

# REAL: Emulating Control Plane at Simulator's Cost

Ze Xia<sup>1</sup>, Hao Li<sup>1</sup>, Jinyu Fu<sup>1</sup>, Xin Wan<sup>1</sup>, Yihan Dang<sup>1</sup>,  
Danfeng Shan<sup>1</sup>, Li Chen<sup>2</sup> and Peng Zhang<sup>1</sup>

1



西安交通大学  
XI'AN JIAOTONG UNIVERSITY

2



HARNETS.AI

# When Control Plane Goes Wrong

The image shows a screenshot of the Cloudflare Blog website. At the top left is the Cloudflare logo, consisting of an orange cloud icon and the word "CLOUDFLARE" in bold black letters. To its right is the text "The Cloudflare Blog". Below the logo is a horizontal navigation menu with the following items: "AI", "Developers", "Radar", "Product News", "Security", "Policy & Legal", "Zero Trust", "Speed & Reliability", "Life at Cloudflare", and "Partners". On the right side of the page, there is a subscription form with the text "Subscribe to receive notifications of new posts:", an input field labeled "Email Address", and an orange "Subscribe" button. Below the navigation menu, the main article title "Understanding how Facebook disappeared from the Internet" is displayed in a large, bold, black font. Underneath the title is the date "2021-10-04". At the bottom of the article header, there are two author profiles: Celso Martinho, shown with a circular profile picture of a man with short hair, and Tom Strickx, shown with a circular profile picture of a man wearing sunglasses.

# When Control Plane Goes Wrong



The image shows a tilted screenshot of a Cloudflare blog post. At the top left is the Cloudflare logo (an orange cloud with a white lightning bolt) and the text 'CLOUDFLARE'. To its right is 'The Cloudflare Blog'. Below these are navigation links: 'AI', 'Developers', 'Radar', and 'Product News'. The main title of the post is 'YouTube Hijacking: A RIPE NCC RIS case study' in large, bold, dark blue font. At the bottom left, the date '17 Mar 2008' is followed by a dot and three tags: 'news', 'RIS', and 'internet governance'. On the right side, the word 'internet' is partially visible. A search icon is located in the top right corner of the page.

## YouTube Hijacking: A RIPE NCC RIS case study

17 Mar 2008

news

RIS

internet governance

internet

# When Control Plane Goes Wrong

CLOUDFLARE

case

OUTAGE ANALYSES

## Twitter Outage Analysis: March 28, 2022

By [Chris Villemez](#) | April 15, 2022 | 14 min read

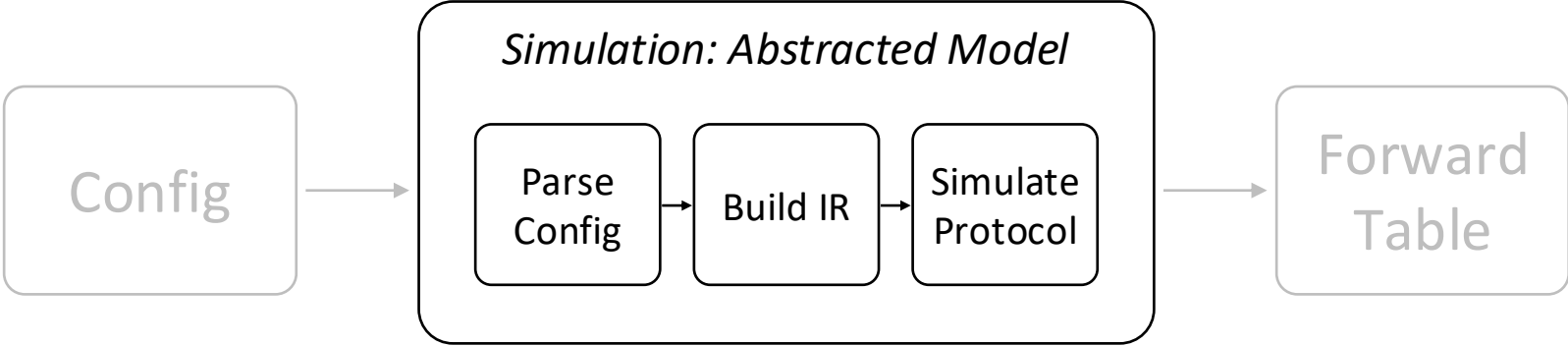


17 Mar 2008 • news

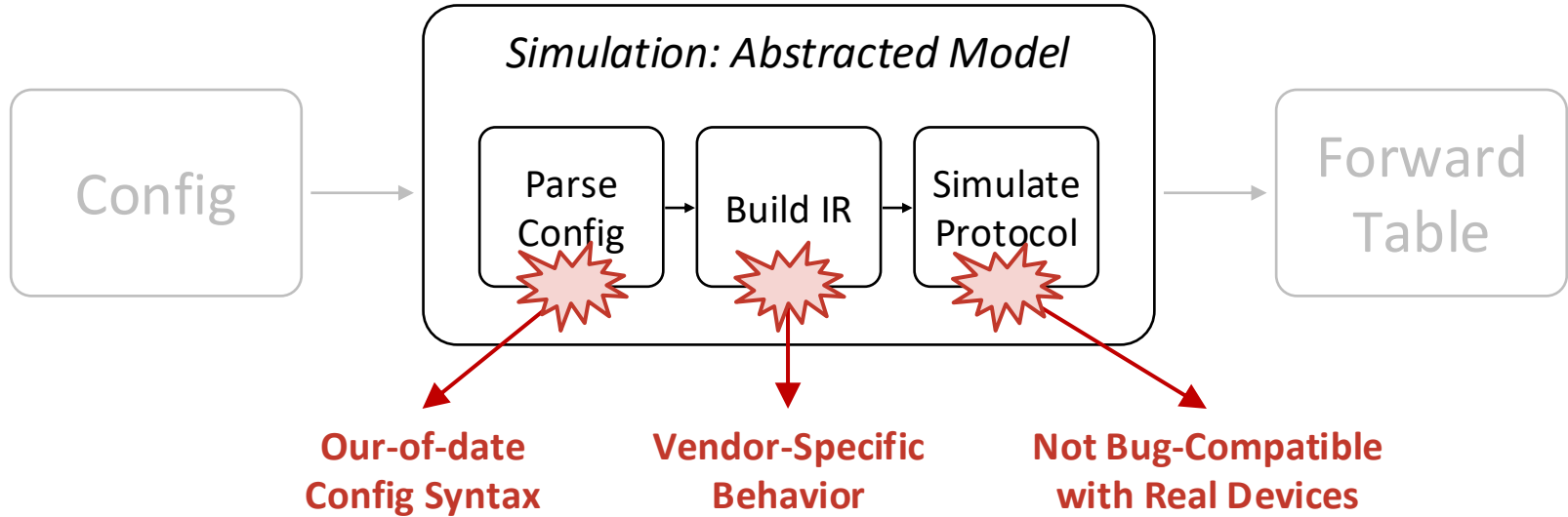
## Control Plane: Configuration to Forward Table



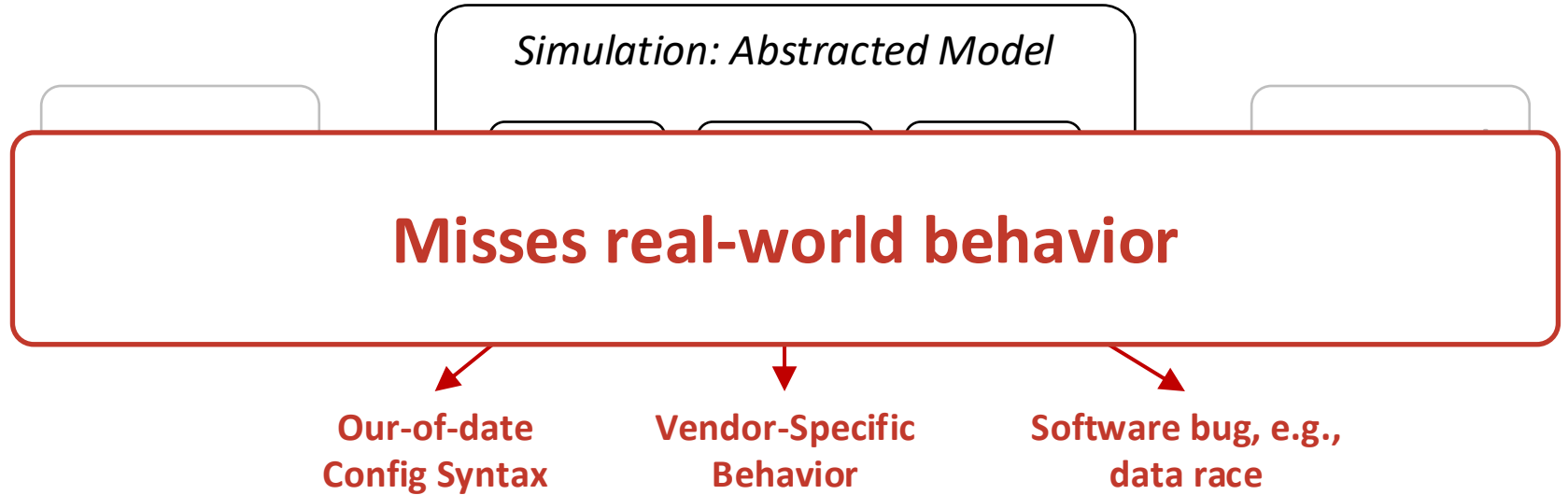
# Modeling Control Plane Behavior: Simulation



# Simulation Misses Real-World Behavior



# Simulation Misses Real-World Behavior



# Emulation: Real Software



**Actual Binaries — No Abstraction Gap**

# Reality Is Expensive

---

	Topology Scale	CPU count	Memory	Time
CrystalNet [SOSP'17]	~4000	2000-4000	4TB-8TB	~30 min
Crescent [NSDI'24]	1000+	512	2TB	~50 min

---

# Reality Is Expensive

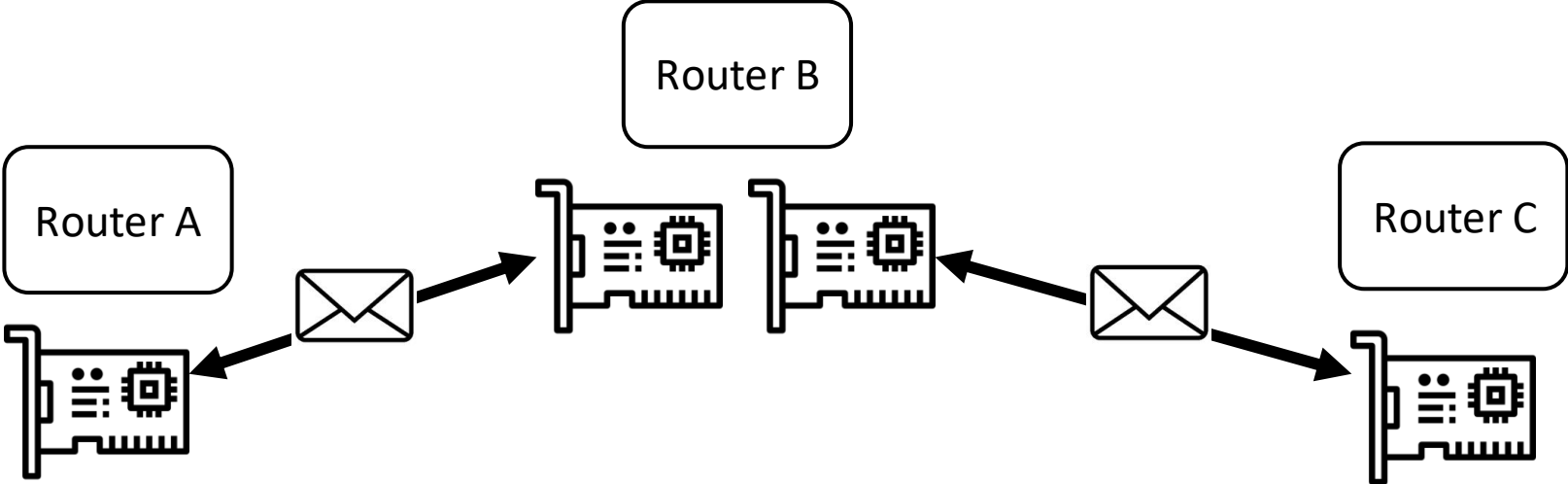
---

	Topology Scale	CPU count	Memory	Time
CrystalNet [SOSP'17]	~4000	2000-4000	4TB-8TB	~30 min
Crescent [NSDI'24]	1000+	512	2TB	~50 min
<b>Batfish</b> <b>[SIGCOMM'23]</b>	<b>~3000</b>	<b>10</b>	<b>64GB</b>	<b>&lt;5 min</b>

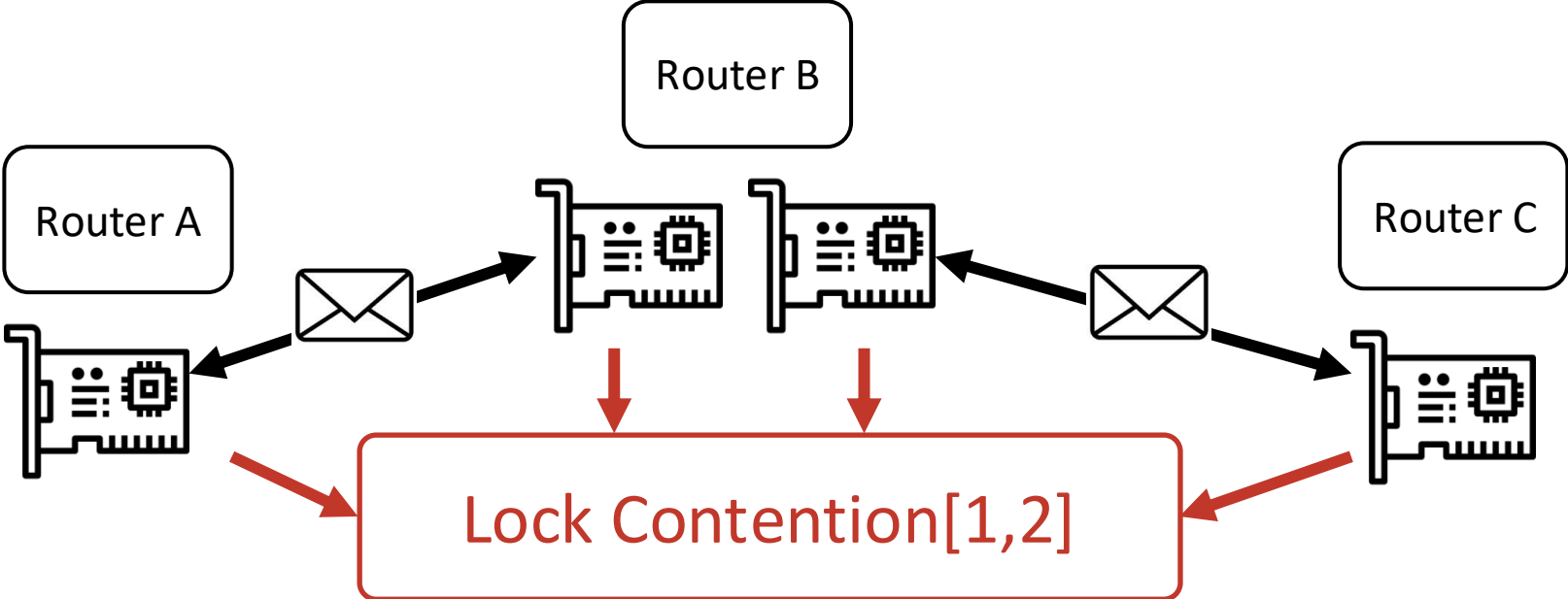
---

Where does the cost come from?

# Cost 1: Network Buildup Time

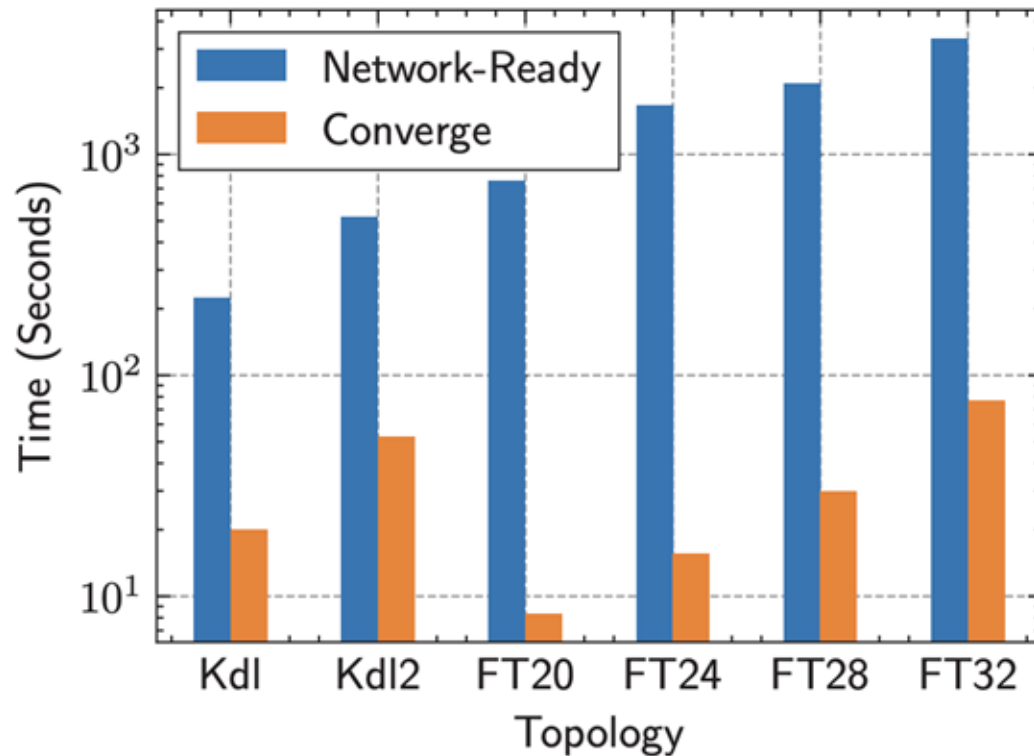


# Cost 1: Network Buildup Time

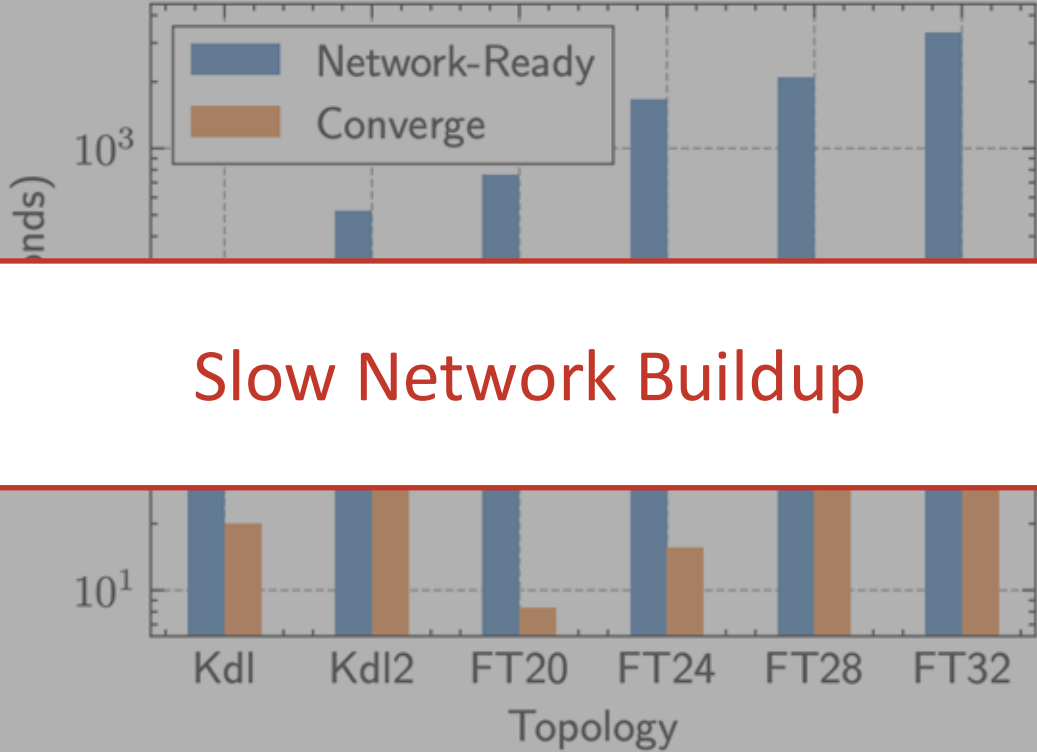


[1] Liu et al. "Understanding Network Startup for Secure Containers in Multi-Tenant Clouds: Performance, Bottleneck and Optimization." IMC '24  
[2] Peng et al. "SplitNN: Single-Machine Network Emulation at Scale with Minute-Level Construction of 10K-Node Virtual Networks." APNET '25

