OPT-Transport Apps of
O3 Orchestrator＆Controller suite
Starting guide

CONTENTS

1 ABBREVIATION

CTP	Connection Termination Point
DON	Dummy Optical Node
DPID	Data Path ID
EMS	Element Management System
HO ODU	Higher Order Optical channel Data Unit
LO ODU	Lower Order Optical channel Data Unit
MPLS	Multi-Protocol Label Switching
OCh	Optical Channel
OCNRM	Optical Core Network Resource Manager
ODU	Optical Data Unit
OMS	Optical Multiplex Section
ONF	Open Networking Foundation
OPT	Optical
Opt.	Optical
OSS	Open Source Software
OTN	Optical Transport Network
PCE	Path Computation Element
Pkt.	Packet
PT	Packet
QoS	Quality of Service
REST	Representational State Transfer
RM	Resource Manager
SDN	Software Defined Networking
SW	Switch
TL1	Transaction Language 1
TPN	Tributary Port Number
TS	Tributary Slot
TTP	Trail Termination Point
WDM	Wavelength Division Multiplexing
XC	Cross Connect

2 INTRODUCTION

If SDN in the optical core network area is realized, it becomes possible to request and utilize low latency and broadband resources on-demand. “OPT-Transport Apps of O3 Controller suite” is one of the applications of “O3 Orchestrator suite” and can make your optical core network with OTN/WDM functions SDN-enabled.

3 Target Network

The target of this software is an optical core network with OTN/WDM functions. Only 10GEther client ports are assumed for the optical network node.

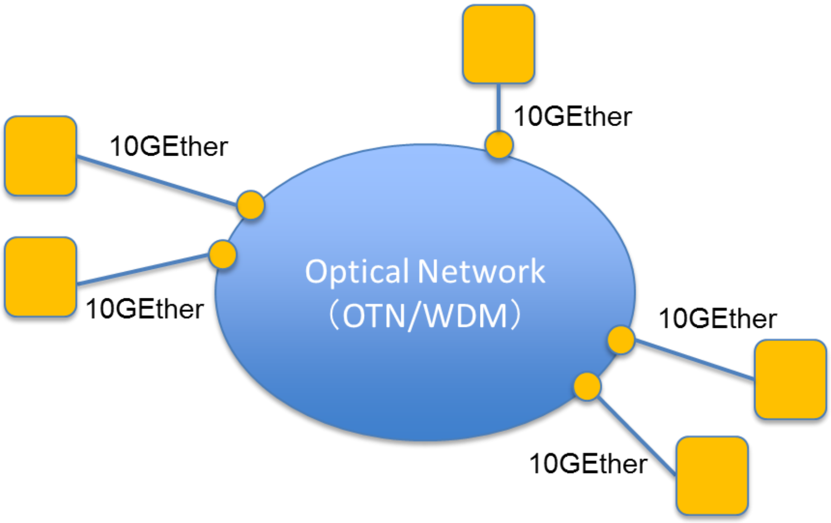


Figure 1 Target network.

4 Outline of Functions

- Virtualization of optical core network resources
  - This function virtualizes nodes, ports, links, and set-up path information and registers the information as a network object of the O3 Orchestrators suite. As a result, the state of the optical core network resources can be treated in the software application.
    - It is assumed that OCh and HO ODU paths are already set.
- Control of optical core network resources
  - "OPT-Transport Apps of O3 Orchestrator＆Controller suite " assigns the requested resources by writing these resources into network objects through the ODENOS interface.
    - RYU[1] OTN Extension (RYU-OE) is needed to control optical core network.
    - Separately, an application that emulates an optical core node is offered. This application establishes a session with RYU-OE and receives control messages.
  - LO ODU path can be assigned dynamically between client ports (10Gbps).
    - Delay constraint can be considered.
    - A cut-through path that does not pass through the electric layer in intermediate nodes can be assigned for a request with severe delay constraint (path computation function may determine appropriate cut-through route).

[1] RYU: RYU is an open source SDN Controller developed by NTT, Japan. Please refer section 7.2.

5 List of the Open Source Software

Main software (O3 Orchestrator & Controller Suite)

1.	OCNRM	Optical core network resource manager

Supplemental software

2.	PseudoManagementFunctions	Dummy software (Physical resource management, PCE)
3.	RYU-OE	RYU OTN extension
4.	DummyOptNode	Dummy optical node
5.	Openflowj-otn	OpenFlowJ OTN extension

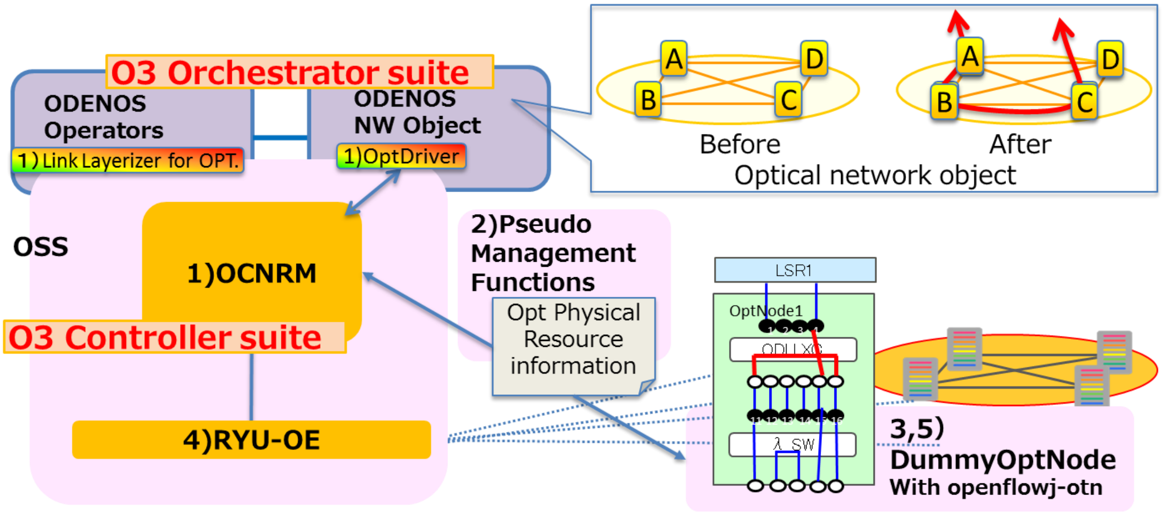


Figure 2 OSS software relationships.

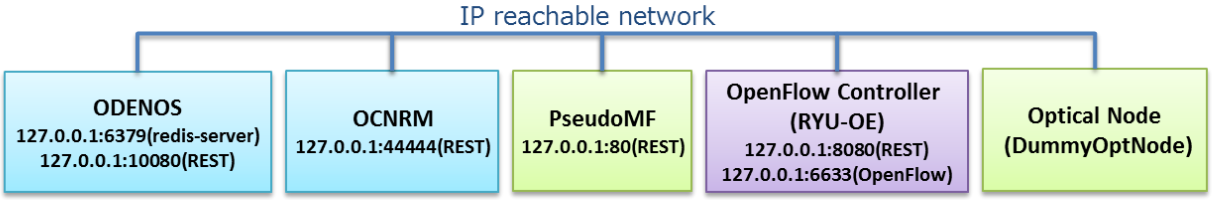


Figure 3 Sample settings.

