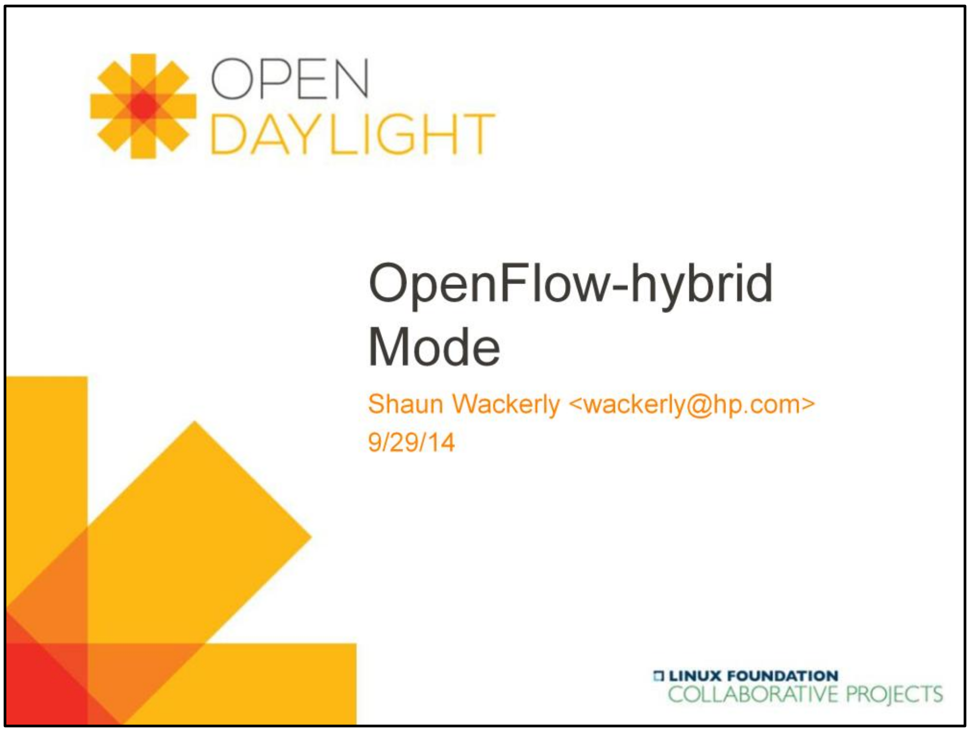
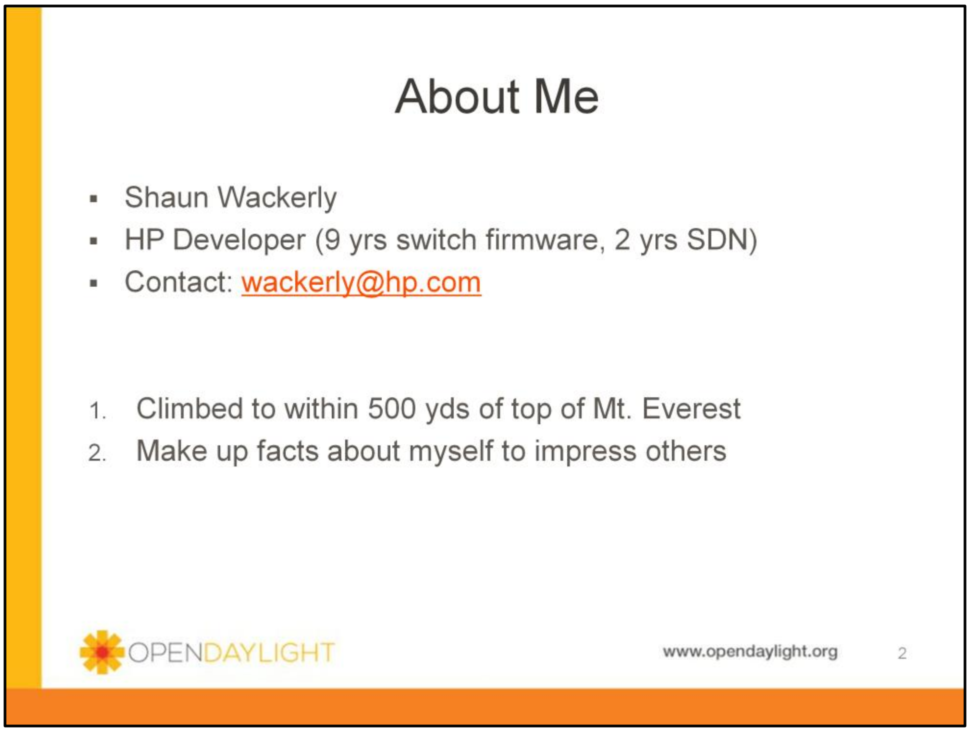


