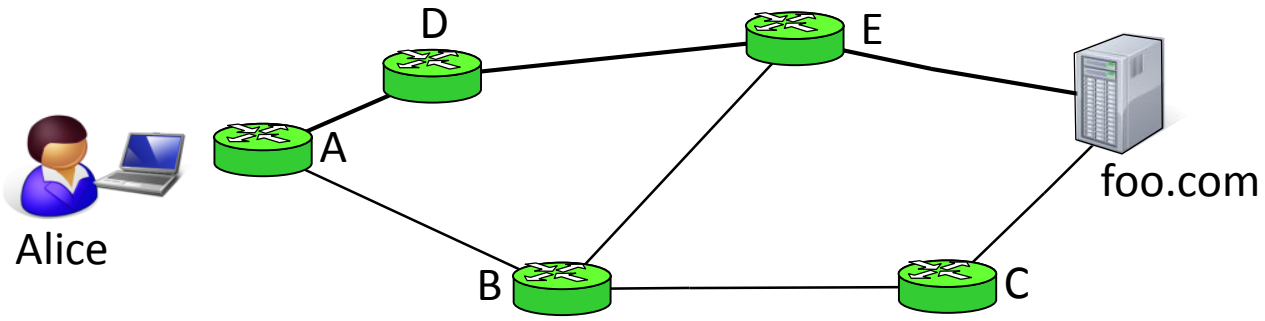
# Data Provenance at Internet Scale: Architecture, Experiences, and the Road Ahead

**Ang Chen, Yang Wu, Andreas Haeberlen,**
**Boon Thau Loo, Wenchao Zhou**

Penn

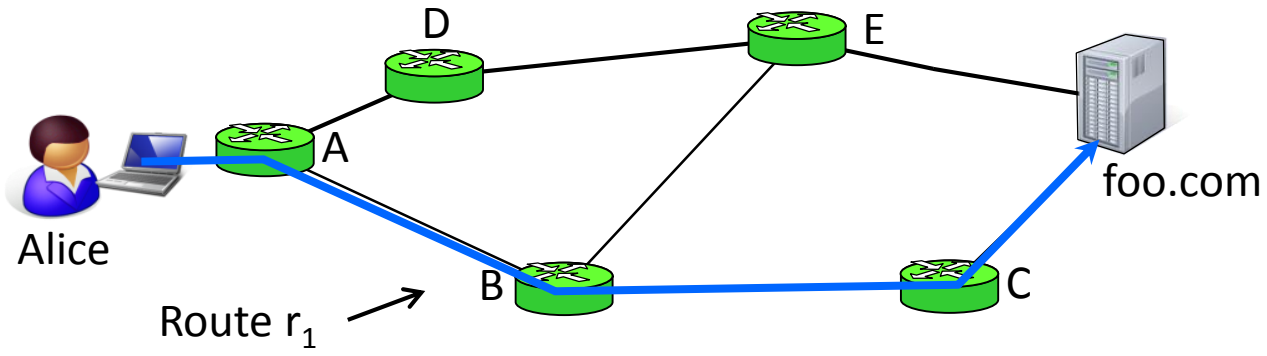
GEORGETOWN UNIVERSITY

# Motivation



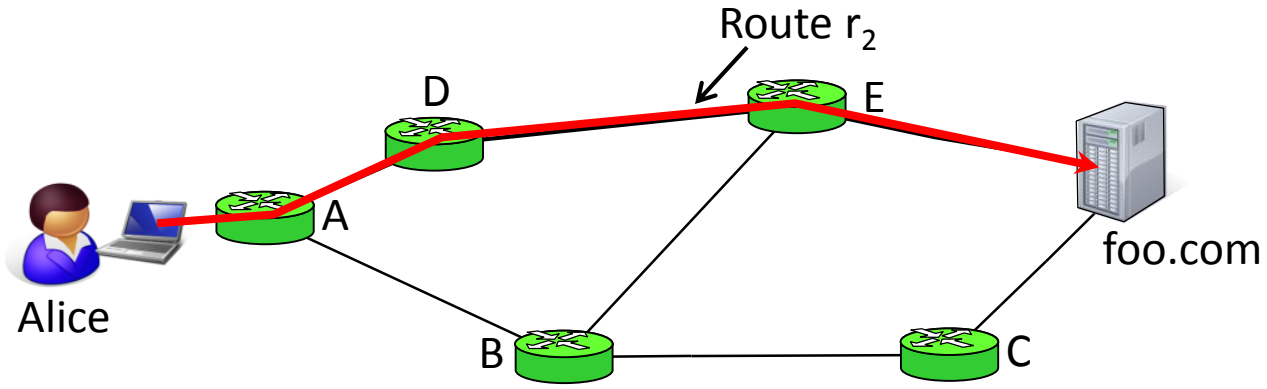

- **An example scenario: network routing**
  - System administrator observes strange behavior
  - Example: the route to foo.com has suddenly changed
  - Anomalies in distributed systems
    - **Need a way to explain system behavior.**

# Motivation



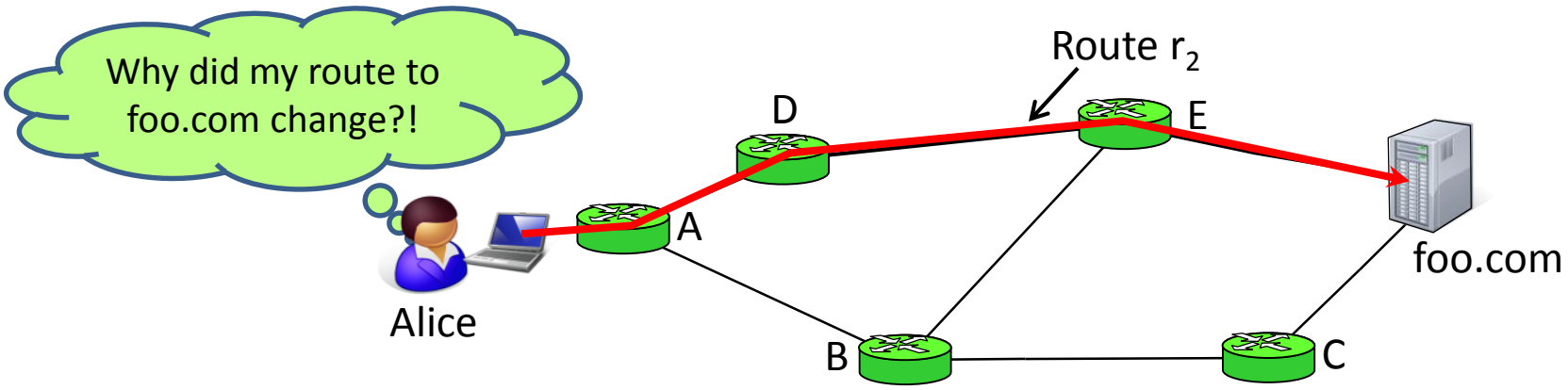

- **An example scenario: network routing**
  - System administrator observes strange behavior
  - Example: the route to foo.com has suddenly changed
  - Anomalies in distributed systems
    - **Need a way to explain system behavior.**

# Motivation



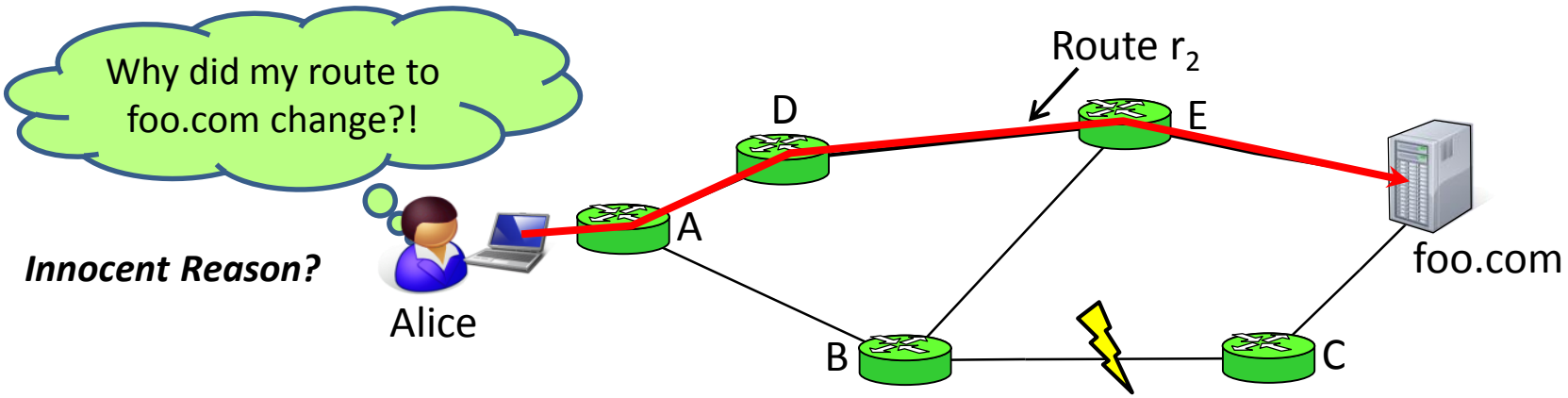

- **An example scenario: network routing**
  - System administrator observes strange behavior
  - Example: the route to foo.com has suddenly changed
  - Anomalies in distributed systems
    - **Need a way to explain system behavior.**

# Motivation



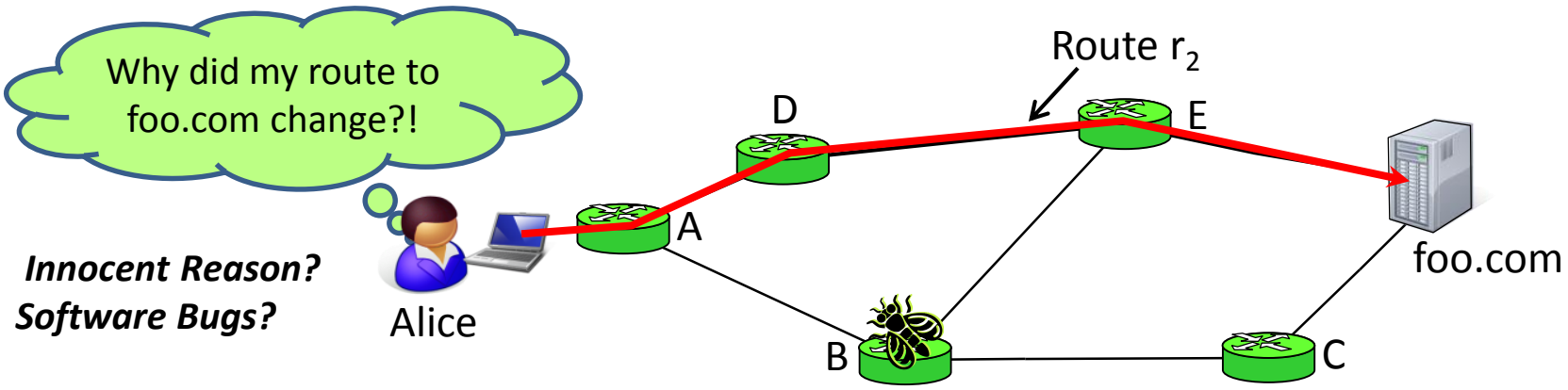

- **An example scenario: network routing**
  - System administrator observes strange behavior
  - Example: the route to foo.com has suddenly changed
  - Anomalies in distributed systems
    - **Need a way to explain system behavior.**

# Motivation



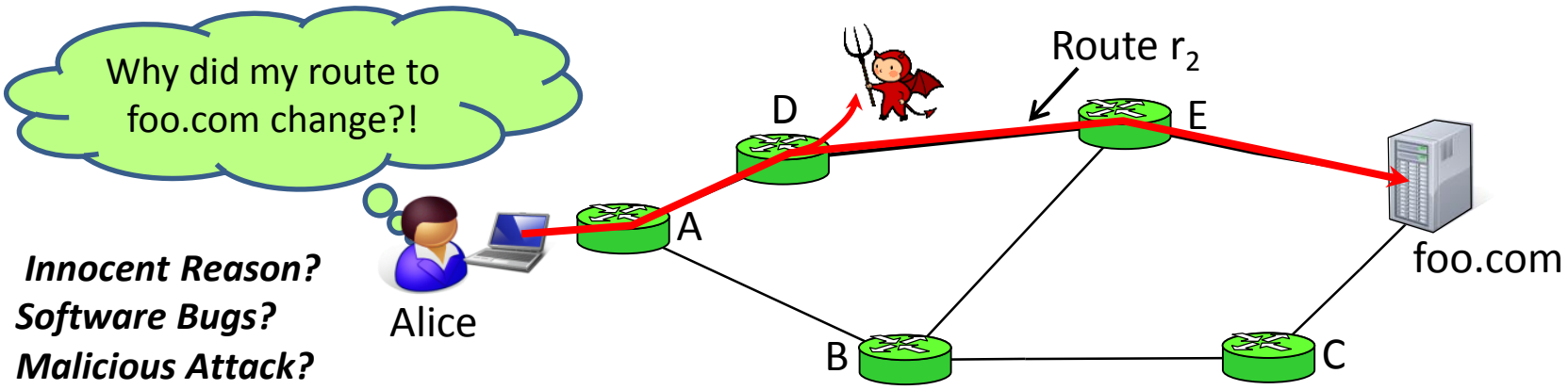

- **An example scenario: network routing**
  - System administrator observes strange behavior
  - Example: the route to foo.com has suddenly changed
  - Anomalies in distributed systems
    - **Need a way to explain system behavior.**

# Motivation



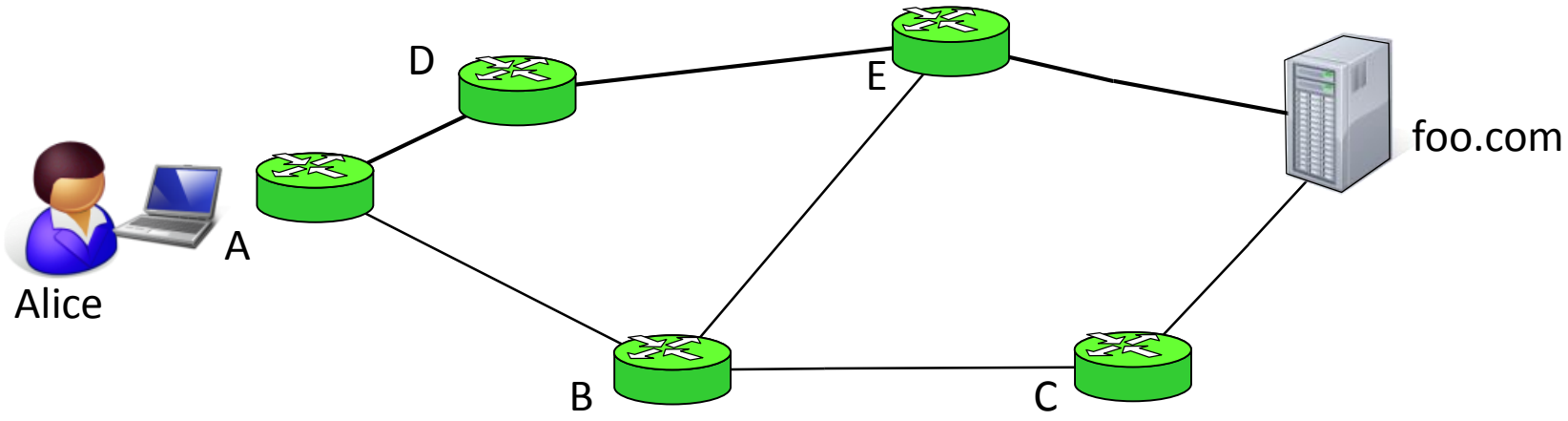

- **An example scenario: network routing**
  - System administrator observes strange behavior
  - Example: the route to foo.com has suddenly changed
  - Anomalies in distributed systems
    - **Need a way to explain system behavior.**

# Motivation



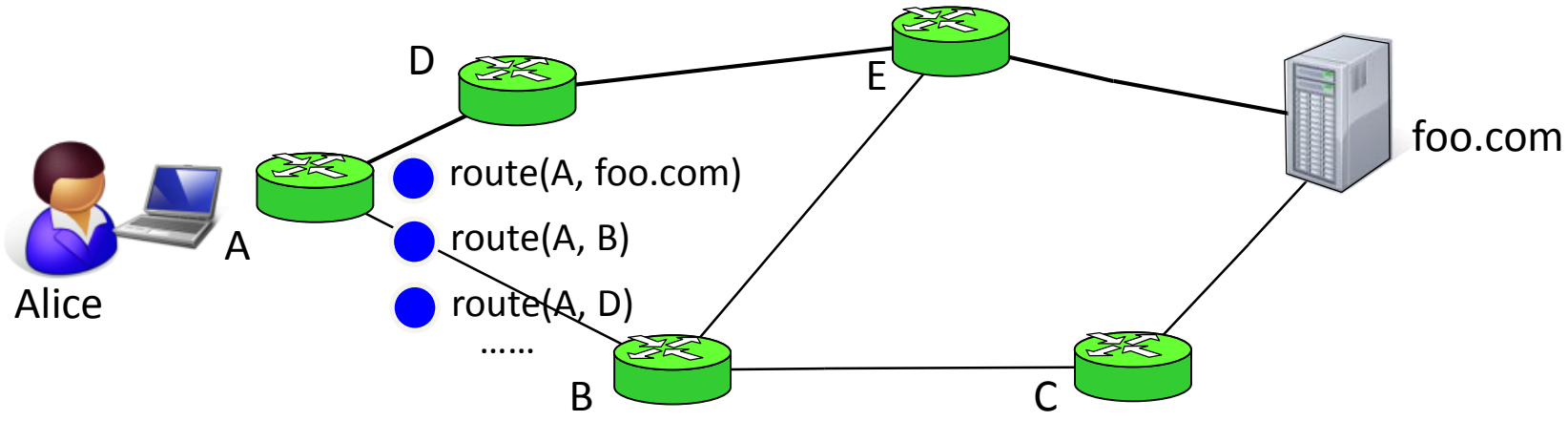

- **An example scenario: network routing**
  - System administrator observes strange behavior
  - Example: the route to foo.com has suddenly changed
  - Anomalies in distributed systems
    - **Need a way to explain system behavior.**

# Data-centric Perspective on Network Debugging



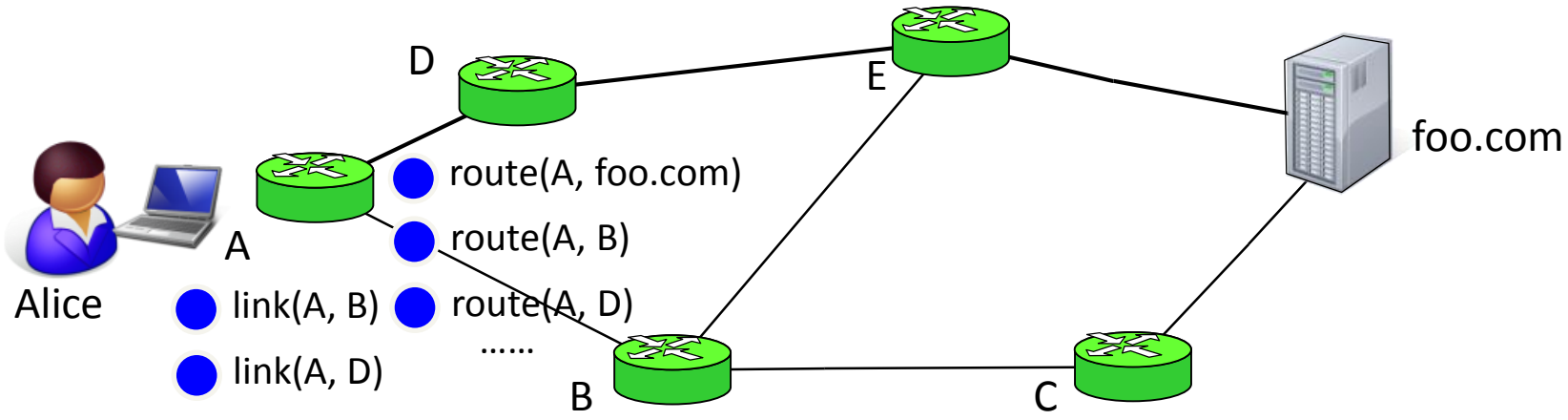

- **We assume a general distributed system**
  - Network consists of nodes (routers, middleboxes, ...)
  - The state of a node is a set of tuples (routes, config, ...)

# Data-centric Perspective on Network Debugging



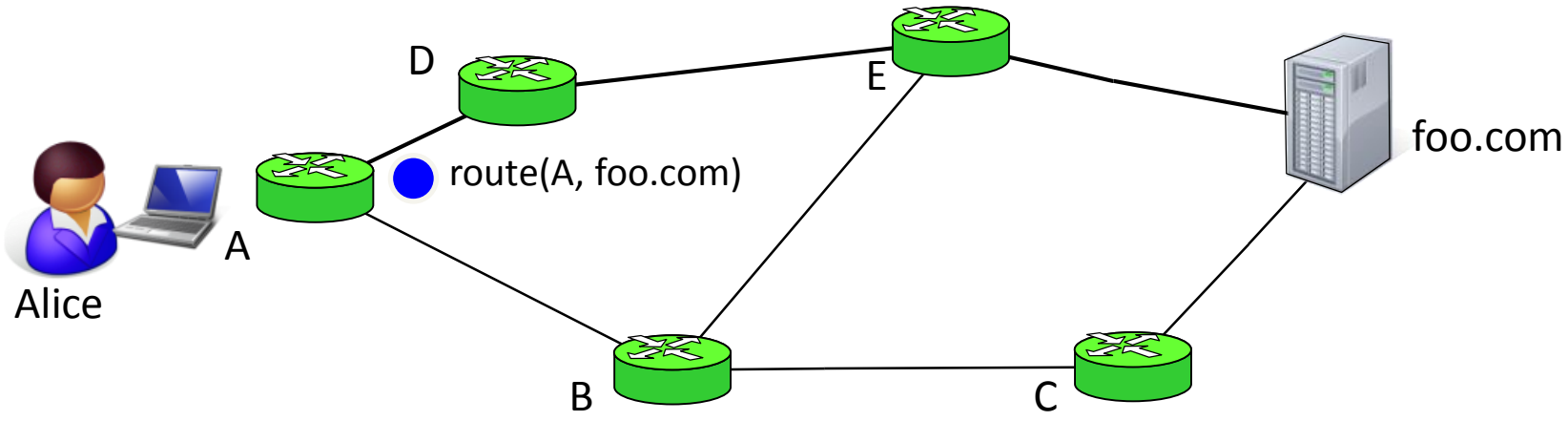

- **We assume a general distributed system**
  - Network consists of nodes (routers, middleboxes, ...)
  - The state of a node is a set of tuples (routes, config, ...)

# Data-centric Perspective on Network Debugging



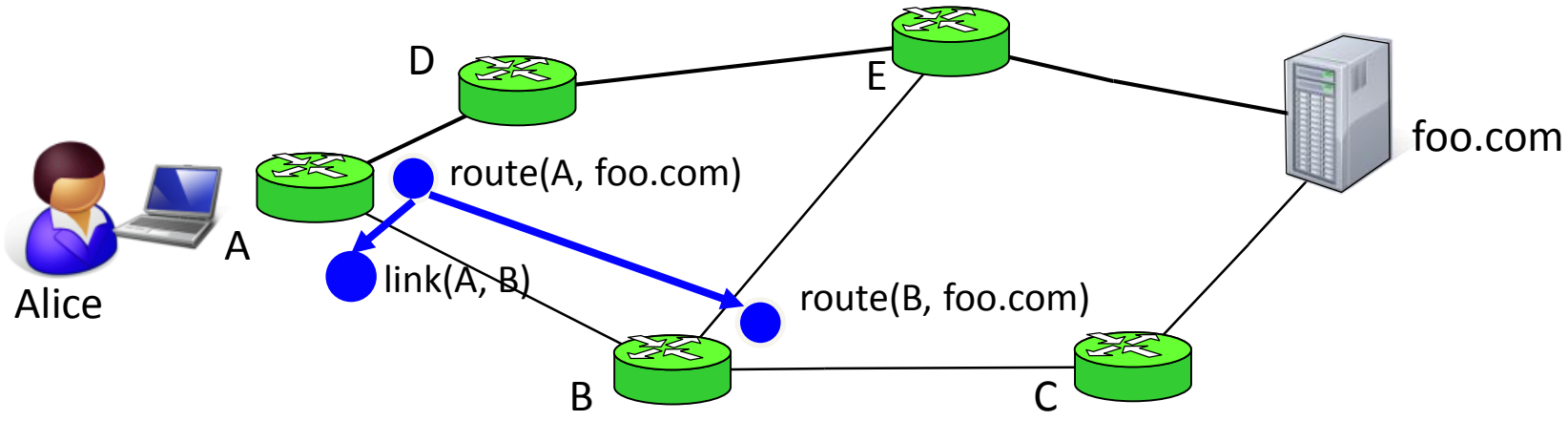

- **We assume a general distributed system**
  - Network consists of nodes (routers, middleboxes, ...)
  - The state of a node is a set of tuples (routes, config, ...)

