Application Layer Traffic Optimization (ALTO) WG

CERN/LHCONE/NRP ALTO Deployment Update

Presenter: Jordi/Kai/Jensen
on behalf of the team and collaborators

IETF 115 Hackathon
11 November 2022, London

I E T F

Outline

- Introduction to OpenALTO and openalto.org
- Deployment update at CERN/LHCONE/NRP
- Challenges and lessons

Context: OpenALTO, openalto.org

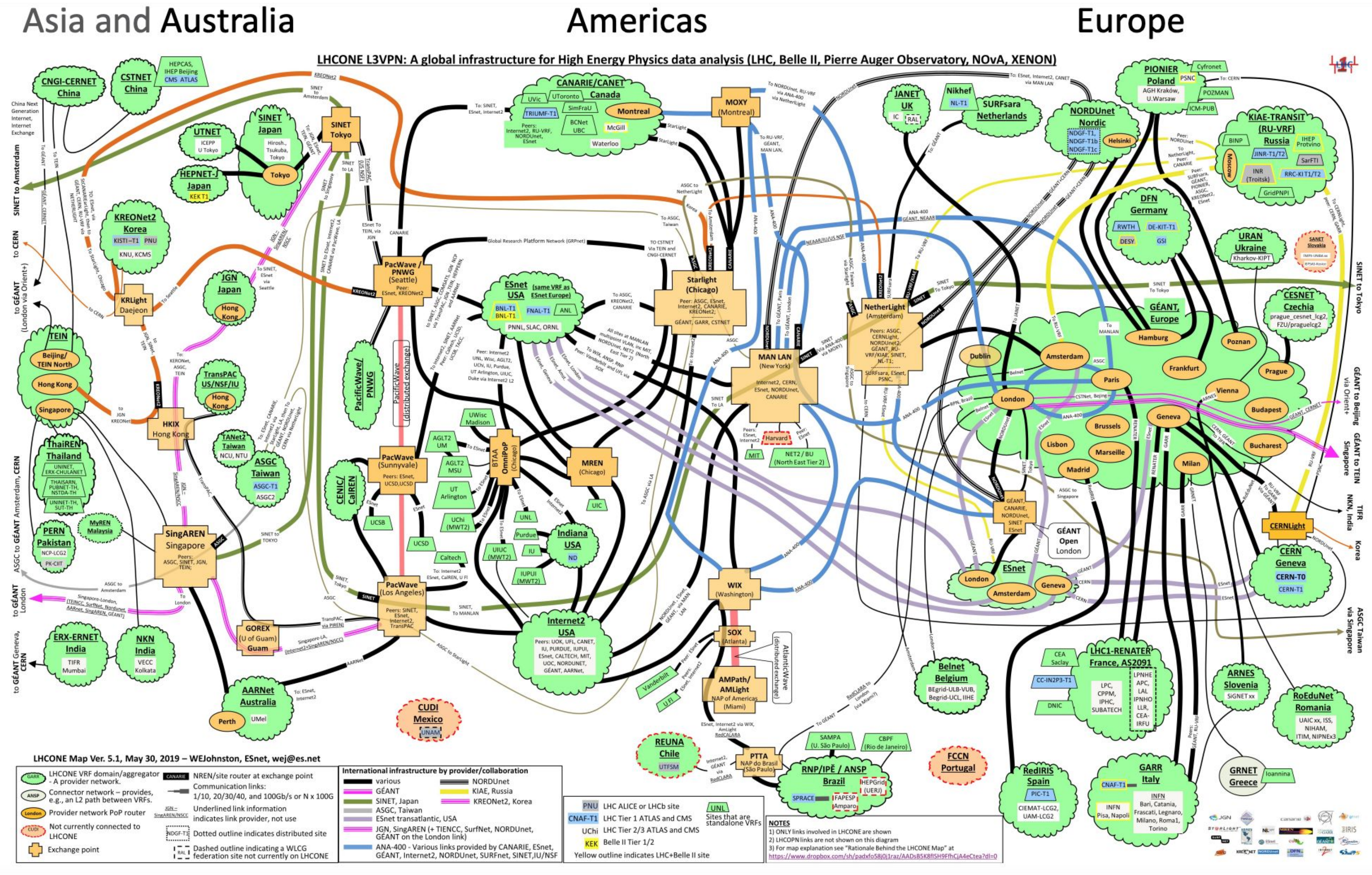
OpenALTO is an open-source **implementation** and platform of ALTO (MIT License).

