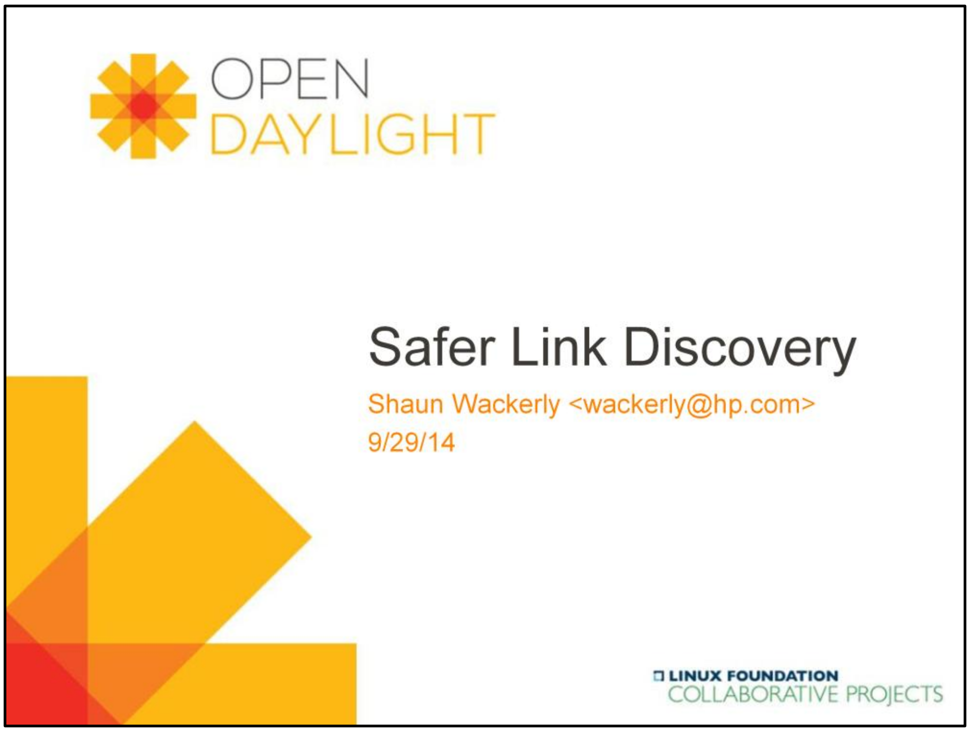


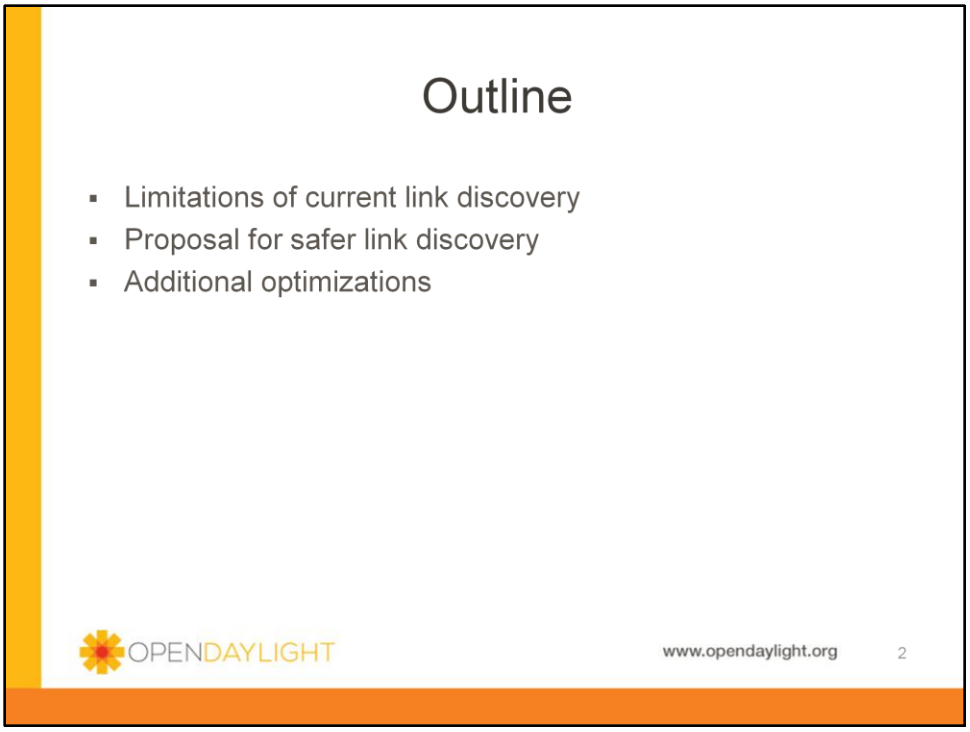
OPEN DAYLIGHT

Safer Link Discovery

Shaun Wackerly <wackerly@hp.com>

9/29/14

LINUX FOUNDATION COLLABORATIVE PROJECTS

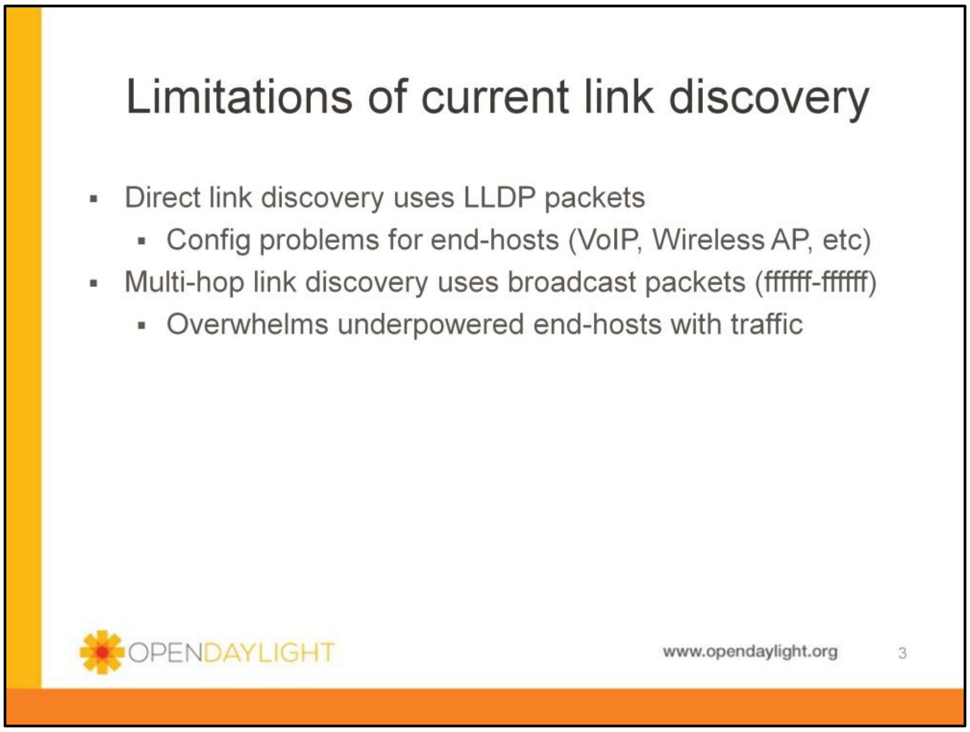


Outline

- Limitations of current link discovery
- Proposal for safer link discovery
- Additional optimizations

OPENDAYLIGHT www.opendaylight.org 2

This presentation will identify the limitations of the current link discovery methods and why it is harmful to end-hosts on the network. Then we'll describe our proposal for a safer link discovery which behaves more gracefully with respect to end-hosts. Lastly, we'll describe some optimizations which might help implement the safer link discovery even more efficiently.



