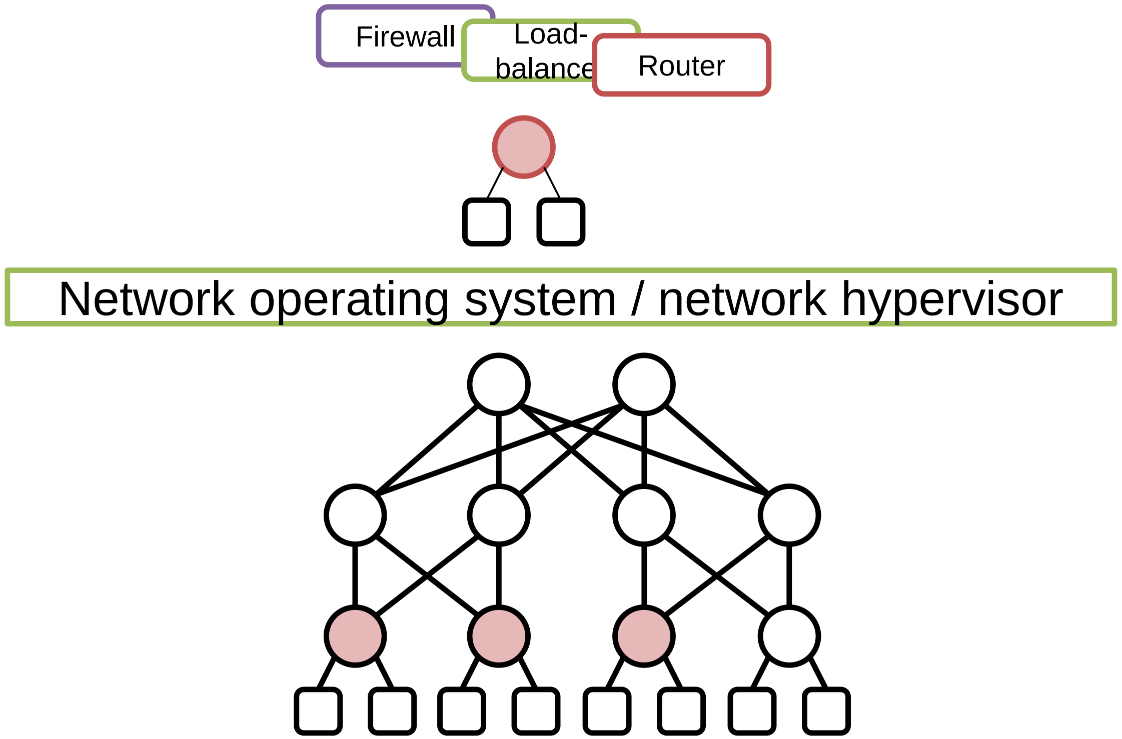
COCONUT: Seamless Scale-out of Network Elements

Soudeh Ghorbani
P. Brighten Godfrey

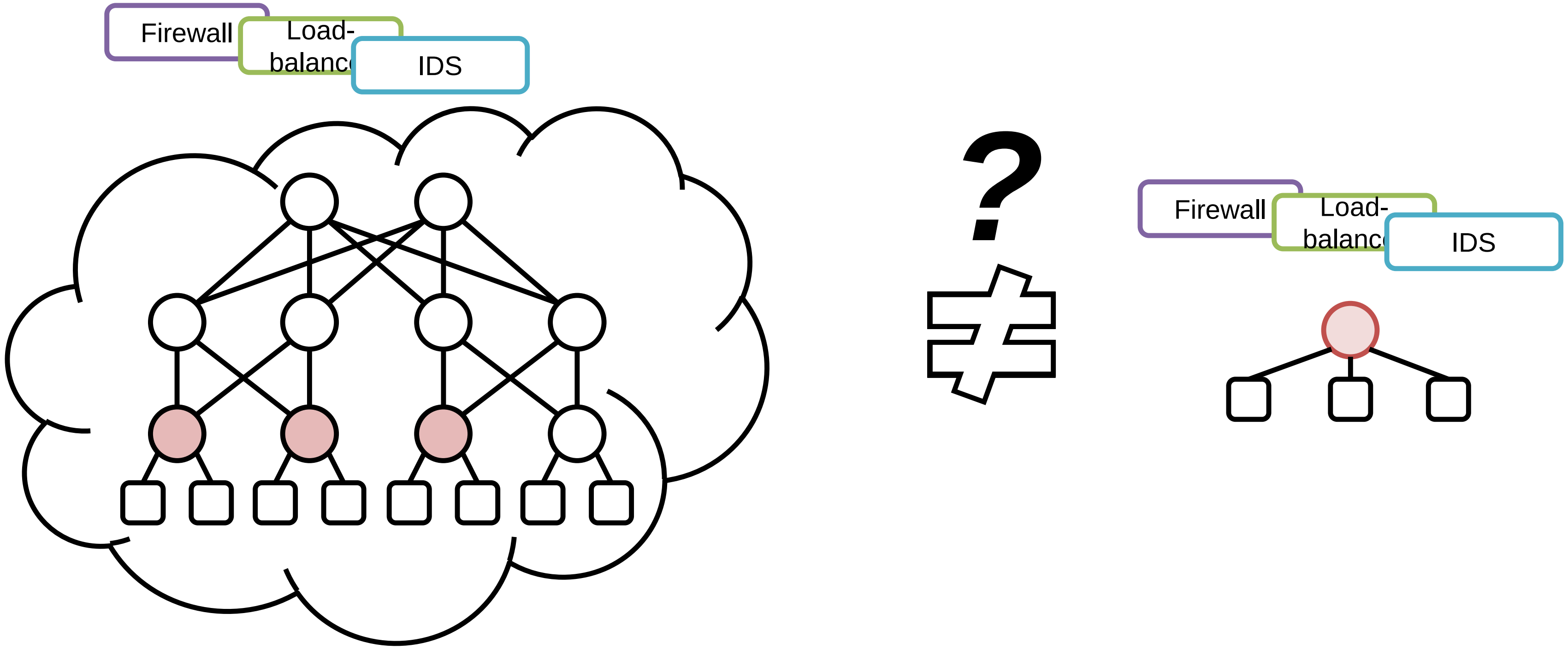
University of Illinois at Urbana-Champaign

Simple abstractions



Scale-out implementation

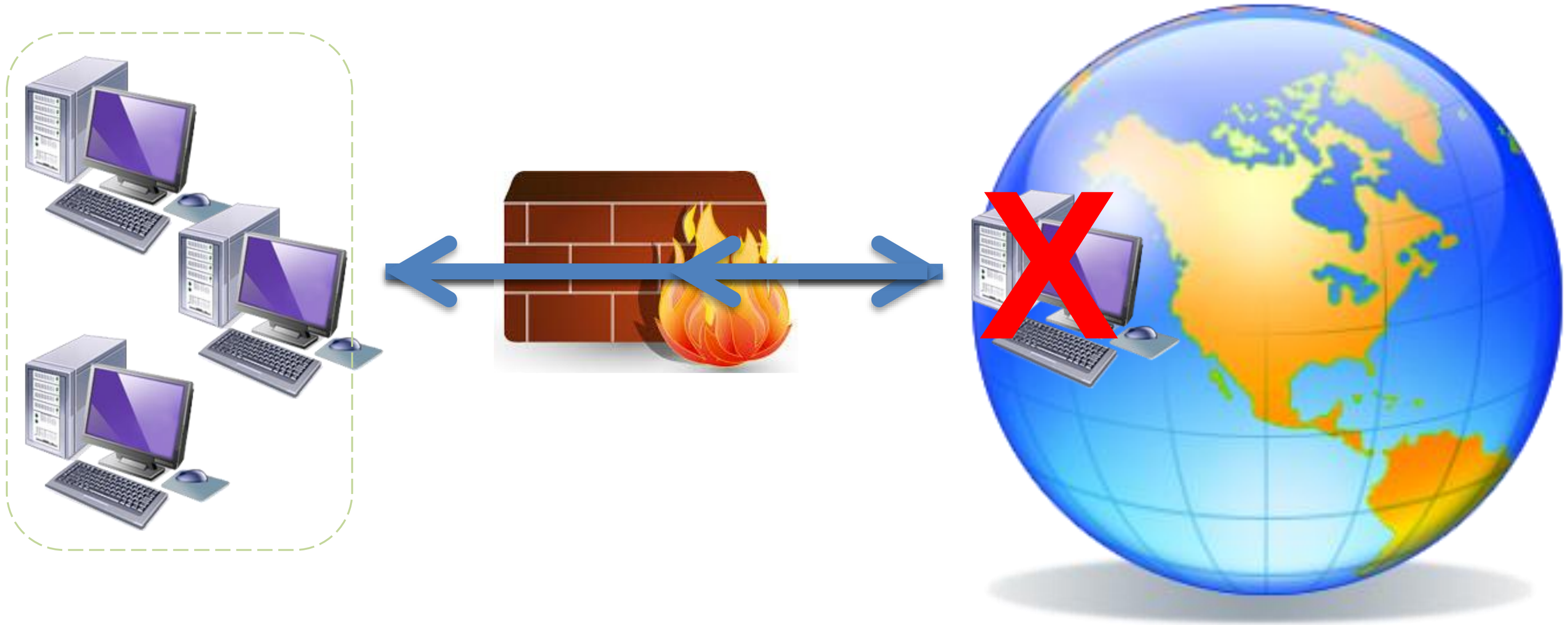
Is the implementation a faithful reproduction of the abstraction, i.e., are the scaling-out techniques *transparent*?