6 OCNRM – OPT-TRANSPORT APPS OF O3 ORCHESTRATOR & CONTROLLER SUITE

6.1 WHAT IS OCNRM

Optical Core Network Resource Manager (OCNRM) is a software application that achieves Software Defined Networking in the OTN/WDM network. OCNRM works with ODENOS, PseudoManagementFunctions (PseudoMF), and the RYU OTN Extension (RYU-OE) and controls the OTN/WDM network nodes that support the OpenFlow OTN extension.

6.2 SOFTWARE STRUCTURE

OCNRM contains the following software.

- ODENOS logic components “LinkLayerizer” and “OptDriver”(O3 Orchestrator suite) for the optical core network.
- Application that manages and controls virtualized resources of the optical core network (O3 Controller suite).

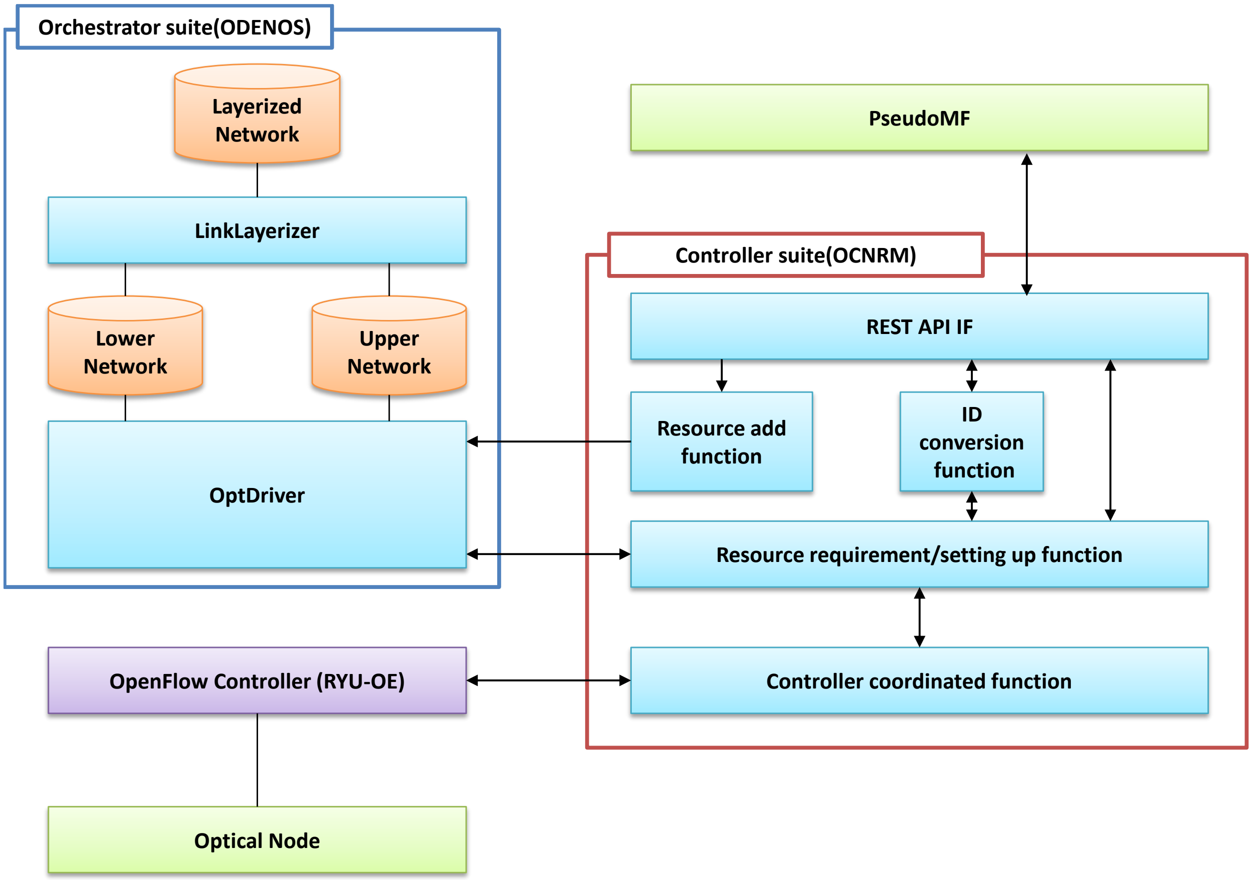


Figure 4 OCNRM component relationships.

6.2.1 ORCHESTRATOR SUITE

Orchestrator suite is a set of programs based on the ODENOS framework (https://github.com/o3project/odenos) that supports the development of SDN applications. OPT-Transport Apps of O3 Orchestrator & Controller suite implements the following two logic components based on the ODENOS framework.

6.2.1.1 Link Layerizer

Link Layerizer is a logic component that abstracts two different layers (upper and lower) as one layer. For the optical core network, Link Layerizer supports the ODU layer as the upper layer and the OCh layer as the lower layer. As a result, these two layers are combined as one layerized network.

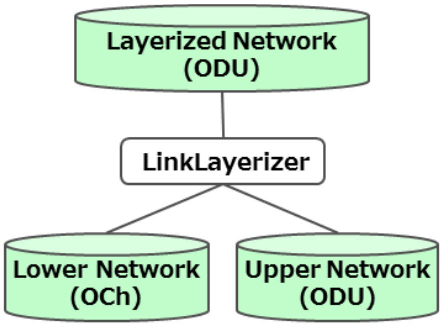


Figure 5 Link Layerizer.

Link Layerizer virtualizes a lower layer flow as the upper layer link. Please see the ODENOS API document (https://github.com/o3project/odenos/blob/develop/doc/api/LinkLayerizer.md) for more detail.

6.2.1.2 OptDriver

OptDriver is a logic component that operates when optical NEs are controlled by ODENOS and the information of NEs is registered.

- OptDriver receives events (e.g., triggers of link/flow settings) from ODENOS and passes them to RM
- OptDriver has a function that registers optical core information to ODENOS through the REST interface

6.2.2 CONTROLLER SUITE

Controller suite is a set of logic components that controls each virtual network resource managed under the Orchestrator suite.

6.2.2.1 Resource manager (RM)

RM is the Controller suite for the optical core network.

- RM decides the control executed to a piece of physical optical equipment according to the notification from OptDriver and then passes RYU-OE the appropriate parameters for control

  - ➢ OpenFlow 1.3 OTN extension is adopted as a southbound interface
- RM converts the information of the optical core network resources (including link and flow settings) into ODENOS format and sends it to OptDriver.

