ALTO
Internet-Draft
Intended status: Standards Track
Expires: 16 September 2020

J. Zhang
Tongji University
K. Gao
Sichuan University
J. Wang

Q. Xiang
Y.R. Yang
Yale University
15 March 2020

ALTO Extension: Flow-based Cost Query
draft-gao-alto-fcs-07

Abstract

ALTO cost maps and endpoint cost services map a source-destination pair into a cost value. However, current filter specifications, which define the set of source-destination pairs in an ALTO query, have two limitations: 1) Only very limited address types are supported (IPv4 and IPv6), which is not sufficient to uniquely identify a flow in networks with fine-grained routing, such as the emerging Software Defined Networks; 2) The base ALTO protocol only defines filters enumerating all sources and all destinations, leading to redundant information in the response; 3) Cannot distinguish transmission types of flows in the query, which makes the server hard to respond the accurate resource consumption. To address these three issues, this document extends the base ALTO protocol with a more fine-grained filter type which allows ALTO clients to select only the concerned source-destination pairs and announce the flow-specific information like data transmission type, and a more expressive address space which allows ALTO clients to make queries beyond the limited IP addresses.

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 https://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 16 September 2020.

Table of Contents

1. Introduction

Application-Layer Traffic Optimization (ALTO) protocol [RFC7285] defines several cost query services, like Filtered Cost Map and Endpoint Cost Service, to allow applications to query path costs. Generally, an ALTO cost query service can be regarded as a function transforming a given subset of a specific query space into a network view abstract, where the subset is specified by a PID filter or an endpoint filter. However, the current specification has some limitations.

First, in the base ALTO protocol [RFC7285], the endpoint filter only contains the source and destination IP addresses. In practice, both Internet Service Providers (ISP) and local network administrators may conduct policy-based routing, e.g., P2P traffic may be constrained and has a smaller bandwidth than HTTP traffic. Also, web services with different QoS requirements may be hosted on the same machine with the same IP address but different paths with different QoS metrics.

Second, in the base ALTO protocol [RFC7285], the query space is defined by a source list and a destination list. For a query with N sources and M destinations, the response contains N*M entries. While such a query schema is well suited for peer-to-peer (P2P) applications where files of the same seed are stored on all hosts, it may lead to a lot of redundancy in use cases such as modern data analytics systems where replicas of the same dataset are stored on only a small subset of servers. Consider a system where the number

of replicas is 3 (the default in HDFS), jointly scheduling N concurrent transfers only needs a maximum of 3N entries but the base ALTO protocol may return up to N^2 entries.

Third, in the base ALTO protocol [RFC7285], the query does not distinguish among the different transmission types like unicast and multicast. For some use cases like the multi-flow scheduling demonstrated by [I-D.ietf-alto-path-vector], the data transmission between endpoints could be beyond unicast. And in those cases, different transmission types may affect the network resource consumption. If applications can receive the path costs distinguishing the different transmission types, it can help applications perform their data transmission decision better.

Thus, we conclude that the following additional requirements (AR) MUST be satisfied to allow ALTO clients make more accurate and efficient cost queries.

AR-1: The ALTO server SHOULD allow the ALTO client to specify accurate query space in cost query services.

The base ALTO protocol only includes IPv4 and IPv6 addresses as endpoint address types, which may not be sufficient to accurately identify an endpoint with emerging flow-based routing mechanisms. ALTO clients MAY suffer from suboptimal decisions because of such inaccuracy. Thus, the ALTO protocol SHOULD be extended so that clients are able to specify accurate query space, i.e., using more fine-grained endpoint address types.

AR-2: The ALTO server SHOULD allow the ALTO client to specify only the essential query space.

Existing PIDFilter (see Sec 11.3.2.3 in [RFC7285]) and EndpointFilter (see Sec 11.5.1.3 in [RFC7285]) represent the cross-product of sources and destinations, and can introduce a lot of redundancy in certain use cases. This limitation greatly harms the scalability of the ALTO protocol. Thus, the ALTO protocol SHOULD be extended so that ALTO clients are able to specify only the essential cost query space, i.e., the concerned source-destination pairs.

AR-3: The ALTO server SHOULD allow the ALTO client to specify different data transmission types for transimissions in the query space.