# Data-centric Perspective on Network Debugging



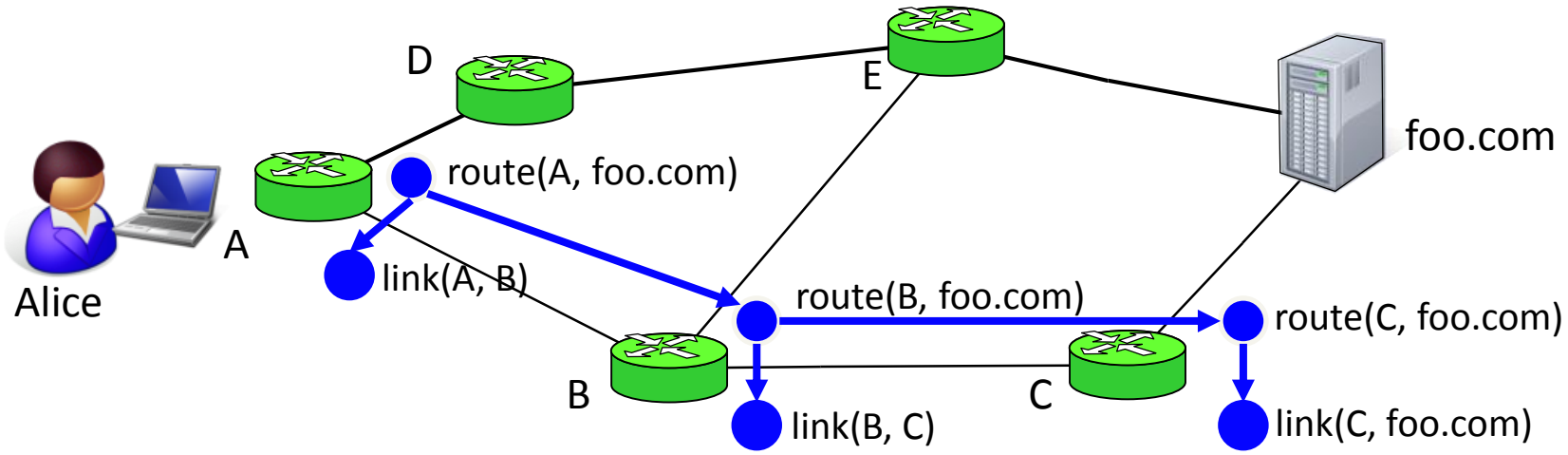

- **We assume a general distributed system**
  - Network consists of nodes (routers, middleboxes, ...)
  - The state of a node is a set of tuples (routes, config, ...)
  - **Idea**: Explanation as reasoning about distributed state dependencies

# Data-centric Perspective on Network Debugging




- **We assume a general distributed system**
  - Network consists of nodes (routers, middleboxes, ...)
  - The state of a node is a set of tuples (routes, config, ...)
  - **Idea**: Explanation as reasoning about distributed state dependencies

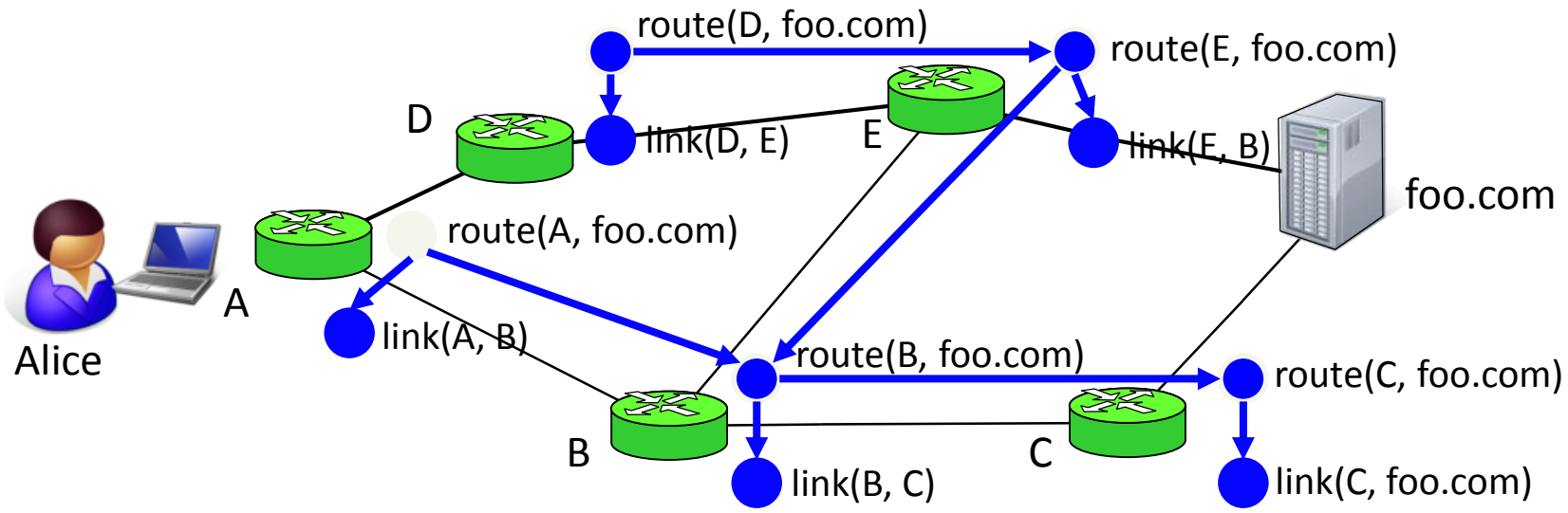
# Data-centric Perspective on Network Debugging




- **We assume a general distributed system**
  - Network consists of nodes (routers, middleboxes, ...)
  - The state of a node is a set of tuples (routes, config, ...)
  - **Idea**: Explanation as reasoning about distributed state dependencies

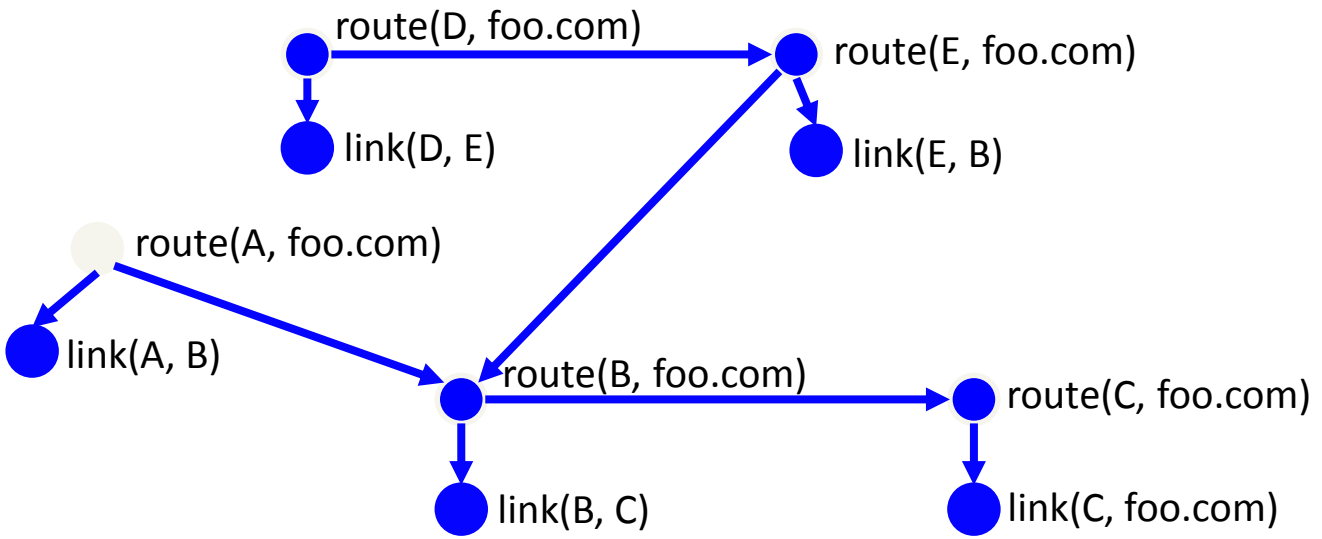
# Network Provenance

[SIGMOD 2010]



# Network Provenance

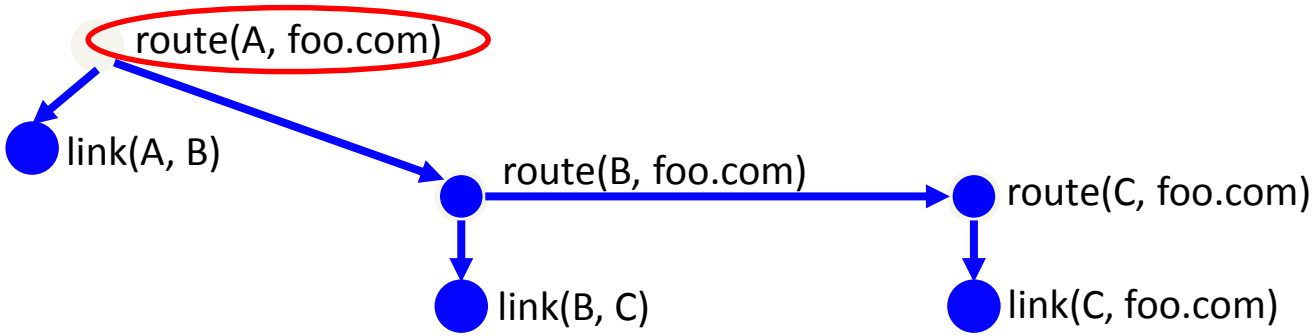
[SIGMOD 2010]




- **Provenance for encoding distributed state dependencies**
  - Explains the derivation of tuples
  - Captures the dependencies between tuples as a graph

# Network Provenance

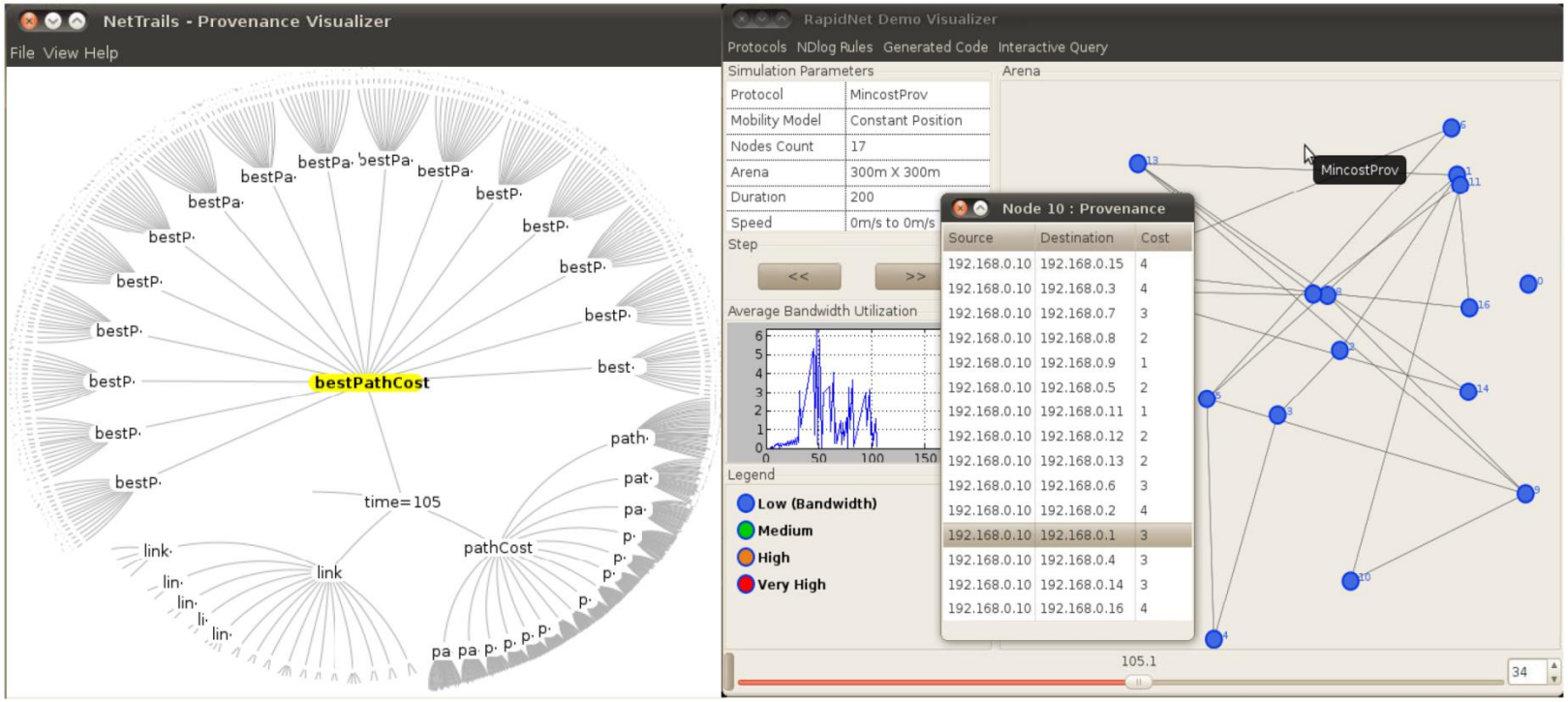
[SIGMOD 2010]




- **Provenance for encoding distributed state dependencies**
  - Explains the derivation of tuples
  - Captures the dependencies between tuples as a graph
  - Explanation of a tuple is an acyclic graph rooted at the tuple

# NetTrails: First Generation Network Provenance Tool

- http://netdb.cis.upenn.edu/nettrails/ [SIGMOD 2011 demo]



# Network Provenance Research (2010 – 2017)

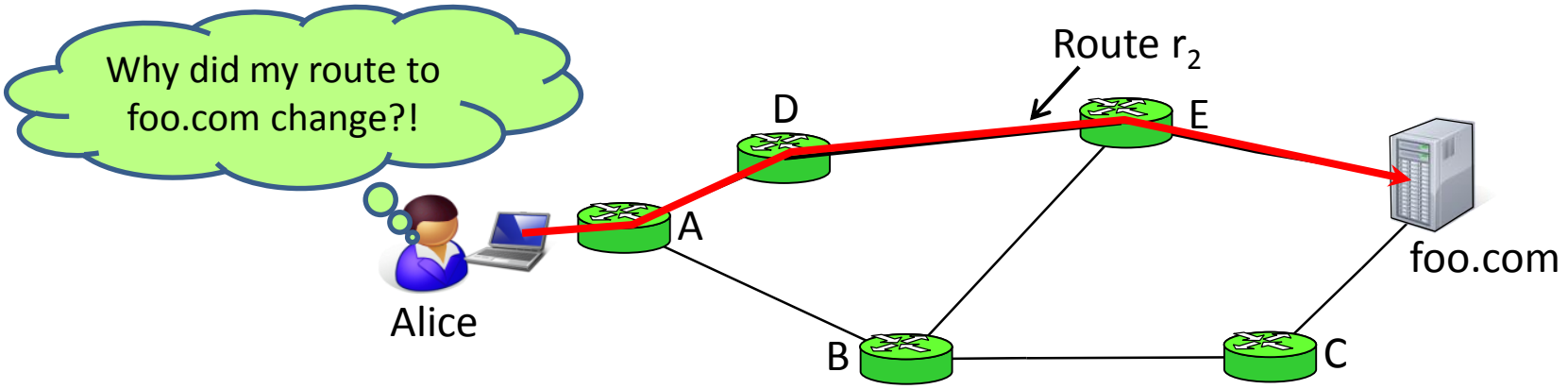
- Network provenance [SIGMOD'10]
- Secure network provenance [SOSP'11]
- Provenance in dynamic environments [VLDB'13]
- Negative provenance [SIGCOMM'14]
- Distributed provenance compression [SIGMOD'17]
- Differential provenance [SIGCOMM'16]
- Meta-provenance [NSDI'17]

Explanations

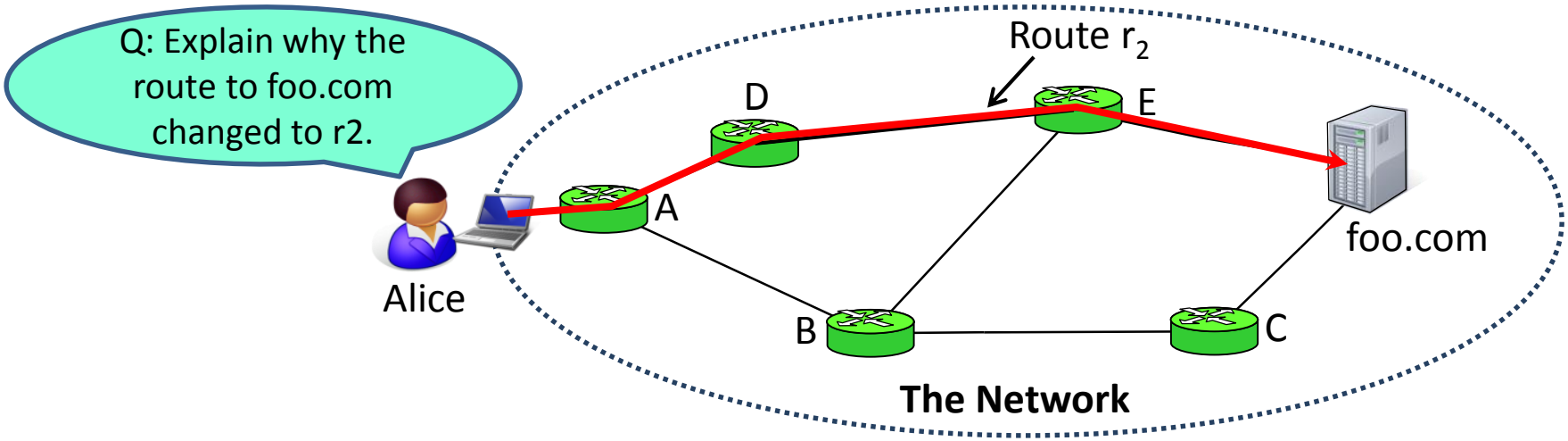
Deeper diagnostics and repair

Ph.D. dissertation work of Ang Chen (2017), Chen Chen (2017), Yang Wu (2017), and Wenchao Zhou (2012).

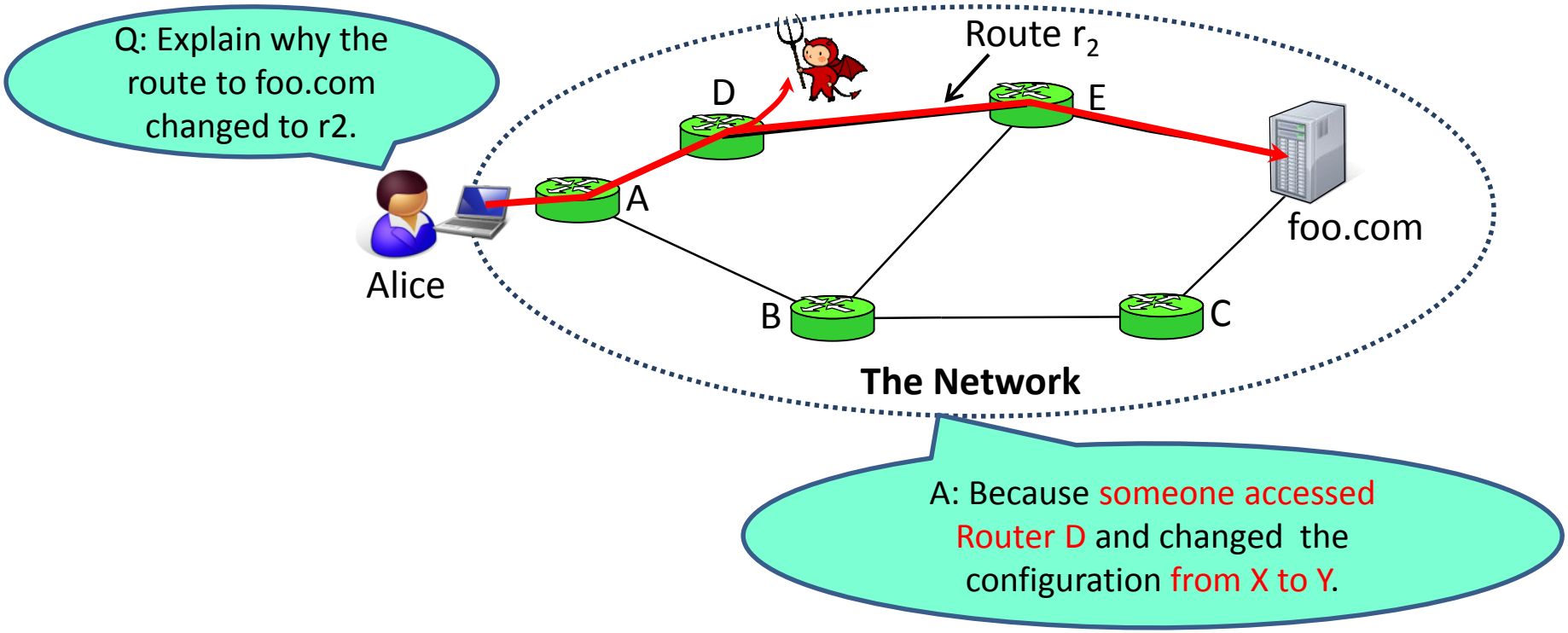
# Assumption #1: All nodes in the network can be trusted



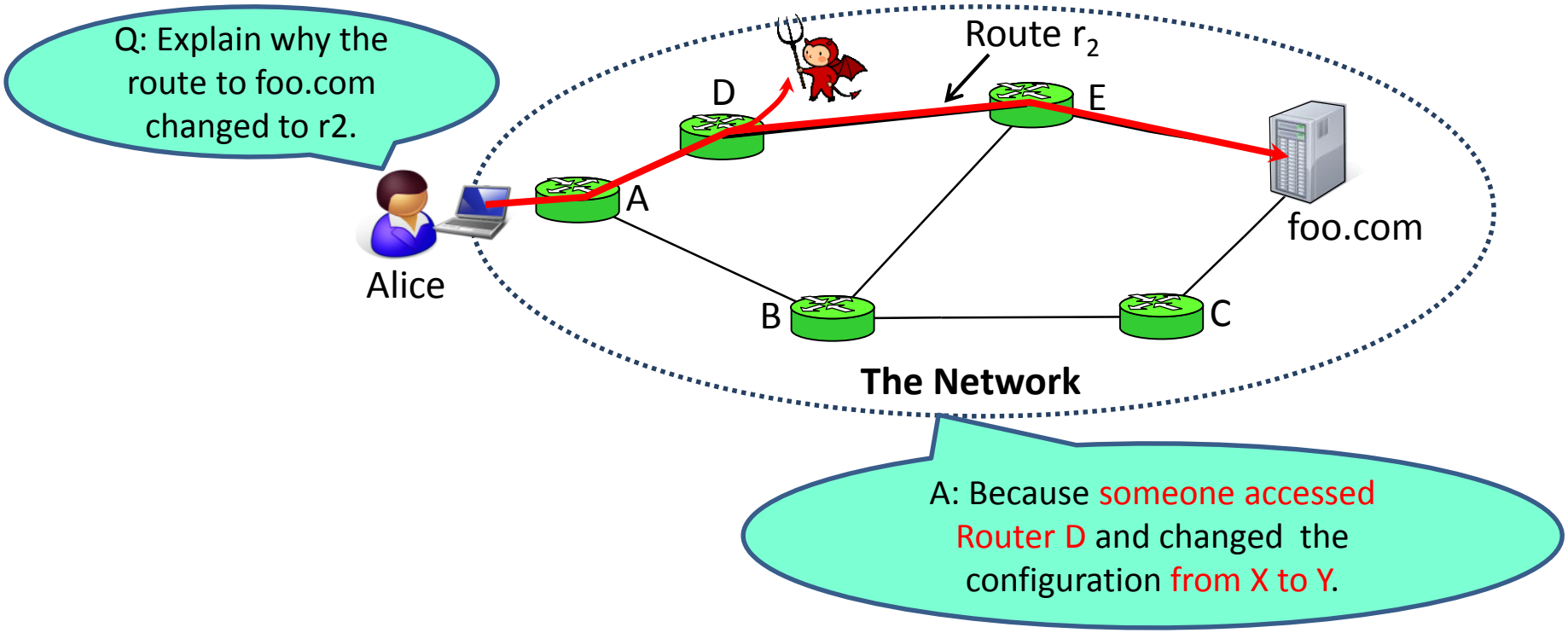
# Assumption #1: All nodes in the network can be trusted



