Flow-based Cost Query

`draft-gao-alto-fcs-04`

Kai Gao[1] J. Jensen Zhang[2]
Qiao Xiang[3] Y. Richard Yang[3]

[1] Tsinghua University [2] Tongji University [3] Yale University

Presenter: Jensen

Dec 18, 2017@IETF 100 Interim

Updates: Overview

- Many updates from -03 (July 03, 2017, IETF 99) to -04 (Dec 13, 2017, IETF 100 Interim)
  - Move cost value extension away from the draft
  - Clean up the address type registry
    - Distinguish term " Protocol" with "AddressType"
    - Add the "Address Type Conflict Registry"
    - Remove application-layer protocol
  - Add a new field "or-required" in FCS capabilities

Requirements on Flow-based Query

General Requirements on ALTO for the Unified Interface (recall IETF99):

- **More flexible input**: Target of **FCS**
- **More flexible output**: Target of Path Vector, Unified Property, Multi-Cost (RFC8189), Cost Calendar

Requirements on Query Input:

- **#1** More flexible shape of query space
- **#2** More expressive encoding of query entry

Basic Proposal of FCS:

- Arbitrary end-to-end query
- Expressive endpoint address
- Extensible flow description and arbitrary flow query

Flexible Shape of Query Space

- Different flexibilities of the query space

Lower Flexibility → Higher Flexibility

Full Mesh Src-Dst Pairs

srcs → dsts

$$\begin{tikzcd}
\text{addr1} \arrow[r] \arrow[rd] \arrow[rdd] & \text{addr3} \\
\text{addr2} \arrow[ru] \arrow[r] \arrow[rd] & \text{addr4} \\
 & \text{addr5}
\end{tikzcd}$$

Partial Mesh Src-Dst Pairs

srcs → dsts

$$\begin{tikzcd}
\text{addr1} \arrow[r] \arrow[rd] & \text{addr3} \\
\text{addr2} \arrow[ru] \arrow[rd] & \text{addr4} \\
 & \text{addr5}
\end{tikzcd}$$

Extensible Header Space

id	header-fields			
	k1	k2	k3	…
f1	v11	*	v13	
f2	v21	v22	*	
f3	v31	v32	v33	

Better Compatibility ← Worse Compatibility

Smaller Request Size ← Larger Request Size

Flexible Shape of Query Space

- Full Mesh Src-Dst Pairs (Base ALTO Protocol)

  -
    ```
    {"srcs": [addr1, addr2]
     "dsts": [addr3, addr4, addr5]}
    ```
- Partial Mesh Src-Dst Pairs (Section 5 of FCS)
  - Advantage:
    - The response can be **compatible** with the base ALTO protocol
    - The size of request **can be reduced** by using multiple smaller full meshes
  - Drawback: Non-endpoint attributes cannot be supported