Scaling-out: What could go wrong?

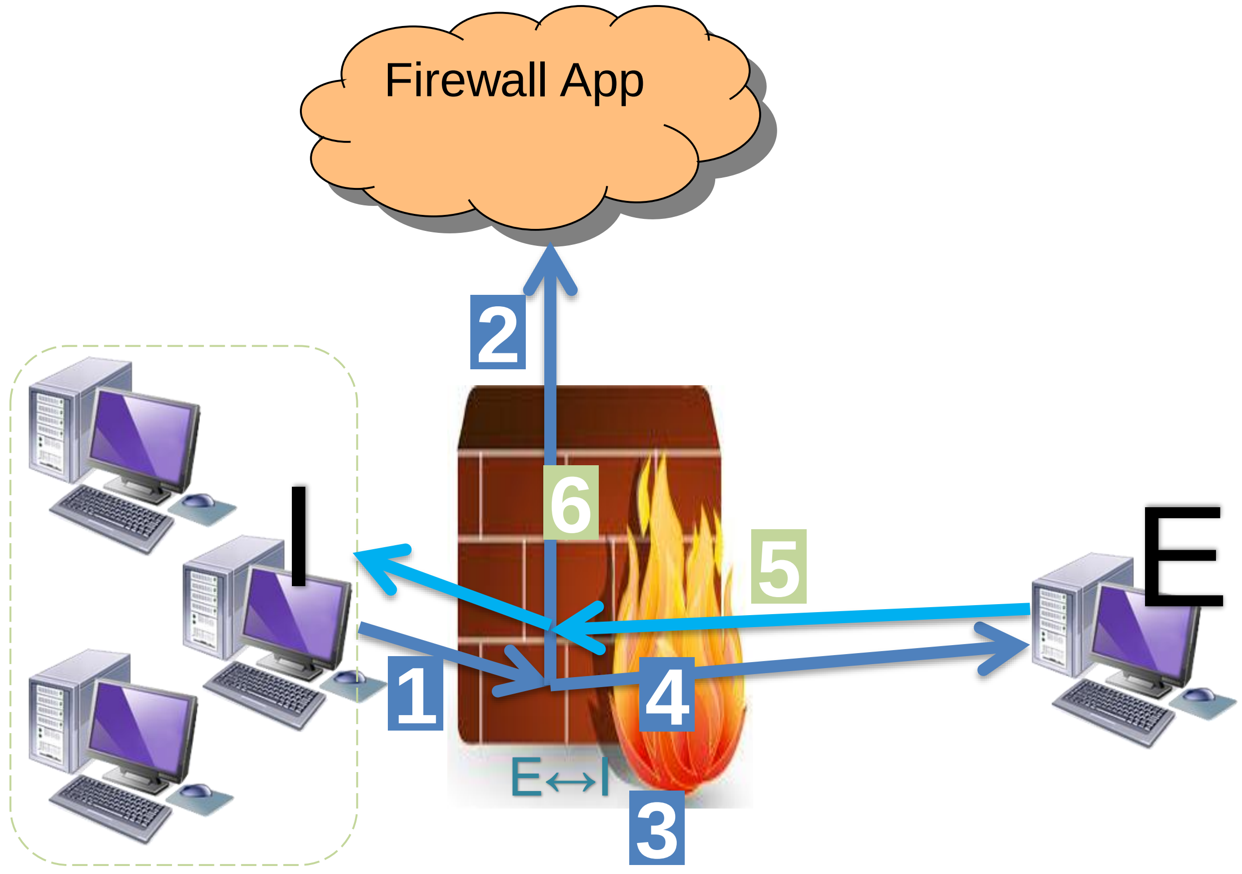
App	Incorrect behavior
Stateful firewall	Blacklisting the legitimate hosts
Load balancer	Dropping connections
IDS	Blacklisting the legitimate hosts

Logical firewall

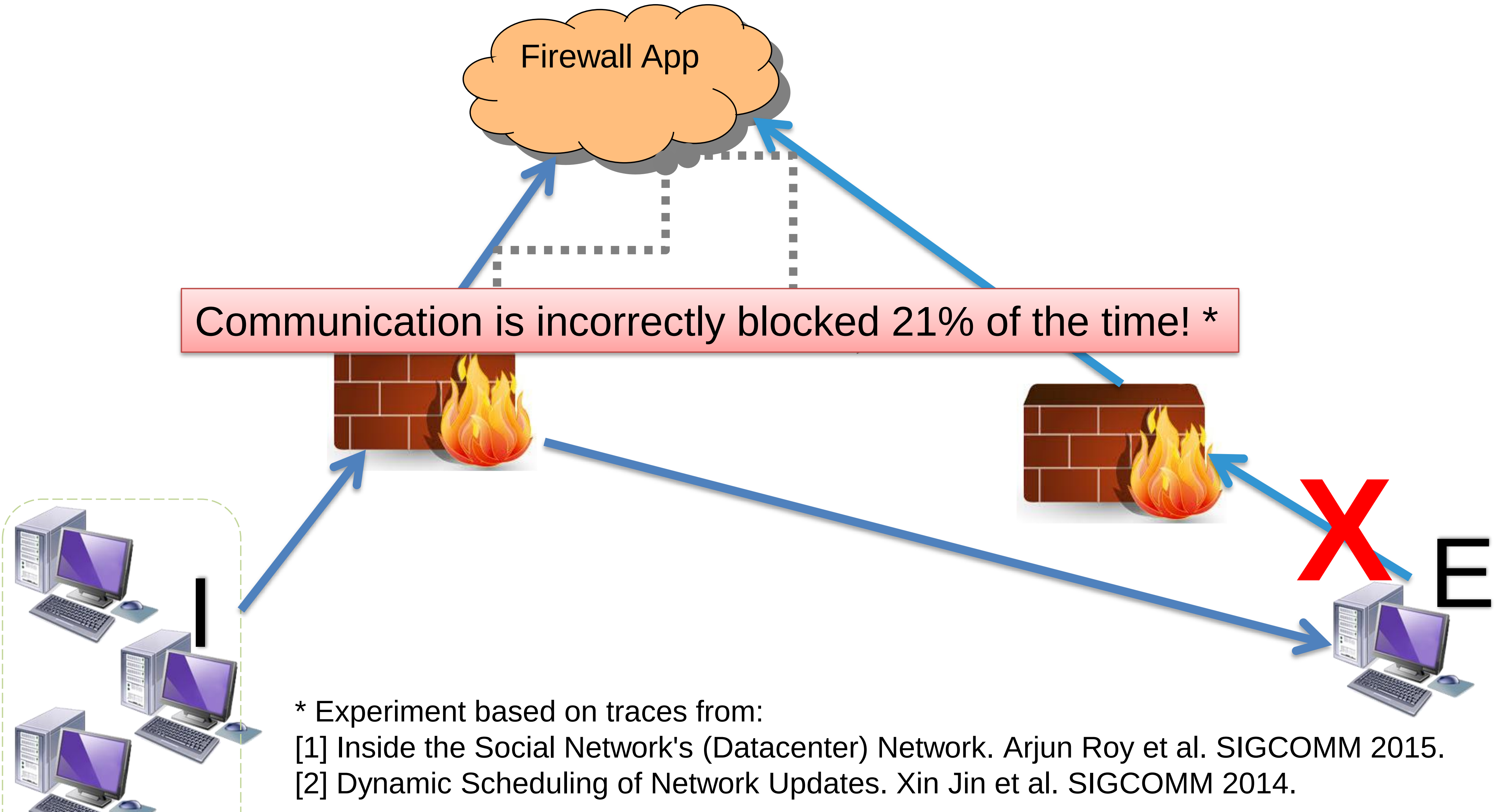


Policy: permit an external server to talk to an internal client if and only if the client has sent a request to the server.