# Assumption #1: All nodes in the network can be trusted

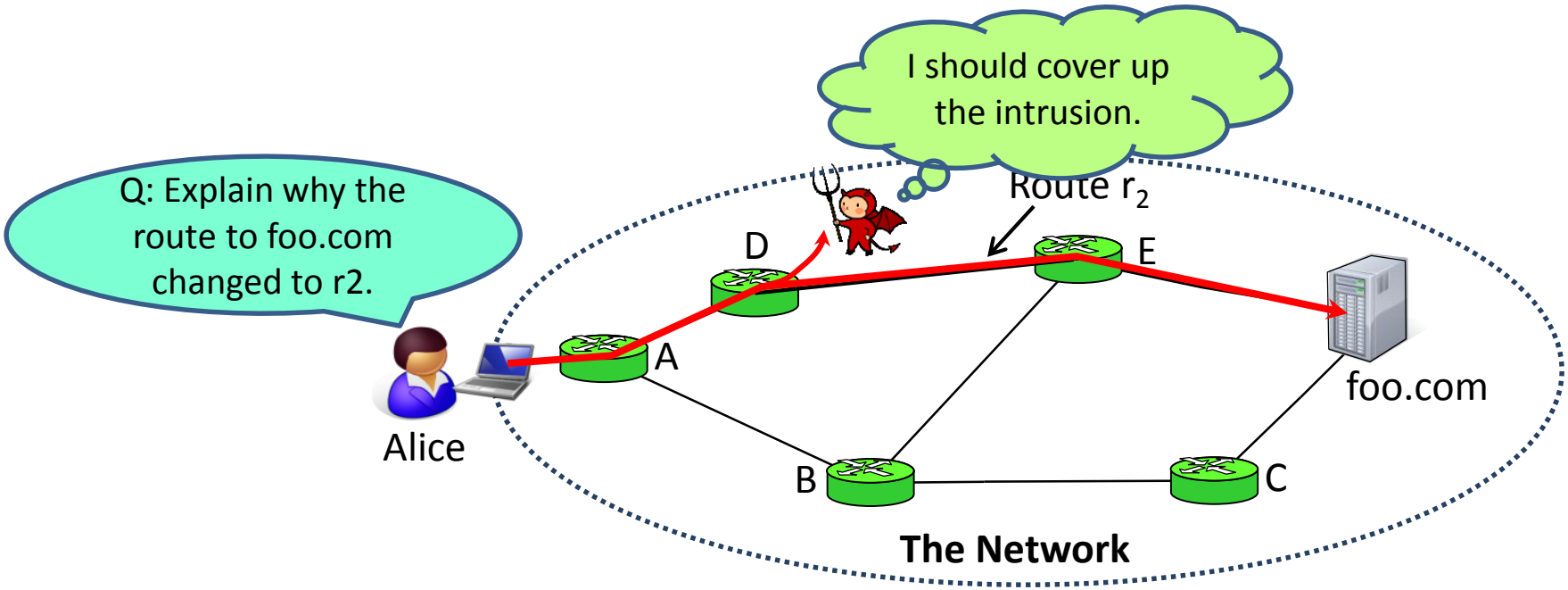


# Assumption #1: All nodes in the network can be trusted



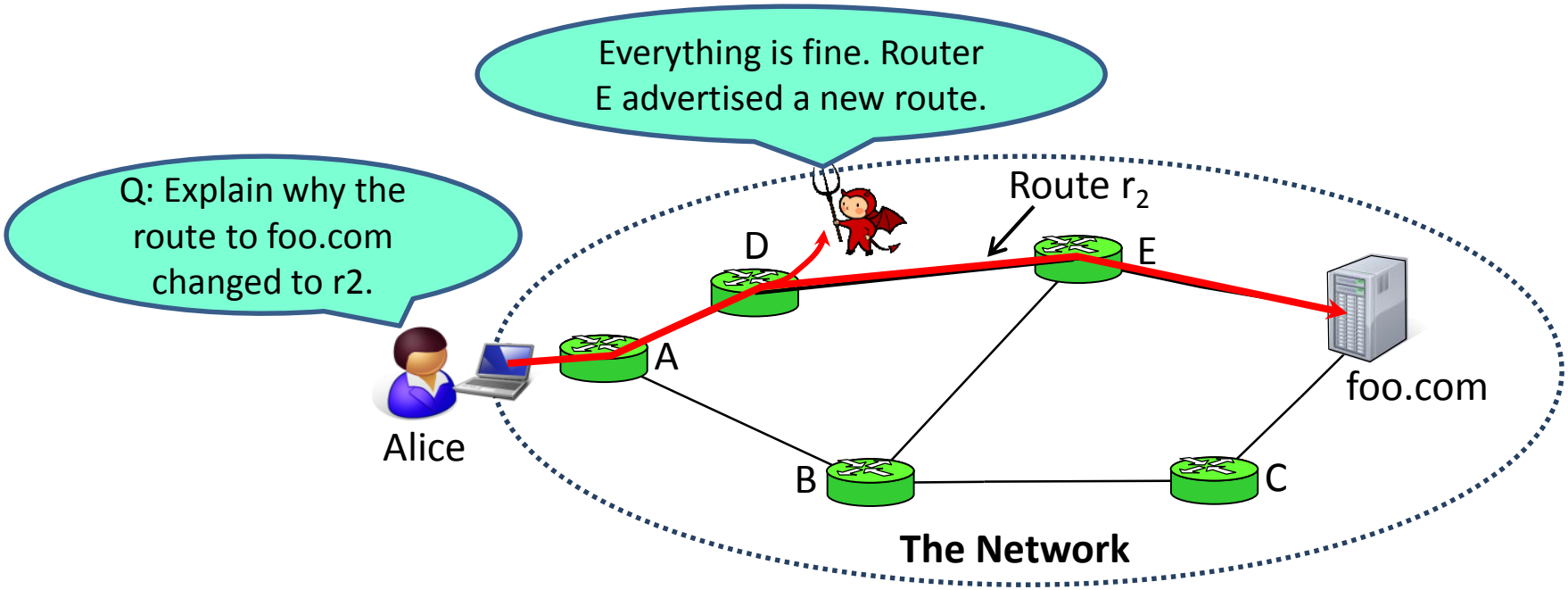

**Not realistic: adversary can tell lies**

# Challenge: Adversaries Can Lie




- **Problem: adversary can …**
  - … fabricate plausible (yet incorrect) response
  - … point accusation towards innocent nodes

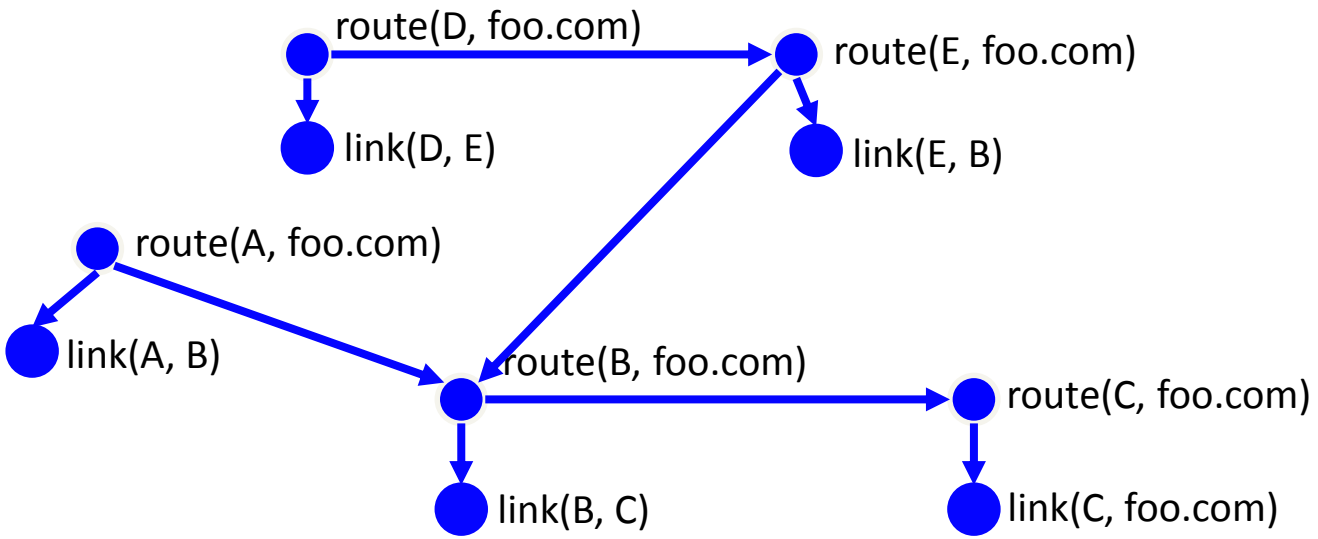
# Challenge: Adversaries Can Lie




- **Problem: adversary can …**
  - … fabricate plausible (yet incorrect) response
  - … point accusation towards innocent nodes

# Secure Network Provenance (SNP)

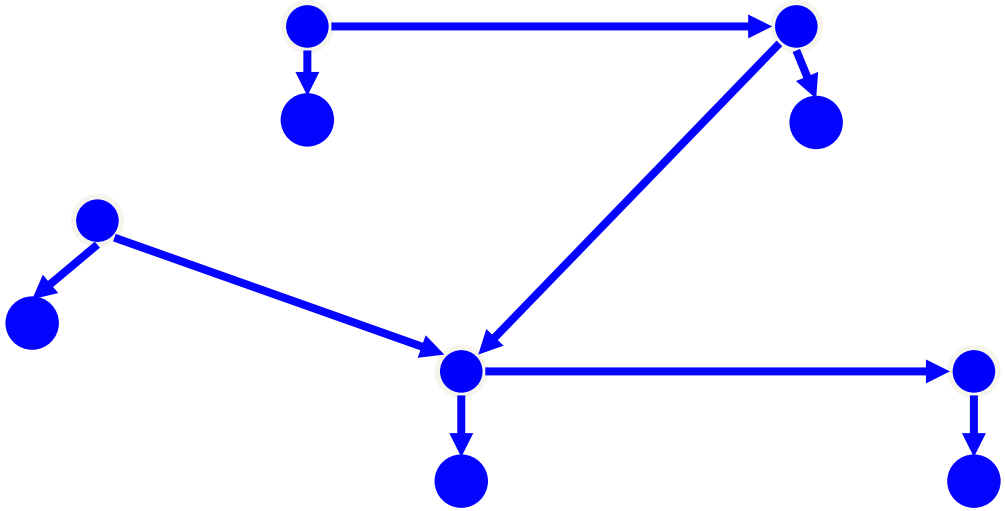
SOSP 2011




- **Step 1: Each node keeps vertices about local actions**
  - Split cross-node communications

# Secure Network Provenance (SNP)

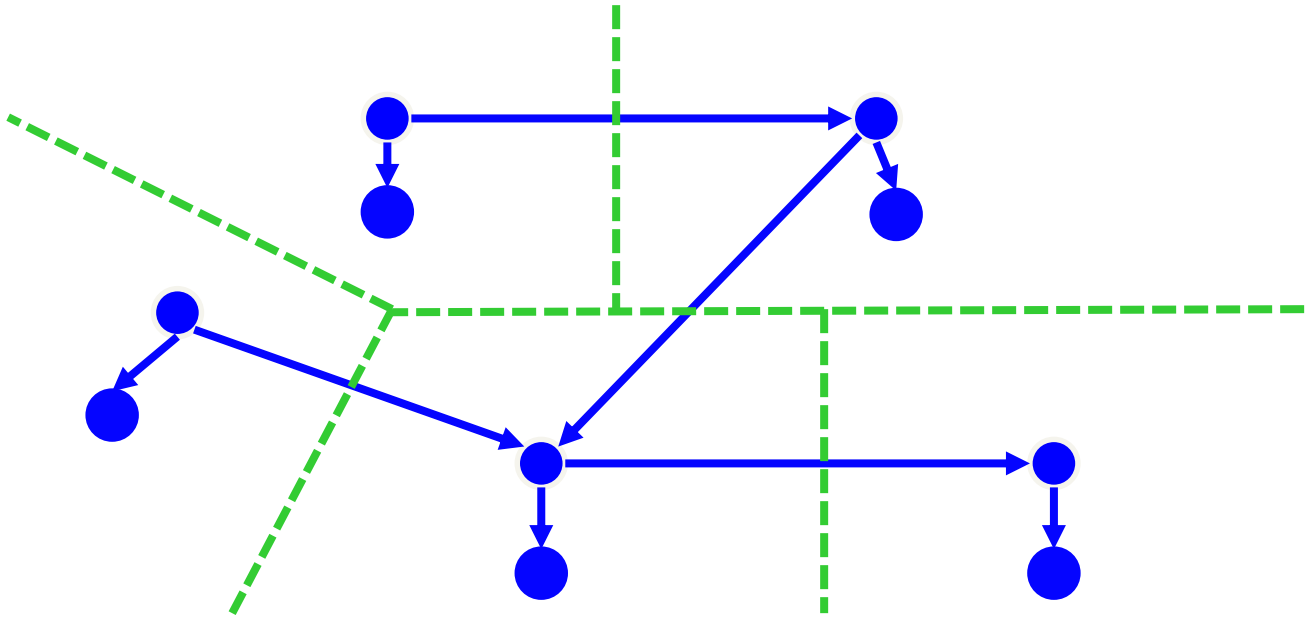
SOSP 2011



- **Step 1: Each node keeps vertices about local actions**
  - Split cross-node communications

# Secure Network Provenance (SNP)

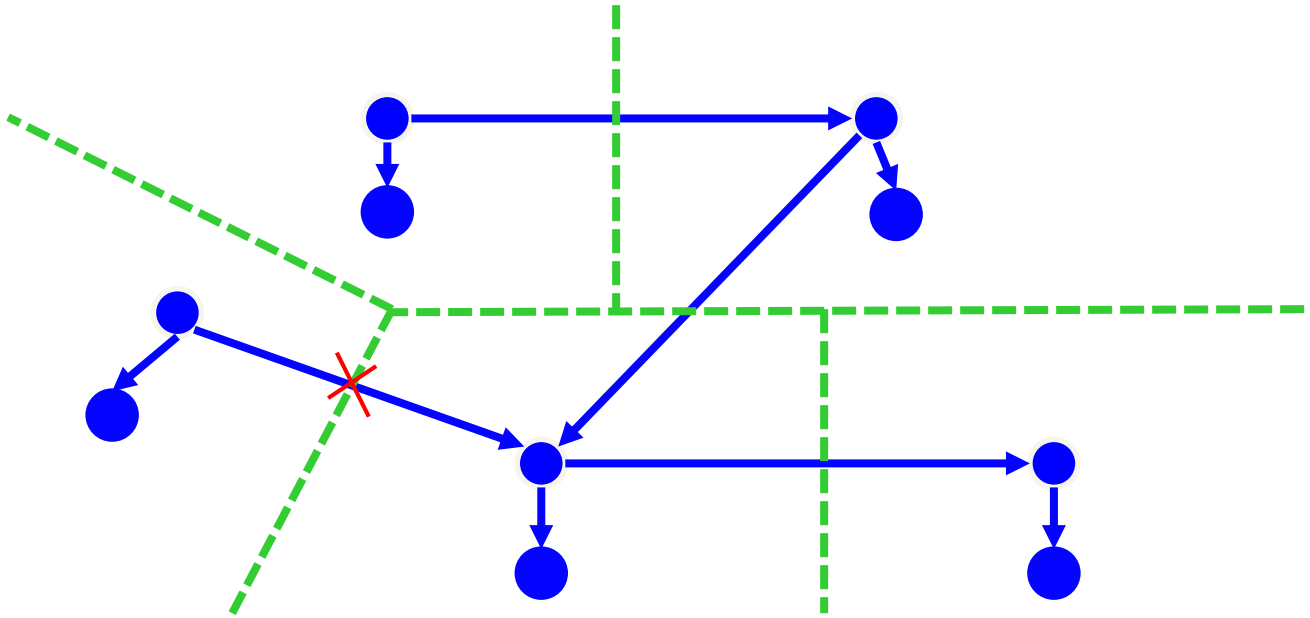
SOSP 2011



- **Step 1: Each node keeps vertices about local actions**
  - Split cross-node communications

# Secure Network Provenance (SNP)

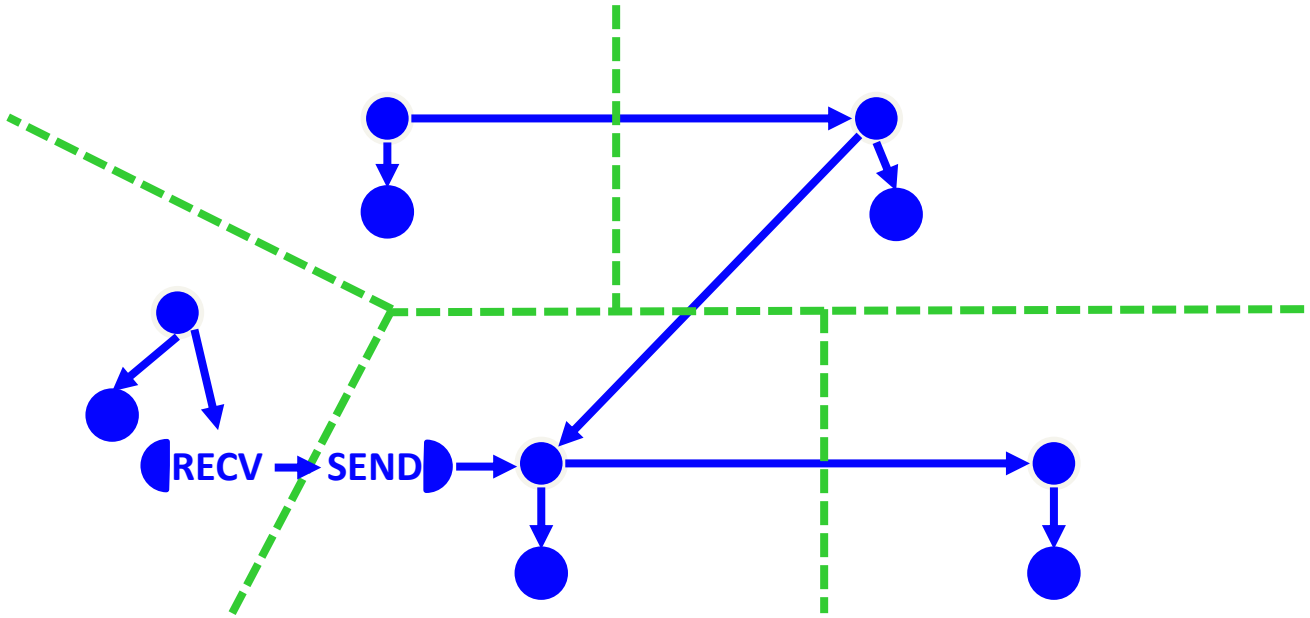
SOSP 2011



- **Step 1: Each node keeps vertices about local actions**
  - Split cross-node communications

# Secure Network Provenance (SNP)

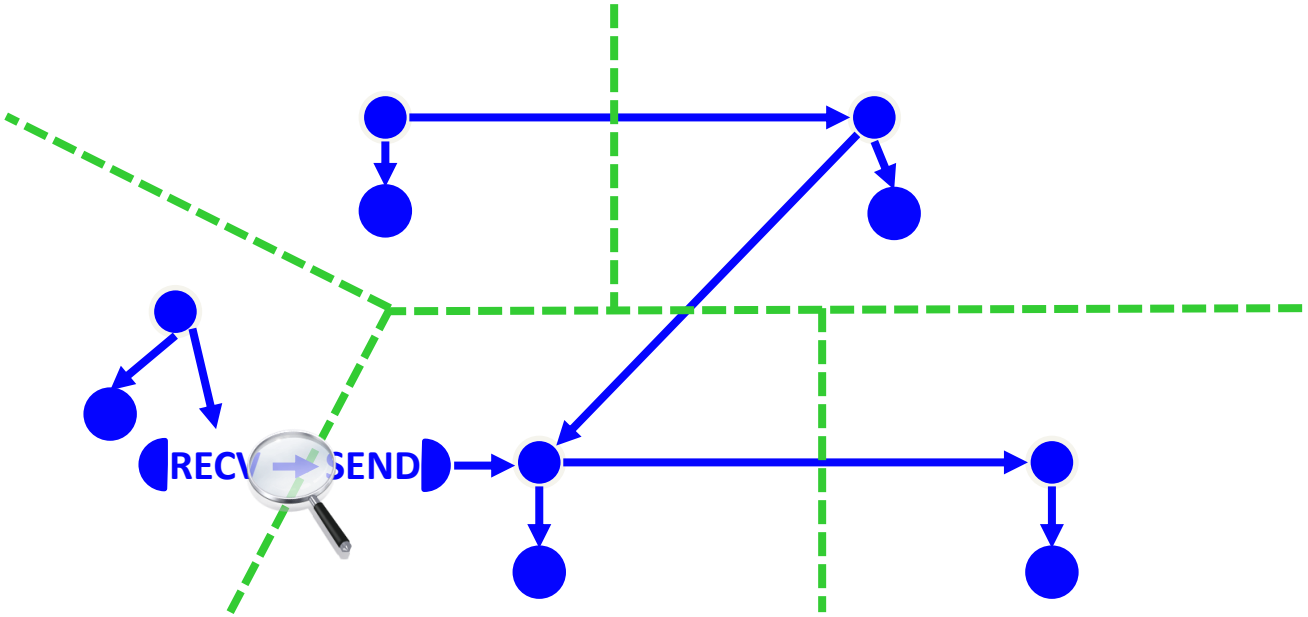
SOSP 2011




- **Step 1: Each node keeps vertices about local actions**
  - Split cross-node communications

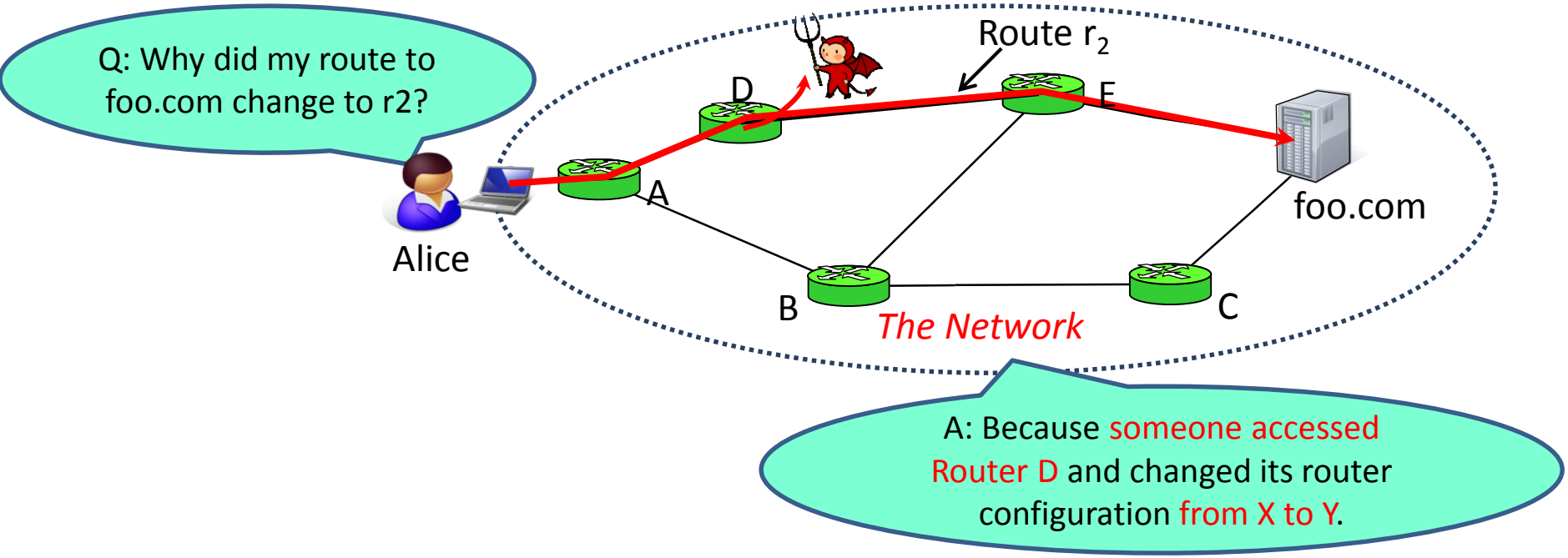
# Secure Network Provenance (SNP)

SOSP 2011



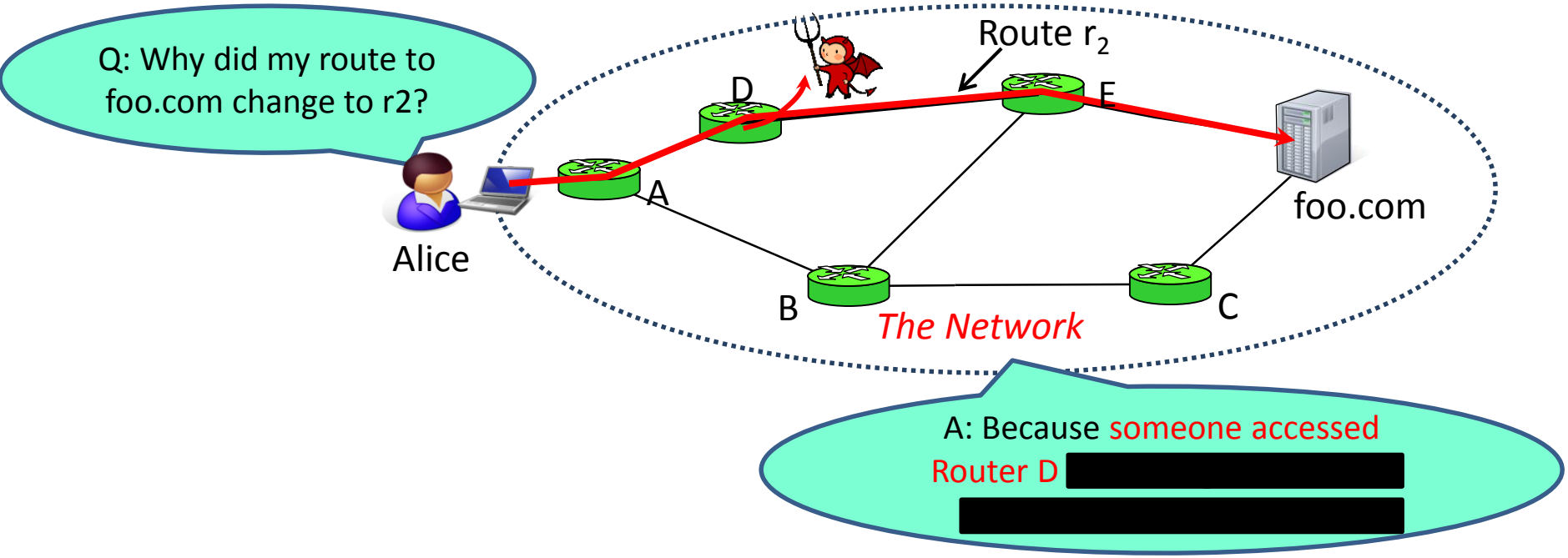

- **Step 1: Each node keeps vertices about local actions**
  - Split cross-node communications
- **Step 2: Make the graph *tamper-evident***

