I2RS Topology Example

Sue Hares
Current Datastores

config true;

config false;

All operational data exists alongside config=true but there is no datastore defined for config=false data nodes
module ietf-network {
    grouping network-ref {
        leaf-network-ref {
            type leafref {path “network/network-id”;}
        }
    }
    grouping node-ref {
        leaf-network-ref {
            type leafref {
                path “network/network-id-current()/…/ntework-ref”+ +”/node/node-id”;}
            uses network-ref;
        }
        list network {
            key “network-id”
            leaf network-id {type network-id; }
            leaf server-provided {type boolean; config false; }
        list supporting-network {
            key “network-ref”;
            leaf network-ref {type leafref; path “/network/node/node-id”; }
        }
    }
    list node {
        key “node-id”;
        leaf node-id {type node-id; }
        list supporting-node {
            key “network-ref node-ref”;
            leaf network-ref {
                type leafref {path “../../..//supporting-network/network-ref”
                }
            leaf node-ref {
                type leafref {
                    path “/network/node/node-id”; }
            }
        }
    }
    Network 1:  
    node: 1  
    nework-1  
    node, 1,2, 3
module ietf-network-topology {
    augment "'/nd:network"
    list link {
        key "link-id";
        container source {
            leaf source-node {
                type leafref { path "./././nd:node/nd:node-id"; }
                mandatory true;
            }
            leaf source-tp {
                type leafref { path "./././nd:node/[nd:node-id=current()/.]/
                    + "source-node]/ termination-point/tp-id"; }
            }
        } /container source
        container destination {
            leaf dest-node {
                type leafref { path "./././nd:node/nd:node-id"; }
                mandatory true;
            }
            leaf dest-tp {
                type leafref { path "./././nd:node/[nd:node-id=current()/.]/
                    + "source-node]/ termination-point/tp-id"; }
            }
        } /container destination
        leaf link-id inet-uri
    list supporting-links {
        key: 1 2
        network-ref
        link-ref
    }
} augment

Network 1:
node: 1
network-1
link [id-5]
source {
    source-node: 1
tp-id:1
}
destination {
    dest-node: 2
tp-id:4
}
leaf link-id inet-uri
list supporting-links
key: 1 2
network-ref
link-ref

Thermostat Model Equivalent

Network 1
Node 1
  igp node-attributes (router-id 192.1.1.2,
     list of prefix: 128.1/16, 128.2)
  Link id 5
  Node 1, 2
  node 2, 4
  supporting links
     1-3, 1-4
  igp-link-attributes (link-flag up; metric 5);
config true;
config false;

Scheduler
Client

intended config

Topology
OSPF topology
ISIS topology
Topology plus ephemeral link

Network 1
Node 1: Link id 5
Node 1, 2; node 2, 4
supporting links 1-3, 1-4
running datastore

Scheduler Client

IPS replaces link

intended config

config true;

ephemeral datastore

network
Node 1: Link id 5
Node 1,3 node, 2,4
Supporting 1-3, 1-5

config false;

Topology link is
Replaced by
IPS route
RESTCONF Example

RESTCONF Running Datastore Edit (Src 1,2 to 2,4)


RESTCONF Ephemeral Datastore Edit of config=true changes to src 1,3

{ “source-tp”:3 }
Open issues

- Large scale uploads (writes)
  - PUT? RPC?
- Model has:
  - Native: learned from OSPF/ISIS or BGP LS
  - Combined (learned + rules of combination)
  - Added from I2RS Client
- Should we have different protocol for large outputs?