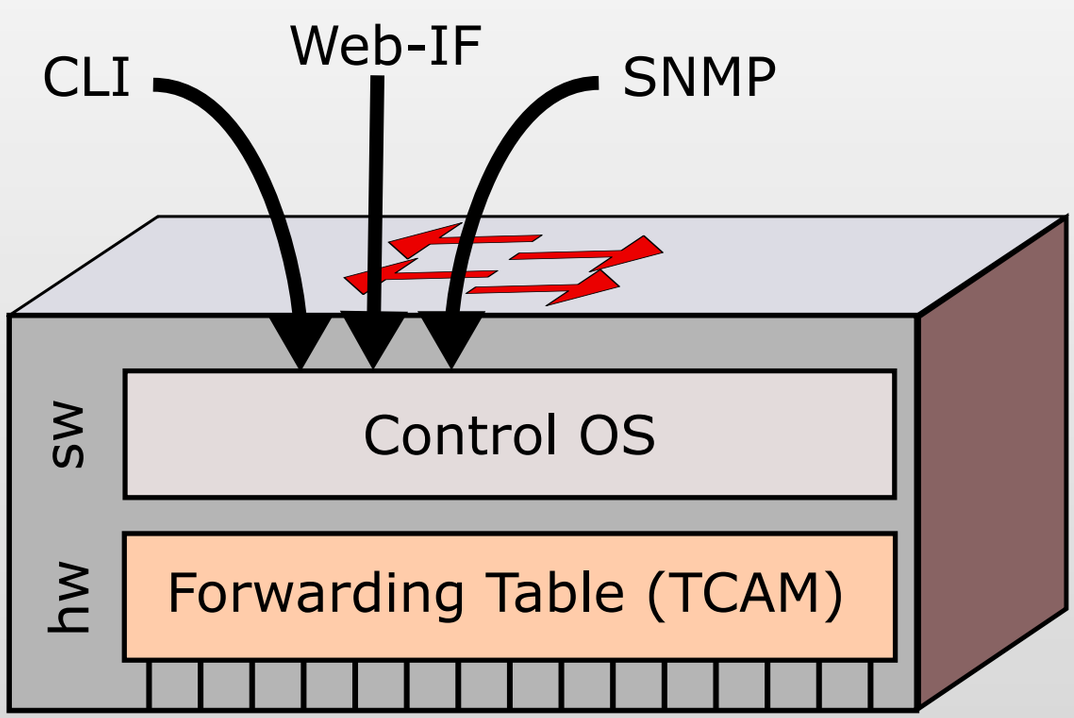
Deutsche Telekom Laboratories
An-Institut der Technischen Universität Berlin