6.3 URL

https://github.com/o3project/ocnrm

6.4 Environment

Operation is validated in the following environments.
OS: Ubuntu 12.04.3 (x64)
Memory: 1GB (without ODENOS)
Middleware: Java 7 and later, Maven2
ODENOS 1.0.0 is required

6.5 Build

```
$ cd ~
$ git clone https://github.com/o3project/ocnrm.git
$ bash ./build.sh
$ tar xvfz ./target/ocnrm-1.0.0-bin.tar.gz –C ~/
```

You can choose your own install directory instead of ~/.

6.6 Configuration

Edit OCNRM.properties. In this document, parameters such as IP address/Port number in sample settings (see Figure 3) are used, for example.

```
$ vi ~/ocnrm-1.0.0-bin/OCNRM.properties
```

Set redis-server's host & port.

```
DISPATCHER_HOST=127.0.0.1
DISPATCHER_PORT=6379
```

Set pseudoMF's REST access points.

```
 REQUEST_ODU_FLOW_URL=http://127.0.0.1:80/DEMO/Generate/L1Path
 REQUEST_OCH_REPLACEMENT_PIECE_URL=http://127.0.0.1:80/DEMO/ID/L0Request
 REQUEST_ODU_REPLACEMENT_PIECE_URL=http://127.0.0.1:80/DEMO/ID/L1Request
 DELETE_ODU_FLOW_URL=http://127.0.0.1:80/DEMO/Delete/L1Path
```

Set RYU-OE's REST access point.

```
 OFCTL_SEND_URL=http://127.0.0.1:8080/stats/flowentry
```

6.7 START AND STOP

Starting OCNRM.

```
$ ~/ocnrm-1.0.0-bin/ocnrm_mn -s
```

Stopping OCNRM.

```
$ ~/ocnrm-1.0.0-bin/ocnrm_mn –q
```

7 Attached Software (PseudoMF, RYU-OE, DummyOptNode)

7.1 PseudoManagementFunctions （PseudoMF）

7.1.1 What Is PseudoMF

PseudoMF is a software application that takes the place of physical network elements management system (EMS) and path computation elements (PCE).

PseudoMF provides optical core network resources information to OCNRM, ID mapping information between OpenFlow and internal ID (which we call Information Model ID), and route information for new resource creation.

- Initial information registration function to ODENOS via OCNRM
- Simple route determination function
- ODENOS/OpenFlow Parameter conversion function

7.1.2 Environment

Same as OCNRM.

7.1.3 URL

https://github.com/o3project/pseudoMF

7.1.4 Build

```
$ cd ~
$ git clone https://github.com/o3project/pseudoMF.git
$ bash ./build.sh
$ tar xvfz ./target/pseudoMf-1.0.0-bin.tar.gz –C ~/
```

You can choose your own install directory instead of ~/.

7.1.5 Configuration

Edit PseudoMf.properties. In this document, parameters such as IP address/Port number in sample settings (see Figure 3) are used, for example.

```
$ sudo vi ~/PseudoMf.properties
```

Set OCNRM’s REST access points.

```
REQUEST_INITIALIZE=http://127.0.0.1:44444/demo/node
```

7.1.6 START AND STOP

Starting PseudoMF.

```
$ ~/pseudoMf-1.0.0-bin/pseudoMf.sh -s
```

Stopping PseudoMF.

```
$ ~/pseudoMf-1.0.0-bin/pseudoMf.sh –q
```

7.2 RYU OTN Extension (RYU-OE)

7.2.1 What Is RYU-OE?

RYU-OE is based on RYU V3.9 developed by NTT.

In RYU-OE, various OTN parameters have been added to the Match Fields of OpenFlow protocol (ver1.3) to support OTN. The details of the extended OpenFlow specification will be published by ONF in the near future.

(1) ODU_SIGTYPE
ODU signal type (ODU0/1/2/2e/3/4 etc.) is set. This software only supports ODU2e.

(2) ODU_SIGID
ODU_SIGID contains in its payload the following parameters:

- TSLEN (the number of TS bits depending on ODU signal type)
- TSMAP (bitmap of TS)
- TPN (Tributary Port Number)

This software only supports TSLEN=8, TSMAP=”1,1,1,1,1,1,1,1”. TPN can be an arbitrary integer.

These parameters can be set in Action by using “Set Field”, which already exists in the OpenFlow protocol. RYU-OE also supports the REST interface in its northbound, and flow entries including these parameters can be set by the REST command as shown in the example below. In the case of setting TSMAP by REST, please write the bit number of TS (counting from the first bit) on which the client signal is mapped.

Example of REST command

```
curl -d '{"dpid":"2",
    "match": {"in_port": "1" , "odu_sigtype": "ODU2E", "odu_sigid":{ "tpn": "1", "tsmap": "1,2,3,4,5,6,7,8",
    "tslen": "8" }},
    "actions":[{"type": "SET_FIELD", "field": ["tpn","tsmap","tslen"],"value": ["2","1,2,3,4,5,6,7,8","8"]},{"type":
    "OUTPUT","port":"2"}]}' http://localhost:8080/stats/flowentry/add
```

7.2.2 URL

https://github.com/o3project/ryu-oe

7.2.3 Build

```
$ cd ~
$ git clone https://github.com/o3project/ryu-oe.git
$ cd ~/ryu-oe
$ sudo python ./setup.py install
```

7.2.4 Start and Stop

```
$ cd ~/ryu-oe/ryu/app
$ ryu-manager ./ofctl_rest.py
```

Press Ctrl+C to stop RYU-OE.

7.3 DummyOptNode (DON) with Openflowj-otn

7.3.1 What Is DummyOptNode

Although “OPT-Transport Apps of O3 Orchestrator& Controller Suite” contains control optical core nodes (OTN/WDM), it is difficult to obtain such nodes. DummyOptNode(DON) is a software application that emulates optical NE with OTN/WDM functions without data plane. DON is connected to RYU-OE, receives Flowmod, and shows the new cross connection status assigned by Flowmod.

Openflowj-otn is a software application that modified OpenFlowJ-Loxi(Ver0.9.0). New OTN parameters are have been added to the OpenFlow ver1.3 code. Openflowj-otn is used by DON in order to handle OpenFlow1.3 with OTN parameters.

1. DON sets up TCP session with the RYU OTN extension and receives the control command of the OpenFlow protocol.
2. DON analyzes the content of flowmod (please see Note below).
3. DON visualizes the ODU XC that is set up by flowmod.

Note: DON can only see the addition (OFPFC_ADD) of flow entry. It does not support deletion (OFPFC_MODIFY, OFPFC_MODIFY_STRICT) or modification (OFPFC_DELETE, OFPFC_DELETE_STRICT).

7.3.2 Environment

Operation is validated in the following environments.
OS: Windows 7 Professional SP1 (x64), Ubuntu desktop 12.04 (x64)
Middleware: Oracle Java VM jdk1.7.0_51, Maven 3

Note: This software works with OCNRM, RYU-OE, and PseudoMF.

