INSTITUTE
OF COMMUNICATION,
INFORMATION
AND PERCEPTION
TECHNOLOGIES

Scuola Superiore
Sant'Anna

Management of Sliceable Transponder with NETCONF and YANG

M. Dallaglio[1], N. Sambo[1], F. Cugini[2], P. Castoldi[1]

1: Scuola Superiore Sant'Anna, Pisa, Italy
2: CNIT, Pisa, Italy

ORCHESTRA

ONDM 2016

Introduction

- Relevant advances in the data and control plane
  - data plane:
    - ✓flexible transponders → configurable/adaptable rate, FEC, format, slice-ability …
    - ✓support of monitoring through Digital Signal Processing (pre-FEC BER, Q factor, etc.)
  - control plane:
    - ✓Software Defined Networking → to remotely set network devices, programming transmission characteristics (such as bit rate) and switching
- Management?
  - innovations have not followed these trends yet [1]:
    - ‣ issues related to the presence of network devices from different vendors
    - ‣ lack of standard solutions (e.g., for operation administration and maintainance – OAM)
- NETCONF based on YANG model is emerging as a standard SDN protocol providing both control (e.g., data plane device configuration) and management (e.g., access to monitoring information) functionalities

In this paper:

- we present and demonstrate a YANG model describing flexible transponders supporting monitoring functionalities
- experimental demonstration: connection setup and OAM through NETCONF and YANG

[1] A. Martinez, M. Yannuzzi, V. Lopez, D. Lopez, W. Ramirez, R. Serral-Gracia, X. Masip-Bruin, M. Maciejewski, and J. Altmann, “Network management challenges and trends in multi-layer and multi-vendor settings for carrier-grade networks,” *Communications Surveys Tutorials, IEEE*, vol. 16, no. 4, 2014.

NETCONF and YANG

- NETCONF: Network configuration and management protocol standardized by IETF [2]
  - Clear separation between configuration and state data
  - Possibility to create and modify configuration data
  - Possibility to retrieve state data and to be notified once particular events occur

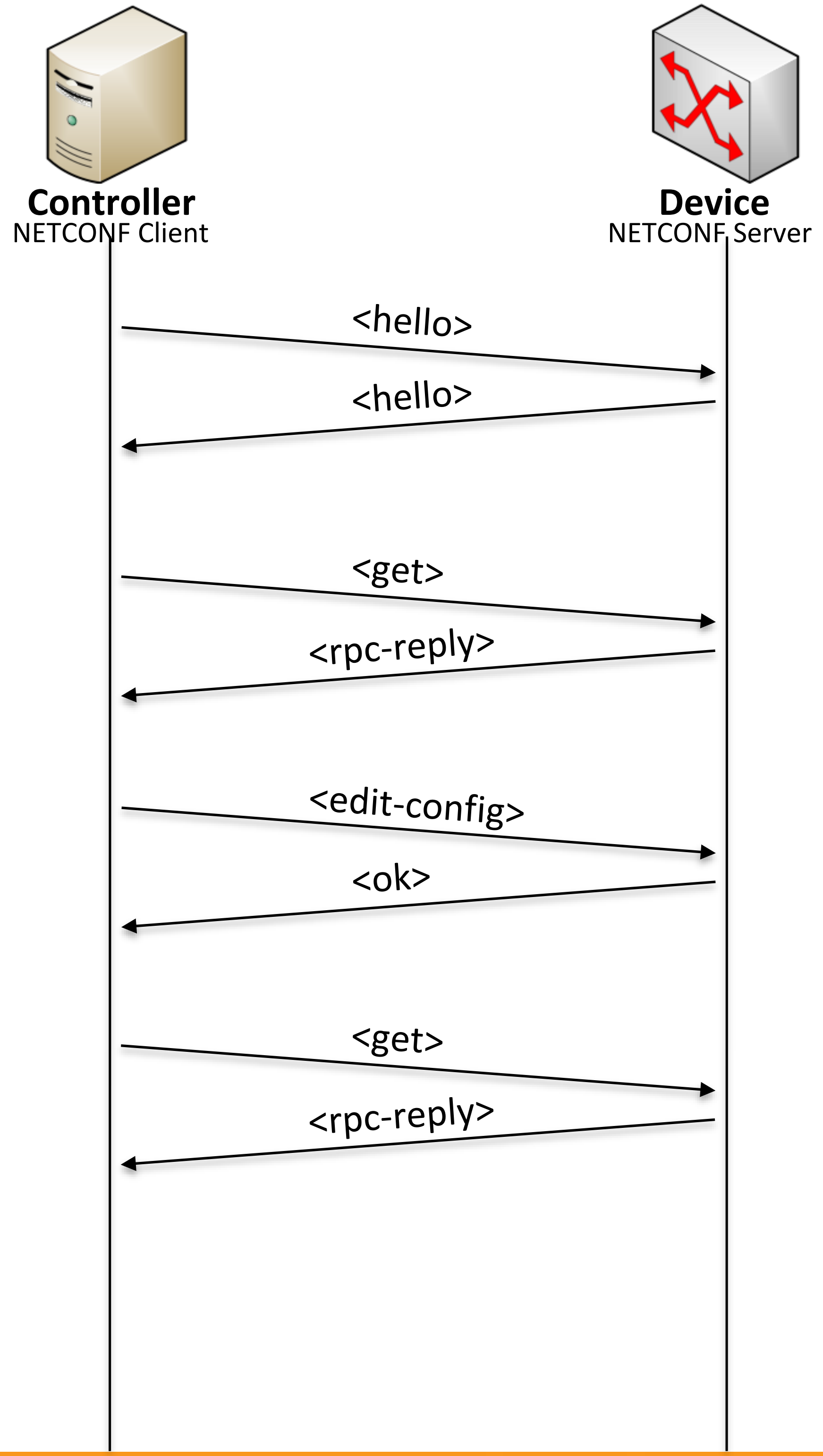
- YANG: data modelling language can be used to describe the structure and semantics of a network device in a vendor-neutral format [3]
  - Ongoing work on YANG model for flexigrid TED (with some transponder information) [4]

[2] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman, “Network configuration protocol (NETCONF),” IETF RFC 6241, June 2011.

