# ALTO Extension: Path Vector

draft-ietf-alto-path-vector-01

Presenter: Dawn Chen

IETF 99
July 20, 2017
Prague

## Overview

- Document goal: address the network graph milestone

- Status at the last IETF
  - Adopted as a Working Group document

- Main updates between IETF98 and IETF99: Finalize 3 remaining design issues

## Recall: Three Main Design Issues and Design Choices Made at IETF 98

- Response
  - Issue 1: What is the information structure of providing path vectors?
  - Issue 2: How to encode the chosen information structure?
- Request
  - Issue 3: What is the query format?

# Issue 1: Information Structure (Problem)

- Fundamentally, path vector response structure consists of two maps
    - to remove redundancy; aka database normalized design should consist of two tables

```
"cost-map":{
   "PID1":  {"PID2": ["ane:L001", "ane:L002"],
     …}
}
```

```
"prop-map":  {
   "ane:L001": {"delay": "10"},
   "ane:L002": {"delay": "30"} ……
}
```

- Strawman: Add "prop-map" in alto-costmap
    - Problem: break existing alto-costmap media type

```
object {
     CostMapData cost-map;
     [PropertyMapData prop-map;]
} InfoResourceCostMap;
```

- Solution Adopted at IETF 98:
    - Send the two maps in two messages:

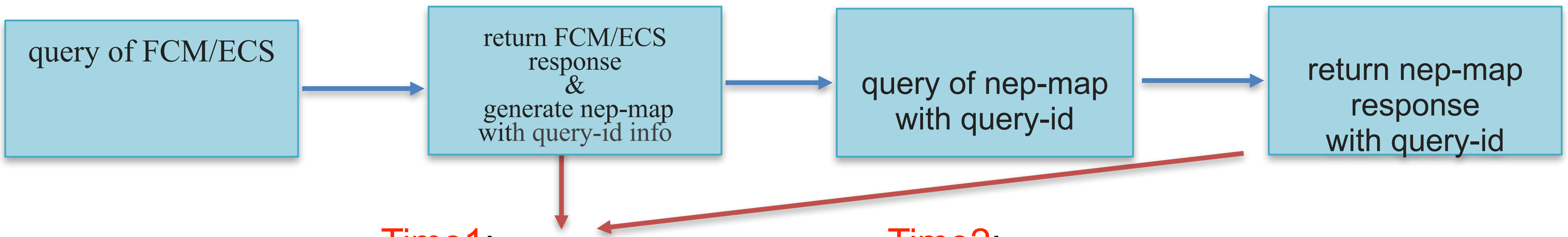
  (1) alto-costmap message and

  (2) alto-propmap message for prop map

```
object {
    CostMapData cost-map;
} InfoResourceCostMap;
```

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

- Possible Problem: Snapshot consistency


query of FCM/ECS
return FCM/ECS response & generate nep-map with query-id info
query of nep-map with query-id
return nep-map response with query-id

Time1:
"ane:L001": {"delay": "10"}

Time2:
"ane:L001": {"delay": "30"}

## Issue 1: Information Structure (Solution)

- -01 solution (Improvements):
  - Keep alto-costmap media type
  - Encode prop map using general alto-propmap media type
  - Two-step query is still applied to support several times of property queries
  - Introduce *MIME multipart/related* [RFC2387] to include both *in a single response*

## Request

```
POST /endpointcostmap/multicost HTTP/1.1
Host: alto.example.com
Accept: multipart/related, application/alto-
costmap+json, application/alto-propmap+json,
application/alto-error+json
Content-Length: [TBD]
Content-Type: application/alto-costmapfilter+json

{
  "multi-cost-types": [
    {  "cost-mode": ”…",
       "cost-metric": ”…"  },
    {  "cost-mode": "numerical",
       "cost-metric": "routingcost" } ],
  "endpoints": {
    "srcs": [ "ipv4:192.0.2.2" ],
    "dsts": [ "ipv4:192.0.2.89",
              "ipv4:203.0.113.45",
              "ipv6:2001:db8::10" ]
  }
}
```

## Response

```
HTTP/1.1 200 OK
Content-Length: [TBD]
Content-Type: multipart/related; boundary=example-2

--example-2
Content-Type: application/alto-endpointcost+json

{
  "meta": {
    "multi-cost-types": [...]
    "vtag": {
      …
      "query-id": "query2"
    }
  },
  "endpoint-cost-map" : …

--example-2
Content-Type: application/alto-propmap+json

{
  "property-map" : …
}

--example-2--
```

# Issue 2: PV Cost Type

**Cost map example:**

```
"cost-map": {
   "PID1": {"PID2": [ "ane:1", "ane:2"]
               "PID3": [ "ane2", "ane3"]
   }
}
```

**Endpoint cost map example:**

```
”endpoint-cost-map": {
   "ipv4:192.168.1.230": {
          ”ipv4:192.168.2.20": [ "ane:L001” ] }
}
```

