ALTO Extension: General Cost Types for Path Vector and Others

draft-yang-alto-general-cost-type-00
draft-yang-alto-pv-topology-01

Presenter: Y. Richard Yang

October 27, 2015 @ ALTO Interim Meeting

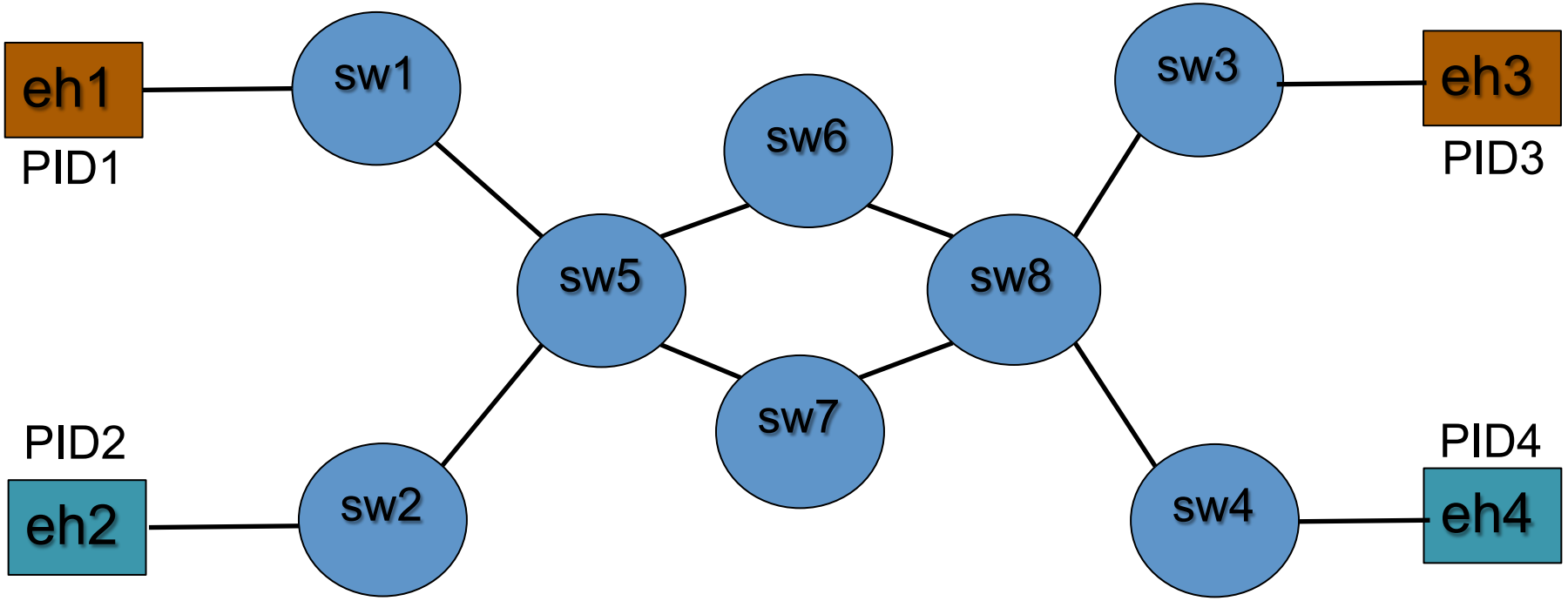
Motivation: Path Vector as Path (Cost) Property

- To make progress on the ALTO path-vector extension, we need to define its encoding
  - In –pv-topology-01, we used a new cost-mode called "path-vector", but this design does not look convincing yet

```
HTTP/1.1 200 OK
…
application/alto-costmap+json

{
   "meta" : {
      "dependent-vtags" : […],
      "cost-type" : {~~"cost-metric": "bw",~~ "cost-mode"  : "path-vector"  }
   },
   "cost-map" : {
      "PID1": { "PID1":[], "PID2":["ne56", "ne67"], "PID3":[], "PID4":["ne57"]},
      "PID2": { "PID1":["ne75"], "PID2":[], "PID3":["ne75"], "PID4":[]}, …
   }
}
```

Motivation: ECMP as Path (Cost) Property



- ECMP for eh1 -> eh3, single path through sw6 for eh2 -> eh4
  - PID1 -> PID3: [{“ne”: “sw5-6”, “w”: 0.5},
    {“ne”: “sw6-8”, “w”: 0.5},
    {“ne”: “sw5-7”, “w”: 0.5},
    {“ne”: “sw7-8”, “w”: 0.5}]
  - PID2 -> PID4: : [{“ne”: “sw5-6”, “w”: 1},
    {“ne”: “sw6-8”, “w”: 1}]

```
object {
   JSONNumber w;      // flow weight
   JSONString  ne;    // network element
} FlowElement;

object {
      cost-map.DstCosts.JSONValue
            -> FlowElement<0,*>;
      meta.cost-mode = ”flow”;
} InfoResourcePVCostMap : InfoResourceCostMap;
```

Summary of Issue

- Authors of [RFC7285] anticipated that Cost may need to be generic and used JSONValue

