Shadow Configurations:
A Network Management Primitive

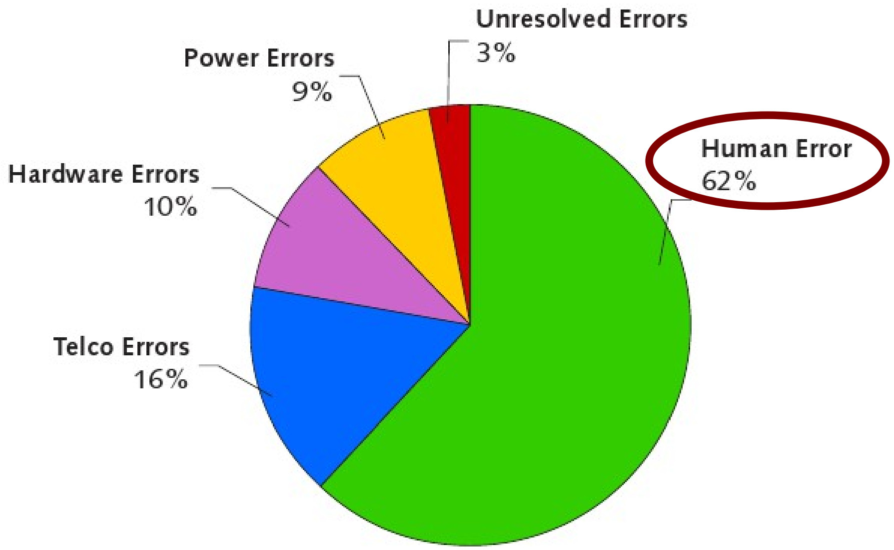
Richard Alimi, Ye Wang, Y. Richard Yang

Laboratory of Networked Systems
Yale University

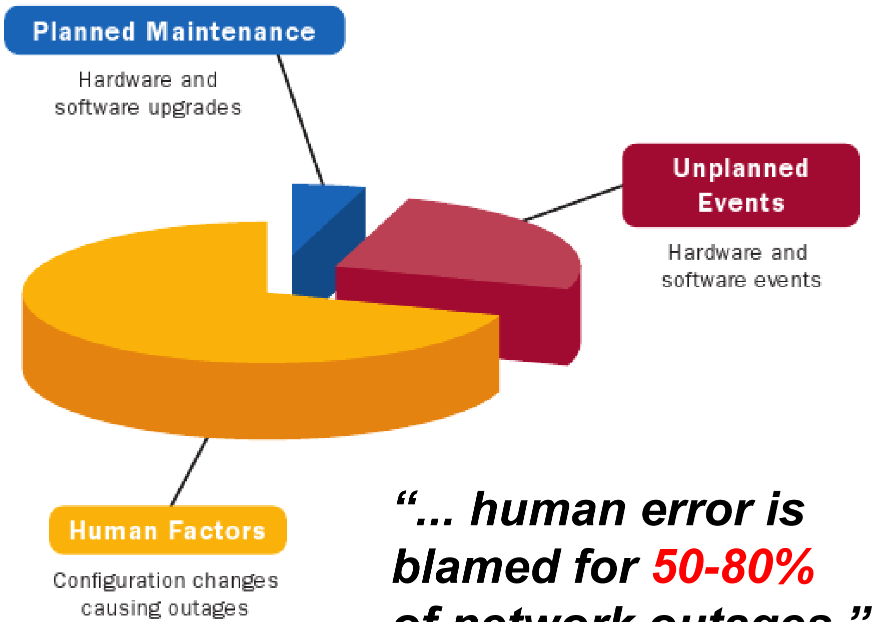
February 16, 2009

Configuration Leads to Errors

"80% of IT budgets is used to maintain the status quo."



Source: The Yankee Group, 2004



"... human error is blamed for 50-80% of network outages."

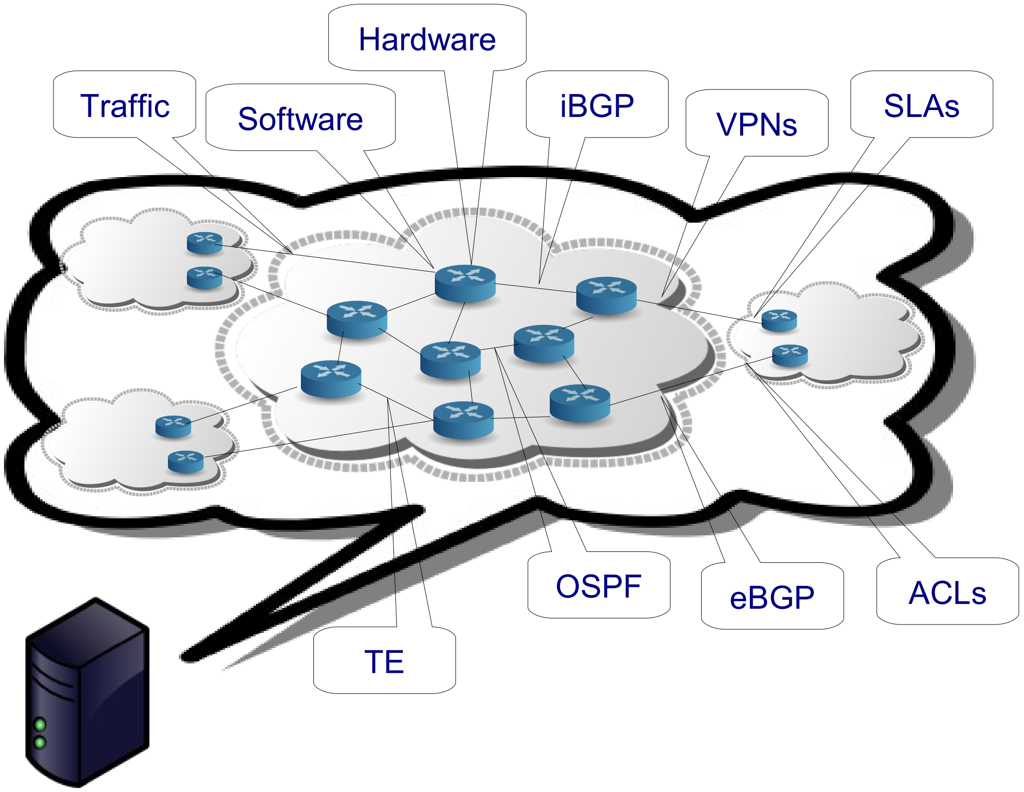
Source: Juniper Networks, 2008

Why is configuration hard today?

Configuration Management Today

Simulation & Analysis

- Depend on simplified models
  - Network structure
  - Hardware and software
- Limited scalability
- Hard to access real traffic



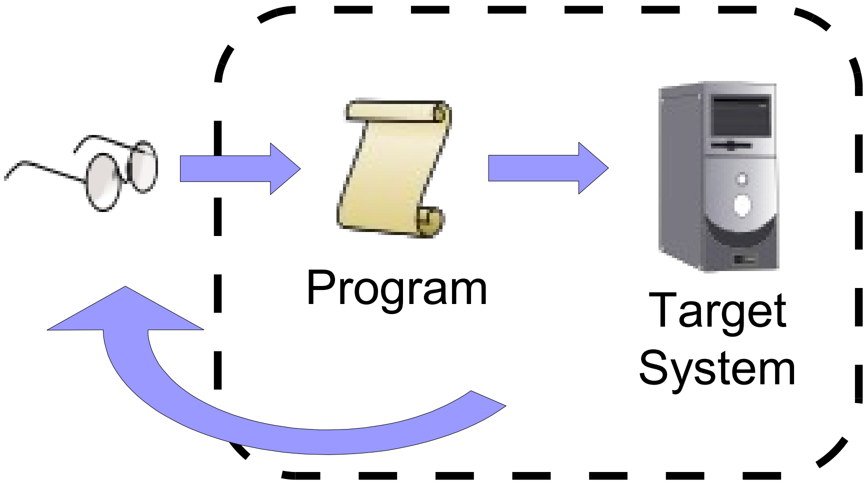
Test networks

- Can be prohibitively expensive

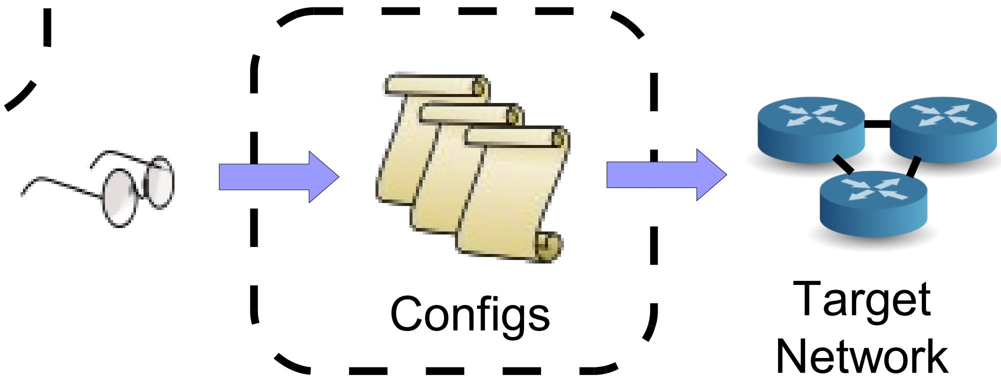
Why are these not enough?

Analogy with Programming

Programming



Network Management



Analogy with Databases

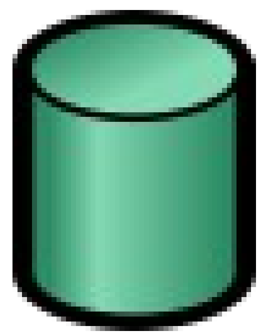
Databases

STATE A

INSERT ...
UPDATE ...
DELETE ...

STATE B

INSERT ...
UPDATE ...
DELETE ...



Network Management

STATE A

ip route ...

STATE B

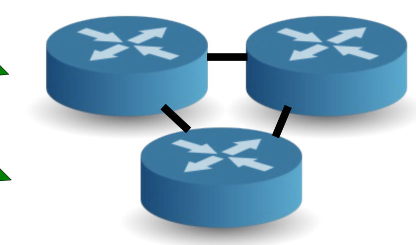
ip addr ...

STATE C

router bgp ...

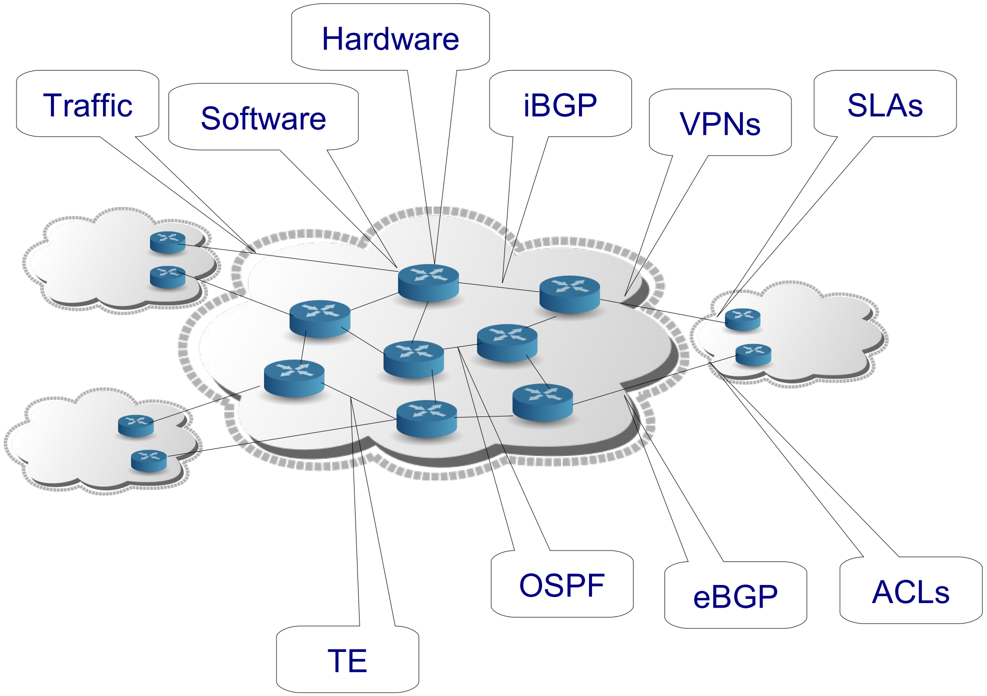
STATE D

router ospf ...



?

