Protocol for I2RS

I2RS WG

IETF #89 London, UK

Dean Bogdanovic deanb@juniper.net

v0.1

Definitions

- Persistent data store – *data store on network element where management system writes configuration data. The persistent data has to survive reboot process, as well it has to provide access to historical config changes*

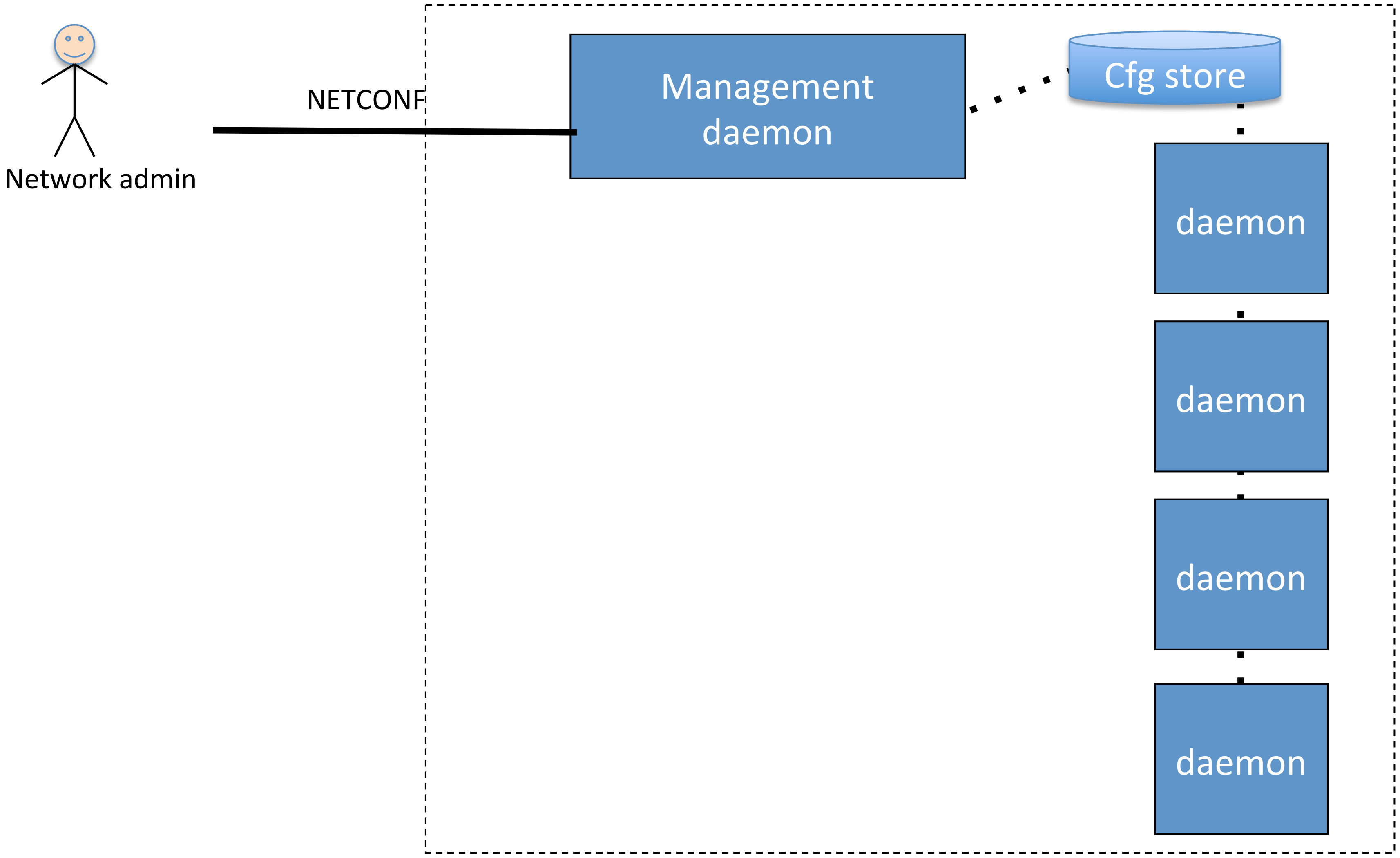
- Ephemeral data store – *data store where management system writes temporary configuration data. The ephemeral data store doesn't survive reboot process, as well doesn't provide access to historical config changes. Ephemeral config data isn't verified*.