7.3.3 URL

https://github.com/o3project/dummyOptNode

DON uses “OpenFlowJ OTN extension” as an external library. The library can be obtained by the following URL.
https://github.com/o3project/openflowj-otn

7.3.4 Build

It is necessary to build “OpenFlowJ OTN extension” before building DON because DON uses the extension for the external library.

```
$ git clone https://github.com/o3project/openflowj-otn.git
$ cd openflowj-otn
$ mvn clean install
$ git clone https://github.com/o3project/dummyoptnode.git
$ cd dummyoptnode
$ mvn clean install
$ tar xfvz ./target/dummyoptnode-1.0.0-bin.tar.gz –C ~/
```

You can choose your own install directory instead of ~/.

7.3.5 Configuration

Edit dummyoptnode-1.0.0/config.properties. In this document, parameters such as IP address/Port number in sample settings (see Figure 3) are used, for example.

```
$ vi ~/dummyoptnode-1.0.0/config.properties
```

Set RYU-OE's host & port for OpenFlow session.

```
 ofcHostname=127.0.0.1
 ofcPortNumber=6633
```

7.3.6 Start and Stop

Starting DummyOptNode.

```
$ cd ~/dummyoptnode-1.0.0/
$ java –jar dummyoptnode-1.0.0.jar
```

Stopping DummyOptNode.

```
 Just close Main GUI window.
```

7.3.7 Software Structure

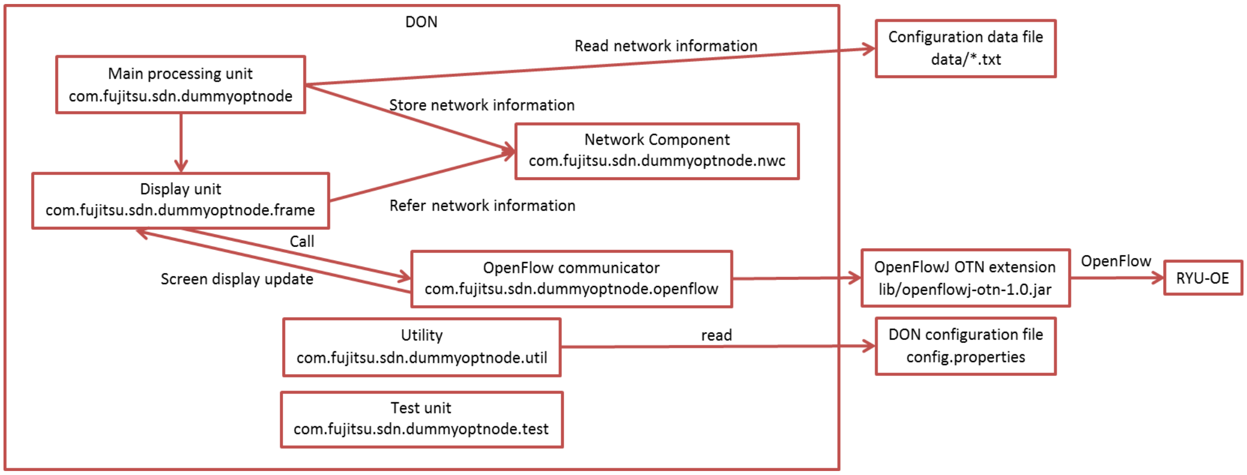

DON
Read network information
Configuration data file
data/*.txt
Main processing unit
com.fujitsu.sdn.dummyoptnode
Store network information
Network Component
com.fujitsu.sdn.dummyoptnode.nwc
Display unit
com.fujitsu.sdn.dummyoptnode.frame
Refer network information
Call
Screen display update
OpenFlow communicator
com.fujitsu.sdn.dummyoptnode.openflow
OpenFlowJ OTN extension
lib/openflowj-otn-1.0.jar
OpenFlow
RYU-OE
Utility
com.fujitsu.sdn.dummyoptnode.util
read
DON configuration file
config.properties
Test unit
com.fujitsu.sdn.dummyoptnode.test

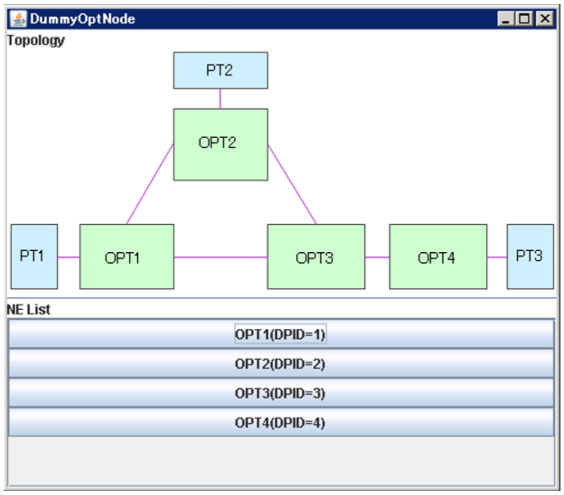
Figure 6 DummyOptNode software structure.

- DON obtains the network configuration information from the definition file in the data folder.

- DON communicates with RYU-OE (session initiation, receiving Flowmod, and sending/receiving Echo Request) using the OpenFlowJ OTN extension (openflowj-otn).

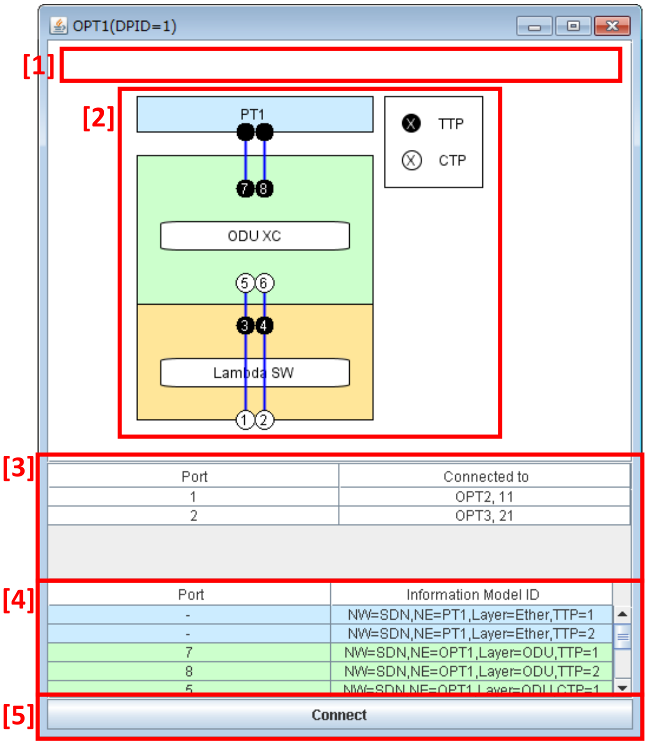
7.3.8 GUI IMAGE

7.3.8.1 Main GUI



Buttons are displayed according to the number of optical core nodes. When a button is clicked, an OptNode GUI is opened.
Network topology and node configurations can be displayed in the GUI if such images are prepared as an img/topology.png file.

