

1

2

Network Reliability in the Software Era

Finding Bugs in OpenFlow-based Software Defined Networks

Marco Canini

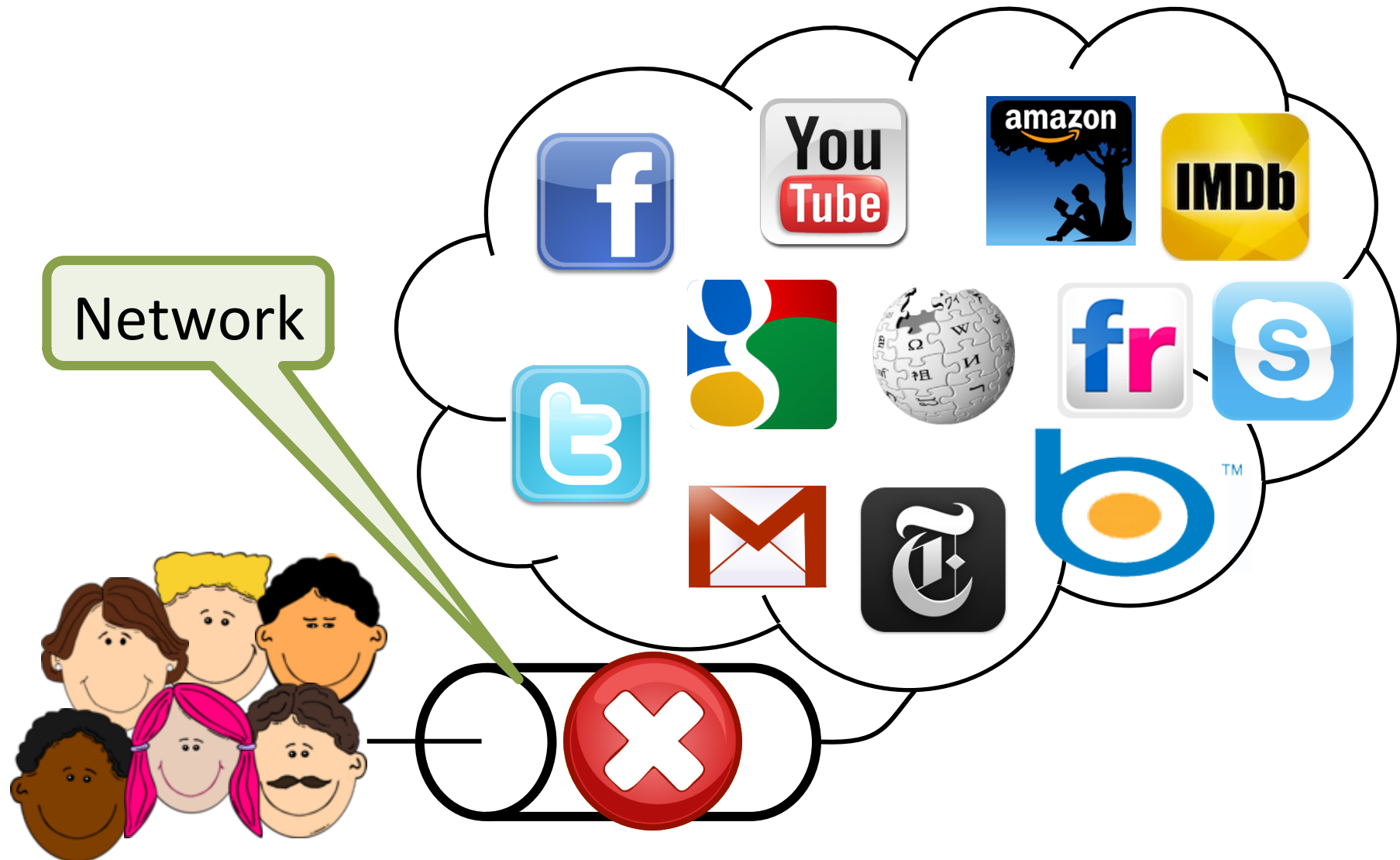
TU Berlin / T-Labs

20 Nov '12

3

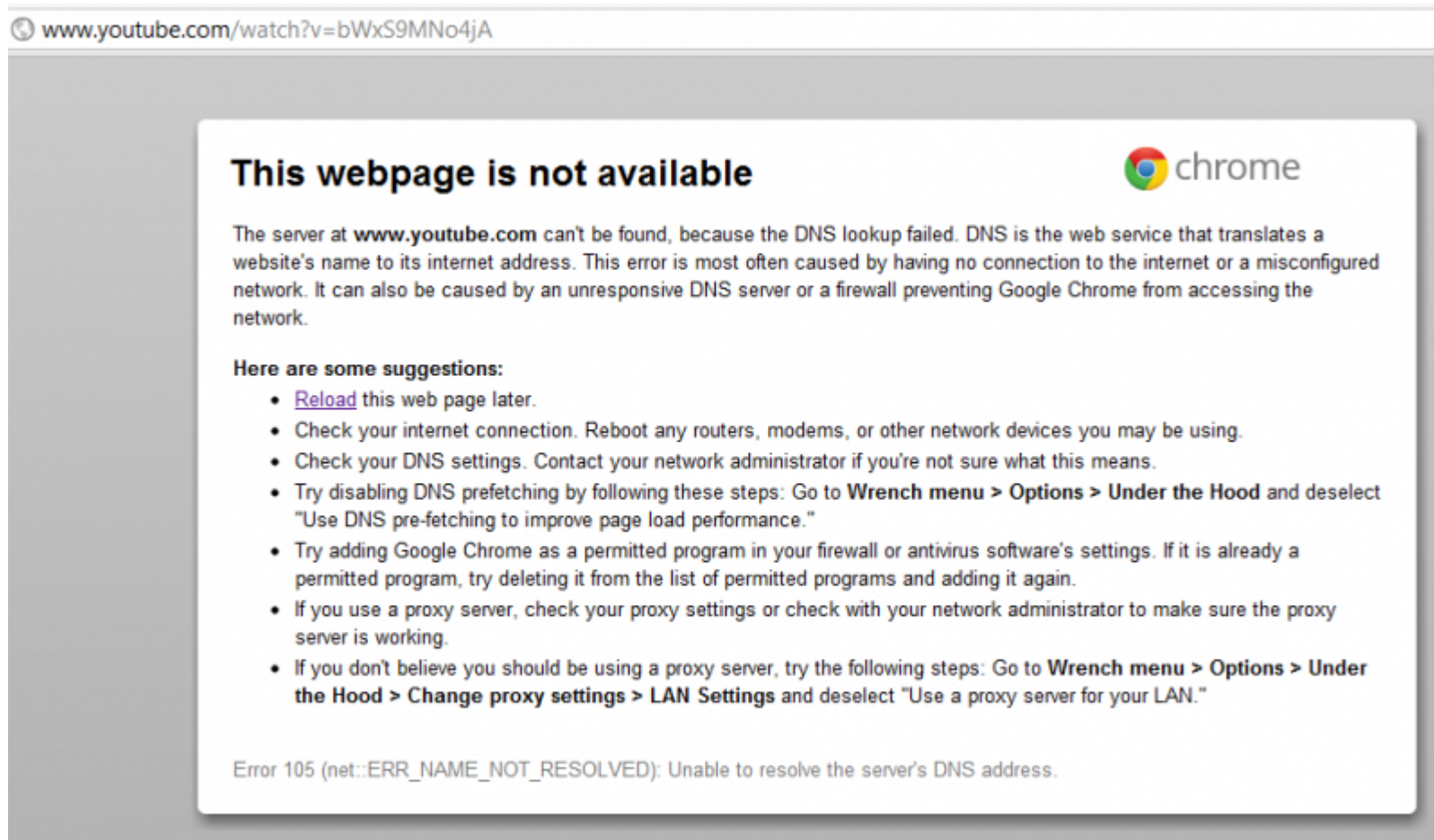
4

We Live in a Connected World




When Users Notice the Network

Like electricity, we assume it is magically always there



The screenshot shows a Chrome browser window with the address bar displaying `www.youtube.com/watch?v=bWxS9MNo4jA`. The main content area is a white error message box with a grey border. The title of the error is "This webpage is not available" in bold black text, followed by the Chrome logo and the word "chrome". The error message explains that the server at `www.youtube.com` cannot be found due to a DNS lookup failure. It provides a list of suggestions for troubleshooting, including reloading the page, checking internet connection, DNS settings, proxy server settings, and firewall settings. At the bottom of the error box, it states "Error 105 (net::ERR_NAME_NOT_RESOLVED): Unable to resolve the server's DNS address."

This webpage is not available 

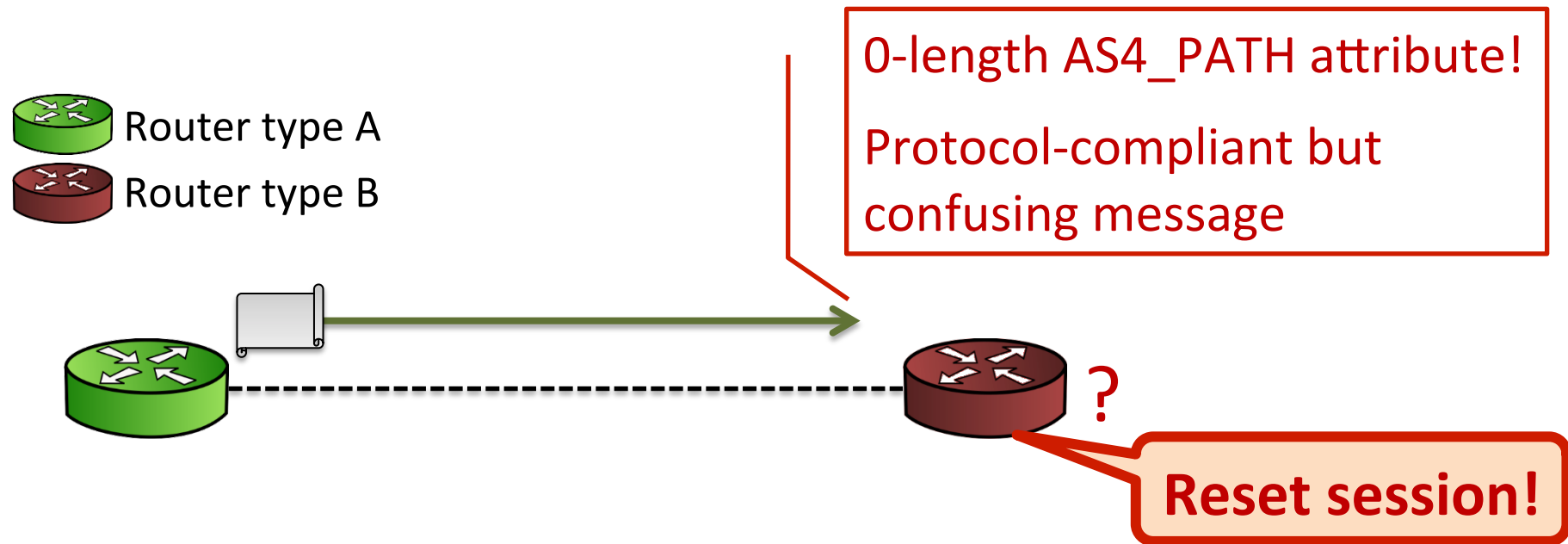
The server at `www.youtube.com` can't be found, because the DNS lookup failed. DNS is the web service that translates a website's name to its internet address. This error is most often caused by having no connection to the internet or a misconfigured network. It can also be caused by an unresponsive DNS server or a firewall preventing Google Chrome from accessing the network.

Here are some suggestions:

- [Reload](#) this web page later.
- Check your internet connection. Reboot any routers, modems, or other network devices you may be using.
- Check your DNS settings. Contact your network administrator if you're not sure what this means.
- Try disabling DNS prefetching by following these steps: Go to **Wrench menu > Options > Under the Hood** and deselect "Use DNS pre-fetching to improve page load performance."
- Try adding Google Chrome as a permitted program in your firewall or antivirus software's settings. If it is already a permitted program, try deleting it from the list of permitted programs and adding it again.
- If you use a proxy server, check your proxy settings or check with your network administrator to make sure the proxy server is working.
- If you don't believe you should be using a proxy server, try the following steps: Go to **Wrench menu > Options > Under the Hood > Change proxy settings > LAN Settings** and deselect "Use a proxy server for your LAN."