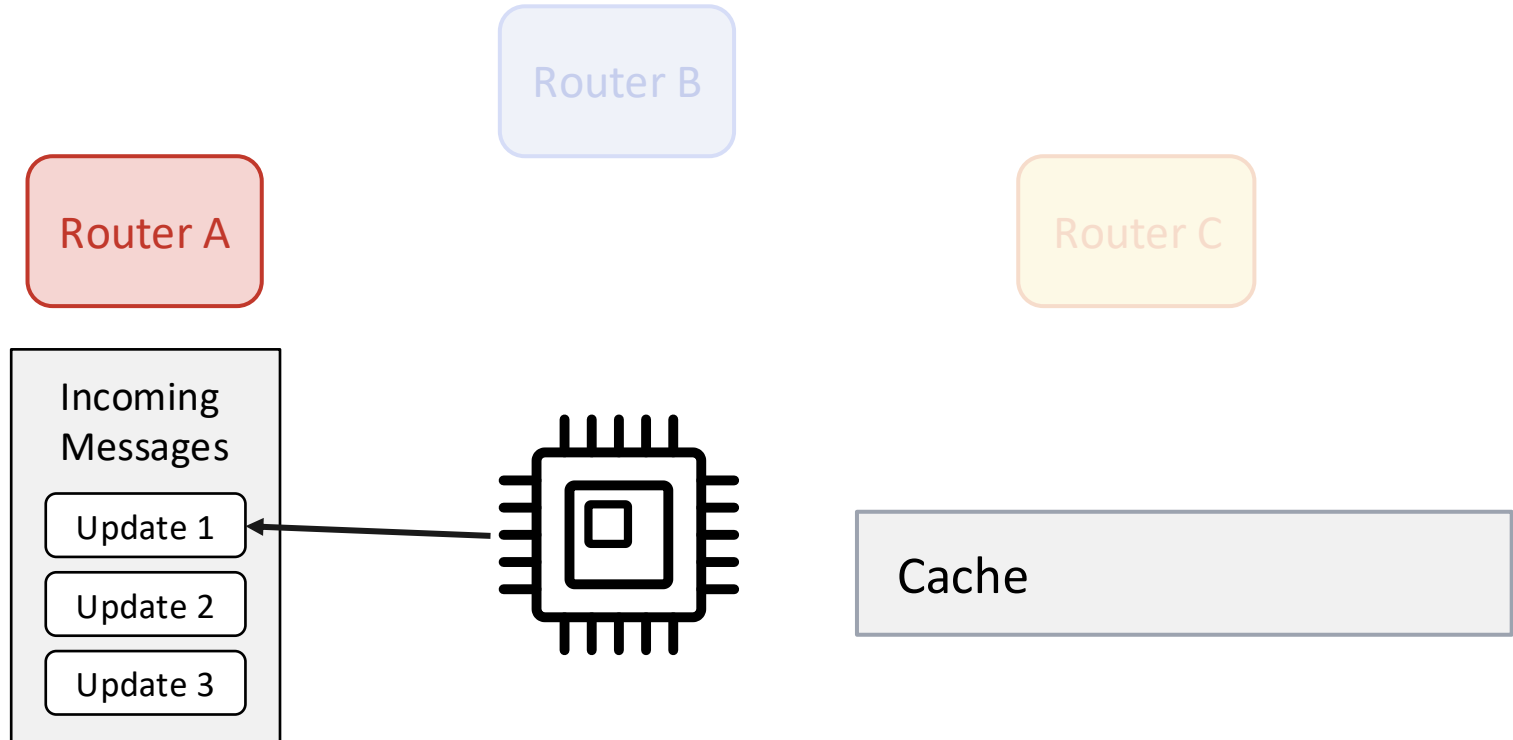
## Cost 1: Network Buildup Time



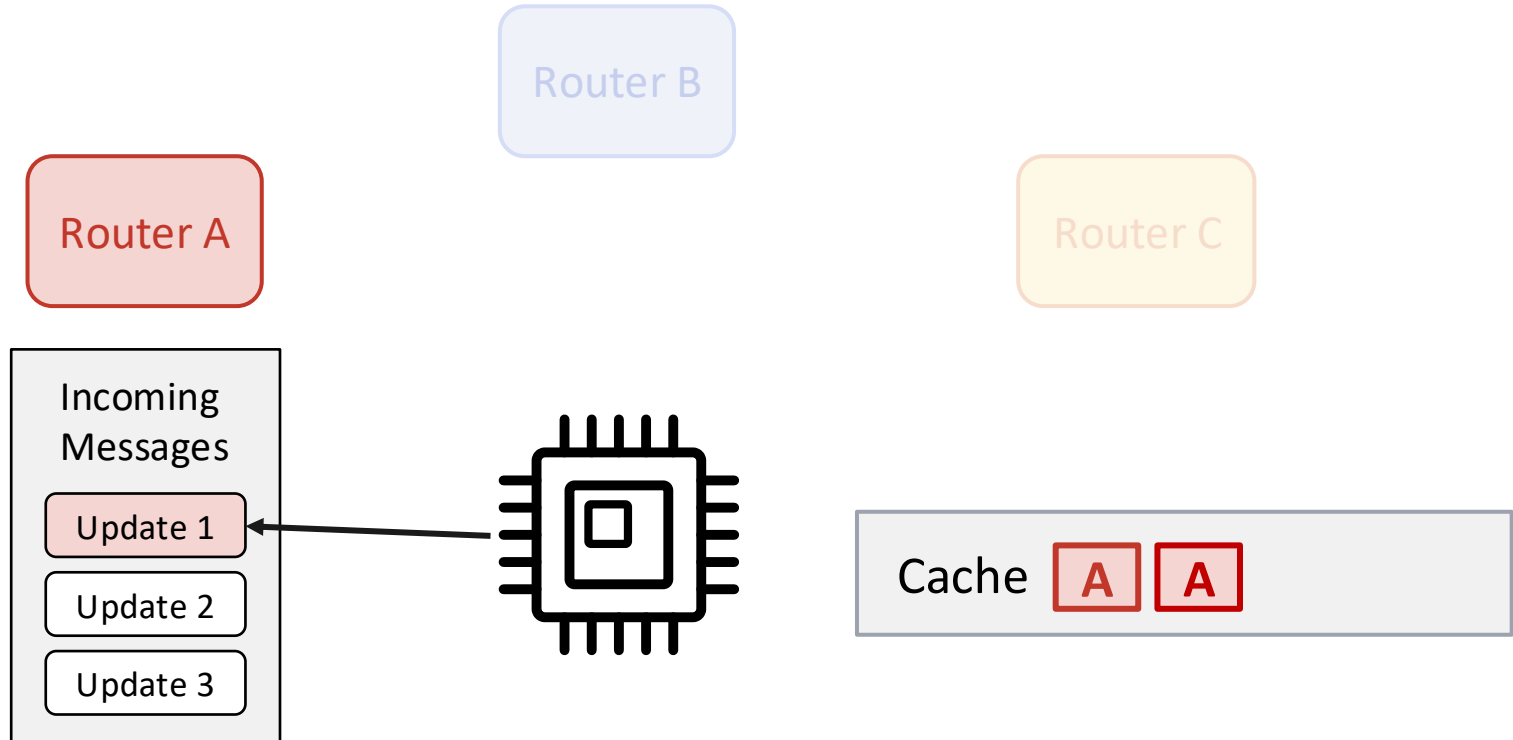
# Cost 1: Network Buildup Time



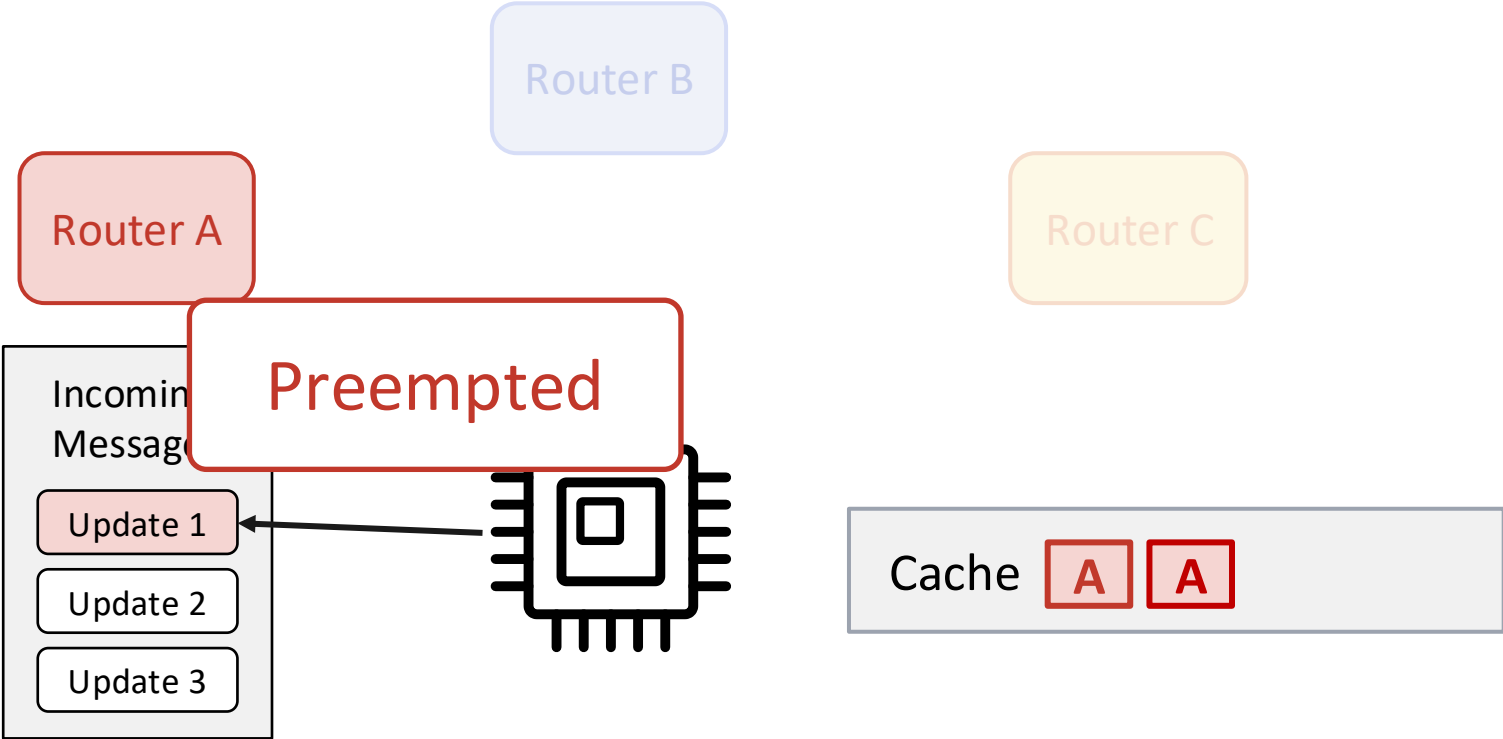
## Cost 2: Context Switch



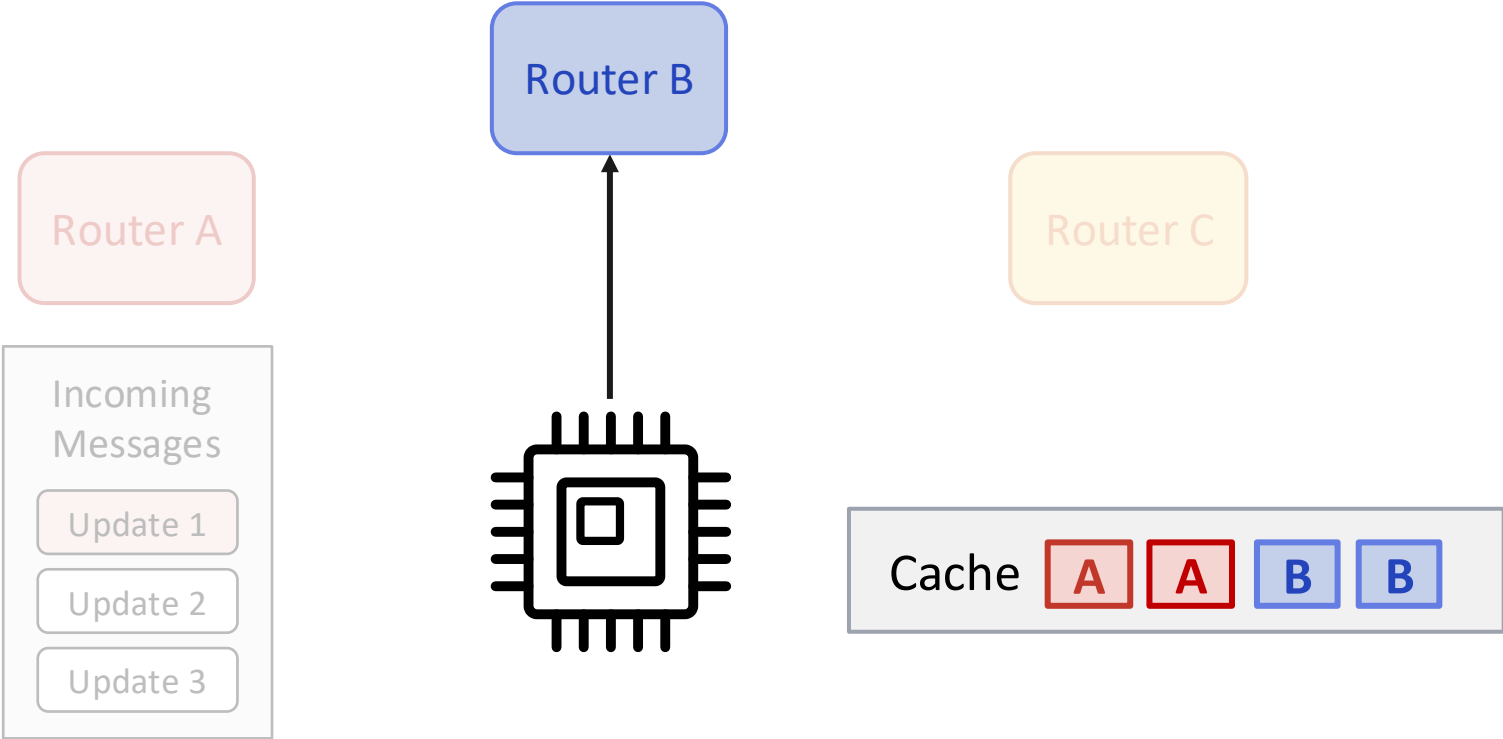
## Cost 2: Context Switch



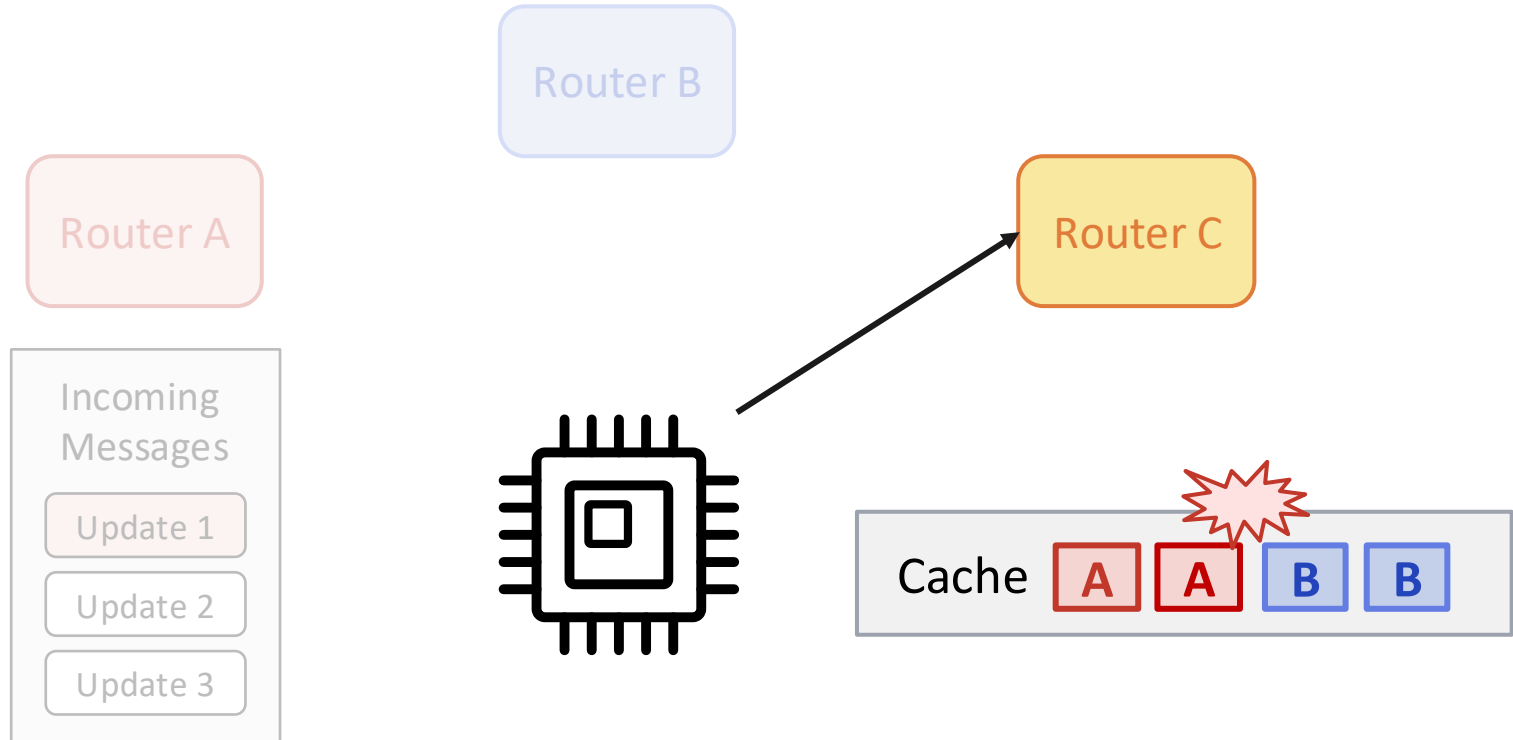
# Cost 2: Context Switch



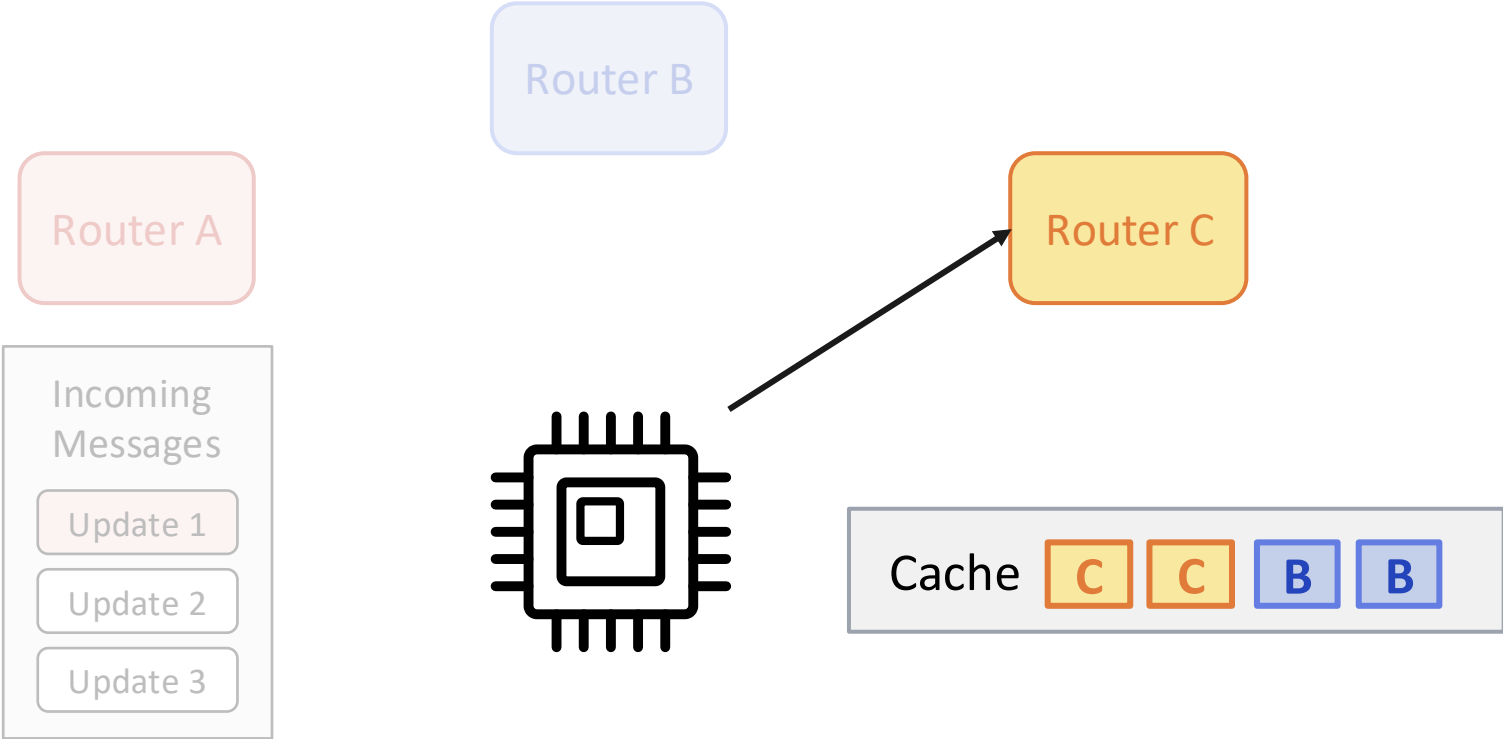
# Cost 2: Context Switch



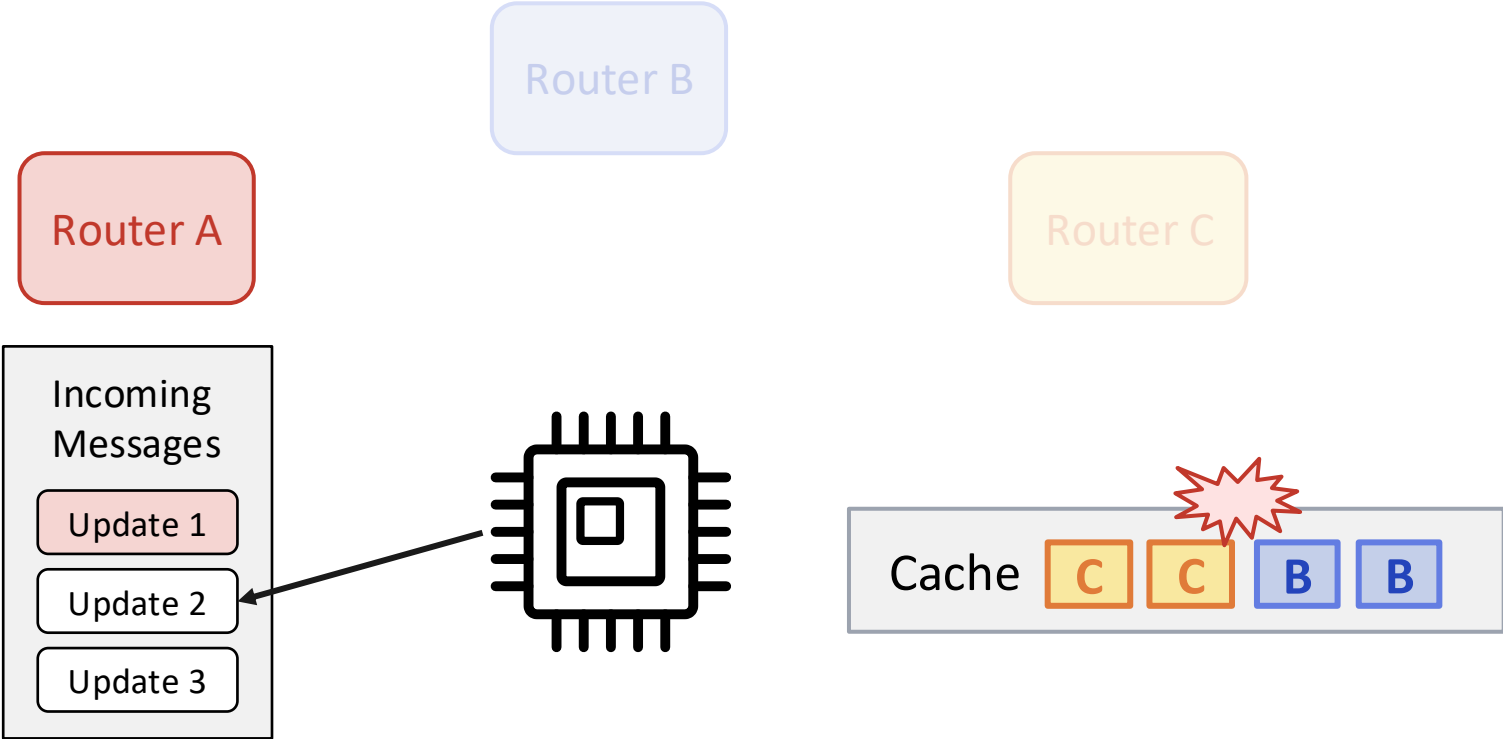
## Cost 2: Context Switch



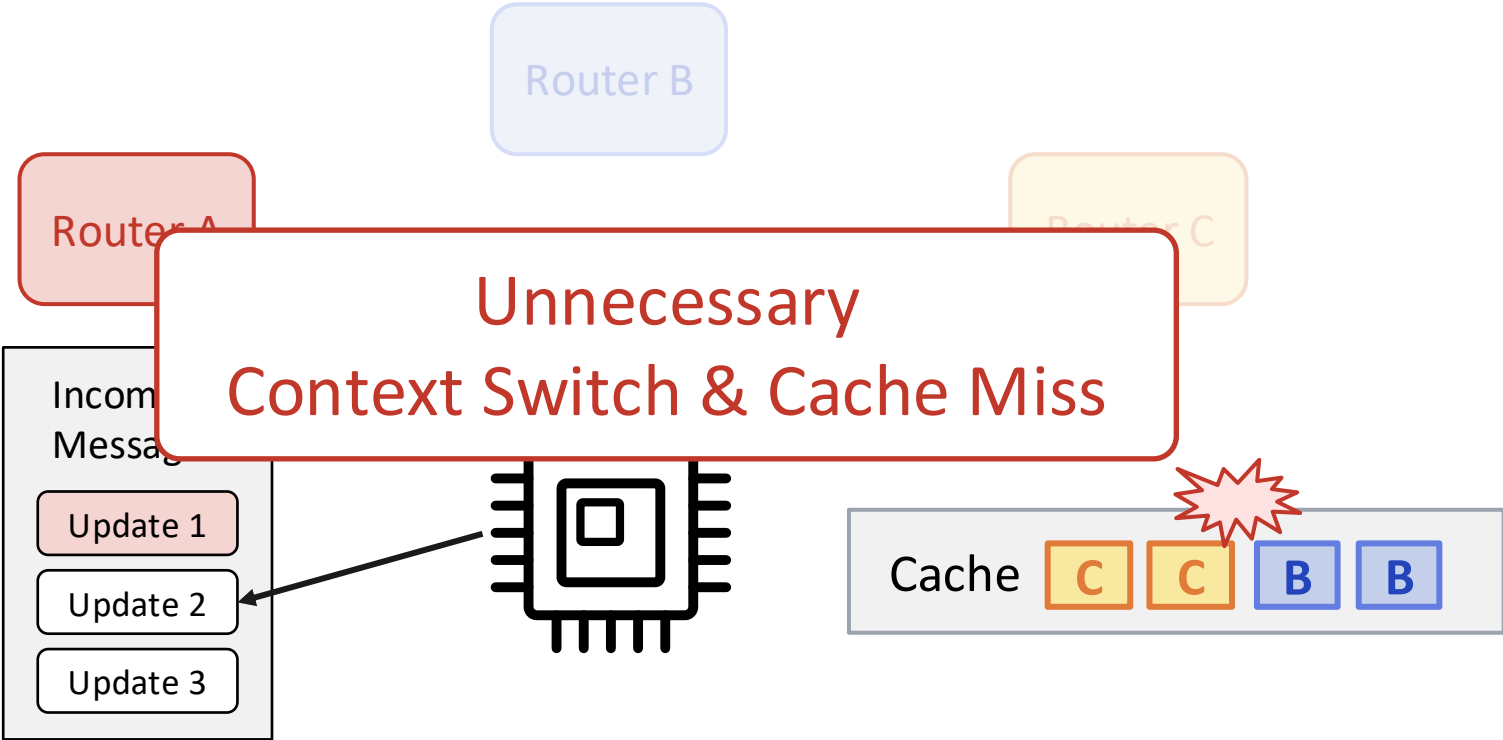
# Cost 2: Context Switch



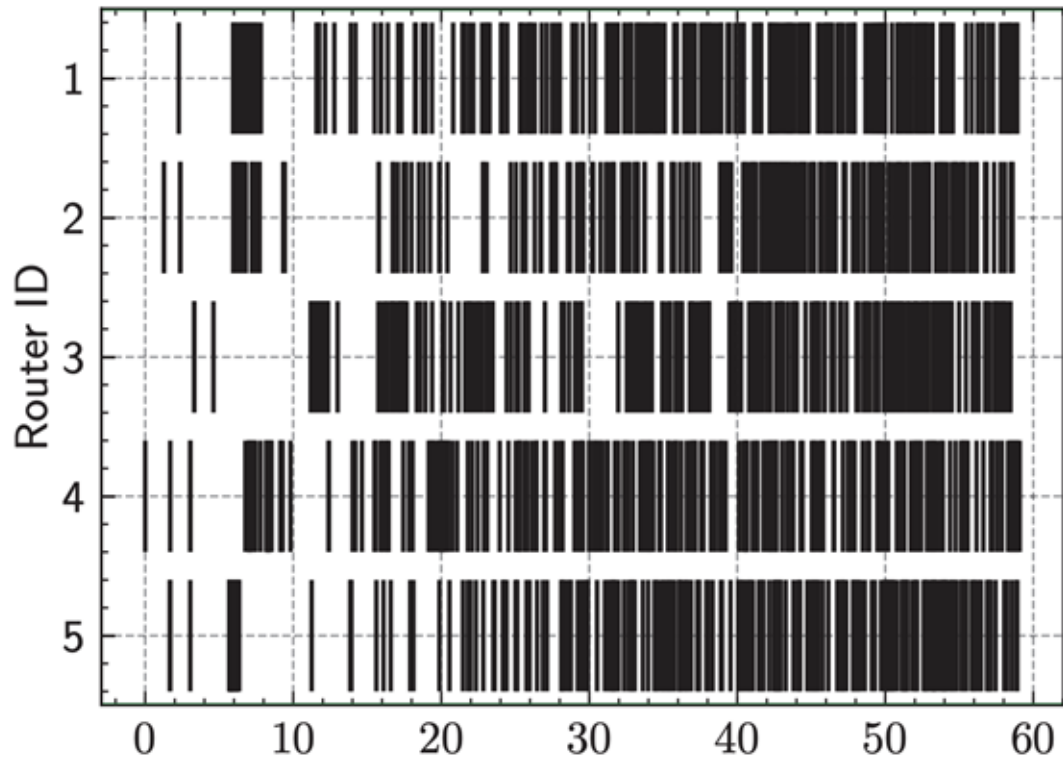
# Cost 2: Context Switch



# Cost 2: Context Switch



## Cost 2: Context Switch

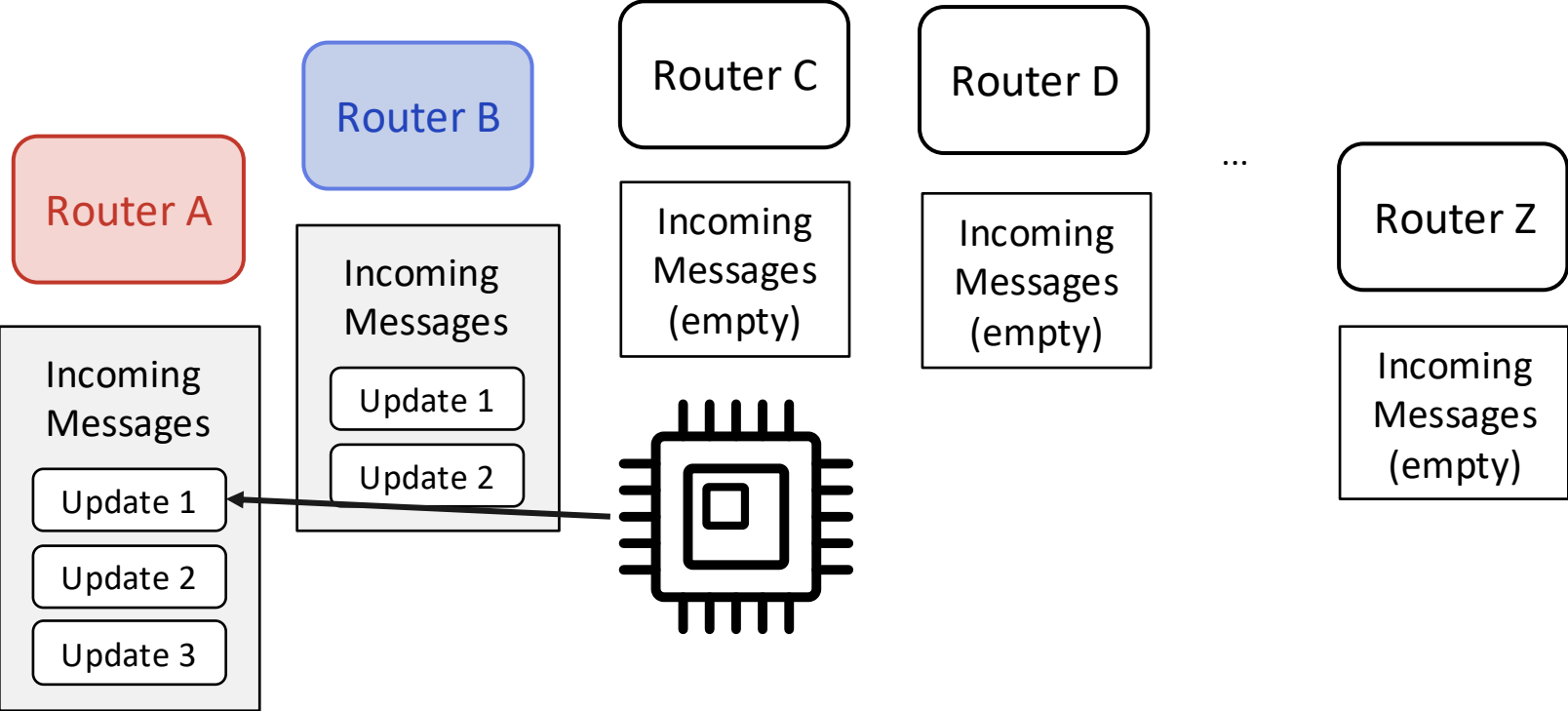


# Cost 2: Context Switch

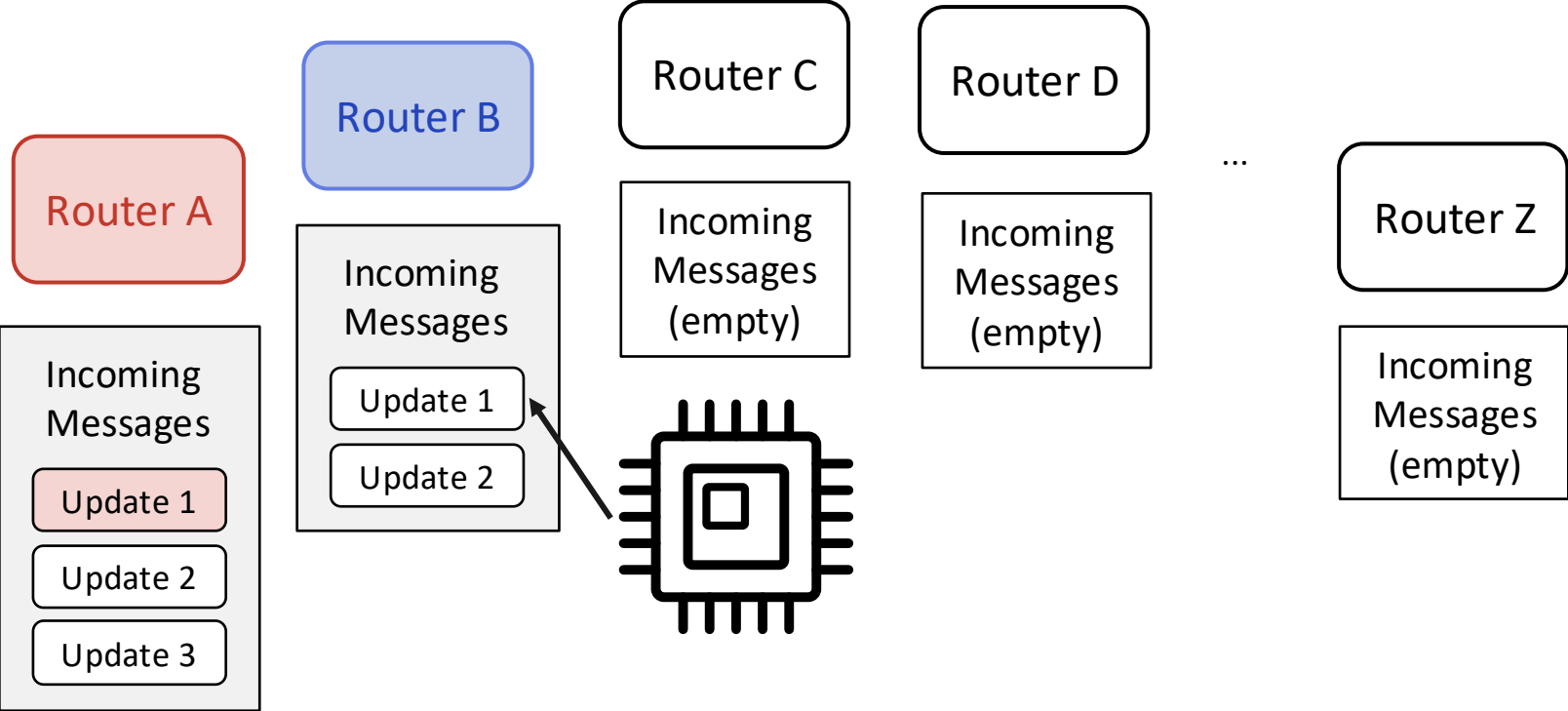


Frequent Context Switch & Cache Miss

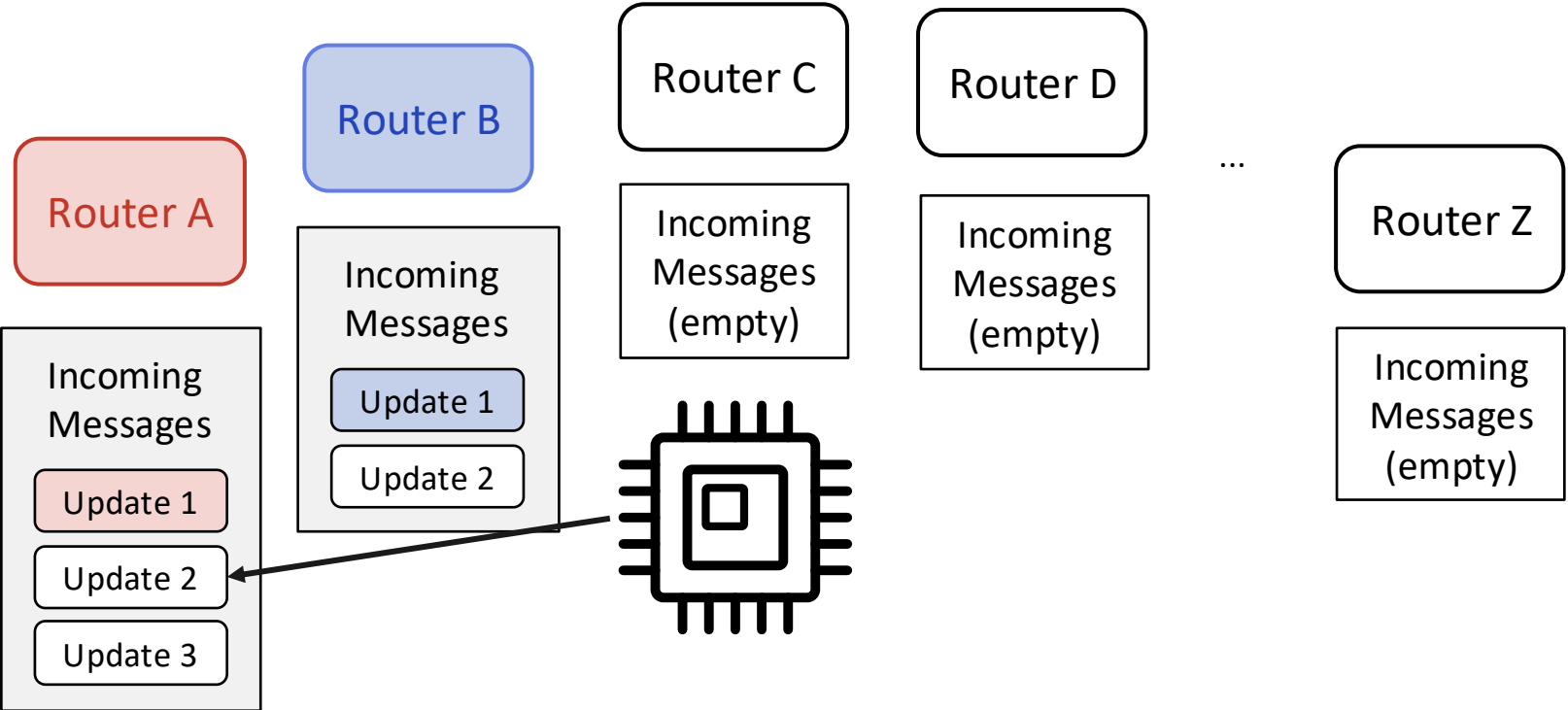
# Cost 3: Memory Residency



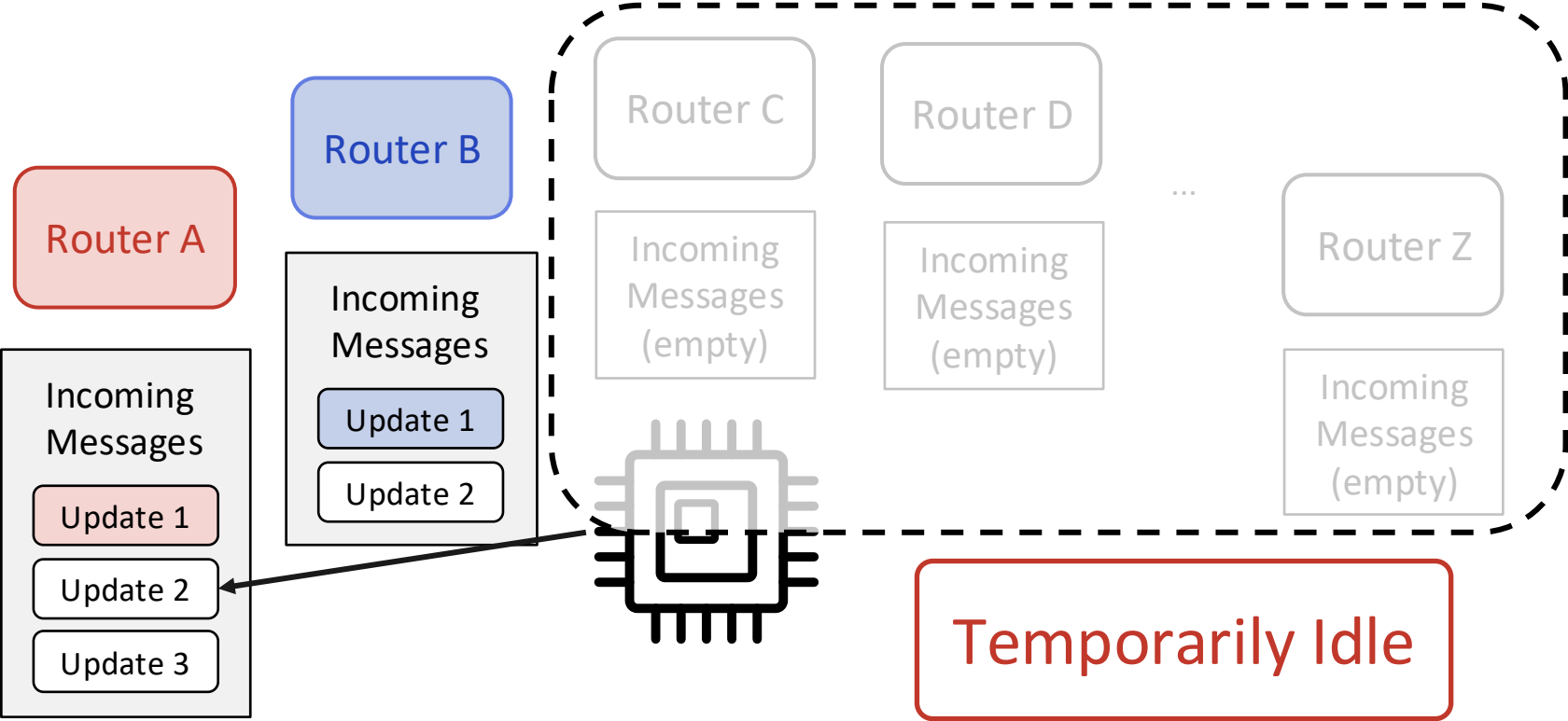
# Cost 3: Memory Residency



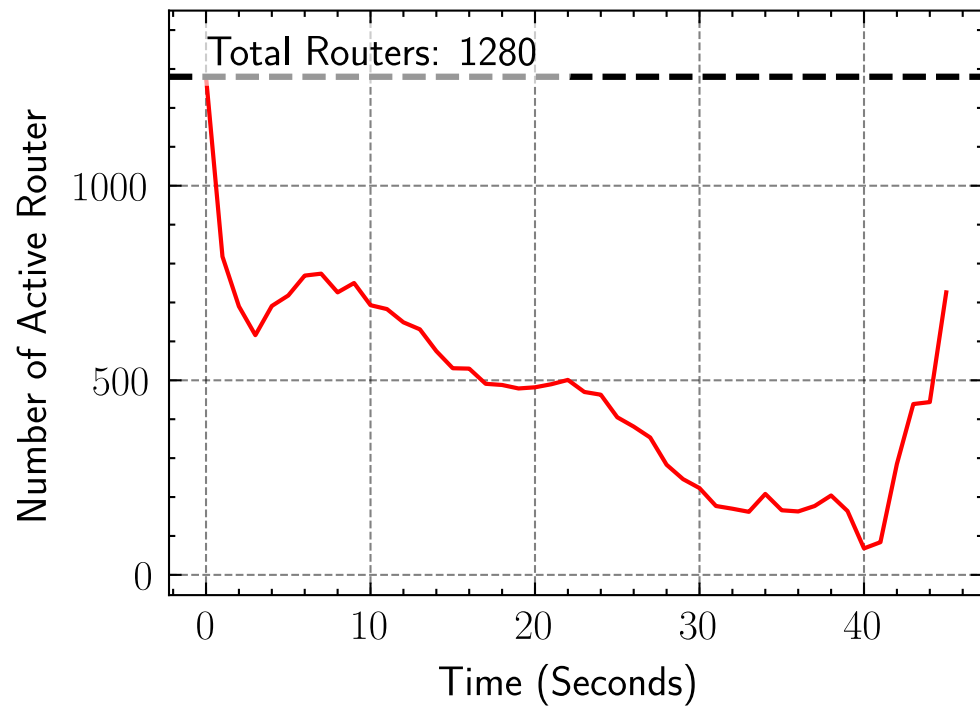
# Cost 3: Memory Residency



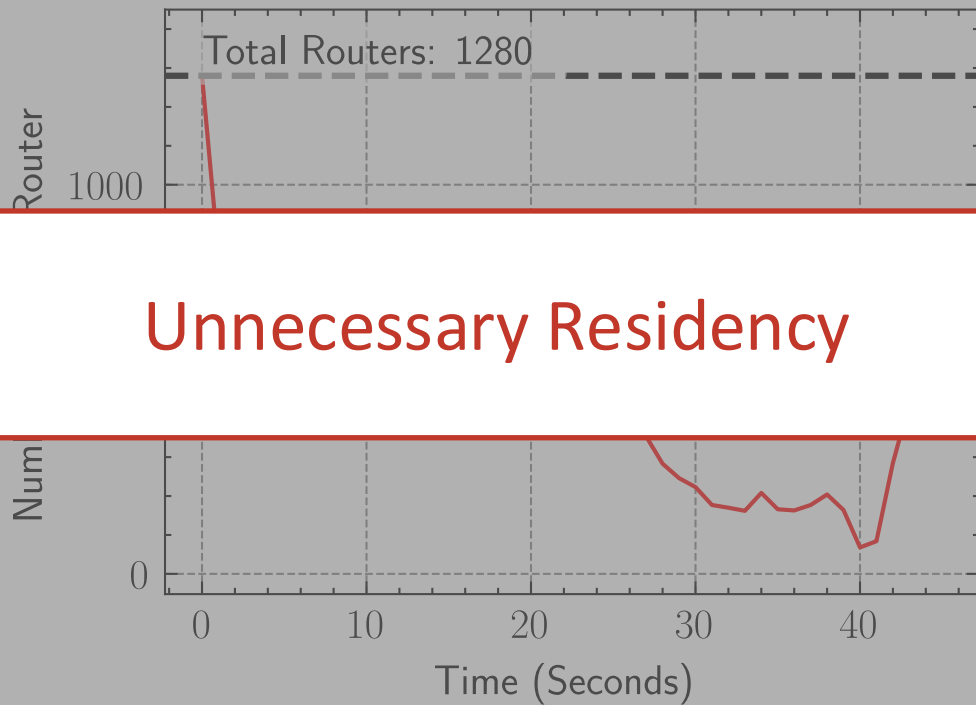
