ro resource-id    resource-id
|        |  |  +--ro tag            string
|        |  +--ro cost-type
|        |     +--ro cost-mode      cost-mode
|        |     +--ro cost-metric    cost-metric
|        |     +--ro description?   string
|        +--ro cost-map* [src]
|           +--ro src          pid-name
|           +--ro dst-costs* [dst]
|              +--ro dst     pid-name
|
```

```
|              +--ro cost
+---x endpoint-property-service
|  +--ro input
|  |  +--ro properties*   endpoint-property-type
|  |  +--ro endpoints*    typed-endpoint-address
|  +--ro output
|     +--ro endpoint-property-service
|        +--ro meta
|        |  +--ro dependent-vtags*
|        |     +--ro resource-id    resource-id
|        |     +--ro tag            string
|        +--ro endpoint-properties* [endpoint]
|           +--ro endpoint      typed-endpoint-address
|           +--ro properties* [property-type]
|              +--ro property-type    endpoint-property-type
|              +--ro property         endpoint-property-value
+---x endpoint-cost-service
   +--ro input
   |  +--ro cost-type
   |  |  +--ro cost-mode      cost-mode
   |  |  +--ro cost-metric    cost-metric
   |  |  +--ro description?   string
   |  +--ro constraints*   constraint
   |  +--ro endpoints
   |     +--ro srcs*   typed-endpoint-address
   |     +--ro dsts*   typed-endpoint-address
   +--ro output
      +--ro endpoint-cost-service
         +--ro meta
         |  +--ro cost-type
         |     +--ro cost-mode      cost-mode
         |     +--ro cost-metric    cost-metric
         |     +--ro description?   string
         +--ro endpoint-cost-map* [src]
            +--ro src          typed-endpoint-address
            +--ro dst-costs* [dst]
               +--ro dst     typed-endpoint-address
               +--ro cost
```

2.3. ALTO/YANG using Data Instance Description

Figure 2 shows the tree diagram of a second YANG model for ALTO, named alto-service-did. Comparing Figure 1 with Figure 2, one can observe that a key difference is that the second model first introduces data instances, listed under resources, that an ALTO server may maintain. We use the word "may" because an ALTO server may not physically store such data. For example, a cost map could be generated on the fly, and hence can be in a sense a virtual data

table. One may observe from our specified instances that we do not specify a data instance to support the endpoint cost service (ECS), as this can be highly inefficient. For ECS, the value of YANG is mainly in RPC specification.

```
module: alto-service-did
   +--ro resources
      +--ro IRD
      |  +--ro meta
      |  |  +--ro cost-types* [cost-type-name]
      |  |  |  +--ro cost-type-name    cost-type-name
      |  |  |  +--ro cost-mode         cost-mode
      |  |  |  +--ro cost-metric       cost-metric
      |  |  |  +--ro description?      string
      |  |  +--ro default-alto-network-map    leafref
      |  +--ro resources* [resource-id]
      |     +--ro resource-id     resource-id
      |     +--ro uri             inet:uri
      |     +--ro media-type      media-type
      |     +--ro accepts*        media-type
      |     +--ro capabilities
      |     |  +--ro cost-constraints?   boolean
      |     |  +--ro cost-type-names*    cost-type-name
      |     |  +--ro prop-types*         endpoint-property-type
      |     +--ro uses*           leafref
      +--ro network-maps*
      |  +--ro meta
      |  |  +--ro vtag
      |  |     +--ro resource-id    resource-id
      |  |     +--ro tag            string
      |  +--ro network-map* [pid]
      |     +--ro pid                        pid-name
      |     +--ro endpoint-address-group* [address-type]
      |        +--ro address-type       endpoint-address-type
      |        +--ro endpoint-prefix*   endpoint-prefix
      +--ro cost-maps*
      |  +--ro meta
      |  |  +--ro dependent-vtags*
      |  |  |  +--ro resource-id    resource-id
      |  |  |  +--ro tag            string
      |  |  +--ro cost-type
      |  |     +--ro cost-mode      cost-mode
      |  |     +--ro cost-metric    cost-metric
      |  |     +--ro description?   string
      |  +--ro cost-map* [src]
      |     +--ro src          pid-name
      |     +--ro dst-costs* [dst]
      |        +--ro dst     pid-name
```

```
         |        +--ro cost
         +--ro endpoint-property-map
            +--ro meta
            |  +--ro dependent-vtags*
            |     +--ro resource-id    resource-id
            |     +--ro tag            string
            +--ro endpoint-properties* [endpoint]
               +--ro endpoint      typed-endpoint-address
               +--ro properties* [property-type]
                  +--ro property-type    endpoint-property-type
                  +--ro property         endpoint-property-value
   rpcs:
      +---x IRD-service
      |  +--ro output
      |     +--ro IRD-service
      |        +--ro meta
      |        |  +--ro cost-types* [cost-type-name]
      |        |  |  +--ro cost-type-name    cost-type-name
      |        |  |  +--ro cost-mode         cost-mode
      |        |  |  +--ro cost-metric       cost-metric
      |        |  |  +--ro description?      string
      |        |  +--ro default-alto-network-map    leafref
      |        +--ro resources* [resource-id]
      |           +--ro resource-id     resource-id
      |           +--ro uri             inet:uri
      |           +--ro media-type      media-type
      |           +--ro accepts*        media-type
      |           +--ro capabilities
      |           |  +--ro cost-constraints?   boolean
      |           |  +--ro cost-type-names*    cost-type-name
      |           |  +--ro prop-types*         endpoint-property-type
      |           +--ro uses*           leafref
      +---x network-map-service
      |  +--ro input
      |  |  +--ro uri    leafref
      |  +--ro output
      |     +--ro network-map-service
      |        +--ro meta
      |        |  +--ro vtag
      |        |     +--ro resource-id    resource-id
      |        |     +--ro tag            string
      |        +--ro network-map* [pid]
      |           +--ro pid                        pid-name
      |           +--ro endpoint-address-group* [address-type]
      |              +--ro address-type       endpoint-address-type
      |              +--ro endpoint-prefix*   endpoint-prefix
      +---x cost-map-service
      |  +--ro input
```

