YANG usage in protocols

Vinod Kumar S

HUAWEI

onos
Open Network Operating System

Agenda

- Sample YANG
- Encoding YANG modeled Data (NETCONF)
- Encoding YANG modeled Data (RESTCONF)
- Logical tree store alternatives
- Logical addressing of data and operations
- Logical addressing relation with protocols
- Issue in exposing device YANG as ONOS NBI (case 1)
- Solution alternatives
- Next step

Sample YANG

Sample YANG

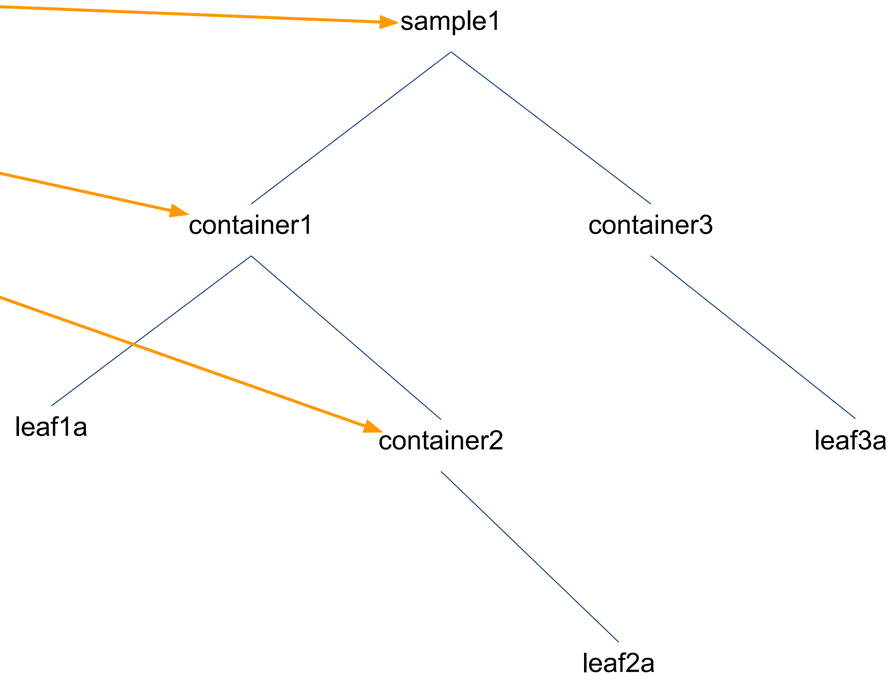
```
module sample1 {
    yang-version 1;
    namespace urn:sample1;
    prefix s1;
    revision 2016-11-11;
    container container1 {
        leaf leaf1a{
           type string;
        }
        container container2{
            leaf  leaf2a{
                type string;
            }
        }
    }

    container container3 {
        leaf leaf3a{
           type string;
        }
    }


    rpc set-node-limit {
        input {
            leaf node-limit {
                type int16;
            }
        }
    }

    notification node-limit-reached {
        leaf node-limit {
            type int16;
        }
    }
}
```

