REAL: Emulating Control Plane at Simulator's Cost

Ze Xia[1], Hao Li[1], Jinyu Fu[1], Xin Wan[1], Yihan Dang[1], Danfeng Shan[1], Li Chen[2] and Peng Zhang[1]

1 西安交通大學 XI'AN JIAOTONG UNIVERSITY

2 HARNETS.AI

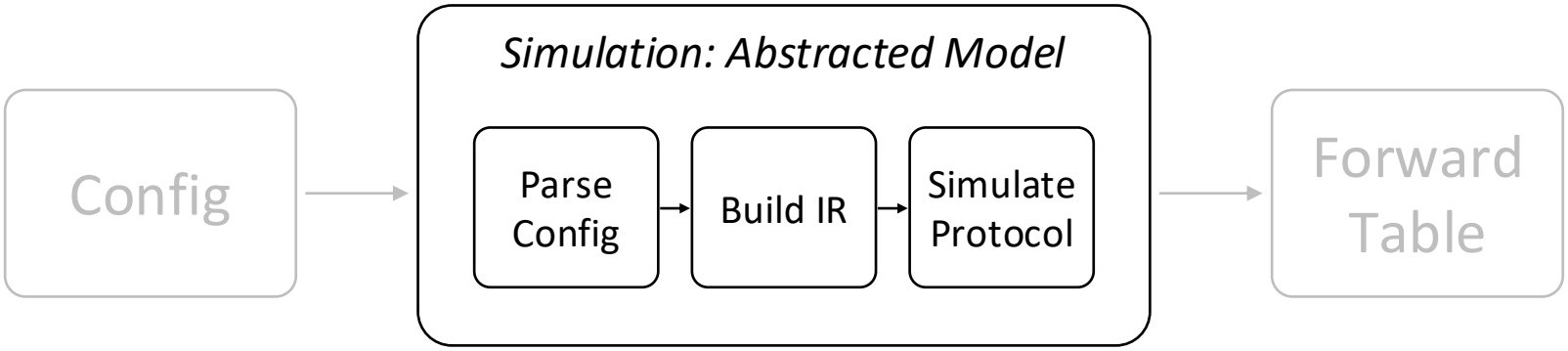
When Control Plane Goes Wrong

When Control Plane Goes Wrong

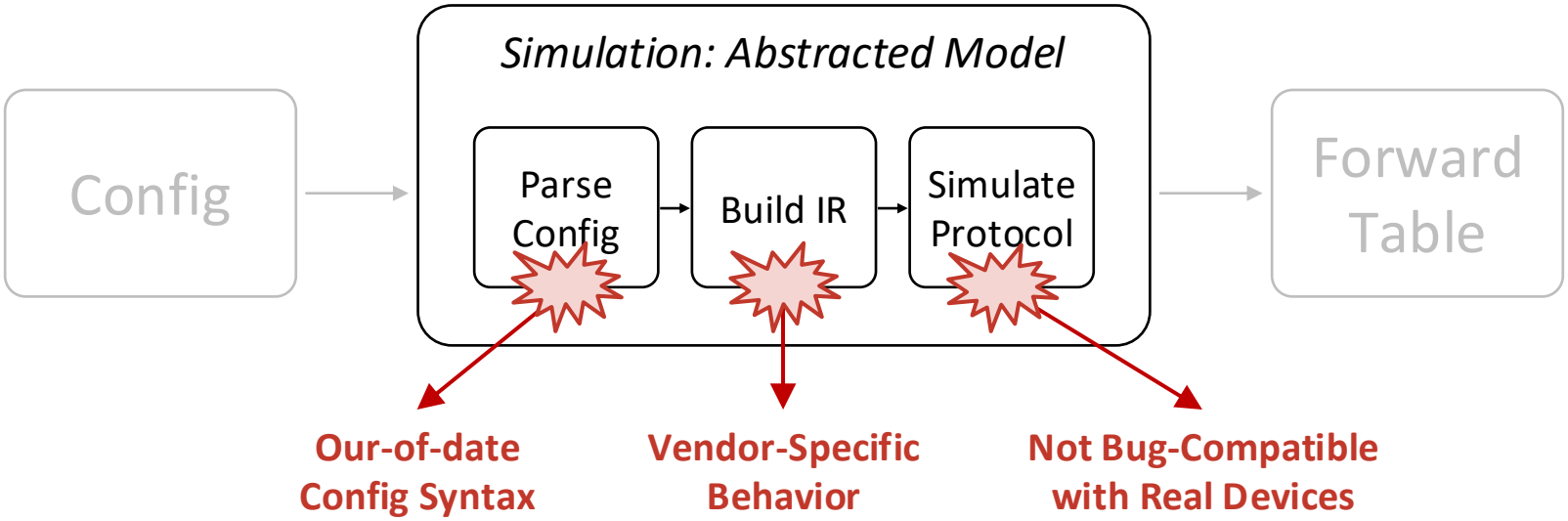
When Control Plane Goes Wrong

Control Plane: Configuration to Forward Table

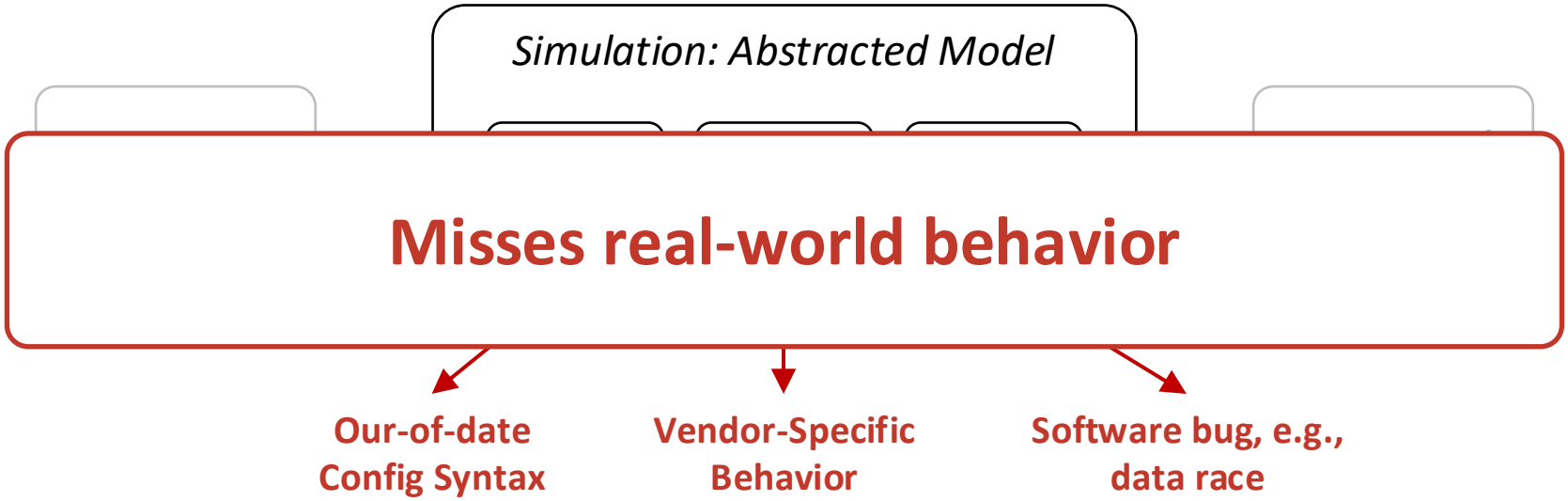
Modeling Control Plane Behavior: Simulation



Simulation Misses Real-World Behavior



Simulation Misses Real-World Behavior



Emulation: Real Software

Actual Binaries — No Abstraction Gap

Reality Is Expensive

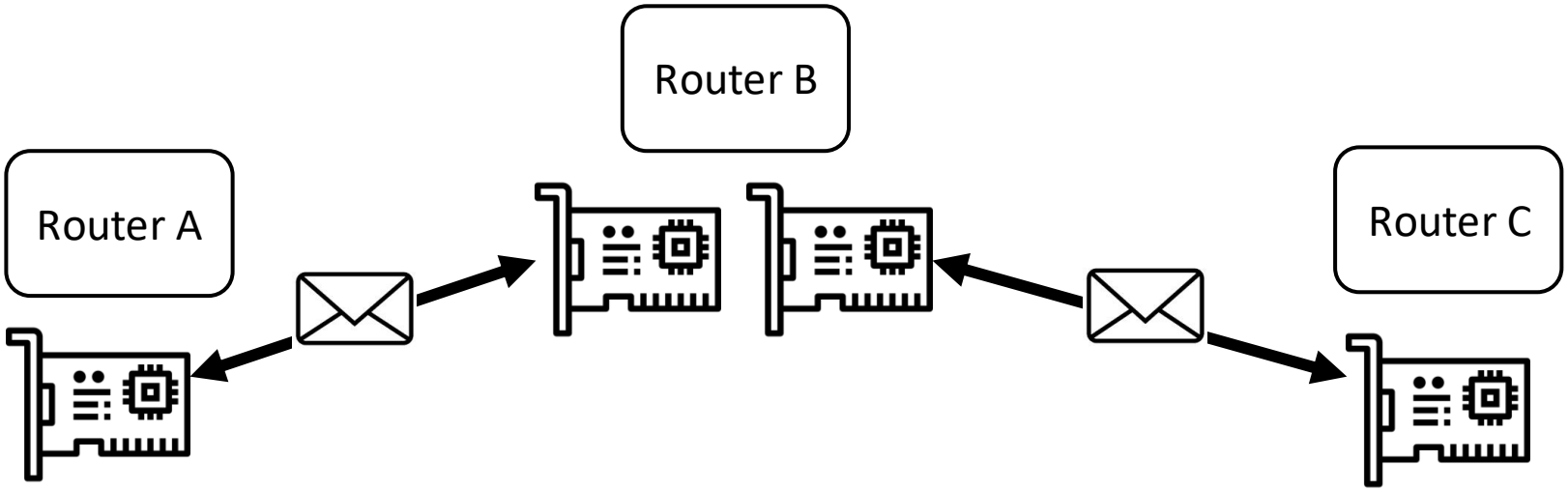
	Topology Scale	CPU count	Memory	Time
CrystalNet [SOSP'17]	~4000	2000-4000	4TB-8TB	~30 min
Crescent [NSDI'24]	1000+	512	2TB	~50 min

Reality Is Expensive

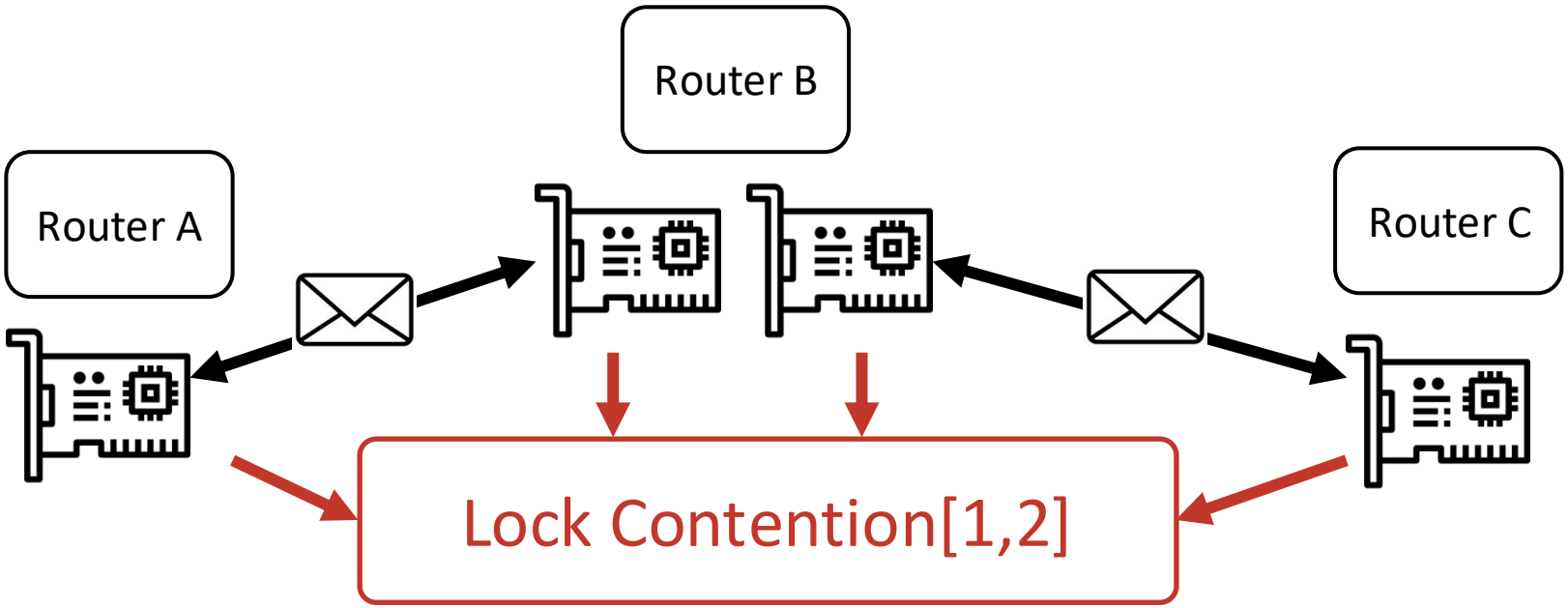
	Topology Scale	CPU count	Memory	Time
CrystalNet [SOSP'17]	~4000	2000-4000	4TB-8TB	~30 min
Crescent [NSDI'24]	1000+	512	2TB	~50 min
Batfish [SIGCOMM'23]	**~3000**	**10**	**64GB**	**<5 min**

Where does the cost come from?

Cost 1: Network Buildup Time

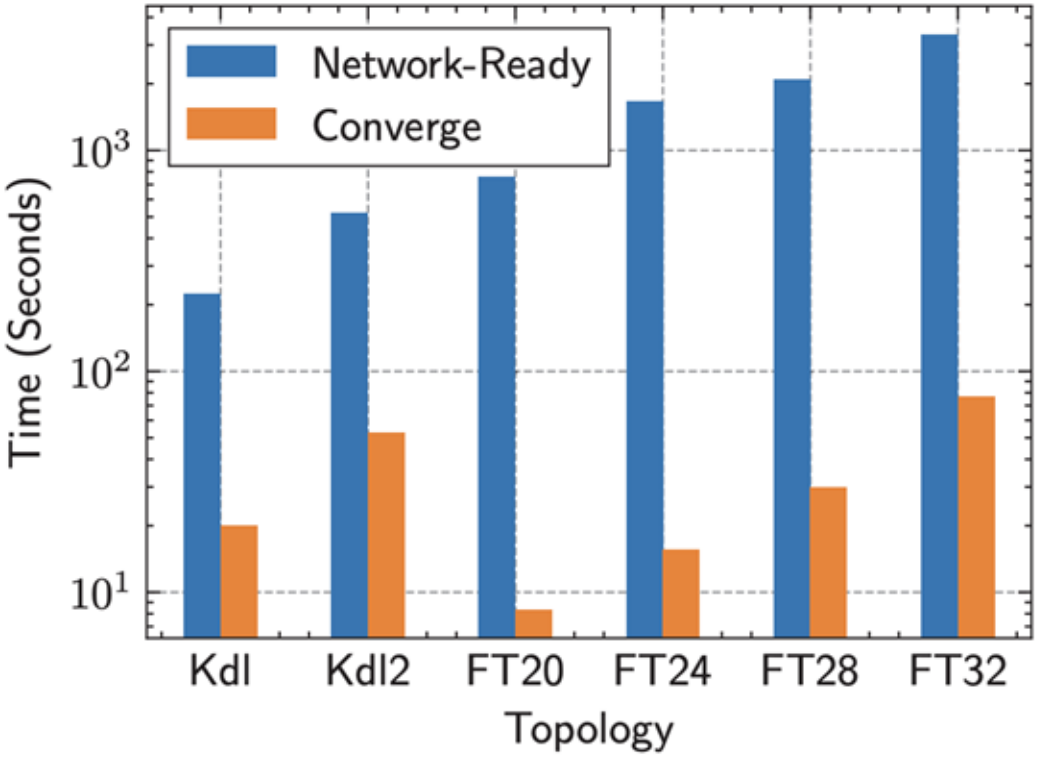


Cost 1: Network Buildup Time

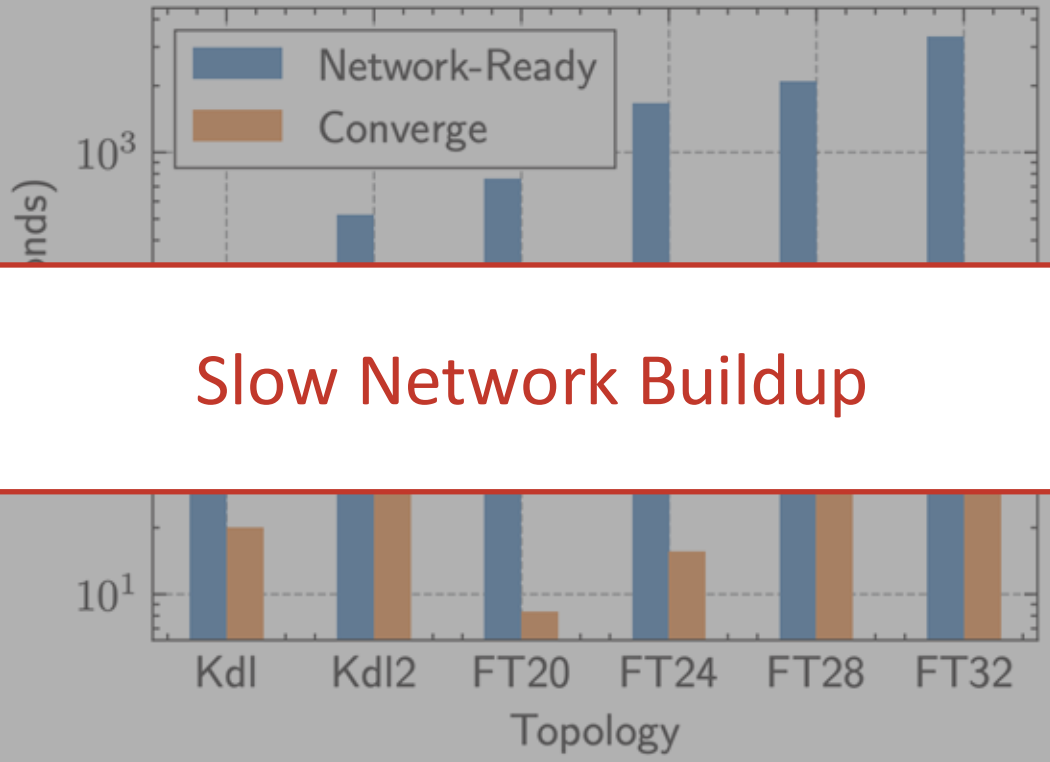


[1] Liu et al. "Understanding Network Startup for Secure Containers in Multi-Tenant Clouds: Performance, Bottleneck and Optimization." IMC '24
[2] Peng et al. "SplitNN: Single-Machine Network Emulation at Scale with Minute-Level Construction of 10K-Node Virtual Networks." APNET '25