The input parameters of existing ALTO cost query services only allow the ALTO client to specify the queried transmissions by sources and destinations. The transmission between each source

and destination will always be considered as the unicast. This limitaiton may make the ALTO client lose the accurate available resources. Thus, the ALTO protocol SHOULD be extended so that ALTO clients are able to speicfy different transmission types.

In this document, we describe an ALTO extension specifying flow-based cost queries. The rest of this document is organized as follows. Section 3 introduces several new address types that extend the query space of ALTO cost services. Section 5 describes the extended schema on Filtered Cost Map (FCM) and Endpoint Cost Service (ECS) to support cost queries of arbitrary source-destination combinations with the optional flow-specific information. Section 6 and Section 7 discuss security and IANA considerations.

1.1. Requirement 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].

1.2. Terminology

This document uses the same terms as defined in [RFC7285] and [RFC8189] with the following additional term:

1.2.1. Flow

In this document, a flow refers to all communications between two endpoints. A flow is "valid" if and only if there CAN be valid communications between the two endpoints, which oftentimes requires that that two endpoint addresses have compatible address types.

1.2.2. Data Transmission Type

This document use the term "Data Transmission Type" or "Transmission Type" to indicate how an application sends the network flows. See Section 4.2.

2. Overview of Approaches

This section presents a non-normative overview of the extension to support flow-based cost query. It assumes the readers are familiar with Filtered Cost Map and Endpoint Cost Service defined in [RFC7285] and their extensions defined in [RFC8189].

2.1. Extended Endpoint Address

To allow ALTO clients specify accurate query space in cost query services (AR-1), this document defines several new endpoint address types. An endpoint address with a new type is referred to as an extended endpoint address.

Since the address types of both the source and the destination correspond to the same network flow, they MUST NOT conflict. This document defines an address type conflict table to indicate conflicts. If some source and destination address types in a query conflict with each others, an ALTO server MUST return the corresponding error.

2.2. Flow-based Filter

To allow ALTO clients specify only the essential query space in cost query services (AR-2), both PIDFilter and EndpointFilter in the base protocol MUST be extended. The extended filters are referred to as flow-based filters.

A straight-forward way of satisfying AR-2 is to have an ALTO client list all its concerned flows. Despite its simplicity, it MAY be too large in size, especially when many flows have common sources or common destinations in the query. Also from the implementation's perspective, it cannot reuse the functionality to parse a PIDFilter/ EndpointFilter.

Thus, the flow-based filters defined in this document allow ALTO clients to include multiple PIDFilter/EndpointFilter objects in the same query. Apparently, if we replace each PIDFilter/EndpointFilter of N sources and M destinations with NM filters that have exactly one source and destination, the two representations refer to the same set of flows. As a result, one can aggregate flows with common sources or destinations in one PIDFilter/EndpointFilter object without introducing redundant flows.

From the implementation's perspective, one MAY reuse an ALTO library which parses PIDFilter/EndpointFilter and/or converts them into a set of source-destination pairs.

2.3. Flow-specific Announcement

Some properties are only related to the transmission and cannot be encoded into endpoints, e.g., the data transmission type of a flow. However, these properties may help the ALTO client get more accurate costs.

To allow the ALTO client to specify these flow-specific properties (AR-3), this document introduces an extensible field in the flow-based filter. The ALTO client can announce the flow-specific information in this field. An announcement, whose type is encoded as a FlowSpecAnnounceType (Section 4.1), can represent transmission type, equal cost multipath assumption and other kinds of flow-specific information.

This document adopts an extensible design for this announcement field. Although only the data transmission type is defined in this document, other announcement properties can be defined in future documents and are not in the scope of this document.

3. Extended Endpoint Address

This document registers new address types and defines the corresponding formats for endpoint addresses of each new address type.

