Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: March 12, 2016

W. Roome
Bell Laboratories,
Alcatel-Lucent
G. Chen
Huawei Technologies
H. Seidel
BENOCS GmbH

September 9, 2015

Interoperability Testing of the ALTO Protocol
draft-roome-alto-interop-ietf93-01

Abstract

The Application-Layer Traffic Optimization (ALTO) protocol is designed to allow entities with knowledge about the network infrastructure to export such information to applications that need to choose one or more endpoints to connect to among large sets of logically equivalent ones. This document defines a data set that may be used to test the functionality and interoperability of ALTO clients and servers.

Status of this Memo

Copyright Notice

Table of Contents

1. Overview

The Application-Layer Traffic Optimization (ALTO) protocol is designed to allow entities with knowledge about the network infrastructure to export such information to applications that need to choose one or more endpoints to connect to among large sets of logically equivalent ones.

This document defines procedures to test the functionality and interoperability of ALTO clients and servers.

This document is informational and is NOT NORMATIVE on any aspects of the ALTO protocol. The normative behavior of ALTO entities is prescribed in [RFC7285].

Section 2 defines the network maps, cost maps and other data necessary to provision an ALTO server. This ensures that all tested servers will return the same results, so a client may verify that a server is operating correctly. Section 3 defines the required and optional resources for an ALTO server to provide. Section 4 describes the actions expected from a client. Section 5 describes a set of invalid client requests, to verify that a server can respond correctly to client errors.

While every effort has been made to catalogue representative test cases, this document does not attempt to codify every test case that arises in ALTO. The aim of the document is to focus on areas that highlight the key offerings of the ALTO protocol.

2. Server Data

This section defines the data necessary to provision a tested ALTO server in a uniform manner. First it defines a default network map, and associated cost maps for the "routingcost" and "hopcount" metrics. Next it defines an optional alternate network map, along with "routingcost" and "hopcount" costs for that map. Finally it defines a set of optional endpoint properties.

The following examples are fictional and do not depict any actual networks or address prefixes. Because an ALTO Network Map must cover all IP addresses, it is not practical to use only addresses in the ranges reserved for documentation [[RFC5737], [RFC3849]].

Appendix A gives network and cost map data defined in this section formatted in JSON.

2.1. Default Network Map And Cost Maps

Every tested ALTO server MUST provide a default network map with the PIDs defined below:

PID	IP Address Block
mine	100.0.0.0/8
mine1	100.0.0.0/10
mine1a	100.0.1.0/24, 100.0.64.0/24, 100.0.192.0/24
mine2	100.64.0.0/10
mine3	100.128.0.0/10
peer1	128.0.0.0/16, 130.0.0.0/16, 2001:DB8:0000::/33
peer2	129.0.0.0/16, 131.0.0.0/16, 2001:DB8:8000::/33
tran1	132.0.0.0/16
tran2	135.0.0.0/16
default	0.0.0.0/0, ::0/0
loopback	127.0.0.0/8, ::1/128
linklocal	169.254.0.0/16, ff80::/10
private	10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16, fc00::/7

Figure 1: Default Network Map

Each ALTO server MUST provide a cost map for the "routingcost" metric. The following table presents the numerical values for those costs. If a server provides a numerical-mode cost map, it MUST use these values. If a server provides an ordinal-mode cost map, the server may use whatever values it wants, provided the ordinal values preserve the order of the numerical values.

	default	linklocal	loopback	mine	mine1	mine1a	mine2	mine3	peer1	peer2	private	tran1	tran2
default	-	-	-	60.0	63.0	65.0	64.0	65.0	-	-	-	-	-
linklocal	-	0.0	-	-	-	-	-	-	-	-	-	-	-
loopback	-	-	0.0	-	-	-	-	-	-	-	-	-	-
mine	60.0	-	-	0.0	3.0	5.0	4.0	5.0	20.0	25.0	-	35.0	45.0
mine1	63.0	-	-	3.0	0.0	2.0	6.5	8.0	23.0	28.0	-	38.0	48.0
mine1a	65.0	-	-	5.0	2.0	0.0	4.5	10.0	25.0	30.0	-	40.0	50.0
mine2	64.0	-	-	4.0	7.0	9.0	0.0	9.0	24.0	29.0	-	39.0	49.0
mine3	65.0	-	-	5.0	8.0	10.0	9.0	0.0	25.0	30.0	-	40.0	50.0
peer1	-	-	-	20.0	23.0	25.0	24.0	25.0	0.0	-	-	-	-
peer2	-	-	-	25.0	28.0	30.0	29.0	30.0	-	0.0	-	-	-
private	-	-	-	-	-	-	-	-	-	-	0.0	-	-
tran1	-	-	-	35.0	38.0	40.0	39.0	40.0	-	-	-	0.0	-
tran2	-	-	-	45.0	48.0	50.0	49.0	50.0	-	-	-	-	0.0

Figure 2: "routingcost" Numerical Cost Map

Note that this is a partial cost map, in that it does not define a cost for every source and destination PID. Also note that the costs are symmetric except for (mine1,mine2) and (mine1a,mine2).

Each ALTO server MAY provide a cost map for the "hopcount" metric. The following table gives the numerical values. As with "routingcost", a numerical-mode cost map MUST use these values, and an ordinal-mode cost map may use any values consistent with this ordering.

	default	linklocal	loopback	mine	mine1	mine1a	mine2	mine3	peer1	peer2	private	tran1	tran2
default	-	-	-	3	4	5	4	4	-	-	-	-	-
linklocal	-	0	-	-	-	-	-	-	-	-	-	-	-
loopback	-	-	0	-	-	-	-	-	-	-	-	-	-
mine	3	-	-	0	2	3	2	2	3	2	-	4	3
mine1	4	-	-	2	0	2	3	3	4	3	-	5	4
mine1a	5	-	-	3	2	0	2	4	5	4	-	6	5
mine2	4	-	-	2	3	4	0	3	4	3	-	5	4
mine3	4	-	-	2	3	4	3	0	4	3	-	5	4
peer1	-	-	-	3	4	5	4	4	0	-	-	-	-
peer2	-	-	-	2	3	4	3	3	-	0	-	-	-
private	-	-	-	-	-	-	-	-	-	-	0	-	-
tran1	-	-	-	4	5	6	5	5	-	-	-	0	-
tran2	-	-	-	3	4	5	4	4	-	-	-	-	0

Figure 3: "hopcount" Numerical Cost Map

Note that the hopcount costs are symmetric except for (mine1a,mine2).