Logical firewall



Logical firewall + scaling out = bug



* Experiment based on traces from:

[1] Inside the Social Network's (Datacenter) Network. Arjun Roy et al. SIGCOMM 2015.

[2] Dynamic Scheduling of Network Updates. Xin Jin et al. SIGCOMM 2014.

Do existing solutions work?

Per-packet consistency and correctness

- *Consistent Updates [Reitblatt et al., SIGCOMM 2012]*
- ***Too weak***: Focus on a single packet/flow.
- Solution requires single point of entry for traffic.
- Can even introduce cross-flow race conditions due to duplicated rules!

Strong consistency of replicas

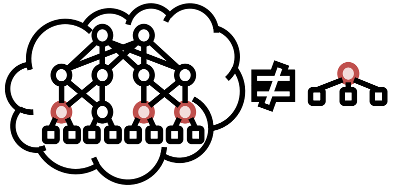
- *LIME [Ghorbani et al., SOCC'14], OpenNF [Gember-Jacobson et al., SIGCOMM'14]*
- Temporarily redirects all data plane traffic through controller during update.
- ***Too inefficient***: strong consistency would be cost prohibitive, e.g., latency can increase 10-100x!

Hard Choices: *Transparency* vs. *Efficiency*

☺ Most efficiency

☹ Least transparency

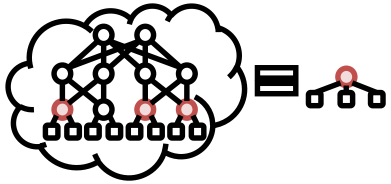
- ❑ Eventual consistency (possibly to an incorrect state)



☹ Least efficiency

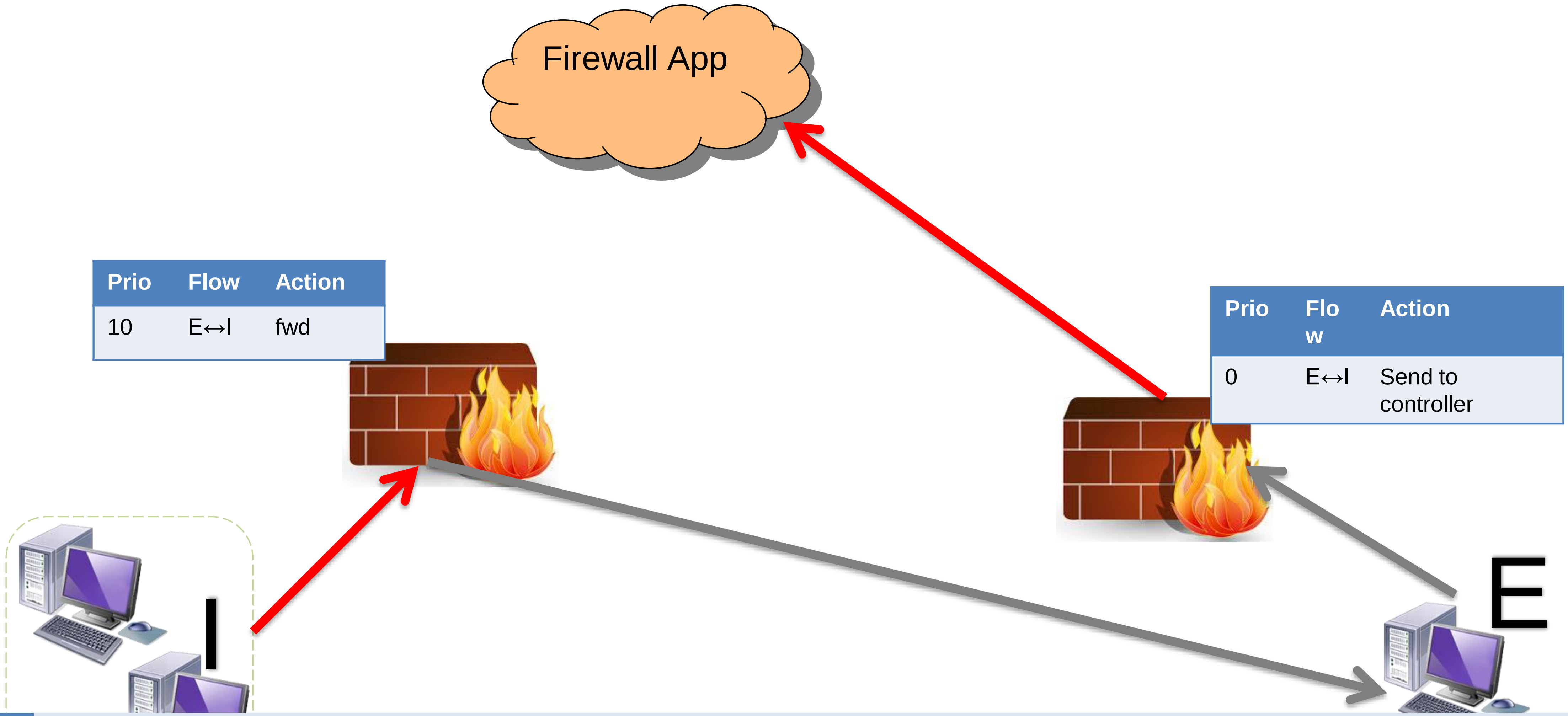
☺ Most transparency

- ❑ Serializability
- ❑ Linearizability
- ❑ Sequential consistency



Root-cause of the incorrect behavior

Logical firewall + scaling out = bug



Prio	Flow	Action
10	E↔I	fwd

Prio	Flo w	Action
0	E↔I	Send to controller

Root cause A packet is handled by a new network state and then triggers a sequence of events leading to other flows' packets being handled by an old state.

COCONUT in one slide

Logical clocks track network state (forwarding rules) versions and restrict the space of executions to those that are causally consistent.

1 Each *packet* carries a vector of logical clocks (VC) showing the latest versions of the rules applied on it or any packet before it.

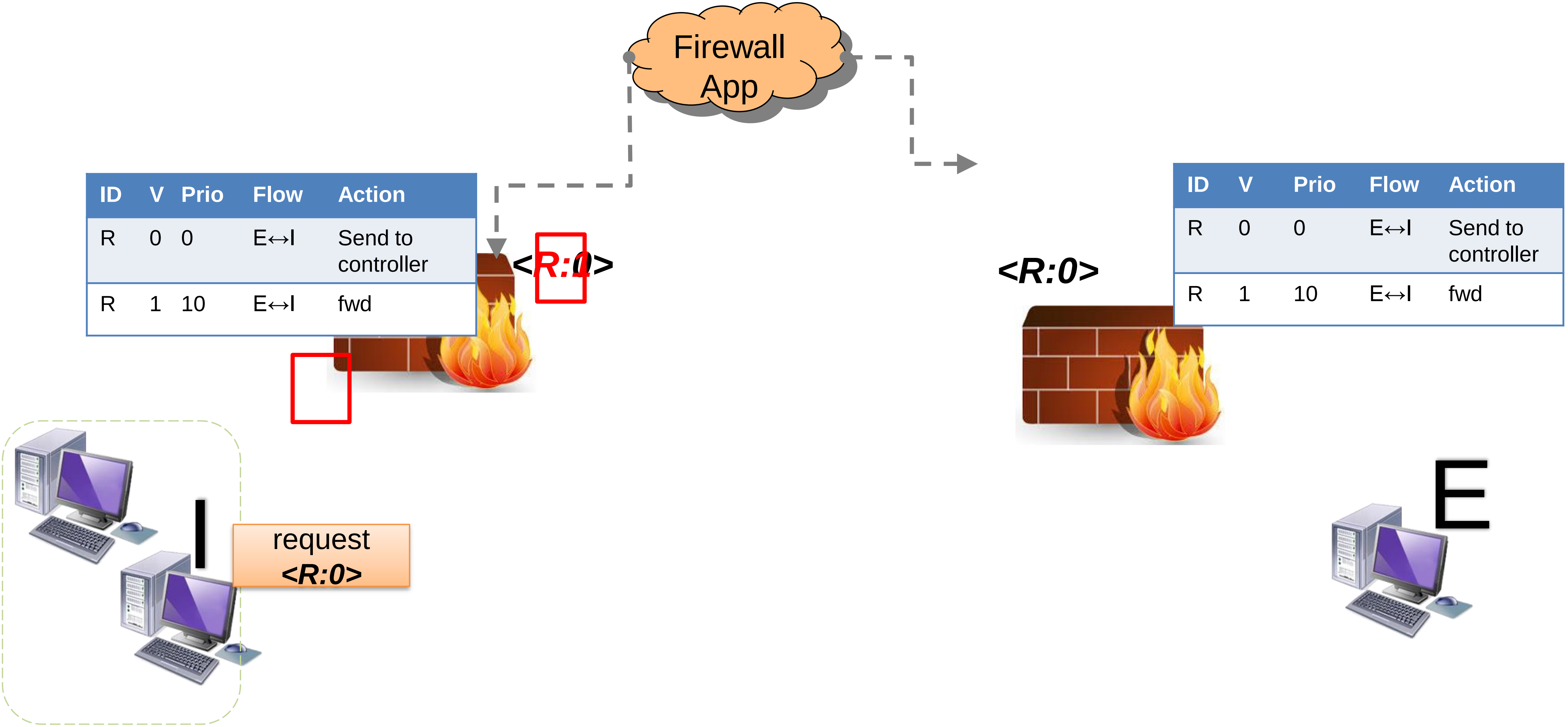
2 *Endpoints* keep VCs showing the latest version of rules they have observed on packets and affix their VCs to packets that they send out.

