Design of LINC-OE

Sandhya Narayan
Infoblox Inc.
[work in progress]

Introduction

This document outlines a design for Optical Switch Emulation described in [1] . LINC-OE is an optical switch emulator which supports extensions to Openflow v1.3 to describe and control optical ports, cross-connect and links. It works with Openflow controllers that support the same extensions.

Development of LINC-OE is expected to be done in three phases. Currently we are focused on completing Phase 1 development by October 1. The scope of the extensions are limited to the needs of Phase 1. This document provides an initial design and setup that can meet the requirements specified in [1].

Phase I

LINC-OE Architecture

Figure 1 shows a LINC-OE switch which contains many logical switches and is configured to emulate a simple optical network. LINC-OE uses LINC switch as a foundation with some modifications discussed below to support emulation of optical network.

We consider LINC-OE as LINC with a new backend, similar to the current user-space implementations (us3, us4, us5 etc.), but having additional ability to move packets (Erlang terms) on special ports called W-Ports, which simulate optical ports. We can call this “linc_us4_oe” . Our code development can then follow what was done going from us3 to us4 and us5. This way we believe that there will not be much change to code in the logical switch.

We use LINC’s logical switch capability to emulate a number of optical switches (called E-ROADMs in Figure 1) interconnected according to some given optical network topology. A config file will provide such topology information and port configurations for the logical switches (actual details will be provided by On.Lab). For our implementation a key difference between optical transport and Ethernet is that an optical transport port can have more than one independent communication channel. Each channel is identified by a wavelength (Lambda) and is separately routable to another port.

LINC-OE will need to maintain a directory of optical connections of the form {E-ROADMa, W-Port1} <->{ E-ROADMb, WPort2}. We would like to use the Map-server for this purpose, but shared ETS table will do for now. Whenever a packet is put on egress W-Port1 of E-ROADMa,

an Erlang message is sent to E-ROADMb. The message will contain the original packet and optical information such as Lambda being used.

T-Ports or transponder ports (shown by green lines) connect the optical network to the packet network. The actual connection between the two is completed by the installation of flow-entries in the E-ROADMs via OF v1.3 experimenter messages.

The emulated optical network gets formed by flow-entries in E-ROADMs to create cross-connects (shown as dotted blue lines) between W-Port$_\lambda$s inside E-ROADMs and Erlang message passing between W-Port of one E-ROADM to W-Port of another (instead of Ethernet packets). The E-ROADMs and the packet OF switches together form a multi-layer packet optical network that can be controlled by an Openflow controller.

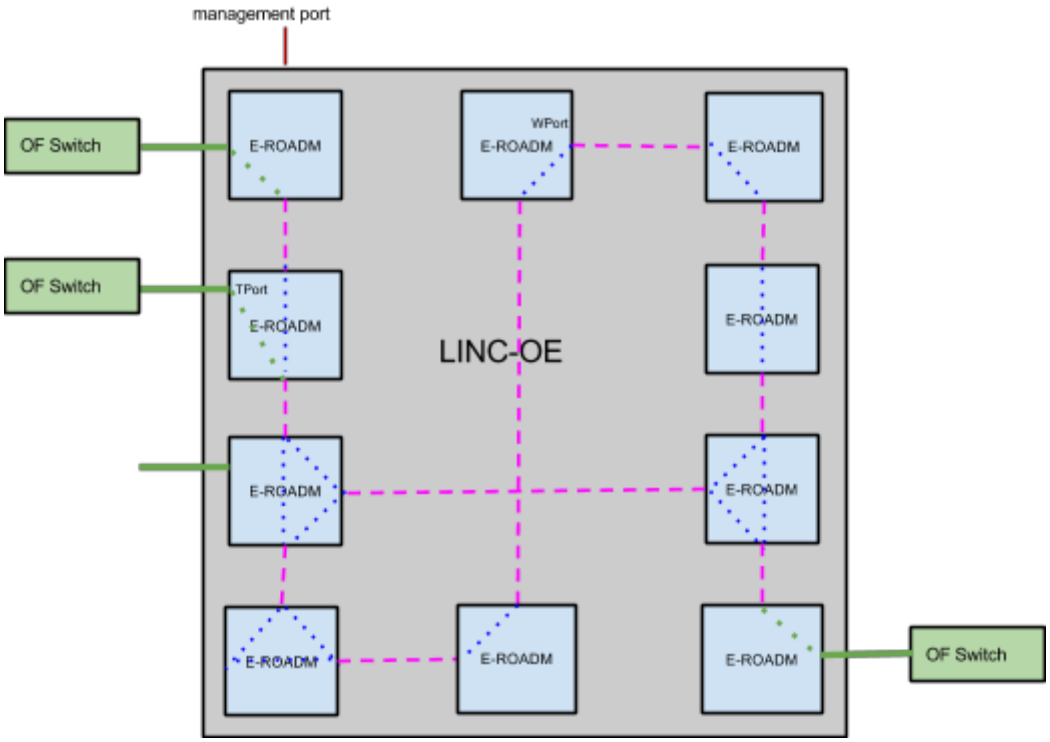

Figure 1

From LINC to LINC-OE

The major changes made to LINC are listed below:

- Support OF v1.3 experimenter messages for optical transport as specified in [2].
- Use Erlang message passing between special ports in LINC called W-Ports to simulate optical links (shown in pink dashed lines).