- Apply the "consistency" principle (i.e., consistency with existing design)
  - Existing design
    - cost mode: numerical, ordinal ⬅ indicate data type of each cost map element: float/int respectively
    - cost metric: routingcost, bw ⬅ indicate semantics
  - Consistent PV response design
    - cost mode indicates each element in the cost map is an array
    - cost metric indicates the semantics of each element is a path consisting of abstract network elements

# PV Cost Type

- Introduce a new cost type, where
  - cost-mode = "array" :

    Indicate each returned cost value is an array

  - cost-metric = "ane-path" :

    Indicate each returned array represents an path consisting of abstract network elements

## Issue 3: Query Format

- Recall, both Filtered Cost Map (FCM) and Endpoint Cost Service (ECS) support only cross product specification of co-flows

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

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

Example:

- Client is interested in resource constraints of two concurrent flows
  - x1: s1 → d1
  - x2: s2 → d2
- But cross product requires
  - x1: s1 → d1
  - x2: s2 → d2
  - x3: s1 → d2
  - x4: s2 → d1

## IETF98: Design Choice: New Query Format to Avoid Cross Product

- Introduce a new field for flows (no new media type), e.g.,

```
object {
   CostType cost-type;
   [JSONString constraints<0..*>;]
   [PIDFilter pids;]
   [PIDFlowFilter pid-flows<1..*>;]
} ReqFilteredCostMap;

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

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

- Comment: acceptable backward compatibility.

# Query Format: -01 Decision

- Move forward with cross product and leave the new co-flow query input in draft-gao-alto-fcs
- Justification
  - Cross product can be less efficient, but can provide the same information as the more specific co-flow spec
  - Theorem: Let $F_1$ and $F_2$ be two sets of flows. $F_1 \subseteq F_2$. Let c(F) be the feasible set returned by PV on bandwidth resource constraints, assuming *non-adaptive flows*. Let $c(F_2)|F_1$ be the projection of $c(F_2)$ with all variables for flows in $F_2 \setminus F_1$ set to 0. Then

$$\boxed{C(F_1) = c(F_2)|F_1}$$

## Protocol Specifications

- VersionTag Extension

```
object {
        ResourceID resource-id;
        JSONString tag;
        [JSONString query-id;]
} VersionTag;
```

- IRDResourceEntry Extension

```
object {
        JSONString uri;
        …
        [ResourceID uses<0..*>;]
        [ResourceID property-map;]
} IRDResourceEntry;
```

- Cost Map/ Endpoint Cost Map Extension
  - **Response**

  1. The "vtag" field MUST be included in the "meta" filed of the response.
  2. The encoding format of each map maintains the same but introduce a new media type multipart/related to encode the multiple resources in a single response.

- Property Map
  - **Accept Input Parameters of IRDResourceEntry**

```
object {
        EntityAddr entities<1..*>
        PropertyName properties<1..*>;
        [JSONString query-id;]
} ReqFilteredPropertyMap;
```

## Next Steps

- Add the wording of the “consistency” principle in the text
- More text to analyze the security of information hiding

## Q & A

Thanks

## Recall: Three Decisions at IETF 98

- **Decision 1**: Define a specific cost type for path vector
  - Cost-mode = "path-vector"
  - Cost-metric = "ane"
- **Decision 2**: A new query format (flow based query format)
- **Decision 3**: Use the reference mode to provide PV network element properties

## Updates

- Item1: Redefine the semantics of the new cost type
  - Cost-mode = "ane-path"
  - Cost-metric = "array"
- Items2: Remove the extension of flow query format extension
- Item3: Support Multi-resources in a single response

# Recall: 3 Main Design Issues and Design Choices Made at IETF 98

- Response
  - Issue 1: What is the information structure of providing path vectors?
  - Issue 2: How to encode information structure?
- Request
  - Issue 3: What is the query format?

| Issue 1 | Issue 2 | Issue 3 |
|---|---|---|
| Define a specific cost type for path vector | Inline | Native FCM/ECS query |
| A unifying scheme supporting multi-cost, and cost calendar | Reference | New flow query format |

- *Our focus from IETF98-99: clean realization of the decisions.*

## Request

```
POST /endpointcostmap/multicost HTTP/1.1
Host: alto.example.com
Accept: multipart/related, application/alto-
costmap+json, application/alto-propmap+json,
application/alto-error+json
Content-Length: [TBD]
Content-Type: application/alto-costmapfilter+json

{
  "multi-cost-types": [
    { "cost-mode": "array",
      "cost-metric": "ane-path" },
    { "cost-mode": "numerical",
      "cost-metric": "routingcost" } ],
  "endpoints": {
    "srcs": [ "ipv4:192.0.2.2" ],
    "dsts": [ "ipv4:192.0.2.89",
              "ipv4:203.0.113.45",
              "ipv6:2001:db8::10" ]
  }
}
```

## Response

```
HTTP/1.1 200 OK
Content-Length: [TBD]
Content-Type: multipart/related; boundary=example-2

--example-2
Content-Type: application/alto-endpointcost+json

{
  "meta": {
    "multi-cost-types": [...]
    "vtag": {
      …
      "query-id": "query2"
    }
  },
  "endpoint-cost-map" : …

--example-2
Content-Type: application/alto-propmap+json

{
  "property-map" : …
}

--example-2--
```