  -
    ```
    [{"srcs": [addr1],
      "dsts": [addr3, addr4]},
     {"srcs": [addr2],
      "dsts": [addr3, addr5]}]
    ```
- Extensible Header Space (Section 6 of FCS)
  - Advantage: non-endpoint attributes can be supported
  - Drawback: The response is **incompatible**; the size of request **cannot be reduced**

  -
    ```
    {"f1": {"ipv4:destination": v11, "ethernet:vlan-id": v13},
     "f2": {"ipv4:destination": v21, "ipv4:source": v22},
     "f3": {"ipv4:destination": v31, "ipv4:source": v32,
            "ethernet:vlan-id": v33}}
    ```

Question: Can we achieve a unified query model?

Expressive Query Entry Encoding

- Expressive Endpoint Address
  - *"An endpoint is an application or host that is capable of communicating (sending and/or receiving messages) on a network."* (RFC7285 Sec 2.1)
  - Encode 5-tuples to endpoint addresses
  - New AddressTypes for ALTO Address Type Registry
    - Use address type identifier to express **protocol semantics**
    - Different address types can use the **same address encoding** with **different semantics** (e.g. "tcp" and "udp")
- Extensible Flow Description
  - ALTO Header Field Registry
    - Current registry is a subset of **OpenFlow match fields**
    - Follow the **TLV dependencies** defined in OpenFlow
    - Allow to register new header fields

The Key Remaining Issue

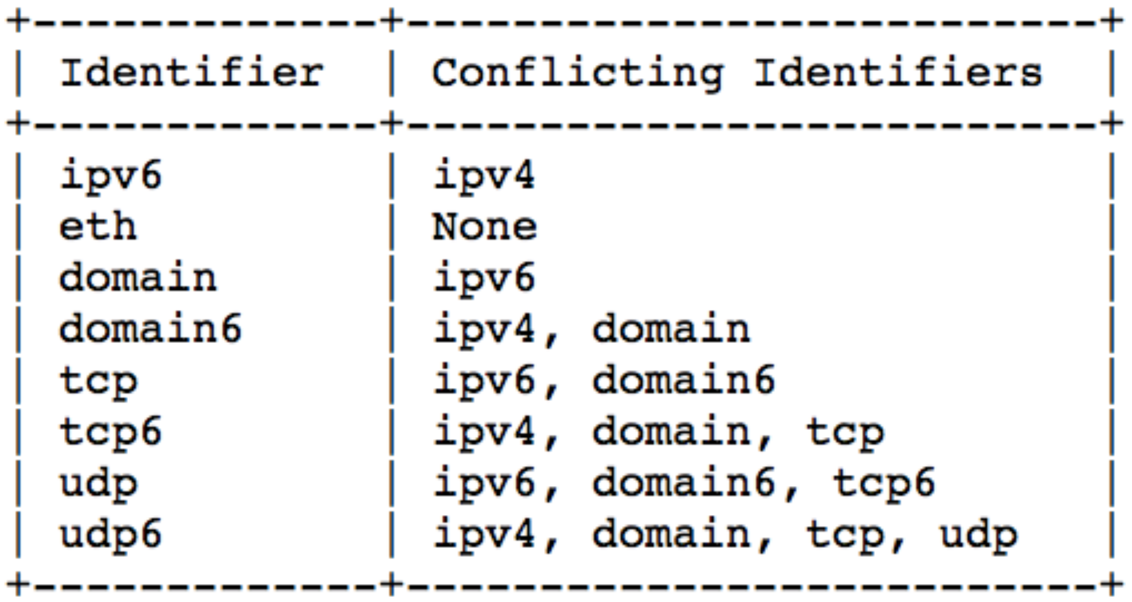
- Validation requirement
  - **Client: I want to query the cost of flow A**
  - **Server: the descriptor of flow A is invalid**
  - *"If the ALTO server does not define a cost value from a source endpoint to a particular destination endpoint, it MAY be omitted from the response"* (RFC7285 Sec 11.5.1.6)
  - General Problem from Client: **Which flows are available from this server?**
- Case1: Endpoint Conflict

