SCL: Simple Coordination Layer
Aurojit Panda, Wenting Zheng, Xiaohe Hu, Arvind Krishnamurthy, Scott Shenker
NYU, UC Berkeley, Tsinghua University, University of Washington, ICSI

Software Defined Networks

- Forwarding implemented by switches.
• Rules computed by controllers.
• Rules depend on policy and network state.
• Policy: What paths are acceptable?
• Network State: Current state of links and switches

How to build controllers?
Single Image Controllers

- Controller runs on a single server.
- Examples: Nox, Pox, Ryu, etc.

Single Image Controllers: Assumptions
- The controller observes a sequence of events.
- **Network state** computed using event sequence.
- Applications react to sequence of events.
- Events and updates sent over TCP channels.
- Events from **different switches** can be **reordered**.
- Updates to different switches can be **reordered**.

How to handle controller failures, scale controllers, etc.?

**Move to distributed controllers.**
How to build distributed controllers?
Why is this Harder?

- Event **ordering** can differ across controllers.
- Rules must converge despite this reordering.
- Two ways to handle this
  - Algorithms are correct despite reordering.
  - Mechanisms so controllers agree on ordering.
- Rely on **ordering mechanisms** for generality.
- How to implement event ordering?

Canonical Solution: Consensus

- Consensus: Protocol to get agreement on a value.
- Rely on **consensus** to agree on event order.
Applications always see events in agreed order.

- Can use same algorithms as single image controller.
- Controllers are Replicated State Machines.
- Adopted by Onix, ONOS, etc.
- How to implement consensus?

Canonical Consensus Mechanism

- Mechanism appoints a leader.
  - Leader receives all network events - decides on order.
  - Leader replicates ordered events at other controllers.
  - Must wait for a quorum of controllers to confirm replication.
  - Once quorum has confirmed delivers events to application.
Several algorithms in use - **ZAB**, **Raft**, **Paxos** variants (e.g., MultiPaxos)

**Canonical Consensus Mechanism**

- If **leader fails** protocol appoints new leader.
- Protocol must ensure leader is one with newest data.
  - Quorum replication ensures order cannot be forgotten.
  - Controller can reconstruct state by replaying events.

- Several algorithms in use - **ZAB**, **Raft**, **Paxos** variants (e.g., MultiPaxos)

**Canonical Consensus Mechanism: Limitations**
• Fault Tolerance: at least one partition fails during network partitions.
• Scalability: Worse performance worsens with more controllers.
• Control Plane Requirements: Performance is sensitive to losses, latency, etc.

Is consensus required?
Consensus Assumption

- Network state (topology and forwarding table) resides in controllers.
- RSMs ensure network state is not lost when controllers fail.
- Similar to distributed key value stores.
Consensus Assumption is Wrong

But we can query the network to discover current network state.

Safe to lose network state!

Distributed Controllers: An Alternative

2. Compute Updates
   Policy Controller
   
2. Compute Updates
   Policy Controller
   - Assume all controllers agree on policy.
<table>
<thead>
<tr>
<th>Challenges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming model: how to write control applications?</td>
</tr>
<tr>
<td>Programming model: how to support existing event-based algorithms?</td>
</tr>
<tr>
<td>Efficiency: how to minimize control traffic?</td>
</tr>
<tr>
<td>Safety: how to ensure critical policies are never violated?</td>
</tr>
<tr>
<td>Safety: how to safely update network policies?</td>
</tr>
</tbody>
</table>
Challenges

- Programming model: how to write control applications?
- Programming model: how to support existing event based algorithms?
- Efficiency: how to minimize control traffic?
- Safety: how to ensure some critical policies are never violated?
- Safety: how to safely update network policies?
- Policies: what classes of policies can be implemented using this mechanism?

SCL: Programming Model and Architecture
SCL Controller Requirements

- **Deterministic**: Controllers compute the same rule for given network state.
- **Idempotent**: The process of computing and updating rules is idempotent.
• **Proactive Applications:** Compute rules based on network state not packet-ins.