```
      |  |  +--ro uri    leafref
      |  +--ro output
      |     +--ro cost-map-service
      |        +--ro meta
      |        |  +--ro dependent-vtags*
      |        |  |  +--ro resource-id    resource-id
      |        |  |  +--ro tag            string
      |        |  +--ro cost-type
      |        |     +--ro cost-mode      cost-mode
      |        |     +--ro cost-metric    cost-metric
      |        |     +--ro description?   string
      |        +--ro cost-map* [src]
      |           +--ro src          pid-name
      |           +--ro dst-costs* [dst]
      |              +--ro dst     pid-name
      |              +--ro cost
      +---x filtered-network-map-service
      |  +--ro input
      |  |  +--ro pids*            pid-name
      |  |  +--ro address-types*   endpoint-address-type
      |  +--ro output
      |     +--ro filtered-network-map-service
      |        +--ro meta
      |        |  +--ro vtag
      |        |     +--ro resource-id    resource-id
      |        |     +--ro tag            string
      |        +--ro network-map* [pid]
      |           +--ro pid                        pid-name
      |           +--ro endpoint-address-group* [address-type]
      |              +--ro address-type       endpoint-address-type
      |              +--ro endpoint-prefix*   endpoint-prefix
      +---x filtered-cost-map-service
      |  +--ro input
      |  |  +--ro cost-type
      |  |  |  +--ro cost-mode      cost-mode
      |  |  |  +--ro cost-metric    cost-metric
      |  |  |  +--ro description?   string
      |  |  +--ro constraints*   constraint
      |  |  +--ro pids
      |  |     +--ro srcs*   pid-name
      |  |     +--ro dsts*   pid-name
      |  +--ro output
      |     +--ro filtered-cost-map-service
      |        +--ro meta
      |        |  +--ro dependent-vtags*
      |        |  |  +--ro resource-id    resource-id
      |        |  |  +--ro tag            string
      |        |  +--ro cost-type
```

```
|        |     +--ro cost-mode      cost-mode
|        |     +--ro cost-metric    cost-metric
|        |     +--ro description?   string
|        +--ro cost-map* [src]
|           +--ro src          pid-name
|           +--ro dst-costs* [dst]
|              +--ro dst     pid-name
|              +--ro cost
+---x endpoint-property-service
|  +--ro input
|  |  +--ro properties*   endpoint-property-type
|  |  +--ro endpoints*    typed-endpoint-address
|  +--ro output
|     +--ro endpoint-property-service
|        +--ro meta
|        |  +--ro dependent-vtags*
|        |     +--ro resource-id    resource-id
|        |     +--ro tag            string
|        +--ro endpoint-properties* [endpoint]
|           +--ro endpoint      typed-endpoint-address
|           +--ro properties* [property-type]
|              +--ro property-type    endpoint-property-type
|              +--ro property         endpoint-property-value
+---x endpoint-cost-service
   +--ro input
   |  +--ro cost-type
   |  |  +--ro cost-mode      cost-mode
   |  |  +--ro cost-metric    cost-metric
   |  |  +--ro description?   string
   |  +--ro constraints*   constraint
   |  +--ro endpoints
   |     +--ro srcs*   typed-endpoint-address
   |     +--ro dsts*   typed-endpoint-address
   +--ro output
      +--ro endpoint-cost-service
         +--ro meta
         |  +--ro cost-type
         |     +--ro cost-mode      cost-mode
         |     +--ro cost-metric    cost-metric
         |     +--ro description?   string
         +--ro endpoint-cost-map* [src]
            +--ro src          typed-endpoint-address
            +--ro dst-costs* [dst]
               +--ro dst     typed-endpoint-address
               +--ro cost
```

2.4. ALTO/YANG with Integration with NETCONF/RESTCONF

Our next model considers standard operations. In other words, if an RPC can be implemented using a standard operation, the model may use the standard operation and not specify the RPC.

Consider NETCONF. One may use the NETCONF <get> operation to retrieve the full ALTO network map, cost map, and the Information Resource Directory; with the <filter> parameter, NETCONF may retrieve filtered maps as well as endpoint properties. Hence, one may omit the corresponding RPCs defined in alto-service-did. The exact encoding of using "standard operations", however, can be less compact.

3. Non-Trivial Specification Efforts

3.1. YANG Expressiveness Issues

3.1.1. Limitation on Modeling JSON key-value store

ALTO is a JSON based protocol and makes extensive use of JSON key-value store, which is useful because of its efficiency, inherent uniqueness constraint on the keys, and its natural correspondence to and from structures such as indexed database tables or hashmaps.

For example, the network map is defined as mapping from a PID to an endpoint address group. Here is an example network map in Section 11.2.1.7 of [RFC7285].

```
"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"
    ]
  }
}
```

As pointed out in [draft-shi-alto-yang-json-00], such JSON objects cannot be modeled in YANG. To achieve the semantical equivalence, we took the approach of modeling it as a "list" with a unique index ("key) in YANG.