The figure below depicts a network the default network and cost maps MAY BE derived from:

```
+------------+     +-------------+
|130.0.0.0/16|     |128.0.0.0/16 |
|    peer1   |     |2001:DB8::/33|
+----+-------+     |    peer1    |
     |             +-----+-------+
     |                   |
     |   +---------------+
     |   |                           +------------+
  +--+---+-+       +--------+      |132.0.0.0/16|
  |  R7   +--------+  R8   +------+    tran1   |
  +---+---+        +--------+      +------------+
          |
          |         +-----------+    +------------+
          |         |100.0.0.0/8|    |100.0.0.0/10|
          |         |    mine   |    |    mine1   |
          |         +-----+-----+    +----+-------+
          |               |               |
      +---+---+       +---+---+      +---+---+        +------------+
      |  R6   +-------+  R1   +------+  R2   |   +----+100.0.1.0/24|
      +---+---+       +-+-+-+-+      +---+---+   |    |   mine1a   |
          |             | | |            |       |    +------------+
          |             | | |            |       |
      +---+---+         | | |            |   +---+---+   +-------------+
      |  R11  |         | | |            +---+  R3   +---+100.0.64.0/24|
      +---+---+         | | |                +-+-+---+   |    mine1a   |
          |             | | |                  | |       +-------------+
          |             | | |                  | |
     +----+----+        | | |                  | |    +--------------+
     |0.0.0.0/0|        | | |                  | +----+100.0.192.0/24|
     |::/0     |        | | |            +-------+    |    mine1a    |
     | default |        | | |            |            +--------------+
     +---------+        | | |            |
                        | | |            |
         +-------+      | | |      +---+---+
         |  R5   +------+ | +------+  R4   |
         +---+---+        |        +---+---+
             |            |            |
     +-------+------+     |     +------+------+
     |100.128.0.0/10|     |     |100.64.0.0/10|
     |    mine3     |     |     |    mine2    |
     +--------------+     |     +-------------+
                          |
   +------------+     +---+---+     +------------------+
   |131.0.0.0/16+-----+  R9   +-----+   129.0.0.0/16   |
   |   peer2    |     +---+---+     |2001:DB8:8000::/33|
   +------------+         |         |      peer2       |
                          |         +------------------+
                          |
                      +---+---+     +------------+
                      |  R10  +-----+135.0.0.0/16|
                      +-------+     |    tran2   |
                                    +------------+
```

Figure 4: Default Network Layout

The links between the routers (R1 - R11) have the following metrics. To get the routingcost between two PIDs, sum the link metrics for all paths between the PIDs, and take the lowest sum. Be aware that the link between R3 and R4 is asymmetric.

	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11
R1	0	3	-	4	5	20	-	-	25	-	-
R2	3	0	2	-	-	-	-	-	-	-	-
R3	-	2	0	4.5	-	-	-	-	-	-	-
R4	4	-	10	0	-	-	-	-	-	-	-
R5	5	-	-	-	0	-	-	-	-	-	-
R6	10	-	-	-	-	0	10	-	-	-	50
R7	-	-	-	-	-	10	0	15	-	-	-
R8	-	-	-	-	-	-	15	0	-	-	-
R9	25	-	-	-	-	-	-	-	0	20	-
R10	-	-	-	-	-	-	-	-	20	0	-
R11	-	-	-	-	-	50	-	-	-	-	0

Figure 5: Default Network Link Weights

2.2. Alternate Network Map And Cost Maps

Every tested ALTO server MAY provide an alternate, or secondary, network map with the PIDs defined below:

PID	IP Address Block
dc1	101.0.0.0/16
dc2	102.0.0.0/16
dc3	103.0.0.0/16
dc4	104.0.0.0/16
user1	201.0.0.0/16
user2	202.0.0.0/16
user3	203.0.0.0/16
user4	204.0.0.0/16
default	0.0.0.0/0, ::0/0
loopback	127.0.0.0/8, ::1/128
linklocal	169.254.0.0/16, ff80::/10
private	10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16, fc00::/7

Figure 6: Alternate Network Map

Each ALTO server MAY provide a cost map for the "routingcost" metric for the alternate network map. The following table presents the numerical values for those costs. If a server provides a numerical-mode cost map, it MUST use these values. If a server provides an ordinal-mode cost map, the server may use whatever values it wants, provided the ordinal values preserve the order of the numerical values.

	dc1	dc2	dc3	dc4	default	user1	user2	user3	user4
dc1	0.0	20.0	20.0	20.0	50.0	10.0	15.0	20.0	25.0
dc2	20.0	0.0	20.0	20.0	55.0	15.0	10.0	15.0	20.0
dc3	20.0	20.0	0.0	20.0	55.0	20.0	15.0	10.0	15.0
dc4	20.0	20.0	20.0	0.0	50.0	25.0	20.0	15.0	10.0
default	50.0	55.0	55.0	50.0	-	-	-	-	-
user1	10.0	15.0	20.0	25.0	-	0.0	-	-	-
user2	15.0	10.0	15.0	20.0	-	-	0.0	-	-
user3	20.0	15.0	10.0	15.0	-	-	-	0.0	-
user4	25.0	20.0	15.0	10.0	-	-	-	-	0.0

Figure 7: "routingcost" Numerical Cost Map

Note that this is a partial cost map, in that it does not define a cost for every source and destination PID.

Each ALTO server MAY provide a cost map for the "hopcount" metric. The following table gives the numerical values. As with "routingcost", a numerical-mode cost map MUST use these values, and an ordinal-mode cost map may use any values consistent with this ordering.

	dc1	dc2	dc3	dc4	default	user1	user2	user3	user4
dc1	0	2	2	2	3	2	3	4	5
dc2	2	0	2	2	4	3	2	3	4
dc3	2	2	0	2	4	4	3	2	3
dc4	2	2	2	0	3	5	4	3	2
default	3	4	4	3	-	-	-	-	-
user1	2	3	4	5	-	0	-	-	-
user2	3	2	3	4	-	-	0	-	-
user3	4	3	2	3	-	-	-	0	-
user4	5	4	3	2	-	-	-	-	0