- Operational state – *state of control daemon. It can be changed by reading persistent or ephemeral config store, control protocols or through exposed APIs.*

Intro

- Reviewing RESTCONF and NETCONF capabilities for I2RS use
- YANG is assumed as Data Language for I2RS
- Assumptions: I2RS Agent and Clients have access to YANG DM

I2RS Agent and Client architecture

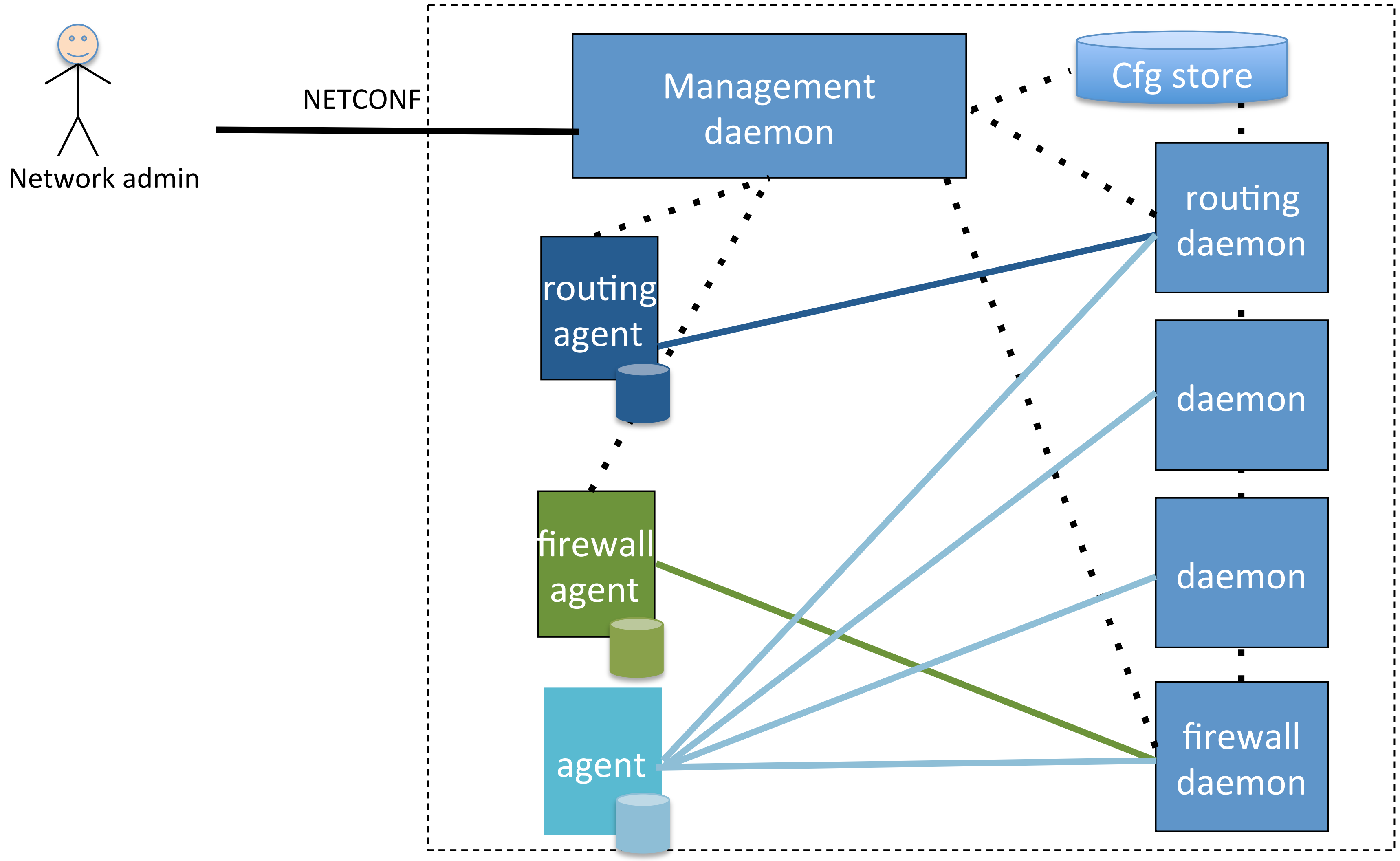


Configuration model

Need mechanism to define service templates to be exposed to the clients via agents

```
agents{
  routing{
    routing-services-template;
  }
  filtering{
    filtering-services-template;
  }
}
```