Logical Data tree



Encoding in NETCONF

Sample YANG

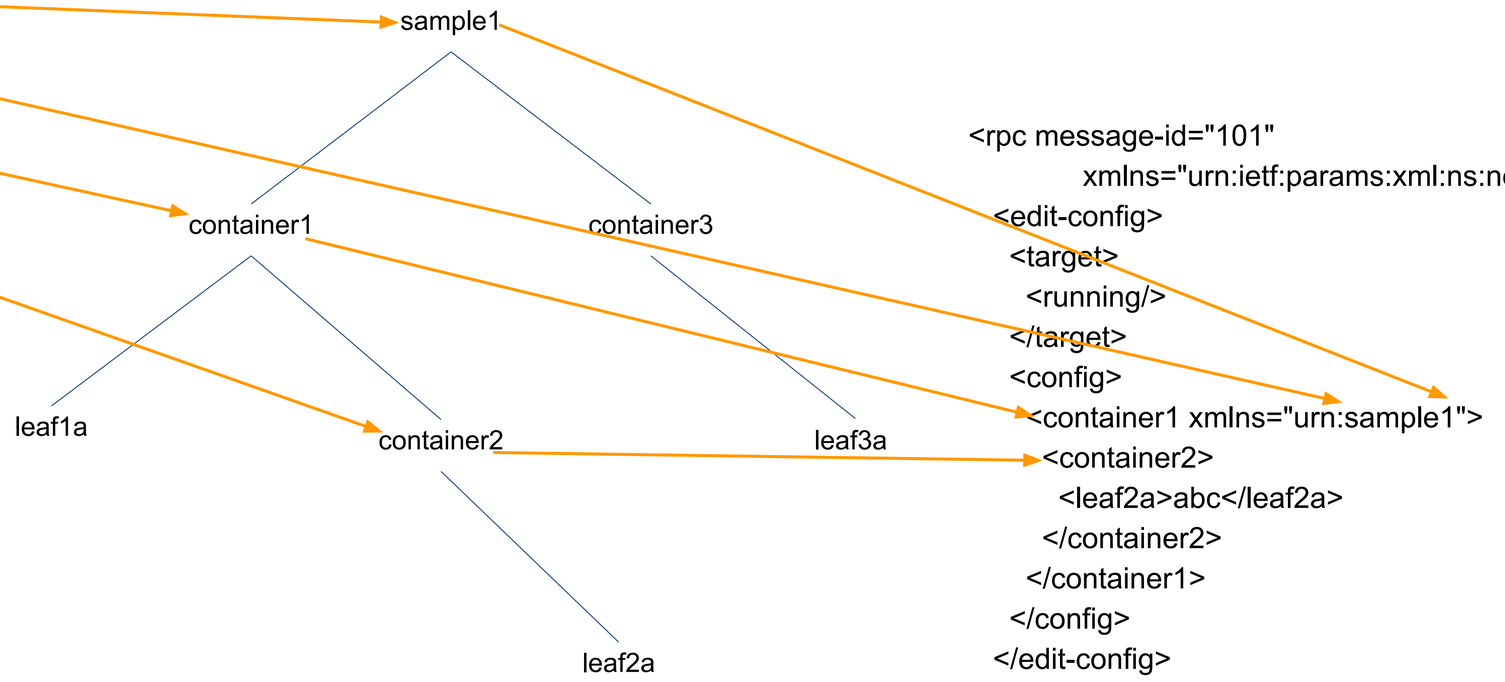
```
module sample1 {
    yang-version 1;
    namespace urn:sample1;
    prefix s1;
    revision 2016-11-11;
    container container1 {
        leaf leaf1a{
           type string;
        }
        container container2{
            leaf  leaf2a{
                type string;
            }
        }
    }

    container container3 {
        leaf leaf3a{
           type string;
        }
    }


    rpc set-node-limit {
        input {
            leaf node-limit {
                type int16;
            }
        }
    }

    notification node-limit-reached {
        leaf node-limit {
            type int16;
        }
    }
}
```

Logical data tree




```
<rpc message-id="101"
        xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
     <running/>
    </target>
    <config>
     <container1 xmlns="urn:sample1">
      <container2>
        <leaf2a>abc</leaf2a>
      </container2>
     </container1>
    </config>
  </edit-config>
</rpc>
```

Encoding in NETCONF

5.1.2.1. NETCONF XML Encoding

NETCONF is capable of carrying any XML content as the payload in the <config> and <data> elements. The top-level nodes of YANG modules are encoded as child elements, in any order, within these elements. This encapsulation guarantees that the corresponding NETCONF messages are always well-formed XML documents.

For example, an instance of:

```
module example-config {
  yang-version 1.1;
  namespace "urn:example:config";
  prefix "co";

  container system { ... }
  container routing { ... }
}
```

could be encoded in NETCONF as:

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
     xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <system xmlns="urn:example:config">
        <!-- system data here -->
      </system>
      <routing xmlns="urn:example:config">
        <!-- routing data here -->
      </routing>
    </config>
  </edit-config>
</rpc>
```

Encoding rules as per YANG

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <container1 xmlns="urn:sample1">
        <container2>
          <leaf2a>abc</leaf2a>
        </container2>
      </container1>
    </config>
  </edit-config>
</rpc>
```

Operations on YANG modeled data is part of protocol (NETCONF). Protocol is aware of following
- config type
- operation

Protocol (NETCONF) is agnostic of modeled data. YANG imposes the rules encoding the modeled data.