Figure 8: "hopcount" Numerical Cost Map

The figure below depicts a network the alternate network and cost maps MAY BE derived from:

```
   +------------+                              +------------+
   |202.0.0.0/16|                              |203.0.0.0/16|
   |   user2    |                              |   user3    |
   +-----+------+                              +-----+------+
         |                                           |
     +---+---+                                   +---+---+
     |   R6  +-----------------------------------+   R7  |
     +---+---+                                   +---+---+
         |  \    +------------+ +------------+    /  |
         |   \   |102.0.0.0/16| |103.0.0.0/16|   /   |
         |    \  |    dc2     | |    dc3     |  /    |
         |     \ +----+-------+ +------+-----+ /     |
         |      \     |                |      /      |
         |       \    |                |     /       |
         |        \   |                |    /        |
         |        +---+----+      +----+---+         |
         |        |   R2   +------+   R3   |         |
         |        +---+----+      +----+---+         |
         |            |    \      /    |             |
         |            |     \    /     |             |
         |            |      \  /      |             |
         |            |       \/       |             |
         |            |       /\       |             |
         |            |      /  \      |             |
         |            |     /    \     |             |
         |            |    /      \    |             |
         |        +---+----+      +----+---+         |
         |        |   R1   +------+   R4   |         |
         |        +---+----+      +----+---+         |
         |        /   |                |    \        |
         |       /    |                |     \       |
         |      /     |                |      \      |
         |     / +----+-------+ +------+-----+ \     |
         |    /  |101.0.0.0/16| |104.0.0.0/16|  \    |
         |   /   |    dc1     | |    dc4     |   \   |
         |  /    +------------+ +------------+    \  |
     +---+---+                                   +---+---+
     |   R5  +-------+     +-------+     +-------+   R8  |
     +---+---+       +-----+   R9  +-----+       +---+---+
         |                 +---+---+                 |
         |                     |                     |
   +-----+------+         +----+----+          +-----+------+
   |201.0.0.0/16|         |0.0.0.0/0|          |204.0.0.0/16|
   |    user1   |         |   ::/0  |          |    user4   |
   +------------+         | default |          +------------+
                          +---------+
```

Figure 9: Alternate Network Layout

The links between the routers (R1 - R11) have the following metrics. To get the routingcost between two PIDs, sum the link metrics for all paths between the PIDs, and take the lowest sum.

	R1	R2	R3	R4	R5	R6	R7	R8	R9
R1	0	20	20	20	10	-	-	-	-
R2	20	0	20	20	-	20	-	-	-
R3	20	20	0	20	-	-	10	-	-
R4	20	20	20	0	-	-	-	10	-
R5	10	-	-	-	0	5	-	-	40
R6	-	10	-	-	5	0	5	-	-
R7	-	-	10	-	-	5	0	5	-
R8	-	-	-	10	-	-	5	0	40
R9	-	-	-	-	40	-	-	40	0

Figure 10: Alternate Network Link Weights

2.3. Endpoint Properties

An ALTO server may provide the private endpoint property "priv:ietf-type" with the following values for endpoints in the indicated address blocks:

Value	IP Address Block
mine	100.0.0.0/8
peer	128.0.0.0/6, 2001:DB8::/32
transit	132.0.0.0/16, 135.0.0.0/16

Figure 11: Values for "priv:ietf-type" endpoint property

3. Server Resources and Configuration

An ALTO server MUST provide the following resources, as required by [RFC7285]:

o An Information Resource Directory (IRD) which describes all of the server’s resources.

o A Network Map resource for the default network defined above.

o A Cost Map resource for the "routingcost" metric for the default network map. The mode may be either "numerical" or "ordinal". If "numerical", the values MUST be identical to those defined above. If "ordinal", the server can use whatever values it wants, but the ordering MUST be consistent with the ordering of the "numerical" values.

- An Endpoint Property Service for the "pid" property for the default network map.

A server MAY provide whatever additional resources it desires, as long as they are consistent with the network maps, cost maps and endpoint properties defined in Section 2. In particular, a server may provide:

- An additional Network Map resource, using the PIDs and address prefixes for the alternate network map defined above.

- Cost Map resources for the "routingcost" and/or "hopcount" metrics, in either "numerical" or "ordinal" modes, using the values defined above.

- Filtered Network Map resources for either or both network maps.

- Filtered Cost Map resources for any combination of "routingcost" and "hopcount" metrics, in either "numerical" and "ordinal" modes, for either or both network maps. The resources may or may not accept constraint tests.

- Endpoint Cost Service(s) or any combination of "routingcost" and "hopcount" metrics, in either "numerical" and "ordinal" modes. The cost values MUST be consistent with those for the default network map. The resources may or may not accept constraint tests.

- Endpoint Property Service(s) for the custom endpoint properties defined above.

However, a server MUST NOT provide more than the two network maps defined in this document. This restriction simplifies testing, because it allows a client to automatically identify the alternate network map (e.g., any network map which is not the default must be the alternate network). If servers could offer three or more network maps, a client would have to be provisioned with the resource id of the alternate network map.

Note that if a server provides a Network Map resource for the alternate network map, [RFC7285] requires the server to also provide a Cost Map resource for the "routingcost" metric, in either "numerical" or "ordinal" mode, and an Endpoint Property Service for that network map’s "pid" property.

A server MAY structure the IRD however it wants. In particular, a server may

- o Use secondary IRDs which the root IRD references.

- o Use arbitrary resource IDs and cost type names.

- o Use arbitrary URIs, in any recognized URI format.

- o Provide multiple versions of POST-mode resources. For example, if a server provides the secondary network map, it must provide an Endpoint Property Service for the "pid" properties for both maps. A server may provide one EPS for both properties, or a separate EPS for each property.

4. Client Actions

When given the URI for an ALTO server's IRD, an ALTO client should read the IRD, and for each resource that it recognizes, verify that the server returns the correct response. Note that most of the data the server returns is determined by the network maps, cost maps and property values specified in Section 2, and hence can be verified by a client. Some data cannot be determined a priori (e.g., resource id and tag of a network map), but a client can verify their consistency (e.g., a cost map's dependent-vtag field should match the vtag field of the associated network map).

Because costs are floating-point values, instead of using an exact equality test, clients should verify that the actual cost is within 0.001 of the expected cost. Also note that although the "hopcount" values are integers, a server may return integral-valued floating point numbers (e.g., 1.0 rather than 1).

It is expected that not every client will be able recognize and verify every possible resource. However, each client MUST be able to verify the default network map and the associated "routingcost" cost map. In particular, although clients are not required to recognize the alternate network map, if presented with an IRD with two network maps, every client MUST be able to distinguish the default network map, and its associated cost map, from the alternate network map.

Ideally clients should be scripted. That is, when given the URI for a server, an ideal client would verify the server automatically, without further operator intervention. A client should log the resources it tested, and clearly highlight any response the client considered incorrect.