OPEN DAYLIGHT

OpenFlow-hybrid Mode

Shaun Wackerly <wackerly@hp.com>
9/29/14

LINUX FOUNDATION COLLABORATIVE PROJECTS



About Me

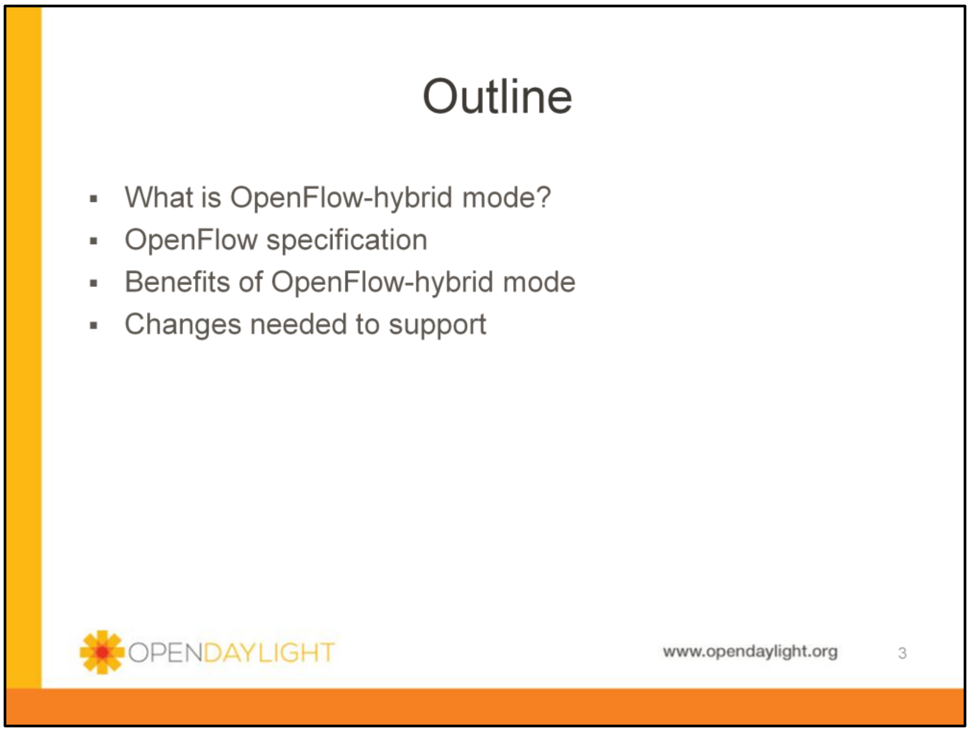
- Shaun Wackerly
- HP Developer (9 yrs switch firmware, 2 yrs SDN)
- Contact: wackerly@hp.com

1. Climbed to within 500 yds of top of Mt. Everest
2. Make up facts about myself to impress others

OPENDAYLIGHT www.opendaylight.org 2

A little about myself since I am a new face to most of you. My name is Shaun Wackerly and I've been an HP Developer for the past 11 years. I developed switch firmware for 9 years, then transitioned to work in SDN for the past 2 years.

I have two facts to share about myself. First, a few years ago I climbed to within 500 yards of the top of Mt. Everest. Second, I make up facts about myself to impress others. Let's move on to the technical stuff.