OFRewind: Enabling Record & Replay Troubleshooting for Networks

Andreas Wundsam • Dan Levin
Srini Seetharaman • Anja Feldmann

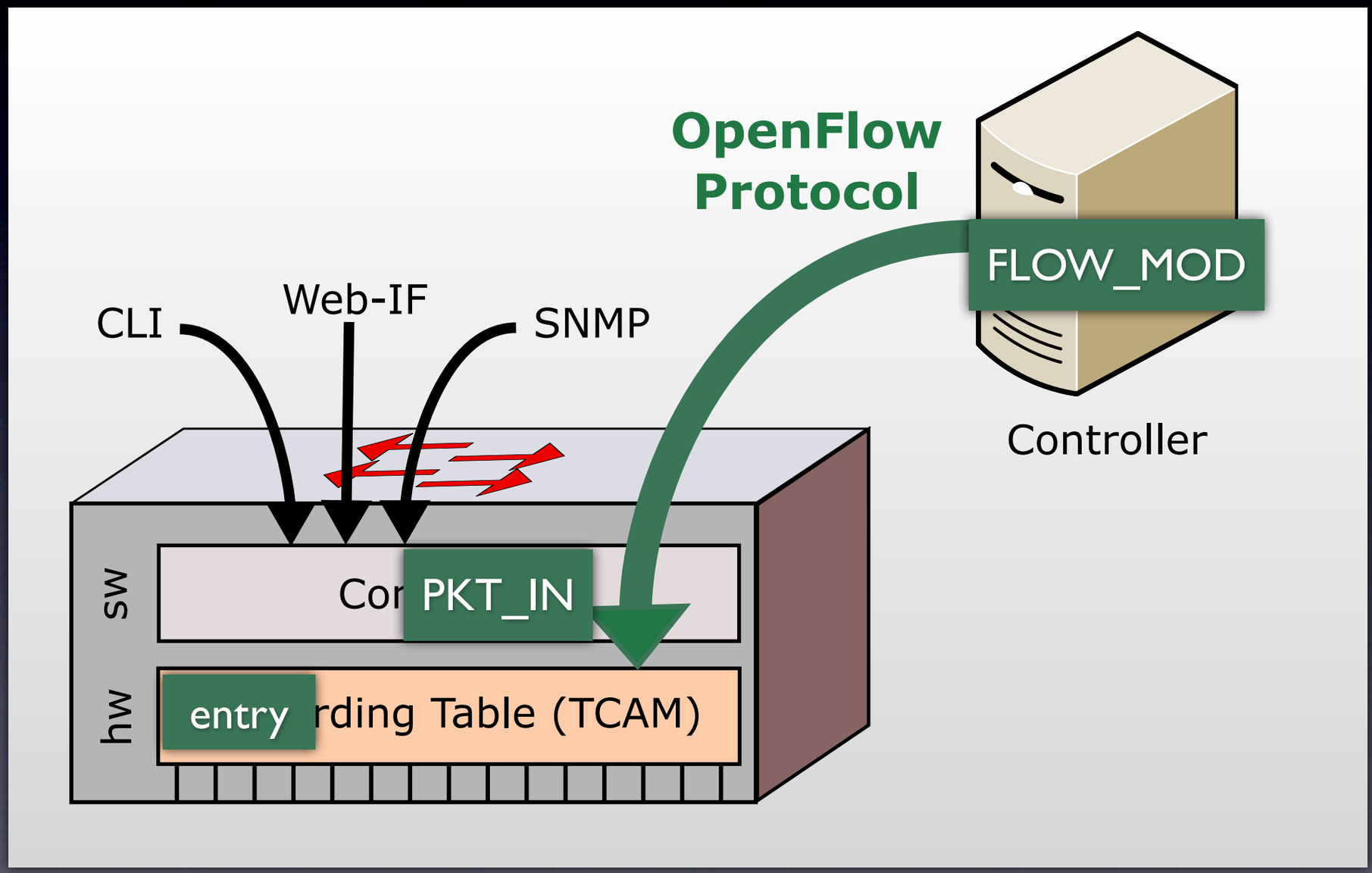
USENIX ATC 2011

Quick OpenFlow 101



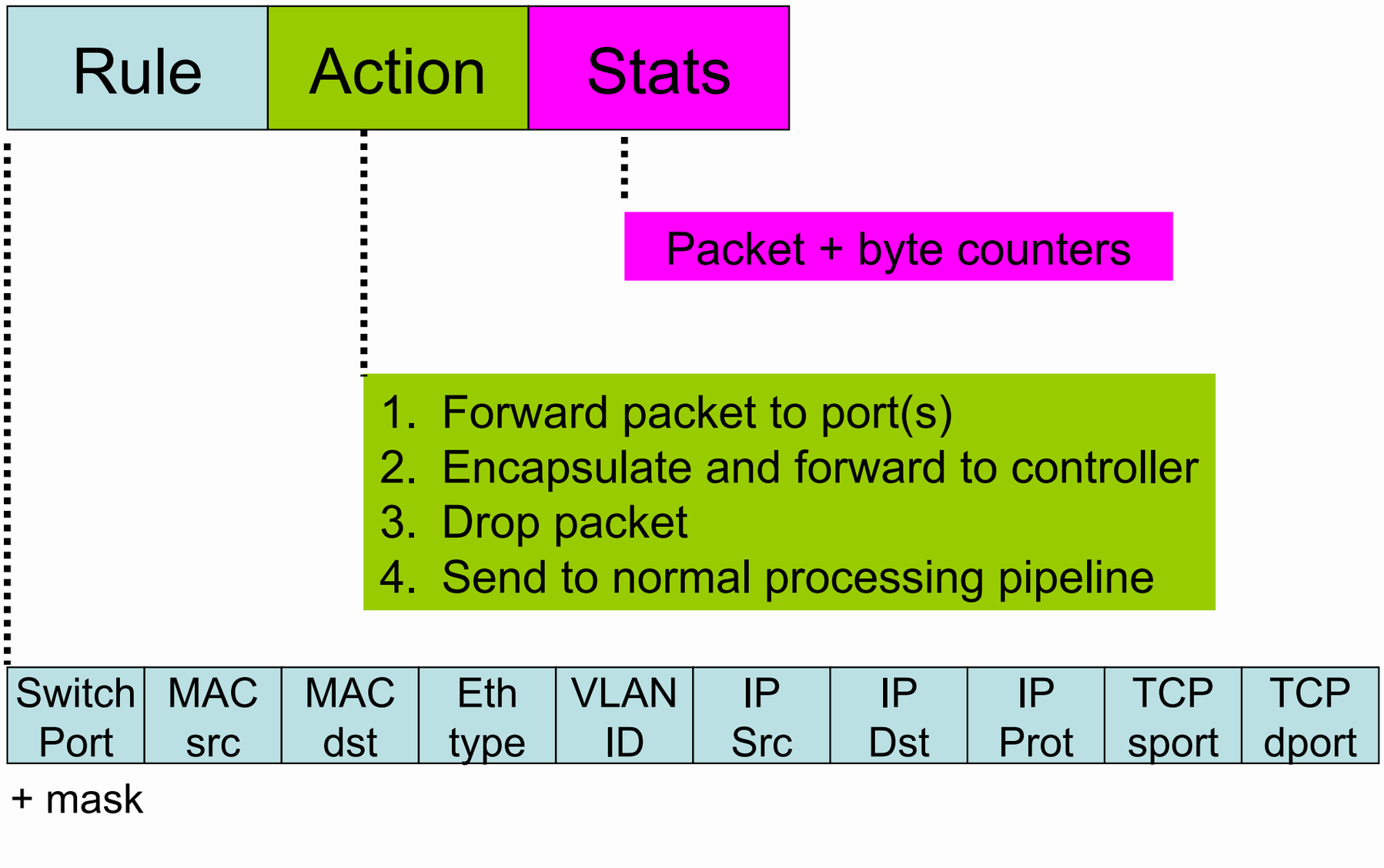
classical switch

Quick OpenFlow 101



OpenFlow switch

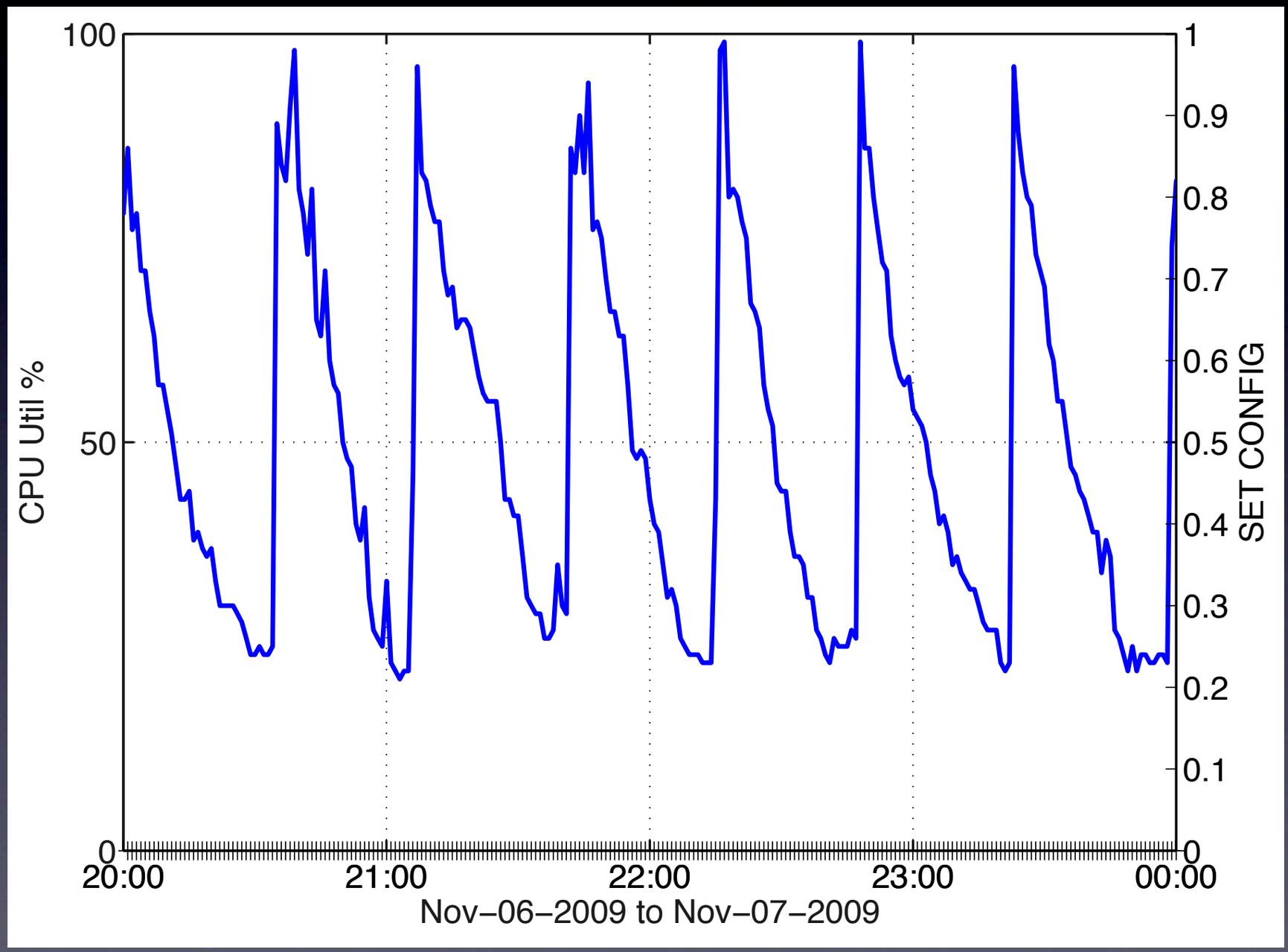
OpenFlow entry



(Figure from the Openflow Intro Presentation, N. McKeown)

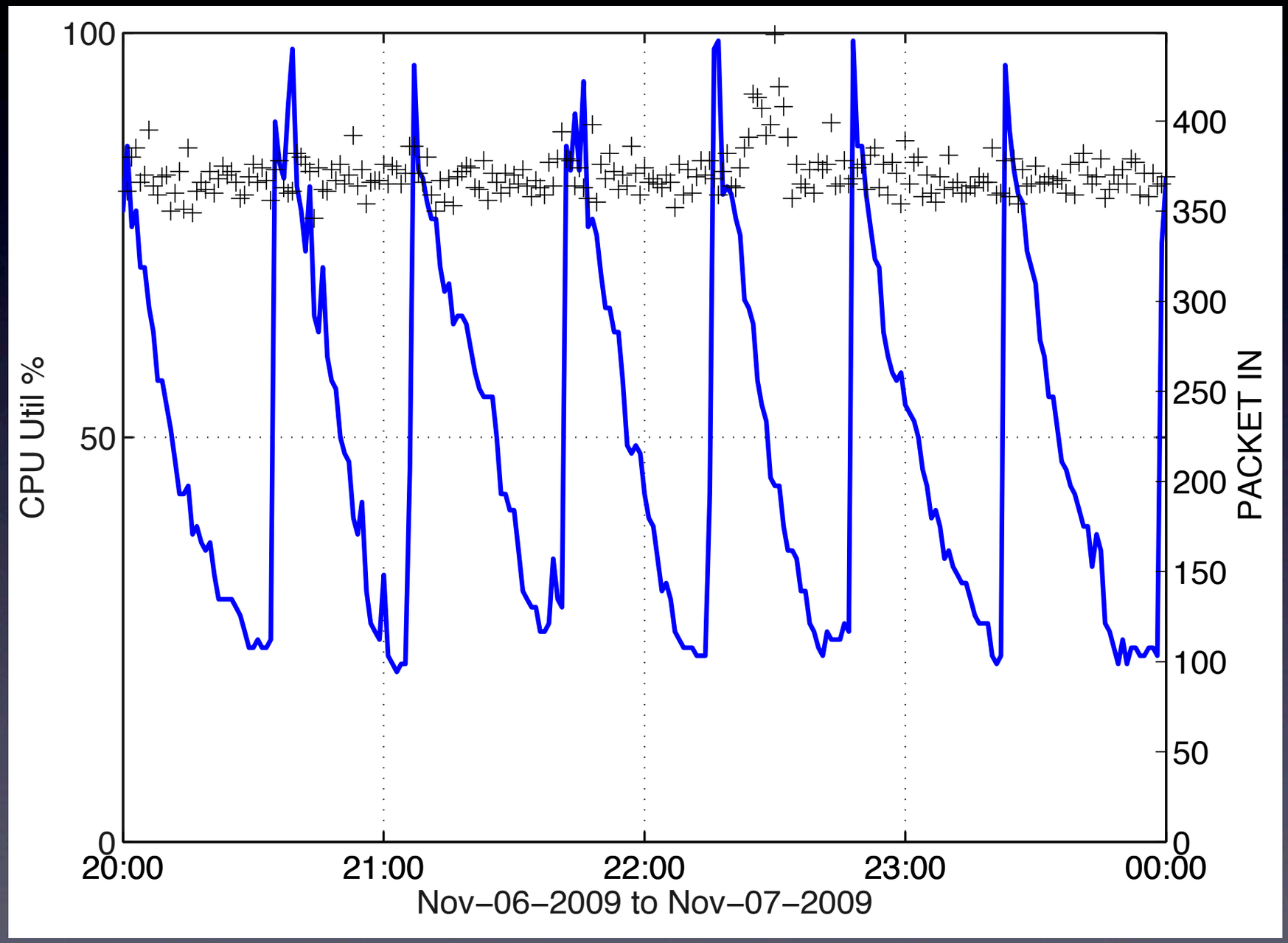
Back to the topic of my talk: OFRewind!