Encoding in RESTCONF

Sample YANG

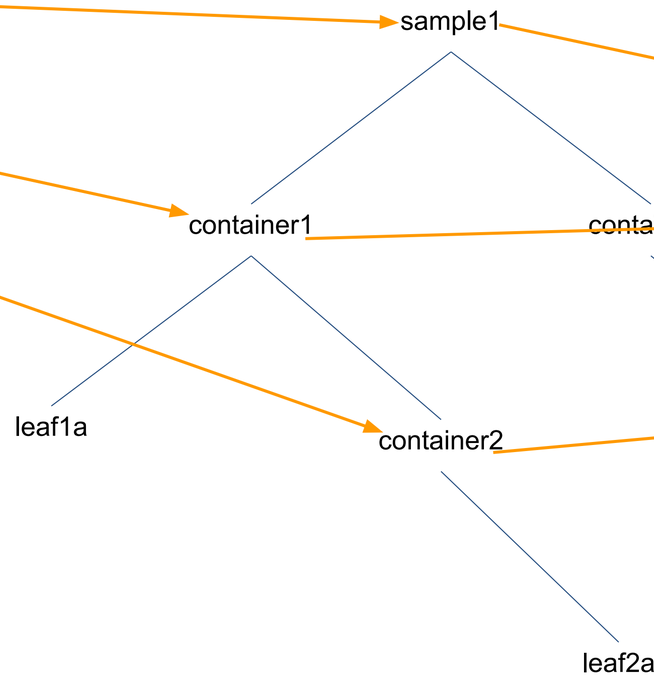
```
module sample1 {
    yang-version 1;
    namespace urn:sample1;
    prefix s1;
    revision 2016-11-11;
    container container1 {
        leaf leaf1a{
           type string;
        }
        container container2{
            leaf  leaf2a{
                type string;
            }
        }
    }

    container container3 {
        leaf leaf3a{
           type string;
        }
    }


    rpc set-node-limit {
        input {
            leaf node-limit {
                type int16;
            }
        }
    }

    notification node-limit-reached {
        leaf node-limit {
            type int16;
        }
    }
}
```

Logical data tree




```
PUT /restconf/data/sample1:container1/ HTTP/1.1
Host: example.com
Content-Type: application/yang-data+json

{
  "sample1:container2" :
    {
      "leaf2a" : "abc"
    }
}
```

Encoding in RESTCONF

3.3. API Resource

The API resource contains the RESTCONF root resource for the RESTCONF datastore and operation resources. It is the top-level resource located at {+restconf} and has the media type "application/yang-data+xml" or "application/yang-data+json".

YANG Tree Diagram for an API Resource:

```
+---- {+restconf}
      +---- data
      | ...
      +---- operations?
      | ...
      +--ro yang-library-version    string
```

The "yang-api" YANG data template is defined using the "yang-data" extension in the "ietf-restconf" module, found in Section 8. It specifies the structure and syntax of the conceptual child resources within the API resource.

Logical organization of data in RESTCONF

Operations on YANG modeled data is part of protocol (RESTCONF). Protocol is aware of following
- config type
- operation

Protocol (RESTCONF) allows data resource addressing in URL

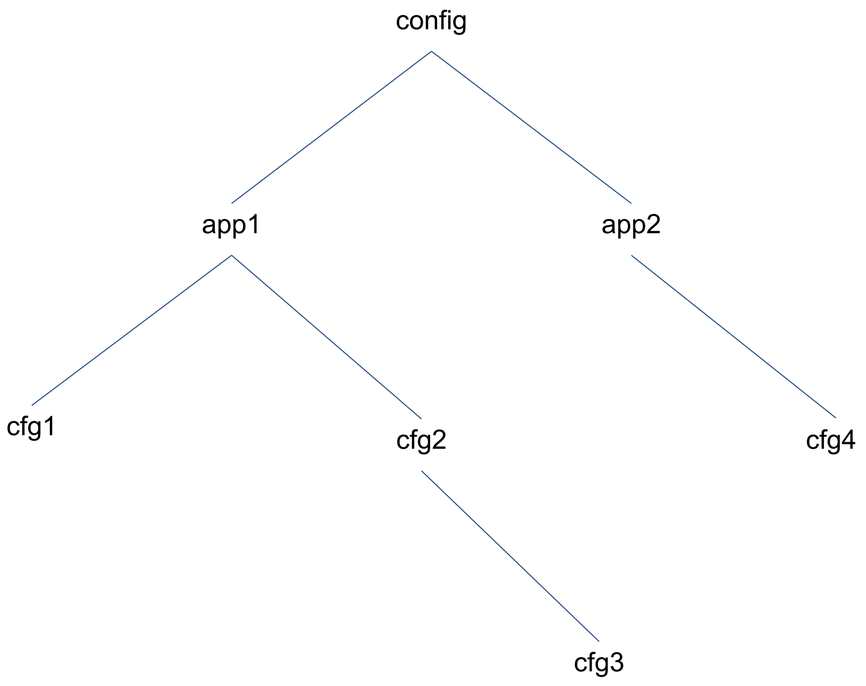
PUT /restconf/data/sample1:container1/ HTTP/1.1
Host: example.com
Content-Type: application/yang-data+json

```
{
  "sample1:container2" :
    {
      "leaf2a" : "abc"
    }
}
```