3.1. Address Type

The new AddressType identifiers defined in this document are as follows:

eth: An endpoint address with type "eth" is the address of an Ethernet interface. It is used to uniquely identify an endpoint in the data link layer.

domain: An endpoint address with type "domain" is the domain name of a web service. It is used to uniquely identify a web service which MAY be translated to one or more IPv4 address(es).

domain6: An endpoint address with type "domain6" is the domain name of a web service. It is used to uniquely identify a web service which MAY be translated to one or more IPv6 address(es).

tcp: An endpoint address with type "tcp" is the address of a TCP socket. It is used to uniquely identify an IPv4 TCP socket in the transport layer.

tcp6: An endpoint address with type "tcp6" is the address of a TCP socket. It is used to uniquely identify an IPv6 TCP socket in the transport layer.

udp: An endpoint address with type "udp" is the address of a UDP socket. It is used to uniquely identify an IPv4 UDP socket in the transport layer.

udp6: An endpoint address with type "udp6" is the address of a UDP socket. It is used to uniquely identify an IPv6 UDP socket in the transport layer.

3.2. Endpoint Address

This document defines EndpointAddr when AddressType is in Section 7.1.

3.2.1. MAC Address

An Endpoint Address of type "eth" is encoded as a MAC address, whose format is encoded as specified by either format EUI-48 in [EUI48] or EUI-64 in [EUI64].

3.2.2. Internet Domain Name

An Endpoint Address of type "domain" or "domain6" is encoded as a domain name in the Internet, as specified in Section 11 of [RFC2181]. It MUST have at least one corresponding A ("domain") or AAAA ("domain6") record in the DNS.

3.2.3. IPv4 Socket Address

An Endpoint Address of type "tcp" or "udp" is encoded as an IPv4 socket address. It is encoded as a string of the format Host:Port with the ":" character as a separator. The Host component of an IPv4 socket address is encoded as specified by either an IPv4 address (see Section 10.4.3.1 of [RFC7285]) or an IPv4-compatible domain name (see Section 3.2.2). The Port component of an IPv4 socket address is encoded as an integer between 1 and 65535.

3.2.4. IPv6 Socket Address

An Endpoint Address of type "tcp6" or "udp6" is encoded as an IPv6 socket address. It is also encoded as a string of the format Host:Port with the ":" character as a separator. The Host component of an IPv6 socket address is encoded as specified by either an IPv6 address (see Section 10.4.3.2 of [RFC7285]) enclosed in the "[" and "]" characters as recommended in [RFC2732] or an IPv6-compatible domain name (see Section 3.2.2). The Port component of IPv6 socket address is encoded as an integer between 1 and 65535.

3.3. Address Type Compatibility

In practice, a flow with endpoint addresses with different types MAY NOT be valid. For example, a source endpoint with an IPv4 address CANNOT establish a network connection with a destination endpoint with an IPv6 address. Neither can a source with a TCP socket address and a destination with a UDP socket address.

Thus, to explicitly define the compatibility between AddressType identifiers, every ALTO AddressType identifier MUST provide a list of AddressType identifiers that are compatible with it in the "ALTO Address Type Compatibility Registry" Section 7.2. For all sources and destinations in a PIDFilter/EndpointFilter, if the AddressType identifiers of a given pair DO NOT appear in the ALTO Address Type Compatibility Registry, an ALTO server MUST return an ALTO error response with the error code "E_INVALID_FIELD_VALUE" with optional information to help diagnose the incompatibility.

3.4. Examples

Some valid endpoint addresses are demonstrated as follows:

```
"eth:98-e0-d9-9c-df-81"
"domain:www.example.com"
"tcp:198.51.100.34:5123"
"udp6:[2000::1:2345:6789:abcd]:8080"
```

4. Flow-specific Announcement

Each flow-specific announcement refers to a property of the traffic between a set of source and destination pairs. The interpretation of a property, however, depends on the meaning, i.e., the type, of the property. This document uses flow-specific announcement type as an indicator of the "type" information, and requires the flow-specific announcement types be registered in an IANA registry called "ALTO Flow-specific Announcement Registry".

4.1. Flow-specific Announcement Type

It is encoded as a JSONString and has the same format as a PID Name defined in Section 10.1 in [RFC7285]. The type FlowSpecAnnounceType is used in this document to indicate a string of that format.

4.2. Data Transmission Type Announcement

This document defines a flow-specific announcement called the data transmission type. The type of this announcement is indicated by the string "transmission-type", and the value of this announcement is a JSONString of either "unicast", "anycast", "broadcast" or "multicast".

5. Extended Cost Query Filters

This section describes extensions to [RFC7285] and [RFC8189] to support flow-based cost queries.