# SNP Guarantees



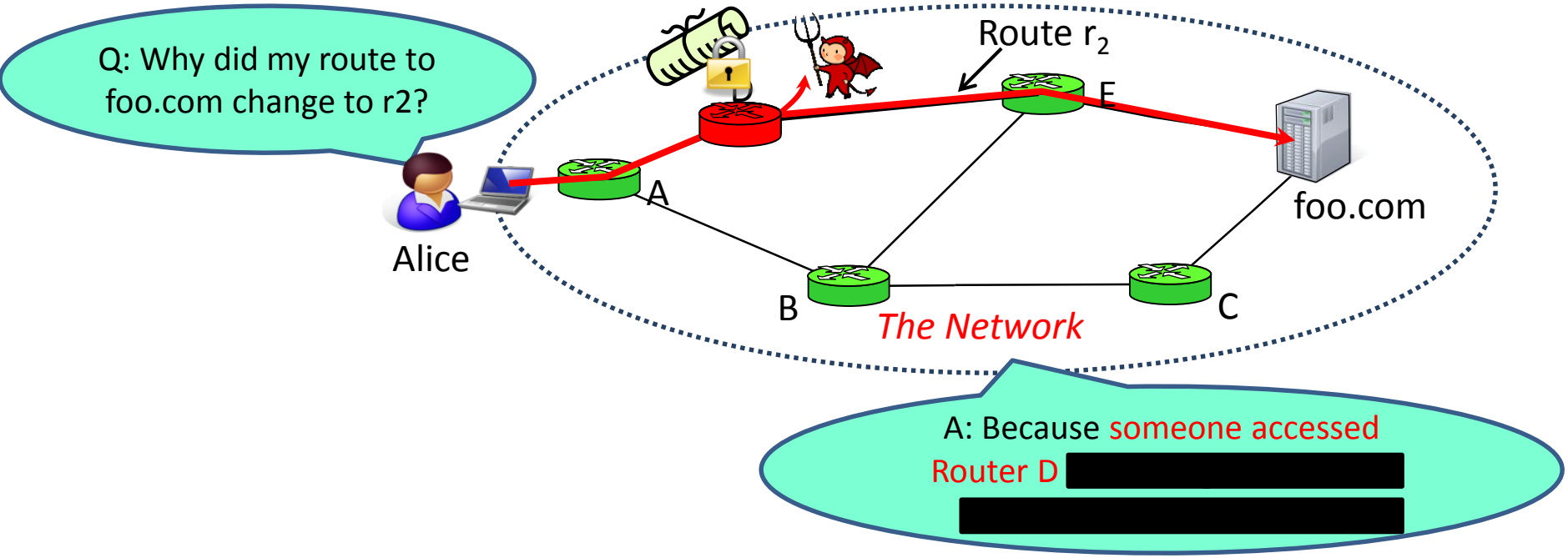

- **No faults: Explanation is complete and accurate**
- **Byzantine fault: Explanation identifies at least one faulty node**

# SNP Guarantees



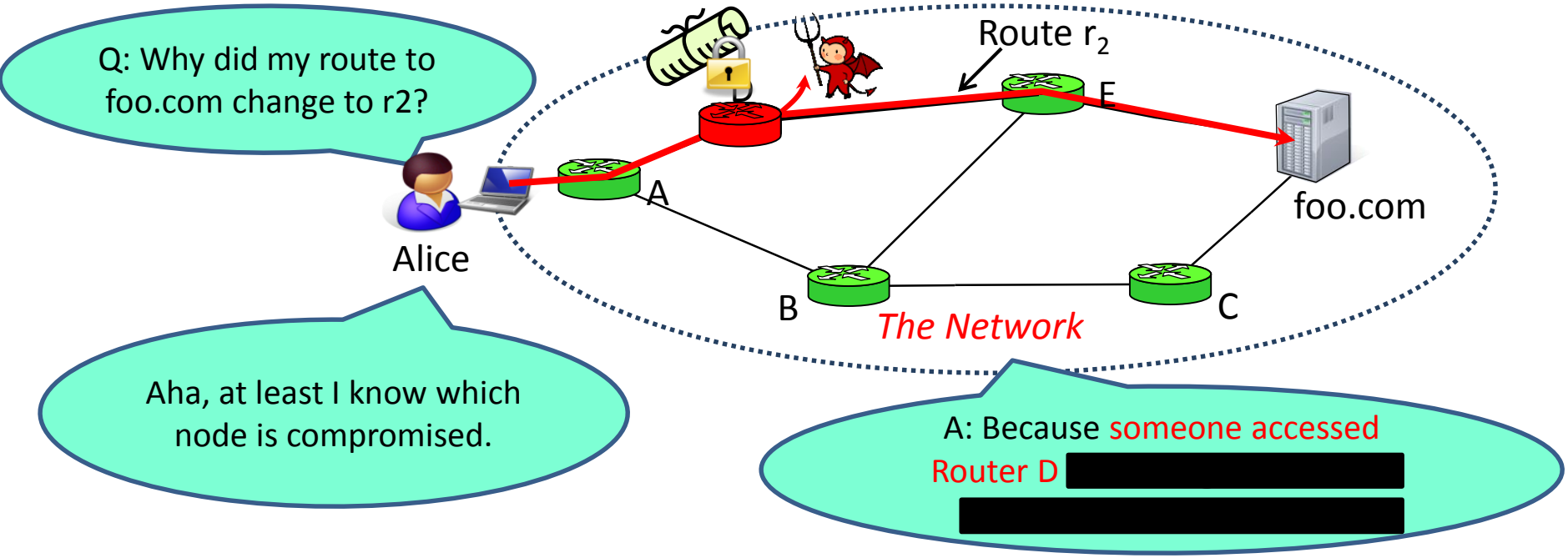

- **No faults: Explanation is complete and accurate**
- **Byzantine fault: Explanation identifies at least one faulty node**

# SNP Guarantees




- **No faults: Explanation is complete and accurate**
- **Byzantine fault: Explanation identifies at least one faulty node**

# SNP Guarantees




- **No faults: Explanation is complete and accurate**
- **Byzantine fault: Explanation identifies at least one faulty node**

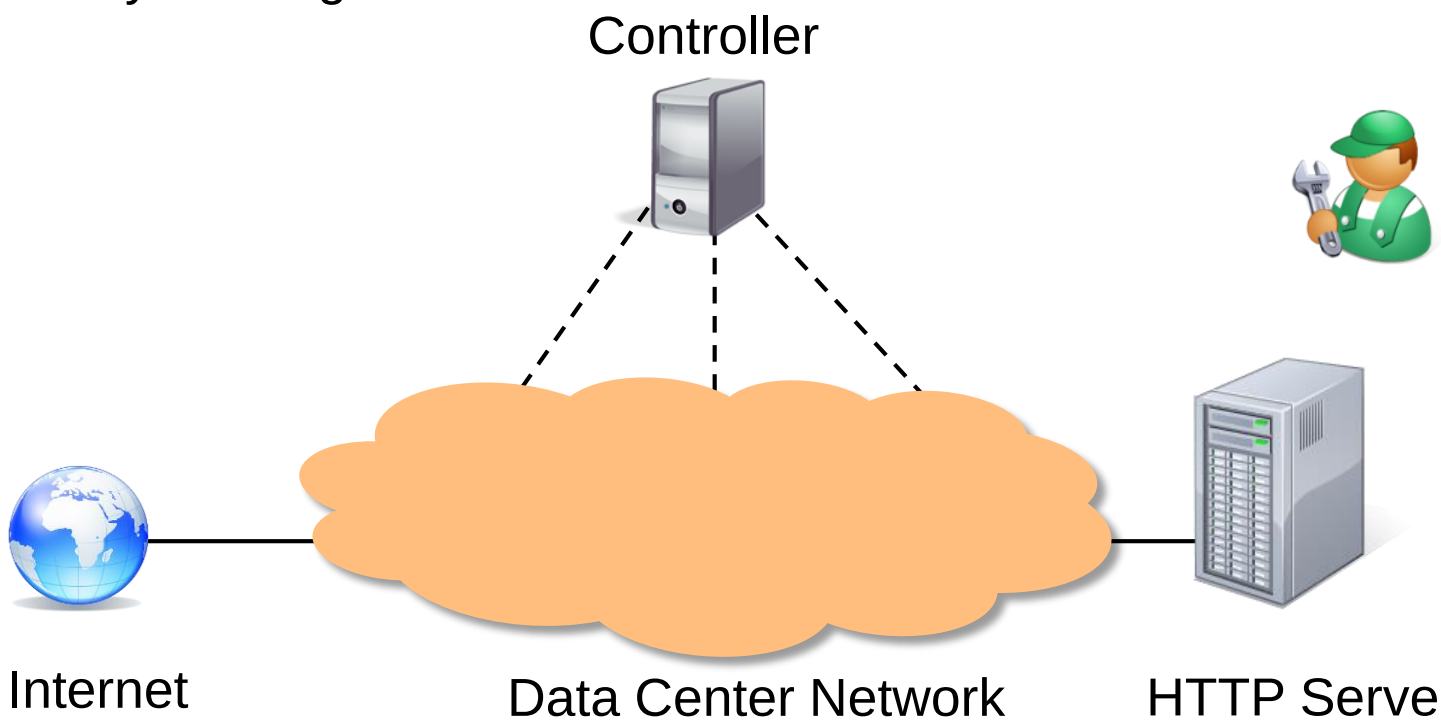
# Assumption #2: Operators react only to presence of anomaly events

# Assumption #2: Operators react only to presence of anomaly events

- What if something expected is not happening?
- Missing events cannot be handled by existing tools

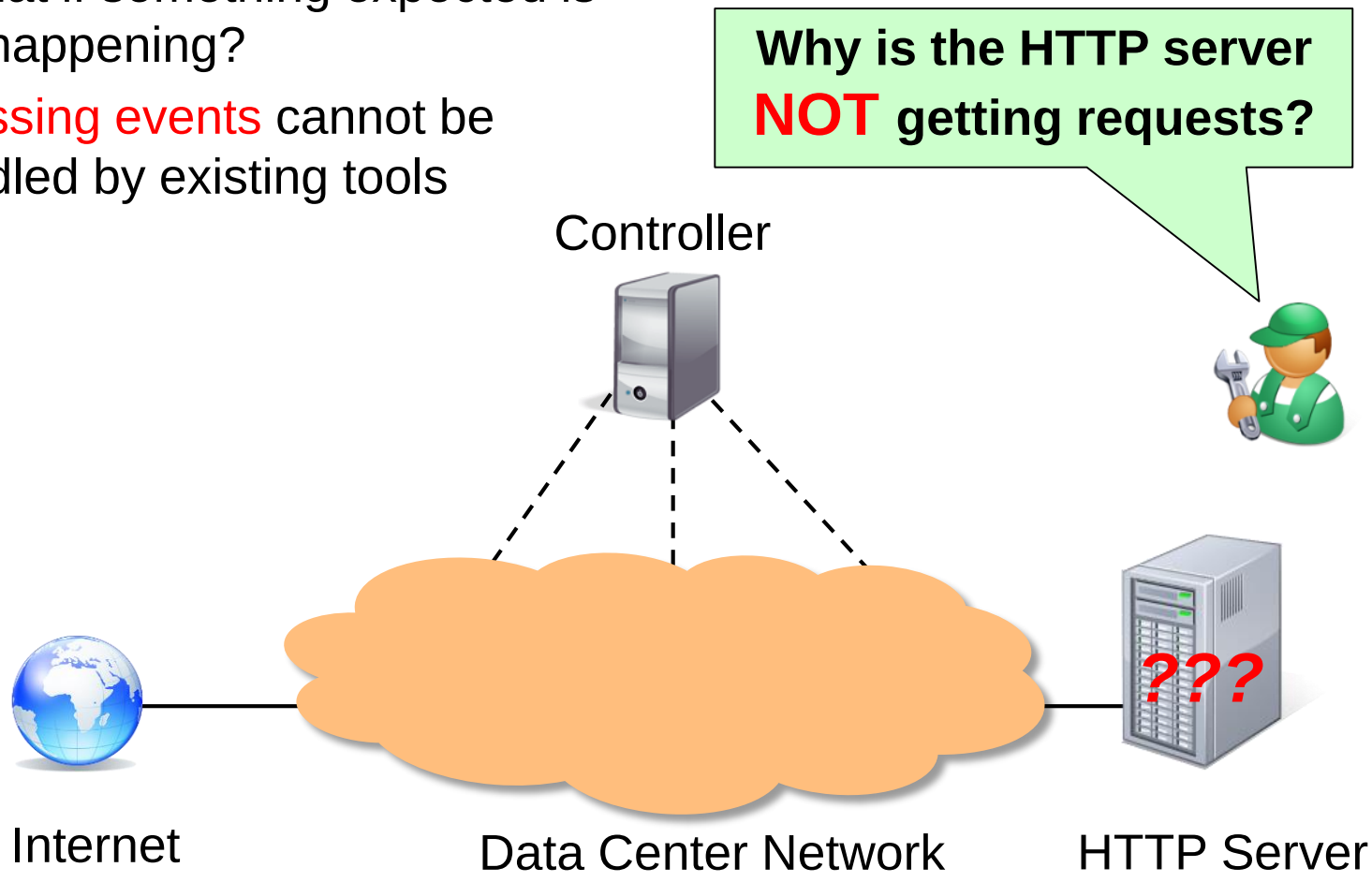
# Assumption #2: Operators react only to presence of anomaly events

- What if something expected is not happening?
- Missing events cannot be handled by existing tools



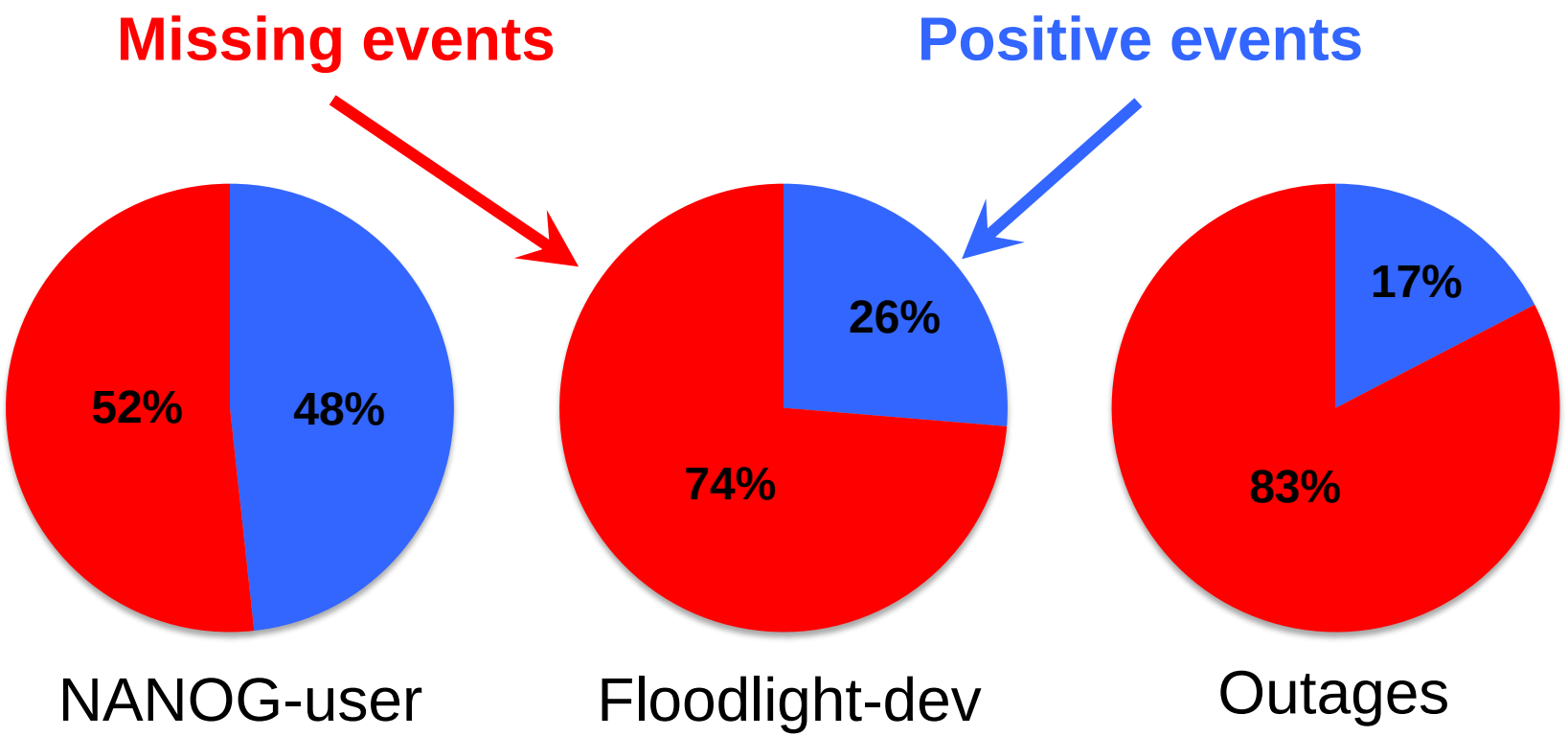
# Assumption #2: Operators react only to presence of anomaly events

- What if something expected is not happening?
- Missing events cannot be handled by existing tools



# How common are missing events?

- Missing events are consistently in the majority
- Lengthier email threads for missing events

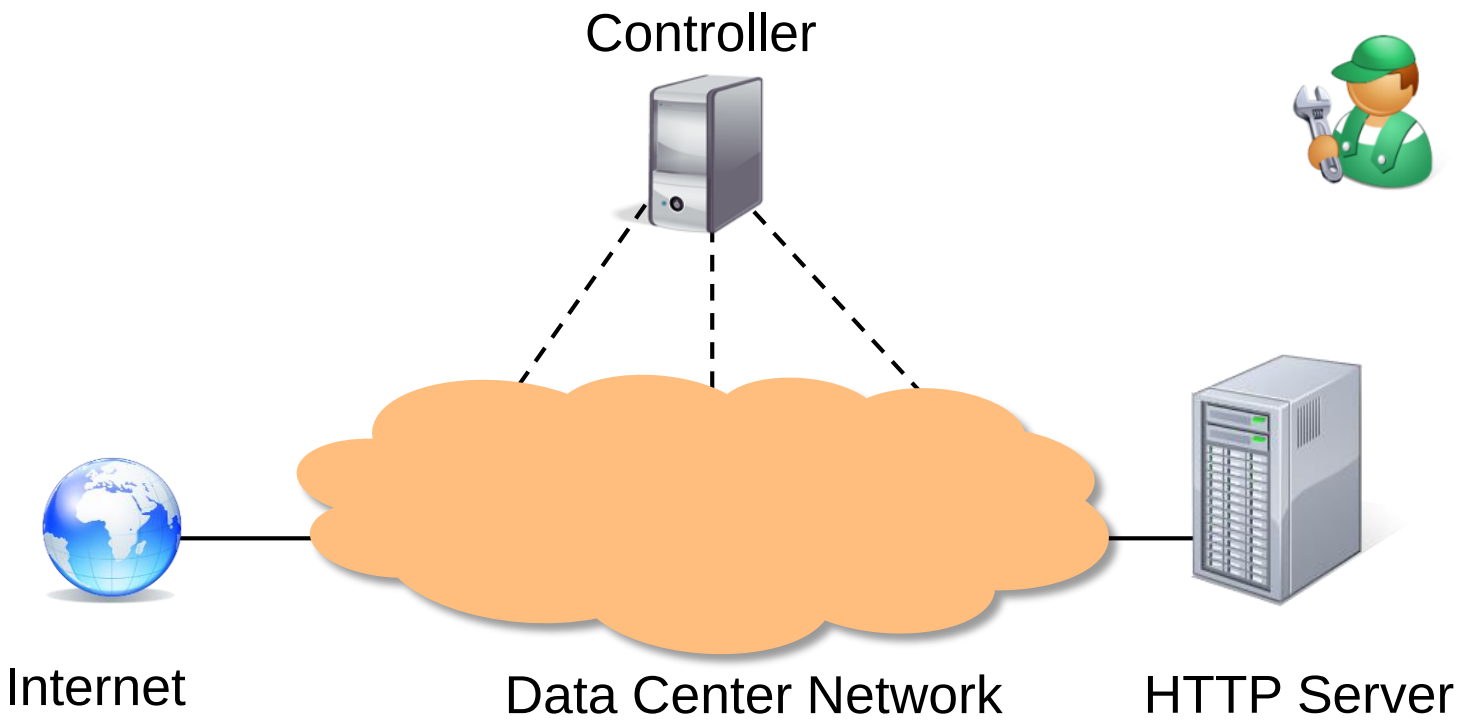


# Negative Provenance

[SIGCOMM 2014]

# Negative Provenance

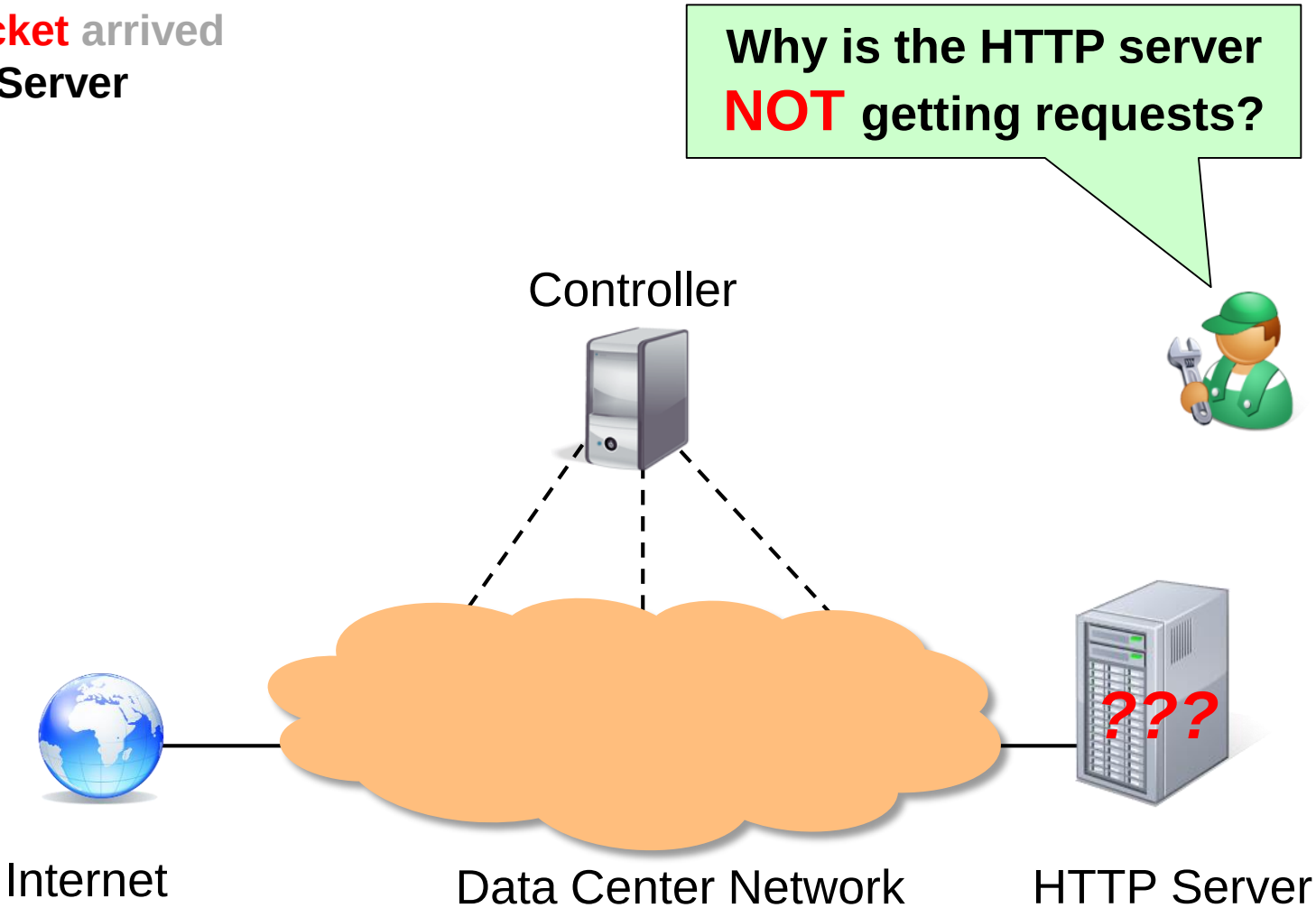
[SIGCOMM 2014]



# Negative Provenance

[SIGCOMM 2014]

**No HTTP Packet** arrived at **HTTP Server**



# Negative Provenance

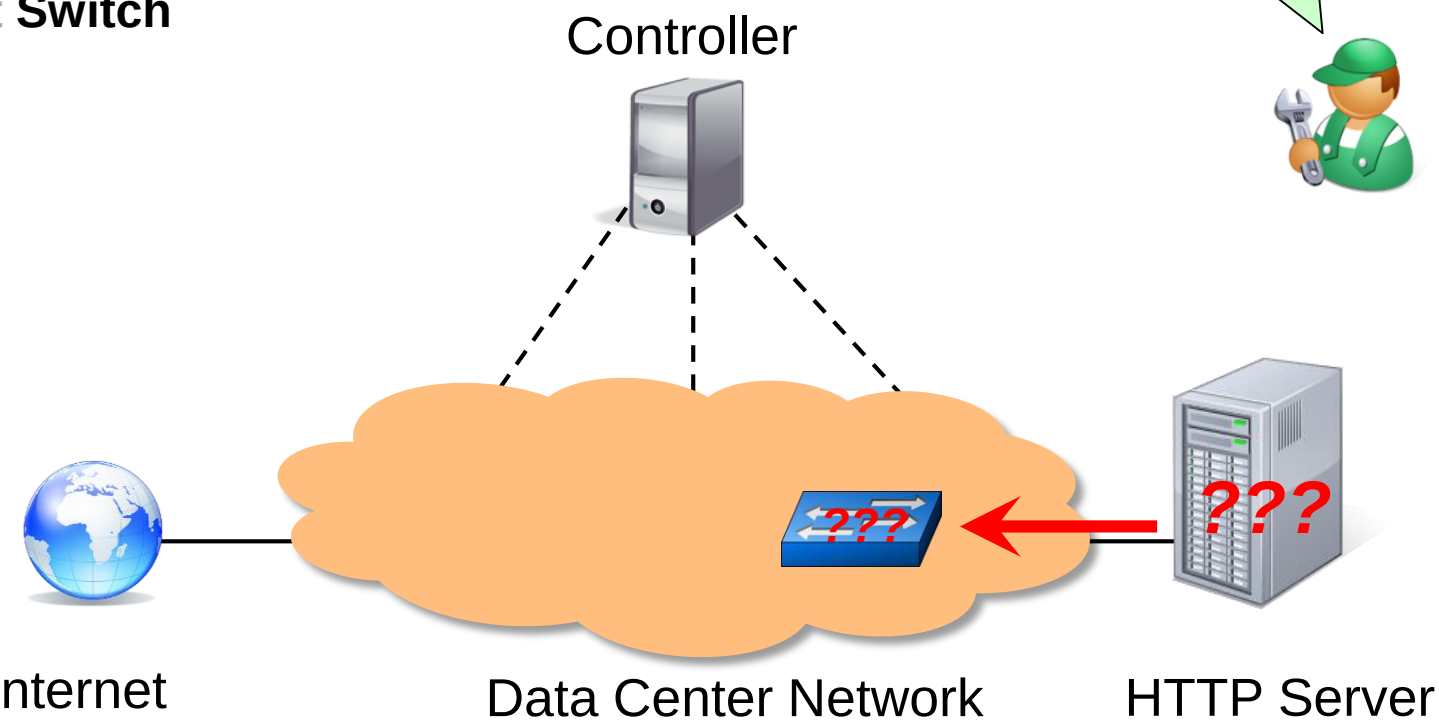
[SIGCOMM 2014]