3 *Switches*

- Switches are preloaded with all versions of the rules.
- Switches keep a clock for each rule showing the local version of the active rule.
- Larger packet clock than the local logical clock of a matching rule prompts the switch to update the rule before applying it.
- Packet's VC is updated if necessary.

COCONUT in one slide

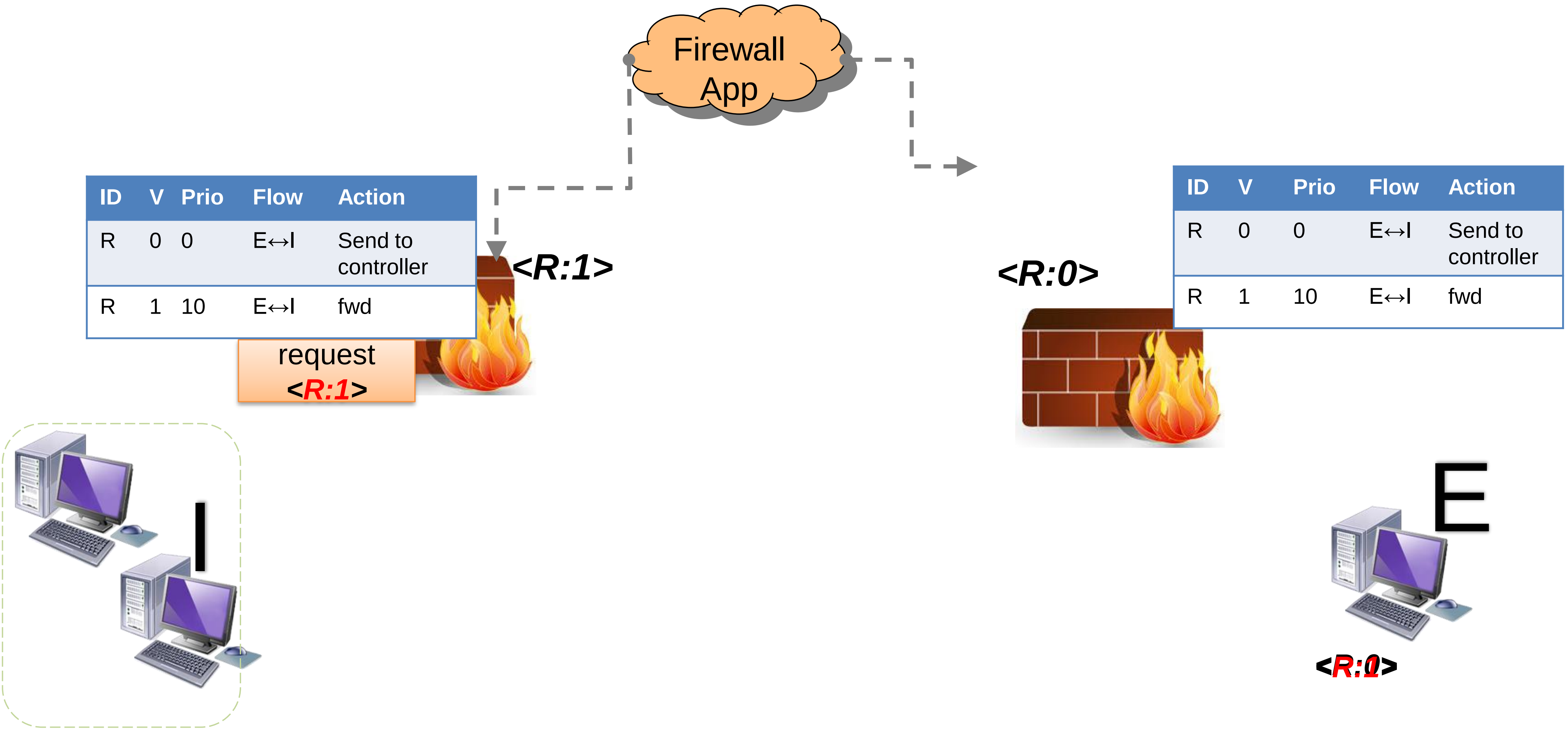
Logical clocks track network state (forwarding rules) versions and restrict the space of executions to those that are causally consistent.