Motivating use case



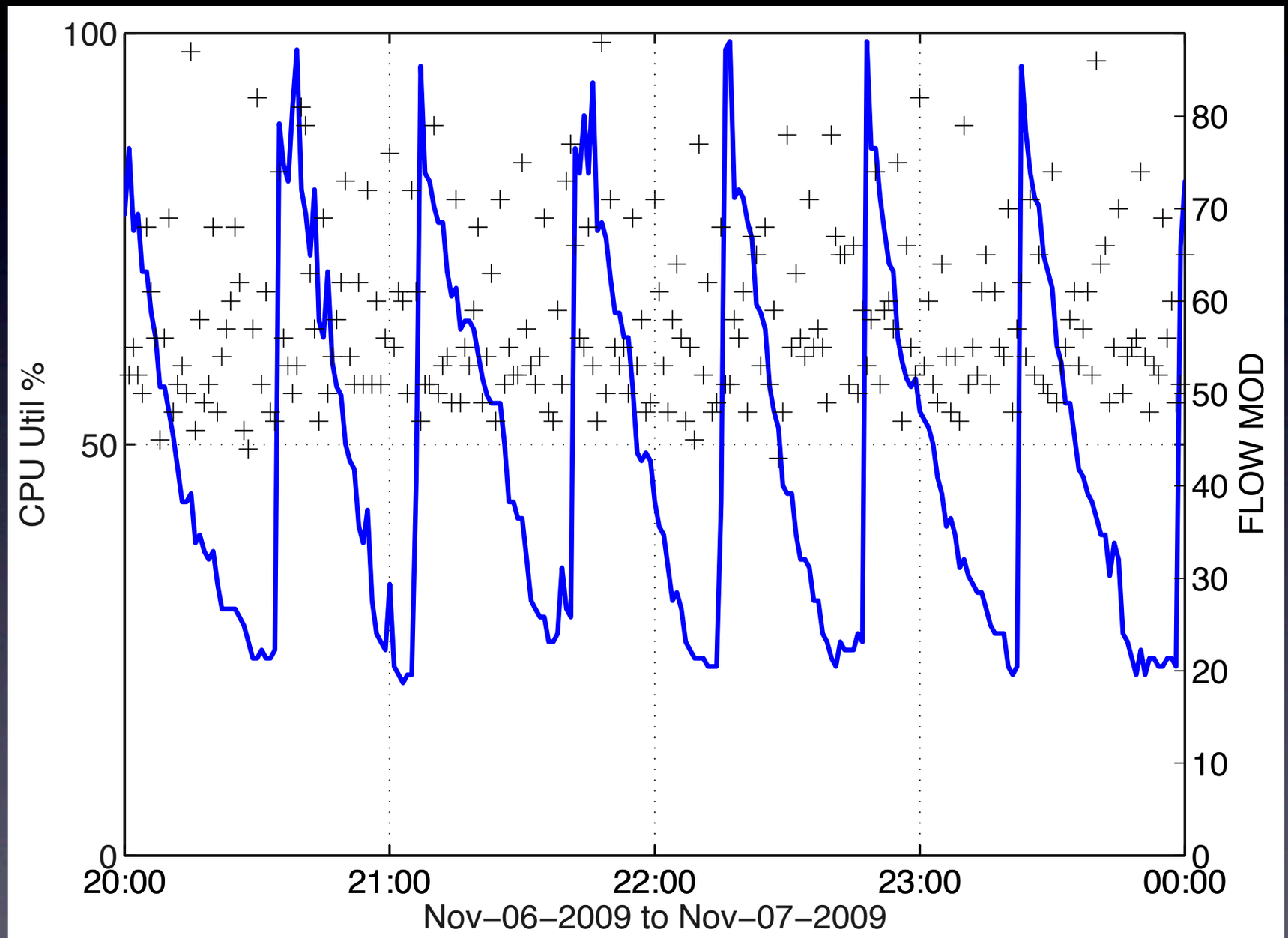
CPU Utilization of an OpenFlow switch

No correlation!



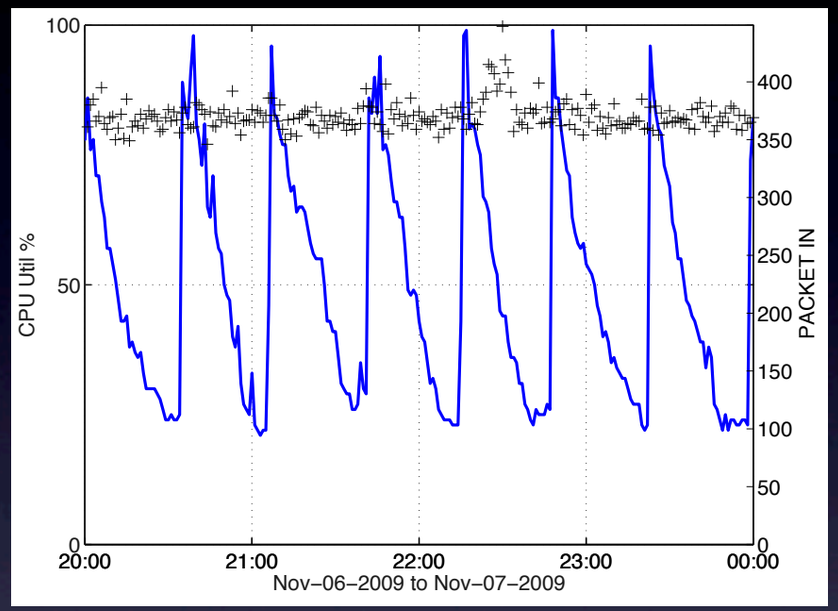
Arrivals of PKT_IN msgs

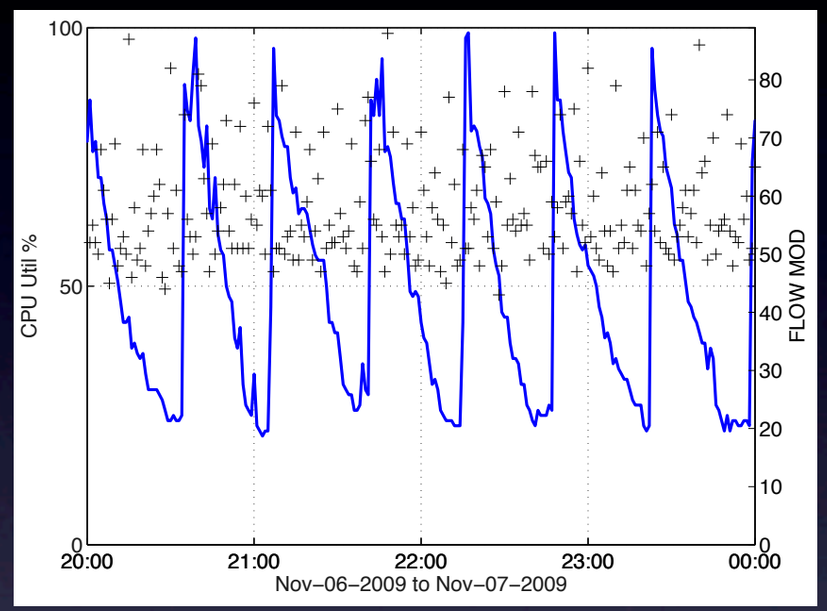
No correlation!

