ALTO WG
Internet-Draft
Intended status: Standards Track
Expires: September 14, 2017

K. Gao
Tsinghua University
J. Zhang
J. Wang
Tongji University
Q. Xiang
Tongji/Yale University
Y. Yang
Yale University
March 13, 2017

ALTO Extension: Flow-based Cost Query
draft-gao-alto-fcs-01.txt

Abstract

The emergence of new networking datapath capabilities has substantially increased the flexibility of networking. For example, OpenFlow has emerged as a major southbound protocol for Software-Defined Networking, and OpenFlow allows flexible routing beyond simple destination-based routing. In this document, we define a new extention to ALTO, namely the Flow Cost Service, for ALTO clients in an OpenFlow-enabled network to query ALTO network information.

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 September 14, 2017.

Table of Contents

1. Introduction 3
2. Changes Since Version -00 4
3. ALTO Flow Cost Specification: Basic Flow-based Query 4
3.1. Flow-based Filtered Cost Map 4
3.1.1. Capabilities 4
3.1.2. Accept Input Parameters 5
3.1.3. Response 6
3.2. Extend Endpoint Abstraction 6
3.3. Flow-based Endpoint Cost Service 7
3.3.1. Capabilities 7
3.3.2. Accept Input Parameters 7
3.3.3. Response 8
3.4. Example 8
3.4.1. IRD Example 8
3.4.2. Flow-based Filtered Cost Map Service Example 10
3.4.3. Flow-based Endpoint Cost Service Example 11
4. ALTO Flow Cost Specification: Advanced Flow-based Query . . . 12
4.1. Basic Data Types 12
4.1.1. Flow ID 12
4.1.2. Typed Header Field 12
4.1.3. Cost Confidence 12
4.2. Flow Cost Service 13
4.2.1. Media Type 13
4.2.2. HTTP Method 13
4.2.3. Accept Input Parameters 13
4.2.4. Capabilities 14
4.2.5. Response 15
4.2.6. Errors 16
4.3. Advanced Flow-based Query Example 17
5. Security Considerations 19
6. IANA Considerations 19

6.1. Media Types 19
6.2. Header Field 20
7. Acknowledgement 20
8. References 21
8.1. Normative References 21
8.2. Informative References 21
8.3. URIs 21
Appendix A. Tables 22
Authors' Addresses 23

1. Introduction

ALTO is now being considered as a solution for more flexible network use cases. The legacy ALTO services defined in [RFC7285] just try to provide the cost information for peer selection. It is enough for the P2P domain. But the network is becoming more and more flexible nowadays. There are two major changes in the coming network evolution:

- Some new network architectures such as SDN are adopting the logically central control solution. It makes the network optimization toward the higher-level view. The traffic optimizer can not only decide the source or the destination of the data transferring, but also make the flow-level traffic scheduling. To solve the flow-level scheduling problem, the cross-product query schema will be redundant.

- With the emerging technologies in the data plane, where multiple header fields can be used to determine the forwarding path, networks are moving to more flexible routing mechanisms beyond the simple destination-based routing. As a consequence, the endpoint cost service (ECS), which depends on only source and destination IP addresses as currently defined, is no longer sufficient to provide accurate cost information.

This document intents to address the following issues in providing fine-grained flow-based endpoint cost query services: 1. The compatibility with the legacy ALTO ECS service; 2. The support for emerging network architectures such as Software Defined Networking; 3. The trade-off between fine-grained queries and query/response redundancy.

In this document, we consider the extensions of ALTO service which provide the flow-based cost query. The basic solution is to extend the legacy ALTO services to support flow-based query schema. Section 3 describes the extended schema on Filtered Cost Map (FCM) and Endpoint Cost Service (ECS) to support endpoint cost queries of flows defined by the 5-tuple of protocol, src/dst name/address and

ports. For networks using a more generic flow concept such as Software-Defined Networks, Section 4 defines a novel ALTO service named the Flow Cost Service (FCS) with the flow-oriented query schema. It SHOULD support the query of any fine-grained routing cost to satisfy the growing demand of obtaining accurate costs in a network using flow-based routing. Section 5 and Section 6 discuss security and IANA considerations.

2. Changes Since Version -00

- Define the basic flow-based query extensions for Filtered Cost Map and Endpoint Cost service. The basic flow-based query is downward compatible with the legacy ALTO service. It does not introduce any new media types.

- Move the service of media-type "application/alto-flowcost+json" to the advanced flow-based query extension. It will ask ALTO server to support the new media type.

3. ALTO Flow Cost Specification: Basic Flow-based Query

This section describes a downward compatible extension for Filtered Cost Map and Endpoint Cost Service to support flow-based query.

3.1. Flow-based Filtered Cost Map

3.1.1. Capabilities

The Filtered Cost Map capabilities are extended with two new members:

- flow-based-filter

- traffic-types

The capability "flow-based-filter" indicates whether this resource supports flow-based query. The capability "traffic-type" list which traffic types are supported to be queried by this resource. The FilteredCostMapCapabilities object in Section 4.1.1 of [I-D.ietf-alto-multi-cost] is extended as follows:

```
object {
  JSONString cost-type-names<1..*>;
  [JSONBool cost-constraints;]
  [JSONNumber max-cost-types;]
  [JSONString testable-cost-type-names<1..*>;]
  [JSONBool flow-based-filter<0..*>;]
  [JSONString traffic-type<0..*>;]
} FilteredCostMapCapabilities;
```

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

max-cost-types and testable-cost-type-names: As defined in Section 4.1.1 of [I-D.ietf-alto-multi-cost].

flow-based-filter: If true, then the ALTO Server allows pid-flows to be queried in the requests. If not present, this field MUST be interpreted as if it is specified false.

traffic-type: If present, the resource allows to query the specified traffic types but only on the traffic types in this array.

3.1.2. Accept Input Parameters

The ReqFilteredCostMap object in Section 4.1.2 of [I-D.ietf-alto-multi-cost] 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..*>;]
    [JSONString traffic-types<0..*>;]
    [PIDFilter  pids;]
    [PIDFlow    pid-flows<0..*>;]
  } ReqFilteredCostMap;

  object {
    PIDName src;
    PIDName dst;
  } PIDFlow;
```

cost-type, multi-cost-types, testable-cost-types, constraints, or-constraints: As defined in Section 4.1.2 of [I-D.ietf-alto-multi-cost].

traffic-types: A JSONArray of JSONStrings, where each string indicates a traffic type. If present, the ALTO server will only return the cost of the traffic with the specified types.

pids: As defined in Section 11.3.2.3 of [RFC7285].

pid-flows: A list of PID src to PID dst for which path costs are to be returned.

Additional requirement is that the Client MUST specify either "pids" or "pid-flows", but MUST NOT specify both.

3.1.3. Response

The response is exactly as defined in Section 4.1.3 of [I-D.ietf-alto-multi-cost].

3.2. Extend Endpoint Abstraction

The typed endpoint address used by ECS is defined in Section 10.4 of [RFC7285]. That section only defines two address types: ipv4 and ipv6 to express IPv4 addresses and IPv6 addresses respectively. However, the flow-extended ECS may contain MAC addresses, domain names and port numbers to give a cost between hosts.

Therefore, this document transforms the typed endpoint address to EndpointURI to measure the cost more precisely. This URI must conform to the syntax for URI defined in [@!RFC3986], in particular with respect to character encoding. The EndpointURI may have one of the following form:

"protocol:name-or-address:port"
"protocol:name-or-address"

And this EndpointURI is defined in [OF15], and it is used to identify a controller connection. To extend endpoint abstraction, we use ConnectionURI to specify a flow with fields:

protocol: The protocol field is REQUIRED. It contains many kinds of protocols, the protocol can be layer two protocols (like PPP, ARP) and layer three protocols (like IPv4, IPv6), it can also be upper-layer protocols (like UDP, TCP, HTTP, FTP). And for different kinds of protocols, there are some provisions. Firstly, if the protocol field is layer two or layer three protocol, client’s query can not fill in the port field in EndpointURI, because the port is unnecessary. Secondly, when the protocol is upper-layer protocol, and if client do not indicate the port, for some special protocol, we will use the default port.

name-or-address: This field is REQUIRED. The value can be an IP address, a domain name or a MAC address.

port: This field is OPTIONAL. It is forbidden when the protocol is layer three (like IPv4 and IPv6) or layer two protocol (like MAC). And it is added for more fine-gained request when the protocol is upper-layer protocol. For some classic protocols, if the port is not

indicated, we use the default port. For example, the default port of SSH is 22, and FTP is 21, HTTP is 80.

A request to query the cost between hosts looks like this:

```
{
  "cost-type": {"cost-mode" : "ordinal",
                "cost-metric" : "routingcost"},
  "endpoints" : {
    "srcs": [ "ipv4:192.0.2.2:22" ],
    "dsts": [
      "http:www.a.com",
      "ipv4:198.51.100.34",
      "telnet:198.51.100.34:23",
      "ipv6:2000::1:2345:6789:abcd"
     ]
  }
}
```

This design of API is fully compatible with ECS. TypedEndpointAddr of EndpointFilter in ECS have the format of "protocal:address", and the protocol only supports IPv4 and IPv6, so it can also be used in this design.

3.3. Flow-based Endpoint Cost Service

3.3.1. Capabilities

The extensions to EndpointCostMapCpabilities are identically to FilteredCostMapCapabilities in Section 3.1.1. But for field "flow-based-filter", the true value means the ALTO server allows to get requests with "endpoint-flows" field. If not present, this field MUST be interpreted as if it is specified false.

3.3.2. Accept Input Parameters

The ReqEndpointCostMap object in Section 4.2.2 of [I-D.ietf-alto-multi-cost] 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..*>;]
      [JSONString traffic-types<0..*>;]
      [EndpointFilter endpoints;]
      [EndpointFlow endpoint-flows<1..*>;]
    } ReqEndpointCostMap;

    object {
      EndpointURI srcs<0..*>;
      EndpointURI dsts<0..*>;
    } EndpointFilter;

    object {
      EndpointURI src;
      EndpointURI dst;
    } EndpointFlow;
```

cost-type, multi-cost-types, testable-cost-types, constraints, or-constraints: As defined in Section 4.1.2 of [I-D.ietf-alto-multi-cost].

traffic-types: As defined in Section 3.1.2 of this document.

endpoints: As defined in Section 11.5.1.3 of [RFC7285].

endpoint-flows: A list of source endpoint to destination endpoint for which path costs are to be returned.

Additional requirement is that the Client MUST specify either "endpoints" or "endpoint-flows", but MUST NOT specify both.

3.3.3. Response

The response is exactly as defined in Section 4.2.3 of [I-D.ietf-alto-multi-cost].

3.4. Example

3.4.1. IRD Example

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

```
HTTP/1.1 200 OK
Content-Length: [TODO]
Content-Type: application/alto-directory+json
{
  "meta" : {
   "default-alto-network-map" : "my-default-network-map",
   "cost-types" : {
     "num-routingcost" : {
       "cost-mode" : "numerial",
       "cost-metric" : "routingcost"},
     "ord-routingcost" : {
       "cost-mode" : "ordinal",
       "cost-metric" : "routingcost"}
   },
    .....
    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"
    },
    "basic-flow-based-cost-map" : {
      "uri" : "http://alto.example.com/costmap/multi/filtered",
      "media-types" : [ "application/alto-costmap+json" ],
      "accepts" : [ "application/alto-costmapfilter+json" ],
      "uses" : [ "my-default-network-map" ],
      "capabilities" : {
        "flow-based-filter" : true,
        "cost-type-names" : [ "ord-routingcost" , "num-routingcost" ]
      }
    },
    "endpoint-advanced-flow-based-cost-map" : {
      "uri" : "http://alto.example.com/endpointcost/lookup",
      "media-types" : [ "application/alto-endpointcost+json" ],
      "accepts" : [ "application/alto-endpointcostparams+json" ],
      "uses" : [ "my-default-network-map" ],
      "capabilities" : {
        "flow-based-filter" : true,
        "cost-type-names" : [ "ord-routingcost" , "num-routingcost" ]
      }
    }
  }
}
```

3.4.2. Flow-based Filtered Cost Map Service Example

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

{
  "cost-type": {
    "cost-mode": "numerical",
    "cost-metric": "routingcost"
  },
  "pid-flows": [
    { "src": "PID1", "dst": "PID2" },
    { "src": "PID1", "dst": "PID3" },
    { "src": "PID3", "dst": "PID4" }
  ]
}

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

{
  "meta": {
    "dependent-vtags": [
      {
        "resource-id": "my-default-network-map",
        "tag": "75ed013b3cb58f896e839582504f622838ce670f"
      },
    ],
    "cost-type": {
      "cost-mode": "numerical",
      "cost-metric": "routingcost"
    }
  },
  "cost-map": {
    "PID1": { "PID2": 100 },
    "PID1": { "PID3": 20 },
    "PID3": { "PID4": 80 }
  }
}
```

3.4.3. Flow-based Endpoint Cost Service Example

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

{
  "cost-type": {
    "cost-mode": "numerical",
    "cost-metric": "hopcount"
  },
  "endpoint-flows": [
    { "src": "ipv4:192.0.2.2",
      "dst": "ipv4:192.0.2.89" },
    { "src": "ipv4:192.0.2.2",
      "dst": "http:cdn1.example.com" },
    { "src": "tcp:203.0.113.45:54321",
      "dst": "http:cdn1.example.com" }
  ]
}


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

{
  "meta": {
    "cost-type": {
      "cost-mode": "numerical",
      "cost-metric": "hopcount"
    }
  },
  "endpoint-cost-map": {
    "ipv4:192.0.2.2": {
      "ipv4:192.0.2.89": 3,
      "http:cdn1.example.com": 6
    },
    "tcp:203.0.113.45:54321": {
      "http:cdn1.example.com": 10
    }
  }
}
```

4. ALTO Flow Cost Specification: Advanced Flow-based Query

The basic flow-based query extends the ECS service to support querying the cost of flows. However, it only supports the cost query of flows defined by the 5-tuple of protocol, source/dst address (hostname, IP address, domain name or MAC address) and ports. However, in the emerging software-defined networking, the concept of flow is not confined by this 5-tuple anymore. Instead, the OpenFlow 1.5 specifications has defined 38 header match fields that could define a flow. This document next introduces an advanced flow-based query to support the flow-based cost queries for such a generic context of flows.

4.1. Basic Data Types

The flow cost service introduces some new basic data types, as defined below.

4.1.1. Flow ID

A flow ID has the same format as a PIDName, as defined in [RFC7285] Section 10.1 [1]. It is used to uniquely identify a flow in a flow cost service request.

4.1.2. Typed Header Field

A typed header field represents a particular field in a network protocol that can be obtained at the application layer. It is represented by the protocol name and the field name, concatenated by the colon (':', U+003A). The typed header fields are case insensitive.

For example, "ipv4:source" and "IPv4:source" both represent the source address field used in IPv4 and "tcp:destination" represents the destination port for a TCP connection.

See Table 2 for a list of proposed typed header fields.

4.1.3. Cost Confidence

A cost confidence is defined as a JSON integer within the range of [0, 100]. It represents the ALTO servers' estimation on the accuracy of the returned costs. The larger the cost confidence is, the more accurate the path cost SHOULD be. If the cost value is very accurate, for example, a unique path can be identified for a flow with the provided information, the ALTO server SHOULD provide a cost confidence of 100.

The cost confidence CAN be used as an evidence of ambiguous paths, which is often associated with insufficient information in a query. If an ALTO client finds that the associated cost confidence value is low, it can narrow down the flow header space in the query, e.g., by adding optional fields or use IP addresses instead of prefixes.

The cost confidence value can be computed in several ways. For example, an ALTO server MAY use the following formula for some cost metrics:

```
    c = 100 * (1 - |deviation / mean|)

                  0        if c <= 0
    confidence =
                  round(c) if c > 0
```

where mean and deviation are computed from the cost values of all possible paths.

4.2. Flow Cost Service

A flow cost service provides information about costs for each individual flow specified in a request.

4.2.1. Media Type

The media type of the flow cost service is "application/alto-flowcost+json".

4.2.2. HTTP Method

The flow cost service is requested using the HTTP POST method.

4.2.3. Accept Input Parameters

The input parameters of the flow cost service MUST be encoded as a JSON object of type FlowCostRequest in the body of an HTTP POST request. The media type of the request MUST be "application/alto-flowcostparams+json".

```
    object {
      FlowFilterMap     flows;
    } FlowCostRequest : MultiCostRequestBase;
```

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

      object-map {
        FlowId -> FlowFilter;
      } FlowFilterMap;

      object-map {
        TypedHeaderField -> JSONValue;
      } FlowFilter;
```

flows: A map of flow filters for which path costs are to be returned. Each flow filter is identified by a unique FlowId, as defined in Section 4.1.1. The value types of a field is protocol-specific, see Table 3 for the value types associated with typed header fields in Table 2.

cost-type: The same as defined in [I-D.ietf-alto-multi-cost] Section 4.2.2 [2].

multi-cost-types: The same as defined in [I-D.ietf-alto-multi-cost] Section 4.2.2 [3].

testable-cost-types, constraints, or-constraints: The same as defined in [I-D.ietf-alto-multi-cost] Section 4.2.2 [4].

4.2.4. Capabilities

The capabilities of the flow cost service is a JSON object of type FlowCostCapabilities:

```
      object {
        TypedHeaderField  required<1..*>;
        [TypedHeaderField optional<1..*>;]
      } FlowCostCapabilities :  FilteredCostMapCapabilities;
```

with fields:

required: A list of required typed header fields. These fields are essential to find the path cost for a given flow and MUST be provided in a flow filter.

optional: A list of optional typed header fields. The ALTO server MAY leverage the values of the optional fields to find more accurate costs.

4.2.5. Response

The "meta" field of a flow cost response MUST contain the same cost type information as defined in [I-D.ietf-alto-multi-cost] Section 4.2.3 [5].

The data component of a flow cost service is named "flow-cost-map", which is a JSON object of type FlowCostMap:

```
object {
  FlowCostMap     flow-cost-map;
  [FlowCostMap    flow-cost-confidences;]
} FlowCostResponse : ResponseEntityBase;

object-map {
  FlowId -> JSONValue;
} FlowCostMap;
```

flow-cost-map: A dictionary map with each key (flow ID) representing a flow specified in the request. For each flow, the cost MUST follow the format defined in [I-D.ietf-alto-multi-cost] Section 4.2.3 [6].

flow-cost-confidences: A dictionary map with each key (flow ID) representing a flow specified in the request. For a single cost, the cost confidence for each flow MUST follow the specification in Section 4.1.3. If the query is using multiple costs where the costs are returned as a JSONArray, the cost confidence MUST also be a JSONArray where each element represents the cost confidence value computed for the corresponding cost type.

4.2.5.1. Ambiguous Paths

Since new forwarding abstractions support fine-grained routing, for example, OpenFlow 1.5 [OF15] has defined 38 header match fields, it is possible that the ALTO server cannot determine the path using the provided header fields. The computation for costs with ambiguous paths is implementation-specific, the servers can choose to return an integrated result of all possible paths, or simply use the cost of a random path. The ALTO servers SHOULD provide cost confidences to justify the accuracy of the provided cost values.

The ALTO server SHOULD be able to determine a unique path when all the optional typed header fields are provided without masks for a flow, however, the client SHOULD NOT assume this always holds.

4.2.6. Errors

The ALTO servers can provide more information to the clients when requests have errors. The FlowCostErrorMap below can provide basic information about two most common errors for the flow cost service. The ALTO servers MAY include it as the data component of an ALTO error response. If multiple errors are identified, the ALTO server MUST return exactly one error code according to [RFC7285] Section 8.5.2 [7].

```
object-map {
  FlowId -> FlowCostError;
} FlowCostErrorMap;

object {
  [TypedHeaderField conflicts<2..*>;]
  [TypedHeadreField missing<2..*>;]
  [TypedHeaderField unsupported<1..*>;]
} FlowFilterError;
```

conflicts: A list of conflicting typed header fields. See Section 4.2.6.1 for details.

missing: A list of missing typed header fields. See Section 4.2.6.2 for details.

unsupported: A list of unsupported typed header fields. See Section 4.2.6.3 for details.

4.2.6.1. Conflicts

Some header fields may have conflicts. For example, IPv4 fields and IPv6 fields can never appear in the same packet, nor can TCP and UDP ports. These header fields MUST not be included in the same flow filter, otherwise the ALTO server MUST return an ALTO error response, with the error code "E_INVALID_FIELD_VALUE". As specified in [RFC7285] Section 8.5.2 [8], the ALTO server MAY include the "field" and the "value" in the "meta" field. In this case, the ALTO server MUST use the flow ID as the "field" and the flow filter as the "value". However, the recommended approach is to use the FlowCostErrorMap, where the server CAN provide the conflicting typed header fields in the "conflicts" field of the FlowFilterError associated with the corresponding flow ID.

4.2.6.2. Missing Fields

The "E_MISSING_FIELD" error code is originally designed to report the absence of required JSON fields. In the flow cost service, the required typed header fields are implementation-specific and the ALTO servers MUST declare the required fields in the capabilities. If any required header field is missing, the ALTO server MUST return an ALTO error response, with the error code "E_MISSING_FIELD". The ALTO server CAN follow the steps defined in [RFC7285] Section 8.5.2 [9] to indicate the location of the missing field. An alternative approach which is also recommended, is that the server provide the missing typed header fields in the "missing" field of the FlowFilterError associated with the corresponding flow ID.

4.2.6.3. Unsupported Fields

If a query contains unsupported typed header fields, e.g., those not in the "required" nor the "optional" capabilities, the ALTO server MUST return an ALTO error response, with the error code "E_INVALID_FIELD_VALUE". Like how the conflicting header fields are handled in Section 4.2.6.1, the ALTO servers CAN report unsupported typed header fields in the "unsupported" field associated with the corresponding flow ID.

4.3. Advanced Flow-based Query Example

```
POST /flowcost/lookup HTTP/1.1
HOST: alto.example.com
Content-Length: 521
Content-Type: application/alto-flowcostparams+json
Accept: application/alto-flowcost+json,application/alto-error+json

{
  "cost-type": {
    "cost-mode": "numerical",
    "cost-metric": "routingcost"
  },
  "flows": {
    "l3-flow": {
      "ipv4:source": "192.168.1.1",
      "ipv4:destination": "192.168.1.2"
    },
    "optional-l2-flow": {
      "ethernet:source": "12:34:56:78:00:01",
      "ethernet:destination": "12:34:56:78:00:02"
    },
    "l3-flow-aggr": {
      "ipv4:source": "192.168.1.0/24",
      "ipv4:destination": "192.168.2.0/24"
    }
  }
}
```

```
HTTP/1.1 200 OK
Content-Length: 312
Content-Type: application/alto-flowcost+json

{
  "meta": {
    "cost-type": {
      "cost-mode": "numerical",
      "cost-metric": "routingcost"
    },
  },
  "flow-cost-map": {
    "l3-flow": 10,
    "l3-flow-aggr": 50
    "optional-l3-flow": 5,
  },
  "flow-cost-confidences": {
    "l3-flow": 70,
    "l3-flow-aggr": 40,
    "optional-l2-flow": 90
  }
}
```

5. Security Considerations

This document has not conducted its security analysis.

6. IANA Considerations

This document defines two new entries to be registered to application/alto-* media types.

6.1. Media Types

This document registers two media types, listed in Table 1.

Type	Subtype	Specification
application	alto-flowcost+json	Section 3.1.3
application	alto-flowcostparam+json	Section 3.3.2

Table 1: ALTO FCS Media Types

Type name: application

Subtype name: This document registers two subtypes, as listed in Table 1.

Required parameters: n/a

Optional parameters: n/a

Encoding considerations: Encoding considerations are identical to those specified for the "applicatoin/json" media type. See [RFC7159].

Security considerations: Security considerations are identical to those specified in [RFC7285] Section 15 [10].

Interoperability considerations: n/a

Published specification: This document is the specification for these media types. See Table 1 for the section documenting each media type.

Applications that use this media type: ALTO servers and ALTO clients with the extension to support the flow cost service, either standalone or embedded within other applications.

Additional information: n/a

Person & email address to contact for further information: See Authors' Addresses.

Intended usage: COMMON

Restrictions on usage: n/a

Author: See Authors' Addresses.

6.2. Header Field

TBD: Create the "ALTO Header Field Name Registry".

7. Acknowledgement

The authors would like to thank Dawn Chen, Haizhou Du, Sabine Randriamasy and Wendy Roome for fruitful discussions and feedback on this document. Shawn Lin provided substantial review feedback and suggestions to the protocol design.

8. References

8.1. 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>.

8.2. Informative References

[I-D.ietf-alto-multi-cost] Randriamasy, S., Roome, W., and N. Schwan, "Multi-Cost ALTO", draft-ietf-alto-multi-cost-05 (work in progress), February 2017.

[I-D.wang-alto-ecs-flows] Shen, X., Zhang, J., Wang, J., and Q. Xiang, "ALTO Extension: Endpoint Cost Service for Flows", draft-wang-alto-ecs-flows-01 (work in progress), April 2016.

[OF15] Foundation, O., "Openflow switch specification v1. 5.0", 2014, <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf>.

[openflow] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and J. Turner, "Openflow: enabling innovation in campus networks", 2008.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

[RFC7285] Alimi, R., Ed., Penno, R., Ed., Yang, Y., Ed., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, DOI 10.17487/RFC7285, September 2014, <http://www.rfc-editor.org/info/rfc7285>.

8.3. URIs

[1] https://tools.ietf.org/html/rfc7285#section-10.1

[2] https://tools.ietf.org/html/draft-ietf-alto-multi-cost-02#4.2.2

[3] https://tools.ietf.org/html/draft-ietf-alto-multi-cost-02#4.2.2

[4] https://tools.ietf.org/html/draft-ietf-alto-multi-cost-02#4.2.2

[5] https://tools.ietf.org/html/draft-ietf-alto-multi-cost-02#4.2.3

[6] https://tools.ietf.org/html/draft-ietf-alto-multi-cost-02#4.2.3

[7] https://tools.ietf.org/html/rfc7285#section-8.5.2

[8] https://tools.ietf.org/html/rfc7285#section-8.5.2

[9] https://tools.ietf.org/html/rfc7285#section-8.5.2

[10] ttps://tools.ietf.org/html/rfc7285#section-15

Appendix A. Tables

Protocol	Field Name	Description
Ethernet	source	The source MAC address
	destination	The destination MAC address
	vlan-id	VLAN-ID from 802.1Q header
IPv4	source	IPv4 source address
	destination	IPv4 destination address
IPv6	source	IPv6 source address
	destination	IPv6 destination address
TCP	source	TCP source port
	destination	TCP destination port
UDP	source	UDP source port
	destination	UDP destination port

Table 2: Protocols and Field Names.

Typed Header Field	Acceptable Value Type
ethernet:source	JSONString as MAC address
ethernet:destination	
ethernet:vlan-id	JSONNumber in the range of [1, 4094]
ipv4:source	JSONString as IPv4 address or IPv4 prefix
ipv4:destination	
ipv6:source	JSONString as IPv6 address or IPv6 prefix
ipv6:destination	
tcp:source	JSONNumber in the range of [0, 65535]
tcp:destination	0 serves as a wildcard value
udp:source	
udp:destination	

Table 3: Value Types for Typed Header Fields

Authors' Addresses

Kai Gao
Tsinghua University
30 Shuangqinglu Street
Beijing 100084
China

Email: gaok12@mails.tsinghua.edu.cn

Jingxuan Jensen Zhang
Tongji University
4800 Cao'an Hwy
Shanghai 201804
China

Email: jingxuan.n.zhang@gmail.com

Junzhuo Austin Wang
Tongji University
4800 Cao'an Hwy, Jiading District
Shanghai
China

Email: wangjunzhuo200@gmail.com

Qiao Xiang
Tongji/Yale University
51 Prospect Street
New Haven, CT
USA

Email: qiao.xiang@cs.yale.edu

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

Email: yry@cs.yale.edu

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 17, 2017

S. Randriamasy
Nokia Bell Labs
R. Yang
Yale University
Q. Wu
Huawei
L. Deng
China Mobile
N. Schwan
Thales Deutschland

February 13, 2017

ALTO Cost Calendar
draft-ietf-alto-cost-calendar-01

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 network informed decisions. The present draft extends the ALTO cost information so as to broaden the decision possibilities of applications to not only decide 'where' to connect to, but also 'when'. This is useful to applications that need to schedule their data transfers and connections and have a degree of freedom to do so. ALTO guidance to schedule application traffic can also efficiently help for load balancing and resources efficiency. Besides, the ALTO Cost Calendar allows to schedule the ALTO requests themselves and thus to save a number of ALTO transactions.

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

Requirements Language

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 3
2. Overview of ALTO Cost Calendars 5
2.1. ALTO Cost Calendar information features 5
2.2. ALTO Calendar design characteristics 6
2.2.1. ALTO Cost Calendar for all cost modes 7
2.2.2. Compatibility with legacy ALTO Clients 7
3. ALTO Calendar specification: IRD extensions 7
3.1. Calendar attributes in the IRD resources capabilities . . 7
3.2. Calendars in a delegate IRD 9
3.3. Example IRD with ALTO Cost Calendars 9
4. ALTO Calendar specification: Service Information Resources . 12
4.1. Calendar extensions for Filtered Cost Maps 12
4.1.1. Calendar extensions in Filtered cost map requests . . 13
4.1.2. Calendar extensions in Filtered Cost map responses . 13

4.1.3. Use case and example for a FCM with a bandwidth Calendar 15
4.2. Calendar extensions in the Endpoint Cost Map Service . . 17
4.2.1. Calendar specific input in Endpoint cost map requests 17
4.2.2. Calendar attributes in the Endpoint Cost Map response 17
4.2.3. Use case and example for the ECS with a routingcost Calendar 18
4.2.4. use case and example for the ECS with a multi-cost calendar for routingcost and latency 21
4.3. Recap of rules related to ALTO Cost Calendars 23
5. Use cases for ALTO Cost Schedule 23
5.1. Bulk Data Transfer scheduling upon bandwidth calendars . 23
5.1.1. Applicable example transaction 24
5.2. Applications with limited connectivity or access to datacenters 24
5.2.1. Applicable example transaction 26
5.3. SDN Controller guided traffic scheduling with Calendars . 26
5.3.1. Applicable example transaction 27
6. IANA Considerations 27
6.1. Information for IANA on proposed Cost Types 27
6.2. Information for IANA on proposed Endpoint Propeeries . . 27
7. Acknowledgements 27
8. References 27
8.1. Normative References 27
8.2. Informative References 28
Authors' Addresses 28

1. Introduction

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

The ALTO protocol in [RFC7285] specifies a Network Map which defines groupings of endpoints in provider-defined network regions (called PIDs). The Cost Map Service, Endpoint Cost Service (ECS) and Endpoint Ranking Service then provide ISP-defined costs and rankings for connections among the specified endpoints and PIDs and thus incentives for application clients to connect to ISP preferred locations, e.g. to reduce their costs. ALTO intentionally avoids provisioning realtime information as explained in the ALTO Problem Statement [RFC5693] and ALTO Requirements [RFC5693].Thus the current

Cost Map and Endpoint Cost Service are providing, for a given Cost Type, exactly one path cost value. Applications have to query one of these two services to retrieve the currently valid cost values. They therefore need to plan their ALTO information requests according to their own estimation of the frequency of cost value change.

With [RFC7285], an ALTO client should interpret the returned costs as those at the query moment. However, Network costs can fluctuate, e.g. due to diurnal patterns of traffic demand or planned events such as network maintenance, holidays or highly publicized events. Providing network costs for only the current time thus may not be sufficient, in particular for applications that can schedule their traffic in a span of time, for example by deferring backup to night during traffic trough.

In case the ALTO Cost value changes are predicable over a certain period of time and the application does not require immediate data transfer, it can save time to get the whole set of cost values over this period in one ALTO response. Using them to schedule data transfers allows optimising the network resources usage and QoE. ALTO Clients and Servers can also minimize their workload by accordingly scheduling their data exchanges.

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

In this draft an "ALTO Cost Calendar" is specified by information resources capabilities that are applicable to time-sensitive ALTO metrics. An ALTO Cost Calendar exposes ALTO Cost Values in JSON arrays where each value corresponds to a given time interval. The time intervals as well as other Calendar attributes are specified in the IRD and in the Server response to allow the ALTO Client to interpret the received ALTO values. Last, the proposed extensions for ALTO Calendars are applicable to any Cost Mode and they ensure backwards compatibility with legacy ALTO clients.

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

2. Overview of ALTO Cost Calendars

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

- an array of values for a given metric, where each value corresponds to a time interval, where the value array can sometimes be a cyclic pattern that repeats a certain number of times.
- attributes describing the time scope of the calendar such as the size and number of the intervals and the date of the starting point of the calendar, allowing an ALTO Client to properly interpret the values.

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

Most likely, the ALTO Cost Calendar would be used for the Endpoint Cost Service, assuming that a limited set of feasible Endpoints for a non-real time application is already identified, that they do not need to be accessed immediately and that their access can be scheduled within a given time period. The Filtered Cost Map service is also applicable as long as the size of the Map allows it.

2.1. ALTO Cost Calendar information features

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

- attributes to interpret the time scope of the Calendar value array:
  - generic time zone,
  - applicable time interval for each calendar value: combining numbers and time units to reflect for example: 1 hour, 2 minutes, 10 seconds, 1 week, 1 month,
  - duration of the Calendar: e.g. the number of intervals provided in the calendar.

- "calendar-start-date": specifying when the calendar starts, that is to which date the first value of the cost calendar is applicable.

- "repeated": an optional attribute indicating for how many iterations the provided calendar will have the same values. The server may use it to allow the client to schedule its next request and thus save its own workload by avoiding to process useless requests.

2.2. ALTO Calendar design characteristics

The protocol extension placeholders for an ALTO Calendar are: the IRD, the ALTO requests and responses for Cost calendars.

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

The calendar-specific capabilities are integrated in the information resources of the IRD and in the "meta" member of ALTO responses to Cost Calendars requests. A calendar and its capabilities are associated to a given information resource and within this information resource to a given cost type. This design has several advantages:

- it does not introduce a new mode,

- it does not introduce new media types,

- it allows an ALTO Server to offer calendar capabilities on a cost type, with attributes values adapted to each information resource.

The Applicable Calendared information resources are:

- the Filtered Cost Map,

- the Endpoint Cost Map.

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

2.2.1. ALTO Cost Calendar for all cost modes

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

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

2.2.2. Compatibility with legacy ALTO Clients

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

For compatibility with legacy ALTO Clients specified in RFC7285, calendared information resources are not applicable for full Cost Maps for the following reason: a legacy ALTO client would receive a Calendared Cost Map via an HTTP ’GET’ command. As specified in section 8.3.7 of RFC7285, it will ignore the Calendar Attributes indicated in the "meta" of the responses. Therefore, lacking information on calendar attributes, it will not be able to correctly interpret and process the values of the received array of calendar cost values.

3. ALTO Calendar specification: IRD extensions

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

3.1. Calendar attributes in the IRD resources capabilities

When for an applicable resource , an ALTO Server provides a Cost Calendar for a given Cost Type, it MUST indicate this in the IRD capabilities of this resource, by an object of type

'CalendarAttributes', associated to this Cost Type and specified below.

The capabilities of a Calendar aware information resource entry have a member named "calendar-attributes" which is an array of objects of type CalendarAttributes. It is necessary to use an array because of resources such as Filtered Cost Map and Endpoint Cost Map, for which the member "cost-type-names" is an array of 1 or more values.

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

CalendarAttributes calendar-attributes <1..*>;

