ALTO YANG Model

draft-shi-alto-yang-model-03

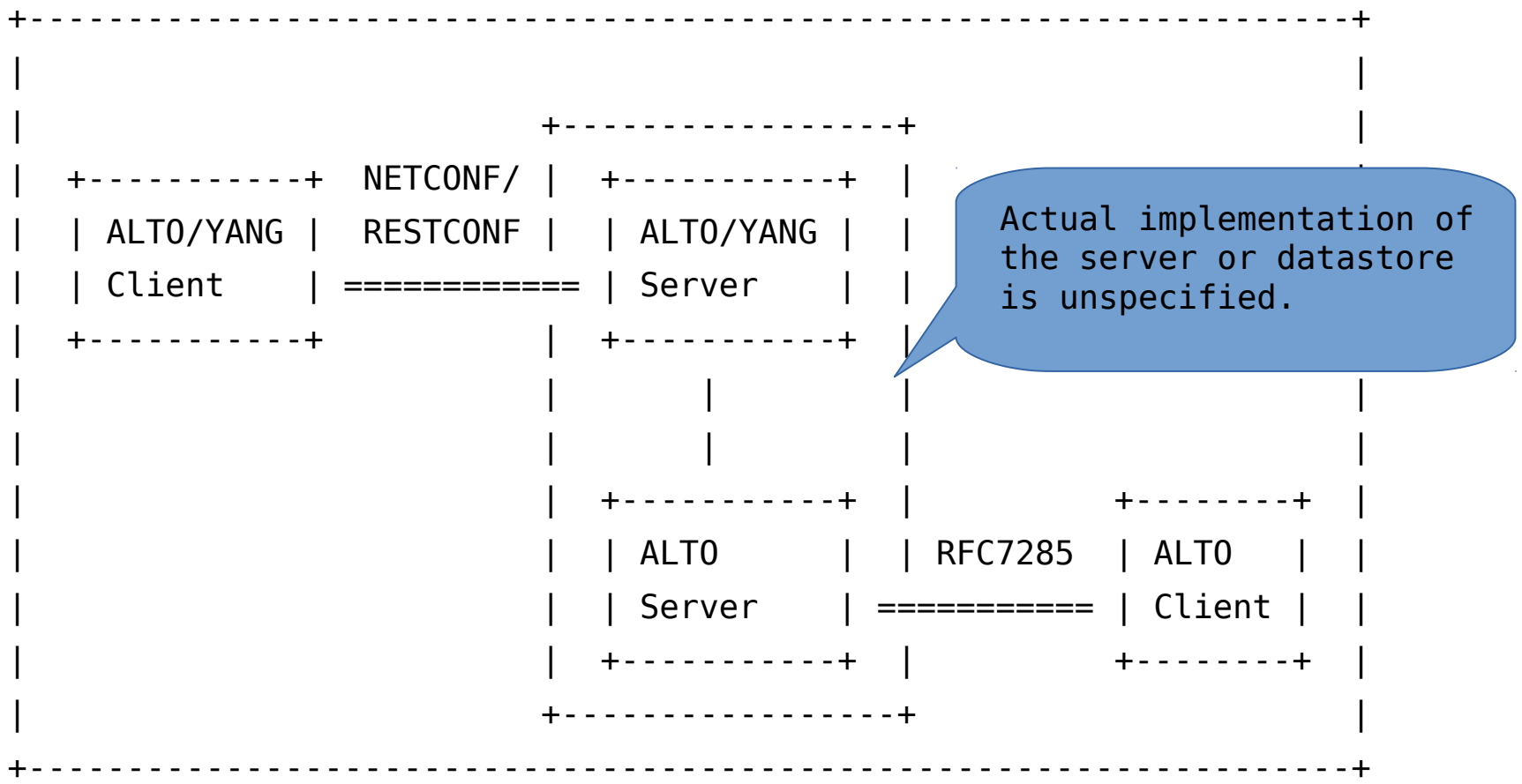
Xiao Shi (xiao.shi@yale.edu)
Y. Richard Yang (yry@cs.yale.edu)
M. Scharf (michael.scharf@alcatel-lucent.com)

IETF 92
March 26, 2015

Motivation

- Information resources on ALTO servers are read-only, would like ability to modify information resources.
  - Standard interface, allowing others to modify: e.g., authorized management apps, provisioning sources, ALTO clients, other ALTO servers, etc.
  - Easy to implement using existing YANG tools
  - Security and authorization
- Management of ALTO server via NETCONF server and YANG model
  - YANG abstracts persistence-layer

