Ravana: Controller Fault-Tolerance in SDN

Software Defined Networking: The Data Centre Perspective Seminar

Michel Kaporin (Mišels Kaporins)

Agenda

- Introduction
- Controller Failures in SDN
- Ravana Protocol
- Correctness
- Performance Optimisations
- Implementation
- Performance Evaluation

Introduction

Single Controller Lacks Reliability

- Single controller can become a single point of failure

- Failures lead to
  - Service disruptions
  - Incorrect packet processing

- Ideal model:
  - Fault-free SDN

Potential Solution

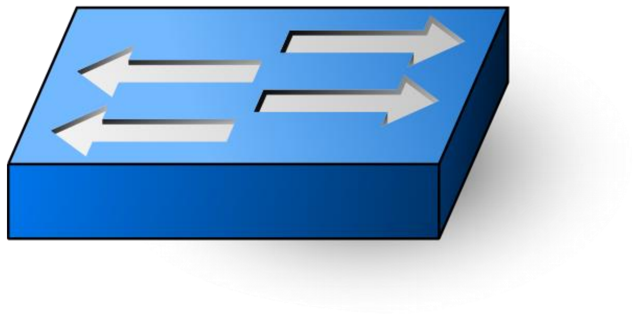
- Apply established distributed systems techniques:
  - Replicate durable state:
    - Two-phase commit or
    - Primary/backup methods with journaling and rollback
  - Or model controller as a *replicated state machine* (RSM)

More to a Solution

- Must ensure that switch state is handled consistently during failures

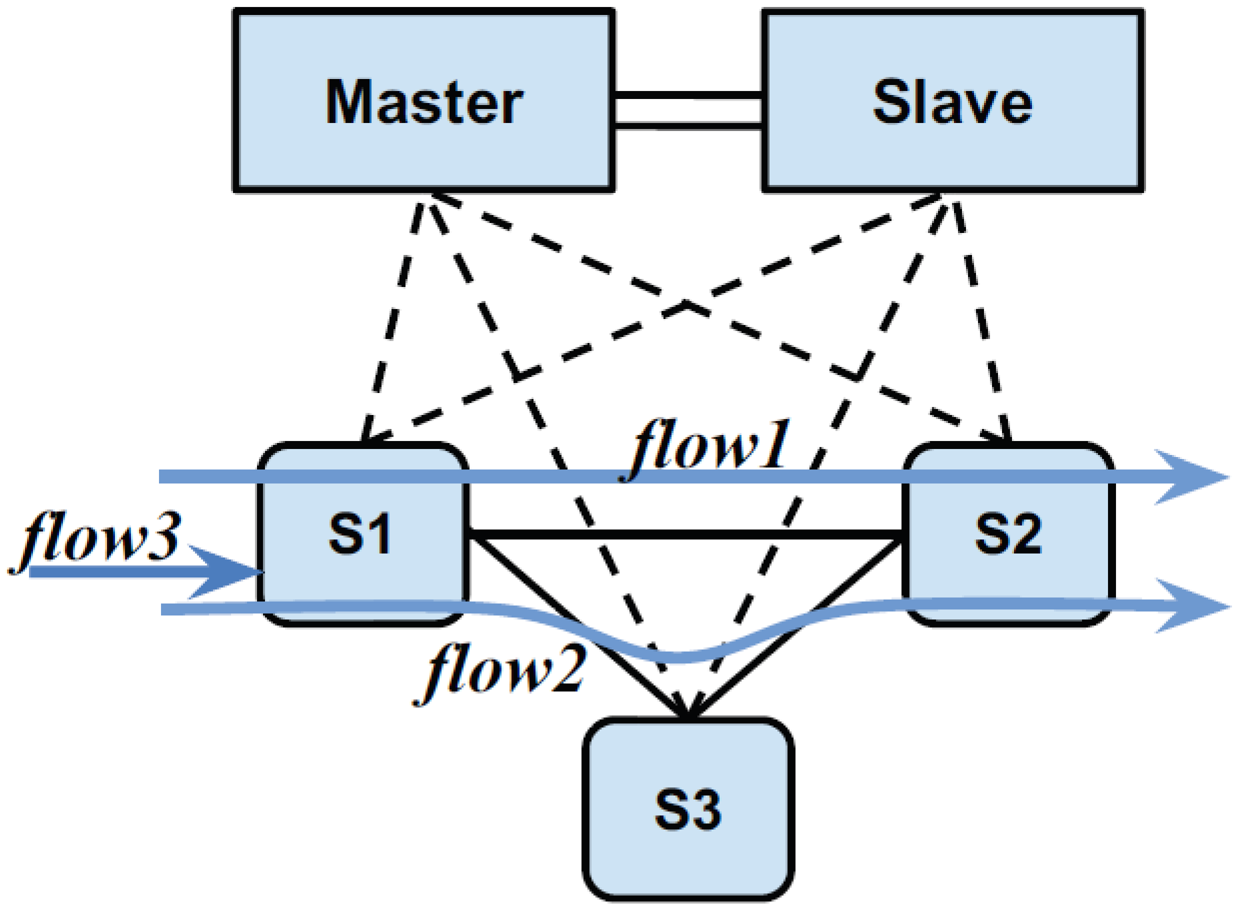
- **Not easy!**
- Switch semantics are different:
  - How to process events and execute commands under failures?

  - How to reason about switch state?
  - Rollback packets?