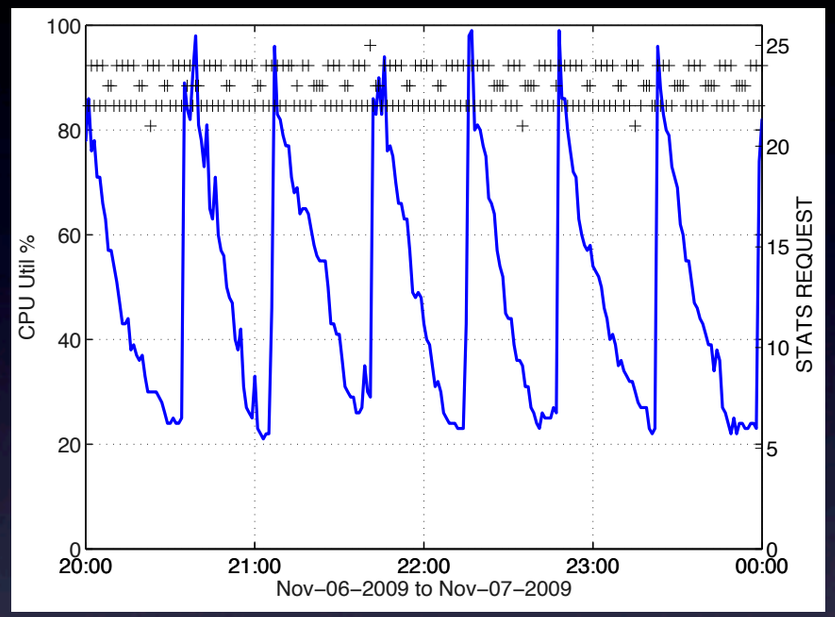


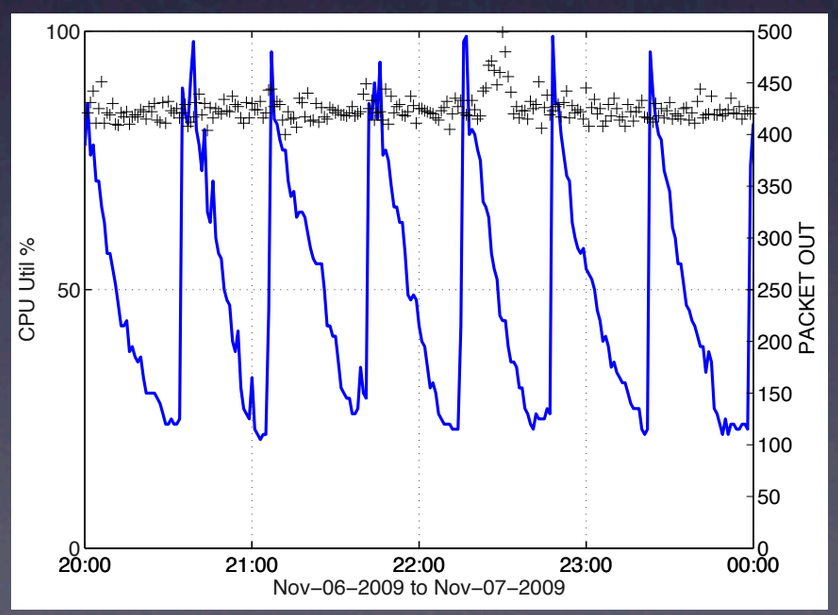
Arrivals of FLOW_MOD msgs

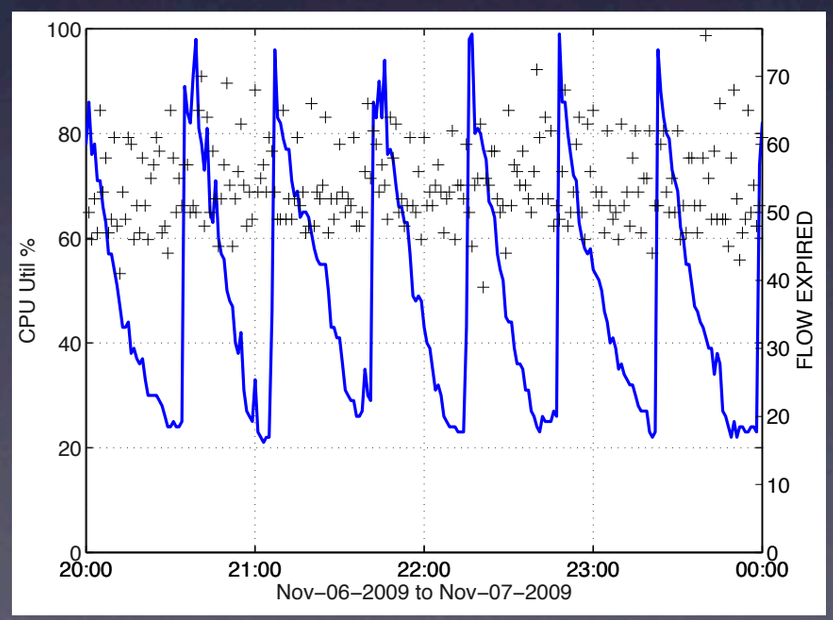
No correlation!

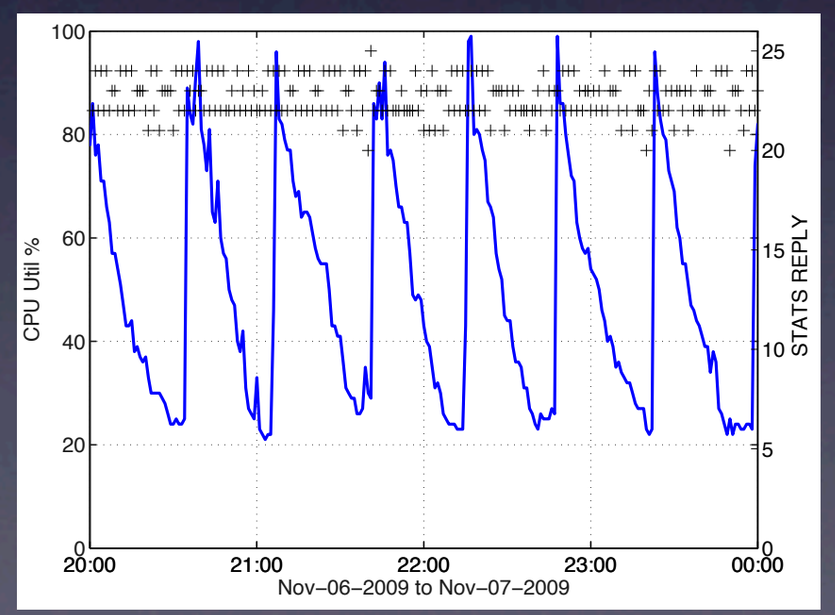












Clueless...

- Switch is a **black box component**
  - Can't inspect internal state, source code
  - No analytical explanation for the behavior
    - Message arrivals do not correlate with symptoms
  - Existing interfaces (CLI, SNMP) too coarse grained

Troubleshooting networks is hard

huge, critical

black boxes

timing / races

A solution?

In production

Reproduce at convenient location / pace

Existing approaches

Endhost Replay Debugging

Fully deterministic replay, via binary instrumentation / virtualization

✗ no black boxes

✗ scalability?

TCPDump / TCPReplay et. al.