```
object{
  [JSONString   cost-type-name;]
  JSONString    time-interval-size;
  JSONNumber    number-of-intervals;
} CalendarAttributes;
```

o "cost-type-name":

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

o "time-interval-size":

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

o "number-of-intervals":

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

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

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

3.2. Calendars in a delegate IRD

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

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

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

3.3. Example IRD with ALTO Cost Calendars

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

The example IRD includes 2 particular URIs providing calendars:

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

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

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

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

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

   {
     "meta" : {
        "cost-types": {
           "num-routingcost": {
              "cost-mode" : "numerical",
              "cost-metric" : "routingcost"
           },
           "num-latency": {
              "cost-mode"  : "numerical",
              "cost-metric": "latency"
           },
           "num-pathbandwidth": {
              "cost-mode"  : "numerical",
              "cost-metric": "Availbandwidth",
           },
           "string-qual-status": {
              "cost-mode"  : "string",
              "cost-metric": "quality-status",
           }
         ... other meta ...
       },

    "resources" : {
       "filtered-cost-map-calendar" : {
           "uri" : "http://custom.alto.example.com/calendar/costmap/filtered",
           "media-type" : "application/alto-costmap+json",
```

```
          "accepts" : "application/alto-costmapfilter+json",
          "capabilities" : {
             "cost-constraints" : true,
             "cost-type-names"  : [ "num-routingcost", "num-pathbandwidth",
                                    "string-service-status" ],
             "calendar-attributes" : [
                {"cost-type-names" : [ "num-routingcost", "num-pathbandwidth" ]
,
                 "time-interval-size" : "1 hour",
                 "number-of-intervals" : 24
                },
                {"cost-type-names" : "string-service-status",
                 "time-interval-size" : "30 minute",
                 "number-of-intervals" : 48
                }
              ] // end calendar-attributes
             "uses": [ "my-default-network-map" ]
             }
          },

       "endpoint-cost-calendar-map" : {
         "uri" : "http://custom.alto.example.com/calendar/endpointcost/lookup",
         "media-types" : [ "application/alto-endpointcost+json" ],
         "accepts" : [ "application/alto-endpointcostparams+json" ],
         "capabilities" : {
           "cost-constraints" : true,
           "cost-type-names" : [ "num-routingcost", "num-latency",
                                 "num-pathbandwidth", "string-service-status" ],
           "calendar-attributes" : [
              {"cost-type-names" : "num-routingcost",
               "time-interval-size" : "1 hour",
               "number-of-intervals" : 24
              },
              {"cost-type-names" : "latency",
               "time-interval-size" : "5 minute",
               "number-of-intervals" : 12
              },
              {"cost-type-names" : "num-pathbandwidth",
               "time-interval-size" : "1 minute",
               "number-of-intervals" : 60
              },
              {"cost-type-names" : "string-service-status",
               "time-interval-size" : "2 minute",
               "number-of-intervals" : 30
              }
             ]
           "uses": [ "my-default-network-map" ]
         } // ECM capab
       } //info resource N
```

```
} // ressources
```

In this example IRD, for the filtered cost map service:

- the Calendar for ’num-routingcost’ and ’num-pathbandwidth’ is an array of 24 values each provided on a time interval lasting 1 hour.
- the Calendar for "string-service-status": "is an array of 48 values each provided on a time interval lasting 30 minutes.

For the endpoint cost map service:

- the Calendar for ’num-routingcost’: is an array of 24 values each provided on a time interval lasting 1 hour.
- the Calendar for ’latency’: is an array of 12 values each provided on a time interval lasting 5 minutes.
- the Calendar for ’num-pathbandwidth’: is an array of 60 values each provided on a time interval lasting 1 minute.
- the Calendar for "string-service-status": "is an array of 30 values each provided on a time interval lasting 2 minutes.

4. ALTO Calendar specification: Service Information Resources

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

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

```
Date: Tue, 15 Nov 2014 08:12:31 GMT
```

4.1. Calendar extensions for Filtered Cost Maps

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

4.1.1. Calendar extensions in Filtered cost map requests

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

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

```
JSONBoolean    calendared<1..*>;
```

This field is an array of 1 to N boolean values, where N is the number of requested metrics. Each boolean value indicates whether or not the ALTO Server should provide the values for this Cost Type as a calendar.

This field MUST NOT be specified if member "calendar-attributes" is not present for this information resource.

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

```
"calendared" : [true];
```

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

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

4.1.2. Calendar extensions in Filtered Cost map responses

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

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

- if the ALTO Client supports cost values for one Cost Type at a time only: the "meta" fields specified in RFC 7285 for these information service responses:
  - "dependent-vtags ",

  * "cost-type" field.

- if the ALTO Client supports cost values for several Cost Types at a time, as specified in [draft-ietf-alto-multi-cost] : the "meta" fields specified in [draft-ietf-alto-multi-cost] for these information service responses:

  * "dependent-vtags ",

  * "cost-type" field with value set to '{}', for backwards compatibility with RFC7285.

  * "multi-cost-types" field.

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

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

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

- "calendar-start-time": indicates the date at which the first value of the calendar applies. By default, the value provided for the "calendar-start-time" attribute SHOULD be no later than the request date.

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

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

- "repeated": is an optional field provided for Calendars. It is an integer N greater or equal to '1' that indicates how many iterations of the calendar value array starting at the date indicated by "calendar-start-time" have the same values. The number N includes the provided iteration.

Using the member "repeated" helps minimizing on the wire data exchange: by providing it, an ALTO Server will avoid unecessary

processing of requests for Calendars with unchanged values while it allows ALTO Clients to save their resources as well.

For example: if the "calendar-start-time" member has value "Mon, 30 Jun 2014 at 00:00:00 GMT" and if the value of member "repeated" is equal to 4, it means that the calendar values are the same values on Monday, Tuesday, Wednesday and Thursday. The ALTO Client thus may use the same calendar for the next 4 duration periods following "calendar-start-time".

4.1.3. Use case and example for a FCM with 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.

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

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

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

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

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


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

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

    "cost-map" : {
       "PID1": { "PID1": [v1,v2, ... v12],
                 "PID2": [v1,v2, ... v12],
                 "PID3": [v1,v2, ... v12] },
       "PID2": { "PID1": [v1,v2, ... v12],
                 "PID2": [v1,v2, ... v12],
                 "PID3": [v1,v2, ... v12] }
    }
  }
```

4.2. Calendar extensions in the Endpoint Cost Map Service

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

4.2.1. Calendar specific input in Endpoint cost map requests

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

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

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

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

4.2.2. Calendar attributes in the Endpoint Cost Map response

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

- if the ALTO Client supports cost values for one Cost Type at a time only: the "meta" fields specified in {11.5.1.6} of RFC 7285 for the Endpoint Cost response:
  - "cost-type" field.
- if the ALTO Client supports cost values for several Cost Types at a time, as specified in [draft-ietf-alto-multi-cost] : the "meta" fields specified in [draft-ietf-alto-multi-cost] for the the Endpoint Cost response:
  - "cost-type" field with value set to ’{}’, for backwards compatibility with RFC7285.

* "multi-cost-types" field.

If the client request does not provide member "calendared" or if it provides it with a value equal to ’false’, then the ALTO Server response is exactly as specified in the above cited references.

If the ALTO client provides member "calendared" with a value equal to ’true’ in the input parameters, the "meta" member of a Calendared Endpoint Cost Map response MUST include the same addifional member "calendar-response-attributes" as specified for the Filtered Cost Map Service. The Server response is thus changed as follows:

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

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

4.2.3. Use case and example for the ECS with a routingcost Calendar

Let us assume an Application Client is 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 predictable time periods. Therefore, it needs to both schedule its resources greedy networking activities and its ALTO transactions.

The Application Client has the choice to trade content or resources with a set of Endpoints and needs to decide with which one it will connect and at what time. For instance, the Endpoints are spread in different time-zones, or have intermittent access. In this example, the ’routingcost’ is assumed to be time sentitive with values provided as ALTO Calendars.

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

For Cost Type ’num-routingcost’, the sollicited 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).

In the following example, the ALTO Client sends its request on Tuesday July 1st 2014 at 13:15.

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

{
  "cost-type" : {"cost-mode" : "numerical", "cost-metric" : "routingcost"},
  "calendared" : [true],
  "endpoints" : {
    "srcs": [ "ipv4:192.0.2.2" ],
    "dsts": [
      "ipv4:192.0.2.89",
      "ipv4:198.51.100.34",
      "ipv4:203.0.113.45",
      "ipv6:2000::1:2345:6789:abcd"
    ]
  }
}
```

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

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

    "endpoint-cost-map" : {
      "ipv4:192.0.2.2": {
        "ipv4:192.0.2.89"    : [v1, v2, ... v24],
        "ipv4:198.51.100.34" : [v1, v2, ... v24],
        "ipv4:203.0.113.45"  : [v1, v2, ... v24],
        "ipv6:2000::1:2345:6789:abcd" : [v1, v2, ... v24]
      }
    }
}
```

When the Client gets the Calendar for "routingcost", it sees that the "calendar-start-time" is Monday at 00h00 GMT and member "repeated" is equal to ’4’. It understands that the provided values are valid until Thursday included and will only need to get a Calendar update on Friday.

4.2.4. use case and example for the ECS with a multi-cost calendar for routingcost and latency

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

As in the previous example, the IRD indicates that the ALTO Server provides "routingcost" Calendars in terms of 24 time intervals of 1 hour each.

For metric "latency", the IRD indicates that the ALTO Server provides Calendars in terms of 12 time intervals values lasting each 5 minutes.

In the following example transaction, the ALTO Client sends its request on Tuesday July 1st 2014 at 13:15.

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

  {
    "cost-type" : {},
    "multi-cost-types" : [
        {"cost-mode" : "numerical", "cost-metric" : "routingcost"},
        {"cost-mode" : "numerical", "cost-metric" : "latency"}
    ],
    "calendared" : [true, true],
    "endpoints" : {
      "srcs": [ "ipv4:192.0.2.2" ],
      "dsts": [
        "ipv4:192.0.2.89",
        "ipv4:198.51.100.34",
        "ipv4:203.0.113.45",
        "ipv6:2000::1:2345:6789:abcd"
      ]
```

```
    }
  }


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

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

      "endpoint-cost-map" : {
        "ipv4:192.0.2.2": {
          "ipv4:192.0.2.89"    : [[r1, r2, ... r24], [l1, l2, ... l12]],
          "ipv4:198.51.100.34" : [[r1, r2, ... r24], [l1, l2, ... l12]],
          "ipv4:203.0.113.45"  : [[r1, r2, ... r24], [l1, l2, ... l12]],
          "ipv6:2000::1:2345:6789:abcd" : [[r1, r2, ... r24], [l1, l2, ... l12]]
        }
      }
  }
```

When receiving the response, the client sees that the calendar values for ’routing cost’ are repeated for 4 iterations. Therefore, in its next requests until the routing cost calendar is expected to change, the client will only need to request a calendar for "latency".

Without the ALTO Calendar extensions, the ALTO client would have no clue on the dynamicity of the metric value change and would spend needless time requesting values at an inappropriate pace. In addition, without the Multi-Cost ALTO capabilities, the ALTO client

would duplicate this waste of time as it would need to send one request per cost metric.

4.3. Recap of rules related to ALTO Cost Calendars

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

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

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

Calendared information resources MUST be requested via a POST method.

5. Use cases for ALTO Cost Schedule

[THIS SECTION NEEDS TO BE SHORTENED]

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

5.1. Bulk Data Transfer scheduling upon bandwidth calendars

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

In the meantime, there is a degree of freedom on when the content is transmitted from the origin server to the caching node, or from the core site to a local site. However, scheduling these data transfers is a non-trivial task as they should not infer with the user peak demand to avoid degradation of user experience and to decrease billing costs for the datacenter operator by leveraging off-peak hours for the transfer.

As a result, these ICPs need to have a good knowledge on the link utilization patterns between the different datacenters before making an efficient scheduling decision. While usage data today is already gathered and used to schedule data transfers, provisioning these data

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

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

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

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

5.1.1. Applicable example transaction

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

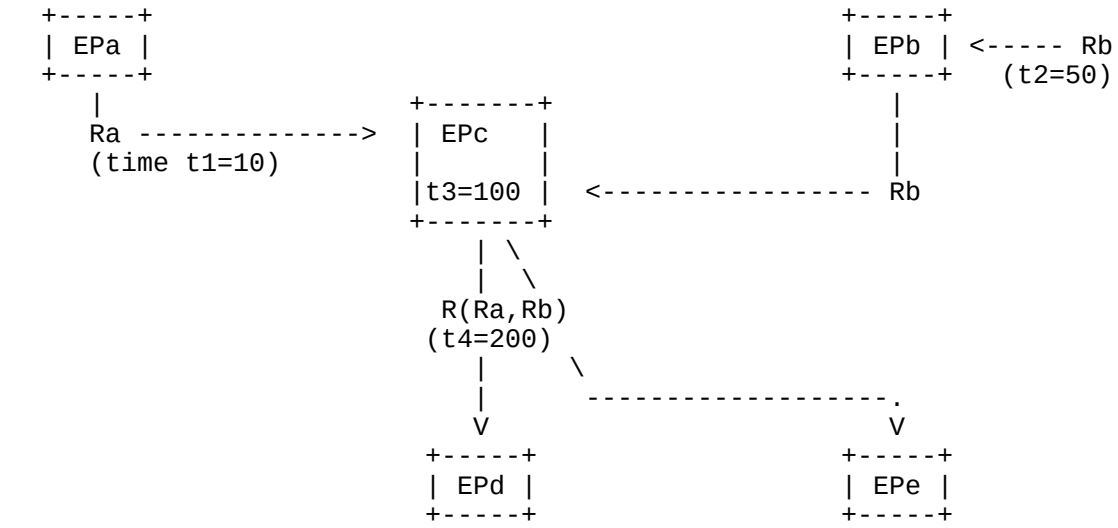
5.2. Applications with limited connectivity or access to datacenters

Some applications are limited in their connectivity either in time or resources or both. For example applications running on devices in remote locations or in developing countries that need to synchronize

their state with a data center periodically, in particular if sometimes there is no connection at all. Example applications are enterprise database update, remote learning, remote computation distributed on several data center endpoints.

Wireless connections have a variable quality and may even be intermittent. On the other hand, the wireless network conditions have a rapid impact on applications while they can sometimes be predicted over a span of time. Non real time applications and time-insensitive data transfers such as client patching, archive syncing, etc. can benefit from careful scheduling. It is thus desirable to provide ALTO clients with routing costs to connection nodes (i.e. Application Endpoints) over different time periods. This would allow end systems using ALTO aware application clients to schedule their connections to application endpoints.

Another challenge arises with applications using data and physical resources scattered around the world. For non-real time applications, the interaction with Endpoints can be orchestratrated and scheduled at the time slots corresponding to the best possible network conditions. 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.



5.2.1. Applicable example transaction

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

5.3. SDN Controller guided traffic scheduling with Calendars

An ALTO Server can assist an SDN Controller by hosting abstracted network information that can be provided to SDN aware applications via an ALTO Client.

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

One key objective of an SDN controller is the ability to balance the application traffic whenever possible. Resources availability may often be predicted and strong incentives for applications to time shift their traffic may be given by network operators appropriately setting routing cost values at different time values, according to their policy on network utilization over time.

To achieve this objective, the SDN controller can:

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

2. abstract the network topology and end to end path costs and store them in an ALTO Server as Network Maps and Cost Calendars

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

This way:

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

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

5.3.1. Applicable example transaction

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

6. IANA Considerations

Information for the ALTO Endpoint property registry maintained by the IANA and related to the new Endpoints supported by the acting ALTO server. These definitions will be formulated according to the syntax defined in Section on "ALTO Endpoint Property Registry" of [RFC7285]
,

Information for the ALTO Cost Type Registry maintained by the IANA and related to the new Cost Types supported by the acting ALTO server. These definitions will be formulated according to the syntax defined in Section on "ALTO Cost Type Registry" of [RFC7285],

6.1. Information for IANA on proposed Cost Types

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

6.2. Information for IANA on proposed Endpoint Propeeries

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

7. Acknowledgements

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

8. References

8.1. Normative References

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

[RFC5693] Seedorf, J. and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement", RFC 5693, DOI 10.17487/RFC5693, October 2009, <http://www.rfc-editor.org/info/rfc5693>.

8.2. Informative References

[draft-ietf-alto-multi-cost]
S. Randriamasy, W. Roome, N. Schwan, , "Multi-Cost ALTO (work in progress), draft-ietf-alto-multi-cost", September 2016.

[draft-ietf-alto-performance-metrics]
Q. Wu, Y. Yang, Y. Lee, D. Dhody, S. Randriamasy, , "ALTO Performance Cost Metrics (work in progress)", September 2016.

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

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

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

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

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

Authors' Addresses

Sabine Randriamasy
Nokia Bell Labs
Route de Villejust
NOZAY 91460
FRANCE

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

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

Email: yry@cs.yale.edu

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

Email: sunseawq@huawei.com

Lingli Deng
China Mobile
China

Email: denglingli@chinamobile.com

Nico Schwan
Thales Deutschland

Email: nico.schwan@thalesgroup.com

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

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

April 27, 2017

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

Abstract

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

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

Status of This Memo

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 October 29, 2017.

Copyright Notice

Table of Contents

1. Introduction 3
1.1. Requirements Language 4
2. Terminology 4
3. Overview Of Approach 5
3.1. Multi-Cost Data Format 5
3.2. Compatibility With Legacy ALTO Clients 5
3.3. Filtered Multi Cost Map Resources 6
3.4. Endpoint Cost Service Resources 6
3.5. Full Cost Map Resources 7
3.6. Extended Constraint Tests 7
3.6.1. Extended constraint predicates 7
3.6.2. Extended logical combination of predicates 7
3.6.3. Testable Cost Types in constraints 8
3.6.4. Testable Cost Type Names in IRD capabilities 9
3.6.5. Legacy ALTO Client issues 9
4. Protocol Extensions for Multi-Cost ALTO Transactions 11
4.1. Filtered Cost Map Extensions 11
4.1.1. Capabilities 11
4.1.2. Accept Input Parameters 12
4.1.3. Response 15
4.2. Endpoint Cost Service Extensions 15
4.2.1. Capabilities 16
4.2.2. Accept Input Parameters 16
4.2.3. Response 17
5. Examples 17
5.1. Information Resource Directory 17
5.2. Multi-Cost Filtered Cost Map: Example #1 19
5.3. Multi-Cost Filtered Cost Map: Example #2 20
5.4. Multi-Cost Filtered Cost Map: Example #3 22
5.5. Multi-Cost Filtered Cost Map: Example #4 23
5.6. Endpoint Cost Service 24
6. IANA Considerations 25
7. Privacy And Security Considerations 26
8. Acknowledgements 26
9. References 26
9.1. Normative References 26
9.2. Informative References 26
Authors' Addresses 27

1. Introduction

IETF has defined ALTO services in [RFC7285] to provide guidance to overlay applications, which have to select one or several hosts from a set of candidates that are able to provide a desired resource. This guidance is based on parameters such as the topological distance, that affect performance of the data transmission between the hosts. The purpose of ALTO is to improve Quality of Experience (QoE) in the application while reducing resource consumption in the underlying network infrastructure. The ALTO protocol conveys a view of the Internet called a Network Map and composed of Provider defined locations spanning from subnets to several Autonomous Systems (AS). ALTO may also convey the Provider determined Costs between Network Map locations or between groups of individual endpoints.

Current ALTO Cost Types provide values such as hopcount and administrative routing cost to reflect ISP routing preferences. Recently, new use cases have extended the usage scope of ALTO to Content Delivery Networks (CDN), Data Centers and applications that need additional information to select their endpoints or network locations. Thus a multitude of new Cost Types that better reflect the requirements of these applications are expected to be specified.

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

It is far more efficient, in terms of Round Trip Time (RTT), traffic, and processing load on the ALTO Client and Server, to get all costs with a single query/response transaction. One Cost Map reporting on N Cost Types is less bulky than N Cost Maps containing one Cost Type each. This is valuable for both the storage of these maps and their transmission. Additionally, for many emerging applications that need information on several Cost Types, having them gathered in one map will save time. Another advantage is consistency: providing values for several Cost Types in one single batch is useful for ALTO Clients needing synchronized ALTO information updates. This document defines how to retrieve multiple cost metrics in a single request for ALTO Filtered Cost Maps and Endpoint Cost Maps. To ensure compatibility with legacy ALTO Clients, only the Filtered Cost Map and Endpoint Cost Map services are extended to return Multi-Cost values.

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

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

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

1.1. Requirements Language

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

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

2. Terminology

- ALTO transaction: A request/response exchange between an ALTO Client and an ALTO Server.
- Client: This term refers to an ALTO client, when used with a capital "C".
- Endpoint (EP): An endpoint is defined as in {2.1} of [RFC7285]. It can be for example a peer, a CDN storage location, a physical server involved in a virtual server-supported application, a party in a resource sharing swarm such as a computation grid or an online multi-party game.
- Server: This term refers to an ALTO server, when used with a capital "S".

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

3. Overview Of Approach

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

3.1. Multi-Cost Data Format

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

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

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

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

3.2. Compatibility With Legacy ALTO Clients

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

3.3. Filtered Multi Cost Map Resources

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

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

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

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

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

3.4. Endpoint Cost Service Resources

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

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

3.5. Full Cost Map Resources

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

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

3.6. Extended Constraint Tests

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

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

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

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

3.6.1. Extended constraint predicates

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

3.6.2. Extended logical combination of predicates

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

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

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

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

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

3.6.3. Testable Cost Types in constraints

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

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

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

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

3.6.4. Testable Cost Type Names in IRD capabilities

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

3.6.5. Legacy ALTO Client issues

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

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

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

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

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

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

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

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

4. Protocol Extensions for Multi-Cost ALTO Transactions

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

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

4.1. Filtered Cost Map Extensions

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

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

4.1.1. Capabilities

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

- max-cost-types,
- testable-cost-type-names

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

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

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

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

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

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

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

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

4.1.2. Accept Input Parameters

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4.1.3. Response

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

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

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

4.2. Endpoint Cost Service Extensions

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

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

4.2.1. Capabilities

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

4.2.2. Accept Input Parameters

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

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

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

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

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

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

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

4.2.3. Response

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

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

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

5. Examples

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

5.1. Information Resource Directory

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

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


HTTP/1.1 200 OK
Content-Length: 2704
Content-Type: application/alto-directory+json

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

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

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

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

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

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

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

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

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

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

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

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

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


HTTP/1.1 200 OK
Content-Type: application/alto-costmap+json
Content-Length: 461

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

5.6. Endpoint Cost Service

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

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

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

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

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

6. IANA Considerations

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

7. Privacy And Security Considerations

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

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

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

8. Acknowledgements

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

9. References

9.1. Normative References

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

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

9.2. Informative References

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

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

Authors' Addresses

Sabine Randriamasy
Nokia Bell Labs
Route de Villejust
NOZAY 91460
FRANCE

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

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

Email: ietf@wdroome.com

Nico Schwan
Thales Deutschland
Lorenzstrasse 10
Stuttgart 70435
Germany

Email: nico.schwan@thalesgroup.com

ALTO Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 4, 2017

Q. Wu
Huawei
Y. Yang
Yale University
Y. Lee
D. Dhody
Huawei
S. Randriamasy
Nokia Bell Labs
March 3, 2017

ALTO Performance Cost Metrics
draft-ietf-alto-performance-metrics-01

Abstract

Cost Metric is a basic concept in Application-Layer Traffic Optimization (ALTO). It is used in both the Cost Map Service and the Endpoint Cost Service.