[3] M. Bjorklund, “YANG - a data modeling language for the network configuration protocol (NETCONF),” IETF RFC 6020.

[4] 1. J. Vergara and et al., IETF draft-vergara-ccamp-flexigrid-yang-02, Oct. 2014.

NETCONF messages



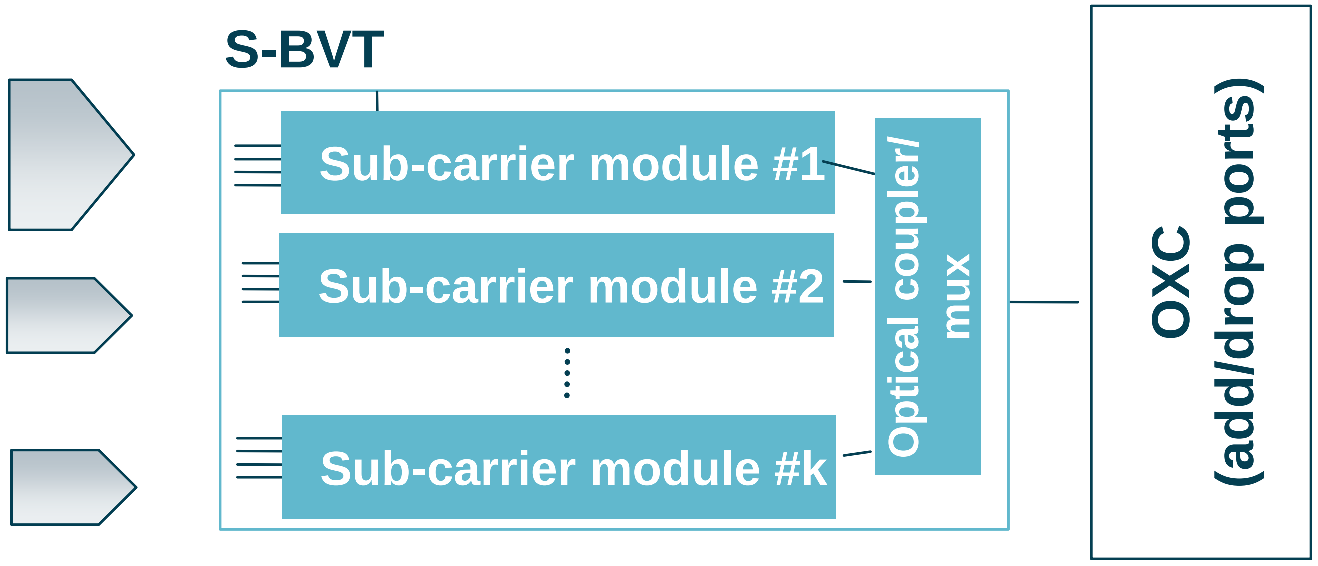
DISCOVERY (supported YANG models)

RUNNING CONFIGURATION + STATE (Functionalities)

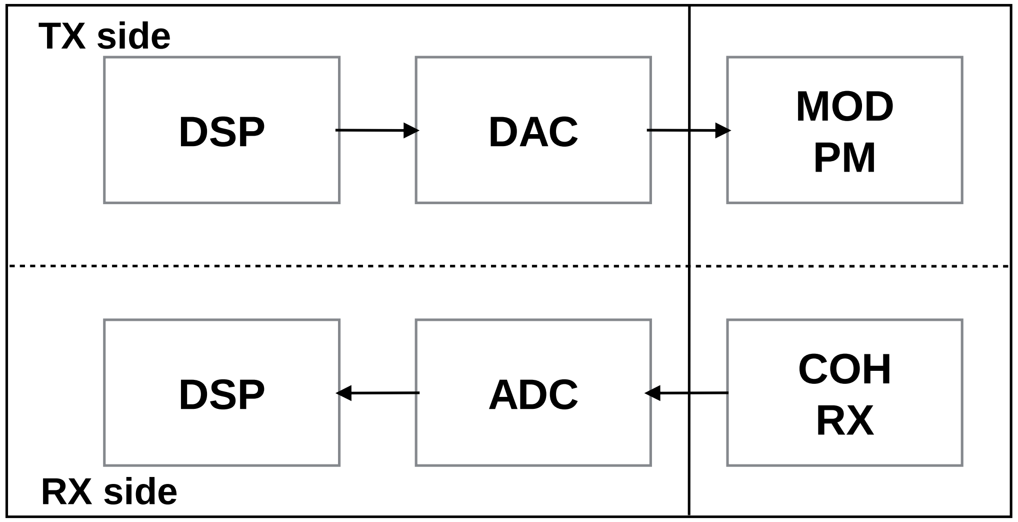
DEVICE CONFIGURATION

PARAMETERS MONITORING

Reference sliceable transponder (S-BVT)