Cost 1: Network Buildup Time

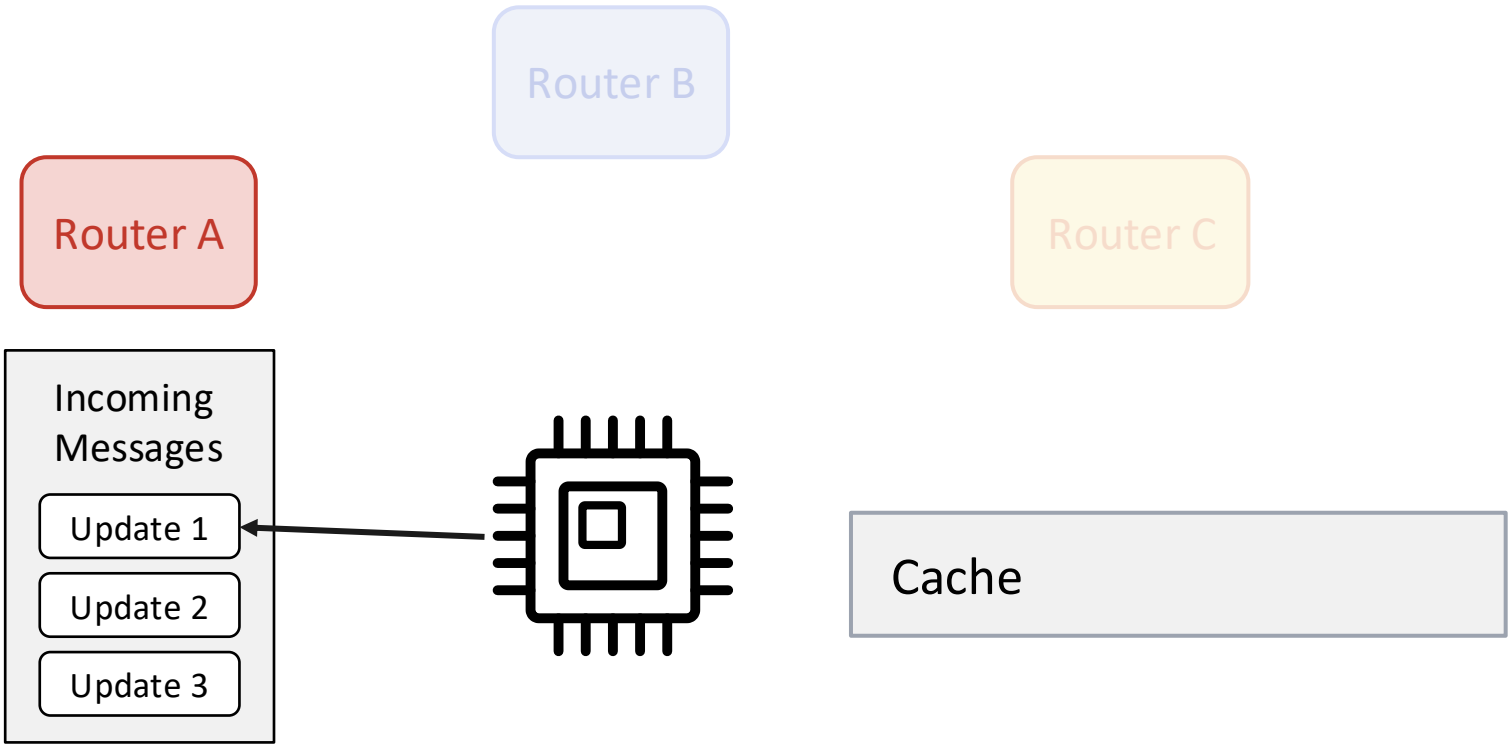


Cost 1: Network Buildup Time

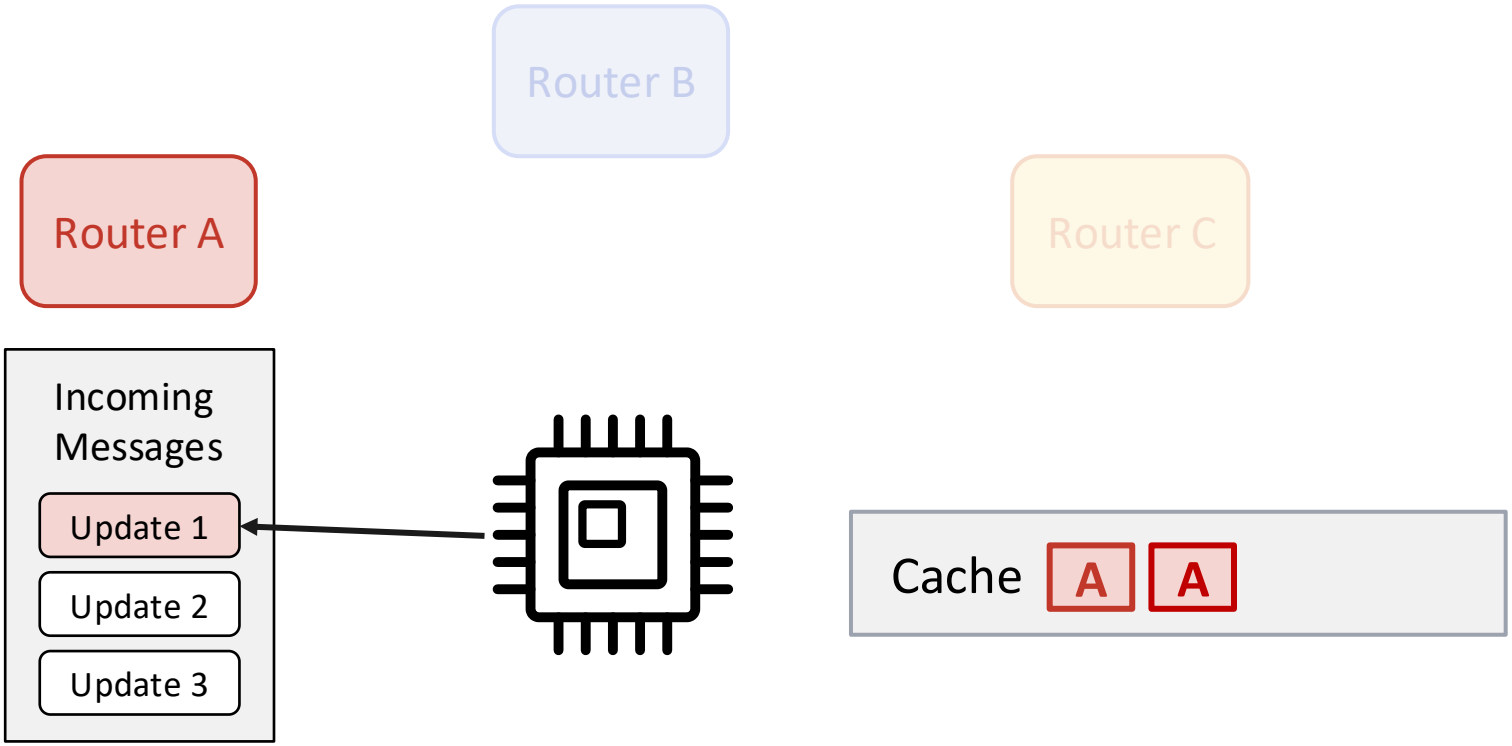


Slow Network Buildup

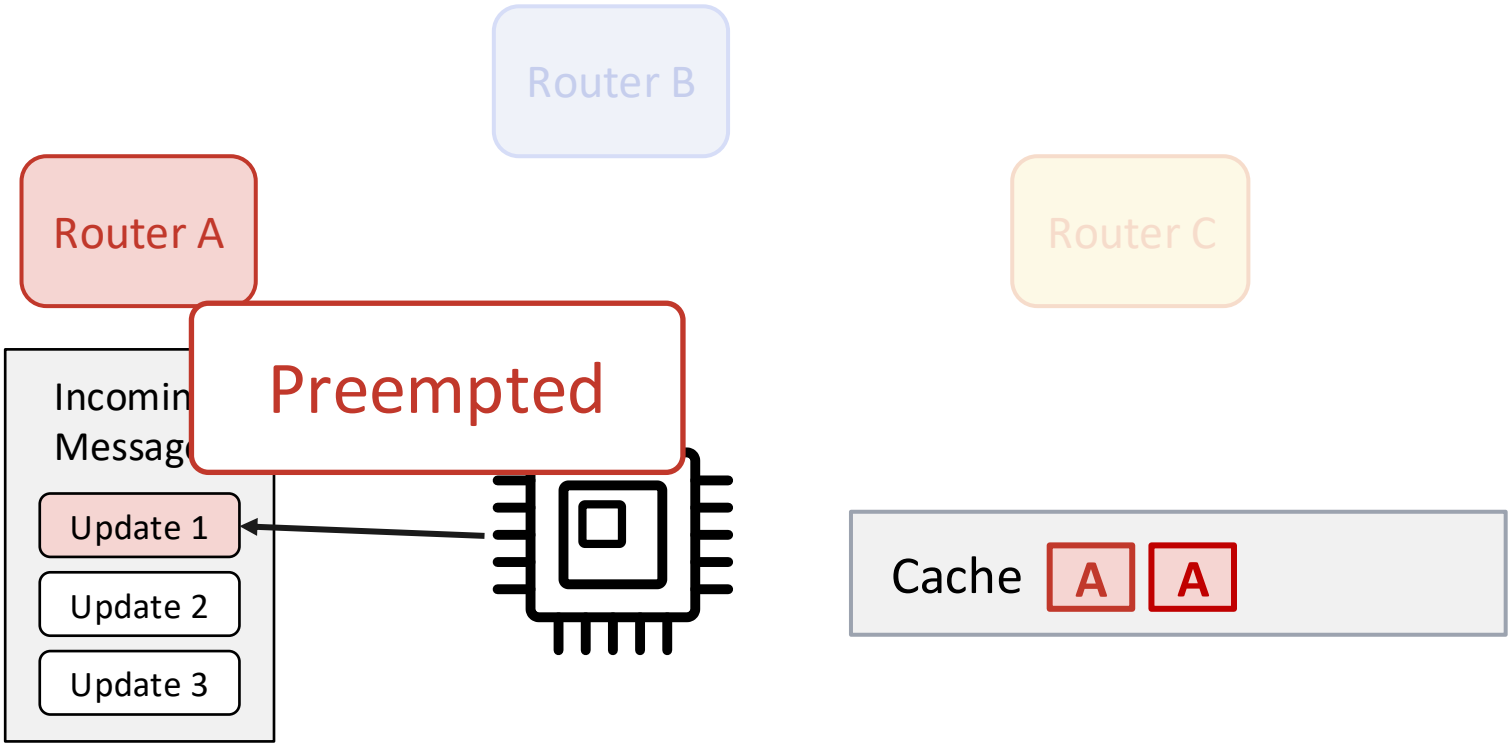
Cost 2: Context Switch



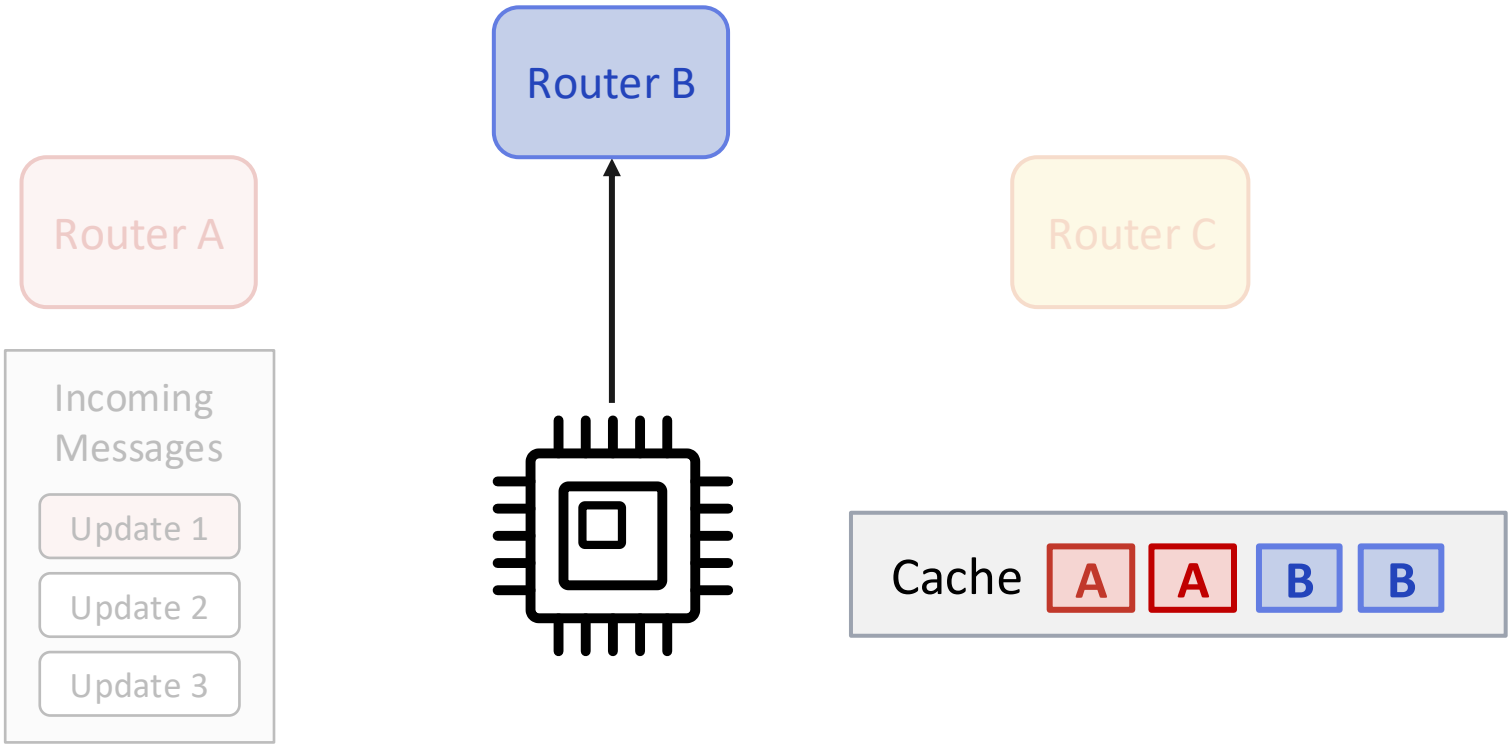
Cost 2: Context Switch



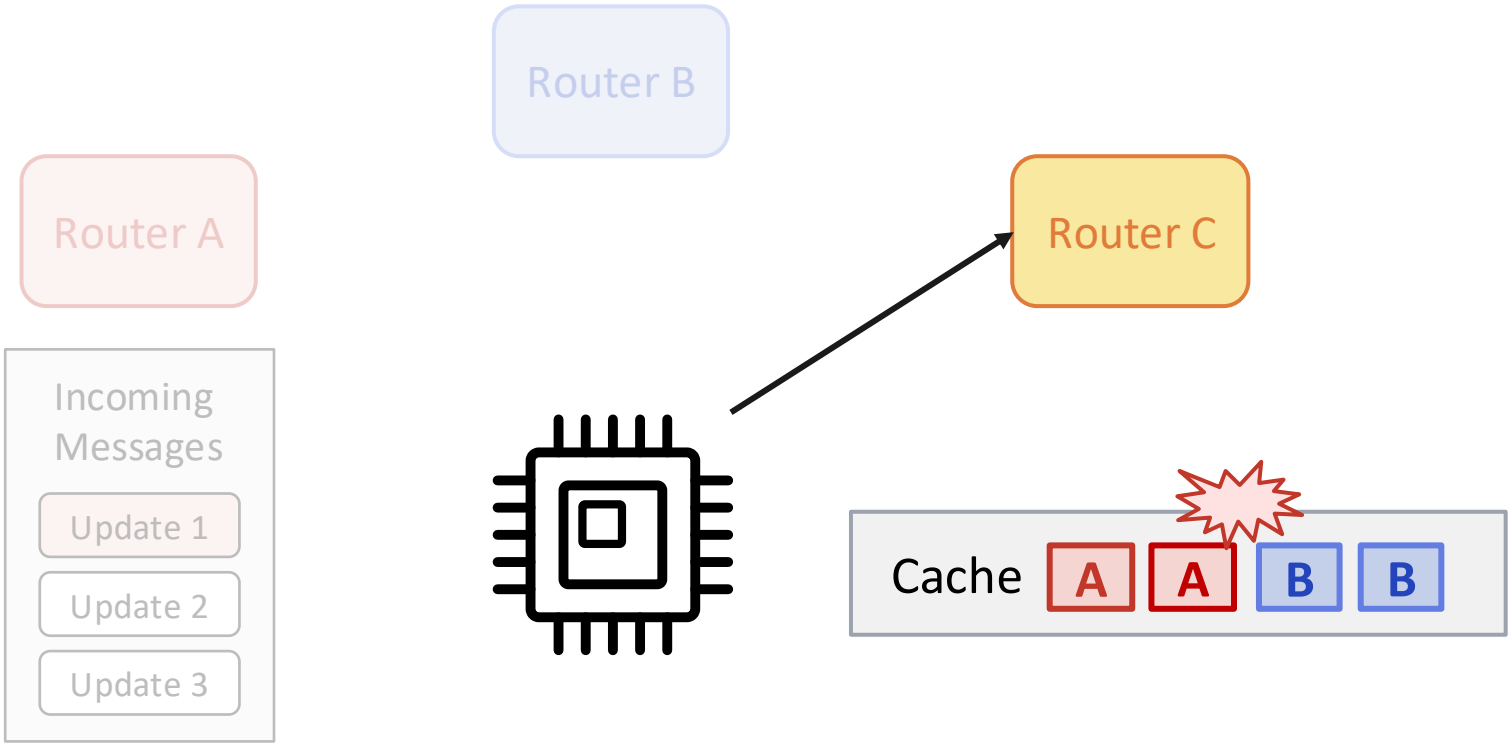
Cost 2: Context Switch



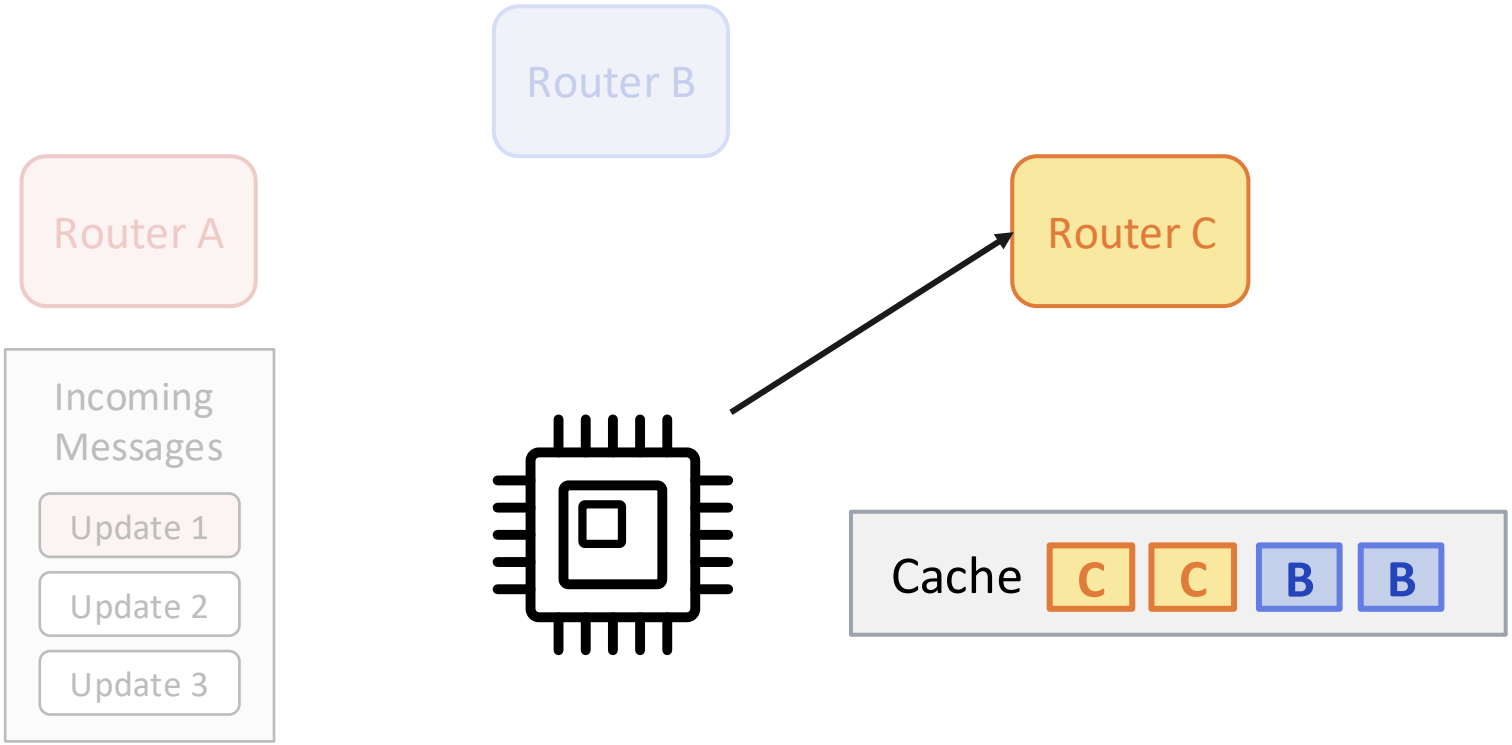
Cost 2: Context Switch



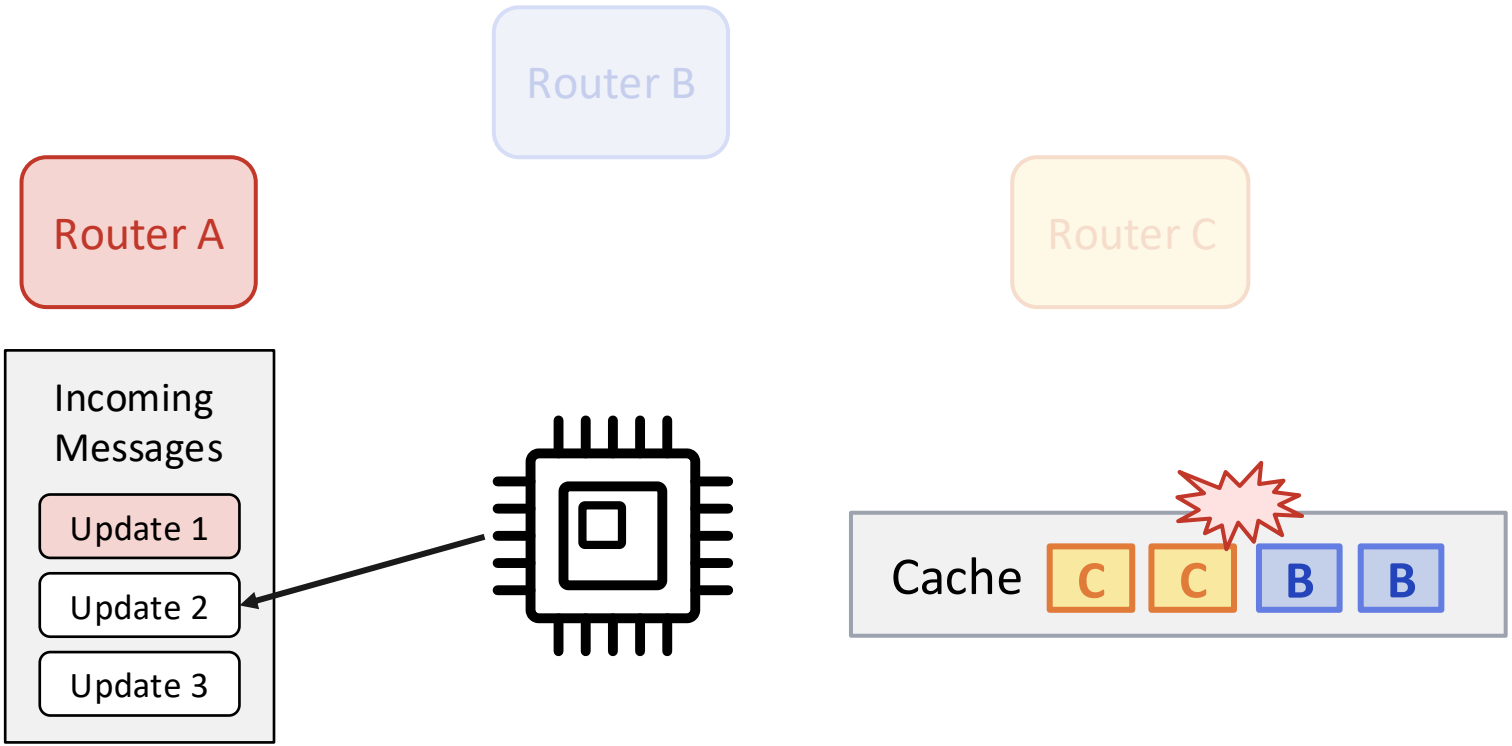
Cost 2: Context Switch



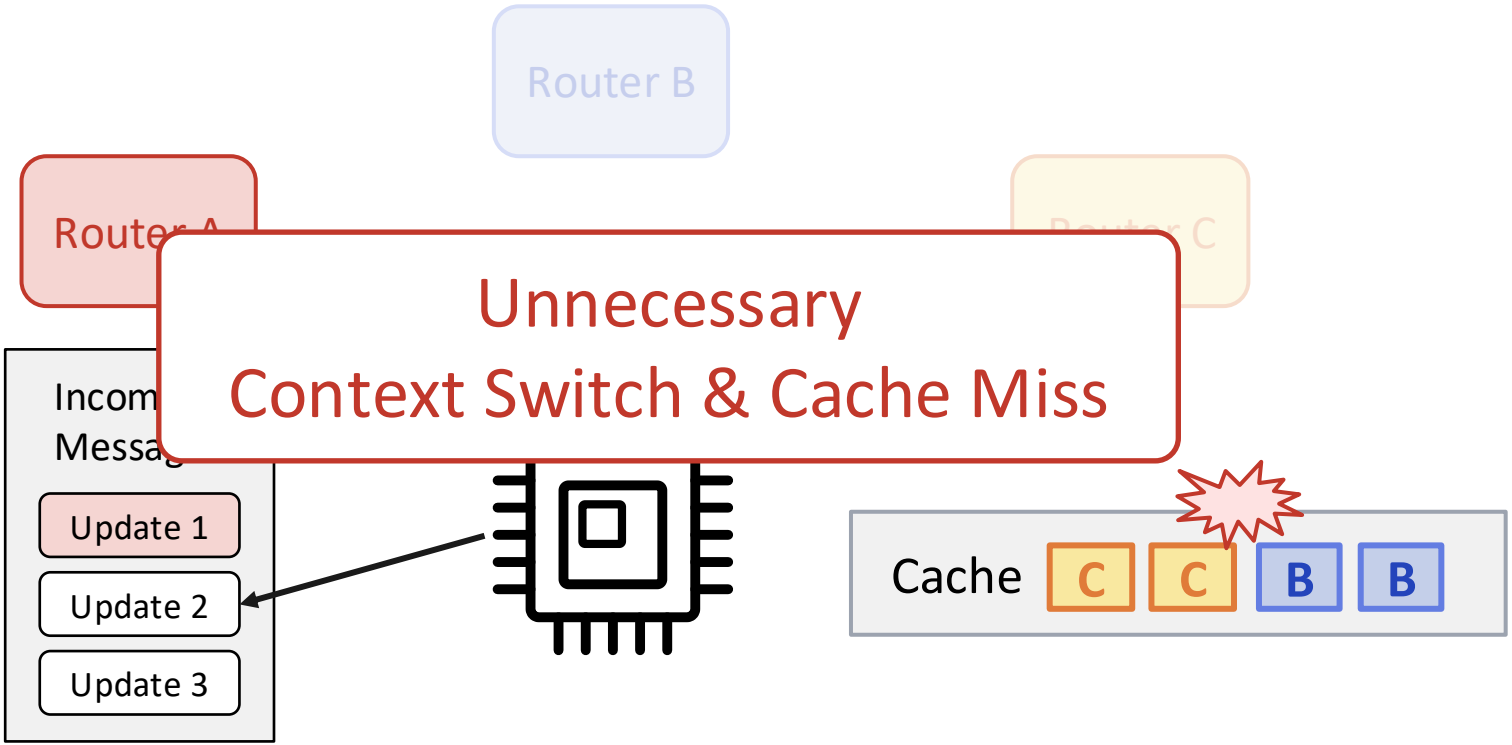
Cost 2: Context Switch



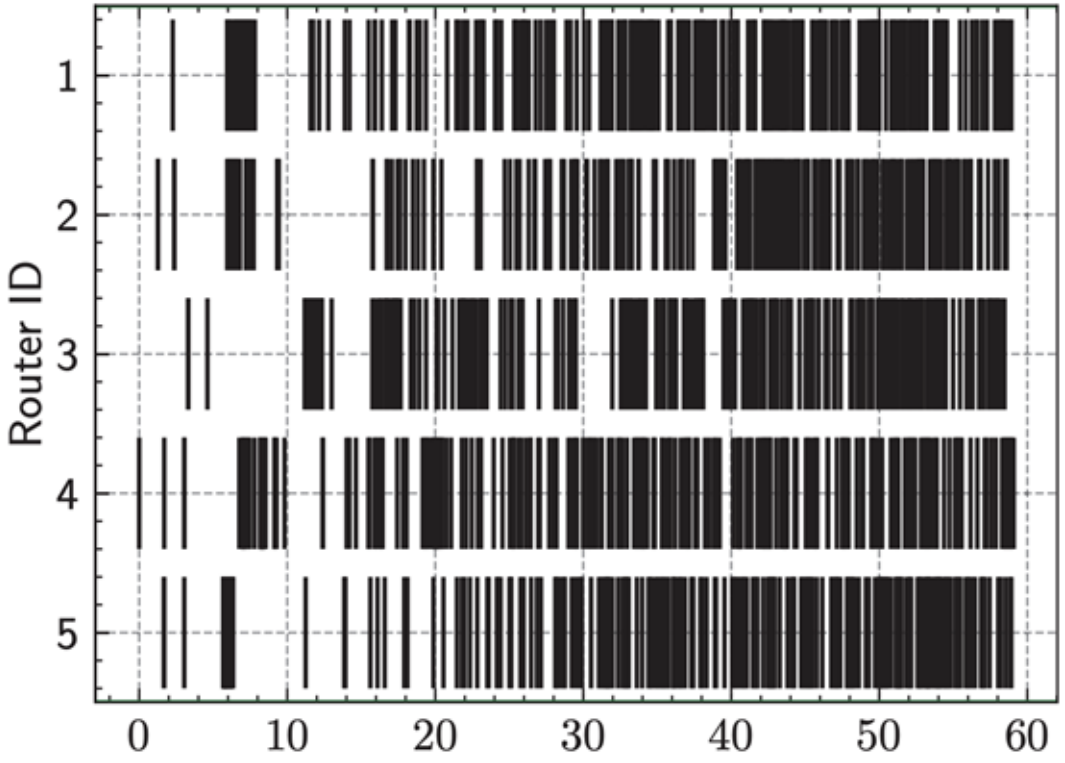
Cost 2: Context Switch



Cost 2: Context Switch



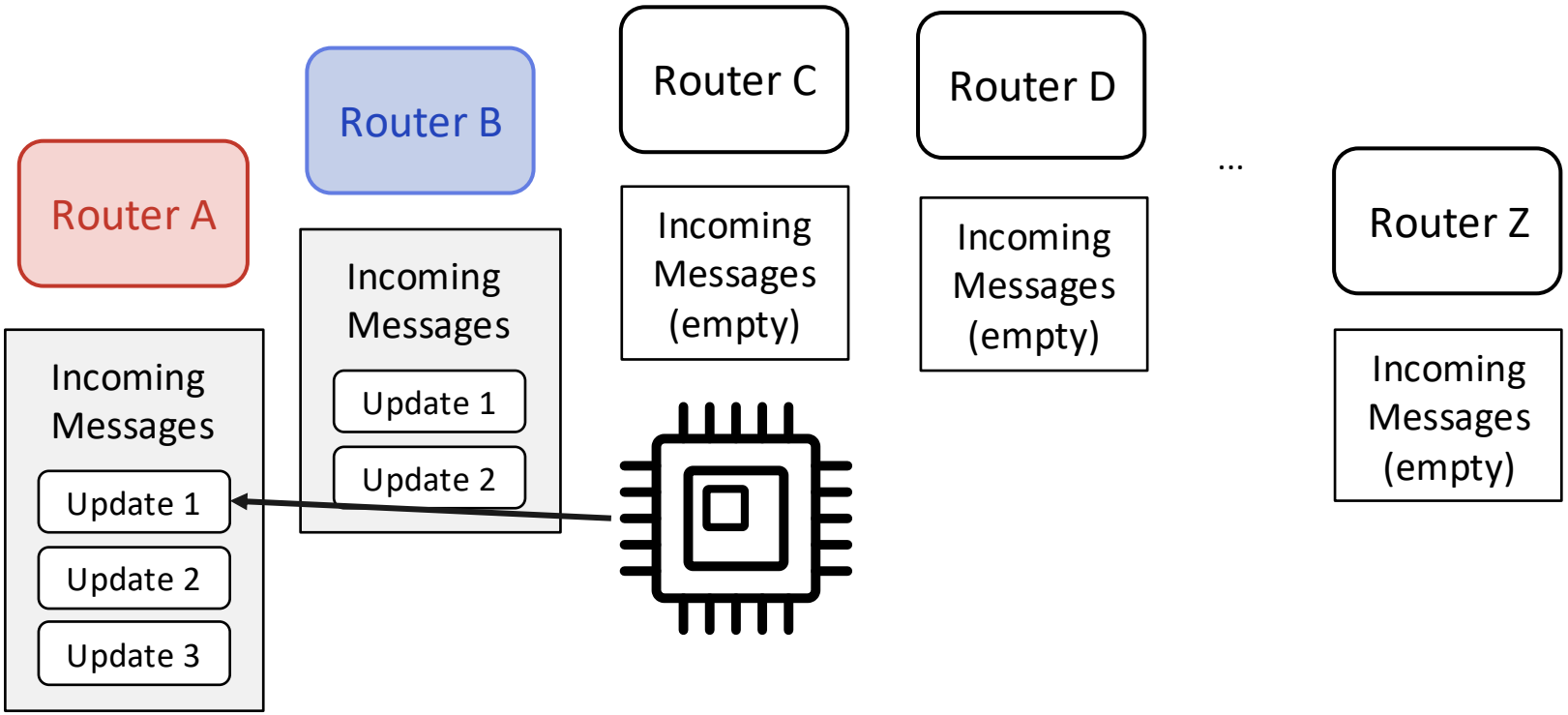
Cost 2: Context Switch



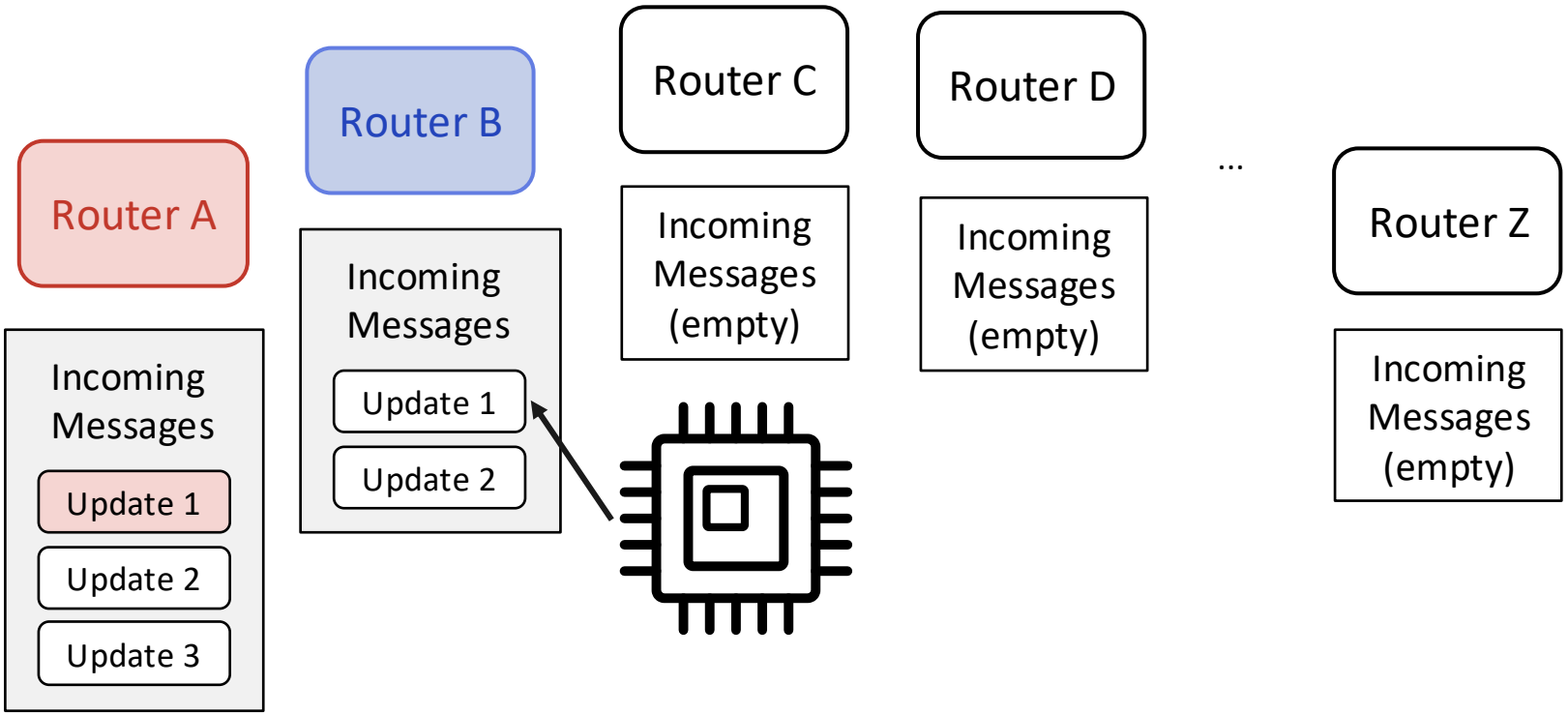
Cost 2: Context Switch

Frequent Context Switch & Cache Miss

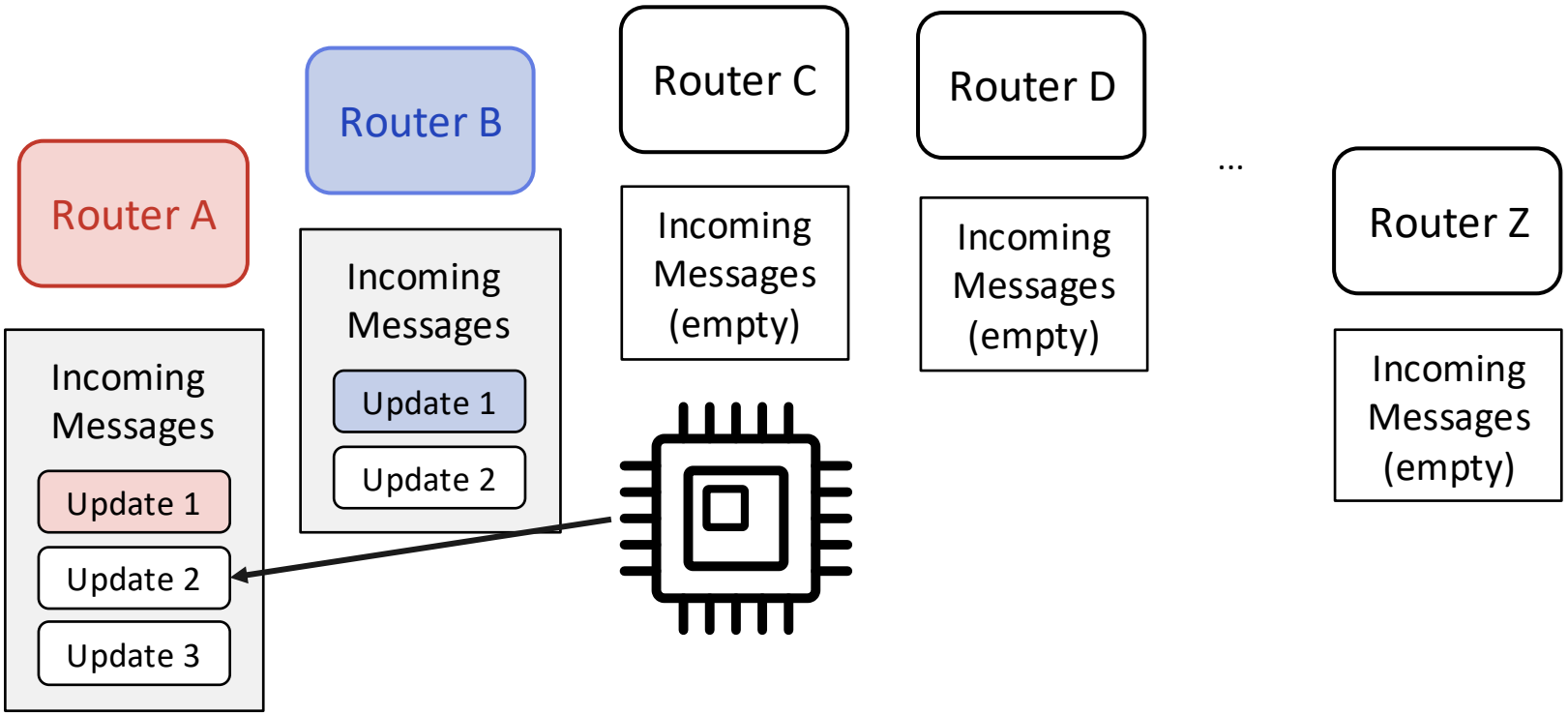
Cost 3: Memory Residency



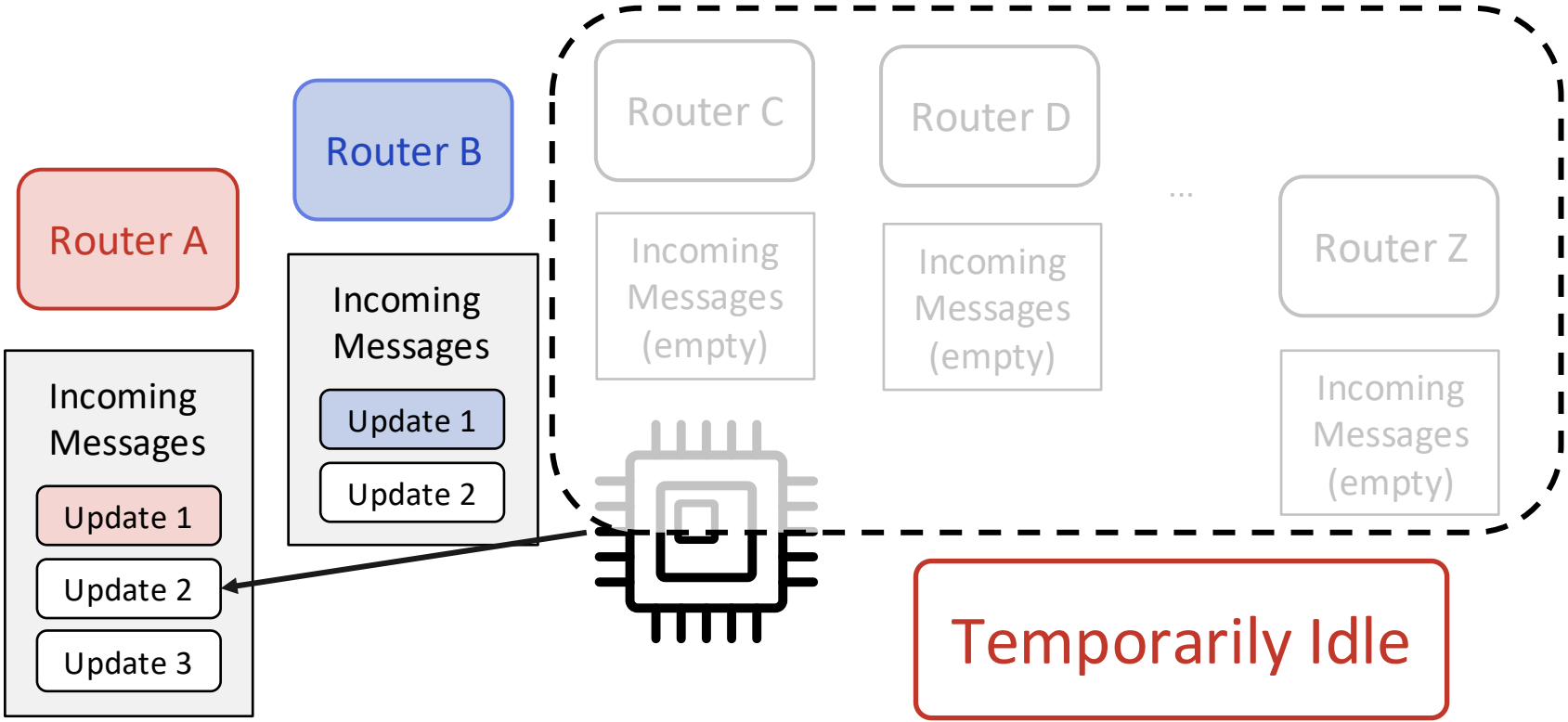