The HTTP GET-mode resources (Network Map and Cost Map) do not require client input, and hence testing is straight-forward: the client sends the appropriate HTTP GET request, and verifies the response.

However, POST-mode resources, such as Filtered Cost Maps and Endpoint Property Services, require client input. The following sections present recommended input parameters for various resources, and clients SHOULD implement as many of these tests as possible. Clients MAY add additional tests, and are encouraged to do so.

4.1. Filtered Network Map Tests

- Empty "pid" array, omitted or empty "address-types" array. The server should return the entire network map.

- Empty "pids" array, "address-types" array containing just "ipv6". The server should return PIDs with ipv6 addresses, and only those PIDs.

- "pids" array with one or more non-existent PID names, such as "not-a-pid". The server should return an empty network map.

- "pids" array with a set of valid PID names (client's choice), plus one or more non-existent PID names. The server should return the valid PIDs and ignore the invalid ones.

4.2. Filtered Cost Map Tests

All tests require an appropriate "cost-type" parameter. At a minimum, clients should run these tests for the "routingcost" metric for the default network map. If possible, clients should also run these tests for the "hopcount" metric and the alternate network map.

Clients should remember that when testing "ordinal" costs, any values are acceptable as long as they are consistent with the order of the "numerical" costs defined in Section 2. Clients are also reminded that ordinal values are only comparable to other values in the same request, and a server may recalculate ordinal values for each request. Hence the same cost point may have ordinal value "6" in a full cost map, but have value "1" in a filtered cost map.

- Empty "srcs" and "dsts" arrays. The server should return the entire cost map.

- Empty "srcs" array, "dsts" array with one or more valid PIDs. The server should return costs from all PIDs to the specified destination PIDs.

- Empty "dsts" array, "srcs" array with one or more valid PIDs. The server should return costs from the specified source PIDs to all destination PIDs.

- "srcs" and "dsts" arrays with only non-existent PID names. The server should return an empty cost map.

- "srcs" and "dsts" arrays with a set of valid PID names (client's choice), plus one or more non-existent PID names in one or the arrays. The server should return costs for the valid PIDs and ignore the non-existent ones.

- The two-element constraint test "ge 20.0", "le 30.0" for the numerical "routingcost" for the default network map, with empty "srcs" and "dsts" arrays (assuming that resource allows constraints, of course). The server should return the all costs in the range, namely:

	mine	mine1	mine1a	mine2	mine3	peer1	peer2
mine	-	-	-	-	-	20.0	25.0
mine1	-	-	-	-	-	23.0	28.0
mine1a	-	-	-	-	-	25.0	30.0
mine2	-	-	-	-	-	24.0	29.0
mine3	-	-	-	-	-	25.0	30.0
peer1	20.0	23.0	25.0	24.0	25.0	-	-
peer2	25.0	28.0	30.0	29.0	30.0	-	-

Figure 12: Filtered Cost Map Constraint Test

4.3. Endpoint Property Service Tests

Every client should verify that the server's EPS resource for the default network's "pid" property returns the correct PID name for a representative set of endpoint addresses. If possible, clients should also verify the alternate network's "pid" property and the "priv:ietf-type" property.

The table below gives the expected values for a set of addresses. Clients are encouraged to test other addresses as well.

Address	def.pid	alt.pid	priv:ietf-type
ipv4:0.0.0.1	default	default	-
ipv4:10.1.2.3	private	private	-
ipv4:100.0.0.1	mine1	default	mine
ipv4:100.0.1.1	mine1a	default	mine
ipv4:100.0.192.1	mine1a	default	mine
ipv4:100.0.64.1	mine1a	default	mine
ipv4:100.130.0.1	mine3	default	mine
ipv4:100.200.0.1	mine	default	mine
ipv4:100.75.0.1	mine2	default	mine
ipv4:101.0.0.1	default	dc1	-
ipv4:101.1.0.1	default	default	-
ipv4:102.0.0.1	default	dc2	-
ipv4:103.0.0.1	default	dc3	-
ipv4:104.0.0.1	default	dc4	-
ipv4:127.0.0.1	loopback	loopback	-
ipv4:127.255.255.255	loopback	loopback	-
ipv4:128.0.0.1	peer1	default	peer
ipv4:129.0.0.1	peer2	default	peer
ipv4:130.0.0.1	peer1	default	peer
ipv4:131.0.0.1	peer2	default	peer
ipv4:132.0.0.1	tran1	default	transit
ipv4:135.0.0.1	tran2	default	transit
ipv4:169.254.1.2	linklocal	linklocal	-
ipv4:201.0.0.1	default	user1	-
ipv4:201.1.2.3	default	default	-
ipv4:202.0.0.1	default	user2	-
ipv4:203.0.0.1	default	user3	-
ipv4:204.0.0.1	default	user4	-
ipv4:99.0.0.1	default	default	-
ipv6:::1	loopback	loopback	-
ipv6:::2	default	default	-
ipv6:2001:db8::	peer1	default	peer
ipv6:2001:db8:8000::1	peer2	default	peer
ipv6:fc00:1::	private	private	-
ipv6:ff80:1:2::	linklocal	linklocal	-

Figure 13: EPS Test Addresses And Property Values

4.4. Endpoint Cost Service Tests

If the ALTO server provides an Endpoint Cost Service (ECS), and if the client supports ECS queries, then the client SHOULD send representative ECS queries to the server. For ECS, the server should use the costs associated with the default network map, so the client can verify the server's response.

An ECS-aware client SHOULD send the following queries to the server, for the "routingcost" and/or "hopcount" metrics, as suppored by the server. A client may add additional tests as desired.