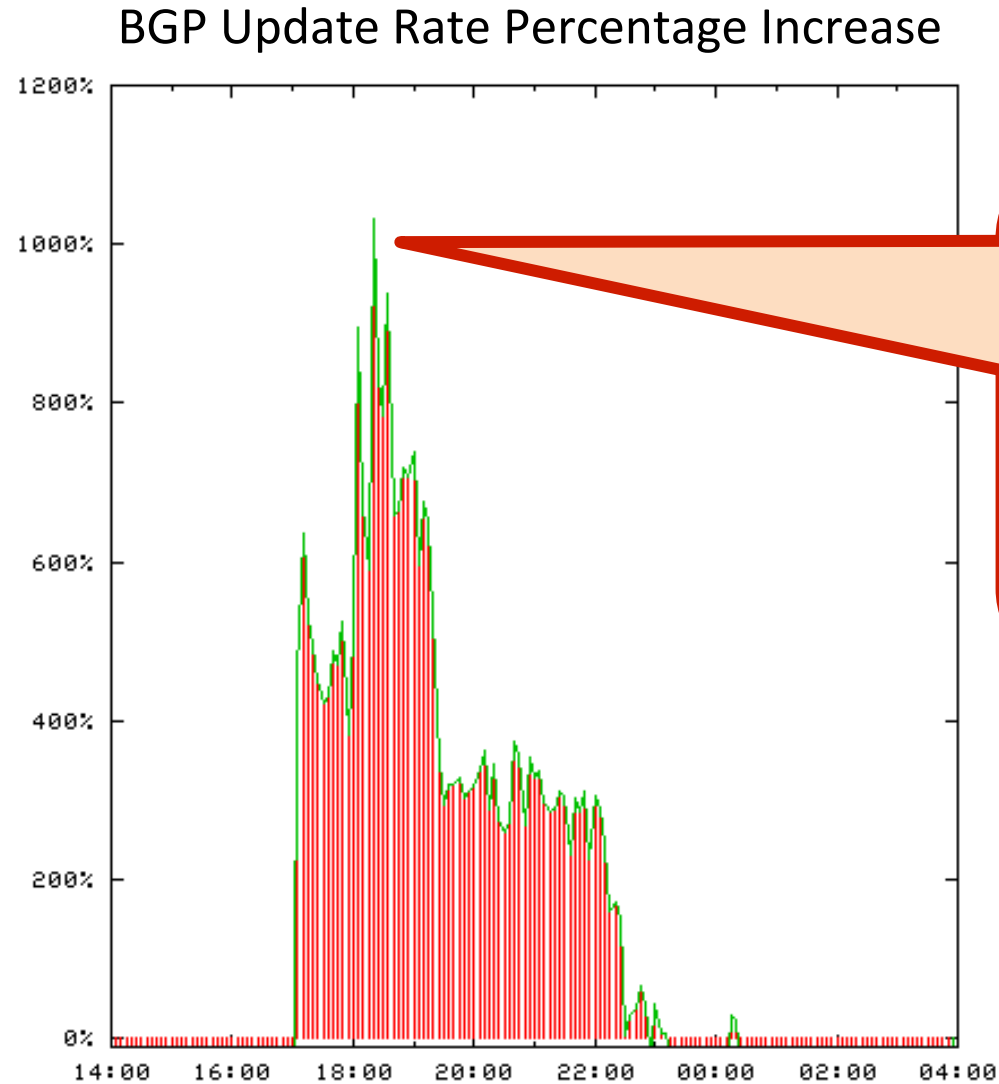
Error 105 (net::ERR_NAME_NOT_RESOLVED): Unable to resolve the server's DNS address.

Network Failure Example 1: Software Bugs in Inter-Domain Routers



On 19th August 2009, CNCI (AS9354),
a small ISP in Japan, **advertised a handful of
BGP updates containing an empty AS4_PATH attribute**

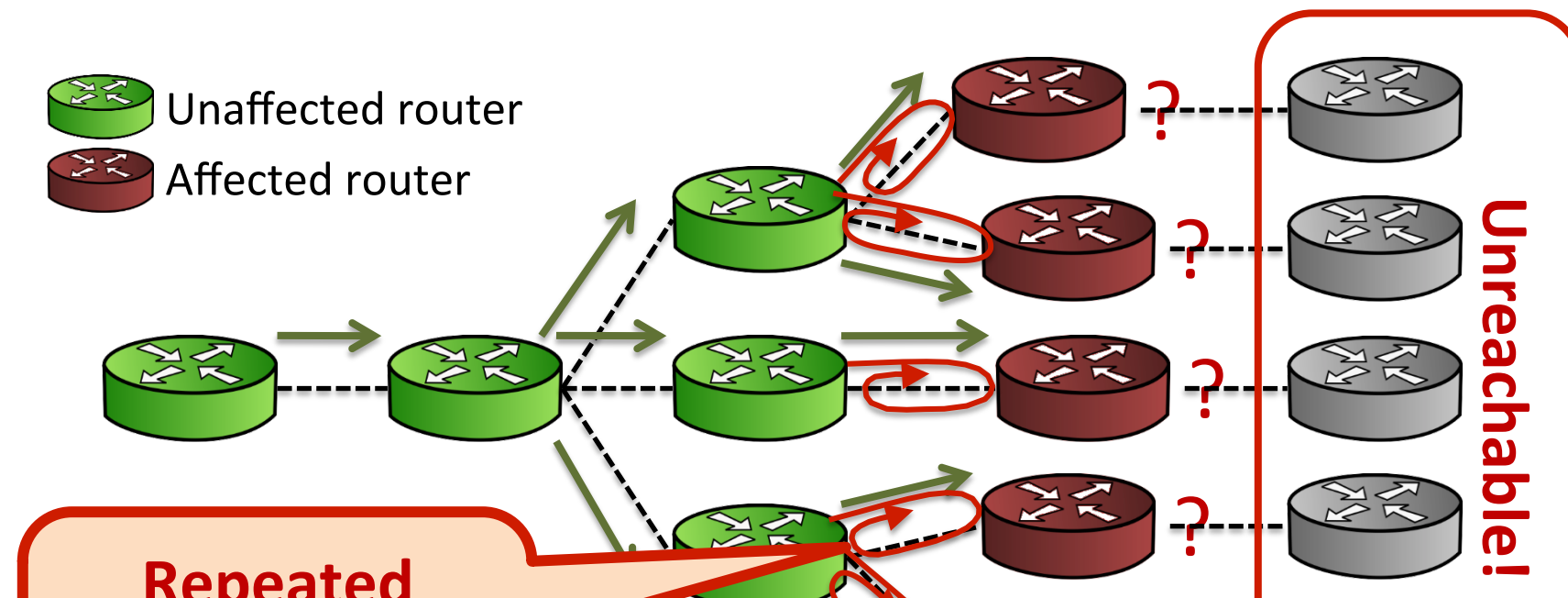
...what could possibly go wrong?



**10x
increase
→
routing
instabilities**

[renesys]

What Went Wrong: (CISCO) Session Reset Flood



Innocuous software fault caused Internet-wide outage

Network Failure Example 2: Planned Network Maintenance

- Amazon EC2 disruption on 21st April 2011
 - Incorrectly executed network change during a planned network capacity upgrade

foursquare

Sorry! We're having technical difficulties

Latest post from status.foursquare.com:

Quora

A continually improving collection of questions and answers created, edited, and organized by everyone who uses it.

We're currently having an unexpected outage, and are working to get the site back up as soon as possible. Thanks for your patience.

We are having troubles

Oh no! It seems much of the Internet is struggling. Our upstream providers are working to resolve the connectivity issues. We appreciate your patience.

In the meantime, if you can't wait to send a Tweet or Facebook post, please let us know how we can help.

Amazon is currently experiencing a degradation. They are **working on it.**

reddit is down.

Misconfiguration caused catastrophic outage

Software- and config-related issues

Affect even well tested, standard
Internet technology

With more software in networks,
need ways to deal with reliability issues

Why is network reliability so difficult to achieve?

Networks are Hard to Manage

New control requirements led to great complexity

- Network virtualization, VM migration, perf. isolation, ...

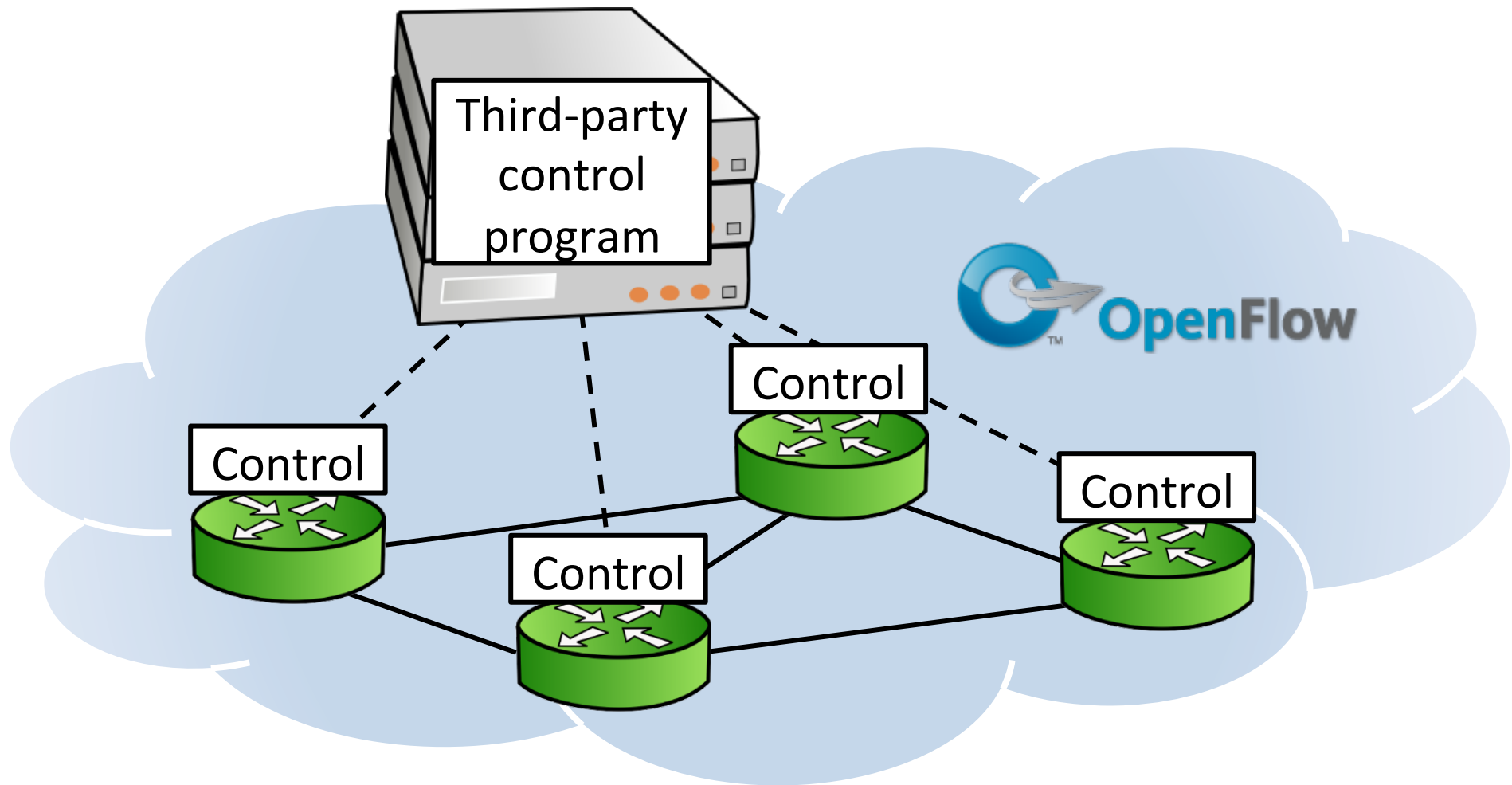
Kept working by “**Masters of Complexity**”

When things don't work?

- Only **limited tools**:

ping, traceroute, tcpdump, SNMP, NetFlow

Software-Defined Networking (SDN)



SDN Promises

Advantages over status quo of management

Reduce complexity

New functionality through programmability

SDN is great, but ...

... at the risk of bugs



Network Operating System

A fatal exception has occurred at 10.3.0.5/C0011E36 in OF(01) + 00010E36. The current OpenFlow application will be terminated.

- * Press any key to terminate the current OpenFlow application
- * Press CTRL+ALT+DEL again to restart your network. Your users will lose all network connectivity.