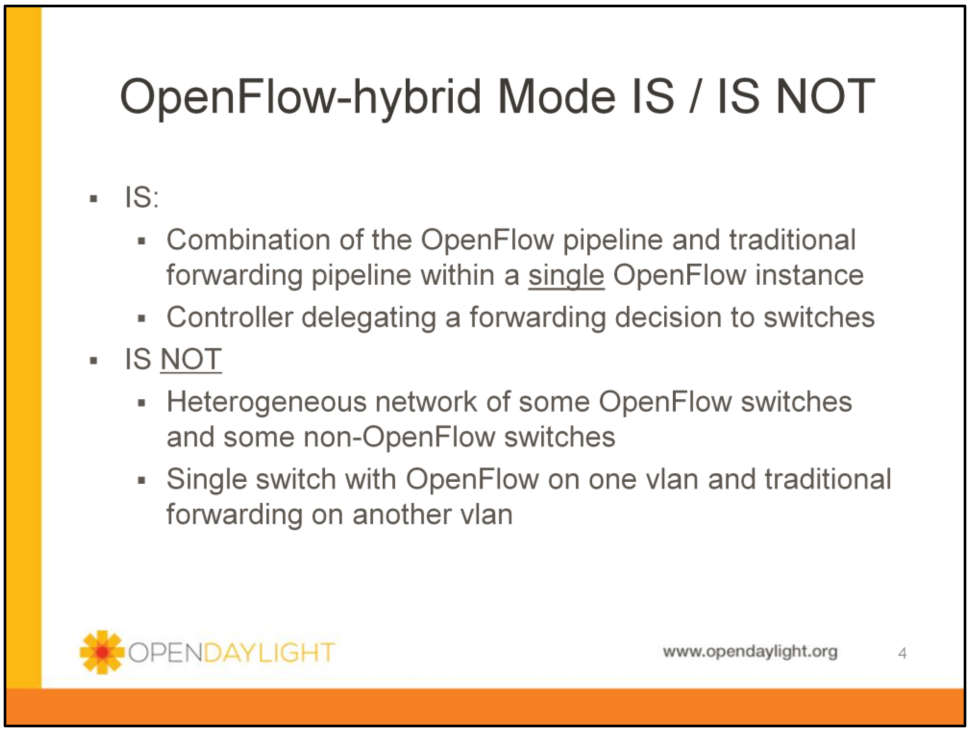


Outline

- What is OpenFlow-hybrid mode?
- OpenFlow specification
- Benefits of OpenFlow-hybrid mode
- Changes needed to support

OPENDAYLIGHT www.opendaylight.org 3

This presentation will define what we mean by “hybrid mode”, how that concept is supported by the OpenFlow specification, and some of the benefits of supporting the concept on an SDN controller. Finally, we’ll discuss some of the changes needed to implement this proposal.



OpenFlow-hybrid Mode IS / IS NOT

- IS:
  - Combination of the OpenFlow pipeline and traditional forwarding pipeline within a single OpenFlow instance
  - Controller delegating a forwarding decision to switches
- IS NOT
  - Heterogeneous network of some OpenFlow switches and some non-OpenFlow switches
  - Single switch with OpenFlow on one vlan and traditional forwarding on another vlan

