Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 15, 2016

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

June 13, 2016

Multi-Cost ALTO
draft-ietf-alto-multi-cost-02

Abstract

The ALTO (Application Layer-Traffic Optimization) Protocol ([RFC7285]) defines several services that return various metrics describing the costs between network endpoints. For example, when downloading a file that is mirrored on several sites, a user application may use these ALTO cost metrics to determine the most efficient mirror site.

An ALTO Server may offer a variety of cost metrics, based on latency, bandwidth, hop count, jitter, or whatever else the ALTO Server deems useful. When selecting a mirror site, a client may consider more than one metric, perhaps trading bandwidth for latency. While the base ALTO Protocol allows a client to use more than one cost metric, to do so, the client must request each metric separately. This document defines a new service that allows a client to retrieve several cost metrics with one request, which is considerably more efficient. In addition, this document extends the ALTO constraint tests to allow a user to specify an arbitrary logical combination of tests on several cost metrics.

Requirements Language

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

Status of This Memo

Table of Contents

1. Introduction

IETF has designed a new service called ALTO that provides guidance to overlay applications, which have to select one or several hosts from a set of candidates that are able to provide a desired resource. This guidance is based on parameters such as the topological distance, that affect performance and efficiency of the data transmission between the hosts. The purpose of ALTO is to improve Quality of Experience (QoE) in the application while reducing resource consumption in the underlying network infrastructure. The ALTO protocol conveys the Internet View from the perspective of a Provider Network region that spans from a region to one or more Autonomous System (AS) and is called a Network Map. ALTO may also provide the Provider determined Cost Map between locations of the Network Map or Endpoint Cost Map between groups of individual endpoints. Last, these costs are provided as numerical or ordinal values.

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

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

The ALTO protocol [RFC7285] restricts ALTO Cost Maps and Endpoint Cost services to only one Cost Type and Cost Mode per ALTO request.

To retrieve information for several Cost Types, an ALTO client must send several separate requests to the server.

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

Along with multi-cost values queries, the filtering capabilities need to be extended to allow constraints on multiple metrics. The base protocol allows a client to provide optional constraint tests for a Filtered Cost Map or the Endpoint Cost Service. In the base protocol, the constraint tests are limited to the AND-combination of simple comparison tests on the value of the (single) requested Cost Type. It is therefore necessary to allow constraints on multiple metrics. Beyond that, applications that are sensitive to several metrics and struggle with complicated network conditions may need to arbitrate between conflicting objectives such as routing cost and network performance. To address this issue, this document proposes to extend the base protocol by extending constraints to test multiple metrics, and by allowing these constraints to be combined with logical 'ORs' as well as logical 'ANDs'. This allows an application to make requests such as: "select solutions with either (moderate "hopcount" AND high "routingcost") OR (higher "hopcount" AND moderate "routingcost")". To ensure compatibility with legacy ALTO Clients, only the Filtered Cost Map and Endpoint Cost Map services are extended to return Multi-Cost values. Full Cost Map services remain unchanged, and are restricted to returning single cost values.

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

2. Terminology

- {1.2.3}: References with curly brackets are to sections in the ALTO protocol specification [RFC7285].

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

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

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

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

- Application Client (AC): This term generalizes the case of a P2P client to include the case of a CDN client, a client of an application running on a virtual server, a Grid application client, or any Client that can choose between several connection points for data or resource exchange.

3. Overview Of Approach

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

3.1. Multi-Cost Data Format

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

Therefore this document extends the definition of a Cost Map to allow a cost to be an array of costs, one per metric, instead of just one number. For example, here is a Cost Map with the "routingcost" and "hopcount" metrics. Note that this is identical to a regular ALTO Cost Map, except that the values are arrays instead of numbers.

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

3.2. Compatibility With Legacy Clients

The multi-cost extensions defined in this document must not break legacy implementations (that is, clients and servers which are not aware of these extensions). One way to achieve that would be to define a new media type for an array-valued Multi Cost Map. However, as indicated above, an array-valued Multi Cost Map is almost identical to a single-valued Cost Map, so it should be simple to write a parser which handles either type of cost map. Hence defining a new media type could result in a lot of wasteful duplication.

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

3.3. Filtered Multi Cost Map Resources

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

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

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

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

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

3.4. Endpoint Cost Service Resources

This document uses the technique described in Section 3.3 to extend the Endpoint Cost Service to return array-valued costs to clients who also are aware of these extensions.