4.4.1. ECS Test 1

This test determines the costs between various endpoints in the "mine*" and "peer*" PIDs:

```
      Query:
        sources:
              ipv4:100.0.0.128     ("mine1")
              ipv4:100.131.39.11   ("mine3")
        destinations:
              ipv4:100.0.0.100     ("mine1")
              ipv4:100.8.1.100     ("mine1")
              ipv4:100.0.1.100     ("mine1a")
              ipv4:100.64.0.100    ("mine2")
              ipv4:100.128.4.100   ("mine3")
              ipv4:130.0.1.100     ("peer1")
              ipv4:132.0.8.100     ("tran1")

      Costs:                     routingcost  hopcount
        ipv4:100.0.0.128 =>
              ipv4:100.0.0.100          0.0         0
              ipv4:100.8.1.100          0.0         0
              ipv4:100.0.1.100          2.0         2
              ipv4:100.64.0.100         6.5         3
              ipv4:100.128.4.100        8.0         3
              ipv4:130.0.1.100         23.0         4
              ipv4:132.0.8.100         38.0         5
        ipv4:100.131.39.11 =>
              ipv4:100.0.0.100          8.0         3
              ipv4:100.8.1.100          8.0         3
              ipv4:100.0.1.100         10.0         4
              ipv4:100.64.0.100         9.0         3
              ipv4:100.128.4.100        0.0         0
              ipv4:130.0.1.100         25.0         4
              ipv4:132.0.8.100         40.0         5
```

Figure 14: ECS Test 1

4.4.2. ECS Test 2

This test determines the costs between endpoints in the "default" PID:

```
      Query:
        sources:
              ipv4:10.0.1.0 ("private")
              ipv6:::2      ("default")
        destinations:
              ipv4:10.0.1.1 ("private")
              ipv6:::1:2    ("default")
```

Figure 15: ECS Test 2

Since no costs are defined between those PIDs the server MUST return an ECS response containing no costs.

4.4.3. ECS Test 3

This test determines the costs between endpoints in the "loopback" PID:

```
      Query:
        sources:
              ipv4:127.0.0.1
              ipv6:::1
        destinations:
              ipv4:127.0.1.0

      Costs:                     routingcost  hopcount
        ipv4:127.0.0.1 =>
              ipv4:127.0.1.0             0.0         0
        ipv6:::1 =>
              ipv4:127.0.1.0             0.0         0
```

Figure 16: ECS Test 3

4.4.4. ECS Test 4

This test determines the cost when the client does not specify any destination addresses. In this case, the server SHOULD use the client’s address as the destination. The costs, however, will depend on the PID for the client’s address, which in turn will depend on the network configuration of the test environment. But in most cases, the client’s PID will be in either the "default" or "private" PIDs.

```
      Query:
        sources:
              ipv4:100.0.0.128     ("mine1")
              ipv4:100.131.39.11   ("mine3")
              ipv4:100.200.0.1     ("mine")
              ipv4:0.0.0.1         ("default")
              ipv4:10.0.0.1        ("private")
              ipv6:::2             ("default")
              ipv6:FC01::          ("private")
        destinations:
              (none specified)

      Costs:                     routingcost  hopcount

             (for clients in "default" PID)
        ipv4:100.0.0.128 =>
              (client address)          63.0         4
        ipv4:100.131.39.11 =>
              (client address)          65.0         4
        ipv4:100.200.0.1 =>
              (client address)          60.0         3
        ipv4:0.0.0.1
              (client address)             -         -
        ipv4:10.0.0.1
              (client address)             -         -
        ipv6:::2
              (client address)             -         -
        ipv6:FC01::
              (client address)             -         -

            (for clients in "private" PID)
        ipv4:100.0.0.128 =>
              (client address)             -         -
        ipv4:100.131.39.11 =>
              (client address)             -         -
        ipv4:100.200.0.1 =>
              (client address)             -         -
        ipv4:0.0.0.1
              (client address)             -         -
        ipv4:10.0.0.1
              (client address)           0.0         0
        ipv6:::2
              (client address)             -         -
        ipv6:FC01::
              (client address)           0.0         0
```

Figure 17: ECS Test 4

4.4.5. ECS Test 5

This test determines the cost when the client does not specify any source addresses. In this case, the server SHOULD use the client's address as the source. The costs, however, will depend on the PID for the client's address, which in turn will depend on the network configuration of the test environment. But in most cases, the client's PID will be in either the "default" or "private" PIDs.

```
   Query:
     sources:
           (none specified)
     destinations:
           ipv4:100.0.0.128     ("mine1")
           ipv4:100.131.39.11   ("mine3")
           ipv4:100.200.0.1     ("mine")
           ipv4:0.0.0.1         ("default")
           ipv4:10.0.0.1        ("private")
           ipv6:::2             ("default")
           ipv6:FC01::          ("private")

   Costs:                     routingcost  hopcount

          (for clients in "default" PID)
     (client address) =>
           ipv4:100.0.0.128          63.0         4
           ipv4:100.131.39.11        65.0         4
           ipv4:100.200.0.1          60.0         3
           ipv4:0.0.0.1                 -         -
           ipv4:10.0.0.1                -         -
           ipv6:::2                     -         -
           ipv6:FC01::                  -         -

         (for clients in "private" PID)
     (client address) =>
           ipv4:100.0.0.128             -         -
           ipv4:100.131.39.11           -         -
           ipv4:100.200.0.1             -         -
           ipv4:0.0.0.1                 -         -
           ipv4:10.0.0.1              0.0         0
           ipv6:::2                     -         -
           ipv6:FC01::                0.0         0
```

Figure 18: ECS Test 5

5. Error Tests

A client may send various invalid requests to a server to verify that the server returns a reasonable response. The following tests are suggested; clients may do additional tests as desired.

Each error test is defined by

description:
A short description of the test.

resource:
The type of server resource to which this test applies. E.g., Filtered Cost Map, Endpoint Property Service, etc.

accept:
The "Accept" HTTP header that the client should send to the server, if something other than the media-type for that resource's response.

content-type:
For POST requests, the Content-Type HTTP header the client should send to the server.

input:
For POST requests, the JSON input the client should send to send to the server.

http-status:
The HTTP status code the server should return.

code, field, value:
The values of the corresponding fields the server should return in an ALTO error message (see Section 8.5.2 of [RFC7285]).

For "property" fields in Endpoint Property Service tests, clients should replace the property name "default.pid" with the resource-specific name of the server's default network map's "pid" property. That is, if the resource id of the server's default network map is "mynet", replace "default.pid" with "mynet.pid".

