"Shortest path forwarding using OpenFlow"

Iwan Hoogendoorn & Joris Soeurt
Supervisor: Ronald van der Pol
Wednesday 8 February 2012
Research Question

In what way can shortest path bridging be implemented using OpenFlow?
Agenda

Approach

Spanning Tree

TRILL / 802.1aq (SPB) differences

TRILL - specific

802.1aq (SPB) - specific

OpenFlow

Routing module

Iwan Hoogendoorn
Part-Time UvA student
Full-Time Cisco Employee

Joris Soeurt
Full-Time UvA student
Approach

• Implement a (basic) shortest path bridging algorithm using OpenFlow
  – First see how TRILL / 802.1aq implement this
• Discovered existing routing module
  – No annotation in code (written in C)
  – Very little, outdated documentation
• Explored and tested this module
• Created improvement proposal
Spanning Tree – Why – What does it do?

“The Spanning Tree Protocol (STP) is a network protocol that ensures a loop-free topology for any bridged Ethernet local area network.”
Spanning Tree
TRILL/802.1aq – Why better?

1. TRILL → IETF & 802.1aq → IEEE
2. IS-IS topology discovery
3. TRILL – unicast traffic – hop by hop
4. TRILL – broadcast, multicast, unknown – pre-calculated distribution trees
5. 802.1aq (SPB) – all traffic – trees calculated up front
TRILL - concept

- RBridges + additional TRILL headers
- IS-IS topology discovery
- 4 MAC learning techniques
- Unicast vs. multicast, broadcast, unknowns
TRILL - diagram
802.1aq (SPB) - Types

SPB

SPB-V
(Q-in-Q)

Access Network
- Reliability
- Bandwidth efficiency
- Unknown or managed addresses

MAC learning in data plane

SPB-M
(MAC-in-MAC)

Metro Core Network
- Reliability
- Auto-discovery
- Load sharing
- Managed addresses

MAC learning in control plane

Enterprise Network
- Plug & Play
- Easy to operate
- Unknown addresses
Each bridge is the “root” of a separate shortest path tree instance

Bridge G is the root of the green tree

Bridge E is the root of the blue tree

Both trees are active AND symmetric at all times
Introduction to OpenFlow

• Vendors generally don’t like to make their firmware open to outsiders.
• No possibility to test new algorithms
• Stanford computer scientist Nick McKeown and colleagues developed a standard called OpenFlow
Introduction to OpenFlow

• Control and dataplane completely separated

• Control plane: Controller
  – NOX
    • Open source
    • Python programmable

• Data plane: OpenFlow aware switches

• Communication via standardized API.
What should I do?

Create rule, action = out port 2

Flow table on switch:

<table>
<thead>
<tr>
<th>Src MAC</th>
<th>Dst MAC</th>
<th>Src IP</th>
<th>Dst IP</th>
<th>In port</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>00-00-01</td>
<td>00-00-02</td>
<td></td>
<td></td>
<td>1</td>
<td>Out 2</td>
</tr>
</tbody>
</table>
Key features/advantages

• You are not limited by the functionality of the proprietary firmware of vendors.
• Computing power of a server
• Strength of a programming language
• You can implement any forwarding algorithm you want.
Testing...

• 6 OpenWrt switches with custom firmware

• Data gathered
  – Wireshark on controller
  – Tcpdump on hosts
  – NOX console output

• Information deduced
  – Communication
  – Innerworking
  – Link failover
  – Path determination
Parts of “routing” module

• Discovery module
  – Controller learns topology

• Spanning tree module
  – For broadcast/multicast/unknown unicast frames
  – Frames are flooded

• Shortest path module
  – For unicast frames
  – Frames are “routed”
Operation of discovery (1)

LLDP
Sent on: SW5, port 1

LLDP
Sent on: SW5, port 1
Received on: SW6, port 3
Broadcast / multicast / unknown

- Rule with action = flood
- Frames are flooded out all ports, except for:
  - Originating port
  - Port with flood flag set to disabled
Operation of flooding (first frame)

Create rule, action = Flood

Src: 00-00-01
Dst: FF-FF-FF

Src: 00-00-01
Dst: FF-FF-FF

Src: 00-00-01
Dst: FF-FF-FF

Src: 00-00-01
Dst: FF-FF-FF

H1
MAC: 00-00-01

H2
MAC: 00-00-02

H3
MAC: 00-00-03

H4
MAC: 00-00-04

What should I do?
Create rule, action = Flood

What should I do?
Create rule, action = Flood

What should I do?
Create rule, action = Flood

What should I do?
Create rule, action = Flood
Operation of flooding (next frames)
Unicast

• Controller knows location of hosts (registered by authenticator module)
• Can program complete path in advance
Operation of unicast (first frame)

Src: 00-00-01
Dst: 00-00-04

What should I do?

Create rule, action = port 1

Src: 00-00-01
Dst: 00-00-02

What should I do?

H1
MAC: 00-00-01

H2
MAC: 00-00-02

H3
MAC: 00-00-03

H4
MAC: 00-00-04
Operation of unicast (next frames)
Shortcomings & improvement

• “Routing” module
  – Loadbalancing algorithm not optimal
  – Link failover not optimal
  – Instable (loops)

• Improvement proposal
  – Pseudocode
Comparison

• Centralized instead of distributed
  – No IS-IS needed
  – But, shortest path algorithm needed
• No standard (like IETF / IEEE)
• Spanning tree programmed to port property
Conclusion

• Powerful but not yet powerful enough?
  – Still in development (1.2 in March, 1.3 in April)
  – Version 1.2:
    • failover group
    • master/slave controller

• OpenFlow is generic, 802.1aq / TRILL are specific
Questions?