**No HTTP Packet** arrived at **HTTP Server**

↑

**No Forwarding-FlowEntry** installed at **Switch**

**Why is the HTTP server NOT getting requests?**



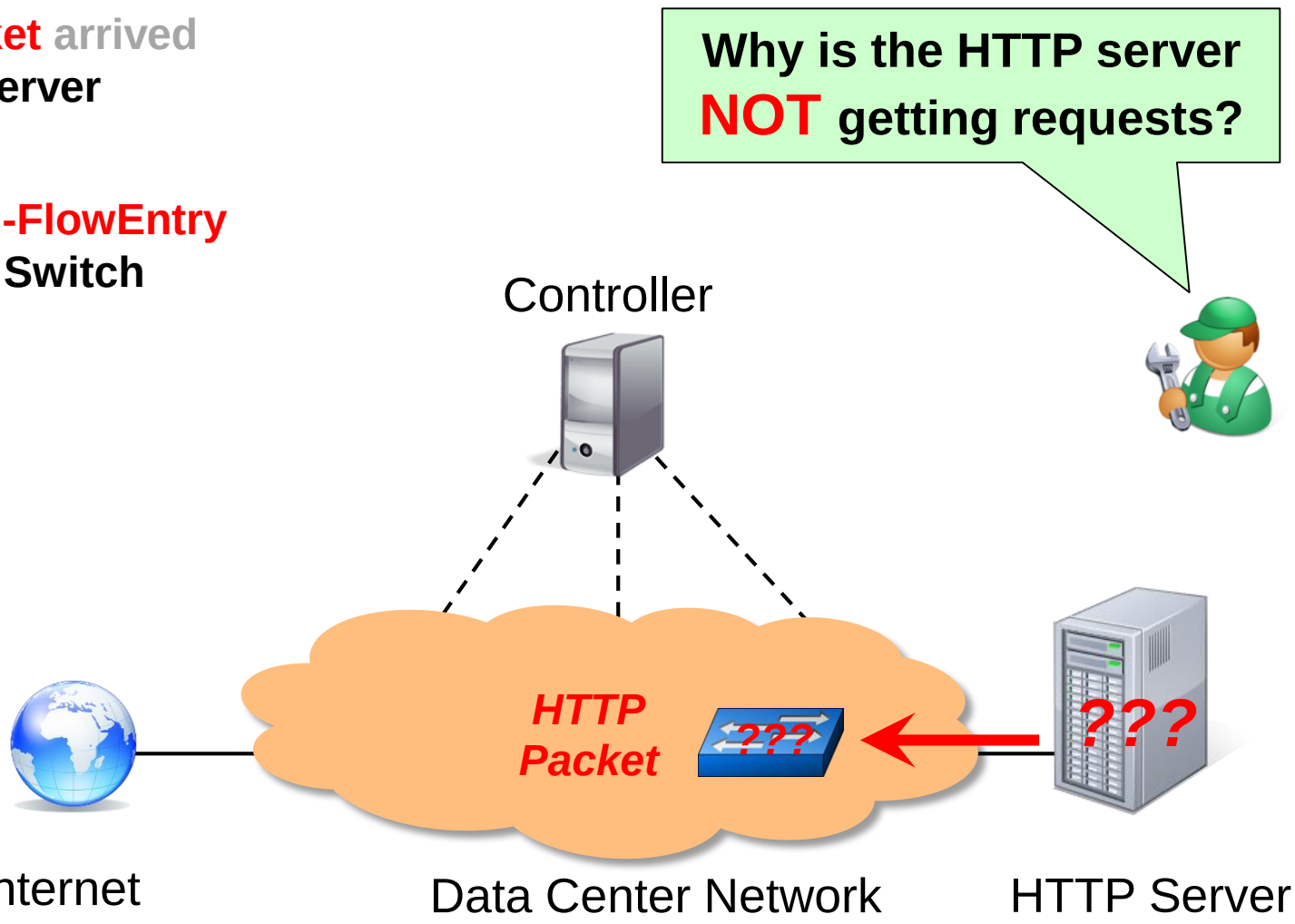
# Negative Provenance

[SIGCOMM 2014]

**No HTTP Packet** arrived at **HTTP Server**

**No Forwarding-FlowEntry** installed at **Switch**

**HTTP Packet** received at **Switch**



# Negative Provenance

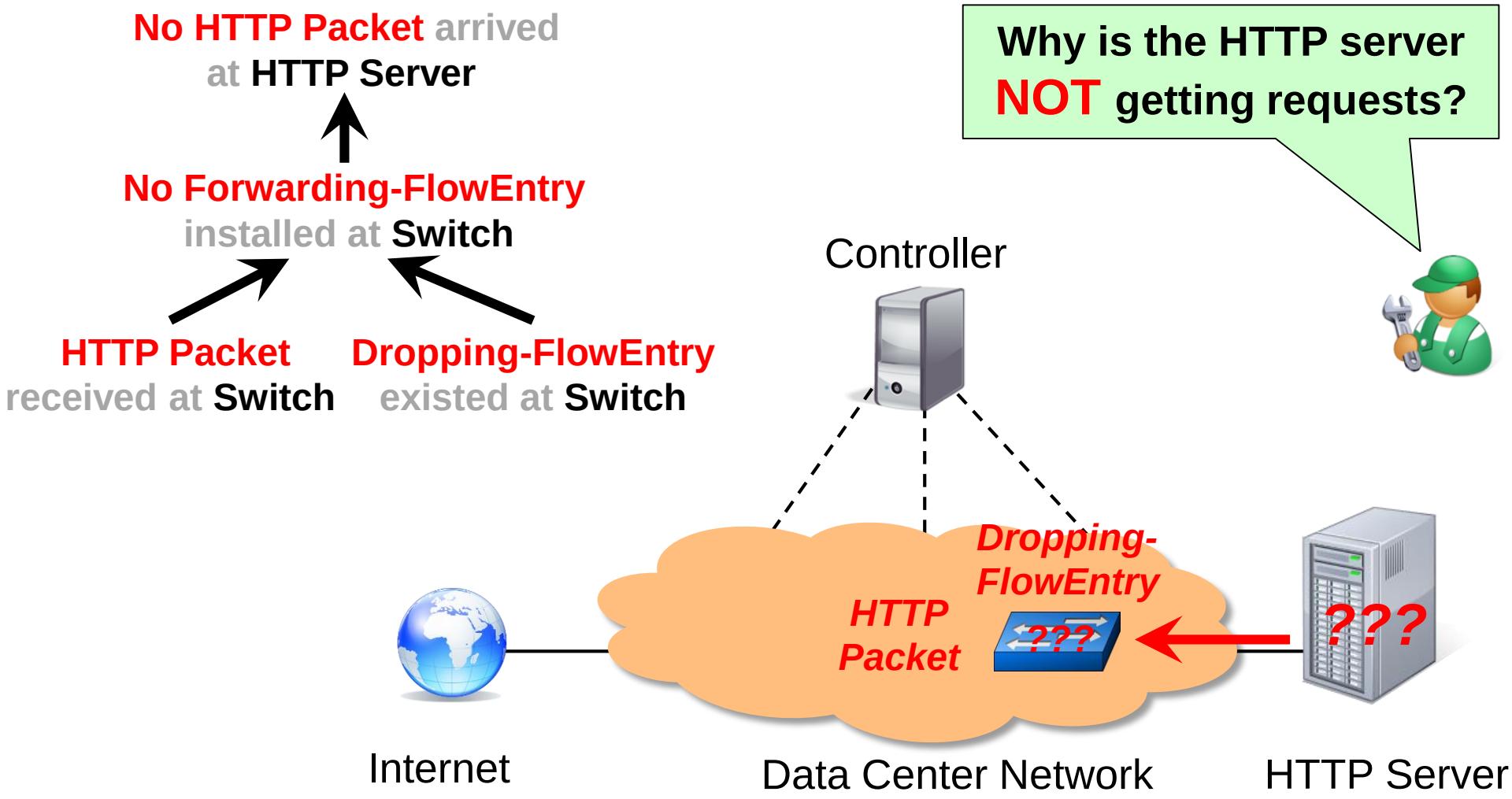
[SIGCOMM 2014]

**No HTTP Packet** arrived at **HTTP Server**

**No Forwarding-FlowEntry** installed at **Switch**

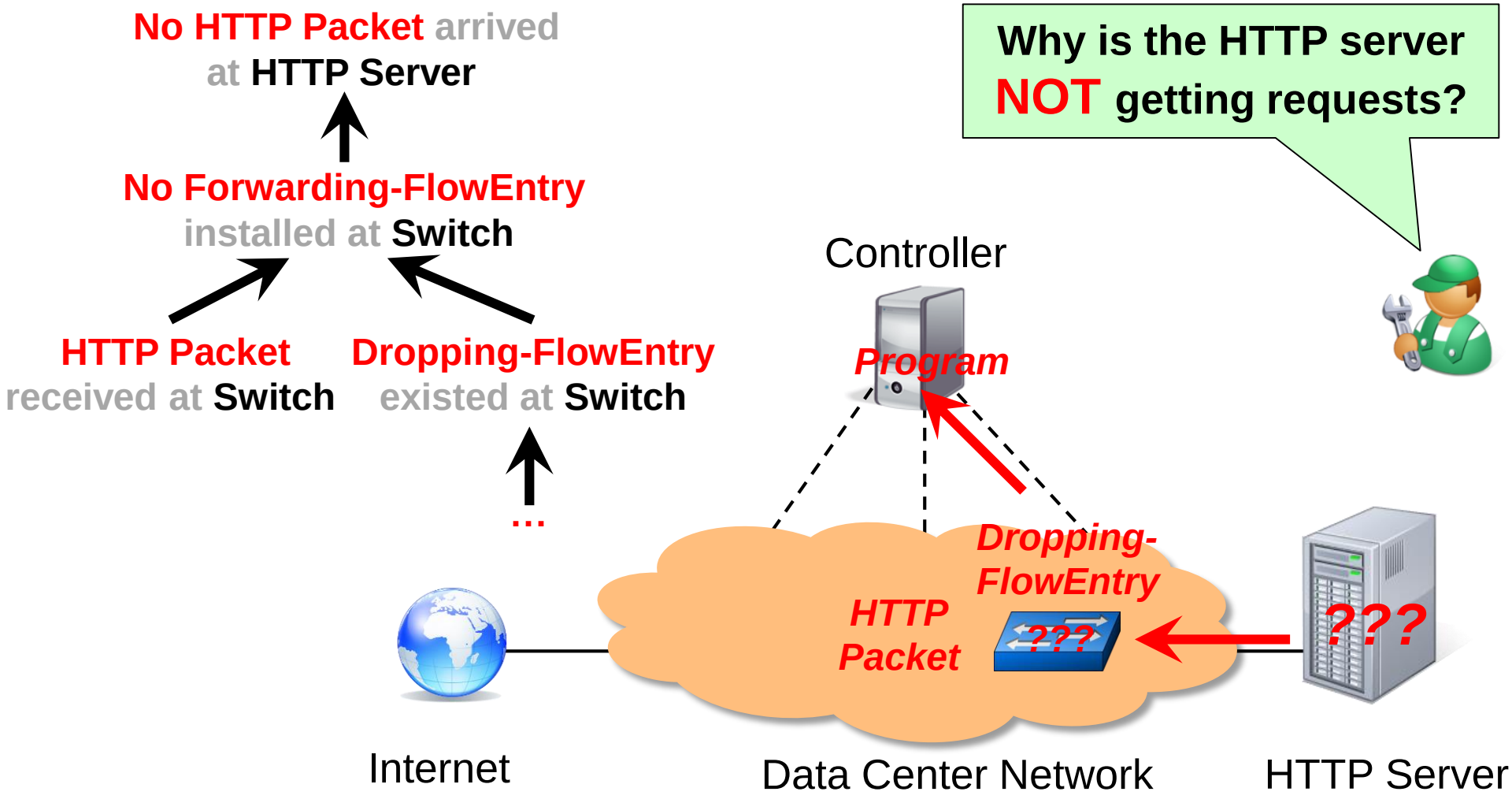
**HTTP Packet** received at **Switch**

**Dropping-FlowEntry** existed at **Switch**



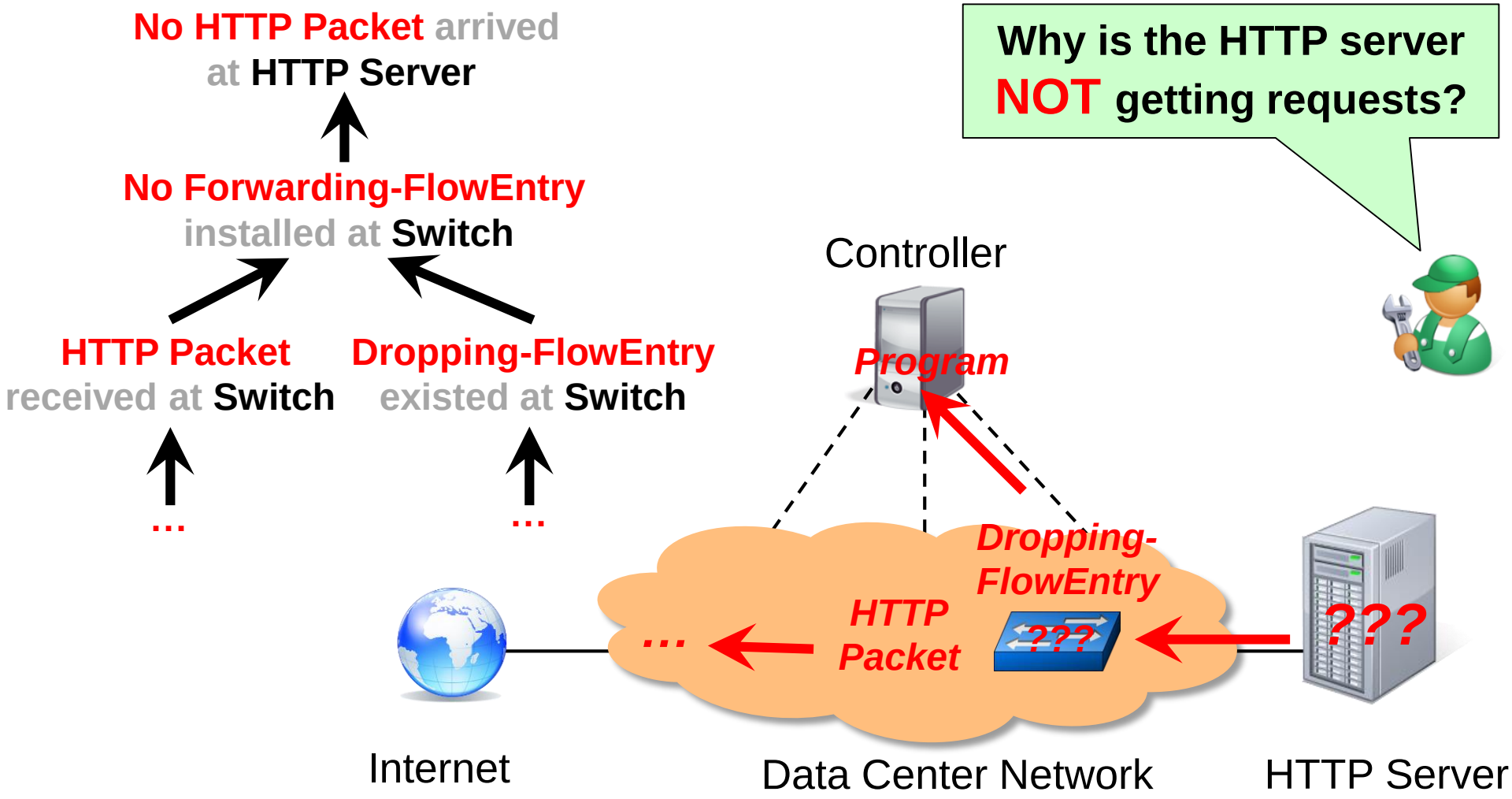
# Negative Provenance

[SIGCOMM 2014]



# Negative Provenance

[SIGCOMM 2014]