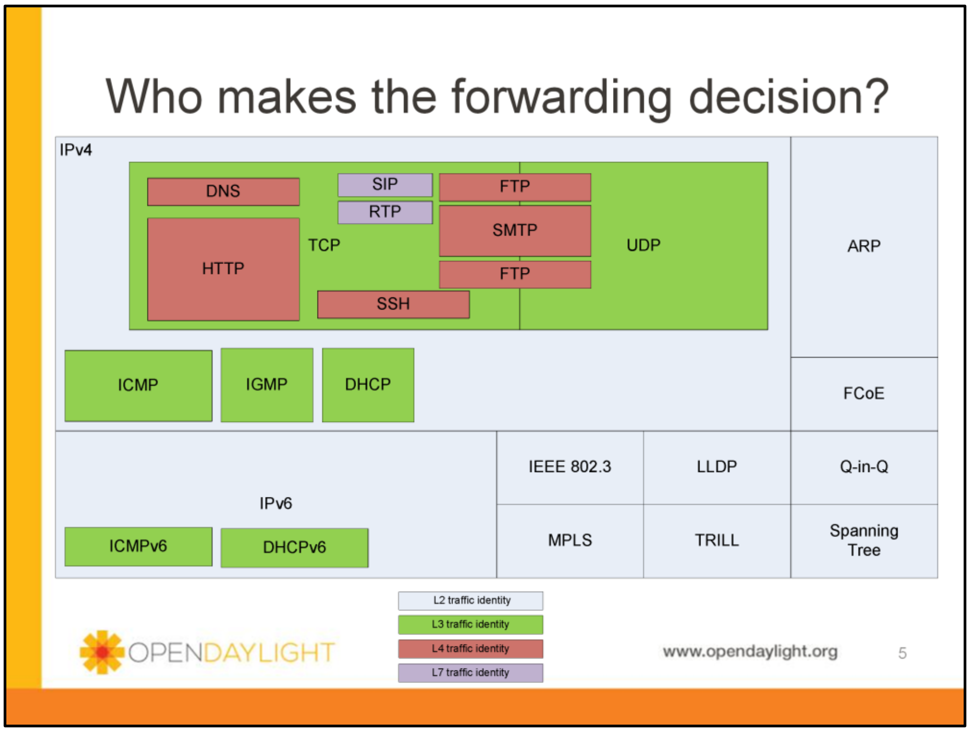
OPENDAYLIGHT www.opendaylight.org 4

The term "hybrid" has been used in multiple ways within the context of SDN, so first I'd like to clarify what we are proposing when we use the term "hybrid mode".

When we use the term "OpenFlow-hybrid mode" we are proposing that the controller use a combination of the OpenFlow pipeline and traditional forwarding pipeline within a single OpenFlow instance. What this reduces to is the controller delegating some forwarding decisions to controlled switches, rather than making those decisions directly itself.

When we use the term "OpenFlow-hybrid mode" we are not referring to a heterogeneous network which is made up of some OpenFlow switches and some traditional switches. We are also not referring to a single switch with OpenFlow on one vlan and traditional forwarding on another vlan.

Hopefully this clarifies what we're proposing.

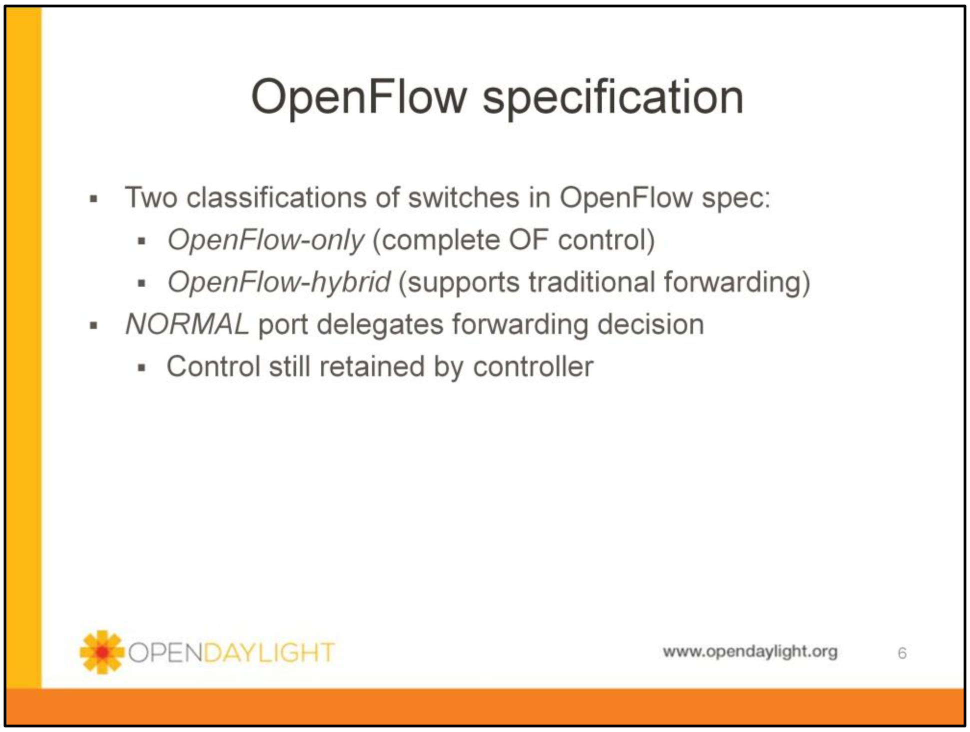


Who makes the forwarding decision?

The key question of hybrid mode is "Who makes the forwarding decision?". The OpenFlow-only approach dictates that the SDN controller make the forwarding decision for every packet on the network. This puts a tremendous amount of responsibility and complexity into the controller, because it needs to correctly forward all packet types that the network might be asked to forward.

With our "OpenFlow-hybrid mode" proposal, we are advocating that the controller only make the forwarding decision for certain packet classes or flows, depending upon the applications installed. The forwarding decision for the remaining packets on the network will be made by the traditional network's forwarding engine. This lets SDN focus on improving the network experience piece-by-piece, rather than completely replacing traditional forwarding.