7.3.8.2 OptNode GUI



Port	Connected to
1	OPT2, 11
2	OPT3, 21

Port	Information Model ID
-	NW=SDN,NE=PT1,Layer=Ether,TTP=1
-	NW=SDN,NE=PT1,Layer=Ether,TTP=2
7	NW=SDN,NE=OPT1,Layer=ODU,TTP=1
8	NW=SDN,NE=OPT1,Layer=ODU,TTP=2
5	NW=SDN,NE=OPT1,Layer=ODU,CTP=1

- [1] Flowmod parameter display.

The following parameters in match/actions are shown in this field. Only the newest Flowmod parameters are shown. Please refer to the log file when you want to see the old Flowmod parameters.

Parameters		Explanation
[match]	in_port	Input port number
	odu_sigtype	ODU SIGTYPE (Number)
	odu_sigid	ODU SIGID (TPN, tslen, tsmap)
[actions]	Output	Output port number
	odu_sigtype	ODU SIGTYPE (Number)
	odu_sigid	ODU SIGID(TPN, tslen, tsmap)

The values shown in TPN, tslen, and tsmap of ODU SIGID are extracted from the received packets and expressed by the following formats:

- TPN: decimal number
- tslen: decimal number
- tsmap: binary number (bitmap)

- [2] Internal connection display

It shows the connection of ODU XC according to Flowmod.

- [3] Adjacent node/port display

It shows the port lists connecting to the other NE.

Row	Content
Port	Source port
Connected to	The NE and port number of destination port E.g.: OPT1,11

- [4] Internal ID Display

It shows the internal ID that corresponds to the port number shown in the figure. Internal ID is expressed as an Information model ID that is based on the internal naming rule in RM and PseudoMF.

- [5] Connect button

When you click the Connect button, RYU will be connected.

8 GETTING STARTED

8.1 OUTLINE

- Sample target network configuration
- Start programs
- Confirm the status of optical core nodes
- Example: Requesting optical core resources

8.2 SAMPLE TARGET NETWORK CONFIGURATION

A definition file that includes the sample network configuration here is included. By changing the definition file, the different network can be configured. How to change the definition file will be shown in the near future.

- Topology

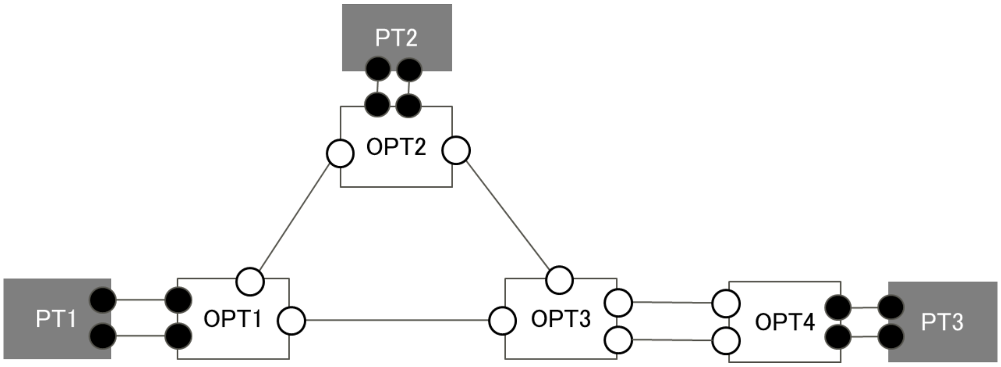


Figure 7 Sample target network topology.

3-packet transport node (PT#1-3) is connected
to the optical core network that consists of 4 nodes (OPT#1-4).

- OCh path set-up

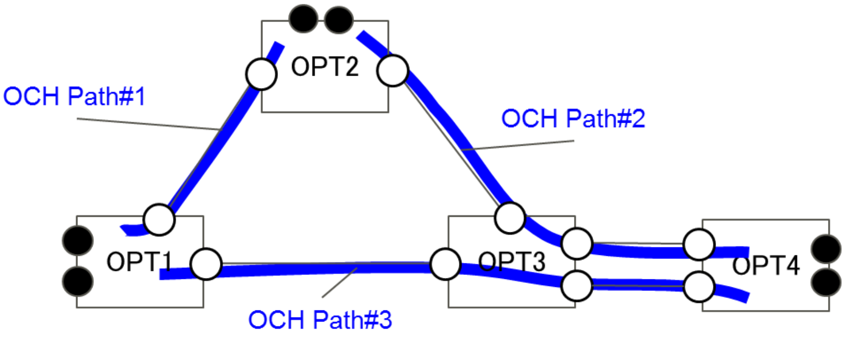


Figure 8 OCh path set-up in sample target network topology.

It is assumed that 3 OCh paths are set up in the optical core network.

Files that define the sample network are shown below.

Table 1 Network resource definition files.

File name	Content	Application using the file
Och_link.txt	OCh path configuration information	DummyOptNode `~/dummyoptnode-1.0.0/data`
ll1.txt	Inter-layer boundary information (ODU layer and OCh layer)	DummyOptNode `~/dummyoptnode-1.0.0/data`
ll2.txt	Inter-layer boundary information (packet transport node and ODU layer)	DummyOptNode `~/dummyoptnode-1.0.0/data`

idex.txt	ID conversion information	PseudoMF, DummyOptNode
		`~/dummyoptnode-1.0.0/data`
		`~/pseudoMF-1.0.0/data`

lower.txt	Lower layer (OCh) resource information	PseudoMF `~/pseudoMF-1.0.0/data`
upper.txt	Upper layer (ODU) resource information	PseudoMF `~/pseudoMF-1.0.0/data`
route.txt	Route information	PseudoMF `~/pseudoMF-1.0.0/data`

NOTE: Please don’t change file name.

8.3 START PROGRAMS

- Start ODENOS.

```
$ odenos_install_dir/odenos –s
```

- Start OCNRM.

```
$ ~/ocnrm-1.0.0-bin/ocnrm_mn –s
```

- Start PseudoMF.

```
$ ~/pseudoMf-1.0.0-bin/pseudoMf.sh -s
```

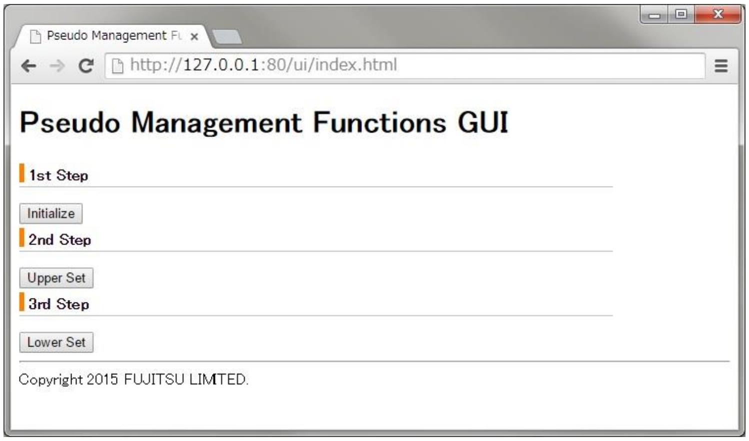
- System configuration (by executing a prepared shell script).

```
$ ~/ocnrm-1.0.0-bin/rm_startup.sh
```

  - Create component managers
  - Create network components
  - Create and connect to OptDriver
  - Create and connect to LinkLayerizer
  - Set up boundary conditions for LinkLayerizer

- Allocate initial resources (L0,L1,L2) using PseudoMF GUI.