Capture/Replay events

✗ Single vantage point, no network wide view

✗ Scalability due to dataplane datarates

Existing approaches

Endhost Replay

Fully
bin

virtualization

✗ no black boxes

✗ scalability?

TCPDump / TCPReplay

nts

t, no
network wide view

✗ Scalability due to dataplane datarates

Full recording of all events feasible?

However...

- Not all traffic is equal
  *(ctrl plane: 1% traffic, 95-99% bugs!)**
- Behavior of many network devices:

Largely Deterministic w.r.t.

Control Plane Network Events

* *Altekar / Stoica, 2010*

Record ●

events + traffic

selective: record important traffic (control)

skip/**aggregate** less important traffic (data plane)

Replay ▶

reinject events + traffic
"best effort replay"

replay partial recordings

reproduce problem at a chosen time / location

Go Network* Wide / Always On!

* *controller domain*

Replay Tweaking

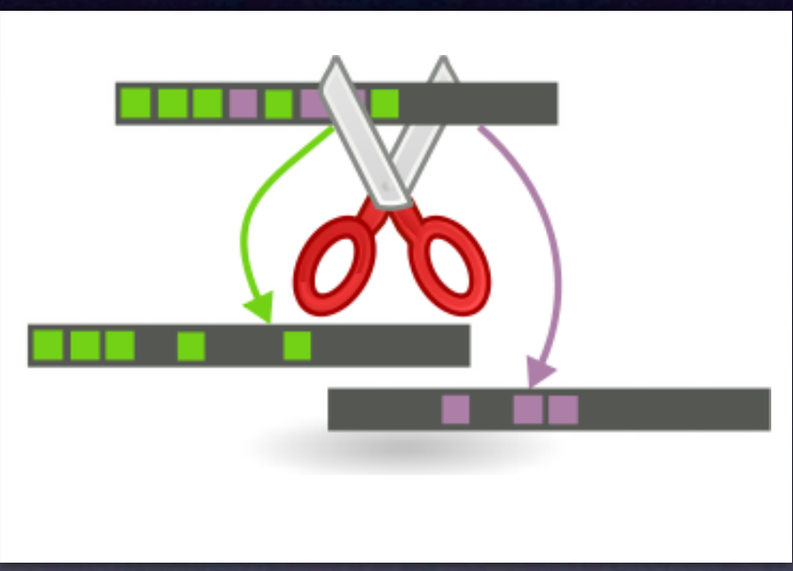
Localize problems through:

Device mapping

different devices / versions
investigate regressions /
vendor implementation issues

Time dilation

Scale time
investigate timing issues



Trace bisection

iteratively replay subselected traffic
localize events that trigger failure

Goals

- ✓ **Record** a controller domain
  - ✓ Scalable, selective, consistent
    - ✓ Even with black boxes
- ✓ coordinated **Replay**
  - ✓ Replay tweaking
  - ✓ Localize problems

Non-Goals

✗ **Root cause** analysis

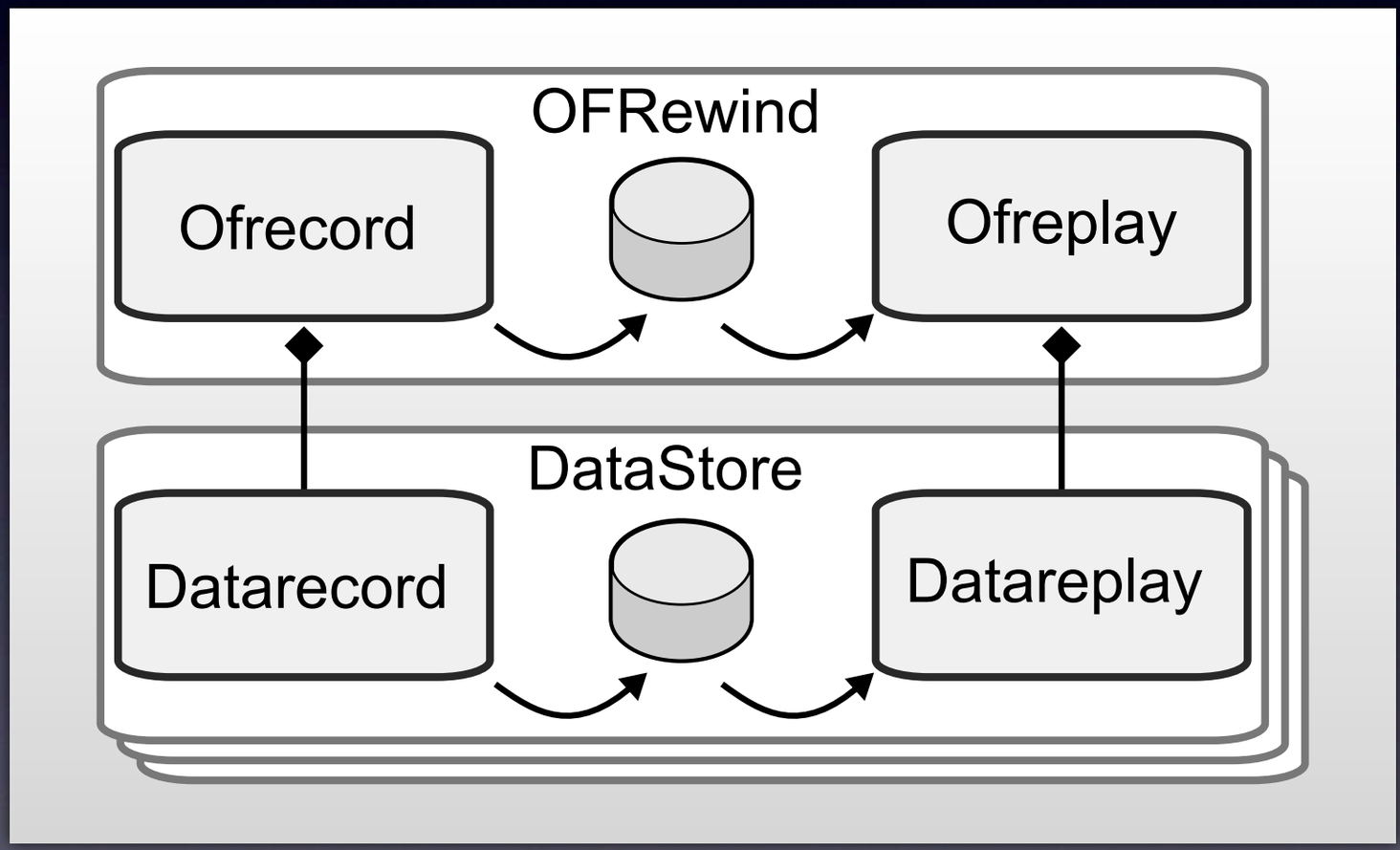
✗ **Automatic configuration** of what to record