This document uses the notation rules specified in Section 8.2 of [RFC7285] and also the notation rules for optional fields in Section 4 of [RFC8189].

5.1. Filtered Cost Map Extension

This document extends the Filtered Cost Map as defined in Section 11.3.2 of [RFC7285] and Section 4.1 of [RFC8189], by adding a new capability and new input parameters.

The media type, HTTP method, and "uses" specifications (described in Sections 11.3.2.1, 11.3.2.2, and 11.3.2.5 of [RFC7285], respectively) are not changed.

The format of the response is the same as defined in Section 4.1.3 of [RFC8189]. But this document recommends how to generate the response based on the extended input parameters.

5.1.1. Capabilities

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

* flow-based-filter

* flow-spec-announce

The capability "flow-based-filter" indicates whether this resource supports flow-based cost queries, and the capability "flow-spec-announce" indicates which flow-specific announcements are supported. The FilteredCostMapCapabilities object in Section 4.1.1 of [RFC8189] 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;]
        [FlowSpecAnnounceType flow-spec-announce<1..*>;]
      } 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 [RFC8189].

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

flow-spec-announce: It MUST NOT be present if "flow-based-filter" is not true. If present, the value is the an array of supported flow-specific announcement types.

5.1.2. Accept Input Parameters

The ReqFilteredCostMap object in Section 4.1.2 of [RFC8189] 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;]
        [ExtPIDFilter  pid-flows<1..*>;]
      } ReqFilteredCostMap;

      object {
        [FlowSpecAnnounceMap flow-spec-announce;]
      } ExtPIDFilter : PIDFilter;

      object-map {
        FlowSpecAnnounceType -> JSONValue;
      } FlowSpecAnnounceMap;
```

cost-type, multi-cost-types, testable-cost-types, constraints, or-constraints: As defined in Section 4.1.2 of [RFC8189].

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

pid-flows: Defined as a list of ExtPIDFilter objects. The ALTO server MUST interpret PID pairs appearing in multiple ExtPIDFilter objects as if they appeared only once. If the capability "flow-spec-announce" is present, the "flow-spec-announce" input parameter can be specified. The value of this field is of type FlowSpecAnnounceMap, which maps a FlowSpecAnnounceType to its corresponding value. The interpretation of the value MUST follow the definition of the flow-specific announcement in the ALTO Flow-specific Announcement Registry.

An ALTO client MUST include either "pids" or "pid-flows" in a query but MUST NOT include both at the same time.

5.2. Response

This document does not change the format of the response entity. But the ALTO server responds the request with "pid-flows" filter as follows:

The ALTO server MUST include the path costs of pairs in each ExtPIDFilter in the "pid-flows" filter. If the "flow-spec-announce" field is specified in some ExtPIDFilter, the path costs for flows in this ExtPIDFilter SHOULD respond the flow-specific information announced by this field.

5.3. Endpoint Cost Service Extension

This document extends the Endpoint Cost Service as defined in Section 11.5.1 of [RFC7285] and Section 4.2 of [RFC8189], by adding a new capability and input parameters.

The media type, HTTP method, and "uses" specifications (described in Sections 11.5.1.1, 11.5.1.2, and 11.5.1.5 of [RFC7285], respectively) are unchanged.

The format of the response is the same as defined in Section 4.2.3 of [RFC8189]. But this document recommends how to generate the response based on the extended input parameters.

5.3.1. Capabilities

The extension to EndpointCostCapabilities includes three additional members:

* flow-based-filter

* address-types

* flow-spec-announce

Only if the capability "flow-based-filter" is present and its value is "true", the ALTO server supports the flow-based extension for this endpoint cost service. The capability "address-types" indicates which endpoint address types are supported by this resource, it MUST NOT be specified if "flow-based-filter" is absent or the value is false. The capability "flow-spec-announce" indicates which flow-specific announcements are supported, just like it works in the Filtered Cost Map resource.