  -
    ```
    {"srcs": ["tcp:203.0.113.45:54321"]
     "dsts": ["udp:8.8.8.8:8080"]}
    ```
- Case2: Invalid Flow Descriptor

  -
    ```
    {"flow1": {"ipv4:source": "203.0.113.45",
               "tcp:source": 54321,
               "udp:destination": 8080}}
    ```

Endpoint Conflict

- Declare conflicts of each address type
  - The conflicting identifier list of the future registered address types could be **longer and longer**
  - Some network with special technologies (e.g. NAT) may **avoid some conflicts**



Identifier	Conflicting Identifiers
ipv6	ipv4
eth	None
domain	ipv6
domain6	ipv4, domain
tcp	ipv6, domain6
tcp6	ipv4, domain, tcp
udp	ipv6, domain6, tcp6
udp6	ipv4, domain, tcp, udp

Table 2: ALTO Address Type Conflict Registry

Invalid Flow Descriptor

- Different cases of invalid flow descriptor
  - Missing required header fields
    - Validation: Declare “required” header fields list in “capabilities”
  - Conflicting header fields/values
    - Validation: Apply the TLV format validation defined in OpenFlow
  - Unsupported header fields
    - Validation: Check “required” and “optional” header fields list
- Limitation of a single “required” list
  - Server: Each flow MUST contain “ipv4 source and destination” **OR** “ipv6 source and destination”
  - A single “required” header fields list cannot express such a validator
  - Introduce “or-required”:

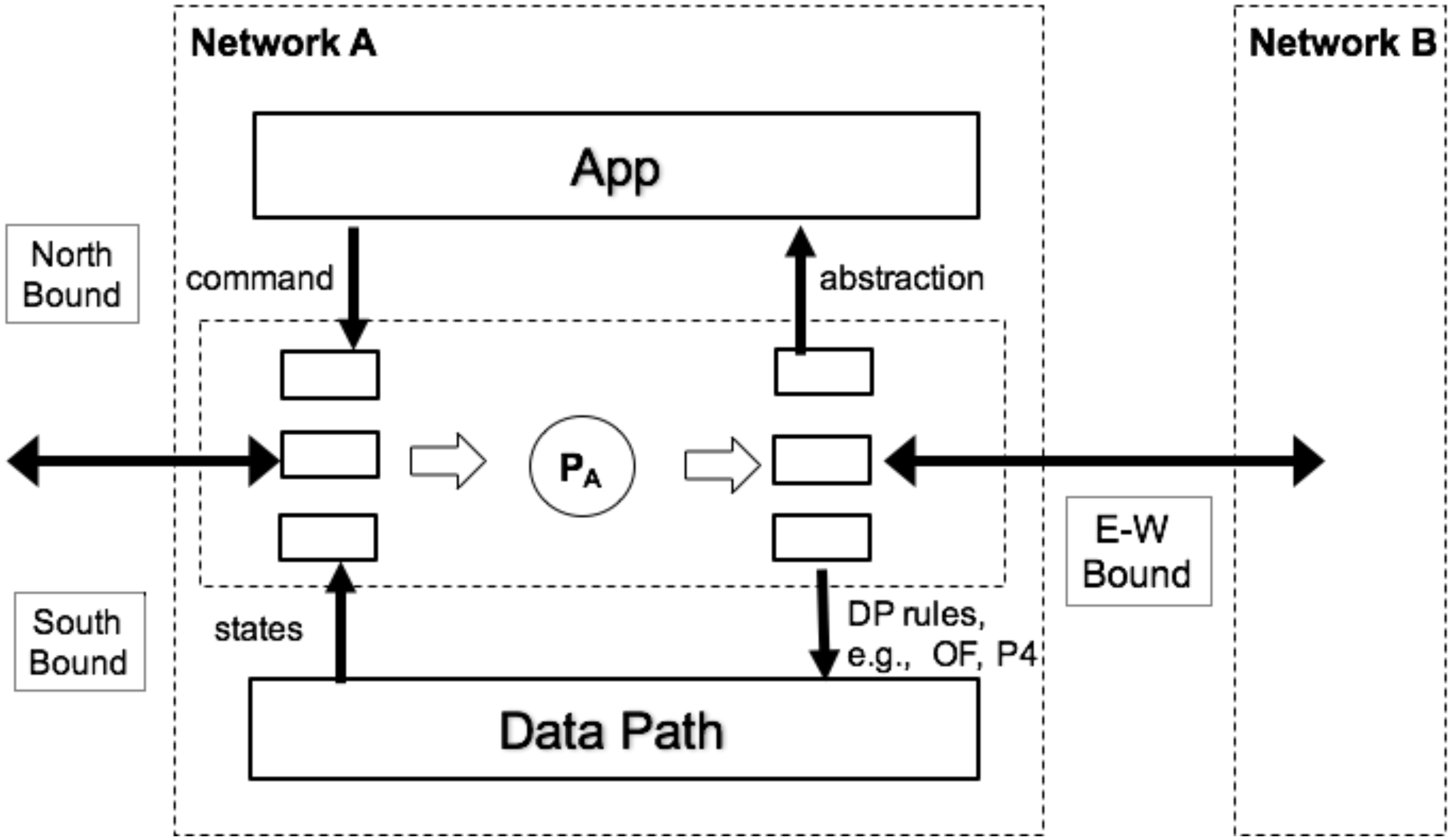
    -
      ```
      {"or-required":
      [["ipv4:source", "ipv4:destination"],
       ["ipv6:source": "ipv6:destination"]]}
      ```

Next Steps

- Move “Address Type Registry” and “Address Type Conflict Registry” to a new draft?
  - Consider other drafts (e.g. cellular addresses) have the same requirement
- Request for reviews/comments
- WG item?

Backup Slides

Architecture: ALTO Providing Unified NorthBound/East-West Views



Big Picture: Unified Model-Views in SDN

ALTO Function: **Network information space → View**

Model-views mapping of different ALTO query services:

- Filtered Network Map Service:
  1-dimensional group region → endpoint set
- Filtered Endpoint Property Service:
  1-dimensional address region → property view
- Filtered Cost Map Service:
  2-dimensional rectangular group region → cost view
- Endpoint Cost Service:
  2-dimensional rectangular address region → cost view

Design Decisions

- #1 Query schema: addr-based vs. spec-based
- #2 Entry encoding: type:addr vs. header-field
- #3 Validation: error or inheritance

Current decisions:

- Co-existence:
  - addr-based + extended type:addr for legacy media-type
  - spec-based + header-field for new media-type
- Return ERROR for all invalid queries

Trade-off between addr-based and spec-based

Extended Legacy Cost Query Schema (address-based schema):

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

object {
  [EndpointFilter  endpoints;]
  [EndpointFilter  endpoint-flows<1..*>;]
} ReqEndpointCostMap  :
MultiCostRequestBase;
```