Cost 3: Memory Residency



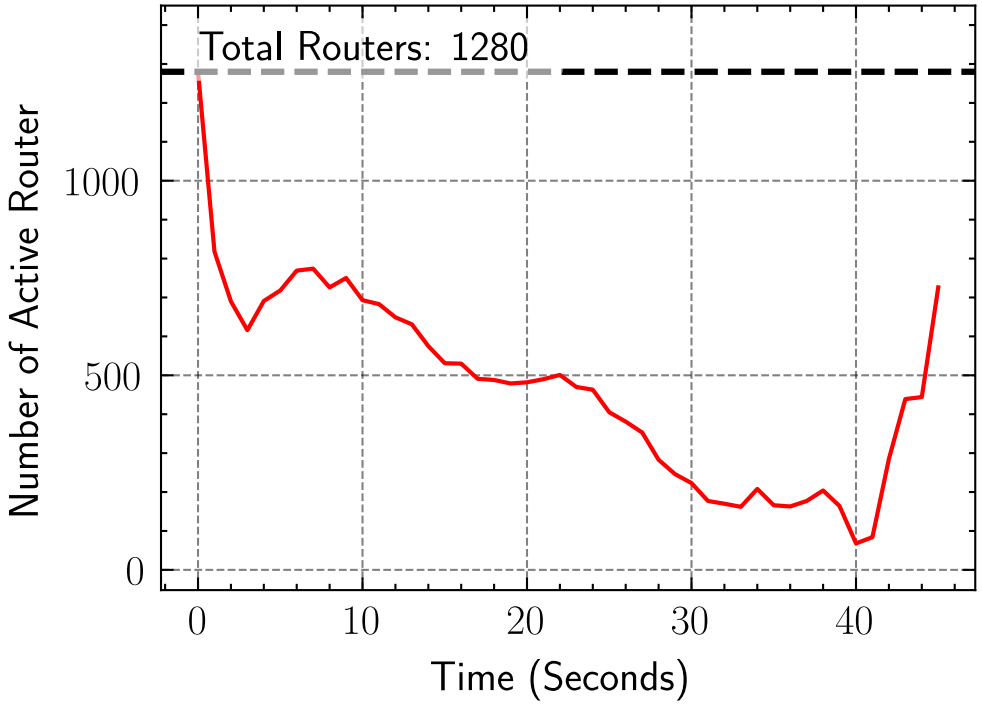
Cost 3: Memory Residency



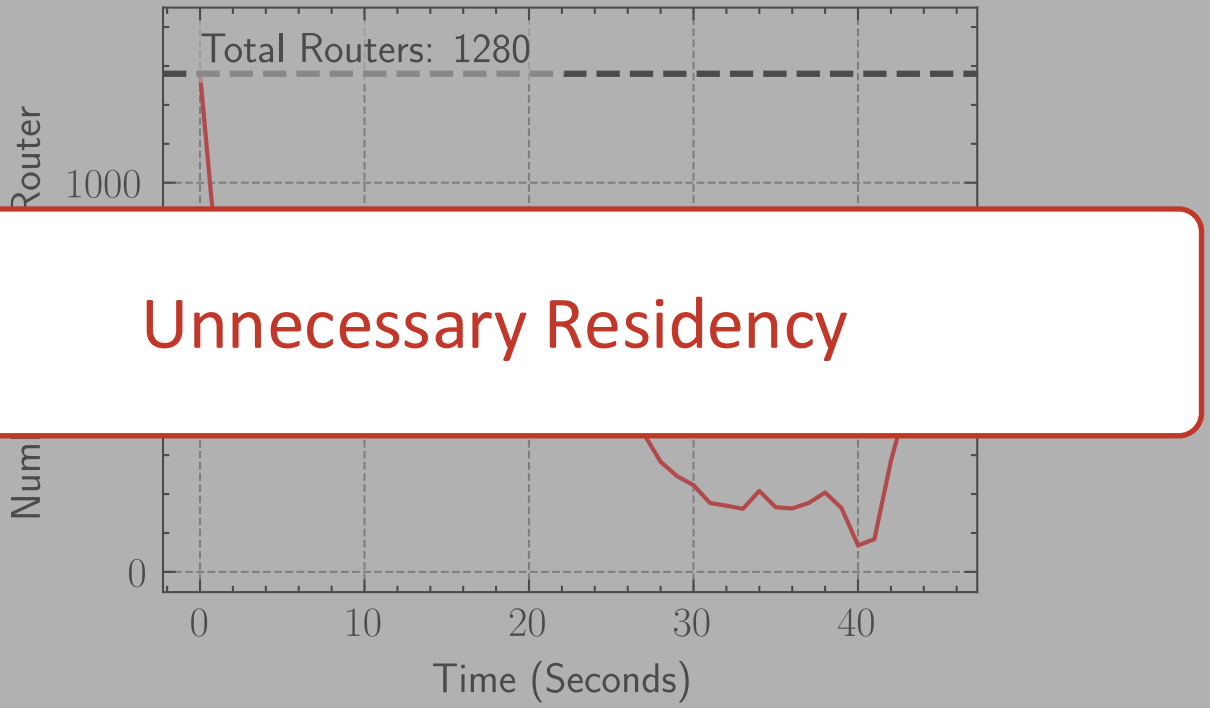
Cost 3: Memory Residency



Cost 3: Memory Residency

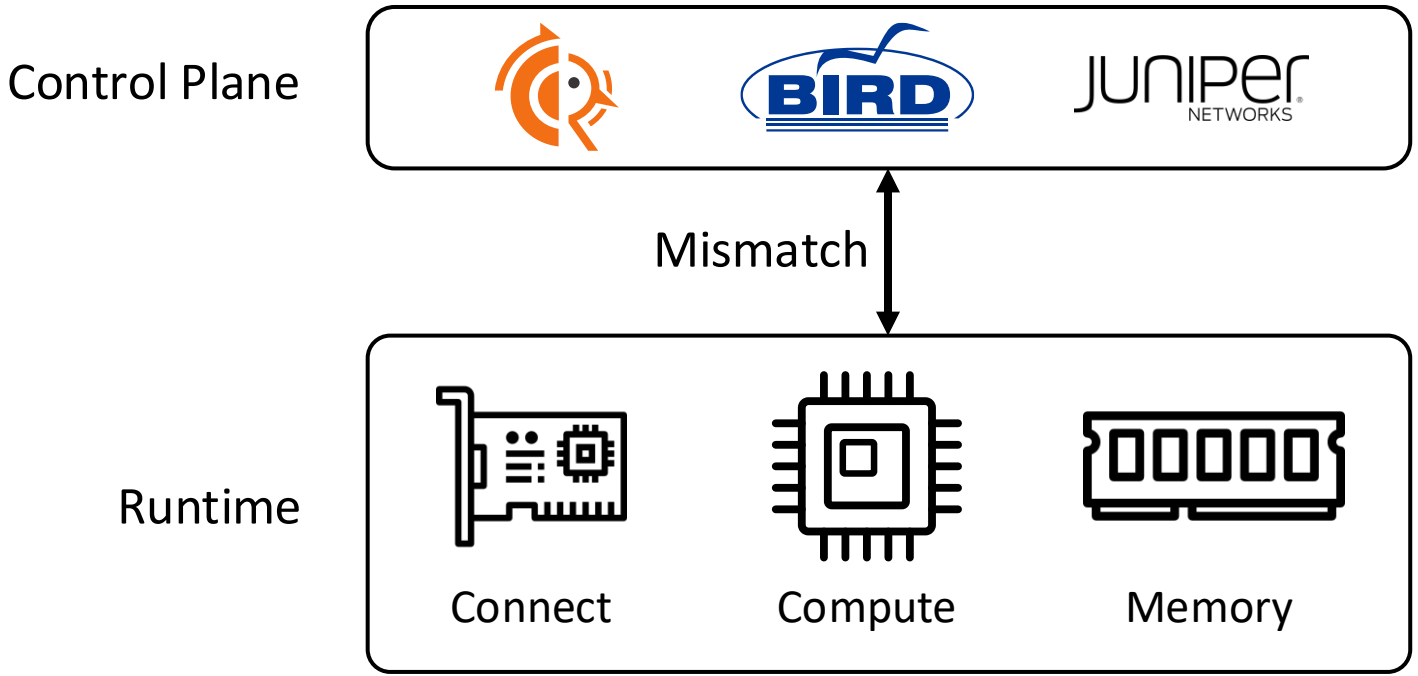


Cost 3: Memory Residency

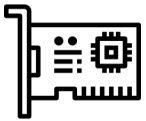


Root Cause: Runtime does not Match Control Plane Needs

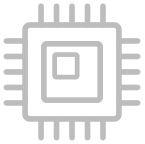
Root Cause: Runtime does not Match Control Plane Needs



Root Cause: Runtime does not Match Control Plane Needs



Connect

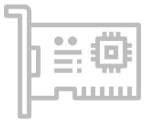


Compute

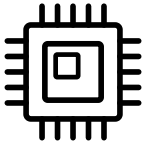
Memory

Runtime Mechanism	Built for	Control Plane Needs
Virtual Ethernet	Real L2 packets	**Message delivery**

Root Cause: Runtime does not Match Control Plane Needs



Connect

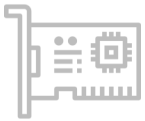


Compute

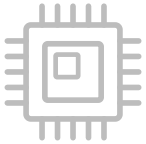
Memory

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?