ID	V	Prio	Flow	Action
R	0	0	E↔I	Send to controller
R	1	10	E↔I	fwd

ID	V	Prio	Flow	Action
R	0	0	E↔I	Send to controller
R	1	10	E↔I	fwd

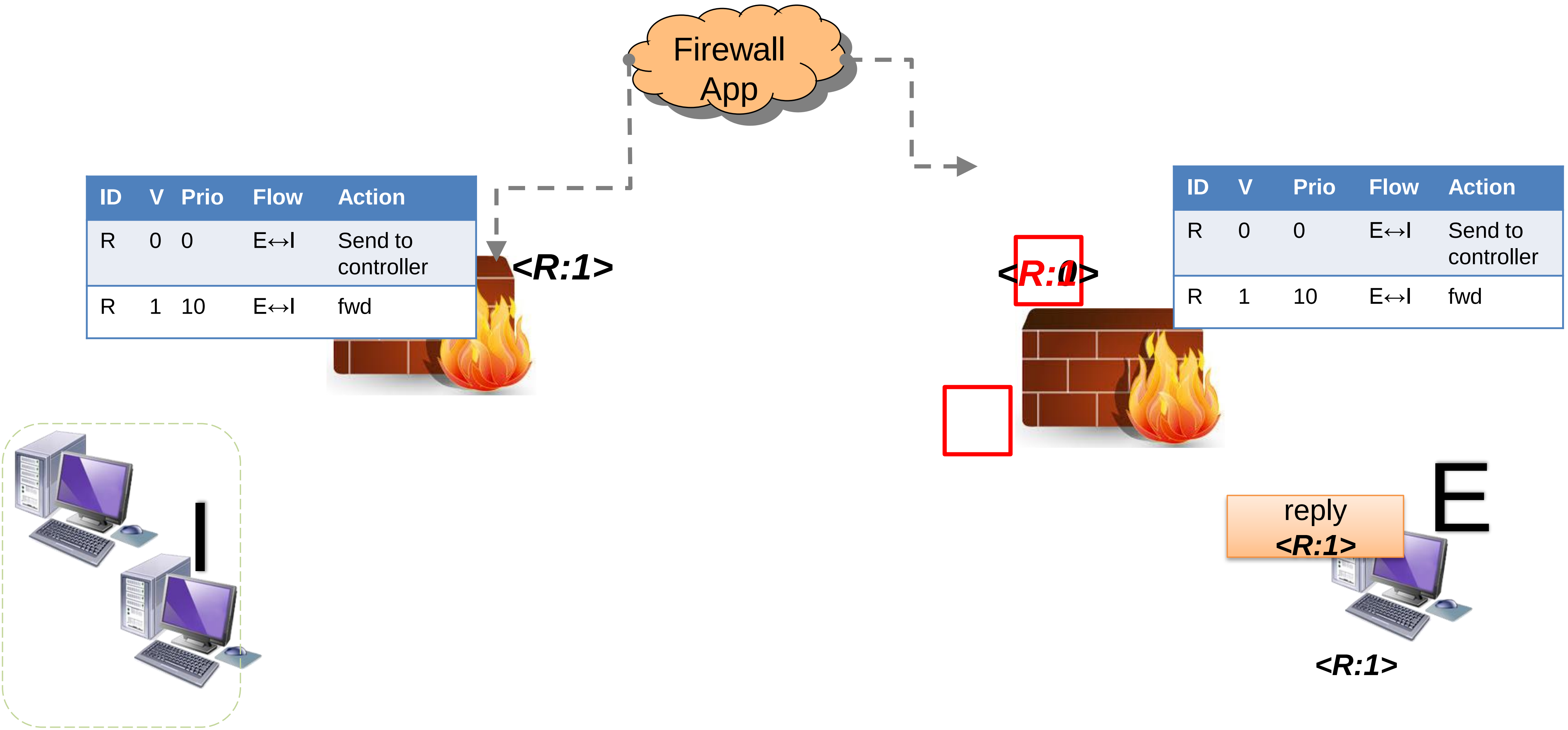
COCONUT in one slide



ID	V	Prio	Flow	Action
R	0	0	E↔I	Send to controller
R	1	10	E↔I	fwd

ID	V	Prio	Flow	Action
R	0	0	E↔I	Send to controller
R	1	10	E↔I	fwd

COCONUT in one slide



ID	V	Prio	Flow	Action
R	0	0	E↔I	Send to controller
R	1	10	E↔I	fwd

ID	V	Prio	Flow	Action
R	0	0	E↔I	Send to controller
R	1	10	E↔I	fwd

Simple but ***impractical***

Impractical for packet to ***carry clocks*** for all rules

- Datacenters with ~100K servers have millions of rules [1].
- Switches and end-hosts cannot perform vector operations on such big vectors for each packet (especially not with integer versions).

Impractical to ***preload switches*** with all rules

- Preloading when the rule is created adds more delay than we want.

[1] Scalable Rule Management for Data Centers. Masoud Moshref et al. NSDI’13.

Down to Earth:
COCONUT within OpenFlow

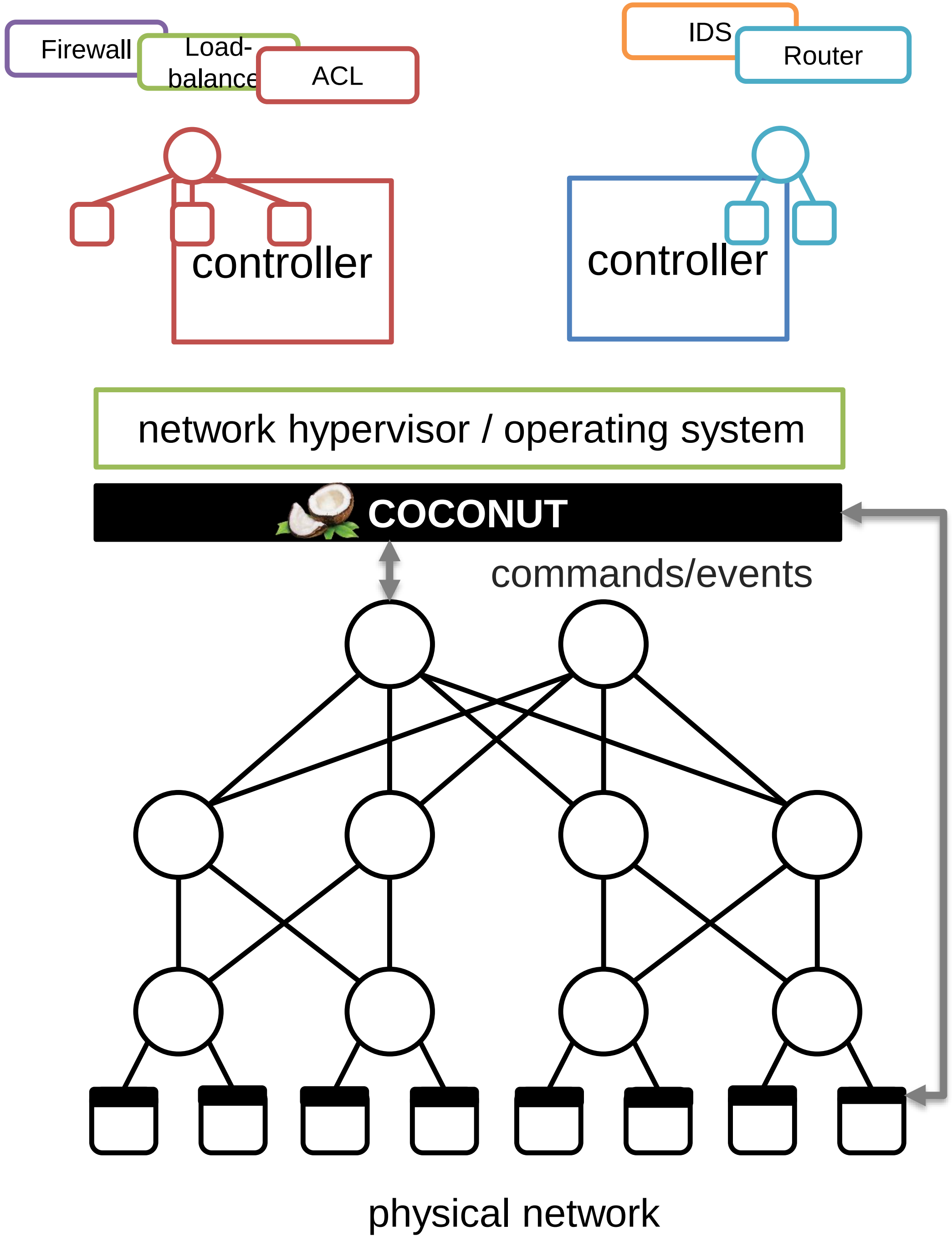
Only concurrent updates require distinct versions

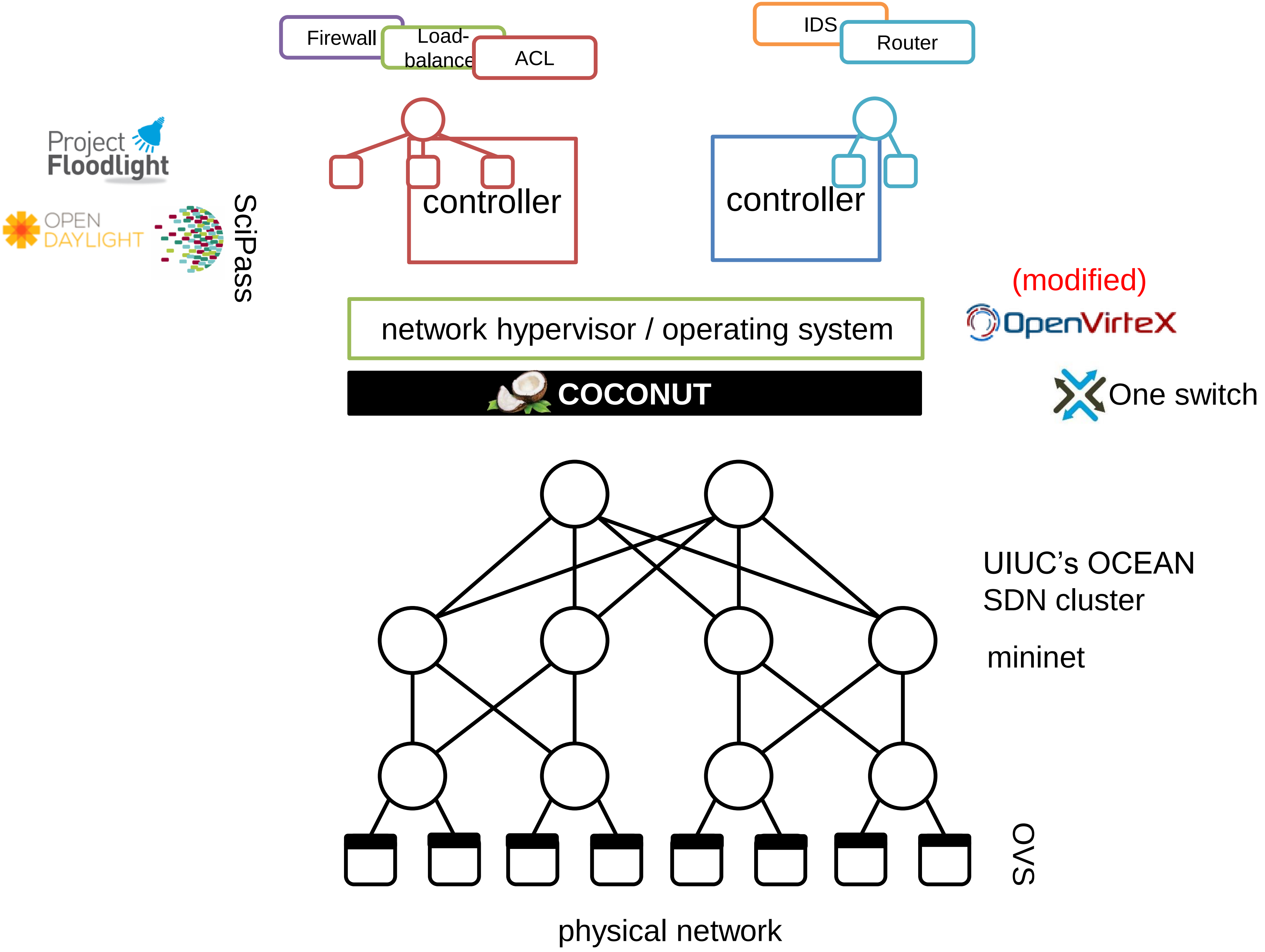
- Large scale networks are updated a few hundred to a few thousand times a day [1].
- Updates with COCONUT take <1s, so a few vector entries are enough.
- Each version can be a single bit (old vs. new), thereafter reused.