# Cost 3: Memory Residency



## Cost 3: Memory Residency



## Cost 3: Memory Residency



Unnecessary Residency

Root Cause: Runtime does not Match Control Plane Needs

# Root Cause: Runtime does not Match Control Plane Needs

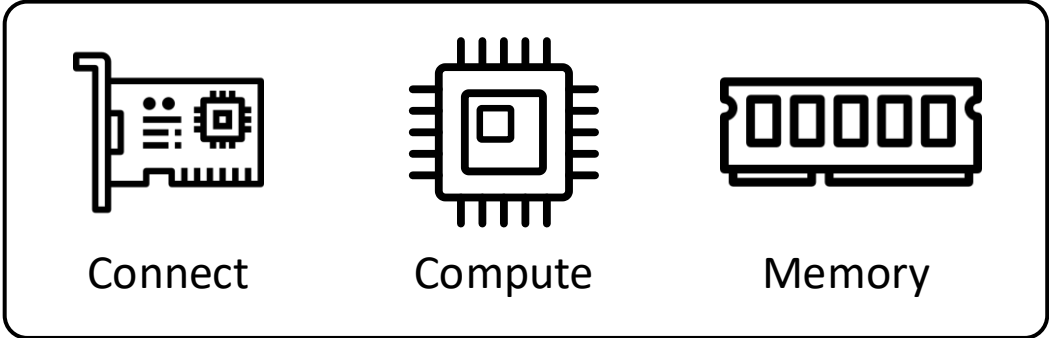
Control Plane



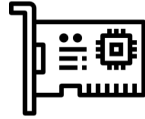
Mismatch



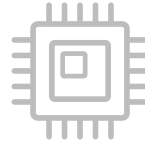
Runtime



# Root Cause: Runtime does not Match Control Plane Needs



Connect



Compute



Memory

---

Runtime Mechanism

Built for

Control Plane Needs

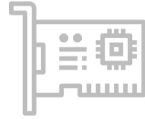
---

Virtual Ethernet

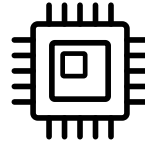
Real L2 packets

**Message delivery**

# Root Cause: Runtime does not Match Control Plane Needs



Connect



Compute



Memory

---

Runtime Mechanism

Built for

Control Plane Needs

Virtual Ethernet

Real L2 packets

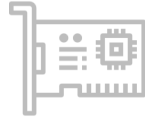
Message delivery

Fair Scheduling

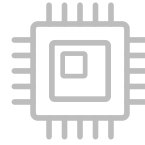
Timely response

**Continuous Execution**

# Root Cause: Runtime does not Match Control Plane Needs



Connect



