ALTO Code Base and Deployments

Jordi Ros-Giralt and Kai Gao, on behalf of the ALTO Code Base Project

https://github.com/openalto

IETF Plenary 113
ALTO WG Session
3/23/2022
Table of Contents

- ALTO Architecture
- ALTO Code Base Project
- Project Management and Approach
- ALTO Deployments
- IETF Hackathon 113: ALTO Project
ALTO Architecture
IETF Application Layer Traffic Optimization WG

- IETF ALTO Charter:
  - Standardizing a network API that applications can query to get the state of the network and to use this information to optimize their performance.
- IETF ALTO history of applications / use cases:
  - Peer-to-peer applications → CDNs → {Cloud, Edge-computing, 5G, V2X, XR, Science}
- Participation from a variety of carriers, vendors and universities:
  - Nokia, Ericsson, Verizon, Comcast, Telefonica, Deutsche Telekom, Huawei, China Telecom, Google, Cisco, Samsung, Qualcomm, Yale University, Tsinghua/Sichuan/Tongji Universities, etc.
IETF ALTO Architecture

https://www.rfc-editor.org/rfc/rfc7285.txt

Figure 1: Basic ALTO Architecture
IETF ALTO Architecture

https://www.rfc-editor.org/rfc/rfc7285.txt

Figure 1: Basic ALTO Architecture
IETF ALTO Architecture

https://www.rfc-editor.org/rfc/rfc7285.txt
IETF ALTO Architecture

https://www.rfc-editor.org/rfc/rfc7285.txt
ALTO Code Base Project
IETF ALTO Code Base

IETF ALTO server stack

- ALTO Services
  - Network map
  - Cost map
  - Property map
  - (...)

Northbound API

Southbound API

IETF ALTO Client

Open-source vendor independent and close-source vendor specific code

Open-source vendor independent code
Project Management and Approach
The ALTO Code Base Project aims at providing a parallel track to the WG’s standardization effort towards implementing the features introduced in the latest RFCs.

IETF Hackathons will be used as 3-checkpoints a year to test interoperability, demo latest standard capabilities and identify issues and improvements for standardization.

Identify and build production, open-source environments for use cases and deployment ("lean startup") to help steer ALTO standardization.
Project Management and Approach

- Adopting industry standard Agile / Scrum methodologies to ensure:
  - Productivity
  - Quality
  - Participation
  - Lean startup, rough consensus running code

- Invoke community participation to develop the ALTO Code Base:
  - Two roles: developers and mentors.
  - Mentors are usually experienced members of the IETF ALTO WG
  - Developers usually come from universities and the industry in general

- Project management resources:
  - Repo: https://github.com/openalto/
  - Project Scrum Dashboard (IETF Hackathon 113): https://github.com/orgs/openalto/projects/1
Scrum dashboard:

- **Todo**
  - ietf-hackathon #1  ... Registration to FRP Kubernetes platform
  - ietf-hackathon #7  ... Document comment TODO items on skeleton code for ALTO python library
  - alto #1  ... Create an basic ALTO client
  - ietf-hackathon #10  ... Documenting demo environment and assigning tasks
  - ietf-hackathon #14  ... Ask Radu feedback on adding ALTO switch to cli + config file

- **In Progress**
  - ietf-hackathon #2  ... Document environment set up for admins
  - ietf-hackathon #3  ... Document environment set up for developers
  - ietf-hackathon #6  ... Integrate Rucio demo containers with Mininet
  - ietf-hackathon #12  ... Update Rucio documentation
  - ietf-hackathon #8  ... Write the 113 ALTO Hackathon Story

- **In Code Review**
  - ietf-hackathon #13  ... Extend rucio CLI to support ALTO switch
  - ietf-hackathon #24  ... Demo 2: Write JSON parser for ALTO flow cost service
  - ietf-hackathon #25  ... Demo 2: Parse G2-mininet configuration files to obtain A (routing matrix) and c (capacity vector) and RTT
  - ietf-hackathon #26  ... Demo 2: Generate flow F from ALTO request query
  - ietf-hackathon #27  ... Demo 2: Call solver passing A, c and F to obtain throughput estimates for each flow

- **Done**
  - alto #2  ... Add unit tests for client with mock ALTO server
  - ietf-hackathon #9  ... Simple HTTP Query to ALTO server
  - ietf-hackathon #10  ... Extend G2-Mininet to support container
  - ietf-hackathon #16  ... Figure out how to monitor the tx rate of Rucio download
ALTO Deployments
ALTO Deployments