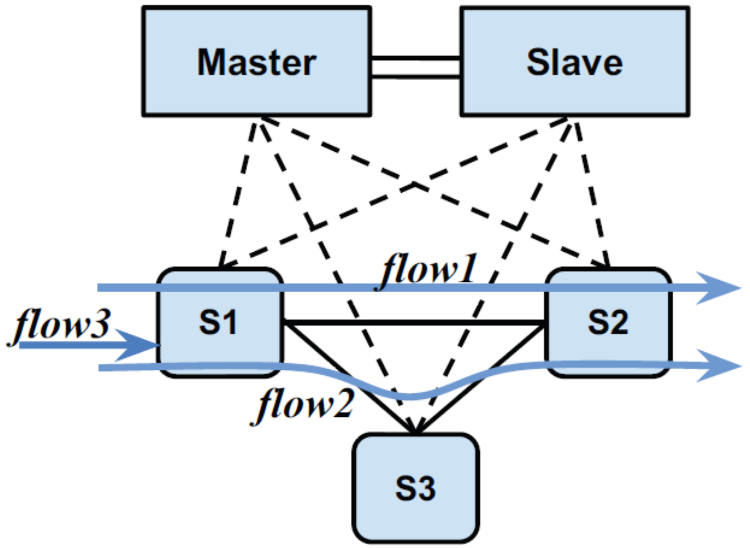
Controller Failures in SDN

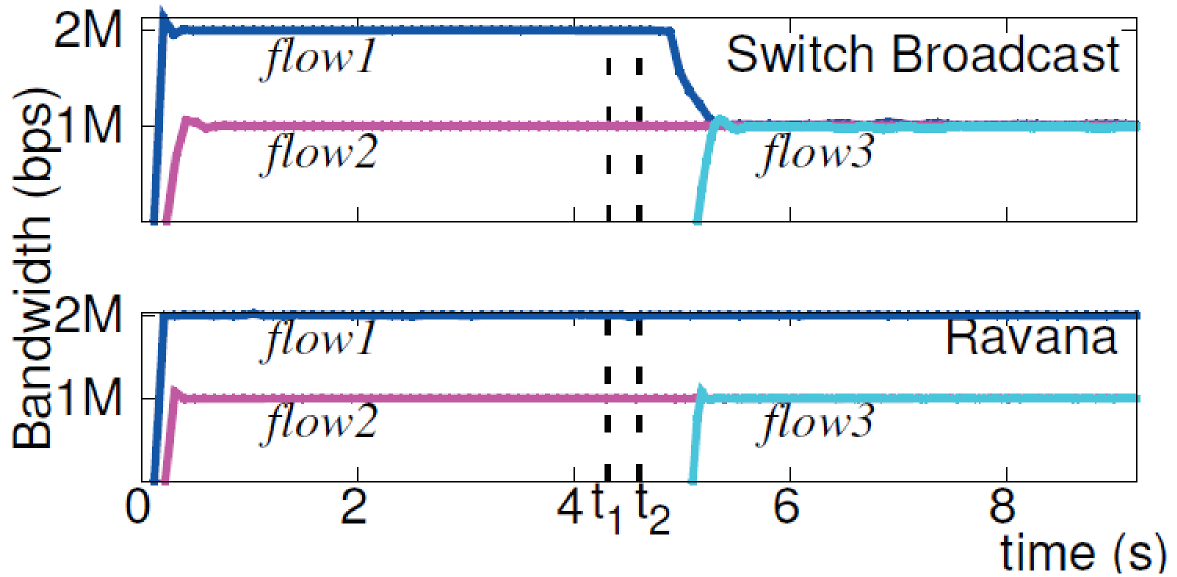
Total Event Ordering



Total Event Ordering – Design Goal #1

- Controller replicas should process events in the same order.
- All controller application instances should reach the same internal state.

