Flow-based Cost Query

draft-gao-alto-fcs-05

Presenter: Jensen Zhang

Mar 19, 2018@IETF 101
Updates: Overview

- Many updates from -04 (Dec 13, 2017, IETF 100 Interim) to -05 (Mar 5, 2017, IETF 101)
  - Improve the clarity of the document by explicitly stating the problems.
  - Make terms used in this document clear.
  - Move Section 6 “Advanced Flow-based Query” out of this document.
  - Change “ALTO Address Type Conflicts Registry” to “ALTO Address Type Compatibility Registry”.
Review Flow-based Query Design

General Requirements on ALTO for the Unified Interface:
- **More flexible input**: Target of FCS
- **More flexible output**: Target of Path Vector, Unified Property, Multi-Cost (RFC8189), Cost Calendar

Requirements on the Input Flexibility:
- #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
Remaining Issues

• Q1: How to achieve a unified query model?
  – We have two design options for the query model:
    • Partial mesh src-dst pairs
    • Extensible header space set

• Q2: How to resolve the flow attribute conflicts?
  – A flow definition may be invalid: A TCP socket source address cannot establish a valid connection with a UDP socket destination address.
  – Allow the server to notify this invalidity to the client as early as possible.
## Q1: Unified Query Model

<table>
<thead>
<tr>
<th>Design Option</th>
<th>Partial Mesh Src-Dst Pairs (Current Option)</th>
<th>Extensible Header Space Set (Another Option)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example</td>
<td><code>[{&quot;srcs&quot;: [addr1], &quot;dsts&quot;: [addr3, addr4]}, {&quot;srcs&quot;: [addr2], &quot;dsts&quot;: [addr3, addr5]}]</code></td>
<td><code>{&quot;f1&quot;: {&quot;ipv4:destination&quot;: v1, &quot;ethernet:vlan-id&quot;: v2}, &quot;f2&quot;: {&quot;ipv4:destination&quot;: v3, &quot;ipv4:source&quot;: v4}, &quot;f3&quot;: {&quot;ipv4:destination&quot;: v5, &quot;ipv4:source&quot;: v6, &quot;ethernet:vlan-id&quot;: v7}}</code></td>
</tr>
<tr>
<td>Compatibility</td>
<td>Response can be compatible</td>
<td>Incompatible</td>
</tr>
<tr>
<td>Request Size</td>
<td>Can be reduced</td>
<td>Cannot be reduced</td>
</tr>
<tr>
<td>Extensibility</td>
<td>Can introduce new endpoint address types</td>
<td>Can introduce new flow attributes</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Cannot request non-endpoint flow attributes</td>
<td>Can support arbitrary flow attributes</td>
</tr>
<tr>
<td>Complexity</td>
<td>Validation is simple (Only need to check source and destination)</td>
<td>Validation is complex (Need to check every shown attributes)</td>
</tr>
</tbody>
</table>

**Comparison between two design options**
# New Registered Address Types

<table>
<thead>
<tr>
<th>Address Type</th>
<th>Encoding</th>
<th>Semantics</th>
<th>Potential Use Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>eth</td>
<td>MAC Address (EUI-48 or EUI-64)</td>
<td>The ethernet address</td>
<td>Layer2 flows between inter DCNs</td>
</tr>
<tr>
<td>domain</td>
<td>Domain Name (RFC2181)</td>
<td>Can be resolved by an A record</td>
<td>CDN</td>
</tr>
<tr>
<td>domain6</td>
<td>Domain Name (RFC2181)</td>
<td>Can be resolved by an AAAA record</td>
<td>CDN</td>
</tr>
<tr>
<td>tcp</td>
<td>IPv4 Socket Address</td>
<td>The client/server address of a tcp socket with an IPv4 address</td>
<td>Flow-level scheduling</td>
</tr>
<tr>
<td>udp</td>
<td>IPv4 Socket Address</td>
<td>The client/server address of a udp socket with an IPv4 address</td>
<td></td>
</tr>
<tr>
<td>tcp6</td>
<td>IPv6 Socket Address</td>
<td>The client/server address of a tcp socket with an IPv6 address</td>
<td></td>
</tr>
<tr>
<td>udp6</td>
<td>IPv6 Socket Address</td>
<td>The client/server address of a udp socket with an IPv6 address</td>
<td></td>
</tr>
</tbody>
</table>
Q2: Flow Attribute Conflicts

- **Original Design:**
  - Declare *conflicts* of new address type with each existing address types.
  - For example: tcp and udp
  - Some network with special technologies (e.g. NAT) may avoid some conflicts. So a server can declare the capability disagree with the conflicts defined in the registry.

- **Key observation:** Most of address types conflict with others.

- **Current Design:**
  - Declare *compatibility* instead of conflicts.
  - If the address type combination of a src-dst pair is not defined in the compatibility registry, it SHOULD be regarded as invalid.
  - A server can extend compatible address type combinations into its own capability.
Next Steps

- Request for reviews/comments
- WG item?
Backup Slides
Flexible Shape of Query Space

- Different flexibilities of the query space

<table>
<thead>
<tr>
<th>Lower Flexibility</th>
<th>Higher Flexibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Mesh</td>
<td>Extensible Header Space</td>
</tr>
<tr>
<td>Src-Dst Pairs</td>
<td></td>
</tr>
<tr>
<td>srcs</td>
<td>srcs</td>
</tr>
<tr>
<td>dsts</td>
<td>dsts</td>
</tr>
<tr>
<td>addr1 &lt;- addr2 -&gt; addr3 &lt;- addr4 -&gt; addr5</td>
<td>addr1 &lt;- addr2 -&gt; addr4 -&gt; addr5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>header-fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>k1</td>
<td>k2</td>
</tr>
<tr>
<td>k3</td>
<td>*</td>
</tr>
<tr>
<td>f1</td>
<td>v11</td>
</tr>
<tr>
<td>v12</td>
<td>v13</td>
</tr>
<tr>
<td>f2</td>
<td>v21, v22</td>
</tr>
<tr>
<td>f3</td>
<td>v31, v32, v33</td>
</tr>
</tbody>
</table>

Better Compatibility
Smaller Request Size

Worse Compatibility
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

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Conflicting Identifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipv6</td>
<td>ipv4</td>
</tr>
<tr>
<td>eth</td>
<td>None</td>
</tr>
<tr>
<td>domain</td>
<td>ipv6</td>
</tr>
<tr>
<td>domain6</td>
<td>ipv4, domain</td>
</tr>
<tr>
<td>tcp</td>
<td>ipv6, domain6</td>
</tr>
<tr>
<td>tcp6</td>
<td>ipv4, domain, tcp</td>
</tr>
<tr>
<td>udp</td>
<td>ipv6, domain6, tcp6</td>
</tr>
<tr>
<td>udp6</td>
<td>ipv4, domain, tcp, udp</td>
</tr>
</tbody>
</table>

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?