Different applications may benefit from different Cost Metrics. For example, a Resource Consumer may prefer Resource Providers that offers a low delay delivery to the Resource Consumer. However the base ALTO protocol [ALTO] has documented only one single cost metric, i.e., the generic "routingcost" metric (Sec. 14.2 of ALTO base specification [ALTO]).

This document, proposes a set of Cost Metrics, derived and aggregated from routing protocols with different granularity and scope, such as BGP-LS,OSPF-TE and ISIS-TE, or from end to end traffic management tools. It currently documents Network Performance Cost Metrics reporting on network delay, jitter, packet loss, hop count, and bandwidth. These metrics may be exposed by an ALTO Server to allow applications to determine "where" to connect based on network performance criteria. Additional Cost Metrics involving ISP specific considerations or other network technologies may be documented in further versions of this draft.

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 3
2. Challenges on data sources and computation of ALTO performance metrics 5
2.1. Data sources 5
2.2. Computation of ALTO performance metrics 5
3. Cost Metric: POWDelay 6
4. Cost Metric: RTT 8
5. Cost Metric: PDV 9
6. Cost Metric: Hop Count 11
7. Cost Metric: Packet Loss 13
8. Traffic Engineering Performance Cost Metrics 15
8.1. Cost Metric: Link Maximum Reservable Bandwidth 16
8.2. Cost Metric: Link Residue Bandwidth 17
8.3. Cost Metric: Link Available Bandwidth 19

8.4. Cost Metric: Link Utilized Bandwidth 21
9. Security Considerations 23
10. IANA Considerations 24
11. References 24
11.1. Normative References 24
11.2. Informative References 26
Authors' Addresses 26

1. Introduction

Cost Metric is a basic concept in Application-Layer Traffic Optimization (ALTO). It is used in both the Cost Map Service and the Endpoint Cost Service. In particular, applications may benefit from knowing network performance measured on several Cost Metrics. For example, a more delay sensitive application may focus on latency, and a more bandwidth-sensitive application may focus on available bandwidth.

This document introduces a set new cost metrics, listed in Table 1, to support the aforementioned applications and allow them to determine "where" to connect based on network performance criteria. Hence, this document extends the base ALTO protocol [ALTO], which defines only a single cost metric, i.e., the generic "routingcost" metric (Sec. 14.2 of ALTO base specification [ALTO]).

Namespace	Property	Reference
	owdelay	See Section 3,[RFC2679] Section 3.6
	rtt	See Section 4,[RFC2681] Section 2.6
	pdv	See Section 5,[RFC3393] Section 2.6
	hopcount	See Section 6,[RFC7285]
	pktloss	See Section 7,[RFC7680] Section 2.6
	maxresbw	See Section 8.1,[RFC5305] Section 3.5
	residbw	See Section 8.2,[RFC7810] Section 4.5
	availbw	See Section 8.3,[RFC7810] Section 4.6
	utilbw	See Section 8.4,[RFC7810 Section 4.7

Table 1.

The purpose of this draft is to list the metrics likely to be exposed to ALTO Clients, including those already specified in other standardization groups and as such it does not claim novelty on all the specified metrics. Some metrics may have values produced by explicitely specified measurement methods such as those specified in IPPM, some may be ISP dependent such as those registered in ISIS or OSPF-TE. In this case, this document will refer to the relevant specifications.

An ALTO server may provide a subset of the cost metrics described in this document. These cost metrics can be retrieved and aggregated from routing protocols or other traffic measurement management tools (See Figure 1). Note that these cost metrics are optional and not all them need to be exposed to applications. If some are subject to privacy concerns, the ALTO server should not provide them to the client.

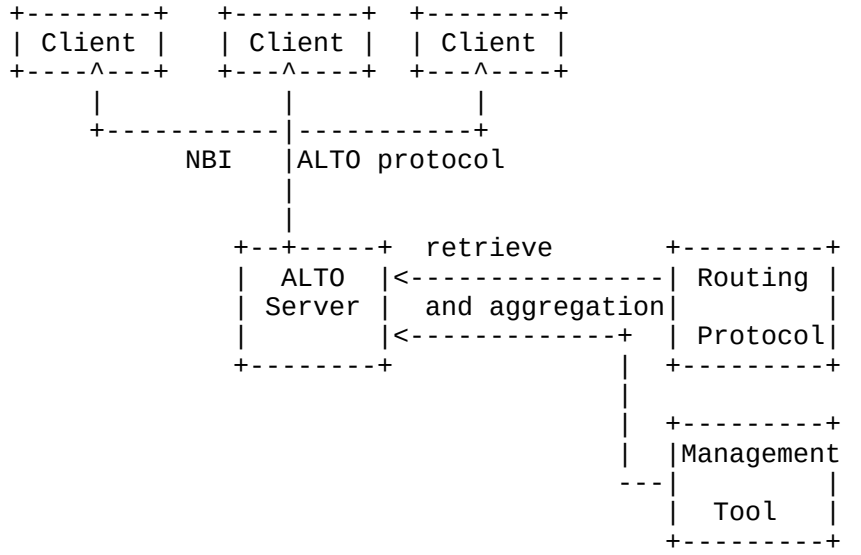


Figure 1.End to End Path Cost Metrics Exposing

When an ALTO server supports a cost metric defined in this document, it SHOULD announce this metric in its IRD.

Additionally, further versions of this document may define network metric values that stem from both measurements and provider policy as for example, many end to end path bandwidth related ALTO metrics. ALTO may convey such information, not available via 3rd party measurement tools. Besides, IPPM informational RFC 5136 points the difficulty to have a unified nomenclature for network capacity related measurements.

As for the reliability and trust in the exposed metric values, applications will rapidly give up using ALTO-based guidance if they feel the exposed information does not preserve their performance level or even degrades it.

Following the ALTO base protocol, this document uses JSON to specify the value type of each defined metric. See [RFC4627] for JSON data type specification.

2. Challenges on data sources and computation of ALTO performance metrics

2.1. Data sources

An ALTO server needs data sources to compute the cost metrics described in this document. This document does not define the exact data sources. For example, the ALTO server may use log servers or the OAM system as its data source [ALTO-DEPLOYMENT]. In particular, the cost metrics defined in this document can be computed using routing systems as the data sources. Mechanisms defined in [RFC3630], [RFC3784], [OSPF-TE], [ISIS-TE], [BGP-LS] and [BGP-PM] that allow an ALTO Server to retrieve and derive the necessary information to compute the metrics that we describe in this document.

One challenge lies in the data sources originating the ALTO metric values. The very purpose of ALTO is to guide application traffic with provider network centric information that may be exposed to ALTO Clients in the form of network performance metric values. Not all of them metrics have values produced by standardized measurement methods or routing protocols. Some of them involve provider-centric policy considerations. Some of them may describe wireless or cellular networks. To reliably guide users and applications while preserving provider privacy, ALTO performance metric values may also add abstraction to measurements or provide unitless performance scores.

2.2. Computation of ALTO performance metrics

The metric values exposed by an ALTO server may result from additional processing on measurements from data sources to compute exposed metrics. This may invlove data processing tasks such as aggregating the results across multiple systems, removing outliers, and creating additional statistics.

One challenge in describing the metrics is that performance metrics often depend on configuration parameters. For example, the value of packet loss rate depends on the measurement interval and varies over time. To handle this issue, an ALTO server may collect data on time periods covering the past and present or only collect data on present time. The ALTO server may further aggregate these data to provide an abstract and unified view that can be more useful to applications. To make the ALTO client better understand how to use these performance data, the ALTO server may provide the client with the validity period of the exposed metric values.

Another challenge relates to the availability of end to end path values for certain metrics. Applications value information relating to bandwidth availability where as bandwidth related metrics can

often be only measured at the link level. This document specifies a set of link-level bandwidth related values that may be exposed as such by an ALTO server. The server may also expose other metrics derived from their aggregation and having different levels of endpoint granularity, e.g. link endpoints or session endpoints. The metric specifications may also expose the utilised aggregation laws.

3. Cost Metric: POWDelay

Metric name:

Periodic One Way Delay

Metric Description:

To specify spatial and temporal aggregated delay of a stream of packets exchanged between the specified source and destination or the time that the packet spends to travel from source to destination. The spatial aggregation level is specified in the query context (e.g., PID to PID, or endpoint to endpoint).

Method of Measurement or Calculation:

See section 8.3 of [I-D.ietf-ippm-initial-registry] for Measurement Method.

Units of Measurement:

See section 8.4.3 of [I-D.ietf-ippm-initial-registry] for Measurement Unit. The unit is expressed in seconds.

Measurement Point(s) with Potential Measurement Domain:

See section 2.1, Data sources.

Measurement Timing:

See section 8.3.5 of [I-D.ietf-ippm-initial-registry] for Measurement Timing.

Use and Applications:

The Metric value Type is a single 'JSONNumber' type value containing a non-negative integer component that may be followed by an exponent part. The Cost Mode is encoded as a US-ASCII string.

This metric could be used as a cost metric constraint attribute used either together with cost metric attribute 'routingcost' or on its own or as a returned cost metric in the response.

Example 1: Delay value on source-destination endpoint pairs

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

{
  "cost-type": {"cost-mode" : "numerical",
                "cost-metric" : "powdelay"},
  "endpoints" : {
    "srcs": [ "ipv4:192.0.2.2" ],
    "dsts": [
      "ipv4:192.0.2.89",
      "ipv4:198.51.100.34",
      "ipv6:2000::1:2345:6789:abcd"
    ]
  }
}

HTTP/1.1 200 OK
Content-Length: TBA
Content-Type: application/alto-endpointcost+json
{
  "meta" :{
    "cost-type": {"cost-mode" : "numerical",
                  "cost-metric" : "powdelay"
     }
   },
    "endpoint-cost-map" : {
      "ipv4:192.0.2.2": {
        "ipv4:192.0.2.89"    : 10,
        "ipv4:198.51.100.34" : 20,
        "ipv6:2000::1:2345:6789:abcd"  : 30,
    }
  }
}
```

4. Cost Metric: RTT

Metric name:

Round Trip Delay

Metric Description:

To specify spatial and temporal aggregated round trip delay between the specified source and destination or the time that the packet spends to travel from source to destination and then from destination to source. The spatial aggregation level is specified in the query context (e.g., PID to PID, or endpoint to endpoint).

Method of Measurement or Calculation:

See section 4.3 of [I-D.ietf-ippm-initial-registry] for Measurement Method.

Units of Measurement:

See section 4.4.3 of [I-D.ietf-ippm-initial-registry] for Measurement Unit. The unit is expressed in seconds.

Measurement Point(s) with Potential Measurement Domain:

See section 2.1, Data sources.

Measurement Timing:

See section 4.3.5 of [I-D.ietf-ippm-initial-registry] for Measurement Timing.

Use and Applications:

See section 3 for use and application.

Example 7: Round Trip Delay value on source-destination endpoint pairs

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

{
  "cost-type": {"cost-mode" : "numerical",
                "cost-metric" : "rtt"},
  "endpoints" : {
    "srcs": [ "ipv4:192.0.2.2" ],
    "dsts": [
      "ipv4:192.0.2.89",
      "ipv4:198.51.100.34",
      "ipv6:2000::1:2345:6789:abcd"
    ]
  }
}

HTTP/1.1 200 OK
Content-Length: TBA
Content-Type: application/alto-endpointcost+json
{
  "meta" :{
    "cost-type": {"cost-mode" : "numerical",
                  "cost-metric" : "rtt"
     }
   },
    "endpoint-cost-map" : {
      "ipv4:192.0.2.2": {
        "ipv4:192.0.2.89"    : 4,
        "ipv4:198.51.100.34" : 3,
        "ipv6:2000::1:2345:6789:abcd"  : 2,
    }
  }
}
```

5. Cost Metric: PDV

Metric name:

Packet Delay Variation

Metric Description:

To specify spatial and temporal aggregated jitter (packet delay variation) with respect to the minimum delay observed on the stream over the specified source and destination. The spatial aggregation level is specified in the query context (e.g., PID to PID, or endpoint to endpoint).

Method of Measurement or Calculation:

See section 5.3 of [I-D.ietf-ippm-initial-registry] for Measurement Method.

Units of Measurement:

See section 5.4.4 of [I-D.ietf-ippm-initial-registry] for Measurement Unit. The unit is expressed in seconds.

Measurement Point(s) with Potential Measurement Domain:

See section 2.1, Data sources.

Measurement Timing:

See section 5.3.5 of [I-D.ietf-ippm-initial-registry] for Measurement Timing.

Use and Applications:

See section 3 for use and application.

Example 2: Delay jitter value on source-destination endpoint pairs

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

{
  "cost-type": {"cost-mode" : "numerical",
   "cost-metric" : "delayjitter"},
  "endpoints" : {
    "srcs": [ "ipv4:192.0.2.2" ],
    "dsts": [
      "ipv4:192.0.2.89",
      "ipv4:198.51.100.34",
      "ipv6:2000::1:2345:6789:abcd"
    ]
  }
}
HTTP/1.1 200 OK
 Content-Length: TBA
 Content-Type: application/alto-endpointcost+json
{
  "meta": {
           "cost type": {
           "cost-mode": "numerical",
           "cost-metric":"delayjitter"
    }
   },
  "endpoint-cost-map": {
           "ipv4:192.0.2.2": {
           "ipv4:192.0.2.89"    : 0
           "ipv4:198.51.100.34" : 1
           "ipv6:2000::1:2345:6789:abcd"  : 5
         }
      }
   }
```

6. Cost Metric: Hop Count

The metric hopcount is mentioned in [ALTO] as an example. This section further clarifies its properties.

Metric name:

Hop count

Metric Description:

To specify the number of hops in the path between the source endpoint and the destination endpoint. The hop count is a basic measurement of distance in a network and can be exposed as Router Hops, IP hops or other hops in direct relation to the routing protocols originating this information. It might also result from the aggregation of such information.

Method of Measurement or Calculation:

See section 2.2, Computation of metrics.

Units of Measurement:

The unit is integer number.

Measurement Point(s) with Potential Measurement Domain:

See section 2.1, Data sources.

Measurement Timing:

See section 2.1, second paragraph for Measurement Timing.

Use and Applications:

See section 3 for use and application.

Example 4: hopcount value on source-destination endpoint pairs

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

  {
    "cost-type": {"cost-mode" : "numerical",
     "cost-metric" : "hopcount"},
    "endpoints" : {
      "srcs": [ "ipv4:192.0.2.2" ],
      "dsts": [
        "ipv4:192.0.2.89",
        "ipv4:198.51.100.34",
        "ipv6:2000::1:2345:6789:abcd"
      ]
    }
  }
HTTP/1.1 200 OK
Content-Length: TBA
Content-Type: application/alto-endpointcost+json
{
    "meta": {
                "cost type": {
              "cost-mode": "numerical",
              "cost-metric":"hopcount"}
       }
    },
   "endpoint-cost-map": {
           "ipv4:192.0.2.2": {
           "ipv4:192.0.2.89"   : 5,
           "ipv4:198.51.100.34": 3,
           "ipv6:2000::1:2345:6789:abcd" : 2,
                              }
            }
 }
```

7. Cost Metric: Packet Loss

Metric name:

Packet loss

Metric Description:

To specify spatial and temporal aggregated packet loss over the specified source and destination. The spatial aggregation level is specified in the query context (e.g., PID to PID, or endpoint to endpoint).

Method of Measurement or Calculation:

See section 2.6 of [RFC7680] for Measurement Method.

Units of Measurement:

The unit is percentile.

Measurement Point(s) with Potential Measurement Domain:

See section 2.1, Data sources.

Measurement Timing:

See section 2 and section3 of [RFC7680] for Measurement Timing.

Use and Applications:

See section 3 for use and application.

Example 3: pktloss value on source-destination endpoint pairs

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

  {
    "cost-type": {"cost-mode" : "numerical",
     "cost-metric" : "pktloss"},
    "endpoints" : {
      "srcs": [ "ipv4:192.0.2.2" ],
      "dsts": [
        "ipv4:192.0.2.89",
        "ipv4:198.51.100.34",
        "ipv6:2000::1:2345:6789:abcd"
      ]
    }
  }
HTTP/1.1 200 OK
Content-Length: TBA
Content-Type: application/alto-endpointcost+json
{
    "meta": {
                "cost type": {
              "cost-mode": "numerical",
              "cost-metric":"pktloss"}
       }
    },
   "endpoint-cost-map": {
           "ipv4:192.0.2.2": {
           "ipv4:192.0.2.89"   : 0,
           "ipv4:198.51.100.34": 1,
           "ipv6:2000::1:2345:6789:abcd" : 2,
                              }
            }
 }
```

8. Traffic Engineering Performance Cost Metrics

This section introduces ALTO network performance metrics that may be aggregated from network metrics measured on links and specified in other documents. In particular, the bandwidth related metrics specified in this section are only available through link level measurements. For some of these metrics, the ALTO Server may further expose aggregated values while specifying the aggregation laws.

8.1. Cost Metric: Link Maximum Reservable Bandwidth

Metric name:

Maximum Reservable Bandwidth

Metric Description:

To specify spatial and temporal maximum reservable bandwidth over the specified source and destination. The value is corresponding to the maximum bandwidth that can be reserved (motivated from RFC 3630 Sec. 2.5.7.). The spatial aggregation unit is specified in the query context (e.g., PID to PID, or endpoint to endpoint).

Method of Measurement or Calculation:

Maximum Reserveable Bandwidth is the bandwidth measured between two directly connected IS-IS neighbors or OSPF neighbor, See section 3.5 of [RFC5305] for Measurement Method.

Units of Measurement:

The unit of measurement is byte per seconds.

Measurement Point(s) with Potential Measurement Domain:

See section 2.1, Data sources.

Measurement Timing:

See section 3.5 of [RFC5305] and section 5 of [RFC7810] for Measurement Timing.

Use and Applications:

See section 3 for use and application.

Example 6: maxresbw value on source-destination endpoint pairs

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

  {
    "cost-type" { "cost-mode":  "numerical",
    "cost-metric":  "maxresbw"},
    "endpoints":  {
      "srcs": [ "ipv4 : 192.0.2.2" ],
      "dsts": [
        "ipv4:192.0.2.89",
        "ipv4:198.51.100.34",
        "ipv6:2000::1:2345:6789:abcd"
      ]
    }
  }
HTTP/1.1 200 OK
Content-Length: TBA
Content-Type: application/alto-endpointcost+json
{
    "meta": {
           "cost-type": {
           "cost-mode": "numerical",
           "cost-metric": "maxresbw"
           }
    },
  " endpoint-cost-map": {
         "ipv4:192.0.2.2" {
         "ipv4:192.0.2.89" :    0,
         "ipv4:198.51.100.34": 2000,
         "ipv6:2000::1:2345:6789:abcd":  5000,
                           }
          }
}
```

8.2. Cost Metric: Link Residue Bandwidth

Metric name:

Residue Bandwidth

Metric Description:

To specify spatial and temporal residual bandwidth over the specified source and destination. The value is calculated by subtracting tunnel reservations from Maximum Bandwidth (motivated from [RFC7810], Sec.4.5.). The spatial aggregation unit is specified in the query context (e.g., PID to PID, or endpoint to endpoint).

Method of Measurement or Calculation:

Residue Bandwidth is the Unidirectional Residue bandwidth measured between two directly connected IS-IS neighbors or OSPF neighbor, See section 4.5 of [RFC7810] for Measurement Method.

Units of Measurement:

The unit of measurement is byte per seconds.

Measurement Point(s) with Potential Measurement Domain:

See section 2.1, Data sources.

Measurement Timing:

See section 5 of [RFC7810] for Measurement Timing.

Use and Applications:

See section 3 for use and application.

Example 8: residuebw value on source-destination endpoint pairs

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

  {
   "cost-type": { "cost-mode":  "numerical",
   "cost-metric":  "residubw"},
   "endpoints":  {
     "srcs": [ "ipv4 : 192.0.2.2" ],
     "dsts": [
       "ipv4:192.0.2.89",
       "ipv4:198.51.100.34",
       "ipv6:2000::1:2345:6789:abcd"
     ]
   }
}

HTTP/1.1 200 OK
Content-Length: TBA
Content-Type: application/alto-endpointcost+json
{
   "meta": {
          "cost-type" {
          "cost-mode": "numerical",
          "cost-metric": "residubw"
        }
    },
"endpoint-cost-map" {
         "ipv4:192.0.2.2" {
         "ipv4:192.0.2.89" :    0,
         "ipv4:198.51.100.34": 2000,
         "ipv6:2000::1:2345:6789:abcd":  5000,
                     }
        }
}
```

8.3. Cost Metric: Link Available Bandwidth

Metric name:

Available Bandwidth

Metric Description:

To specify spatial and temporal availaible bandwidth over the specified source and destination. The value is calculated by subtracting the measured bandwidth used for the actual forwarding of best effort traffic from Residue Bandwidth (motivated from [RFC7810], Sec.4.6.). The spatial aggregation level is specified in the query context (e.g., PID to PID, or endpoint to endpoint).

Method of Measurement or Calculation:

Available bandwidth is the Unidirectional Available bandwidth measured between two directly connected IS-IS neighbors or OSPF neighbor, See section 4.6 of [RFC7810] for Measurement Method.

Units of Measurement:

The unit of measurement is byte per seconds.

Measurement Point(s) with Potential Measurement Domain:

See section 2.1, Data sources.

Measurement Timing:

See section 5 of [RFC7810] for Measurement Timing.

Use and Applications:

See section 3 for use and application. Besides, knowledge about available bandwidth is essential for applications to distribute or schedule their transmissions. The example below illustrates how this metric is provided in the form of an ALTO calendar, as specified in [XXXX] to help deciding "where" and "when" to transmit.

Example 9: availbw value on source-destination endpoint pairs

This example assumes that the ALTO Server provides the values for metric "availbw" in the form of an ALTO calendar and declares it in its IRD.

```
POST /endpointcost/lookup HTTP/1.1
Host: alto.example.com
```

```
  Content-Length: TBA
  Content-Type: application/alto-endpointcostparams+json
  Accept: application/alto-endpointcost+json,application/alto-error+json

    {
     "cost-type": { "cost-mode":  "numerical",
                    "cost-metric":  "availbw"},
     "calendared" : [true],
     "endpoints":  {
        "srcs": [ "ipv4 : 192.0.2.2" ],
        "dsts": [
          "ipv4:192.0.2.89",
          "ipv4:198.51.100.34",
          "ipv6:2000::1:2345:6789:abcd"
        ]
     }
       }

  HTTP/1.1 200 OK
  Content-Length: TBA
  Content-Type: application/alto-endpointcost+json
  {
     "meta": {
           "cost-type": {
               "cost-mode": "numerical", "cost-metric": "availbw"
           }
           "calendar-response-attributes" : [
              "calendar-start-time" : Tue, 1 Mar 2017 13:00:00 GMT,
              "time-interval-size" : "1 hour",
              "numb-intervals" : 8
         ]
     },

     "endpoint-cost-map": {
             "ipv4:192.0.2.2" {
             "ipv4:192.0.2.89" : [6,5,7,8,4,10,7,6],
             "ipv4:198.51.100.34" : [7,4,6,8,5,9,6,7],
             "ipv6:2000::1:2345:6789:abcd" : [7,6,8,5,7,9,6,8],
             }
           }
  }
```

8.4. Cost Metric: Link Utilized Bandwidth

Metric name:

Utilized Bandwidth

Metric Description:

To specify spatial and temporal utilized bandwidth over the specified source and destination. The value is corresponding to the actual measured bandwidth used for all traffic (motivated from [RFC7810], Sec.4.7.). The spatial aggregation level is specified in the query context (e.g., PID to PID, or endpoint to endpoint).

Method of Measurement or Calculation:

Link Utilizated bandwidth is Unidirectional utilization bandwidth measured between two directly connected IS-IS neighbors or OSPF neighbor, See section 4.7 of [RFC7810] for Measurement Method.

Units of Measurement:

The unit of measurement is byte per seconds.

Measurement Point(s) with Potential Measurement Domain:

See section 2.1, Data sources.

Measurement Timing:

Link Utilized bandwidth is Unidirectional utilization bandwidth measured between two directly connected IS-IS neighbors or OSPF neighbor, See section 5 of [RFC7810] for Measurement Timing.

Use and Applications:

See section 3 for use and application.

Example 10: utilbw value on source-destination endpoint pairs

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

 {
  "cost-type": {"cost-mode" : "numerical",
  "cost-metric" :  "utilbw"},
  "endpoints":  {
       "srcs" : [ "ipv4 : 192.0.2.2" ],
       "dsts" : [
         "ipv4:192.0.2.89",
         "ipv4:198.51.100.34",
         "ipv6:2000::1:2345:6789:abcd"
      ]
    }
 }

HTTP/1.1 200 OK
Content-Length: TBA
Content-Type: application/alto-endpointcost+json
 {
  "meta": {
         "cost type": {
         "cost-mode": "numerical",
         "cost-metric": "utilbw"
        }
  },
"endpoint-cost-map": {
          "ipv4:192.0.2.2" {
          "ipv4:192.0.2.89" :   0,
          "ipv4:198.51.100.34" : 2000,
          "ipv6:2000::1:2345:6789:abcd" :  5000,
                        }
        }
}
```

9. Security Considerations

The properties defined in this document present no security considerations beyond those in Section 15 of the base ALTO specification [ALTO].

However concerns addressed in Sections "15.1 Authenticity and Integrity of ALTO Information", "15.2 Potential Undesirable Guidance

from Authenticated ALTO Information" and "15.3 Confidentiality of ALTO Information" remain of utmost importance. Indeed, TE performance is a highly sensitive ISP information and sharing TE metric values in numerical mode requires full mutual confidence between the entities managing the ALTO Server and Client. Numerical TE performance information will most likely be distributed by ALTO Servers to Clients under strict and formal mutual trust agreements. On the other hand, ALTO Clients must be cognizant on the risks attached to such information that they would have acquired outside formal conditions of mutual trust.

10. IANA Considerations

IANA has created and now maintains the "ALTO Cost Metric Registry", listed in Section 14.2, Table 3 of [RFC7285]. This registry is located at <http://www.iana.org/assignments/alto-protocol/alto-protocol.xhtml#cost-metrics>. This document requests to add the following entries to "ALTO Cost Meric Registry".

Namespace	Property	Reference
	owdelay	[thisdraft] Section 3,[RFC2679] Section 3.6
	rtt	[thisdraft] Section 4,[RFC2681],Section 2.6
	pdv	[thisdraft] Section 5,[RFC3393],Section 2.6
	hopcount	[thisdraft] Section 6,[RFC7285]
	pktloss	[thisdraft] Section 7,[RFC7680],Section 2.6
	maxresbw	[thisdraft] Section 8.1,[RFC5305],Section 3.5
	residbw	[thisdraft] Section 8.2,[RFC7810],Section 4.5
	availbw	[thisdraft] Section 8.3,[RFC7810],Section 4.6
	utilbw	[thisdraft] Section 8.4,[RFC7810,Section4.7]

11. References

11.1. Normative References

[I-D.ietf-idr-te-pm-bgp]
Previdi, S., Wu, Q., Gredler, H., Ray, S., jefftant@gmail.com, j., Filsfils, C., and L. Ginsberg, "BGP-LS Advertisement of IGP Traffic Engineering Performance Metric Extensions", draft-ietf-idr-te-pm-bgp-04 (work in progress), October 2016.

[I-D.ietf-ippm-initial-registry]
Morton, A., Bagnulo, M., Eardley, P., and K. D'Souza, "Initial Performance Metric Registry Entries", draft-ietf-ippm-initial-registry-02 (work in progress), October 2016.

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

[RFC2679] Almes, G., Kalidindi, S., and M. Zekauskas, "A One-way Delay Metric for IPPM", RFC 2679, DOI 10.17487/RFC2679, September 1999, <http://www.rfc-editor.org/info/rfc2679>.

[RFC2681] Almes, G., Kalidindi, S., and M. Zekauskas, "A Round-trip Delay Metric for IPPM", RFC 2681, DOI 10.17487/RFC2681, September 1999, <http://www.rfc-editor.org/info/rfc2681>.

[RFC3393] Demichelis, C. and P. Chimento, "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)", RFC 3393, DOI 10.17487/RFC3393, November 2002, <http://www.rfc-editor.org/info/rfc3393>.

[RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, DOI 10.17487/RFC4627, July 2006, <http://www.rfc-editor.org/info/rfc4627>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <http://www.rfc-editor.org/info/rfc5234>.

[RFC5305] Li, T. and H. Smit, "IS-IS Extensions for Traffic Engineering", RFC 5305, DOI 10.17487/RFC5305, October 2008, <http://www.rfc-editor.org/info/rfc5305>.

[RFC7285] Alimi, R., Ed., Penno, R., Ed., Yang, Y., Ed., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, DOI 10.17487/RFC7285, September 2014, <http://www.rfc-editor.org/info/rfc7285>.

[RFC7471] Giacalone, S., Ward, D., Drake, J., Atlas, A., and S. Previdi, "OSPF Traffic Engineering (TE) Metric Extensions", RFC 7471, DOI 10.17487/RFC7471, March 2015, <http://www.rfc-editor.org/info/rfc7471>.