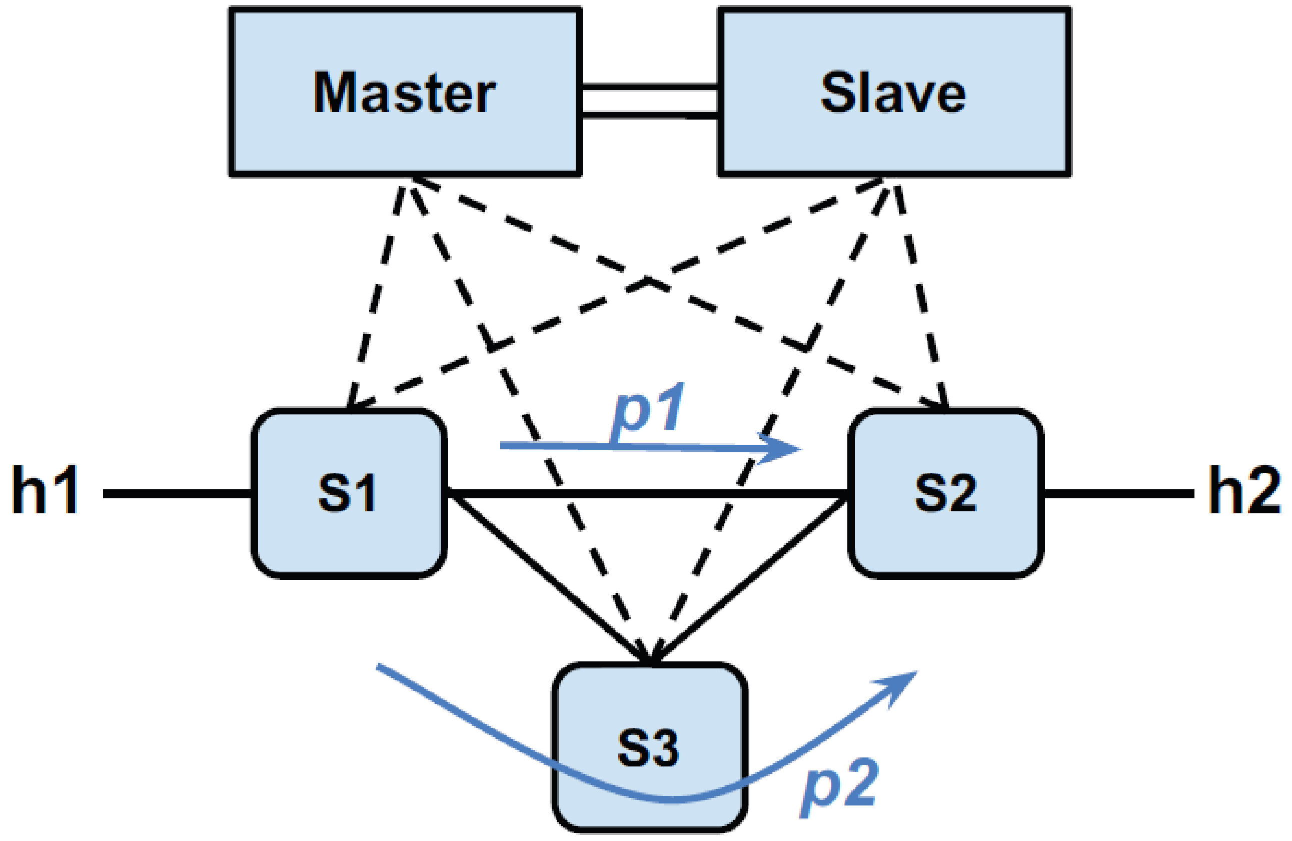




Bandwidth Allocation

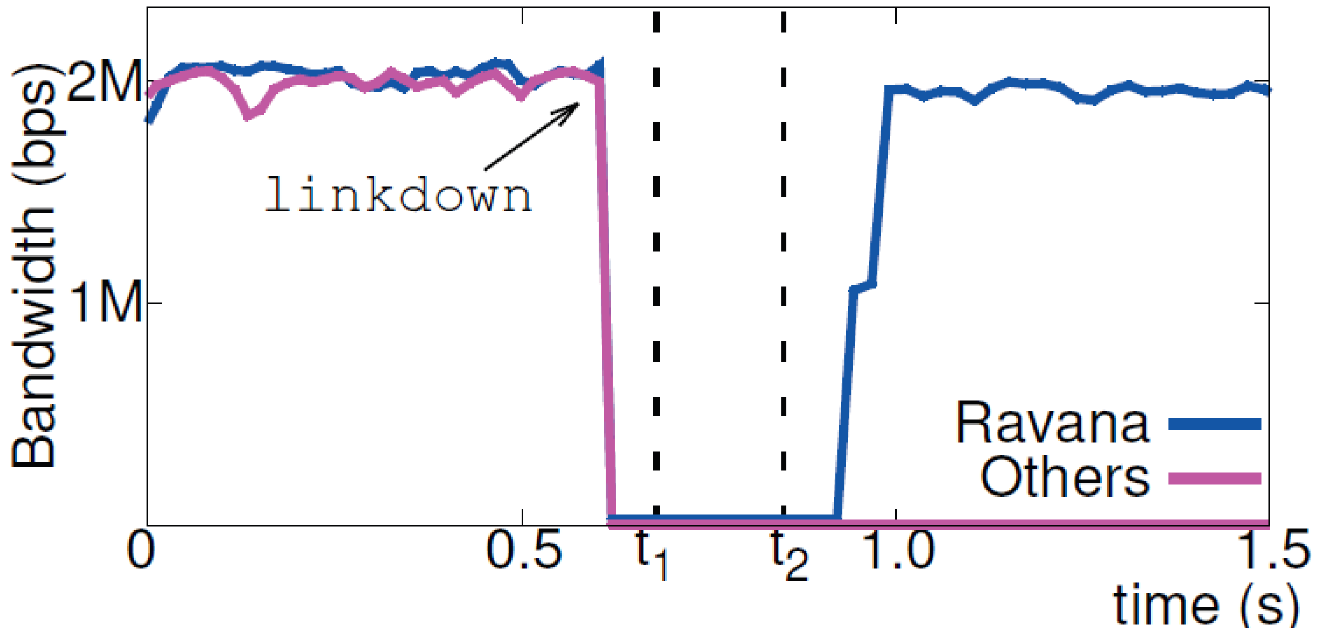
ETH zürich

Exactly-Once Event Processing



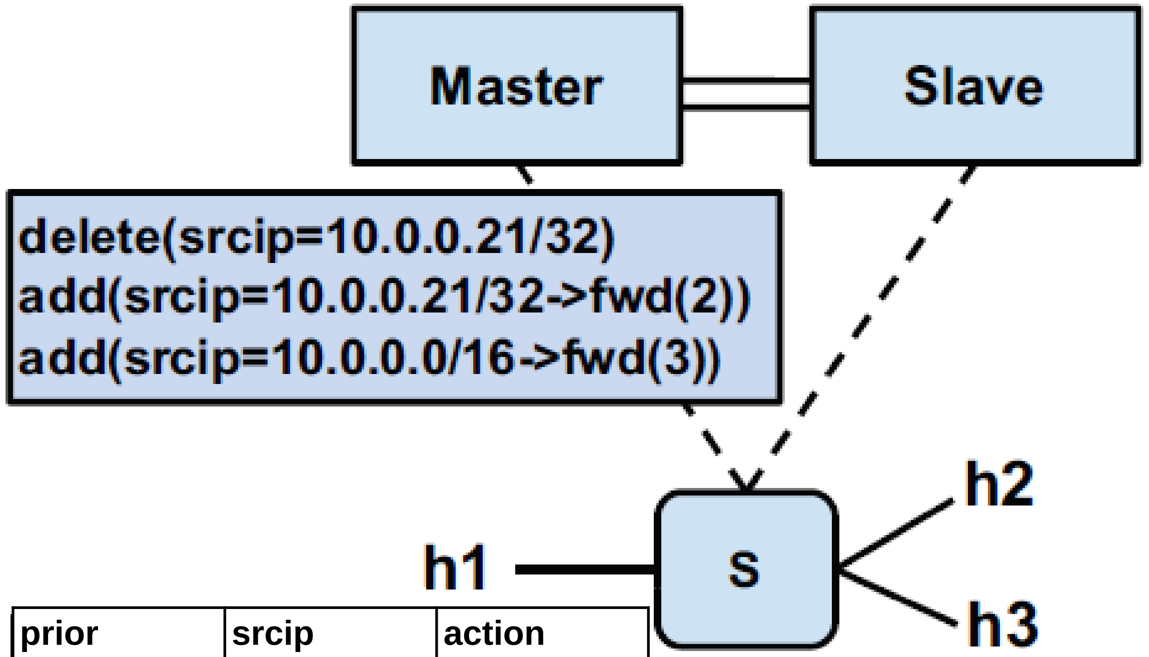
Exactly-Once Event Processing – Design Goal #2

All the events are processed, and neither lost nor processed repeatedly.



linkdown Under Failures

Exactly-Once Execution of Commands

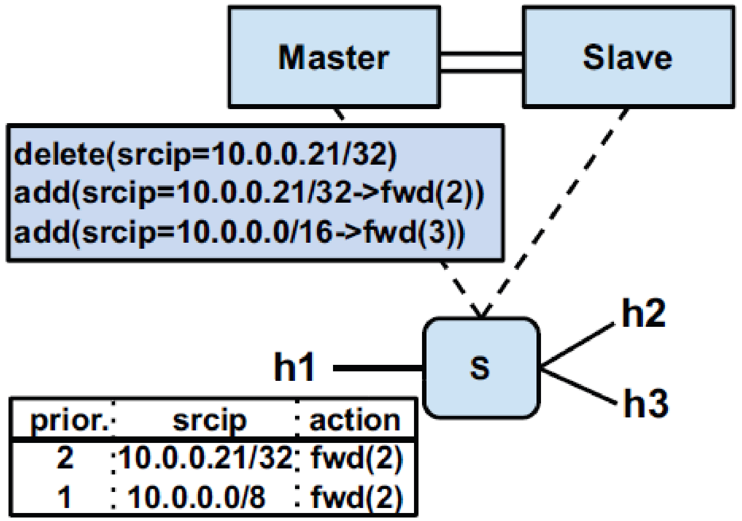


delete(srcip=10.0.0.21/32)
add(srcip=10.0.0.21/32->fwd(2))
add(srcip=10.0.0.0/16->fwd(3))

prior	srcip	action
3	10.0.0.21/32	fwd(2)
2	10.0.0.0/16	fwd(3)
1	10.0.0.0/8	fwd(2)

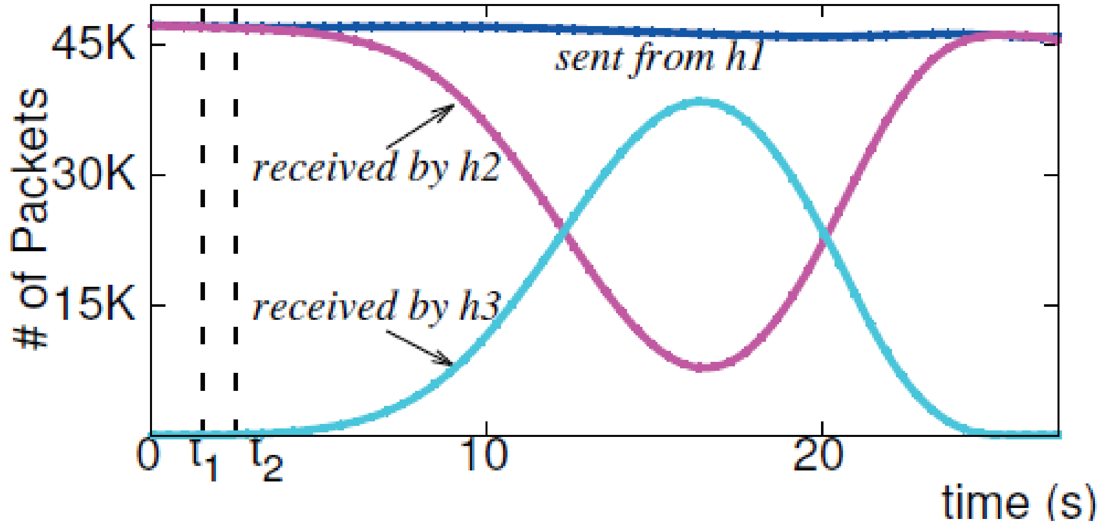
Exactly-Once Execution of Commands – Design Goal #3