```
    object {
      [JSONBool    flow-based-filter;]
      [JSONString  address-types<0..*>;]
      [FlowSpecAnnounceType flow-spec-announce<1..*>;]
    } EndpointCostCapabilities : FilteredCostMapCapabilities;
```

flow-based-filter: If true, an ALTO Server MUST accept field "endpoint-flows" in the requests. If not present, this field MUST be interpreted as if it is specified false.

address-types: Defines a list of AddressType identifiers encoded as a JSONArray of JSONString. All AddressType identifiers MUST be registered in the "ALTO Address Type Registry" (see Section 14.4 of [RFC7285]). An ALTO server SHOULD NOT claim "ipv4" and "ipv6" in this field explicitly, because they are supported by default. If not present, this field MUST be interpreted as if it is an empty array, i.e., the ALTO server only supports the default "ipv4" and "ipv6" address types.

flow-spec-announce: It MUST NOT be present if "flow-based-filter" is not true. If present, the value is the an array of supported flow-specific announcement types.

5.3.2. Accept Input Parameters

The ReqEndpointCostMap object in Section 4.2.2 of [RFC8189] 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;]
  [ExtEndpointFilter endpoint-flows<1..*>;]
} ReqEndpointCostMap;

object {
  [FlowSpecAnnounceMap flow-spec-announce;]
} ExtEndpointFilter : EndpiontFilter;
```

cost-type, multi-cost-types, testable-cost-types, constraints, or-constraints:
As defined in Section 4.1.2 of [RFC8189].

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

endpoint-flows: Defined as a list of ExtEndpointFilter objects. The ALTO server MUST interpret endpoint pairs appearing in multiple ExtEndpointFilter objects as if they appeared only once. If the capability "flow-spec-announce" is present, the "flow-spec-announce" input parameter can be specified. The interpretation of this field is the same as in Section 5.1.2.

If the AddressType of the source and destination in the same EndpointFilter do not conform the compatibility rule defined in Table 1 of Section 7.1, an ALTO server MUST return an ALTO error response with the error code "E_INVALID_FIELD_VALUE".

An ALTO client MUST specify either "endpoints" or "endpoint-flows", but MUST NOT specify both in the same query.

5.4. Response

This document does not change the format of the response entity. But the ALTO server responds the request with "pid-flows" filter as follows:

The ALTO server MUST include the path costs of pairs in each ExtPIDFilter in the "pid-flows" filter. If the "flow-spec-announce" field is specified in some ExtPIDFilter, the path costs for flows in this ExtPIDFilter SHOULD respond the flow-specific information announced by this field. Especially, if "transmission-type" is specified as "multicast", the ALTO server SHOULD expose all the destination address as a multicast group address, and append the

shared trees to the multicast destination addresses into the response if possible.

5.5. Examples

5.5.1. Information Resource Directory

The following is an example of IRD with relevant resources of the ALTO server. It provides a default network map, a property map of "ane" domain, a filtered cost map and two endpoint cost resources. All of three cost query resources (filtered cost map and endpoint cost resources) support "flow-based-filter". One endpoint cost resource support "flow-spec-announce" and the compound query extension defined in I-D.ietf-alto-path-vector.

Examples followed this section use the same IRD in this document.

```
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: [TBD]
Content-Type: application/alto-directory+json

{
  "meta" : {
    "default-alto-network-map" : "my-default-network-map",
    "cost-types" : {
      "num-hopcount" : {
        "cost-mode" : "numerial",
        "cost-metric" : "hopcount"},
      "num-routingcost" : {
        "cost-mode" : "numerial",
        "cost-metric" : "routingcost"},
      "ord-routingcost" : {
        "cost-mode" : "ordinal",
        "cost-metric" : "routingcost"},
      "path-vector" : {
        "cost-mode" : "array",
        "cost-metric" : "ane-path"}
    },
    .....
    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"
      },
      "flow-based-cost-map" : {
        "uri" : "http://alto.example.com/costmap/multi/filtered",
        "media-type" : "application/alto-costmap+json",
        "accepts" : "application/alto-costmapfilter+json",
        "uses" : [ "my-default-network-map" ],
        "capabilities" : {
          "max-cost-types" : 2,
          "flow-based-filter" : true,
          "cost-type-names" : [ "num-hopcount",
                                "num-routingcost" ]
        }
      },
      "flow-based-endpoint-cost" : {
        "uri" : "http://alto.example.com/endpointcost/lookup",
        "media-type" : "application/alto-endpointcost+json",
        "accepts" : "application/alto-endpointcostparams+json",
        "capabilities" : {
          "address-types": ["tcp", "udp"],
          "flow-based-filter" : true,
          "cost-type-names" : [ "ord-routingcost",
                                "num-routingcost" ]
        }
      },
      "path-vector-endpoint-cost" : {
        "uri" : "http://alto.example.com/pathvector/lookup",
        "media-type": "multipart/related;
                       type=application/alto-costmap+json",
        "accepts" : "application/alto-endpointcostparams+json",
        "capabilities" : {
          "address-types": ["tcp", "tcp6"],
          "flow-based-filter" : true,
          "flow-spec-announce" : [ "transmission-type" ]
          "cost-type-names" : [ "path-vector" ],
          "ane-property-names": [ "maxresbw" ],
        }
      }
    }
  }
```