[RFC7680] Almes, G., Kalidindi, S., Zekauskas, M., and A. Morton, Ed., "A One-Way Loss Metric for IP Performance Metrics (IPPM)", STD 82, RFC 7680, DOI 10.17487/RFC7680, January 2016, <http://www.rfc-editor.org/info/rfc7680>.

[RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <http://www.rfc-editor.org/info/rfc7752>.

[RFC7810] Previdi, S., Ed., Giacalone, S., Ward, D., Drake, J., and Q. Wu, "IS-IS Traffic Engineering (TE) Metric Extensions", RFC 7810, DOI 10.17487/RFC7810, May 2016, <http://www.rfc-editor.org/info/rfc7810>.

11.2. Informative References

[I-D.ietf-alto-deployments] Stiemerling, M., Kiesel, S., Scharf, M., Seidel, H., and S. Previdi, "ALTO Deployment Considerations", draft-ietf-alto-deployments-16 (work in progress), July 2016.

[RFC6390] Clark, A. and B. Claise, "Framework for Performance Metric Development", RFC 6390, July 2011.

Authors' Addresses

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

Email: bill.wu@huawei.com

Y. Richard Yang
Yale University
51 Prospect St
New Haven, CT 06520
USA

Email: yry@cs.yale.edu

Young Lee
Huawei
1700 Alma Drive, Suite 500
Plano, TX 75075
USA

Email: leeyoung@huawei.com

Dhruv Dhody
Huawei
Leela Palace
Bangalore, Karnataka 560008
INDIA

Email: dhruv.ietf@gmail.com

Sabine Randriamasy
Nokia Bell Labs
Route de Villejust
Nozay 91460
FRANCE

Email: sabine.randriamasy@nokia-bell-labs.com

ALTO
Internet-Draft
Intended status: Standards Track
Expires: September 14, 2017

S. Randriamasy
Nokia Bell Labs

March 13, 2017

ALTO Contextual Cost Values
draft-randriamasy-alto-cost-context-01

Abstract

The Application-Layer Traffic Optimization (ALTO) Service has defined network and cost maps to provide basic network information, where the cost maps allow only one JSON value for a requested metric.

This document introduces several protocol extensions to allow ALTO clients to support use cases such as context based connection selection in cellular networks and calendaring for unattended data. This document refers to other extension proposals posted in the ALTO WG that can support the present use cases as well. Likewise, some of the proposed extensions may serve other ALTO use cases.

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 September 14, 2017.

Table of Contents

1. Introduction 3
2. Use cases 4
2.1. Use Case 1: conditional RF costs in cellular networks . . 4
2.2. Use case 2: access-aware endpoint selection 5
3. Required ALTO extensions 6
4. Design options and examples 6
4.1. Overview of context features 7
4.1.1. Applicable ALTO services 7
4.2. Example IRD 7
4.3. Use case 1: Example scenario for the FCM Service 10
4.4. Design option: Network Map with cells as PIDs 11
4.4.1. Network Map: FCM Request for contextual 'RFcost' . . 11
4.4.2. Network Map: FCM Response for contextual RFcost . . . 12
4.5. Use case 2: example ALTO transactions for the ECS 13
4.5.1. Use case 2: example with logical context parameter combinations 13
5. Deployment case: local ALTO Server cascaded with global ALTO Server 16
5.1. Cascaded ALTO Servers with one network map each 16
6. IANA Considerations 17
7. Security Considerations 17
8. Acknowledgements 17
9. References 17
9.1. Normative References 17
9.2. Informative References 17
Appendix A. An Appendix 18
Author's Address 18

1. Introduction

The IETF ALTO protocol specified in [RFC7285] provides guidance to over the top applications which have to select one or several hosts or endpoints from a set of candidates that are able to provide a desired data resource, or which need some provider-centric insight on the cost of application paths to these endhosts. The ALTO Service has defined network and cost maps to provide basic network information, where the cost maps allow only one JSON value for a requested metric.

This draft brings a use case where providing different values for a same cost metric can help in optimizing the application path selection. Typically, when an end host can connect to the network via multiple technologies or access points, the path performance for a metric may be accordingly impacted.

The present draft proposes to extend the cost information specified in [RFC7285] by providing, for a same cost metric, several possible cost values. Which value to provide depends on qualitative criteria as opposed to quantitative criteria such as time. The purpose is to allow a finer grained decision on which application endpoint or sub network to access.

Previous ALTO WG discussions have suggested to introduce "the ability to "name" cost maps so that a single Information Resource Directory can link multiple cost maps with the same cost type to a single network map." The goal was to provide, for a given cost metric, multiple cost values depending on qualitative conditions named "circumstance", where a circumstance reflects a given policy.

For applications such as video download or streaming, a user equipment (UE) may use [RFC7285] to choose the best possible application resource location.

Currently, the insight of ALTO information on the path between a UE and a connection node (or say Endpoint) does not provide details below IP hops. However the major QoE challenges of wireless network users arise in the access network, that is, in the first hop between the UE and its one or more serving packet data network gateway (PGW). The path of a UE to its serving PGW(s) impacts the path to the content and thus the related QoE. Therefore, it is necessary to inform the UE, which could take the appropriate decisions w.r.t. the utilized access path. The access technology in current ALTO documents is accounted at the content location (last hop) side by distinguishing whether the requested content is located in a fixed or a wireless access network, as described in [draft-ietf-alto-deployment]

This document introduces several protocol extensions to allow ALTO clients to support use cases such as context-based connection selection in cellular networks and calendaring for unattended data. This document refers to other extension proposals posted in the ALTO WG that can support the present use cases as well. Likewise, some of the proposed extensions may serve other ALTO use cases.

2. Use cases

This section presents motivating use cases for contextual ALTO Costs with a focus on conditional RF costs in cellular networks. In these 2 use cases, a terminal UE is located in a LTE network and associated to a "local" ALTO Server(LAOS) that covers this access network, say up to the Packet Data Network (PDN) Gateway PGW and can itself connect to another ALTO Server having a more global view covering up to the whole ISP network. Such a deployment is proposed in section Section 5 of this draft.

2.1. Use Case 1: conditional RF costs in cellular networks

Let’s assume a terminal UE located in a cellular network. An ALTO Client (LAOC) associated to the UE queries the local ALTO Server in order to know via which cell it should connect to the network. So in a first place, LAOC will query the connection cost associated to cells C1,.. CK. This example assumes that this cost is a unitless value abstracting a (RF) cost to a cell. Our example however includes 2 additional considerations:

- the RF cost to a cell may be impacted by its load,

- a UE usually transmits a fair amount of "unattended data" (UD).

UD is considered in one of the key features for LTE enhancements in Release 13 and defined in 3GPP TS22.101 as follows: "Unattended Data Traffic : Data traffic of which the user is unaware he/she initiated, e.g. based on the screen/keypad lock being activated, length of time since the UE last received any input from the user, known type of app (e.g. an application monitoring a user’s health "mHealth" may need its data never treated as Unattended Data Traffic.)". UD traffic is often delay tolerant and it would be beneficial for the network if the UE can schedule its transmission. To this end, the UE can use an instant UD Indicator (UDI) sent by the LTE network. The UDI, accepted for LTE Release 13 is a single bit sent to the UE indicating whether UD in a cell is allowed (UDA) or not (UDNA). The status change of a UDI from UDA to UDNA is presumably triggered when the cell load exceeds a given threshold T(udna). The value of T(udna) may change across cells and in time but is not provided to UEs. If the UE had an ALTO calendar for either T(udna) or for the abstracted

cell load values, it could appropriately schedule the transmission of its UD, that will have to occur anyway. The UE could combine this calendar with the UDI it receives from the cellular network. The UDI state may change within sub-seconds and impact the data exchange. What is missing in the provided LTE information is:

- knowing whether the UDI threshold relates to downlink or uplink congestion.

- knowing the level of congestion that triggers a change in UDI and how it may evolve in time.

The UE thus can advatageously combine the non-real time ALTO information with the real-time UDI provided by the LTE network. The present draft illustrates how ALTO can fill these gaps with the support of:

- ALTO Cost Calendars,

- the proposed protocol extension providing context-dependent ALTO Cost values.

In this use case: ALTO calendars need to be requested via for the ALTO Filtered Cost Map (FCM) Service, the context parameters impacting the cost values are: "uda" (Unattended Data Allowed), "udna" (Unattended Data Not Allowed), "uplink", "downlink".

2.2. Use case 2: access-aware endpoint selection

In a second use case, an end-system called UEP is located in a LTE network and may connect via several access technologies, e.g. Cellular or WiFi. UEP may also benefit from a given Service Level Agreement SLA-m. Other parameters may characterize the UEP generated traffic.

Currently the insight of ALTO information in the path between a UE and a connection node (or say Endpoint) does not provide details below IP hops. However the major QoE challenges of wireless network users arise in the access network, that is, in the first hop between the UE and its one or more serving packet data network gateway (PGW). The path of a UE to its serving PGW(s) impacts the path to the content and thus the related QoE. Therefore, it is necessary to inform the UE, which could take the appropriate decisions w.r.t. the utilized access path. The access technology in current ALTO proposals is accounted at the content location (last hop) side by distinguishing whether the requested content is located in a fixed or a wireless access network, as described in [draft-ietf-alto-deployments] that states: "For ISPs with mobile network and fixed

network, the traffic optimizing problems they focus will be optimizing the mobile traffic, except problems on last hop section."

For Mobile Network Operators (MNO) and their users, being connected via e.g. cellular or trusted Wifi can hugely impact the QoE and routing cost. Sometimes a 4G connection is preferable for users than a poor WiFi connection although potentially more expensive. Sometimes, MNOs have spare data resources or offer them for given SLAs. For both parties, access-aware Endpoint selection for Users is thus beneficial. One way to achieve this is that ALTO provides cost values depending on qualitative contextual parameters such as access technology and the access technology and SLA.

3. Required ALTO extensions

The aforementionned use cases can be supported with a few simple extensions to the ALTO protocol. A number of them have already been discussed in other WG drafts and use cases. The proposed extensions include:

- Cost value context parameters: a capability to allow exposing several possible context-dependent values for one metric, as proposed in the present document,

- Entities with associated domain and properties for cellular and wireless networks, that could be added to [draft-roome-alto-unified-props],

- Cost metrics for cellular and wireless networks: these features would extend current proposals in the WG,that could be added to [draft-ietf-alto-performance-metrics],

- Extended input for the Filtered Cost Map Service: to allow the input to comprise several(source-array, destination-array) pairs, as it has been proposed in [draft-yang-alto-path-vector].

4. Design options and examples

Similarly to Multi-Cost and Cost Calendar ([draft-ietf-alto-cost-calendar]), this proposal does not introduce new cost modes or new media-types. It ensures backwards compatibility with legacy ALTO Clients, that is: "A legacy ALTO Client must be able to send legacy requests to a Cost Context aware ALTO Server and get legacy responses as specified in RFC7285".

"A Cost Context aware ALTO Server must be able to receive and process requests sent by legacy ALTO Clients, as specified in RFC 7285".

Besides, the proposed extension is designed to be compatible with Multi-Cost ALTO and ALTO Cost Calendars ([draft-ietf-alto-cost-calendar]).

In the present draft version, the IRD indicates the supported context parameters as values encoded in JSON strings. The idea is that this design simplifies the transactions, as it applies to context attributes that take a limited number of values, say 1 to 5. Context attributes taking numerous or unpredictible values should be handled as values properties or metrics expressed in constraints.

4.1. Overview of context features

Cost context attributes are strings with values such as "wifi", "cellular", "uda".

Cost context attributes are indicated in the IRD as capabilities of an information resource. They are associated to cost type names.

4.1.1. Applicable ALTO services

Draft [draft-bertz-alto-mobilitynets] proposes to identify network points of attachment (PoA) such as cells to PIDs, as PoAs are endpoint types not currently supported in ALTO. The current proposal is to represent cellular PIDs in an ALTO Network Map with no routes. PID properties as specified in [draft-roome-alto-unified-props] could be used to indicate the type of the PoA, together with other properties. ALTO properties are well suited for almost static attributes such as access type.

To indicate connection properties with frequently changing values such as RF Cost, load or congestion, the ALTO Filtered Cost Map service can be used. Connection properties may also be conveyed with the Endpoint property service or its extensions defined in [draft-roome-alto-unified-props].

Costs and properties with the extensions proposed in this document may be conveyed with different values depending on the context parameter. The present version of this draft focuses on context patrameters associated to costs.

4.2. Example IRD

The purpose of ALTO is to guide the behavior of the end systems or applications without the need for networks to explicitely expose their performance values. In this example, the IRD does not expose the real load percentage of a cell to UE. Instead, it abstracts the cell congestion by a metric called ’RFcost’ represented by a number

between 0 and 100. The values of ’RFcost’ are provided as a an ALTO Calendar as specified in [draft-ietf-alto-cost-calendar-00] in shorter time intervals. In addition they differ, depending on the the context values "uda" and "udna".

Besides, the IRD provides metric ’routingcost’ as a MUST specfied in [RFC7285], that may represent a more administrative or monetary access cost.

The IRD could publish the capability of a resource to provide context dependent ’routingcost’ values as expressed for resource "filtered-cost-calendar-map".

```
   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-RFcost": {
              "cost-mode"  : "numerical",
              "cost-metric": "RFcost",
           }
         }
         ... other meta ...
     },
     "resources" : {
           "filtered-cost-calendar-map" : {
           "uri" : "http://alto.local.example.com/costmap/filtered/calendar/cont
ext",
           "media-types" : [ "application/alto-endpointcost+json" ],
           "accepts" : [ "application/alto-endpointcostparams+json" ],
           "capabilities" : {
              "cost-constraints" : true,
              "cost-type-names" : [ "num-routingcost",
                                    "num-RFcost"],// ++NEW
              "calendar-attributes" : [
                 {"cost-type-names" : "num-routingcost",
                  "time-interval-size" : "1 hour",
                  "number-of-intervals" : 24}, // MAY ALSO BE SINGLE VALUE
                  {"cost-type-names" : "num-RFcost", // ++NEW
                  "time-interval-size" : "5 minute",
                  "number-of-intervals" : 12}
              ],
              "cost-context" : [ // ++NEW
                 {"cost-type-names" : "num-RFcost",
                  "context-params" : ["uda", "udna", "uplink", "downlink"]
                  }
                ] // ++NEW
              "uses": [ "my-default-network-map" ]
                    } // end ECM capab
        ... other resources ...
     } // end resources
   } // end IRD
```

4.3. Use case 1: Example scenario for the FCM Service

We assume an example scenario where a UE has the choice to connect to 2 cells C1 and C2.

As suggested in [draft-bertz-alto-mobilitynets], we may represent the cellular topology with an ALTO Network Map comprising PIDs representing the cells and named "Cell1", "Cell2", ... "Celln". A format for a cell identifier has been proposed in [draft-rauschenbach-alto-wireless-access] and is not being discussed here.

As a Network Map may cover a large number of cells, the Filtered Cost Map Service can be used to reduce data exchange and get information on a restricted number of cells, say PID1 and PID2.

We assume that the ALTO Client in the UE wants to get calendared values for ALTO metric "RFcost" in order to appropriately schedule its unattended data transmission. The ALTO information resource 'ALTO Calendar' provides an array of time-dependent cost values and is being specified in [draft-ietf-alto-cost-calendar]. In addition, the ALTO Client wants these values for both the "uda" and "udna" context. Last, we assume that the UE needs the Cost values for both the uplink (UE to Cell-k) and downlink (Cell-k to UE) directions. We assume that the UE is located in the PID called "Cell1".

In this scenario, C1 is limited by its uplink capacity, C2 is limited by its downlink capacity. ALTO can be used to convey the following information:

At time interval T1 of the next Calendar:

- if C1 indicates "unattended data allowed" the downlink RF cost is 20, and the uplink RF cost is 70

- if C1 indicates "unattended data NOT allowed", the downlink RF cost is 20, and the uplink RF cost is 90

- if C2 indicates "unattended data allowed" the downlink RF cost is 70, and the uplink RF cost is 20

- if C2 indicates "unattended data NOT allowed", the downlink RF cost is 90, in the uplink RF cost is 20.

The ALTO Calendar provides values for the other 11 time intervals Ti.

4.4. Design option: Network Map with cells as PIDs

In this design, the cellular topology is represented with an ALTO Network Map comprising PIDs named "Cell1", "Cell2", ... "Celln". The UE is located in one of these PIDs. A Cost Map is associated to this Network Map and conveys metrics indicated in the IRD. The Cost Map is requested to convey connection costs between firstly the UE to its serving cell (that is the PID to itself) and secondly the UE and neighboring cells (that is the PID to another one) and last, for both uplink and downlink directions.

The ALTO Server can regularly update the Cost Map and send filtered information to the ALTO Client. The proposed IRD design announces additional context attributes "uplink", "downlink". In this case and other potential cases, the context parameters need to be arranged w.r.t. their possible combinations (to be further specified in the IRD). For example, the IRD may announce that costs are provided for contexts "uda" and "udna" and this in both contexts "uplink" and "downlink". Or that costs are provided for contexts "uplink" and "downlink" and this in both contexts "udna" and "uda". In such a case, the IRD capability member may list the possible combinations in the capabilities as follows:

```
"cost-context" : [ // ++NEW
   { "cost-type-names" : "num-RFcost",
     "context-params"[["uda", "uplink"],
                      ["uda", "downlink"],
                      ["udna", "uplink"],
                      ["udna", "downlink"]] // ++NEW
    }
  ]
```

This arrangement indicates that for the metric named "num-RFcost", the ALTO Server can provide 4 different values: v1 for ["uda" AND "uplink"], ... v4 for ["udna" AND "downlink"].

Further versions may specify more elaborated logical combinations of context parameters.

4.4.1. Network Map: FCM Request for contextual ’RFcost’

The ALTO Client can specify the desired cost value context parameters in the request input. In particular, it can select one or more combinations indicated in the IRD. Its input parameter "cost-context-params" is an array of all the desired combinations. In this

example, the ALTO Client wants 4 values, corresponding to all the indicated combinations.

```
POST /costmap/filtered/calendar/context 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": "RFcost"},
  "calendared" : true,
  "context-params" : [["uda", "uplink"], // ++NEW
                      ["uda", "downlink"],
                      ["udna", "uplink"],
                      ["udna", "downlink"]],
  "pids" : [
    {"srcs" : [ "Cell1"], "dsts" : [ "Cell1", "Cell2"]},
    {"srcs" : [ "Cell2"], "dsts" : [ "Cell1", "Cell2"]}
  ]
}
```

4.4.2. Network Map: FCM Response for contextual RFcost

The ALTO response provides, for each requested ("src", "dest") pair, a calendar of 12 JSON values, where each is an array of cost values arranged as specified in the "meta" of the ALTO response.

```
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": "RFcost"},
    "calendar-response-attributes" :
       { "calendar-start-time" : Tue, 1 Sept 2016 13:00:00 GMT,
         "time-interval-size" : "5 minute",
         "numb-intervals" : 12},
         "context-params" : [["uda", "uplink"], // ++NEW
                             ["uda", "downlink"],
                             ["udna", "uplink"],
                             ["udna", "downlink"]]
   } // end meta
   "cost-map" : {
      "Cell1": { "Cell1": [[70, 20, 90, 20], ... ,[50, 20, 70, 20]],
      "Cell2": { "Cell2": [[20, 70, 20, 90], ... ,[20, 50, 20, 70]]
   }
}
```

4.5. Use case 2: example ALTO transactions for the ECS

In this use case, the UE requests the ECS to a local ALTO server for the routingcost to the PGW and wants the metric values varying w.r.t. the "access-type" and "SLA-id". Note that the "context" related design feature can be easily transposed for the Cost Map Service.

4.5.1. Use case 2: example with logical context parameter combinations

This section proposes a design, allowing a Client to arrange input context parameters in logical combinations. The purpose is to show how such combinations of context parameters avoids specifying as many metrics and moderates the amount of exchanged data.

In this example the ALTO Server indicates in its IRD that it can provide endpoint cost maps for metrics "routingcost" and "bandwidthscore". Values for metric "routingcost" are provided w.r.t. 2 types of context parameters. The ALTO Client may query values for metric "routingcost" for either of these types of parameters or both or none.

For each type, the parameters are listed in an array. We have 2 arrays:

- ["cell", "wifi", "lan"]

- ["SLA-1", "SLA-2", "SLA-3"]

This indicates that in each array, the client can pick one or more parameters and combine them with one or more parameters in the second array. The ALTO Server will provide costs w.r.t. the AND combination accross and within arrays.

In the present example, if the Client requests cost values for the combination:

[["cell", "wifi"], ["SLA-3"]]

The server will provide 2 values: one for ("cell" AND "SLA-3")and the second one for ("wifi" and "SLA"-3").

4.5.1.1. Example IRD with logical context parameter combinations

The IRD below specifies the possibility to combine parameters from the two arrays of the example above.

```
     "resources" : {
           "filtered-cost-calendar-map" : {
           "uri" : "http://alto.local.example.com/endpointcostmap/lookup/context
",
           "media-types" : [ "application/alto-endpointcost+json" ],
           "accepts" : [ "application/alto-endpointcostparams+json" ],
           "capabilities" : {
              "cost-constraints" : true,
              "cost-type-names" : [ "num-routingcost",
                                    "num-bandwidthscore"],
              "cost-context" : [// ++NEW
                {"cost-type-names" : "num-routingcost",
                 "context-params" : [["cell", "wifi", "lan"],
                                     ["SLA-1", "SLA-2", "SLA-3"]]
                 }
                ]
                    } // end ECM capab
          ... other resources ...
     } // end resources
```

4.5.1.2. Use case 2: example ECS request with logical context parameter combinations

The ALTO Client queries the ECS between 2 endpoints for the following combinations: ("cell" AND "SLA-3")and ("wifi" and "SLA"-3") and thus arranges its input context parameters as follows:

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

{
  "cost-type" : {"cost-mode" : "numerical", "cost-metric" : "routingcost"},
  "context-params" : [["cell", "wifi"], ["SLA-3"]],
  "endpoints" : {
    "srcs": [ "ipv4:192.0.2.2" ],
    "dsts": [
      "ipv4:192.0.2.89",
      "ipv6:2000::1:2345:6789:abcd"
    ]
  }
}
```

4.5.1.3. Use case 2: example ECS response with logical context parameter combinations

Following the ALTO Client request of the above example, The ALTO Server provides a response as follows:

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

       {
        "meta" : {
          "cost-type" : {"cost-mode" : "numerical", "cost-metric" : "routingcost
"},
          "context-params" : [["cell", "wifi"], ["SLA-3"]]
        } // end meta

        "endpoint-cost-map" : {
          "ipv4:192.0.2.2": {
            "ipv4:192.0.2.89" : [10, 4],
            "ipv6:2000::1:2345:6789:abcd" : [4, 6]
          }
        }
       }
```

5. Deployment case: local ALTO Server cascaded with global ALTO Server

To maintain scalability, the ALTO coverage network zone can be decomposed in one "local"ALTO Server part covering a restricted local network zone, for instance within the first IP hop range and another "global" part covering the rest of the ISP network, similarly to what is proposed in [draft-ietf-alto-deployments]. The local ALTO server may include the guidance given by the ISP ALTO server and compose it with the "global" guidance in its replies to its ALTO clients. Recent ALTO WG discussions open the possibility for one IRD to indicate multiple network maps having different levels of detail.

5.1. Cascaded ALTO Servers with one network map each

In the "cascaded" use case, the ALTO Service is preferably distributed among two ALTO Servers as follows:

The ALTO Client serving the UE is referred to as the LAOC and can be located either in the UE or in the network.

1. A Local ALTO Server (LAOS)

   * Hosts the information on the local EPS network, covering the paths between e.g. the UEs and the cells or the PGWs,

   * Hosts an ALTO Client that sends an ALTO request to a "global" ALTO Server, covering the zone beyond the PGW. It can possibly get the global Server updates using the extensions specified in [draft-ietf-alto-incr-update-sse].

* receives the ALTO request issued by the ALTO Client associated to the UE.

2. a "core" ALTO Server covers the whole ISP network view, as it would if the "local ALTO Service" is not available or deactivated.

6. IANA Considerations

This document makes no request of IANA.

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

7. Security Considerations

8. Acknowledgements

9. References

9.1. Normative References

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

[RFC7285] 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.

9.2. Informative References

[draft-bertz-alto-mobilitynets] Bertz, L., "Mobility Network Models in ALTO", October 2015.

[draft-ietf-alto-cost-calendar] Randriamasy, S., Yang, Y., Wu, Q., Deng, L., and N. Schwan, "ALTO Cost Calendar", February 2017.

[draft-ietf-alto-deployment] Stiemerling, M., Kiesel, S., Scharf, M., Seidel, H., and S. Previdi, "draft-ietf-alto-deployments-16", July 2016.

[draft-ietf-alto-incr-update-sse]
Roome, W. and Y. Yang, "ALTO Incremental Updates Using Server-Sent Events (SSE)", Septembre 2016.

[draft-ietf-alto-performance-metrics]
Wu, Q., Yang, Y., Lee, Y., Dhody, D., and S. Randriamasy, "ALTO Performance Cost Metrics", March 2017.

[draft-rauschenbach-alto-wireless-access]
Rauschenbach, U., "ALTO in wireless access networks", October 2014.

[draft-roome-alto-unified-props]
Roome, W., "Extensible Property Maps for the ALTO Protocol", July 2016.

[draft-yang-alto-path-vector]
Bernstein, G., Gao, K., Lee, Y., Roome, W., Scharf, M., and Y. Yang, "ALTO Extension: Path Vector Cost Mode", July 2016.

Appendix A. An Appendix

Author’s Address

Sabine Randriamasy
Nokia Bell Labs
Route de Villejust
Nozay 91460
FRANCE

Email: sabine.randriamasy@nokia-bell-labs.com

ALTO WG
Internet-Draft
Intended status: Standards Track
Expires: September 14, 2017

W. Roome
Nokia Bell Labs
R. Yang
Yale University

March 13, 2017

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

Abstract

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

Requirements Language

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

Status of This Memo

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 September 14, 2017.

Copyright Notice

Table of Contents

1. Introduction 3
2. Definitions and Concepts 4
2.1. Entities 4
2.2. Domains 4
2.3. Entity Addresses 4
2.4. Domain Names 5
2.5. Property Names 5
2.6. Relationship to Network Maps 6
3. Entity Domains 6
3.1. Internet Address Domains 7
3.1.1. IPV4 Domain 7
3.1.2. IPV6 Domain 7
3.1.3. Heirarchy And Inheritance 7
3.1.4. Relationship To Network Maps 8
3.2. PID Domain 9
3.2.1. Domain Name 9
3.2.2. Domain-Specific Entity Addresses 9
3.2.3. Heirarchy And Inheritance 9
3.2.4. Relationship To Internet Addresses Domains 9
3.3. Internet Address Properties vs. PID Properties 9
4. Property Map Resource 10
4.1. Media Type 10
4.2. HTTP Method 10
4.3. Accept Input Parameters 10
4.4. Capabilities 10
4.5. Uses 10
4.6. Response 10
5. Filtered Property Map Resource 12
5.1. Media Type 12
5.2. HTTP Method 12
5.3. Accept Input Parameters 12
5.4. Capabilities 13
5.5. Uses 13
5.6. Response 13
6. Impact On Legacy Servers And Clients 13
6.1. Impact on Endpoint Property Service 13
6.2. Impact on Resource-Specific Properties 13
6.3. Impact on the "pid" Property 14
6.4. Impact on Other Properties 14

7. Examples 14
7.1. Network Map 14
7.2. Property Definitions 15
7.3. Information Resource Directory (IRD) 15
7.4. Property Map Example 17
7.5. Filtered Property Map Example #1 17
7.6. Filtered Property Map Example #2 18
7.7. Filtered Property Map Example #3 19
8. Security Considerations 20
9. IANA Considerations 21
9.1. application/alto-* Media Types 21
9.2. ALTO Entity Domain Registry 22
9.3. ALTO Endpoint Property Type Registry 23
10. References 24
Authors' Addresses 24

1. Introduction

The ALTO protocol [RFC7285] introduced the concept of "properties" attached to "endpoint addresses," and defined the Endpoint Property Service (EPS) to allow clients to retrieve those properties. While useful, the EPS, as defined in RFC7285, has at least two limitations.

First, it only allows properties to be associated with a particular domain of entities, namely individual IP addresses. It is reasonable to think that collections of endpoints, as defined by CIDRs or PIDs, may also have properties. Furthermore, recent proposals ([ID-draft-yang-alto-path-vector-02] and [ID-draft-yang-alto-topology-06]) have suggested new classes of entities with properties. The EPS cannot be extended to new entity domains. Instead, new services, with new request and response messages, would have to be defined for each new entity domain.

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

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

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

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

2. Definitions and Concepts

2.1. Entities

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

2.2. Domains

A domain is a family of entities. Examples are the internet address and PID domains (see Section 3.1 and Section 3.2). Another example is the proposed domain of Abstract Network Elements associated with topology and routing, as suggested by [ID-draft-yang-alto-path-vector-02].

2.3. Entity Addresses

Each entity has a unique name of the form:

domain-name : domain-specific-entity-address

Examples from the IP domain include individual addresses such as "ipv4:192.0.2.14" and "ipv6:2001:db8::12", as well as address blocks such as "ipv4:192.0.2.0/26" and "ipv6:2001:db8:1/48".