# Approach: Counter-factual reasoning

**Find all the ways a missing event could have occurred, and show why each of them did not happen.**

# Approach: Counter-factual reasoning

**Find all the ways a missing event could have occurred, and show why each of them did not happen.**

**Philly**

# Approach: Counter-factual reasoning

**Find all the ways a missing event could have occurred, and show why each of them did not happen.**

**Philly**

**Santa Cruz**

# Approach: Counter-factual reasoning

**Find all the ways a missing event could have occurred, and show why each of them did not happen.**

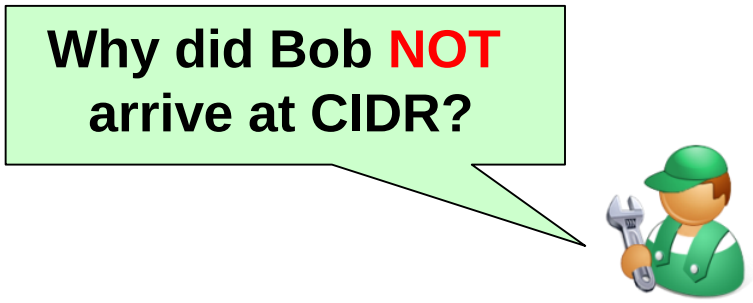
**Why did Bob NOT arrive at CIDR?**

**Philly**

**Santa Cruz**

# Approach: Counter-factual reasoning

**Find all the ways a missing event could have occurred, and show why each of them did not happen.**

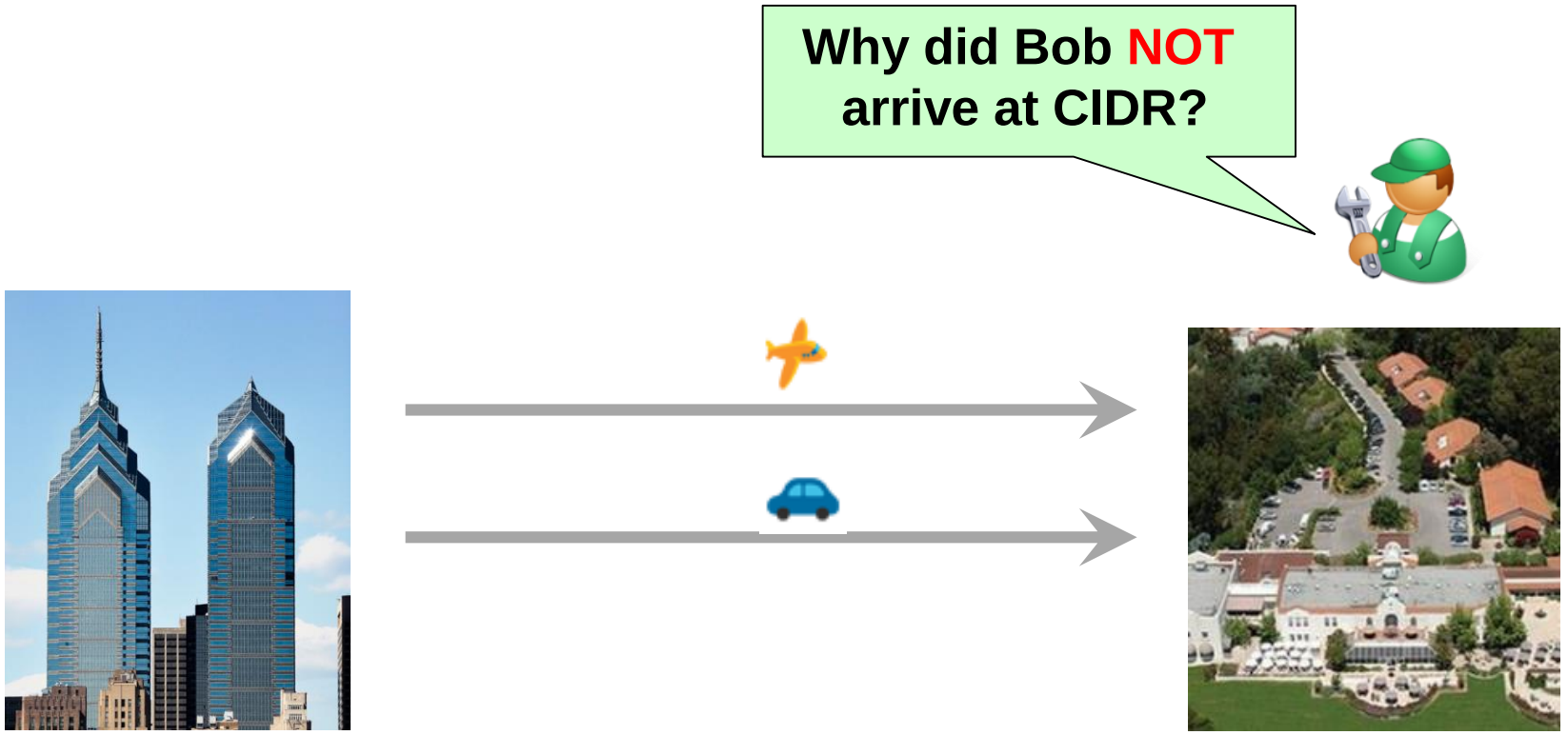



Philly

Santa Cruz

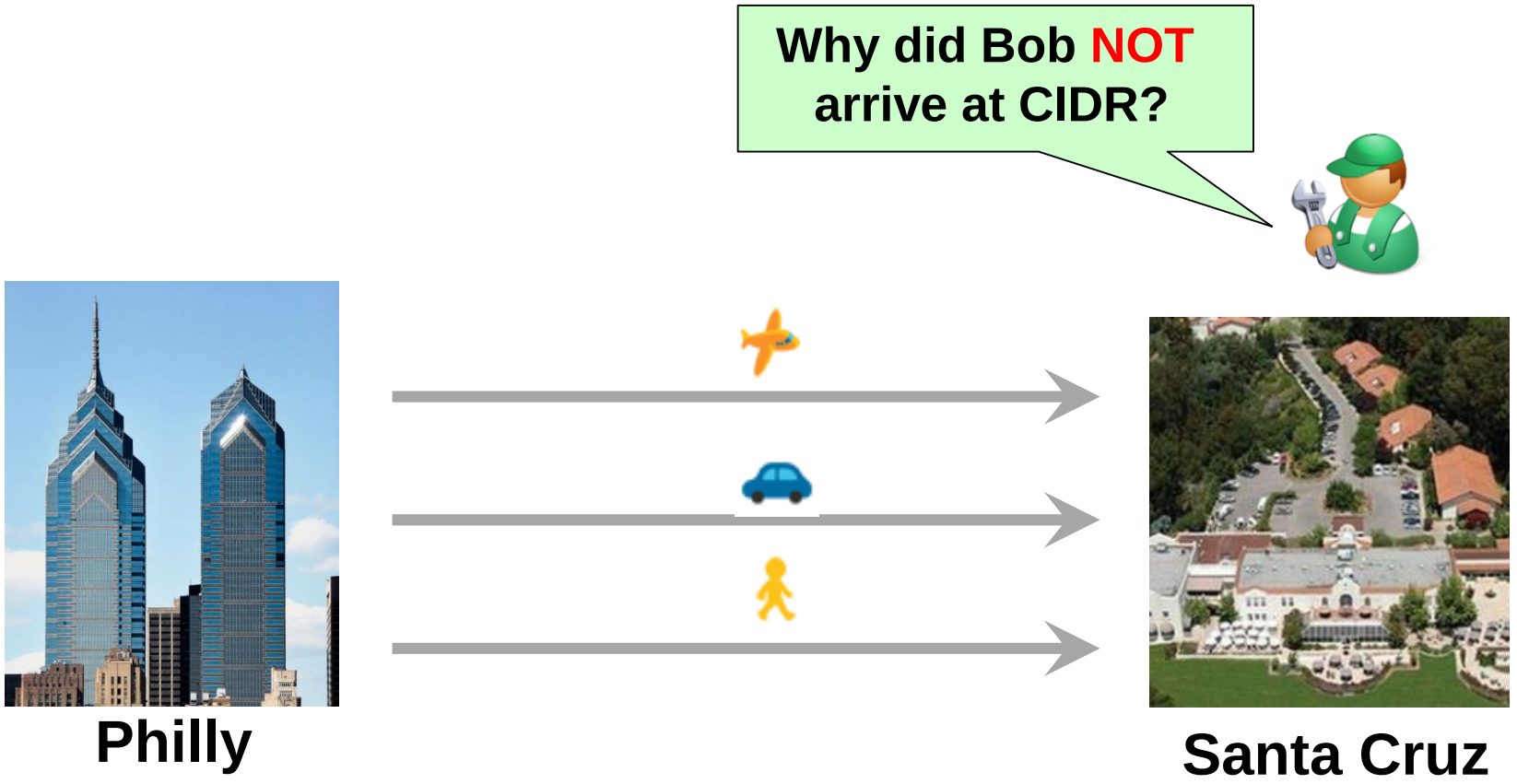
# Approach: Counter-factual reasoning

**Find all the ways a missing event could have occurred, and show why each of them did not happen.**



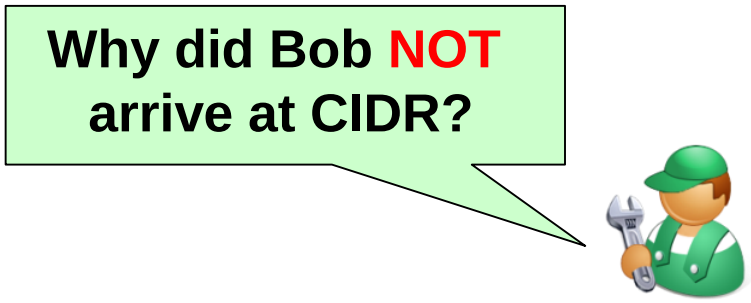
# Approach: Counter-factual reasoning

**Find all the ways a missing event could have occurred, and show why each of them did not happen.**

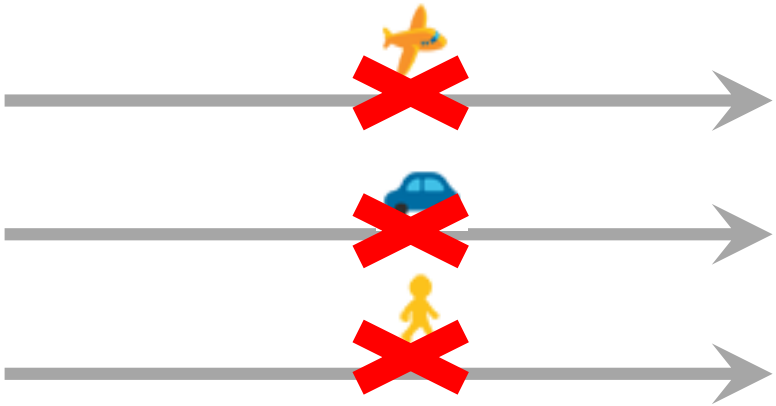


# Approach: Counter-factual reasoning

**Find all the ways a missing event could have occurred, and show why each of them did not happen.**



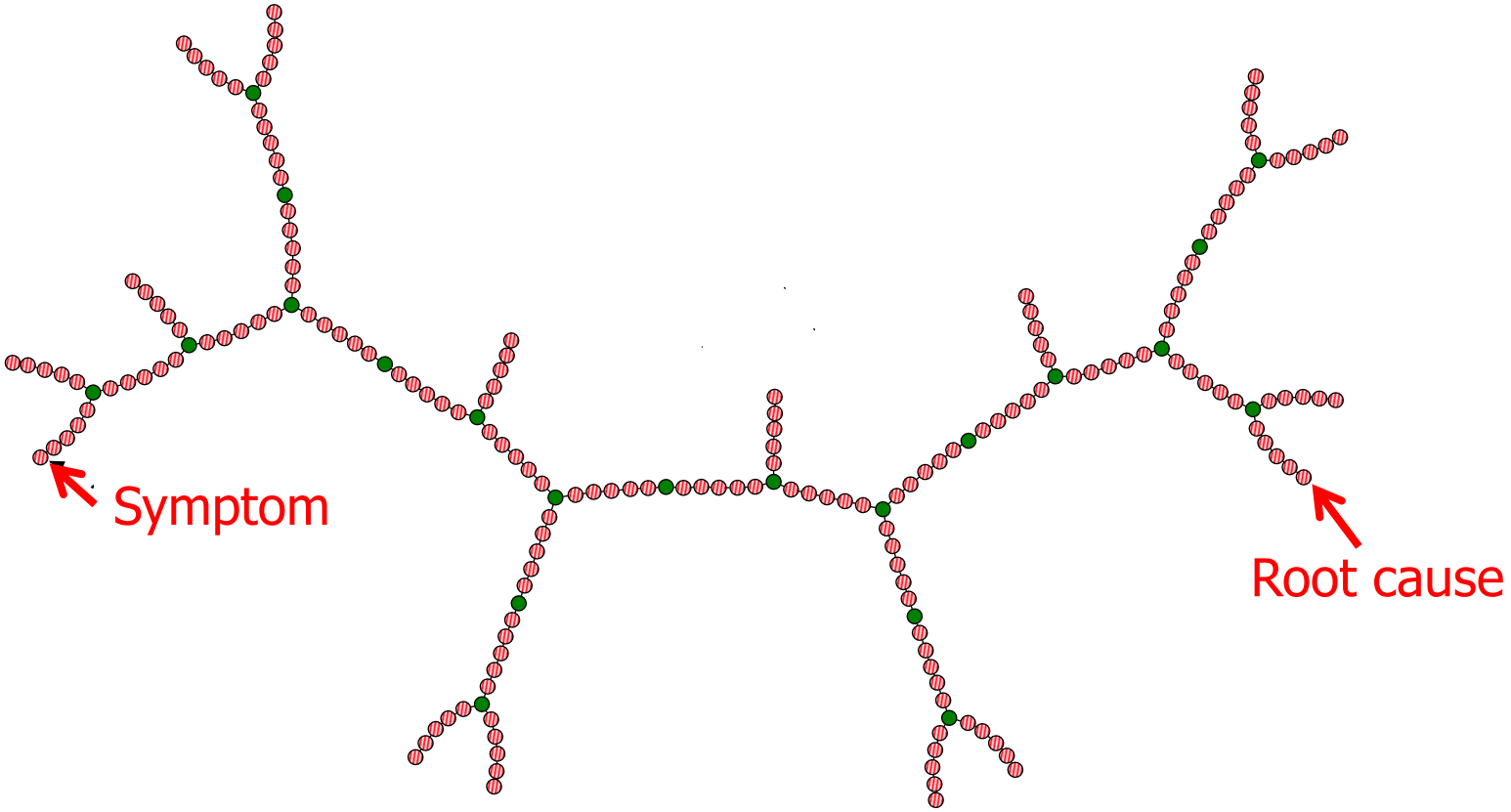

**Philly**



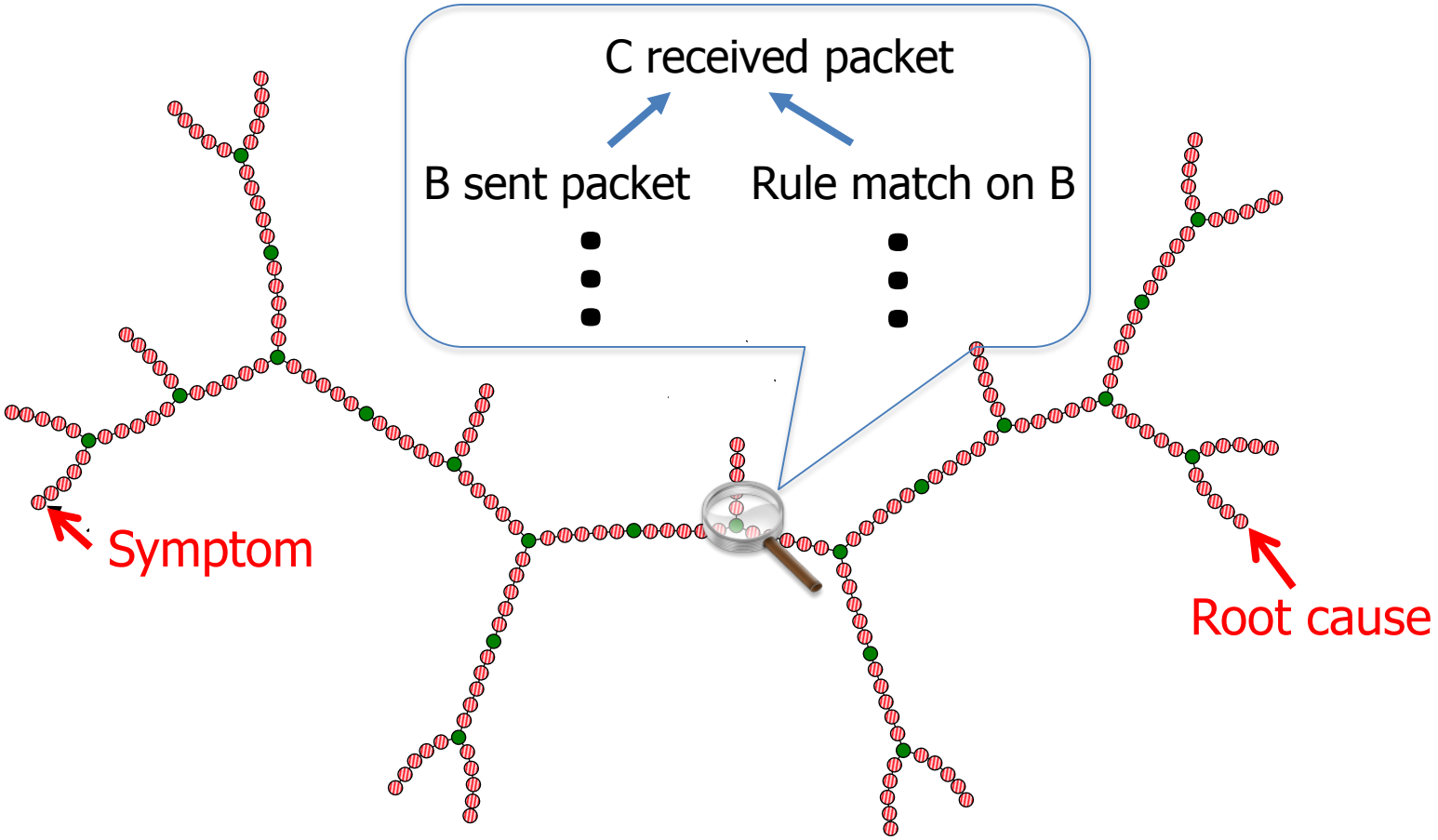
**Santa Cruz**

# Assumption #3: Provenance trees alone are sufficient for diagnostics

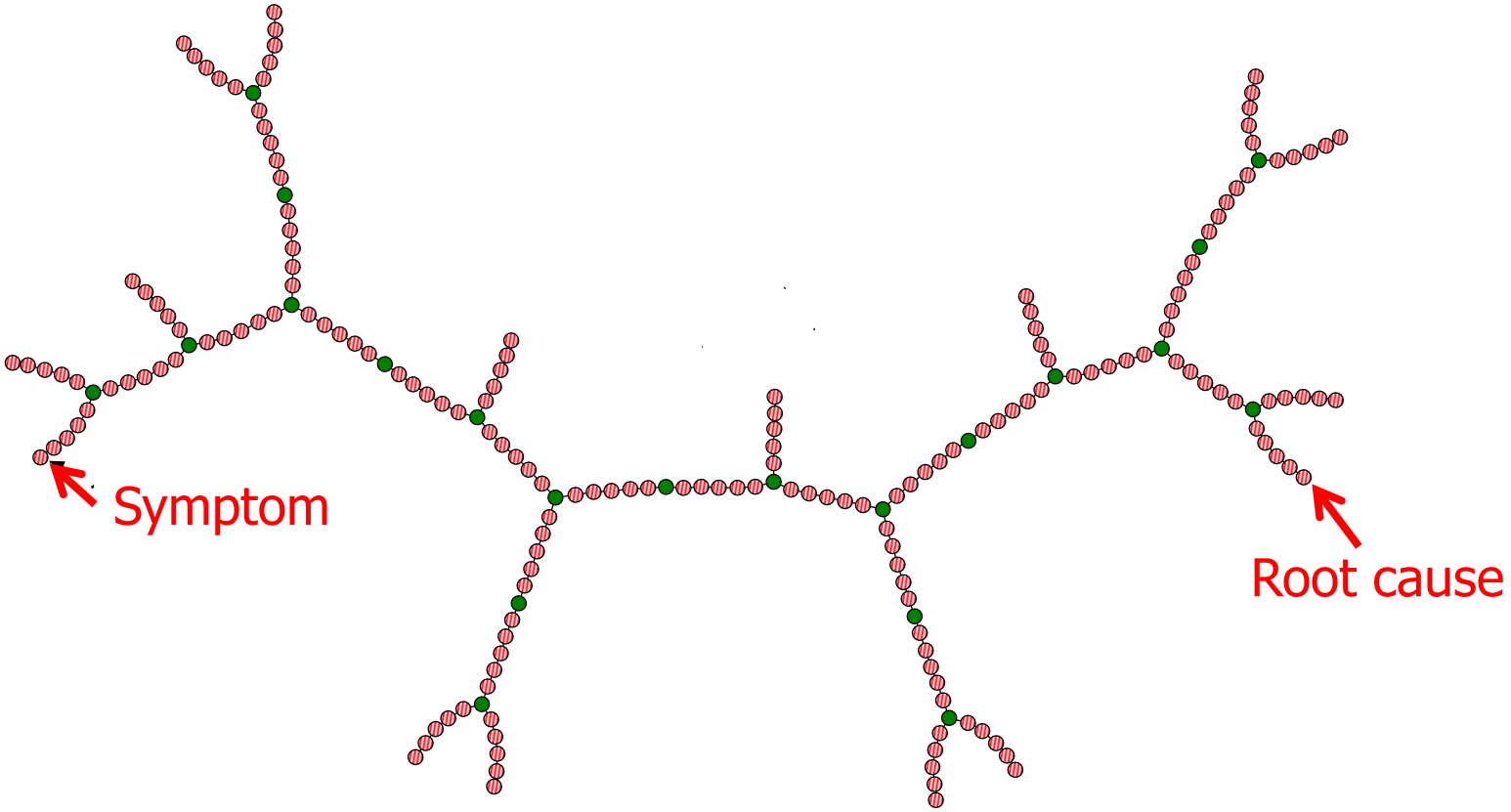
# Assumption #3: Provenance trees alone are sufficient for diagnostics



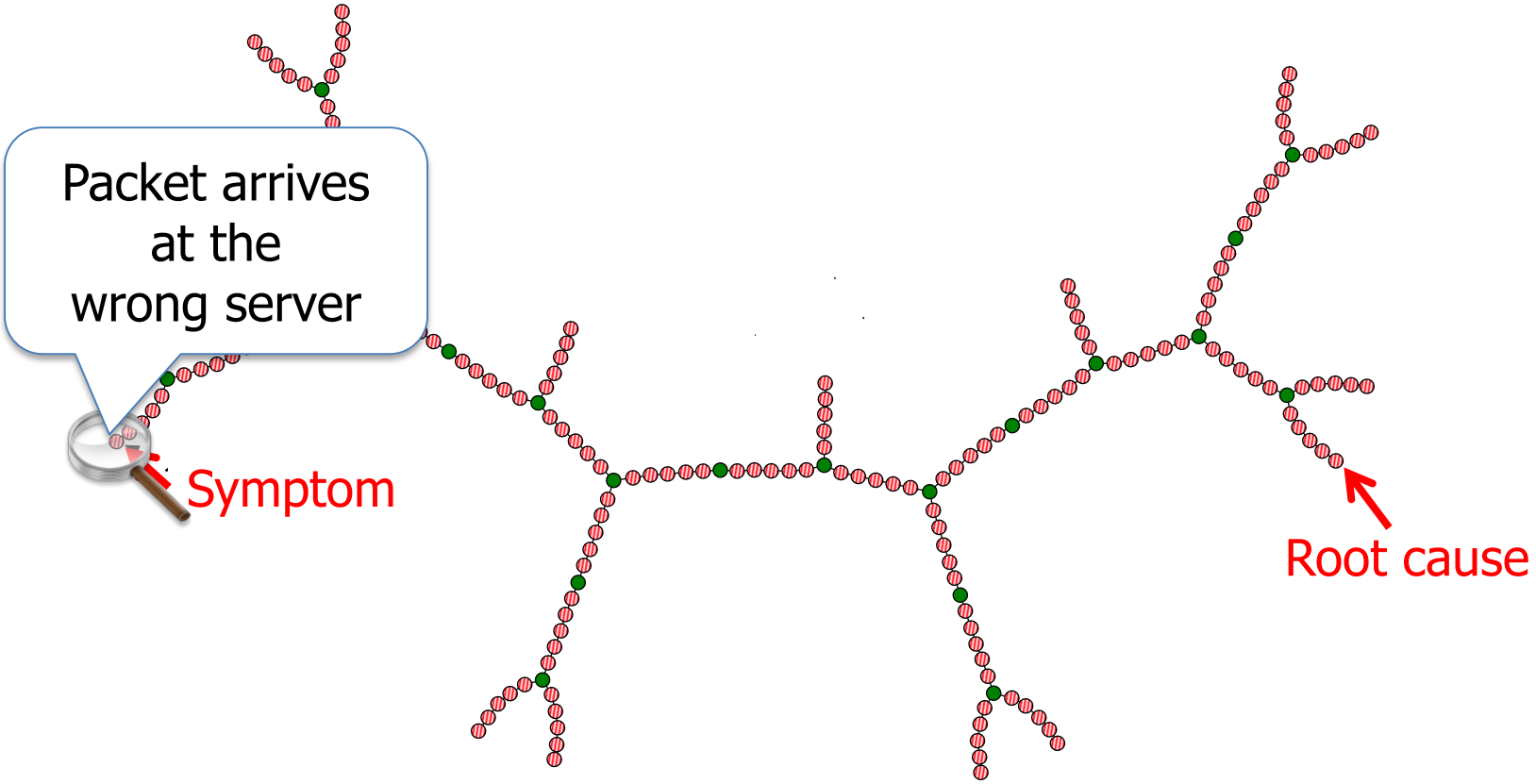
# Assumption #3: Provenance trees alone are sufficient for diagnostics



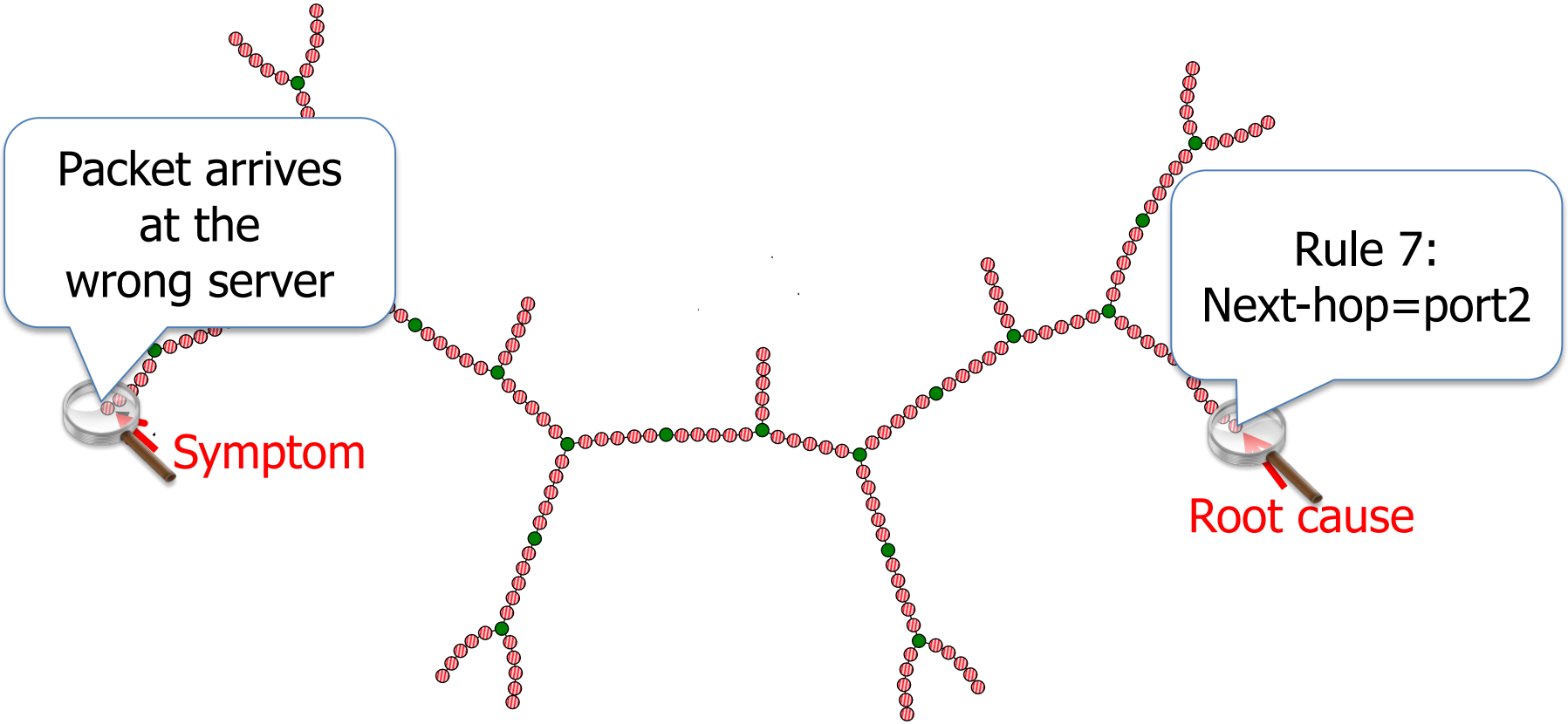
# Assumption #3: Provenance trees alone are sufficient for diagnostics



# Assumption #3: Provenance trees alone are sufficient for diagnostics



# Assumption #3: Provenance trees alone are sufficient for diagnostics



# What can we do?

# What can we do?

From: alice@xyz.com
To: Admin (bob@xyz.com)
Title: Help!

My server is receiving suspicious traffic from 4.3.2.0/24--it should have been sent to the low-security server. Packets from 4.3.3.0/24 are still being routed correctly. Can you help?

# What can we do?

From: alice@xyz.com
To: Admin (bob@xyz.com)
Title: Help!

My server is receiving suspicious traffic from 4.3.2.0/24--it should have been sent to the low-security server. Packets from 4.3.3.0/24 are still being routed correctly. Can you help?

**Working reference!**

# What can we do?

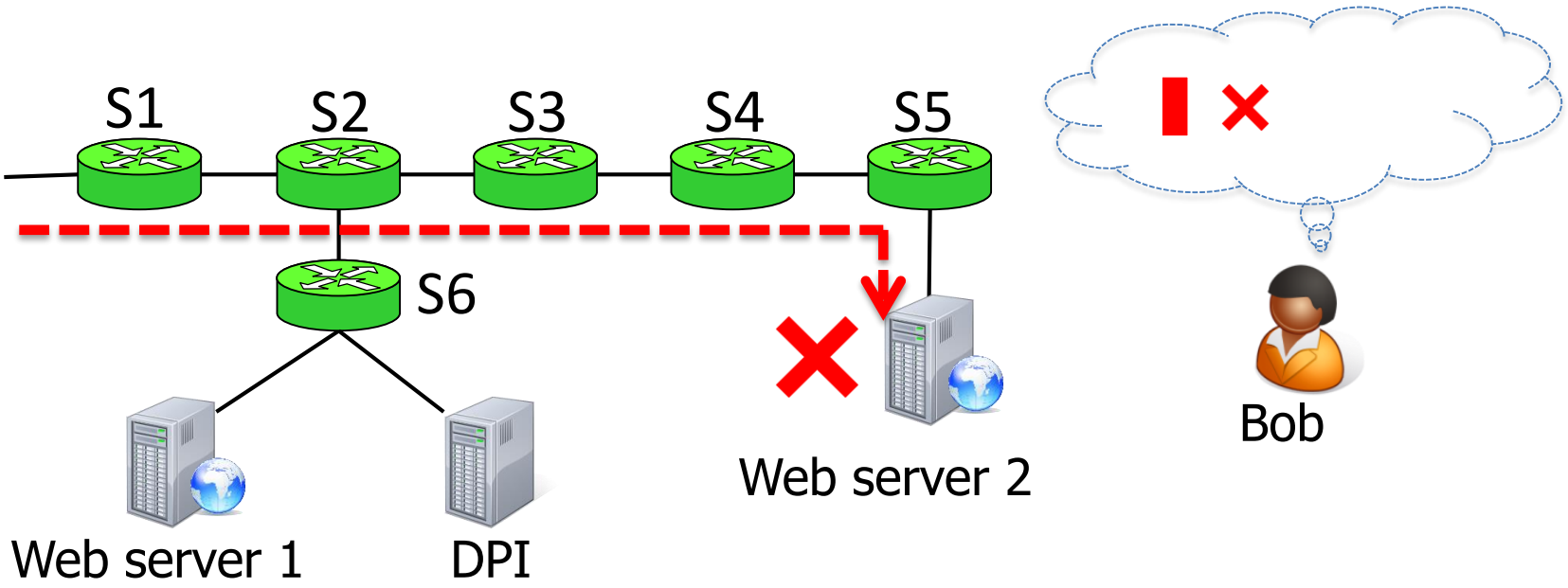
From: alice@xyz.com
To: Admin (bob@xyz.com)
Title: Help!

My server is receiving suspicious traffic from 4.3.2.0/24--it should have been sent to the low-security server. Packets from 4.3.3.0/24 are still being routed correctly. Can you help?