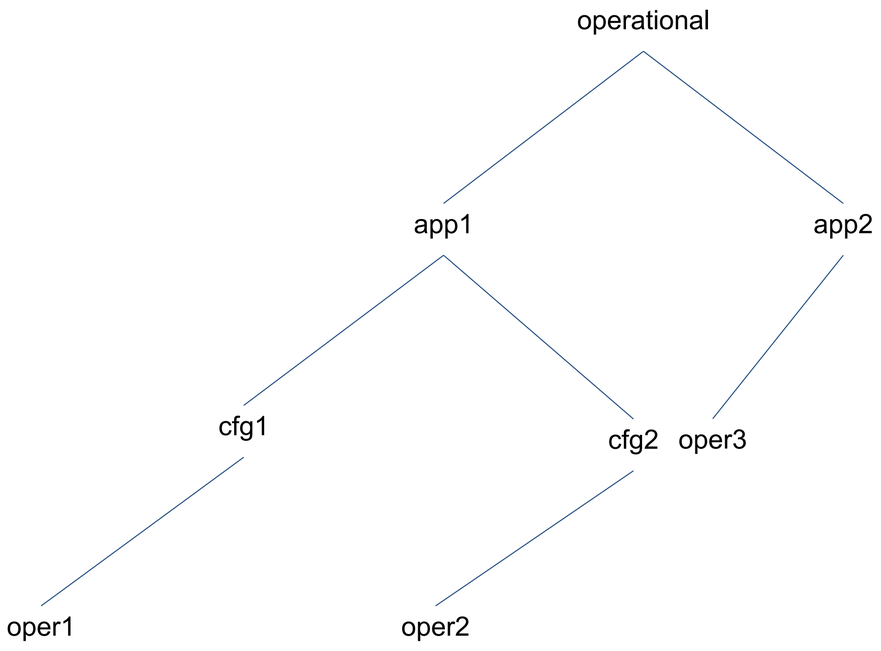
Protocol (RESTCONF) allows resource information in encoding body JSON/XML