5.5.2. Flow-based Filtered Cost Map Example

This example shows how an ALTO client requests a filtered cost map using the "pid-flows" filter. In this case, the ALTO client receives a sparse cost map, which cuts 50% useless cost values from the full mesh.

```
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": [
    { "srcs": ["PID1"], "dsts": ["PID2", "PID3"] },
    { "srcs": ["PID3"], "dsts": ["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": "hopcount"
    }
  },
  "cost-map": {
    "PID1": { "PID2": 6, "PID3": 2 },
    "PID3": { "PID4": 1 }
  }
}
```

5.5.3. Flow-based Endpoint Cost Service Example #1

This example shows how the ALTO client requests endpoint cost using "flow-based-filter" and extended endpoint addresses. In this case, the ALTO client specifies tcp socket address to get more accurate path cost.

```
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": [
    { "srcs": ["ipv4:192.0.2.2"],
      "dsts": ["ipv4:192.0.2.89", "tcp:cdn1.example.com:21"] },
    { "srcs": ["tcp:203.0.113.45:54321"],
      "dsts": ["tcp:cdn1.example.com:21"] }
  ]
}

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

{
  "meta": {
    "cost-type": {
      "cost-mode": "numerical",
      "cost-metric": "routingcost"
    }
  },
  "endpoint-cost-map": {
    "ipv4:192.0.2.2": {
      "ipv4:192.0.2.89": 100,
      "tcp:cdn1.example.com:21": 20
    },
    "tcp:203.0.113.45:54321": {
      "tcp:cdn1.example.com:21": 80
    }
  }
}
```

5.5.4. Flow-based Endpoint Cost Service Example #2

This example shows the integration of the path vector extension and the flow-based query. And in this example, the ALTO client specifies the flow from "tcp6:203.0.113.45:54321" to "tcp6:group1.example.com:21" is multicast. So the ALTO server will expose the destination IP as a multicast group IP, and find the multicast destinations "fe80::40e:9594:da3d:34b" and "fe80::826:daff:feb8:1bb". Then the ALTO server will append the cost for the shared tree into the "endpoint-cost-map".

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

{
  "cost-type": {
    "cost-mode": "array",
    "cost-metric": "ane-path"
  },
  "endpoint-flows": [
    { "srcs": ["ipv4:192.0.2.2"],
      "dsts": ["tcp:192.0.2.89:21",
               "tcp:cdn1.example.com:21"] },
    { "srcs": ["tcp6:203.0.113.45:54321"],
      "dsts": ["tcp6:group1.example.com:21"],
      "flow-spec-announce": {
        "transmission-type": "multicast" } }
  ],
  "ane-property-names": ["maxresbw"]
}

HTTP/1.1 200 OK
Content-Length: [TBD]
Content-Type: multipart/related; boundary=example;
              type=application/alto-endpointcost+json

--example
Resource-Id: ecs
Content-Type: application/alto-endpointcost+json

{
  "meta": {
    "vtags": {
```

```
          "resource-id": "path-vector-endpoint-cost.ecs",
          "tag": "bb6bb72eafe8f9bdc4f335c7ed3b10822a391cef"
        },
        "cost-type": {
          "cost-mode": "array",
          "cost-metric": "ane-path"
        }
      },
      "endpoint-cost-map": {
        "ipv4:192.0.2.2": {
          "tcp:192.0.2.89:21": [ "ane:S1", "ane:D1" ],
          "tcp:cdn1.example.com:21": [ "ane:S1", "ane:D2", "ane:D3" ]
        },
        "tcp6:203.0.113.45:54321": {
          "tcp6:group1.example.com:21": [ "ane:S2", "ane:D3" ]
        },
        "tcp6:group1.example.com:21": {
          "tcp6:[fe80::40e:9594:da3d:34b]:21": [ "ane:G1" ],
          "tcp6:[fe80::826:daff:feb8:1bb]:21": [ "ane:G2" ],
        }
      }
    }

