ALTO WG
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2017

K. Gao
Tsinghua University
J. Zhang
Tongji University
Y. Yang
Yale University

July 8, 2016

ALTO Flow Cost Service
draft-gao-alto-fcs-00.txt

Abstract

OpenFlow [openflow] is the current standard southbound protocol for Software-Defined Networking. In this document, we define a new service, namely the Flow Cost Service, for clients in a OpenFlow-enabled network to query the network information.

Requirements 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 RFC 2119 [RFC2119].

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 http://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 January 9, 2017.

Copyright Notice

Table of Contents

1. Introduction

With the emerging technologies in the data plane, where multiple header fields can be used to determine the forwarding path, networks are moving to more flexible routing mechanism beyond the simple destination-based routing. As a consequence, the endpoint cost service (ECS), which depends on only source and destination IP addresses as currently defined, is no longer sufficient to provide accurate cost information.

In this document, we consider the extension of ALTO service which provides the cost service, for networks using flow-based routing such as Software-Defined Networks using OpenFlow switches. The flow-based routing, in general, provides a more fine-grained control over the packets than destination-based routing. Consider those packets from a specific source to a specific destination. With destination-based routing, these packets will always use the same path, and hence ECS can provide the same routing cost. Flow-based routing, however, can partition these packets into multiple subsets (flows), where different subsets (flows) can go through different paths, and hence have different routing cost values. For example, large science data flows may go through a DMZ path with low cost, and other traffic may need to go through more security checks, with higher costs. Although one may still use ECS, which may provide an aggregated cost (e.g., average), the result can be inaccurate and misleading.

To satisfy the growing demand of obtaining accurate costs in a network using flow-based routing, a new ALTO service named the flow cost service (FCS) is defined.

2. Basic Data Types

The flow cost service introduces some new basic data types, as defined below.

2.1. Flow ID

A flow ID has the same format as a PIDName, as defined in [RFC7285] Section 10.1 [1]. It is used to uniquely identify a flow in a flow cost service request.

2.2. Typed Header Field

A typed header field represents a particular field in a network protocol that can be obtained at the application layer. It is represented by the protocol name and the field name, concatenated by the colon (':', U+003A). The typed header fields are case insensitive.

For example, "ipv4:source" and "IPv4:source" both represent the source address field used in IPv4 and "tcp:destination" represents the destination port for a TCP connection.

See Table 2 for a list of proposed typed header fields.

2.3. Cost Confidence

A cost confidence is defined as a JSON integer within the range of [0, 100]. It represents the ALTO servers' estimation on the accuracy of the returned costs. The larger the cost confidence is, the more accurate the path cost SHOULD be. If the cost value is very accurate, for example, a unique path can be identified for a flow with the provided information, the ALTO server SHOULD provide a cost confidence of 100.

The cost confidence CAN be used as an evidence of ambiguous paths, which is often associated with insufficient information in a query. If the ALTO clients find the associated cost confidence value is low, it can narrow down the flow header space in the query by adding optional fields or use IP addresses instead of prefixes.

The cost confidence value can be computed in several ways. For example, ALTO servers MAY use the following formula for some cost metrics:

```
    c = 100 * (1 - |deviation / mean|)

                  0        if c <= 0
    confidence =
                  round(c) if c > 0
```

where mean and deviation are computed from the cost values of all possible paths.

3. Flow Cost Service

A flow cost service provides information about costs for each individual flows specified in the requests.

3.1. Media Type

The media type of the flow cost service is "application/alto-flowcost+json".

3.2. HTTP Method

The flow cost service is requested using the HTTP POST method.

3.3. Accept Input Parameters

The input parameters of the flow cost service MUST be encoded as a JSON object of type FlowCostRequest in the body of an HTTP POST

request. The media type of the request MUST be "application/alto-flowcostparams+json".

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

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

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

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

flows: A map of flow filters for which path costs are to be returned. Each flow filter is identified by a unique FlowId, as defined in Section 2.1. The value types of a field is protocol-specific, see Table 3 for the value types associated with typed header fields in Table 2.

cost-type: The same as defined in [I-D.ietf-alto-multi-cost] Section 4.2.2 [2].

multi-cost-types: The same as defined in [I-D.ietf-alto-multi-cost] Section 4.2.2 [3].

testable-cost-types, constraints, or-constraints: The same as defined in [I-D.ietf-alto-multi-cost] Section 4.2.2 [4].

3.4. Capabilities

The capabilities of the flow cost service is a JSON object of type FlowCostCapabilities:

```
    object {
      TypedHeaderField  required<1..*>;
      [TypedHeaderField optional<1..*>;]
    } FlowCostCapabilities :  FilteredCostMapCapabilities;
```

with fields:

required: A list of required typed header fields. These fields are essential to find the path cost for a given flow and MUST be provided in a flow filter.

optional: A list of optional typed header fields. The ALTO server MAY leverage the values of the optional fields to find more accurate costs.