Sub-carrier module

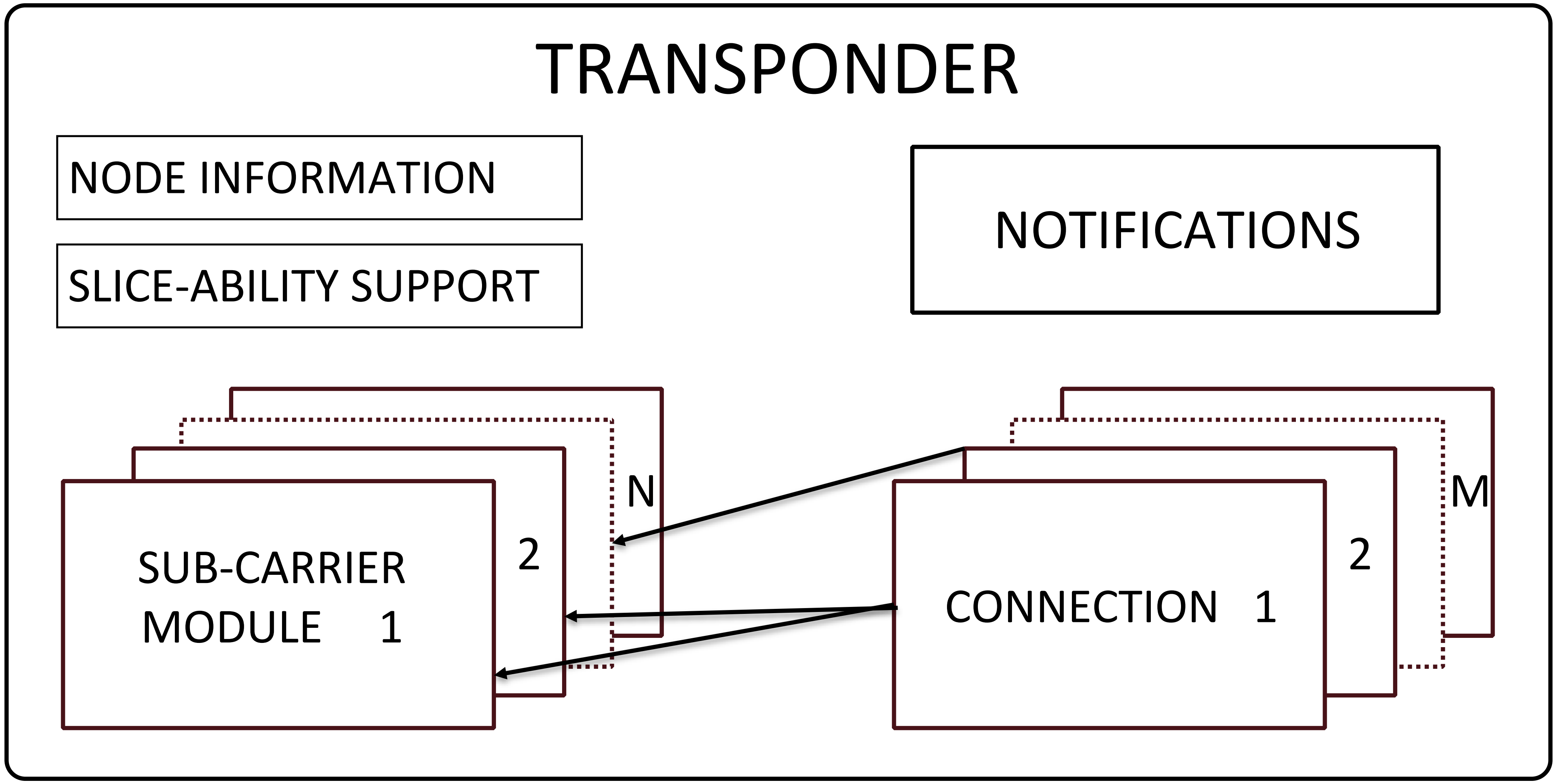


N. Sambo and et al., “Next generation sliceable bandwidth variable transponders,” Communications Magazine, IEEE, vol. 53, no. 2, pp. 163–171, Feb 2015.

TRANSPONDER YANG SCHEME

Following : YANG standardization guidelines IETF [5,6] and OpenConfig working group [7].

We propose:



[5] A. Bierman, “Guidelines for Authors and Reviewers of YANG Data Model Documents”, IETF RFC 6087, 2011.
[6] R. Shakir, “Consistent Modeling of Operational State Data in YANG draft-openconfig-netmod-opstate-01”, IETF Draft, 2015.
[7] http://www.openconfig.net

