Internet Engineering Task Force 104, Prague

INTERFACING ASYNCHRONOUS AND SYNCHRONOUS NETWORK MANAGEMENT PROTOCOLS

Jean CHORIN
Tutors: Edward BIRRANE (JHU/APL), Marius FELDMANN (TU Dresden)
26.03.19
Motivation

- Network Management: on **conventional** and Delay/Disruption-Tolerant Networks (DTN): communication optimized for special needs:
- Conventional network: make the most of **low delay, high bandwidth**, ...
- DTN: must use **automation** $\rightarrow$ reduce the number of messages exchanged

- No approach is suitable for both cases. How to manage DTN nodes from conventional network? $\rightarrow$ **Combine the two mechanisms**
Goals

• **Interface** the Asynchronous Management Protocol (AMP) and Network Configuration Protocol (NETCONF)
• Create a **Gateway** between an IP network and a DTN
• **Translate** the Application Data Models (ADM) into the Yet Another Next Generation (YANG) data model
• Final purpose:
  • Being able to **send commands** to the AMP Agents and **fetch data** using the NETCONF Managers
General Architecture

- Two networks: IP network managed by NETCONF
- DTN managed by AMP

- NETCONF is synchronous ("pull" mechanism)
- AMP is asynchronous ("push" mechanism)
Protocol Stack
Gateway Architecture

Diagram: Gateway Architecture with NETCONF Agent 1 and NETCONF Agent n connected to Gateway, which includes Translation functions, Controller, AMP Manager, and a cached database for storing values and requests.
Translation approach (1/3)

Example of yangcli command

```bash
$ gen_rpts ids="Amp/Agent/Edd/num_sbr" rxmgrs="ipn:4.5" rxmgrs="ipn:6.5"
```
Translation approach (2/3)

XML translation of the command (received by the Gateway):

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="7"
 xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <gen_rpts xmlns="Amp/Agent">
   <ids>Amp/Agent/Edd/num_sbr</ids>
   <rxmgrs>ipn:4.5</rxmgrs>
   <rxmgrs>ipn:6.5</rxmgrs>
 </gen_rpts>
</rpc>
```

Python dictionary used for the AMP message:

```python
{
   'control_timestamp': 0,
   'ari_dict': {
     'adm': 'amp_agent',
     'ari_type': 'CTRL',
     'object_name': 'gen_rpts',
     'parameters': {
       0: [{
         'adm': 'amp_agent',
         'ari_type': 'EDD',
         'object_name': 'num_sbr'
       }],
       1: ['ipn:4.5', 'ipn:6.5']
     }
   }
}
```
**Translation approach (3/3)**

A report received by the AMP Manager

---

**AMP DATA REPORT**

Sent to:  
Rpt Name: num_sbr  
Timestamp: Tue Feb 30 13:13:42 2019

# Entries: 1

---

num_sbr: 10

---

**get_AMP_value_uint32**  
`cache_timeout=1000`  
`value_ref=Amp/Agent/Edd/num_sbr`

**Command to fetch a value**

---

**rpc-reply**

`value 10`

`asked_time 1553634318`

`processed_time 1553634367`

`received_time 1553634387`

---

**notification**

`eventTime 2019-02-30T17:20:28Z`

`amp_value_received_uint32 {`

`value_received 11`

`asked_time 1553634345`

`processed_time 1553634370`

`received_time 1553634428`

`}`

**Cached value sent to the NETCONF Manager**

**Updated value sent later on to the NETCONF Manager in a notification**
Demonstration

- Start an **AMP Agent** on a DTN, a **NETCONF Manager** on an IP network, the **Gateway** at the junction

- With a **NETCONF RPC**, create a **Control** to have a new **Time-Based Rule** on Agent, for sending **EDD** (Externally Defined Data)

- The RPC is **translated** on the Gateway into an **AMP Command**

- Data is sent from the Agent to the Gateway

- A **specific RPC command** requests the current value
Video

- https://youtu.be/-AT7mF8Gn94
Conclusion

• Can send **any control** to the AMP Agent through the Gateway

• Can send ARI with **parameters**

• Can retrieve data from the Gateway

• Sent **notification** when the updated data is present
Future works

• Support retrieval of AMP’s Table Templates (TBLT) (under development)

• Support several Gateways

• Better rationale for: Cache Timeout, staleness of data and timestamp

• Optimizations
Bibliography