• **Triggered Updates:** Can trigger rule recomputation based on event log.

---

**SCL Proxies and Controllers**

- **Proxies** maintain a log of all prior network events.
- All switch events are sent to all **proxies**.
- **Proxy** triggers **controller** computation.
- Computation based on current log.
- **Controller** sends updates to **proxy**.
- **Proxy** maintains state about installed rules.
- Deduplicates updates before applying them.
SCL Proxies and Controllers: Challenges

- **Agreement:** Proxies must eventually agree on order.
- **Agreement:** Must eventually agree on the set of events.
- **Awareness:** Controllers and network state agrees eventually.

Addressing SCL Challenges

- Address these with two mechanisms.
- **Gossip** between controllers
Gossip between controllers

- Eventual agreement on observed events.
- Also assures agreement on ordering.
- Periodically query network for state.
- Awareness of network state.

Why abandon consensus?
Conceptually Unnecessary

- **RSM assumption**: Truth about network lies in the controller.
- **Reality**: Truth about the network lies within the network (dataplane).
  - Packets are processed by dataplane not by controllers.

Improves Performance and Resilience
<table>
<thead>
<tr>
<th></th>
<th>Responsiveness</th>
<th>Scalability</th>
<th>Fault Tolerance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consensus</td>
<td>At least 1 RTT between controllers</td>
<td>Latency increases with participants</td>
<td>Quorum must be available for progress</td>
</tr>
<tr>
<td>SCL</td>
<td>Respond immediately</td>
<td>Does not increase with # of participants</td>
<td>Functional as long as a controller is available</td>
</tr>
</tbody>
</table>

What about Route Convergence?
When Does Everyone Agree?

Convergence time in AS1221

Convergence time in Fat Tree
In the Paper

- Proof that gossip and periodic update are sufficient to guarantee convergence.
- Broadcast based in-band control channels.
- Mechanisms for policy update.
- Interaction with other types of policies.
- Other performance results.

Related Work

<table>
<thead>
<tr>
<th>Control Plane Consistency</th>
<th>Data Plane Consistency (Orthogonal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serializability</td>
<td>Labels: Reitblatt et al. (SIGCOMM ’12)</td>
</tr>
<tr>
<td>OpenFlow, ONIX (Cordys), SDxCO</td>
<td></td>
</tr>
<tr>
<td>Consensus: ONIX (OSDI'10), ONOS</td>
<td></td>
</tr>
<tr>
<td>----------------------------------</td>
<td></td>
</tr>
<tr>
<td>Atomic registers: Schiff et al (CCR'16)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stronger Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exactly-Once: Ravana (SOSR'15)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ordered Updates:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mahajan et al. (HotNets '13)</td>
</tr>
<tr>
<td>McClurg et al. (PLDI '15)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Synchronized Clocks:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mirzahi et al. (SOSR '15)</td>
</tr>
</tbody>
</table>

Back to the Future
Network Control as State Synchronization

- Model defined here: one update per network event.
- Might sound crazy, but this is what lots of SDN controllers do today.
- Why? RSM -- each decision must feed into next decision.
- Problem: What happens in periods when network is rapidly changing.
- Generate responses to old events, care about controller response rate.
• SCL model: Agree on log, compute updates based on log.
  • Enables coalescing events to generate a single log.
  • Take this idea further: switches do not send individual events.
  • Periodically synchronize state with controllers.
  • Synchronization period determines control channel requirements.

Network Control as State Synchronization

• Consequences:
  • Correctness due to the SCL argument.
  • When network is stable: minimal difference from current networks.
• When network is unstable: do not overwhelm control channels.

• One application explored in a recent submission.

Conclusion

• **Conventional wisdom**: Distributed SDN controllers need consensus.

• **This talk**: no consensus required.
  
  • Can use existing single image controllers with SCL.

• **Implication**
  
  • Simplifies controllers.

  • Improves convergence time, responsiveness, robustness.