YANG CONFIG AND STATE DATA

Configuration data

- Writable (NETCONF <edit-config>)
- Explicitly set by an external entity

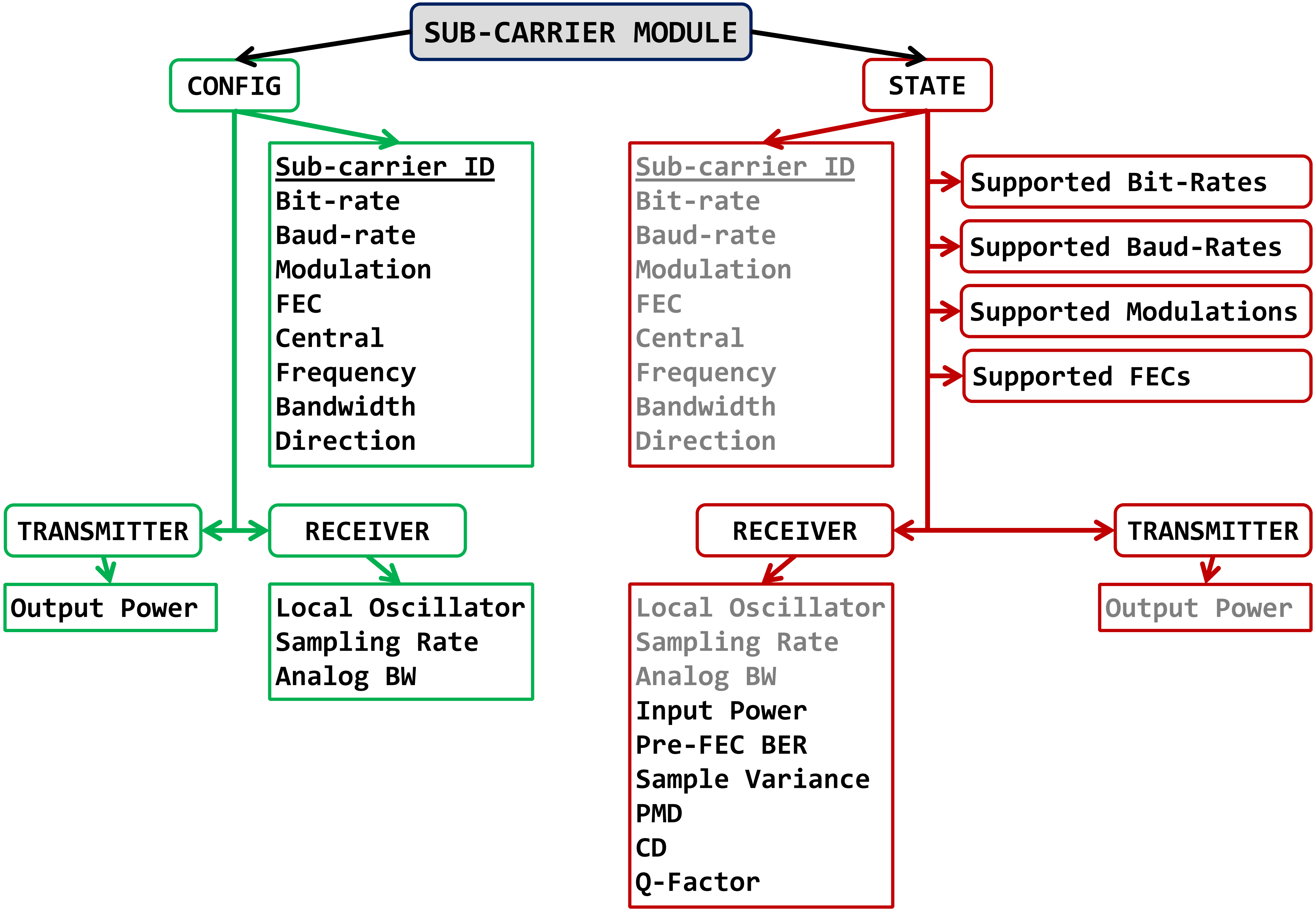
State data

- Read only (NETCONF <get>)
- Parameters that cannot be set by an external entity
- Monitoring information / parameters supported by the device

Intended configuration: the state that the network operator intends the system to be in.
Applied configuration: the state that the network element is actually in.

Configuration data is replicated into State data

SUB-CARRIER MODULE



SUB-CARRIER MODULE

CONFIG

Sub-carrier ID
Bit-rate
Baud-rate
Modulation
FEC
Central Frequency
Bandwidth
Direction

TRANSMITTER

Output Power

RECEIVER

Local Oscillator
Sampling Rate
Analog BW

STATE

Sub-carrier ID
Bit-rate
Baud-rate
Modulation
FEC
Central Frequency
Bandwidth
Direction

Supported Bit-Rates

Supported Baud-Rates

Supported Modulations

Supported FECs

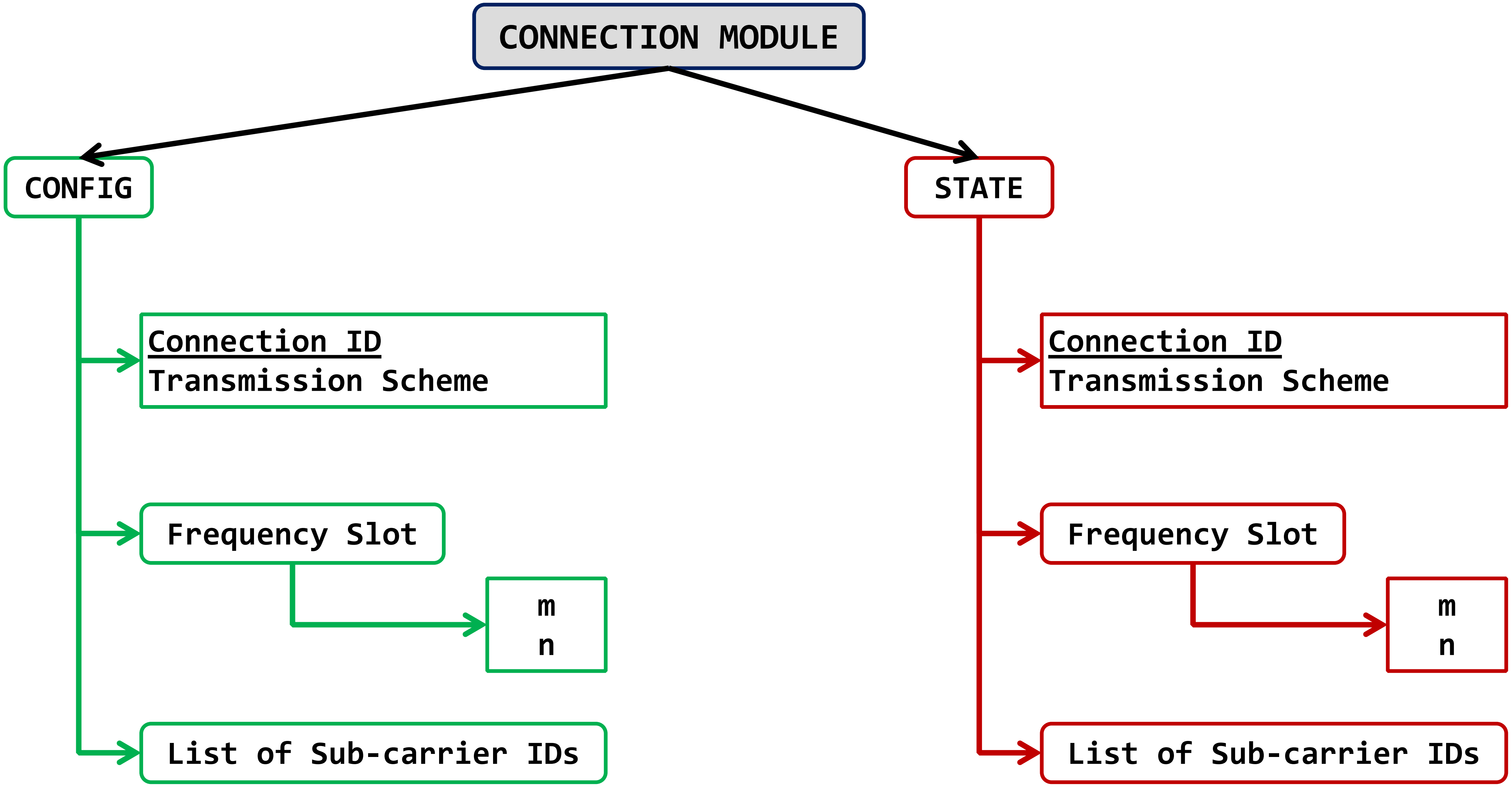
RECEIVER

Local Oscillator
Sampling Rate
Analog BW
Input Power
Pre-FEC BER
Sample Variance
PMD
CD
Q-Factor