✗ **Fully** deterministic replay

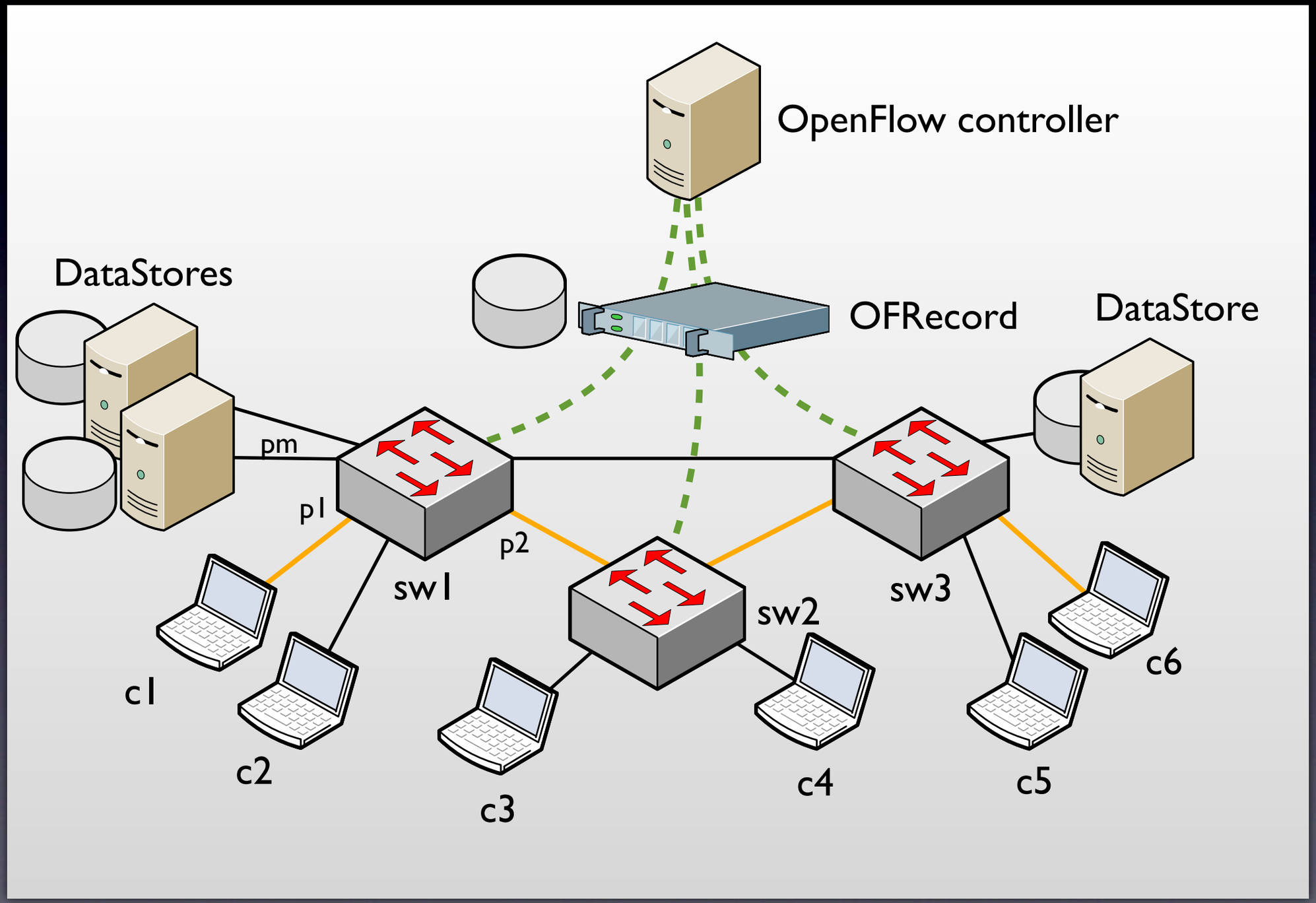
Introducing the tool

System design

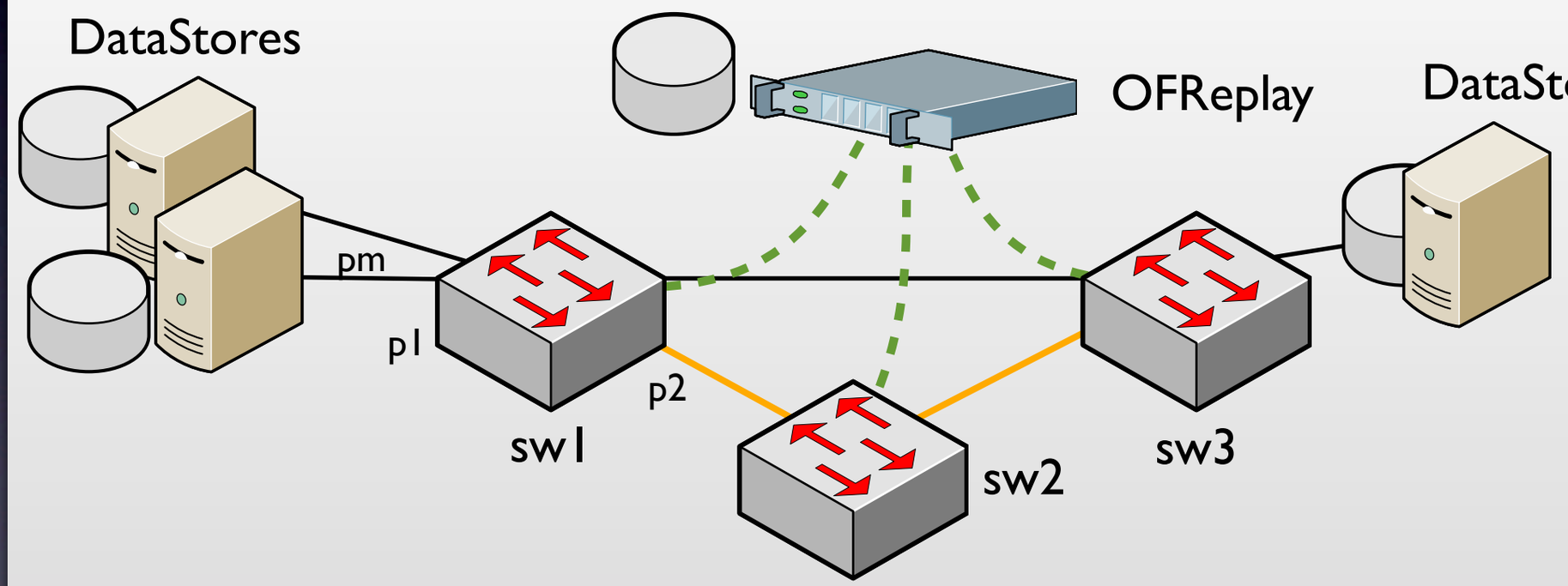
2 components of 2 modules each:



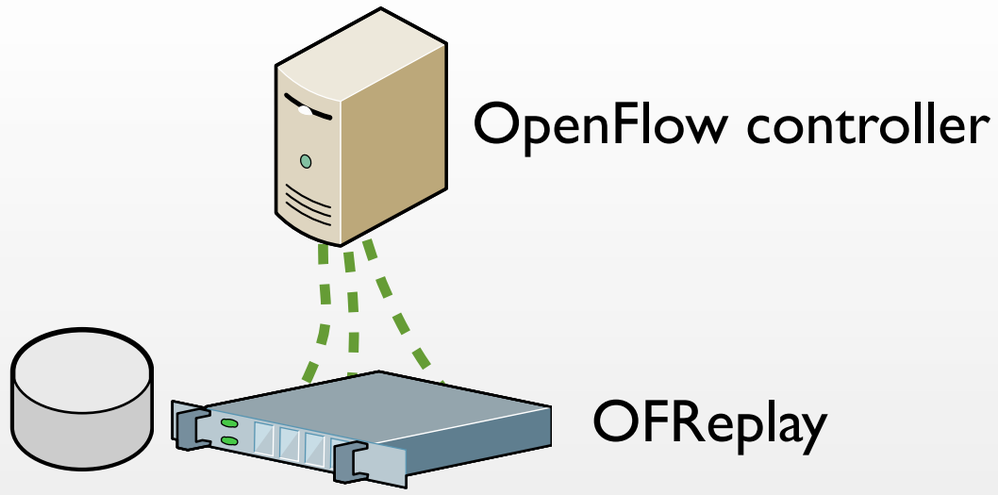
OFRecord


OpenFlow controller
DataStores
OFRecord
DataStore
pm
p1
p2
sw1
sw2
sw3
c1
c2
c3
c4
c5
c6