Press any key to continue

Software Faults



- Will make communication unreliable



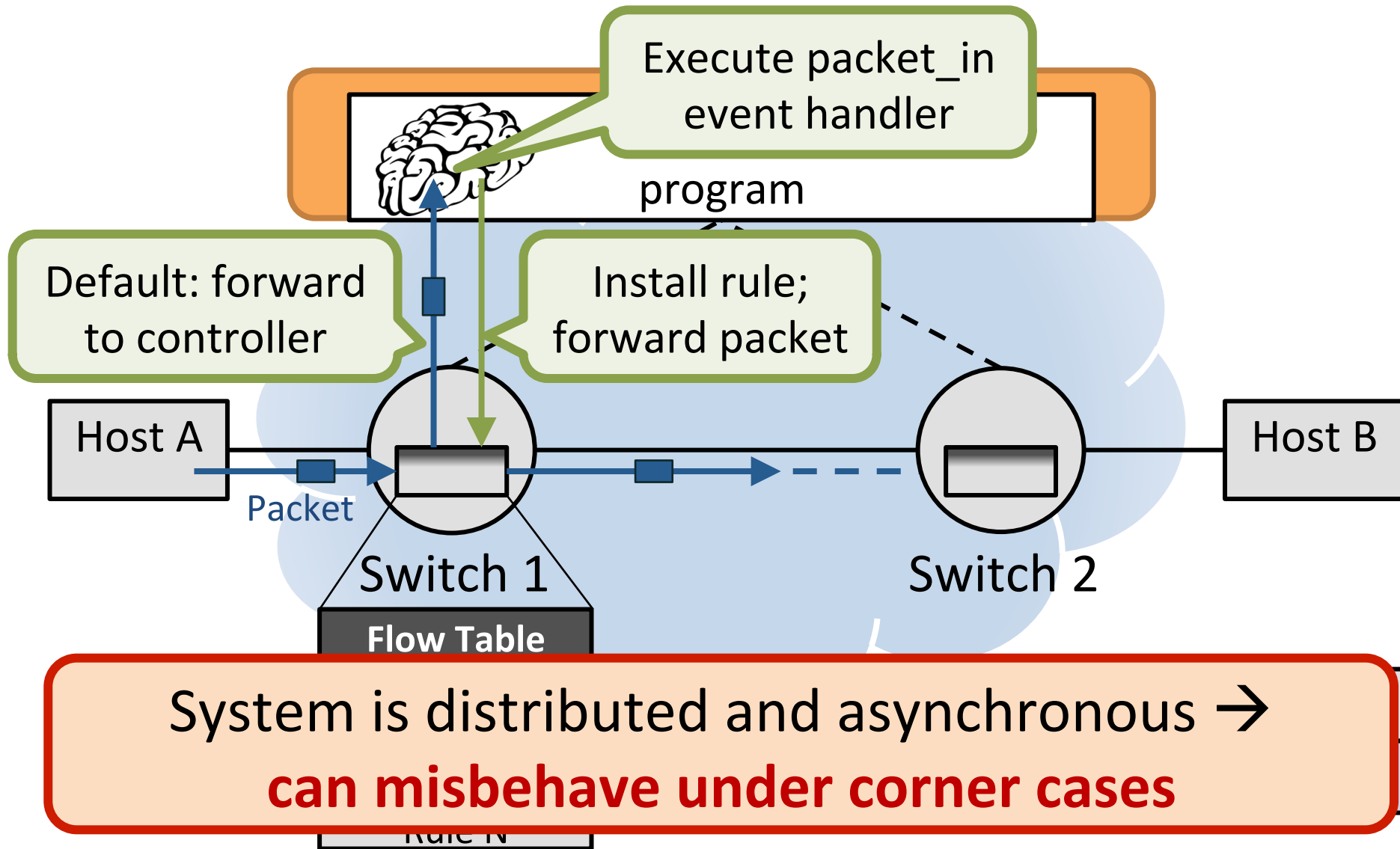
- Major hurdle for success of SDN

We need effective ways to test SDN networks

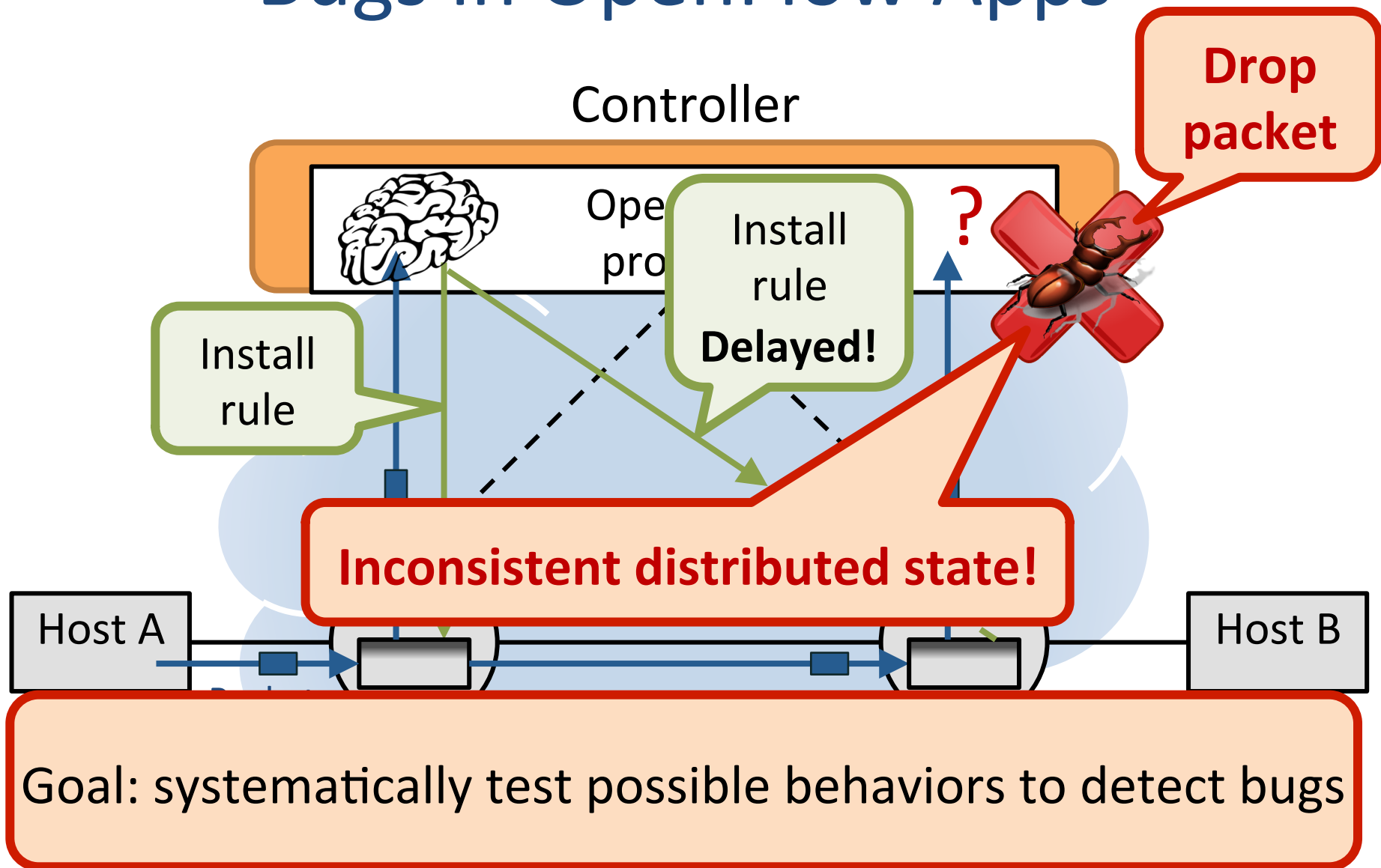
Roadmap

- Intro
- OpenFlow background
- NICE [NSDI'12]: systematically testing OpenFlow Apps
- SOFT [CoNEXT'12]: automating interop testing of OpenFlow Agents
- Conclusions

Quick OpenFlow 101



Bugs in OpenFlow Apps



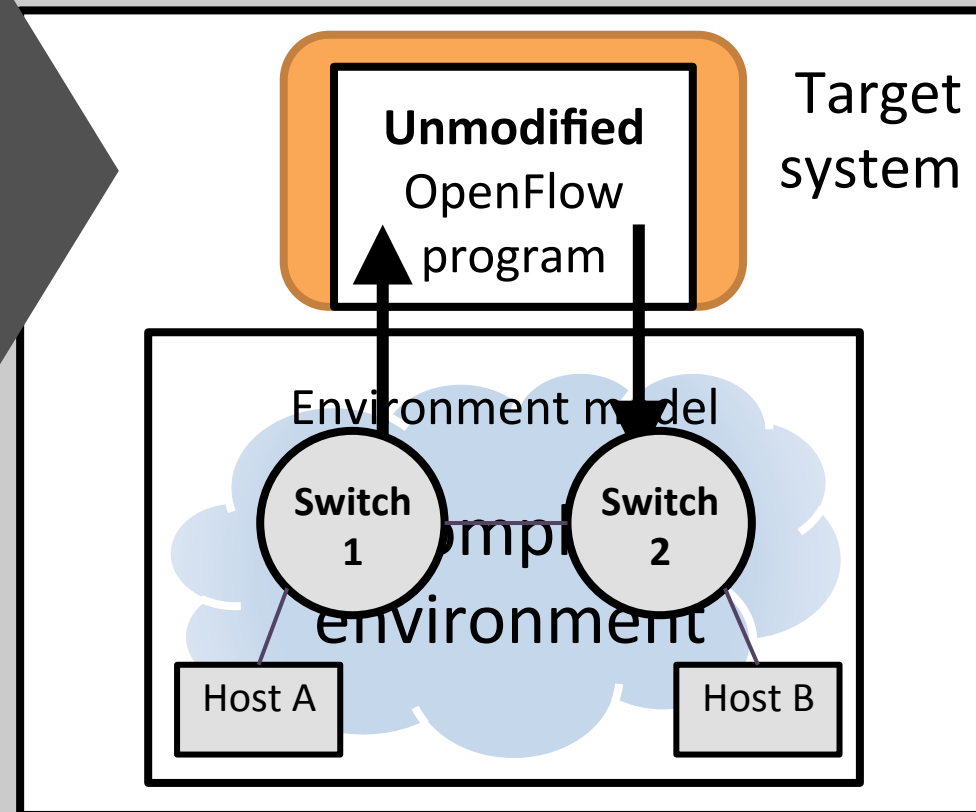
Roadmap

- Intro
- OpenFlow background
- NICE [NSDI'12]: systematically testing OpenFlow Apps
- SOFT [CoNEXT'12]: automating interop testing of OpenFlow Agents
- Conclusions

Systematically Testing OpenFlow Apps

- Carefully-crafted streams of packets
- Many orderings of packet arrivals and events

State-space exploration via Model Checking (MC)