TRANSMITTER

Output Power

CONNECTION MODULE



CONNECTION MODULE

CONFIG

Connection ID
Transmission Scheme

Frequency Slot

m
n

List of Sub-carrier IDs

STATE

Connection ID
Transmission Scheme

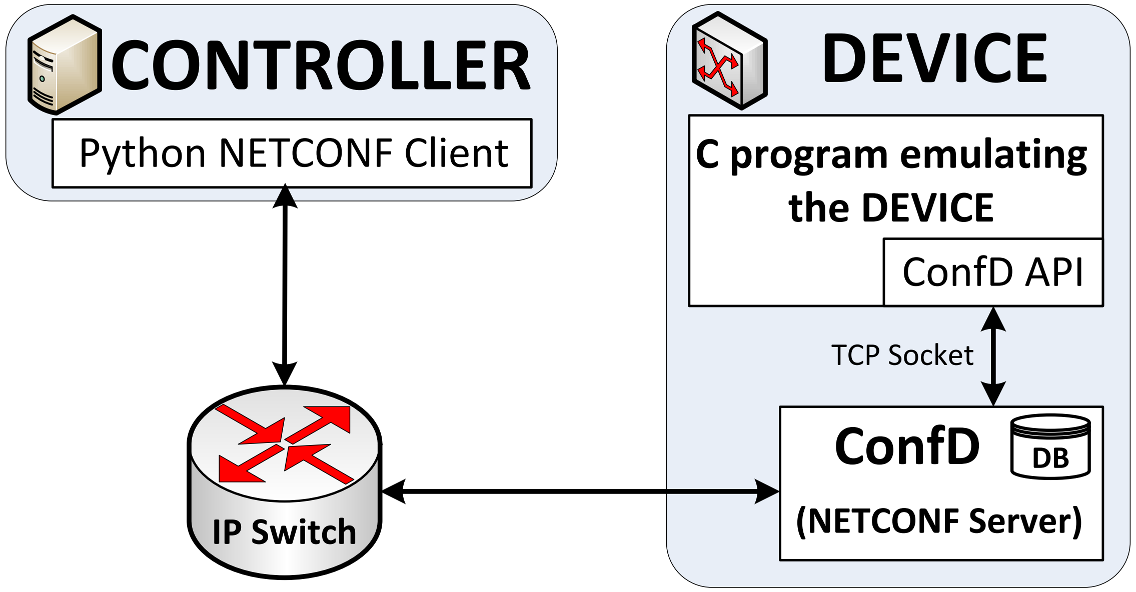
Frequency Slot

m
n

List of Sub-carrier IDs

Experimental demonstration

TESTBED

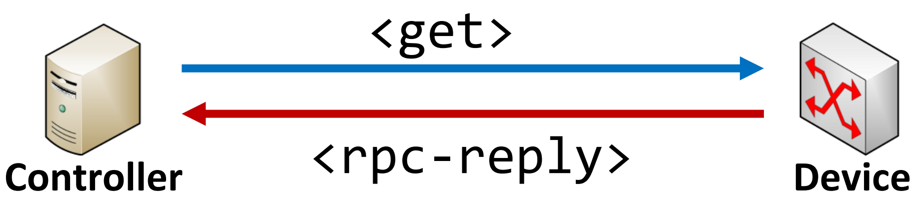


- Transponder Discovery
- Connection Setup
- Connection Monitoring

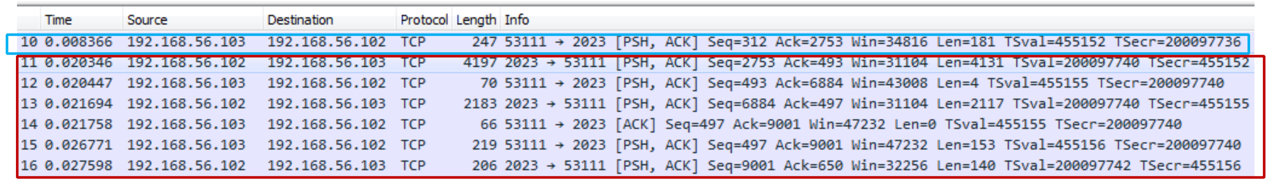
Transponder discovery

The controller issues a <get> message to retrieve the device’s current state

(e.g. installed sub-carriers modules, supported transmission parameters).



Wireshark Capture



	Time	Source	Destination	Protocol	Length	Info
10	0.008366	192.168.56.103	192.168.56.102	TCP	247	53111 → 2023 [PSH, ACK] Seq=312 Ack=2753 Win=34816 Len=181 TSval=455152 TSecr=200097736
11	0.020346	192.168.56.102	192.168.56.103	TCP	4197	2023 → 53111 [PSH, ACK] Seq=2753 Ack=493 Win=31104 Len=4131 TSval=200097740 TSecr=455152
12	0.020447	192.168.56.103	192.168.56.102	TCP	70	53111 → 2023 [PSH, ACK] Seq=493 Ack=6884 Win=43008 Len=4 TSval=455155 TSecr=200097740
13	0.021694	192.168.56.102	192.168.56.103	TCP	2183	2023 → 53111 [PSH, ACK] Seq=6884 Ack=497 Win=31104 Len=2117 TSval=200097740 TSecr=455155
14	0.021758	192.168.56.103	192.168.56.102	TCP	66	53111 → 2023 [ACK] Seq=497 Ack=9001 Win=47232 Len=0 TSval=455155 TSecr=200097740
15	0.026771	192.168.56.103	192.168.56.102	TCP	219	53111 → 2023 [PSH, ACK] Seq=497 Ack=9001 Win=47232 Len=153 TSval=455156 TSecr=200097740
16	0.027598	192.168.56.102	192.168.56.103	TCP	206	2023 → 53111 [PSH, ACK] Seq=9001 Ack=650 Win=32256 Len=140 TSval=200097742 TSecr=455156