Compute



Memory

---

Runtime Mechanism

Built for

Control Plane Needs

Virtual Ethernet

Real L2 packets

**Message delivery**

Fair Scheduling

Timely response

**Continuous Execution**

Memory residency

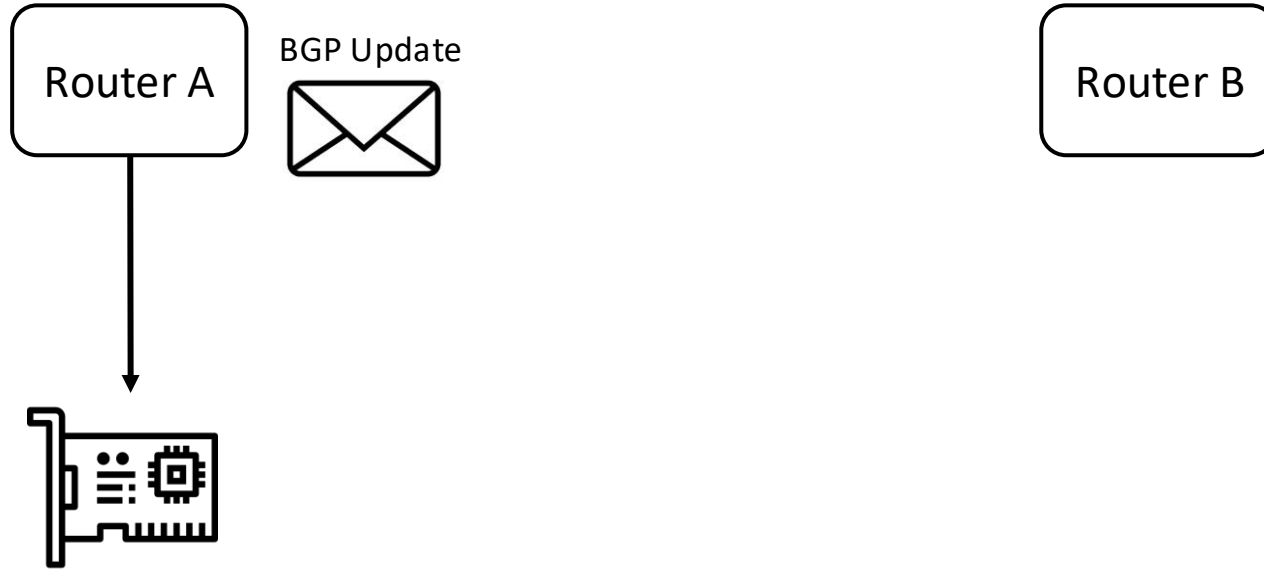
Always-on nodes

**On-demand access**

---

What should the runtime look like?

# Message-Level Delivery: Eliminate Network Buildup



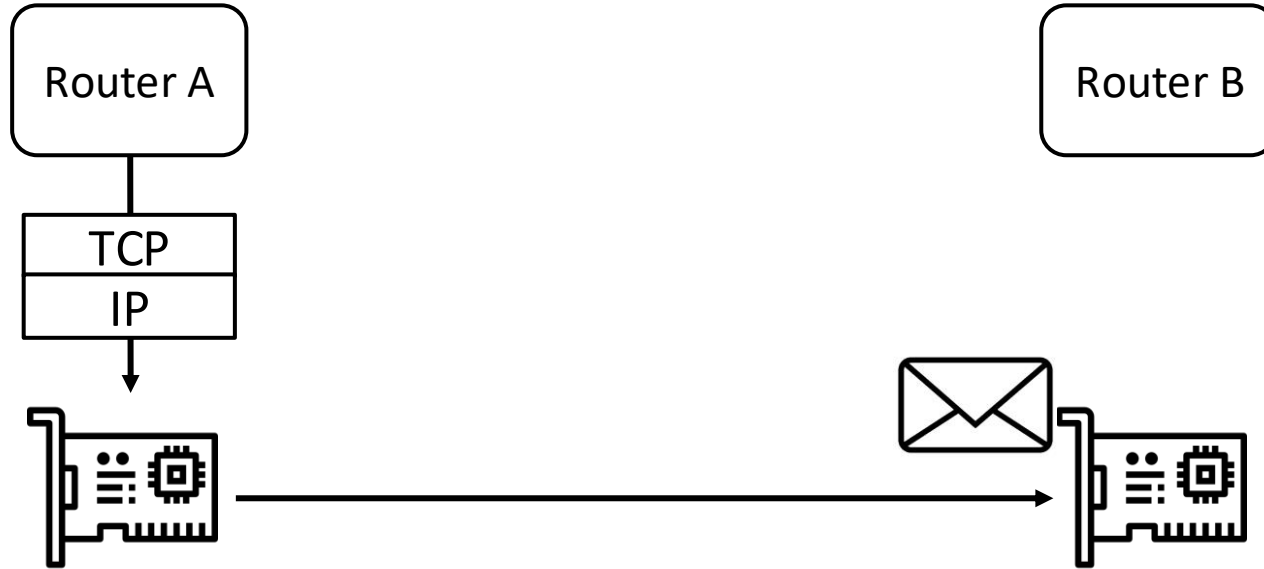
Normal Workflow

# Message-Level Delivery: Eliminate Network Buildup



Normal Workflow

# Message-Level Delivery: Eliminate Network Buildup



Normal Workflow

# Message-Level Delivery: Eliminate Network Buildup

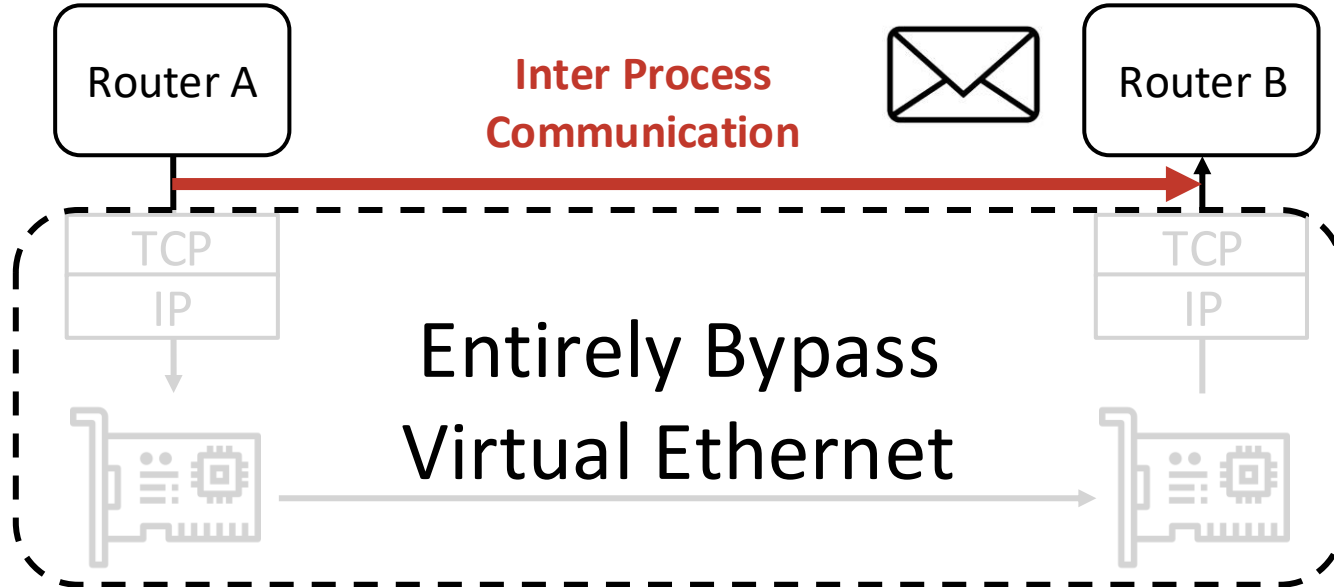


Normal Workflow

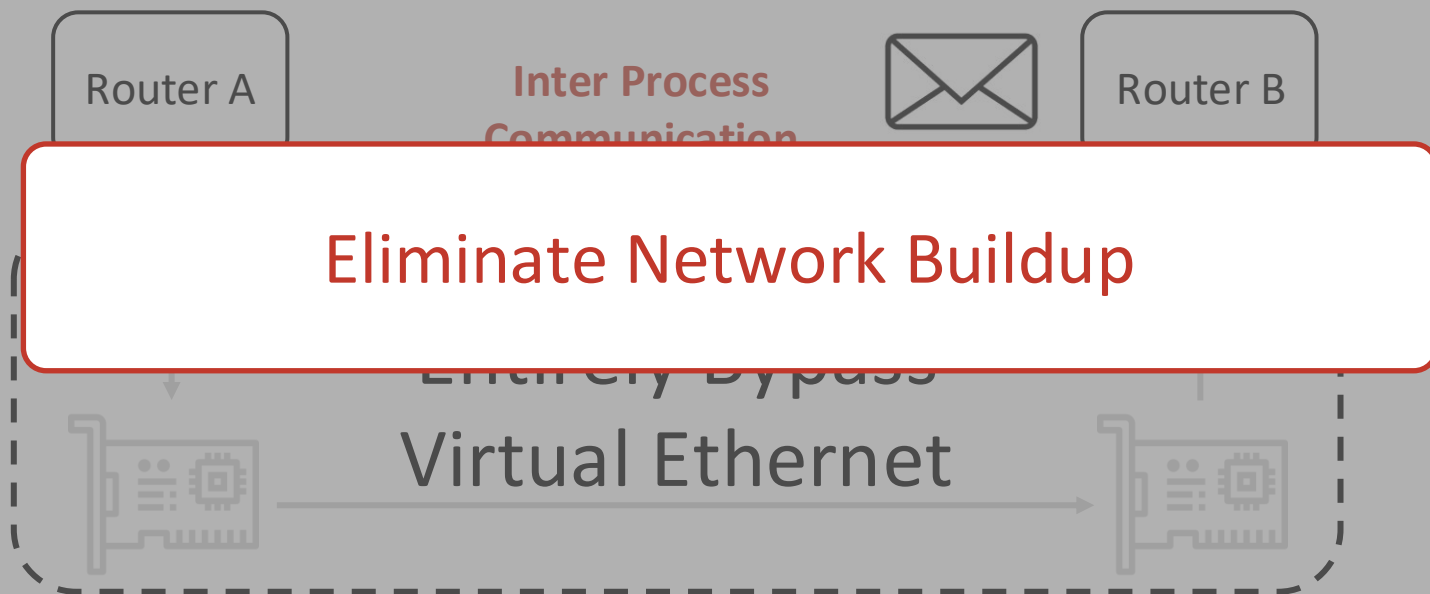
# Message-Level Delivery: Eliminate Network Buildup



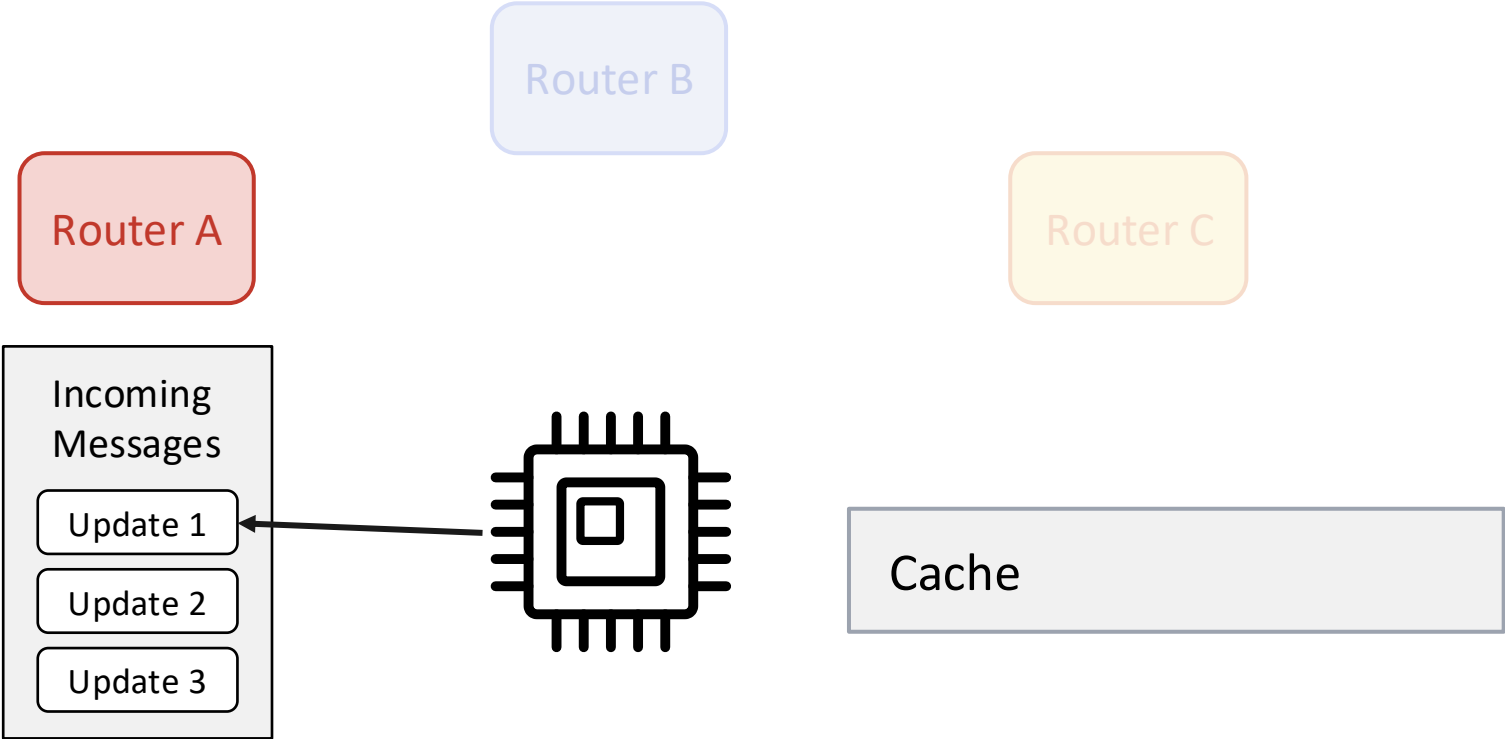
# Message-Level Delivery: Eliminate Network Buildup



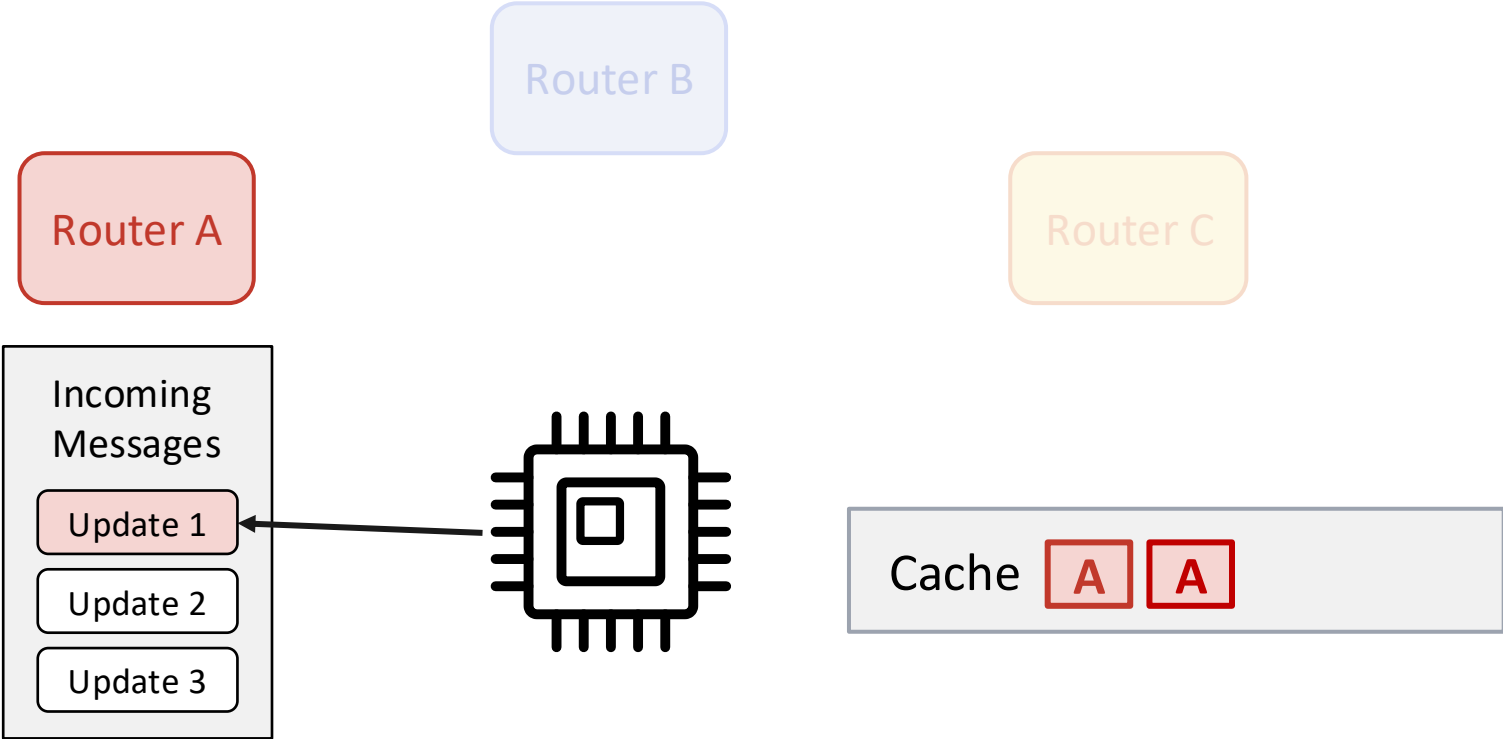
# Message-Level Delivery: Eliminate Network Buildup



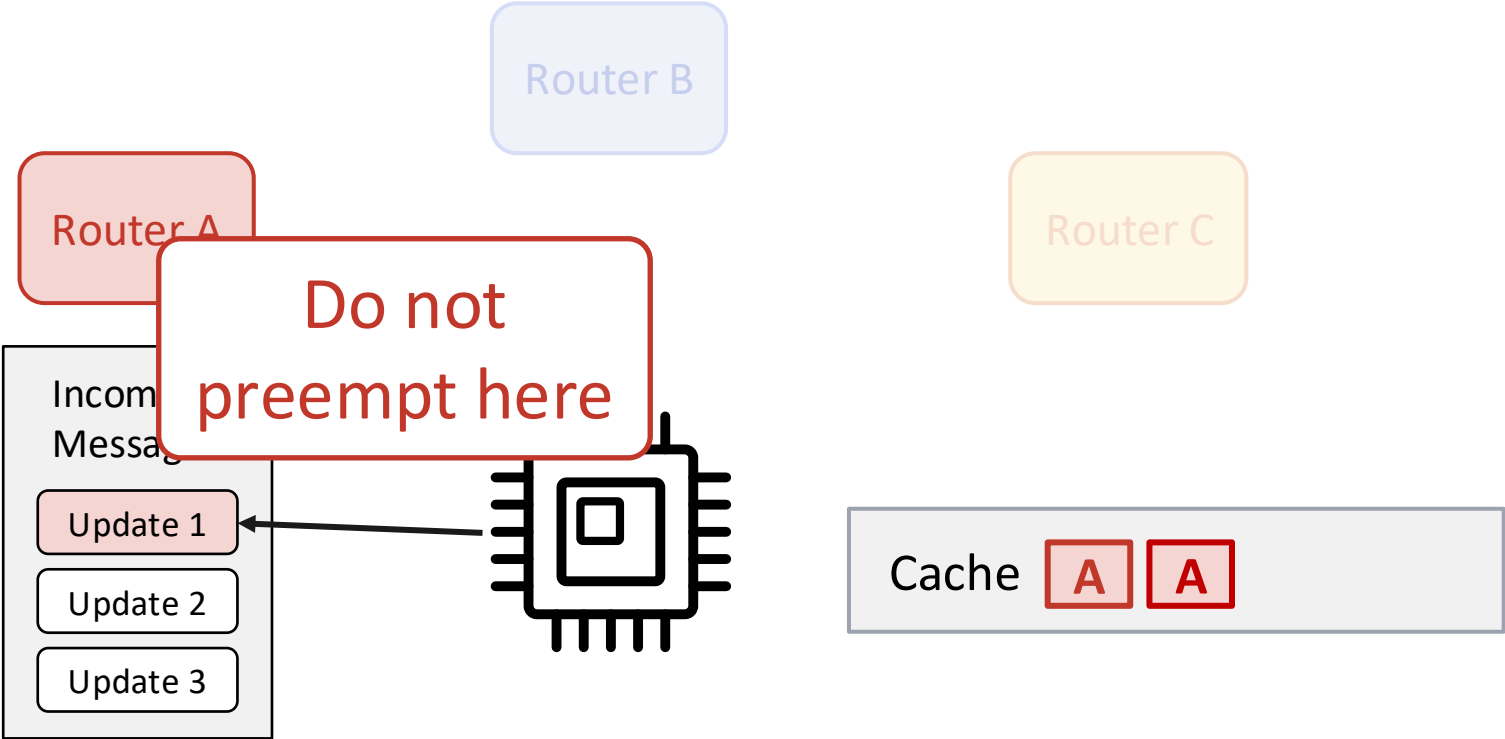
# Run-to-Idle Scheduling: Avoid Unnecessary Cache Miss