```
list network-map {
  key "pid";
  leaf pid {
    type string;
  }
  list endpoint-address-group {
    key address-type;
    leaf address-type {
      type endpoint-address-type;
    }
    leaf-list endpoint-prefix {
      type endpoint-prefix;
    }
  }
}
```

According to [draft-ietf-netmod-yang-json-01], the above YANG model correspond to the following JSON text:

```
"network-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"]
      }
    ]
  }
]
```

We immediately notice a few disadvantages.

(1) From the YANG validated JSON message alone, it is unclear that the "pid" field is the key to the list "network-map";

(2) The YANG-validated JSON message is much more verbose, which increases the payload, especially when the model scales.

(3) The consistency between address-type and endpoint-prefix is not enforced in the above YANG model.

3.1.2. Limits of Leafrefs

Leafrefs in YANG are heavily tied with XML and XPATH expressions. However, leafrefs cannot refer to an rpc node (or structures within a rpc node). For example, referring to the input parameters in rpc input from rpc output could be very useful.

3.2. Extensibility Issues

3.2.1. String Pattern Extensibility

For string types that have inherant relationships (e.g., extension, concatenation), it is useful to be able to extend the patterns of the strings for modularity and extensibility. For example, TypedEndpointAddress in [RFC7285] is defined as a string of the format AddressType, followed by the ':' separator, followed by a ip-address EndpointAddress. Definitions of the pattern of an ip-address is readily available in Common YANG Data Types [RFC6991]. It is not possible in YANG to inherit, extend, or refer to a pattern of a different string type. Another example is the resource-specific endpoint property, which is defined as a resource ID, followed by the '.' separator (U+002E), followed by a name obeying the same rules as for global endpoint property names.

3.2.2. Type Extensibility

YANG is unable to augment a typedef or enum type. Hence there is not a good way to add a cost metric or type of address if the model is fixed.

4. Applications of ALTO/YANG

4.1. Other Applications

4.1.1. Verifier/Validator on the Client Side

Using tools like pyang (https://code.google.com/p/pyang/), we can not only validate the YANG module, but also translate the module to DSDL schema and validate instance documents. The limitation is that pyang can only validate XML files and the JSON-XML translation is not well-defined. We provide a few examples of pyang-validated ALTO messages in the appendix.

4.1.2. Code Generator

OpenDayLight (ODL) provides automatic code generation for YANG models. ODL yangtools generates a Java OSGi bundle for a given YANG module, including typedefs and groupings, data instances, and RPCs.

One may integrate this bundle in the ODL controller which provides JSON parser and a RESTful API for the YANG RPCs. The datastore is automatically created and managed by ODL as well. The only manual code needed are the bundle activators and the server computation implementation.

5. Security Considerations

This document does not introduce security or privacy concerns.

6. IANA Considerations

This document does not have IANA considerations.

7. References

[RFC7159] Bray, T., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, March 2014.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, June 2010.

[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.

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

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6020, October 2010.

[draft-ietf-netmod-yang-json-01]
Lhotka, L., "JSON Encoding of Data Modeled with YANG", October 2014.

[draft-shi-alto-yang-json-00]
Shi, X. and Y. Yang, "Modeling JSON Messages Using YANG", October 2014.

[RESTCONF]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", October 2014.

Appendix A. YANG Data Model for ALTO Protocol

A.1. Revision 1: RPC Only

```
module alto-service-rpc {
  yang-version 1;

  namespace "urn:ietf:params:xml:ns:yang:alto-service";
  // TODO: replace with IANA namespace when assigned

  prefix "as";

  import ietf-inet-types {
    prefix inet;
  }

  organization "ALTO WG";
  contact "alto@ietf.org";

  description
    "This module defines a semantically equivalent data model
     for the ALTO services defined in RFC7285.";

  /* Potential issue for future EXTENSION: YANG 1.0 is unable to
   * augment a typedef or enum type. e.g., there is not a good way to
   * add a cost metric or type of address if this document is
   * fixed. We have marked such places with "EXTENSION." */

  revision 2014-10-24 {
    description "Initial revision: RPC only.";
  }

  /*******************
   * 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 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 string {
        length "1..64";
        pattern "[!-~]+";
      }
      mandatory true;
      description
        "Tag. RFC7285 Sec. 10.3. U+0021-U+007E";
    }
    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 {
      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;";
  }

  /***********************************
   * RPCs                            *
   ***********************************/

  rpc IRD-service {
    output {
      container IRD-service {
        uses IRD;
      }
    }
  }

  rpc network-map-service {
    input {
      leaf uri {
        type inet:uri;
        mandatory true;
        description
          "The uri of the request for the full network map.";
      }
    }
    output {
      container network-map-service {
        uses network-map;
      }
    }
  }

  rpc cost-map-service {
    input {
      leaf uri {
        type inet:uri;
        mandatory true;
        description
          "The uri of the request for the full network map.";
      }
    }
    output {
```

```
      container cost-map-service {
        uses cost-map;
      }
    }
  }

  rpc filtered-network-map-service {
    description
      "inquiries on filtered network map" +
      "ReqFilteredNetworkMap. RFC7285 Sec. 11.3.1.3." +
        " object {
            PIDName pids<0..*>;
            [AddressType address-types<0..*>;]
          } ReqFilteredNetworkMap;";
    input {
      leaf-list pids {
        must "current()";
        type pid-name;
      }
      leaf-list address-types {
        type endpoint-address-type;
      }
    }
    output {
      container filtered-network-map-service {
        uses network-map;
      }
    }
  }

  rpc filtered-cost-map-service {
    input {
      container cost-type {
        must "current()";
        uses cost-type;
      }
      leaf-list constraints {
        type constraint;
        description
          "RFC7285 Sec. 11.3.2.3.";
      }
      container pids {
        leaf-list srcs {
          type pid-name;
          description
            "Source endpoint addresses.";
        }
        leaf-list dsts {
```

