Peer Mount

Eric Voit
Alexander Clemm

5-Nov-2015

Overview

- Peer-mount was first introduced in 2013
- Allows to super-impose new structures on top of existing YANG models
- Original purpose: Allow YANG Datastores to reference information in remote datastores
  - Insert (remote) subtrees under a mount point in a datastore
    - Mount client: a YANG server that maintains the mounted "view"
    - Mount server: the original "authoritative" owner of the data
    - For on-demand object access, mount server does not need to be aware of mount client
  - Defines an alternative path to access data nodes
    - Clients of the YANG server with mounted structure access it like "native" information
- Original draft emphasized remotability of data
  - YANG Server allows its clients to access data that is conceptually federated across a network
    - (Note: Peer-mount is also the basis for MD-SAL in Open Daylight, and is now proven/robust)
  - After initial discussions failed to ignite enthusiasm, we left the drafts "low-key"
  - However, mount points could also be defined for local data → renewed interest

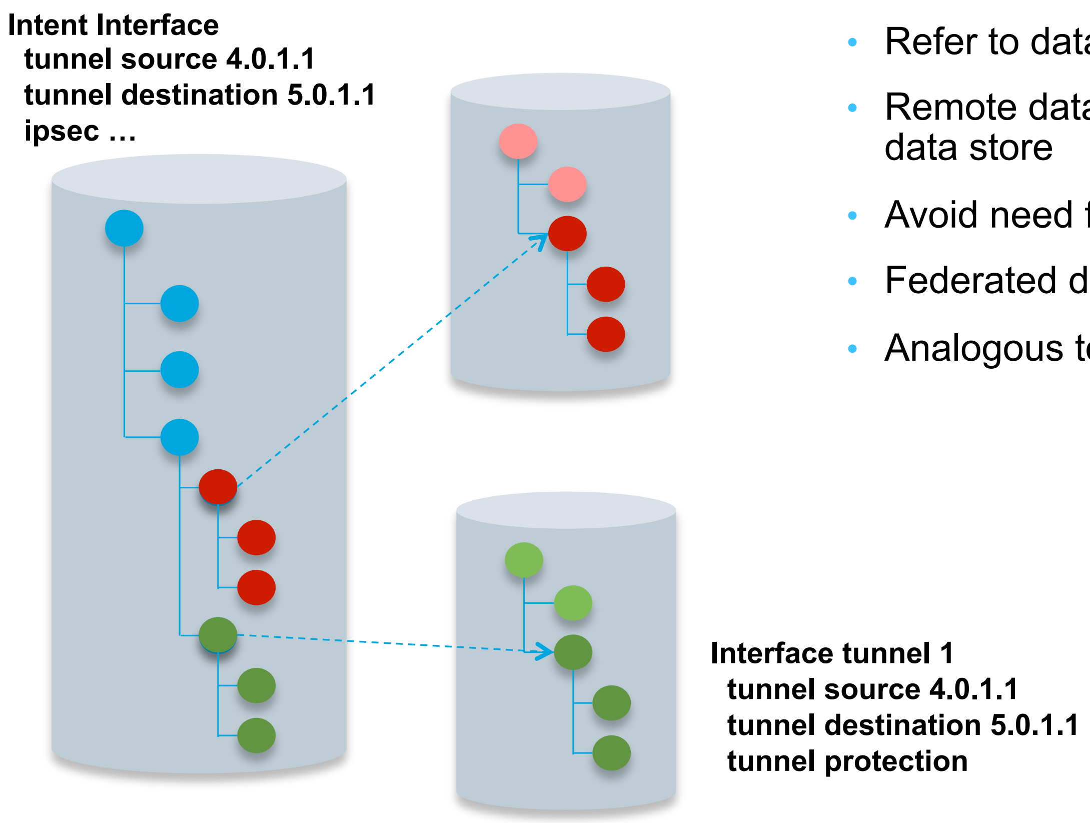
Current draft status

Requirements **draft-voit-netmod-peer-mount-requirements-03** (with Sander Mertens, recently refreshed)

Technical spec **draft-clemm-netmod-mount-03** (with Jan Medved, needs refreshing, recently expired)

