Software Defined Networking for access networks

ALIEN

Richard G. Clegg, Imperial College, London

Co authors: Jason Spencer,Raul Landa, Manoj Thakur, John Mitchell, Miguel Rio

Talk to EWSDN 2014

(Prepared using LaTeX and beamer.)

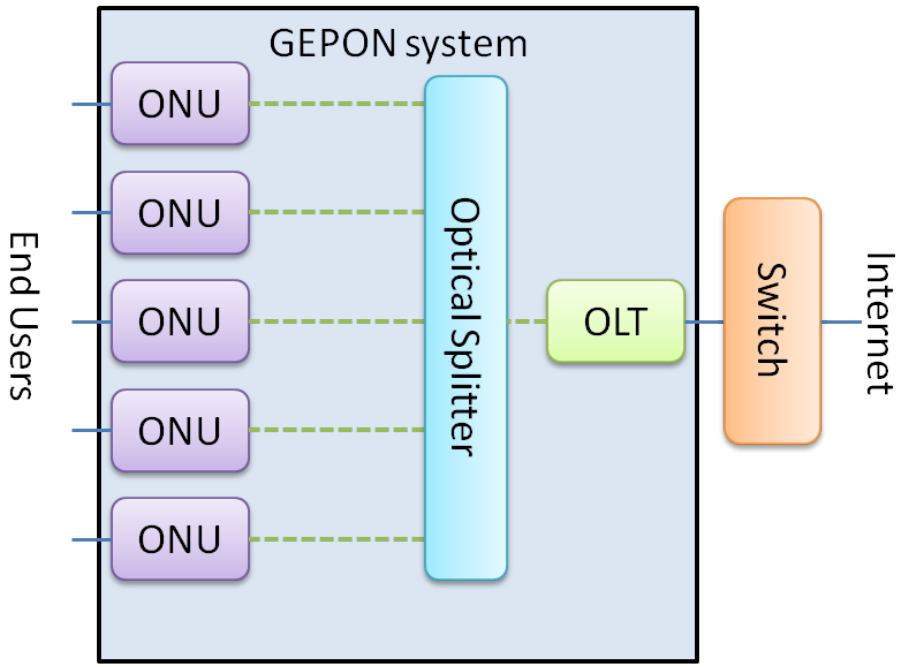
Introduction

SDN for access equipment

Access equipment: relatively modern, slow replacement cycles, involves end-user equipment, not typically programmable. How can it be made OpenFlow ready.

- Equipment cannot be programmed (custom chip, not documented).
- Typically access equipment has a “head end” device and tail end consumer premesis units.
- Replacing it is difficult, but want the benefits of OpenFlow at this part of the network.

The GEPON (Gigabit Ethernet Passive Optical Network)