5.1. Invalid Field Type

For an EPS request, the "endpoints" input field should be a JSON array of one or more addresses. In this test, it is a scalar JSON string.

```
      resource:     Endpoint Property Service
      accept:       application/alto-endpointprop+json
      content-type: application/alto-endpointpropparams+json
      input:        { "properties": ["default.pid"],
                      "endpoints": "ipv4:1.2.3.4" }
      http-status:  400 Bad Request
      code:         E_INVALID_FIELD_TYPE
      field:        endpoints
```

5.2. Missing "properties" Field

This test omits the required "properties" input field.

```
      resource:     Endpoint Property Service
      accept:       application/alto-endpointprop+json
      content-type: application/alto-endpointpropparams+json
      input:        { "endpoints": ["ipv4:1.2.3.4"] }
      http-status:  400 Bad Request
      code:         E_MISSING_FIELD
      field:        properties
```

5.3. Invalid Property Name

This test requests the (presumably!) invalid property "no-such-property".

```
      resource:     Endpoint Property Service
      accept:       application/alto-endpointprop+json
      content-type: application/alto-endpointpropparams+json
      input:        { "properties": ["no-such-property"],
                      "endpoints": "ipv4:1.2.3.4" }
      http-status:  400 Bad Request
      code:         E_INVALID_FIELD_VALUE
      field:        properties
      value:        no-such-property
```

5.4. Invalid Endpoint Addresses

These tests verify that a server rejects various invalid endpoint addresses.

```
         resource:     Endpoint Property Service
         accept:       application/alto-endpointprop+json
         content-type: application/alto-endpointpropparams+json
         input:        { "properties": ["default.pid"],
                         "endpoints": ["ipv4:1.2.3.256"] }
         http-status:  400 Bad Request
         code:         E_INVALID_FIELD_VALUE
         field:        endpoints
         value:        ipv4:1.2.3.256


         resource:     Endpoint Property Service
         accept:       application/alto-endpointprop+json
         content-type: application/alto-endpointpropparams+json
         input:        { "properties": ["default.pid"],
                         "endpoints": ["ipv6:2001:db800::"] }
         http-status:  400 Bad Request
         code:         E_INVALID_FIELD_VALUE
         field:        endpoints
         value:        ipv6:2001:db800::


         resource:     Endpoint Property Service
         accept:       application/alto-endpointprop+json
         content-type: application/alto-endpointpropparams+json
         input:        { "properties": ["default.pid"],
                         "endpoints": ["ipv4:2001:db8::"] }
         http-status:  400 Bad Request
         code:         E_INVALID_FIELD_VALUE
         field:        endpoints
         value:        ipv4:2001:db8::


         resource:     Endpoint Property Service
         accept:       application/alto-endpointprop+json
         content-type: application/alto-endpointpropparams+json
         input:        { "properties": ["default.pid"],
                         "endpoints": ["ipv6:1.2.3.4"] }
         http-status:  400 Bad Request
         code:         E_INVALID_FIELD_VALUE
         field:        endpoints
         value:        ipv6:1.2.3.4
```

5.5. Invalid Cost Type

This test requests the (presumably!) invalid cost metric "no-such-metric". If the server's Filtered Cost Map resource only provides "ordinal" mode cost types, the client should change "numerical" mode to "ordinal" mode, to prevent the server from rejecting the request because of an invalid cost mode.

```
      resource:     Filtered Cost Map
      accept:       application/alto-costmap+json
      content-type: application/alto-costmapfilter+json
      input:        { "cost-type": [
                           "cost-metric": "no-such-cost",
                           "cost-mode": "numerical" ],
                        "endpoints": {
                           "srcs": [],
                           "dsts": [] }
                    }
      http-status:  400 Bad Request
      code:         E_INVALID_FIELD_VALUE
      field:        cost-type/cost-metric
      value:        no-such-cost
```

5.6. Invalid Cost Mode

This test requests the invalid cost mode "no-such-mode". The client should direct this request to a Filtered Cost Map resource which can return the "routingcost" metric, to prevent the server from rejecting the request because of an invalid cost metric.

```
      resource:     Filtered Cost Map
      accept:       application/alto-costmap+json
      content-type: application/alto-costmapfilter+json
      input:        { "cost-type": [
                           "cost-metric": "routingcost",
                           "cost-mode": "no-such-mode" ],
                        "endpoints": {
                           "srcs": [],
                           "dsts": [] }
                    }
      http-status:  400 Bad Request
      code:         E_INVALID_FIELD_VALUE
      field:        cost-type/cost-mode
      value:        no-such-mode
```

5.7. Invalid Cost Constraints

This test uses a constraint test with the undefined "ne" operator. The client should direct this request to a Filtered Cost Map resource which can return the "routingcost" metric in "numerical" mode, to prevent the server from rejecting the request because of an invalid cost metric or mode.

```
      resource:     Filtered Cost Map which accepts constraints
      accept:       application/alto-costmap+json
      content-type: application/alto-costmapfilter+json
      input:        { "cost-type": [
                           "cost-metric": "routingcost",
                           "cost-mode": "numerical" ],
                        "endpoints": {
                           "srcs": [],
                           "dsts": [] },
                        "constraints": ["ne 10"]
                      }
      http-status:  400 Bad Request
      code:         E_INVALID_FIELD_VALUE
      field:        constraints
      value:        no-such-mode
```

5.8. JSON Syntax Error

This test gives syntactically incorrect JSON input to the server.

```
      resource:     Endpoint Property Service
      accept:       application/alto-endpointprop+json
      content-type: application/alto-endpointpropparams+json
      input:        { "properties": }
      http-status:  400 Bad Request
      code:         E_SYNTAX
```

5.9. Invalid Accept Header In GET Request

This test attempts to GET the Full Network Map, without including the appropriate media-type ("application/alto-networkmap+json") in the "Accept" HTTP header. Note that the client must ensure that the HTTP library does not automatically append "*/*" to the "Accept" header. Also note that because this is an HTTP error, [RFC7285] does not specify the content the server is expected to return.

```
      resource:     Full Network Map
      accept:       text/html
      http-status:  406 Not Acceptable
```

5.10. Invalid Accept Header In POST Request

This test requests a property without including the appropriate media-type ("application/alto-endpointprop+json") in the "Accept" HTTP header. Note that the client must ensure that the HTTP library does not automatically append "*/*" to the "Accept" header. Note that because this is an HTTP error, [RFC7285] does not specify the content the server is expected to return.

```
resource:     Endpoint Property Service
accept:       text/html
content-type: application/alto-endpointpropparams+json
input:        { "properties": ["default.pid"],
                "endpoints": ["ipv4:1.2.3.4"] }
http-status:  406 Not Acceptable
```

5.11. Invalid Content-Type Header In POST Request

This test requests a property but provides input with an incorrect Content-Type. Note that because this is an HTTP error, [RFC7285] does not specify the content the server is expected to return.

```
resource:     Endpoint Property Service
accept:       application/alto-endpointprop+json
content-type: text/text
input:        { "properties": ["default.pid"],
                "endpoints": ["ipv4:1.2.3.4"] }
http-status:  415 Unsupported Media Type
              (or 404 Not Found or 400 Bad Request)
```

6. Security considerations

This document does not present any new security considerations above and beyond what is documented in the ALTO protocol [RFC7285].