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

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

Note that entity addresses do NOT have a unique textual representation. For example, the strings "ipv6:2001:db8::1" and "ipv6:2001:db8:0:0:0:0:0:1" refer to the same entity.

2.4. Domain Names

Each domain has a unique name. A domain name MUST be no more than 32 characters, and MUST NOT contain characters other than US-ASCII alphanumeric characters (U+0030-U+0039, U+0041-U+005A, and U+0061-U+007A), hyphen ('-', U+002D), and low line ('_', U+005F). For example, the names "ipv4" and "ipv6" identify objects in the internet address domain (Section 3.1).

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

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

2.5. Property Names

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

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

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

necessary to represent the bounding box of a set of hosts, another property, such as "geo-region", should be defined.

2.6. Relationship to Network Maps

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

This document takes a different approach. Instead of defining the dependency by qualifying the property name, this document attaches the dependency to the property map as a whole. Thus all properties in a given property map depend on the same resource. Furthermore, entity addresses may depend on a network map (for example, the Abstract Network Elements suggested by [ID-draft-yang-alto-path-vector-02]). Associating the dependency with the property map handles any entity address dependencies as well.

The "uses" field in an IRD entry defines the dependencies of a property map resource, and the "dependent-vtags" field in a property map response defines the dependencies of that map. These fields are defined in Sections 9.1.5 and 11.1 of [RFC7285], respectively.

This is similar to how RFC7285 handles dependencies between cost metrics and network maps. Recall that cost maps present the costs between PIDs, and PID names depend on a network map. If an ALTO server provides the "routingcost" metric for the network maps "net1" and "net2", then the server defines two separate cost maps, one for "net1" and the other for "net2".

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

3. Entity Domains

This document defines the following entity domains.

3.1. Internet Address Domains

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

3.1.1. IPV4 Domain

3.1.1.1. Domain Name

ipv4

3.1.1.2. Domain-Specific Entity Addresses

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

For the purpose of defining properties, an individual internet address and the corresponding full-length prefix are considered aliases for the same entity. Thus "ipv4:192.0.2.0" and "ipv4:192.0.2.0/32" are equivalent, and have the same set of properties, as are "ipv6:2001:db8::" and "ipv6:2001:db8::/128".

3.1.2. IPV6 Domain

3.1.2.1. Domain Name

ipv6

3.1.2.2. Domain-Specific Entity Addresses

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

For the purpose of defining properties, an individual internet address and the corresponding 128-bit prefix are considered aliases for the same entity. That is, "ipv6:2001:db8::1" and "ipv:2001:db8::1/128" are equivalent, and have the same set of properties.

3.1.3. Heirarchy And Inheritance

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

such block defines a value for P, IP inherits the value of P in the block with the longest prefix.

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

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

```
ipv4:192.0.2.0/26: P=v1
ipv4:192.0.2.0/28: P=v2
ipv4:192.0.2.0/30: P=v3
ipv4:192.0.2.0:    P=v4
```

Defined Property Values

Then the following entities have the indicated values:

```
ipv4:192.0.2.0:    P=v4
ipv4:192.0.2.1:    P=v3
ipv4:192.0.2.16:   P=v2
ipv4:192.0.2.32:   P=v1
ipv4:192.0.2.64:   (not defined)
ipv4:192.0.2.0/32: P=v4
ipv4:192.0.2.0/31: P=v3
ipv4:192.0.2.0/29: P=v2
ipv4:192.0.2.0/27: P=v1
ipv4:192.0.2.0/25: (not defined)
```

Inherited Property Values

3.1.4. Relationship To Network Maps

An internet address domain may or may not be associated with an ALTO network map resource. Logically, there is a map of internet address entities to property values for each network map defined by the ALTO server, plus an additional property map for internet address entities which are not associated with a network map. These maps are separate from each other. The prefixes in the property map do not have to correspond to the prefixes defining the network map's PIDs. For example, the property map for a network map may assign properties to "ipv4:192.0.2.0/24" even if that prefix is not associated with any PID in the network map.

3.2. PID Domain

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

3.2.1. Domain Name

pid

3.2.2. Domain-Specific Entity Addresses

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

3.2.3. Heirarchy And Inheritance

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

3.2.4. Relationship To Internet Addresses Domains

The PID domain and the internet address domains are completely independent; the properties associated with a PID have no relation to the properties associated with the prefixes or endpoint addresses in that PID. An ALTO server MAY choose to assign some or all properties of a PID to the prefixes in that PID, but is not required to do so.

For example, suppose "PID1" consists of the prefix "ipv4:192.0.2.0/24", and has the property "P" with value "v1". The internet address entities "ipv4:192.0.2.0" and "ipv4:192.0.2.0/24" may or may not have a value for the property "P", and if they do, it is not necessarily "v1".

3.3. Internet Address Properties vs. PID Properties

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

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

For example, because internet addresses are allocated to server providers by blocks of prefixes, an "ISP" property would be best associated with the internet address domain. On the other hand, a

property that explains why a PID was formed, or how it relates the a provider's network, would best be associated with the PID domain.

4. Property Map Resource

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

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

4.1. Media Type

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

4.2. HTTP Method

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

4.3. Accept Input Parameters

None.

4.4. Capabilities

The capabilities are defined by an object of type PropertyMapCapabilities:

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

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

4.5. Uses

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

4.6. Response

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

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

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

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

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

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

Specifically, a PropertyMapData object has one member for each entity in the Property Map. The entity's properties are encoded in the corresponding EntityProps object. EntityProps encodes one name/value pair for each property, where the property names are encoded as strings of type PropertyName. A protocol implementation SHOULD assume that the property value is either a JSONString or a JSON "null" value, and fail to parse if it is not, unless the implementation is using an extension to this document that indicates when and how property values of other data types are signaled.

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

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

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

5. Filtered Property Map Resource

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

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

5.1. Media Type

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

5.2. HTTP Method

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

5.3. Accept Input Parameters

The input parameters for a Filtered Property Map request are supplied in the entity body of the POST request. This document specifies the input parameters with a data format indicated by the media type "application/alto-propmapparams+json", which is a JSON object of type ReqFilteredPropertyMap:

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

with fields:

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

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

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

5.4. Capabilities

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

5.5. Uses

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

5.6. Response

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

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

6. Impact On Legacy Servers And Clients

6.1. Impact on Endpoint Property Service

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

6.2. Impact on Resource-Specific Properties

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

6.3. Impact on the "pid" Property

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

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

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

Second, a client MAY request the "pid" property for a block of addresses. An ALTO server determines the value of "pid" for an address block C as follows. Let CS be the set of all address blocks in the network map. If C is in CS, then the value of "pid" is the name of the PID associated with C. Otherwise, find the longest block C' in CS such that C' prefix-matches C, but is shorter than C. If there is such a block C', the value of "pid" is the name of the PID associated with C'. If not, then "pid" has no value for block C.

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

6.4. Impact on Other Properties

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

7. Examples

7.1. Network Map

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

```
defaultpid:  ipv4:0.0.0.0/0  ipv6::0/0
pid1:        ipv4:192.0.2.0/25
pid2:        ipv4:192.0.2.0/28  ipv4:192.0.2.16/28
```

Figure 1: Example Network Map

7.2. Property Definitions

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

```
                          ISP    ASN   country   state
ipv4:192.0.2.0/24:  BitsRus      -        us       -
ipv4:192.0.2.0/28:        -  12345         -      NJ
ipv4:192.0.2.16/28:       -  12345         -      CT
ipv4:192.0.2.0:           -      -         -      PA
```

Figure 2: Example Property Values

7.3. Information Resource Directory (IRD)

The following IRD defines the relevant resources of the ALTO server. It provides two Property Map resources, one for the "ISP" and "ASN" properties, and another for the "country" and "state" properties. The server could have provided a Property Map resource for all four properties, but did not, presumably because the organization that runs the ALTO server believes any given client is not interested in all four properties.

The server provides two Filtered Property Maps. The first returns all four properties, and the second just returns the "pid" property for the default network map.

The Property Maps for the "ISP", "ASN", "country" and "state" properties do not depend on the default network map (they do not have a "uses" capability), because the definitions of those properties do not depend on the default network map. The Filtered Property Map for the "pid" property does have a "uses" capability for the default network map, because that defines the values of the "pid" property.

Note that for legacy clients, the ALTO server provides an Endpoint Property Service for the "pid" property for the default network map.

```
  "meta": { ... },
  "resources" : {
     "default-network-map" : {
        "uri" : "http://alto.example.com/networkmap",
        "media-type" : "application/alto-networkmap+json"
```

```
        },
        .... cost map resources ....
        "country-state-property-map" : {
           "uri" : "http://alto.example.com/propmap/full/inet-cs",
           "media-type" : "application/alto-propmap+json",
           "capabilities" : {
             "domain-types": [ "ipv4", "ipv6" ],
             "prop-types" : [  "country", "state" ]
           }
        },
        "isp-asn-property-map" : {
           "uri" : "http://alto.example.com/propmap/full/inet-ia",
           "media-type" : "application/alto-propmap+json",
           "capabilities" : {
             "domain-types": [ "ipv4", "ipv6" ],
             "prop-types" : [ "ISP", "ASN" ]
           }
        },
        "iacs-property-map" : {
           "uri" : "http://alto.example.com/propmap/lookup/inet-iacs",
           "media-type" : "application/alto-propmap+json",
           "accepts" : "application/alto-propmapparams+json",
           "capabilities" : {
             "domain-types": [ "ipv4", "ipv6" ],
             "prop-types" : [ "ISP", "ASN", "country", "state" ]
           }
        },
        "pid-property-map" : {
           "uri" : "http://alto.example.com/propmap/lookup/pid",
           "media-type" : "application/alto-propmap+json",
           "accepts" : "application/alto-propmapparams+json",
           "uses" : [ "default-network-map" ]
           "capabilities" : {
             "domain-types": [ "ipv4", "ipv6" ],
             "prop-types" : [ "pid" ]
           }
        },
        "legacy-pid-property" : {
           "uri" : "http://alto.example.com/legacy/eps-pid",
           "media-type" : "application/alto-endpointprop+json",
           "accepts" : "application/alto-endpointpropparams+json",
           "capabilities" : {
             "prop-types" : [ "default-network-map.pid" ]
           }
        }
     }
```

Example IRD

7.4. Property Map Example

The following example uses the properties and IRD defined above to retrieve a property map for entities with the "ISP" and "ASN" properties. Note that the response does not include the entity "ipv4:192.0.2.0", because it does not have a value for either of those properties. Also note that the entities "ipv4:192.0.2.0/28" and "ipv4:192.0.2.16/28" are refinements of "ipv4:192.0.2.0/24", and hence inherit its value for "ISP" property. But because that value is inherited, it is not explicitly listed in the property map.

```
GET /propmap/full/inet-ia HTTP/1.1
Host: alto.example.com
Accept: application/alto-propmap+json,application/alto-error+json


HTTP/1.1 200 OK
Content-Length: ###
Content-Type: application/alto-propmap+json
{
  "property-map": {
    "ipv4:192.0.2.0/24":   {"ISP: "BitsRus"},
    "ipv4:192.0.2.0/28":   {"ASN": "12345"},
    "ipv4:192.0.2.16/28":  {"ASN": "12345"}
  }
}
```

7.5. Filtered Property Map Example #1

The following example uses the Filtered Property Map resource to request the "ISP", "ASN" and "state" properties for several IPv4 addresses. Note that the value of "state" for "ipv4:192.0.2.0" is the only explicitly defined property; the other values are all derived by the inheritance rules for internet address entities.

```
POST /propmap/lookup/inet-iacs HTTP/1.1
Host: alto.example.com
Accept: application/alto-propmap+json,application/alto-error+json
Content-Length: ###
Content-Type: application/alto-propmapparams+json

{
  "entities" : [ "ipv4:192.0.2.0",
                 "ipv4:192.0.2.1",
                 "ipv4:192.0.2.17" ],
  "properties" : [ "ISP", "ASN", "state" ]
}


HTTP/1.1 200 OK
Content-Length: ###
Content-Type: application/alto-propmap+json
{
  "property-map": {
    "ipv4:192.0.2.0":
           {"ISP": "BitsRus", "ASN": "12345", "state": "PA"},
    "ipv4:192.0.2.1":
           {"ISP": "BitsRus", "ASN": "12345", "state": "NJ"},
    "ipv4:192.0.2.17":
           {"ISP": "BitsRus", "ASN": "12345", "state": "CT"}
  }
}
```

7.6. Filtered Property Map Example #2

The following example uses the Filtered Property Map resource to request the "ASN", "country" and "state" properties for several IPv4 prefixes. Note that none of the returned property values were explicitly defined; all values are derived by the inheritance rules for internet address entities.

Also note the "ASN" property has the value "12345" for both the blocks "ipv4:192.0.2.0/28" and "ipv4:192.0.2.16/28", so every address in the block "ipv4:192.0.2.0/27" has that property value. However the block "ipv4:192.0.2.0/27" itself does not have a value for "ASN": address blocks cannot inherit properties from blocks with longer prefixes, even if every such block has the same value.

```
POST /propmap/lookup/inet-iacs HTTP/1.1
Host: alto.example.com
Accept: application/alto-propmap+json,application/alto-error+json
Content-Length: ###
Content-Type: application/alto-propmapparams+json

{
  "entities" : [ "ipv4:192.0.2.0/26",
                 "ipv4:192.0.2.0/27",
                 "ipv4:192.0.2.0/28" ],
  "properties" : [ "ASN", "country", "state" ]
}


HTTP/1.1 200 OK
Content-Length: ###
Content-Type: application/alto-propmap+json
{
  "property-map": {
    "ipv4:192.0.2.0/26":  {"country": "us"},
    "ipv4:192.0.2.0/27":  {"country": "us"},
    "ipv4:192.0.2.0/28":  {"ASN": "12345",
                           "country": "us",
                           "state": "NJ"}
 }
}
```

7.7. Filtered Property Map Example #3

The following example uses the Filtered Property Map resource to request the "pid" property for several IPv4 addresses and prefixes.

Note that the value of "pid" for the prefix "ipv4:10.0.0.0/15" is "pid1", even though all addresses in that block are in "pid2", because "ipv4:10.0.0.0/8" is the longest prefix in the network map which prefix-matches "ipv4:10.0.0.0/15", and that prefix is in "pid1".

```
POST /propmap/lookup/pid HTTP/1.1
Host: alto.example.com
Accept: application/alto-propmap+json,application/alto-error+json
Content-Length: ###
Content-Type: application/alto-propmapparams+json

{
  "entities" : [
                "ipv4:192.0.2.0  10.0.0.0",
                "ipv4:192.0.2.16 10.1.0.0",
                "ipv4:192.0.2.64  10.3.0.0",
                "ipv4:192.0.2.128  11.0.0.0",
                "ipv4:192.0.2.0/26 10.0.0.0/15",
                "ipv4:192.0.2.0/30 10.0.0.0/17" ],
  "properties" : [ "pid" ]
}


HTTP/1.1 200 OK
Content-Length: ###
Content-Type: application/alto-propmap+json
{
  "meta" : {
    "dependent-vtags" : [
       {"resource-id": "default-network-map",
        "tag": "7915dc0290c2705481c491a2b4ffbec482b3cf62"}
    ]
  },
  "property-map": {
    "ipv4:192.0.2.0":     {"pid": "pid2"},
    "ipv4:192.0.2.16":    {"pid": "pid2"},
    "ipv4:192.0.2.64":    {"pid": "pid1"},
    "ipv4:192.0.2.128":   {"pid": "defaultpid"},
    "ipv4:192.0.2.0/26":  {"pid": "pid1"},
    "ipv4:192.0.2.0/30":  {"pid": "pid2"}
  }
}
```

8. Security Considerations

As discussed in Section 15 of [RFC7285], properties may have sensitive customer-specific information. If this is the case, an ALTO Server may limit access to those properties by providing several different property maps. For non-sensitive properties, the ALTO Server would provide a URI which accepts requests from any client. Sensitive properties, on the other hand, would only be available via a secure URI which would require client authentication.

Also, while technically this document does not introduce any security risks not inherent in the Endpoint Property Service defined by [RFC7285], the GET-mode property map resource defined in this document does make it easier for a client to download large numbers of property values. Accordingly, an ALTO Server should limit GET-mode property maps for to properties which do not contain sensitive data.

9. IANA Considerations

This document defines additional application/alto-* media types, and extends the ALTO endpoint property registry.

9.1. application/alto-* Media Types

This document registers two additional ALTO media types, listed in Table 1.

Type	Subtype	Specification
application	alto-propmap+json	Section 4.1
application	alto-propmapparams+json	Section 5.3

Table 1: Additional ALTO Media Types

Type name: application

Subtype name: This documents registers multiple subtypes, as listed in Table 1.

Required parameters: n/a

Optional parameters: n/a

Encoding considerations: Encoding considerations are identical to those specified for the "application/json" media type. See [RFC7159].

Security considerations: Security considerations relating to the generation and consumption of ALTO Protocol messages are discussed in Section 15 of [RFC7285].

Interoperability considerations: This document specifies format of conforming messages and the interpretation thereof.

Published specification: This document is the specification for these media types; see Table 1 for the section documenting each media type.

Applications that use this media type: ALTO servers and ALTO clients either stand alone or are embedded within other applications.

Additional information:

Magic number(s): n/a

File extension(s): This document uses the mime type to refer to protocol messages and thus does not require a file extension.

Macintosh file type code(s): n/a

Person & email address to contact for further information: See Authors' Addresses section.

Intended usage: COMMON

Restrictions on usage: n/a

Author: See Authors' Addresses section.

Change controller: Internet Engineering Task Force (mailto:iesg@ietf.org).

9.2. ALTO Entity Domain Registry

This document requests IANA to create and maintain the "ALTO Entity Domain Registry", listed in Table 2.

Identifier	Entity Address Encoding	Hierarchy & Inheritance
ipv4	See Section 3.1.1	See Section 3.1.3
ipv6	See Section 3.1.2	See Section 3.1.3
pid	See Section 3.2	None

Table 2: ALTO Entity Domain Names

This registry serves two purposes. First, it ensures uniqueness of identifiers referring to ALTO entity domains. Second, it states the requirements for allocated domain names.

New ALTO entity domains are assigned after IETF Review [RFC5226] to ensure that proper documentation regarding the new ALTO entity domains and their security considerations has been provided. RFCs defining new entity domains should indicate how an entity in a registered domain is encoded as an EntityName, and, if applicable, the rules defining the entity hierarchy and property inheritance. Updates and deletions of ALTO entity domains follow the same procedure.

Registered ALTO entity domain identifiers MUST conform to the syntactical requirements specified in Section 2.4. Identifiers are to be recorded and displayed as strings.

Requests to add a new value to the registry MUST include the following information:

- Identifier: The name of the desired ALTO entity domain.
- Entity Address Encoding: The procedure for encoding the address of an entity of the registered type as an EntityAddr (see Section 2.3).
- Hierarchy: If the entities form a hierarchy, the procedure for determining that hierarchy.
- Inheritance: If entities can inherit property values from other entities, the procedure for determining that inheritance.
- Security Considerations: In some usage scenarios, entity addresses carried in ALTO Protocol messages may reveal information about an ALTO client or an ALTO service provider. Applications and ALTO service providers using addresses of the registered type should be made aware of how (or if) the addressing scheme relates to private information and network proximity.

This specification requests registration of the identifiers "ipv4", "ipv6" and "pid", as shown in Table 2.

9.3. ALTO Endpoint Property Type Registry

The ALTO Endpoint Property Type Registry was created by [RFC7285]. If possible, the name of that registry should be changed to "ALTO Entity Property Type Registry", to indicate that it is not restricted to Endpoint Properties. If it is not feasible to change the name, the description must be amended to indicate that it registers properties in all domains, rather than just the internet address domain.

10. References

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

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", RFC 3986, January 2005.

[RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", RFC 4632, BCP 122, August 2006.

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

[RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, August 2010.

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

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

[ID-draft-yang-alto-path-vector-02] Bernstein, G., Gao, K., Lee, Y., Roome, W., Scharf, M., and Y. Yang, "ALTO Topology Extension: Path Vector as a Cost Mode", July 2016.

[ID-draft-yang-alto-topology-06] Bernstein, G., Lee, Y., Roome, W., Scharf, M., and Y. Yang, "ALTO Topology Extensions: Node-Link Graphs", March 2015.

Authors' Addresses

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

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

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

Phone: +1-203-432-6400
Email: yry@cs.yale.edu

CDNI
Internet-Draft
Intended status: Informational
Expires: September 6, 2015

J. Seedorf
NEC
Y. Yang
Yale
J. Peterson
Neustar

March 5, 2015

CDNI Footprint and Capabilities Advertisement using ALTO
draft-seedorf-cdni-request-routing-alto-08

Abstract

Network Service Providers (NSPs) are currently considering to deploy Content Delivery Networks (CDNs) within their networks. As a consequence of this development, there is a need for interconnecting these local CDNs. The necessary interfaces for inter-connecting CDNs are currently being defined in the Content Delivery Networks Interconnection (CDNI) WG. This document focuses on the CDNI Footprint & Capabilities Advertisement interface (FCI). Specifically, this document specifies a new Application Layer Traffic Optimization (ALTO) service to facilitate Footprint & Capabilities Advertisement in a CDNI context.

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 September 6, 2015.

Copyright Notice

Table of Contents

1. Introduction 2
2. ALTO within CDNI Request Routing 3
3. Assumptions and High-Level Design Considerations 4
3.1. General Assumptions and Considerations 4
3.2. Semantics for Footprint/Capabilities Advertisment 5
3.3. Advantages of using ALTO as the CDNI FCI protocol 7
3.4. Selection of a Downstream CDN with ALTO 7
4. CDNI FCI ALTO Service 8
4.1. Server Response Encoding 8
4.1.1. CDNI FCI Map 8
4.1.2. Meta Information 8
4.1.3. Data Information 8
4.2. Protocol Errors 9
4.3. Example 9
5. Useful ALTO extensions for CDNI Request Routing 10
6. Security Considerations 12
7. Acknowledgements 12
8. References 12
8.1. Normative References 12
8.2. Informative References 13
Authors' Addresses 14

1. Introduction

Many Network Service Providers (NSPs) are currently considering or have already started to deploy Content Delivery Networks (CDNs) within their networks. As a consequence of this development, there is a need for interconnecting these local CDNs. Content Delivery Networks Interconnection (CDNI) has the goal of standardizing protocols to enable such interconnection of CDNs [RFC6707].

The CDNI problem statement [RFC6707] envisions four interfaces to be standardized within the IETF for CDN interconnection:

- CDNI Request Routing Interface

- CDNI Metadata Interface
- CDNI Logging Interface
- CDNI Control Interface

This document focuses solely on the CDNI Request Routing Interface, which can be further divided into two interfaces (see [RFC6707] for a detailed description): the CDNI Request Routing Redirection interface (RI), and the CDNI Footprint & Capabilities Advertisement interface (FCI). This document specifies a new Application Layer Traffic Optimization (ALTO) [RFC7285] service called 'CDNI Footprint & Capabilities Advertisement Service'. This service is used to transport CDNI FCI JSON objects, which are defined in a separate document [I-D.ma-cdni-capabilities]. An abstraction for managing individual CDNI capabilities in an opaque manner is defined as 'FCIBase' object in [I-D.ietf-cdni-footprint-capabilities-semantics].

Throughout this document, we use the terminology for CDNI defined in [RFC6707].

2. ALTO within CDNI Request Routing

The main purpose of the CDNI Request Routing Interface is described in [RFC6707] as follows: "The CDNI Request Routing interface enables a Request Routing function in an Upstream CDN to query a Request Routing function in a Downstream CDN to determine if the Downstream CDN is able (and willing) to accept the delegated Content Request. It also allows the Downstream CDN to control what should be returned to the User Agent in the redirection message by the upstream Request Routing function." On a high level, the scope of the CDNI Request Routing Interface therefore contains two main tasks:

- A) Determining if the downstream CDN is willing to accept a delegated content request
- B) Redirecting the content request coming from an upstream CDN to the proper entry point or entity in the downstream CDN

More precisely, in [RFC7336] the request routing interface is broadly divided into two functionalities:

- 1) the asynchronous advertisement of footprint and capabilities by a dCDN that allows a uCDN to decide whether to redirect particular user requests to that dCDN (the CDNI FCI)
- 2) the synchronous operation of actually redirecting a user request (the CDNI RI)

Application Layer Traffic Optimization (ALTO) [RFC7285] is an approach for guiding the resource provider selection process in distributed applications that can choose among several candidate resources providers to retrieve a given resource. By conveying network layer (topology) information, an ALTO server can provide important information to "guide" the resource provider selection process in distributed applications. Usually, it is assumed that an ALTO server conveys information these applications cannot measure themselves [RFC5693].

Originally, ALTO was motivated by the huge amount of cross-ISP traffic generated by P2P applications [RFC5693]. Recently, however, ALTO is also being considered for improving the request routing in CDNs [I-D.jenkins-alto-cdn-use-cases]. In this context, it has also been proposed to use ALTO for selecting an entry-point in a downstream NSP's network (see section 3.4 "CDN delivering Over-The-Top of a NSP's network" in [I-D.jenkins-alto-cdn-use-cases]). Also, the CDNI problem statement explicitly mentions ALTO as a candidate protocol for "algorithms for selection of CDN or Surrogate by Request-Routing systems" [RFC6707].

3. Assumptions and High-Level Design Considerations

In this section we list some assumptions and design issues to be considered when using ALTO for the CDNI Footprint and Capabilities Advertisement interface.

3.1. General Assumptions and Considerations

Below we list some general assumptions and considerations:

- As explicitly being out-of-scope for CDNI [RFC6707], it is assumed that ingestion of content or acquiring content across CDNs is not part of request routing as considered within CDNI standardization work. The focus of using ALTO (as considered in this document) is hence on request routing only, assuming that the content (desired by the end user) is available in the downstream CDN (or can be aquired by the downstream CDN by some means).

- Federation Model: "Footprint and Capabilities Advertisement" and in general CDN request routing depends on the federation model among the CDN providers. Designing a suitable solution thus depends on whether a solution is needed for different settings, where CDNs consist of both NSP CDNs (serving individual ASes) and general, traditional CDNs (such as Akamai). We assume that CDNI is not designed for a setting where only NSP CDNs each serve a single AS only.

- In this document, it is assumed that the upstream CDN (uCDN) makes the decision on selecting a downstream CDN, based on information that each downstream CDN has made available to the upstream CDN. Further, we assume that in principle more than one dCDN may be suitable for a given end-user request (i.e. different dCDNs may claim "overlapping" footprints). The uCDN hence potentially has to select among several candidate downstream CDNs for a given end user request.

- It is not clear what kind(s) of business, contract, and operational relationships two peering CDNs may form. For the Internet, we see provider-customer and peering as two main relations; providers may use different charging models (e.g., 95-percentile, total volume) and may provide different SLAs. Given such unknown characteristics of CDN peering business agreements, we should design the protocol to support as much diverse potential business and operational models as possible.

3.2. Semantics for Footprint/Capabilities Advertisment

The CDNI document on "Footprint and Capabilities Semantics" [I-D.ietf-cdni-footprint-capabilities-semantics] defines the semantics for the CDNI FCI. It thus provides guidance on what Footprint and Capabilities mean in a CDNI context and how a protocol solution should in principle look like. Here we briefly summarize the key points of the semantics of Footprint and Capabilities (for a detailed discussion, the reader is referred to [I-D.ietf-cdni-footprint-capabilities-semantics]):

- Often, footprint and capabilities are tied together and cannot be interpreted independently from each other. In such cases, i.e. where capabilities must be expressed on a per footprint basis, it may be beneficial to combine footprint and capabilities advertisement.

- Given that a large part of Footprint and Capabilities Advertisement will actually happen in contractual agreements, the semantics of CDNI Footprint and Capabilities advertisement refer to answering the following question: what exactly still needs to be advertised by the CDNI FCI? For instance, updates about temporal failures of part of a footprint can be useful information to convey via the CDNI request routing interface. Such information would provide updates on information previously agreed in contracts between the participating CDNs. In other words, the CDNI FCI is a means for a dCDN to provide changes/updates regarding a footprint and/or capabilities it has prior agreed to serve in a contract with a uCDN.

- It seems clear that "coverage/reachability" types of footprint must be supported within CDNI. The following such types of footprint are mandatory and must be supported by the CDNI FCI:

  * List of ISO Country Codes

  * List of AS numbers

  * Set of IP-prefixes

  A 'set of IP-prefixes' must be able to contain full IP addresses, i.e., a /32 for IPv4 and a /128 for IPv6, and also IP prefixes with an arbitrary prefix length. There must also be support for multiple IP address versions, i.e., IPv4 and IPv6, in such a footprint.

- For all of these mandatory-to-implement footprint types, footprints can be viewed as constraints for delegating requests to a dCDN: A dCDN footprint advertisement tells the uCDN the limitations for delegating a request to the dCDN. For IP prefixes or ASN(s), the footprint signals to the uCDN that it should consider the dCDN a candidate only if the IP address of the request routing source falls within the prefix set (or ASN, respectively). The CDNI specifications do not define how a given uCDN determines what address ranges are in a particular ASN. Similarly, for country codes a uCDN should only consider the dCDN a candidate if it covers the country of the request routing source. The CDNI specifications do not define how a given uCDN determines the country of the request routing source. Multiple footprint constraints are additive, i.e. the advertisement of different types of footprint narrows the dCDN candidacy cumulatively.