3.5. Full Cost Map Resources

Full Cost Map resources are GET-mode requests, with no capabilities other than the name of the cost type they return. Therefore unless we create a new media type for array-valued Cost Maps, it is not possible to define a Multi-Cost Full Cost Map resource so that multi-cost-aware clients can recognize it and legacy clients will ignore it. Indeed, the response for a Full Cost Map conveying multiple cost types would include a "meta" field that would itself include a "cost-type" field, that would list several values corresponding to the cost types of the cost map. A legacy client would not be able to understand this list. It would not know what the cost type of the map is and neither would it be able to interpret the cost values array provided by a Multi-Cost full maps.

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

3.6. Extended Constraint Tests

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

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

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

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

3.6.1. Extended constraint predicates

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

3.6.2. Extended logical combination of predicates

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

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

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

3.6.3. Testable Cost Types in constraints

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

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

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

In this example, "[0]" means the constraint applies to "hopcount" because that is the first cost type in the "testable-cost-types" parameter. (If "testable-cost-types" is omitted, it is assumed to be the same as "multi-cost-types".)

3.6.4. Testable Cost Type Names in IRD capabilities

In [RFC7285], when a resource's capability "constraints" is true, the server accepts constraints on all the cost types listed in the "cost-type-names" capability. However, some ALTO Servers may not be willing to allow contraint tests on all available cost metrics. Therefore the Multi-Cost ALTO protocol extension defines the capability field "testable-cost-type-names". Like "cost-type-names",

it is an array of cost type names. If present, that resource only allows constraint tests on the cost types in that list. "testable-cost-type-names" MUST be a subset of "cost-type-names".

3.6.5. Legacy client issues

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

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

In the IRD example below, the resource "filtered-cost-map-extended" provides values for three metrics: "num-routingcost", "num-hopcount" and "num-bwscore". The capability "testable-cost-type-names" indicates that the server only allows constraints on "routingcost" and "hopcount". A multi-cost capable client will see this capability, and will limit its constraint tests to those metrics. Because capability "cost-constraints" is false (by default), a legacy client will not use constraint tests on this resource at all.

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

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

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

4. Protocol Extensions for Multi-Cost ALTO Transactions

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

4.1. Filtered Cost Map Extensions

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

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

4.1.1. Capabilities

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

o max-cost-types,

o testable-cost-type-names

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

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

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

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

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

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

4.1.2. Accept Input Parameters

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

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

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

cost-type: As defined in {11.3.2.3} of [RFC7285], with the additional requirement that the client MUST specify either "cost-type" or "multi-cost-types", but MUST NOT specify both.

multi-cost-types: If present, the ALTO Server MUST return array-valued costs for the cost types in this list. For each entry, the "cost-metric" and "cost-mode" fields MUST match one of the supported cost types indicated in member "cost-type-names" of this resource's "capabilities" field (Section 4.1.1). The client MUST NOT use this field unless this resource's "max-cost-types" capability exists and has a value greater than 0. This field MUST NOT have more than "max-cost-types" cost types. The client MUST specify either "cost-type" or "multi-cost-types", but MUST NOT specify both.

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

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

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

This feature is useful when a client wants to test a cost type whose actual value is irrelevant, as long as it satisfies the tests. For example, a client may want the cost metric

"routingcost" for those PID pairs whose "hopcount" is less than 10. The exact hopcount does not matter.

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

If this resource's "max-cost-types" capability has a value greater than 0, then a "constraints" parameter with the array of extended predicates [P1, P2, ...] is equivalent to an "or-constraints" parameter with the value [[P1, P2, ...]]. In this case, the "constraints" parameter MUST NOT be specified if the "or-constraints" parameter is specified.

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

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

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

An extended constraint predicate consists of two or three entities separated by white space: (1) an optional cost type index, of the form "[#]", with default value "[0]", (2) a required operator, and (3) a required target value. The operator and target value are as defined in {11.3.2.3} of [RFC7285]. The cost type index, i, specifies the cost type to test. If the "testable-cost-type" parameter is present, the test applies to the i'th cost type in "testable-cost-types", starting with index 0. Otherwise if the "multi-cost-types" parameter is present, the test applies to the i'th cost type in that array. If neither parameters are present, the test applies to the cost type in the "cost-type" parameter, in which this case the index MUST be 0. Regardless of how the tested cost type is selected, it MUST be in the resource's "testable-cost-type-names" capability, or, if not present, in the "cost-type-names" capability.