Available at *https://github.com/openalto/alto*

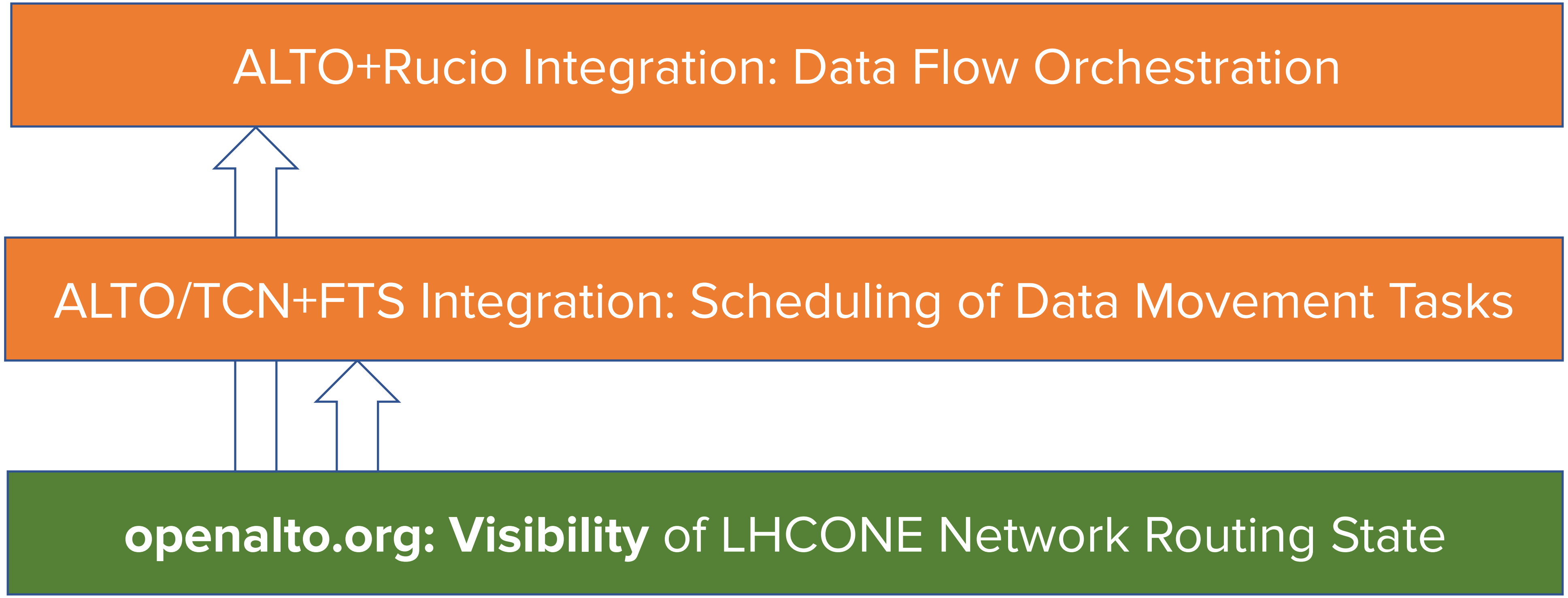
openalto.org is a running instance of deployment of OpenALTO, providing network information, in particular, in the context of data-intensive sciences, such as LHCONE.

Available at *https://{service}.openalto.org* (ALTO only)