- Updates
  - Renamed “Peer-Mount” to “YANG-Mount”
  - Separation of two complementary yet orthogonal concepts:
    - Alias-mount: inserting (“mounting”) a subtree under a mountpoint, for an alternative path within a device
    - Peer-mount: the ability to mount a subtree that resides on a remote system
  - Updated structure allows to address alias-mount first and leave peer-mount for later
    - Peer-mount is an extension of alias mount, in which subtrees can be remote
- Alias-mount is conceptually simpler yet still useful
  - Expose YANG objects via alternative structures, referenced via alternative application-intuitive paths
  - Doesn’t require mirroring or replication of the underlying data
- Renewed interest
  - Alias-Mount provides the ability to impose alternative structures over existing models without redefinition
  - This is what is needed e.g. in the context of Open Config

Mount Concept – peer mount



- Refer to data nodes / subtrees in remote datastores
- Remote data nodes conceptually treated as part of local data store
- Avoid need for data replication and orchestration
- Federated datastore - treat network as a system
- Analogous to Network File System

Datastore mount concept

- Mount client
  - Contains mount points at which to attach (local or remote) subtrees into data tree
  - Requests whose scope contains mounted data are redirected to the authoritative data
- Mount server
  - Authoritative owner of the data
  - May not be aware that mounting occurs (mount client is “just another application”)
- YANG module defines YANG mountpoint extensions and data model for mountpoint management
  - Date models can be defined that use the extensions to impose their own “super-structure”
- Notes
  - Caching optimizations possible (e.g. YANG pub/sub)
  - Circular mounting prohibited
  - Focus on data nodes (not notifications)

Usage example

```
rw controller-network
    +-- rw network-elements
        +-- rw network-element [element-id]
            +-- rw element-id
            +-- rw element-address
            |   +-- ...
            +-- M interfaces
```

Module structure

```
...
 list network-element {
     key "element-id";
     leaf element-id {
          type element-ID;
     }
     container element-address {
          ...
     }
     mnt:mountpoint "interfaces" {
         mnt:target "./element-address";
         mnt:subtree "/if:interfaces";
     }
  }
...
```

Mountpoint declaration

- YANG module defines YANG mount extensions + data model for mountpoint management
- YANG extensions:

  Mountpoint: Defined under a containing data node (e.g. container, list)

  Target: References data node that identifies remote server [peer-mount only]

  Subtree: Defines root of remote subtree to be attached

```
<network-element>
   <element-id>NE1</element-id>
    <element-address> .... </element-address>
    <interfaces>
       <if:interface>
           <if:name>fastethernet-1/0</if:name>
           <if:type>ethernetCsmacd</if:type>
           <if:location>1/0</if:location>
            ...
       </if:interface>
        ...
```

Instance information

Alias mount example

```
rw my-container
    +-- rw sub-container
        +-- M interfaces


...
```

Module structure

```

container my-container {
  container subcontainer {
     mnt:mountpoint "interfaces" {
          mnt:subtree "/if:interfaces";
     }
  }
...
```

Mountpoint declaration

Mount point is local

```
<my-container>
   <sub-container>
   <interfaces>
       <if:interface>
           <if:name>fastethernet-1/0</if:name>
           <if:type>ethernetCsmacd</if:type>
           <if:location>1/0</if:location>
            ...
       </if:interface>
        ...
```

Instance information

Mountpoint management

```
rw mount-server-mgmt
  +-- rw mountpoints
  |   +-- rw mountpoint [mountpoint-id]
  |       +-- rw mountpoint-id  string
  |       +-- rw mount-target
  |       |   +--: (IP)
  |       |   |    +-- rw target-ip  yang:ip-address
  |       |   +--: (URI)
  |       |   |    +-- rw uri  yang:uri
  |       |   +--: (host-name)
  |       |   |    +-- rw hostname  yang:host
  |       |   +-- (node-ID)
  |       |   |    +-- rw node-info-ref  mnt:subtree-ref
  |       |   +-- (other)
  |       |        +-- rw opaque-target-id  string
  |       +-- rw subtree-ref  mnt:subtree-ref
  |       +-- ro mountpoint-origin enumeration
  |       +-- ro mount-status  mnt:mount-status
  |       +-- rw manual-mount? empty
  |       +-- rw retry-timer? uint16
  |       +-- rw number-of-retries? uint8
  +-- rw global-mount-policies
      +-- rw manual-mount? empty
      +-- rw retry-time? uint16
      +-- rw number-of-retries? uint8
```

- Mountpoints can be system-administered
- Applications&users are not exposed to this
- System administration can add bindings

  Update on-demand, periodic, on-change
- Not shown:

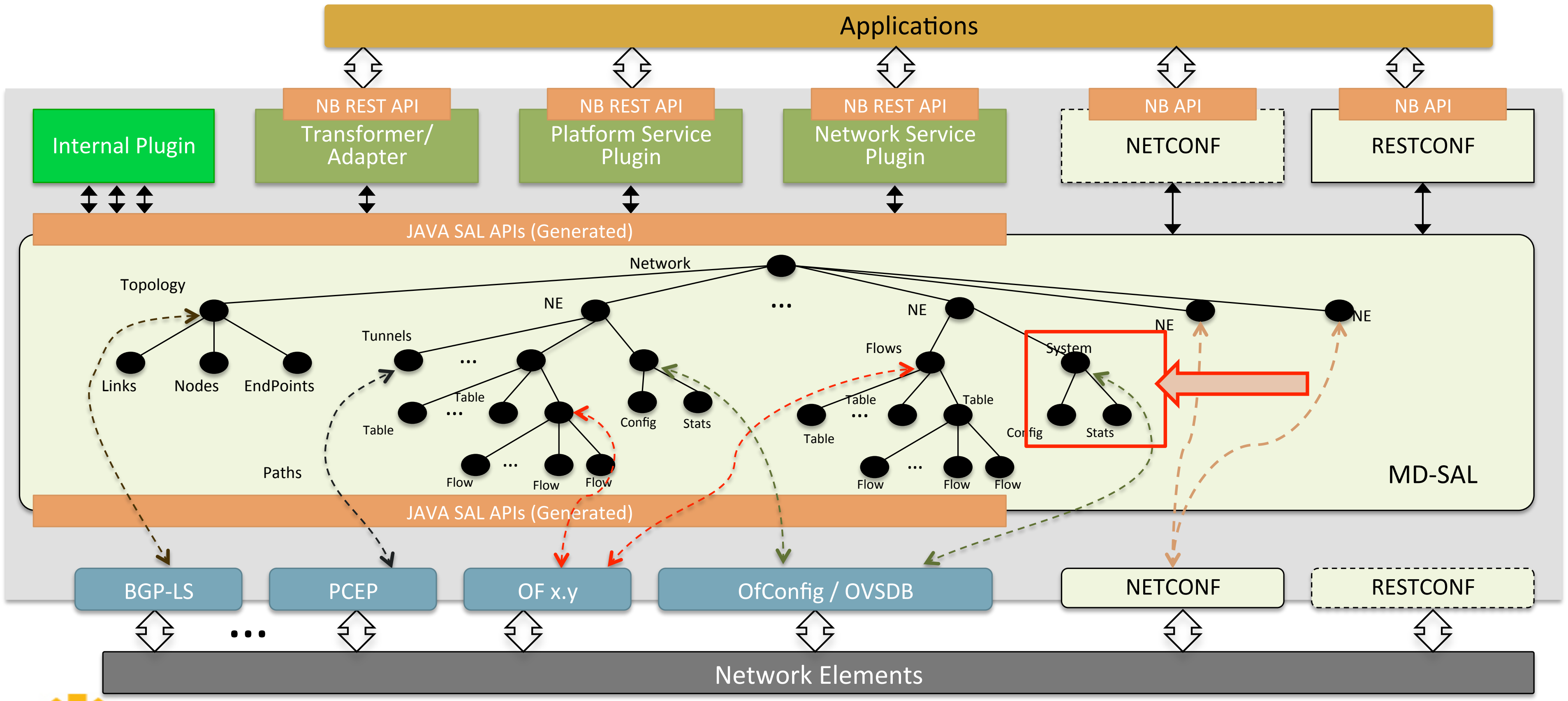
  Mount bindings - data update subscriptions
- Model needs updating to distinguish alias and peer mount

```
RPCs for manual mount, unmount
```

Application example:
Network controller

- Provide consolidated network view to applications north of controller without replicating information from controlled nodes
  - Mount information from devices and interfaces below nodes inventory
  - Allow to change containment hierarchy
    - E.g. place top level “system” information underneath list of nodes
    - Device and network abstractions complement one another in same data tree
  - No need for replicated device models
  - Dynamic discovery and support of new device features
    - Controller not a bottle neck for the adoption of new feature

Open Daylight - Model-Driven SAL



OPENDAYLIGHT

www.opendaylight.org

Next steps

- Update technical draft to better reflect distinction between alias and peer mount
  - Peer mount as an extension to alias mount
- Explore new alias mount use cases further
- Investigate support for notifications referring to objects under their aliased name
- Solicit feedback from working group