- The following capabilities seem useful as 'base' capabilities, i.e. ones that are needed in any case and therefore constitute mandatory capabilities to be supported by the CDNI FCI:

  * Delivery Protocol (e.g., HTTP vs. RTMP)

  * Acquisition Protocol (for aquiring content from a uCDN)

  * Redirection Mode (e.g., DNS Redirection vs. HTTP Redirection as discussed in [RFC7336])

  * Capabilities related to CDNI Logging (e.g., supported logging mechanisms)

* Capabilities related to CDNI Metadata (e.g., authorization algorithms or support for proprietary vendor metadata)

3.3. Advantages of using ALTO as the CDNI FCI protocol

The following reasons make ALTO a suitable candidate protocol for downstream CDN selection as part of CDNI request routing and in particular for an FCI protocol:

- CDN request routing is done at the application layer. ALTO is a protocol specifically designed to improve application layer traffic (and application layer connections among hosts on the Internet) by providing additonal information to applications that these applications could not easily retrieve themselves. For CDNI, this is exactly the case: a uCDN wants to improve application layer CDN request routing by using dedicated information (provided by a dCDN) that the uCDN could not easily obtain otherwise.

- The semantics of an ALTO network map are an exact match for the needed information to convey a footprint by a downstream CDN, in particular if such a footprint is being expressed by IP-prefix ranges.

- Security: ALTO maps can be signed and hence provide inherent integrity protection (see Section 6)

- RESTful-Design: The ALTO protocol has undergone extensive revisions in order to provide a RESTful design regarding the client-server interaction specified by the protocol. A CDNI FCI interface based on ALTO would inherit this RESTful design.

- Error-handling: The ALTO protocol has undergone extensive revisions in order to provide sophisticated error-handling, inparticular regarding unexpected cases. A CDNI FCI interface based on ALTO would inherit this thought-through and mature error-handling.

- Filtered network map: The ALTO Map Filtering Service (see [RFC7285] for details) would allow a uCDN to query only for parts of an ALTO map.

3.4. Selection of a Downstream CDN with ALTO

Under the considerations stated in Section 3, ALTO can help the upstream CDN provider to select a proper downstream CDN provider for a given end user request as follows: Each downstream CDN provider

hosts an ALTO server which provides ALTO services which convey CDNI FCI information to an ALTO client at the upstream CDN provider.

4. CDNI FCI ALTO Service

The ALTO protocol is based on an ALTO Information Service Framework which consists of several services, where all ALTO services are 'provided through a common transport protocol, messaging structure and encoding, and transaction model' [RFC7285]. The ALTO protocol specification [RFC7285] defines several such services, e.g. the ALTO map service.

This document defines a new ALTO Service called 'CDNI Footprint & Capabilities Advertisement Service' which conveys JSON objects of media type 'application/alto-fcimap+json'. This media type and JSON object format is defined in [I-D.ma-cdni-capabilities]; this document specifies how to transport such JSON objects via the ALTO protocol with the ALTO 'CDNI Footprint & Capabilities Advertisement Service'. An abstraction for managing individual CDNI capabilities in an opaque manner is defined as 'FCIBase' object in [I-D.ietf-cdni-footprint-capabilities-semantics].

4.1. Server Response Encoding

4.1.1. CDNI FCI Map

The media type of the CDNI FCI Map is 'application/alto-cdni-fcimap+json'. The HTTP Method, Accept Input Parameters, Capabilities, Uses, and Response of the CDNI FCI Map are specified in [I-D.ma-cdni-capabilities].

4.1.2. Meta Information

The 'meta' field of a FCIMapData response MUST include 'vtag', which is an ALTO Version Tag of the retrieved FCIMapData according to [RFC7285] (Section 10.3.). It thus contains a 'resource-id' attribute, and a 'tag' is an identifier string.

4.1.3. Data Information

The data component of a CDNI FCI Map resource is named 'fcimap' which is a JSON object of type FCIMapData. This JSON object of type FCIMapData is derived from ResponseEntityBase as specified in the ALTO protocol [RFC7285] (Section 8.4.) and specified in [I-D.ma-cdni-capabilities].

4.2. Protocol Errors

Protocol errors are handled as specified in the ALTO protocol [RFC7285] (Section 8.5.).

4.3. Example

The following example shows an CDNI FCI Map as in [I-D.ma-cdni-capabilities], however with meta-information as defined in Section 4.1.2 of this document.