Don’t guarantee that rules are preloaded

- Unlucky race condition ⇒ Versioning catches it & sends to controller.

[1] "Ananta: cloud scale load balancing." Patel, Parveen, et al. *SIGCOMM 2013.*

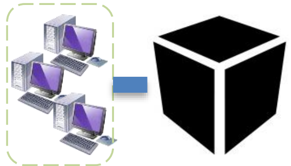

Firewall
Load-
balance
ACL
IDS
Router
controller
controller
network hypervisor / operating system
COCONUT
commands/events
physical network



Firewall
Load-balance
ACL
IDS
Router
Project Floodlight
OPEN DAYLIGHT
SciPass
controller
controller
(modified)
OpenVirteX
network hypervisor / operating system
COCONUT
One switch
UIUC's OCEAN SDN cluster
mininet
OVS
physical network

Is this enough for transparency?

COCONUT guarantees transparency



- A rigorous notion of ***behavior***.
  - *Trace*: sequence of ***externally-visible actions.***
  - ***Behavior***: set of all plausible traces.

- **Transparency:** The possible behavior of P is a subset of the possible behavior of a non-replicated implementation of L.

- Theorem: Coconut provides transparency, i.e., ***any behavior of COCONUT's implementation of replicated networks could have happened in the logical network.***

Experimental Evaluation

Metrics

- Transparency
- Update delay
- Rule overhead
  - How much?
  - Where?
  - For how long?

Topologies

- VL2
- Fat-tree

Workload

"Inside the social network's (datacenter) network." Roy, Arjun, et al. *SIGCOMM* 2015

Schemes

- Strong Consistency (SC) [1]
- Simple Replication (SR)
- COCONUT

[1] "Transparent, live migration of a software-defined network." Ghorbani, Soudeh, et al. SoCC 2014.

With COCONUT:

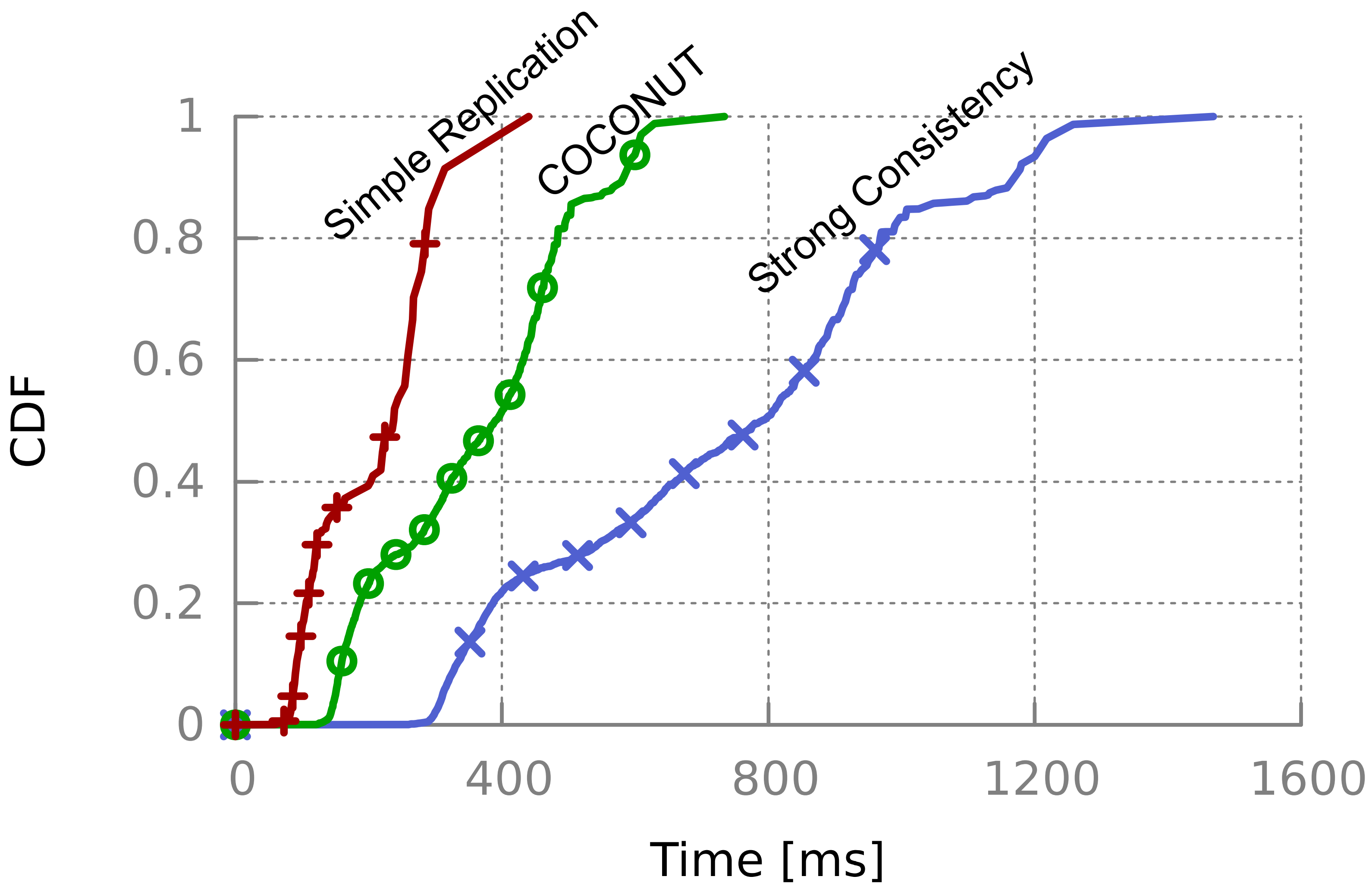
- ✓ **Transparency**

- ✓ **No data plane performance overhead**.
  - Up to **20x** increase in latency with SC
  - Up to **12Gbps** bandwidth overhead with SC

- ✓ **Modest update delay**
  - **1.2x** higher than SR
  - **3.5x** lower than SC

- ✓ **Modest rule overhead**
  - **1.6x** higher than SR
  - **2x** lower than SC