For example, if someone wanted to write an application to inspect DNS packets, the controller would only need to make the forwarding decision for DNS packets, rather than all packets on the network. The application and controller wouldn't need to know anything about DHCPv6, ARP, or SSH.



OpenFlow specification

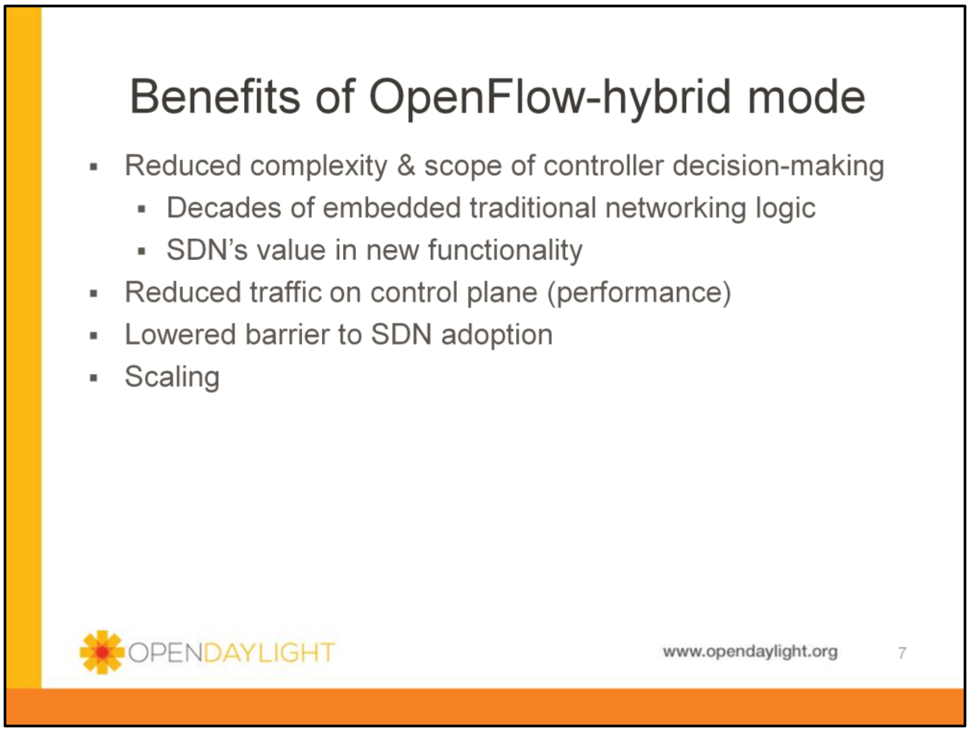
- Two classifications of switches in OpenFlow spec:
  - *OpenFlow-only* (complete OF control)
  - *OpenFlow-hybrid* (supports traditional forwarding)
- *NORMAL* port delegates forwarding decision
  - Control still retained by controller

OPENDAYLIGHT www.opendaylight.org 6

The OpenFlow specification has supported "OpenFlow-hybrid mode" since version 1.3 of the spec. The spec defines two classifications of switches. OpenFlow-only switches are those switches which completely rely upon the controller to make all forwarding decisions. OpenFlow-hybrid switches are those switches which support OpenFlow, but also support a traditional networking pipeline for protocols like spanning-tree, OSPF, etc.

The OpenFlow specification also supports "OpenFlow-hybrid mode" via the NORMAL port. The NORMAL port tells the switch to forward the packet according to the traditional networking pipeline. By instructing the switch to forward to the NORMAL port, the controller can delegate the forwarding decision to the switch on a per-flow or per-class basis, or any other subset of all packets on the network.

It is important to note that when the controller makes use of the NORMAL port, the controller is still retaining control over the network, because the controller is instructing the switches how to forward the packet. The controller retains control without itself making the forwarding decision.