<get> message

```
<?xml version="1.0" encoding="UTF-8" ?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <get><filter type='xpath' select=' /transponder' /></get>
</rpc>
```

<rpc-reply> message

```
<?xml version="1.0" encoding="UTF-8" ?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <data>
    <transponder xmlns="http://sssup.it/transponder">
      <node-id>1</node-id>
      <add-drop-id>1</add-drop-id>
      <slice-ability-support>true</slice-ability-support>
      <subcarrier-module>
        <subcarrier-id>1</subcarrier-id>
        <state>
          <supported-bit-rates>
            <bit-rate>112.0</bit-rate>
            <bit-rate>124.0</bit-rate>
            <bit-rate>224.0</bit-rate>
            <bit-rate>248.0</bit-rate>
          </supported-bit-rates>
          <supported-baud-rates>
            <baud-rate>28.0</baud-rate>
            <baud-rate>31.0</baud-rate>
          </supported-baud-rates>
          <supported-modulations>
            <modulation xmlns:mdfrms="/sssup/mdfrms">mdfrms:dp-qpsk</modulation>
            <modulation xmlns:mdfrms="/sssup/mdfrms">mdfrms:dp-16qam</modulation>
          </supported-modulations>
          <supported-fec>
            <fec xmlns:fec="/sssup/fec-types">fec:ldpc</fec>
            <fec xmlns:fec="/sssup/fec-types">fec:golay</fec>
          </supported-fec>
        </state>
      </subcarrier-module>
      .........
      <subcarrier-module>
        <subcarrier-id>4</subcarrier-id>
        .........
      </subcarrier-module>
      <connections></connections>
    </transponder>
  </data>
</rpc-reply>
```

Connection Setup

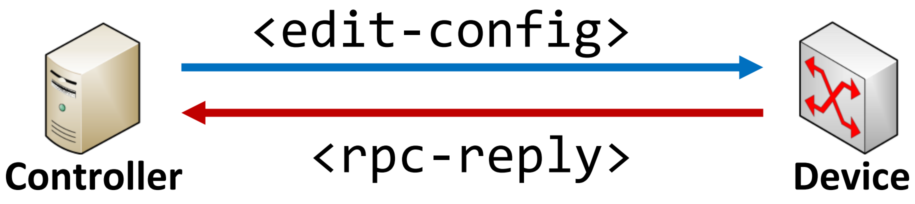
<edit-config> message

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0“ message-id="1">
<edit-config xmlns:nc='urn:ietf:params:xml:ns:netconf:base:1.0'>
<target><running/></target><config>
<transponder xmlns="http://sssup.it/transponder" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
    <subcarrier-module>
        <subcarrier-id>1</subcarrier-id>
        <config>
            <direction>RX</direction>
            <bit-rate>112</bit-rate>
            <baud-rate>28</baud-rate>
            <modulation xmlns:mf="/sssup/mdfrms">mf:dp-qpsk</modulation>
            <fec-in-use>
                <name xmlns:fec="/sssup/fec-types">fec:ldpc</name>
                <rate> <message-length>14</message-length> <block-length>15</block-length> </rate>
            </fec-in-use>
            <central-frequency>193100</central-frequency>
            <bandwidth>33.6</bandwidth>
            <receiver>
                <sampling-rate>35</sampling-rate>
                <local-oscillator>193100</local-oscillator>
                <analog-bw>10.0</analog-bw>
            </receiver>
        </config>
    </subcarrier-module>
    <connections>
        <connection nc:operation="create">
            <connection-id>1</connection-id>
            <config>
                <connection-id>1</connection-id>
                <transmission-scheme>NWDM</transmission-scheme>
                <subcarrier> <subcarrier-id>1</subcarrier-id> </subcarrier>
                <frequency-slot> <n>0</n> <m>3</m> </frequency-slot>
            </config>
        </connection>
    </connections>
</transponder>
</config></edit-config>
</rpc>
```

The controller issues a <edit-config> message to create a new connection.

Connection parameters:

- 112Gbps
- DP-QPSK
- LDPC 14/15 FEC

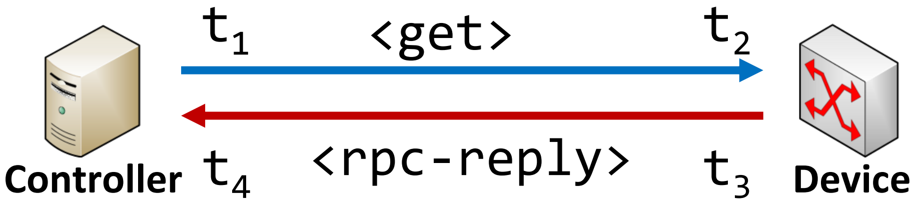


<rpc-reply> message

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
<ok/>
</rpc-reply>
```

Monitoring

The controller monitors the Q-Factor of the connection by periodically issuing a <get> command (polling)



$t_{TOT} = t_4 - t_1 \approx 11ms$ $\qquad$ $t_{PROC} = t_3 - t_2 \approx 8.5ms$

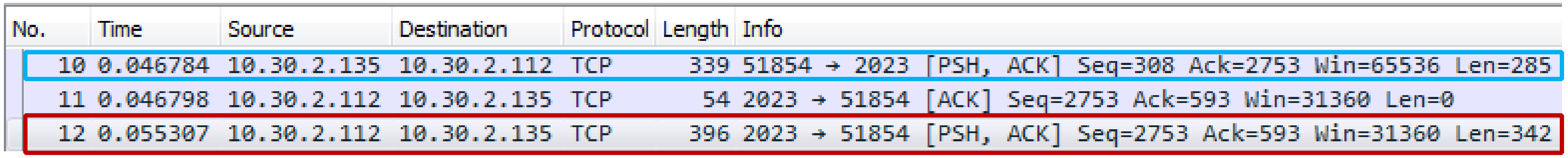
<get> message

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
message-id="1">
    <get>
    <filter type='xpath' select=' /transponder/subcarrier-
module[subcarrier-id=1]/state/receiver/q-factor'/>
    </get>
</rpc>
```

<rpc-reply> message

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-
id="1">
  <data>
    <transponder xmlns="http://sssup.it/transponder">
      <subcarrier-module>
        <subcarrier-id>1</subcarrier-id>
        <state>
          <receiver>
            <q-factor>6.0</q-factor>
          </receiver>
        </state>
      </subcarrier-module>
    </transponder>
  </data>
</rpc-reply>
```