I2RS Agent and Client architecture



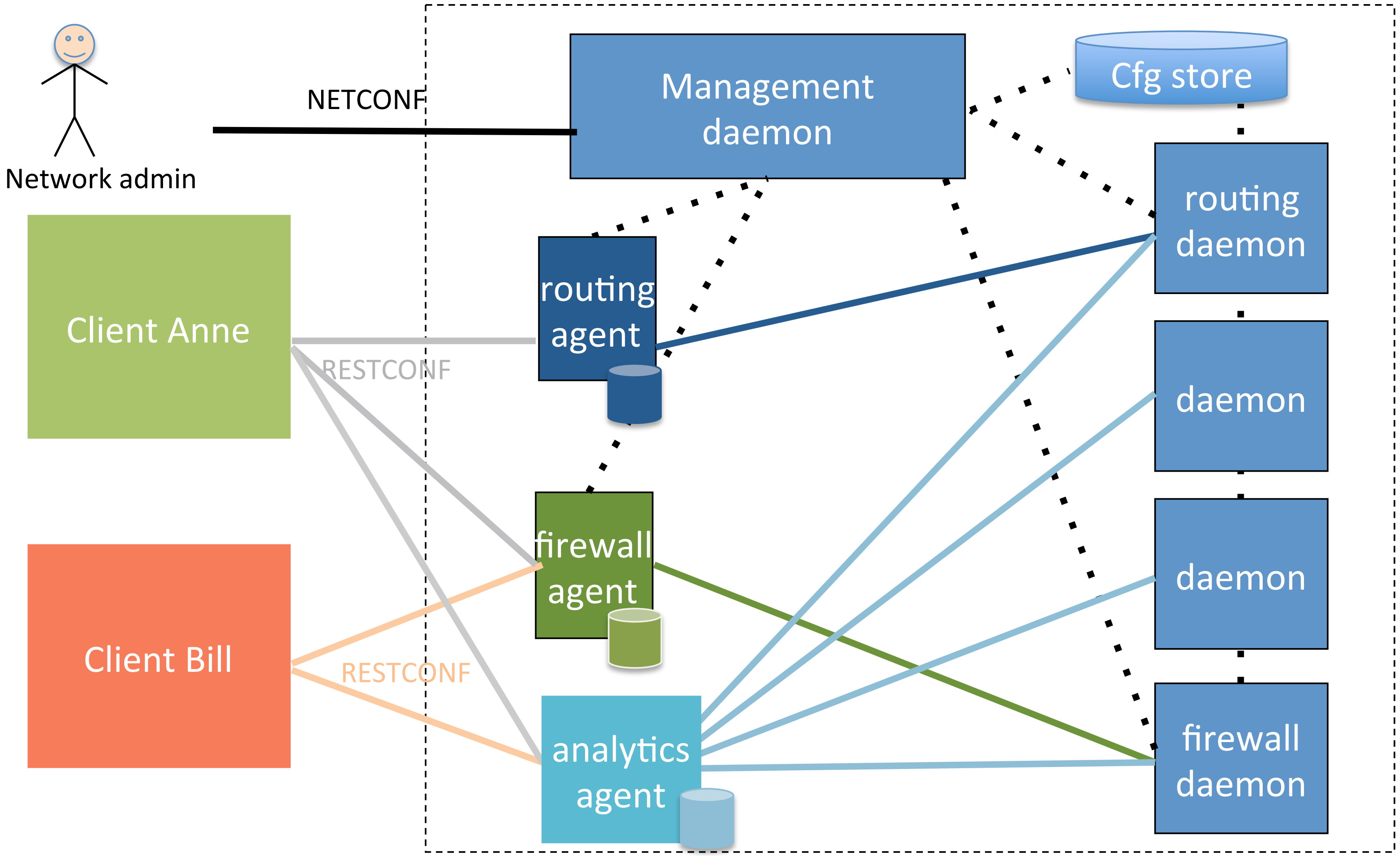
Client policy definition

Define policy for clients which agents they can access

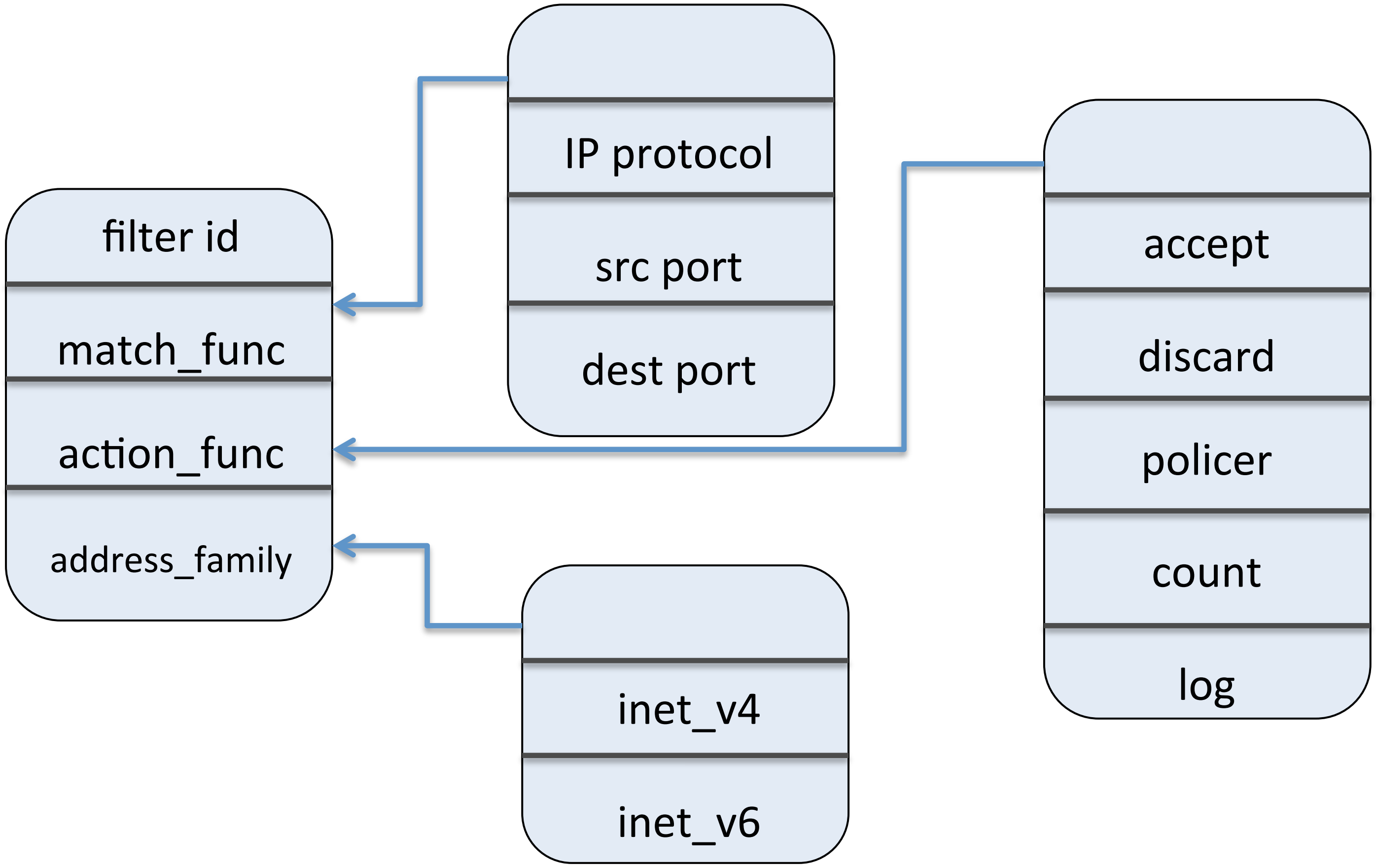
```
clients {
  client-Anne{
    authentication local;
    access {
      filtering;
      routing;
      analytics;
    }
  }
```

```
client-Bill{
    authentication RADIUS;
    access {
      filtering;
      analytics;
    }
  }
}
```

I2RS Agent and Client architecture



Filter model (simple)