Context: LHCONE

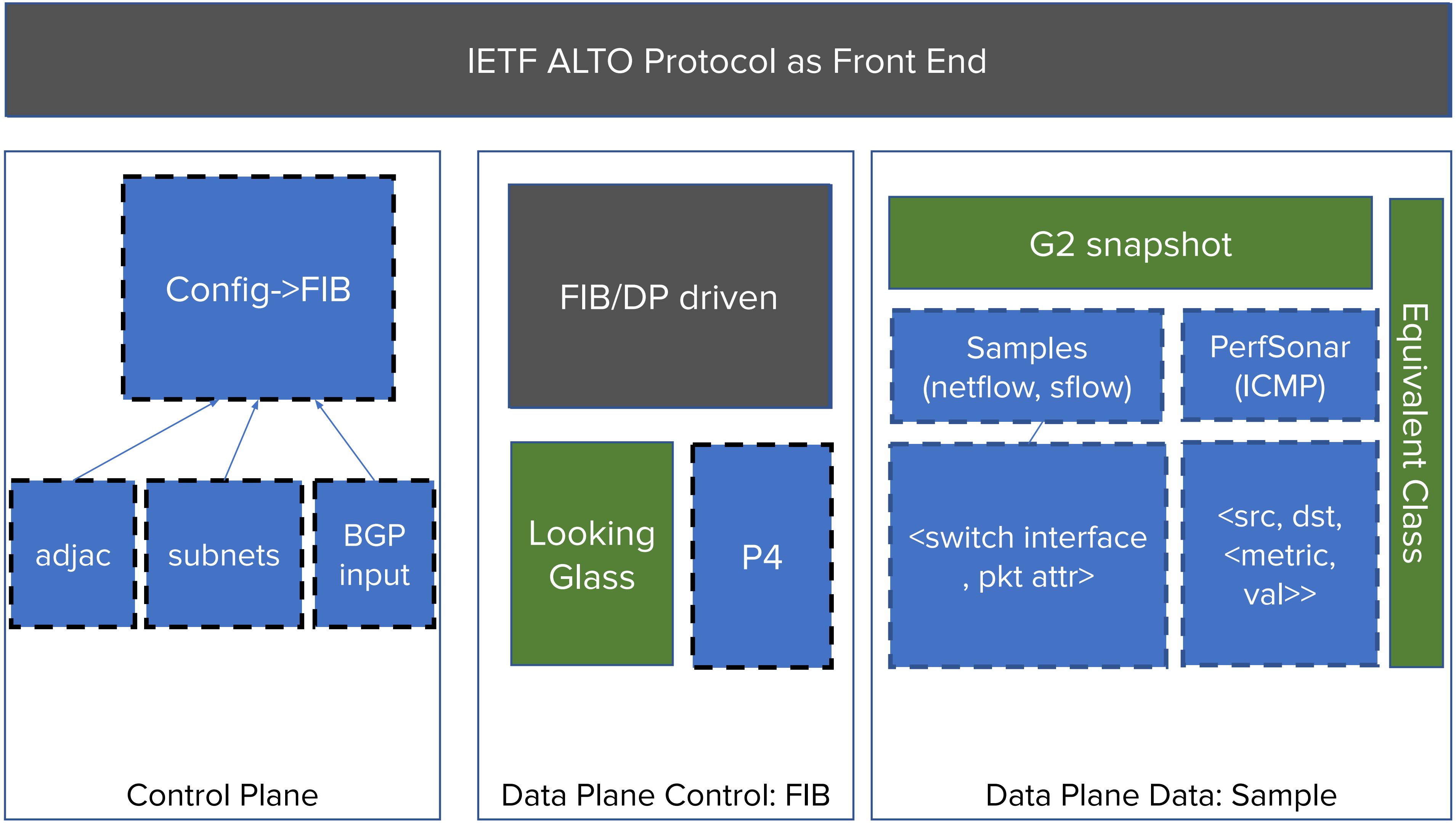


Context: LHCONE, openalto.org Use Cases



OpenALTO Architecture

Components before IETF 114
Components before IETF 115
Components to be developed

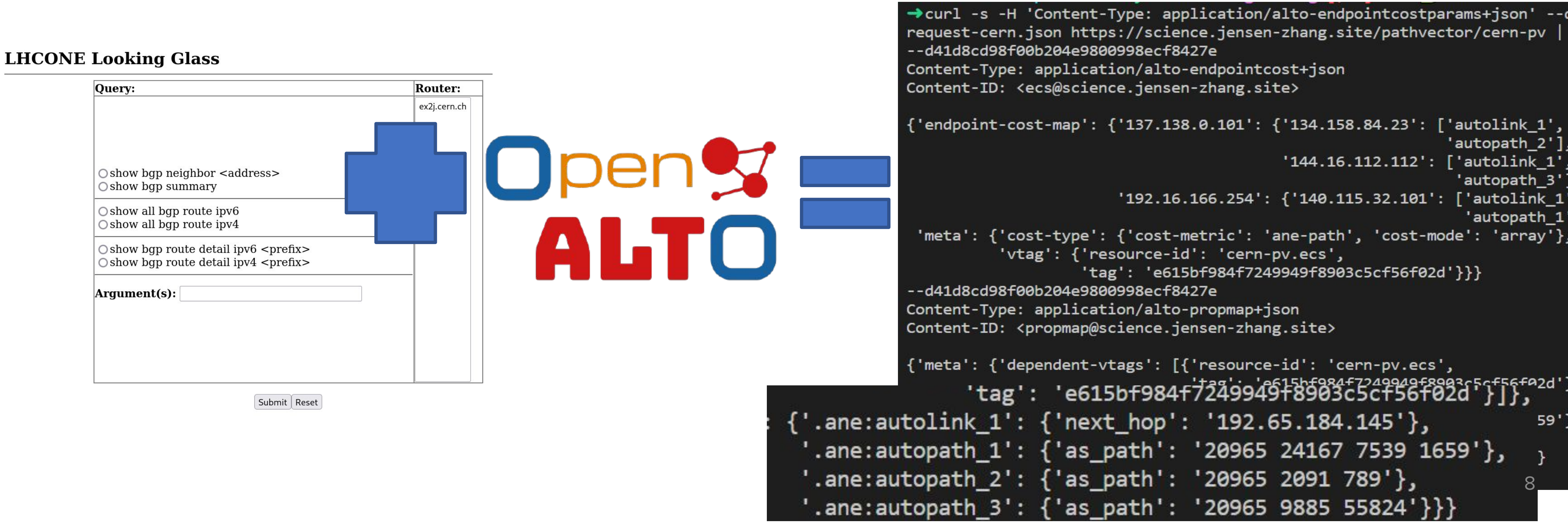


Deployment Update

- CERN/LHCONE deployment (server)
  - Online: October 25
  - **(CERN internal server) https://alto-cern.cern.ch**
  - (public mirror) https://as513.openalto.org/pathvector/cern-pv
  - Supported standards: RFC 7285, RFC 9275
- NRP deployment (server)
  - Online: November 6
  - **(NRP server) https://alto.nrp-nautilus.io/pathvector/nrp**
  - (public mirror) https://nrp.openalto.org/pathvector/nrp-pv
  - Supported standards: RFC 7285, RFC 9240
- Rucio/FTS integration (client)
  - (Forked repository) https://github.com/fno2010/rucio

CERN/LHCONE Deployment Update

- Data source: CRIC database, LHCONE Looking Glass server
- Available information: AS path, next hop router



CERN/LHCONE Deployment

- An agent fetches the prefixes of LHCONE sites from the CRIC database
- and periodically queries the CERN LHCONE Looking Glass server

```
14.139.119.64/26   *[BGP/170] 4d 02:07:52, localpref 100, from 62.40.126.19
                      AS path: 20965 9885 55824 I, validation-state: unverified
                    > to 192.65.184.145 via irb.183
                    [BGP/170] 4d 02:07:52, localpref 100, from 62.40.126.21
                      AS path: 20965 9885 55824 I, validation-state: unverified
                    > to 192.65.184.145 via irb.183
                    [BGP/170] 00:12:46, localpref 100, from 192.65.183.30
                      AS path: 2603 20965 9885 55824 I, validation-state: unverified
                    > to 192.65.183.46 via xe-0/0/29:0.0
18.12.0.0/20       *[BGP/170] 7w5d 12:38:01, localpref 100, from 198.124.80.21
                      AS path: 293 3 I, validation-state: unverified
                    > to 192.65.183.46 via xe-0/0/29:0.0
                    [BGP/170] 1w6d 16:11:46, localpref 100, from 62.40.126.19
                      AS path: 20965 293 3 I, validation-state: unverified
                    > to 192.65.184.145 via irb.183
                    [BGP/170] 3w5d 15:42:38, localpref 100, from 62.40.126.21
                      AS path: 20965 293 3 I, validation-state: unverified
                    > to 192.65.184.145 via irb.183
                    [BGP/170] 4w0d 20:51:33, localpref 100, from 144.206.255.142
                      AS path: 57484 293 3 I, validation-state: unverified
                    > to 192.65.183.46 via xe-0/0/29:0.0
                    [BGP/170] 7w5d 12:38:05, localpref 100
                      AS path: 20641 293 3 I, validation-state: unverified
                    > to 192.65.183.46 via xe-0/0/29:0.0
                    [BGP/170] 00:12:46, localpref 100, from 192.65.183.30
                      AS path: 2603 11537 3 I, validation-state: unverified
                    > to 192.65.183.46 via xe-0/0/29:0.0
```