Getting OpenFlow on GEPON

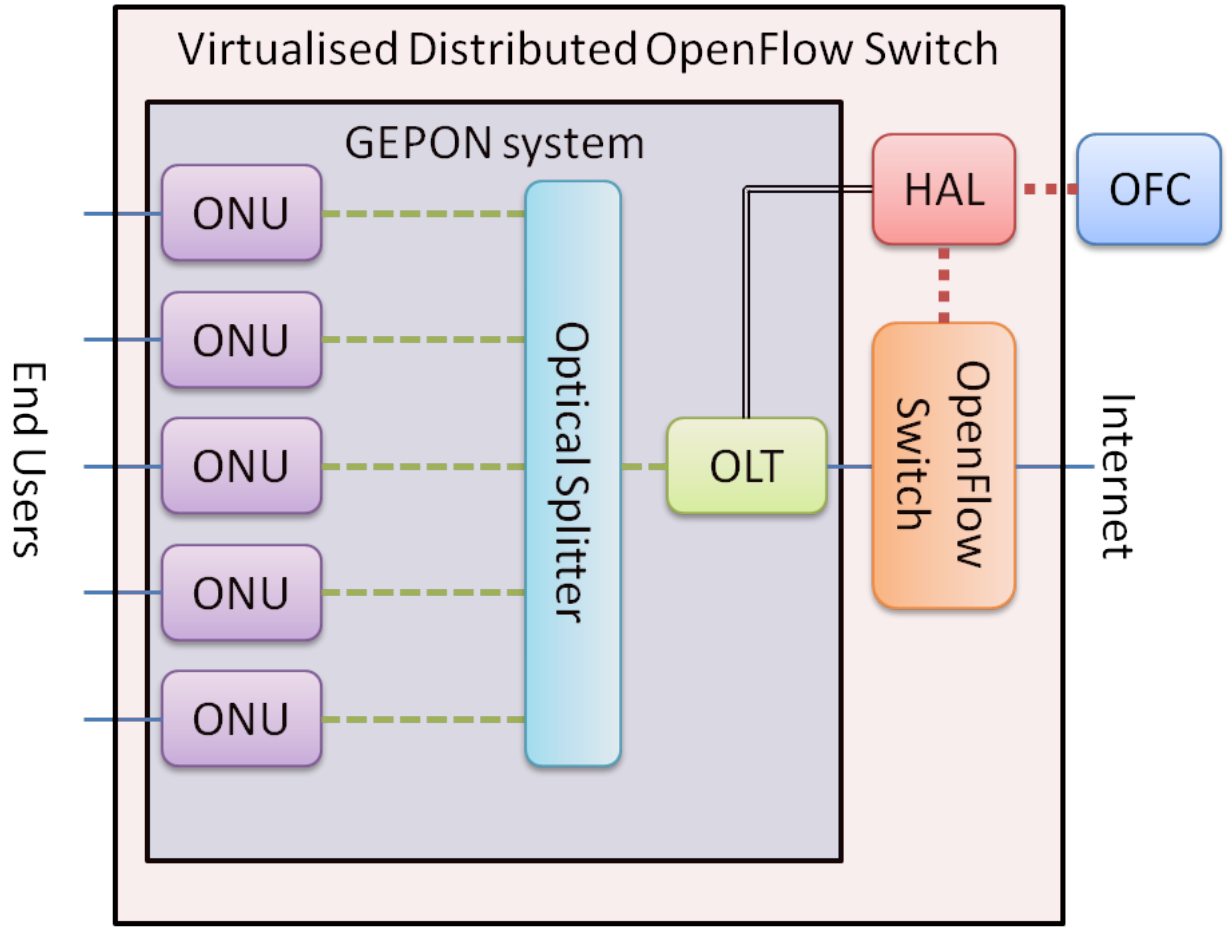
- Want to make whole system of switch, OLT, splitter ONU present as single massively distributed OF switch.
- Problem with making GEPON OpenFlow capable:
  - Proprietary device, no knowledge of chips or drivers.
  - Not intended as programmable.
  - ONU cheap low power consumer unit.
- Solution:
  - Make use of fact that OLT usually needs switch before it anyway.
  - Give OLT Open Flow capable front end switch (xdpd on NetFPGA).
  - OLT can switch to ONU on VLANs – e.g. VLAN 10 goes to ONU 4.
  - Use a mapping to VLANs in lower level open flow switch.
  - Controller sees only higher level abstract switch.

Mapping physical ports and tags

- Create a virtual OpenFlow switch with one port for the head-end device and one port for each tail-end device.
- Each virtual port on the virtual OpenFlow switch maps to exactly one physical port and possibly a tag.
- Here we talk in terms of VLAN tag but any alteration that OF can match on will do.

$$
\begin{aligned}
P_1, T_1 &\leftrightarrow V_1 \\
P_1, T_2 &\leftrightarrow V_2 \\
P_1, T_3 &\leftrightarrow V_3 \\
P_1, T_4 &\leftrightarrow V_4 \\
P_1, T_5 &\leftrightarrow V_5 \\
P_2 &\leftrightarrow V_6
\end{aligned}
$$

The GEPON with OpenFlow



- Note – approach is generic – could work for many access devices.
- Another approach would be to surround the device with OF switches.

Design in detail

Target is OF 1.0

For simplicity we chose to implement OpenFlow 1.0 but there is no reason we could not implement later versions with the same approach.

Front end switch

Any OpenFlow capable switch can sit at the front end of the OLT. This is easier for an ISP to replace than any other part of the system.

Hardware abstraction layer

We abstract the hardware using xCPd – the eXtensible Control Path daemon, written to translate the control path.

eXtensible Control Path daemon xCPd

- Based on xDPd and ROFL libraries.
- Translates OpenFlow messages to/from switch to/from controller.
- E.g. Action output a packet on port 4 may becomes actions to tag the packet with VLAN 10 and output to port 2.
- E.g. A PacketIn from port 2 with VLAN 10 becomes an untagged packet from port 4.