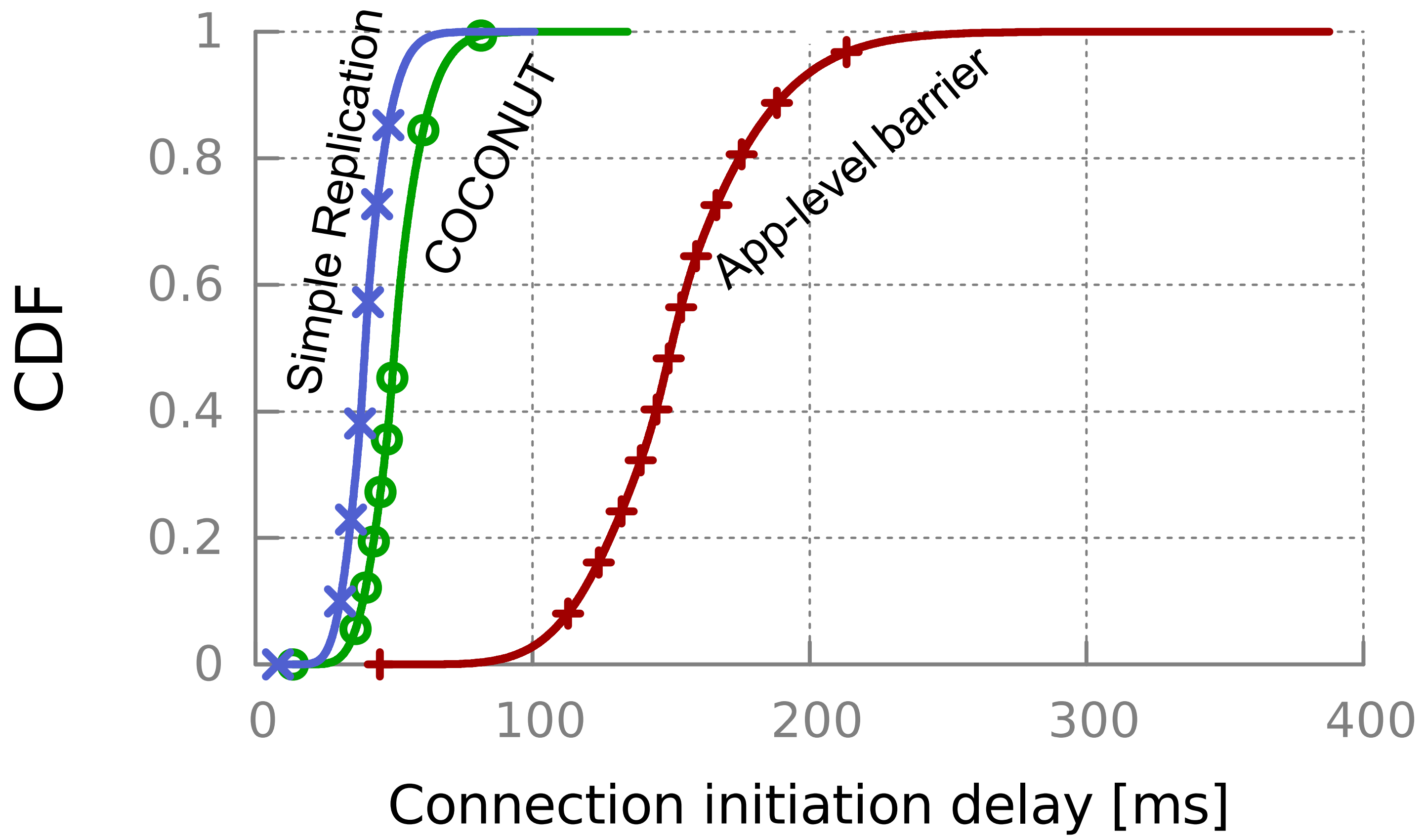
Faster updates than Strong Consistency



Experiment:

- Hardware testbed
- 20 switch fat tree topology
- 16 hosts
- IDS application

Faster updates than application-level barrier solution



Experiment:

- Mininet
- Tree with 20 switchs at leaf
- 100 hosts
- IDS application
- Switch update delay distribution drawn from measurements of HP ProCurve switches

Conclusion:

1. We identified the **problem:** incorrect application-level behavior under the existing techniques for scaling-out.

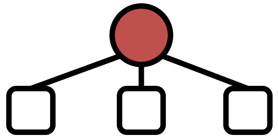
2. We identified its **root cause:** causality violation.