Enter, Shadow Configurations



Key ideas

- Allow additional (shadow) config on each router
- In-network, interactive shadow environment
- “Shadow” term from computer graphics

Key Benefits

- Realistic (no model)
- Scalable
- Access to real traffic
- Transactional

Roadmap

Motivation and Overview

System Basics and Usage

System Components

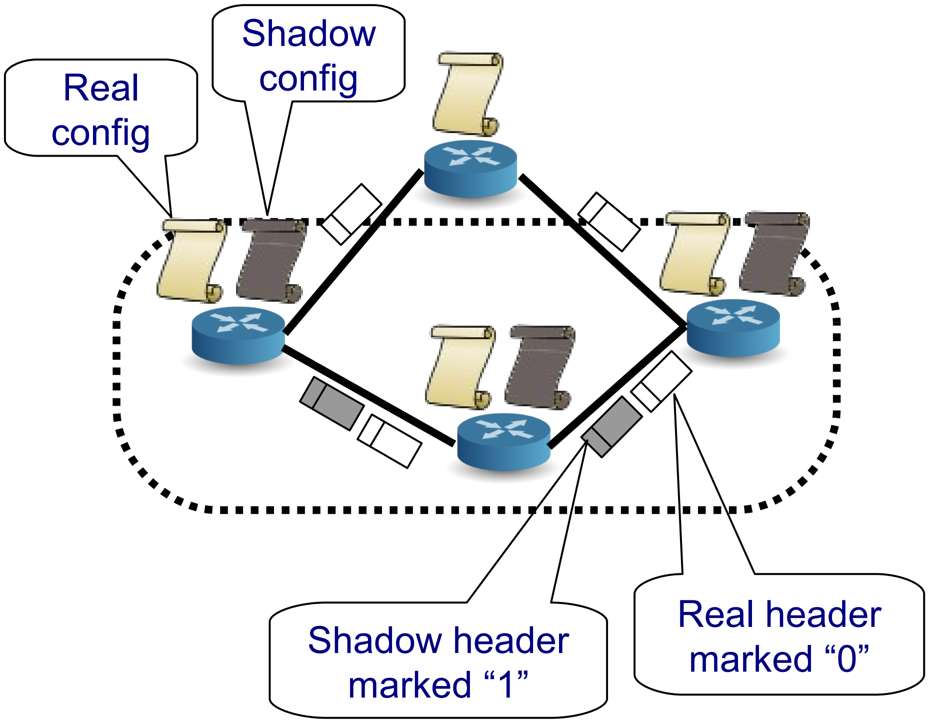
- Design and Architecture
- Performance Testing
- Transaction Support

Implementation and Evaluation

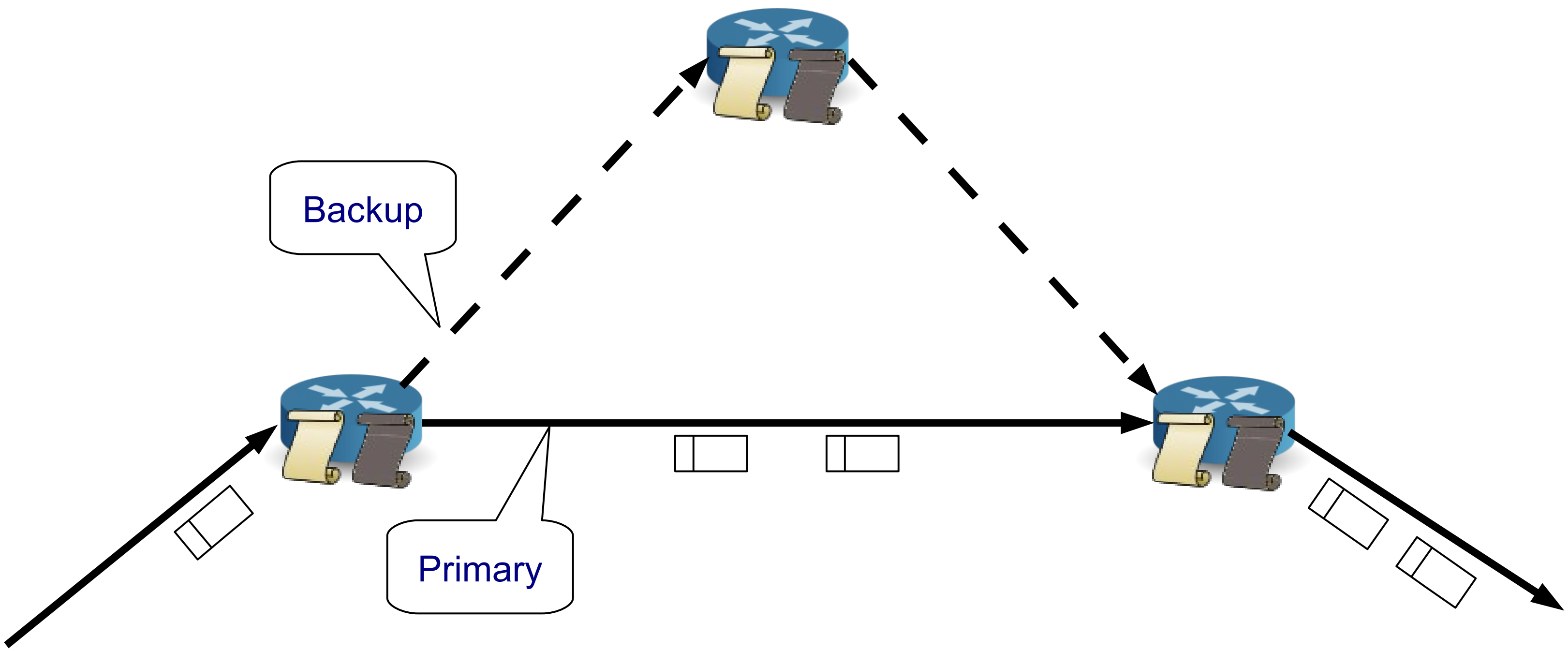
System Basics

What's in the shadow configuration?

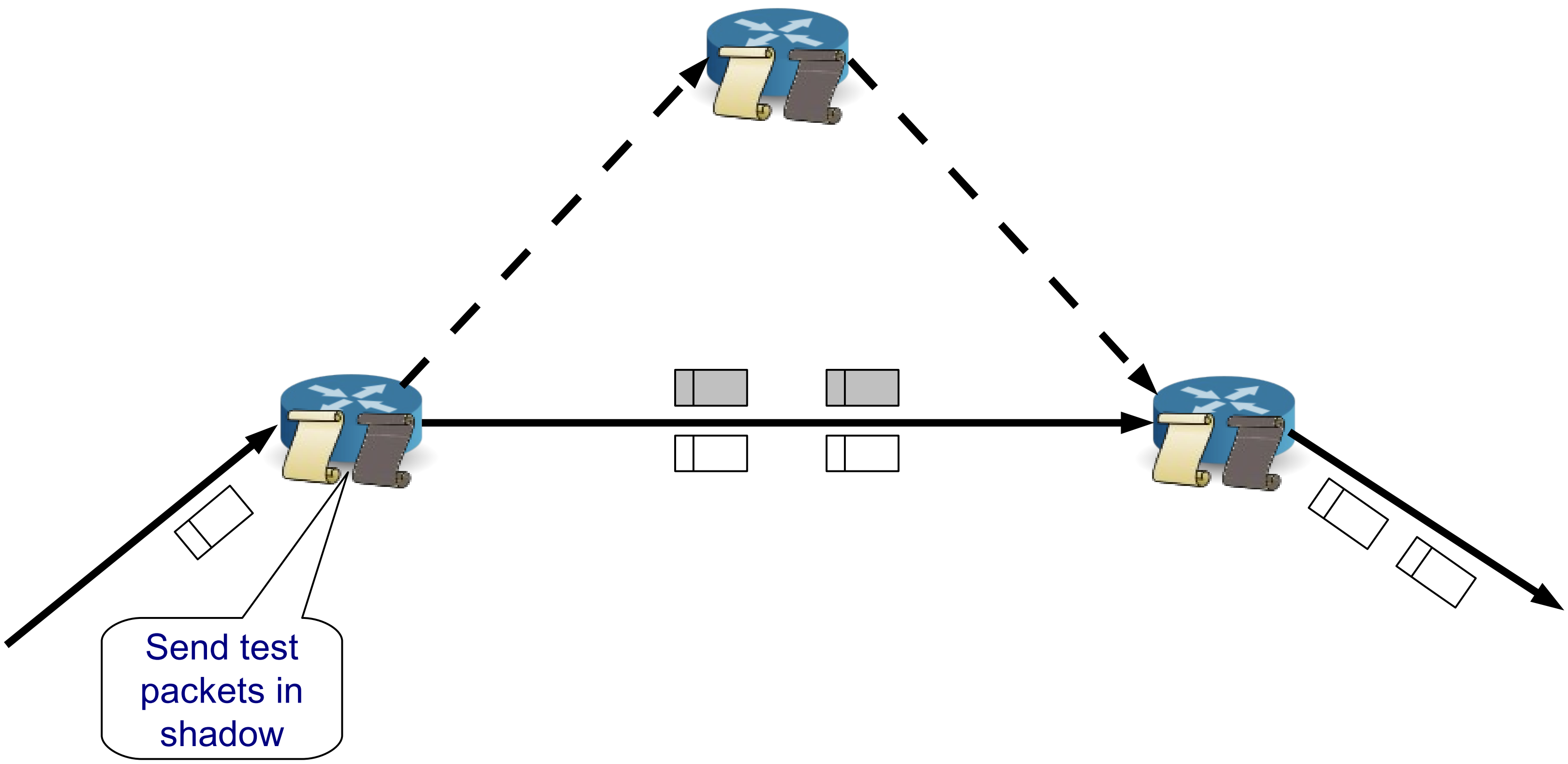
- Routing parameters
- ACLs
- Interface parameters
- VPNs
- QoS parameters



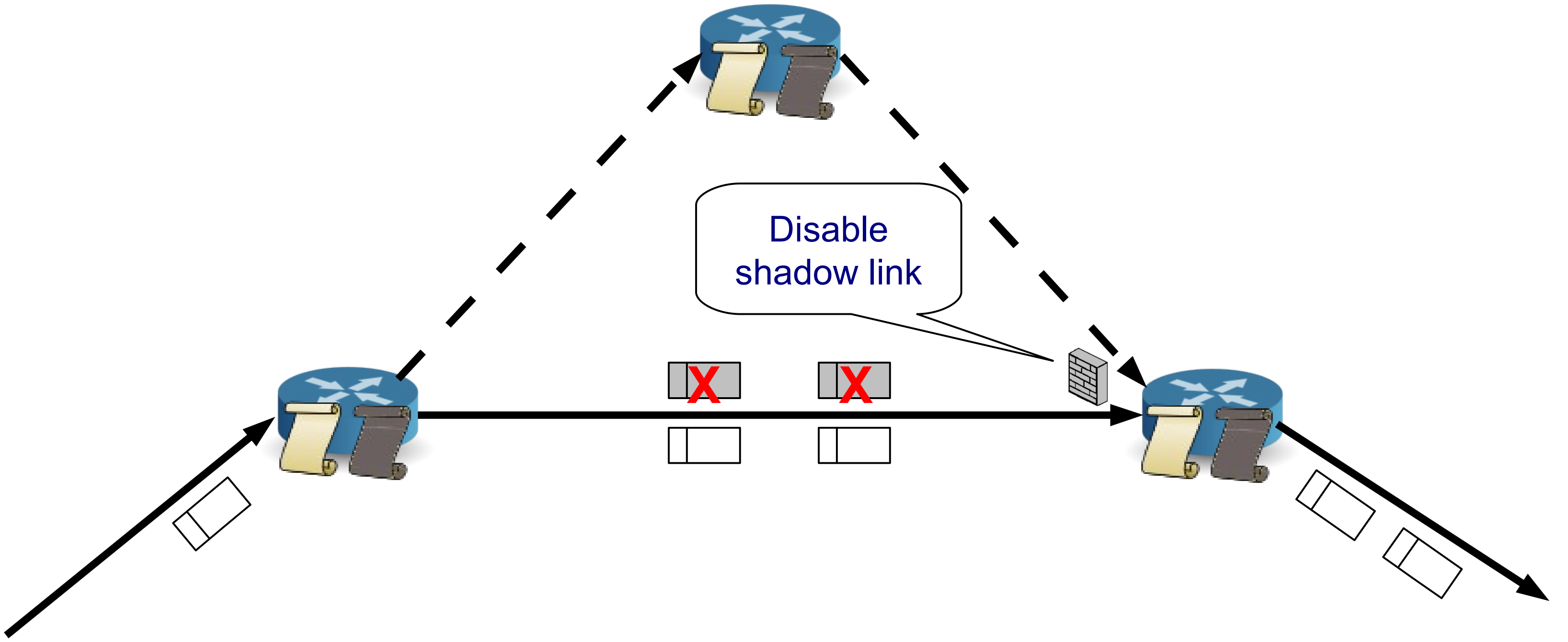
Example Usage Scenario: Backup Path Verification



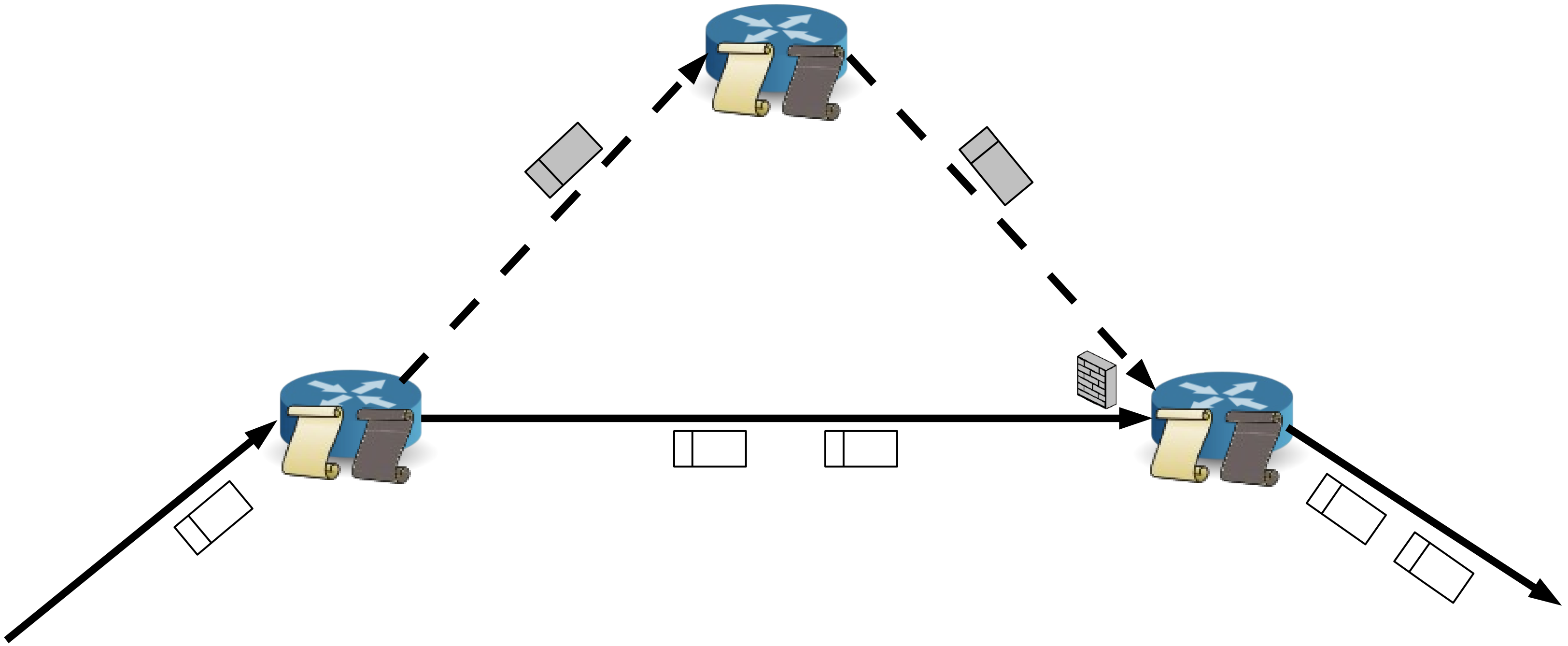
Example Usage Scenario: Backup Path Verification