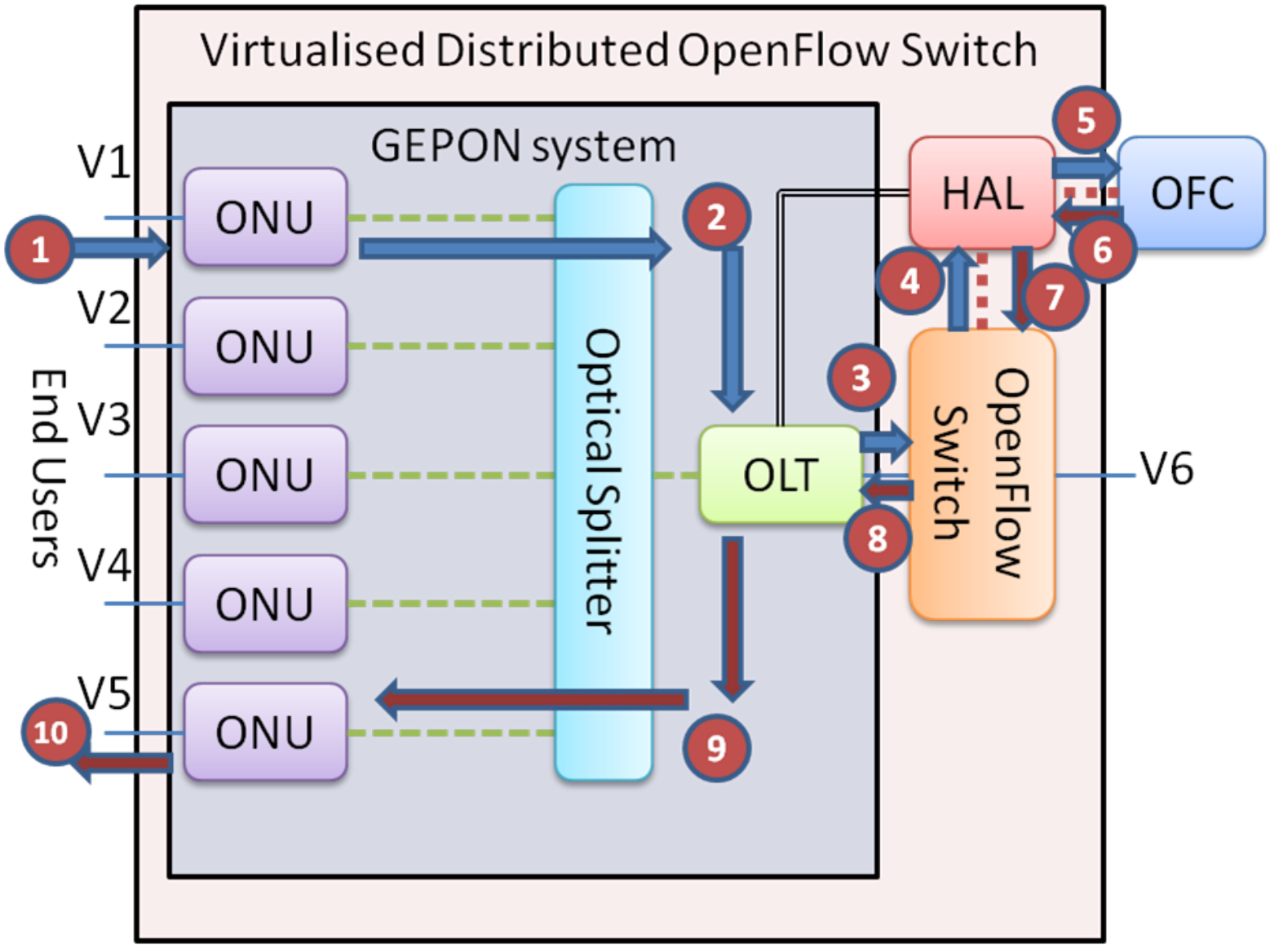
ROFL libraries

The Revised OpenFlow Library abstracts OpenFlow concepts as C++ objects and methods. Not your standard OFC northbound – targetted at data path implementations or building new controllers.
`https://github.com/bisdn/rofl-core`

xCPd in detail

- Read and store a mapping of real ports and vlan tags to virtual ports.
- Store all flowmods on the switch in their ‘original’ and ‘translated’ forms so translated flowmods can be deleted.
- Connect to the switch and to the OFC as transparently as possible.
- Forward messages from switch to OFC (suitably translated).
- Forward messages from OFC to switch (suitably translated).
- Where translation not possible provide way to get messages to access device to implement.