- Any given series of commands are executed once and only once on the switches.



```
delete(srcip=10.0.0.21/32)
add(srcip=10.0.0.21/32->fwd(2))
add(srcip=10.0.0.0/16->fwd(3))
```

prior.	srcip	action
2	10.0.0.21/32	fwd(2)
1	10.0.0.0/8	fwd(2)



Routing Under Repeated Commands

Ravana Protocol

Ravana

- **Controller platform that provides an abstraction of a fault-free centralised controller**

- Entire event-processing cycle = Transaction
  - All or none of the transaction components are executed.

- Uses existing distributed systems' techniques in SDN

Ravana Contributions

- Two-phase replication protocol
- OpenFlow interface extensions
- Correctness properties for centralised controller
- Real transparent runtime prototype with low overhead

Ravana Components – Replication Protocol

- Two-phase replication protocol
  - Extends RSM
  - Each phase adds event-processing information to a replicated in-memory log

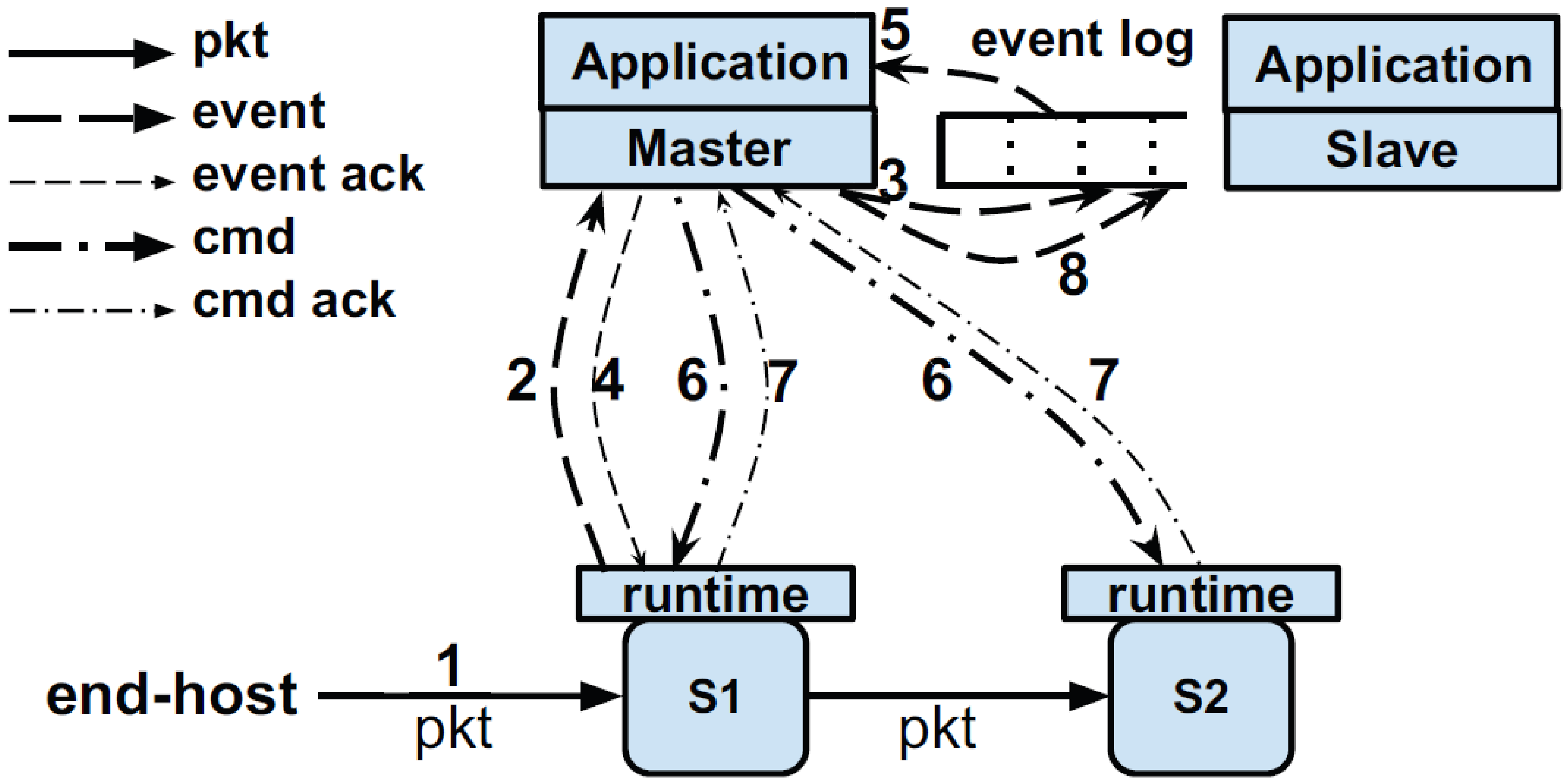
- **1st stage**
  - Ensures every received event is replicated
- **2nd stage**
  - Conveys that the event-processing transaction has completed.

Ravana Components – Extended Interface

- Extended Control Channel Interface
  - The channel between controller and switches

1. RPC level ACKs and retransmission mechanisms
   - Ensures message delivery at least once
2. Each message has unique ID, receive-side filtering
   - Guarantees at most once messages