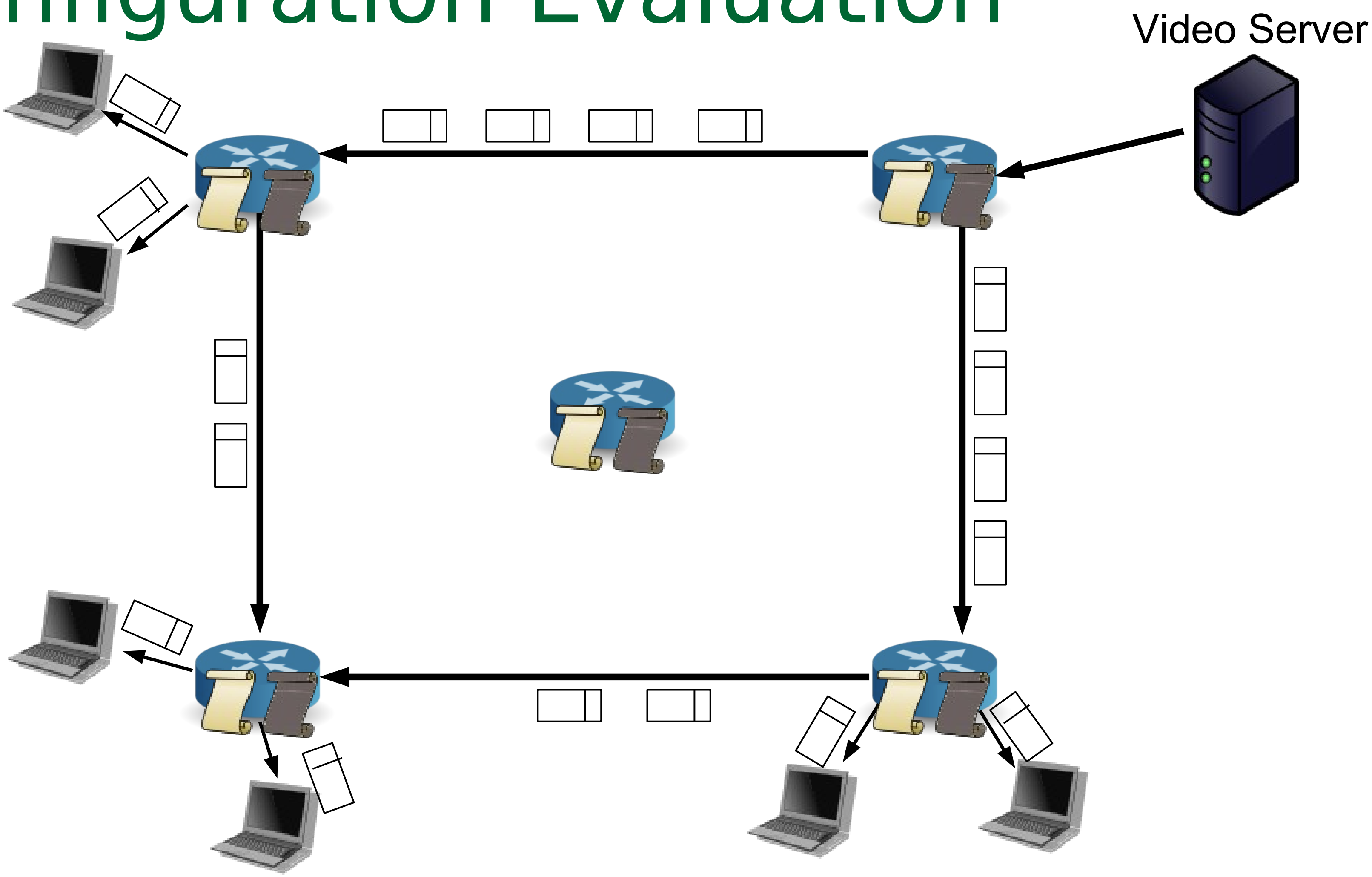
Example Usage Scenario: Backup Path Verification



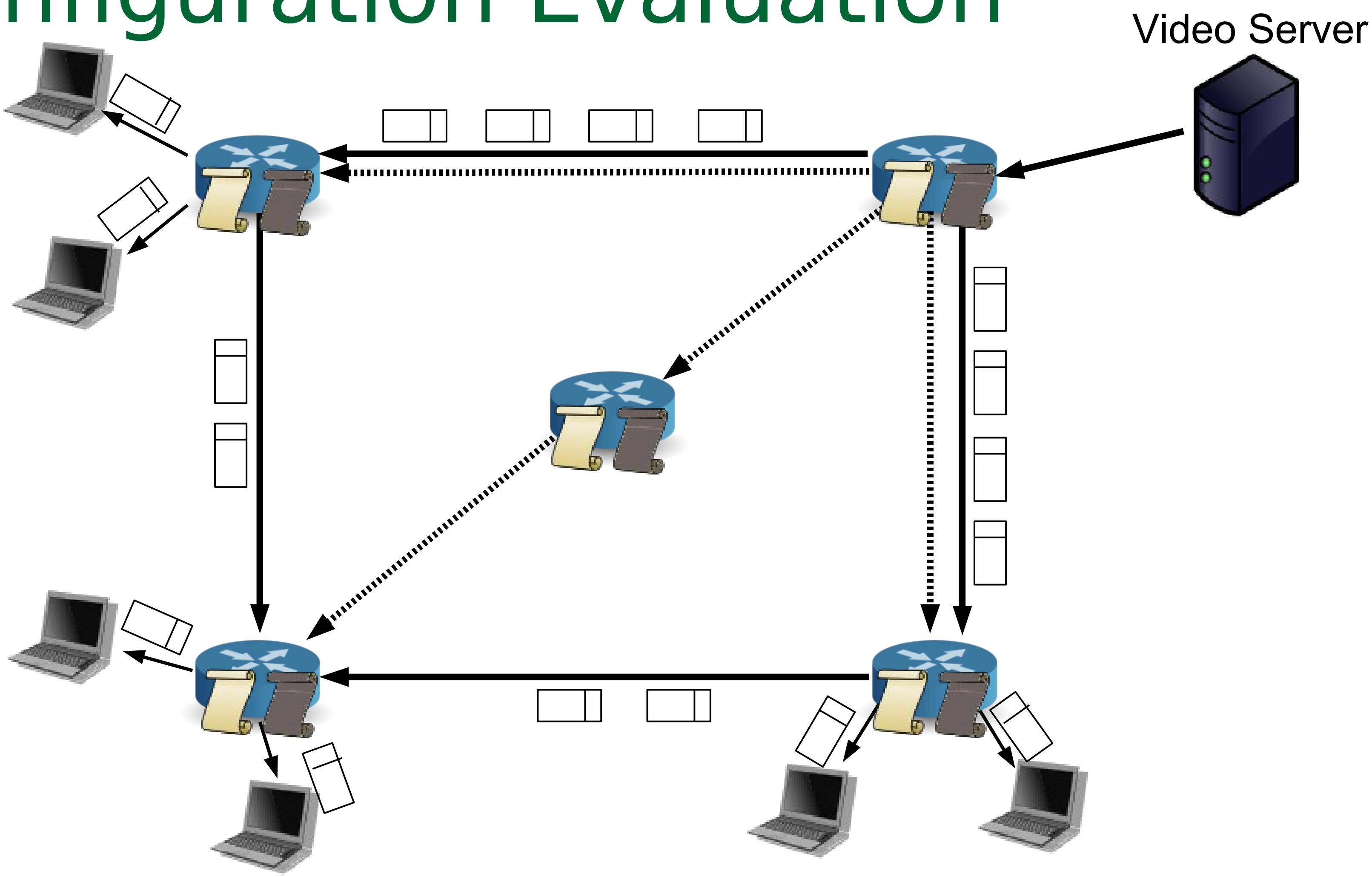
Example Usage Scenario: Backup Path Verification



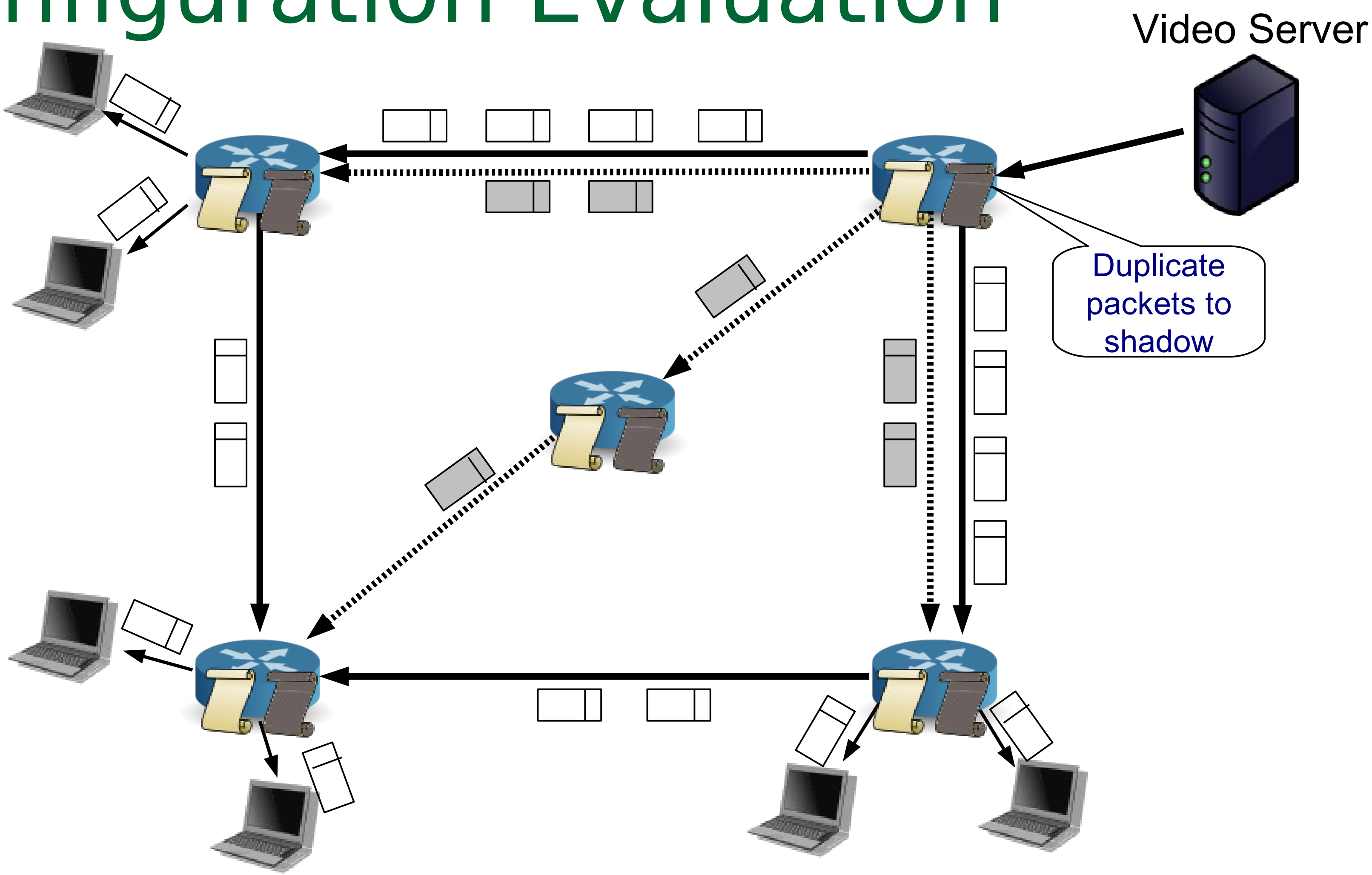
Example Usage Scenario: Configuration Evaluation