Dest-Prefix: 14.139.119.64/26
AS-PATH: 20965 9885 55824
Next-Hop: 192.65.184.145

Dest-Prefix: 18.12.0.0/20
AS-PATH: 293 3
Next-Hop: 192.65.183.46

CERN/LHCONE Deployment: Examples

Query Example (ECS with path vector extension)

```
→ cat request-cern.json
{
  "cost-type": {
    "cost-metric": "ane-path",
    "cost-mode": "array"
  },
  "endpoint-flows": [
     {
        "srcs": [ "ipv4:137.138.0.101" ],
        "dsts": [ "ipv4:134.158.84.23", "ipv4:144.16.112.112" ]
     },
     {
        "srcs": [ "ipv4:192.16.166.254" ],
        "dsts": [ "ipv4:140.115.32.101" ]
     }
  ],
  "ane-property-names": [ "next_hop", "as_path" ]
}
```

Response Example (ECS with path vector extension)

```
→curl -s -H 'Content-Type: application/alto-endpointcostparams+json' --data-ascii @
request-cern.json https://science.jensen-zhang.site/pathvector/cern-pv | ./pprint
--d41d8cd98f00b204e9800998ecf8427e
Content-Type: application/alto-endpointcost+json
Content-ID: <ecs@science.jensen-zhang.site>

{'endpoint-cost-map': {'137.138.0.101': {'134.158.84.23': ['autolink_1',
                                                           'autopath_2'],
                                         '144.16.112.112': ['autolink_1',
                                                            'autopath_3']},
                       '192.16.166.254': {'140.115.32.101': ['autolink_1',
                                                             'autopath_1']}},
 'meta': {'cost-type': {'cost-metric': 'ane-path', 'cost-mode': 'array'},
          'vtag': {'resource-id': 'cern-pv.ecs',
                   'tag': 'e615bf984f7249949f8903c5cf56f02d'}}}
--d41d8cd98f00b204e9800998ecf8427e
Content-Type: application/alto-propmap+json
Conte
{'met
 'pro
--d41
```

```
endent-vtags': [{'resource-id': 'cern-pv.ecs',
                 'tag': 'e615bf984f7249949f8903c5cf56f02d'}]},
': {'.ane:autolink_1': {'next_hop': '192.65.184.145'},
    '.ane:autopath_1': {'as_path': '20965 24167 7539 1659'},
    '.ane:autopath_2': {'as_path': '20965 2091 789'},
    '.ane:autopath_3': {'as_path': '20965 9885 55824'}}}
b204e9800998ecf8427e--
```

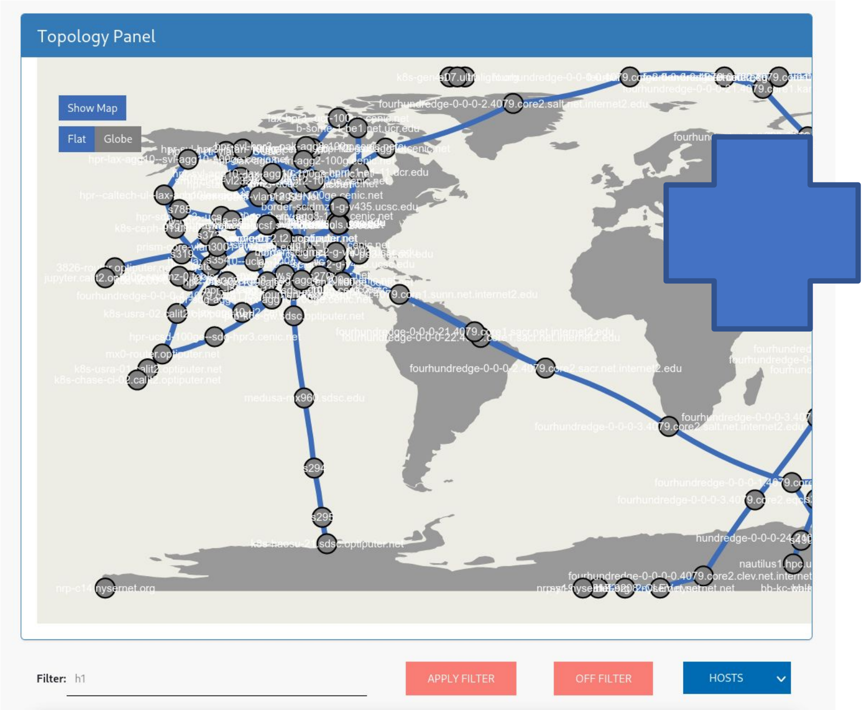
FIB Retrieval (LG; deployment at CERN and GEANT)

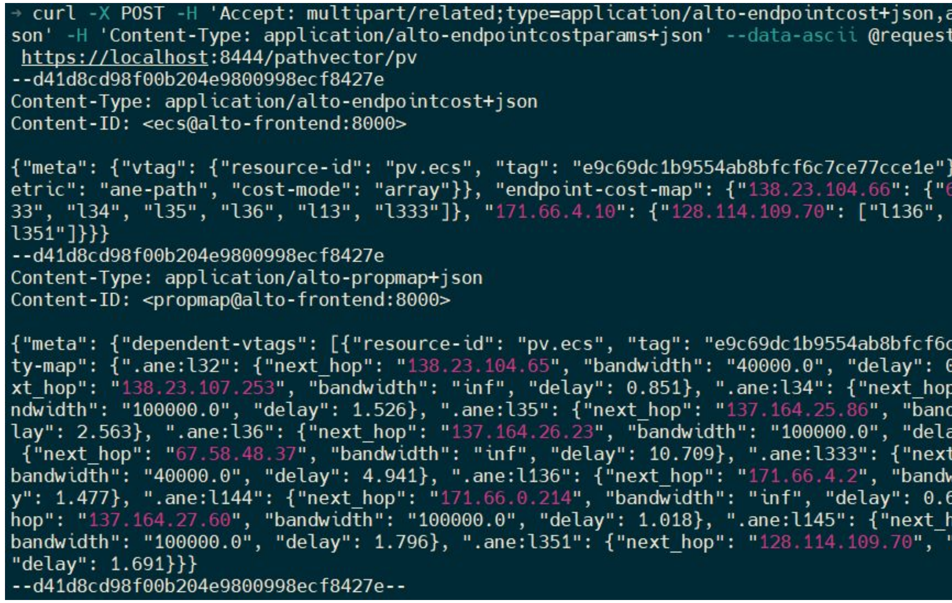
Implementation

```
etc > {} lg-agent.json > ...
{
    "namespace": "default",
    "agent_class": "alto.agent.cernlg.LookingGlassAgent",
    "uri": "http://lhcone-lg.cern.ch/lg.cgi",
    "default_router": "ex2j.cern.ch:juniper",
    "refresh_interval": 300
}
```