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

HTTP/1.1 200 OK
Content-Length: 439
Content-Type: application/alto-fcimap+json
{
  "meta" : {
    "vtag": {
      "resource-id": "my-default-fcimap",
       "tag": "da65eca2eb7a10ce8b059740b0b2e3f8eb1d4785"
    }
  },
  "fcimap": [
    { "name": "delivery_protocol",
      "values": [
        "HTTP",
        "RTSP",
        "MMS"
      ]
    },
    { "name": "delivery_protocol",
      "values": [
        "RTMP",
        "HTTPS"
      ],
      "footprint": [
        { "type": "IPv4CIDR",
          "values": [
            "10.1.0.0/16",
            "10.10.10.0/24"
          ]
        }
      ]
    }
  ]
}
```

5. Useful ALTO extensions for CDNI Request Routing

It is envisioned that yet-to-be-defined ALTO extensions will be standardized that make the ALTO protocol more suitable and useful for applications other than the originally considered P2P use case [I-D.marocco-alto-next]. Some of these extensions to the ALTO protocol would be useful for ALTO to be used as a protocol within CDNI request routing, and in particular within the "Footprint and

Capabilities Advertisment" part of the CDNI request routing interface.

The following proposed extensions to ALTO would be beneficial to facilitate CDNI request routing with ALTO as outlined in Section 3.4:

- Server-initiated Notifications and Incremental Updates: In case the footprint or the capabilities of a downstream CDN change abruptly (i.e. unexpectedly from the perspective of an upstream CDN), server initiated notifications would enable a dCDN to directly inform an upstream CDN about such changes. Consider the case where - due to failure - part of the footprint of the dCDN is not functioning, i.e. the CDN cannot serve content to such clients with reasonable QoS. Without server-initiated notifications, the uCDN might still use a very recent network and cost map from dCDN, and therefore redirect request to dCDN which it cannot serve. Similarly, the possibility for incremental updates would enable efficient conveyance of the aforementioned (or similar) status changes by the dCDN to the uCDN. A proposal for server-initiated ALTO updates can be found in [I-D.marocco-alto-ws]. A discussion of incremental ALTO updates can be found in [I-D.schwan-alto-incr-updates].

- Content Availability on Hosts: A dCDN might want to express CDN capabilties in terms of certain content types (e.g. codecs/ formats, or content from certain content providers). A new endpoint property for ALTO that would be able to express such "content availability" would enable a dCDN to make available such information to an upstream CDN. This would enable a uCDN to determine if a given dCDN actually has the capabilities for a given request with respect to the type of content requested.

- Resource Availability on Hosts or Links: The capabilities on links (e.g. maximum bandwidth) or caches (e.g. average load) might be useful information for an upstream CDN for optimized dowmstream CDN selection. For instance, if a uCDN receives a streaming request for content with a certain bitrate, it needs to know if it is likely that a dCDN can fulfill such stringent application-level requirements (i.e. can be expected to have enough consistent bandwidth) before it redirects the request. In general, if ALTO could convey such information via new endpoint properties, it would enable more sophisticated means for downstream CDN selection with ALTO.

6. Security Considerations

One important security consideration is the proper authentication of advertisement information provided by a downstream CDN. The ALTO protocol provides a specification for a signature of ALTO information (see 8.2.2. of [RFC7285]. ALTO thus provides a proper means for protecting the integrity of FCI information.

More Security Considerations will be discussed in a future version of this document.

7. Acknowledgements

The authors would like to thank Kevin Ma, Daryl Malas, and Matt Caulfield for their timely reviews and invaluable comments.

Jan Seedorf is partially supported by the GreenICN project (GreenICN: Architecture and Applications of Green Information Centric Networking), a research project supported jointly by the European Commission under its 7th Framework Program (contract no. 608518) and the National Institute of Information and Communications Technology (NICT) in Japan (contract no. 167). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the GreenICN project, the European Commission, or NICT.

8. References

8.1. Normative References

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

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

[RFC6770] Bertrand, G., Stephan, E., Burbridge, T., Eardley, P., Ma, K., and G. Watson, "Use Cases for Content Delivery Network Interconnection", RFC 6770, November 2012.

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

[RFC7336] Peterson, L., Davie, B., and R. van Brandenburg, "Framework for Content Distribution Network Interconnection (CDNI)", RFC 7336, August 2014.

[RFC7337] Leung, K. and Y. Lee, "Content Distribution Network Interconnection (CDNI) Requirements", RFC 7337, August 2014.

8.2. Informative References

[I-D.peterson-CDNI-strawman] Peterson, L. and J. Hartman, "Content Distribution Network Interconnection (CDNI) Problem Statement", draft-peterson-CDNI-strawman-01 (work in progress), May 2011.

[I-D.marocco-alto-next] Marocco, E. and V. Gurbani, "Extending the Application-Layer Traffic Optimization (ALTO) Protocol", draft-marocco-alto-next-00 (work in progress), January 2012.

[I-D.marocco-alto-ws] Marocco, E. and J. Seedorf, "WebSocket-based server-to-client notifications for the Application-Layer Traffic Optimization (ALTO) Protocol", draft-marocco-alto-ws-02 (work in progress), February 2014.

[I-D.schwan-alto-incr-updates] Schwan, N. and B. Roome, "ALTO Incremental Updates", draft-schwan-alto-incr-updates-02 (work in progress), July 2012.

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

[I-D.ma-cdni-capabilities] Ma, K. and J. Seedorf, "CDNI Footprint & Capabilities Advertisement Interface", draft-ma-cdni-capabilities-06 (work in progress), June 2014.

[I-D.liu-cdni-cost] Liu, H., "A Cost Perspective on Using Multiple CDNs", draft-liu-cdni-cost-00 (work in progress), October 2011.

[I-D.ietf-cdni-metadata]
Niven-Jenkins, B., Murray, R., Caulfield, M., and K. Ma, "CDN Interconnection Metadata", draft-ietf-cdni-metadata-09 (work in progress), March 2015.

[I-D.ietf-cdni-logging]
Faucheur, F., Bertrand, G., Oprescu, I., and R. Peterkofsky, "CDNI Logging Interface", draft-ietf-cdni-logging-15 (work in progress), February 2015.

[I-D.ietf-cdni-footprint-capabilities-semantics]
Seedorf, J., Peterson, J., Previdi, S., Brandenburg, R., and K. Ma, "CDNI Request Routing: Footprint and Capabilities Semantics", draft-ietf-cdni-footprint-capabilities-semantics-05 (work in progress), March 2015.

Authors' Addresses

Jan Seedorf
NEC Laboratories Europe, NEC Europe Ltd.
Kurfuersten-Anlage 36
Heidelberg 69115
Germany

Phone: +49 (0) 6221 4342 221
Email: jan.seedorf@neclab.eu
URI: http://www.neclab.eu

Y.R. Yang
Yale University
51 Prospect Street
New Haven 06511
USA

Email: yry@cs.yale.edu
URI: http://www.cs.yale.edu/~yry/

Jon Peterson
NeuStar
1800 Sutter St Suite 570
Concord CA 94520
USA

Email: jon.peterson@neustar.biz

ALTO WG
Internet-Draft
Intended status: Informational
Expires: September 14, 2017

Q. Xiang
Tongji/Yale University
H. Newman
California Institute of Technology
G. Bernstein
Grotto Networking
A. Mughal
J. Balcas
California Institute of Technology
J. Zhang
Tongji University
H. Du
Y. Yang
Tongji/Yale University

March 13, 2017

Traffic Optimization for ExaScale Science Applications
draft-xiang-alto-exascale-network-optimization-01.txt

Abstract

Massive datasets continue to be acquired, simulated, processed and analyzed by globally distributed scientific collaborations, and the volume of this data is growing exponentially. These datasets need to be exchanged through a global network infrastructure. Applications that manage and analyze such massive data volumes can benefit substantially from the information about networking, computing and storage resources from each member sites, and more directly from network-resident services that optimize and load balance resource usage among multiple data transfer and analytic requests, and achieve a better utilization of multi-resources in clusters.

The Application-Layer Traffic Optimization (ALTO) protocol can provide via extensions the network information about different clusters/sites, to both users and proactive network management services where applicable, with the goal of improving both application performance and network resource utilization. However, it has been verified in both science networks and commercial data center networks that network resource in many cases is not the bottleneck preventing the efficiency of large dataset transfer and data-intensive analytics. To achieve a greater overall efficiency of the science programs' workflows information about different resources, such as computing, storage and networking, should be provided to data intensive applications simultaneously.

In this document, we propose that it is feasible to use existing ALTO services to provides not only network information, but also

information about other resources in science networks including computing and storage. We introduce an Exascale Science Application Orchestrator (ExaO), which achieves an efficient multi-resource allocation to support low-latency dataset transfer and data intensive analytics in exascale science networks. ExaO provides simple APIs for users to submit and manage dataset transfer and analytic requests and to monitor the status of each request, along with fine-grained local and global network and site state information in real-time. It collects cluster information from multiple ALTO services utilizing topology extensions and leverages emerging SDN control capabilities to orchestrate the resource allocation for dataset transfers and analytic tasks, leading to improved transfer and analytic latency as well as more efficient utilization of multi-resources in clusters/ sites.

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 September 14, 2017.

Copyright Notice

Table of Contents

1. Introduction 3
2. Requirements Language 5
3. Changes Since Version -00 5
4. Problem Settings 5
4.1. Motivation 6
4.2. Challenges 6
5. Basic Idea 7
5.1. Using ALTO topology services to provide multi-resource information 7
5.2. Example: encode storage bandwidth into path vector . . . 7
6. Key Issues 9
7. Exascale Dataset Transfer Orchestrator Framework 10
7.1. Architecture 10
7.2. Request parser 12
7.2.1. User API 12
7.3. ALTO Client 13
7.3.1. Query Mode 13
7.4. ALTO Server 13
7.5. Dataset Transfer Agents 14
7.6. Request Execution Agents 14
7.7. Multi-Resource Orchestrator 14
7.7.1. Orchestration Algorithms 15
7.7.2. Online, Dynamic Orchestration 15
7.7.3. Example: A Max-Min Fairness Resource Allocation Algorithm 15
8. Discussion 16
8.1. Deployment 17
8.2. Benefiting From ALTO Extension Topology Services 17
8.3. Constraints of the MFRA Algorithm 18
9. Security Considerations 18
10. IANA Considerations 18
11. Acknowledgments 19
12. References 19
12.1. Normative References 19
12.2. Informative References 19
Authors' Addresses 20

1. Introduction

Scientific innovation continues to exponentially increase the production of valuable research data. Exchange of this information typically involves the worldwide network infrastructure. One leading example is the Large Hadron Collider (LHC) high energy physics (HEP) program, which aims to find new particles and interactions in a previously inaccessible range of energies. The scientific collaborations that have built and operate large HEP experimental

facilities at the LHC, such as the Compact Muon Solenoid (CMS) and A Toroidal LHC ApparatuS (ATLAS), currently have more than 300 petabytes of data under management at hundreds of sites around the world, and this volume is expected to grow to one exabyte by approximately 2018.

With such an increasing data volume, how to manage the storage and analytics of these data in a globally distributed infrastructure has become an increasingly challenging issue. Applications such as the Production ANd Distributed Analysis system (PanDA) in ATLAS and the Physics Experiment Data Export system (PhEDEX) in CMS have been developed to manage the data transfers among different cluster sites. Given a data transfer request, these applications make data transfer decisions based on the availability of dataset replicas at different sites and initiate retransmission from a different replica if the original transmission fails or is excessively delayed. And HTCondor is deployed to achieve coarse-grained data analytics parallelization across these sites. When a data analytic task is submitted, HTCondor adopts a match-making process to assign the task to a certain set of servers in one site, based on the coarse-grained description of resource availability, such as the number of cores, the size of memory, the size of hard disk, etc. However, neither dataset transfers nor data analytic task parallelization takes fine-grained information of cluster resources, such as data locality, memory speed, network delay, network bandwidth, etc., into account, leading to high data transfer and analytic latency and underutilization of cluster resources.

The Application-Layer Traffic Optimization (ALTO) services defined in [RFC7285] provide network information with the goal of improving the network resource utilization while maintaining or improving application performance. Though ALTO is not designed to provide information about other resources, such as computing and storage resources, in cluster networks, in this document we propose that exascale science networks can leverage existing ALTO services defined in [RFC7285] and ALTO topology extension services defined in network graph [DRAFT-NETGRAPH], path vector [DRAFT-PV], routing state abstraction[DRAFT-RSA], multi-cost [DRAFT-MC] and cost-calendar [DRAFT-CC] and etc. to encode information about multiple types of resources in science networks, such as memory I/O speed, CPU utilization, network bandwidth, and provides such information to orchestration applications to improve the performance of dataset transfer and data analytic tasks, including throughput, latency, etc.

This document introduces a centralized resource orchestration service, Exascale Science Application Orchestrator (ExaO), which provides an efficient multi-resource allocation to support low-latency dataset transfer and data-intensive analytics in exascale

science networks. ExaO provides a set of simple API for authorized users to submit, update and delete dataset transfer requests and data intensive analytic requests. One important proposal we make in this document is that it is feasible to use ALTO services to provide not only network information, but also information on other resources in science networks including computing and storage.

An ExaO prototype with the dataset transfer scheduling component has been implemented on a single-domain Caltech SDN development testbed, where the ALTO OpenDaylight controller is used to collect topology information. We are currently designing the resource orchestration components to achieve low-latency data-intensive analytics.

This document is organized as follows: Section 3 summarizes the change of this document since version -00. Section 4 elaborates on the motivation and challenges for coordinating storage, computing and network resources in a globally distributed science network infrastructure. Section 5 discusses the basic idea of encoding multi-resource information into ALTO path vector and abstraction services and gives an example. Section 6 lists several key issues to address in order to realize the proposal of providng multi-resource information by ALTO topology services. Section 7 gives the details of ExaO architecture for orchestrating exascale dataset transfer. Section 8 discusses current development progress of ExaO and next steps.

2. Requirements Language

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

3. Changes Since Version -00

- Add the basic idea of ExaO, i.e., use ALTO topology service to provide a multi-resource abstraction of clusters/sites in science networks.
- Add an example to show the feasibility of encoding storage resources in existing ALTO services, e.g., ALTO path-vector.
- Add the data analytic component in the orchestration framework.

4. Problem Settings

4.1. Motivation

Exascale science programs usually involve the participation of countries and sites all over the world. The CMS experiment in the LHC physics program is a typical example. The site located at the LHC laboratory is called a Tier-0 site, which processes the data selected and stored locally by the online systems that select and record the data in real-time as it comes off the particle detector, archives it and transfers it to over 10 Tier-1 sites around the globe. Raw datasets and processed datasets from Tier-1 sites are then transferred to over 160 Tier-2 sites around the world based on users' requests. Different sites have different resources and belong to different administration domains. With the exponentially increasing data volume in the CMS experiment, the management of large data transfers and data intensive analytics in such a global multi-domain science network has become an increasingly challenging issue. Allocating resources in different clusters to fulfill different users' dataset transfer requests and data analytic requests require careful orchestrating as different requests are competing for limited storage, computation and network resources.

4.2. Challenges

Orchestrating exascale dataset transfers and analytics in a globally distributed science network is non-trivial as it needs to cope with two challenges.

- Different sites in this network belong to different administration domain. Sharing raw site/cluster information would violate sites' privacy constraints. Orchestrating data transfers and analytic requests based on highly abstracted, non-real-time network information may lead to suboptimal scheduling decisions. Hence the orchestrating framework must be able to collect sufficient resource information about different clusters/sites in real-time as well as over the longer term, to allow reasonably optimized network resource utilization without violating sites' privacy requirements.

- Different science programs tend to adopt different software infrastructures for managing dataset transfers and analytics, and may place different requirements. Hence the orchestrating framework must be modular so that it can support different dataset management systems and different orchestrating algorithms.

The orchestrating framework must support the interaction between the multi-resource orchestration module, the dataset transfer module, and the data analytic execution module. The key information to be exchanged between modules includes dataset information, the resource

state of different clusters and sites, the transfer and analytic requests in progress, as well as trends and network-segment and site performance from the network point of view. Such interaction ensures that (1) the various programs can adapt their own data transfer and analytic systems to be multi-resource-aware, and more efficient in achieving their goals; and (2) the various orchestrating algorithms can achieve a reasonably optimized utilization on not only the network resource but also the computing and storage resources.

5. Basic Idea

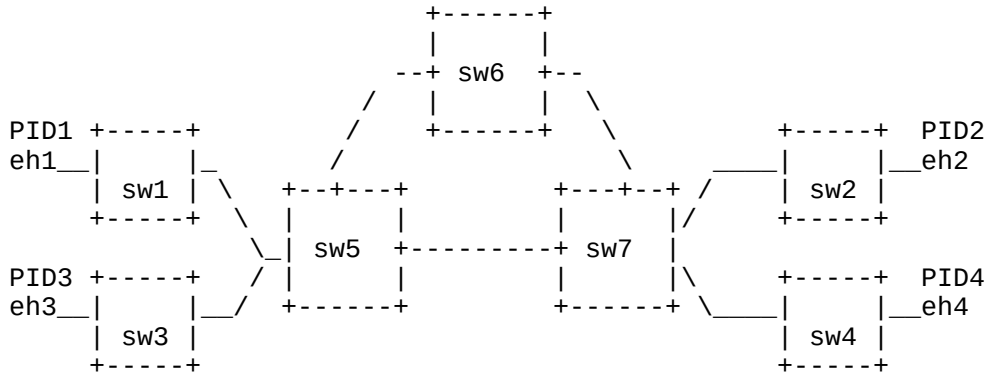
5.1. Using ALTO topology services to provide multi-resource information

The ALTO protocol is designed to provide network information to applications so that applications can achieve a better performance. Different ALTO topology services including path vector, routing state abstraction, multi-cost, cost calendar, etc. have been proposed to provide fine-grained network information to applications. In this document, we propose that not only can ALTO provide network information of different cluster sites, it can also provides information of multiple resources, including computing resource and storage resources. To this end, the basic "one-big-switch" abstraction provided by the base ALTO protocol is not sufficient. Several examples have already been given in [DRAFT-PV] and [DRAFT-RSA] to demonstrate that. There has been a similar proposal before about using ALTO to provide resource information of data centers [DRAFT-DC]. However, that proposal requires a new information model for clusters or data centers, which may affect the compatibility of ALTO. The solution of this proposal is simpler. Its basic idea is that each computer node and storage node can be seen as a "network element" or an "abstract network element" defined in ALTO-path-vector [DRAFT-PV]. In this way, ExaO can fully reuse all existing ALTO services by introducing only one cost-mode (pv) and two cost-metrics (ne and ane), instead of introducing a new information model.

5.2. Example: encode storage bandwidth into path vector

We use the same dumbbell topology in [DRAFT-RSA] as an example to show the feasibility of using ALTO topology service to provide multi-resource information. In this topology, we assume the bandwidth of each network cable is 1Gbps, including the cables connecting end hosts to switches. Consider a dataset transfer request which needs to schedule the traffic among a set of end host source-destination pairs, say eh1 -> eh2, and eh3 -> eh4. Assume that the transfer application receives from the ALTO Cost Map service that both eh1 -> eh2 and eh3 -> eh4 have bandwidth 1Gbps. In [DRAFT-RSA], it is shown that whether each of the two traffic flows can receive 1Gbps

bandwidth depends on whether the routes of two flows share a bottleneck link. Path vector and routing state abstraction services provide additional information about network state encoded in abstract network elements. If the returned state is ane1 + ane2 <= 1Gbps, it means two flows cannot each get 1Gbps bandwidth at the same time. If the returned state is ane1 <= 1Gbps and ane2 <= 1Gbps, it means two flows each can get 1Gbps bandwidth.



Other than network resource, assume in this topology eh1 and eh3 are equipped with commodity hard drive disk (HDD) while eh2 and eh4 are equipped with SSD. Because the bandwidth of HDD is typically 0.8Gbps and that of SSD is typically 3Gbps. Even if the returned routing state is ane1 <= 1Gbps and ane2 <=1Gbps, the actual bottleneck of each traffic flow is the storage I/O bandwidth at source host. As a result, the total bandwidth of both traffic flows can only reach 1.6Gbps.

It has been verified in the CMS experiment, and also several studies on commercial data centers that network resource is not always the bottleneck of large dataset transfer and data analytics. Many have reported that storage resources and computing resources become the bottleneck in a fair large percent of dataset transfers and data analytic tasks in science networks and commercial data centers.

In this example, if we see the end hosts as network elements, the storage I/O bandwidth of each host can also be encoded as an abstract element into the path-vector. And under the storage and route settings above, the returned cluster state would be ane1 <=0.8Gbps and ane2 <=0.8Gbps, which provides a more accurate capacity region for the requested traffic flows.

6. Key Issues

Last section describes the basic idea of using ALTO topology services to provide multi-resource information and gives an example to demonstrate its feasibility. Next we list and discuss several key issues to address in this proposal.

- Can ALTO topology services provide data locality information? Existing ALTO topology services do not provide such information. Many studies have pointed out that such information plays a vital role in reducing the latency of data-intensive analytics. If ALTO topology services can encode such information together with information of other resources together, data-intensive applications can benefit a great deal in terms of information aggregation and communication overhead.

- How to quickly map applications' resource allocation decision on abstract multi-resource view back to the physical multi-resource view of clusters/sites? Fine-grained resource information can be encoded into abstract network elements to reduce overhead and provide certain privacy protection of clusters. Such information can be highly compressed (see the dumbbell example used in this document as well as in [DRAFT-PV] and [DRAFT-RSA]). In preliminary evaluations on RSA, the network element compression ratio can be as high as 80 percent. This ratio is expected to be even higher in large-scale data center or cluster setting, e.g. a fat-tree topology with k=48. Therefore a fast mapping from the resource orchestration decisions on the abstract view back to the physical view is needed to satisfy the stringent latency requirement of large dataset transfers and data-intensive analytics.

- How much privacy, including key resource configurations, raw topology, intra-cluster scheduling policy, etc., will be exposed? Compared with the "one-big-switch" abstraction, other ALTO topology services such as path-vector [DRAFT-PV] and routing state abstraction [DRAFT-RSA] provides fine-grained resource information to applications. Even if such information can be encoded into abstract network elements, it still risks exposing private information of different clusters/sites. Current internet drafts of these services did not provide any formal privacy analysis or performance measurement. This would be one key issue this document plan to investigate in the future.

- How does current ALTO services such as path-vector and RSA scale when they are used to provide abstract information of multiple resources in clusters? Another issue along this line is how to balance the liveness of fine-grained resource information and the

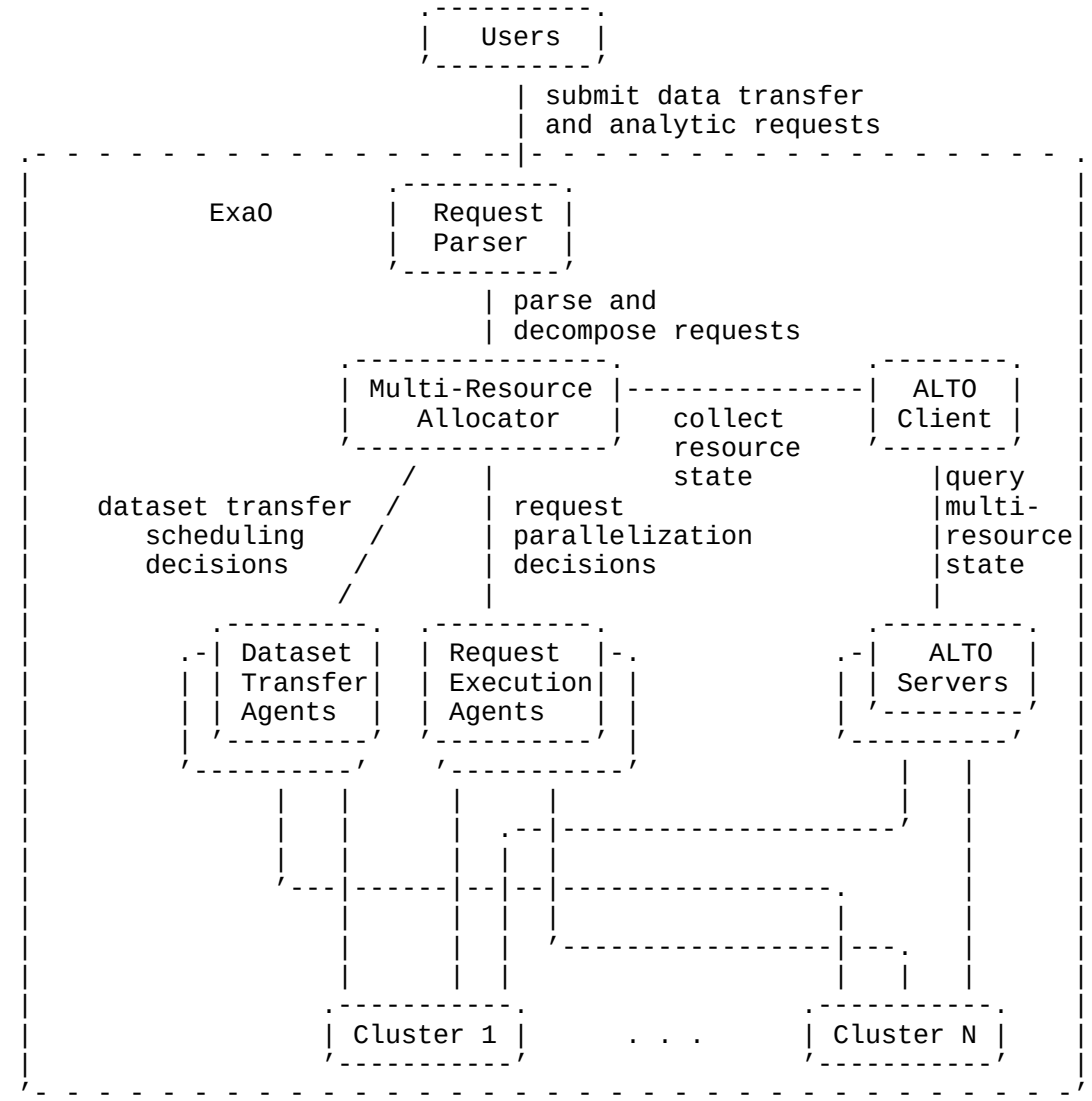
corresponding information delivery overhead? Although encoding information of network elements into abstract network elements can achieve a very competitive information compression ratio, a large dataset transfer or analytic application always involve many network elements in multiple clusters/sites and the absolute number of involved network elements keep increasing as the scale of clusters increase. In addition, when resource information in a cluster changes, the ALTO services need to inform all related applications. In either cases, delivering fine-grained resource information would cause high communication overhead. There still lacks of an analytics or experimental understanding on the scalability of path-vector and RSA services.

7. Exascale Dataset Transfer Orchestrator Framework

7.1. Architecture

This section describes the design details of key components of the ExaO framework: the request parser, the ALTO client, the ALTO servers, the multi-resource orchestrator, the dataset transfer agents and the request execution agents. Among these modules, the request parser provides a set of simple APIs for authorized users to submit, update and cancel dataset transfer requests and data-intensive analytic requests. Depending on the programming model of each request, e.g., Map-Reduce, the parser decompose it into multiple smaller sub-requests. The ALTO client collects information about multiple resources in different clusters from ALTO servers deployed at different cluster sites.

Both the decomposed sub-requests and the collected information of different clusters are sent to the multi-resource orchestrator, which makes dataset transfer scheduling decisions, including replica selection, routing path computation and bandwidth allocation, and request parallelization decisions, such as which cluster each sub-request should be placed at the multi-resource orchestrator. These decisions are then sent to data transfer agents and request execution agents, who act these decisions on behalf of ExaO. Figure 1 shows the whole process.



The benefits of ExaO include:

- It provides a set of simple APIs for authorized users to submit and manage dataset transfer requests i and data-intensive analytic requests, and enables real-time requests' status monitoring.

- It improves a better resource utilization and achieves low-latency for dataset transfer and data intensive analytics in science

networks, by collecting the resource information of different clusters/sites and orchestrating the resource allocation for all submitted requests in a centralized framework.

- The architecture of ExaO is modular to support different resource allocation algorithms, data transfer agents and request execution agents. It also supports the deployment of different ALTO servers.

7.2. Request parser

The request parser is the front end of ExaO, and is responsible for collecting dataset transfer requests and data-intensive analytic requests from users and passing them to the multi-resource orchestrator for further processing. It provides a set of simple APIs for users to submit and manage requests, and to track the status of requests in real-time.

7.2.1. User API

- submitReq(request, [options])

  This API allows users to submit a request and specify corresponding options. The request can be a data transfer request or a data analytic request. Request options include priority, delay, etc. It returns a request identifier reqID that allows users to update, delete this request or track its status. The additional options may or may not be approved, and the relative priorities may be modified by the resource orchestrator depending on the role of users (regular users or administrators at different levels), the resource availability and the status of other ongoing requests.

- updateReq(requestID, [options])

  This API allows users to update the options of requests. It will return a SUCCESS if the new options are received by the request parser. But these new options may or may not be approved, and may be modified by the resource orchestrator depending on the role of users (regular users or administrators), the resource availability and the status of other ongoing requests.

- deleteReq(requestID)

  This API allows users to delete a request by passing the corresponding requestID. A completed request cannot be deleted. An ongoing request will be stopped and the output data will be deleted.

- o getReqStatus(requestID)

  This API allows users to query the status of a request by specifying the corresponding requestID. The returned status information includes whether the request has started, the assigned priority, the percentage of finished sub-requests, transmission statistics, the expected remaining time to finish, etc.

7.3. ALTO Client

The ALTO client is in the back end of ExaO and is responsible for retrieving cluster resource information through querying ALTO servers deployed at different sites. The resource information needed in ExaO includes the topology, link bandwidth, computing node memory I/O speed, computing node CPU utilization, etc. The base ALTO protocol [RFC7285] provides an extreme single-node abstraction for this information, which only supports the multi-resource orchestrator making coarse-grained resource allocation decisions. To enable fine-grained multi-resource orchestration for dataset transfer and data analytics in cluster networks, ALTO topology extension services such as routing state abstraction (RSA) [DRAFT-RSA], path vector [DRAFT-PV], network graph [DRAFT-NETGRAPH], multi-cost [DRAFT-MC] and cost-calendar [DRAFT-CC] are needed to provide fine-grained information about different types of resources in clusters.

7.3.1. Query Mode

The ALTO client should operate in different query modes depending on the implementation of ALTO servers. If an ALTO server does not support incremental updates using server-sent events (SSE) [DRAFT-SSE], the ALTO client sends queries to this server periodically to get the latest resource information. If the cluster state changes after one query, the ALTO client will not be aware of the change until next query. If an ALTO server supports SSE, the ALTO client only sends one query to the ALTO server to get the initial cluster information. When the resource state changes, the ALTO client will be notified by the ALTO server through SSE.

7.4. ALTO Server

ALTO servers are deployed at different sites around the world, and at strategic locations in the network itself, to provide information about different types of resources in the cluster networks in response to queries from the ALTO client deployed in ExaO. Such information include topology, link bandwidth, memory I/O speed and CPU utilization at computing nodes, storage constraints in storage nodes and etc. Each ALTO server must provide basic information services as specified in [RFC7285] such as network map, cost map,

endpoint cost service (ECS), etc. To support the fine-grained multi-resource allocation in ExaO, each ALTO server should also provide more fine-grained information about different resources in clusters through ALTO extension services such as the routing state abstraction [DRAFT-RSA], path vector [DRAFT-PV], network graph [DRAFT-NETGRAPH], multi-cost [DRAFT-MC] and cost-calendar [DRAFT-CC] services.

7.5. Dataset Transfer Agents

Dataset transfer agents are deployed at each site and in the network as needed, and are responsible for the following functions:

- Receive and process instructions from the multi-resource orchestrator, e.g. starting a new transfer, aborting a running transfer and adjusting transfer parameters such as transfer rate and the number of connections.

- Monitor the status of dataset transfers and sends the updated status to the multi-resource orchestrator.

Different systems can adopt different dataset transfer agents, or different structured agent subsystems, depending on specific needs. For instance, in the CMS experiment, these agents are PhEDEx distributed agents.

7.6. Request Execution Agents

Request execution agents are deployed at each site and are responsible for the following functions:

- Receive and process instructions from the multi-resource orchestrator, e.g. placement and execution of data analytic sub-requests, abortion of running analytic tasks and etc.

- Monitor the status of data analytic tasks and send the updated status to the multi-resource orchestrator.

Depending on the supporting data analytic frameworks, different request execution agents may be deployed in each cluster. For instance, in the CMS experiment, both MPI and Hadoop execution agents are deployed.

7.7. Multi-Resource Orchestrator

The multi-resource orchestrator takes the decomposed dataset transfer and data analytic requests from the request parser and the cluster resource information collected by the ALTO client as input. It then makes (1) dataset transfer scheduling decisions, including dataset

replica selection, path selection, and bandwidth allocation, for all data transfer request, and (2) request parallelization decisions, such as which cluster each sub-request should be placed at the multi-resource orchestrator. These decisions are sent to data transfer agents and the request execution agents at different clusters for execution.

7.7.1. Orchestration Algorithms

The modular design of ExaO allows the adoption of different orchestration algorithms and methodologies, depending on the specific performance requirements. In Section 7.7.3, a max-min fairness resource allocation algorithm for dataset transfer is described as an example.

7.7.2. Online, Dynamic Orchestration

The multi-resource orchestrator should adjust the resource allocation decisions based on the progress of ongoing requests, the utilization and dynamics of cluster resources. In normal cases, the multi-resource orchestrator periodically collects such information and executes the orchestration algorithm. When it is notified of events such as request status update, cluster state update and etc., the orchestrator will also execute the orchestration algorithm to adjust resource allocations.

7.7.3. Example: A Max-Min Fairness Resource Allocation Algorithm

In this section, we describe a max-min fair resource allocation (MFRA) scheduling algorithm which aims to minimize the maximal time to complete a dataset transfer subject to a set of constraints. To make resource allocation decisions, MFRA requires sufficient network information including topology, link bandwidth and recent historical information in some cases. In a small-scale single-domain network, an SDN controller can provide the raw complete topology information for the MFRA algorithm. However, in a large-scale multi-domain science network such as CMS, providing the raw network topology is infeasible because (1) it would incur significant communication overhead; and (2) it would violate the privacy constraints of some sites. Several ALTO extension topology services including Abstract Path Vector [DRAFT-PV], Network Graphs [DRAFT-NETGRAPH] and RSA [DRAFT-RSA] can provide the fine-grained yet aggregated/abstract topology information for MFRA to efficiently utilize bandwidth resources in the network.

Ongoing pre-production deployment efforts of ExaO in the CMS network involve the implementation of the RSA service. Other than topology information, the additional input of the MFRA algorithm is the

priority of each class of flows, expressed in terms of upper and lower limits on the allocated bandwidth between the source and the destination for each data transfer requests.

The basic idea of the MFRA algorithm is to iteratively maximize the volume of data that can be transferred subject to the constraints. It works in quantized time intervals such that it schedules network paths and data volumes to be transferred in each time slot. When the DTR scheduler is notified of events such as the cancellation of a DTR, the completion of a DTR or network state changes, the MFRA algorithm will also be invoked to make updated network path and bandwidth allocation decisions.

In each execution cycle, MFRA first marks all transfers as unsaturated. Then it solves a linear programming model to find the common minimum transfer satisfaction rate (i.e., the ratio of transferred data volume in a time interval over the whole data volume of this request) that is satisfied by all transfer requests. With this common rate found, MFRA then randomly selects an unsaturated request in each iteration, increases its transfer rate as much as possible by finding residual paths available in the network, or by increasing the allocated bandwidth along an existing path, until it reaches its upper limit or can otherwise not be increased further, so it is saturated. At each iteration, newly saturated requests are removed from the subsequent process by fixing their corresponding rate value, and completed transfers are removed from further consideration. After all the data transfer rates are saturated in the given time slot, then a feasible set of data transfer volumes scheduled to be transferred in the slot across each link in the network can be derived.

The MFRA algorithm yields a full utilization of limited network resources such as bandwidth so that all DTR can be completed in a timely manner. It allocates network resources fairly so that no DTR suffers starvation. It also achieves load balance among the sites and the network paths crossing a complex network topology so that no site and no network link is oversubscribed. Moreover, MFRA can handle the case where particular routing constraints are specified, e.g., where all routes are fixed ahead of time, or where each transfer request only uses one single path in each time slot, by introducing an additional set of linear constraints.

8. Discussion

8.1. Deployment

The ExaO framework is the first step towards a new class of intelligent, SDN-driven global systems for data-intensive science programs involving a worldwide ensemble of sites and networks, such as CMS and ATLAS. ExaO relies heavily on the ALTO services for collecting and expressing abstract up-to-date cluster information, and the SDN centralized control capability to orchestrate the flows of dataset transfer requests and data analytic requests. It aims to provide a new operational paradigm in which science programs can use complex network and computing infrastructures with high throughput, while allowing for coexistence with other network traffic.

A prototype case study implementation of ExaO has been demonstrated on the Caltech/StarLight/Michigan/Fermilab SDN development testbed. Because this testbed is a single-domain network, the current ExaO prototype leverages the ALTO OpenDaylight controller, to collect topology information. The CMS experiment is currently exploring pre-production deployments of ExaO, looking towards future widespread production use. To achieve this goal, it is imperative to collect sufficient topology information from the various sites in the multi-domain CMS network, without causing any privacy leak. To this end, the ALTO RSA service [DRAFT-RSA] is under development. Furthermore, as will be discussed next, other ALTO topology extension services can also substantially improve the performance of ExaO.

8.2. Benefiting From ALTO Extension Topology Services

The current ALTO base protocol [RFC7285] exposes 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 end hosts called endpoints. Such an extreme abstraction leads to significant information loss on network topology [DRAFT-PV], which is key information for ExaO to make dynamic scheduling and resource allocation decisions. Though ExaO can still allocate resource for data transfer and analytic requests on this abstract view, the resource allocation decisions are suboptimal. Alternatively, feeding the raw, complete network topology of each site to ExaO is not desirable, either. First, this would violate privacy constraints of different sites. Secondly, a raw network topology would significantly increase the problem space and the solution space of the orchestrating algorithm, leading to a long computation time. Hence, ExaO desires an ALTO topology service that is able to provide only enough fine-grained topology information.

Several ALTO topology extension services including Abstract Path Vector [DRAFT-PV], Network Graphs [DRAFT-NETGRAPH] and RSA

[DRAFT-RSA] are potential candidates for providing fine-grained abstract network formation to ExaO. In addition, we propose that these services can also be used to provide information about computing and storage resources of different cluster/sites by viewing each computing node and storage node as a network element or abstract network element. For instance, the path vector service supports the capacity region query, which accepts multiple concurrent data flows as the input and returns the information of bottleneck resources, which could be a set of links, computing devices or storage devices, for the given set of concurrent flows. This information can be interpreted as a set of linear constraints for the multi-resource orchestrator, which can help data transfer and analytic requests better utilize multiple types of resources in different clusters.

8.3. Constraints of the MFRA Algorithm

The first constraint of the MFRA algorithm is computation overhead. The execution of MFRA involves solving linear programming problems repeatedly at every time slot. The overhead of computation time is acceptable for small sets of dataset transfer requests, but may increase significantly when handling large sets of requests, e.g., hundreds of transfer requests. Current efforts towards addressing this issue include exploring the feasibility of incremental computation of scheduling policies, and reducing the problem scale by finding the minimal equivalent set of constraints of the linear programming model. The latter approach can benefit substantially from the ALTO RSA service [DRAFT-RSA].

The second constraint is that the current version of MFRA does not involve dataset replica selection. Simply denoting the replica selection as a set of binary constraint will significantly increases the computation complexity of the scheduling process. Current efforts focus on finding efficient algorithms to make dataset replica selection.

9. Security Considerations

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

10. IANA Considerations

This document does not define any new media type or introduce any new IANA consideration.

11. Acknowledgments

The authors thank discussions with Kai Gao, Linghe Kong, Xiao Lin, Xin Wang, Y. Richard Yang and Jingxuan Zhang.

12. References

12.1. 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>.

12.2. Informative References

[DRAFT-CC] Randriamasy, S., Yang, R., Wu, Q., Deng, L., and N. Schwan, "ALTO Cost Calendar", 2017, <https://datatracker.ietf.org/doc/draft-ietf-alto-cost-calendar>.

[DRAFT-DC] Lee, Y., Bernstein, G., Dhody, D., and T. Choi, "ALTO Extensions for Collecting Data Center Resource Information", 2014, <https://datatracker.ietf.org/doc/draft-lee-alto-ext-dc-resource/>.

[DRAFT-MC] Randriamasy, S., Roome, W., and N. Schwan, "Multi-Cost ALTO", 2017, <https://datatracker.ietf.org/doc/draft-ietf-alto-multi-cost/>.

[DRAFT-NETGRAPH] Bernstein, G., Lee, Y., Roome, W., Scharf, M., and Y. Yang, "ALTO Topology Extensions: Node-Link Graphs", 2015, <https://tools.ietf.org/html/draft-yang-alto-topology-06>.

[DRAFT-PV] Bernstein, G., Lee, Y., Roome, W., Scharf, M., and Y. Yang, "ALTO Extension: Abstract Path Vector as a Cost Mode", 2015, <https://tools.ietf.org/html/draft-yang-alto-path-vector-01>.

[DRAFT-RSA]
Gao, K., Wang, X., Yang, Y., and G. Chen, "ALTO Extension: A Routing State Abstraction Service Using Declarative Equivalence", 2015, <https://datatracker.ietf.org/doc/draft-gao-alto-routing-state-abstraction/>.

[DRAFT-SSE]
Roome, W. and Y. Yang, "ALTO Incremental Updates Using Server-Sent Events (SSE)", 2015, <https://datatracker.ietf.org/doc/draft-ietf-alto-incr-update-sse/>.

[RFC7285] Alimi, R., Ed., Penno, R., Ed., Yang, Y., Ed., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, DOI 10.17487/RFC7285, September 2014, <http://www.rfc-editor.org/info/rfc7285>.

Authors' Addresses

Qiao Xiang
Tongji/Yale University
51 Prospect Street
New Haven, CT
USA

Email: qiao.xiang@cs.yale.edu

Harvey Newman
California Institute of Technology
1200 California Blvd.
Pasadena, CA
USA

Email: newman@hep.caltech.edu

Greg Bernstein
Grotto Networking
Fremont, CA
USA

Email: gregb@grotto-networking.com

Azher Mughal
California Institute of Technology
1200 California Blvd.
Pasadena, CA
USA

Email: azher@hep.caltech.edu

Justas Balcas
California Institute of Technology
1200 California Blvd.
Pasadena, CA
USA

Email: justas.balcas@cern.ch

Jingxuan Jensen Zhang
Tongji University
4800 Cao’an Hwy
Shanghai 201804
China

Email: jingxuan.n.zhang@gmail.com

Haizhou Du
Tongji/Yale University
51 Prospect Street
New Haven, CT
USA

Email: duhaizhou@gmail.com

Y. Richard Yang
Tongji/Yale University
51 Prospect Street
New Haven, CT
USA

Email: yry@cs.yale.edu

ALTO WG
Internet-Draft
Intended status: Standards Track
Expires: September 14, 2017

G. Bernstein
Grotto Networking
S. Chen
Tongji University
K. Gao
Tsinghua University
Y. Lee
Huawei
W. Roome
M. Scharf
Nokia
Y. Yang
Yale University
J. Zhang
Tongji University
March 13, 2017

ALTO Extension: Path Vector Cost Mode
draft-yang-alto-path-vector-04.txt

Abstract

The Application-Layer Traffic Optimization (ALTO) Service has defined network and cost maps to provide basic network information, where the cost maps allow only scalar (numerical or ordinal) cost mode values. This document introduces a new cost mode called path-vector to allow ALTO clients to support use cases such as capacity regions for applications. This document starts with a non-normative example called multi-flow scheduling (or capacity region) to illustrate that ALTO cost maps without path vectors cannot provide sufficient information. This document then defines path-vector as a new cost mode.

Requirements Language

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

Status of This Memo

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 September 14, 2017.

Table of Contents

1. Introduction 4
2. Overview of Approach 5
2.1. Graph Model and Path Vector Data Format 5
2.2. Network Element Property Map Data Format 6
2.3. Flow-based Query Data Format 6
2.4. Routing State Abstraction Service 6
3. Changes Since Version -03 6
4. Use Case: Capacity Region for Multi-Flow Scheduling 6
5. Protocol Extensions 8
5.1. Cost Type 8
5.1.1. Cost Metric 8
5.1.2. Cost Mode: Path Vector 9
5.2. Network Element Name 9
5.3. Entity Domains 9
5.3.1. NE Domain 9
5.3.2. ANE Domain 10
5.4. (Filtered) Network Element Property Map Resource 10
5.4.1. Media Type 10
5.4.2. HTTP Method 10
5.4.3. Accept Input Parameters 10
5.4.4. Capabilities 11

5.4.5. Uses 11
5.4.6. Response 11
5.5. Cost Map Extensions 11
5.5.1. Media Types 11
5.5.2. HTTP Method 11
5.5.3. Accept Input Parameters 11
5.5.4. Capabilities 11
5.5.5. Uses 12
5.5.6. Response 12
5.6. Filtered Cost Map Extensions 12
5.6.1. Media Type 12
5.6.2. HTTP Method 12
5.6.3. Accept Input Parameters 12
5.6.4. Capabilities 13
5.6.5. Uses 14
5.6.6. Response 14
5.7. Endpoint Cost Service Extensions 14
5.7.1. Media Type 14
5.7.2. HTTP Method 15
5.7.3. Accept Input Parameters 15
5.7.4. Capabilities 15
5.7.5. Uses 16
5.7.6. Response 16
5.8. Optional: RSA Extension 16
5.8.1. Media Type 16
5.8.2. HTTP Method 16
5.8.3. Accept Input Parameters 17
5.8.4. Capabilities 17
5.8.5. Response 17
6. Examples 17
6.1. Information Resource Directory Example 17
6.2. Network Element Property Map Example 19
6.3. Filtered Cost Map Example #1 19
6.4. Filtered Cost Map Example #2 21
6.5. Endpoint Cost Map Example #1 22
6.6. Endpoint Cost Map Example #2 23
7. Compatibility 24
7.1. Compatibility with Legacy ALTO Clients/Servers 24
7.2. Compatibility with Multi-Cost Extensions 25
7.3. Compatibility with Incremental Update 25
8. Design Decisions and Discussions 25
8.1. Path Vector or Path Graph? 25
8.2. Provide More General Calendar Extension? 26
9. Security Considerations 26
10. IANA Considerations 27
10.1. ALTO Cost Mode Registry 27
10.2. ALTO Cost Metric Registry 27
10.3. ALTO Entity Domain Registry 27

11. Acknowledgments 28
12. References 28
12.1. Normative References 28
12.2. Informative References 28
Authors' Addresses 29

1. Introduction

The ALTO base protocol [RFC7285] is designed for exposing network information through services such as the Network Map service and the Cost Map service. These services use the extreme "single-node" abstraction, which represents a whole network with a single node and hosts are connected to the node's access ports.

Although the "single-node" abstraction works well for many settings, new use cases, such as inter-datacenter data transfers and scientific high-performance computing data transfers, require additional network information beyond the single-node abstraction, to support application capabilities, in particular, the ability of application flow scheduling.

Specifically, providing network information to support application flow scheduling introduces multiple complexities. First, the underlying assumption of existing ALTO services is single-commodity flows. Hence, given the flow from a source to a destination, ALTO computes the network metrics of the flow and returns them to the application. The metrics for different flows are independent. Application flow scheduling, however, requires network information to compute application-desirable multi-commodity flows, where multiple flows under the control of the same application may share common network bottlenecks. This requirement is beyond the capability of the single-node abstraction adopted by the base ALTO protocol. Second, some flow scheduling problems may consider end-to-end metrics at the same time and thus require multiple costs to be updated simultaneously. Such a requirement, even though already addressed by [I-D.ietf-alto-multi-cost], still needs to be handled very carefully. Third, flow scheduling can be conducted with several independent sets of flows. Using the cross product of "src" and "dst" lists will introduce a lot of redundancies.

To address these complexities in supporting the new flow scheduling use case, this document specifies a path vector extension to the base ALTO protocol. This extension introduces a new family of cost types, which uses "path-vector" as cost mode and "ne" (network element) or "ane" (abstract network element) as cost metric. It also extends "Unified Property Map" defined in [I-D.roome-alto-unified-props] to address the issue of scalability and consistency in providing path vectors with fine-grained information, and declares "pid-flows" and

"endpoint-flows" to support queries for multiple independent flow sets. This document also registers new domains, entity specification and properties in the ALTO Entity Domain Registry.

The document is organized as follows. Section 2 gives an overview of the path vector extension. Section 4 gives an example of flow scheduling to illustrate the need to introduce cost mode "path-vector" and new cost metrics. Section 5 specifies the new cost mode and cost metrics, new domain types and entity properties, new resource Network Element Property Map, and protocol extensions for (Filtered) Cost Maps and Endpoint Cost Services. Section 6 presents examples of Information Resources, requests and responses. Section 7 discusses the compatibility issues with some other proposed ALTO extensions. Section 8 lists several to-be-determined design decisions. Section 9 discusses about security and Section 10 discusses about IANA Considerations.

2. Overview of Approach

This section presents a non-normative overview of the path vector extension defined in this document. It assumes the readers are familiar with Cost Map and Endpoint Cost Service defined in [RFC7285], and Unified Property Map defined in [I-D.roome-alto-unified-props].

2.1. Graph Model and Path Vector Data Format

In this document, the graph model presented to ALTO clients is represented by path vectors. A path vector between two entities (either PIDs or endpoints) is an array of Network Element Names, where each Network Element Name represents a network element (usually a link) in the path between the two entities.

A specific Network Element Name MUST represent the same network element in the same ALTO Network Element Property Map resource. Thus, ALTO clients can find the flows that share this specific network element by finding the source-destination pairs whose corresponding path vectors contain the Network Element Name.

The cost entries contained in an ALTO Cost Map or Endpoint Cost Map are formally defined in Section 11.2.3.6 [RFC7285] to be any type of JSON value. But the section also suggests that implementations may assume the cost values are numbers unless specifically defined by an extension. This document extends the definition of Cost Map and Endpoint Cost Map to allow the returned cost to be a path vector, which is a JSONArray of Network Element Names.

An example can be found in Section 6.3.

2.2. Network Element Property Map Data Format

An ALTO client may need not know all attributes associated with all network elements. Thus, Network Element Property Map is provided so after an ALTO client obtains the path vectors, it can use Network Element Names to selectively query the associated attributes in the corresponding Network Element Property Map.

2.3. Flow-based Query Data Format

Flow scheduling may involve multiple sets of flows which have different source-destination combinations. Using source and destination lists can lead to a lot of redundancies. To allow more flexible specification of path vector requests, two new filter types for ReqFilteredCostMap and ReqEndpointCostMap are specified in this document.

2.4. Routing State Abstraction Service

For security and scalability considerations, this document specifies an optional feature called Routing State Abstraction (RSA). Routing State Abstraction feature compresses the original path vector information without loss of equivalence. A Routing State Abstraction compression feature MUST be applied only when a (Filtered) Cost Map or Endpoint Cost Service provides the path vector cost type with cost metric being "ane".

3. Changes Since Version -03

- Define "ne" and "ane" as the only cost metrics of the cost mode "path-vector".
- Define new entity domains for network property map and add the "availbw", "delay" property to these domains.
- Use Unified Property service to query properties of network elements.
- Augment the request schema of the (Filtered) Cost Map and Endpoint Cost Service.

4. Use Case: Capacity Region for Multi-Flow Scheduling

Consider the case that routing is given. Then what application-layer traffic optimization will focus on is traffic scheduling among application-layer paths. Specifically, assume that an application has control over a set of flows $F = \{f_1, f_2, ..., f_{|F|}\}$. If routing is given, what the application can control is $x_1, x_2, ...,$

x_|F|, where x_i is the amount of traffic for flow i. Let x = [x_1, ..., x_|F|] be the vector of the flow traffic amounts. Due to shared links, feasible values of x where link capacities are not exceeded can be a complex polytype.

Specifically, consider a network as shown in Figure 1. The network has 7 switches (sw1 to sw7) forming a dumb-bell topology. Switches sw1/sw3 provide access on one side, sw2/sw4 provide access on the other side, and sw5-sw7 form the backbone. End hosts eh1 to eh4 are connected to access switches sw1 to sw4 respectively. Assume that the bandwidth of each link is 100 Mbps.

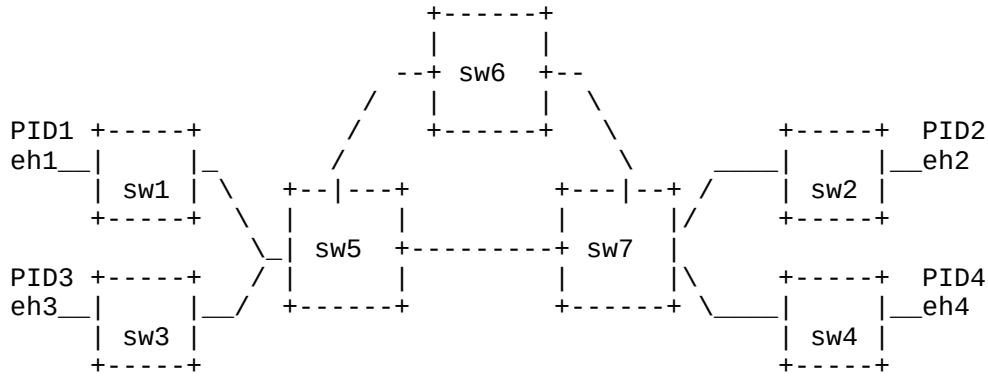


Figure 1: Raw Network Topology.

The single-node ALTO topology abstraction of the network is shown in Figure 2.

Figure 2: Base Single-Node Topology Abstraction.

Consider an application overlay (e.g., a large data analysis system) which needs to schedule the traffic among a set of end host source-destination pairs, say eh1 -> eh2, and eh3 -> eh4. The application

can request a cost map providing end-to-end available bandwidth, using 'availbw' as cost-metric and 'numerical' as cost-mode.

Assume that the application receives from the ALTO server that the bandwidth of eh1 -> eh2 and eh3 ->eh4 are both 100 Mbps. It cannot determine that if it schedules the two flows together, whether it will obtain a total of 100 Mbps or 200 Mbps. This depends on whether the routing paths of the two flows share a bottleneck in the underlying topology:

- Case 1: If eh1 -> eh2 and eh3 -> eh4 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 eh1 -> eh2 and eh3 -> eh4 share a 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 aforementioned cases, the network needs to provide more details. The path vector extension defined in this document resolves this issue.

See [I-D.bernstein-alto-topo] for a survey of use-cases where extended network topology information is needed.

5. Protocol Extensions

This section formally specifies the path vector extension.

5.1. Cost Type

The path vector extension defined in this document extends the Cost Types as specified in Section 6.1 of [RFC7285] .

5.1.1. Cost Metric

This document specifies two new cost metrics: "ne" and "ane". Both cost metrics are of type CostMetric as defined in Section 10.6 of [RFC7285]. These cost metrics MUST NOT be used when the cost mode is not "path-vector" unless it is explicitly specified in a future extension. An ALTO server with path vector extension MUST support at least one of these two cost metrics.

Cost metric "ne": This cost metric MUST be encoded as the JSONString "ne". When cost metric is "ne", Network Element Names contained in the path vectors MUST be resource-specific. In this case, different path vector queries to the same (Filtered) Cost Map or

Endpoint Cost Service MUST have the same Network Element Property Map in the responses.

Cost metric "ane": This cost metric MUST be encoded as the JSONString "ane". When cost metric is "ane", Network Element Names contained in the path vector MUST be query-specific. In this case, different path vector queries to the same (Filtered) Cost Map or Endpoint Cost Service MAY have different Network Element Property Maps.

5.1.2. Cost Mode: Path Vector

This document extends the cost mode as defined in Section 10.5 of [RFC7285] by allowing an ALTO server to provide a new cost mode other than "numerical" and "ordinal". The path vector cost mode is of type CostMode and is encoded as the JSONString "path-vector".

A (Filtered) Cost Map resource or Endpoint Cost Service, when queried with this cost mode, MUST return a CostMapData or EndpointCostMapData whose costs are JSONArrays of type NetworkElementName as specified in Section 5.2.

This cost mode MUST be used with either cost metric "ne" or "ane" unless it is explicitly specified by a future extension.

5.2. Network Element Name

This document also extends [RFC7285] with a new basic data type: Network Element Name. A Network Element Name is of type EntityAddr as defined in Section 2.3 of [I-D.roome-alto-unified-props] and is encoded as a JSONString. A Network Element Name MUST be an EntityAddr either of the NE domain (Section 5.3.1) or of the ANE domain (Section 5.3.2).

5.3. Entity Domains

This document specifies two new domains in addition to the ones in [I-D.roome-alto-unified-props].

5.3.1. NE Domain

5.3.1.1. Domain Name

ne

5.3.1.2. Domain-Specific Entity Addresses

Entity address of NE domain MUST be encoded as a JSONString with the same format as PID name defined in Section 10.1 of [RFC7285].

5.3.2. ANE Domain

5.3.2.1. Domain Name

ane

5.3.2.2. Domain-Specific Entity Addresses

The same as Section 5.3.1.2.

5.4. (Filtered) Network Element Property Map Resource

This document extends the base ALTO protocol with a new (filtered) resource: (Filtered) Network Element Property Map.

A Network Element Property Map MUST be a Property Map as defined in Section 4 of [I-D.roome-alto-unified-props] and a Filtered Network Element Property Map MUST be a Filtered Property Map as defined in Section 5 of [I-D.roome-alto-unified-props].

5.4.1. Media Type

The media type of a (Filtered) Network Element Property Map resource is "application/alto-propmap+json".

5.4.2. HTTP Method

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

An ALTO Filtered Network Element Property Map is requested using the HTTP POST method.

5.4.3. Accept Input Parameters

Network Element Property Map does not accept any input parameters.

The input parameters of a Filtered Network Element Property Map MUST conform to the format in Section 5.3 of [I-D.roome-alto-unified-props]. The EntityAddr in the request MUST have the same format as Network Element Name specified in Section 5.2.

5.4.4. Capabilities

A (Filtered) Network Element Property Map MUST have capabilities "domain-types" and "prop-types" as defined in Section 4.4 of [I-D.roome-alto-unified-props]. The "domain-types" capability MUST contain either "ne" or "ane" but not both at the same time and the "prop-types" capability MUST contain property type "availbw".

5.4.5. Uses

None.

5.4.6. Response

The "vtag" field MUST be included in the "meta" field of a response to a (Filtered) Network Element Map, with the same format as defined in Section 11.2.1.6 of [RFC7285].

The response is the same as for the Property Map, as defined in Section 4.6 of [I-D.roome-alto-unified-props], except that only the requested entities and properties are returned for Filtered Network Element Map. Examples can be found in Section 6.2.

5.5. Cost Map Extensions

5.5.1. Media Types

The same as Section 11.2.3.1 of [RFC7285].

5.5.2. HTTP Method

The same as Section 11.2.3.2 of [RFC7285].

5.5.3. Accept Input Parameters

The same as Section 11.2.3.3 of [RFC7285].

5.5.4. Capabilities

If a Cost Map resource supports the path vector extension defined in this document, its "cost-type-names" capability MUST have exactly one single cost type with the cost mode being "path-vector" and the cost metric being either "ne" or "ane", unless it is explicitly specified by a future extension.

5.5.5. Uses

The same as Section 11.2.3.5 of [RFC7285].

5.5.6. Response

The response has the same format as in Section 4.1.3 of [I-D.ietf-alto-multi-cost], except the follows.

- If cost mode is "path-vector", the costs MUST be JSONArrays of Network Element Names.
- If cost mode is "path-vector", the "meta" field MUST include the "nep-map" field, whose value is of type IRDResourceEntry as defined in Section 9.2.2 of [RFC7285] and represents the Filtered Network Element Property Map associated with this Cost Map as defined in Section 5.4. An ALTO server providing this resource MUST verify the following conditions are met:
- If cost metric of this Cost Map resource is "ne", the "nep-map" entry MUST have "domain-types" capability which includes domain name "ne".
- If cost metric of this Cost Map resource is "ane", the "nep-map" entry MUST have "domain-types" capability which includes domain name "ane".

ALTO servers MUST also verity the conditions in Section 5.1.1 are also met.

5.6. Filtered Cost Map Extensions

5.6.1. Media Type

The same as Section 11.3.2.1 of [RFC7285].

5.6.2. HTTP Method

The same as Section 11.3.2.2 of [RFC7285].

5.6.3. Accept Input Parameters

This document extends the ReqFilteredCostMap as follows:

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

  object {
    PIDName src;
    PIDName dst;
  } PIDFlow;
```