Example Usage Scenario: Configuration Evaluation



Example Usage Scenario: Configuration Evaluation



Roadmap

Motivation and Overview

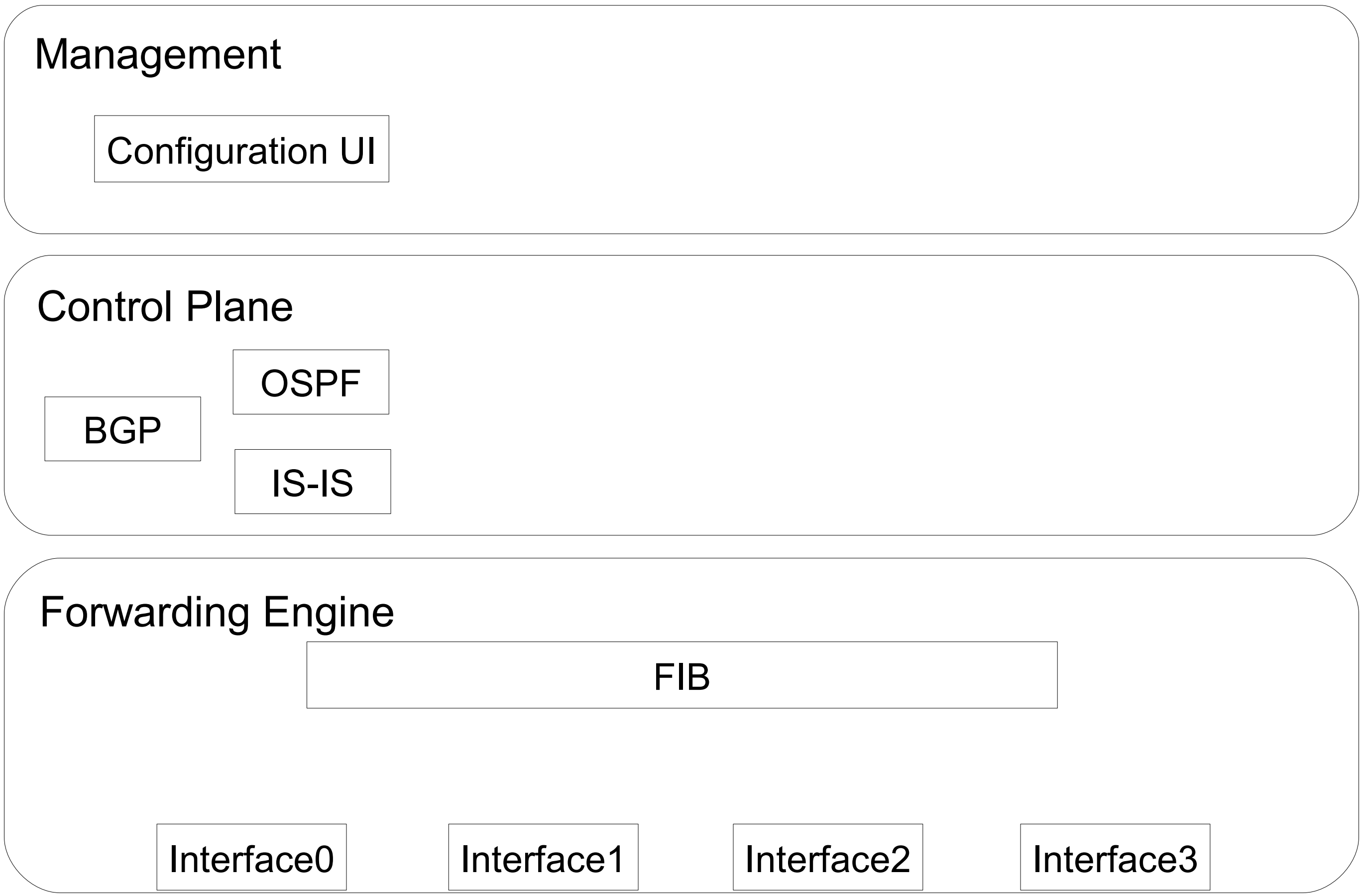
System Basics and Usage

System Components

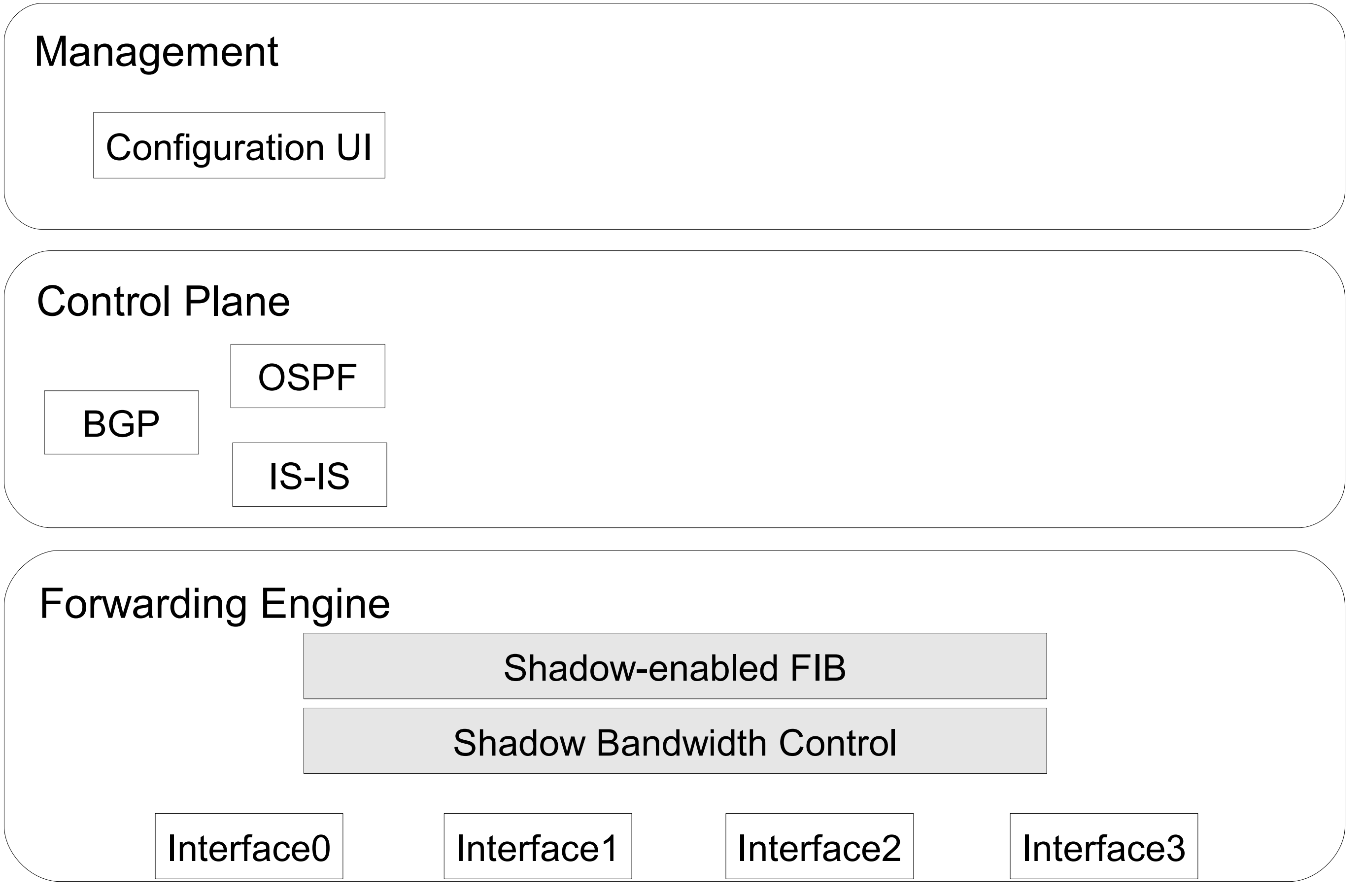
- ***Design and Architecture***
- ***Performance Testing***
- ***Transaction Support***

