Ground Control to Major Faults:
Towards Fault Tolerant and Adaptive SDN Control Network

Liron Schiff (Tel Aviv University)
Stefan Schmid (TU Berlin, Germany & Aalborg University, Denmark)
Marco Canini (Université catholique de Louvain)
Software Defined Network (SDN)

Logically centralized control

Control plane network

Fast data plane
• Main function:
  – Connect the controller with each switch
Main function:
  - Connect the controller with each switch

Can be distributed
  - Handle failures
  - Load balancing
  - Need synchronization
SDN control-plane

• Main functions:
  – Connect the controller with each switch
  – Inter-connect the controllers

• Can be distributed
  – Handle failures
  – Load balancing
  – Need synchronization
• Main function:
  – Connect the controller with each switch
  – Inter-connect the controllers

• Can be distributed
  – Handle failures
  – Load balancing
  – Need synchronization

• Can be in-band
  – Cheaper
  – More provisioned (redundancy)
  – More flexible (TE, unicast, etc.)
Switch Structure (Model)

- Control traffic is sent in-band.
- The switch identifies and forward it to the control module.
- Supported by OpenFlow.
Challenge: Boot Up

- Switches start as unmanaged.
- Switches should be configured to forward control in-band.
Challenge: Boot Up

- Switches start as unmanaged.
- Switches should be configured to forward control in-band.
Challenge: Plug&Play

• Support new links / switches / controllers

• Switches can’t be configured with all possible controllers.
Challenge: Plug & Play

- Support new links / switches / controllers

- Switches can’t be configured with all possible controllers.
Challenge: Handle Failures
Goal: Network should return to a good state.
“Good network state” :=

• Every switch is connected to a controller.
• Controllers can communicate and make joint decisions.
Our Contributions

A Plug & Play Distributed SDN Control Plane

• Flexible controller membership (additions, removals, failures)
• Automatic switch discovery & topology awareness
• Supports ONIX, ElastiCon, Beehive, STN, and more.

Self Adjusting

• Converges to “good state” from unmanaged states.
• Tolerates failures and delays: low re-convergence times
The Medieval Scheme

• Controllers aim to continuously grow their management regions...
• … and “conquer” unmanaged switches.
Switch States

1. Broadcast
2. Any controller can respond

Session established
No keep-alive timeout

1. Controller traffic is passed through
2. Other controllers are blocked
Switch State Configurations

A priori configured

<table>
<thead>
<tr>
<th>Rules</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Managed</td>
<td>Priority 2, with timeout</td>
</tr>
<tr>
<td>Unmanaged</td>
<td>Priority 1, no timeout</td>
</tr>
</tbody>
</table>

Maintained by controller
Controller uses a managed switch, R, to detect and establish connection to a new switch S.
The Medieval Scheme

- Controllers aim to continuously grow their management regions...
- ... and “conquer” unmanaged switches.
The Medieval Scheme

• Controllers aim to continuously grow their management regions...

• … and “conquer” unmanaged switches.

• Management with two spanning tree types:
  (1) Per-region spanning tree
    (bidirectional, owned by controller)
Controllers “conquer” switches adjacent to their regions of control and build a spanning tree for controller-to-switch connectivity.
The Medieval Scheme

• Controllers aim to continuously grow their management regions...
• … and “conquer” unmanaged switches.
• Management with two spanning tree types:
  (1) Per-region spanning tree
      (bidirectional, owned by controller)
  (2) Network-wide spanning tree
      (to connect controllers)
Per-controller global spanning trees provide controller-to-controller connectivity.
Prototype Implementation

- Emulator in Java
- OpenFlow switches and controllers: light-weight threads
- Links modelled by message queues
- Fat-tree topology \((k=4)\), 1-8 controllers
- Measured time to manage switches

<table>
<thead>
<tr>
<th># ctrls</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time(ms)</td>
<td>9382</td>
<td>6983</td>
<td>6150</td>
<td>4224</td>
<td>6035</td>
<td>5104</td>
<td>3704</td>
<td>3680</td>
</tr>
</tbody>
</table>
Prototype Implementation
Conclusions

• Medieval: a robust distributed SDN control plane.

• Fully supported by OpenFlow.

• Convergence can be proved and easily tested.

• Extended analysis and simulation are coming soon.
fin