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

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

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

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

4.1.3. Response

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

- o In "meta", the field "cost-type" is replaced with the field "multi-cost-types", with the same value as the "multi-cost-types" input parameter.

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

4.2. Endpoint Cost Service Extensions

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

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

4.2.1. Capabilities

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

4.2.2. Accept Input Parameters

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

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

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

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

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

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

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

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

4.2.3. Response

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

- o In "meta", the field "cost-type" is replaced with the field "multi-cost-types", with the same value as the "multi-cost-types" input parameter.

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

5. Examples

5.1. Information Resource Directory

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

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


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

{
  "meta" : {
    "default-alto-network-map" : "my-default-network-map",
    "cost-types" : {
       "num-routing" : {
         "cost-mode" : "numerical",
         "cost-metric" : "routingcost"
       },
```

```
          "num-hopcount" : {
            "cost-mode" : "numerical",
            "cost-metric" : "hopcount"
          },
          "num-bwscore" : {
            "cost-mode" : "numerical",
            "cost-metric" : "bandwidthscore"
          },
            .....
            Other ALTO cost types as described in RFC7285
            .....
          }
     },
     "resources" : {
       "my-default-network-map" : {
         "uri" : "http://alto.example.com/networkmap",
         "media-type" : "application/alto-networkmap+json"
       },
       "numerical-routing-cost-map" : {
         "uri" : "http://alto.example.com/costmap/num-routing",
         "media-types" : [ "application/alto-costmap+json" ],
         "uses" : [ "my-default-network-map" ],
         "capabilities" : {
           "cost-type-names" : [ "num-routing" ]
         }
       },
       "numerical-hopcount-cost-map" : {
         "uri" : "http://alto.example.com/costmap/num-hopcount",
         "media-types" : [ "application/alto-costmap+json" ],
         "uses" : [ "my-default-network-map" ],
         "capabilities" : {
           "cost-type-names" : [ "num-hopcount" ]
         }
       },
       .........
       Other resources as described in RFC7285
       .........
       "filtered-multicost-map" : {
         "uri" : "http://alto.example.com/multi/costmap/filtered",
         "media-types" : [ "application/alto-costmap+json" ],
         "accepts" : [ "application/alto-costmapfilter+json" ],
         "uses" : [ "my-default-network-map" ],
         "capabilities" : {
           "cost-constraints" : true,
           "max-cost-types" : 2,
           "cost-type-names" : [ "num-routingcost",
                                 "num-hopcount" ]
         }
```

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

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

This example illustrates a simple multi-cost ALTO transaction. The ALTO Server provides two Cost Types, "routingcost" and "hopcount", both in "numerical" mode. The ALTO Server does not know the value of the "routingcost" between PID2 and PID3, and hence uses "null" for those costs.

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

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


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

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

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

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

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

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


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

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

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

This example uses extended constraints to limit the response to cost points with ("routingcost" <= 10 and "hopcount" <= 2), or else ("routingcost" <= 3 and "hopcount" <= 6). Unlike the previous

example, the client is only interested in the "routingcost" cost type, and uses the "cost-type" parameter instead of "multi-cost-types" to tell the server to return scalar costs instead of array costs:

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

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

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

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

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

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

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

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


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

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

5.6. Endpoint Cost Service

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

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

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


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

{
  "meta" : {
    "multi-cost-types" : [
      {"cost-mode": "numerical", "cost-metric": "routingcost"},
      {"cost-mode": "numerical", "cost-metric": "hopcount"}
    ]
  }
  "endpoint-cost-map" : {
```

```
       "ipv4:192.0.2.2": {
         "ipv4:192.0.2.89":    [15, 5],
         "ipv4:203.0.113.45":  [4, 23]
       }
       "ipv6:2001:db8::1:0": {
         "ipv4:198.51.100.34": [16, 5],
         "ipv6:2001:db8::10":  [10, 2]
       }
     }
   }
```

6. IANA Considerations

This document does define any new media types or introduce any new IANA considerations.

7. Privacy And Security Considerations

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

8. Acknowledgements

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

9. References

9.1. Normative References

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

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

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

9.2. Informative References

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

Authors' Addresses

Sabine Randriamasy
Nokia Bell Labs
Route de Villejust
NOZAY 91460
FRANCE

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

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

Phone: +1-908-582-7974
Email: w.roome@nokia-bell-labs.com

Nico Schwan
Thales Deutschland
Lorenzstrasse 10
Stuttgart 70435
Germany

Email: nico.schwan@thalesgroup.com