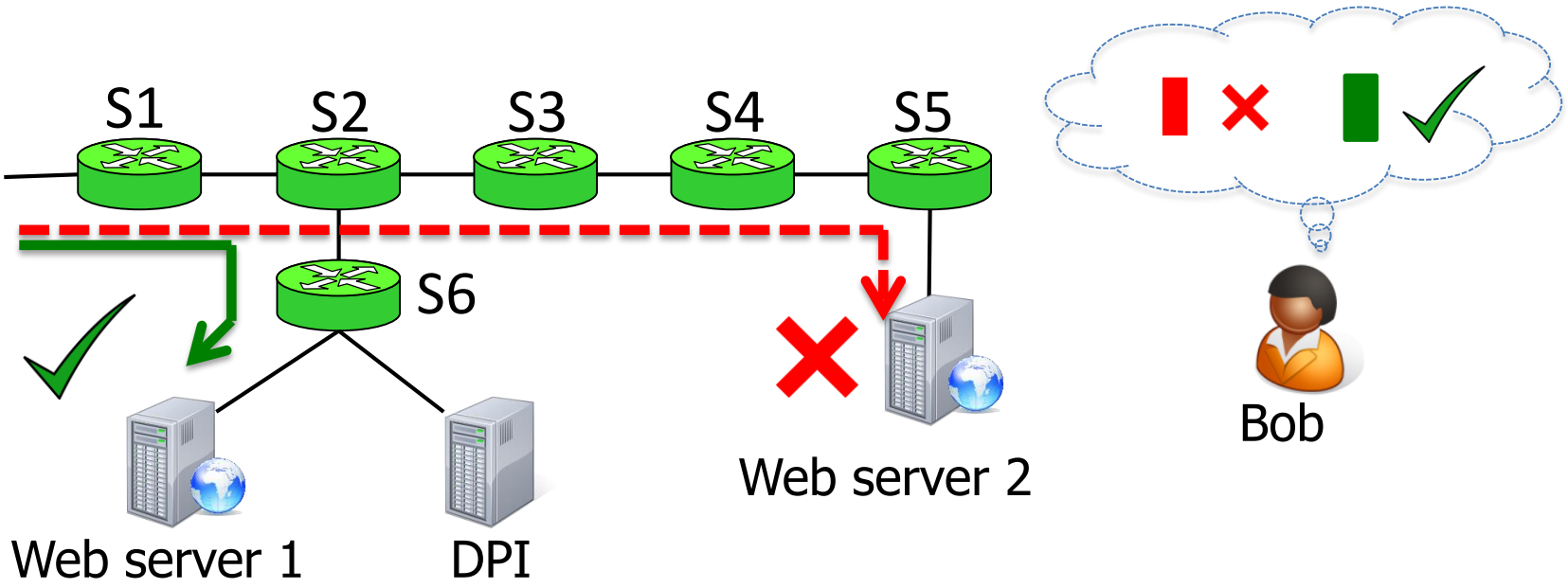
**Working reference!**

Outages mailing list Sept.—Dec. 2014:
**66%** have references!

# What can we do?



# What can we do?




- Idea: Reason about the differences between the symptom and the reference

# Differential provenance

[SIGCOMM 2016]

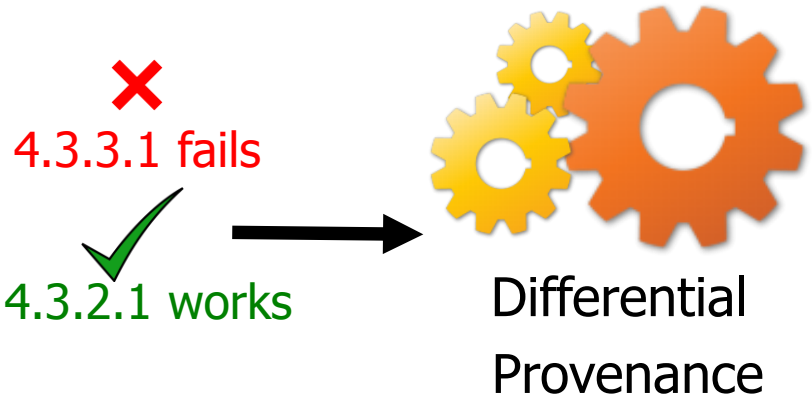
4.3.3.1 fails

4.3.2.1 works

- Input: a bad symptom and a good reference

# Differential provenance

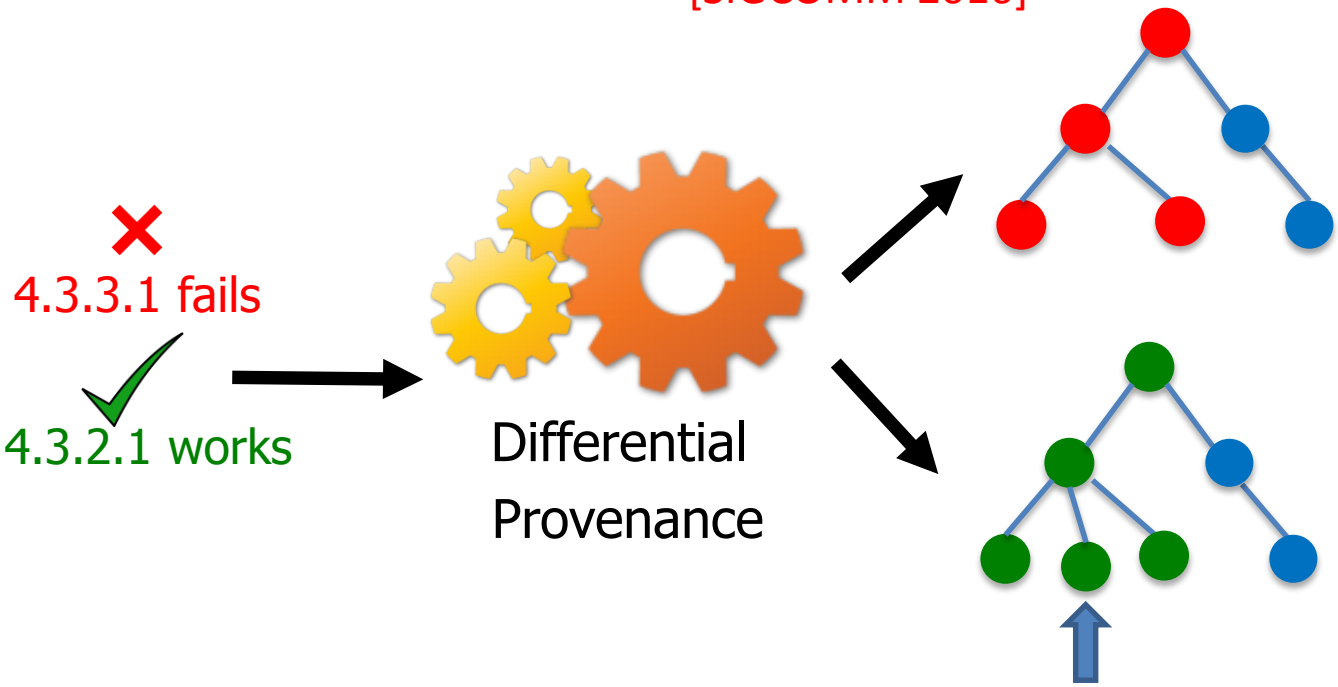
[SIGCOMM 2016]




- Input: a bad symptom and a good reference

# Differential provenance

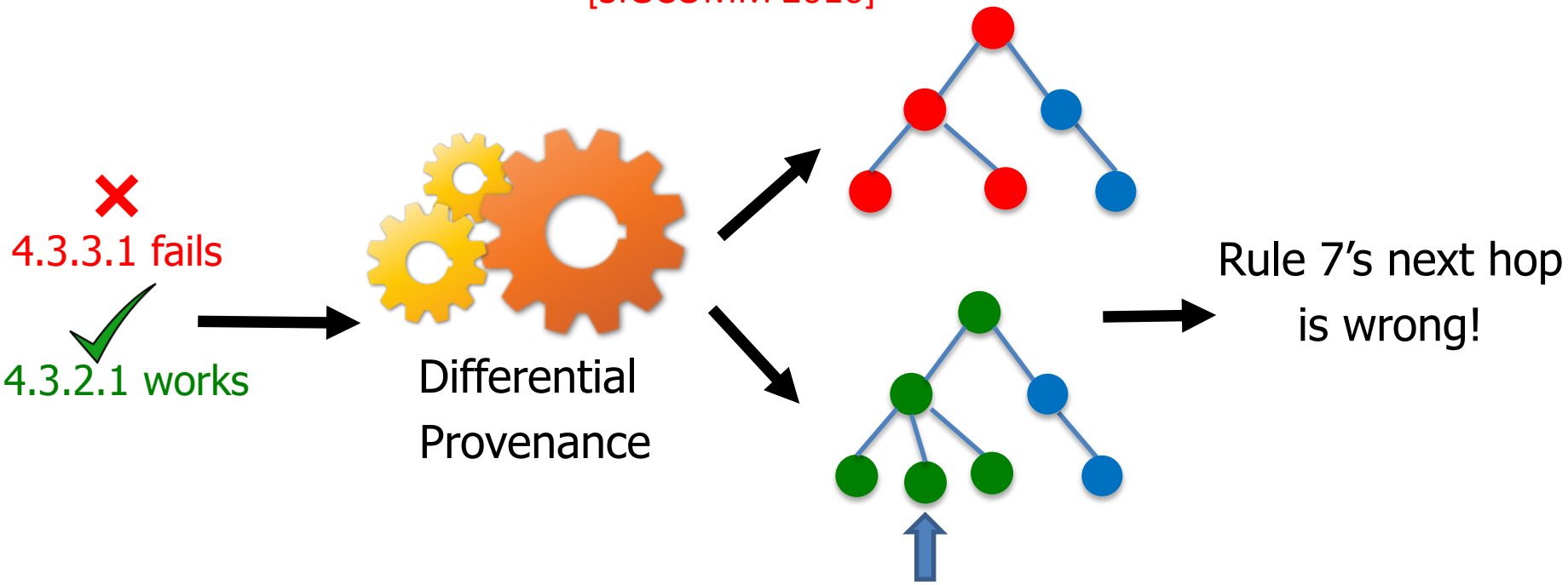
[SIGCOMM 2016]




- Input: a bad symptom and a good reference
- Debugger reasons about the differences

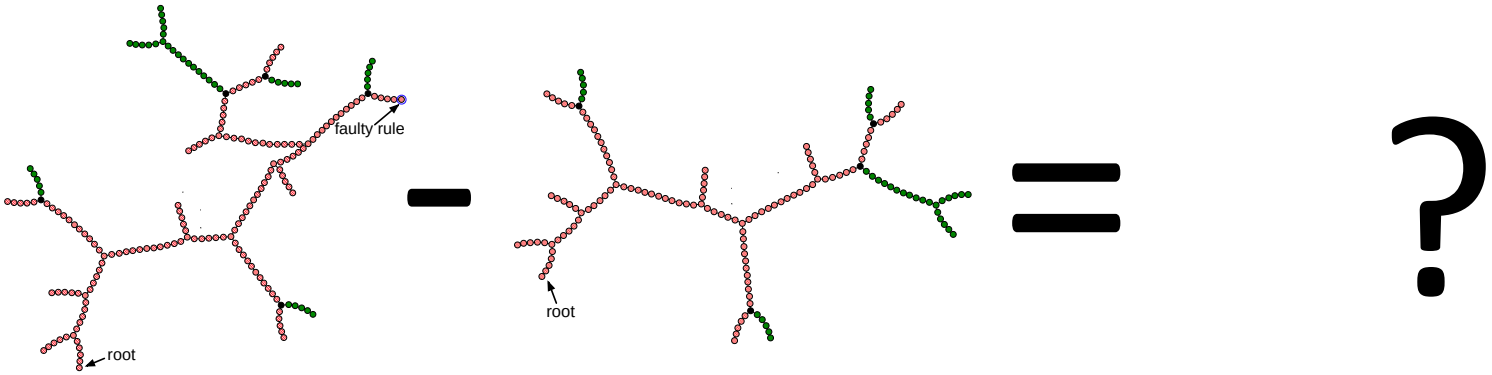
# Differential provenance

[SIGCOMM 2016]




- Input: a bad symptom and a good reference
- Debugger reasons about the differences
- Output: root cause

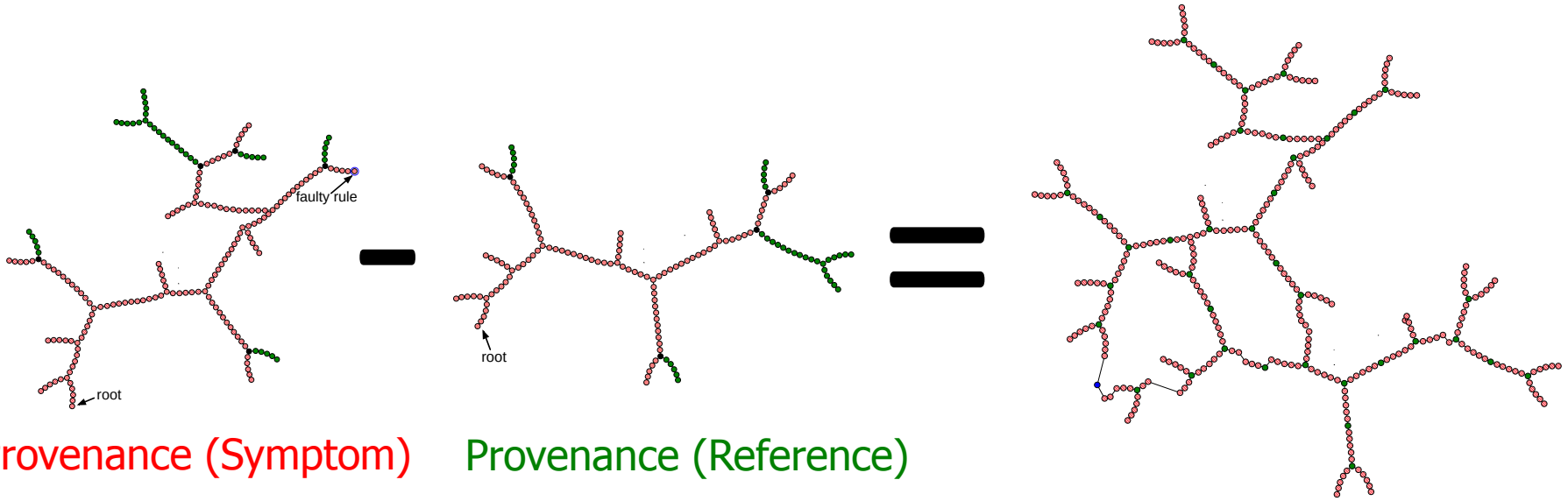
# Strawman solution




Provenance (Symptom) Provenance (Reference)

- Strawman solution: Find vertexes that are different in the two trees

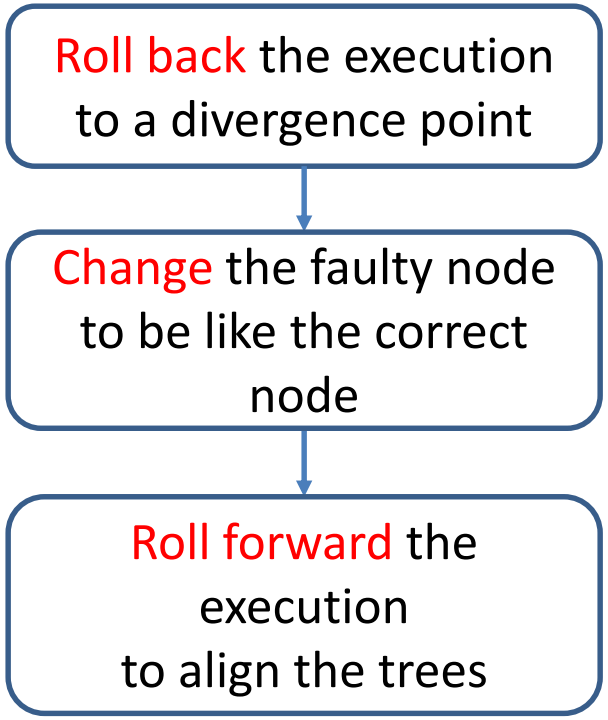
# Strawman solution




Provenance (Symptom) Provenance (Reference)

- Strawman solution: Find vertexes that are different in the two trees
- Problem: The diff can be larger than the individual trees!

# Overly Simplified Approach in a nutshell

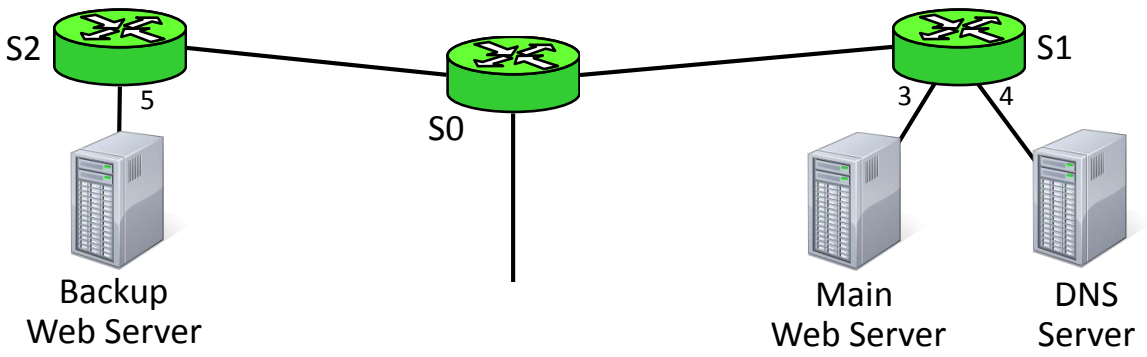


# Assumption #4: Software is correct and static

- Networks are software and can have bugs

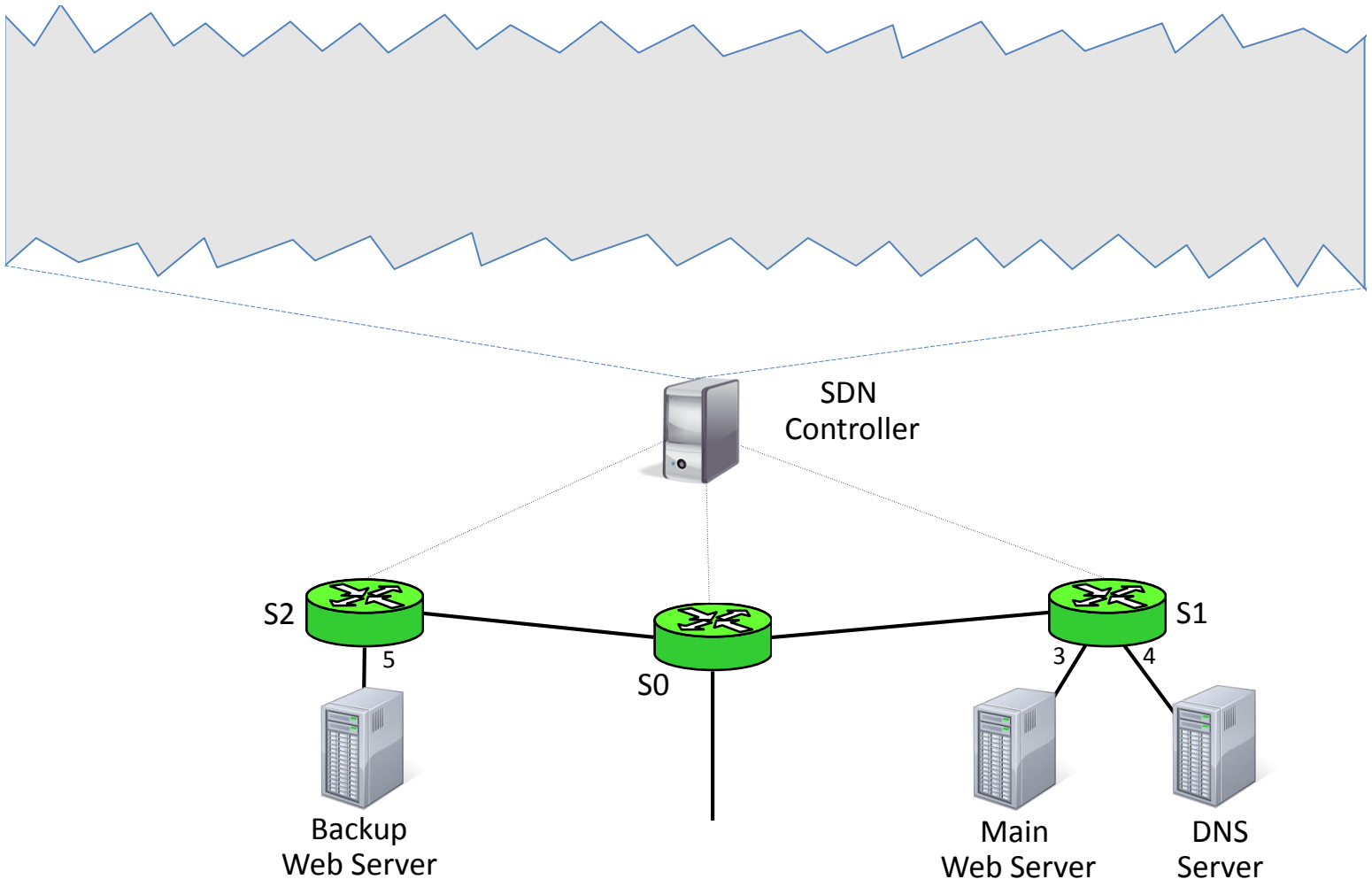
# Assumption #4: Software is correct and static

- Networks are software and can have bugs



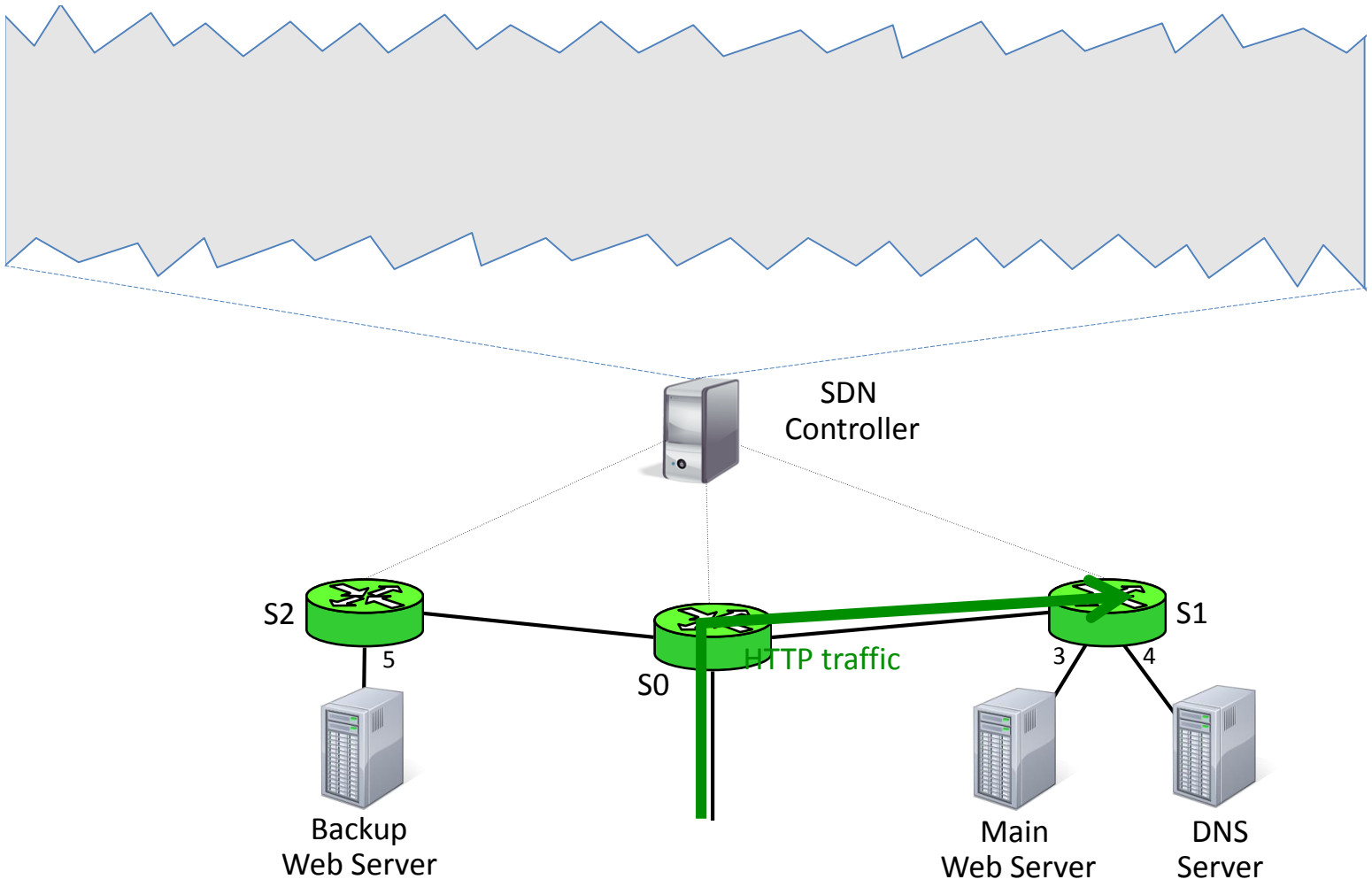
# Assumption #4: Software is correct and static