3.5. Response

The "meta" field of a flow cost response MUST contain the same cost type information as defined in [I-D.ietf-alto-multi-cost] Section 4.2.3 [5].

The data component of a flow cost service is named "flow-cost-map", which is a JSON object of type FlowCostMap:

```
object {
  FlowCostMap     flow-cost-map;
  [FlowCostMap    flow-cost-confidences;]
} FlowCostResponse : ResponseEntityBase;

object-map {
  FlowId -> JSONValue;
} FlowCostMap;
```

flow-cost-map: A dictionary map with each key (flow ID) representing a flow specified in the request. For each flow, the cost MUST follow the format defined in [I-D.ietf-alto-multi-cost] Section 4.2.3 [6].

flow-cost-confidences: A dictionary map with each key (flow ID) representing a flow specified in the request. For a single cost, the cost confidence for each flow MUST follow the specification in Section 2.3. If the query is using multiple costs where the costs are returned as a JSONArray, the cost confidence MUST also be a JSONArray where each element represents the cost confidence value computed for the corresponding cost type.

3.5.1. Ambiguous Paths

Since new forwarding abstractions support fine-grained routing, for example, OpenFlow 1.5 [OF15] has defined 38 header match fields, it is possible that the ALTO server cannot determine the path using the provided header fields. The computation for costs with ambiguous paths is implementation-specific, the servers can choose to return an

integrated result of all possible paths, or simply use the cost of a random path. The ALTO servers SHOULD provide cost confidences to justify the accuracy of the provided cost values.

The ALTO server SHOULD be able to determine a unique path when all the optional typed header fields are provided without masks for a flow, however, the client SHOULD NOT assume this always holds.

3.6. Errors

The ALTO servers can provide more information to the clients when requests have errors. The FlowCostErrorMap below can provide basic information about two most common errors for the flow cost service. The ALTO servers MAY include it as the data component of an ALTO error response. If multiple errors are identified, the ALTO server MUST return exactly one error code according to [RFC7285] Section 8.5.2 [7].

```
object-map {
  FlowId -> FlowCostError;
} FlowCostErrorMap;

object {
  [TypedHeaderField conflicts<2..*>;]
  [TypedHeadreField missing<2..*>;]
  [TypedHeaderField unsupported<1..*>;]
} FlowFilterError;
```

conflicts: A list of conflicting typed header fields. See Section 3.6.1 for details.

missing: A list of missing typed header fields. See Section 3.6.2 for details.

unsupported: A list of unsupported typed header fields. See Section 3.6.3 for details.

3.6.1. Conflicts

Some header fields may have conflicts. For example, IPv4 fields and IPv6 fields can never appear in the same packet, nor can TCP and UDP ports. These header fields MUST not be included in the same flow filter, otherwise the ALTO server MUST return an ALTO error response, with the error code "E_INVALID_FIELD_VALUE". As specified in [RFC7285] Section 8.5.2 [8], the ALTO server MAY include the "field" and the "value" in the "meta" field. In this case, the ALTO server MUST use the flow ID as the "field" and the flow filter as the "value". However, the recommended approach is to use the

FlowCostErrorMap, where the server CAN provide the conflicting typed header fields in the "conflicts" field of the FlowFilterError associated with the corresponding flow ID.

3.6.2. Missing Fields

The "E_MISSING_FIELD" error code is originally designed to report the absence of required JSON fields. In the flow cost service, the required typed header fields are implementation-specific and the ALTO servers MUST declare the required fields in the capabilities. If any required header field is missing, the ALTO server MUST return an ALTO error response, with the error code "E_MISSING_FIELD". The ALTO server CAN follow the steps defined in [RFC7285] Section 8.5.2 [9] to indicate the location of the missing field. An alternative approach which is also recommended, is that the server provide the missing typed header fields in the "missing" field of the FlowFilterError associated with the corresponding flow ID.

3.6.3. Unsupported Fields

If a query contains unsupported typed header fields, e.g., those not in the "required" nor the "optional" capabilities, the ALTO server MUST return an ALTO error response, with the error code "E_INVALID_FIELD_VALUE". Like how the conflicting header fields are handled in Section 3.6.1, the ALTO servers CAN report unsupported typed header fields in the "unsupported" field associated with the corresponding flow ID.

3.7. Example

```
POST /flowcost/lookup HTTP/1.1
HOST: alto.example.com
Content-Length: 521
Content-Type: application/alto-flowcostparams+json
Accept: application/alto-flowcost+json,application/alto-error+json

{
  "cost-type": {
    "cost-mode": "numerical",
    "cost-metric": "routingcost"
  },
  "flows": {
    "l3-flow": {
      "ipv4:source": "192.168.1.1",
      "ipv4:destination": "192.168.1.2"
    },
    "optional-l3-flow": {
      "ipv4:source": "192.168.1.1",
      "ipv4:destination": "192.168.1.2"
      "ethernet:source": "12:34:56:78:00:01",
      "ethernet:destination": "12:34:56:78:00:02"
    },
    "l3-flow-aggr": {
      "ipv4:source": "192.168.1.0/24",
      "ipv4:destination": "192.168.2.0/24"
    }
  }
}
```

```
HTTP/1.1 200 OK
Content-Length: 312
Content-Type: application/alto-flowcost+json

{
  "meta": {
    "cost-type": {
      "cost-mode": "numerical",
      "cost-metric": "routingcost"
    },
  },
  "flow-cost-map": {
    "l3-flow": 10,
    "l3-flow-aggr": 50
    "optional-l3-flow": 5,
  },
  "flow-cost-confidences": {
    "l3-flow": 70,
    "l3-flow-aggr": 40,
    "optional-l3-flow": 90
  }
}
```

4. Security Considerations

This document has not conducted its security analysis.