Logical tree store alternatives

config tree



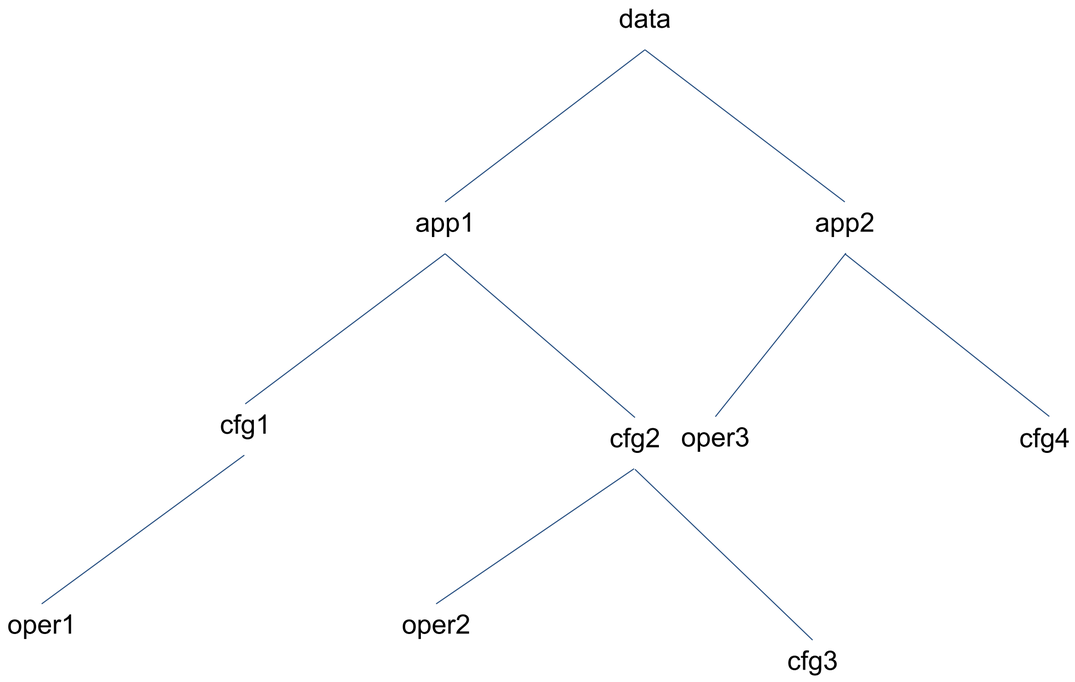
operational tree



Disjoint config and operational store