```
          type pid-name;
          description
            "Destination endpoint addresses.";
        }
        description
          "PIDFilter: Endpoint addresses. RFC7285 Sec. 11.3.2.3." +
          " object {
              PIDName srcs<0..*>;
              PIDName dsts<0..*>;
            } PIDFilter;";
      }
    }
    output {
      container filtered-cost-map-service {
        uses cost-map;
      }
    }
  }

  rpc endpoint-property-service {
    description
      "inquiries on properties of an endpoint" +
      " object {
            EndpointPropertyType  properties<1..*>;
            TypedEndpointAddr     endpoints<1..*>;
          } ReqEndpointProp;";
    input {
      leaf-list properties {
        type endpoint-property-type;
        min-elements 1;
      }
      leaf-list endpoints {
        type typed-endpoint-address;
        min-elements 1;
      }
    }
    output {
      container endpoint-property-service {
        uses endpoint-property-map;
      }
    }
  }


  rpc endpoint-cost-service {
    description
      "ReqEndpointCostMap. RFC7285 Sec. 11.5.1.3." +
      " object {
```

```
          CostType          cost-type;
          [JSONString       constraints<0..*>;]
          EndpointFilter    endpoints;
        } ReqEndpointCostMap;";
    input {
      container cost-type {
        must "current()";
        uses cost-type;
      }
      leaf-list constraints {
        type constraint;
        description
          "RFC7285 Sec. 11.5.1.3.";
      }
      container endpoints {
        must "current()";
        leaf-list srcs {
          type typed-endpoint-address;
          description
            "Source endpoint addresses.";
        }
        leaf-list dsts {
          type typed-endpoint-address;
          description
            "Destination endpoint addresses.";
        }
        description
          " EndpointFilter: Endpoint addr. RFC7285 Sec. 11.5.1.3." +
          " object {
              [TypedEndpointAddr srcs<0..*>;]
              [TypedEndpointAddr dsts<0..*>;]
            } EndpointFilter;";
      }
    }
    output {
      container endpoint-cost-service {
        container meta {
          container cost-type {
            uses cost-type;
          }
        }
        list endpoint-cost-map {
          leaf src {
            type typed-endpoint-address;
            description
              "Source endpoint address.";
          }
          key "src";
```

```
          list dst-costs {
            leaf dst {
              type typed-endpoint-address;
              description
                "Destination endpoint address.";
            }
            key "dst";
            uses alto-cost {
              description
                "Cost from source to destination.";
            }
            description
              "The list represents the inner part of the cost matrix." +
              "EndpointDstCosts. RFC7285 Sec. 11.5.1.6." +
              " object-map {
                  TypedEndpointAddr -> JSONValue;
                } EndpointDstCosts;";
          }
          description
            "The list represents the outer part of the cost matrix." +
            "EndpointCostMapData. RFC7285 Sec. 11.5.1.6." +
            " object {
                EndpointCostMapData endpoint-cost-map;
              } InfoResourceEndpointCostMap : ResponseEntityBase;
              object-map {
                TypedEndpointAddr -> EndpointDstCosts;
              } EndpointCostMapData;";
        }
      }
    }
  }

}
```

A.2. Revision 2: Custom Data Instances

```
module alto-service-did {
  yang-version 1;

  namespace "urn:ietf:params:xml:ns:yang:alto-service";
  // TODO: replace with IANA namespace when assigned

  prefix "as";

  import ietf-inet-types {
    prefix inet;
  }
```

```
  organization "ALTO WG";
  contact "alto@ietf.org";

  description
    "This module defines a semantically equivalent data model
     for the ALTO services defined in RFC7285.";

  /* Potential issue for future EXTENSION: YANG 1.0 is unable to
   * augment a typedef or enum type. e.g., there is not a good way to
   * add a cost metric or type of address if this document is
   * fixed. We have marked such places with "EXTENSION." */

  revision 2014-10-27 {
    description "Revision 2: Custom Data Instances";
  }

  revision 2014-10-24 {
    description "Initial revision: RPC only.";
  }

  /********************
   * 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 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 string {
        length "1..64";
        pattern "[!-~]+";
      }
      mandatory true;
      description
        "Tag. RFC7285 Sec. 10.3. U+0021-U+007E";
    }
    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 {
      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 leafref {
        path "/resources/IRD/resources/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 leafref {
        path "/resources/IRD/resources/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;";
  }
```

```
  /*************************************************
     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 {
    config false;

    /* Information Resource Directory */
    container IRD {
      uses IRD;
    }

    list network-maps {
      unique "meta/vtag/resource-id meta/vtag/tag";
      uses network-map;
    }

    list cost-maps {
      /*
        The dependent-vtags and cost-types in the meta uniquely
        identify a cost map. However, the unique statement in YANG
        does not allow reference to a list (dependent-vtags).
      */
      uses cost-map;
    }

    container endpoint-property-map {
      uses endpoint-property-map;
    }
  }


  /***********************************
   * RPCs                            *
   ***********************************/

  rpc IRD-service {
    output {
      container IRD-service {
        uses IRD;
      }
    }
```

```
  }

  rpc network-map-service {
    input {
      leaf uri {
        type leafref {
          path "/resources/IRD/resources/uri";
        }
        mandatory true;
        description
          "The uri of the request for the full network map.";
      }
    }
    output {
      container network-map-service {
        uses network-map;
      }
    }
  }

  rpc cost-map-service {
    input {
      leaf uri {
        type leafref {
          path "/resources/IRD/resources/uri";
        }
        mandatory true;
        description
          "The uri of the request for the full network map.";
      }
    }
    output {
      container cost-map-service {
        uses cost-map;
      }
    }
  }

  rpc filtered-network-map-service {
    description
      "inquiries on filtered network map" +
      "ReqFilteredNetworkMap. RFC7285 Sec. 11.3.1.3." +
        " object {
            PIDName pids<0..*>;
            [AddressType address-types<0..*>;]
          } ReqFilteredNetworkMap;";
    input {
      leaf-list pids {
```