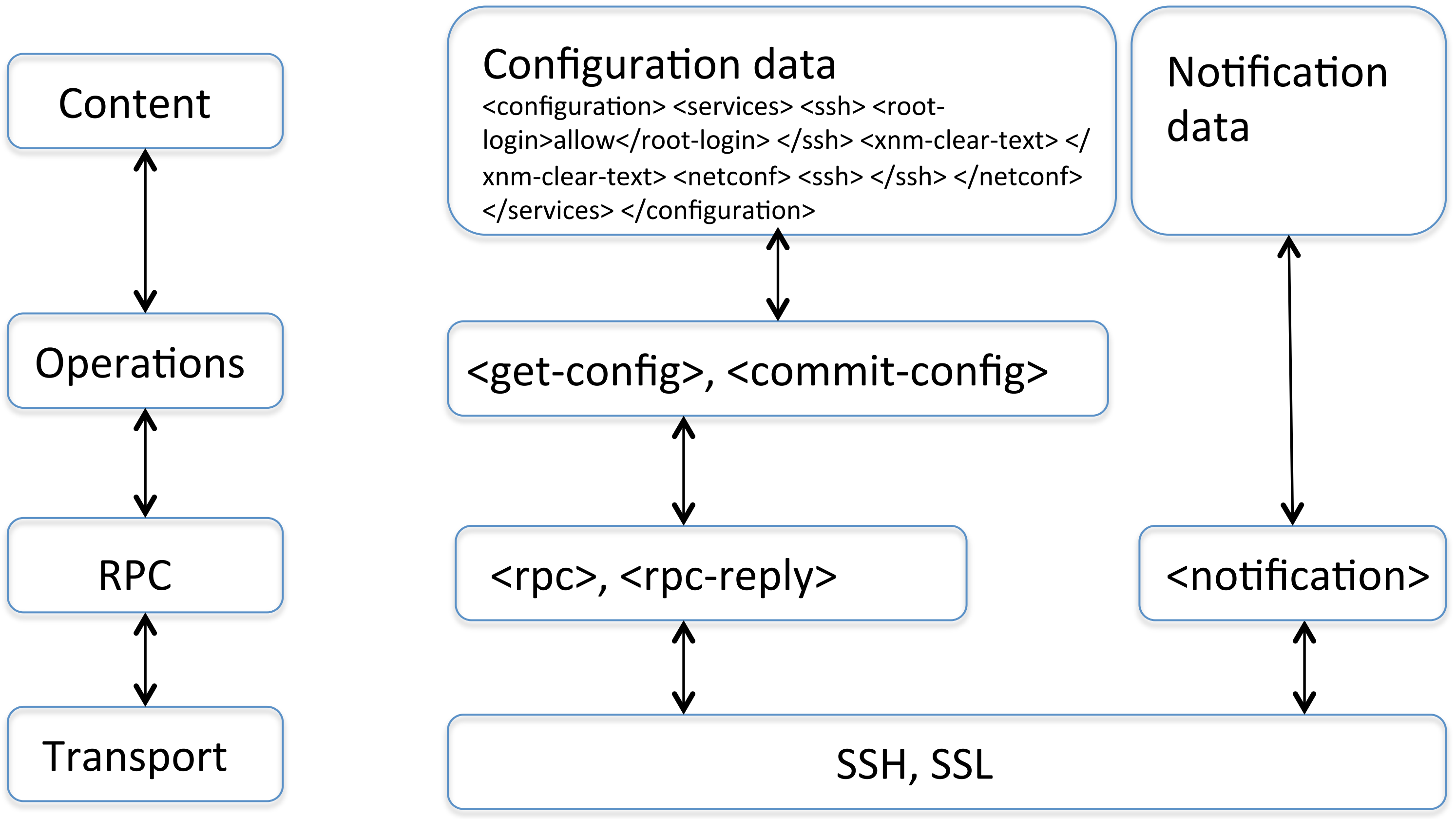
Device configuration vs modifying operational state

- NETCONF and RESTCONF provide mechanism to configure devices, but not to change operational state

NETCONF

- Network Configuration Protocol (RFC 6241)
- Operations are realized on top of RPCs
- Uses XML for configuration data as well as protocol messages
- The NETCONF protocol can be conceptually partitioned into four layers:
    1. The Content layer consists of configuration data and notification data
    2. The Operations layer defines a set of base protocol operations to retrieve and edit the configuration data.
    3. The Messages layer provides a mechanism for encoding remote procedure calls (RPCs) and notifications
    4. The Secure Transport layer provides a secure and reliable transport of messages between a client and a server.

NETCONF cnt'd


Content
Operations
RPC
Transport
Configuration data
<configuration> <services> <ssh> <root-login>allow</root-login> </ssh> <xnm-clear-text> </xnm-clear-text> <netconf> <ssh> </ssh> </netconf> </services> </configuration>
Notification data
<get-config>, <commit-config>
<rpc>, <rpc-reply>
<notification>
SSH, SSL

NETCONF

Cons

- Can’t modify operational daemon state directly
- Multiple configuration data stores
- Commit model
- RPC based

Pros

- IETF Standards track configuration protocol
- Selective data retrieval with filtering
- Provides data validation and verification

RESTCONF

- IETF draft draft-bierman-netconf-restconf-04
- defined as simplified interface to resource-oriented device abstractions
- not intended to provide full capabilities as NETCONF

RESTCONF

Cons

- No network locking model
- Can’t modify operational state of network device
- Using JSON only simple meta-data is supported

Pros

- Unified data store
- Provides atomicity of transaction
- Simplified defaults handling
- Allows multiple edits (with PATCH) within single message
- Providing abstracted simplified config model
- Supports XML and JSON
- Streaming via Server-Sent-Events
- Edit collision detection

RESTCONF and NETCONF Gaps

- Both protocols will need some new mechanism in order to be able to install operational state on the device:
  - <edit-operational> in NETCONF

  or
  - PUT/POST/PATCH to the operational resource in RESTCONF