Benefits of OpenFlow-hybrid mode

- Reduced complexity & scope of controller decision-making
  - Decades of embedded traditional networking logic
  - SDN's value in new functionality
- Reduced traffic on control plane (performance)
- Lowered barrier to SDN adoption
- Scaling

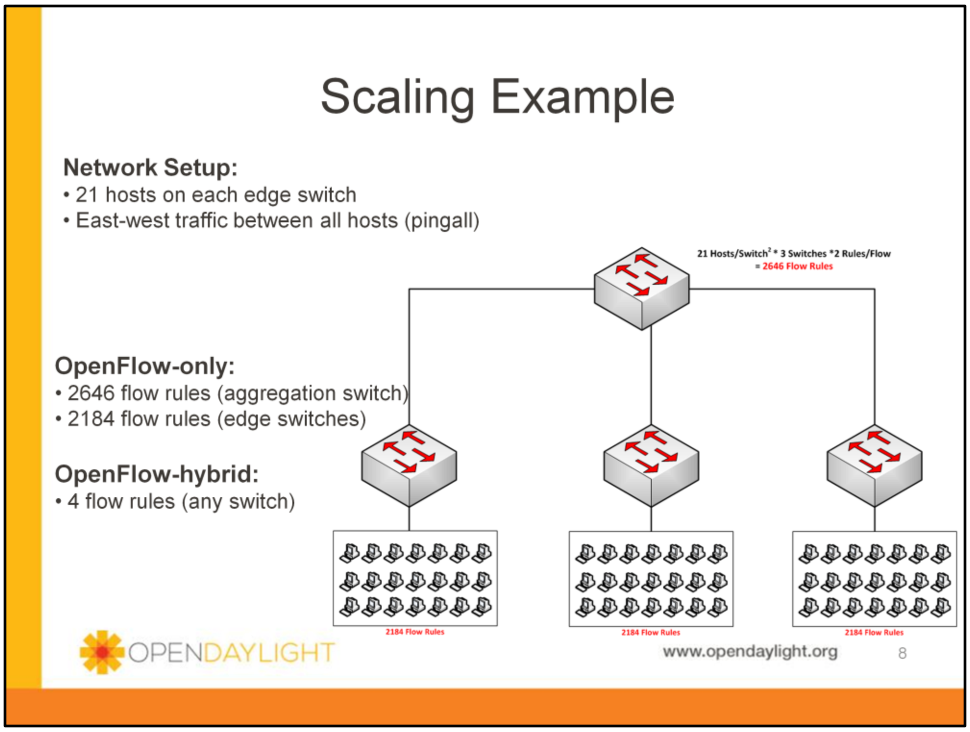
OPENDAYLIGHT www.opendaylight.org 7

There are several benefits to supporting hybrid mode in an SDN controller. First, hybrid mode reduces the complexity and scope of the forwarding decisions which the controller makes. Any OpenFlow-hybrid switches we control will have decades of embedded traditional forwarding logic in their ASICs and firmware. Rather than re-implementing that logic in the controller, we are proposing that the controller make use of the embedded traditional forwarding that's already available. SDN's value is in adding new functionality to networks. If all SDN does is centralize the same forwarding decision, then there is no compelling reason for administrators to move to SDN.

Another benefit of hybrid mode is the reduced amount of traffic on the control plane. Since the controller only needs to make the forwarding decision for specific packet types/flows, there are many fewer PACKET_IN messages being sent to the controller from the controlled switches. This yields better performance overall for the solution.

Hybrid mode also lowers the barrier to SDN adoption, because it allows administrators to adopt SDN piece-by-piece, as needed. This gradual migration to SDN is much easier to adopt than the notion of completely replacing all forwarding decisions in their traditional network with those made by the SDN controller.

Lastly, hybrid mode scales much more gracefully in terms of the number of OpenFlow rules it pushes to switches, as you will see on the next slide.



Scaling Example

Network Setup:
- 21 hosts on each edge switch
- East-west traffic between all hosts (pingall)

OpenFlow-only:
- 2646 flow rules (aggregation switch)
- 2184 flow rules (edge switches)