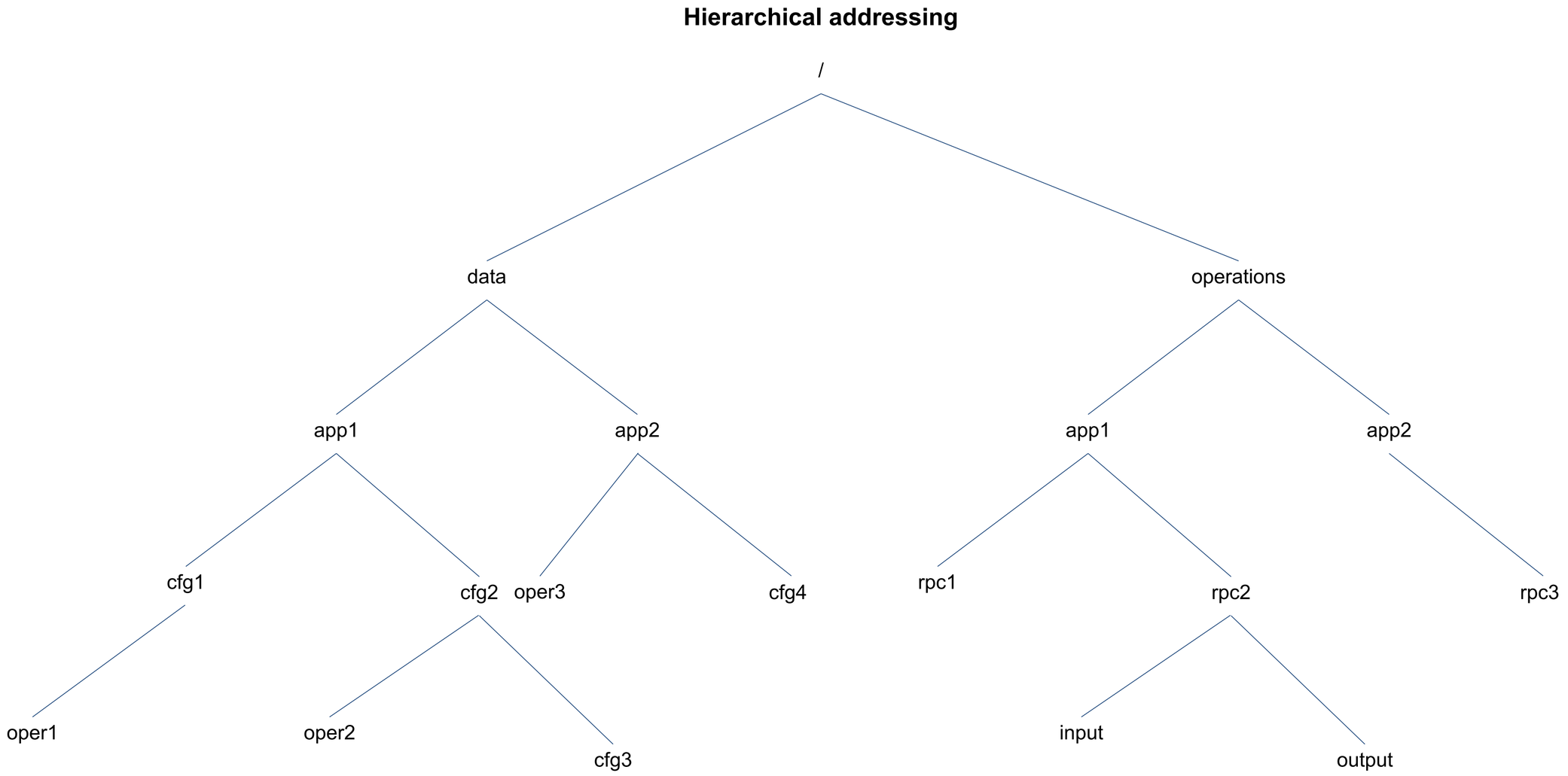
Logical tree store alternatives

data tree

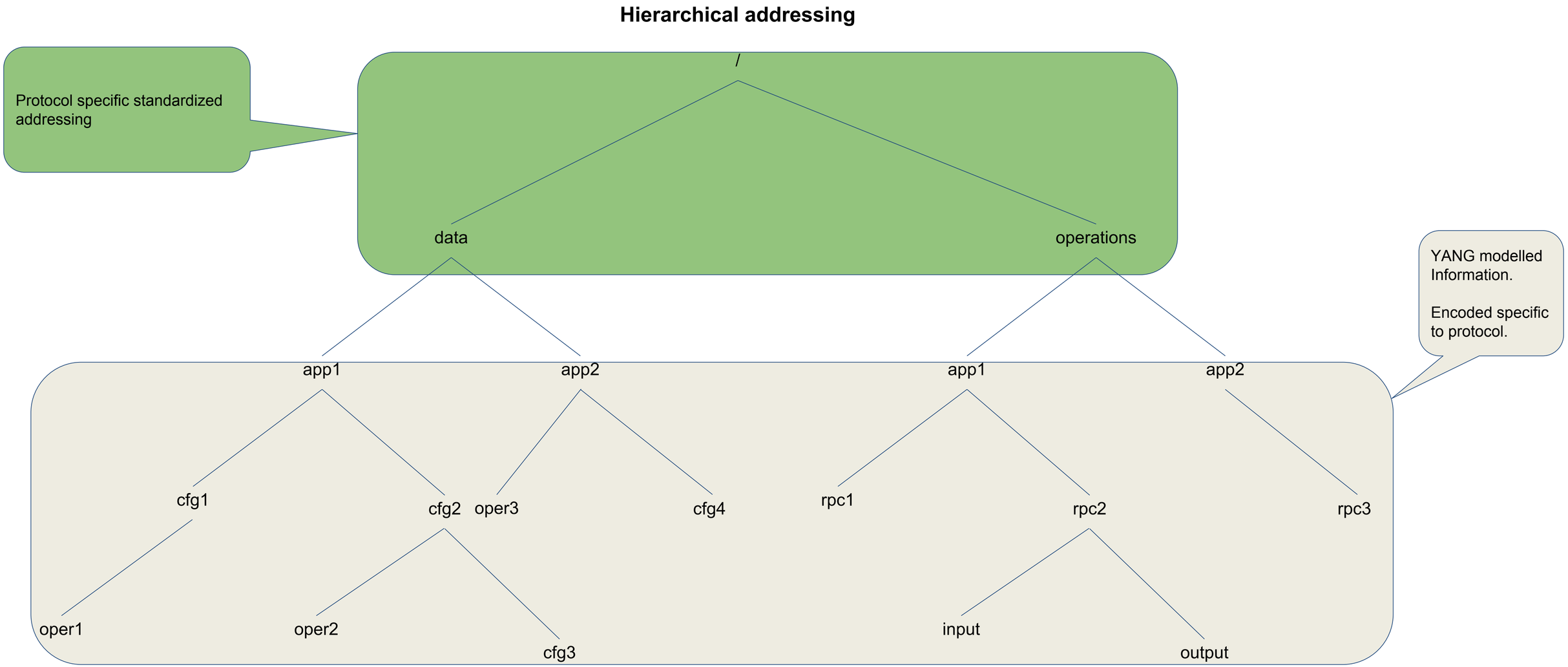


unified data store (config and operational)