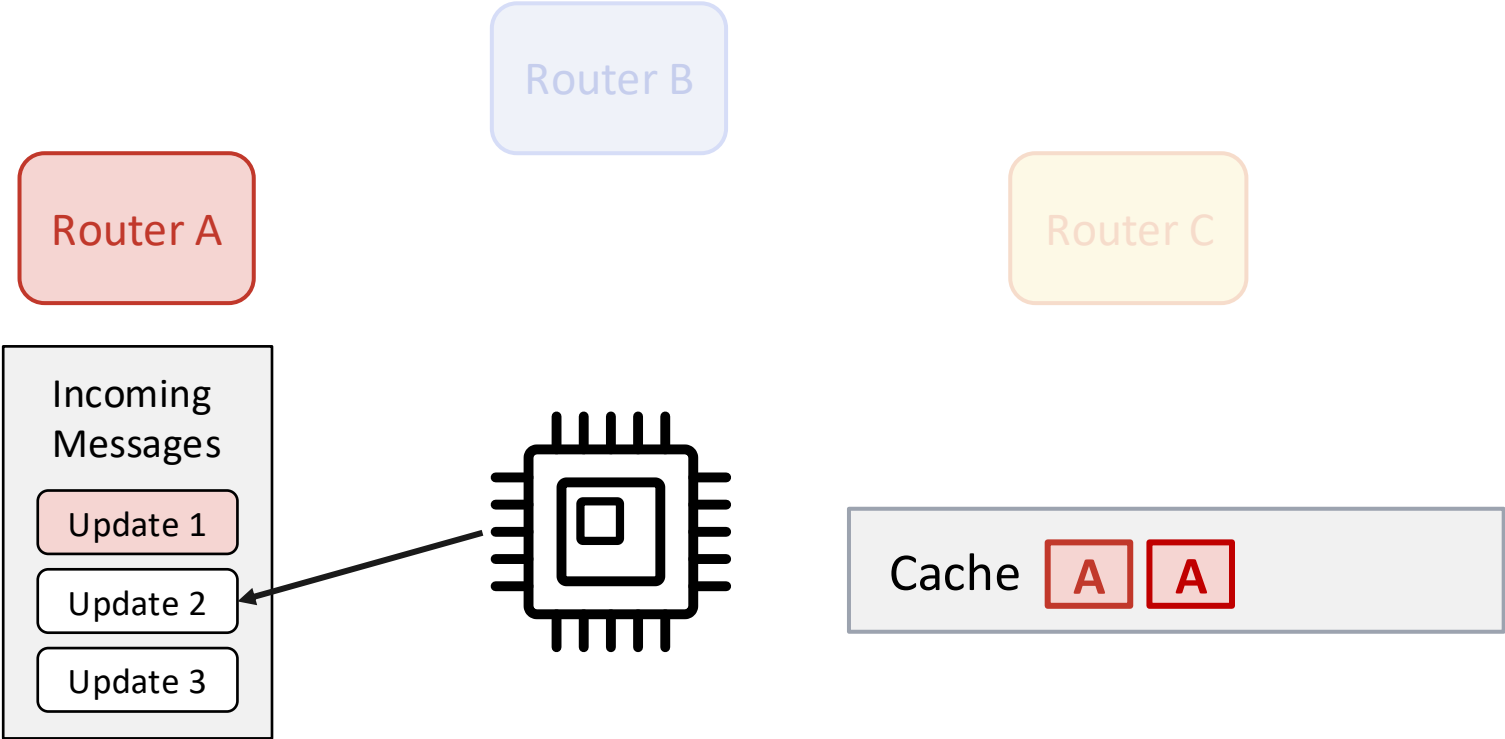
# Run-to-Idle Scheduling: Avoid Unnecessary Cache Miss



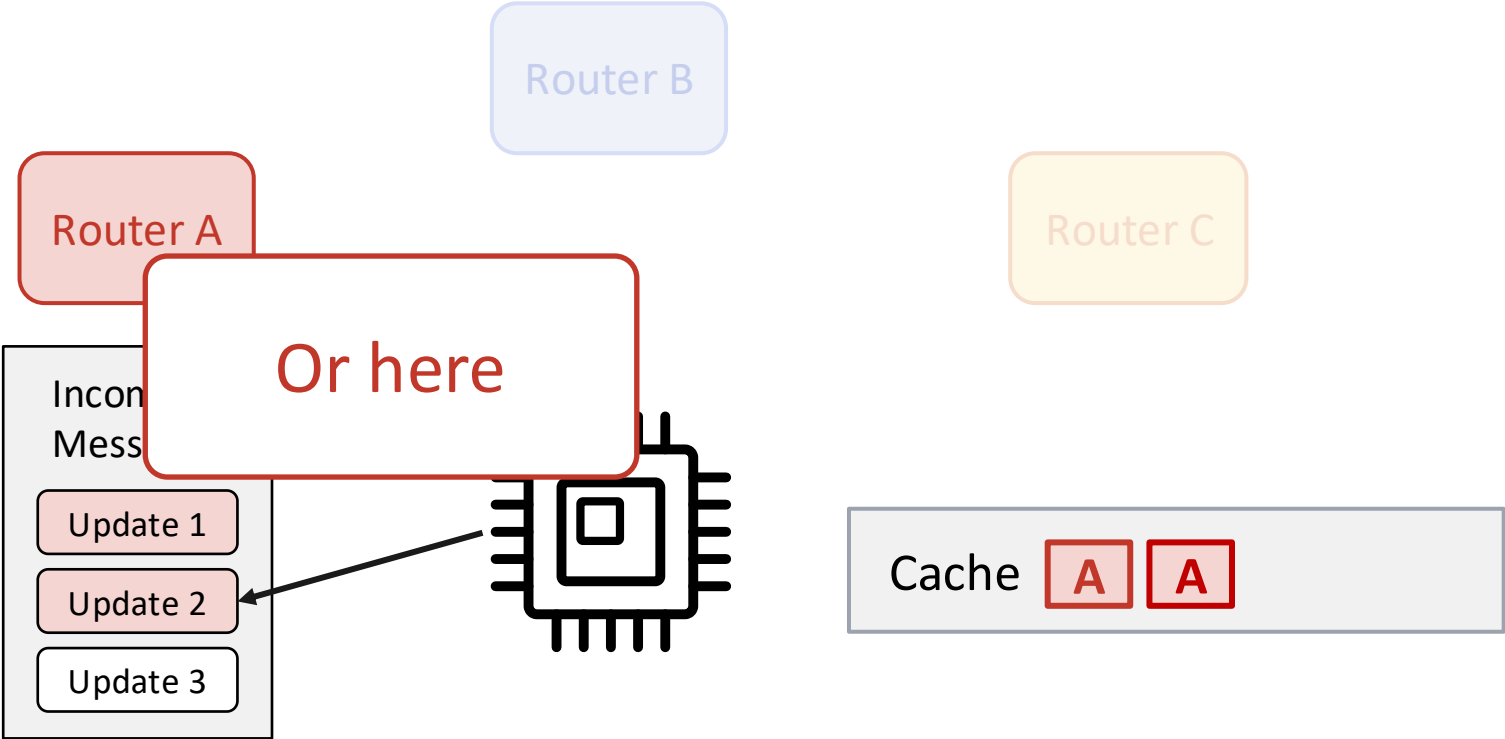
# Run-to-Idle Scheduling: Avoid Unnecessary Cache Miss



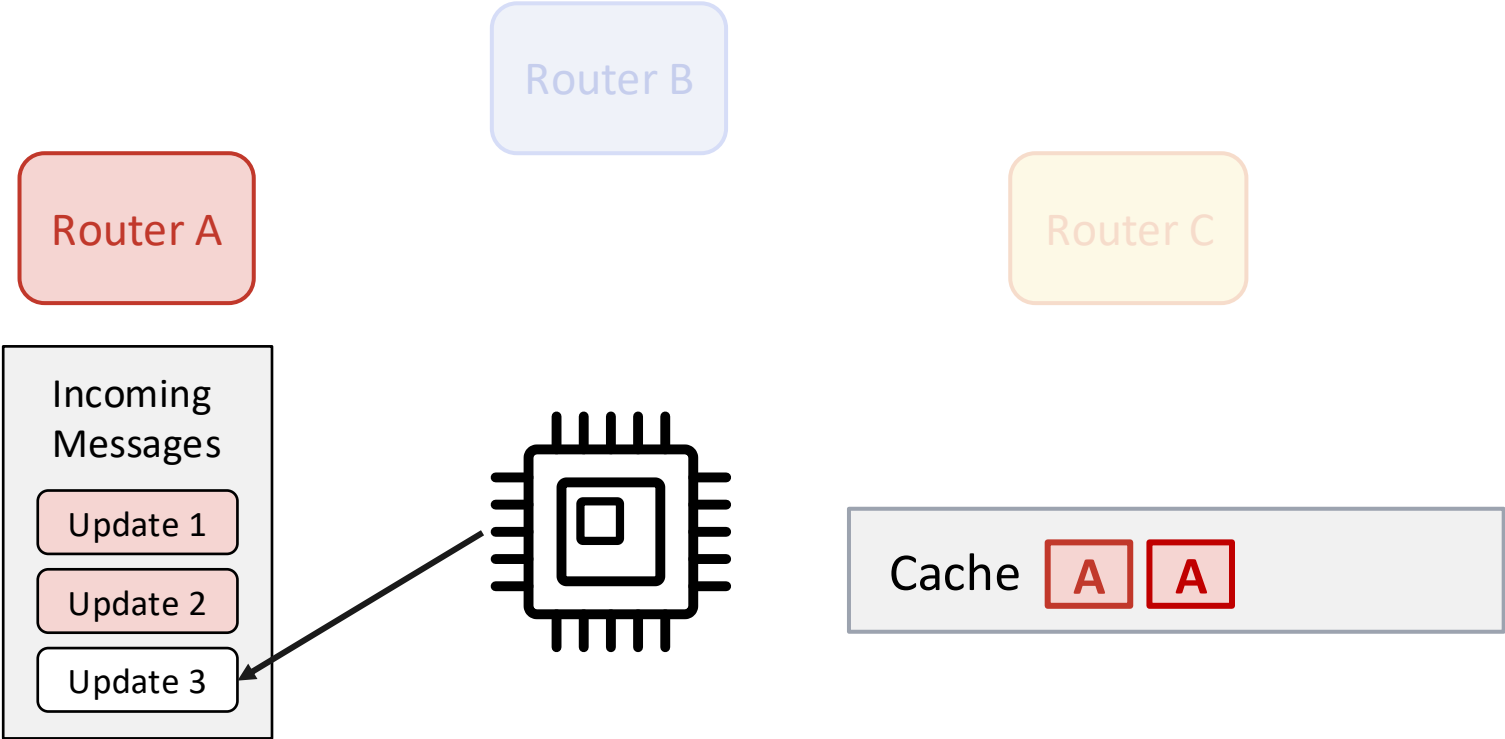
# Run-to-Idle Scheduling: Avoid Unnecessary Cache Miss



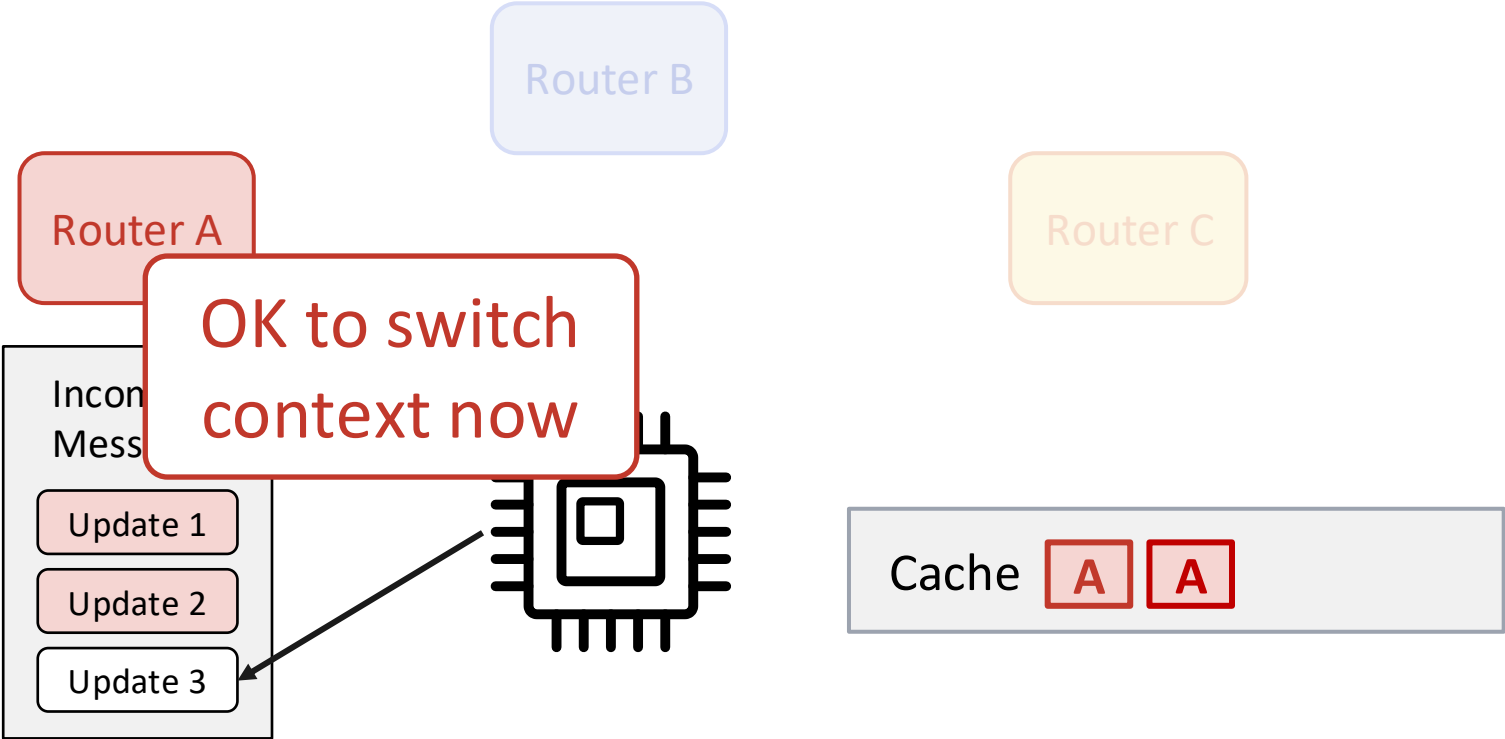
# Run-to-Idle Scheduling: Avoid Unnecessary Cache Miss



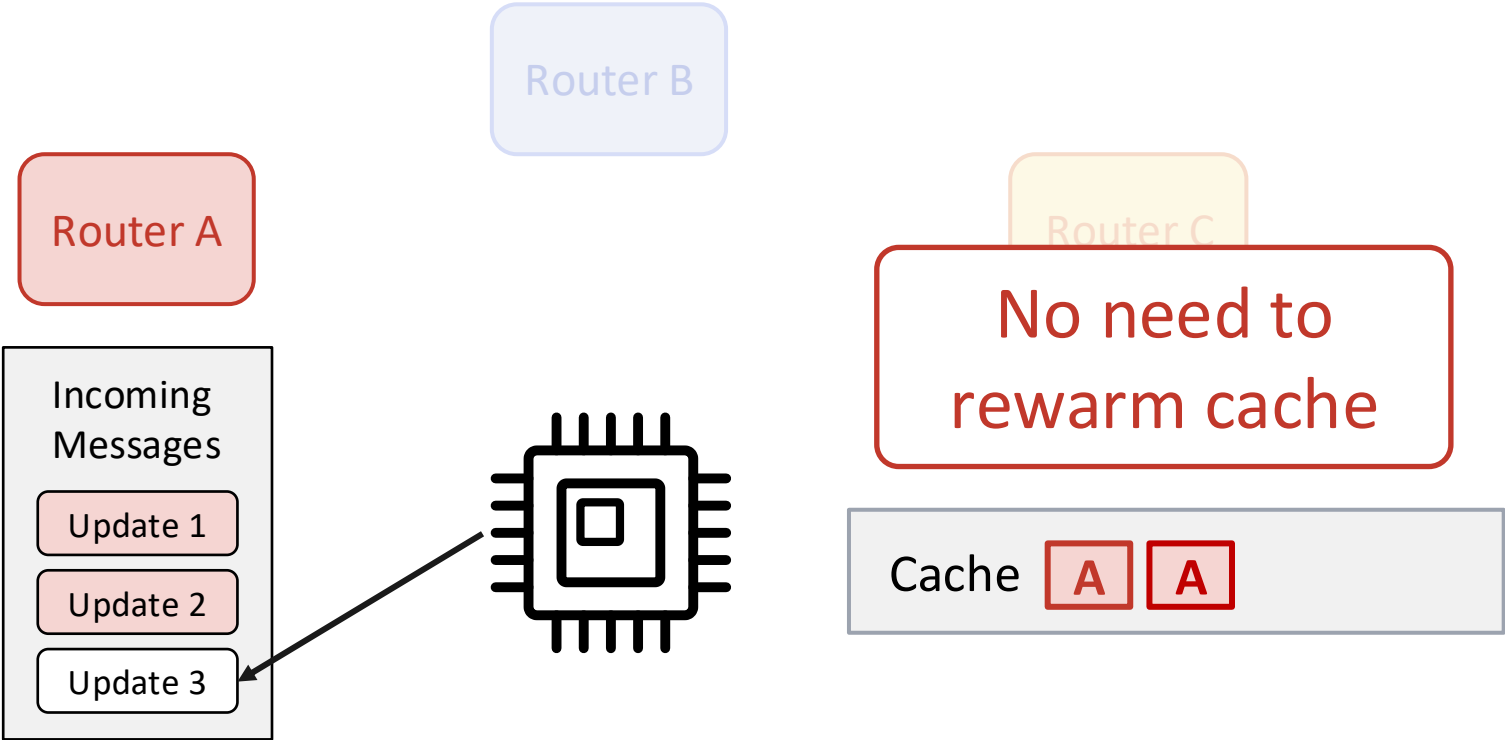
# Run-to-Idle Scheduling: Avoid Unnecessary Cache Miss



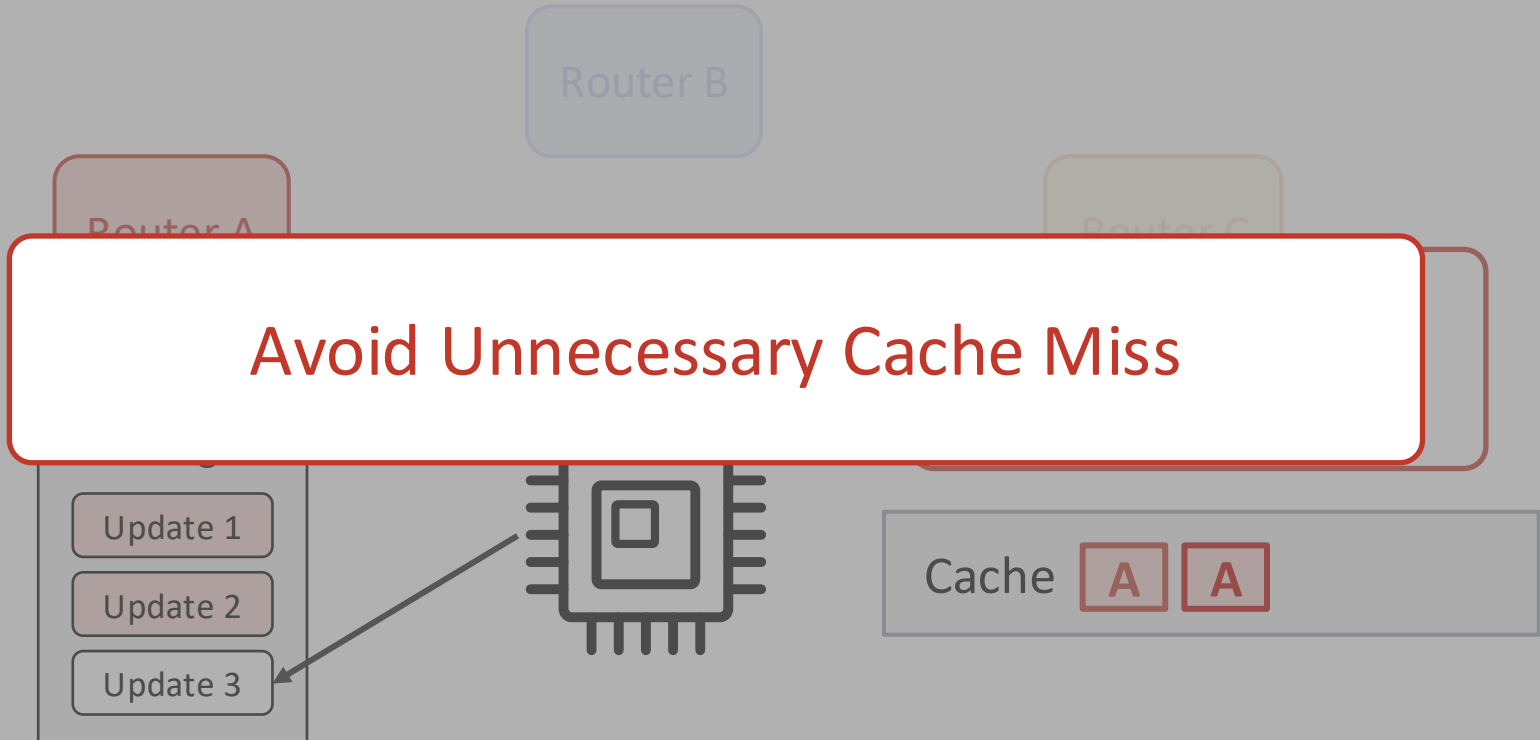
# Run-to-Idle Scheduling: Avoid Unnecessary Cache Miss