    --example
    Resource-Id: propmap
    Content-Type: application/alto-propmap+json

    {
      "meta": {
        "dependent-vtags": [
          {
            "resource-id": "path-vector-endpoint-cost.ecs",
            "tag": "bb6bb72eafe8f9bdc4f335c7ed3b10822a391cef"
          }
        ]
      },
      "property-map": {
        "ane:S1": { "maxresbw": 100 },
        "ane:S2": { "maxresbw": 100 },
        "ane:D1": { "maxresbw": 150 },
        "ane:D2": { "maxresbw": 80 },
        "ane:D3": { "maxresbw": 150 },
        "ane:G1": { "maxresbw": 100 },
        "ane:G2": { "maxresbw": 100 }
      }
    }
```

6. Security Considerations

As discussed in Section 15.4 of [RFC7285], an ALTO server or a third party who is able to intercept the flow-based cost query messages MAY store and process the obtained information in order to analyze user behaviors and communication patterns. Since flow-based cost queries MAY potentially provide more accurate information, an ALTO client should be cognizant about the trade-off between redundancy and privacy.

7. IANA Considerations

This document defines new address types to be registered to an existing ALTO registry, and a new registry for their compatible address types.

7.1. ALTO Address Type Registry

This document defines several new address types to be registered to "ALTO Address Type Registry", listed in Table 1.

Identifier	Address Encoding	Prefix Encoding	Mapping to/from IPv4/v6
eth	See Section 3.2.1	None	Mapping to/from IPv4 by [RFC0903] and [RFC0826]; Mapping to/from IPv6 by [RFC3122] and [RFC4861]
domain	See Section 3.2.2	None	Mapping to/from IPv4 by [RFC1034]
domain6	See Section 3.2.2	None	Mapping to/from IPv6 by [RFC3596]
tcp	See Section 3.2.3	None	No mapping
tcp6	See Section 3.2.4	None	No mapping
upd	See Section 3.2.3	None	No mapping
udp6	See Section 3.2.4	None	No mapping

Table 1: ALTO Address Type Registry

7.2. ALTO Address Type Compatibility Registry

This document proposes to create a new registry called "ALTO Address Type Compatibility Registry", whose purpose is stated in Section 3.3.

The compatible address type identifiers of the ones registered in the ALTO Address Type Registry are listed in Table 2.

Identifier	Compatible Identifiers
eth	ipv4, ipv6
domain	eth, ipv4
domain6	eth, ipv6
tcp	eth, ipv4, domain
tcp6	eth, ipv6, domain6
udp	eth, ipv4, domain
udp6	eth, ipv6, domain6

Table 2: ALTO Address Type Compatibility Registry

The entry of an address type identifier SHOULD only include the identifiers registered before it. The compatibility between address types are bidirectional. For example, although "eth" does not register "tcp" as its compatible identifier, an ALTO server MUST recognize them as compatible because "eth" is registered as a compatible identifier of "tcp".

Any new ALTO address type identifier registered after this document MUST register their compatible identifiers in this registry simultaneously.

7.3. ALTO Flow-specific Announcement Registry

This document proposes to create a new registry called "ALTO Flow-specific Announcement Registry", whose purpose is stated in Section 4. An initial registry entry is also documented as shown in Table 3.

Type	Interpretation
transmission-type	See Section 4.2

Table 3: ALTO Flow-specific Announcement Registry

8. References

8.1. Normative References

[EUI48] IEEE, ., "Guidelines for use of a 48-bit Extended Unique Identifier (EUI-48)", 2012.

[EUI64] IEEE, ., "Guidelines for use of a 64-bit Extended Unique Identifier (EUI-64)", 2012.

[I-D.ietf-alto-path-vector] Gao, K., Randriamasy, S., Yang, Y., and J. Zhang, "ALTO Extension: Path Vector", Work in Progress, Internet-Draft, draft-ietf-alto-path-vector-10, 9 March 2020, <http://www.ietf.org/internet-drafts/draft-ietf-alto-path-vector-10.txt>.

[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, <https://www.rfc-editor.org/info/rfc7285>.