```
        must "current()";
        type pid-name;
      }
      leaf-list address-types {
        type endpoint-address-type;
      }
    }
    output {
      container filtered-network-map-service {
        uses network-map;
      }
    }
  }

  rpc filtered-cost-map-service {
    input {
      container cost-type {
        must "current()";
        uses cost-type;
      }
      leaf-list constraints {
        type constraint;
        description
          "RFC7285 Sec. 11.3.2.3.";
      }
      container pids {
        leaf-list srcs {
          type pid-name;
          description
            "Source endpoint addresses.";
        }
        leaf-list dsts {
          type pid-name;
          description
            "Destination endpoint addresses.";
        }
        description
          "PIDFilter: Endpoint addresses. RFC7285 Sec. 11.3.2.3." +
          " object {
              PIDName srcs<0..*>;
              PIDName dsts<0..*>;
            } PIDFilter;";
      }
    }
    output {
      container filtered-cost-map-service {
        uses cost-map;
      }
```

```
    }
  }

  rpc endpoint-property-service {
    description
      "inquiries on properties of an endpoint" +
      " object {
            EndpointPropertyType  properties<1..*>;
            TypedEndpointAddr     endpoints<1..*>;
          } ReqEndpointProp;";
    input {
      leaf-list properties {
        type endpoint-property-type;
        min-elements 1;
      }
      leaf-list endpoints {
        type typed-endpoint-address;
        min-elements 1;
      }
    }
    output {
      container endpoint-property-service {
        uses endpoint-property-map;
      }
    }
  }


  rpc endpoint-cost-service {
    description
      "ReqEndpointCostMap. RFC7285 Sec. 11.5.1.3." +
      " object {
          CostType          cost-type;
          [JSONString       constraints<0..*>;]
          EndpointFilter    endpoints;
        } ReqEndpointCostMap;";
    input {
      container cost-type {
        must "current()";
        uses cost-type;
      }
      leaf-list constraints {
        type constraint;
        description
          "RFC7285 Sec. 11.5.1.3.";
      }
      container endpoints {
        must "current()";
```

```
        leaf-list srcs {
          type typed-endpoint-address;
          description
            "Source endpoint addresses.";
        }
        leaf-list dsts {
          type typed-endpoint-address;
          description
            "Destination endpoint addresses.";
        }
        description
          " EndpointFilter: Endpoint addr. RFC7285 Sec. 11.5.1.3." +
          " object {
              [TypedEndpointAddr srcs<0..*>;]
              [TypedEndpointAddr dsts<0..*>;]
            } EndpointFilter;";
      }
    }
    output {
      container endpoint-cost-service {
        container meta {
          container cost-type {
            uses cost-type;
          }
        }
        list endpoint-cost-map {
          leaf src {
            type typed-endpoint-address;
            description
              "Source endpoint address.";
          }
          key "src";
          list dst-costs {
            leaf dst {
              type typed-endpoint-address;
              description
                "Destination endpoint address.";
            }
            key "dst";
            uses alto-cost {
              description
                "Cost from source to destination.";
            }
            description
              "The list represents the inner part of the cost matrix." +
              "EndpointDstCosts. RFC7285 Sec. 11.5.1.6." +
              " object-map {
                  TypedEndpointAddr -> JSONValue;
```

```
                } EndpointDstCosts;";
          }
          description
            "The list represents the outer part of the cost matrix." +
            "EndpointCostMapData. RFC7285 Sec. 11.5.1.6." +
            " object {
                EndpointCostMapData endpoint-cost-map;
              } InfoResourceEndpointCostMap : ResponseEntityBase;
              object-map {
                TypedEndpointAddr -> EndpointDstCosts;
              } EndpointCostMapData;";
        }
      }
    }
  }

}
```

Appendix B. YANG-Validated JSON Messages for ALTO Examples

We established that the YANG validated messages cannot be syntactically equivalent to the ALTO protocol messages. For a selection of the examples in the ALTO protocol, we provide the YANG validated version of the message for comparison purposes.

B.1. IRD Response Example

The ALTO example of IRD response as specified in Section 9.2.3 of [RFC7285].

```
{
  "rpc-reply": {
    "-xmlns": "urn:ietf:params:xml:ns:netconf:base:1.0",
    "-message-id": "1",
    "IRD-service": {
      "-xmlns": "urn:ietf:params:xml:ns:yang:alto-service",
      "meta": {
        "cost-types": [
          {
            "cost-type-name": "num-routing",
            "cost-mode": "numerical",
            "cost-metric": "routingcost",
            "description": "My default"
          },
          {
            "cost-type-name": "num-hop",
            "cost-mode": "numerical",
```