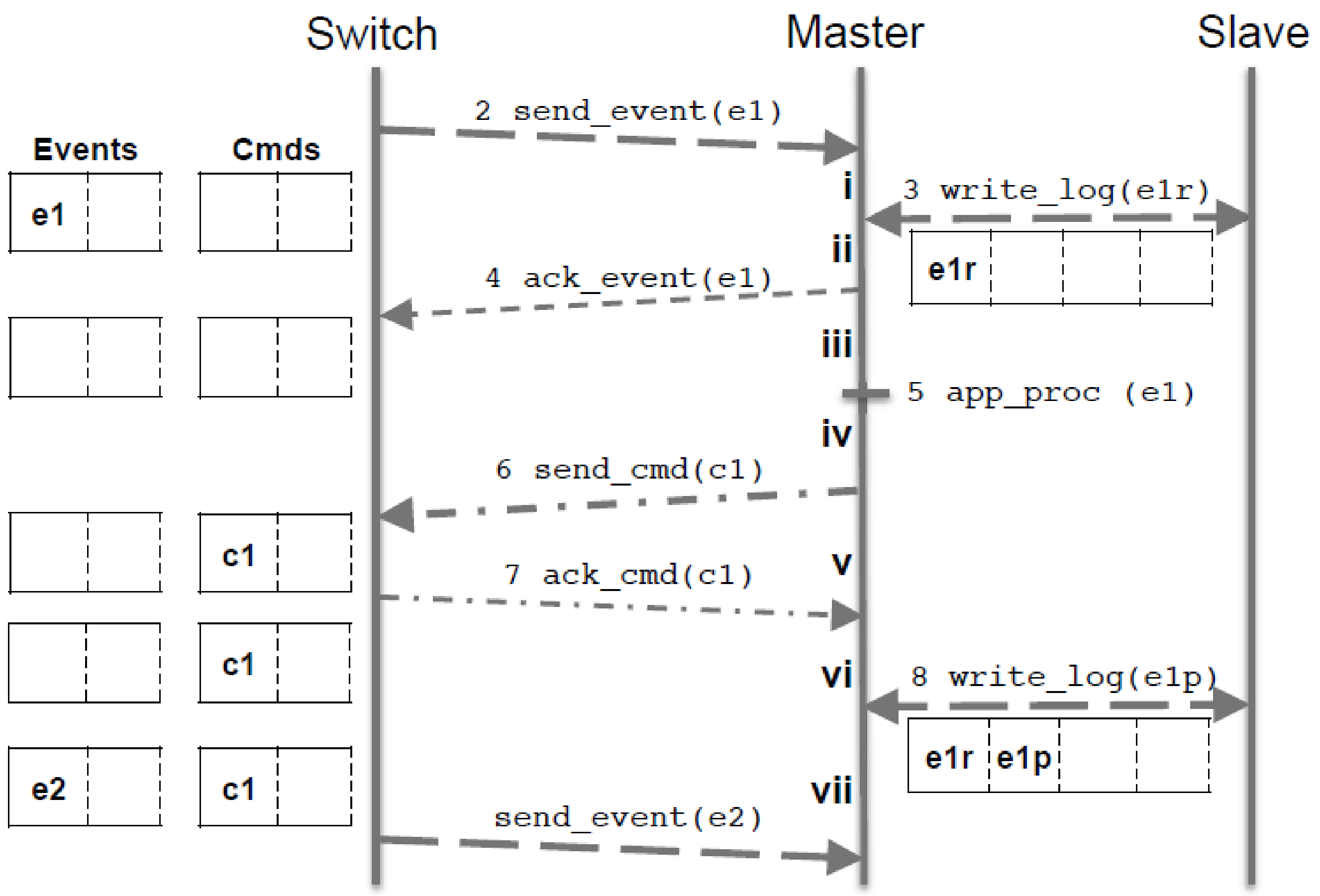
Protocol Overview



Master Controller Failure Case

- If master fails:
  1. A leader election component elects new master.
  2. New master finishes processing any logged events to catch up with failed master state.
  3. New master registers itself as a master with switches.
  4. Proceeds with normal controller operation.

Protocol Insights – Potential Fail Cases



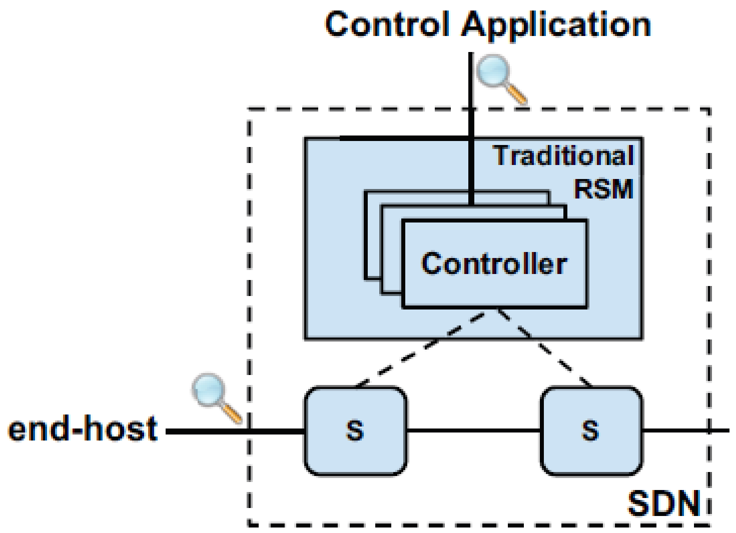
- Exactly-Once Event Processing
  - Crash case (i) ✓
  - Crash case (ii) ✓
- Total Event Ordering
  - Crash case (iii) and (iv) ✓
- Exactly-Once Command Execution
  - Crash case (v) and (vi) ✓

Correctness

Observational Indistinguishability

- *If the trace of observations made by users in the fault-tolerant system is a possible trace in the fault-free system, then the fault-tolerant system is* ***observationally indistinguishable from a fault-free system****.* ✓

- Two properties:
  - Safety
  - Liveness

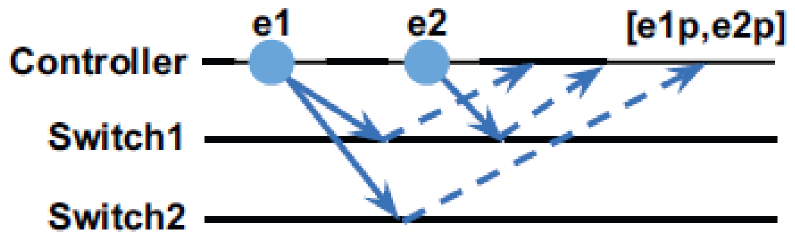


- Ravana provides transactional semantics to the entire “control loop”
  - Event delivery, ordering and processing
  - Command execution

Performance Optimisations

Performance Optimisations

- Parallel logging of events
  - Total order is imposed by IDs
  - Multiple threads write events in parallel
- Processing multiple transactions in parallel
  - Pipelining multiple commands without waiting for ACKs
  - TCP sorts out ordering
- Clearing switch buffers
  - Event buffer (Ebuf)
  - Command buffer (Cbuf)



Implementation

ETH zürich

1. Controller Runtime

- Ryu
  - Message-parsing library
  - Raw messages -> OpenFlow messages
- Leader election
  - ZooKeeper
  - Failure detected with a help of *hearbeat* messages
  - Election as a competition for a master lock
- Event logging
- Event batching

Modifications:

1. **Controller runtime**
2. Switch runtime
3. Control channel

2. Switch Runtime

- Event and command buffers
  - Modified Open vSwitch (v1.10)

  - If master fails, connection manager sends buffered events.
  - Filters to check if a newly received command has been executed already.