Wireshark Capture



No.	Time	Source	Destination	Protocol	Length	Info
10	0.046784	10.30.2.135	10.30.2.112	TCP	339	51854 → 2023 [PSH, ACK] Seq=308 Ack=2753 Win=65536 Len=285
11	0.046798	10.30.2.112	10.30.2.135	TCP	54	2023 → 51854 [ACK] Seq=2753 Ack=593 Win=31360 Len=0
12	0.055307	10.30.2.112	10.30.2.135	TCP	396	2023 → 51854 [PSH, ACK] Seq=2753 Ack=593 Win=31360 Len=342

Conclusions

- This paper presented a YANG model for transponders with monitoring capabilities, slice-ability, and variable:
  - Bit-rate
  - Baud-rate
  - FEC
  - Modulation Format

  Model: https://github.com/mattedallo/sssa/tree/master/yang-models
- Experiments have shown transponder state/features discovery and management

ACK: The work has been supported by the ORCHESTRA project.

TRANSPONDER YANG

```
module transponder {
namespace "http://sssup.it/transponder";
prefix tran;

  import modulation-formats {
    prefix mdfrms;
  }

  import fec-types {
    prefix fec;
  }

  import ietf-yang-types {
    prefix yang;
  }

  organization
   "Scuola Superiore Sant'Anna Network and Services Laboratory";

  description
   "This module contains a YANG definitions for configuring Optical Transponder.";

  revision 2015-09-15 {
    description "Initial Revision.";
    reference   "TBD";
  }

  typedef transmission-type {
    description "The transmission method";
    type enumeration {
      enum NWDM;
      enum O-OFDM;
      enum TFP;//Time-frequency packing
    }
  }
```

TRANSPONDER YANG

```
typedef direction-type {
  description "Indicates the direction";
  type enumeration {
    enum TX;
    enum RX;
  }
}

typedef bit-rate-type {
  type decimal64 {
    fraction-digits 3;
    range "0..max";
  }
  units "Gb/s";
}

typedef baud-rate-type {
  type decimal64 {
    fraction-digits 3;
    range "0..max";
  }
  units "Gbaud";
}

typedef modulation-type {
  type identityref {
    base mdfrms:modulation-format;
  }
}

typedef fec-type {
  type identityref {
    base fec:fec-type;
  }
}
```

TRANSPONDER YANG

```
typedef frequency-ghz-type {
  type decimal64 {
    fraction-digits 8;
    range "0..max";
  }
  units "GHz";
}


grouping fec-config {
  description "Configuration data for forward error correction";

  container fec-in-use {
    description "FEC in use";
    presence "Enables FEC";

    leaf name {
      type fec-type;
    }

    container rate {
      description
      "The code rate is given by message-length/block-length";
      leaf message-length {
        type int16 {
          range "1..max";
        }
      }
      leaf block-length {
        type int16 {
          range "1..max";
        }
      }
      must "block-length >= message-length" {
          error-message "block-length must be greater or equal to message-length";
      }
    }//container rate

  }//container fec-in-use
}//grouping fec-config
```

TRANSPONDER YANG

```
grouping fec-state {
  description "Operational state data for forward error correction";
  container supported-fec {
    description "List of supported FEC schemes";
    leaf-list fec {
      type fec-type;
    }
  }//supported
}//grouping fec-state


grouping transmitter-config {
  description "Configuration data for the transmitter";
  leaf output-power {
    description "launch power at the transmitter";
    type int16;
    units "dBm";
  }
}//grouping transmitter-config

grouping transmitter-state {
  description "Operational state data for the transmitter";
}


grouping receiver-config {
  description "Configuration data for the receiver";

  leaf local-oscillator {
    type frequency-ghz-type;
  }

  leaf sampling-rate {
    description "Minimum hardware requirements in terms of sampling rate";
    type uint32;
    units "GS/s";
  }
```

TRANSPONDER YANG

```
    leaf analog-bw {
      description "Minimum hardware requirements in terms of analog bandwidth";
      type frequency-ghz-type;
    }

  }//grouping receiver-config

  grouping receiver-state {
    description "Operational state data for the receiver";

    leaf input-power {
      description "per-channel received optical power at the receiver";
      type int16;
      units "dBm";
    }

    leaf pre-fec-ber {
      description
      "Pre-FEC Bit Error Rate.";
      type decimal64 {
        fraction-digits 18;
        range "0..max";
      }
    }

    leaf sample-variance {
      type decimal64 {
        fraction-digits 18;
        range "0..max";
      }
      reference
        "http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7035536";
    }
```

TRANSPONDER YANG

```
  leaf pmd {
    description
    "Polarization Mode Dispersion.";
    type decimal64 {
      fraction-digits 8;
      range "0..max";
    }
    units "ps/(km)^0.5";
  }

  leaf cd {
    description
    "Chromatic Dispersion.";
    type decimal64 {
      fraction-digits 5;
    }
    units "ps/nm/km";
  }

  leaf q-factor{
    type decimal64 {
      fraction-digits 5;
    }
    units "dB";
  }
}//grouping receiver-state


grouping subcarrier-module-config {
  description "Configuration data for the optical subcarrier-module";

  leaf direction {
    description "Defines whether the subcarrier is received or transmitted";
    type direction-type;
  }

  leaf bit-rate {
    description "The bit-rate in use";
    type bit-rate-type;
  }
```