NRP Deployment Update

- Data source: G2 snapshot, NRP NetSage
- Available information: link capacity, link delay in the overlay network





```
→ curl -X POST -H 'Accept: multipart/related;type=application/alto-endpointcost+json,a
son' -H 'Content-Type: application/alto-endpointcostparams+json' --data-ascii @request
 https://localhost:8444/pathvector/pv
--d41d8cd98f00b204e9800998ecf8427e
Content-Type: application/alto-endpointcost+json
Content-ID: <ecs@alto-frontend:8000>

{"meta": {"vtag": {"resource-id": "pv.ecs", "tag": "e9c69dc1b9554ab8bfcf6c7ce77cce1e"}
etric": "ane-path", "cost-mode": "array"}}, "endpoint-cost-map": {"138.23.104.66": {"6
33", "l34", "l35", "l36", "l13", "l333"]}, "171.66.4.10": {"128.114.109.70": ["l136",
l351"]}}}
--d41d8cd98f00b204e9800998ecf8427e
Content-Type: application/alto-propmap+json
Content-ID: <propmap@alto-frontend:8000>

{"meta": {"dependent-vtags": [{"resource-id": "pv.ecs", "tag": "e9c69dc1b9554ab8bfcf6c
ty-map": {".ane:l32": {"next_hop": "138.23.104.65", "bandwidth": "40000.0", "delay": 6
xt_hop": "138.23.107.253", "bandwidth": "inf", "delay": 0.851}, ".ane:l34": {"next_hop
ndwidth": "100000.0", "delay": 1.526}, ".ane:l35": {"next_hop": "137.164.25.86", "band
lay": 2.563}, ".ane:l36": {"next_hop": "137.164.26.23", "bandwidth": "100000.0", "dela
 {"next_hop": "67.58.48.37", "bandwidth": "inf", "delay": 10.709}, ".ane:l333": {"next
bandwidth": "40000.0", "delay": 4.941}, ".ane:l136": {"next_hop": "171.66.4.2", "bandw
y": 1.477}, ".ane:l144": {"next_hop": "171.66.0.214", "bandwidth": "inf", "delay": 0.6
hop": "137.164.27.60", "bandwidth": "100000.0", "delay": 1.018}, ".ane:l145": {"next_h
bandwidth": "100000.0", "delay": 1.796}, ".ane:l351": {"next_hop": "128.114.109.70", "
"delay": 1.691}}}
--d41d8cd98f00b204e9800998ecf8427e--
```

NRP/G2 Deployment

- G2 snapshot contains an overlay topology, overlay paths for active flows, and bottleneck structures


```
▼ flows: {flowgroups: [{end: "h1", exp_share: "331.99", id: "f1179156",…},…], num_flowgroups: 2
  ▼ flowgroups: [{end: "h1", exp_share: "331.99", id: "f1179156",…},…]
    ▼ [0 … 99]
      ▼ 0: {end: "h1", exp_share: "331.99", id: "f1179156",…}
          end: "h1"
          exp_share: "331.99"
          id: "f1179156"
          info: "{'src_ip': '138.23.104.66', 'src_port': 'TCP:6802', 'dst_ip': '171.66.4.10', '
        ▶ links: [{group: 1, id: "l070", source: "h7", target: "s8"},…]
          num_bytes: 1424560000
          num_flows: 1
          qos_class: null
          start: "h7"
          start_time: 1667516443.4553416
```


EC rule:

src_prefix: 138.23.104.64/30,
dst_prefix: 171.66.4.8/30

src: 138.23.104.64/30,
dst: 171.66.4.8/30,
ane-path: [...]

NRP/G2 Deployment: Examples

Query Example (ECS with path vector extension)

```
→ cat request.json
{
  "cost-type": {
    "cost-metric": "ane-path",
    "cost-mode": "array"
  },
  "endpoint-flows": [
    {
      "srcs": [ "ipv4:138.23.104.66" ],
      "dsts": [ "ipv4:67.58.50.67" ]
    },
    {
      "srcs": [ "ipv4:171.66.4.10" ],
      "dsts": [ "ipv4:128.114.109.70" ]
    }
  ],
  "ane-property-names": [ "next_hop", "bandwidth", "delay" ]
}
```

Response Example (ECS with path vector extension)

```
"endpoint-cost-map": {"138.23.104.66": {"67.58.50.67": ["l32", "l
"171.66.4.10": {"128.114.109.70": ["l136", "l144", "l9", "l145", "
```

```
etric": "ane-path", "cost-mode": "array"}}, "endpoint-cost-map": {"138.23.104.66": {"67.58.50.67": ["l32", "l
33", "l34", "l35", "l36", "l13", "l333"]}, "171.66.4.10": {"128.114.109.70": ["l136", "l144", "l9", "l145", "
l351"]}}}
--d41d8cd98f00b204e9800998ecf8427e
Content-Type: application/alto-propmap+json
Content-ID: <propmap@alto-frontend:8000>

{"meta": {"dependent-vtags": [{"resource-id": "pv.ecs", "tag": "e9c69dc1b9554ab8bfcf6c7ce77cce1e"}]}, "proper
ty-map": {".ane:l32": {"next_hop": "138.23.104.65", "bandwidth": "40000.0", "delay": 0.737}, ".ane:l33": {"ne
xt_hop": "138.23.107.253", "bandwidth": "inf", "delay": 0.851}, ".ane:l34": {"next_hop": "137.164.27.17", "ba
ndwidth": "100000.0", "delay": 1.526}, ".ane:l35": {"next_hop": "137.164.25.86", "bandwidth": "100000.0", "de
lay": 2.563}, ".ane:l36": {"next_hop": "137.164.26.23", "bandwidth": "100000.0", "delay": 2.575}, ".ane:l13":
 {"next_hop": "67.58.48.37", "bandwidth": "inf", "delay": 10.709}, ".ane:l333": {"next_hop": "67.58.50.67", "
```

```
ne:l32": {"next_hop": "138.23.104.65", "bandwidth": "40000.0", "delay": 0.737},
```

```
"delay": 1.691}}}
--d41d8cd98f00b204e9800998ecf8427e--
```