OFReplay



OFReplay



Typical Usage

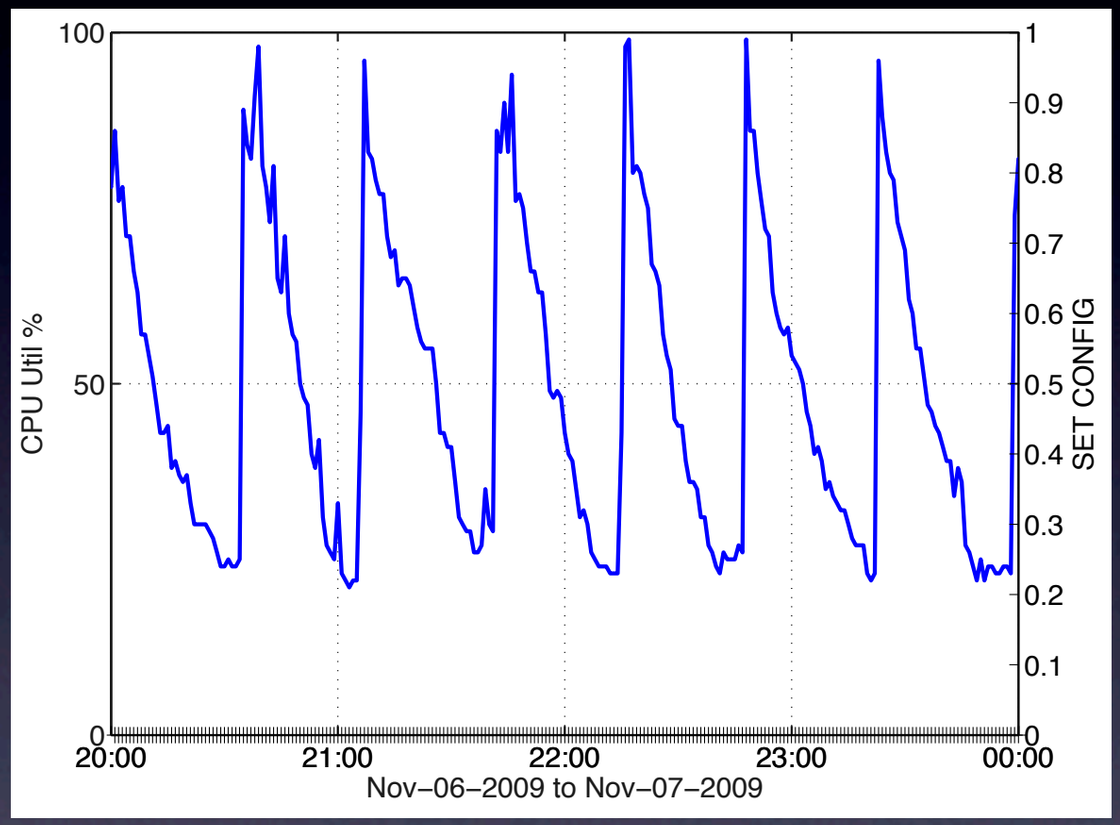
- Deploy **Ofrecord** in production environment -> proxy to 'regular' controller
  - **Always-on** OF messages, control plane, data plane summaries
  - Alter selection rules as necessary
- Deploy **Ofreplay** in lab environment
  - Localize bugs / validate bug fixes

Case studies

1. Debugging Black box components
   - *CPU inflation in an OpenFlow switch*
2. Debugging OpenFlow controllers
   - *NOX problem*

+ Others (see poster/paper)

Back to CPU inflation

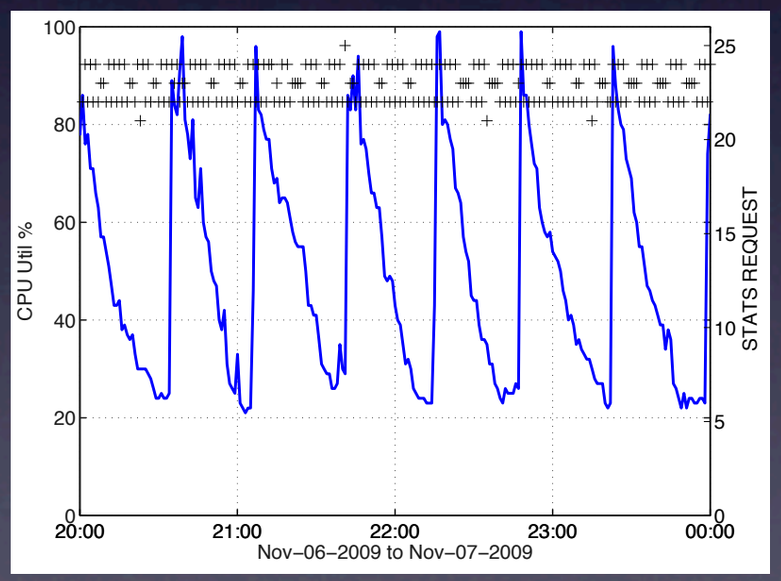


- Replay and bisect the trace by message type

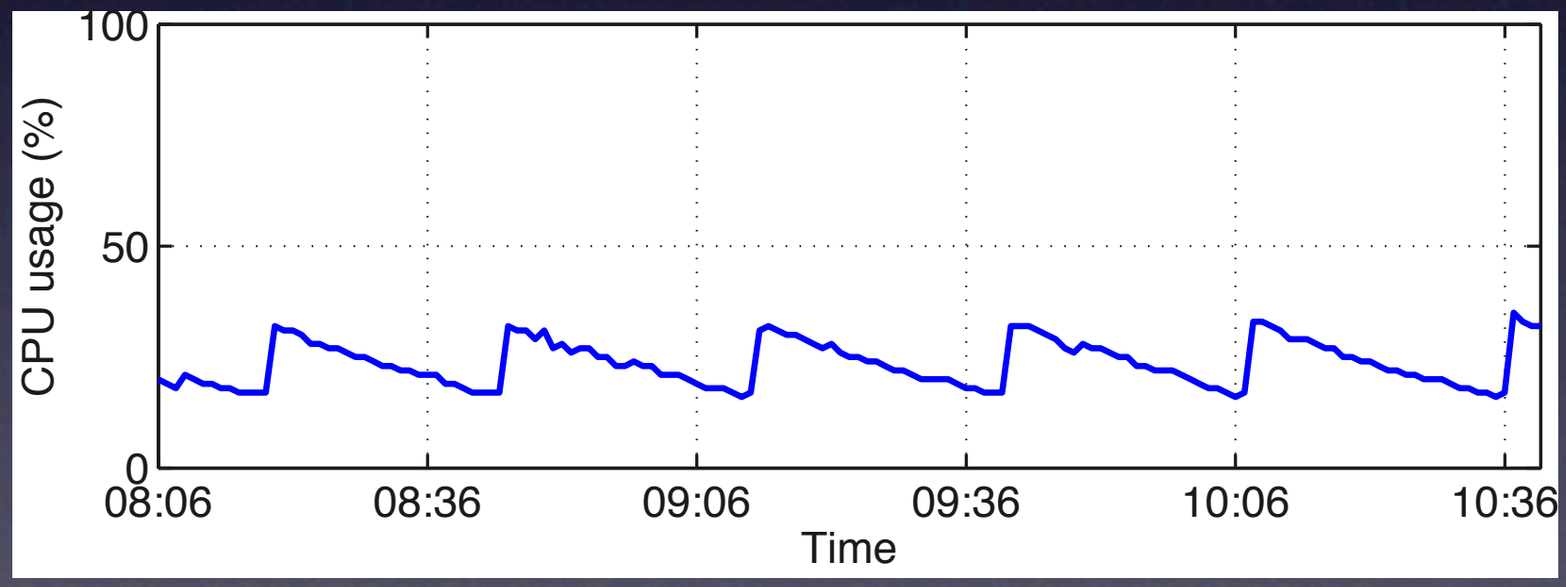
Back to CPU inflation

- Replay and bisect the trace by message type
- When replaying STATS_REQ msgs...

Record ●



Replay ▶



STATS_REQ msgs reproduce the problem
even though there is no correlation in arrival times

Debugging controllers: NOX problem

- Problem record: Messages initiated by one **specific device** don't reach NOX controller module
- Not reproducible at the lab

Debugging controllers: NOX problem

- Record at end user site
- Replay at lab towards NOX
- Use host-level debugging to analyze NOX behavior