7. IANA considerations

This document does not require any action from IANA.

8. Normative References

[RFC3849] Huston, G., Lord, L., and P. Smith, "IPv6 Address Prefix Reserved for Documentation", RFC 3849, July 2004.

[RFC5737] Arkko, J., Cotton, M., and L. Vegoda, "IPv4 Address Prefix Reserved for Documentation", RFC 5737, January 2010.

[RFC7159] Bray, T., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, March 2014.

[RFC7285] Almi, R., Penno, R., Yang, Y., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, September 2014.

Appendix A. Appendix: JSON Network And Cost Maps

This section presents the network and cost maps defined in Section 2 formatted as JSON ([RFC7159]) objects.

A.1. Default Network Map

```
"network-map": {
   "default": {
      "ipv4": ["0.0.0.0/0"],
      "ipv6": ["::/0"] },
   "linklocal": {
      "ipv4": ["169.254.0.0/16"],
      "ipv6": ["FF80::/10"] },
   "loopback": {
      "ipv4": ["127.0.0.0/8"],
      "ipv6": ["::1/128"] },
   "mine": {
      "ipv4": ["100.0.0.0/8"] },
   "mine1": {
      "ipv4": ["100.0.0.0/10"] },
   "mine1a": {
      "ipv4": ["100.0.64.0/24", "100.0.192.0/24", "100.0.1.0/24"] },
   "mine2": {
      "ipv4": ["100.64.0.0/10"] },
   "mine3": {
      "ipv4": ["100.128.0.0/10"] },
   "peer1": {
      "ipv4": ["130.0.0.0/16", "128.0.0.0/16"],
      "ipv6": ["2001:DB8::/33"] },
   "peer2": {
      "ipv4": ["131.0.0.0/16", "129.0.0.0/16"],
      "ipv6": ["2001:DB8:8000::/33"] },
   "private": {
      "ipv4": ["10.0.0.0/8", "172.16.0.0/12", "192.168.0.0/16"],
      "ipv6": ["FC00::/7"] },
   "tran1": {
      "ipv4": ["132.0.0.0/16"] },
   "tran2": {
      "ipv4": ["135.0.0.0/16"] }
}
```

Figure 19: Default Network Map, in JSON

A.2. Default "routingcost" Cost Map

```
   "cost-map": {
      "default": {
         "mine": 60.0,  "mine1": 63.0,  "mine2": 64.0,  "mine1a": 65.0,
         "mine3": 65.0 },
      "linklocal": {
         "linklocal": 0.0 },
      "loopback": {
         "loopback": 0.0 },
      "mine": {
         "mine": 0.0,  "mine1": 3.0,  "mine2": 4.0,  "mine1a": 5.0,
         "mine3": 5.0,  "peer1": 20.0,  "peer2": 25.0,  "tran1": 35.0,
         "tran2": 45.0,  "default": 60.0 },
      "mine1": {
         "mine": 3.0,  "mine1": 0.0,  "mine2": 6.5,  "mine1a": 2.0,
         "mine3": 8.0,  "peer1": 23.0,  "peer2": 28.0,  "tran1": 38.0,
         "tran2": 48.0,  "default": 63.0 },
      "mine1a": {
         "mine": 5.0,  "mine1": 2.0,  "mine2": 4.5,  "mine1a": 0.0,
         "mine3": 10.0,  "peer1": 25.0,  "peer2": 30.0,  "tran1": 40.0,
         "tran2": 50.0,  "default": 65.0 },
      "mine2": {
         "mine": 4.0,  "mine1": 7.0,  "mine2": 0.0,  "mine1a": 9.0,
         "mine3": 9.0,  "peer1": 24.0,  "peer2": 29.0,  "tran1": 39.0,
         "tran2": 49.0,  "default": 64.0 },
      "mine3": {
         "mine": 5.0,  "mine1": 8.0,  "mine2": 9.0,  "mine1a": 10.0,
         "mine3": 0.0,  "peer1": 25.0,  "peer2": 30.0,  "tran1": 40.0,
         "tran2": 50.0,  "default": 65.0 },
      "peer1": {
         "mine": 20.0,  "mine1": 23.0,  "mine2": 24.0,  "mine1a": 25.0,
         "mine3": 25.0,  "peer1": 0.0
         },
      "peer2": {
         "mine": 25.0,  "mine1": 28.0,  "mine2": 29.0,  "mine1a": 30.0,
         "mine3": 30.0,  "peer2": 0.0 },
      "private": {
         "private": 0.0 },
      "tran1": {
         "mine": 35.0,  "mine1": 38.0,  "mine2": 39.0,  "mine1a": 40.0,
         "mine3": 40.0,  "tran1": 0.0 },
      "tran2": {
         "mine": 45.0,  "mine1": 48.0,  "mine2": 49.0,  "mine1a": 50.0,
         "mine3": 50.0,  "tran2": 0.0 }
      }
```

Figure 20: Default "routingcost" Cost Map, in JSON

A.3. Default "hopcount" Cost Map

```
   "cost-map": {
      "default": {
         "mine": 3,  "mine1": 4,  "mine2": 4,  "mine3": 4,
         "mine1a": 5 },
      "linklocal": {
         "linklocal": 0 },
      "loopback": {
         "loopback": 0 },
      "mine": {
         "mine": 0,  "mine1": 2,  "mine2": 2,  "mine3": 2,
         "mine1a": 3,  "peer2": 2,  "default": 3,  "peer1": 3,
         "tran2": 3,  "tran1": 4 },
      "mine1": {
         "mine": 2,  "mine1": 0,  "mine2": 3,  "mine3": 3,
         "mine1a": 2,  "peer2": 3,  "default": 4,  "peer1": 4,
         "tran2": 4,  "tran1": 5 },
      "mine1a": {
         "mine": 3,  "mine1": 2,  "mine2": 2,  "mine3": 4,
         "mine1a": 0,  "peer2": 4,  "default": 5,  "peer1": 5,
         "tran2": 5,  "tran1": 6 },
      "mine2": {
         "mine": 2,  "mine1": 3,  "mine2": 0,  "mine3": 3,
         "mine1a": 4,  "peer2": 3,  "default": 4,  "peer1": 4,
         "tran2": 4,  "tran1": 5 },
      "mine3": {
         "mine": 2,  "mine1": 3,  "mine2": 3,  "mine3": 0,
         "mine1a": 4,  "peer2": 3,  "default": 4,  "peer1": 4,
         "tran2": 4,  "tran1": 5 },
      "peer1": {
         "mine": 3,  "mine1": 4,  "mine2": 4,  "mine3": 4,
         "mine1a": 5,  "peer1": 0 },
      "peer2": {
         "mine": 2,  "mine1": 3,  "mine2": 3,  "mine3": 3,
         "mine1a": 4,  "peer2": 0 },
      "private": {
         "private": 0 },
      "tran1": {
         "mine": 4,  "mine1": 5,  "mine2": 5,  "mine3": 5,
         "mine1a": 6,  "tran1": 0 },
      "tran2": {
         "mine": 3,  "mine1": 4,  "mine2": 4,  "mine3": 4,
         "mine1a": 5,  "tran2": 0 }
   }
```