TRANSPONDER YANG

```
    leaf baud-rate {
      description "The baud-rate in use";
      type baud-rate-type;
    }

    leaf modulation {
      description "Modulation format in use";
      type modulation-type;
    }
    uses fec-config;
    leaf central-frequency {
      description
        "The central frequency of the subcarrier.";
      type frequency-ghz-type;
    }

    leaf bandwidth {
      description
        "The bandwidth occupied.";
      type frequency-ghz-type;
    }
  }//subcarrier-module-config

  grouping subcarrier-module-state {
    description "Operational state data for the optical subcarrier-module";
    container supported-bit-rates{
      description "List of supported bit-rates";
      leaf-list bit-rate {
        description "the bit rate value";
        type bit-rate-type;
      }
    }

    container supported-baud-rates {
      description "List of supported baud-rates";
      leaf-list baud-rate {
        description "the baud rate value";
        type baud-rate-type;
      }
    }
```

TRANSPONDER YANG

```
  container supported-modulations {
    description "List of supported modulation formats";
    leaf-list modulation {
      description "Name of the supported modulation";
      type modulation-type;
    }
  }
  uses fec-state;
}//subcarrier-module-state

grouping subcarrier-module {
  description "Top-level grouping for optical subcarrier-module";

  container config {
    description
      "Configuration data for subcarrier-module";
    uses subcarrier-module-config;

    container transmitter {
      when "../direction = 'TX'";
      uses transmitter-config;
    }

    container receiver {
      when "../direction = 'RX'";
      uses receiver-config;
    }
  }
  container state {
    config false;
    description
      "Operational state data for subcarrier-module";
    uses subcarrier-module-config;
    uses subcarrier-module-state;

    container transmitter {
      when "../direction = 'TX'";
      uses transmitter-config;
      uses transmitter-state;
    }
```

TRANSPONDER YANG

```
      container receiver {
        when "../direction = 'RX'";
        uses receiver-config;
        uses receiver-state;
      }
    }

  }//subcarrier-module

  grouping connection-config {
    description "Configuration data for a connection";
    leaf connection-id{
      type uint32;
    }

    leaf transmission-scheme {
      description "The scheme adopted for the transmission";
      type transmission-type;
    }

    list subcarrier {
      description "List of ids of the involved subcarriers";
      key "subcarrier-id";
      leaf subcarrier-id {
        type leafref {
          path "/tran:transponder/tran:subcarrier-module/tran:subcarrier-id";
        }
      }
    }

    container frequency-slot {
      description
        "The frequency range allocated to a slot
        within the flexible grid and unavailable to other slots.  A
        frequency slot is defined by its nominal central frequency and its
        slot width.";
      reference "draft-ietf-ccamp-flexi-grid-fwk-07";
```

TRANSPONDER YANG

```
    leaf nominal-central-frequency-granularity {
      description
        "It is the spacing between allowed nominal central frequencies.";
      type frequency-ghz-type;
      default 6.25;
    }//leaf nominal-central-frequency-granularity

    leaf slot-width-granularity {
      description "It is the minimum slot width.";
      type frequency-ghz-type;
      default 12.5;
    }//leaf slot-width-granularity

    leaf n {
      description
        "n gives the nominal central frequency (ncf) using the following formula:
        ncf = 193.1THz + n x nominal-central-frequency-granularity[THz].";
      type int16;
      mandatory true;
    }//leaf n
    leaf m {
      description
        "m gives the slot width. A slot width is constrained to be
        m x slot-width-granularity";
      type int16 {
        range "1..max";
      }
      mandatory true;
    }//leaf m
  }//container frequency-slot

  leaf source-address {
    description "The IP address of the source node";
    Type inet:ip-address;
  }//leaf source-address
  leaf destination-address {
    description "The IP address of the destination node";
    Type inet:ip-address;
  }//leaf source-address
}//grouping connection-config
```

TRANSPONDER YANG

```
grouping connection-state {
  description "Operational state data for a connection";
}//grouping connection-state

grouping connections {
  description "List of all connections served by the transponder";
  list connection {
    key "connection-id";
    leaf connection-id {
      description "references the configured connection-id";
      type leafref {
        path "../config/connection-id";
      }
    }
    container config {
      description "Configuration parameters for connection";
      uses connection-config;
    }

    container state {
      config false;
      description "State variables for connection";
      uses connection-config;
      uses connection-state;
    }
  }//list connection
}//grouping connections
```

TRANSPONDER YANG

```
//---------- MAIN TREE ------------//
container transponder {
    list subcarrier-module {
      description
        "List of all the subcarrier modules installed in the transponder";
      key "subcarrier-id";
      leaf subcarrier-id {
        type uint32;
      }
      uses subcarrier-module;
    }

    leaf slice-ability-support {
      when "count(../subcarrier-module) > 1";
      type boolean;
      config false;
      description "Determines if the transponder is slice-able.";
    }


    leaf node-id {
      description "ID of the node where the transponder is installed";
      type uint16;
    }

    leaf add-drop-id {
      description "Add/drop ID inside the node";
      type uint16;
    }

    container connections {
      uses connections;
    }
}
```

TRANSPONDER YANG

```
  //------------- NOTIFICATIONS ----------------//
  notification pre-fec-ber-change {
    leaf subcarrier-module-id {
      description
        "An existing subcarrier-module in the list";
      type leafref {
        path "/tran:transponder/tran:subcarrier-module/tran:subcarrier-id";
      }
      mandatory true;
    }
    leaf pre-fec-ber {
      type leafref {
        path "/transponder/subcarrier-module[subcarrier-id=current()/../subcarrier-module-id]/state/receiver/pre-fec-ber";
      }
      mandatory true;
    }
  }
  notification pmd-change {
    leaf subcarrier-module-id {
      description
        "An existing subcarrier-module in the list";
      type leafref {
        path "/tran:transponder/tran:subcarrier-module/tran:subcarrier-id";
      }
      mandatory true;
    }
    leaf pmd {
      type leafref {
        path "/transponder/subcarrier-module[subcarrier-id=current()/../subcarrier-module-id]/state/receiver/pmd";
      }
      mandatory true;
    }
  }
}//module transponder
```

email: matteo.dallaglio@sssup.it