ALTO/YANG Components



YANG Model

- Models datastore for ALTO/YANG server
  - Allows NETCONF operations on modeled resources: <get-config>, <edit-config>, etc.
  - NETCONF protocol messages are encoded in XML, but the ALTO/YANG server need not store data in XML.
  - NETCONF uses persistent connections and underlying transport protocols such as SSH/TLS, providing the desired authentication, confidentiality, etc.
- Given the goal is only to modify resources, we only model IRD, network maps, cost maps, endpoint properties
  - Will not directly provide other ALTO services: e.g., filtered maps, endpoint cost service (underlying data is unspecified in RFC7285)

Example: Cost Map YANG Model

```
+--rw cost-maps
|  +--rw cost-map* [resource-id]
|     +--rw resource-id    alto:resource-id
|     +--rw tag            alto:tag-string
|     +--rw meta
|     |  +--rw dependent-vtags* [resource-id tag]
|     |  |  +--rw resource-id    resource-id
|     |  |  +--rw tag            tag-string
|     |  +--rw cost-type
|     |     +--rw cost-mode      cost-mode
|     |     +--rw cost-metric    cost-metric
|     |     +--rw description?   string
|     +--rw map* [src]
|        +--rw src          alto:pid-name
|        +--rw dst-costs* [dst]
|           +--rw dst     alto:pid-name
|           +--rw cost
```

Each resource is only keyed by resource-id, but MUST have a tag.

Same semantics as RFC7285 protocol messages, slightly different encoding.

Example: Update Cost Map via NETCONF

```
<rpc message-id=SEQ-NUM xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
      <resources xmlns="urn:ietf:params:xml:ns:yang:alto-service">
        <cost-maps>
          <cost-map>
            <resource-id>myCostMap1</resource-id>
            <map>
              <src>PID3</src>
              <dst-costs>
                <dst>PID2</dst>
                <cost>10</cost>
              </dst-costs>
            </map>
          </cost-map>
        </cost-maps>
      </resources>
    </config>
  </edit-config>
</rpc>
```

change the cost from PID3 to PID2 from 15 to 10 in myCostMap1

Example: Endpoint Properties YANG Model

```
+--rw endpoint-property-map
   +--rw meta
   |  +--rw dependent-vtags* [resource-id tag]
   |     +--rw resource-id    resource-id
   |     +--rw tag            tag-string
   +--rw endpoint-properties* [endpoint]
      +--rw endpoint      typed-endpoint-address
      +--rw properties* [property-type]
         +--rw property-type    endpoint-property-type
         +--rw property         endpoint-property-value
```

Example: Update Endpoint Properties

```
<rpc message-id=SEQ-NUM xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target><running/></target>
    <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
      <resources xmlns="urn:ietf:params:xml:ns:yang:alto-service">
        <endpoint-property-map><endpoint-properties>
            <endpoint>ipv4:192.0.2.34</endpoint>
            <properties>
              <property-type>my-default-network-map.pid</property-type>
              <property>PID2</property>
            </properties>
        </endpoint-properties></endpoint-property-map>
        <endpoint-property-map><endpoint-properties>
            <endpoint>ipv4:203.0.113.129</endpoint>
            <properties>
              <property-type xc:operation="create">
                priv:ietf-example-prop
              </property-type>
              <property xc:operation="create">2</property>
            </properties>
        </endpoint-properties></endpoint-property-map>
      </resources>
    </config>
  </edit-config>
</rpc>
```

Change the “my-default-network-map.pid” property of “ipv4:192.0.2.34” to “PID2”

Add “priv:ietf-example-prop” property to “ipv4:203.0.113.129”

Note the sub-operations: create, delete, remove, merge (default)

Consistency Discussions: Resource Tagging

- ALTO resources are keyed by `resource-ids` only in the YANG model, `tag` should be generated by the server and hence should be read-only.
  - All current versions of the resources are in the config datastore (r/w) whereas all history versions are in the state data (read-only)
  - Each time an ALTO resource updates, the ALTO/YANG server MUST generate a new tag (e.g., a persistent sequence number) and assign to the resource.
  - ALTO/YANG server MUST check that update operations do not include writes to `tag` (reply with <rpc-error> if they do).
- Edits to `dependency-vtags` fields MUST use NETCONF <lock>/<unlock> mechanisms to ensure atomicity of updates.

Consistency: ALTO/YANG & ALTO Server

- The ALTO/YANG server keeps a transactional log of all updates. It does not push to the ALTO server datastore until the updates reach a consistent state.
- The ALTO/YANG puts TTL on received updates, and rolls back once TTL is exceeded.

Experiences: Implementing ALTO/YANG

- Using existing tools
  - Automatic code-generation based on YANG models defined in this document
  - Basic functionality under OpenDayLight: ~1k lines of manually written code

Next Steps

- Specify authorization scheme
- Extend charter to include YANG model?

Thank you