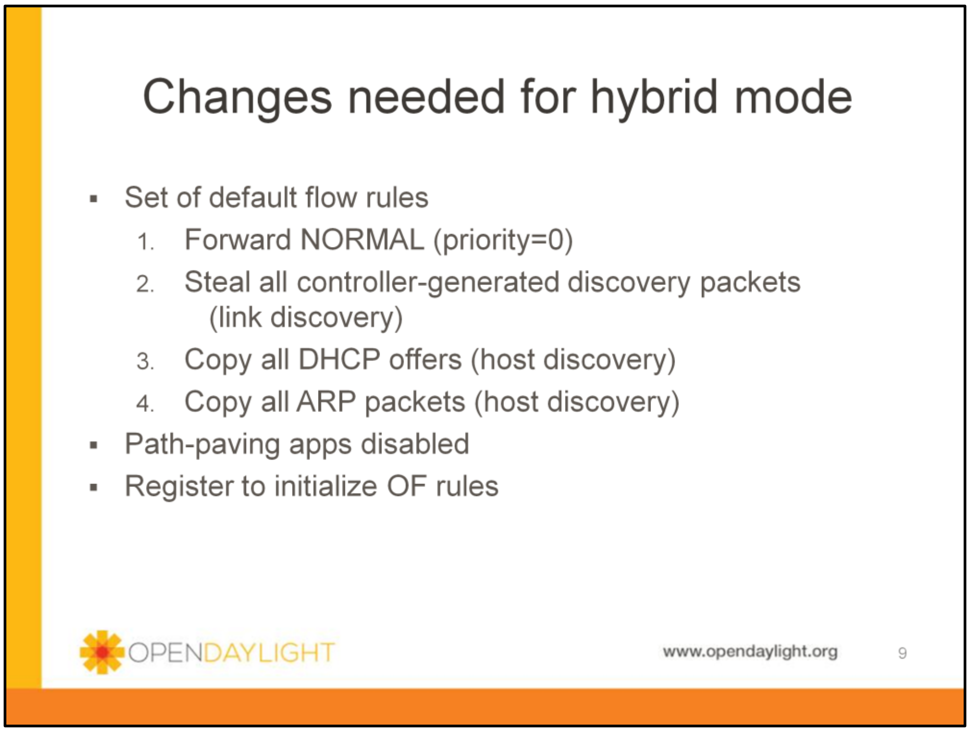
OpenFlow-hybrid:
- 4 flow rules (any switch)

21 Hosts/Switch^2 * 3 Switches *2 Rules/Flow
= 2646 Flow Rules
2184 Flow Rules
2184 Flow Rules
2184 Flow Rules

OPENDAYLIGHT www.opendaylight.org 8

This example shows a network with 1 aggregation switch and 3 edge switches. If each edge switch has 21 hosts connected, there are a total of 63 hosts connected to the network. If the equivalent of a "pingall" is executed where each host pings all of the other hosts, there would be 2K rules on each edge switch and over 2600 rules on the aggregation switch.

It is not scalable for 63 hosts to fill the tables of an aggregation switch with 2600+ flow rules. By comparison, this same setup with hybrid mode would use only 4 rules on each edge switch and aggregation switch. Those 4 rules will be described on the next slide. If the number of hosts increases to 1,000 on each edge switch, the switches will still each only have 4 rules.



Changes needed for hybrid mode

- Set of default flow rules
  1. Forward NORMAL (priority=0)
  2. Steal all controller-generated discovery packets (link discovery)
  3. Copy all DHCP offers (host discovery)
  4. Copy all ARP packets (host discovery)
- Path-paving apps disabled
- Register to initialize OF rules

OPENDAYLIGHT www.opendaylight.org 9

The changes needed in OpenDaylight to support hybrid mode mainly consist of adjusting the flow mods (OpenFlow forwarding rules) which we send to switches. We would need to adjust the rules that are sent initially after a switch connects, and in an ongoing basis in a running environment.

The controller would send a default rule which tells the switch to forward packets to the NORMAL port. This rule delegates the forwarding decision to the controlled switches, but it means that the controller would receive ZERO packet_in messages if no other rules were pushed. For this reason, we'd put this rule at priority 0 in the last hardware OF table of the pipeline. Without this rule, the default behavior for OF 1.0 is to steal to the controller and the default behavior for OF 1.3 is to drop all packets.

Link discovery would need to send a rule which steals all controller-generated link discovery packets to the controller. This would not include LLDP generated by the switches themselves, assuming we can distinguish controller-generated packets from switch-generated packets based upon ethertype (see the Safe Link Discovery presentation for details).

Host discovery would need to send a rule which copies (not steals) all ARP requests and replies. Host discovery would also need to send a rule which copies (not steals) all DHCP offers. Since these packets are copied, the controller would not re-inject these packets.

Any path-paving applications would need to be disabled, because the controller would not want to pave paths except where the controller's path paving was functionally desirable (from an end-host perspective) over traditional forwarding.

We also propose that we implement a mechanism in OpenDaylight whereby applications can register to supply initial OpenFlow rules when a switch connects. These initial rules would be used to get packets to the application so the application can make the forwarding decision.