1. Access PseudoMF via GUI (http://127.0.0.1:80/ui/index.html)
2. Push GUI 1st , 2nd, 3rd buttons in order



Pseudo Management Functions GUI

1st Step

2nd Step

3rd Step

Copyright 2015 FUJITSU LIMITED.

- ➢ Start RYU-OE.

```
$ ryu-manager ~ryu-oe/ryu/app/ofctl_rest.py
```

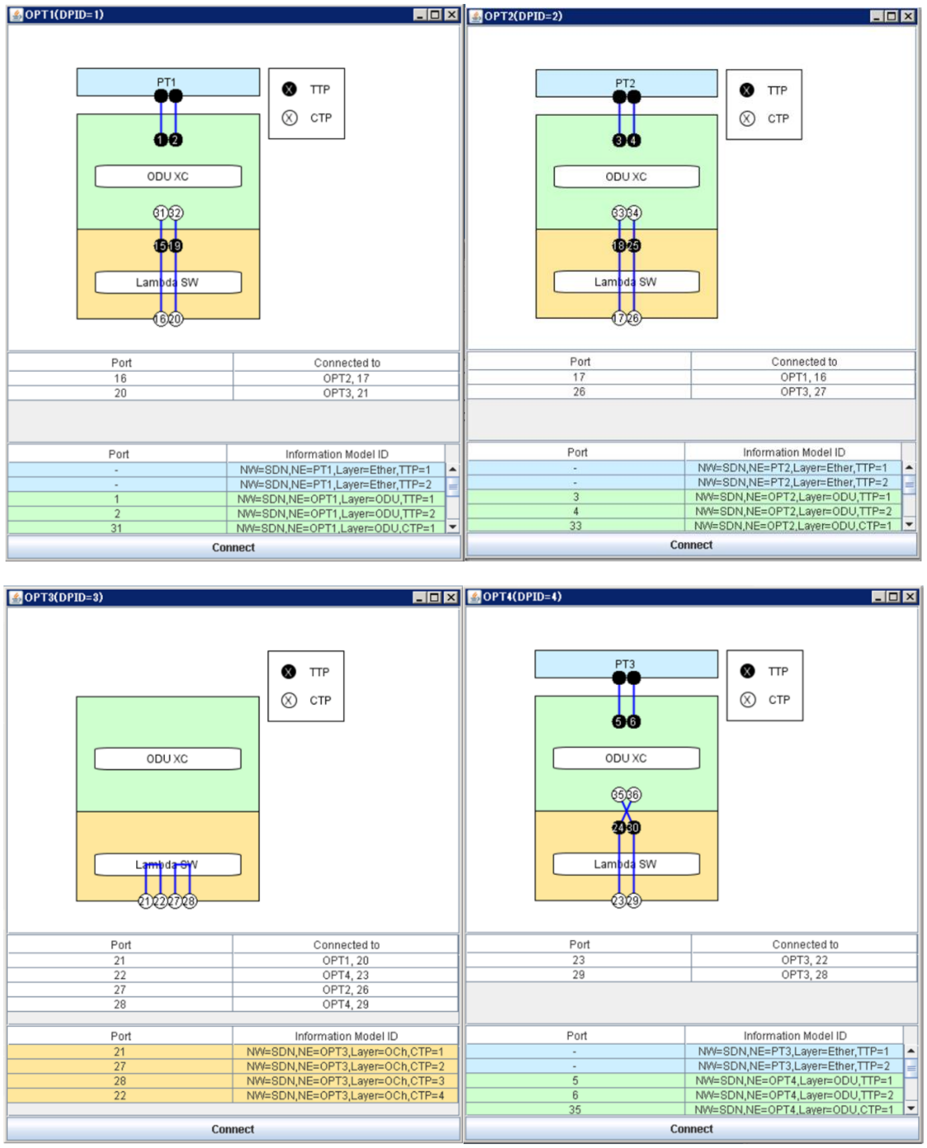
- ➢ Start Dummy Opt Node and connect to RYU-OE.

1. $ cd ~/dummyoptnode-1.0.0/
   $ java –jar dummyoptnode-1.0.0.jar
2. Push button with network entity name of NE List in Main GUI
3. Push connect button at each OptNode GUI

8.4 CONFIRM STATUS OF OPT NODE

Confirm initial set-up of Opt Node via DummyOptNode GUI.

- You can see 4 core nodes with sample configuration.
- You can see connection between OPT node and PT node and initial connections of 3 OCh paths (bidirectional)
- Port numbers shown in DummyOptNode GUI are the same as those used in OpenFlow messages.



OPT1(DPID=1)

Port	Connected to
16	OPT2, 17
20	OPT3, 21

Port	Information Model ID
-	NW=SDN,NE=PT1,Layer=Ether,TTP=1
-	NW=SDN,NE=PT1,Layer=Ether,TTP=2
1	NW=SDN,NE=OPT1,Layer=ODU,TTP=1
2	NW=SDN,NE=OPT1,Layer=ODU,TTP=2
31	NW=SDN,NE=OPT1,Layer=ODU,CTP=1

OPT2(DPID=2)

Port	Connected to
17	OPT1, 16
26	OPT3, 27

Port	Information Model ID
-	NW=SDN,NE=PT2,Layer=Ether,TTP=1
-	NW=SDN,NE=PT2,Layer=Ether,TTP=2
3	NW=SDN,NE=OPT2,Layer=ODU,TTP=1
4	NW=SDN,NE=OPT2,Layer=ODU,TTP=2
33	NW=SDN,NE=OPT2,Layer=ODU,CTP=1

OPT3(DPID=3)

Port	Connected to
21	OPT1, 20
22	OPT4, 23
27	OPT2, 26
28	OPT4, 29

Port	Information Model ID
21	NW=SDN,NE=OPT3,Layer=OCh,CTP=1
27	NW=SDN,NE=OPT3,Layer=OCh,CTP=2
28	NW=SDN,NE=OPT3,Layer=OCh,CTP=3
22	NW=SDN,NE=OPT3,Layer=OCh,CTP=4

OPT4(DPID=4)

Port	Connected to
23	OPT3, 22
29	OPT3, 28

Port	Information Model ID
-	NW=SDN,NE=PT3,Layer=Ether,TTP=1
-	NW=SDN,NE=PT3,Layer=Ether,TTP=2
5	NW=SDN,NE=OPT4,Layer=ODU,TTP=1
6	NW=SDN,NE=OPT4,Layer=ODU,TTP=2
35	NW=SDN,NE=OPT4,Layer=ODU,CTP=1

Figure 9 Initial status with sample configuration.