Limitations of current link discovery

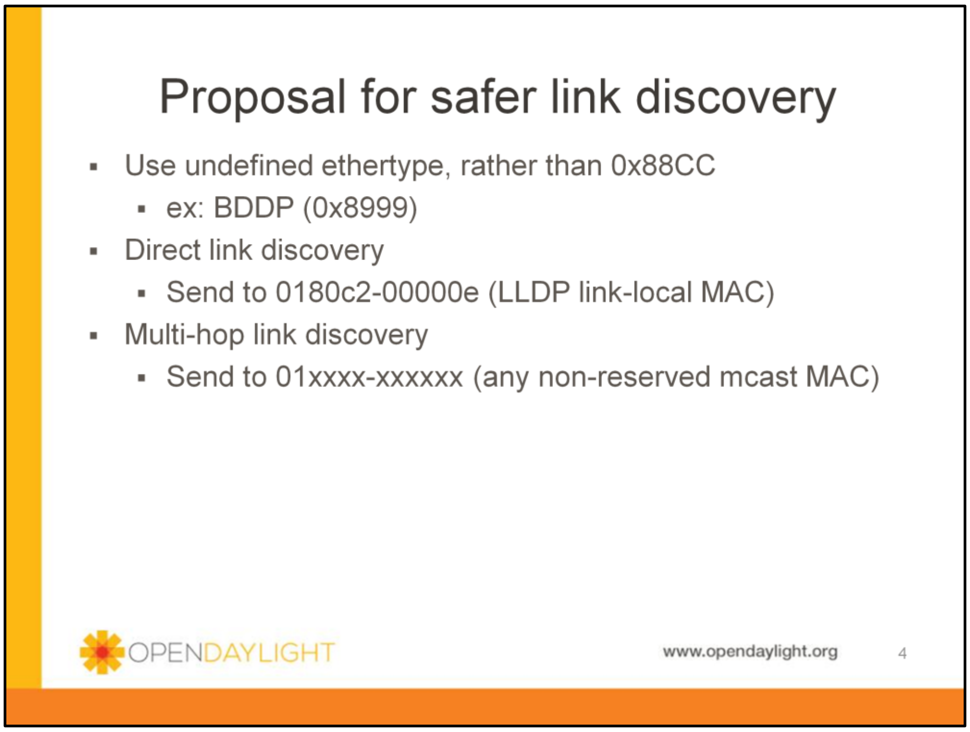
- Direct link discovery uses LLDP packets
  - Config problems for end-hosts (VoIP, Wireless AP, etc)
- Multi-hop link discovery uses broadcast packets (ffffff-ffffff)
  - Overwhelms underpowered end-hosts with traffic

OPENDAYLIGHT www.opendaylight.org 3

The limitations of the current link discovery center around behaving gracefully toward end-hosts.

Direct link discovery injects LLDP packets into the network, but those LLDP packets may cause problems for end-hosts which receive their configuration via LLDP. Some specific examples of such end-hosts are VoIP phones or wireless access points. If these devices receive one of the LLDP packets we've injected, it may clear their configuration.

Multi-hop link discovery injects broadcast packets into the network. Those broadcast packets are received by all hosts on the network. This can cause issues for underpowered end-hosts if we're injecting broadcast packets too often, because it will consume a larger percentage of that host's CPU.



Proposal for safer link discovery

- Use undefined ethertype, rather than 0x88CC
  - ex: BDDP (0x8999)
- Direct link discovery
  - Send to 0180c2-00000e (LLDP link-local MAC)
- Multi-hop link discovery
  - Send to 01xxxx-xxxxxx (any non-reserved mcast MAC)

