• Forwarding pipeline design to allow
  o E2E proactive protection independent from controller reachability
  o Fully programmable failure detection and recovery in the fast-path

• Programmable failure detection
  o BFD-like

• Fast reroute
  o Inspired by MPLS
  o for both local and non-local preplanned failures

• Based on stateful data plane
  o OpenState
  o OVS + learn action
  o P4$_{14}$
Failure detection

Assumption:
As long as packets are received from a given port, that port can be also used to transmit packets

- If no packet is received from port x within a $\delta_1$ interval:
  - Next data packet towards port x is tagged with a special value (Heartbeat request)
  - Port x is declared down if adjacent node does not send back a copy (Heartbeat reply) within a $\delta_2$ interval

- Configurable trade off: overhead vs failover responsiveness
  - $\delta_1$: Heartbeat requests generation timeout
  - $\delta_2$: Heartbeat reply timeout before the port is declared down

- Guaranteed max detection delay:
  - $\delta_1 + \delta_2$
Fast Reroute

- MPLS label used to distinguish between different forwarding:
  - No tag → forward packet on the primary path
  - tag=Fi → forward packet on the detour for the i-th failure

- Zero losses after failure detection
- No controller intervention for all pre-planned failures
- What if no local alternative path is available?
Fast Reroute (2)

- Packets are tagged and bounced back up to a proper redirect point
- Tagged packets trigger a state transition:
  - updating the routing of the involved connections
- Still zero losses after failure detection!
- Tagged data packets as signalling
- No controller intervention!

Lookup-scope = [ETH_SRC, ETH_DST]
Update-scope = [ETH_SRC, ETH_DST]
• How to restore the forwarding on the primary path?
• Programmable periodic probing for primary path availability
• Failover activation/deactivation can be postponed
  • In order to minimize out-of-sequence, packets are kept on the primary path up to expiration of a burst of packets
  • Programmable idle timeout/hard timeout
Putting all together: Fast reroute FSM
Redirect FSM

FSM: NO backup path is locally available

Lookup-scope = [ETH_SRC, ETH_DST]
Update-scope = [ETH_SRC, ETH_DST]

Failure Detect FSM

Lookup-scope = [METADATA]
Update-scope = [METADATA]
Redirect & Failure Detect FSM

FSM (2): backup path is locally available

Lookup-scope = [METADATA]
Update-scope = [METADATA]
Software implementation

• OpenState
  • Ryu* controller
  • CPqD OpenFlow 1.3 softswitch*
  • https://github.com/OpenState-SDN/spider

• P4$_{14}$ based on openstate.p4 library
  • https://github.com/OpenState-SDN/openstate.p4

*modified with OpenState support http://openstate-sdn.org
• Testing ping from h1 to h2
  o h1→h2 forwarded on path S1-S2-S3
  o h2→h1 forwarded on path S3-S4-S1

• Failure of link S2-S3
  o Backup path S1-S4-S3
  o Checking Heartbeat mechanism for failure detection
  o Checking Probe mechanism for primary path availability

<table>
<thead>
<tr>
<th>VLAN tag</th>
<th>forwarding</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>Normal</td>
</tr>
<tr>
<td>17</td>
<td>Failure</td>
</tr>
<tr>
<td>20</td>
<td>HB_request</td>
</tr>
<tr>
<td>21</td>
<td>HB_reply</td>
</tr>
<tr>
<td>22</td>
<td>Probe</td>
</tr>
</tbody>
</table>
OpenState pipeline

Legend:
- \( \rightarrow \) Packets
- \( \Rightarrow \) State updates

Table 0 \rightarrow Table 1 \rightarrow Table 2 RF FSM \rightarrow Table 3 LF FSM \rightarrow Output port(s)
P4 pipeline

Redirect FSM
- Forwarding on primary/tagging on backup path according to flow state
- State transitions based on tagged packets

Failure Detect FSM
- State update for input port
- State lookup for output port
- Forwarding/tagging/bouncing back according to port state
- State transitions based on tagged packets and timeout expiration

Clone table egress

Registers
- Lookup flow state
- Forward
- Update flow state

Write inport to metadata
Update inport state
Write outport to metadata
Lookup outport state
Forward
Update outport state

Clone table egress