Scalability Challenges

Data-plane driven

Huge space of possible
packets

**Equivalence
classes of
packets**

Complex network behavior

Huge space of possible
event orderings

**Domain-specific
search
strategies**

Enumerating all inputs and event orderings is intractable

Input

Unmodified
OpenFlow
program

Network
topology

NICE

No bugs
In
Controller
Execution

State-space
search

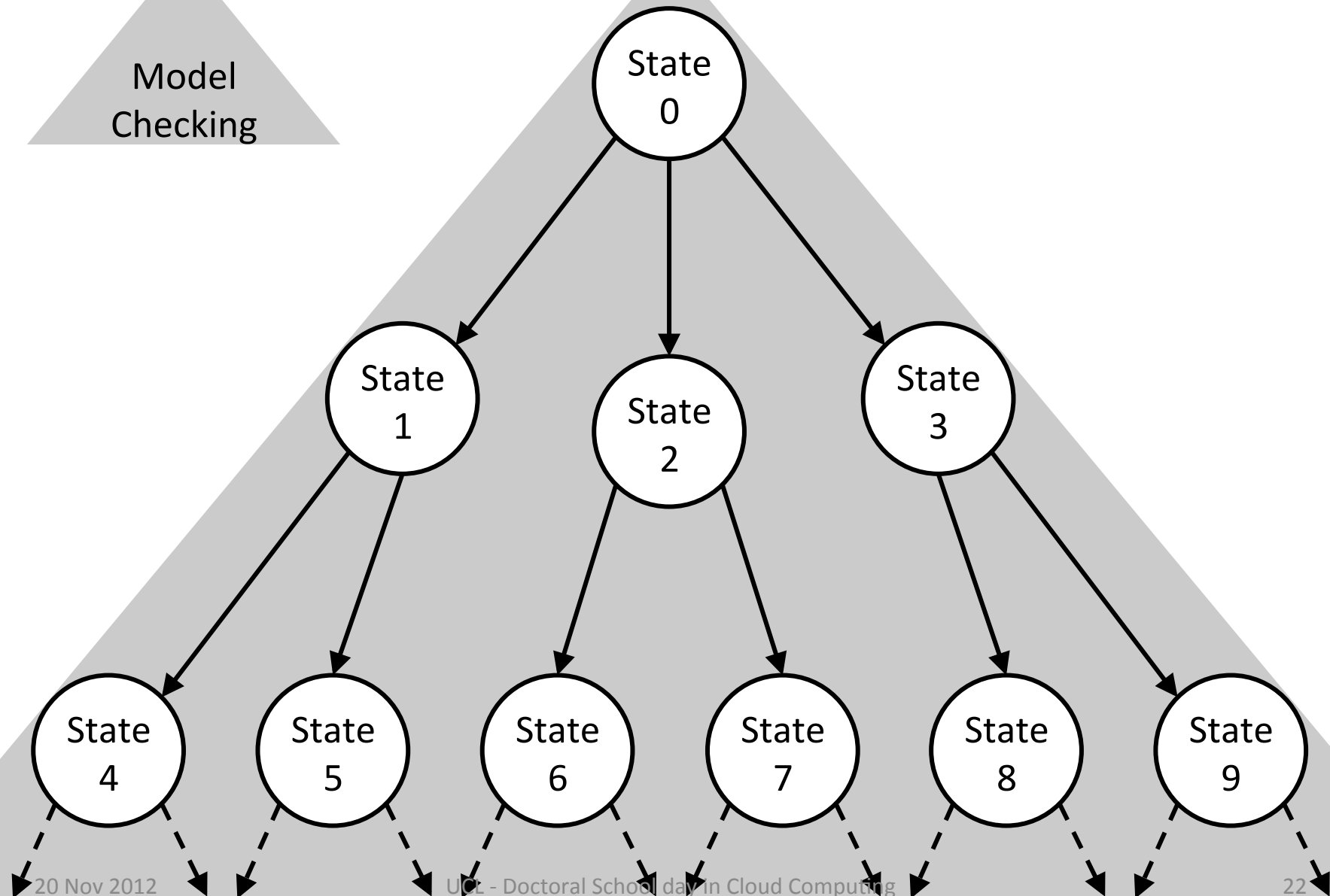
Output

Traces of
property
violations

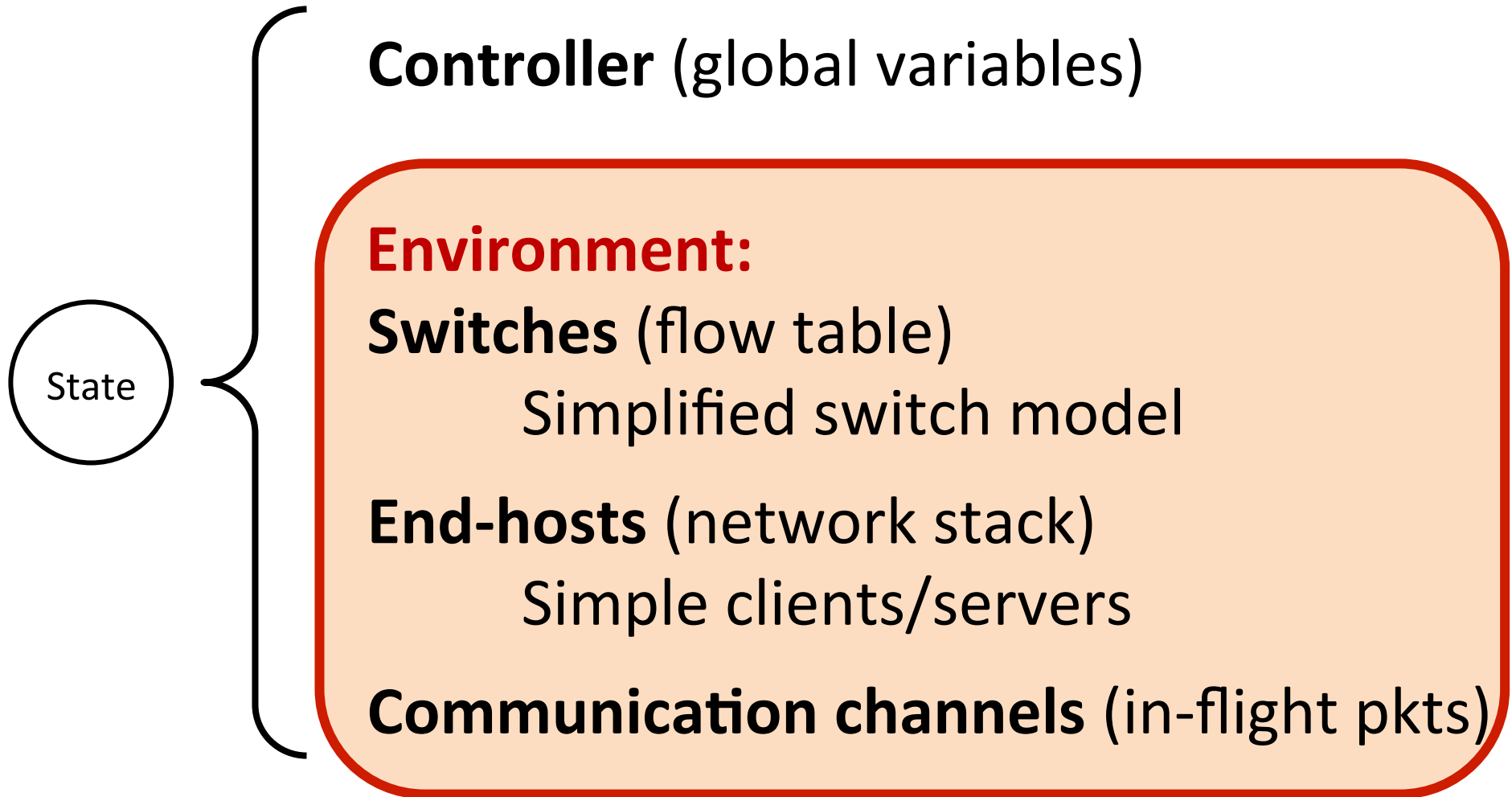
NICE found 11 bugs in 3 real OpenFlow Apps

State-Space Model

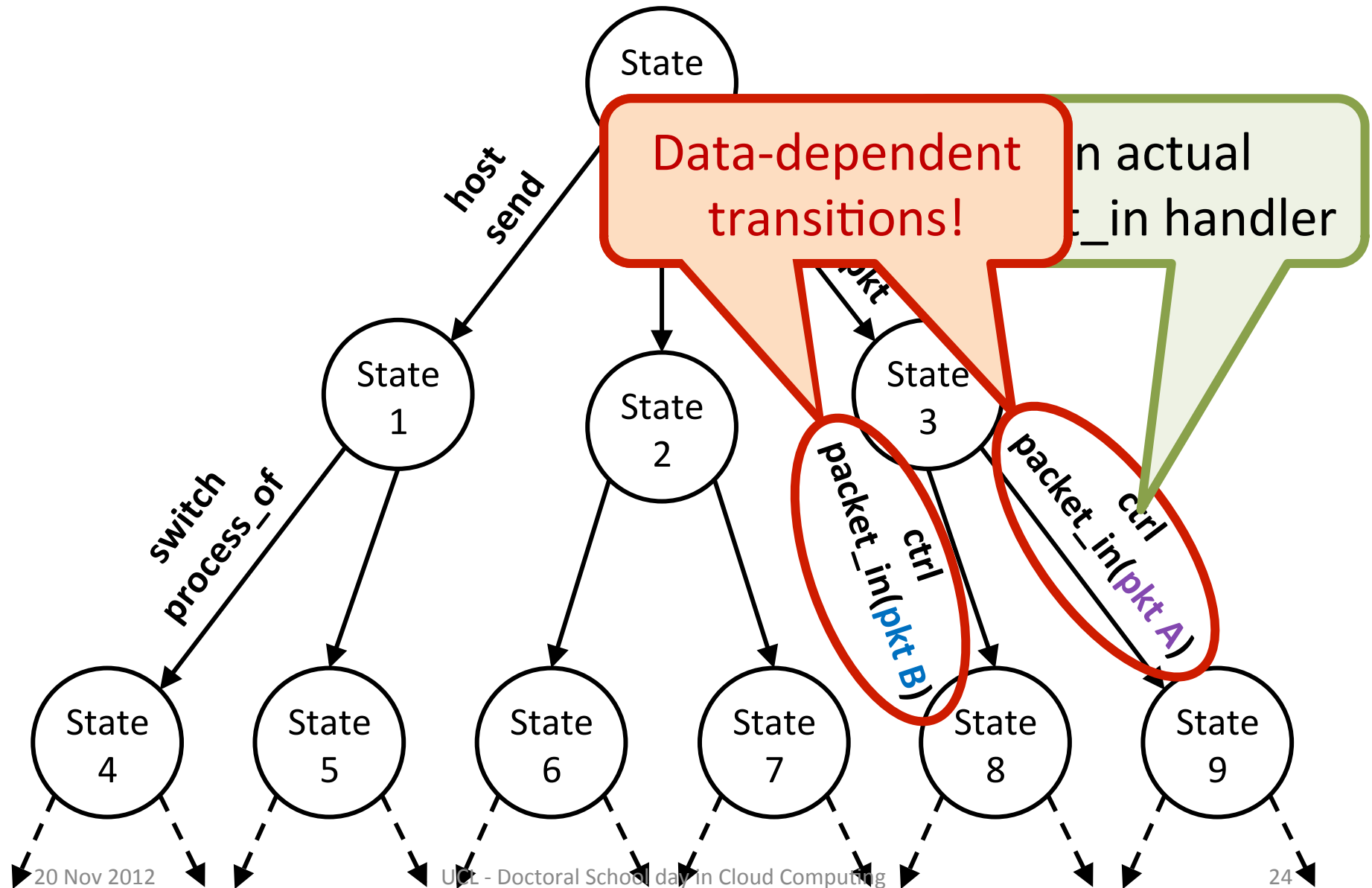
Model
Checking



System State



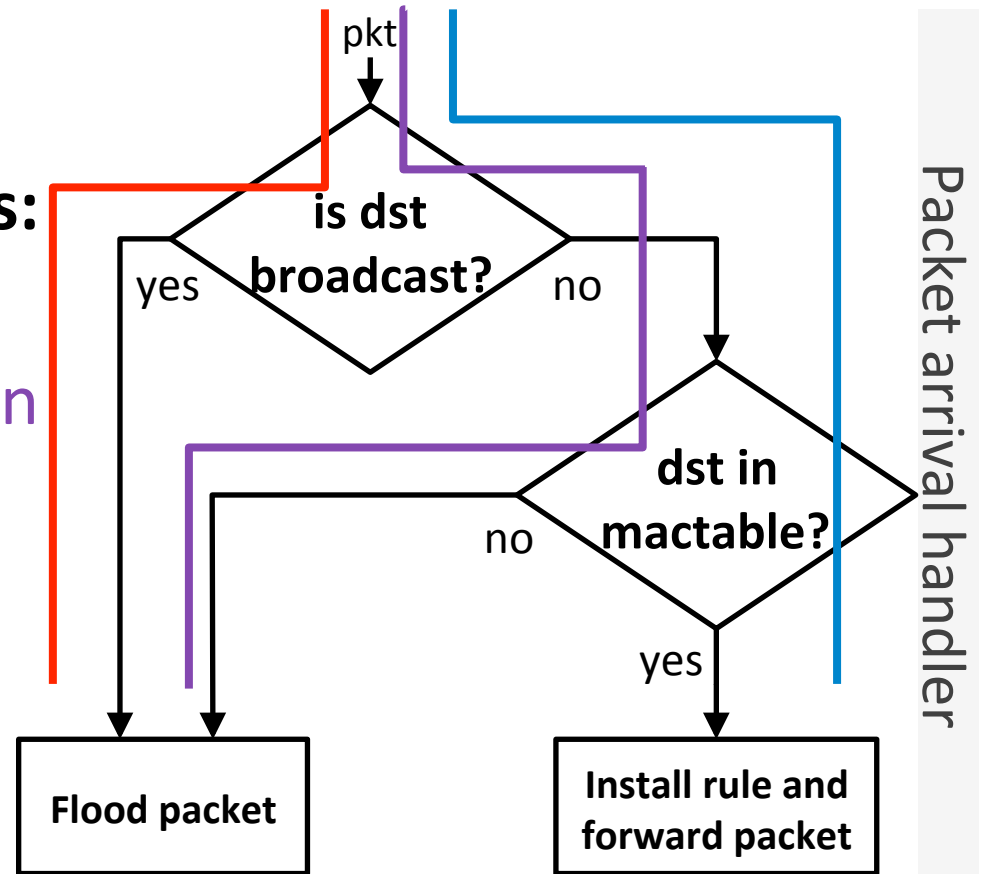
Transition System



Combating Huge Space of Packets

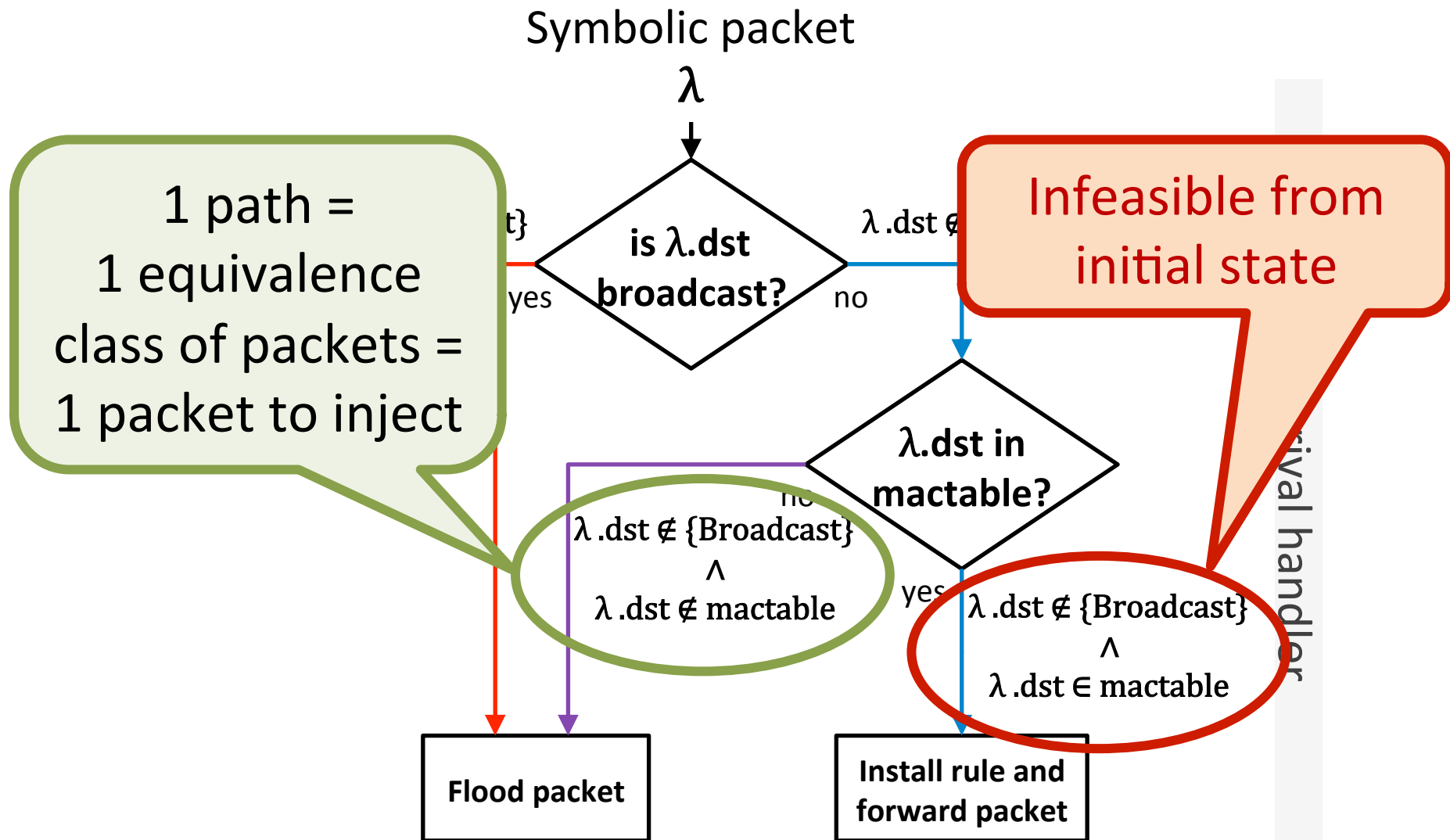
Equivalence classes of packets:

1. Broadcast destination
2. Unknown unicast destination
3. Known unicast destination

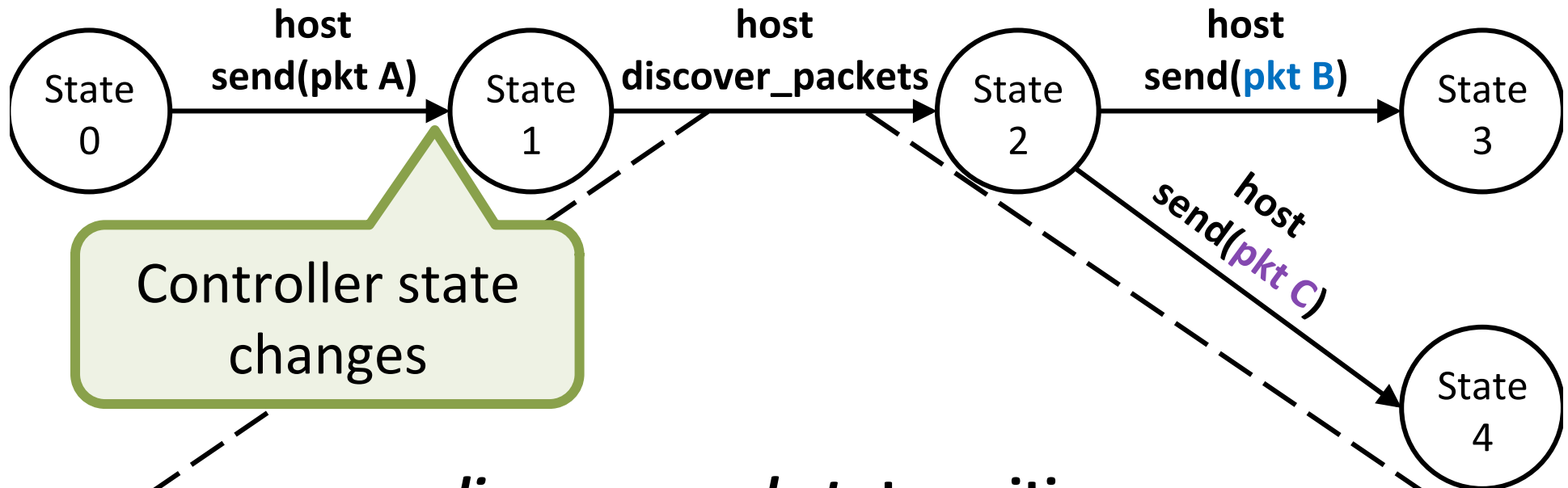


Code itself reveals equivalence classes of packets

Code Analysis: Symbolic Execution (SE)



Combining SE with Model Checking

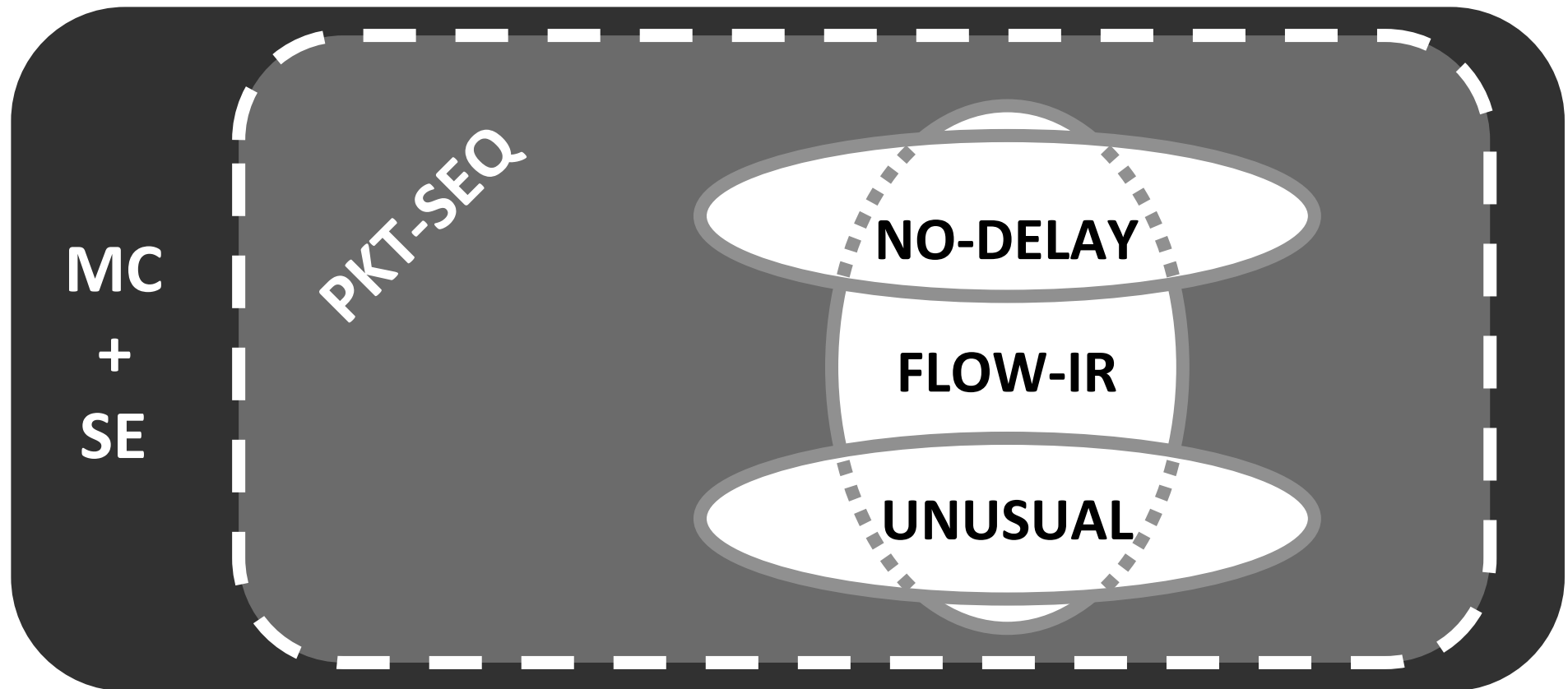


discover_packets transition:



Combating Huge Space of Orderings

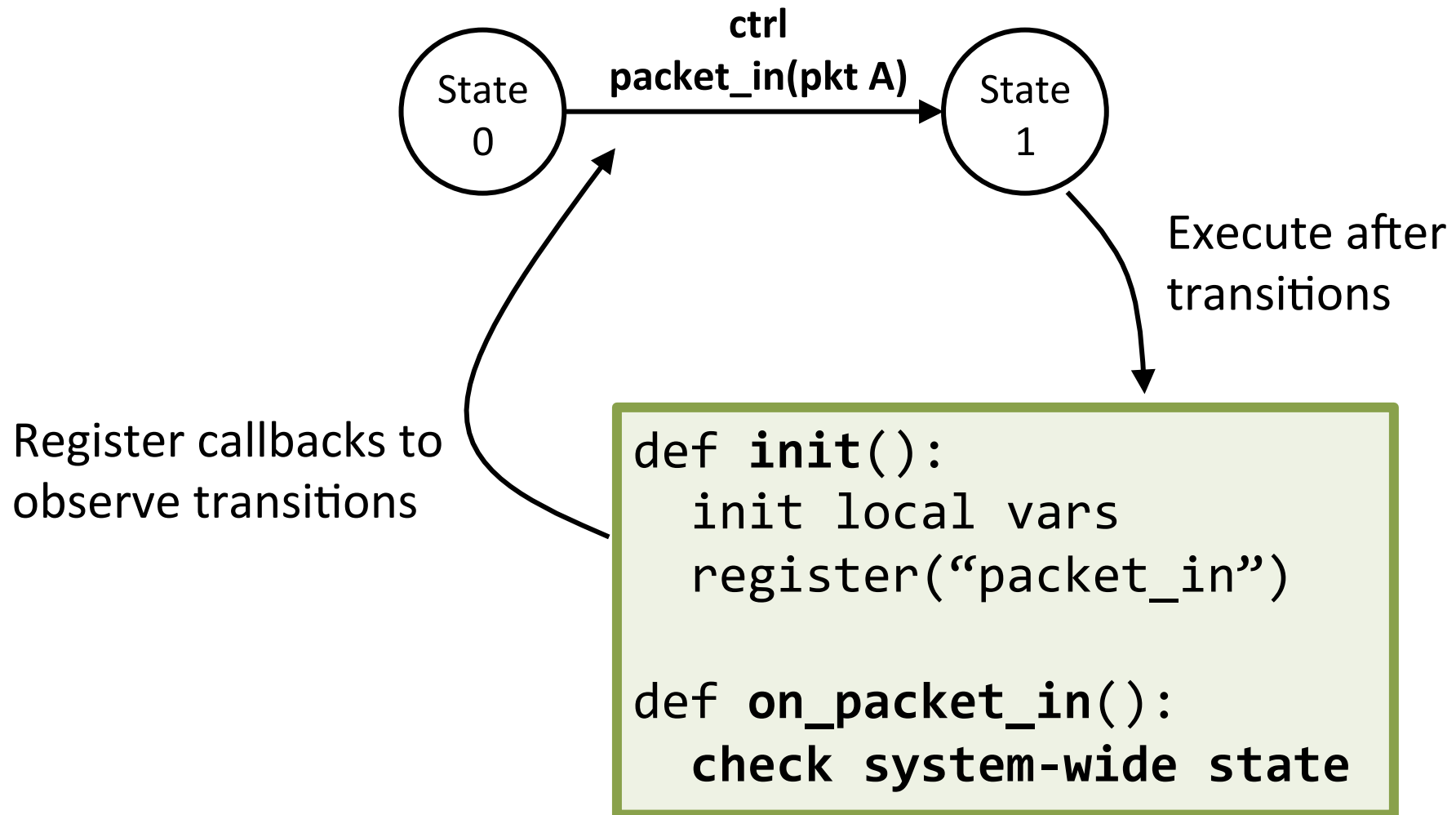
OpenFlow-specific search strategies for
up to 20x state-space reduction:



Specifying App Correctness

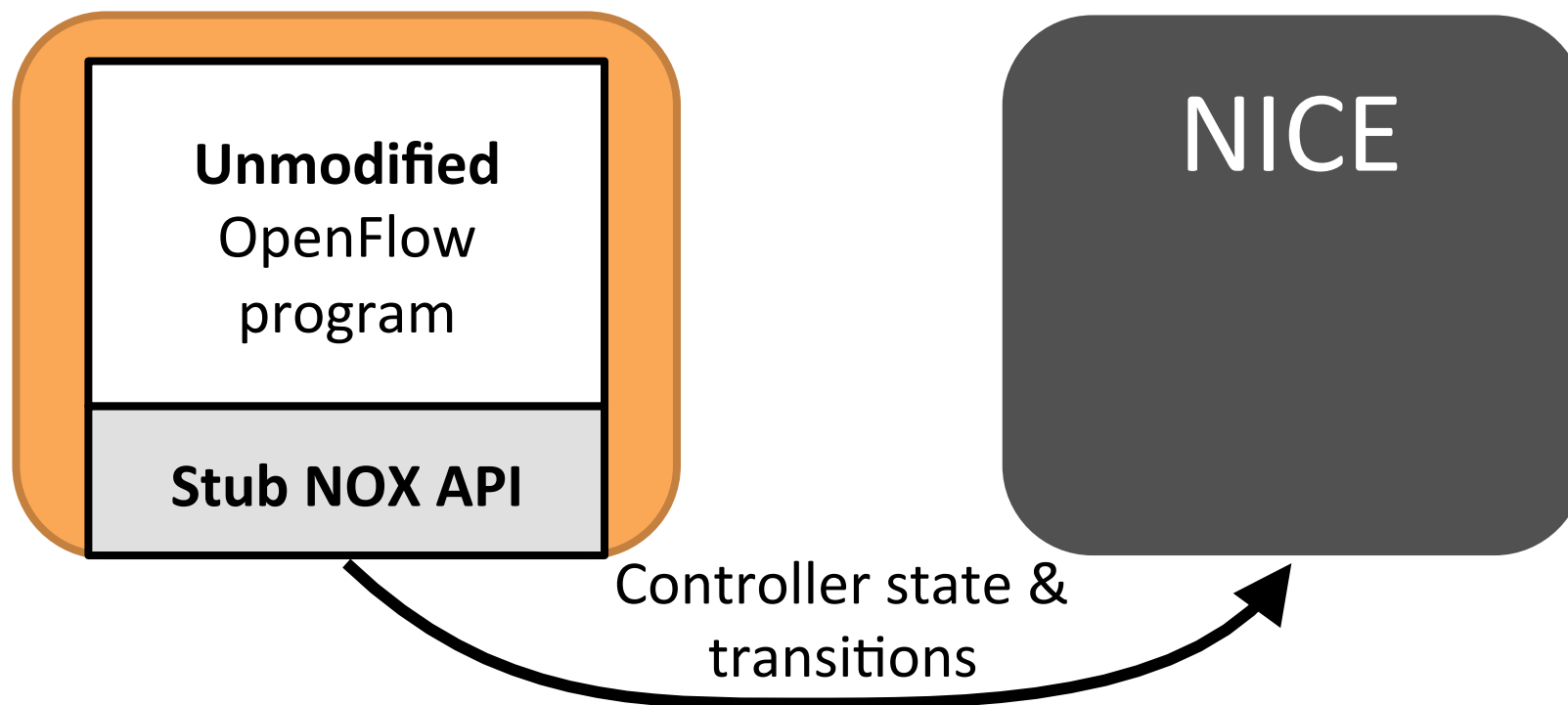
- Library of **common properties**
 - No forwarding loops
 - No black holes
 - Direct paths (no unnecessary flooding)
 - Etc...
- Correctness is app-specific in nature