The packet's big adventure



Results

Unit tests

We pass them – lots of them. Oftest a standard but often not easy to work with `http://www.projectfloodlight.org/oftest/`

- Oftest basic tests passed, many other action and match tests pass.
- Sometimes underlying switch does not pass oftest tests.
- Sometimes test failure is to do with assumptions oftest makes about timing not a real failure.
- Sometimes test failes for "wrong" reason – e.g. oftest PacketIn test also test VLAN tags.

OFtest basic tests

Test	**Result**
Echo	Passes using xCPd
EchoWithData	Passes using xCPd
PacketIn	Fails due to VLAN tag (would pass with QinQ)
PacketInBC	Passes using xCPd
PacketOut	Passes using xCPd port mapping
PacketOutMC	Passes using xCPd port mapping
FlowStatsGet	Passes using xCPd port mapping
TableStatsGet	Passes using xCPd
DescStatsGet	Passes using xCPd
FlowMod	Passes using xCPd port mapping
PortConfigMod	Requires hardware specific code
PCMErr	Requires hardware specific code
BadMessage	Passes using xCPd
TableModConfig	Passes using xCPd

OF 1.0 commands by status

- Aim was not to get a single GEPON working but to provide a method for any sufficient access device.
- OF1.0 commands split into four groups:
  1. Automatic pass: Hello, EchoRequest, EchoReply, Vendor, FeaturesRequest, FeaturesReply, BarrierRequest, BarrierReply, GetConfigRequest, GetConfigReply, SetConfig, PortStatus and QueueGetConfigReply.
  2. Require translation: PacketIn, PacketOut, StatsRequest/StatsReply (for a FlowMod), Error and FlowRemoved.
  3. Require hardware specific code: PortMod, StatsRequest/StatsReply (for a Port) and QueueGetConfigRequest.
- Can never work without QinQ – Action VLAN tag/untag, match on VLAN (with Q-in-Q these should just work).

Porting to your device (requirements for new hardware/software)

- All traffic between tail-end devices must travel via the head-end OF enabled switch.
- The OpenFlow switch can add the tag and match any packet against that tag in combination with the real underlying port.
- The head-end device can add the tag to any packet entering the head-end device from a tail end device.
- The head-end device can route the packet according to the tag and remove the tag.
- Without Q-in-Q then you cannot use all of the tag mechanism that is used to signal.
- New equipment: replace head-end switch with OF capable switch.
- New software: Run xCPd, probably on same machien where OFC runs.

Porting to your device (code you need to write)

- Code to query the head-end device for port statistics specific to the tail-end devices – without this port stats are for underlying physical port (combines several virtual ports).
- Code which can modify the links to the end user devices as required by the OpenFlow PortMod command.
- Queue commands either:
  1. Ensure that a user configuring the queue on the underlying hardware also gives xCPd the same information.
  2. Write code to query the head-end device about its queue configurations (this is optional in the OF 1.0 specification).
- Possible extra: Add automatic detection of new tail-end devices as new ports.

Conclusions: Lessons learned

- Implementing the whole protocol is hard.
- OpenFlow commands in this approach group to:
  - Just work – no changes.
  - Work with tags – VLAN tags added/removed.
  - Just don't work – need hardware commands.
  - Will never work – VLAN tags when no QinQ available.
- ROFL is an excellent place to start if you need more than "just another OpenFlow northbound".
  `https://github.com/bisdn/rofl-core`.
- Provides sensible abstractions not just for messages to/from controller.
- OpenFlow is actually really fun to play with despite the pain.
- More details in EWSDN paper, preprint:
  `http://www.richardclegg.org/access_sdn`.
- Our code is available:
  `https://github.com/richardclegg/xcpd`