Agent Configuration: DP sampling and EC configuration

Implementation

etc > {} nrp-agent.json > ...

```
{
    "namespace": "default",
    "agent_class": "alto.agent.g2.G2Agent",
    "base_uri": "https://g2.nrp-nautilus.io/api/",
    "username": "admin",
    "password": "admin",
    "ec_rule": "/etc/ec-rule.json",
    "refresh_interval": 300
}
```

{} ec-rule.json > {} 3

```
[
    {
        "src_prefix": "128.114.109.70/24",
        "dst_prefix": "163.253.70.0/24"
    },
    {
        "src_prefix": "128.114.109.70/24",
        "dst_prefix": "163.253.71.0/24"
    },
    {
        "src_prefix": "128.114.109.70/24",
        "dst_prefix": "163.253.72.0/24",
        "dst_port": 80
    },
```

Rucio/FTS Integration Update

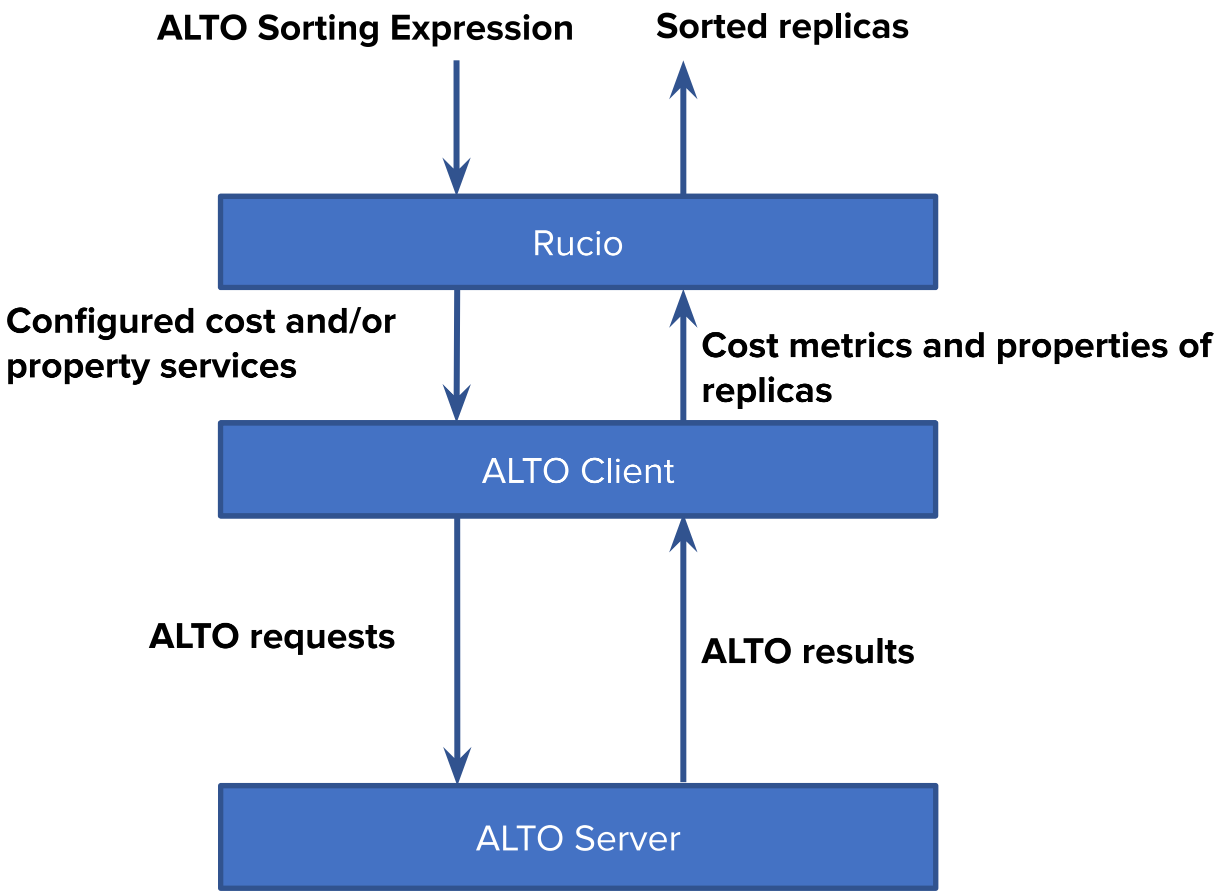
- Recap: In IETF 113, we have added a new option to sort replicas in Rucio based on a single ALTO routing cost
- In IETF 115, we are extending the capability of replica sorting and filtering using multiple ALTO resources
  - Entity properties: geolocation information (country, continent, etc.)
  - Endpoint cost: geo distance, AS hop count
- Additional data sources: Maxmind geoip database

ALTO-based Sorting Expression in Rucio

Currently: replica sorting based on a single metric

Our goal: replica sorting based on multiple metrics and property constraints

ALTO provides a unified interface to query these cost metrics and properties



ALTO Sorting Expression

Example:
BY as_hopcount, geodist
WHERE geo_country="UK"

Select replicas in UK and sort them first by AS-level hopcount, and then by geo distance if the AS paths to two replicas have the same number of hops