API to Define App-Specific Properties



Prototype Implementation

- Built a NICE prototype in Python
- Target the Python API of NOX



Experiences

- Tested 3 unmodified NOX OpenFlow Apps
 - MAC-learning switch
 - LB: Web server load balancer [Wang et al., HotICE'11]
 - TE: Energy-aware traffic engineering [CoNEXT'11]
- Setup
 - Iterated with 1, 2 or 3-switch topologies; 1,2,... pkts
 - App-specific properties
 - LB: All packets of same request go to same server replica
 - TE: Use appropriate path based on network load

Results

- NICE found **11 property violations** → **bugs**
 - Few secs to find 1st violation of each bug (max 30m)
 - Few simple mistakes (not freeing buffered packets)
 - **3 insidious bugs** due to network race conditions

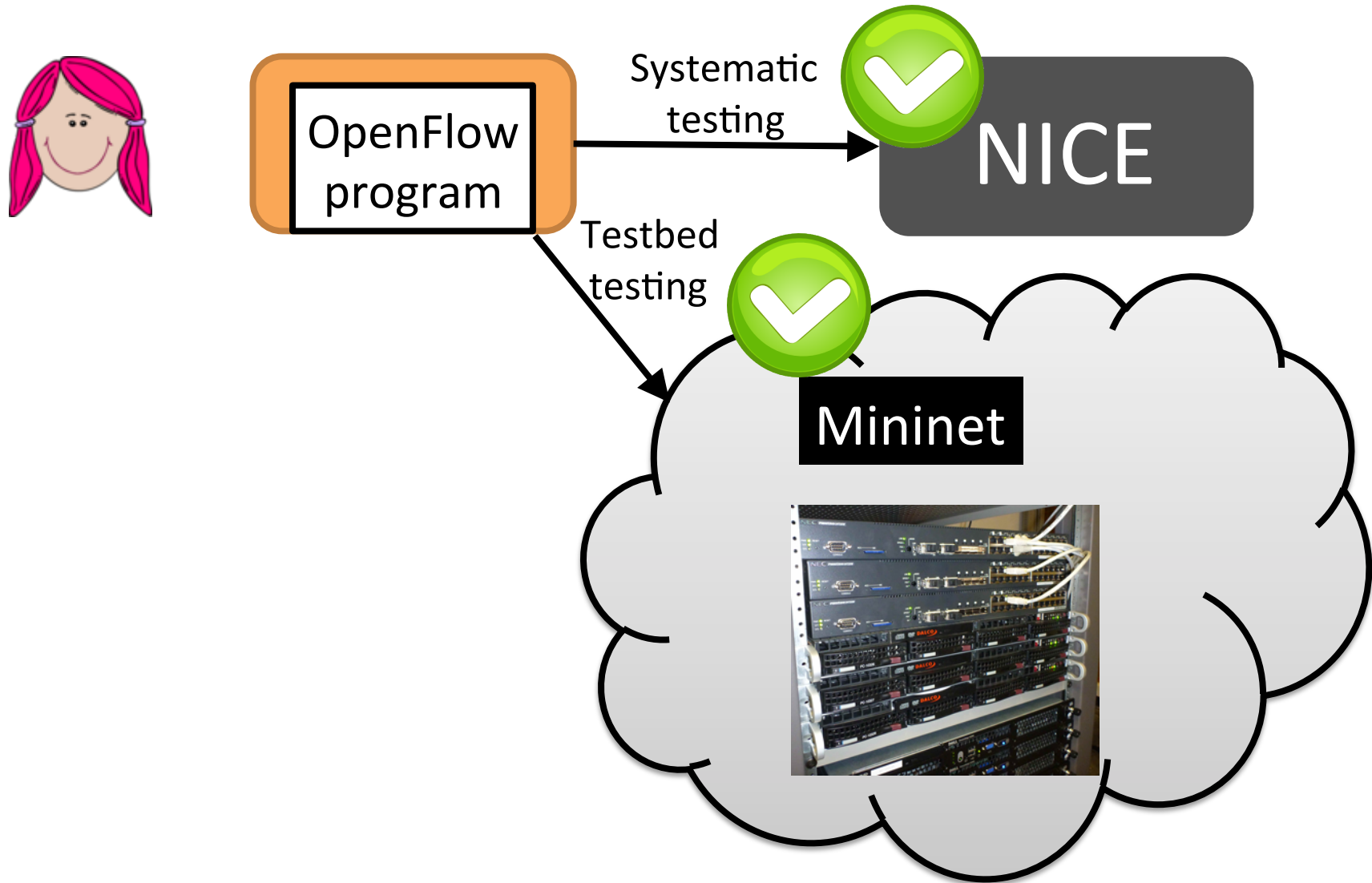
Take Aways

- Why were mistakes easy to make?
 - Centralized programming model only an abstraction
- Why the programmer could not detect them?
 - Bugs don't always manifest
 - TCP masks transient packet loss
 - Platform lacks runtime checks
- Why NICE easily found them?
 - Makes corner cases as likely as normal cases

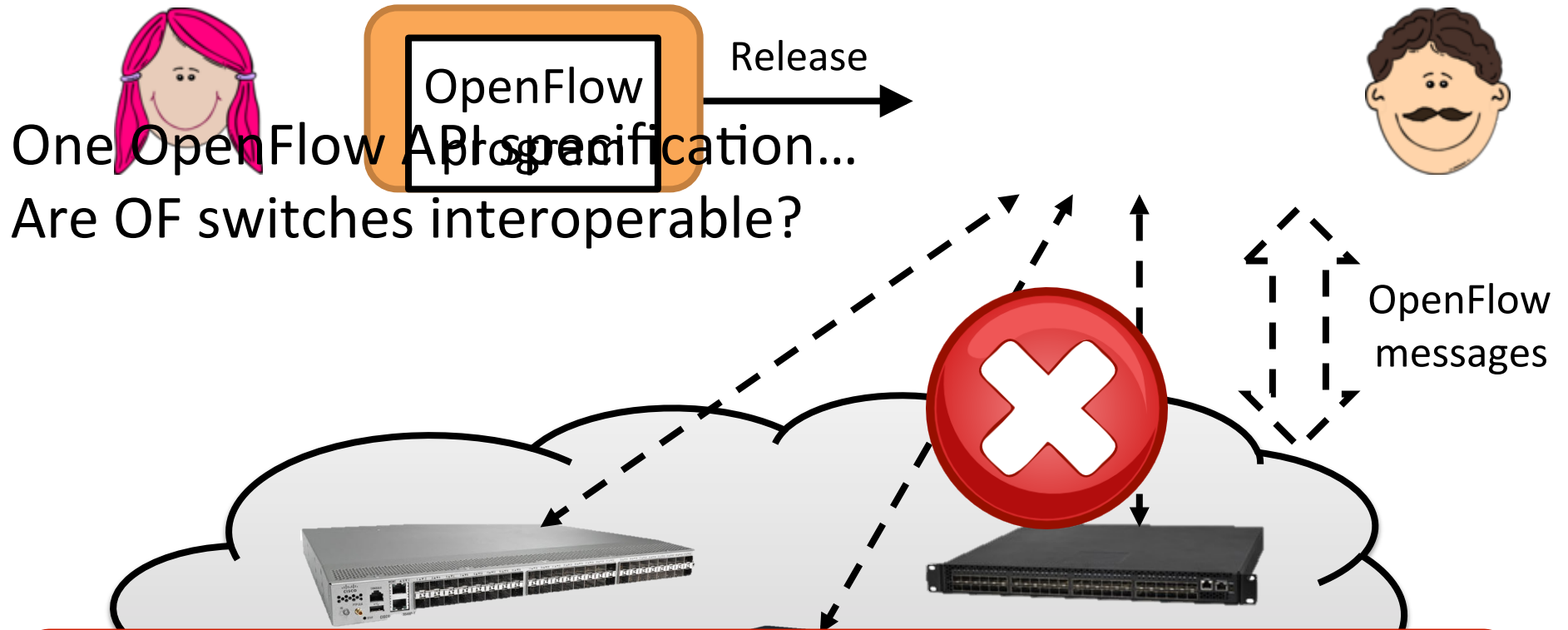
Roadmap

- Intro
- OpenFlow background
- NICE [NSDI'12]: systematically testing OpenFlow Apps
- **SOFT [CoNEXT'12]: automating interop testing of OpenFlow Agents**
- **Conclusions**

Interoperability at Deployment Time



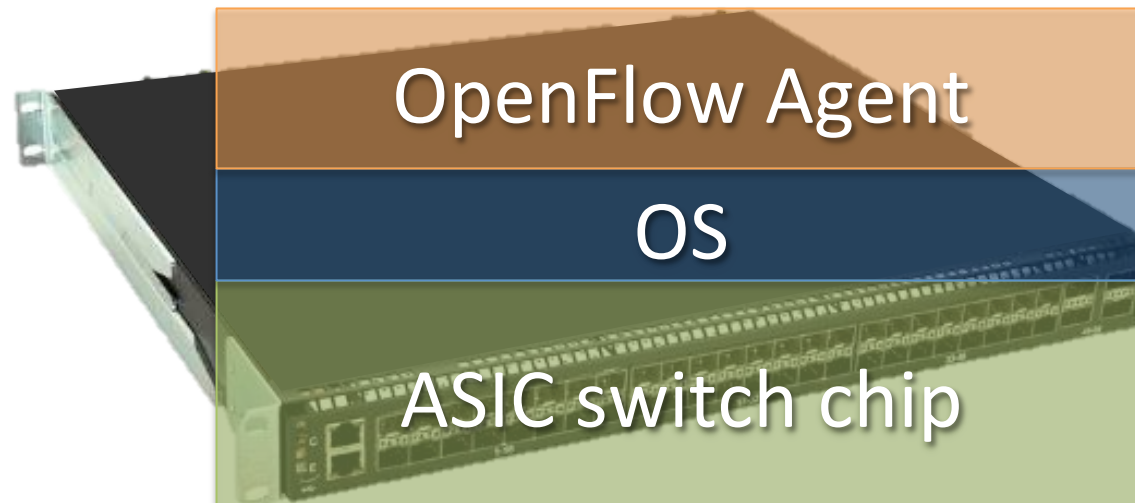
Interoperability at Deployment Time



Interop is critical for the success of SDN

Interop: How Hard Can It Be?