Modifications:

1. Controller runtime
2. **Switch runtime**
3. Control channel

ETH zürich

3. Control Channel

- Modified OpenFlow 1.3 controller-switch interface
  - EVENT_ACK
  - CMD_ACK
  - Ebuf_CLEAR
  - Cbuf_CLEAR
- Unique transaction IDs (XID)
  - XID field increment on Open vSwitch

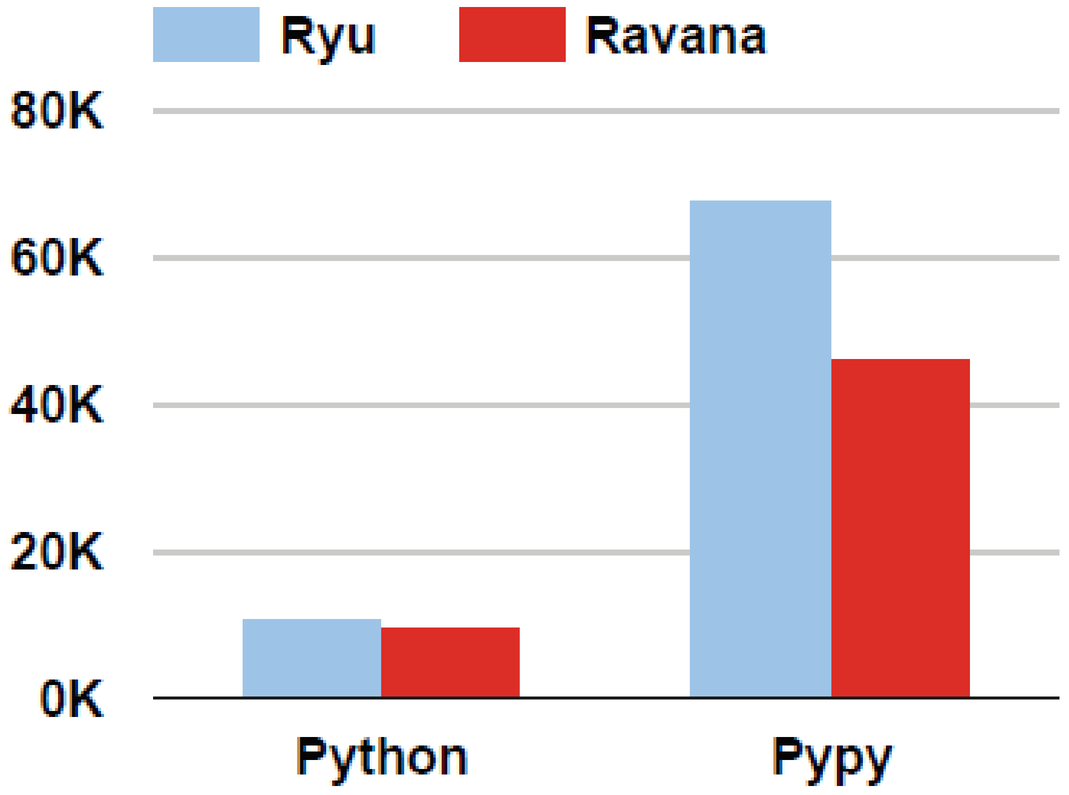
Modifications:

1. Controller runtime
2. Switch runtime
3. **Control channel**

Performance Evaluation

ETH zürich

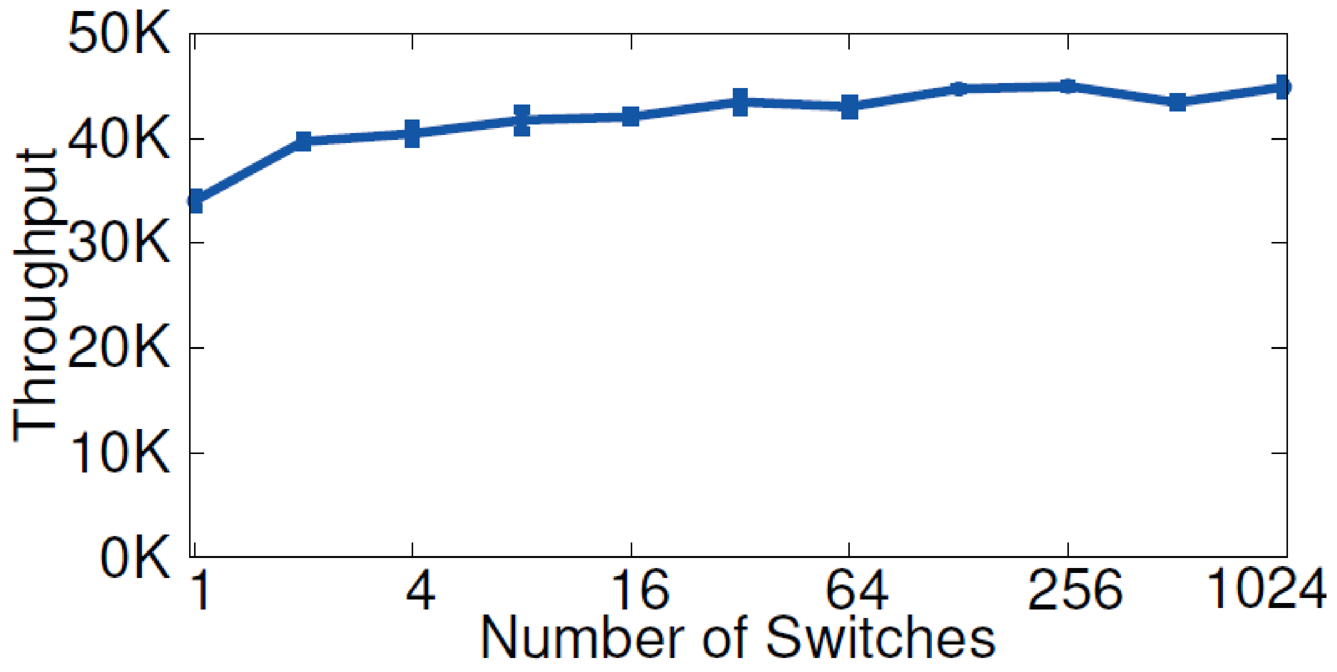
Throughput



Throughput Overhead
(flow responses per second)