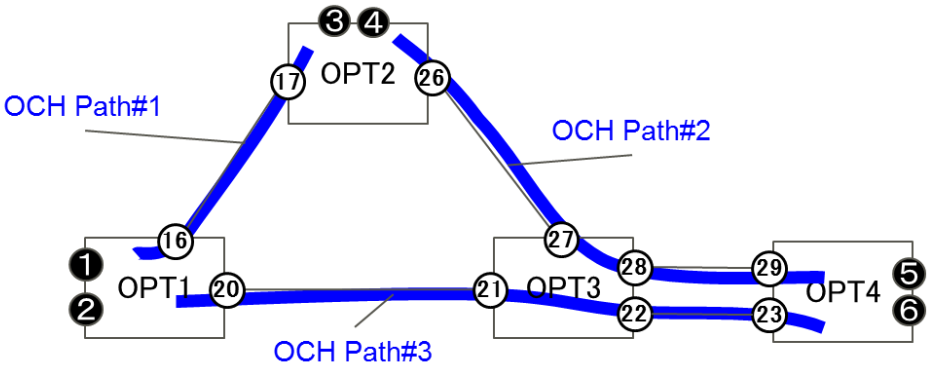


Figure 10 OCh path set up in sample target network topology with port numbers for GUI.

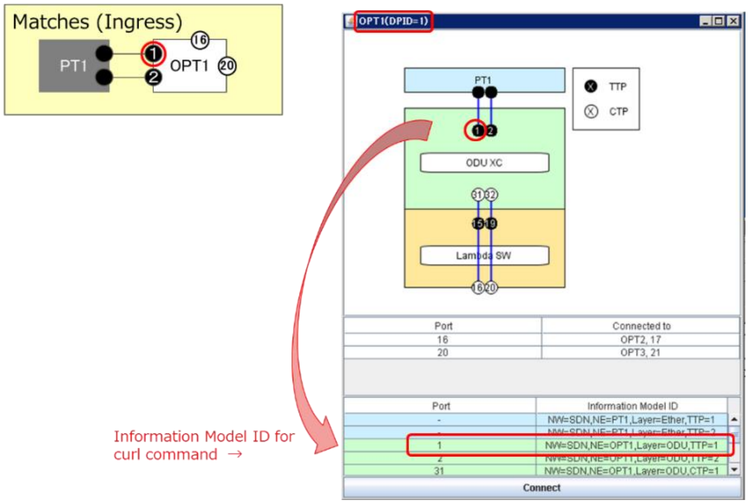
8.5 EXAMPLE: REQUESTING OPTICAL CORE RESOURCES (1)

< Create LO ODU path without delay constraints >

The following example shows how to create LO ODU (optical core resource to which client signal is mapped) without delay constraints on demand.

Curl command is used to request a flow (Flow1) in a network component in the ODU layer (L01).

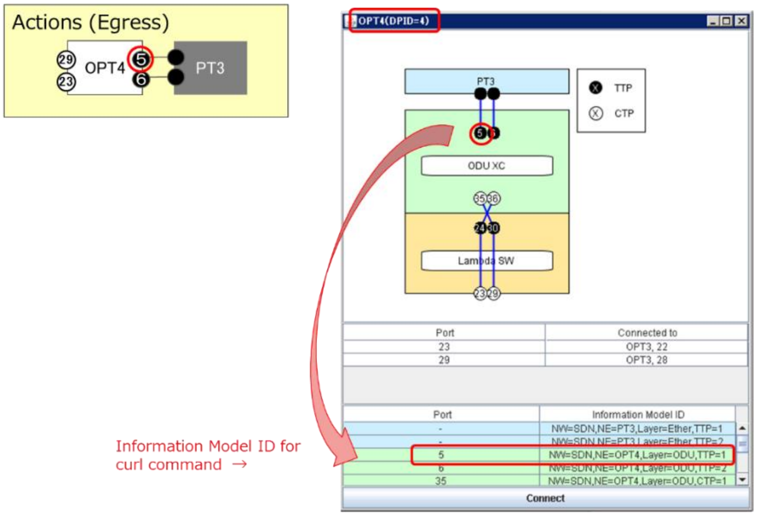
- Designate port 1 of ingress node (OPT1) as a match and port 5 of egress node (OPT4) as edge_actions. (Caution: It's necessary to designate the Information Model ID of those ports, e.g., "NW=SDN,NE=OPT1,Layer=ODU,TTP=1".)
  - "in_node":"NW=SDN,NE=OPT1","in_port":"NW=SDN,NE=OPT1,Layer=ODU,TTP=1"
  - "output":"NW=SDN,NE=OPT4,Layer=ODU,TTP=1"



Port	Connected to
16	OPT2, 17
20	OPT3, 21

Port	Information Model ID
-	NW=SDN,NE=PT1,Layer=Ether,TTP=1
-	NW=SDN,NE=PT1,Layer=Ether,TTP=2
1	NW=SDN,NE=OPT1,Layer=ODU,TTP=1
2	NW=SDN,NE=OPT1,Layer=ODU,TTP=2
31	NW=SDN,NE=OPT1,Layer=ODU,CTP=1

Figure 11 Ingress port name used for LO ODU creation.



Port	Connected to
23	OPT3, 22
29	OPT3, 28

Port	Information Model ID
-	NW=SDN,NE=PT3,Layer=Ether,TTP=1
-	NW=SDN,NE=PT3,Layer=Ether,TTP=2
5	NW=SDN,NE=OPT4,Layer=ODU,TTP=1
6	NW=SDN,NE=OPT4,Layer=ODU,TTP=2
35	NW=SDN,NE=OPT4,Layer=ODU,CTP=1

Figure 12 Egress port name used for LO ODU creation.

- Bandwidth [Mbps] and delay [msec] can be designated. Bandwidth is fixed at 10,000 (ODU0 is assumed).
  - "req_bandwidth":"10000","req_latency":"9999"
- Setting the status “Establishing” means that flows are to be created.
  - "status":"establishing"