```
            "cost-metric": "hopcount"
          },
          {
            "cost-type-name": "ord-routing",
            "cost-mode": "ordinal",
            "cost-metric": "routingcost"
          },
          {
            "cost-type-name": "ord-hop",
            "cost-mode": "ordinal",
            "cost-metric": "hopcount"
          }
        ],
        "default-alto-network-map": "my-default-network-map"
      },
      "resources": [
        {
          "resource-id": "my-default-network-map",
          "uri": "http://alto.example.com/networkmap",
          "media-type": "application/alto-networkmap+json"
        },
        {
          "resource-id": "numerical-routing-cost-map",
          "uri": "http://alto.example.com/costmap/num/routingcost",
          "media-type": "application/alto-costmap+json",
          "capabilities": { "cost-type-names": [ "num-routing" ] },
          "uses": [ "my-default-network-map" ]
        },
        {
          "resource-id": "numerical-hopcount-cost-map",
          "uri": "http://alto.example.com/costmap/num/hopcount",
          "media-type": "application/alto-costmap+json",
          "capabilities": { "cost-type-names": [ "num-hop" ] },
          "uses": [ "my-default-network-map" ]
        },
        {
          "resource-id": "custom-maps-resources",
          "uri": "http://custom.alto.example.com/maps",
          "media-type": "application/alto-directory+json"
        },
        {
          "resource-id": "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"
              ]
            }
          },
          {
            "resource-id": "endpoint-cost",
            "uri": "http://alto.example.com/endpointcost/lookup",
            "media-type": "application/alto-endpointcost+json",
            "accepts": "application/alto-endpointcostparams+json",
            "capabilities": {
              "cost-constraints": "true",
              "cost-type-names": [
                "num-routing",
                "num-hop",
                "ord-routing",
                "ord-hop"
              ]
            }
          }
        ]
      }
    }
  }
```

The corresponding XML file which is validated by YANG.

```
<?xml version="1.0" encoding="UTF-8" ?>
  <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
    message-id="1">
    <IRD-service xmlns="urn:ietf:params:xml:ns:yang:alto-service">
      <meta>
        <cost-types>
          <cost-type-name>num-routing</cost-type-name>
          <cost-mode>numerical</cost-mode>
          <cost-metric>routingcost</cost-metric>
          <description>My default</description>
        </cost-types>
        <cost-types>
          <cost-type-name>num-hop</cost-type-name>
          <cost-mode>numerical</cost-mode>
          <cost-metric>hopcount</cost-metric>
        </cost-types>
        <cost-types>
          <cost-type-name>ord-routing</cost-type-name>
          <cost-mode>ordinal</cost-mode>
          <cost-metric>routingcost</cost-metric>
        </cost-types>
        <cost-types>
```

```
            <cost-type-name>ord-hop</cost-type-name>
            <cost-mode>ordinal</cost-mode>
            <cost-metric>hopcount</cost-metric>
          </cost-types>
          <default-alto-network-map>
            my-default-network-map
          </default-alto-network-map>
        </meta>
        <resources>
          <resource-id>my-default-network-map</resource-id>
          <uri>http://alto.example.com/networkmap</uri>
          <media-type>application/alto-networkmap+json</media-type>
        </resources>
        <resources>
          <resource-id>numerical-routing-cost-map</resource-id>
          <uri>http://alto.example.com/costmap/num/routingcost</uri>
          <media-type>application/alto-costmap+json</media-type>
          <capabilities>
            <cost-type-names>num-routing</cost-type-names>
          </capabilities>
          <uses>my-default-network-map</uses>
        </resources>
        <resources>
          <resource-id>numerical-hopcount-cost-map</resource-id>
          <uri>http://alto.example.com/costmap/num/hopcount</uri>
          <media-type>application/alto-costmap+json</media-type>
          <capabilities>
            <cost-type-names>num-hop</cost-type-names>
          </capabilities>
          <uses>my-default-network-map</uses>
        </resources>
        <resources>
          <resource-id>custom-maps-resources</resource-id>
          <uri>http://custom.alto.example.com/maps</uri>
          <media-type>application/alto-directory+json</media-type>
        </resources>
        <resources>
          <resource-id>endpoint-property</resource-id>
          <uri>http://alto.example.com/endpointprop/lookup</uri>
          <media-type>application/alto-endpointprop+json</media-type>
          <accepts>application/alto-endpointpropparams+json</accepts>
          <capabilities>
            <prop-types>my-default-network-map.pid</prop-types>
            <prop-types>priv:ietf-example-prop</prop-types>
          </capabilities>
        </resources>
        <resources>
          <resource-id>endpoint-cost</resource-id>
```

```
          <uri>http://alto.example.com/endpointcost/lookup</uri>
          <media-type>application/alto-endpointcost+json</media-type>
          <accepts>application/alto-endpointcostparams+json</accepts>
          <capabilities>
            <cost-constraints>true</cost-constraints>
            <cost-type-names>num-routing</cost-type-names>
            <cost-type-names>num-hop</cost-type-names>
            <cost-type-names>ord-routing</cost-type-names>
            <cost-type-names>ord-hop</cost-type-names>
          </capabilities>
        </resources>
      </IRD-service>
    </rpc-reply>
```

B.2. Network Map Service Response Example

The ALTO example of a network map response as specified in Section 11.2.1.7 of [RFC7285].

```
{
  "rpc-reply": {
    "-xmlns": "urn:ietf:params:xml:ns:netconf:base:1.0",
    "-message-id": 2,
    "network-map-service": {
      "-xmlns": "urn:ietf:params:xml:ns:yang:alto-service",
      "media-type": "application/alto-networkmap+json",
      "meta": {
        "vtag": {
          "resource-id": "my-default-network-map",
          "tag": "da65eca2eb7a10ce8b059740b0b2e3f8eb1d4785"
        }
      },
      "network-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"]
            }
          ]
        }
      ]
    }
  }
}
```

The corresponding YANG-validated XML file:

```
<?xml version="1.0" encoding="UTF-8" ?>
  <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
    message-id="2">
    <network-map-service
      xmlns="urn:ietf:params:xml:ns:yang:alto-service">
      <meta>
        <vtag>
          <resource-id>my-default-network-map</resource-id>
          <tag>da65eca2eb7a10ce8b059740b0b2e3f8eb1d4785</tag>
        </vtag>
      </meta>
      <network-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>
      </network-map>
      <network-map>
        <pid>PID2</pid>
        <endpoint-address-group>
          <address-type>ipv4</address-type>
          <endpoint-prefix>198.51.100.128/25</endpoint-prefix>
        </endpoint-address-group>
      </network-map>
      <network-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>
      </network-map>
    </network-map-service>
  </rpc-reply>
```

B.3. Filtered Cost Map Response Example

The ALTO example of a filtered cost map response as specified in Section 11.3.2.7 of [RFC7285].