OF Switch

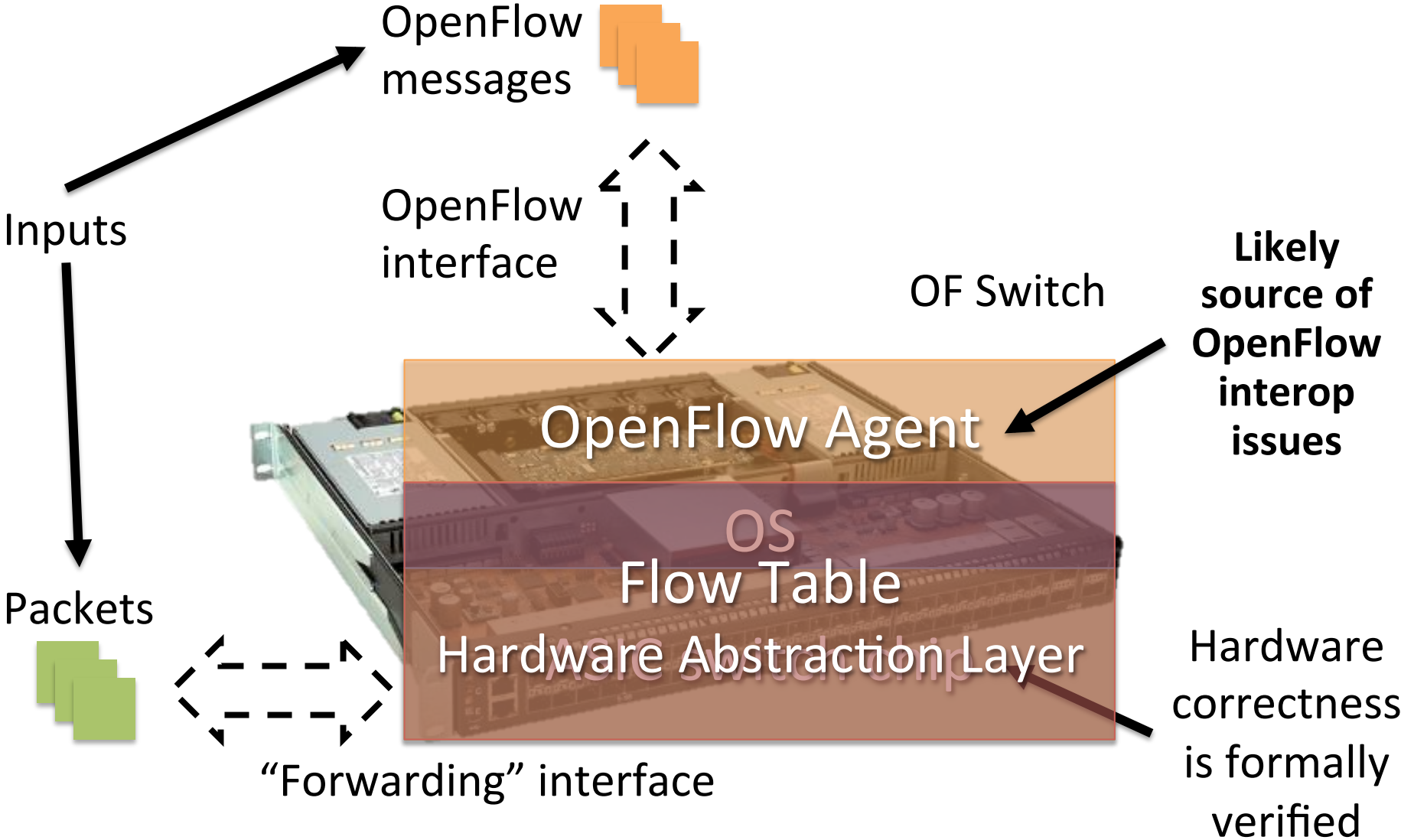


Definition of Interoperability*

“Being able to **accomplish end-user applications** using different types of systems, whose **interfaces are completely understood**, in a manner that requires the user to have **little or no knowledge of the unique characteristics of those systems**”

* NB: Many other definitions exist

Interop: How Hard Can It Be?



OpenFlow Software Agent

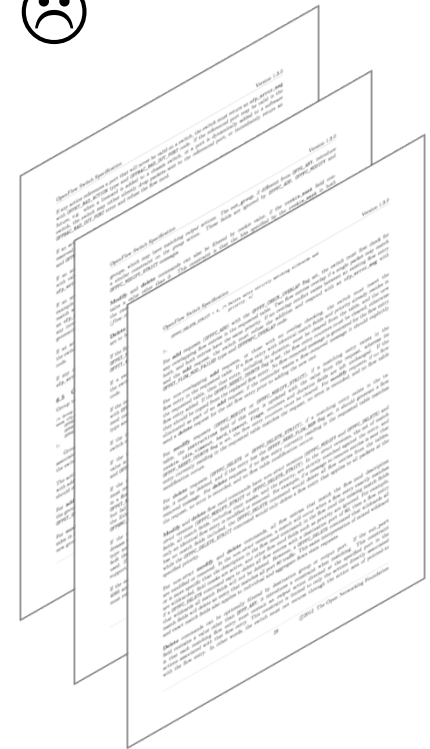
Switch software is not provably correct ☹️

Specifications

- Rapid flux (3 revisions in ~ 1 year)
- Ambiguities (FlowMod is 2.5 pages long)

Specifications → Implementation

- Implementation freedom
- Vendors may not follow the specs



Testing, testing and testing...

Interop Testing Today



LET'S GO TO VEGAS

Interop'12 Testing Event



- Gather various vendors in Vegas
- Hook up switches and controllers
- Create and run test cases
- See what breaks and ...

~~What happens in Vegas,
stays in Vegas~~

FIX IT!

- Very high manual effort
- Test cases are not exhaustive
- It is not a one time thing

Automating Interop Testing

Insight:
systematically crosscheck OF implementations

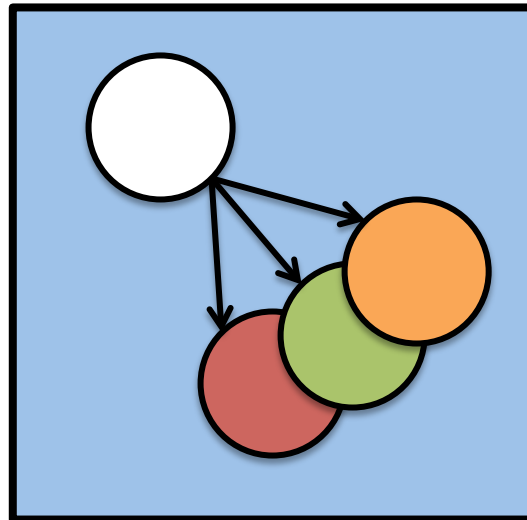
The 10,000 foot view

Test inputs

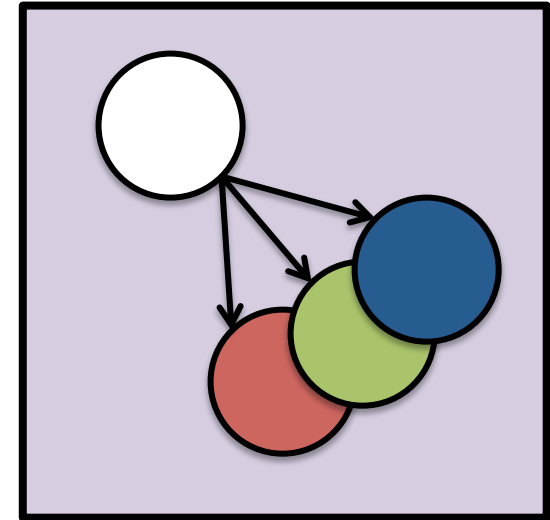


Input-driven execution

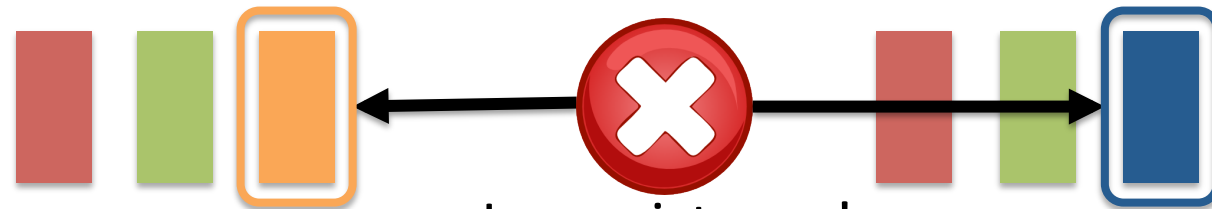
OF Agent 1



OF Agent 2



Observable behaviors



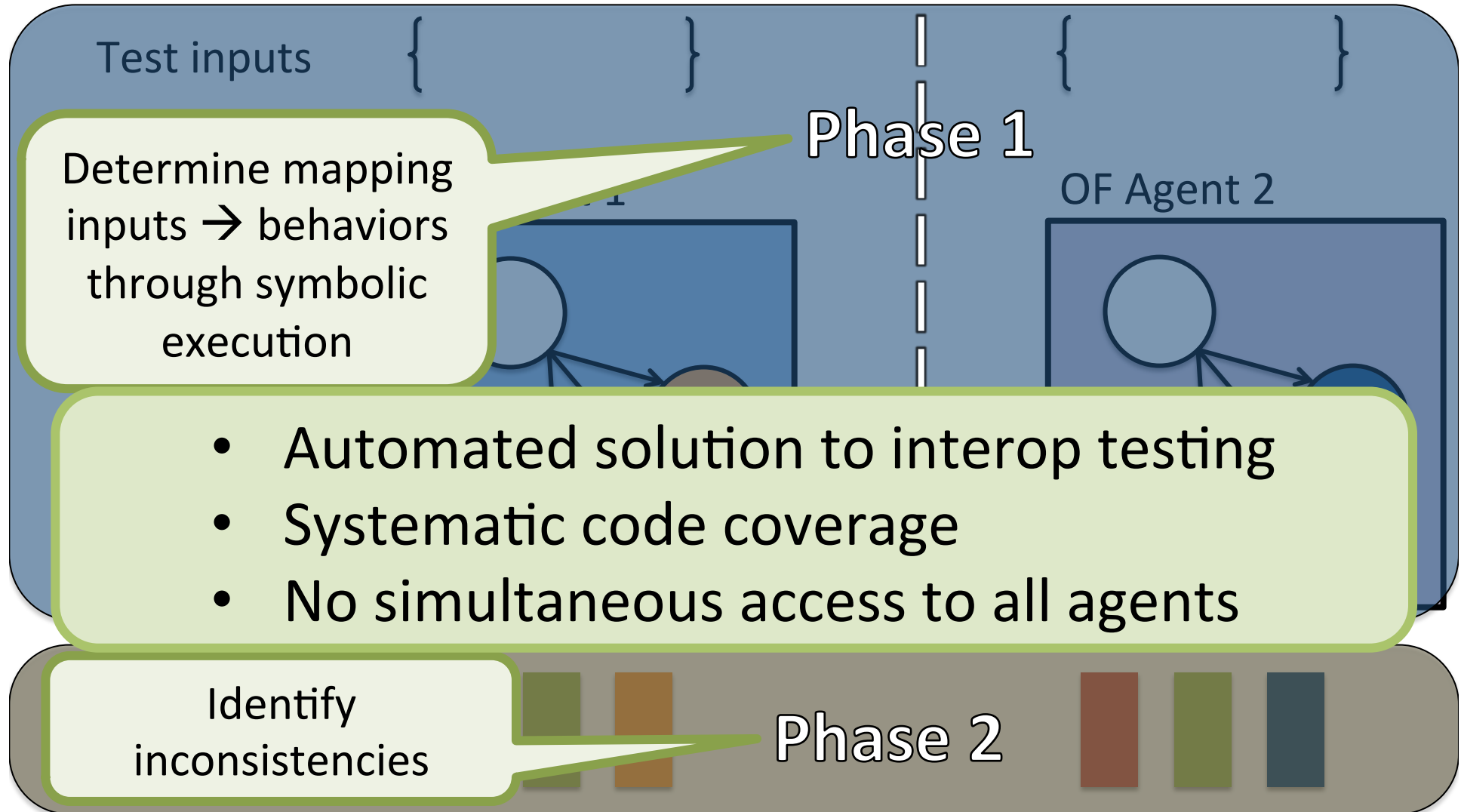
Inconsistency!

Challenges

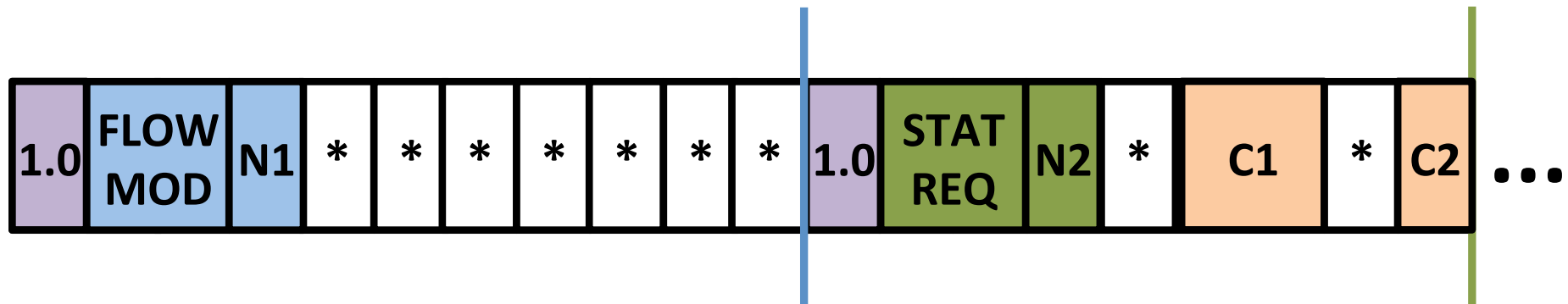
- Manage test inputs and coverage efficiently
 - Or manage “path explosion”
- Capture behaviors
- Avoid simultaneous access to all code

SOFT

(Systematic OpenFlow Testing)



Structured Inputs



Further reductions

- Some inputs are independent
- Many inputs are entirely concrete
- Small number of messages
- Concrete values at cost of completeness

Capturing Behaviors

Externally observable outputs

- OpenFlow reply messages
- Data plane packets
- Normalize harmless nondeterminism (e.g., Buffer IDs)

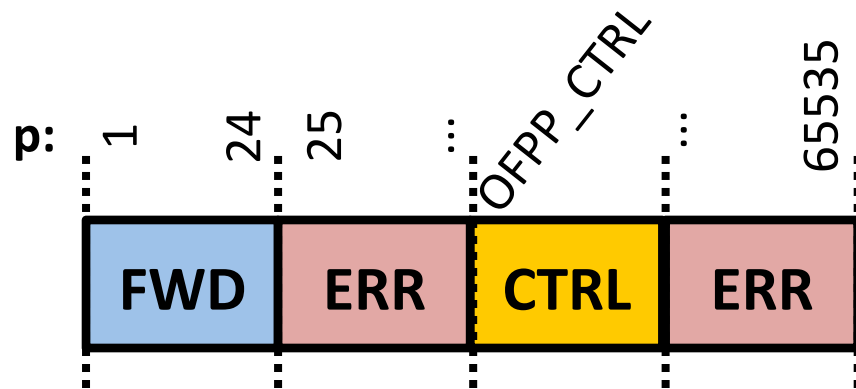
Internal state changes affect successive inputs