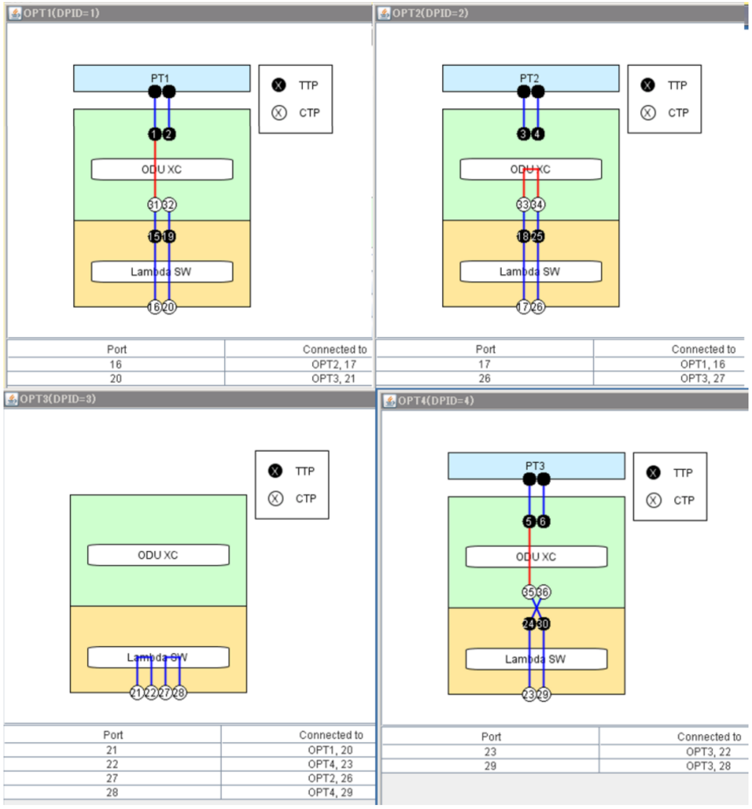
- Command (ODENOS REST API)
  This command is very long, so please copy and paste below:

```
curl -X PUT http://127.0.0.1:10080/L01/flows/Flow1 -d '{"type":"BasicFlow","version":"1","flow_id":"Flow1","owner":"ANY","enabled":true,"status":"establishing","matches":[{"type":"BasicFlowMatch","in_node":"NW=SDN,NE=OPT1","in_port":"NW=SDN,NE=OPT1,Layer=ODU,TTP=1"}],"path":[],"edge_actions":{"NW=SDN,NE=OPT4":[{"type":"FlowActionOutput","output":"NW=SDN,NE=OPT4,Layer=ODU,TTP=1"}]},"attributes":{"direction":"unidirectional","req_bandwidth":"10000","req_latency":"9999"}}'
```

[Reference] https://github.com/o3project/odenos/blob/develop/doc/api/Network.md#PUTflow_id

- Confirmation

In the DummyOptNode GUI, the Flowmod command is shown at the upper side and the new setting of the cross connection is indicated by a red line. LO ODU is created between the designated ports by ODU XC and OCh paths no. 1 and 2.



OPT1(DPID=1)

Port	Connected to
16	OPT2, 17
20	OPT3, 21

OPT2(DPID=2)

Port	Connected to
17	OPT1, 16
26	OPT3, 27

OPT3(DPID=3)

Port	Connected to
21	OPT1, 20
22	OPT4, 23
27	OPT2, 26
28	OPT4, 29

OPT4(DPID=4)

Port	Connected to
23	OPT3, 22
29	OPT3, 28

Figure 13 GUI showing the cross connection (red line) set up for the request (1).

8.6 EXAMPLE: REQUESTING OPTICAL CORE RESOURCES (2)

< Create LO ODU path with small delay >

The following example shows how to create LO ODU with delay constraints.

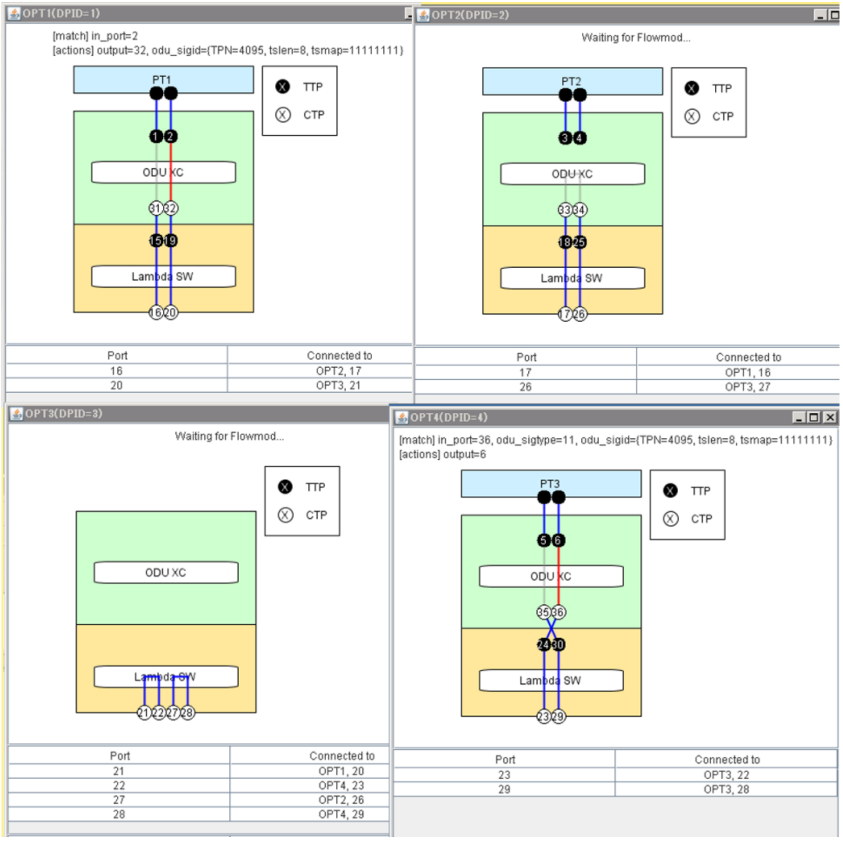
This can be done simply by changing the QoS parameters using the request in 8.5. Curl command is used to set up a flow (Flow2) in the network component in the ODU layer (networkcomponent01).

- Command

```
curl -X PUT http://127.0.0.1:10080/L01/flows/Flow2 -d '{"type":"BasicFlow","version":"1","flow_id":"Flow1","owner":"ANY","enabled":true,"status":"establishing","matches":[{"type":"BasicFlowMatch","in_node":"NW=SDN,NE=OPT1","in_port":"NW=SDN,NE=OPT1,Layer=ODU,TTP=2"}],"path":[],"edge_actions":{"NW=SDN,NE=OPT4":[{"type":"FlowActionOutput","output":"NW=SDN,NE=OPT4,Layer=ODU,TTP=2"}]},"attributes":{"direction":"unidirectional","req_bandwidth":"10000","req_latency":"10"}}'
```

- Confirmation

In the DummyOptNode GUI, a Flowmod command is shown at the upper side and the new setting of the cross connection is indicated by the red line. LO ODU is created between the designated ports by ODU XC and OCh path no. 3.



OPT1(DPID=1)

```
[match] in_port=2
[actions] output=32, odu_sigid={TPN=4095, tslen=8, tsmap=11111111}
```

Port	Connected to
16	OPT2, 17
20	OPT3, 21

OPT2(DPID=2)

Waiting for Flowmod...

Port	Connected to
17	OPT1, 16
26	OPT3, 27

OPT3(DPID=3)

Waiting for Flowmod...

Port	Connected to
21	OPT1, 20
22	OPT4, 23
27	OPT2, 26
28	OPT4, 29

OPT4(DPID=4)

```
[match] in_port=36, odu_sigtype=11, odu_sigid={TPN=4095, tslen=8, tsmap=11111111}
[actions] output=6
```

Port	Connected to
23	OPT3, 22
29	OPT3, 28

Figure 14 GUI showing the cross connection (red line) set up for the request (2).