- Ravana’s overhead
  - Python - 16.4%
  - PyPy - 31.4%

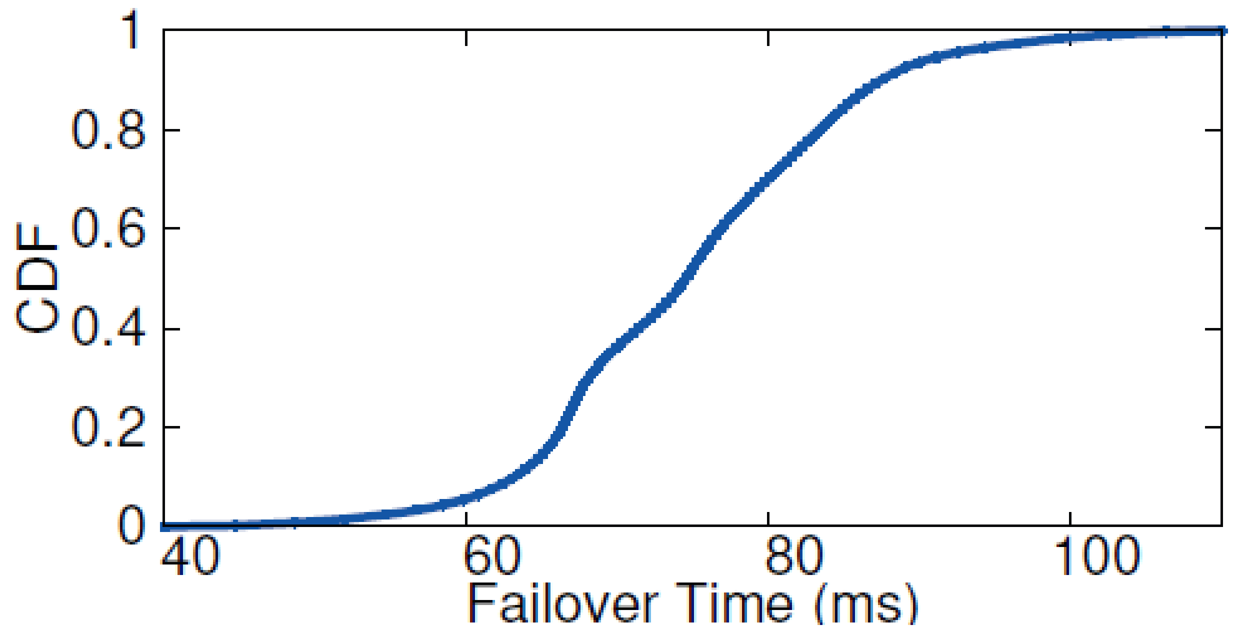
Scalability



Throughput with different number of switches

- Controller runtime can manage large number of parallel switch connections efficiently.

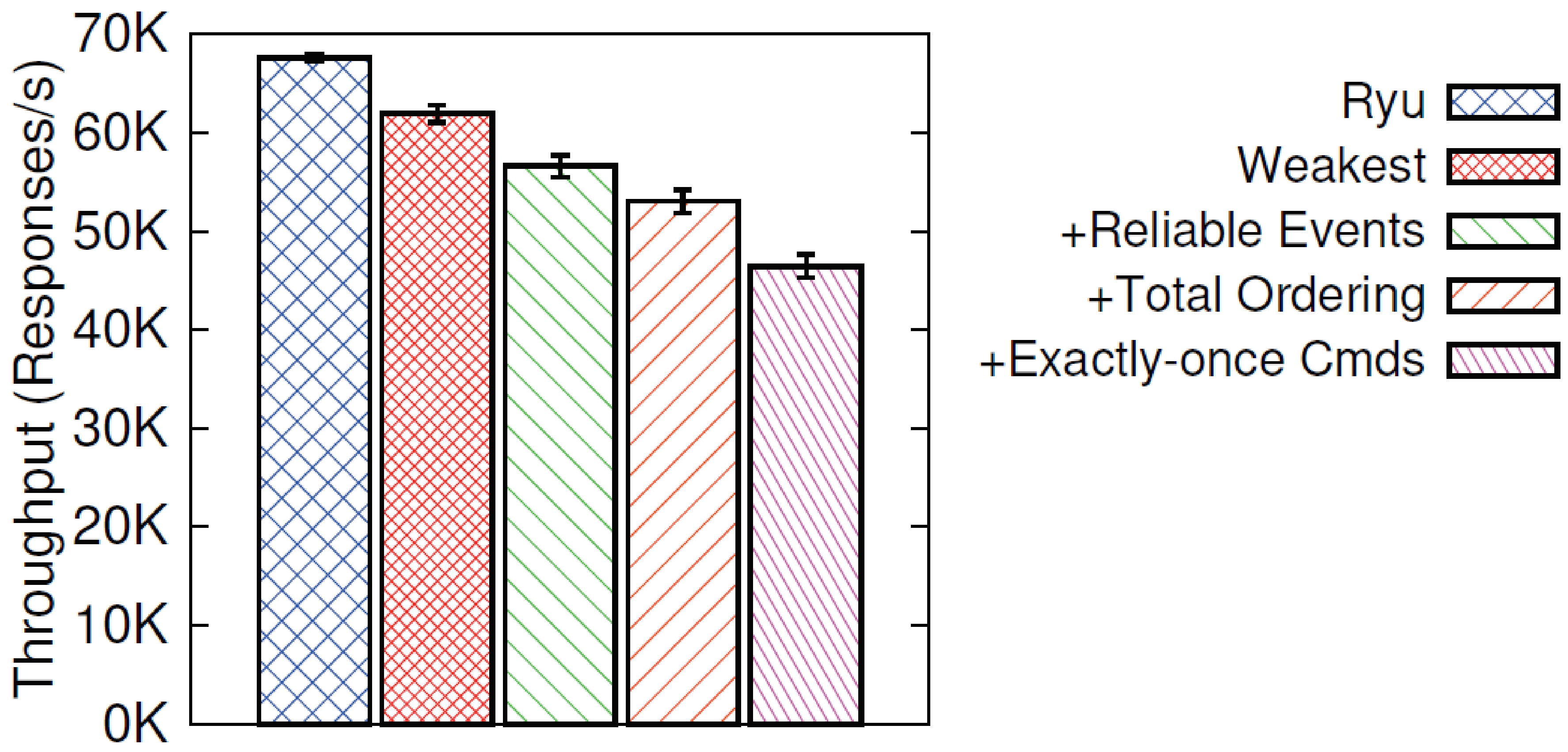
Failover Times



CDF for Failover Time

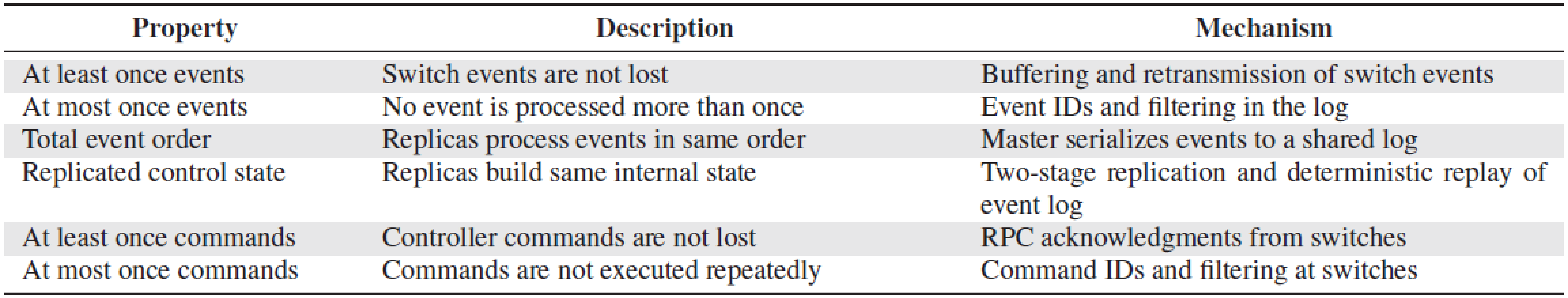
- Average failover time is **75ms**
  - **40ms** to detect failure and elect new leader
  - **25ms** to catch up with old master
  - **10ms** to register role with switches