Key Syntax of Sorting Expression

```
expr := BY metrics [WHERE cond]
metrics := [metrics “,”] cost_def
cond := def op val
      | val op def
      | def op def
      | “(” cond “)”
      | cond OR cond
      | cond AND cond
def := cost_def | prop_def
```

Rucio Integration Example

Configuration Example

```
[client]
# ALTO server
default_ird = https://science.jensen-zhang.site/directory/default
metrics = {
  "as_hopcount": {
    "resource_type": "path-vector",
    "resource_id": "cern-pv",
    "prop_name": "as_path",
    "prop_transformer": "tolist | len",
    "aggr_transformer": "sum"
  },
  "delay_ow": {
    "resource_type": "cost-map",
    "resource_id": "delay-ow",
    "dependent_network_map": "default-networkmap"
  }}
```

Map properties of ANEs into end-to-end metrics

Sorting Expression Example

```
s --sort='alto;stmt="BY as_hopcount,delay_ow"'
```

Result Example

```
-replicas --sort='alto;stmt="BY as_hopcount,delay_ow"' --metalink test:file1
<?xml version="1.0" encoding="UTF-8"?>
<metalink xmlns="urn:ietf:params:xml:ns:metalink">
 <file name="file1">
  <identity>test:file1</identity>
  <hash type="adler32">69fe2b13</hash>
  <hash type="md5">12969016e761864f30f97dd5fb259e30</hash>
  <size>1048576</size>
  <glfn name="/atlas/rucio/test:file1"></glfn>
  <url location="XRD1" domain="wan" priority="1" client_extract="false">root://xrd1:1094//rucio/test/80/25/file1</url>
  <url location="XRD3" domain="wan" priority="2" client_extract="false">root://xrd3:1096//rucio/test/80/25/file1</url>
  <url location="XRD4" domain="wan" priority="3" client_extract="false">root://xrd4:1097//rucio/test/80/25/file1</url>
 </file>
</metalink>
```

Future Deployment Plans

- CERN/LHCONE:
  - Multi-domain endpoint cost service
  - Milestone: IETF 116
  - Deployment at other LHCONE networks (ESNET, GEANT, etc.)
- NPR/G2:
  - Flow prediction service (exposing fair share as cost)
  - Provide bottleneck structure
  - Milestone: IETF 116
- Rucio/FTS integration:
  - Finalize the unified replica sorting feature and send PR to Rucio
  - ALTO-assisted FTS scheduling for resource control in science networks
  - Milestone: IETF 116

Experiences, Challenges and Lessons

- Data source heterogeneity
- Data source conflicts
- Data fragmentation
- Incomplete information
- Data source availability

Heterogeneous Data Sources

We identify 4 types of heterogeneity during our deployment efforts

- **Heterogeneous information (H1)**: Different data sources provide different types of information
- **Heterogeneous data formats (H2)**: Different data sources provide the same type of information in different formats
- **Heterogeneous collection methodologies (H3)**: Different data sources collect the same type of information using different methodologies with different authority scopes and levels
- **Heterogeneous quality-of-service (H4)**: Different data sources have different performances

Experiences and lessons

- Engineering-wise: Provide common abstractions/data schemas (e.g., FIB-like abstraction and redis key encoding) and handle heterogeneity through plugins
- Sufficient to handle *H1: heterogeneous information, H2: heterogeneous data formats* when there are no conflicts, and *H4: Heterogeneous quality-of-service*
- Cannot handle *H3: Heterogeneous collection methodologies*

Data Source Conflicts

Different data sources provide different values for the same type of information for the same entity or endpoint pairs

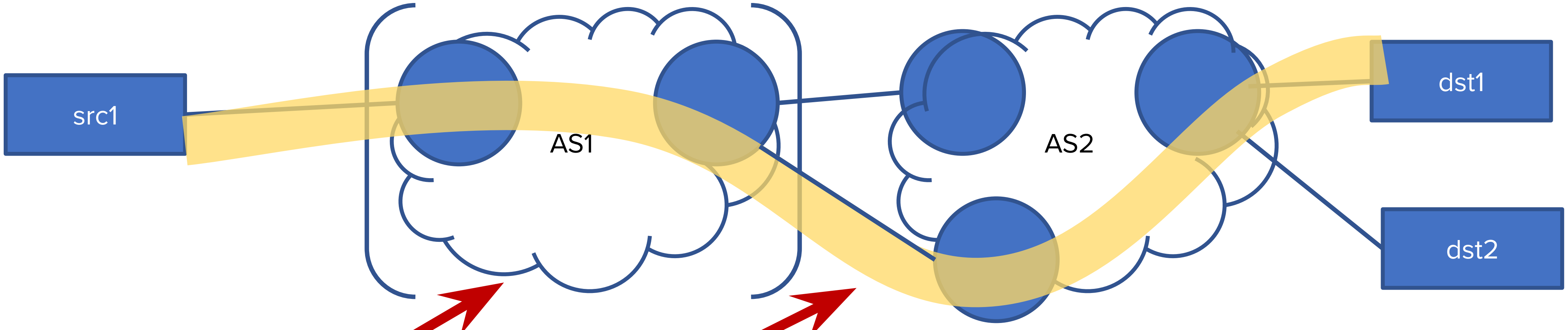
- Example: geo-location
  - Source 1: NRP Net Sage
  - Source 2: Maxmind geolocation database
- For IP address 139.182.103.11
  - NetSage: (34.108345, -117.289765)
  - Maxmind: (37.751, -97.822)

Experiences and Lessons

- ALTO server implementations should provide mechanisms to resolve conflicts (e.g., through prioritization)
- A follow-up question: who should set the priorities and how?
    - Solution 1 (simple): Operators of the ALTO servers can manually specify the global priority of data sources
    - Solution 2 (advanced): The priority depends authority levels of the data sources (e.g., NetSage has higher authority than Maxmind for NRP devices), which may be different in different prexies/regions/...
- Suggestion: Include data source prioritization in the ALTO OAM document?