5. IANA Considerations

This document defines two new entries to be registered to application/alto-* media types.

5.1. Media Types

This document registers two media types, listed in Table 1.

Type	Subtype	Specification
application	alto-flowcost+json	Section 3.5
application	alto-flowcostparam+json	Section 3.3

Table 1: ALTO FCS Media Types

Type name: application

Subtype name: This document registers two subtypes, as listed in Table 1.

Required parameters: n/a

Optional parameters: n/a

Encoding considerations: Encoding considerations are identical to those specified for the "applicatoin/json" media type. See [RFC7159].

Security considerations: Security considerations are identical to those specified in [RFC7285] Section 15 [10].

Interoperability considerations: n/a

Published specification: This document is the specification for these media types. See Table 1 for the section documenting each media type.

Applications that use this media type: ALTO servers and ALTO clients with the extension to support the flow cost service, either standalone or embedded within other applications.

Additional information: n/a

Person & email address to contact for further information: See Authors' Addresses.

Intended usage: COMMON

Restrictions on usage: n/a

Author: See Authors' Addresses.

5.2. Header Field

TBD: Create the "ALTO Header Field Name Registry".

6. Acknowledgement

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

7.2. Informative References

[I-D.ietf-alto-multi-cost]
Randriamasy, S., Roome, W., and N. Schwan, "Multi-Cost ALTO", draft-ietf-alto-multi-cost-02 (work in progress), June 2016.

[I-D.wang-alto-ecs-flow]
Shen, X., Zhang, J., Wang, J., and Q. Xiang, "ALTO Extension: Endpoint Cost Service for Flows", draft-wang-alto-ecs-flows-01 (work in progress), April 2016.

[OF15] Foundation, O., "Openflow switch specification v1. 5.0", 2014, <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf>.

[openflow]
McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and J. Turner, "Openflow: enabling innovation in campus networks", 2008.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

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

7.3. URIs

[1] https://tools.ietf.org/html/rfc7285#section-10.1

[2] https://tools.ietf.org/html/draft-ietf-alto-multi-cost-02#4.2.2

[3] https://tools.ietf.org/html/draft-ietf-alto-multi-cost-02#4.2.2

[4] https://tools.ietf.org/html/draft-ietf-alto-multi-cost-02#4.2.2

[5] https://tools.ietf.org/html/draft-ietf-alto-multi-cost-02#4.2.3

[6] https://tools.ietf.org/html/draft-ietf-alto-multi-cost-02#4.2.3

[7] https://tools.ietf.org/html/rfc7285#section-8.5.2

[8] https://tools.ietf.org/html/rfc7285#section-8.5.2

[9] https://tools.ietf.org/html/rfc7285#section-8.5.2

[10] ttps://tools.ietf.org/html/rfc7285#section-15

Appendix A. Tables

Protocol	Field Name	Description
Ethernet	source	The source MAC address
	destination	The destination MAC address
	vlan-id	VLAN-ID from 802.1Q header
IPv4	source	IPv4 source address
	destination	IPv4 destination address
IPv6	source	IPv6 source address
	destination	IPv6 destination address
TCP	source	TCP source port
	destination	TCP destination port
UDP	source	UDP source port
	destination	UDP destination port

Table 2: Protocols and Field Names.

Typed Header Field	Acceptable Value Type
ethernet:source	JSONString as MAC address
ethernet:destination	
ethernet:vlan-id	JSONNumber in the range of [1, 4094]
ipv4:source	JSONString as IPv4 address or IPv4 prefix
ipv4:destination	
ipv6:source	JSONString as IPv6 address or IPv6 prefix
ipv6:destination	
tcp:source	JSONNumber in the range of [0, 65535]
tcp:destination	0 serves as a wildcard value
udp:source	
udp:destination	

Table 3: Value Types for Typed Header Fields

Authors' Addresses

Kai Gao
Tsinghua University
30 Shuangqinglu Street
Beijing 100084
China

Email: gaok12@mails.tsinghua.edu.cn

Jingxuan Jensen Zhang
Tongji University
4800 CaoAn Road
Shanghai 201804
China

Email: jingxuan.n.zhang@gmail.com

Y. Richard Yang
Yale University
51 Prospect St
New Haven CT
USA

Email: yry@cs.yale.edu