Throughput Overhead



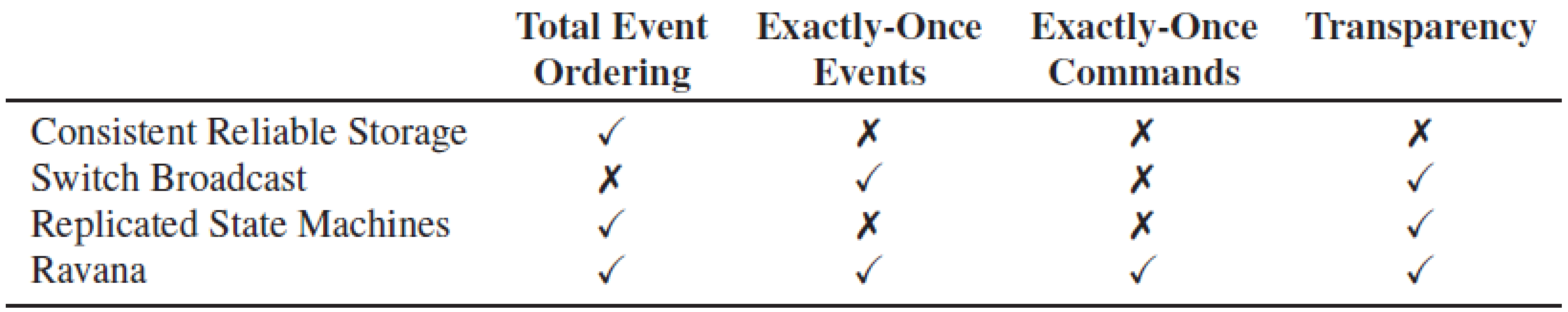
Throughput Overhead for Correctness Guarantees

Ravana: Summary



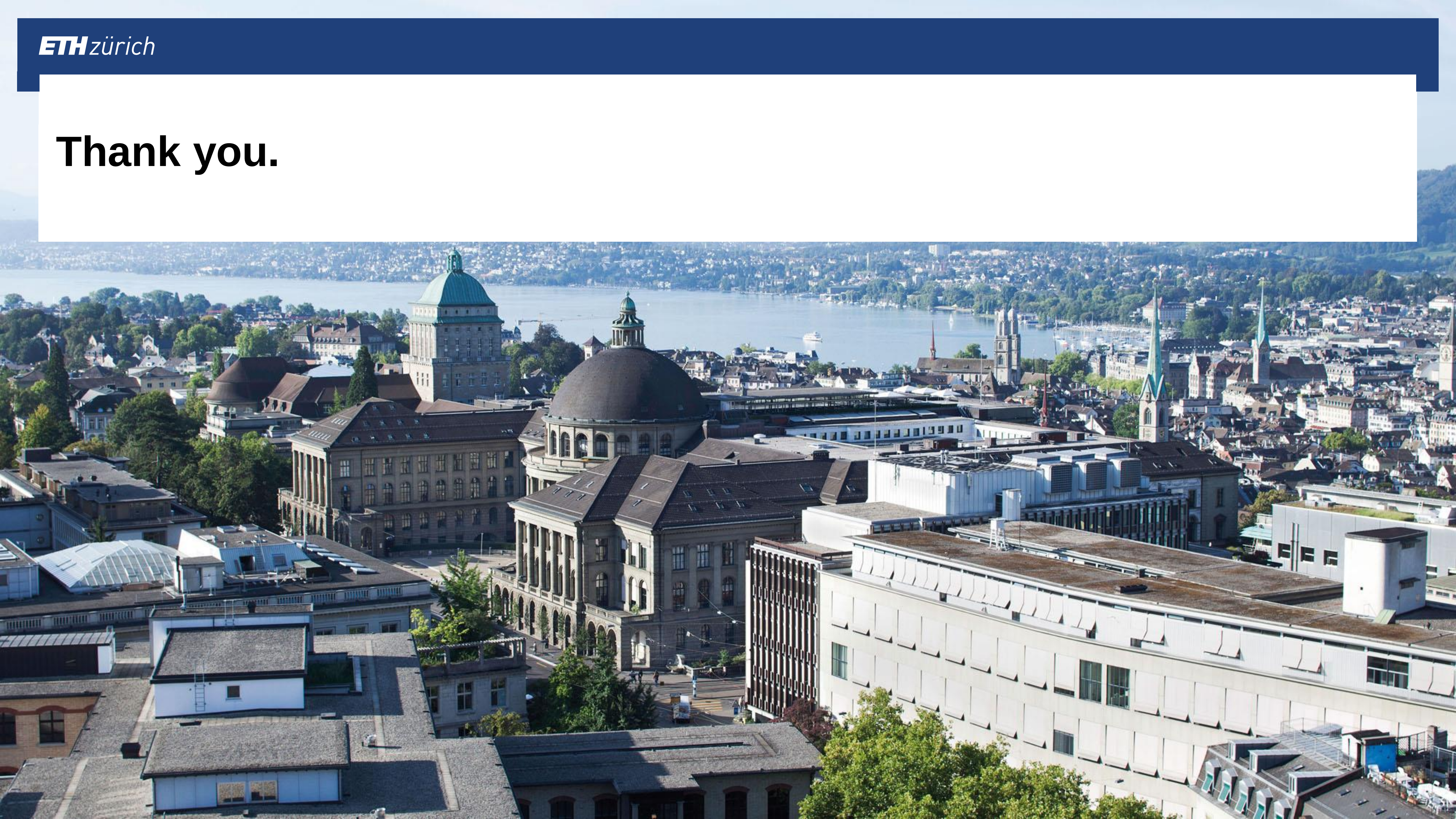
Property	Description	Mechanism
At least once events	Switch events are not lost	Buffering and retransmission of switch events
At most once events	No event is processed more than once	Event IDs and filtering in the log
Total event order	Replicas process events in same order	Master serializes events to a shared log
Replicated control state	Replicas build same internal state	Two-stage replication and deterministic replay of event log
At least once commands	Controller commands are not lost	RPC acknowledgments from switches
At most once commands	Commands are not executed repeatedly	Command IDs and filtering at switches

Design Goals and Mechanisms



	Total Event Ordering	Exactly-Once Events	Exactly-Once Commands	Transparency
Consistent Reliable Storage	✓	✗	✗	✗
Switch Broadcast	✗	✓	✗	✓
Replicated State Machines	✓	✗	✗	✓
Ravana	✓	✓	✓	✓

Different solutions for fault-tolerant controllers



Thank you.

ETH zürich

References

- Naga Katta, Haoyu Zhang, Michael Freedman, and Jennifer Rexford. 2015. Ravana: controller fault-tolerance in software-defined networking. In Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research (SOSR '15). ACM, New York, NY, USA, Article 4, 12 pages.

- Wikipedia contributors, "Ravana" Wikipedia, The Free Encyclopedia, https://en.wikipedia.org/w/index.php?title=Ravana&oldid=719503575 (accessed May 10, 2016).