- Use concrete probe packets

Example

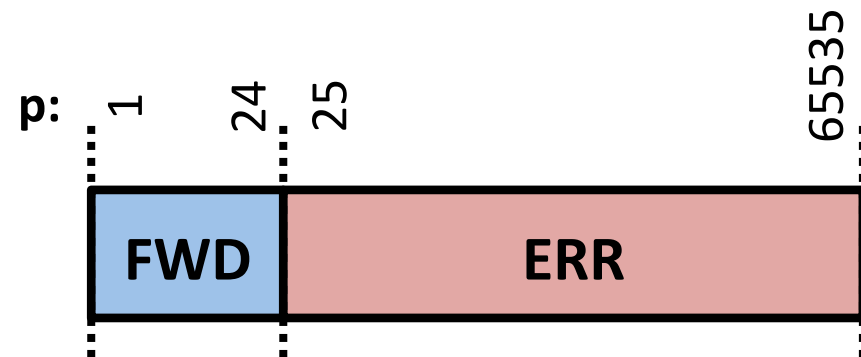
Agent 1

```
If ( p == OFPP_CTRL )
    send_to_ctrl ( )
else if ( p < 25 )
    send_to_port( p )
else
    error( BAD_PORT )
```



Agent 2

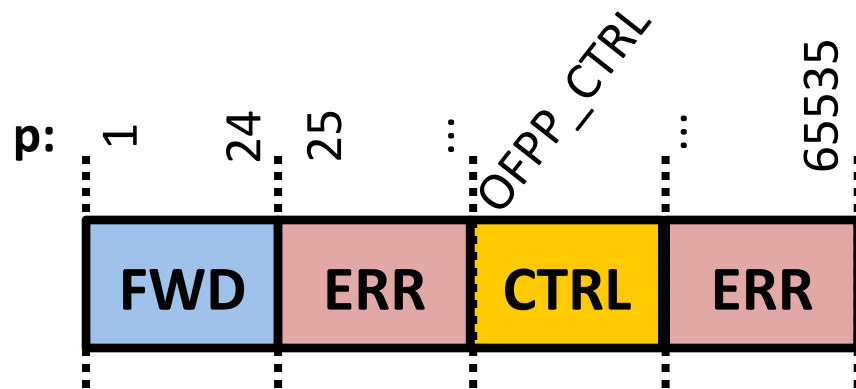
```
if ( p < 25 )
    send_to_port( p )
else
    error( BAD_PORT )
```



N-version Comparison

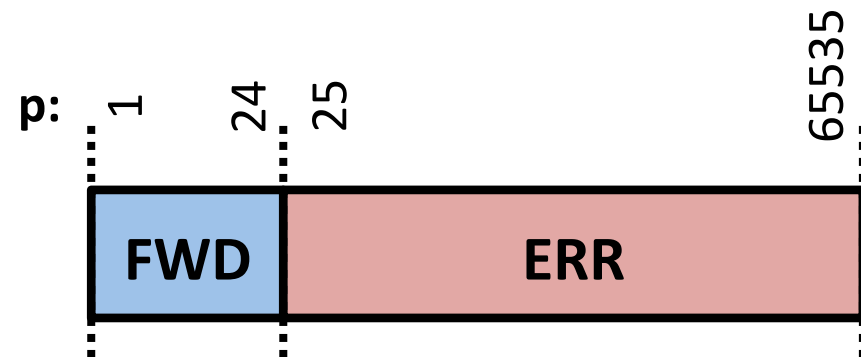
Agent 1

```
If ( p == OFPP_CTRL )  
    send_to_ctrl ( )  
else if ( p < 25 )  
    send_to_port( p )  
else  
    error( BAD_PORT )
```

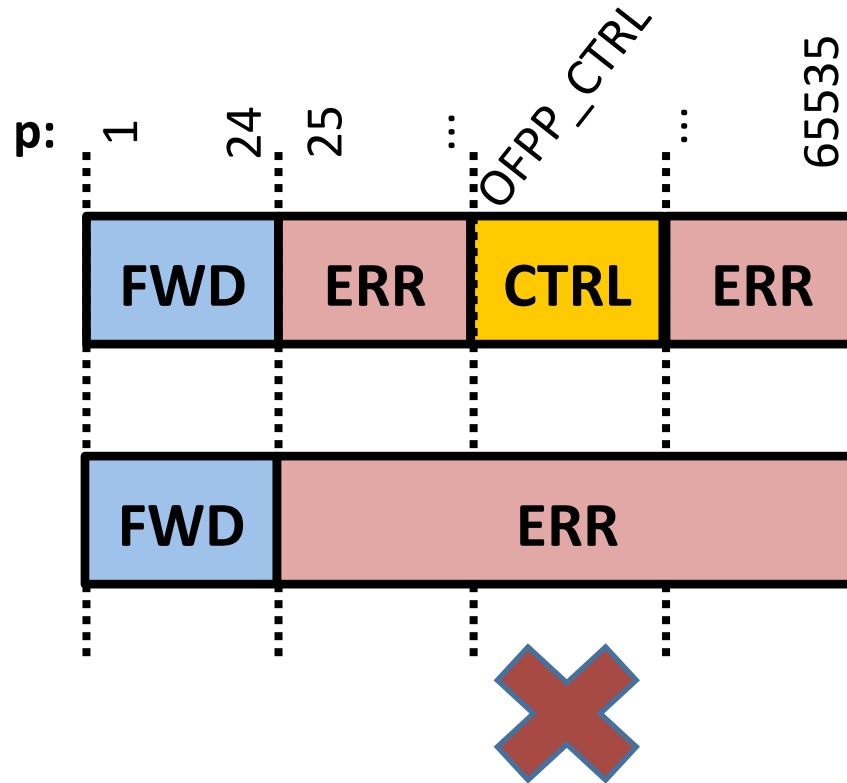


Agent 2

```
if ( p < 25 )  
    send_to_port( p )  
else  
    error( BAD_PORT )
```



N-version Comparison



Results

- Compared
 - OpenFlow 1.0 Switch Reference Implementation
 - Open VSwitch 1.0.0

- Input Sequences containing 1 - 4 messages

Results

Found 7 classes of inconsistencies

Mostly related to message validation

Result of underspecification

- No expected behavior in the specification
- Inconsistent interpretation of the specification

Results - Example

FlowMod message

1. Modify VLAN to value greater than 2^{12}
2. Forward packet

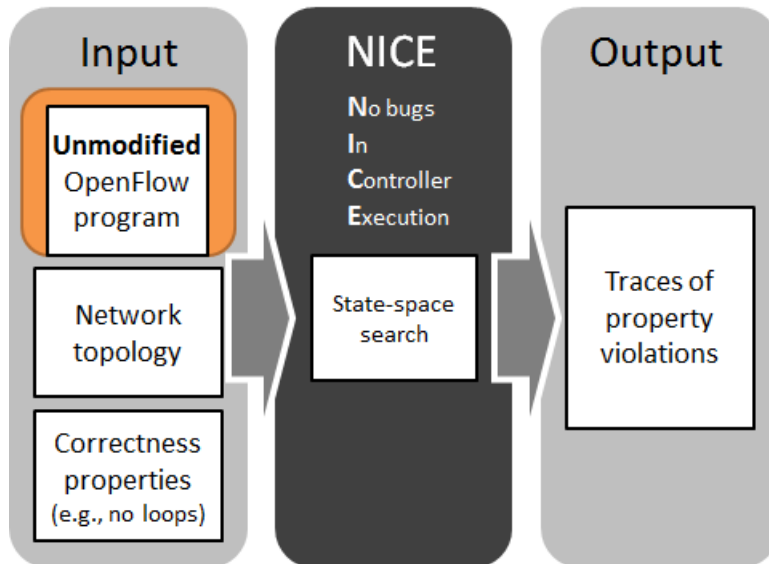
Network in 2 different states
Which one is assumed by the controller?

1. Trim VLAN value to 12 bits
2. Install the rule

1. Silently ignore the message

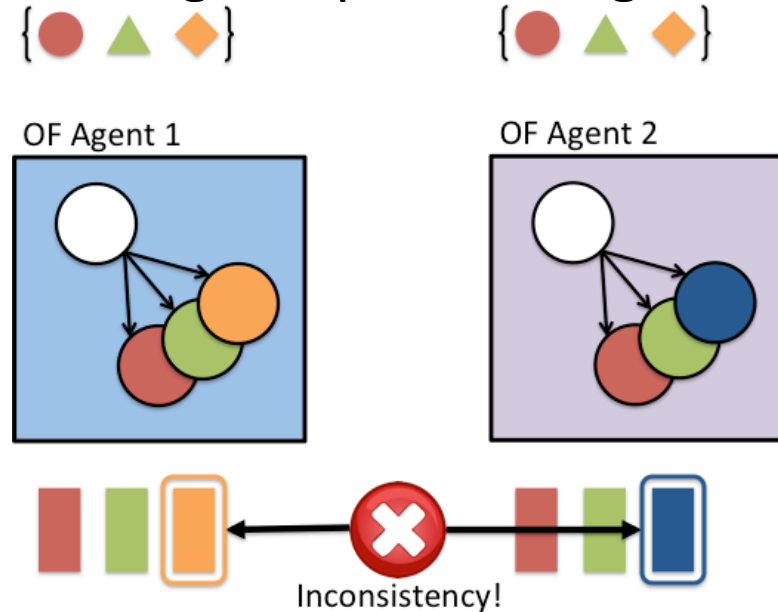
Conclusions

NICE automates the testing of OpenFlow Apps



<http://code.google.com/p/nice-of/>

SOFT automates interop testing of OpenFlow Agents



SDN: a new role for software tool chains to make networks more dependable. NICE and SOFT are a step in this direction!

Thanks



Peter Perešini
(EPFL)



Maciej Kuźniar
(EPFL)



Daniele Venzano
(EPFL)



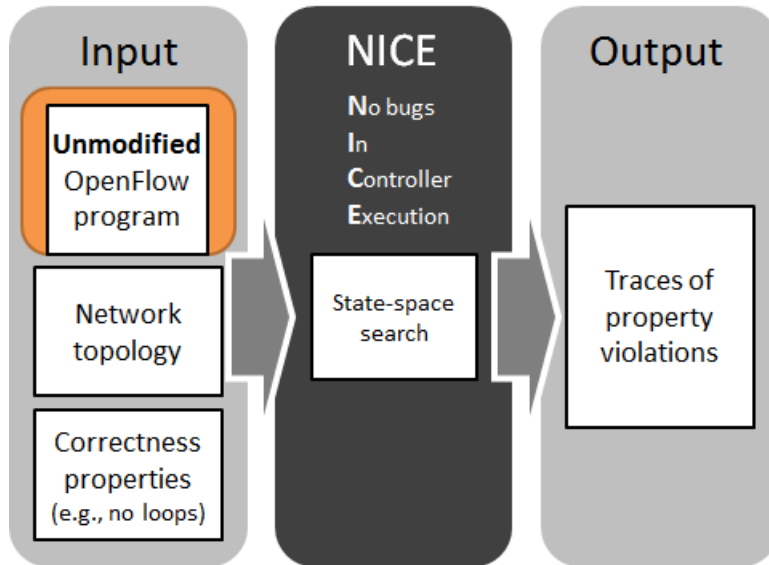
Dejan Kostić
(EPFL → IMDEA Networks)



Jennifer Rexford
(Princeton)

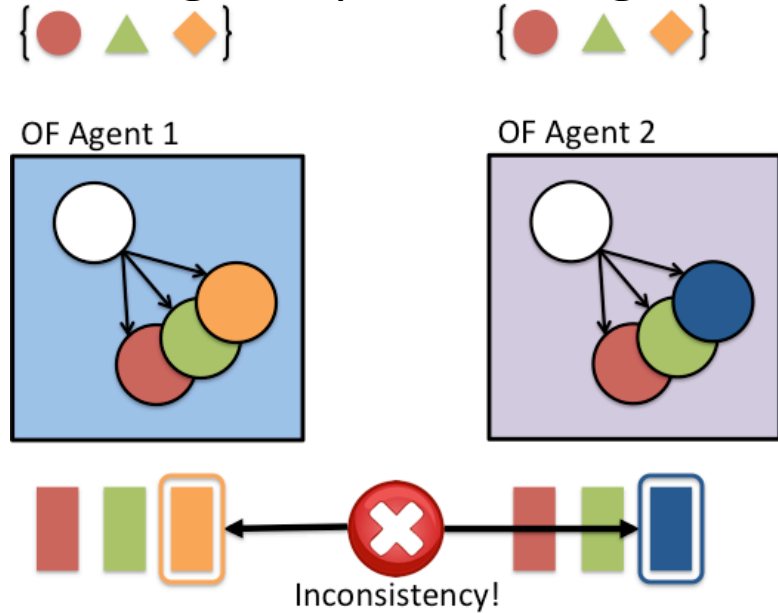
Thank you! Questions?

NICE automates the testing of OpenFlow Apps



<http://code.google.com/p/nice-of/>

SOFT automates interop testing of OpenFlow Agents



SDN: a new role for software tool chains to make networks more dependable. NICE and SOFT are a step in this direction!