Implementation and Evaluation

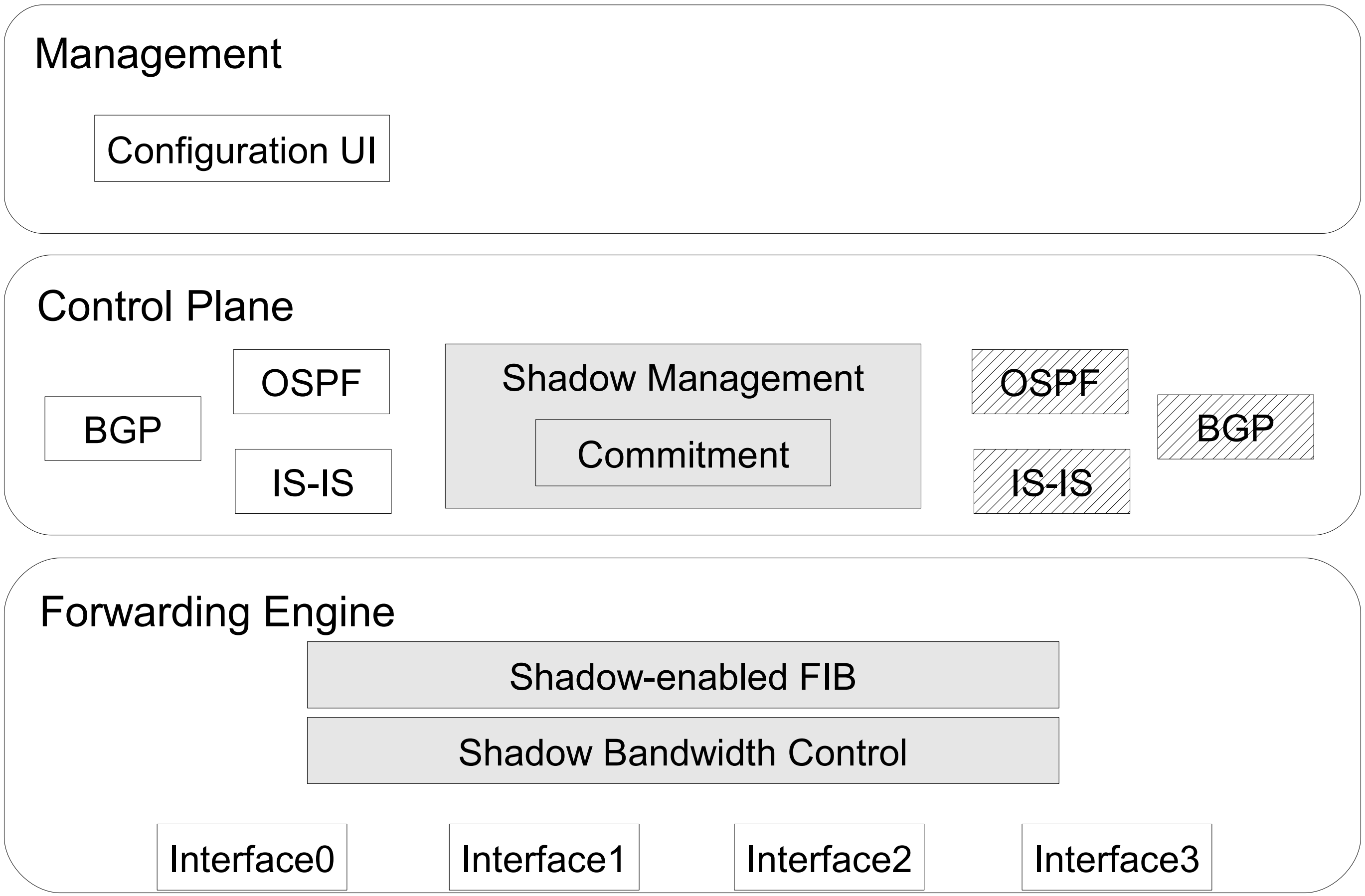
Design and Architecture



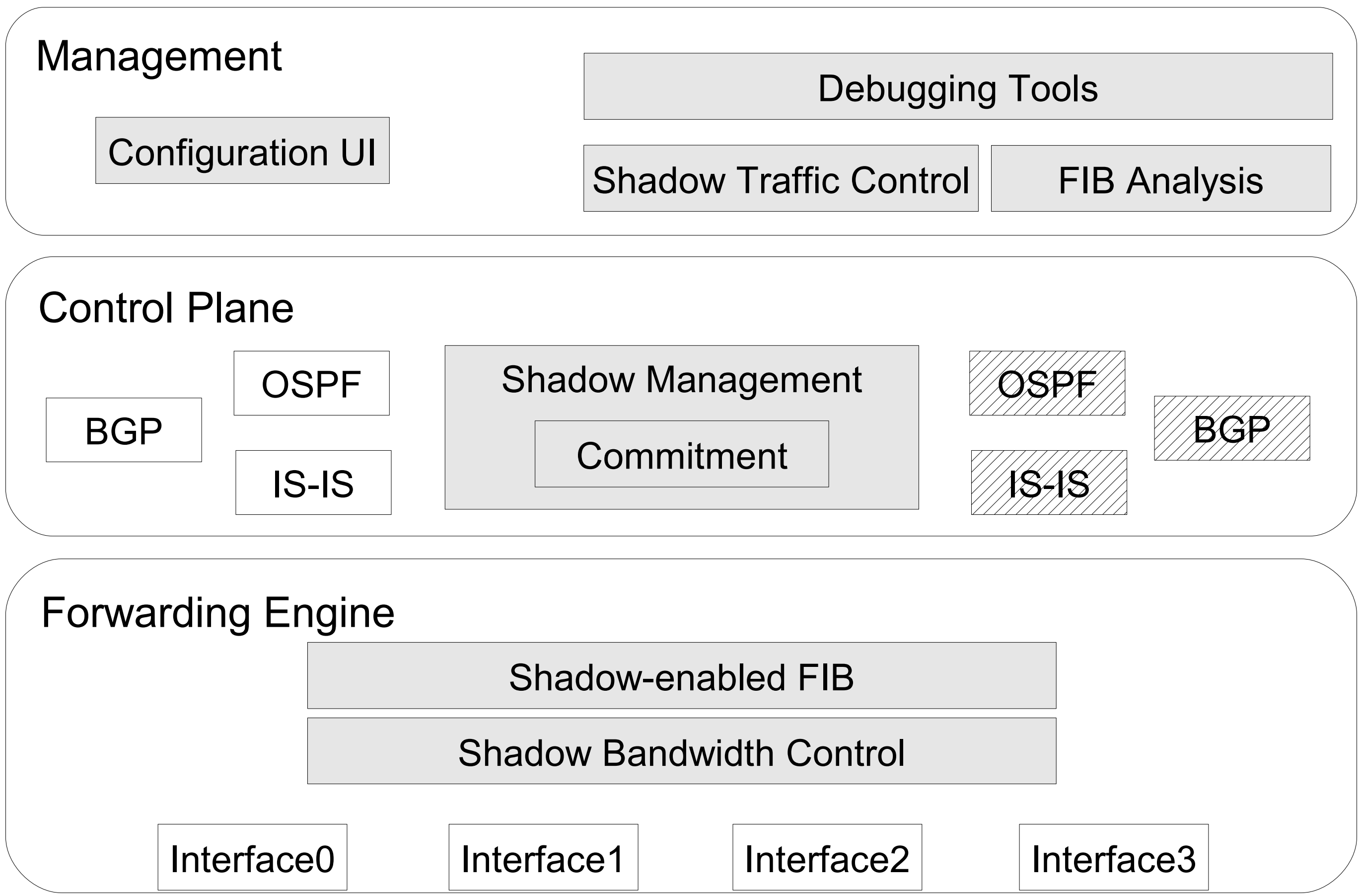
Design and Architecture



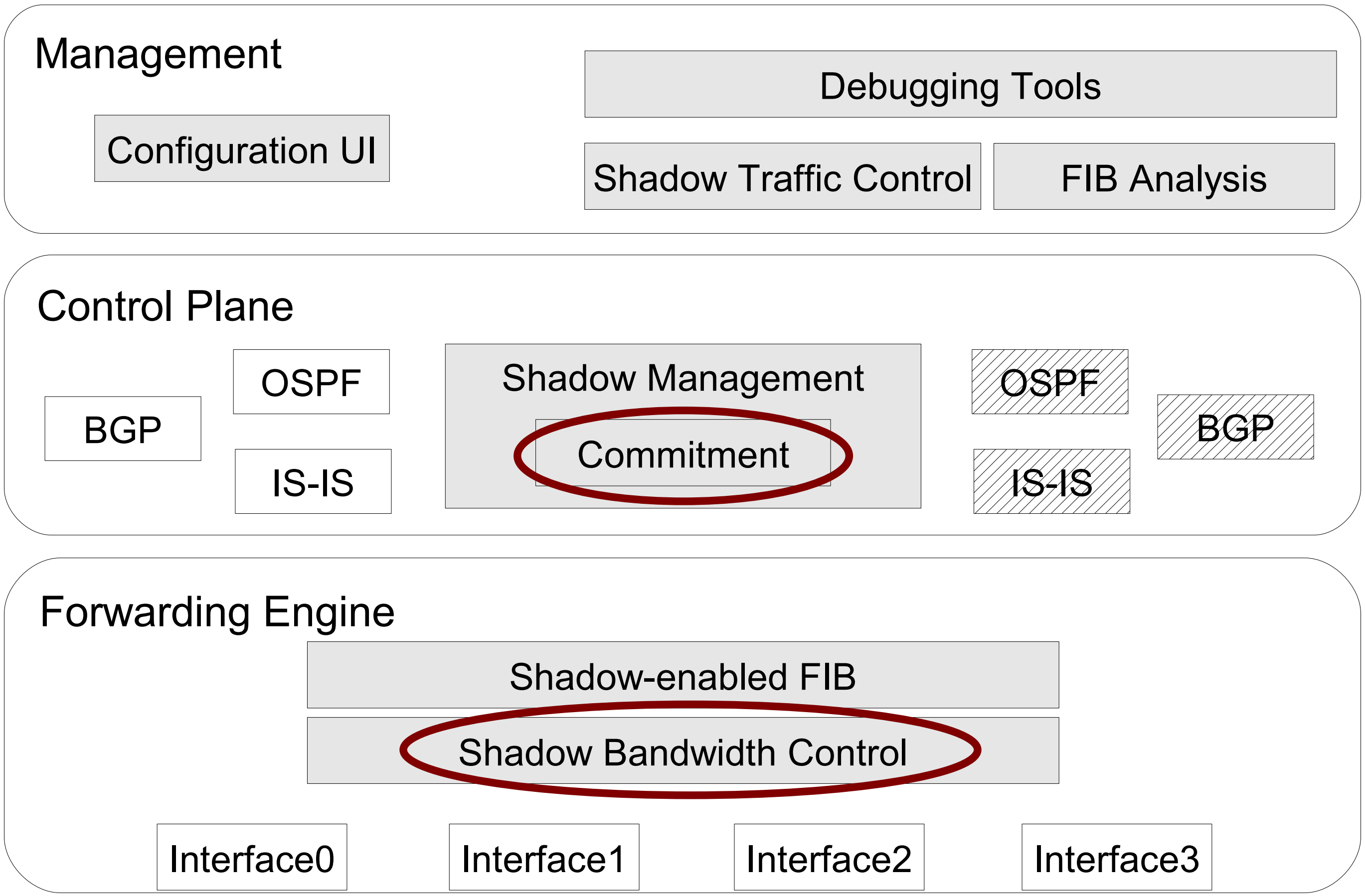
Design and Architecture



Design and Architecture



Design and Architecture



Shadow Bandwidth Control

Requirements

- Minimal impact on real traffic
- Accurate performance measurements of shadow configuration

Supported Modes

- Priority
- Bandwidth Partitioning
- Packet Cancellation

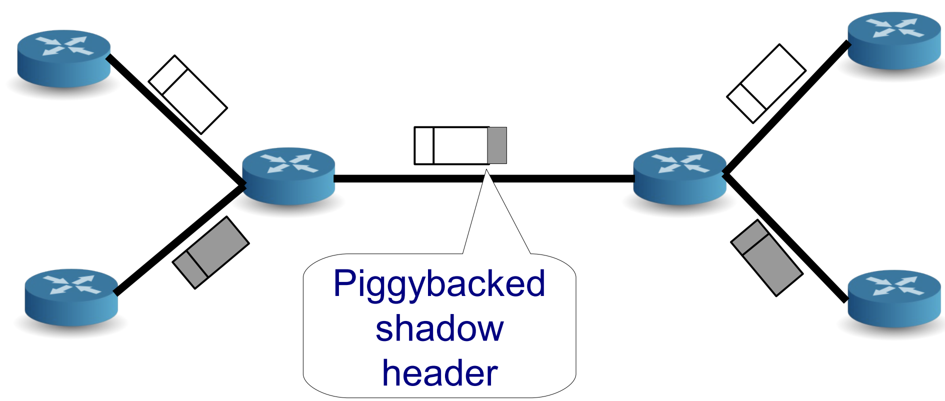
Packet Cancellation

Observation: in many network performance testing scenarios,

- Content of payload is not important
- Only payload size matters

Idea: only need headers for shadow traffic

Piggyback shadow headers on real packets

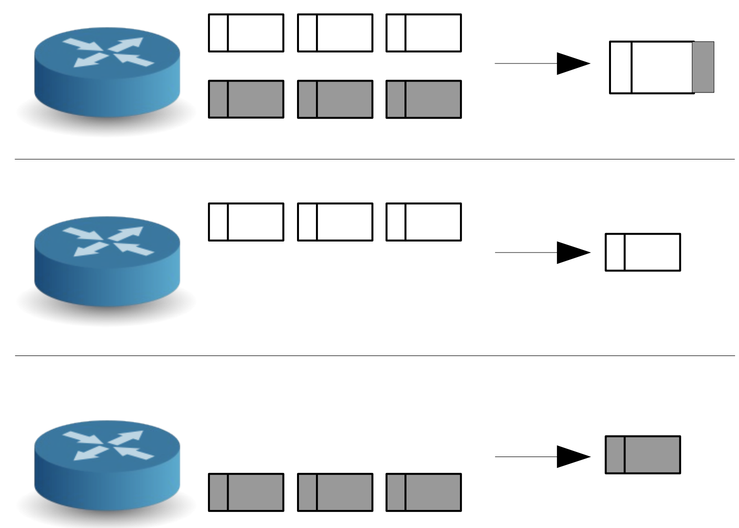


Packet Cancellation Details

Output interface maintains real and shadow queues

- Q_r and Q_s

```
pktsched() – packet cancellation and scheduling.
if not empty(Q_r) then
    p ← dequeue(Q_r) // Select real packet
    // Append shadow packet headers
    for 1 ... MAX_CANCELLABLE do
        if not virtual_clock_expired(peek(Q_s))
            break
        p ← append(p, ip_hdr(dequeue(Q_s)))
    endfor
    transmit(p)
elseif not empty(Q_s) then
    // Send shadow packet if available
    if virtual_clock_expired(peek(Q_s))
        transmit(dequeue(Q_s))
endif
```

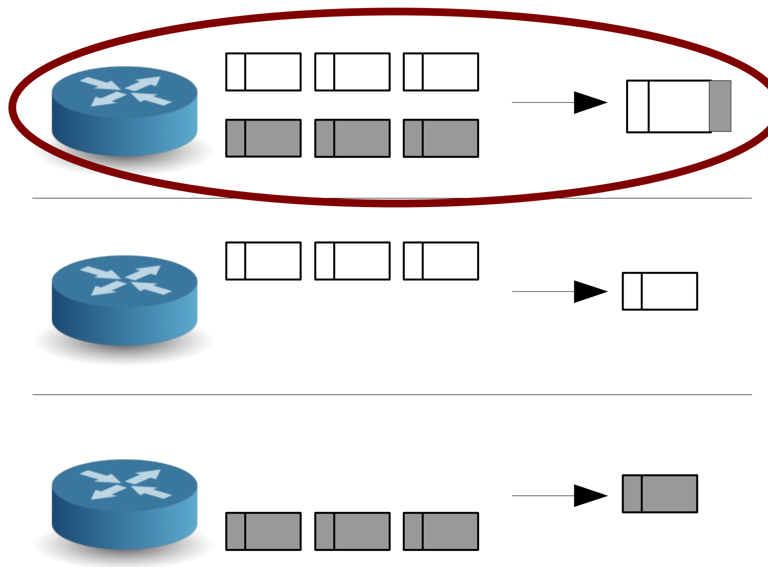


Packet Cancellation Details

Output interface maintains real and shadow queues

- Q_r and Q_s

```
pktsched() – packet cancellation and scheduling.
if not empty(Q_r) then
    p ← dequeue(Q_r) // Select real packet
    // Append shadow packet headers
    for 1 ... MAX_CANCELLABLE do
        if not virtual_clock_expired(peek(Q_s))
            break
        p ← append(p, ip_hdr(dequeue(Q_s)))
    endfor
    transmit(p)
elseif not empty(Q_s) then
    // Send shadow packet if available
    if virtual_clock_expired(peek(Q_s))
        transmit(dequeue(Q_s))
endif
```



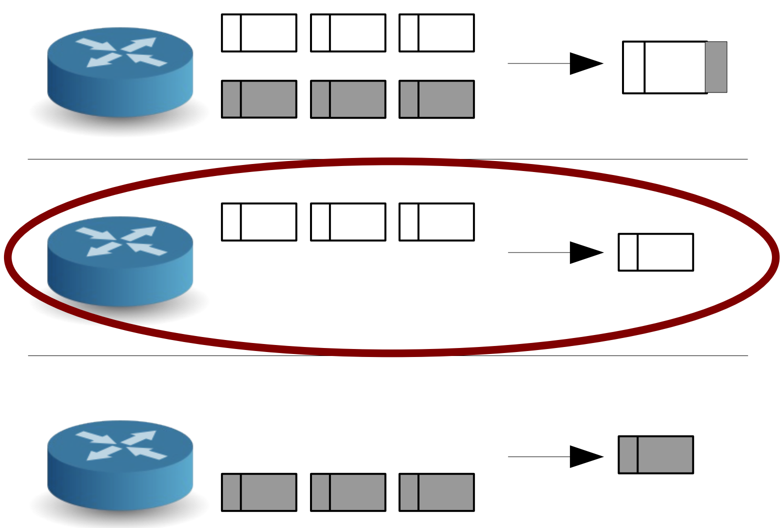
Packet Cancellation Details

Output interface maintains real and shadow queues

- Q_r and Q_s

pktsched() – packet cancellation and scheduling.

```
if not empty(Q_r) then
    p ← dequeue(Q_r) // Select real packet
    // Append shadow packet headers
    for 1 ... MAX_CANCELLABLE do
        if not virtual_clock_expired(peek(Q_s))
            break
        p ← append(p, ip_hdr(dequeue(Q_s)))
    endfor
    transmit(p)
elseif not empty(Q_s) then
    // Send shadow packet if available
    if virtual_clock_expired(peek(Q_s))
        transmit(dequeue(Q_s))
endif
```



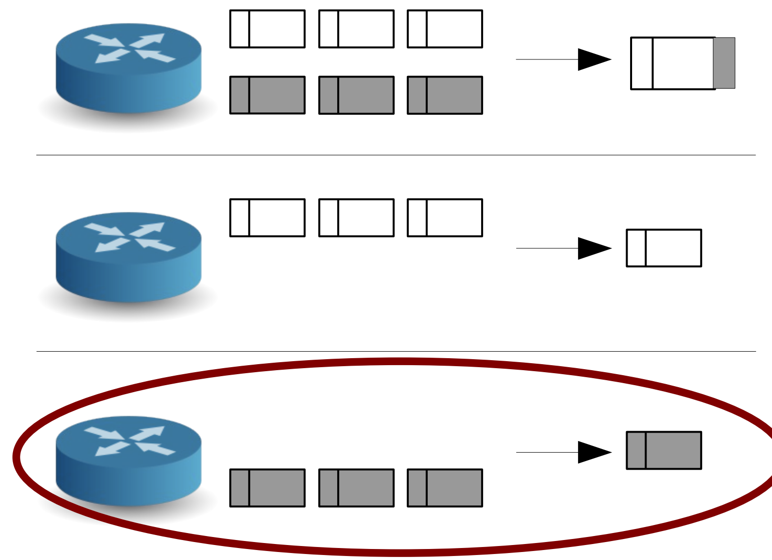
Packet Cancellation Details

Output interface maintains real and shadow queues

- Q_r and Q_s

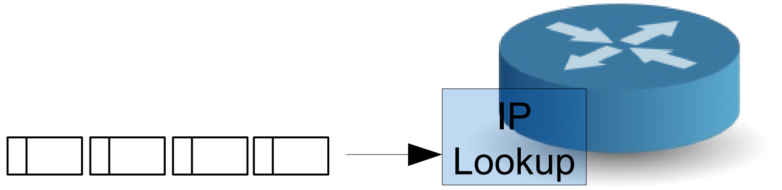
pktsched() – packet cancellation and scheduling.

```
if not empty(Q_r) then
    p ← dequeue(Q_r) // Select real packet
    // Append shadow packet headers
    for 1 ... MAX_CANCELLABLE do
        if not virtual_clock_expired(peek(Q_s))
            break
        p ← append(p, ip_hdr(dequeue(Q_s)))
    endfor
    transmit(p)
elseif not empty(Q_s) then
    // Send shadow packet if available
    if virtual_clock_expired(peek(Q_s))
        transmit(dequeue(Q_s))
endif
```

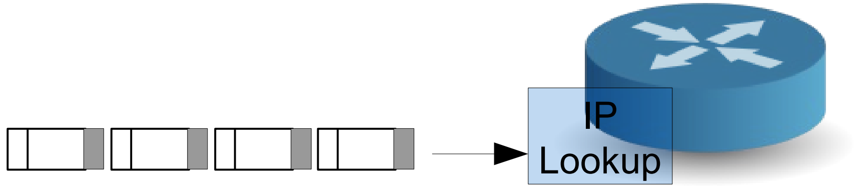


Forwarding Overhead

Without Packet Cancellation:



With Packet Cancellation:



Cancellation may require routers to process more packets.
Can routers support it?