• Current implementations/deployments:
  • Wiki list of implementations: https://trac.ietf.org/trac/alto/wiki/Impl
  • Examples:
  • Forthcoming new deployments (work in progress):
    • Pacific Research Platform
    • CERN Rucio
    • UCSD 5G
    • NY City Cosmos 5G
    • ESnet
IETF Hackathon 113: ALTO Project
Summary

• Demo setup
  • Network is simulated by Mininet
  • Applications are running as virtual hosts/containers
  • The demo environment is packed as multiple containers for future enhancement

• Capabilities
  • Source selection based on network map and cost map
  • Costs computed as inverse of bandwidth

• Code base
  • An open-source python client library is developed
  • Add the ALTO-based replica selection support to the Rucio scientific data management system
Goals in the 113 Hackathon

• Use ALTO Cost Maps to optimize dataset transfers for rucio, the main data management tool for LHC and other large projects.

• Integrate ALTO Northbound Interface with Rucio to provide visibility and achieve better performance.

• Show that it works.

* ESnet / LHCONE source: https://www.es.net/about/
RFCs Involved During the Hackathon

  https://datatracker.ietf.org/doc/rfc7285/
- I-Draft ALTO Extension: Flow-based Cost Query
- I-Draft ALTO Performance Cost Metrics
What Got Done

- Implementation of an ALTO Client in Python (RFC 7285)
- Integration with CERN Rucio replica download
  - Submitted pull request to Rucio Project:
    - [https://github.com/rucio/rucio/pull/5364](https://github.com/rucio/rucio/pull/5364)
- 3 Demos [https://github.com/openalto/ietf-hackathon/issues/8](https://github.com/openalto/ietf-hackathon/issues/8)
  - [D1] Single-flow replica node selection using ALTO BW Cost Map
  - [D2] ALTO Estimator: Multi-flow BW prediction
  - [D3] ALTO Scheduler: SLA-constrained multi-flow node selection
- Southbound ALTO integration with SDN:
  - Mininet/Pox, OpenDaylight
- Scrum dashboard: [https://github.com/orgs/openalto/projects/1/views/1](https://github.com/orgs/openalto/projects/1/views/1)
- Lots of really interesting architecture discussions
## ALTO Metrics


<table>
<thead>
<tr>
<th>Metric</th>
<th>Definition in this doc</th>
<th>Semantics Based On</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-way Delay</td>
<td>Section 3.1 Base: [RFC7471, 8570, 8571] sum Unidirectional Delay</td>
<td></td>
</tr>
<tr>
<td>Round-trip Delay</td>
<td>Section 3.2 Base: Sum of two directions from above</td>
<td></td>
</tr>
<tr>
<td>Delay Variation</td>
<td>Section 3.3 Base: [RFC7471, 8570, 8571] sum of Unidirectional Delay Variation</td>
<td></td>
</tr>
<tr>
<td>Loss Rate</td>
<td>Section 3.4 Base: [RFC7471, 8570, 8571] sum Unidirectional Link Loss</td>
<td></td>
</tr>
<tr>
<td>Residual Bandwidth</td>
<td>Section 4.2 Base: [RFC7471, 8570, 8571] min Unidirectional Residual BW</td>
<td></td>
</tr>
<tr>
<td>Available Bandwidth</td>
<td>Section 4.3 Base: [RFC7471, 8570, 8571] min Unidirectional Avail. BW</td>
<td></td>
</tr>
<tr>
<td>Utilized Bandwidth</td>
<td>Section 4.4 Base: [RFC7471, 8570, 8571] max Unidirectional Utilized BW</td>
<td></td>
</tr>
<tr>
<td>TCP Throughput</td>
<td>Section 4.1 [RFC8312bis]</td>
<td></td>
</tr>
<tr>
<td>Hop Count</td>
<td>Section 3.5 [RFC7285]</td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Cost Metrics Defined in this Document.
## IETF ALTO Code Base

### Container Environment Description

**docker-compose-with-rucio.yml**

It provides the following containers:

<table>
<thead>
<tr>
<th>Service/Container Name</th>
<th>Scope</th>
<th>Short Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mininet</td>
<td>Mininet</td>
<td>Extended mininet container for network simulation</td>
</tr>
<tr>
<td>sfow</td>
<td>Mininet</td>
<td>Network flow sampling and monitoring</td>
</tr>
<tr>
<td>odl</td>
<td>ALTO</td>
<td>ALTO server over OpenDaylight controller</td>
</tr>
<tr>
<td>rucio</td>
<td>Rucio</td>
<td>Basic Rucio components</td>
</tr>
<tr>
<td>ruciodb</td>
<td>Rucio</td>
<td>ProgresQL database for Rucio</td>
</tr>
<tr>
<td>fts</td>
<td>Rucio</td>
<td>File transfer system for Rucio</td>
</tr>
<tr>
<td>ftsdb</td>
<td>Rucio</td>
<td>MySQL database for fts</td>
</tr>
<tr>
<td>activemq</td>
<td>Rucio</td>
<td>Message queue for Rucio and fts scheduler</td>
</tr>
<tr>
<td>xrd(i)</td>
<td>Storage</td>
<td>Xrootd storage node as Rucio RSE</td>
</tr>
<tr>
<td>ssh(i)</td>
<td>Storage</td>
<td>SSH storage node as Rucio RSE</td>
</tr>
<tr>
<td>minio</td>
<td>Storage</td>
<td>MinIO storage node as Rucio RSE</td>
</tr>
<tr>
<td>graphite</td>
<td>Monitoring</td>
<td>Graphtie server for Rucio internal monitoring</td>
</tr>
</tbody>
</table>

**docker-compose-with-rucio-monit.yml**

Besides the containers above, it also provides the following additional containers:

<table>
<thead>
<tr>
<th>Service/Container Name</th>
<th>Scope</th>
<th>Short Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>elasticsearch</td>
<td>Monitoring</td>
<td>ElasticSearch engine for more complicated monitoring data processing</td>
</tr>
<tr>
<td>kibana</td>
<td>Monitoring</td>
<td>Kibana dashboard for customized visualization</td>
</tr>
<tr>
<td>logstash</td>
<td>Monitoring</td>
<td>Data processing pipeline for ElasticSearch</td>
</tr>
<tr>
<td>grafana</td>
<td>Monitoring</td>
<td>Another dashboard for visualization</td>
</tr>
</tbody>
</table>
Demo 1: Single-flow Replica Node Selection Using ALTO BW Cost Map

1. Look up the host by finding the longest-prefix match

2. Bandwidth between hosts as the ALTO cost

3. Replicas sorted by bandwidth from ALTO

4. Download from the replica with the smallest cost (0.57 MBps = 4.56 Mbps)
Demo 1: Single-flow Replica Node Selection Using ALTO BW Cost Map

4-5X speed up by using ALTO network visibility
Demo 1: Single-flow Replica Node Selection Using ALTO Latency Cost Map

1. Look up the host by finding the longest-prefix match

2. One-way latency between hosts as the ALTO cost

3. Replicas sorted by latency from ALTO

4. Download from the replica with the smallest latency
Demo 2: ALTO Estimator: Multi-flow Tput/BW Prediction

TCP throughput computed from network topology and TCP throughput modeling for bulk flows ([G2, PROPHET]).
Demo 3: ALTO Scheduler: SLA-constrained Multi-flow Node Selection

- Problem
  - Multiple datasets replicated on multiple hosts.
  - Rucio dataset automation workflow requires a given SLA (e.g., time-bound constrained data transfers)

- Demo (partially finished)
  - ALTO ESTIMATOR (Demo) provides cost map predicting replication throughput
  - ALTO SCHEDULER searches among possible download configuration one that guarantees the SLA requirement.

Example:
- Goal: download datasets file1, file3
- Replica selections: red versus yellow. Pick one replica that satisfies the SLA.
Wrap Up and Looking Forward

- **ALTO WG Contact:**
  - IETF ALTO WG: [https://datatracker.ietf.org/wg/alto/about/](https://datatracker.ietf.org/wg/alto/about/)

- **ALTO Code Base Project:**
  - Repo: [https://github.com/openalto/](https://github.com/openalto/)
  - IETF Hackathon 113 ALTO Scrum Dashboard: [https://github.com/orgs/openalto/projects/1/views/1](https://github.com/orgs/openalto/projects/1/views/1)

- **Potential Tasks/demos at IETF 114 hackathon:**
  - Finishing Demo 3, ALTO with HTTP/2
  - ALTO for multiple experiments for Rucio and more production use cases

- **Want to contribute to OpenALTO as a developer?** Reach us out: jros at qti.qualcomm.com

Looking forward to seeing you in Philadelphia!