# Run-to-Idle Scheduling: Avoid Unnecessary Cache Miss

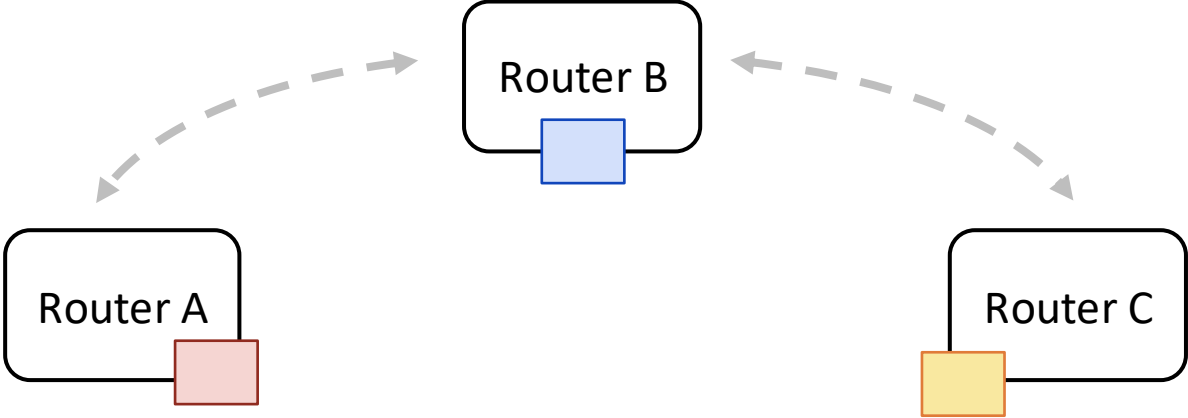


# Run-to-Idle Scheduling: Avoid Unnecessary Cache Miss

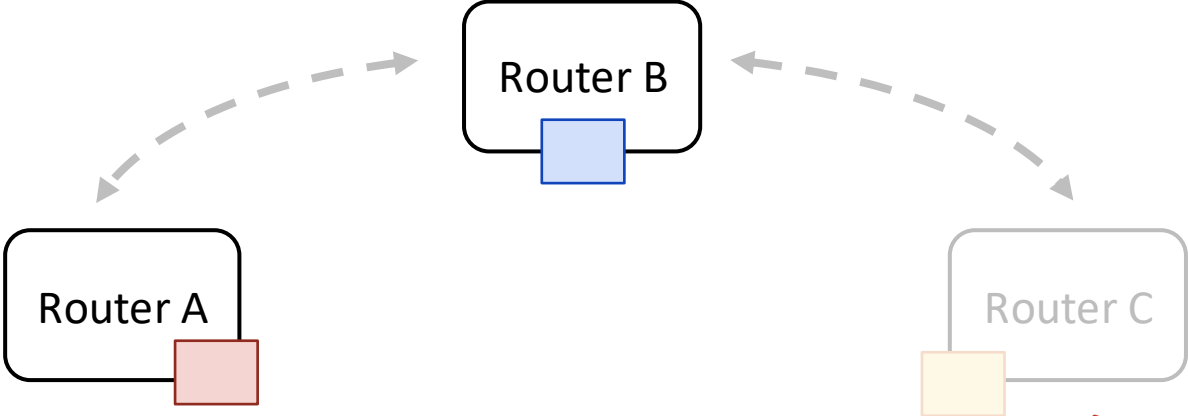


Iterative Converge: Reduce Memory Residency

# Iterative Converge: Reduce Memory Residency

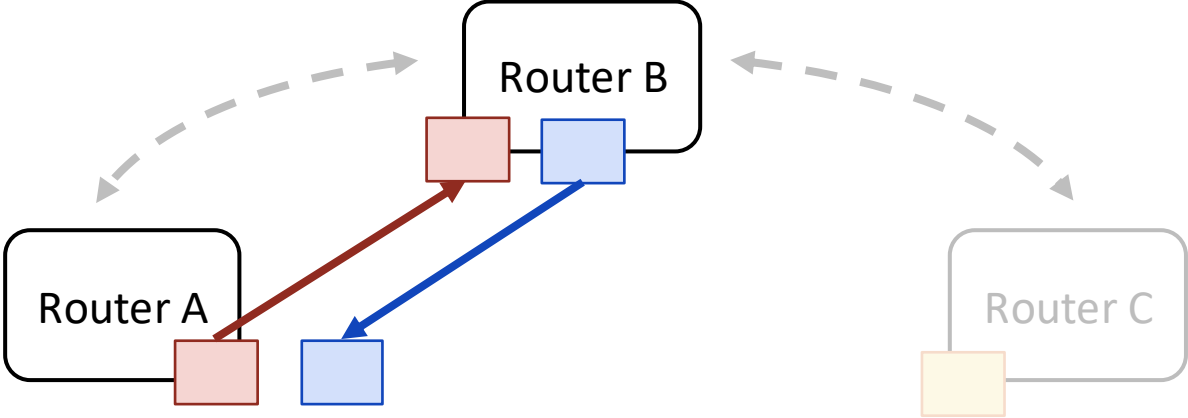


# Iterative Converge: Reduce Memory Residency

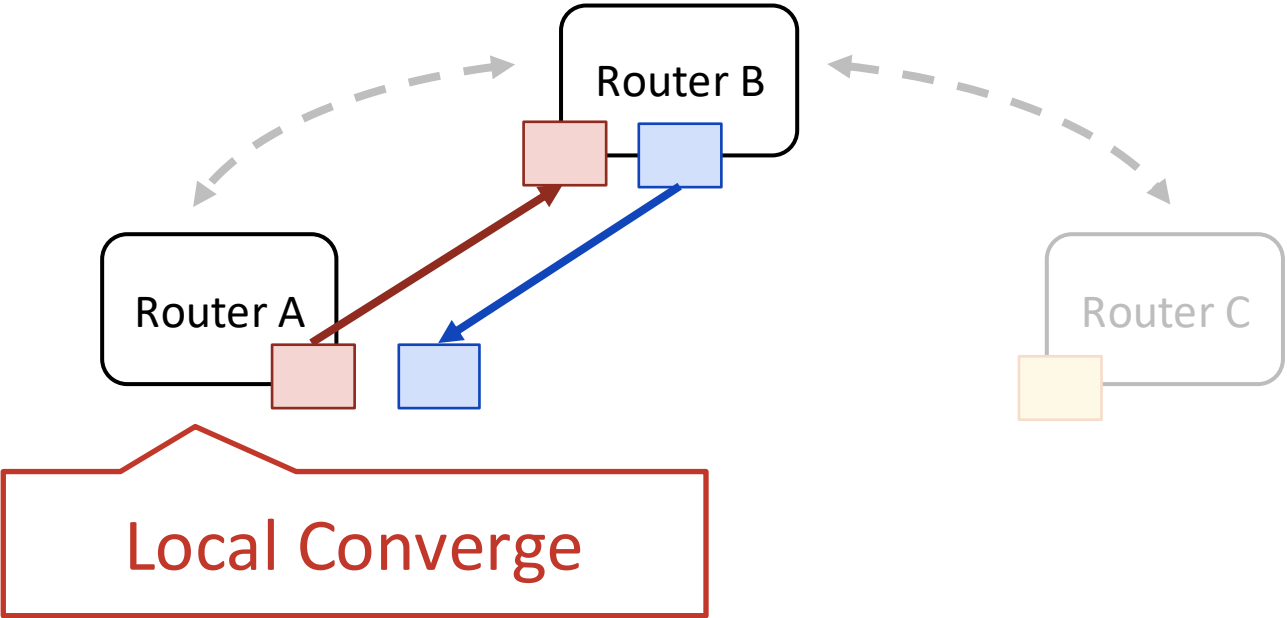


Not Resident yet with Iterative Converge

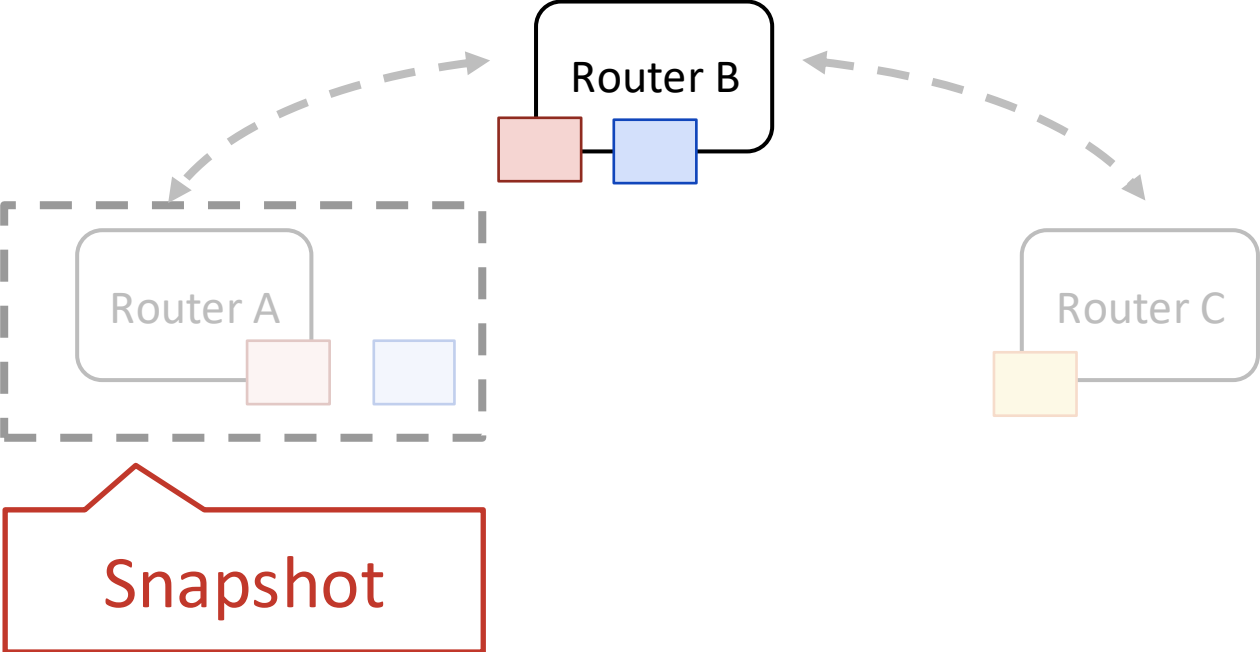
# Iterative Converge: Reduce Memory Residency



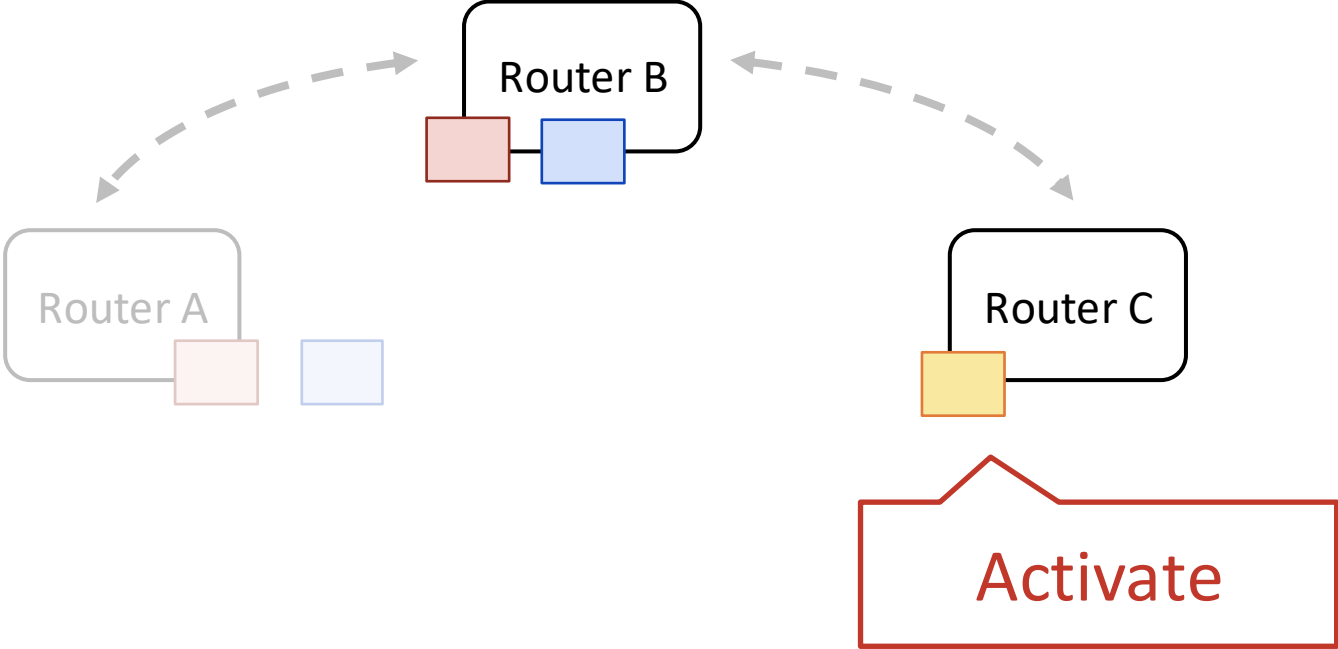
# Iterative Converge: Reduce Memory Residency



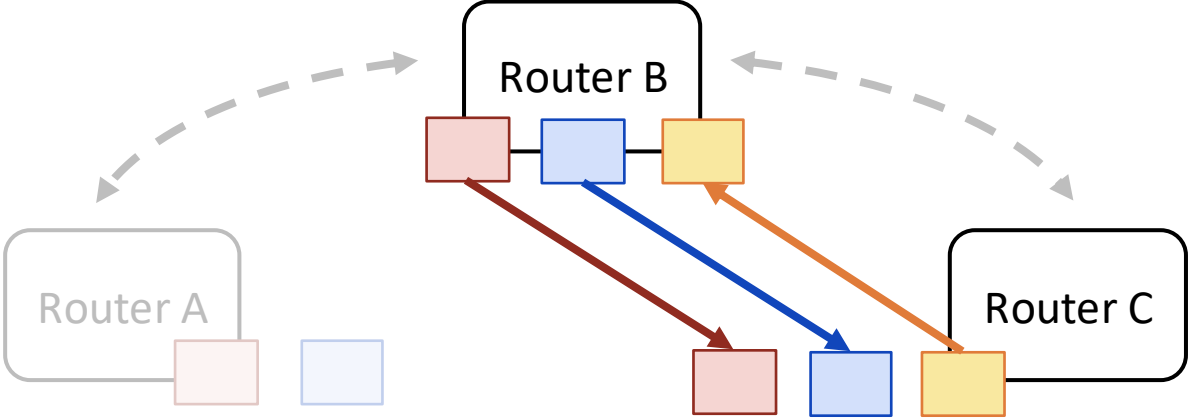
# Iterative Converge: Reduce Memory Residency



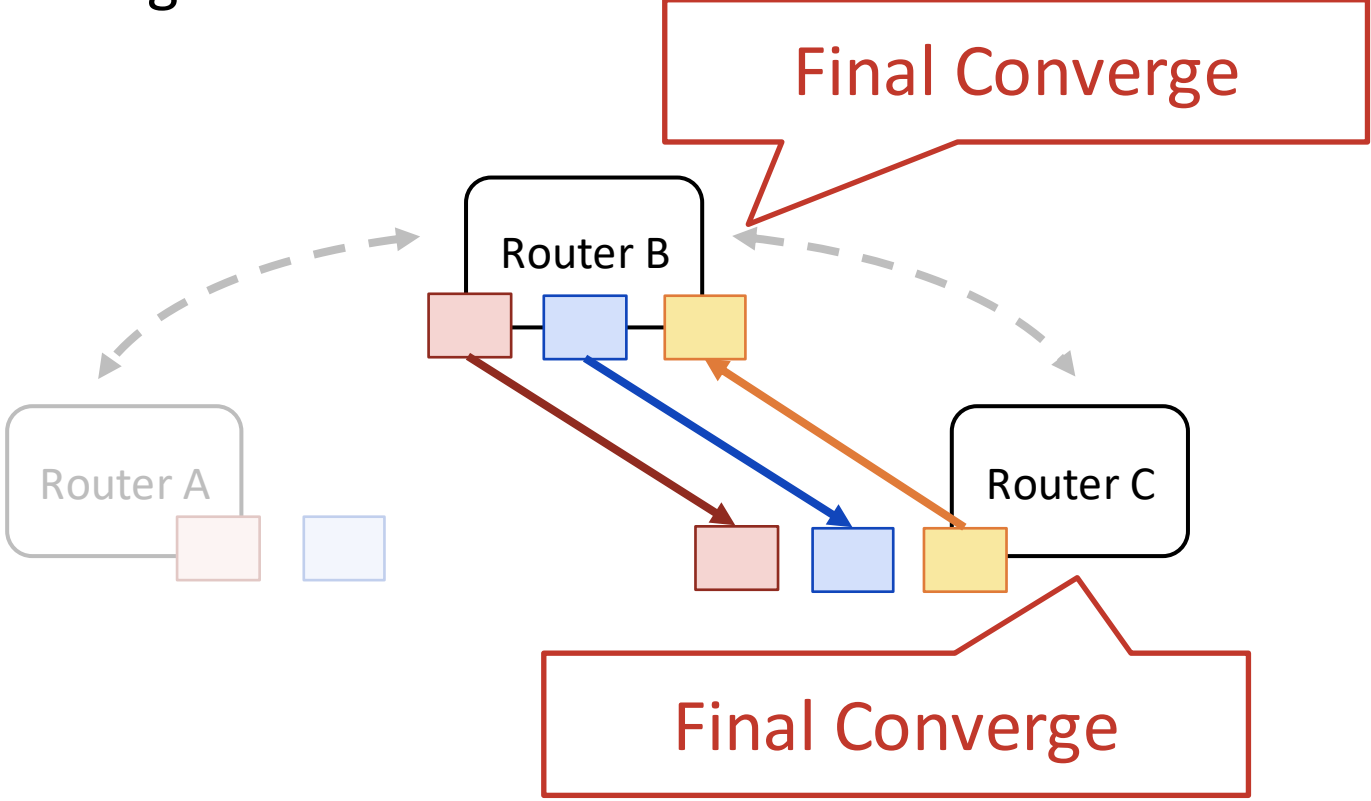
# Iterative Converge: Reduce Memory Residency



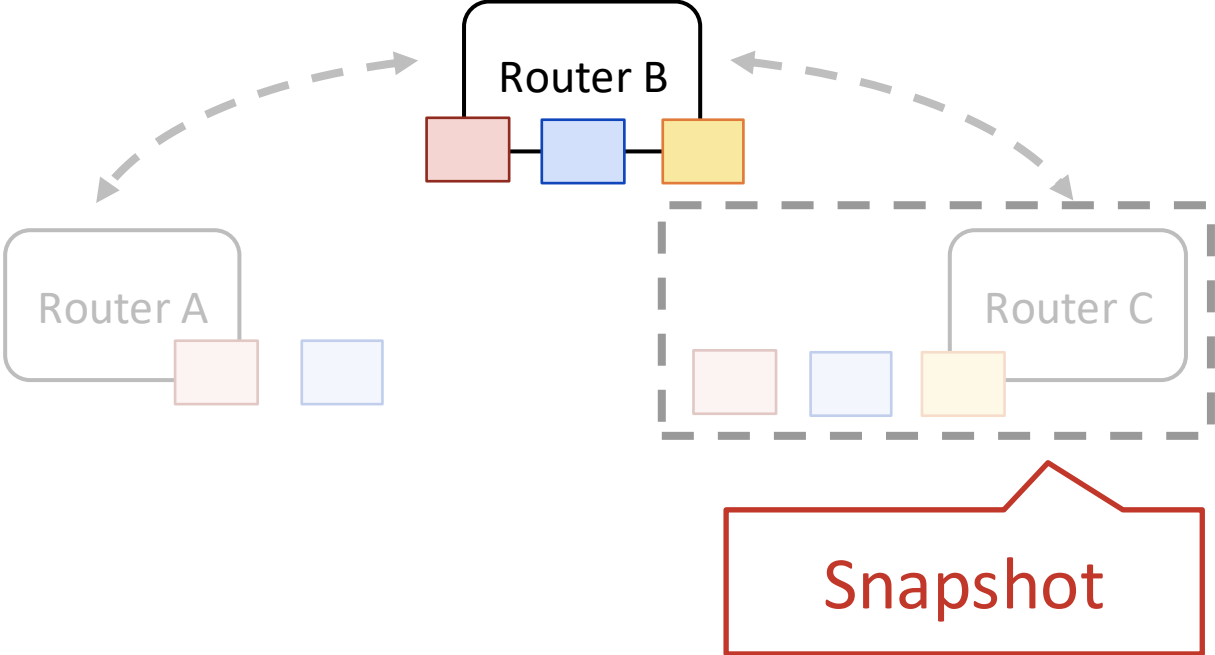
# Iterative Converge: Reduce Memory Residency



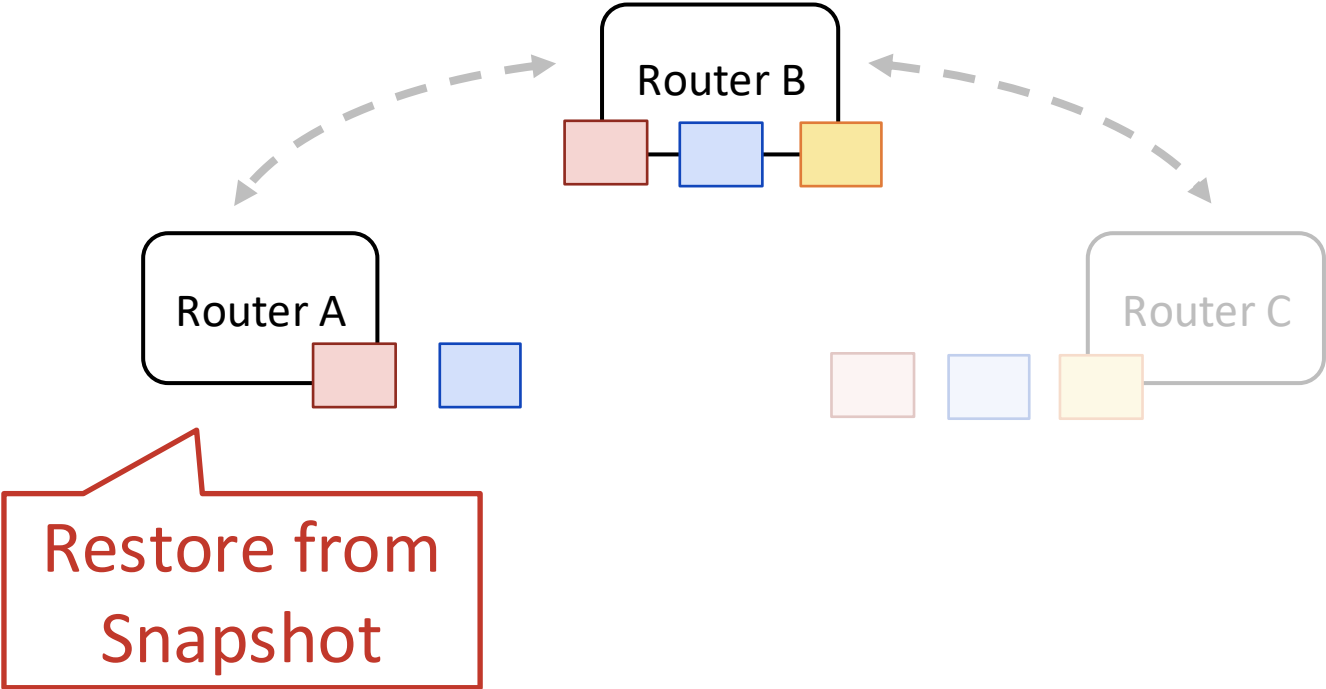
# Iterative Converge: Reduce Memory Residency



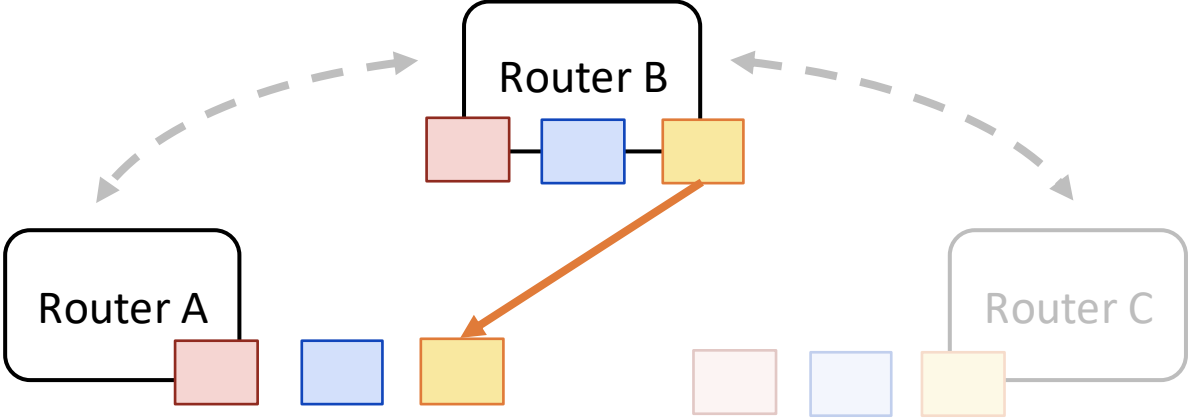
# Iterative Converge: Reduce Memory Residency