Forwarding Overhead Analysis

Routers can be designed for worst-case

- L : Link speed
- K_{min} : Minimum packet size
- Router supports $\alpha \frac{L}{K_{min}}$ packets per second

Load typically measured by link utilization

- α_r : Utilization due to real traffic (packet sizes k_r)
- α_s : Utilization due to shadow traffic (packet sizes k_s)

We require:

$$\mathbb{E}\left[\frac{\alpha_r L}{k_r}\right] + \mathbb{E}\left[\frac{\alpha_s L}{k_s}\right] < \alpha \frac{L}{K_{min}}$$

Forwarding Overhead Analysis

Routers can be designed for worst-case

- L : Link speed
- K_{min} : Minimum packet size
- Router supports $\alpha \frac{L}{K_{min}}$ packets per second

Load typically measured by link utilization

- α_r : Utilization due to real traffic (packet sizes k_r)
- α_s : Utilization due to shadow traffic (packet sizes k_s)

We require:

$$\mathbb{E}\left[\frac{\alpha_r L}{k_r}\right] + \mathbb{E}\left[\frac{\alpha_s L}{k_s}\right] < \alpha \frac{L}{K_{min}}$$

Example:
With α = 70%, and 80% real traffic utilization
Support up to ***75% shadow traffic utilization***

Commitment

Objectives

- Smoothly swap real and shadow across network
  - Eliminate effects of reconvergence due to config changes
- Easy to swap back

Commitment

Objectives

- Smoothly swap real and shadow across network
  - Eliminate effects of reconvergence due to config changes
- Easy to swap back

Issue

- Packet marked with *shadow* bit
  - 0 = Real, 1 = Shadow
- Shadow bit determines which FIB to use
- Routers swap FIBs asynchronously
- Inconsistent FIBs applied on the path

Commitment Protocol

Idea: Use tags to achieve consistency

- Temporary identifiers

Basic algorithm has 4 phases

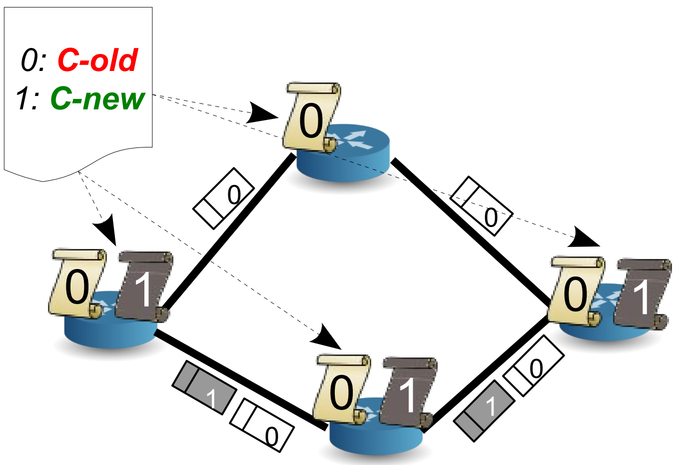
Commitment Protocol

Idea: Use tags to achieve consistency

- Temporary identifiers

Basic algorithm has 4 phases

- Distribute tags for each config
  - **C-old** for current real config
  - **C-new** for current shadow config



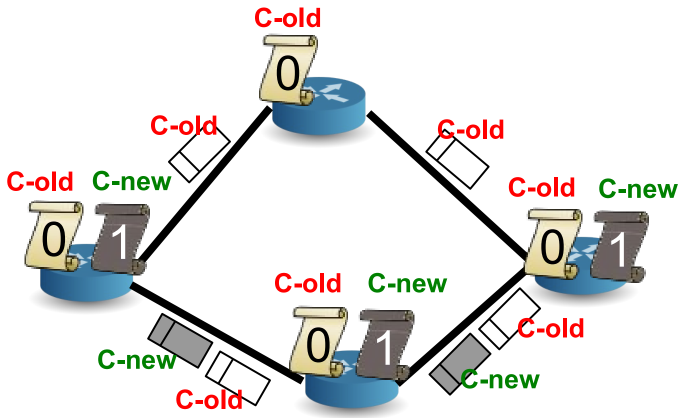
Commitment Protocol

Idea: Use tags to achieve consistency

- Temporary identifiers

Basic algorithm has 4 phases

- Distribute tags for each config
  - **C-old** for current real config
  - **C-new** for current shadow config
- Routers mark packets with tags
  - Packets forwarded according to tags



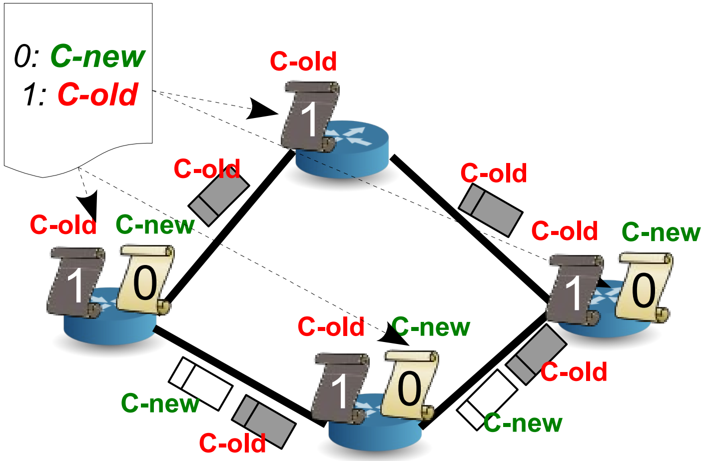
Commitment Protocol

Idea: Use tags to achieve consistency

- Temporary identifiers

Basic algorithm has 4 phases

- Distribute tags for each config
  - **C-old** for current real config
  - **C-new** for current shadow config
- Routers mark packets with tags
  - Packets forwarded according to tags
- Swap configs (tags still valid)



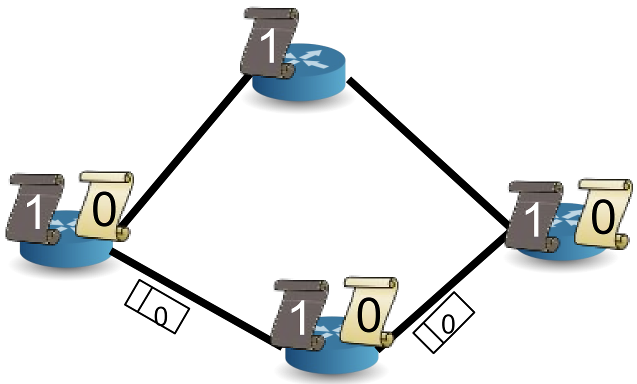
Commitment Protocol

Idea: Use tags to achieve consistency

- Temporary identifiers

Basic algorithm has 4 phases

- Distribute tags for each config
  - **C-old** for current real config
  - **C-new** for current shadow config
- Routers mark packets with tags
  - Packets forwarded according to tags
- Swap configs (tags still valid)
- Remove tags from packets
  - Resume use of shadow bit



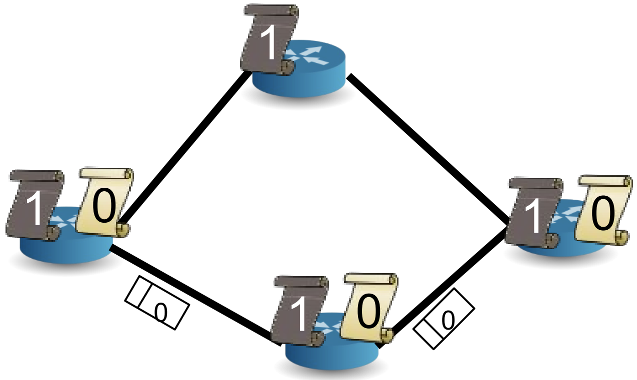
Commitment Protocol

Idea: Use tags to achieve consistency

- Temporary identifiers

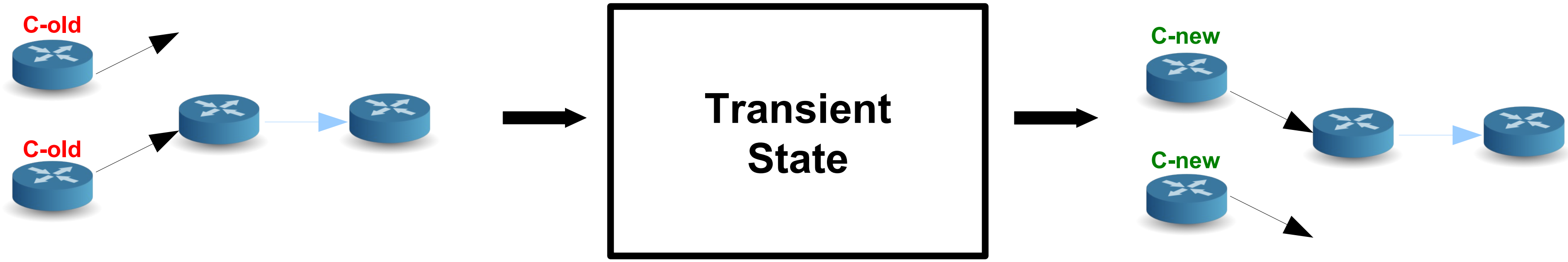
Basic algorithm has 4 phases

- Distribute tags for each config
  - **C-old** for current real config
  - **C-new** for current shadow config
- Routers mark packets with tags
  - Packets forwarded according to tags
- Swap configs (tags still valid)
- Remove tags from packets
  - Resume use of shadow bit



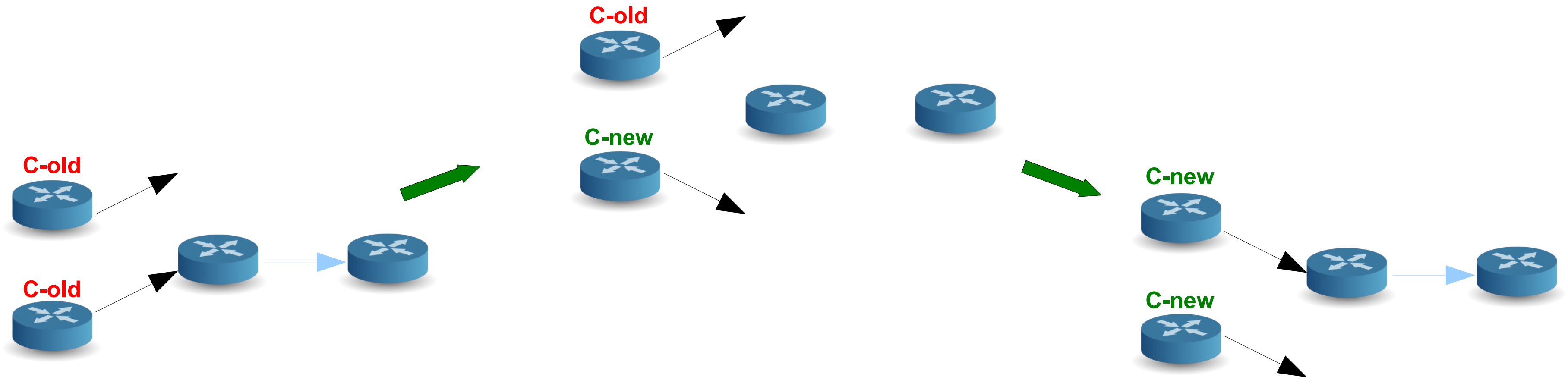
Transient States

Definition: State in which some packets use **C-old** and others use **C-new**.



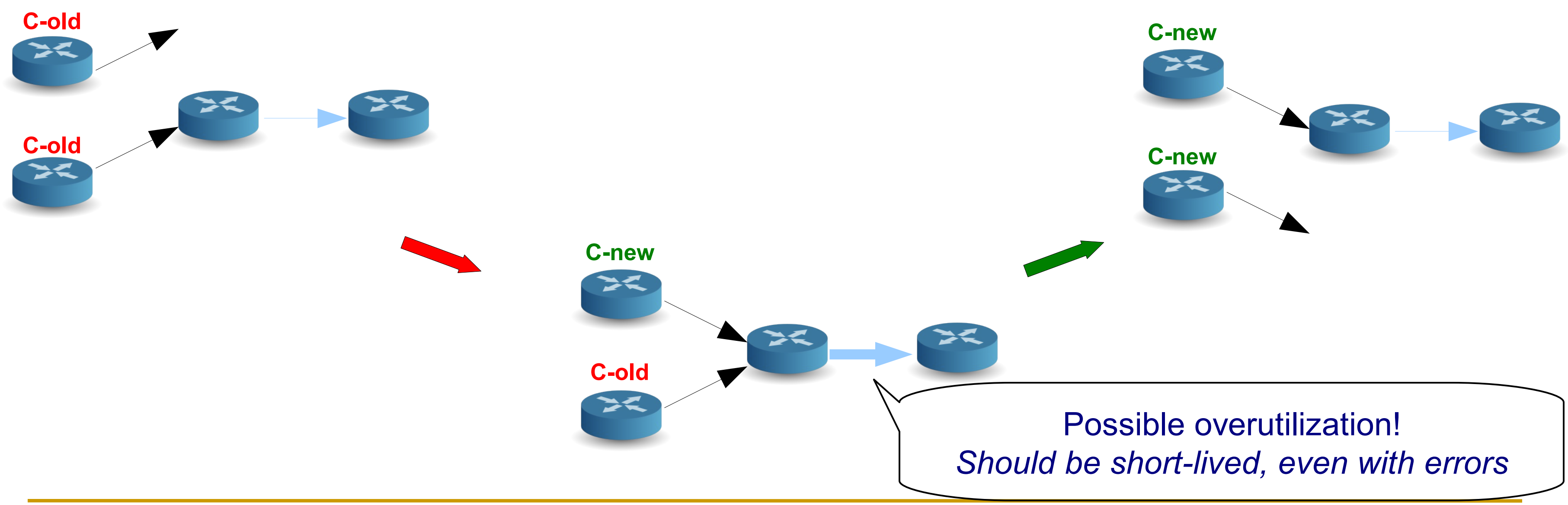
Transient States

Definition: State in which some packets use **C-old** and others use **C-new**.