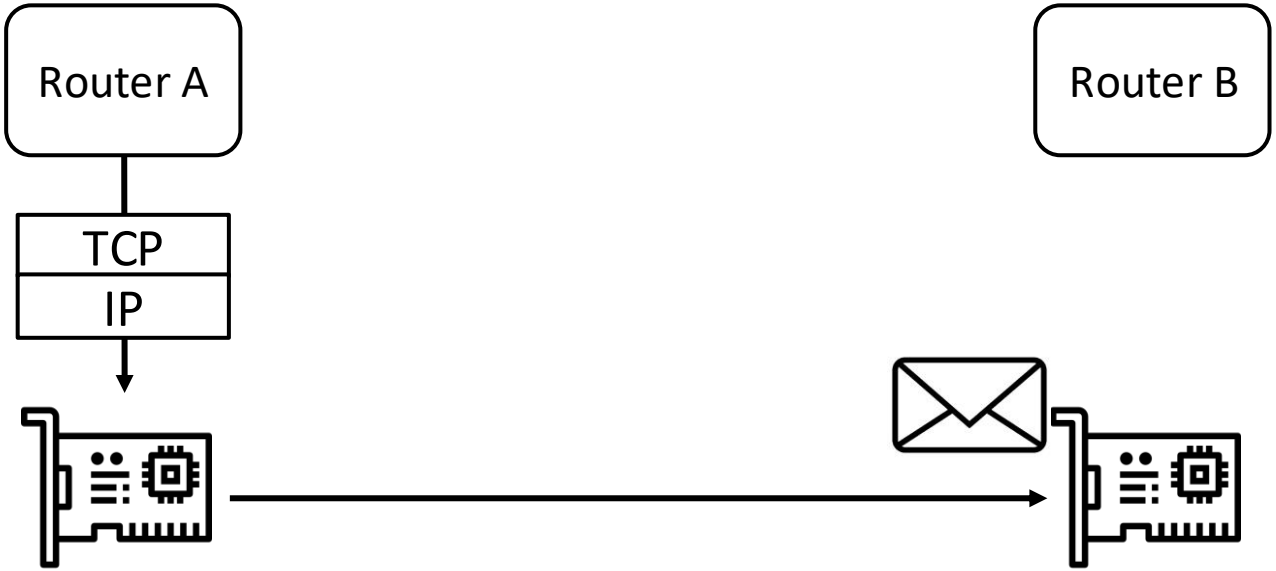
Message-Level Delivery: Eliminate Network Buildup

Normal Workflow

Message-Level Delivery: Eliminate Network Buildup

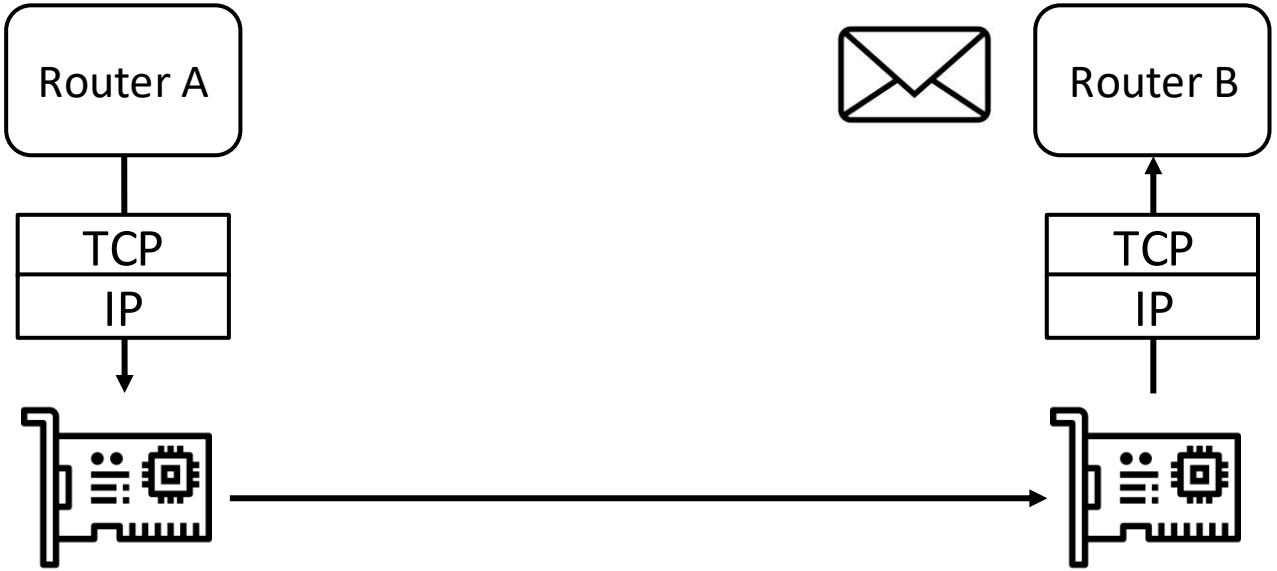
Normal Workflow

Message-Level Delivery: Eliminate Network Buildup



Normal Workflow

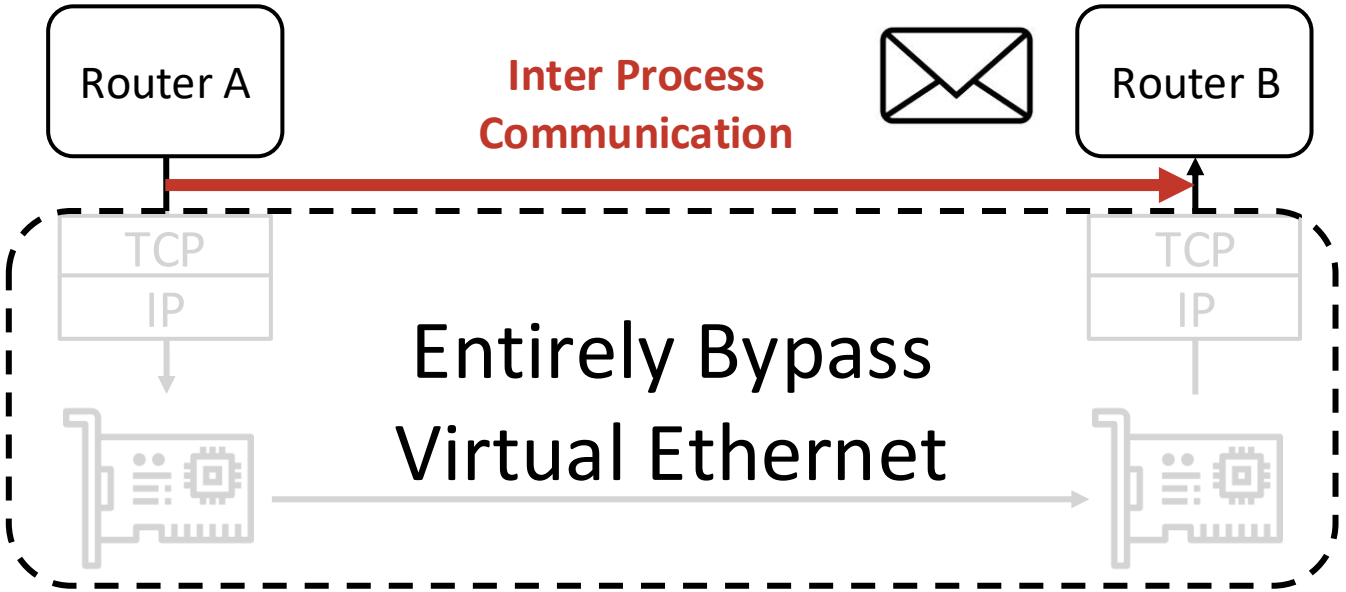
Message-Level Delivery: Eliminate Network Buildup



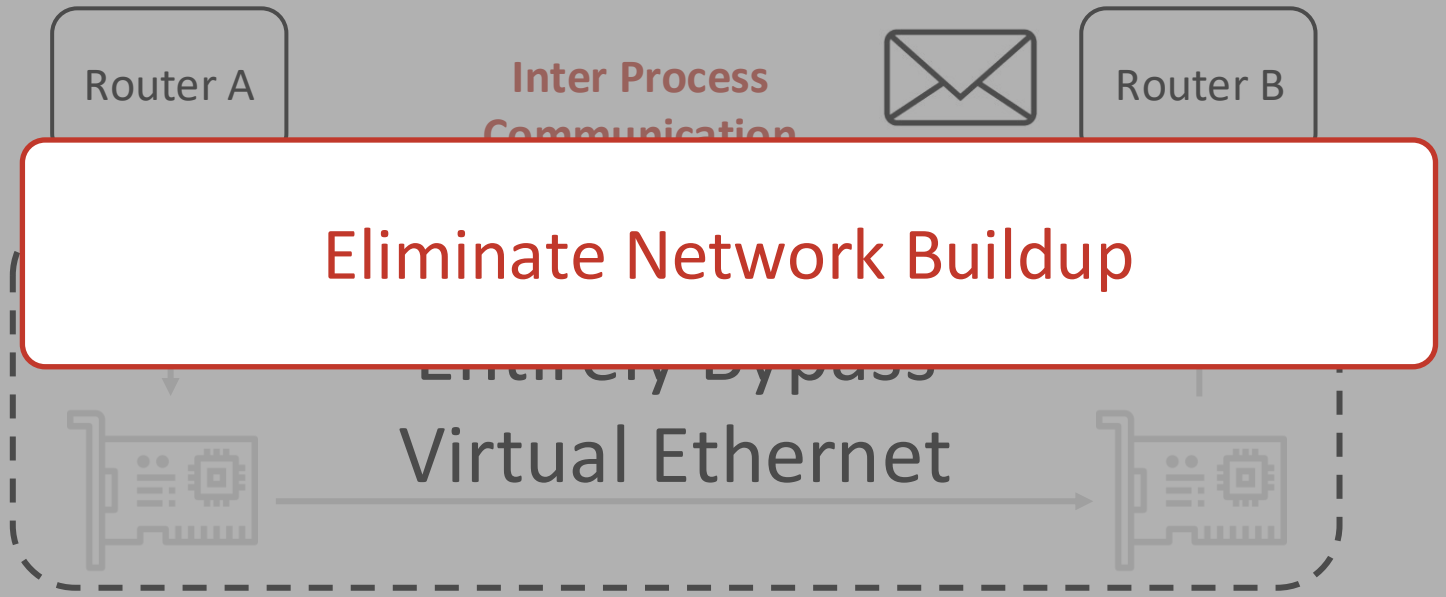
Normal Workflow

Message-Level Delivery: Eliminate Network Buildup

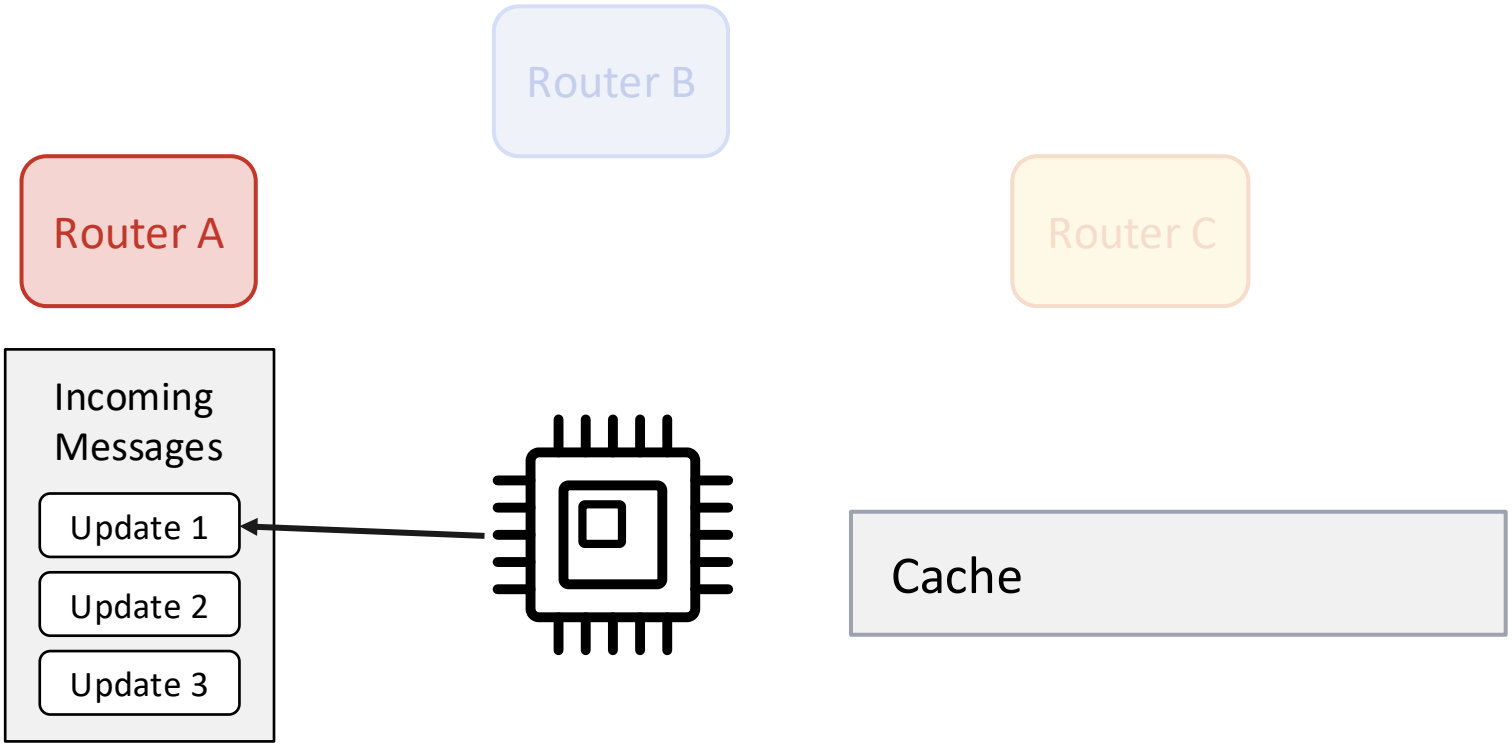
Message-Level Delivery: Eliminate Network Buildup



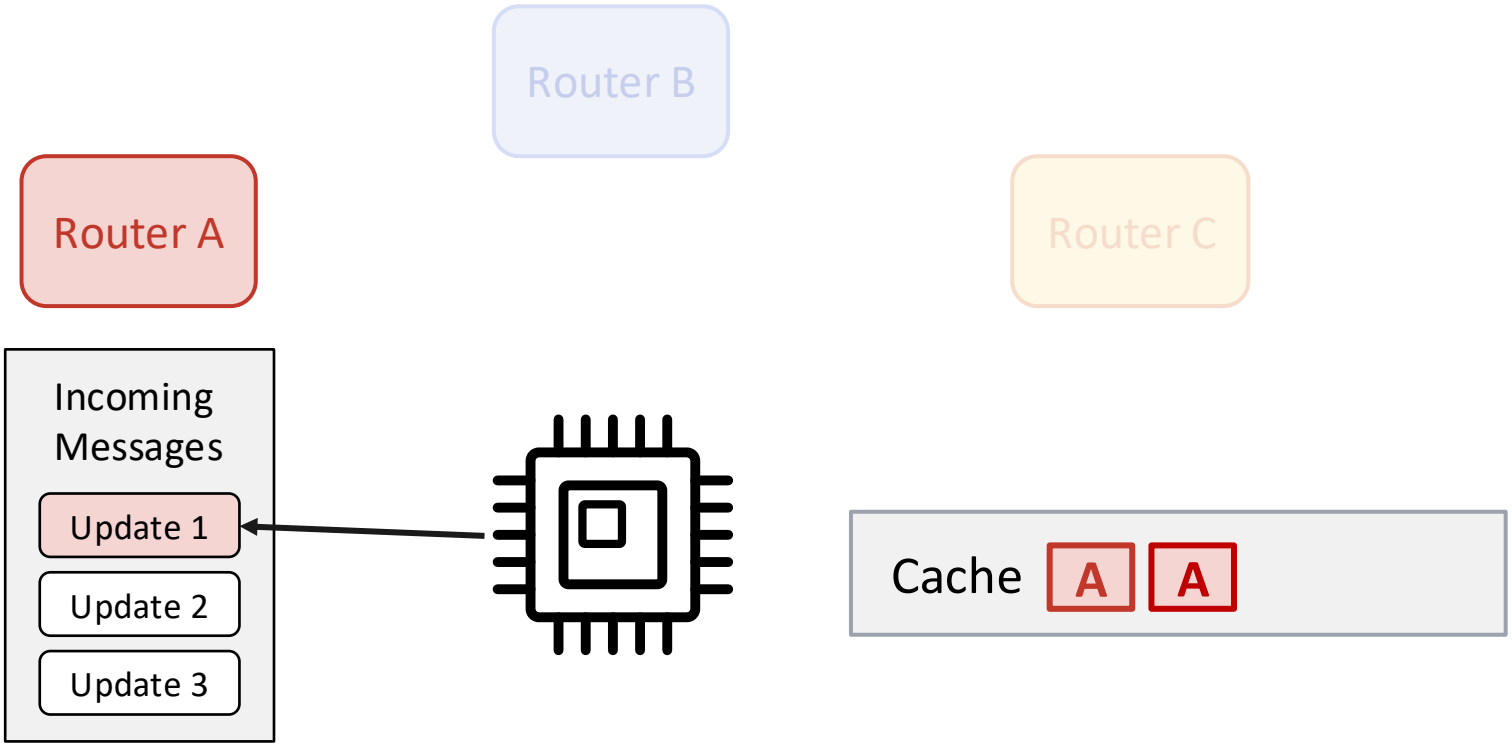
Message-Level Delivery: Eliminate Network Buildup



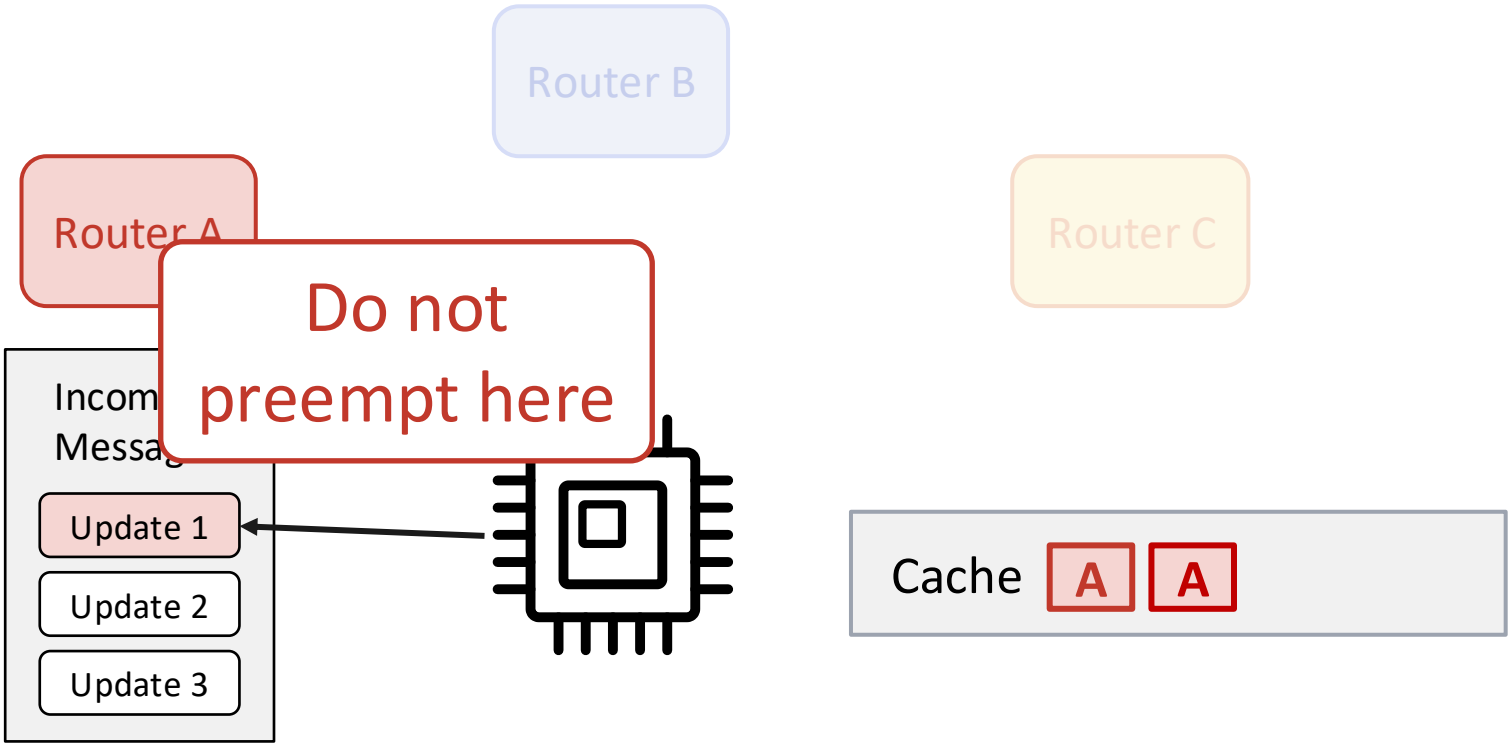
Run-to-Idle Scheduling: Avoid Unnecessary Cache Miss



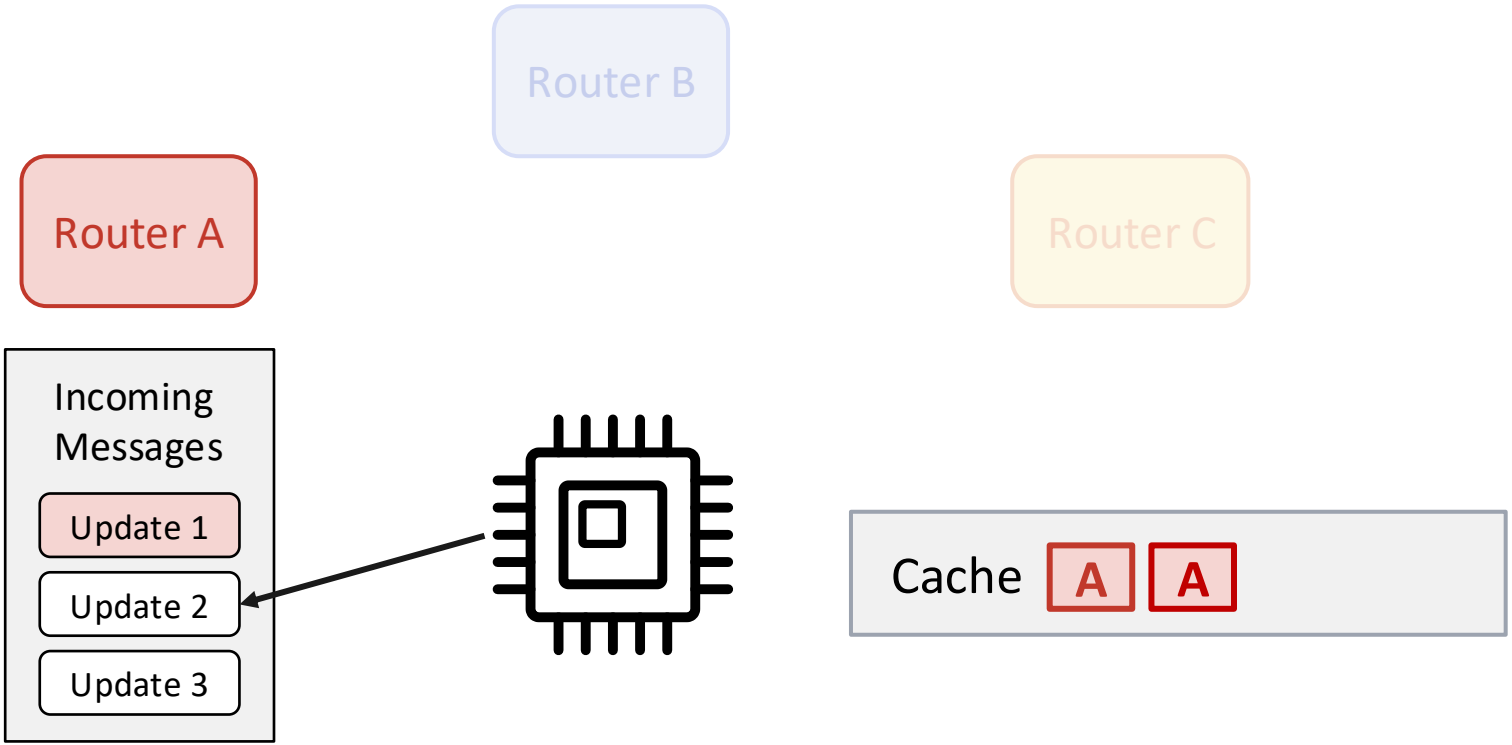
Run-to-Idle Scheduling: Avoid Unnecessary Cache Miss