Logical addressing of data and operations



Logical addressing of data and operations

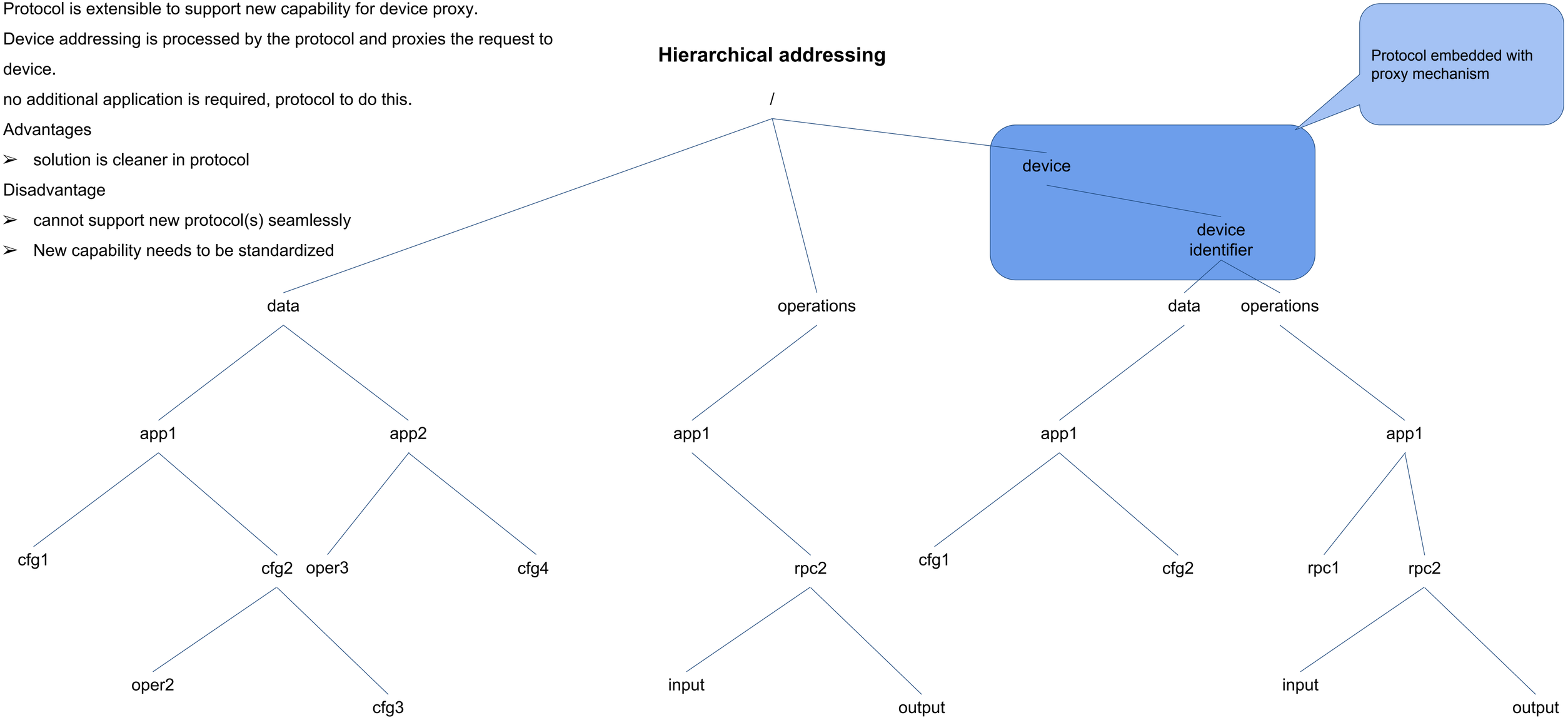


Issue in exposing device YANG as ONOS NBI

- YANG exposed interfaces belongs to the device exposing it
  - ONOS is exposing on behalf of device, there is no inbuilt mechanism for this.
  - ONOS exposing the device YANG, makes ONOS to own the resources as per the YANG rules.
  - potential reuse of same application name / namespace across different vendor, with proprietary structure.
  - Device specific yang library management.
- Device addressing is new requirement for ONOS NBI.
- Device addressing is not part of device YANG.
- Solution 1: Protocol support device proxy.
- Solution 2: ONOS app acts as proxy to manage device.