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

  object-map {
    PIDName -> DstCosts;
  } CostMapData;

  object-map {
    PIDName -> JSONValue;
  } DstCosts;
```

```
object {
    EndpointCostMapData endpoint-cost-map;
  } InfoResourceEndpointCostMap
      : ResponseEntityBase;

  object-map {
    TypedEndpointAddr -> EndpointDstCosts;
  } EndpointCostMapData;

  object-map {
    TypedEndpointAddr -> JSONValue;
  } EndpointDstCosts;
```

- To allow a client to correctly parse JSONValue, ALTO protocol must indicate its type: CostMap and EndpointCostMap are polymorphic (generic) types that need type indicator:
  - CostMap<T>
  - EndpointCostMap<T>

One Interpretation of Current Design

- The “cost-mode” field of the “cost-type” field of “meta” is the type indicator

```
object {
      CostMetric  cost-metric;
      CostMode    cost-mode;
      [JSONString description;]
} CostType;
```

- Hence, one interpretation of current design is CostMap<cost-mode>, i.e., CostMap<numerical> and CostMap<ordinal>
  - “numerical” are floating point numbers {6.1.2.1}
  - “ordinal” are “non-negative” integers {6.1.2.2}

- “cost-metric”: indicates the semantics (routingcost, bw, …)
  - CostMap<numerical> routingcost, bw
  - CostMap<ordinal> routingcost, bw
  - EndpointCostMap<numerical> routingcost, bw
  - EndpointCostMap<ordinal> routingcost, bw

Consistent Generalization of Current Design

- For path vector, the “cost-mode” field is a new generic, reusable data type, e.g., string-vector (or vector<string>), that is similar to numerical, ordinal, so that the data type can be reused by others

```
HTTP/1.1 200 OK
…
application/alto-costmap+json

{
  "meta" : {
     "dependent-vtags" : […],
     "cost-type" : {“cost-metric”: “path-elements”, "cost-mode"  : ”string-vector”  }
  },
  "cost-map" : {
     "PID1": { "PID1":[], "PID2":["ne56", "ne67"], "PID3":[], "PID4":["ne57”]},
     "PID2": { "PID1":["ne75"], "PID2":[], "PID3":["ne75"], "PID4":[]}, …
  }
}
```

One More Example

- How do we allow statistics on a cost metric (e.g., routingcost)

```
HTTP/1.1 200 OK
…
application/alto-costmap+json

{
   "meta" : {
      "dependent-vtags" : […],
      "cost-type" : {“cost-metric”: “latency-stat”, "cost-mode"  : ”basic-stat-object”  }
   },
   "cost-map" : {
      "PID1": {
            "PID1”:{“min”; 1, “max”: 2, “avg”: 1.5} ,
            "PID2”:{“min”; 2, “max”: 5, “avg”: 2.5} ,
            …
      ...
   }
}
```

Summary and Remaining Issue

- We interpret current design of CostMap and EndpointCostMap as polymorphic types parameterized by cost-mode
  - CostMap<cost-mode> EndpointCostMap<cost-mode>
- We plan to generalize cost-mode to a broader set of data types (e.g., string-vector, numerical-vector, basic-stat-object)
- Remaining issue: How to specify the schema (i.e., cost-mode)
  - Use an existing data model language, e.g., YANG, JSON Schema
  - Develop one for ALTO

Backup Slides

Examples of CostMap, EndpointCostMap

```
POST /costmap/filtered HTTP/1.1
Host: custom.alto.example.com
Content-Type: application/alto-costmapfilter+json
Content-Length: 181
Accept: application/alto-costmap+json,application/alto-
error+json

{
  "cost-type" : {"cost-mode": "numerical",
                 "cost-metric": "routingcost”
   },
   "pids" : {
     "srcs" : [ "PID1" ],
     "dsts" : [ "PID1", "PID2", "PID3" ]
    }
}

HTTP/1.1 200 OK
Content-Length: 341
Content-Type: application/alto-costmap+json

{
  "meta" : {
    "dependent-vtags" : [
       {"resource-id": "my-default-network-map",
        "tag": "75ed013b3cb58f896e83958258ce670f"
       }
     ],
    "cost-type": {"cost-mode" : "numerical",
                  "cost-metric" : "routingcost”}
   },
   "cost-map" : {
     "PID1": { "PID1": 0,  "PID2": 1,  "PID3": 2 }
   }
}
```

```
POST /endpointcost/lookup HTTP/1.1
Host: alto.example.com
Content-Length: 248
Content-Type: application/alto-endpointcostparams+json
Accept: application/alto-endpointcost+json,application/alto-error+json

{
  "cost-type": {"cost-mode" : "ordinal",
              "cost-metric" : "routingcost"},
  "endpoints" : {
    "srcs": [ "ipv4:192.0.2.2" ],
    "dsts": [
      "ipv4:192.0.2.89”, "ipv4:198.51.100.34”, "ipv4:203.0.113.45"
    ]}
}

HTTP/1.1 200 OK
Content-Length: 274
Content-Type: application/alto-endpointcost+json

{
  "meta" : {
    "cost-type": {"cost-mode" : "ordinal",
                  "cost-metric" : "routingcost"
    } },
  "endpoint-cost-map" : {
    "ipv4:192.0.2.2": {
      "ipv4:192.0.2.89"    : 1, "ipv4:198.51.100.34" : 2,
      "ipv4:203.0.113.45"  : 3
    }}
}
```