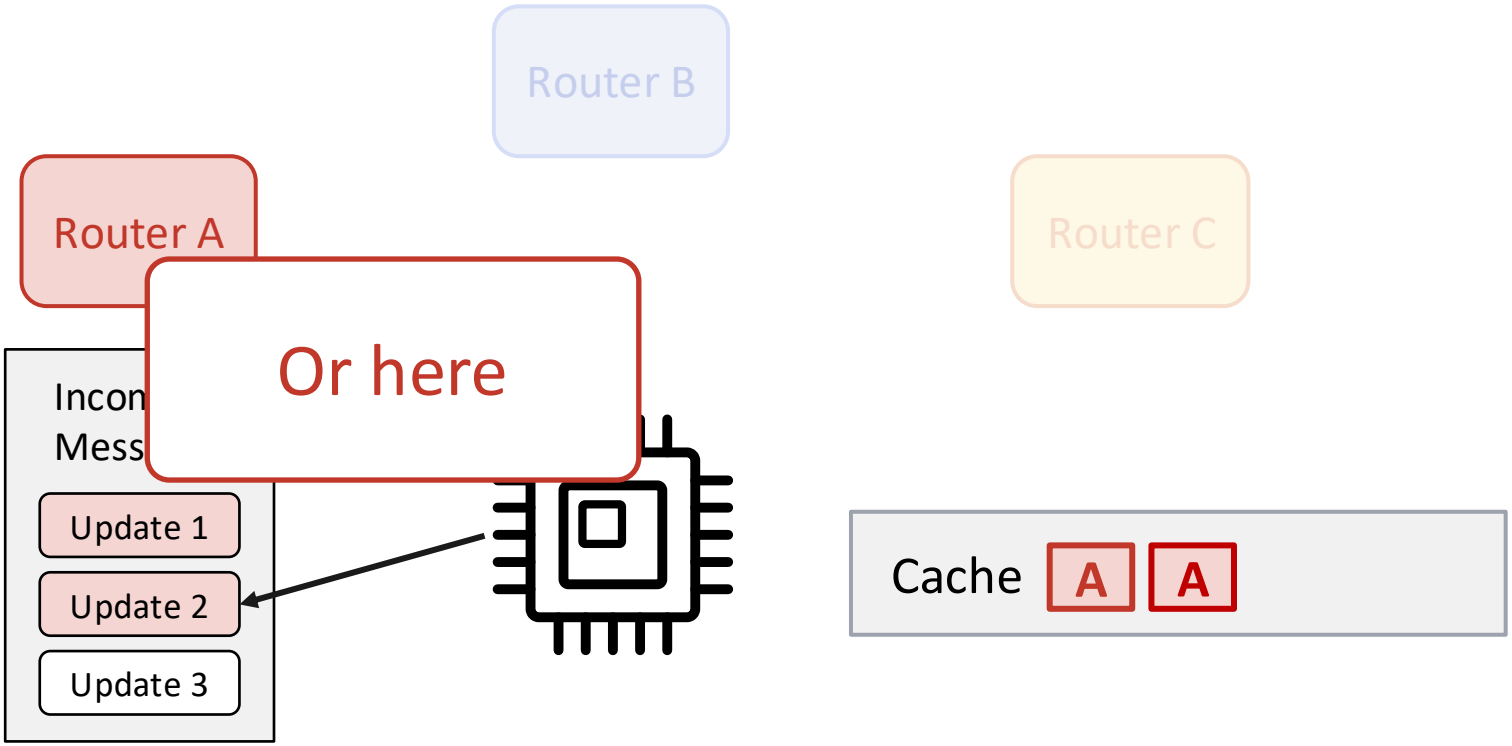
Run-to-Idle Scheduling: Avoid Unnecessary Cache Miss



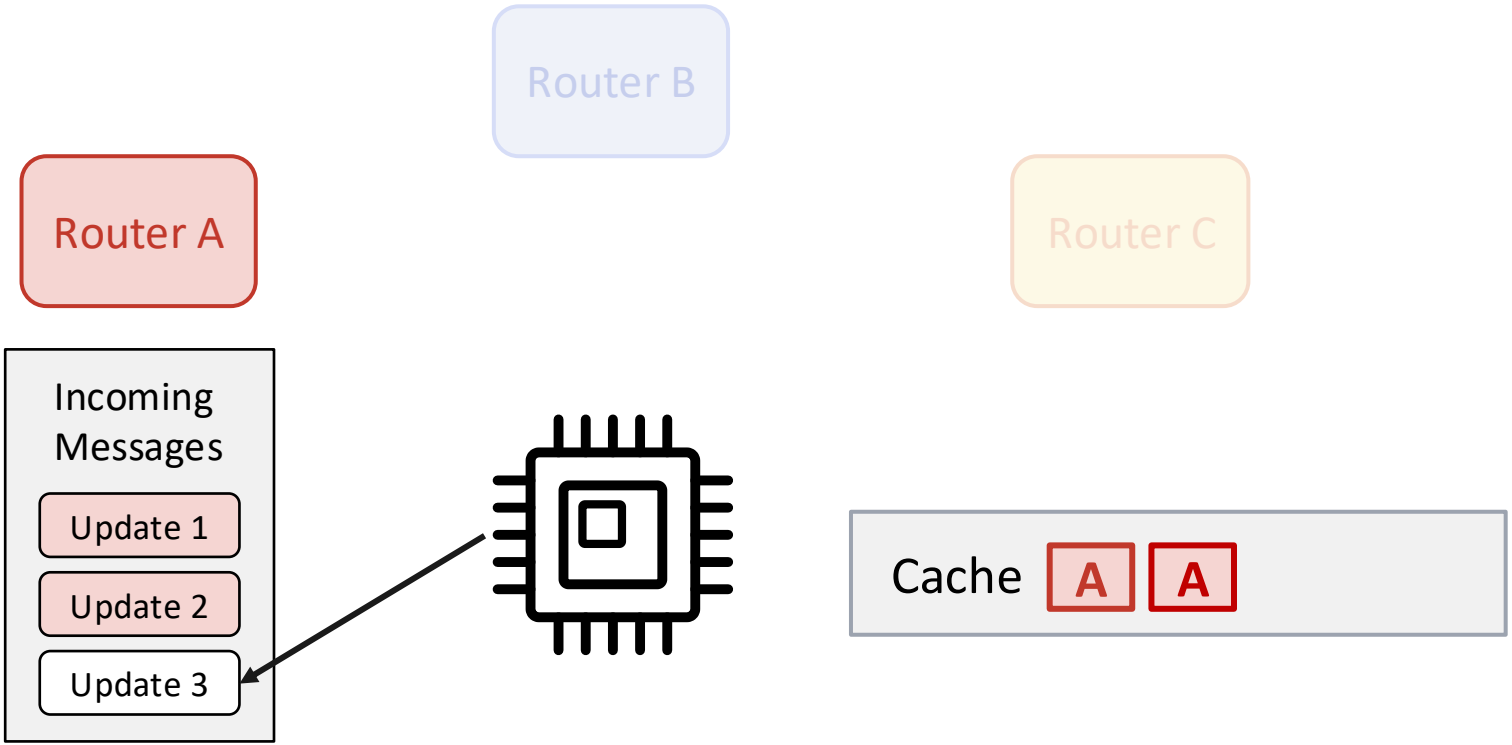
Run-to-Idle Scheduling: Avoid Unnecessary Cache Miss



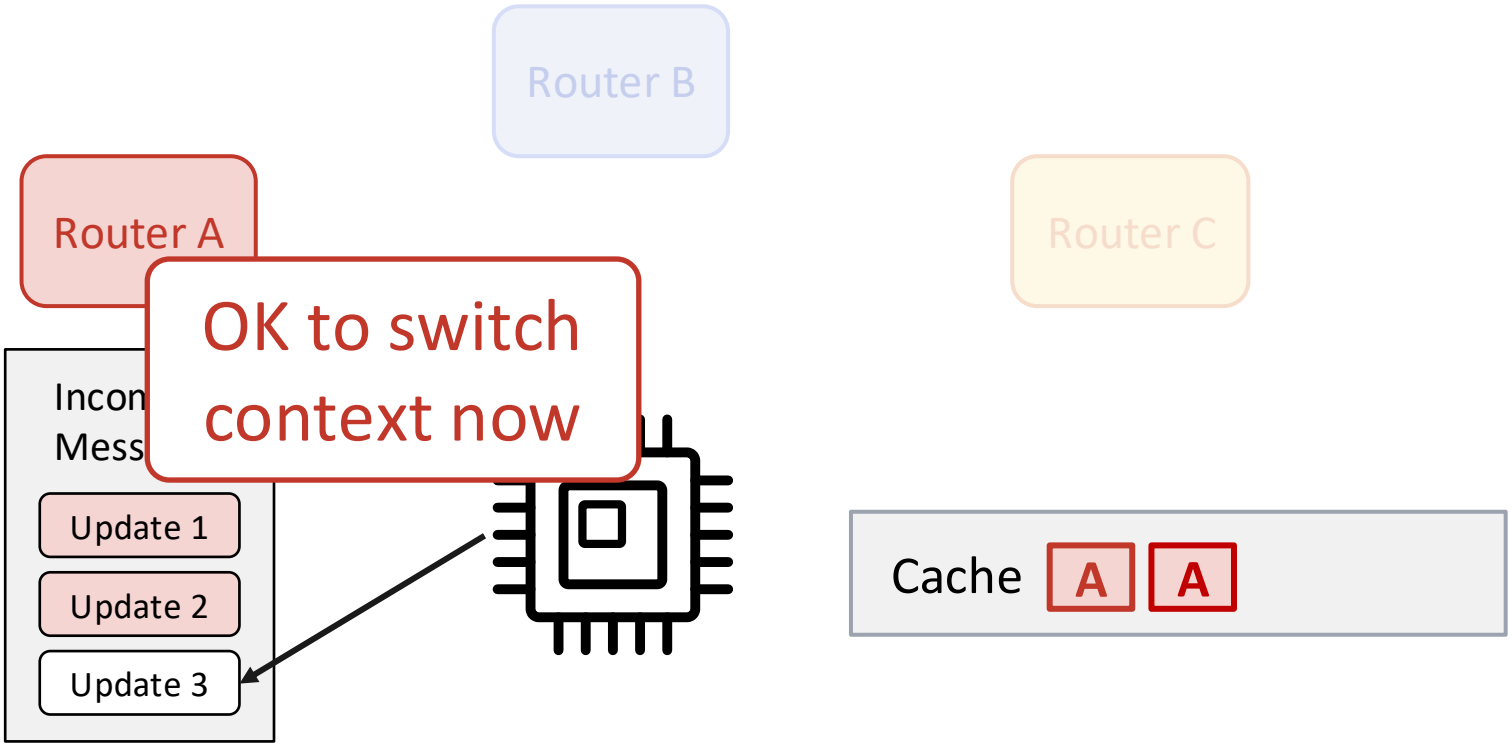
Run-to-Idle Scheduling: Avoid Unnecessary Cache Miss



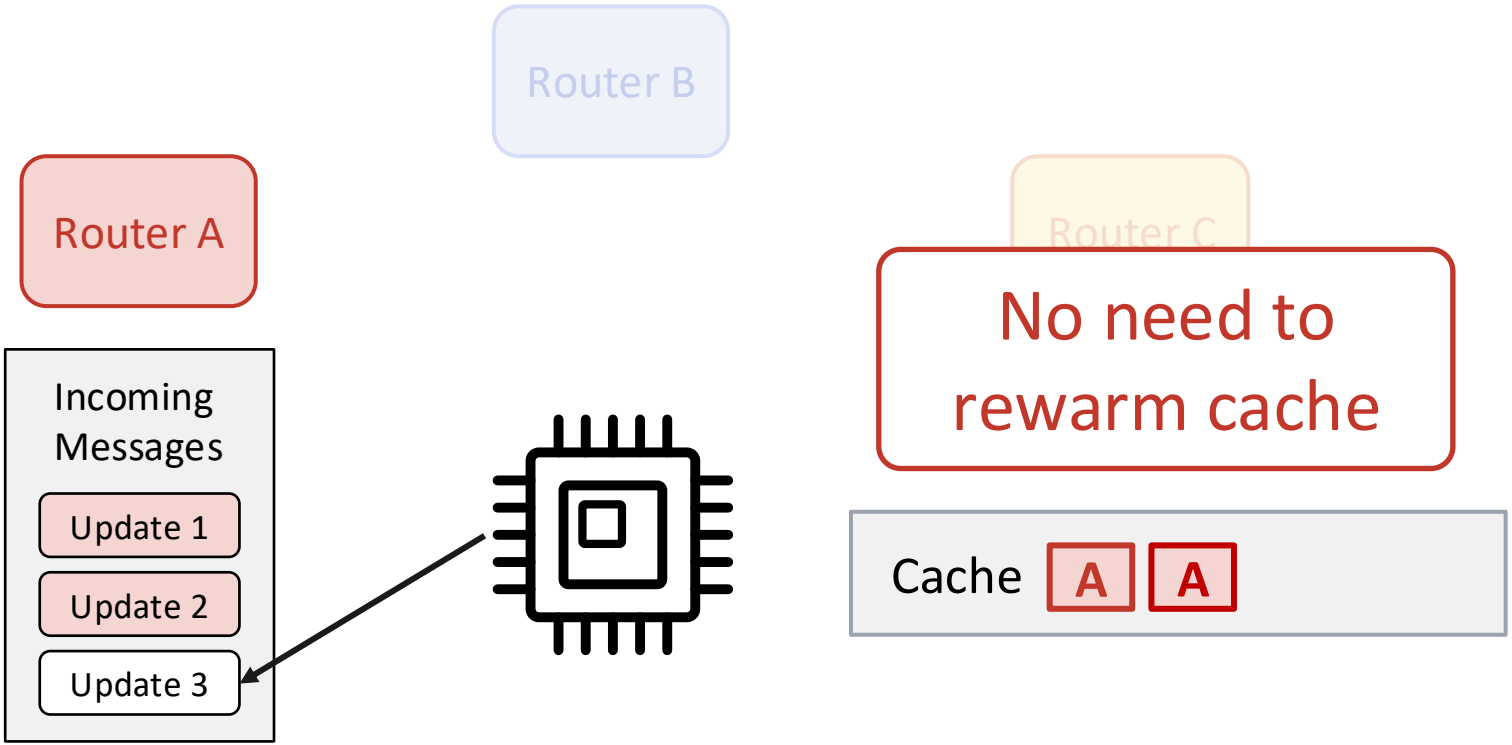
Run-to-Idle Scheduling: Avoid Unnecessary Cache Miss



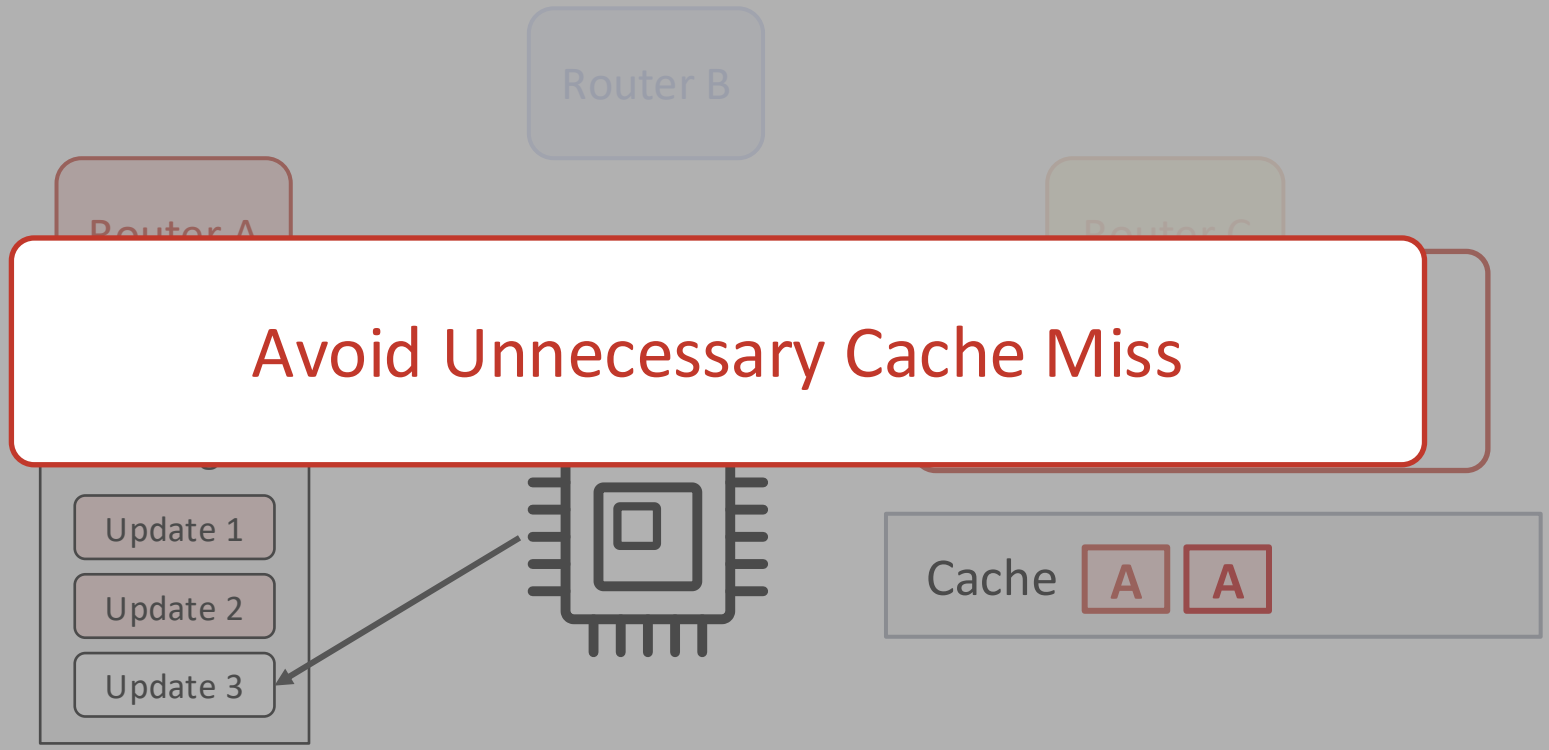
Run-to-Idle Scheduling: Avoid Unnecessary Cache Miss



Run-to-Idle Scheduling: Avoid Unnecessary Cache Miss

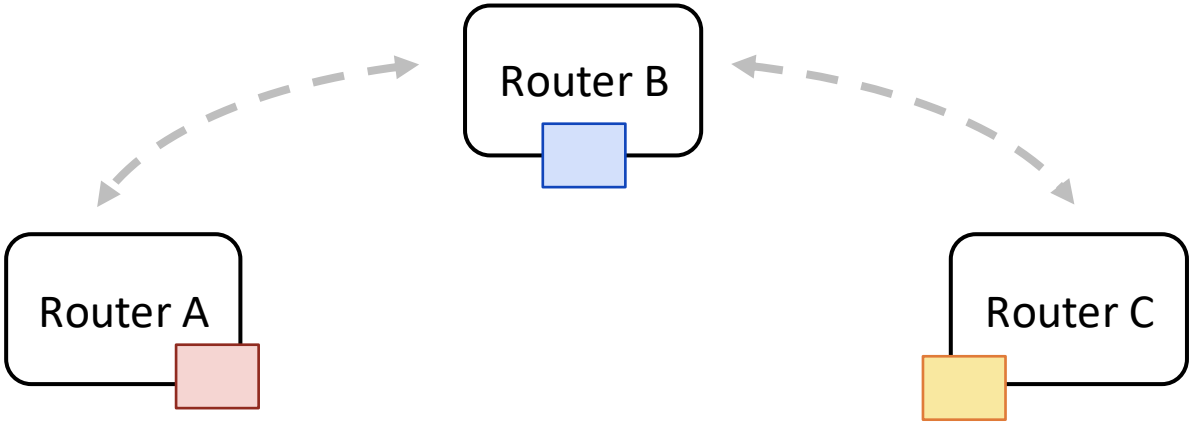


Run-to-Idle Scheduling: Avoid Unnecessary Cache Miss

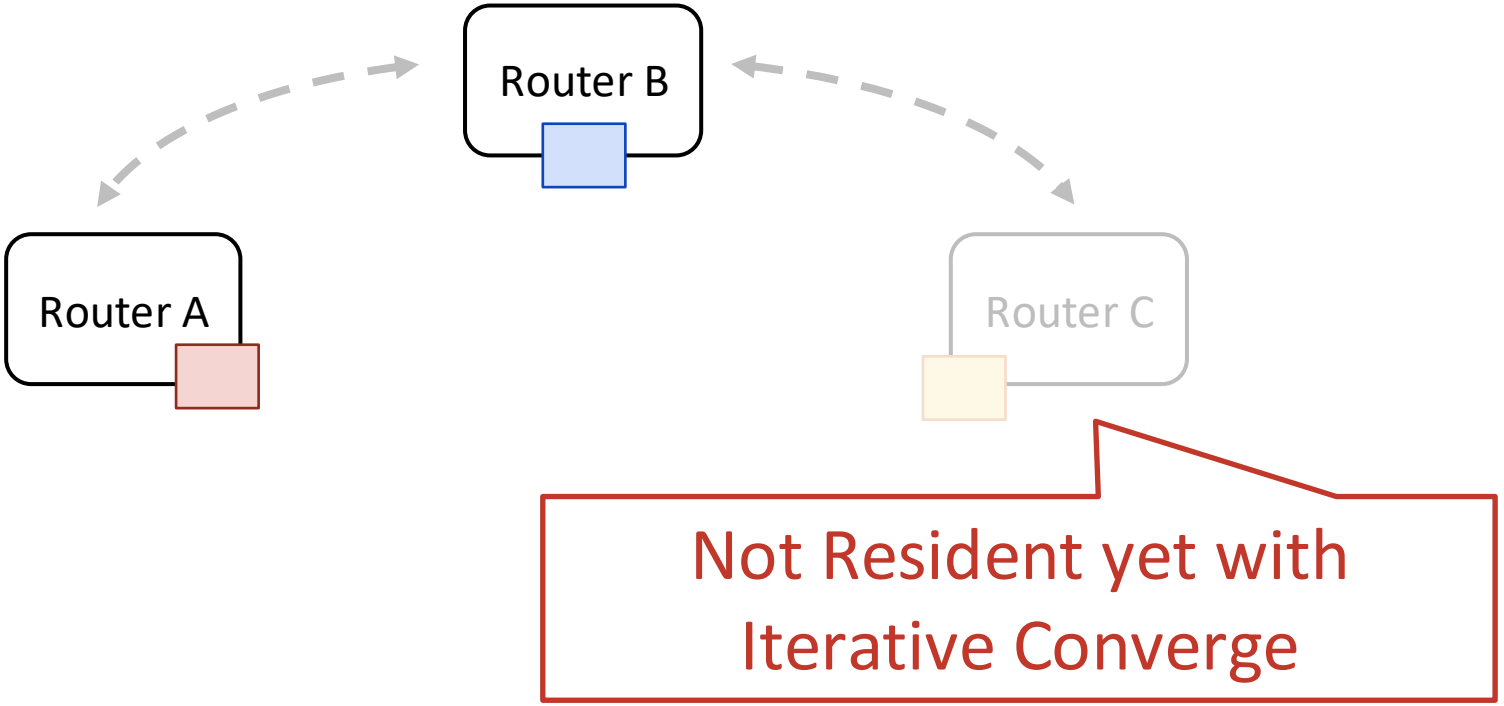


Iterative Converge: Reduce Memory Residency

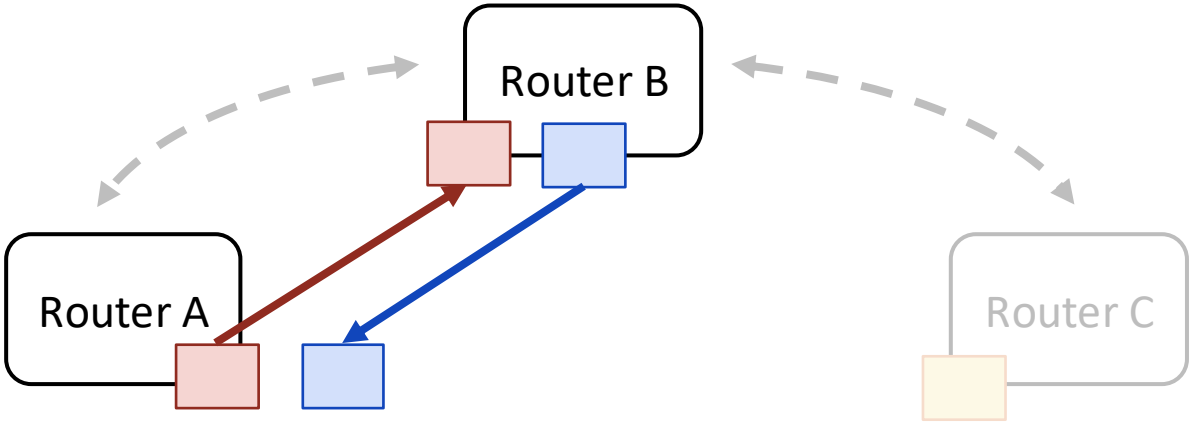
Iterative Converge: Reduce Memory Residency