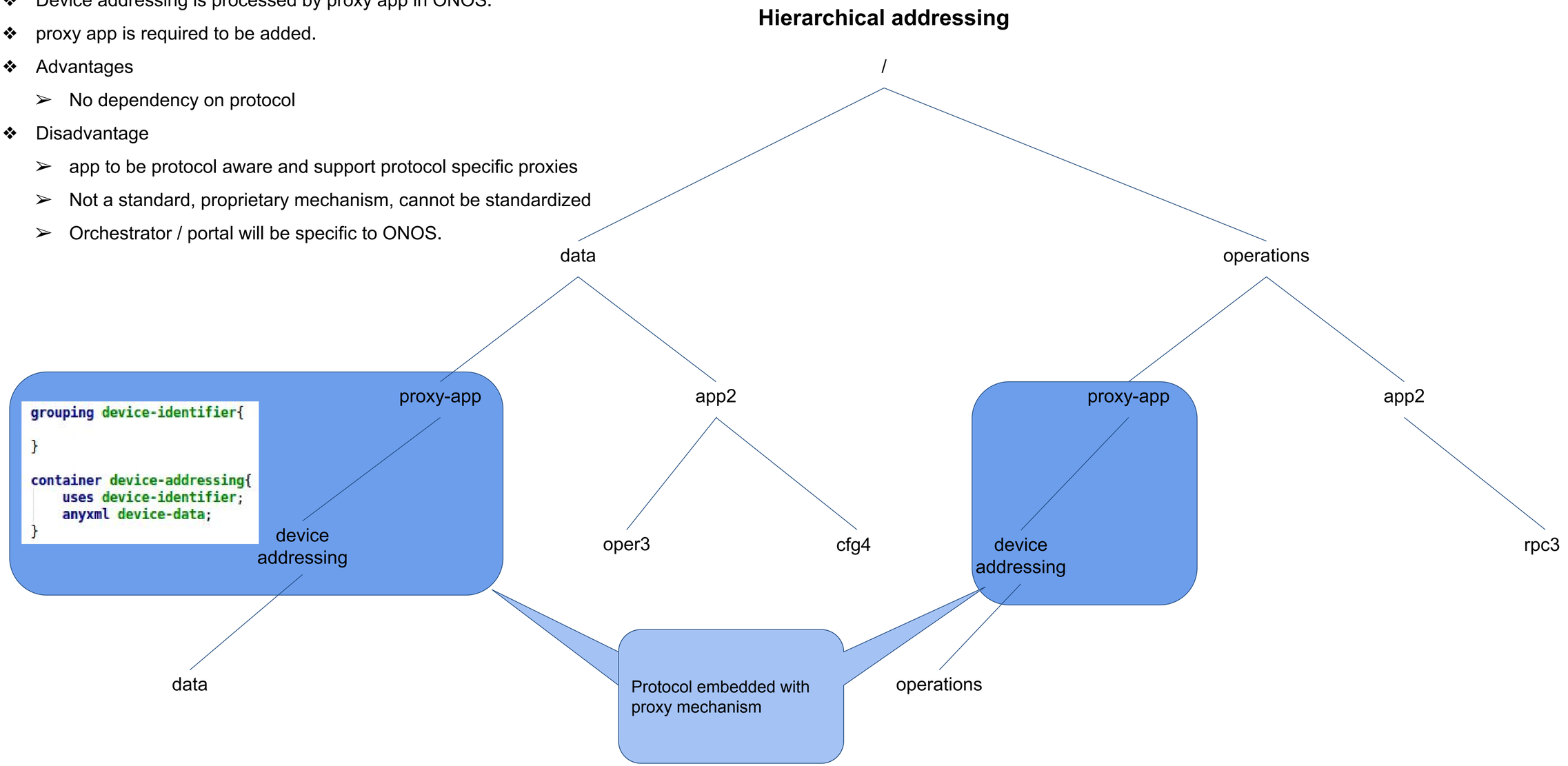
Solution 1: Protocol support device proxy

- Protocol is extensible to support new capability for device proxy.
- Device addressing is processed by the protocol and proxies the request to device.
- no additional application is required, protocol to do this.
- Advantages
  - solution is cleaner in protocol
- Disadvantage
  - cannot support new protocol(s) seamlessly
  - New capability needs to be standardized



Solution 2: ONOS app acts as proxy

- No impact to protocol.
- Device addressing is processed by proxy app in ONOS.
- proxy app is required to be added.
- Advantages
  - No dependency on protocol
- Disadvantage
  - app to be protocol aware and support protocol specific proxies
  - Not a standard, proprietary mechanism, cannot be standardized
  - Orchestrator / portal will be specific to ONOS.



Next step

- Derive or finalize the solution to be chosen.
- Based on the solution standardize it, if generic.
- Any other alternatives???

Thank You