FCS Query Schema (specification-based schema):

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

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

Trade-off between type:addr and header-field

Compatible Query Entry Descriptor:
AddressType:EndpointAddr

New ALTO Address Type Registry (Section 8.1 of draft-gao-alto-fcs-03)

Valid query entries:

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

Address type conflict:

```
{
  "srcs":  ["ftp:192.168.0.2:5123"],
  "dsts":  ["http:www.google.com"]
}
```

New Query Entry Descriptor:

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

Valid query entry:
(We can define a query entry without any information about the source point.)

```
{
  "ipv4:dst":  "192.168.1.3",
  "tcp:dst":  22,
  "eth:vlan-id":  20
}
```

Remaining Issue: Fault Tolerance

Consider the following query:

```
"endpoint-flows":  [
  {
    "srcs":  ["ipv4:192.0.2.2"],
    "dsts":  ["ipv4:192.0.2.89",
              "http:cdn1.example.com"]
  },  ...  (1)
  {
    "srcs":  ["udp:203.0.113.45:54321"],
    "dsts":  ["http:cdn1.example.com"]
  }   ...  (2)
]
```

Only filter **(2)** conflicts, but the ALTO server won't return the cost of **(1)**.

The ALTO client has to re-send **(1)** in the revised query.

Is it possible to return the response of the valid part with the error message for the invalid part?

Option 1: Augment error message into the [endpiont]cost-map response.

Option 2: Automatic conflict avoidance.

e.g. `"udp"` is a specific type of `"ipv4"/"ipv6"`, so the ALTO server reduce the src endpoint address to `"ipv4:203.0.113.45"` and return the cost between it and `"http:cdn1.example.com"`.