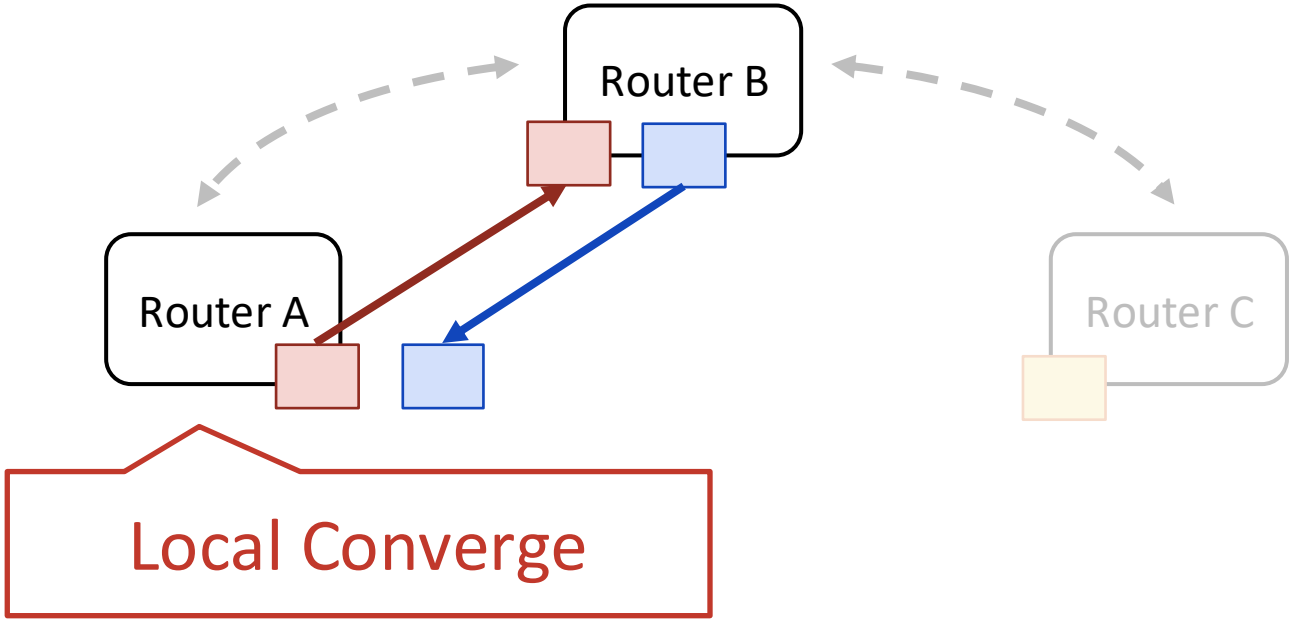
Iterative Converge: Reduce Memory Residency



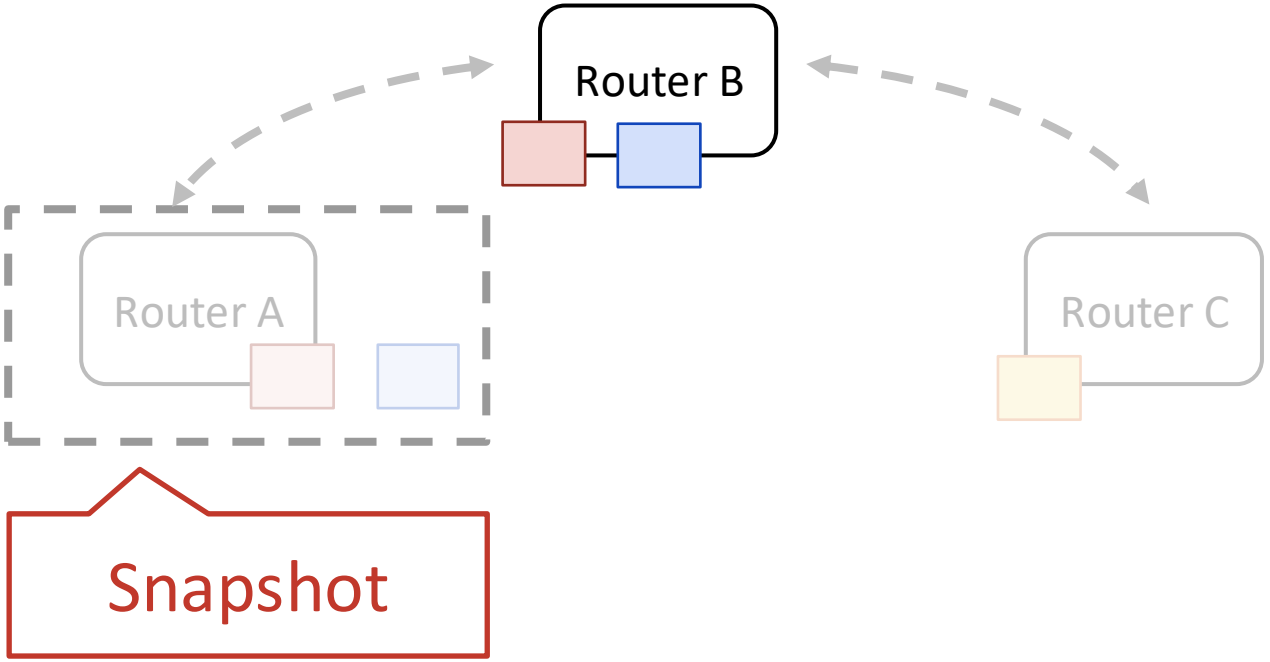
Iterative Converge: Reduce Memory Residency



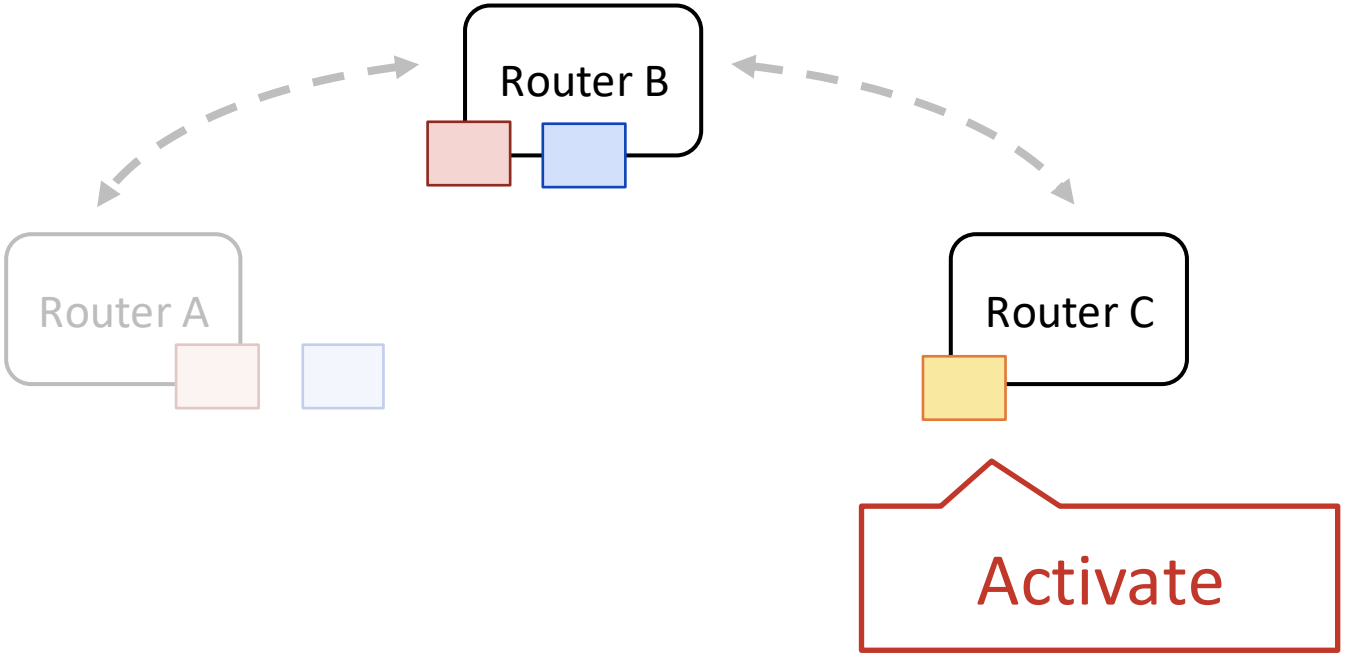
Iterative Converge: Reduce Memory Residency



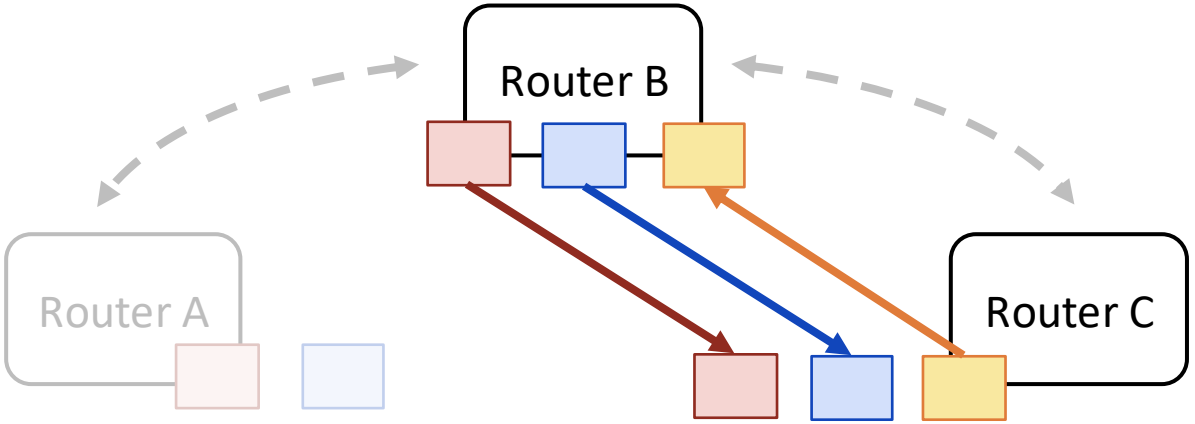
Iterative Converge: Reduce Memory Residency



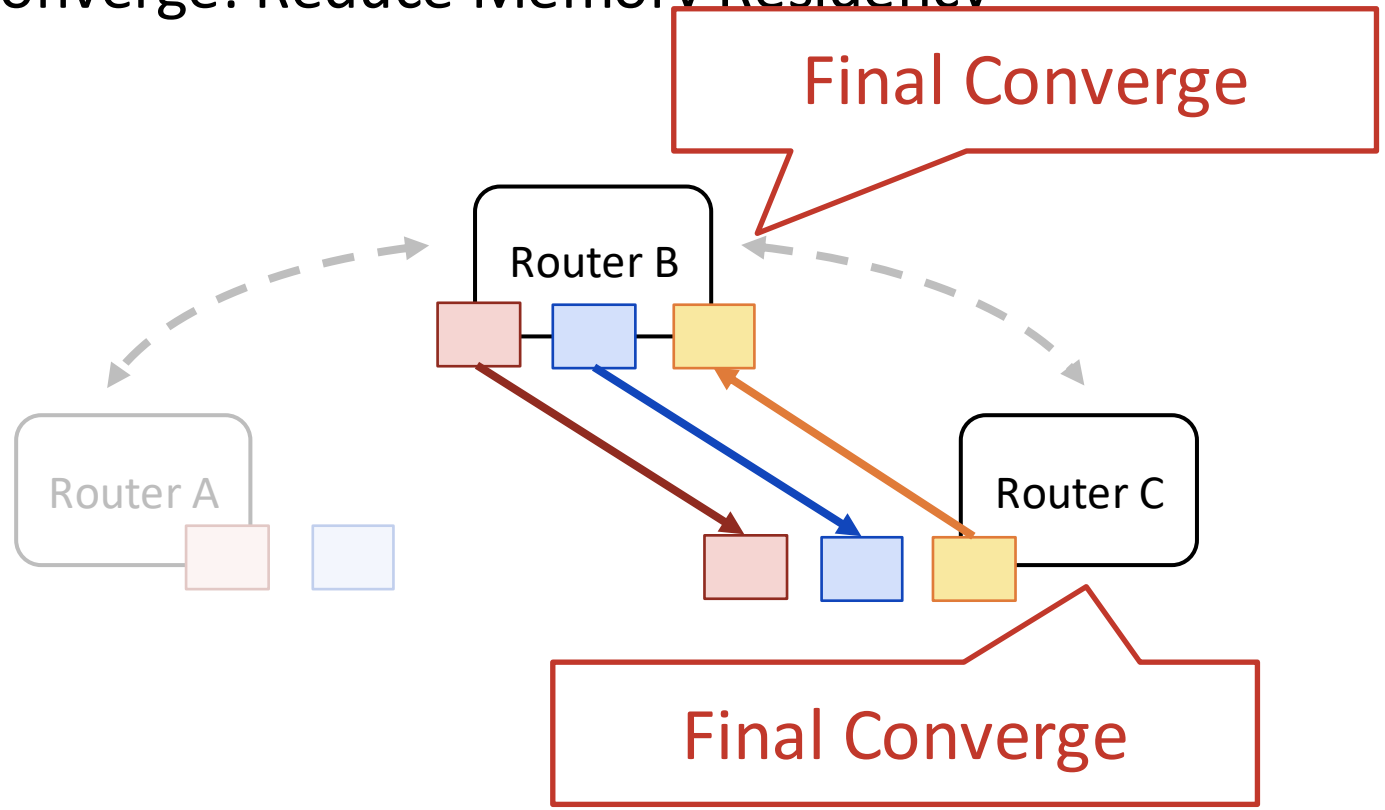
Iterative Converge: Reduce Memory Residency



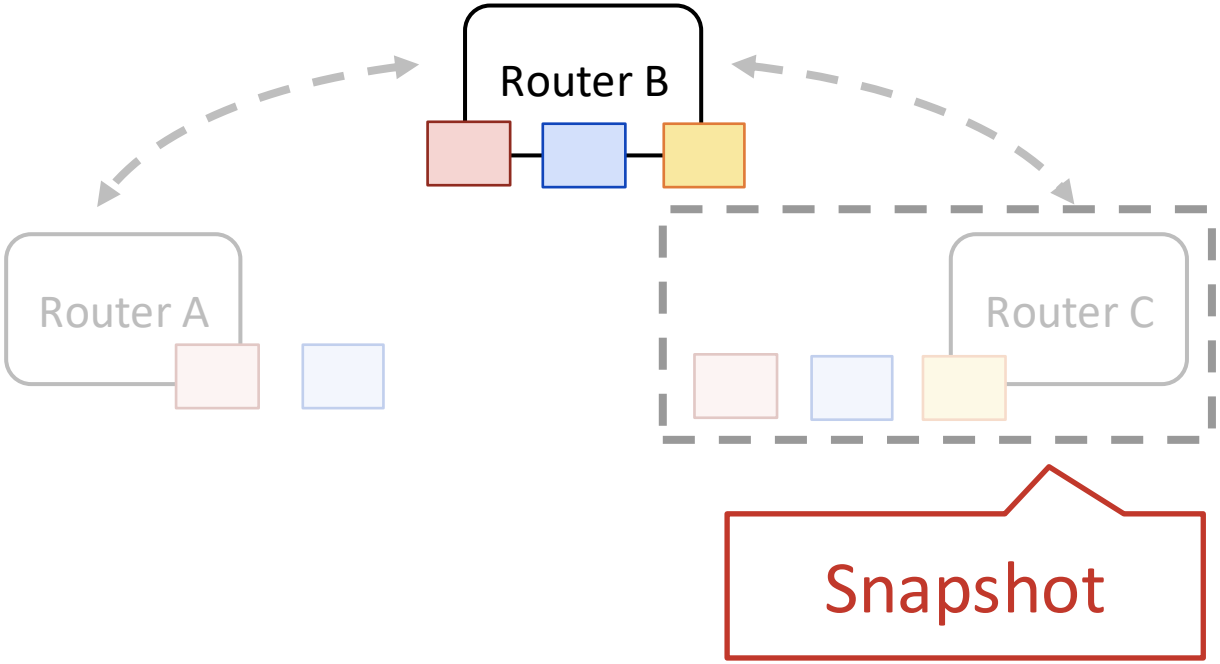
Iterative Converge: Reduce Memory Residency



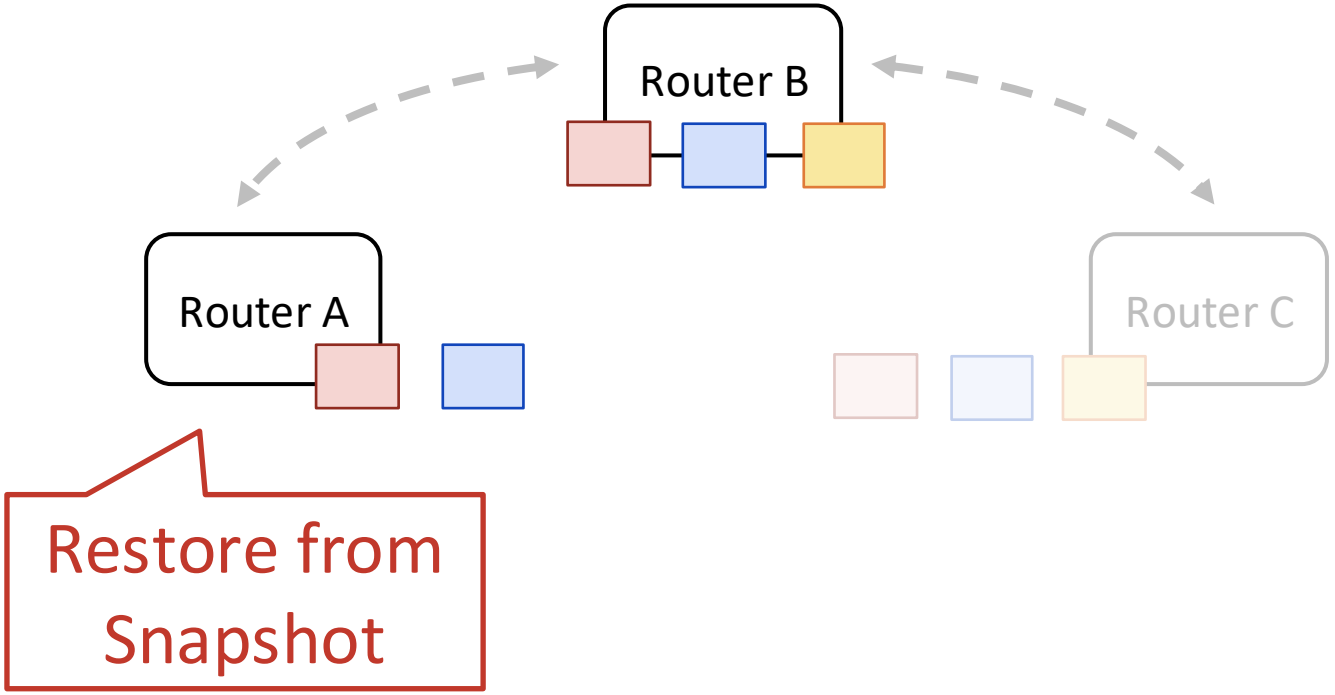
Iterative Converge: Reduce Memory Residency



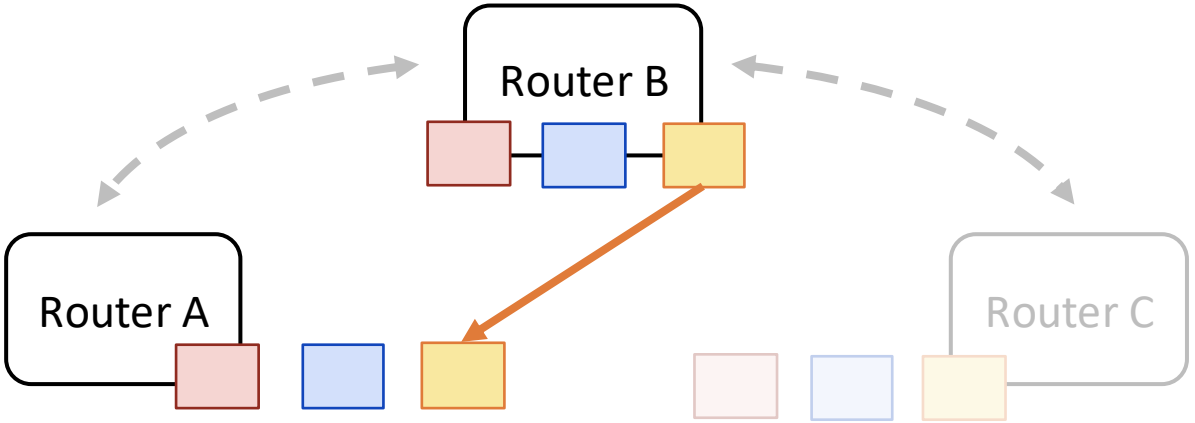
Iterative Converge: Reduce Memory Residency



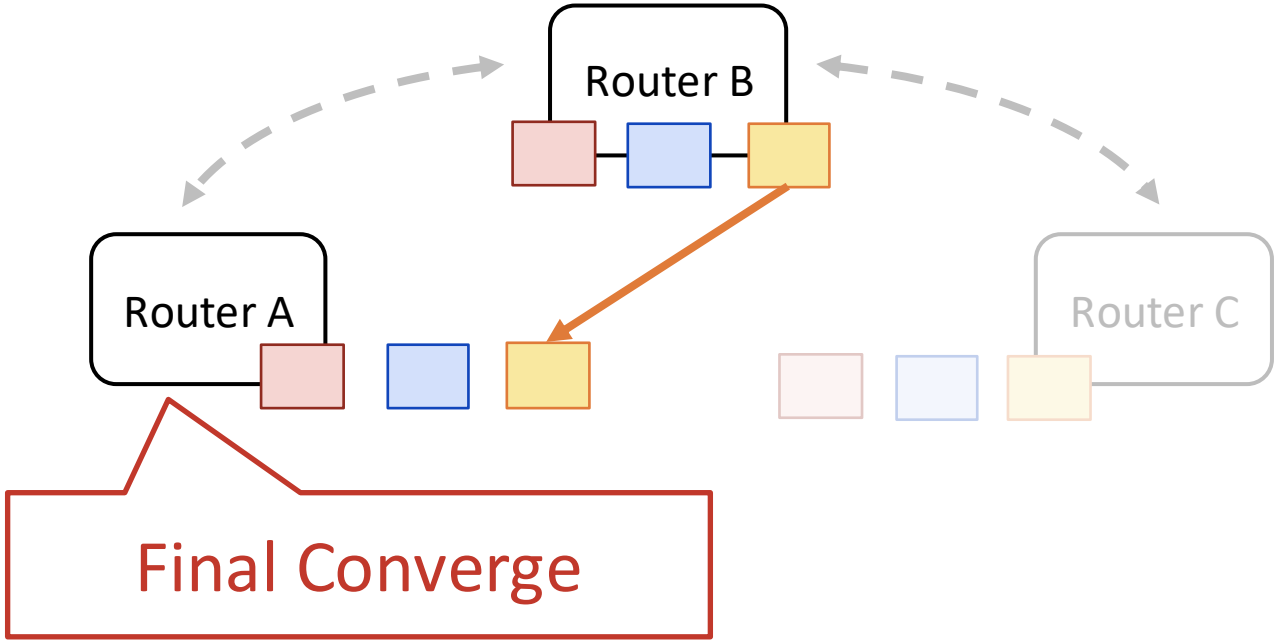
Iterative Converge: Reduce Memory Residency