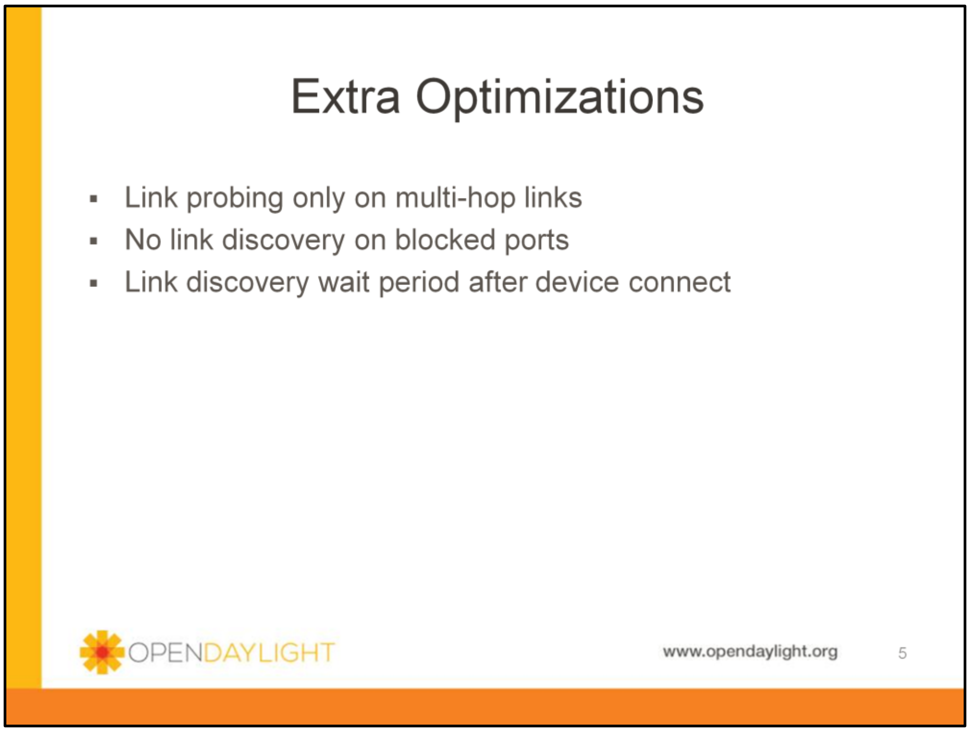
OPENDAYLIGHT www.opendaylight.org 4

Our proposal for safer link discovery involves changes to the packets injected by direct link discovery and multi-hop link discovery. We are proposing that all link discovery packets use an undefined ethertype (for example, the value 0x8999 which is used by BDDP). Using an undefined ethertype lets us send to any destination MAC address without the packet being mis-interpreted as a protocol packet sent by the traditional network.

We propose that direct link discovery use the LLDP link-local MAC (as is done today) but with an undefined ethertype**. This prevents end-hosts from processing the injected link discovery packets as LLDP packets, but also restricts the packets from being forwarded across multiple hops.

We also propose that multi-hop link discovery packets are sent to a non-reserved multicast MAC (of the form 01xxxx-xxxxxx)**. This will prevent end-hosts from processing these packets, but will allow them to be forwarded across multiple hops.

** NOTE: These methods have been filed by Hewlett-Packard with the US Patent & Trademark Office, but Hewlett-Packard has granted license to these patents under EPL 1.0.



Extra Optimizations

- Link probing only on multi-hop links
- No link discovery on blocked ports
- Link discovery wait period after device connect

www.opendaylight.org 5

Some extra optimizations to link discovery that could accompany this proposal would be to only probe multi-hop links**. Direct link state can be derived from the OpenFlow port status change messages (ie: when the port goes down). By only probing multi-hop links, we reduce the number of packets the controller is injecting into the controlled network, thus reducing the number of packets the controller receives from the controlled network.

Another optimization is to not perform link discovery on blocked ports**. This would only apply to a hybrid environment where a traditional protocol like spanning-tree is running on ports which the controller controls. We would not want to attempt link discovery over ports which wouldn't forward packets, because that would imply we could use those links for forwarding path calculations.

The final optimization we're suggesting is a wait period after switch connect before attempting to initiate link discovery on that device**. When a controller reboots or when a section of the network has a hiccup, switches will tend to join the controller in bunches. By waiting a short period after the switch connects, we save work in the controller by avoiding re-discovery on all switches after each switch connects.

** NOTE: These methods have been filed by Hewlett-Packard with the US Patent & Trademark Office, but Hewlett-Packard has granted license to these patents under EPL 1.0.