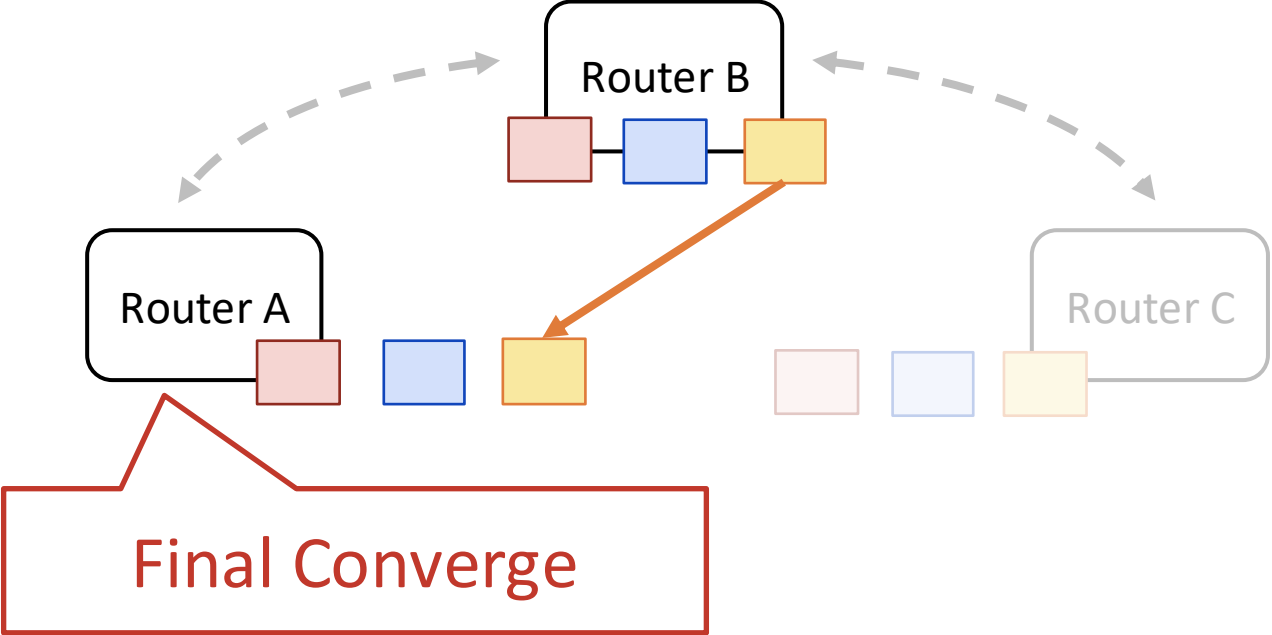
# Iterative Converge: Reduce Memory Residency



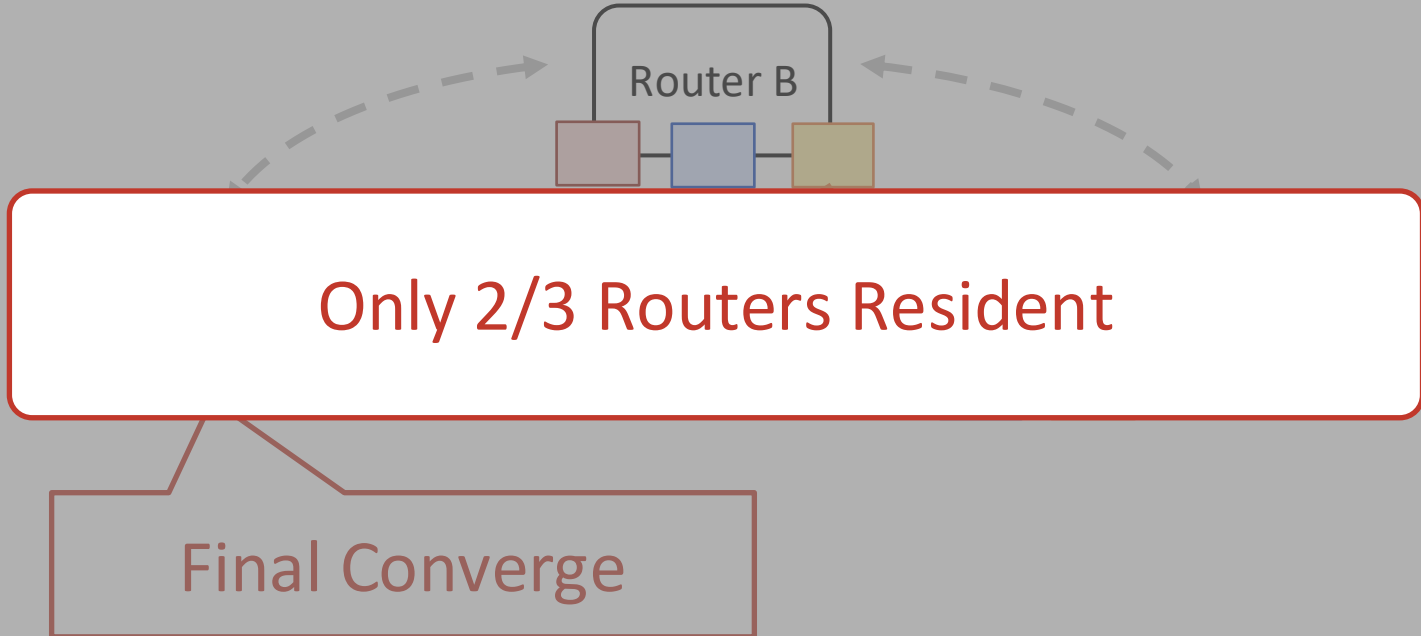
# Iterative Converge: Reduce Memory Residency



# Iterative Converge: Reduce Memory Residency



# Iterative Converge: Reduce Memory Residency

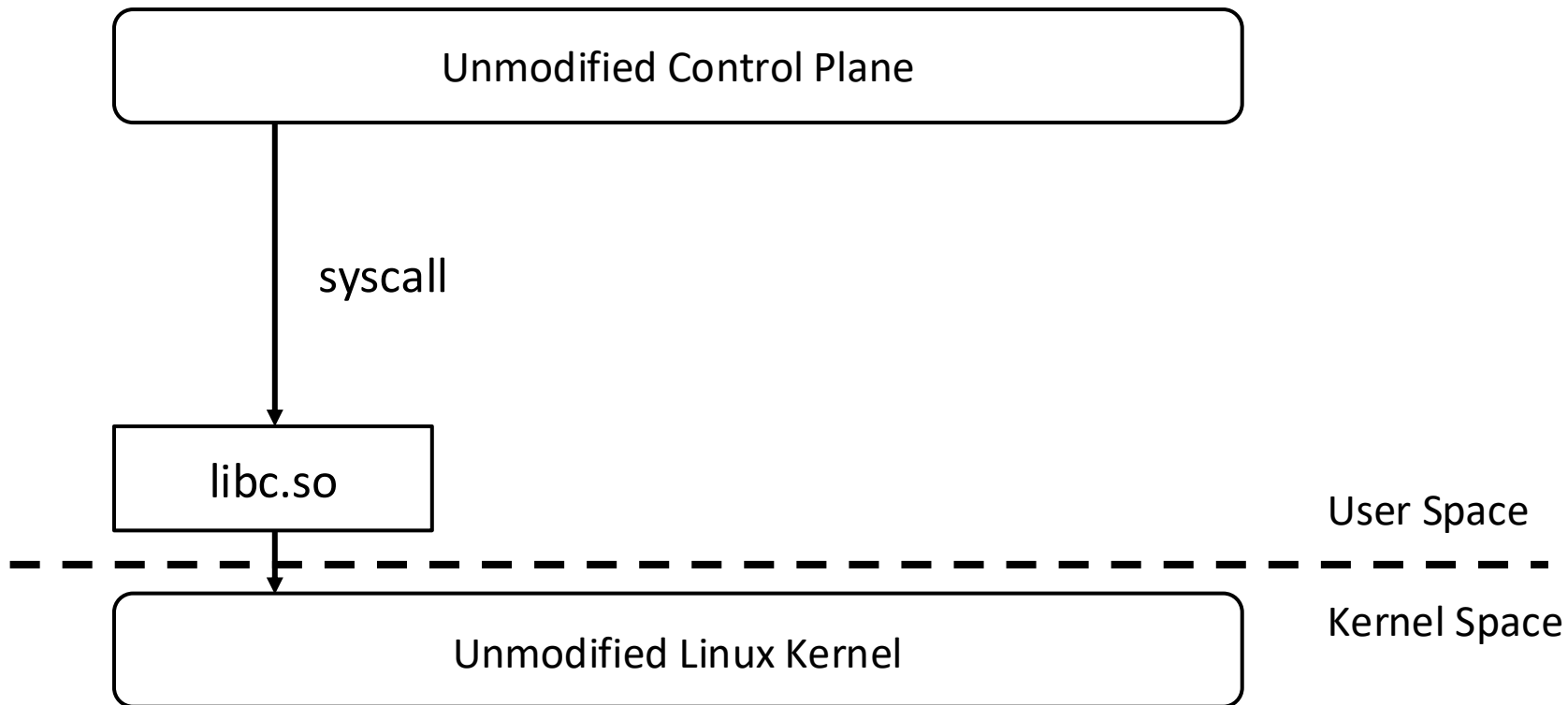


How to realize such a runtime?

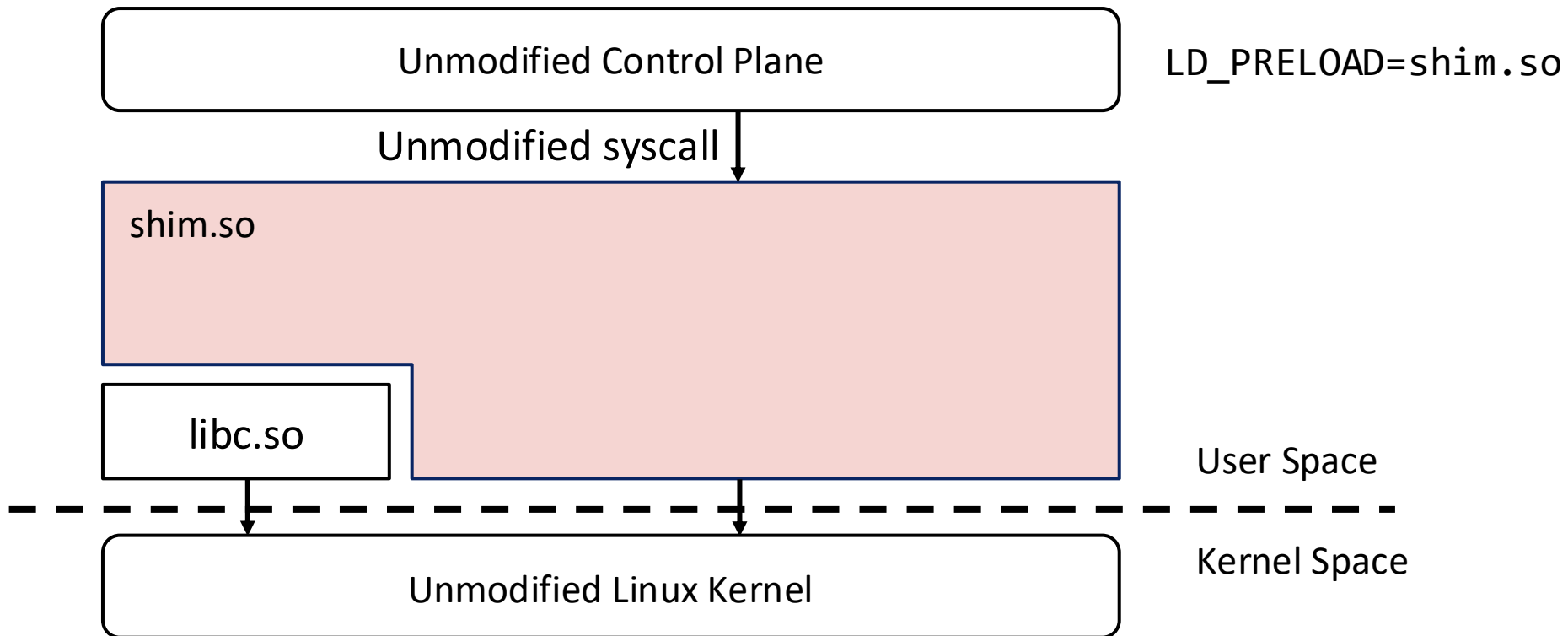
## Goal: Not Touching Software

- Modifying vendor image **hurts fidelity**, if we can at all
- Modifying Linux kernel means **hard deployment**
- **Our approach: Transparent Syscall Interposition**

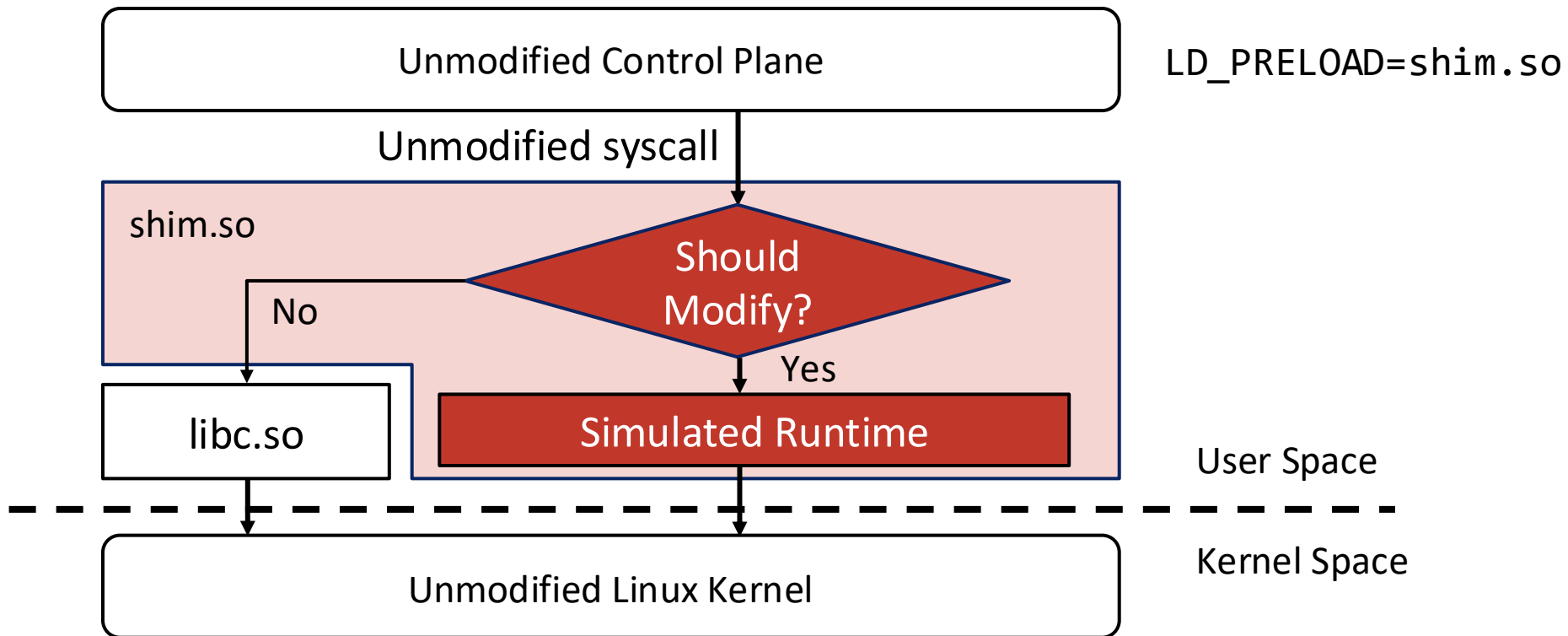
# Transparent Syscall Interposition via LD\_PRELOAD



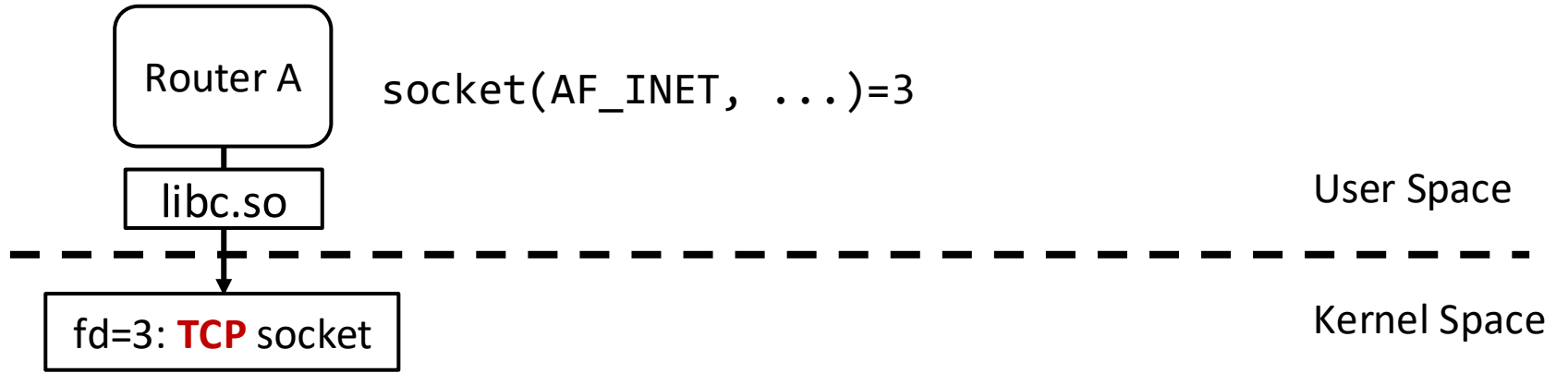
# Transparent Syscall Interposition via LD\_PRELOAD



# Transparent Syscall Interposition via LD\_PRELOAD

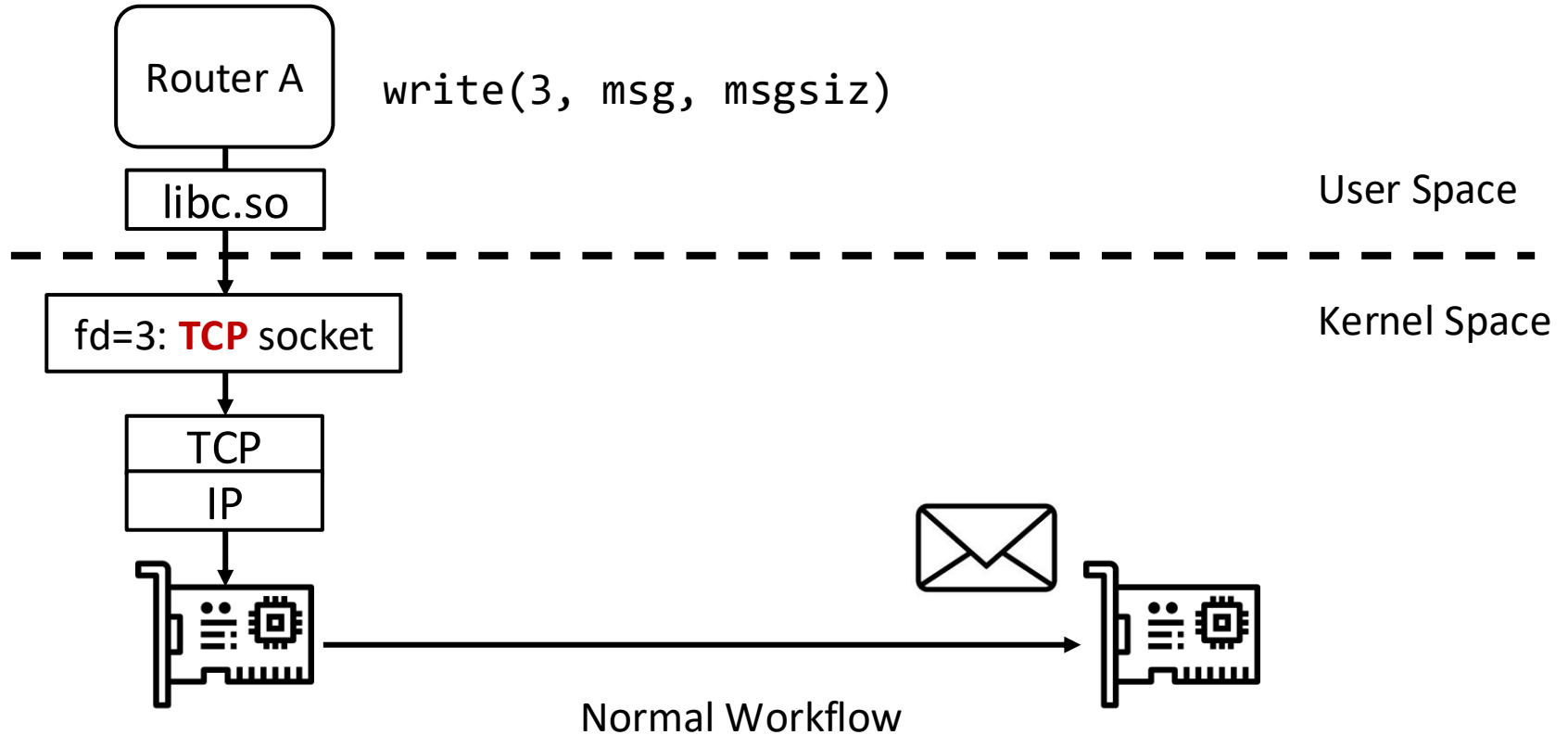


# Simulate TCP Semantics with IPC

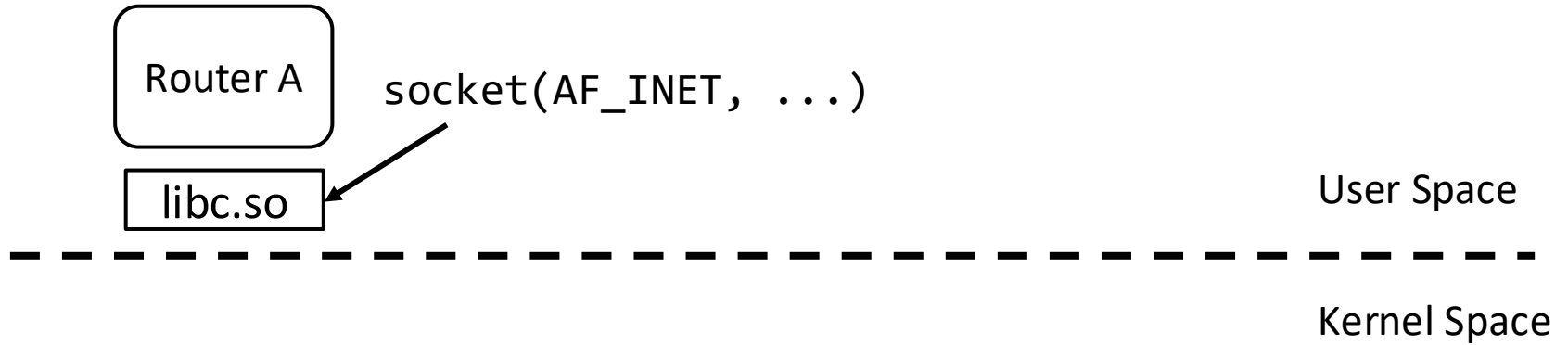


Normal Workflow

# Simulate TCP Semantics with IPC

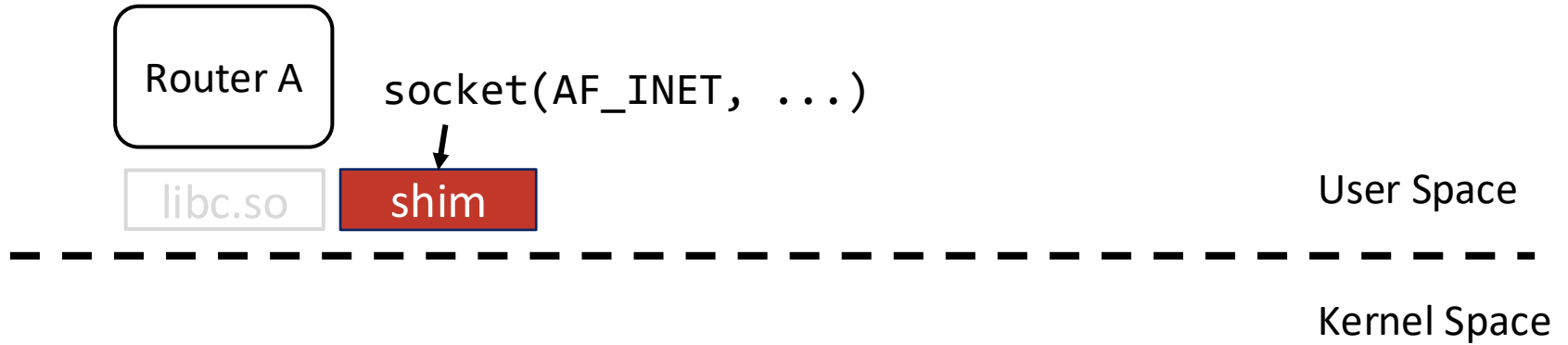


# Simulate TCP Semantics with IPC



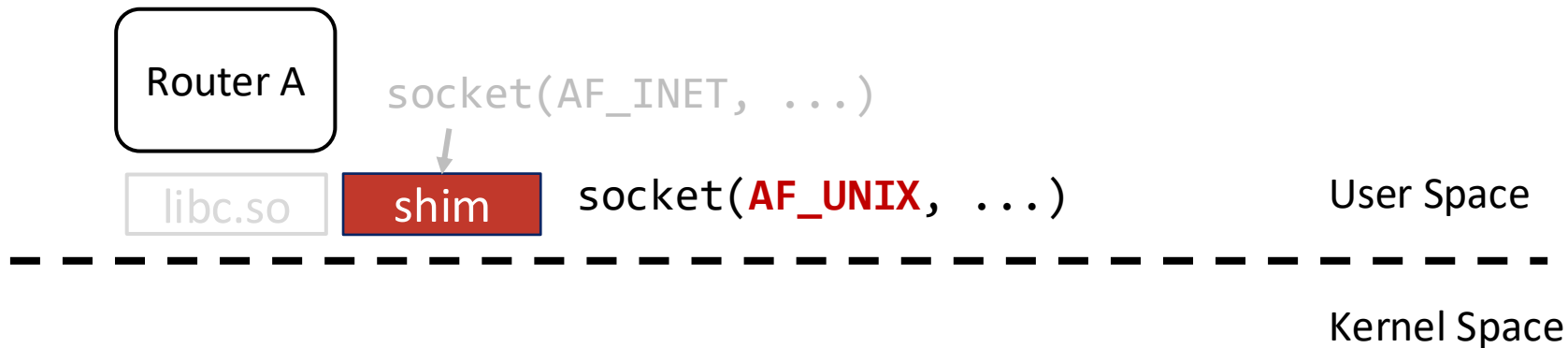
linker dynamically finds implementation of socket()