Transient States

Definition: State in which some packets use **C-old** and others use **C-new**.

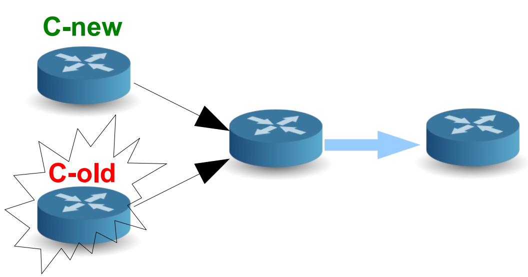


Error Recovery During Swap

If ACK missing from at least one router, two cases:

(a) Router completed SWAP but ACK not sent

(b) Router did not complete SWAP ***Transient State***



Error Recovery During Swap

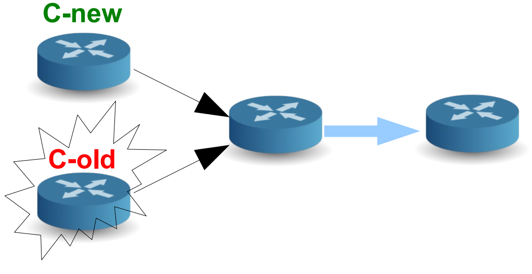
If ACK missing from at least one router, two cases:

(a) Router completed SWAP but ACK not sent

(b) Router did not complete SWAP ***Transient State***

Detect (b) and rollback quickly

- Querying router directly may be impossible



Error Recovery During Swap

If ACK missing from at least one router, two cases:

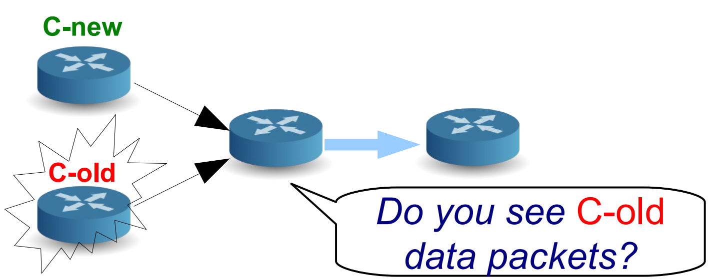
(a) Router completed SWAP but ACK not sent

(b) Router did not complete SWAP ***Transient State***

Detect (b) and rollback quickly

- Querying router directly may be impossible

Solution: Ask neighboring routers



If YES:

Case (b): rollback other routers

Otherwise,

Case (a): no transient state

Roadmap

Motivation and Overview

System Basics and Usage

System Components

- Design and Architecture
- Performance Testing
- Transaction Support

Implementation and Evaluation

Implementation

Kernel-level (based on Linux 2.6.22.9)

- TCP/IP stack support
- FIB management
- Commitment hooks
- Packet cancellation

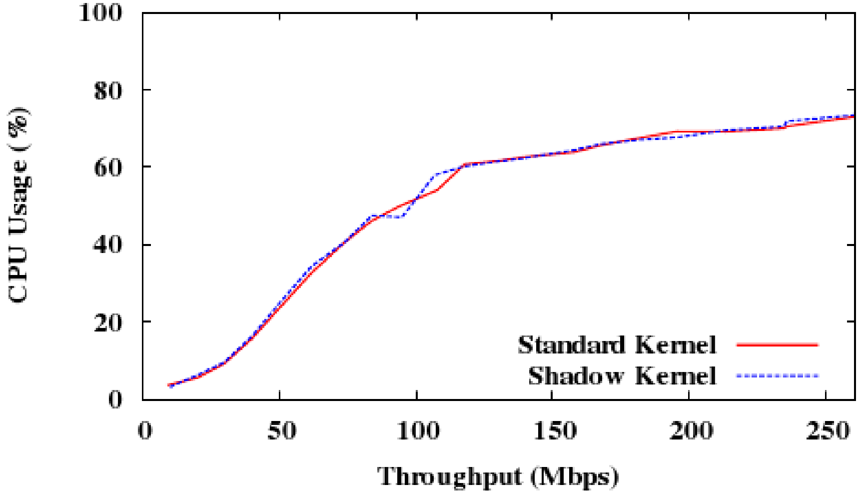
Tools

- Transparent software router support (Quagga + XORP)
- Full commitment protocol
- Configuration UI (command-line based)

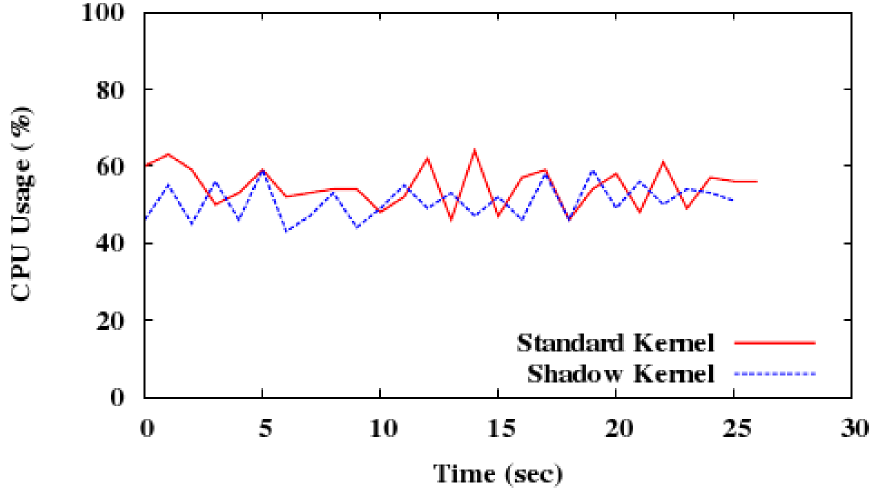
Evaluated on Emulab (3Ghz HT CPUs)

Evaluation: CPU Overhead

Static FIB
- 300B pkts
- No route caching

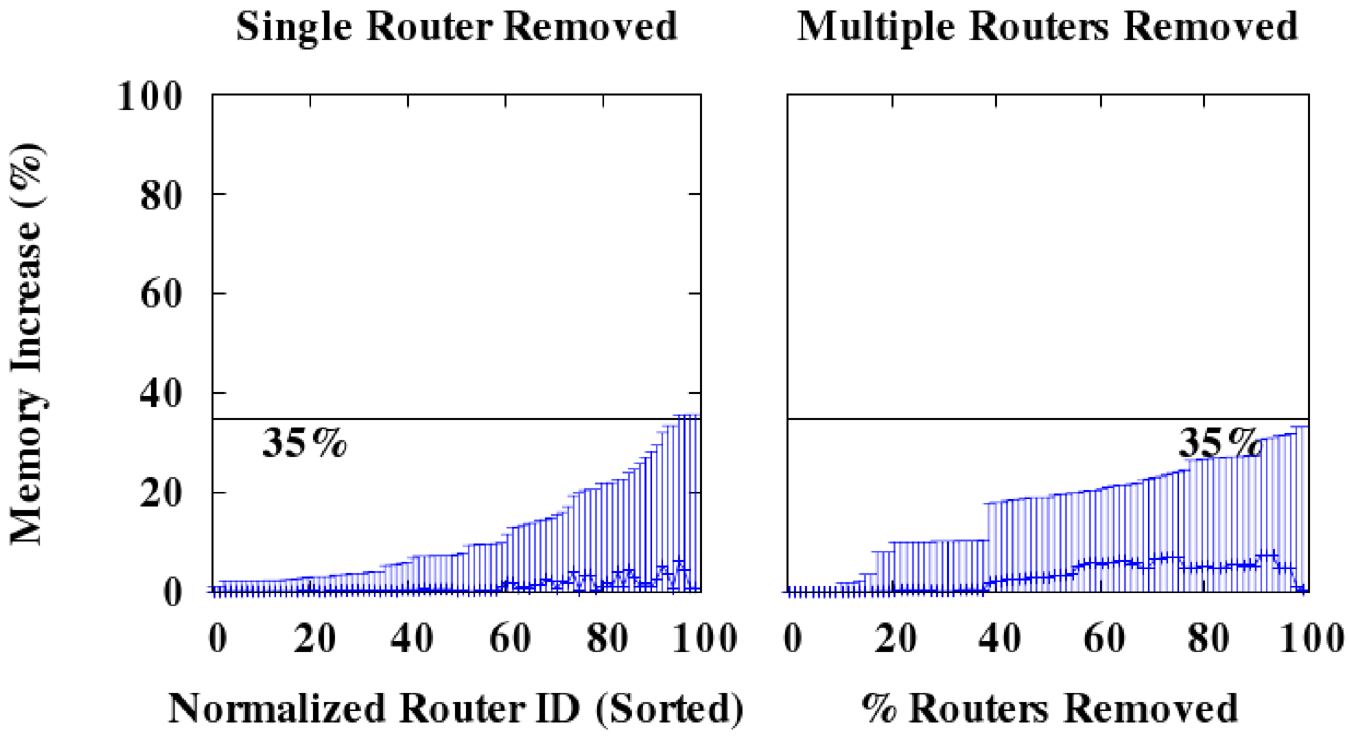


With FIB updates
- 300B pkts @ 100Mbps
- 1-100 updates/sec
- No route caching

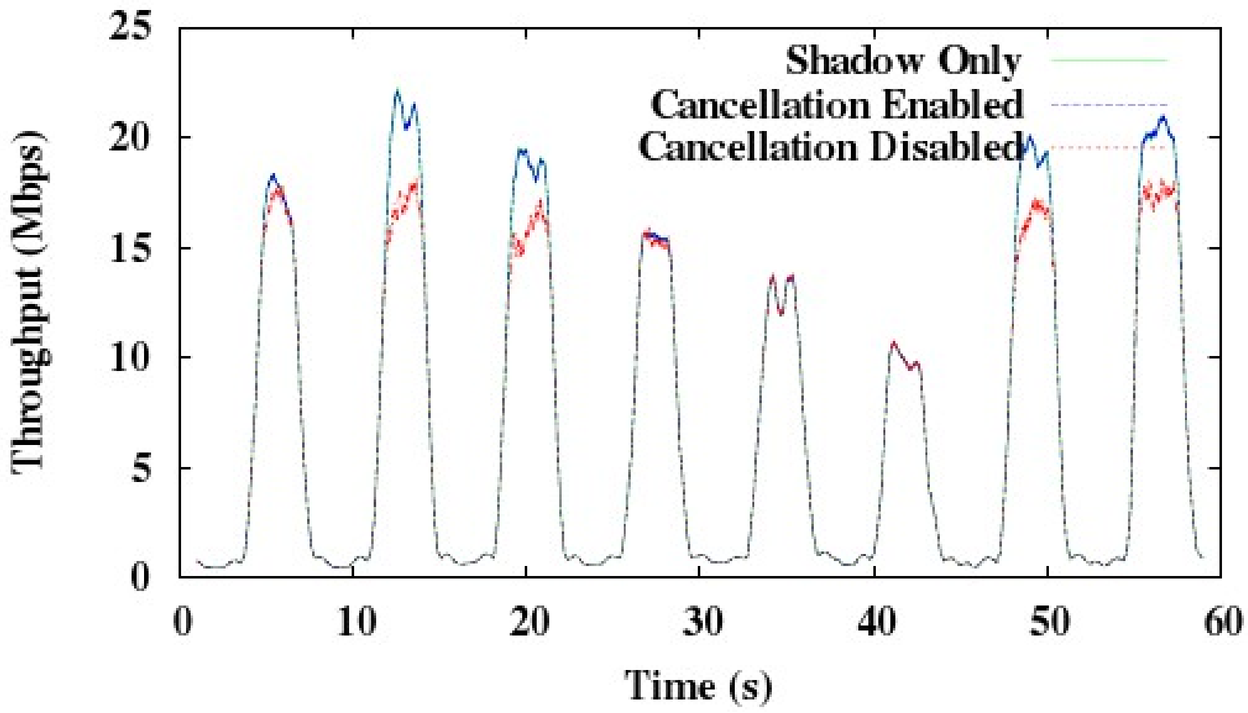


Evaluation: Memory Overhead

FIB storage overhead for US Tier-1 ISP



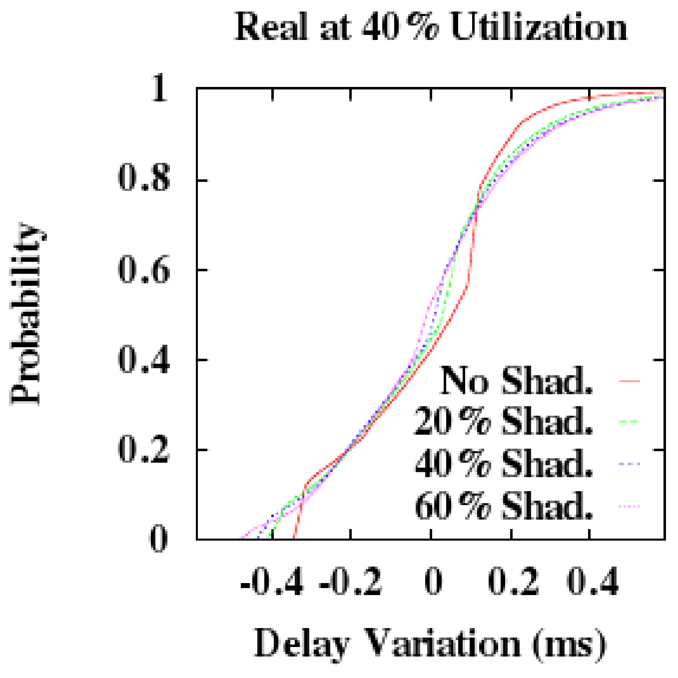
Evaluation: Packet Cancellation



Accurate streaming throughput measurement

- Abilene topology
- Real transit traffic duplicated to shadow
- Video streaming traffic in shadow