```
{
  "rpc-reply": {
    "-xmlns": "urn:ietf:params:xml:ns:netconf:base:1.0",
    "-message-id": 2,
    "filtered-cost-map-service": {
      "-xmlns": "urn:ietf:params:xml:ns:yang:alto-service",
      "meta": {
        "dependent-vtags": {
          "resource-id": "my-default-network-map",
          "tag": "75ed013b3cb58f896e839582504f622838ce670f"
        },
        "cost-type": {
          "cost-mode": "numerical",
          "cost-metric": "routingcost"
        }
      },
      "cost-map": [
        {
          "src": "PID1",
          "dst-costs": [
            {
              "dst": "PID1",
              "cost": "0"
            },
            {
              "dst": "PID2",
              "cost": "1"
            },
            {
              "dst": "PID3",
              "cost": "2"
            }
          ]
        }
      ]
    }
  }
}
```

The corresponding YANG-validated XML file:

```
<?xml version="1.0" encoding="UTF-8" ?>
  <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
    message-id="2">
    <filtered-cost-map-service
      xmlns="urn:ietf:params:xml:ns:yang:alto-service">
      <meta>
        <dependent-vtags>
          <resource-id>my-default-network-map</resource-id>
          <tag>75ed013b3cb58f896e839582504f622838ce670f</tag>
        </dependent-vtags>
        <cost-type>
          <cost-mode>numerical</cost-mode>
          <cost-metric>routingcost</cost-metric>
        </cost-type>
      </meta>
      <cost-map>
        <src>PID1</src>
        <dst-costs>
          <dst>PID1</dst>
          <cost>0</cost>
        </dst-costs>
        <dst-costs>
          <dst>PID2</dst>
          <cost>1</cost>
        </dst-costs>
        <dst-costs>
          <dst>PID3</dst>
          <cost>2</cost>
        </dst-costs>
      </cost-map>
    </filtered-cost-map-service>
  </rpc-reply>
```

B.4. Endpoint Property Service Response Example

The ALTO example of an endpoint property service response as specified in Section 11.4.1.7 of [RFC7285].

```
{
  "rpc-reply": {
    "-xmlns": "urn:ietf:params:xml:ns:netconf:base:1.0",
    "-message-id": "2",
    "endpoint-property-service": {
      "-xmlns": "urn:ietf:params:xml:ns:yang:alto-service",
      "meta" : {
        "dependent-vtags" : [
          {"resource-id": "my-default-network-map",
            "tag": "7915dc0290c2705481c491a2b4ffbec482b3cf62"
          }
        ]
      },
      "endpoint-properties": [
        {
          "endpoint": "ipv4:192.0.2.34",
          "properties" : [
            {
              "property-type": "my-default-network-map.pid",
              "property": "PID1"
            },
            {
              "property-type": "priv:ietf-example-prop",
              "property": "1"
            }
          ]
        },
        {
          "endpoint": "ipv4:203.0.113.129",
          "properties": [
            {
              "property-type": "my-default-network-map.pid",
              "property": "PID3"
            }
          ]
        }
      ]
    }
  }
}
```

The corresponding YANG-validated XML file:

```
<?xml version="1.0" encoding="UTF-8" ?>
  <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
    message-id="2">
    <endpoint-property-service
      xmlns="urn:ietf:params:xml:ns:yang:alto-service">
      <meta>
        <dependent-vtags>
          <resource-id>my-default-network-map</resource-id>
          <tag>7915dc0290c2705481c491a2b4ffbec482b3cf62</tag>
        </dependent-vtags>
      </meta>
      <endpoint-properties>
        <endpoint>ipv4:192.0.2.34</endpoint>
        <properties>
          <property-type>my-default-network-map.pid</property-type>
          <property>PID1</property>
        </properties>
        <properties>
          <property-type>priv:ietf-example-prop</property-type>
          <property>1</property>
        </properties>
      </endpoint-properties>
      <endpoint-properties>
        <endpoint>ipv4:203.0.113.129</endpoint>
        <properties>
          <property-type>my-default-network-map.pid</property-type>
          <property>PID3</property>
        </properties>
      </endpoint-properties>
    </endpoint-property-service>
  </rpc-reply>
```

Authors' Addresses

Xiao Shi
Yale University
51 Prospect Street
New Haven, CT 06511
USA

Email: xiao.shi@yale.edu

Y. Richard Yang
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: April 30, 2015

G. Bernstein
Grotto Networking
Y. Lee
Huawei
W. Roome
M. Scharf
Alcatel-Lucent
Y. Yang
Yale University
October 27, 2014

ALTO Topology Extensions
draft-yang-alto-topology-05.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 graph representations of network topology. We start with a basic application use case of multi-flow scheduling using ALTO. We show that ALTO cost maps alone cannot provide sufficient information. We then define one key, generic component to address the issues: introducing path vectors in cost maps. We specify two approaches to complement path vectors and achieve a complete design: an approach using opaque network elements and another using a graph (node-link) representation.

Requirements Language

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

Status of This Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any

time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 30, 2015.

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 Section 2) defined in this document.

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 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. Then in Section 3, we give the multi-flow scheduling use case as an example. In Section 4, we specify path vector as a key component to handle multi-flow

scheduling. In Sections 5 and 6, we give two graph representations to complete the design. Section 7 gives a framework of topology transformations to help with the understanding of deriving multiple representations of the topology of the same network infrastructure, for applications.

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.

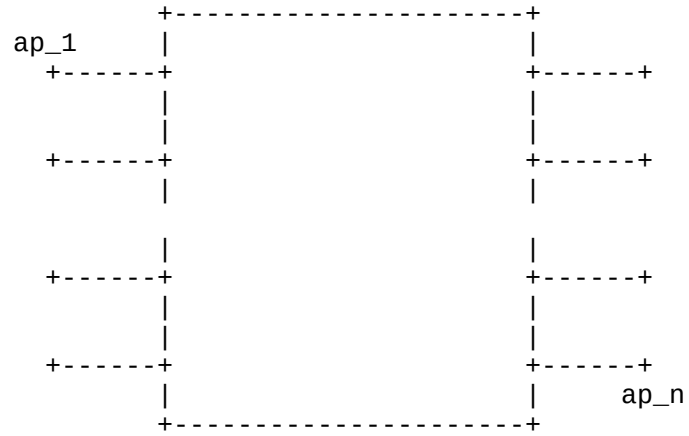


Figure 1: Base Single-Node Topology Abstraction.

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. The Multi-flow Scheduling Use Case

There are use cases where simple cost metrics cannot 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 idea.

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. A simple cost metric such as 'available bw' for eh1 -> eh2 and eh3 -> eh4 may not reflect whether the two paths for eh1 -> eh2 and eh3 -> eh4 share a bottleneck.

More concretely, assume that 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.

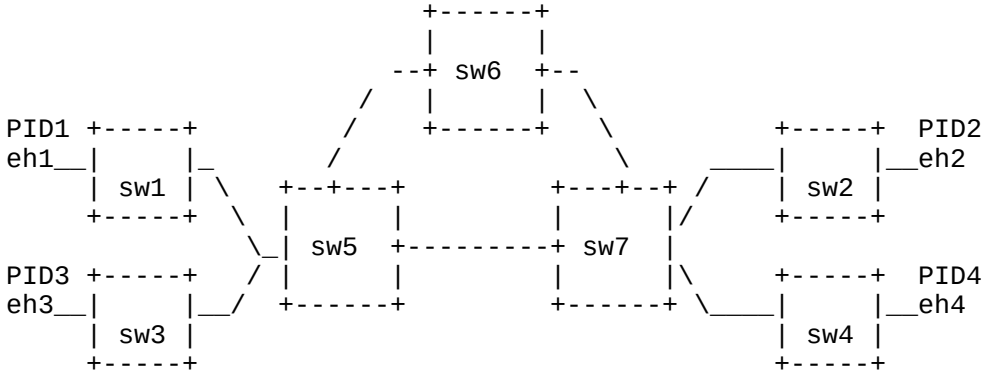


Figure 2: Base Single-Node Topology Abstraction.

Now, consider a cost map providing end-to-end available bandwidth. There can be two possible interpretations on the semantics of the value of PIDi -> PIDj reported by the cost map: (1) it represents reserved bandwidth from PIDi -> PIDj, or (2) it represents possible

bandwidth for PIDi -> PIDj, if no other applications use shared resources. The common understanding is (2), just as when we look at the number of available seats on a flight.

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 flows share a bottleneck:

- 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.

- 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.

4. Path-Vector as Cost Metric Representation

A key component to address the problem in the preceding section is to introduce path vectors as a cost metric, which is a set of path vectors from a source PID to a destination PID, where each path vector is a sequence (array) of network elements. Note that this design does not specify that a path vector is a sequence of network links. Rather, as a general design, a path is a sequence of network elements.

A schema for introducing path vectors in cost maps is the 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-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":[]}
  }
}
```

The example illustrates that there are two key extensions to the ALTO base protocol:

o It introduces a new "cost-mode" named "path-vector";

- To indicate the resource that provides information on the elements of path vectors (e.g., ["ne5", "ne67"] for the path vector from PID1 to PID2, it introduces a new dependency. In the example, it is indicated by a resource named "my-topology-map".

5. Minimal Topology through 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
  }
}
```

An advantage of the representation is that it does not need to distinguish between network nodes vs network links, as an application in typical cases do not need to make the distinction between network nodes and network links. At the same time, the design introduces an optional "type" field, which can indicate the type (e.g., link, layer 2 switch, layer 3 router), of the network element.

6. Topology using a Graph (Node-Link) Representation

6.1. Use Case: Compact Representation

A potential problem of the path vector representation 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.

6.2. Use Case: Application Path Selection

Another setting where a node-link graph approach is more complete (than the partial NEP approach) can be motivated by the multi-flow scheduling use case discussed in Section 3. In particular, consider

that the network routing is Case 2 (only 100 Mbps total bandwidth), and the application can benefit from the routing in Case 1 (200 Mbps). With a topology graph, the 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.

6.3. 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;
```

An example using the schema:

```
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",
                   ...
          }
        }
      }
    }
```

6.4. 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 such as our topology in Section 3. 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.

7. Security Considerations

This document has not conducted its security analysis.

8. IANA Considerations

This document does not specified its IANA considerations, yet.

9. Acknowledgments

The author thanks discussions with Xiao Shi, Xin Wang, Erran Li, Tianyuan Liu, Andreas Voellmy, Haibin Song, and Yan Luo.

10. References

10.1. Normative References

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

10.2. 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):

- O1: Deletion of a switch/port/link from G(n-1);

- 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;

- 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;

- 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