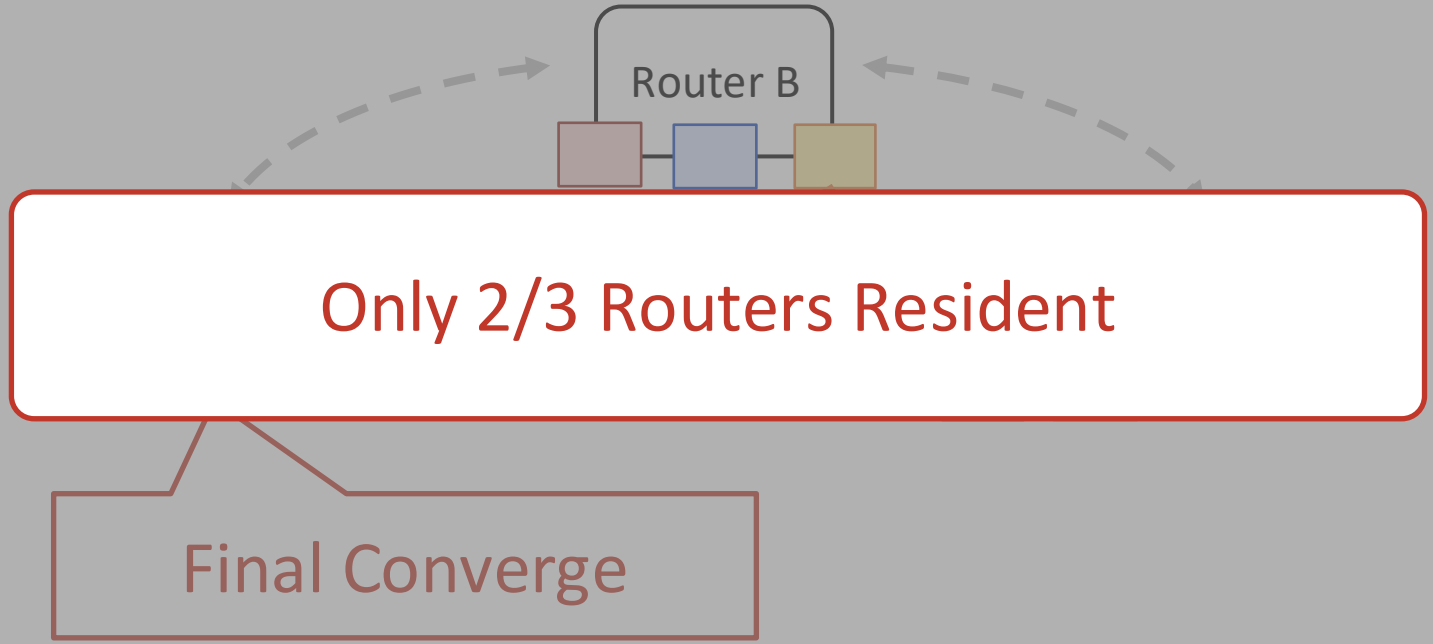
Iterative Converge: Reduce Memory Residency



Iterative Converge: Reduce Memory Residency



Iterative Converge: Reduce Memory Residency

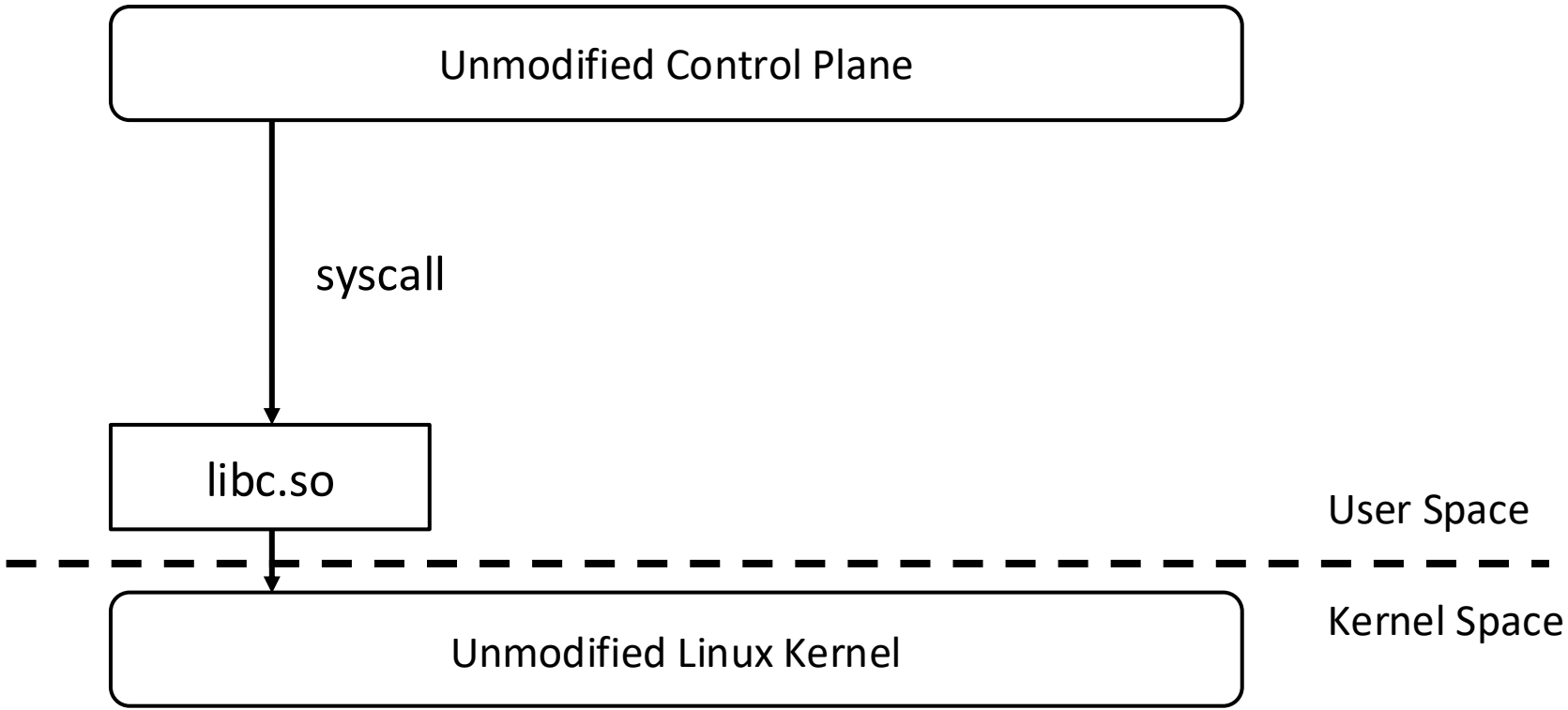


How to realize such a runtime?

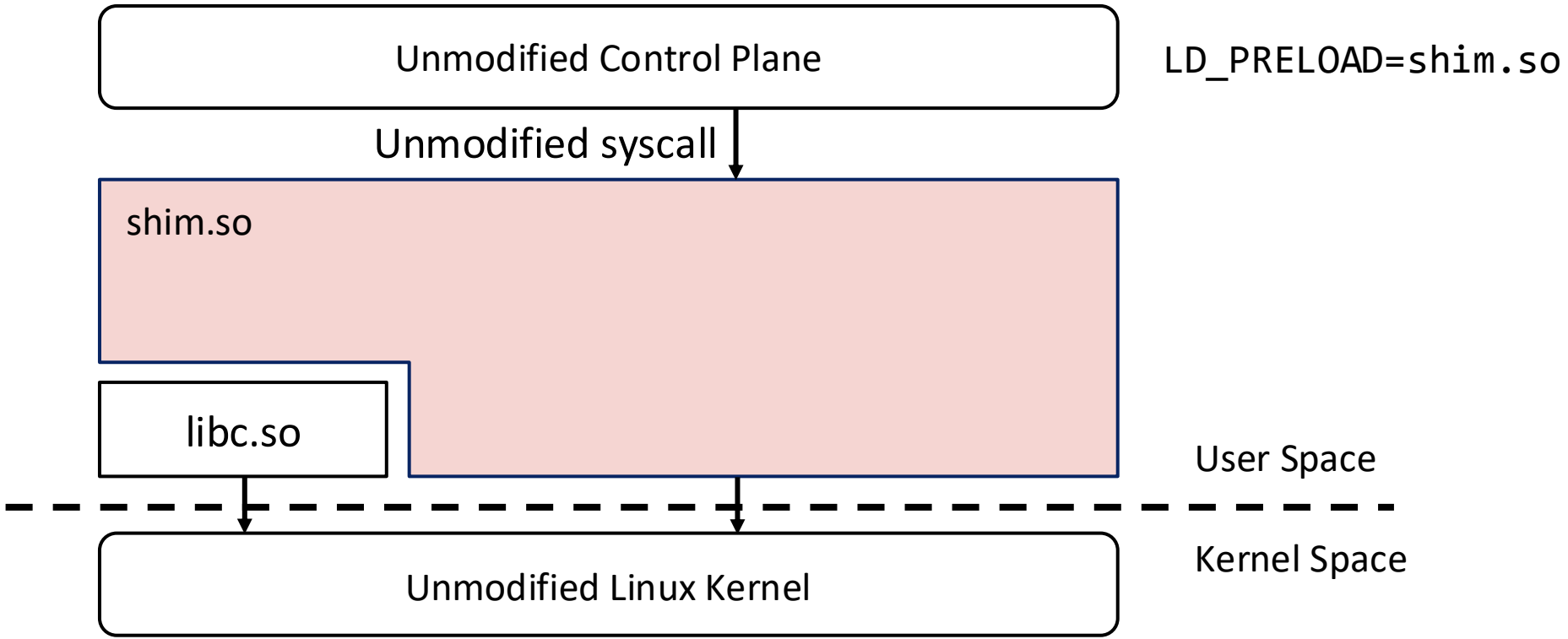
Goal: Not Touching Software

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

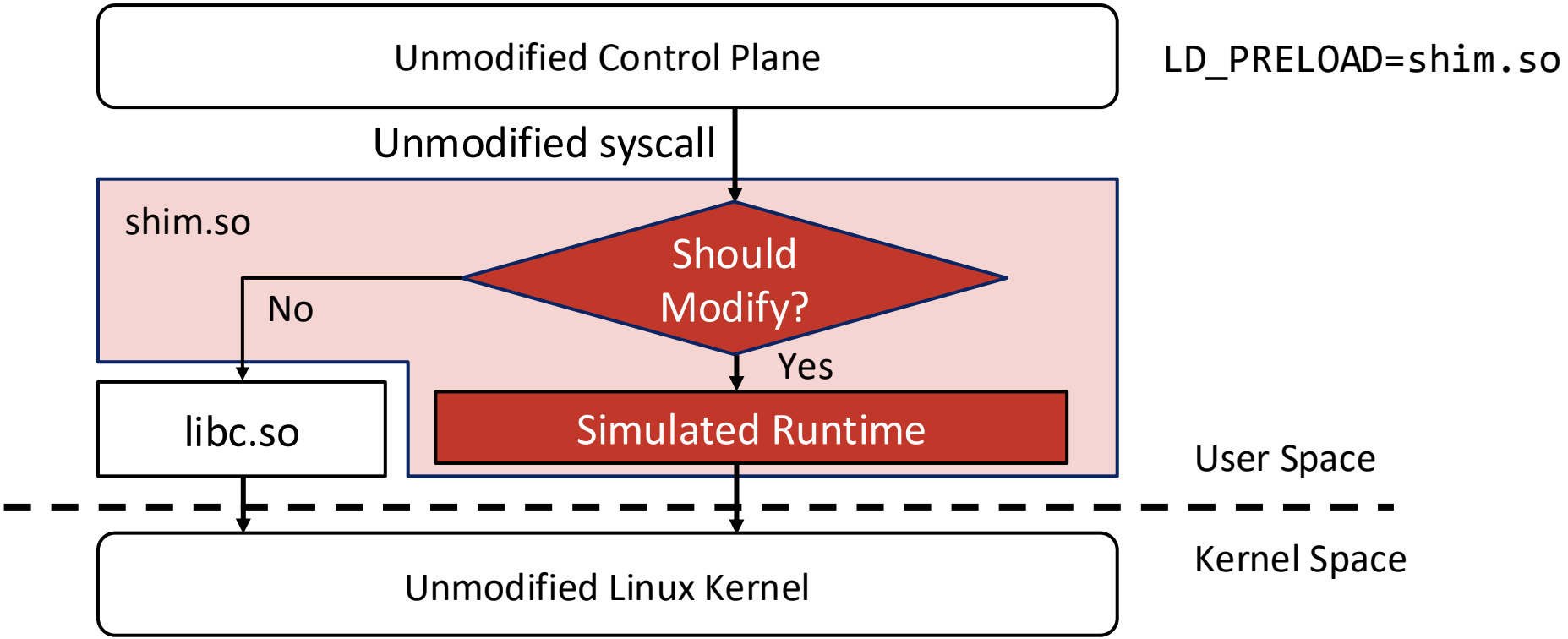
Transparent Syscall Interposition via LD_PRELOAD



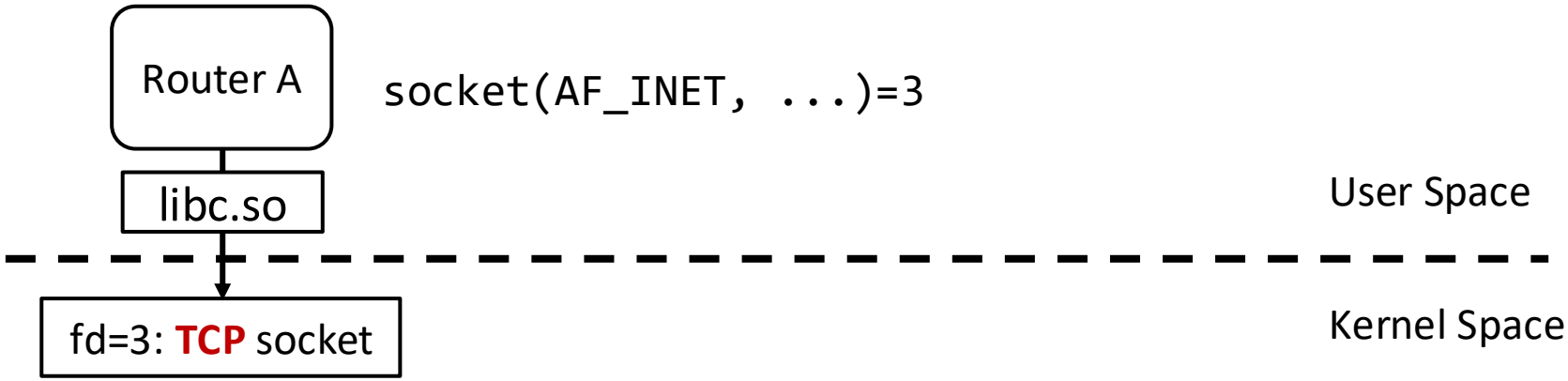
Transparent Syscall Interposition via LD_PRELOAD



Transparent Syscall Interposition via LD_PRELOAD

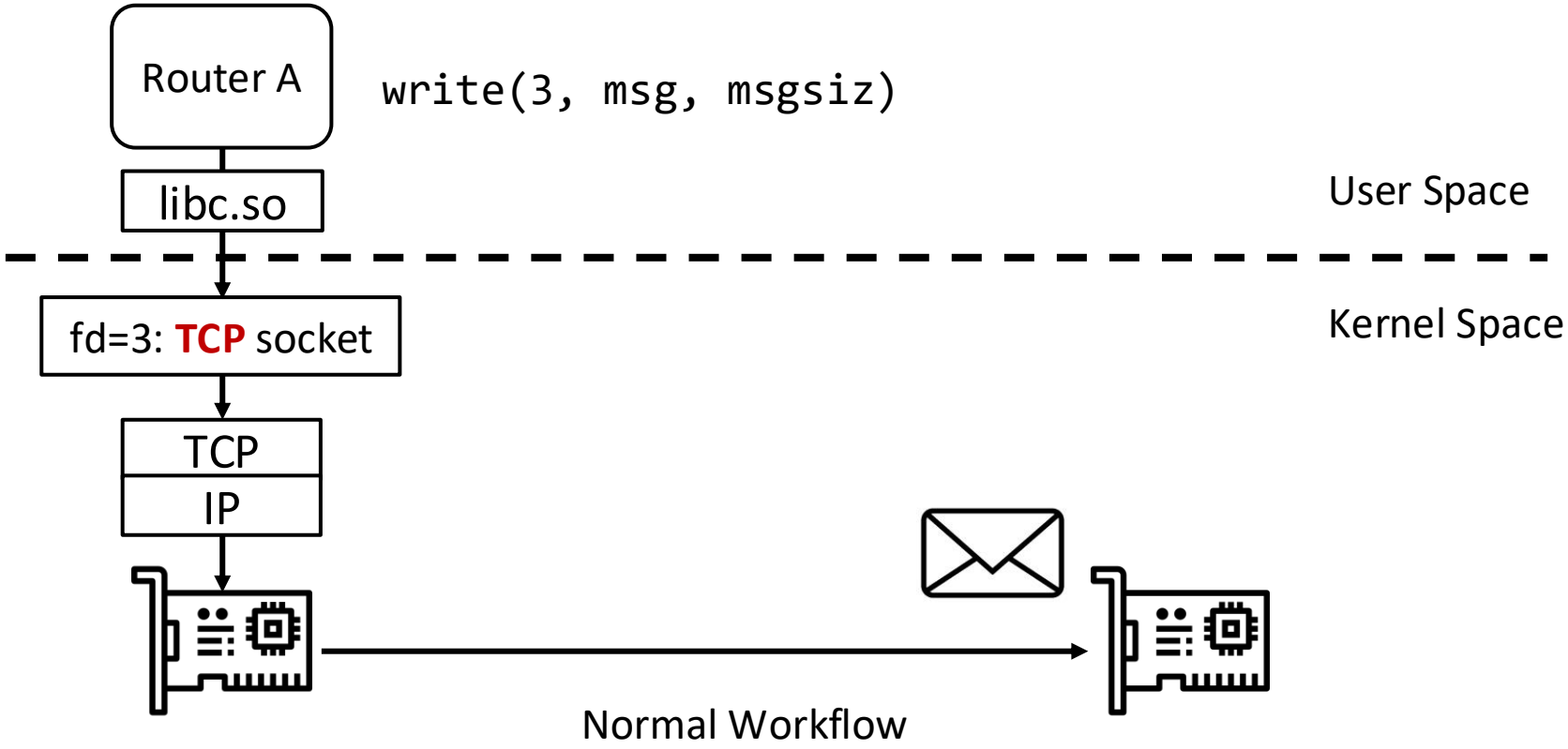


Simulate TCP Semantics with IPC

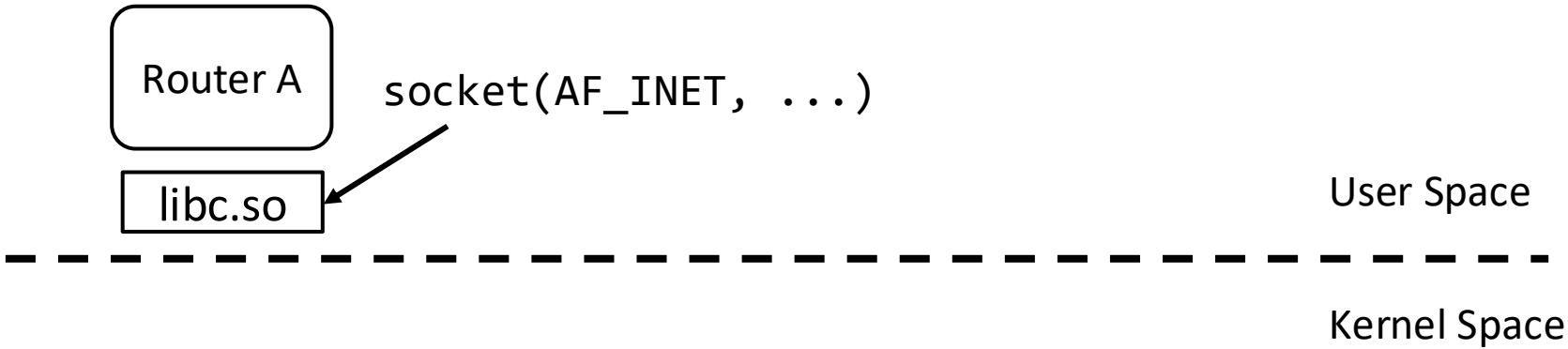


Normal Workflow

Simulate TCP Semantics with IPC

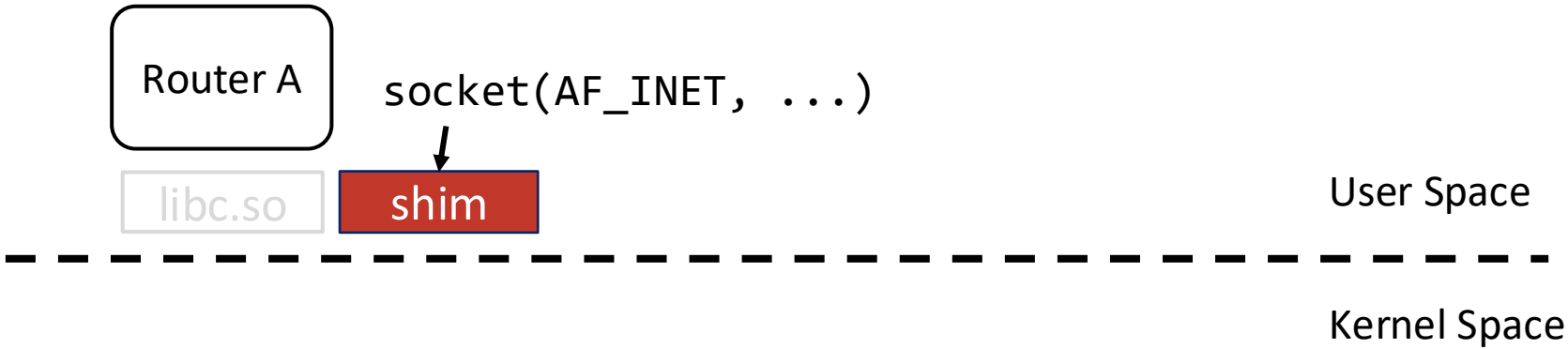


Simulate TCP Semantics with IPC



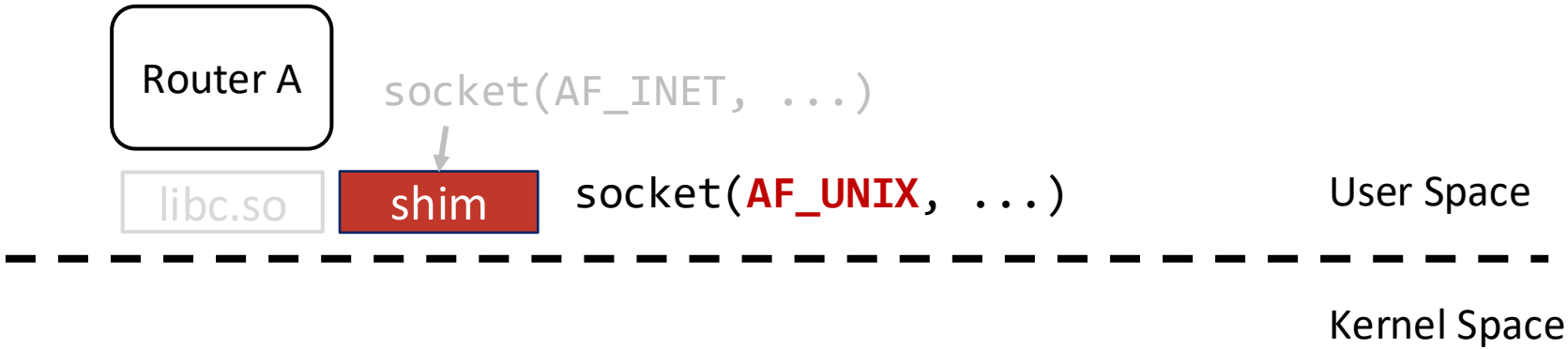
linker dynamically finds implementation of socket()

Simulate TCP Semantics with IPC



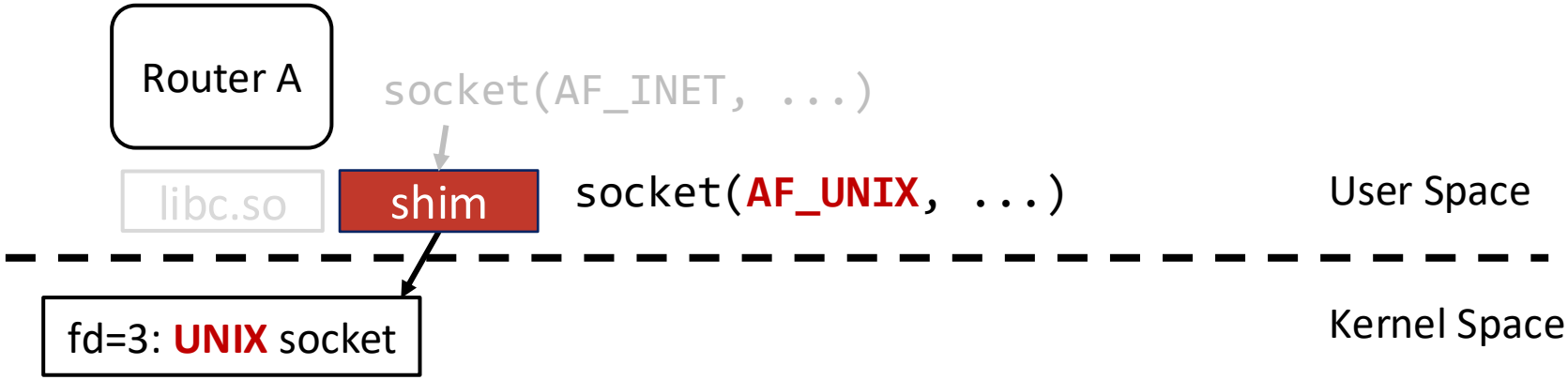
... which can be redirected to custom implementation