# Simulate TCP Semantics with IPC



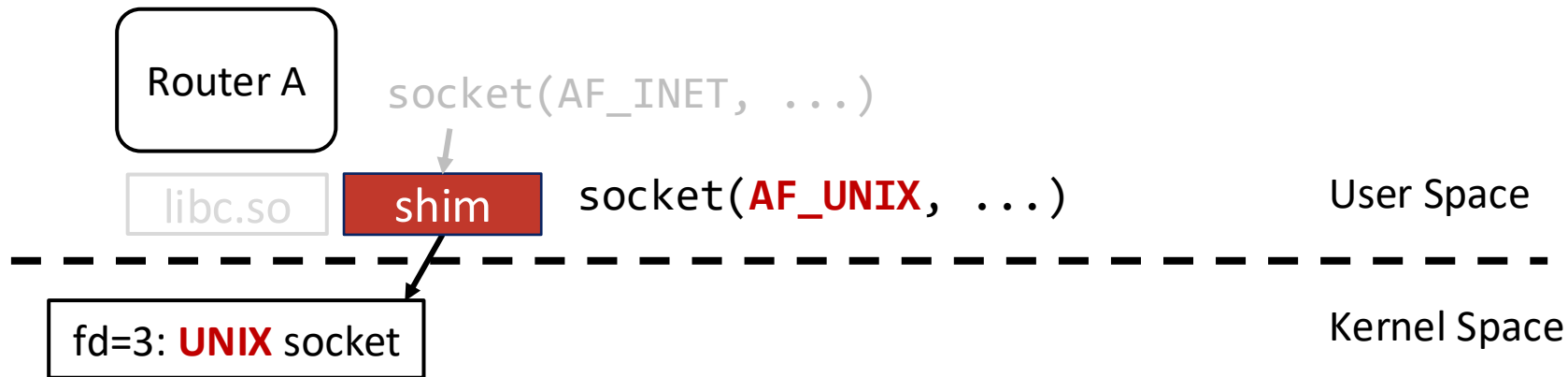
... which can be redirected to custom implementation

# Simulate TCP Semantics with IPC



... which can be redirected to custom implementation

# Simulate TCP Semantics with IPC



# Simulate TCP Semantics with IPC



`write(3, msg, msgsiz)`

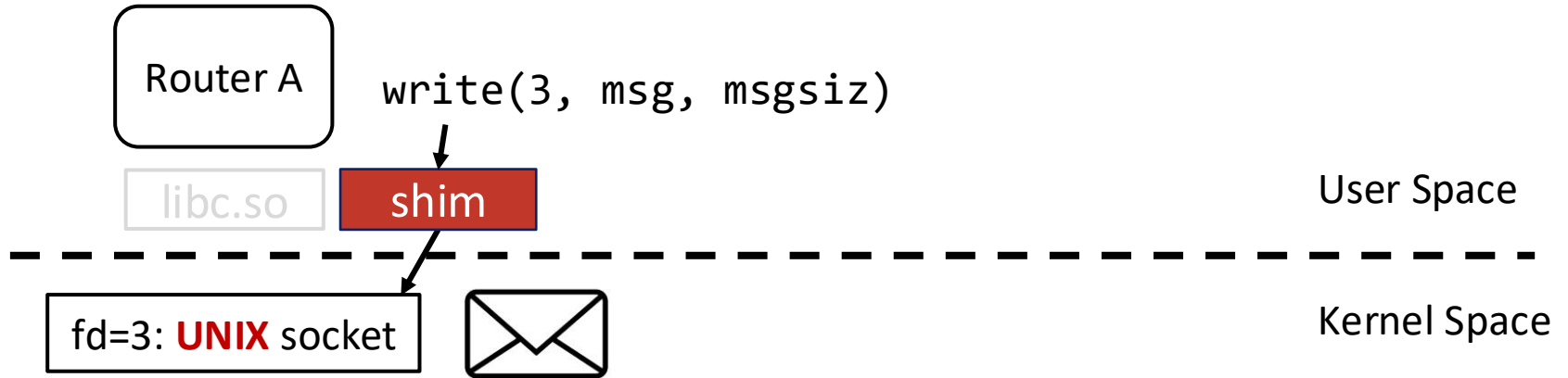
User Space



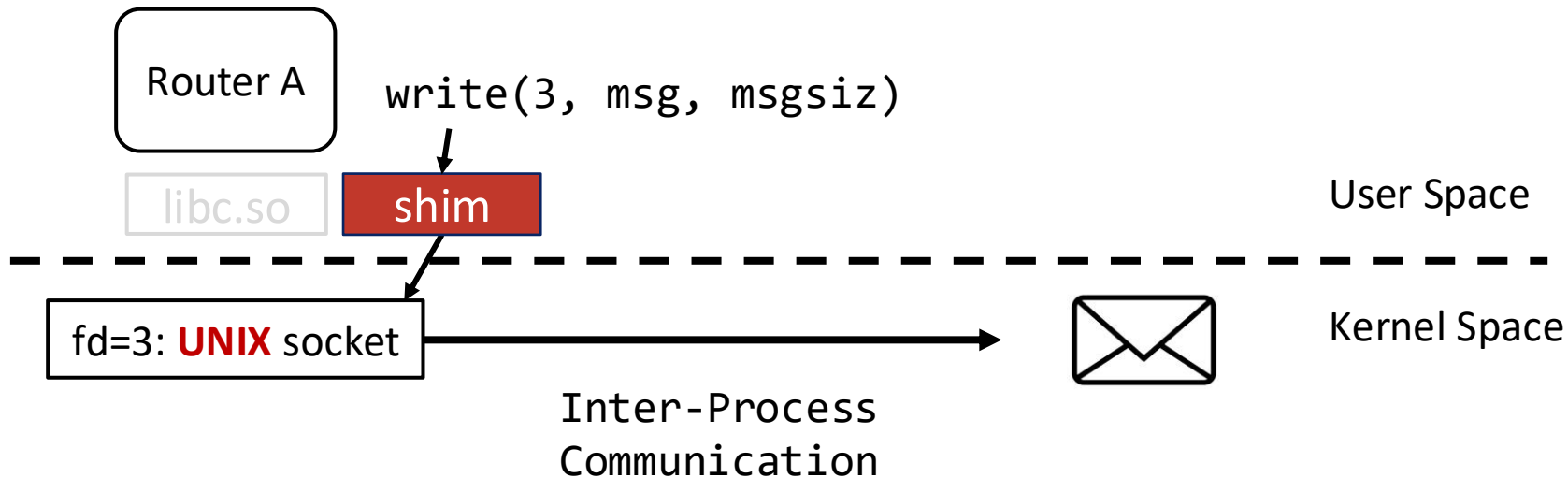
fd=3: **UNIX** socket

Kernel Space

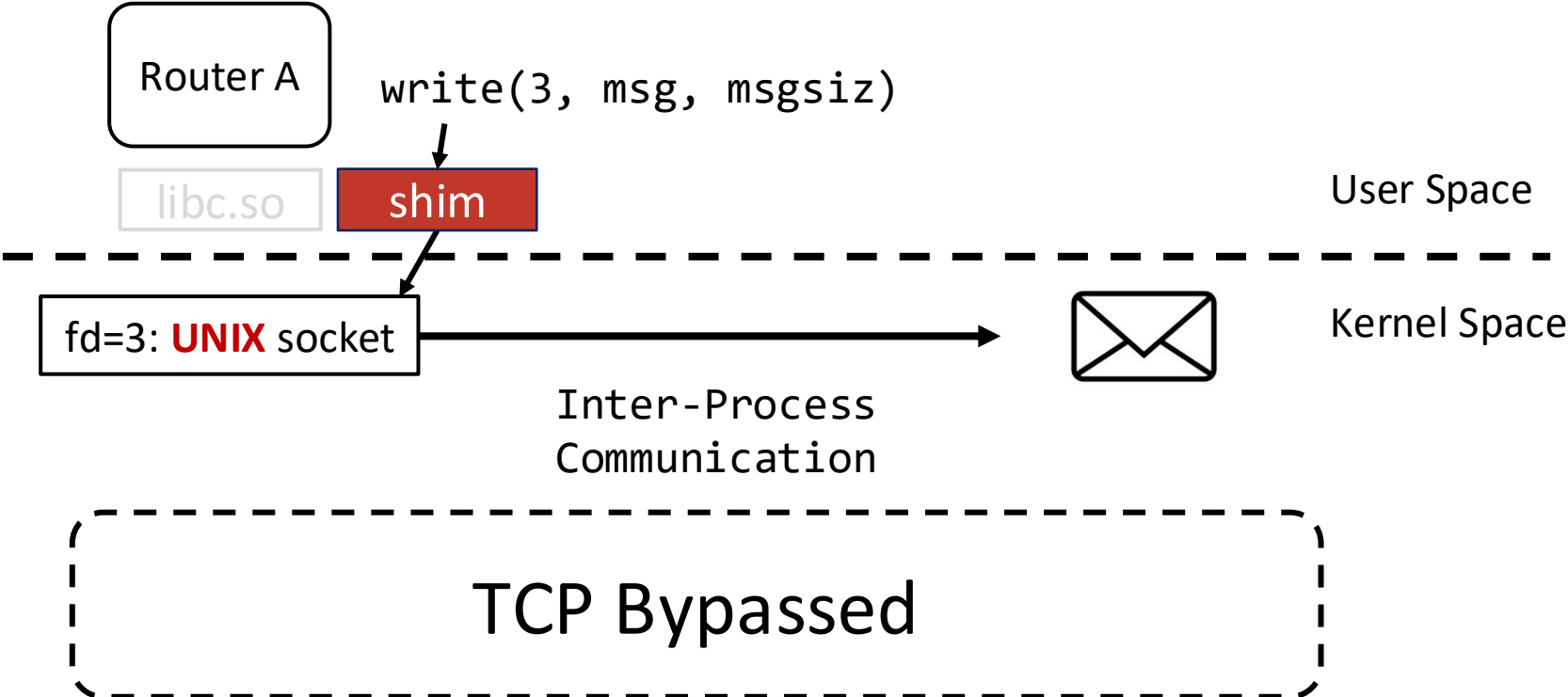
# Simulate TCP Semantics with IPC



# Simulate TCP Semantics with IPC



# Simulate TCP Semantics with IPC



## More in the Paper

- Simulate Scheduling with Controlled Event Delivery
- Simulate Snapshot & Restore with History Message Replay
- Scale-out to Multiple Hosts

## Evaluation: Fidelity

Found a new **data race** in FRRouting

# Evaluation: Fidelity

Found a new **data race** in FRRouting

May happen with every BGP teardown, e.g. a peer router reboots

## Main Thread

```
close(fd=3)
```

```
socket()=3, connect(3,...)
```

## I/O Thread

```
ppoll(fd=[3,...]) blocks...
```

```
... returns POLLOUT on fd 3
```

# Evaluation: Fidelity

Found a new **data race** in FRRouting  
Confirmed by community maintainer

Open

Possible bug in event system? #17507



mjstapp on Nov 26, 2024

Contributor



wow, that's a beauty - thanks for all that detailed info. I'd be very surprised if we expected this kind of fd reuse; I've seen in happen, but I think we'll have to do some work in bgpd to get this fixed.



## Evaluation: Fidelity

Found a new **data race** in FRRouting

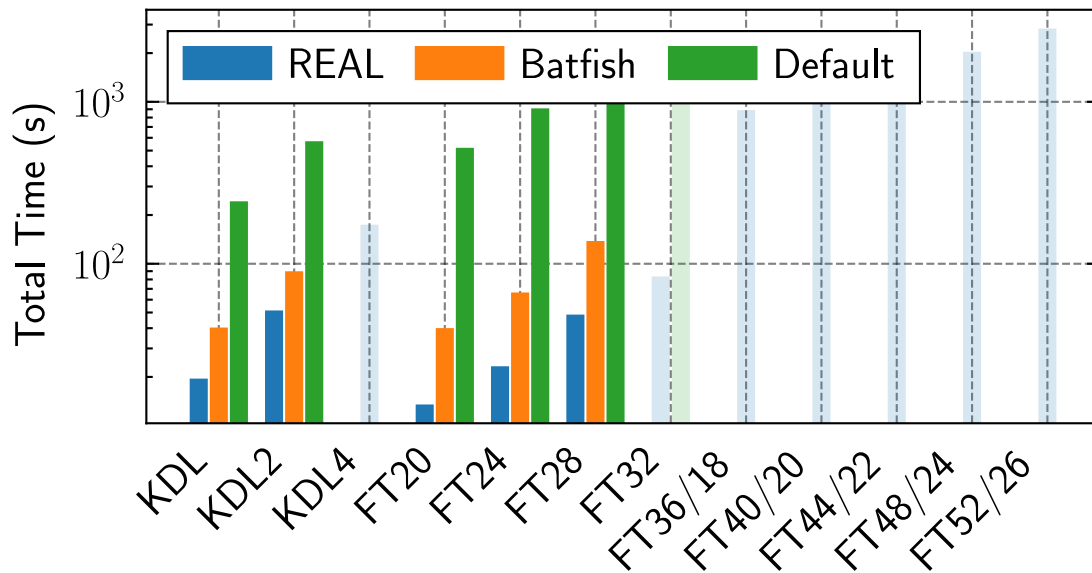
Confirmed by community maintainer

Beyond simulators' capacity: they don't model thread or fd

## Evaluation: Time

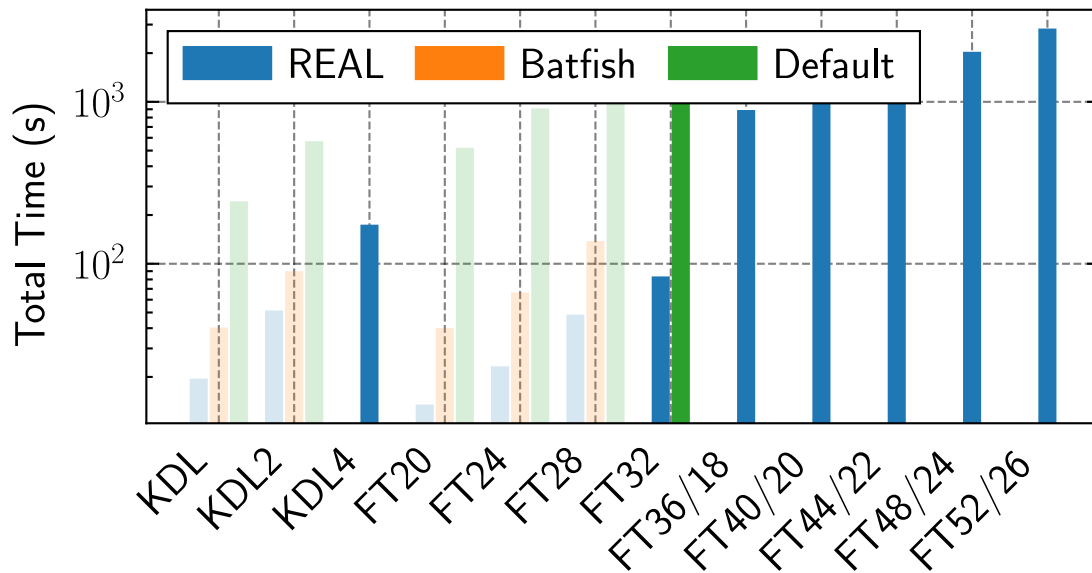
Up to **47x** faster than default Linux runtime

Even faster than Batfish



# Evaluation: Memory

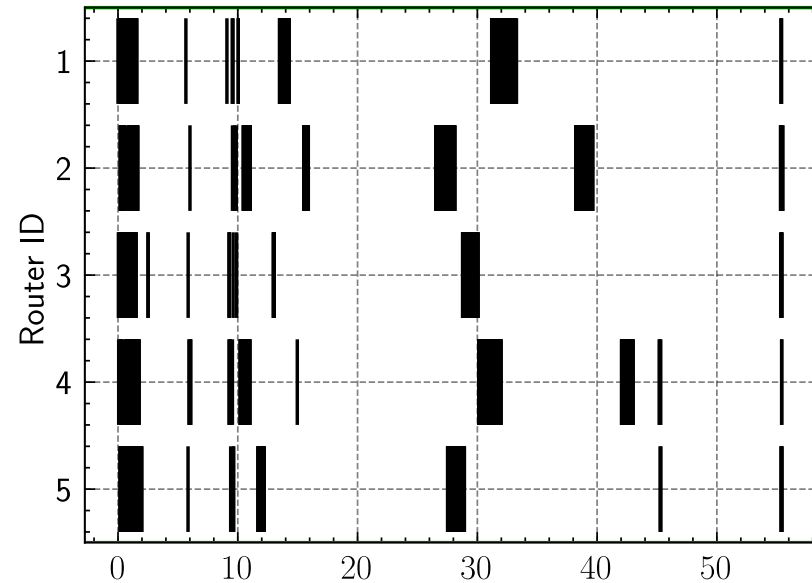
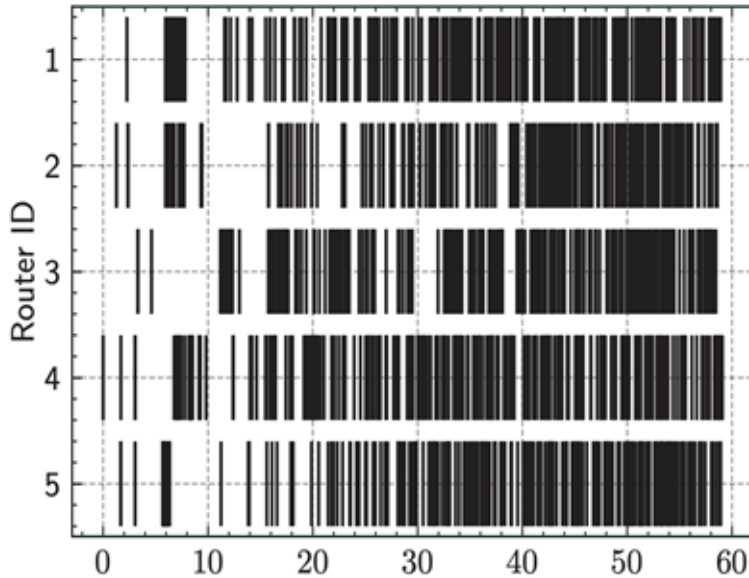
Enables larger topology (**1K**→**3K**) with 256G RAM



# MicroBenchmark: Run-to-Idle Scheduling

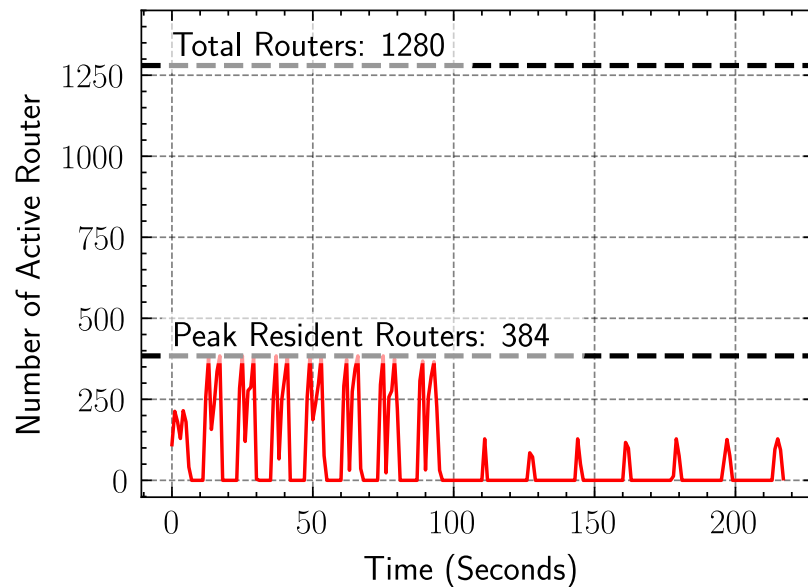
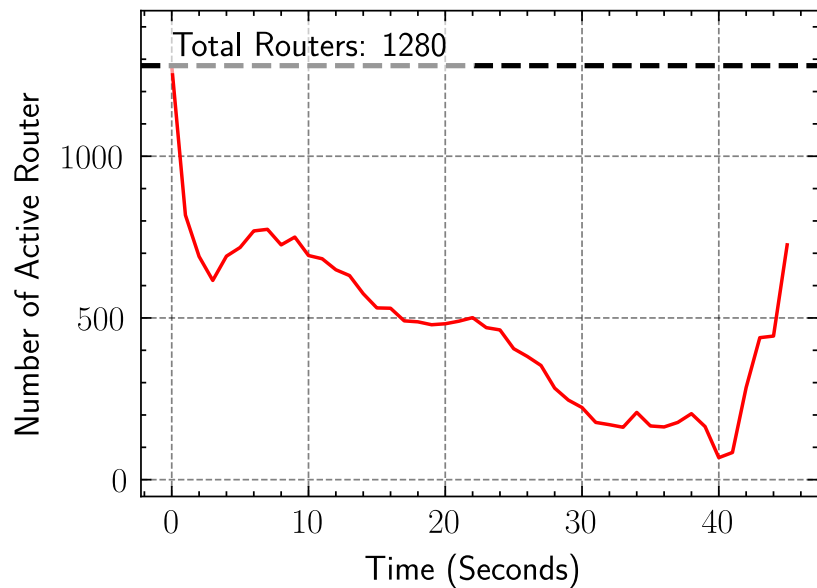
Continuous Execution: Reduced Context Switch

Last-Level-Cache miss **60% → 48%**



# MicroBenchmark: Iterative Converge

Controlled Residency: 1280  $\rightarrow$  384, **>3x** Reduced



# Conclusion

Efficiency

● Simulation

● **REAL**

Fidelity of Emulation,  
Efficiency of Simulation.

● Emulation

Fidelity



Conclusion

Efficiency

Insight: Emulate the Control Plane,  
Simulate the Runtime

tion,  
ulation.

○ Emulation

Fidelity

## Conclusion

- Emulation cost is rooted in a mismatched runtime
- **REAL**: simulated runtime without image modification
- Faster, Larger topology, Less cache & memory footprint



<https://github.com/ants-xjtu/REAL-artifact-evaluation>

Email: xiaze2017@stu.xjtu.edu.cn