Debugging controllers: NOX problem

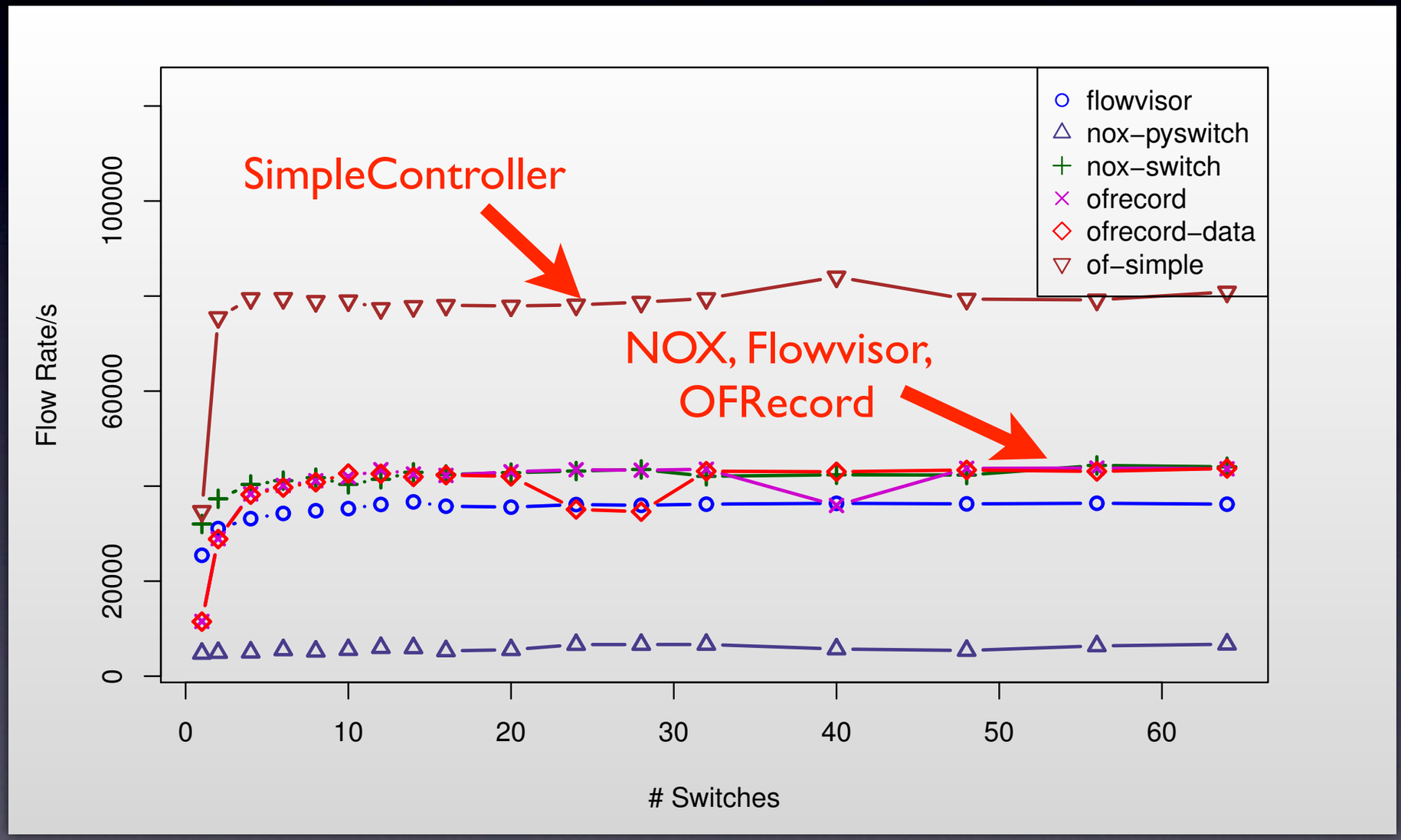
- Trigger: specific source MAC address

- NOX has an 'intelligent' MAC address parser that handles both binary and ASCII MAC addresses
- '0x3a' is the ASCII representation of ':' and appeared in the binary form of this MAC :)

Performance Evaluation

- **Record**: *production environment*
  - OFRecord controller performance
  - Impact on switch performance
- **Replay**: *lab environment*
  - Timing accuracy

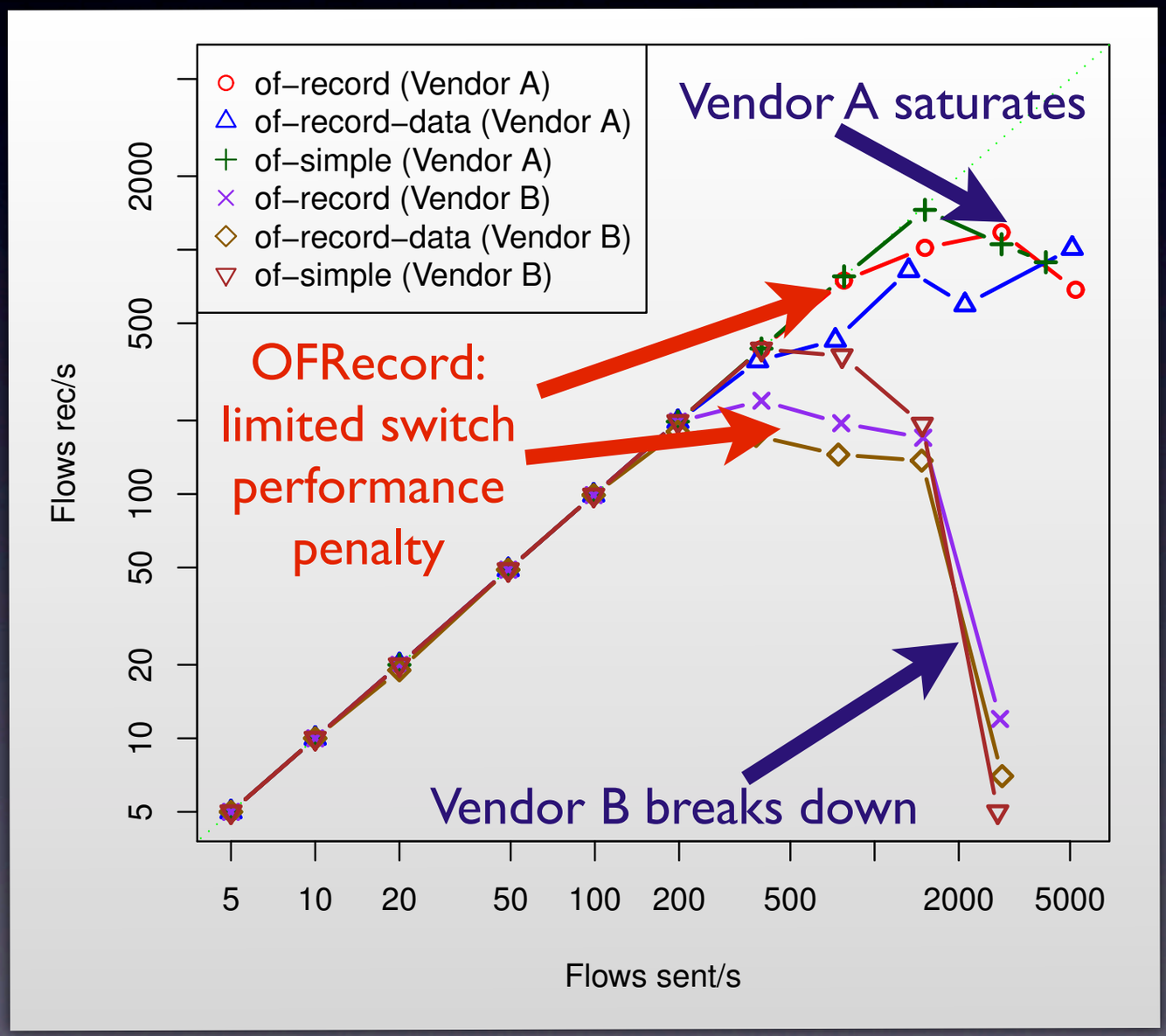
OFRecord controller performance



Median # Flows handled by different controllers (measured with cbench)

Impact on switch performance

- Single UDP packet flows created using hping
- sent to switches of two different vendors
- measure # flows successfully forwarded
- compare OFRecord vs. SimpleCtrl



End-to-end performance

Rate [Flows/s]	Drop %	sd (timing) [ms]
5	0	4.5
10	0	15.6
20	0	21,1
50	0	23,4
100	0	10,9
200	0	13,9
400	19 %	15,8

Summary

New Primitives:

Selective, consistent, multigranularity **Network Recording**

&

Adaptive coordinated best-effort **Network Replay**

- reproduce problems at convenient time and place
- Combined in **OfRewind,** an Open-Flow based tool for Network Record & Replay
  http://www.openflow.org/wk/index.php/OFRewind
- Enables practical record and replay of network domains

Future work

- Scale to larger topology sizes, more complex networks
- Extend to production quality tool
  - Improve timing for very fast flow rates
- Automated regression tests through standard sets of traces

Thank you.

Summary

New Primitives:

Selective, consistent, multigranularity **Network Recording**

&

Adaptive coordinated best-effort **Network Replay**

- **reproduce** problems at convenient time and location
- Combined in **OfRewind,** an Open-Flow based tool for Network, Record & Replay
- Enables practical record and replay of network domains
- http://www.openflow.org/wk/index.php/OFRewind