pid-flows: A list of PID src to PID dst for which path costs are to be returned.

pids: As defined in Section 11.3.2.3 of [RFC7285].

cost-type, multi-cost-types, testable-cost-types, constraints, or-constraints:
As defined in Section 4.1.2 of [I-D.ietf-alto-multi-cost]. A valid query MUST be considered a path vector query, either when "cost-type" of this query exists with "path-vector" cost mode, or when "multi-cost-types" exists and exact one cost type uses "path-vector" cost mode. For a path vector query, the path vector cost type MUST follow the definition in Section 5.1, otherwise the ALTO server SHOULD return an error with error code "E_INVALID_FIELD_VALUE". If "multi-cost-types" contains multiple path vector cost types, ALTO servers SHOULD return an error with the error code "E_INVALID_FIELD_VALUE". If a query is a path vector query and its "constraints" or "or-constraints" field is present, the "testable-cost-types" field MUST be explicitly specified and MUST NOT include any path vector cost type. If a path vector cost type is included, an ALTO server SHOULD return an error with the error code "E_INVALID_FIELD_VALUE".

An ALTO client MUST specify either "pids" or "pid-flows", but MUST NOT specify both in a single query.

5.6.4. Capabilities

A Filtered Cost Map with the path vector extension MAY have the "flow-based-filter" in its IRD capabilities.

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

flow-based-filter: If the value is true, the ALTO server allows ALTO clients to use "pid-flows" to query the Filtered Cost Map. If false or not present, ALTO clients MUST NOT include this field in the queries and the ALTO server SHOULD return an error with the error code "E_INVALID_FIELD_TYPE" for the queries including this field.

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

max-cost-types and testable-cost-type-names: As defined in Section 4.1.1 of [I-D.ietf-alto-multi-cost]. If a cost type with "path-vector" mode is included in "cost-type-names", either the "testable-cost-type-names" field is explicitly specified which MUST NOT include any path vector cost type, or the "testable-cost-type-names" field is empty where ALTO clients MUST interpret this as specified in Section 4.1.1 of [I-D.ietf-alto-multi-cost] except that the resource MUST NOT accept tests on any path vector cost type.

5.6.5. Uses

The same as Section 5.5.5.

5.6.6. Response

The response MUST have the same format as defined in Section 5.5.6 with the supplement that if a query uses the field "pid-flows", the response MUST still conform to the CostMapData format defined in Section 11.2.3.6 of [RFC7285]. Examples can be found in Section 6.3.

5.7. Endpoint Cost Service Extensions

5.7.1. Media Type

The same as Section 11.5.1.1 of [RFC7285].

5.7.2. HTTP Method

The same as Section 11.5.1.2 of [RFC7285].

5.7.3. Accept Input Parameters

This document extends the input parameters of Endpoint Cost Service as follows:

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

  object {
    TypedEndpointAddr src;
    TypedEndpointAddr dst;
  } EndpointFlow;
```

endpoint-flows: A list of source-destination endpoint pairs for which path costs are to be returned.

endpoints: As defined in Section 11.5.1.3 of [RFC7285].

cost-type, multi-cost-types, testable-cost-types, constraints, or-constraints:
As defined in Section 5.6.3.

ALTO clients MUST specify either "endpoints" or "endpoint-flows", but MUST NOT specify both in the same query.

5.7.4. Capabilities

This document defines EndpointCostMapCpabilities the same as FilteredCostMapCapabilities in Section 5.6.4.

If the value of capability flow-based-filter is true, the ALTO server MUST be able to process "endpoint-flows" in a query. If the value is false or not present, ALTO clients MUST assume the "endpoint-flows" is not supported and ALTO servers SHOULD return an error with the error code "E_INVALID_FIELD_TYPE" if "endpoint-flows" is included in the query.

5.7.5. Uses

The same as Section 11.5.1.5 of [RFC7285].

5.7.6. Response

The response has the same format as in Section 4.2.3 of [I-D.ietf-alto-multi-cost] for compatibility, except the follows.

- If cost mode is "path-vector", the costs MUST be JSONArrays of Network Element Names.
- If cost mode is "path-vector", the "meta" field MUST include the "nep-map" field, whose value is of type IRDResourceEntry as defined in Section 9.2.2 of [RFC7285] and represents the Filtered Network Element Property Map associated with this Endpoint Cost Service as defined in Section 5.4. An ALTO server providing this resource MUST verify the following conditions are met:
- If cost metric of this Endpoint Cost Service is "ne", the "nep-map" entry MUST have "domain-types" capability which includes domain name "ne".
- If cost metric of this Endpoint Cost Service is "ane", the "nep-map" entry MUST have "domain-types" capability which includes domain name "ane".

ALTO servers MUST also verity the conditions in Section 5.1.1 are also met.

Examples can be found in Section 6.5.

5.8. Optional: RSA Extension

5.8.1. Media Type

RSA extension MUST NOT change media types of (Filtered) Cost Map resource or Endpoint Cost Service.

5.8.2. HTTP Method

RSA extension MUST NOT change HTTP method for (Filtered) Cost Map or Endpoint Cost Service.

5.8.3. Accept Input Parameters

RSA extension SHOULD NOT change the input parameters of a Filtered Cost Map or an Endpoint Cost Service unless it is explicitly specified by a future extension.

5.8.4. Capabilities

If a (Filtered) Cost Map or an Endpoint Cost Service supports RSA extension, the "cost-type-names" MUST have one cost type with "path-vector" cost mode and "ane" cost metric, and ALTO clients MUST ignore this field when no such path vector cost type exists. The resource/service MUST also have the field "rsa" explicitly specified to "true" in the "capabilities" field. If the "rsa" field has a value of "false" or is not present, ALTO clients MUST assume this resource/service does not provide RSA compression.

An example can be found in Section 6.1.

5.8.5. Response

RSA extension SHOULD NOT change the output of a (Filtered) Cost Map or an Endpoint Cost Service unless it is explicitly specified in a future extension.

6. Examples

6.1. Information Resource Directory Example

Here is an example of an ALTO server’s Information Resource Directory.

```
  "meta" {
    "cost-types": {
      "pv-ne": {
        "cost-mode": "pv",
        "cost-metric": "ne"
      },
      "pv-ane": {
        "cost-mode": "pv",
        "cost-metric": "ane"
      },
      "num-hopcount": {
        "cost-mode": "numerical",
        "cost-metric": "hopcount"
      },
      "num-routingcost": {
        "cost-mode": "numerical",
```

```
        "cost-metric": "routingcost"
      },
      "num-delay": {
          "cost-mode": "numerial",
          "cost-metric": "delay"
      },
      "num-availbw": {
        "cost-mode": "numerical",
        "cost-metric": "availbw"
      }
    }
  },
  "resource": {
    "default-network-map": {
      "uri": "http://alto.example.com/networkmap",
      "media-type": "application/alto-networkmap+json"
    },

    ... Filtered Cost Map Resource ...

    "filtered-multi-cost-map1": {
      "uri": "http://alto.example.com/costmap/multi/filtered1",
      "media-type": "application/alto-costmap+json",
      "accepts": "application/alto-costmapfilter+json",
      "uses": ["default-network-map"],
      "capabilities": {
        "max-cost-types": 3,
        "cost-type-names": ["pv-ne", "num-availbw", "num-hopcount"],
        "testable-cost-types-names": ["num-availbw", "num-hopcount"]
      }
    },

    "filtered-multi-cost-map2": {
      "uri": "http://alto.example.com/costmap/multi/filtered2",
      "media-type": "application/alto-costmap",
      "accepts": "application/alto-costmapfilter+json",
      "uses": ["default-network-map"],
      "capabilities": {
        "max-cost-types":2,
        "cost-type-names": ["pv-ne", "num-routingcost", "num-delay"],
        "cost-constraints": true
      }
    }
    ... Endpoint Cost Map Resource ...

    "default-endpoint-cost-map": {
      "uri": "http://alto.example.com//endpointcost/lookup/ne-ane",
      "media-type": "application/alto-endpointcostmap+json",
```

```
      "accepts": "application/alto-endpointcostparams+json",
      "capabilities": {
        "rsa": true,
        "max-cost-types": 4,
        "cost-type-names": ["pv-ne", "pv-ane", "num-routingcost", "num-hopcount"
],
        "testable-cost-types-names": ["num-hopcount", "num-routingcost"]
      }
    }
  }
```

6.2. Network Element Property Map Example

```
POST /propmap/lookup/nep-map HTTP/1.1
Host: alto.example.com
Accept: application/alto-propmap+json,application/alto-error+json
Content-Length: [TBD]
Content-Type: application/alto-propmapparams+json

{
  "entities" : [ "ne:ne12",
                 "ne:ne23",
                 "ne:ne34" ],
  "properties" : [ "availbw", "delay" ]
}

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

{
  "meta": {
      "vtag": {
          "resource-id": "default-network-element-prop-map",
          "tag": "babbc14521772381472bffefff627813909875dd"
      }
  }
  "property-map": {
    "ne:ne12": { "availbw": "90", "delay": "30" },
    "ne:ne23": { "availbw": "80", "delay": "15" },
    "ne:ne34": { "availbw": "70", "delay": "25" }
  }
}
```

6.3. Filtered Cost Map Example #1

Assume that the available bandwidth between PID1 and PID3 is less than 50.

```
POST /costmap/multi/filtered1 HTTP/1.1
Host: alto.example.com
Accept: application/alto-costmap+json,application/alto-error+json
Content-Length: [TBD]
Content-Type: application/alto-costmapfilter+json

{
  "cost-type": {
    "cost-mode": "path-vector",
    "cost-metric": "ne"
  },
  "testable-cost-types": [
    { "cost-mode": "numerical", "cost-metric": "availbw" },
    { "cost-mode": "numerical", "cost-metric": "hopcount" }
  ],
  "or-constraints": [
    ["[0] ge 50", "[1] le 100"]
  ],
  "pid-flows": [
    { "src": "PID1", "dst": "PID2" }
    { "src": "PID1", "dst": "PID3" }
    { "src": "PID3", "dst": "PID4" }
  ]
}

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

{
  "meta": {
    "dependent-vtags": [
      {
        "resource-id": "my-default-network-map",
        "tag": "75ed013b3cb58f896e839582504f622838ce670f"
      }
    ],
    "cost-type": {
      "cost-mode": "path-vector",
      "cost-metric": "ne"
    },
    "nep-map": {
      "uri": "http://alto.example.com/propmap/lookup/nep-map",
      "media-type": "application/alto-prompmap+json",
      "accepts": "application/alto-propmapparams+json",
      "capabilities": {
        "domain-types": ["ne"],
        "prop-types": ["availbw", "delay"]
```

```
          }
        }
      },
      "cost-map": {
        "PID1": { "PID2": [ "ne:ne12", "ne:ne23" ] },
        "PID3": { "PID4": [ "ne:ne23", "ne:ne34" ] }
      }
    }
```

6.4. Filtered Cost Map Example #2

Assume that the delay between PID1 and PID2 is greater than 80.

```
  POST /costmap/multi/filtered2 HTTP/1.1
  Host: alto.example.com
  Accept: application/alto-costmap+json,application/alto-error+json
  Content-Length: [TBD]
  Content-Type: application/alto-costmapfilter+json

  {
    "multi-cost-types": [
      { "cost-mode": "path-vector", "cost-metric": "ne" },
      { "cost-mode": "numerical", "cost-metric": "delay" }
    ],
    "testable-cost-types": [
      { "cost-mode": "numerical", "cost-metric": "delay" }
    ],
    "constraints": [ "[0] le 80" ],
    "pid-flows": [
      { "src": "PID1", "dst": "PID2" },
      { "src": "PID1", "dst": "PID3" },
      { "src": "PID3", "dst": "PID4" }
    ]
  }

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

  {
    "meta": {
      "dependent-vtags": [
        {
          "resource-id": "my-default-network-map",
          "tag": "75ed013b3cb58f896e839582504f622838ce670f"
        }
      ],
      "cost-type": {},
```

```
        "multi-cost-types": [
          { "cost-mode": "path-vector", "cost-metric": "ne"},
          { "cost-mode": "numerical", "cost-metric": "delay"}
        ],
        "nep-map": {
          "uri": "http://alto.example.com/propmap/lookup/nep-map",
          "media-type": "application/alto-prompmap+json",
          "accepts": "application/alto-propmapparams+json",
          "capabilities": {
            "domain-types": ["ne"],
            "prop-types": ["availbw", "delay"]
          }
        }
      },
      "cost-map": {
        "PID1": { "PID3": [ [ "ne:ne23", "ne:ne45" ], 60 ] },
        "PID3": { "PID4": [ [ "ne:ne23", "ne:ne34" ], 40 ] }
      }
    }
```

6.5. Endpoint Cost Map Example #1

Assume that the delay between ipv4:203.0.113.45 and ipv4:198.51.100.34 is greater than 100.

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

{
  "cost-type": {
    "cost-mode": "path-vector",
    "cost-metric": "ne"
  },
  "testable-cost-types": [
    {
      "cost-mode": "numerical",
      "cost-metric": "delay"
    }
  ],
  "constraints": [
    "[0] le 100"
  ],
  "endpoint-flows": [
    { "src": "ipv4:192.0.2.2", "dst": "ipv4:192.0.2.89" },
    { "src": "ipv4:192.0.2.2", "dst": "ipv4:198.51.100.34" },
```

```
      { "src": "ipv4:203.0.113.45", "dst": "ipv4:198.51.100.34" }
    ]
  }


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

  {
      "meta": {
      "cost-type": {
        "cost-mode": "path-vector",
        "cost-metric": "ne"
      },
      "nep-map": {
        "uri": "http://alto.example.com/propmap/lookup/nep-map",
        "media-type": "application/alto-prompmap+json",
        "accepts": "application/alto-propmapparams+json",
        "capabilities": {
          "domain-types": ["ne"],
          "prop-types": ["availbw", "delay"]
        }
      }
    },
    "endpoint-cost-map": {
      "ipv4:192.0.2.2": {
        "ipv4:192.0.2.89": [ "ne:ne12", "ne:ne23" ],
        "ipv4:198.51.100.34": [ "ne:ne12", "ne:ne24", "ne:ne45" ]
      }
    }
  }
```

6.6. Endpoint Cost Map Example #2

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

  {
    "multi-cost-types": [
    {
      "cost-mode": "path-vector",
      "cost-metric": "ane"
    },
    {
```

```
      "cost-mode": "numerical",
      "cost-metric": "delay"
    }],
    "endpoint-flows": [
      { "src": "ipv4:192.0.2.2", "dst": "ipv4:192.0.2.89" },
      { "src": "ipv4:192.0.2.2", "dst": "ipv4:198.51.100.34" },
      { "src": "ipv4:203.0.113.45", "dst": "ipv4:198.51.100.34" }
    ]
  }

  HTTP/1.1 200 OK
  Content-Length: [TBD]
  Content-Type: application/alto-endpointcost+json
  {
    "meta": {
      "cost-type": {},
      "multi-cost-types": [
      { "cost-mode": "path-vector", "cost-metric": "ane"},
      { "cost-mode": "numerical", "cost-metric": "delay"}
      ],
      "nep-map": {
        "uri": "http://alto.example.com/propmap/lookup/nep-map/31415926",
        "media-type": "application/alto-prompmap+json",
        "accepts": "application/alto-propmapparams+json",
        "capabilities": {
          "domain-types": ["ane"],
          "prop-types": ["availbw", "delay"]
        }
      }
    },
    "endpoint-cost-map": {
      "ipv4:192.0.2.2": {
        "ipv4:192.0.2.89": [ ["ane:ane12", "ane:ane23"], 45 ],
        "ipv4:198.51.100.34": [ ["ane:ane12", "ane:ane24"], 90 ]
      },
      "ipv4:203.0.113.45": {
        "ipv4:198.51.100.34": [ ["ane:ane34"], 120 ]
      }
    }
  }
```

7. Compatibility

7.1. Compatibility with Legacy ALTO Clients/Servers

Legacy ALTO clients SHOULD NOT send queries with path vector extension and ALTO servers with this extension SHOULD NOT have any compatibility issue. Legacy ALTO servers do not support cost types

with the "path-vector" cost mode and MUST NOT announce the extended cost types in IRD. Thus, ALTO clients MUST NOT send queries specified in this extension to legacy ALTO servers according to Section 11.3.2.3 [RFC7285].

7.2. Compatibility with Multi-Cost Extensions

Path Vector extension SHOULD be fully compatible with Multi-Cost extensions.

7.3. Compatibility with Incremental Update

There is no compatibility issue with incremental update extension.

8. Design Decisions and Discussions

8.1. Path Vector or Path Graph?

When we introduce the "path-vector" as a cost mode in the Cost Map, an unavoidable problem is how to handle multipath. Because a PID is a group of endpoints, it is common that there are multiple paths between two PIDs. The valid routing state information is all of the accessible paths. So in this scenario, the Cost Map Resource SHOULD provide the cost values including of the multiple paths.

A natural solution is to provide an array of path vectors as the cost value. Every path vector in this array means an accessible path between the source PID and the destination PID. It is different from the solution of the path vector extension which provides an array of network elements. So it requires to introduce a different cost mode. This document proposes this new cost mode named "path-graph".

However, the "path-graph" will increase the complexity of the Cost Map Response. Since the applications select ALTO as the protocol to get the network information rather than other topology-based solution such as I2RS, the major reason should be the simplicity. If we provide "path-graph" for each PID pairs, the ALTO client has to handle the complex data structure.

What’s more, the "path-vector" is powerful enough to express multiple paths. The simple solution is to list the network elements of all accessible paths in a single path vector. This solution will lose the information about paths. Another solution is to define the path as a new type of network elements. In this way, the path vector can provide an array of paths. Each element of this array contains a path vector of network elements in the Network Element Property Map.

So in this document, we just introduce "path-vector" as the only required cost mode for routing state information.

8.2. Provide More General Calendar Extension?

Cost Calendar is proposed as a useful ALTO extension to provide the historical cost values for Filtered Cost Map Service and Endpoint Cost Service. Since path vector is an extension to these services, it SHOULD be compatible with Cost Calendar extension.

However, the calendar of a path-vector (Endpoint) Cost Map is insufficient for the application which requires the historical data of routing state information. The (Endpoint) Cost Map can only provide the changes of the paths. But more useful information is the history of network element properties which are recorded in the dependent Network Element Property Map.

Before the Unified Property Map is introduced as a new ALTO service, Filtered Cost Map Service and Endpoint Cost Service are the only resources which require the calendar supported. Because other resources don’t have to be updated frequently. But Network Element Property Map as a use case of Unified Property Map will collect the real-time information of the network. It SHOULD be updated as soon as possible once the metrics of network elements change.

So the requirement is to provide a general calendar extension which not only meets the Filtered Cost Map and Endpoint Cost Service but also applies to the Property Map Service.

9. Security Considerations

We can identify multiple potential security issues. A main security issue is network privacy, as the path-vector information may reveal more network internal structures than the more abstract single-node abstraction. The network should consider protection mechanisms to reduce information exposure, in particular, in settings where the network and the application do not belong to the same trust domain. On the other hand, in a setting of the same trust domain, a key benefit of the path-vector abstraction is reduced information transfer from the network to the application.

The path-vector query may also reveal more information about the application. In particular, the application may reveal all potential transfers sites (e.g., where the data source is replicated, and where the potential replication sites are). The application should evaluate the potential privacy concerns.

Beyond the privacy issues, the computation of the path-vector is unlikely to be cachable, in that the results will depend on the particular requests (e.g., where the flows are distributed). Hence, this service may become an entry point for denial of service attacks on the availability of an ALTO server. Hence, authenticity and authorization of this ALTO service may need to be better protected.

10. IANA Considerations

10.1. ALTO Cost Mode Registry

This document specifies a new cost mode "path-vector". However, the base ALTO protocol does not have a Cost Mode Registry where new cost mode can be registered. This new cost mode will be registered once the registry is defined either in a revised version of [RFC7285] or in another future extension.

10.2. ALTO Cost Metric Registry

Two new cost metrics need to be registered in the "ALTO Cost Metric Registry", listed in Table 1.

Identifier	Intended Semantics
ne	See Section 5.1.1
ane	See Section 5.1.1

Table 1: ALTO Cost Metrics

10.3. ALTO Entity Domain Registry

As proposed in Section 9.2 of [I-D.roome-alto-unified-props], "ALTO Entity Domain Registry" is requested. Besides, two new domains are to be registered, listed in Table 2.

Identifier	Entity Address Encoding	Hierarchy & Inheritance
ne	See Section 5.3.1.2	None
ane	See Section 5.3.2.2	None

Table 2: ALTO Entity Domain

11. Acknowledgments

The authors would like to thank discussions with Andreas Voellmy, Erran Li, Haibin Son, Haizhou Du, Qiao Xiang, Tianyuan Liu, Xiao Shi, Xin Wang, and Yan Luo.

12. References

12.1. 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>.

12.2. 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.bernstein-alto-topo]
Bernstein, G., Yang, Y., and Y. Lee, "ALTO Topology Service: Uses Cases, Requirements, and Framework", draft-bernstein-alto-topo-00 (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.ietf-alto-cost-calendar]
Randriamasy, S., Yang, Y., Wu, W., Lingli, D., and N. Schwan, "ALTO Cost Calendar", draft-ietf-alto-cost-calendar-01 (work in progress), February 2017.

[I-D.ietf-alto-incr-update-sse]
Roome, W. and Y. Yang, "ALTO Incremental Updates Using Server-Sent Events (SSE)", draft-ietf-alto-incr-update-sse-03 (work in progress), September 2016.

[I-D.ietf-alto-multi-cost]
Randriamasy, S., Roome, W., and N. Schwan, "Multi-Cost ALTO", draft-ietf-alto-multi-cost-05 (work in progress), February 2017.

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

[I-D.roome-alto-unified-props]
Roome, W., "Extensible Property Maps for the ALTO Protocol", draft-roome-alto-unified-props-01 (work in progress), July 2016.

[RFC7285] Alimi, R., Ed., Penno, R., Ed., Yang, Y., Ed., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, DOI 10.17487/RFC7285, September 2014, <http://www.rfc-editor.org/info/rfc7285>.

Authors' Addresses

Greg Bernstein
Grotto Networking
Fremont, CA
USA

Email: gregb@grotto-networking.com

Shiwei Dawn Chen
Tongji University
4800 Caoan Road
Shanghai 201804
China

Email: dawn_chen_f@hotmail.com

Kai Gao
Tsinghua University
Beijing Beijing
China

Email: gaok12@mails.tsinghua.edu.cn

Young Lee
Huawei
TX
USA

Email: leeyoung@huawei.com

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

Phone: +1-908-582-7974
Email: wendy.roome@nokia.com

Michael Scharf
Nokia
Germany

Email: michael.scharf@nokia.com

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

Email: yry@cs.yale.edu

Jingxuan Jensen Zhang
Tongji University
4800 Caoan Road
Shanghai 201804
China

Email: jingxuan.n.zhang@gmail.com