- Support optical forwarding from ingress W-Port to egress W-Port by installing Openflow match-action flow entries.
- Support add/drop of packet to optical traffic and vice versa by use of cross-connects installed by means of Openflow match-action flow entries.
- Make changes to support special utilities to make and break optical links, turn off and bring up LINC logical switches.
- Provide utility to convert a network topology file into a configuration file (sys.config) for LINC-OE.

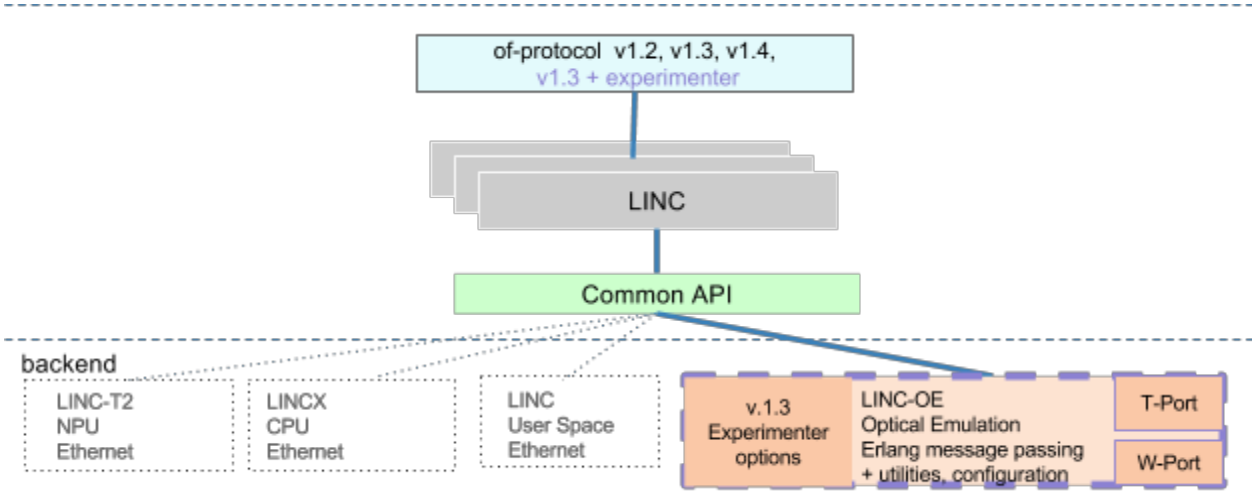

Figure 2

Use of experimenter match and action to emulate optical cross connect

- When a packet comes in on T-Port of a E-ROADM:

  Match inport =T-Port, Action= outport ({W-Port, Lambda}) -> Make an emulated-packet with Lambda and original packet, send to W-Port (which sends an Erlang message containing the emulated-packet to a “connected” ROADM.

- When a packet comes in on a W-Port of a ROADM (Erlang message received from another ROADM), we send it either to the T-Port or across the cross-connect to a W-Port:

  Match inport ={W-Port, Lambda}, Action= outport {T-Port} -> Unpack emulated-packet, match Lambda, send original packet to T-Port if matched

  Match inport ={W-Port1, Lambda1}, Action= outport ({W-Port2, Lambda2}) -> unpack emulated-packet, match Lambda1, if matched then change to Lambda2, make new emulated-packet and send to W-Port2.

Note that in Phase I we will allow any ingress Lambda to use any other egress Lambda, provided that Lambda is not already in use. In actual practice this may not be allowed unless there is a regenerator available in the ROADM, but that will be work for Phase 2.

Experimenter Flow-mod message

The optical emulation has restrictions on allowed flows. When the controller tries to create a flow that violates one of these restrictions, LINE-OE returns an a flow-mod error message to the controller and does not create the flow.

1. A Lambda on a egress W-Port may only be used in one flow as a egress W-Port
2. match action to write to a W-Port must have a Lambda (e.g., match actions on ingress T-Port must set a lambda to write to a W-Port)
3. T-Port and W-Port match actions can have only one W-Port write action (i.e., flow cannot be duplicated)

OF v1.3 extensions

OF v1.3 extensions based on OF v1.5 proposal take advantage of experimenter messages:

- multipart request/reply for port descriptions
- match and match action to match on lambda
- experimenter message for port status change remains compatible with OF v1.3

Mechanisms to emulate failures

LINC-OE has to provide special utility programs to create failures listed below.

- Emulated optical links
- Emulated T-Port
- ROADM

Demo Network

(copying from requirements document [1])
1. *m* client side T-ports*
2. *n* network W-Ports*
3. Operation state for T-Port and W-Port
4. Each W-Port can have *k* lambda's.
5. Each Lambda is assumed to carry 100G Bandwidth.

*Sample values: m=4, n=10, k=10

Future Work

- Regenerator functionality
- Ability to emulate Channel Signal/Power Measurement
- Scale
- Layer 1
- Mininet Integration

References

[1] OpticalSwitchEmulatorRequirementsV2.pdf
[2] OpenFlow Protocol Extensions for LINC-OE
[3] https://rs.opennetworking.org/wiki/display/PUBLIC/ONF+Registry
Experimenter ID = 0x74:87:71 - Infoblox Inc.