[RFC8189] Randriamasy, S., Roome, W., and N. Schwan, "Multi-Cost Application-Layer Traffic Optimization (ALTO)", RFC 8189, DOI 10.17487/RFC8189, October 2017, <https://www.rfc-editor.org/info/rfc8189>.

8.2. Informative References

[RFC0826] Plummer, D., "An Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware", STD 37, RFC 826, DOI 10.17487/RFC0826, November 1982, <https://www.rfc-editor.org/info/rfc826>.

[RFC0903] Finlayson, R., Mann, T., Mogul, J.C., and M. Theimer, "A Reverse Address Resolution Protocol", STD 38, RFC 903, DOI 10.17487/RFC0903, June 1984, <https://www.rfc-editor.org/info/rfc903>.

[RFC1034] Mockapetris, P.V., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

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

DOI 10.17487/RFC2119, March 1997,
<https://www.rfc-editor.org/info/rfc2119>.

[RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, DOI 10.17487/RFC2181, July 1997, <https://www.rfc-editor.org/info/rfc2181>.

[RFC2732] Hinden, R., Carpenter, B., and L. Masinter, "Format for Literal IPv6 Addresses in URL's", RFC 2732, DOI 10.17487/RFC2732, December 1999, <https://www.rfc-editor.org/info/rfc2732>.

[RFC3122] Conta, A., "Extensions to IPv6 Neighbor Discovery for Inverse Discovery Specification", RFC 3122, DOI 10.17487/RFC3122, June 2001, <https://www.rfc-editor.org/info/rfc3122>.

[RFC3596] Thomson, S., Huitema, C., Ksinant, V., and M. Souissi, "DNS Extensions to Support IP Version 6", STD 88, RFC 3596, DOI 10.17487/RFC3596, October 2003, <https://www.rfc-editor.org/info/rfc3596>.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <https://www.rfc-editor.org/info/rfc4861>.

Appendix A. Acknowledgment

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

Appendix B. Change Logs

Note to Editor: Please remove this section prior to publication.

This section records the change logs of the draft updates.

Changes since -06 versions:

* Clarify the type of "flow-spec-announcement" in both the request and the response

* Modify examples to be compatible with the latest path vector document

* Add a new registry for flow-specific announcements

* Fix some typos

Changes since -05 versions:

* Add flow-specific information announcement in the flow-based filter.

* Modify examples and add descriptions to Make them clear.

* Rename the address type "Domain Name" to "Internet Domain Name" to distinguish it with the "Domain Name" in the unified properties draft.

Changes since older versions:

Changes since -04 revision:

* Improve the clarity of the document by explicitly stating the problems.

* Keep only "flow" in the terminology section.

* Move section 6 "Advanced Flow-based Query" out of this document.

* Change "ALTO Address Type Conflicts Registry" to "ALTO Address Type Compatibility Registry".

Since -03 revision:

* Remove some irrelevant content from the draft.

* Improve the description of the new endpoint address type identifier registry. And add a new registry to declare the conflicting address type identifiers.

Since -02 revision:

* Change "EndpointURI" to "AddressType::EndpointAddr" for consistency.

* Replace "Cost Confidence" by "Cost Statistics" for compatibility.

Since -01 revision:

* 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.

Since -00 revision:

* Change the schema of "pid-flows" and "endpoint-flows" fields from pair list to pair mesh list.

Authors' Addresses

Jingxuan Jensen Zhang
China
201804
Shanghai
4800 Caoan Road
Tongji University

Email: jingxuan.n.zhang@gmail.com

Kai Gao
China
610000
Chengdu
No.24 South Section 1, Yihuan Road
Sichuan University

Email: kaigao@scu.edu.cn

Junzhuo Wang

Qiao Xiang
Yale University
51 Prospect Street
New Haven, CT
United States of America

Email: qiao.xiang@cs.yale.edu

Yang Richard Yang
Yale University

51 Prospect Street
New Haven, CT
United States of America

Email: yry@cs.yale.edu