Data Source Fragmentation

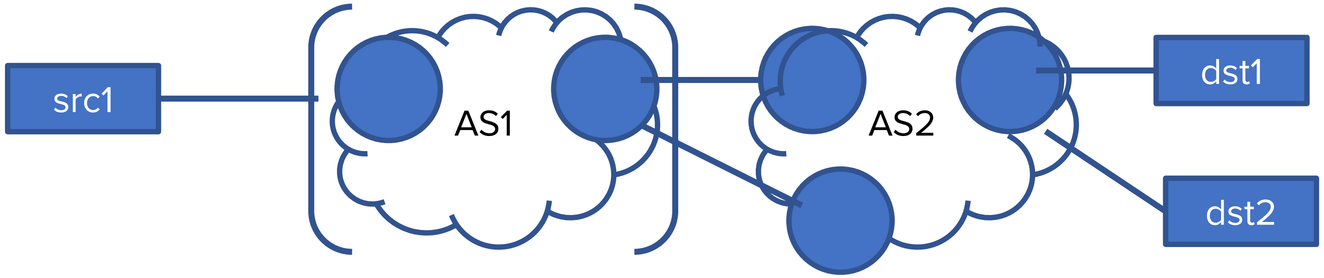
We identify two types of fragmentation



- “Vertical fragmentation”: Each data source only knows information in its own administrative domain
- “Horizontal fragmentation”: Each data source only knows information between some source and destination pairs (especially for sampling-based methods such as traceroute or sflow)
- Example: CERN LookingGlass, GEANT LookingGlass, PerfSonar

Experiences and Lessons

- For vertical fragmentation, cross-domain ALTO coordination is essential, e.g., recursive queries
- Current ALTO query interface (e.g., using src and dst IP addresses) is not sufficient for cross-domain server discovery and queries
- Example: using (src,dst) is not sufficient to determine the path in AS2



- Potential solution and suggestion: Adding extra attributes to support recursive queries, including ingress IP address, virtual network identifier, etc.

Incomplete Information

There are two types of incomplete information

- Type 1: The information does not cover the query space (usually a consequence of horizontal fragmentation caused by sampling-based methods)
    - Example: traceroute/sflow only provides flow-level path information
- Type 2: The information has missing fragments
    - Example: Some routers do not respond to traceroute (i.e., ICMP)

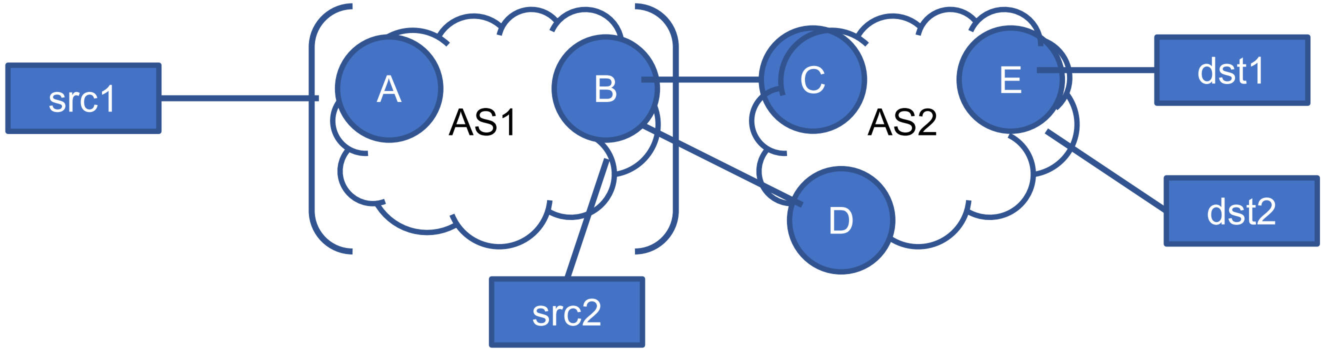
```
},
{
    "src_prefix": "128.114.109.70/24",
    "dst_prefix": "163.253.71.0/24"
},
{
    "src_prefix": "128.114.109.70/24",
    "dst_prefix": "163.253.72.0/24",
    "dst_port": 80
},
```


Experiences and Lessons (I)

- For incomplete query space, the basic idea is to map sampled results as representatives of an atomic query space
- A follow-up question: How to determine the mapping?
- Solution 1: Configure ALTO servers to specify atomic query space (motivated by equivalent classes in network verification literature), either manually or based on routing configurations (as in our NRP deployment)
- Solution 2: For some sampling data (e.g., RTT/hop count between src/dst), use learning models to determine the atomic query spaces (e.g., by minimizing prediction error)

Experiences and Lessons (II)

- For missing fragments, the ALTO server may not be able to return a **deterministic** result
- Example: assume we have the following traceroute results:
  - src1->dst1: A, B, ?, ?, E
  - Both B->C and B->D are potentially on the path



- Solution 1: Synthesize with other samples, e.g., with src2->dst1: B, C, ?, E, the server may infer that B->C is on src1->dst1. But with src2->dst1: B, ?, ?, E, the server cannot determine
- Solution 2: Return all potential results (e.g., as a DAG in this case, see discussion at https://mailarchive.ietf.org/arch/msg/alto/2RMZgqSI2-wQ-eHKcnPysIPnzvs/)

Data Source Availability

Data sources and agents may fail and put the ALTO server out of service

- Case 1: Server returns unexpected results (e.g., server internal errors or inconsistent formats) that crash the agent
- Case 2: Some data sources are not designed for highly frequent queries and may fail/hang upon agent requests

Lessons

- The ALTO server must be able to handle data source failures
- The ALTO server should start with a lower frequency

Feedback to the WG

- OAM
  - Prioritization of data sources
  - Specification of atomic query spaces
- Protocol extensions
  - Advanced query filters
    - support cross-domain scenarios: server discovery and recursive queries
    - flow-level queries
  - New data formats to efficiently represent non-deterministic query results
- Implementation guideline
  - Handling data source failures

Thanks!

Q&A

alto@ietf.org