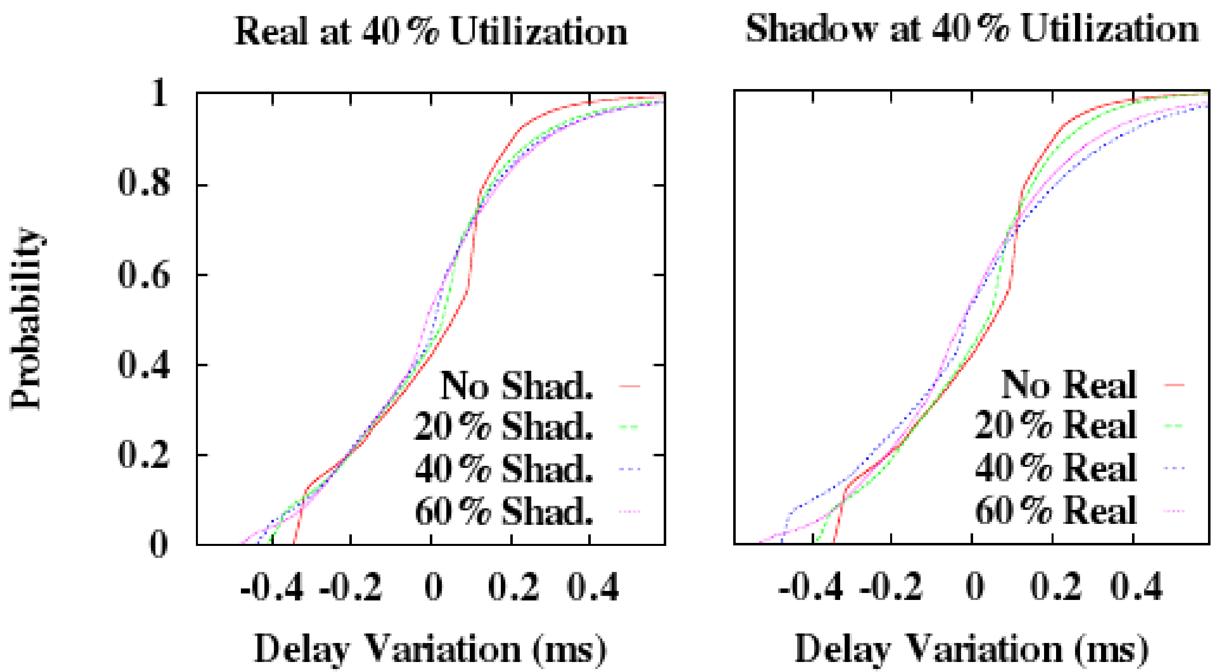
Evaluation: Packet Cancellation



Limited interaction of real and shadow

- Intersecting real and shadow flows
  - CAIDA traces
- Vary flow utilizations

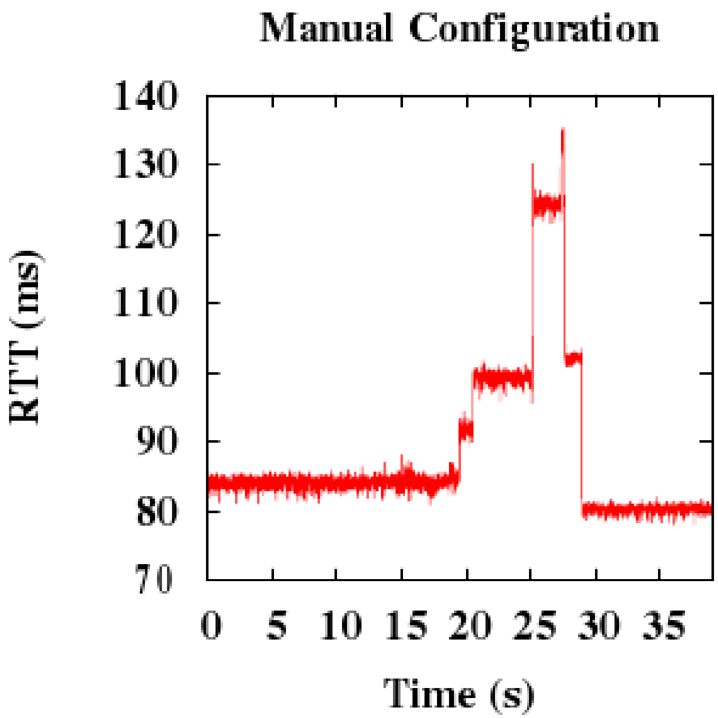
Evaluation: Packet Cancellation



Limited interaction of real and shadow

- Intersecting real and shadow flows
  - CAIDA traces
- Vary flow utilizations

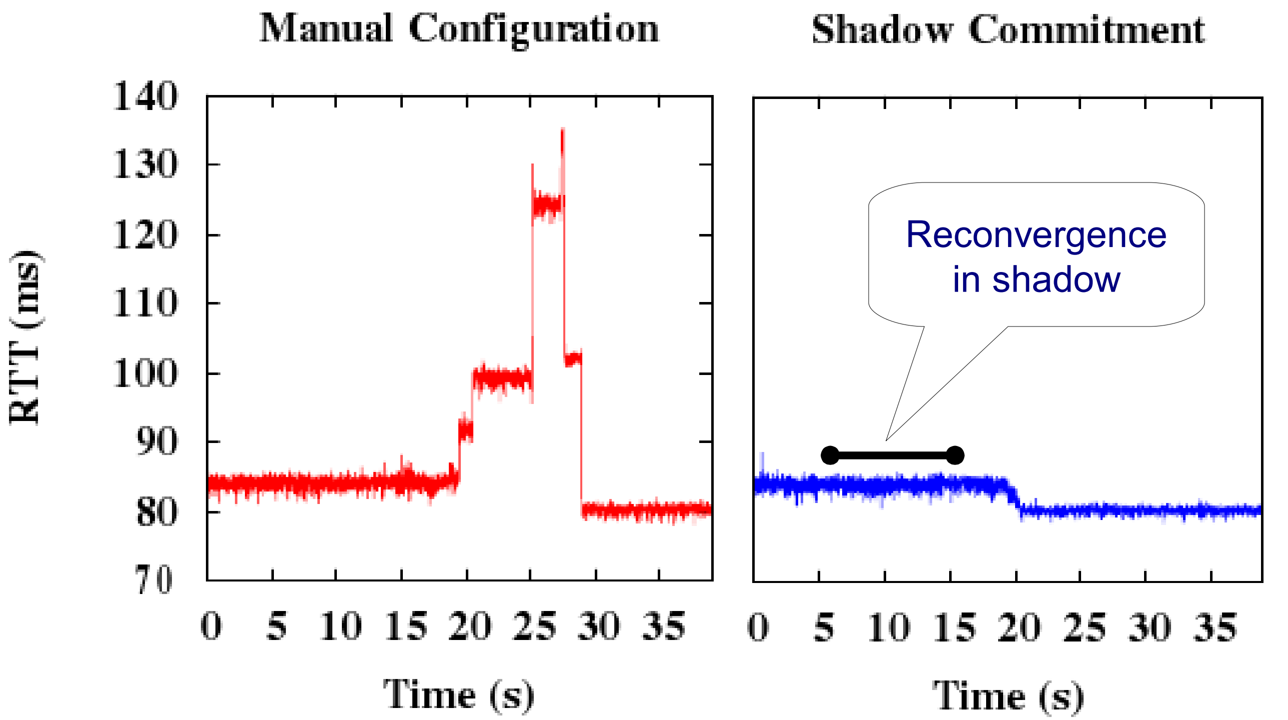
Evaluation: Commitment



Applying OSPF link-weight changes

- Abilene topology with 3 external peers
  - Configs translated to Quagga syntax
  - Abilene BGP dumps

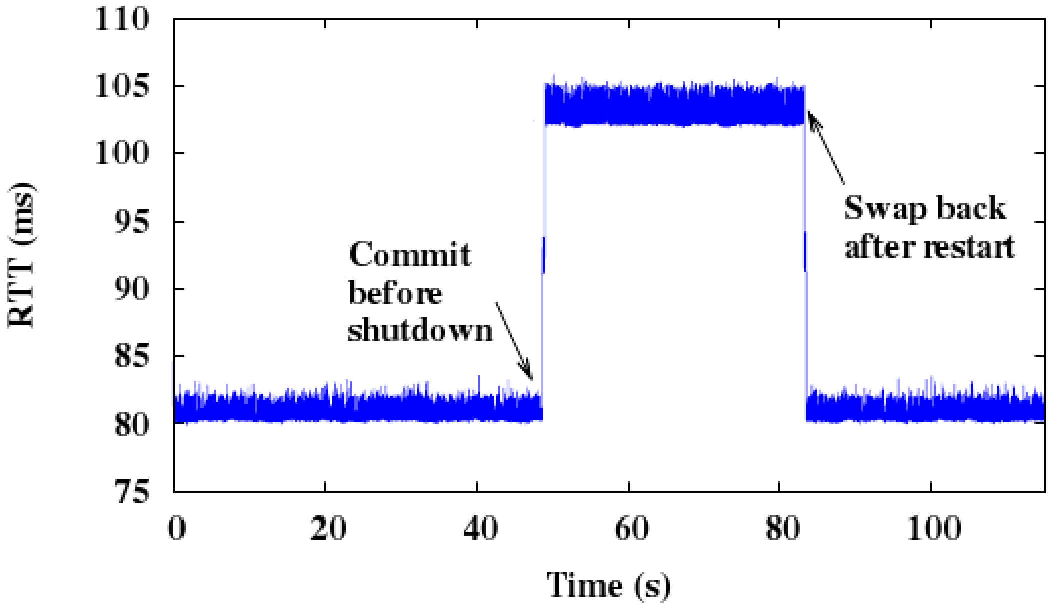
Evaluation: Commitment



Applying OSPF link-weight changes

- Abilene topology with 3 external peers
  - Configs translated to Quagga syntax
  - Abilene BGP dumps

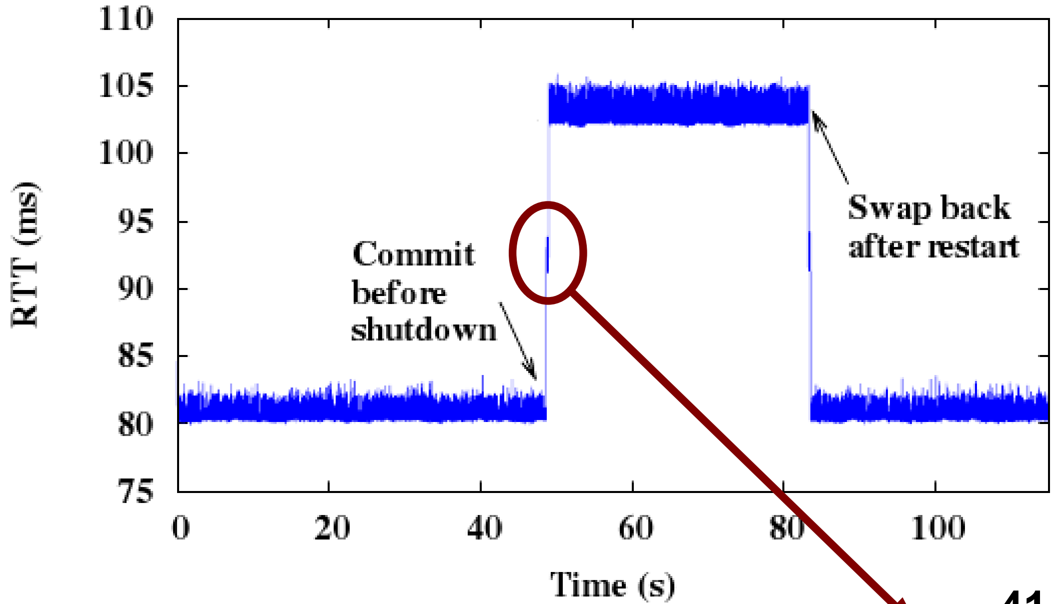
Evaluation: Router Maintenance

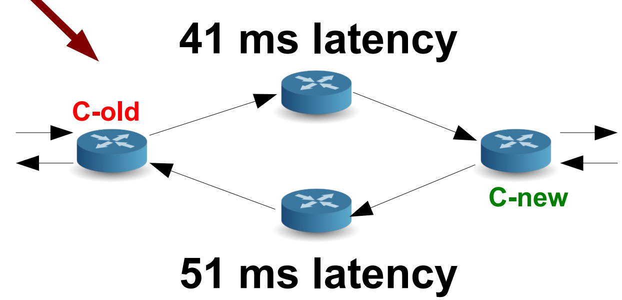


Temporarily shutdown router

- Abilene topology with 3 external peers
  - Configs translated to Quagga syntax
  - Abilene BGP dumps

Evaluation: Router Maintenance





Temporarily shutdown router

- Abilene topology with 3 external peers
  - Configs translated to Quagga syntax
  - Abilene BGP dumps

Conclusion and Future Work

Shadow configurations is new management primitive

- Realistic in-network evaluation
- Network-wide transactional support for configuration

Future work

- Evaluate on carrier-grade installations
- Automated proactive testing
- Automated reactive debugging

Thank you!