- Networks are software and can have bugs



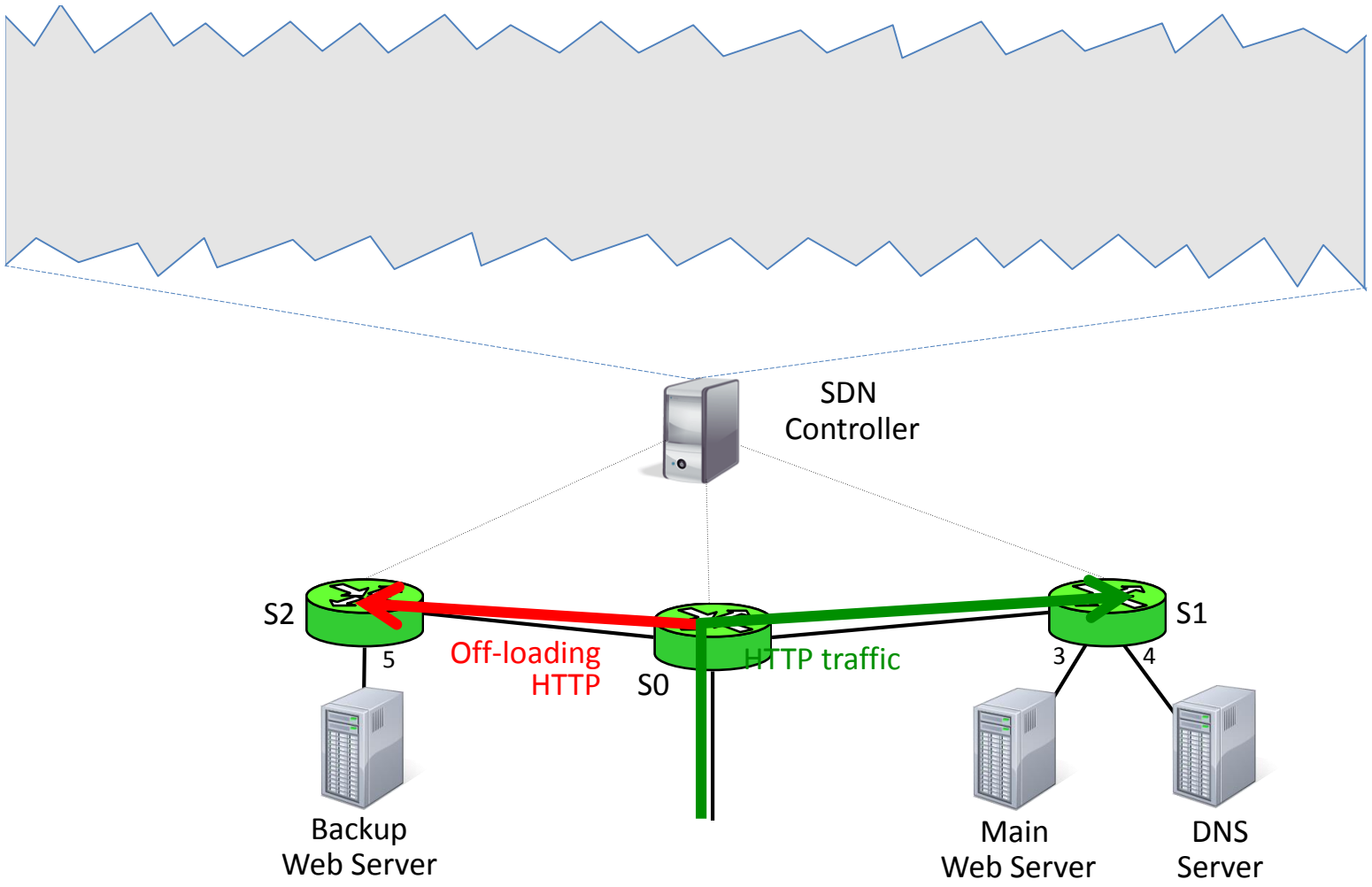
# Assumption #4: Software is correct and static

- Networks are software and can have bugs



# Assumption #4: Software is correct and static

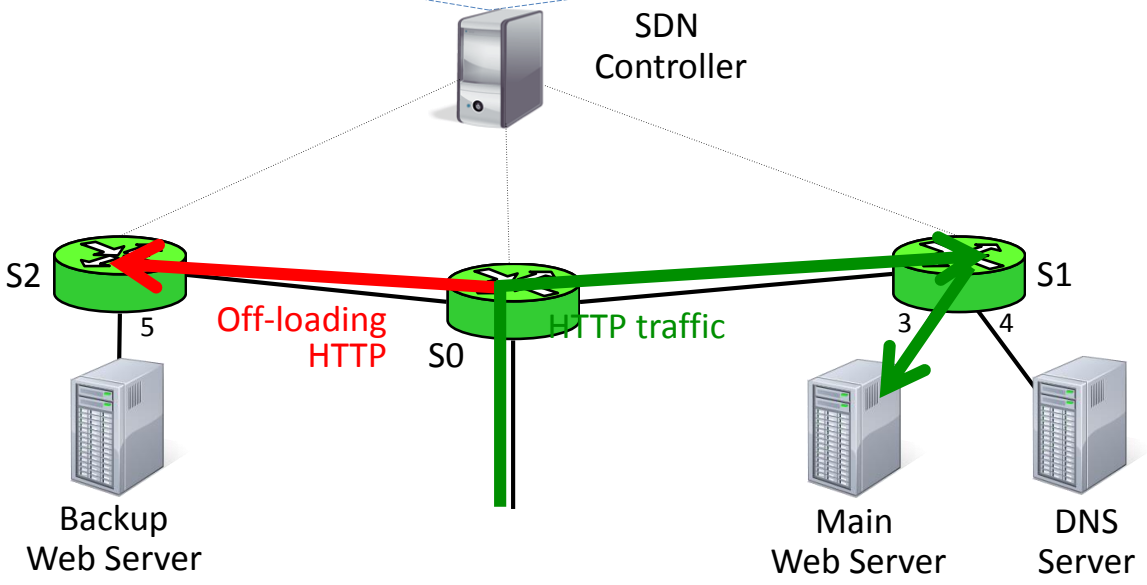
- Networks are software and can have bugs



# Assumption #4: Software is correct and static

- Networks are software and can have bugs

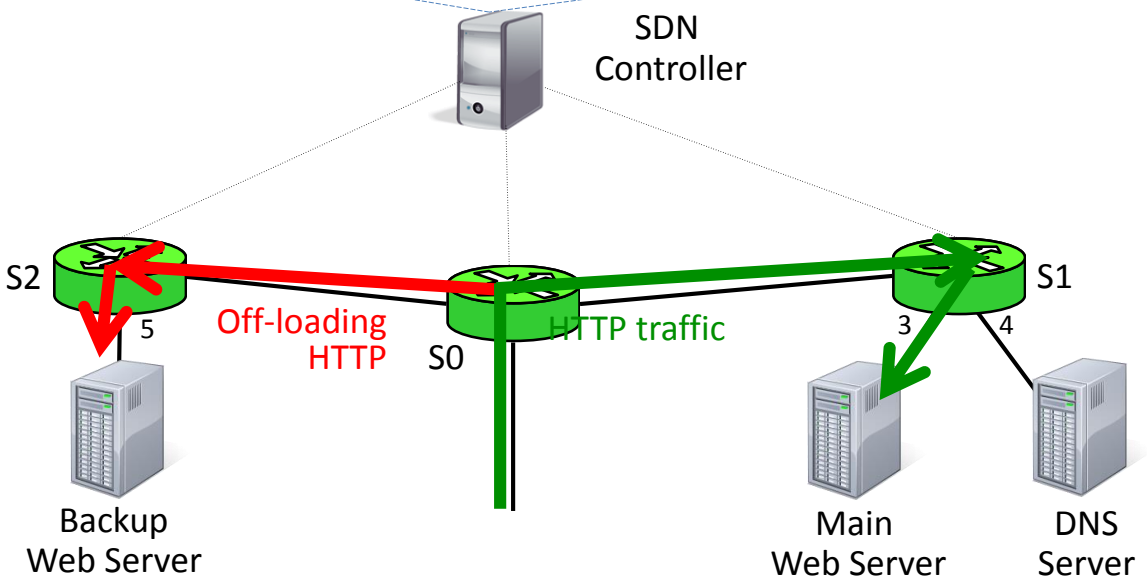
*else if* (switch == S1 && protocol == HTTP) *then* action = output:3.



# Assumption #4: Software is correct and static

- Networks are software and can have bugs

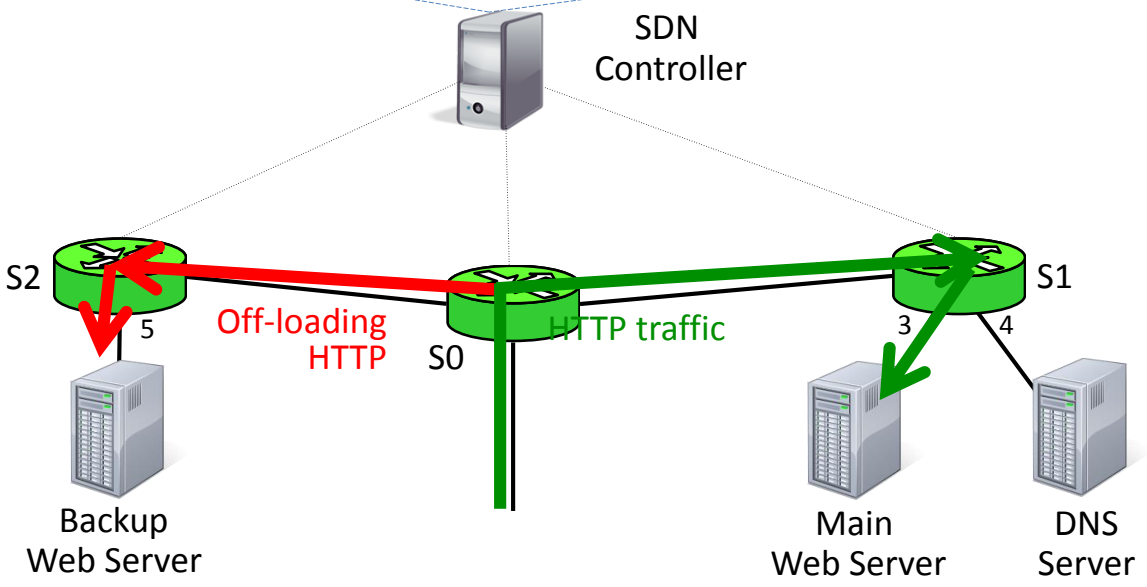
*else if* (switch == S1 && protocol == HTTP) *then* action = output:3.



# Assumption #4: Software is correct and static

- Networks are software and can have bugs

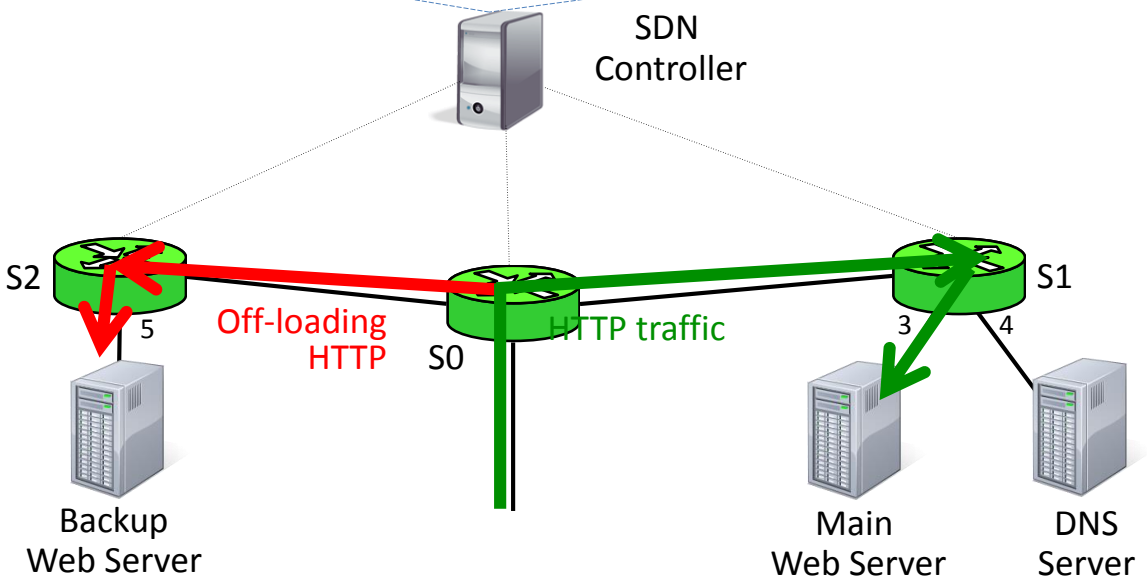
*else if* (switch == S1 && protocol == HTTP) *then* action = output:3.
*else if* (switch == S1 && protocol == HTTP) *then* action = output:5.



# Assumption #4: Software is correct and static

- Networks are software and can have bugs

*else if* (switch == S1 && protocol == HTTP) *then* action = output:3.
*else if* (switch == S1 && protocol == HTTP) *then* action = output:5.



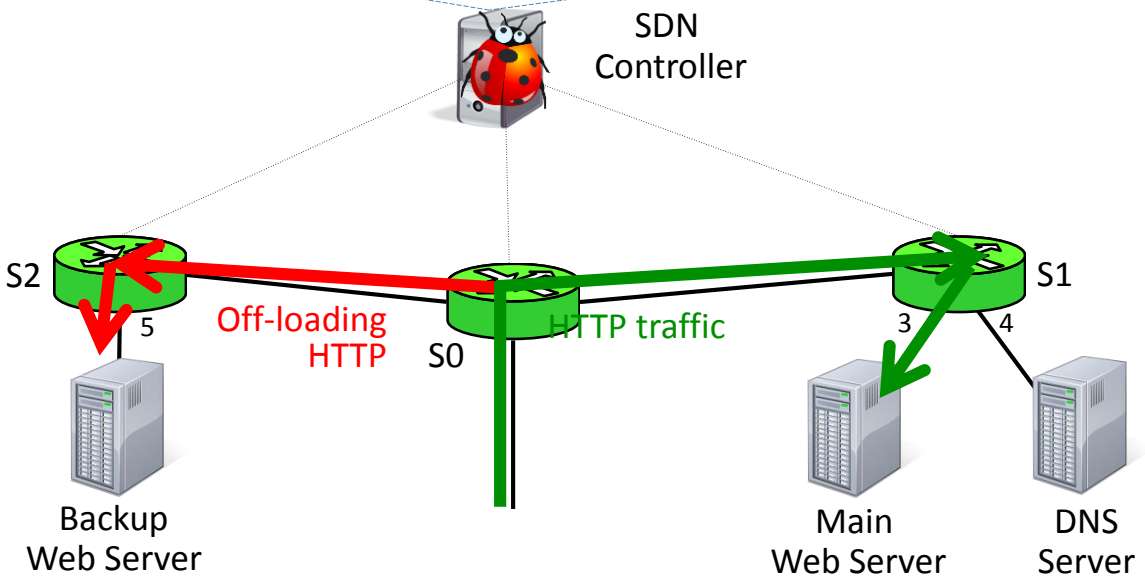
# Assumption #4: Software is correct and static

- Networks are software and can have bugs

*else if* (switch == S1 && protocol == HTTP) *then* action = output:3.

*else if* (switch == S1 && protocol == HTTP) *then* action = output:5.

**Copy-and-paste bug!!!**

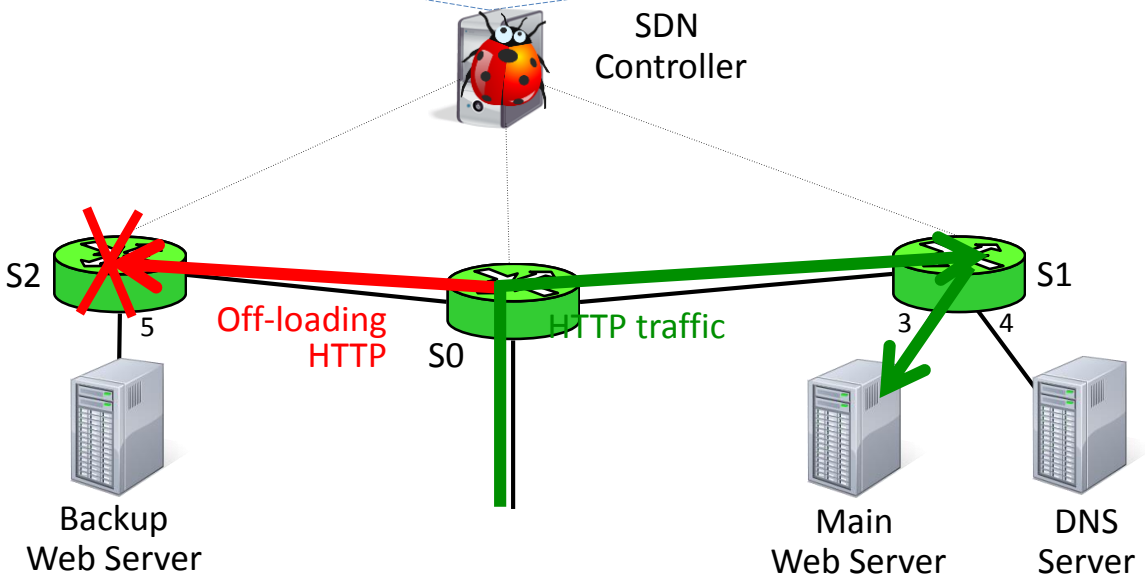


# Assumption #4: Software is correct and static

- Networks are software and can have bugs

*else if* (switch == S1 && protocol == HTTP) *then* action = output:3.
*else if* (switch == S1 && protocol == HTTP) *then* action = output:5.

**Copy-and-paste bug!!!**

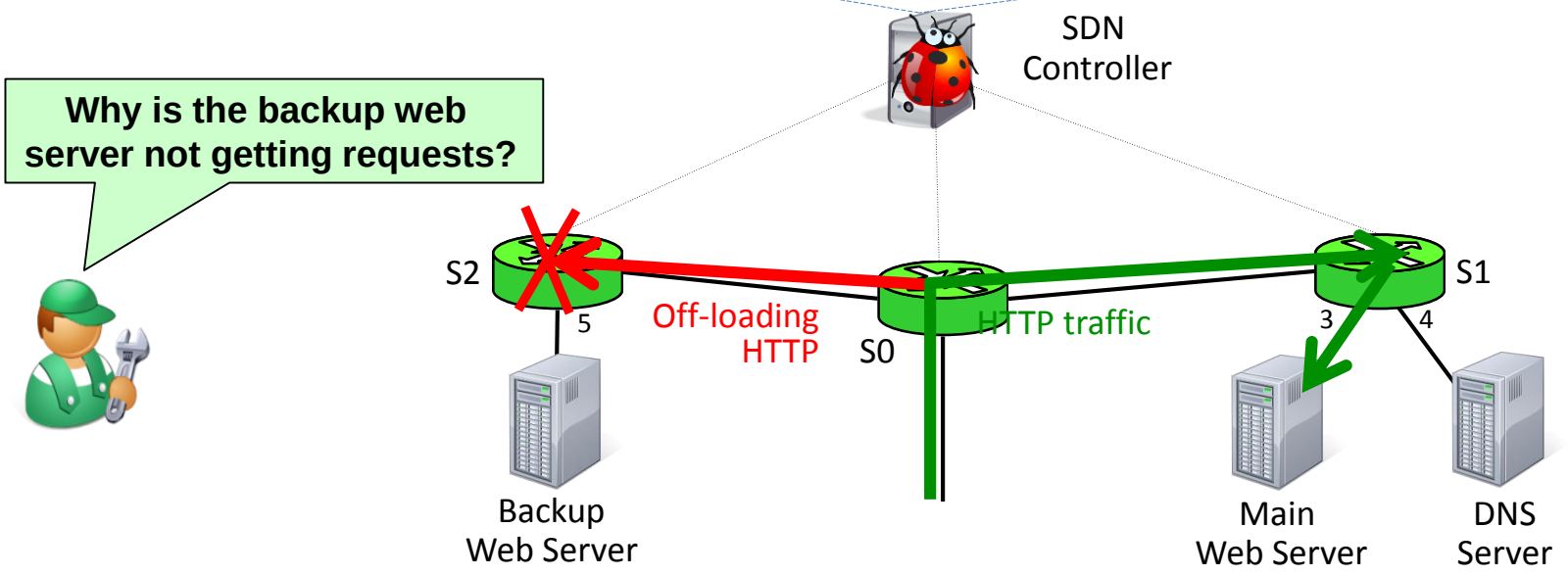


# Assumption #4: Software is correct and static

- Networks are software and can have bugs

*else if* (switch == S1 && protocol == HTTP) *then* action = output:3.
*else if* (switch == S1 && protocol == HTTP) *then* action = output:5.

**Copy-and-paste bug!!!**

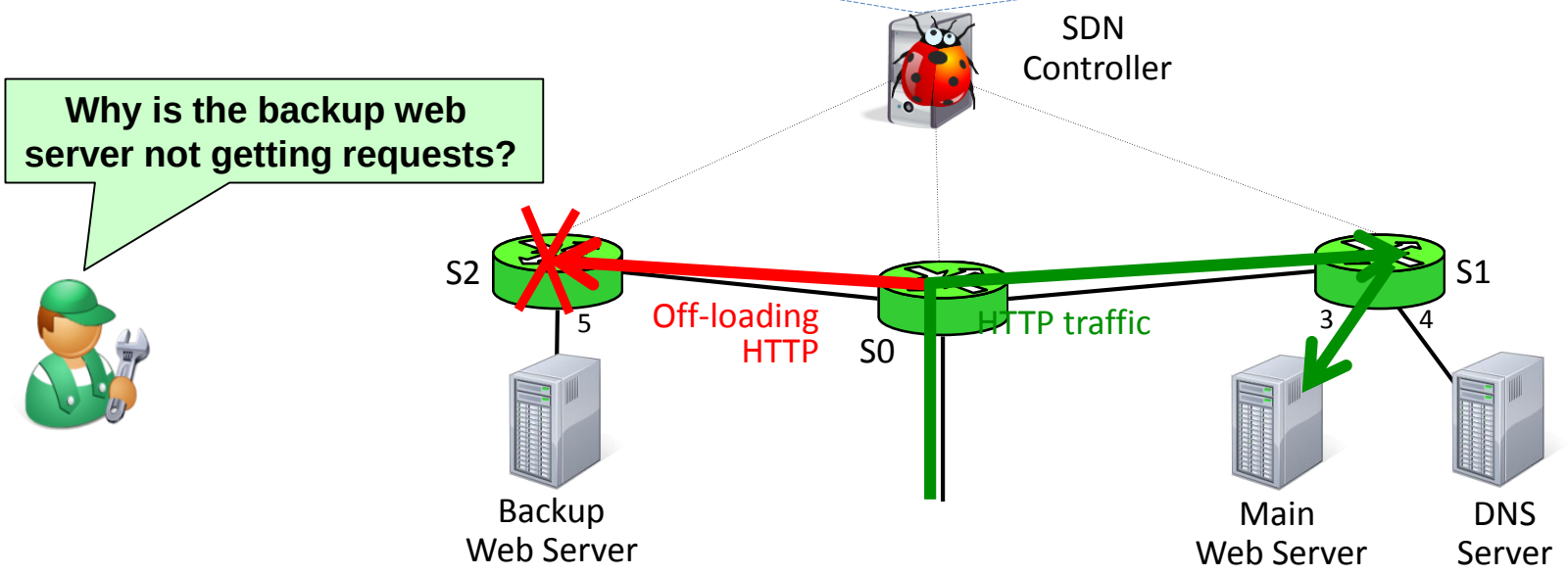


# Assumption #4: Software is correct and static

- Networks are software and can have bugs
- How can we find and fix bugs quickly?

*else if* (switch == S1 && protocol == HTTP) *then* action = output:3.
*else if* (switch == S1 && protocol == HTTP) *then* action = output:5.

**Copy-and-paste bug!!!**

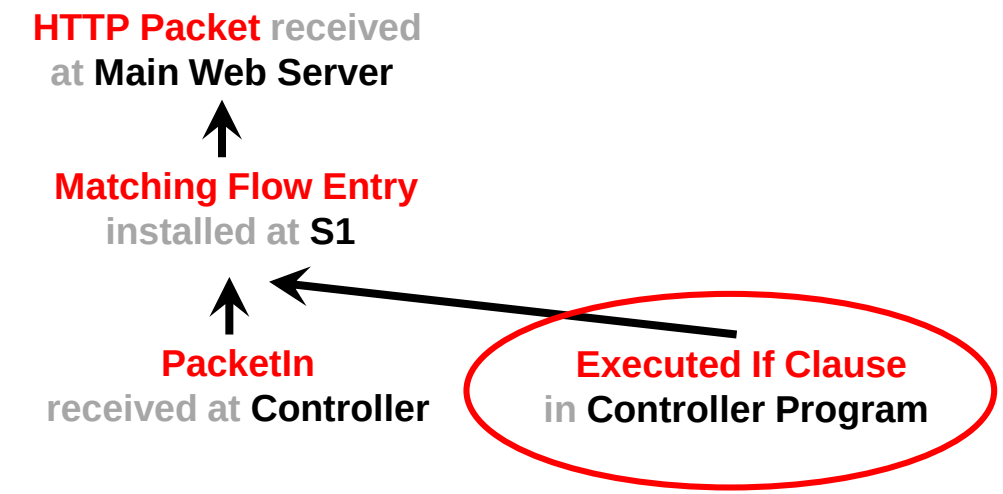


# Approach: Meta provenance

[NSDI 2017]

- Problem: Finding fixes is hard
- Idea: Provenance can pinpoint the root cause
- But previous provenance focus exclusively on data
- Key idea: Treating program as data

meta provenance



# Repairing Software-defined Networking Programs

Full support of Network Datalog (declarative)

Support a subset of Pyretic (Python + DSL)  frenetic ≫

Support a subset of Trema (imperative Ruby) 

Uses Z3 SMT solver to enumerate repairs

More details are in [HotNets'15, NSDI'17]

# Network Provenance Research (2010 – 2017)

- Network provenance model [SIGMOD'10]
- Secure network provenance [SOSP'11]
- Provenance in dynamic environments [VLDB'13]
- Negative provenance [SIGCOMM'14]
- Distributed provenance compression [SIGMOD'17]
- Differential provenance [SIGCOMM'16]
- Meta-provenance [NSDI'17]

Explanations

Deeper diagnostics and repair

Ph.D. dissertation work of Ang Chen (2017), Chen Chen (2017), Yang Wu (2017), and Wenchao Zhou (2012).

# The Road Ahead

- Network forensics meets data provenance is a rich area of exploration!
- Sampling of the problems we are working on:
  - Network forensics on the data plane
  - Privacy-preserving provenance on sensitive networks
  - Probabilistic provenance
  - Automated repairs of complex events
  - Timing-based provenance
  - and more….

# Thank You!

- Network provenance team at Penn/Georgetown:
  - Ang Chen, Chen Chen, Ling Ding, Qiong Fei, Andreas Haeberlen, Zachary Ives, Yang Li, Boon Thau Loo, Suyog Mapara, Arjun Narayan, Yiqing Ren, Micah Sherr, Shengzhi Sun, Tao Tao, Yang Wu, Mingchen Zhao, Wenchao Zhou