Figure 21: Default "hopcount" Cost Map, in JSON

A.4. Alternate Network Map

```
   "network-map": {
      "dc1": {
         "ipv4": ["101.0.0.0/16"] },
      "dc2": {
         "ipv4": ["102.0.0.0/16"] },
      "dc3": {
         "ipv4": ["103.0.0.0/16"] },
      "dc4": {
         "ipv4": ["104.0.0.0/16"] },
      "default": {
         "ipv4": ["0.0.0.0/0"],
         "ipv6": ["::/0"] },
      "linklocal": {
         "ipv4": ["169.254.0.0/16"],
         "ipv6": ["FF80::/10"] },
      "loopback": {
         "ipv4": ["127.0.0.0/8"],
         "ipv6": ["::1/128"] },
      "private": {
         "ipv4": ["10.0.0.0/8", "172.16.0.0/12", "192.168.0.0/16"],
         "ipv6": ["FC00::/7"] },
      "user1": {
         "ipv4": ["201.0.0.0/16"] },
      "user2": {
         "ipv4": ["202.0.0.0/16"] },
      "user3": {
         "ipv4": ["203.0.0.0/16"] },
      "user4": {
         "ipv4": ["204.0.0.0/16"] }
   }
```

Figure 22: Alternate Network Map, in JSON

A.5. Alternate "routingcost" Cost Map

```
   "cost-map": {
      "dc1": {
         "dc1": 0.0,  "user1": 10.0,  "user2": 15.0,  "dc2": 20.0,
         "dc3": 20.0,  "dc4": 20.0,  "user3": 20.0,  "user4": 25.0,
         "default": 50.0 },
      "dc2": {
         "dc1": 20.0,  "user1": 15.0,  "user2": 10.0,  "dc2": 0.0,
         "dc3": 20.0,  "dc4": 20.0,  "user3": 15.0,  "user4": 20.0,
         "default": 55.0 },
      "dc3": {
         "dc1": 20.0,  "user1": 20.0,  "user2": 15.0,  "dc2": 20.0,
         "dc3": 0.0,  "dc4": 20.0,  "user3": 10.0,  "user4": 15.0,
         "default": 55.0 },
      "dc4": {
         "dc1": 20.0,  "user1": 25.0,  "user2": 20.0,  "dc2": 20.0,
         "dc3": 20.0,  "dc4": 0.0,  "user3": 15.0,  "user4": 10.0,
         "default": 50.0 },
      "default": {
         "dc1": 50.0,  "dc2": 55.0,  "dc3": 55.0,  "dc4": 50.0 },
      "linklocal": {
         "linklocal": 0.0 },
      "loopback": {
         "loopback": 0.0 },
      "private": {
         "private": 0.0 },
      "user1": {
         "dc1": 10.0,  "user1": 0.0,  "dc2": 15.0,  "dc3": 20.0,
         "dc4": 25.0 },
      "user2": {
         "dc1": 15.0,  "user2": 0.0,  "dc2": 10.0,  "dc3": 15.0,
         "dc4": 20.0 },
      "user3": {
         "dc1": 20.0,  "dc2": 15.0,  "dc3": 10.0,  "dc4": 15.0,
         "user3": 0.0 },
      "user4": {
         "dc1": 25.0,  "dc2": 20.0,  "dc3": 15.0,  "dc4": 10.0,
         "user4": 0.0 }
   }
```

Figure 23: Alternate "routingcost" Cost Map, in JSON

A.6. Alternate "hopcount" Cost Map

```
   "cost-map": {
      "dc1": {
         "dc1": 0,  "dc2": 2,  "dc3": 2,  "dc4": 2,
         "user1": 2,  "default": 3,  "user2": 3,  "user3": 4,
         "user4": 5 },
      "dc2": {
         "dc1": 2,  "dc2": 0,  "dc3": 2,  "dc4": 2,
         "user1": 3,  "default": 4,  "user2": 2,  "user3": 3,
         "user4": 4 },
      "dc3": {
         "dc1": 2,  "dc2": 2,  "dc3": 0,  "dc4": 2,
         "user1": 4,  "default": 4,  "user2": 3,  "user3": 2,
         "user4": 3 },
      "dc4": {
         "dc1": 2,  "dc2": 2,  "dc3": 2,  "dc4": 0,
         "user1": 5,  "default": 3,  "user2": 4,  "user3": 3,
         "user4": 2 },
      "default": {
         "dc1": 3,  "dc2": 4,  "dc3": 4,  "dc4": 3 },
      "linklocal": {
         "linklocal": 0 },
      "loopback": {
         "loopback": 0 },
      "private": {
         "private": 0 },
      "user1": {
         "dc1": 2,  "dc2": 3,  "dc3": 4,  "dc4": 5,
         "user1": 0 },
      "user2": {
         "dc1": 3,  "dc2": 2,  "dc3": 3,  "dc4": 4,
         "user2": 0 },
      "user3": {
         "dc1": 4,  "dc2": 3,  "dc3": 2,  "dc4": 3,
         "user3": 0 },
      "user4": {
         "dc1": 5,  "dc2": 4,  "dc3": 3,  "dc4": 2,
         "user4": 0 }
   }
```

Figure 24: Alternate "hopcount" Cost Map, in JSON

Authors' Addresses

Wendy Roome
Bell Laboratories, Alcatel-Lucent
600 Mountain Ave, Rm 3B-324
Murray Hill, NJ 07974
USA

Phone: +1-908-582-7974
Email: w.roome@alcatel-lucent.com

GuohaiChen
Huawei Technologies
101 Software Avenue, Yuhua District
Nanjing,
China

Phone: +8615805180590
Email: chenguohai@huawei.com chenguohai67@outlook.com

Hans Seidel
BENOCS GmbH
Winterfeldtstr. 21
Berlin,
Germany

Phone: +493057700040
Email: hseidel@benocs.com