Simulate TCP Semantics with IPC



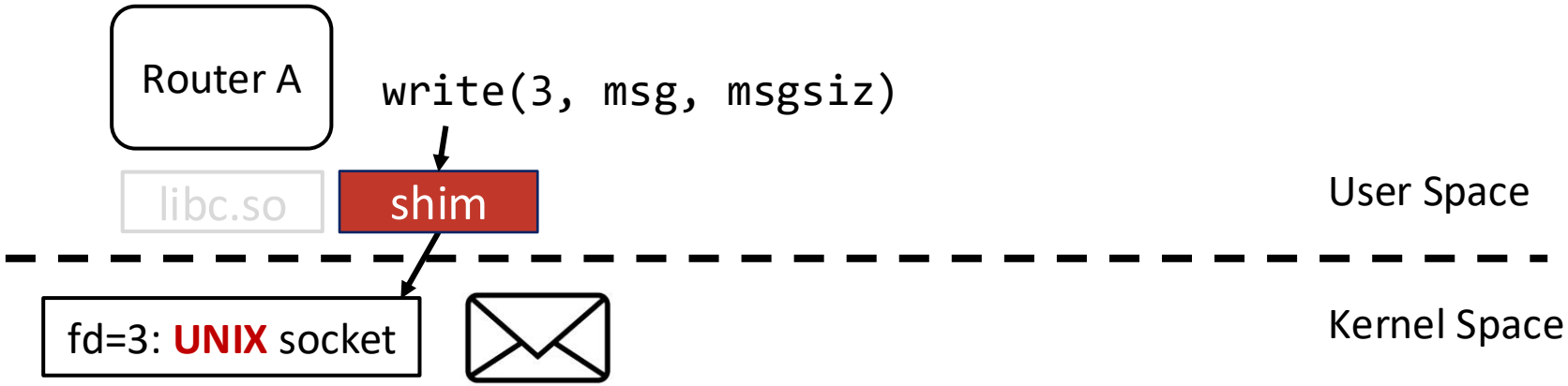
... which can be redirected to custom implementation

Simulate TCP Semantics with IPC

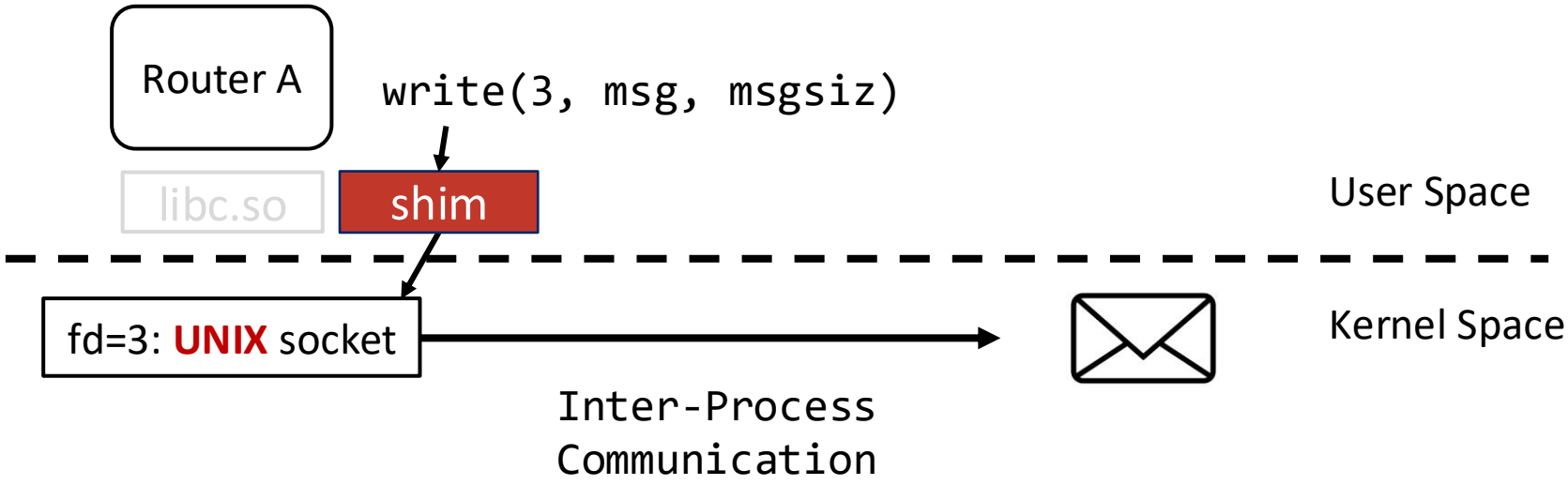


Simulate TCP Semantics with IPC

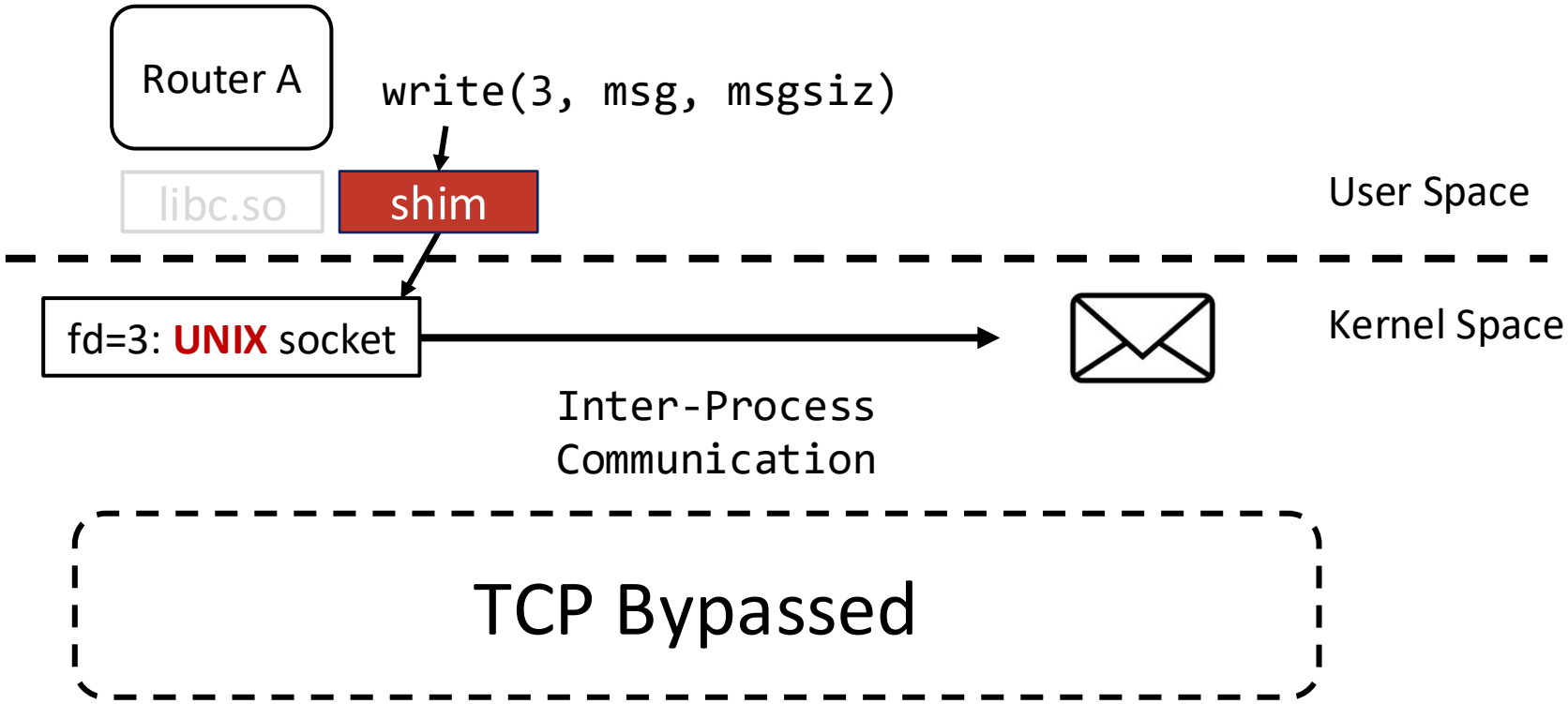
Simulate TCP Semantics with IPC



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

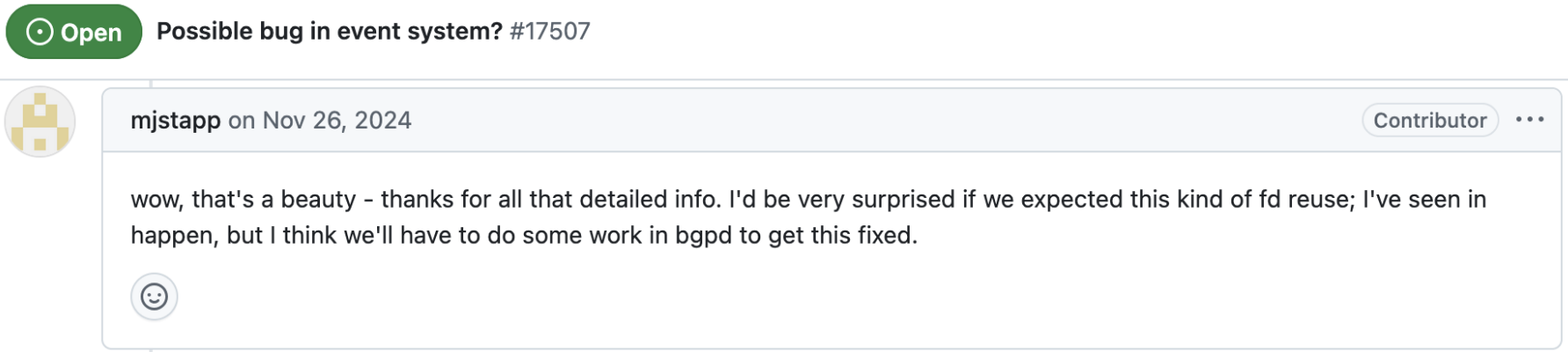
May happen with every BGP teardown, e.g. a peer router reboots

Main Thread	**I/O Thread**
	`ppoll(fd=[3,…]) blocks…`
`close(fd=3)`	
`socket()=3, connect(3,…)`	
	`… returns POLLOUT on fd 3`

Evaluation: Fidelity

Found a new **data race** in FRRouting

Confirmed by community maintainer



Evaluation: Fidelity

Found a new **data race** in FRRouting

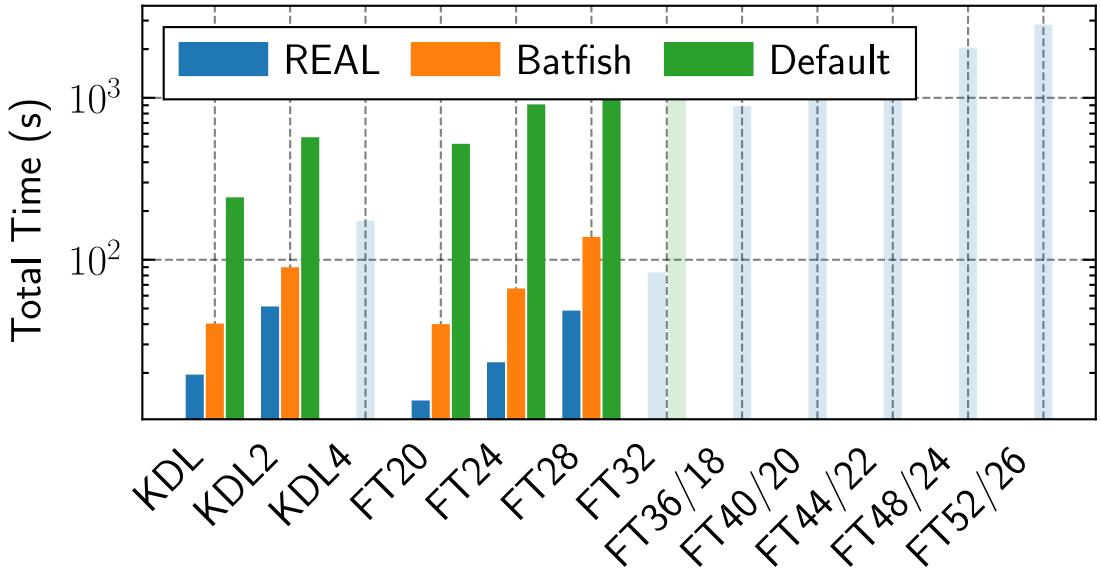
Confirmed by community maintainer

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

Evaluation: Time

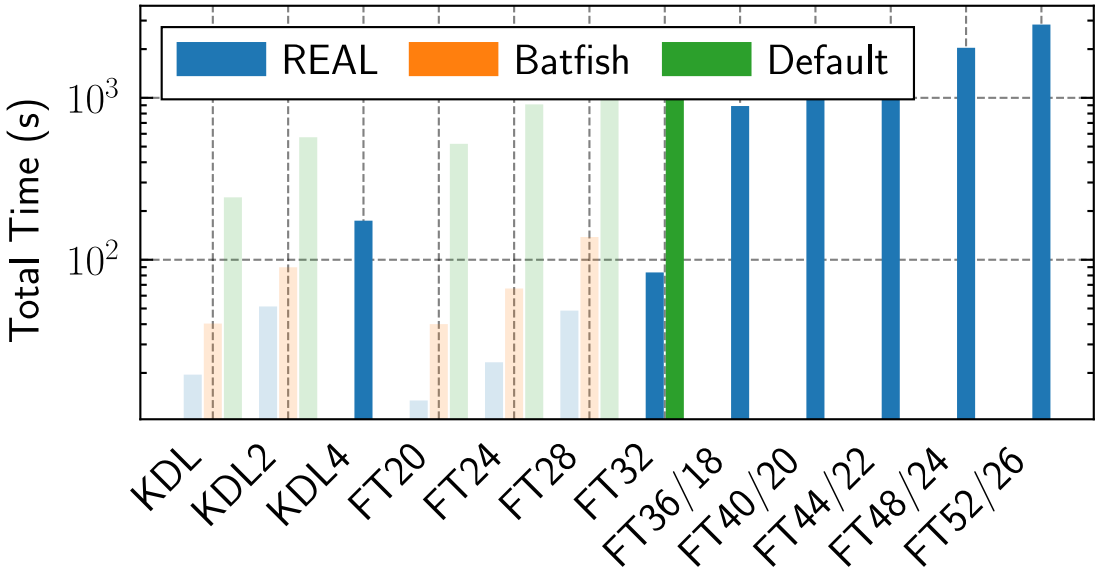
Up to **47x** faster than default Linux runtime

Even faster than Batfish



Evaluation: Memory

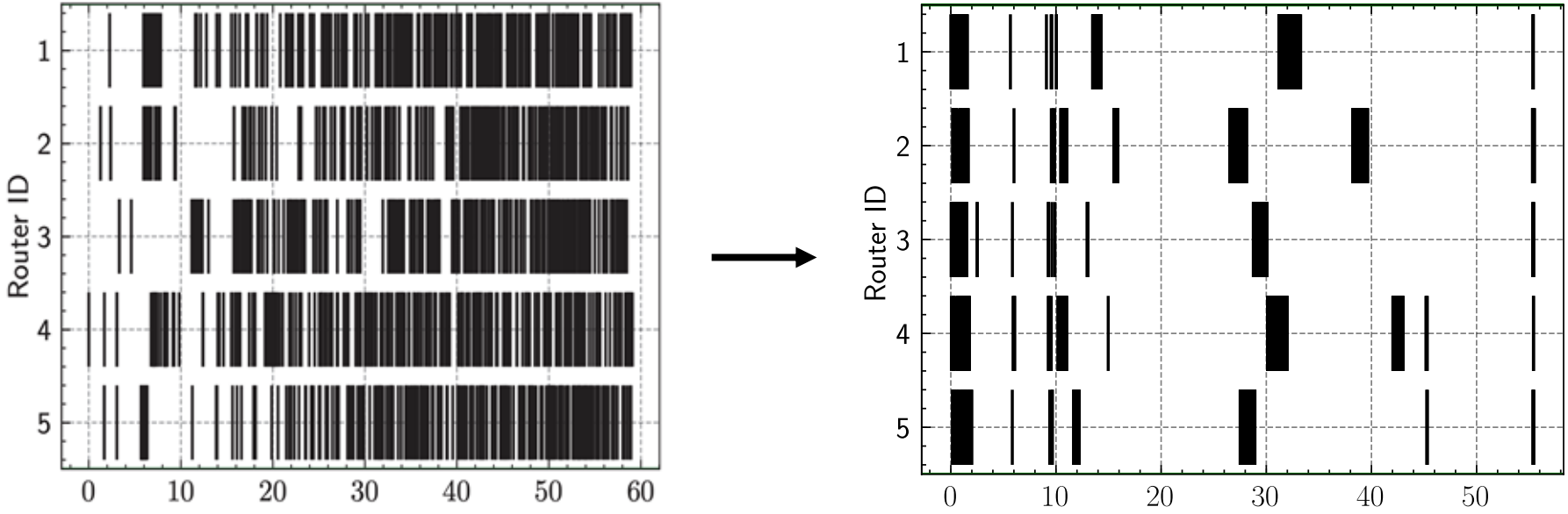
Enables larger topology (**1K→3K**) with 256G RAM



MicroBenchmark: Run-to-Idle Scheduling

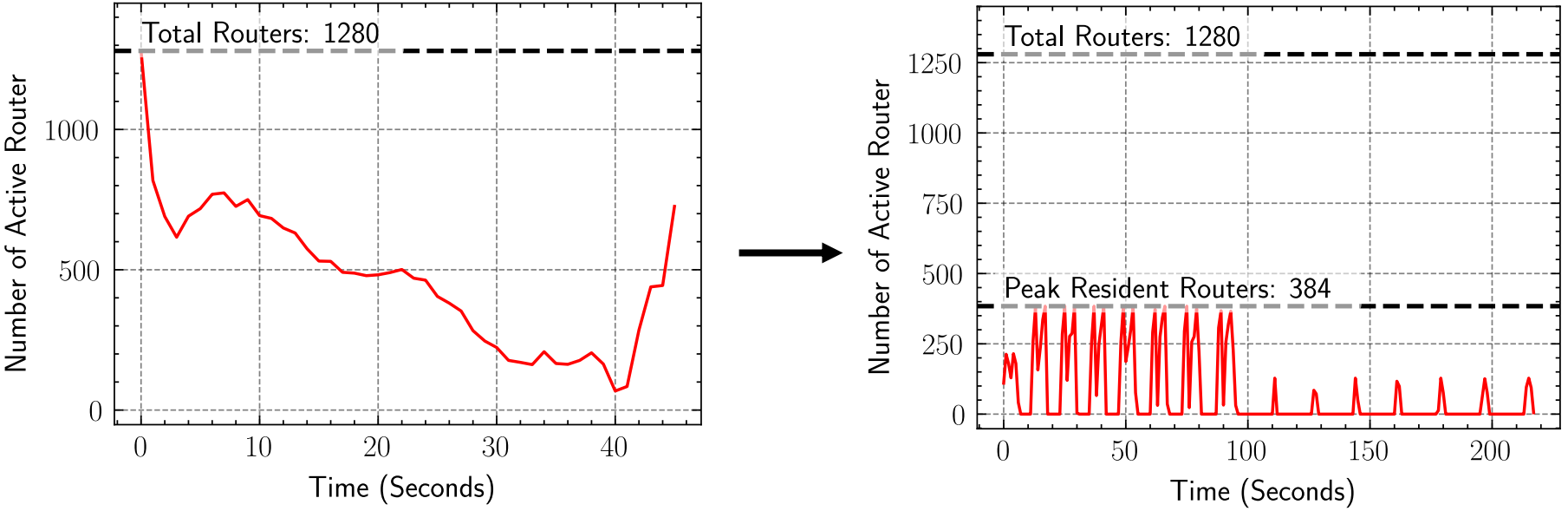
Continuous Execution: Reduced Context Switch

Last-Level-Cache miss **60% → 48%**

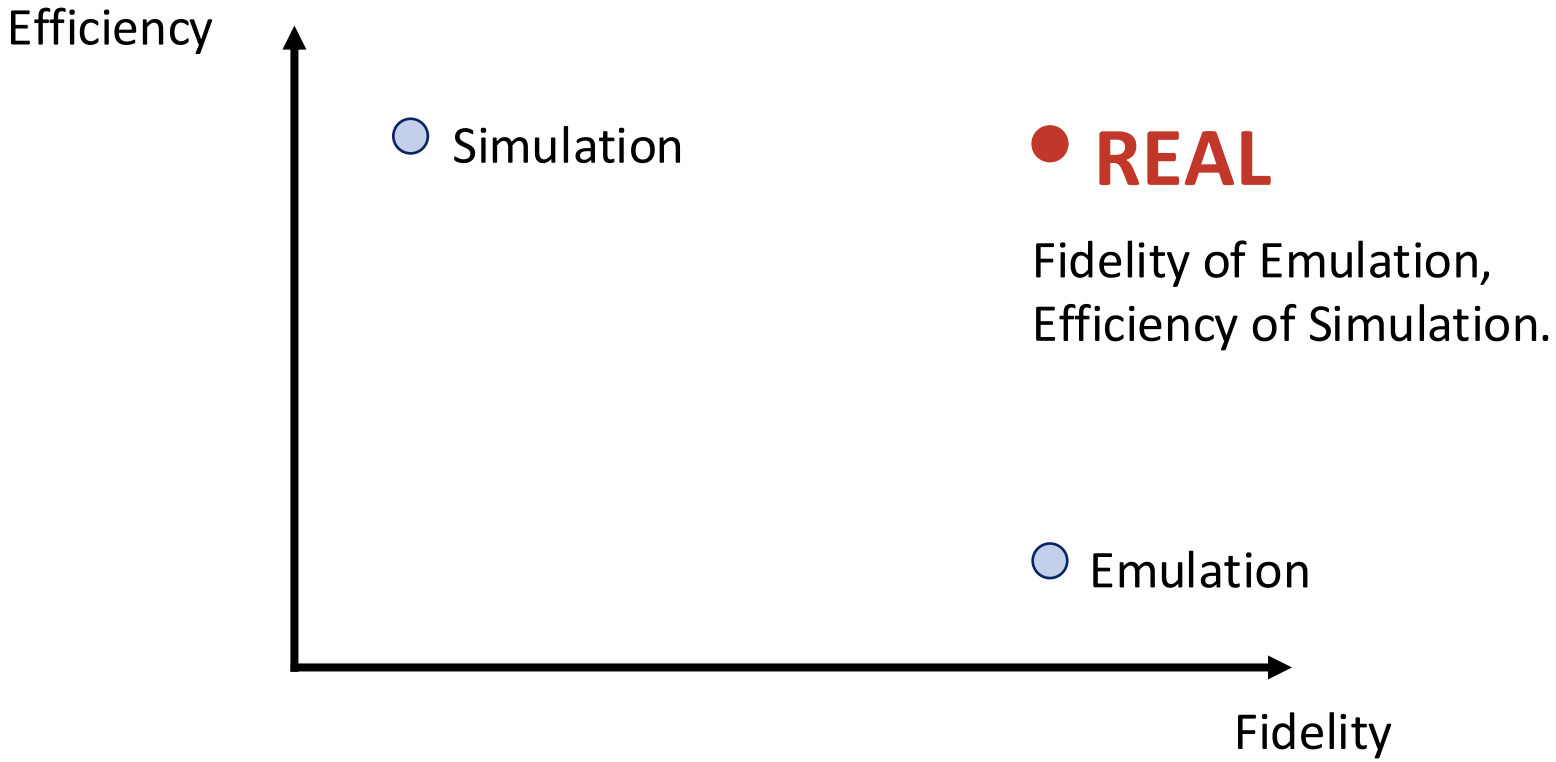


MicroBenchmark: Iterative Converge

Controlled Residency: 1280 → 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