3. We developed an analytical **framework** to reason about the problem.

4. We developed a algorithms and a system, **COCONUT,** to efficiently scale out networks transparently.

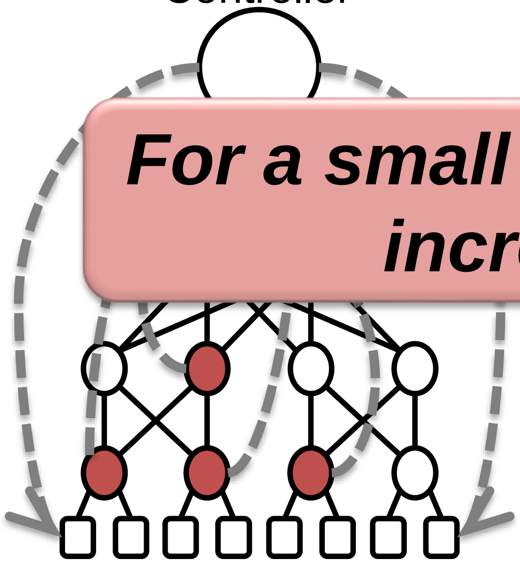
Backup

Strong Consistency (SC) via centralization



One big switc

Controller



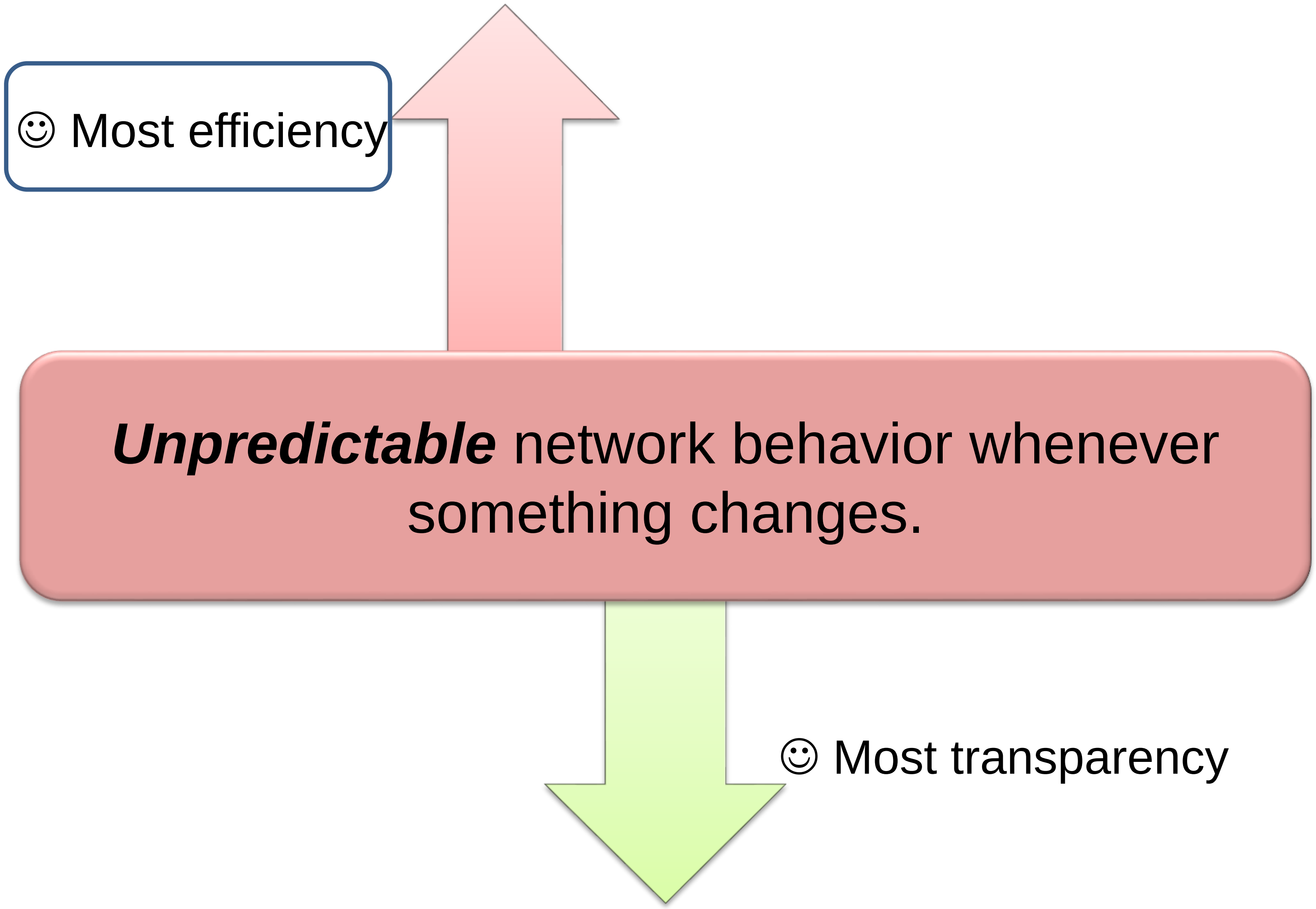
Scaled out physical network

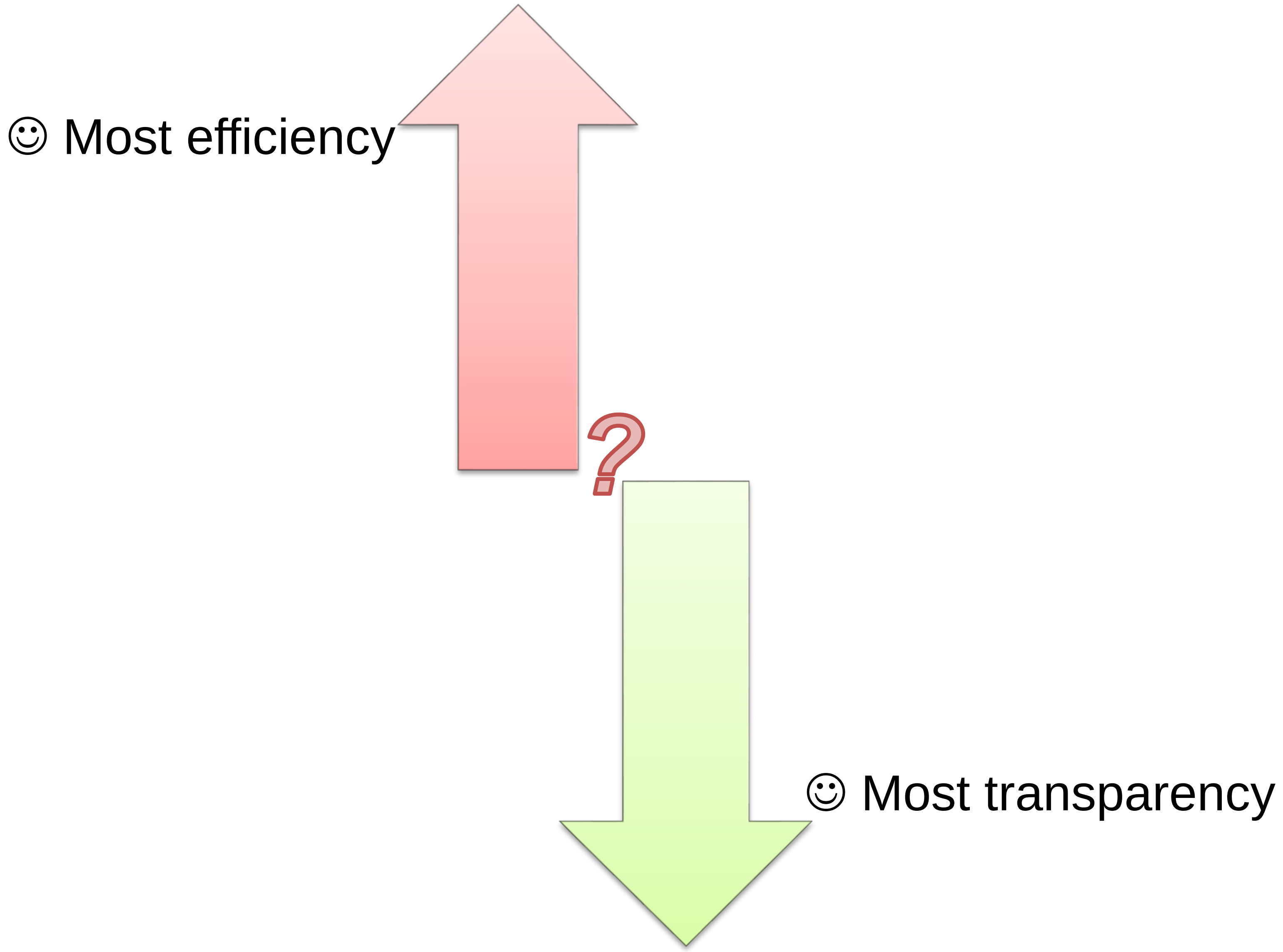
It doesn’t scale.

For a small network, latency can increase 10-100x!

- “Transparent live migration of a ... network.” ...i, Cole ... thew Monaco, ... hew Caesar, Jennifer Rexford, David Walker, SoCC 2014.

- ... Prakash, Robert Grandl, Junaid Khalid, Sourav Das, and Aditya Akella, SIGCOMM 2014

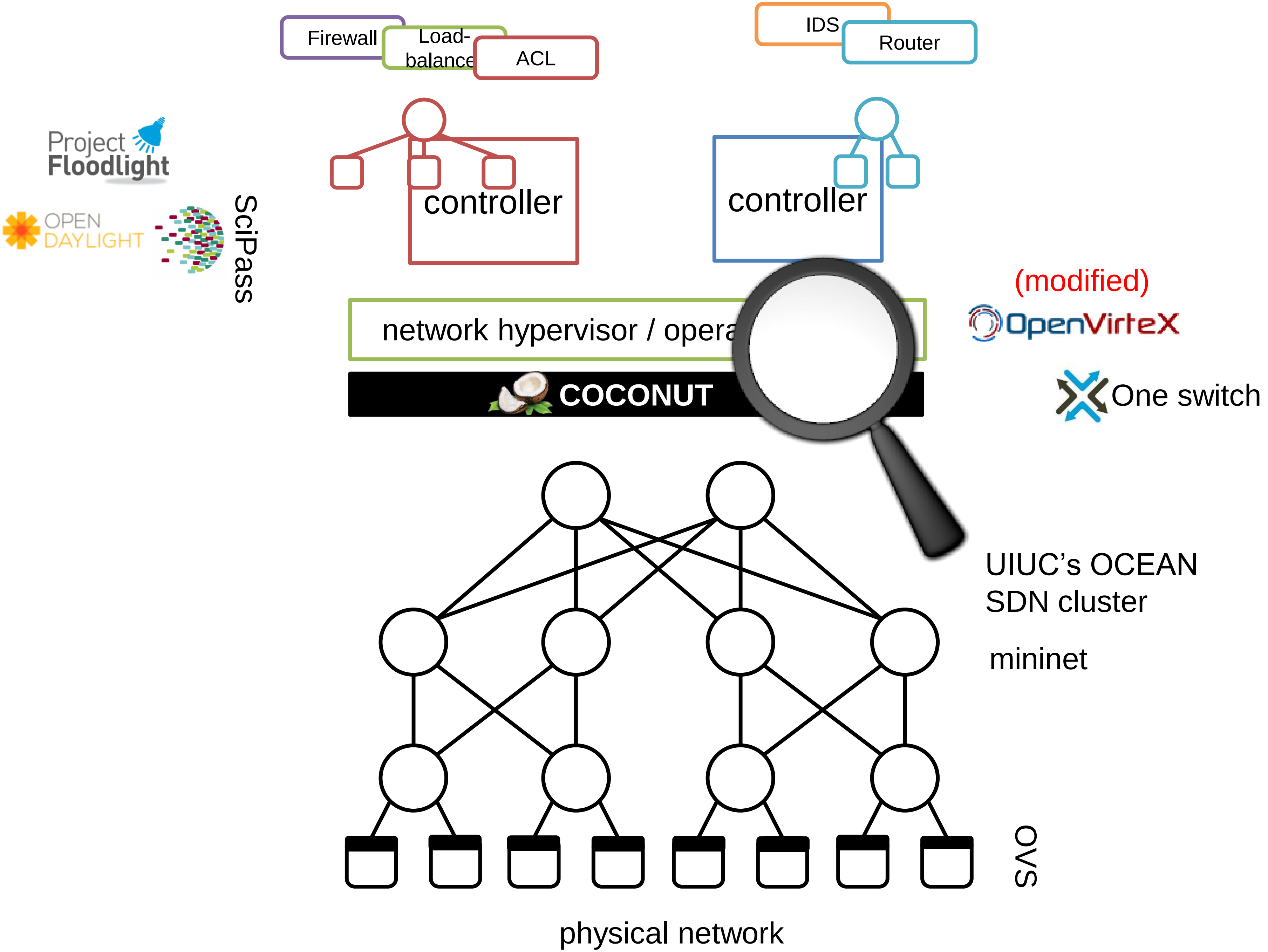

☺ Most efficiency
Unpredictable network behavior whenever something changes.
☺ Most transparency


☺ Most efficiency
?
☺ Most transparency

Change is a nightmare.

- A survey of network operators shows [1]:
  - *89%* are *never* completely certain that changes will not introduce a new bug.
  - *82%* are concerned that changes might *break existing functionality unrelated to the changes*.
- A big problem given the frequency and criticality of change:
  - 100s to 1000s of changes per day [2,3].
  - Majority of them are critical, e.g., related to fixing security issues [3].

[1] “Tomorrow’s network operators will be programmers.” Nick Feamster. *OOPSALA keynote 2015.*
[2] "Ananta: cloud scale load balancing." Parveen Patal et al. *SIGCOMM* 2013.
[3] “Networking Options and Challenges in a Multi-Data Center and Multi-Cloud Provider Environment.” Mark Bluhm. *FutureNet 2016.*


Firewall
Load-balance
ACL
IDS
Router
Project Floodlight
OPEN DAYLIGHT
SciPass
controller
controller
(modified)
OpenVirteX
network hypervisor / opera
COCONUT
One switch
UIUC's OCEAN SDN cluster
mininet
OVS
physical network