Distribution, fault-tolerance & consistency in SDN control: Pick three.

André Mantas
Context
Context
Context
Context
Context
Introduction to SDN
OpenFlow switch

1. Forward packet to port(s)
2. Encapsulate and forward to controller
3. Drop packet
4. Send to normal processing pipeline

<table>
<thead>
<tr>
<th>Switch Port</th>
<th>MAC src</th>
<th>MAC dst</th>
<th>Eth type</th>
<th>VLAN ID</th>
<th>IP Src</th>
<th>IP Dst</th>
<th>IP Prot</th>
<th>TCP sport</th>
<th>TCP dport</th>
</tr>
</thead>
</table>

Packet + byte counters
Motivation: Exactly once semantics
Related Work: Ravana
Related Work: Ravana
Summing up...

Exactly once events
- Buffering and retransmission
- IDs and filtering in the log

Exactly once commands
- Buffering and filtering by switches
- ACK from switches

![Events and Commands Diagram](image_url)
Proposal

- Distributed architecture: scale with the number of switches
- Fault-tolerant: correct and consistent operation of controllers and switches
- Transparent for applications
- No changes to OpenFlow or switches
OpenFlow 1.4 Bundles

- Available flags: atomic, ordered
- Number of messages can be optimized
Protocol for Fault-Tolerance
Exactly once events with faults
Exactly once commands with faults
Comparison with Ravana

- Doesn’t require changes to switches or OpenFlow
- Distributed architecture scales better

- More messages in the network
- More synchronization between controllers
Plans

- Implementation: Floodlight + BFT-SMaRt
- Evaluation: Cbench
  - Normal case + impact of faults
  - Compare with Ravana
  - Compare with Floodlight
Questions?