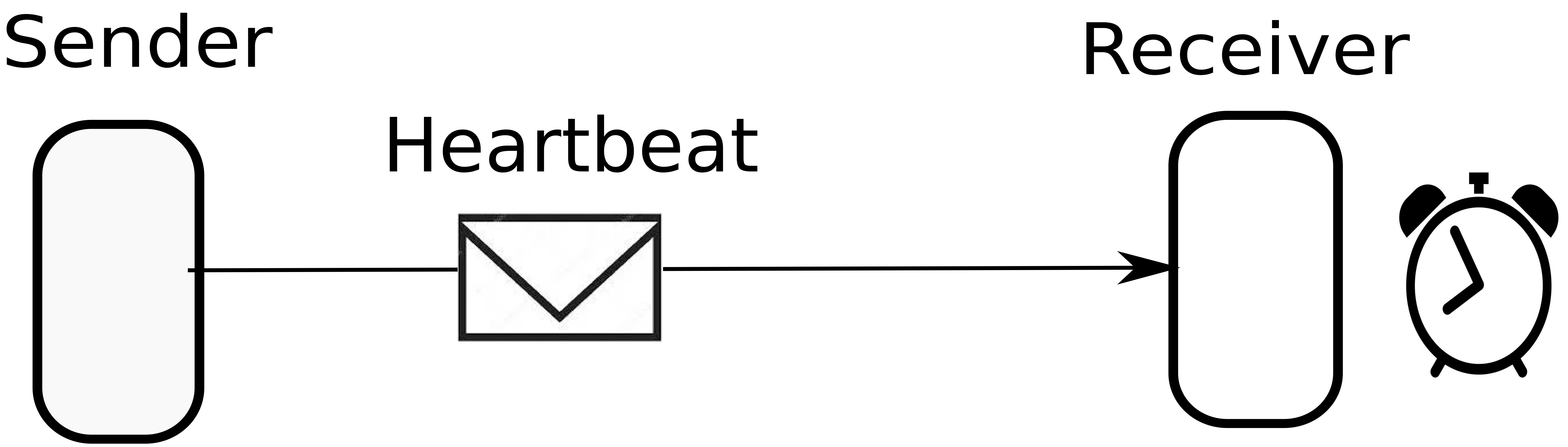
# Accurate Timeout Detection Despite Arbitrary Processing Delays

**Sixiang Ma**, Yang Wang

The Ohio State University

# Timeout is Widely Used in Failure Detection

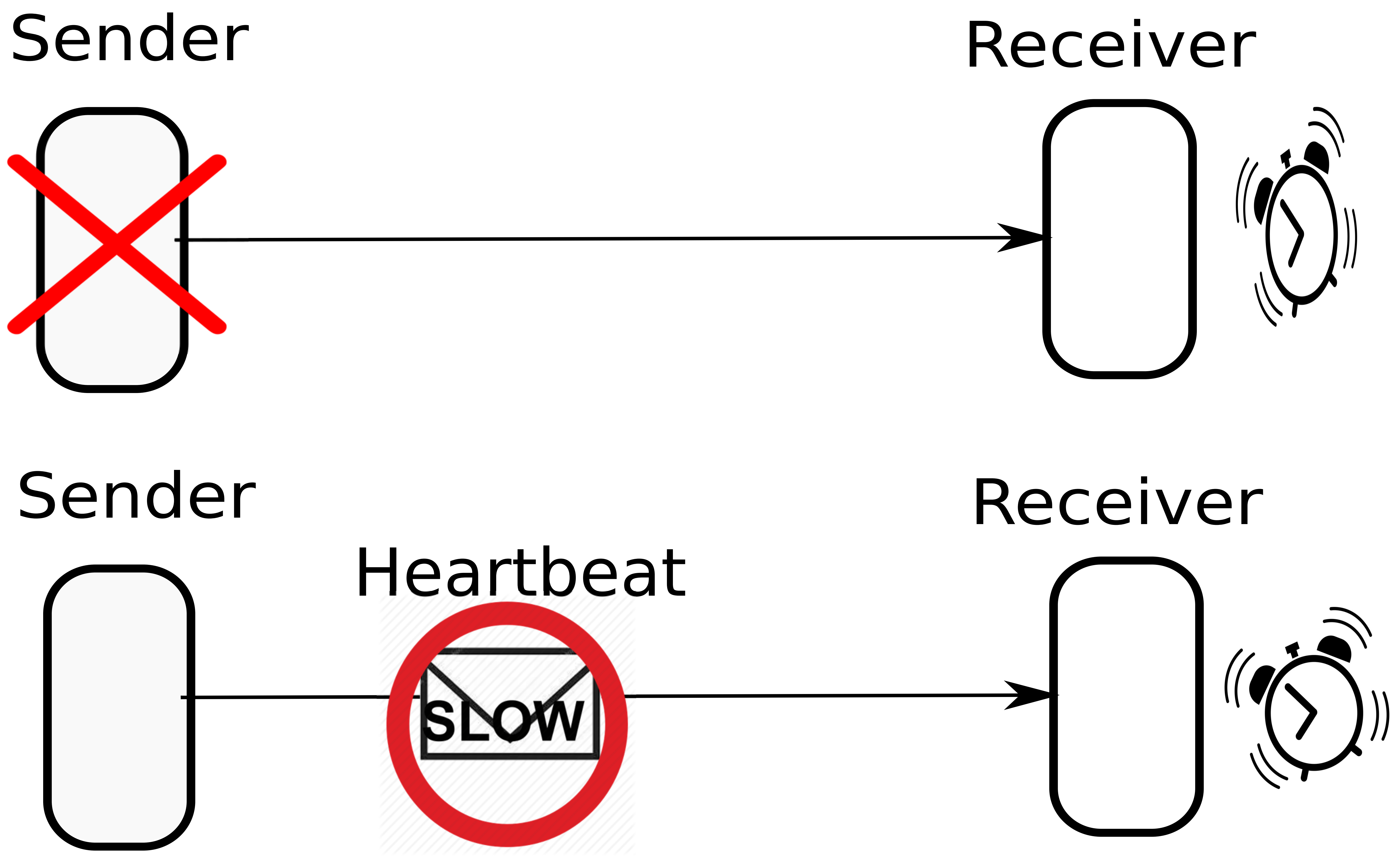


# Timeout Detection Can be Inaccurate

When **timeout happens**, it is hard to tell between:

- sender crash failure
- <u>heartbeat delay</u>

**Accuracy**: when receiver reports timeout, sender mush have failed. [Chandra, Journal of ACM' 96]



# How to Ensure System Correctness

Approach 1: Paxos-based consensus

- ensure correctness despite inaccurate timeout detection
- high cost and complexity
- examples: ZooKeeper, Chubby, Spanner, etc.

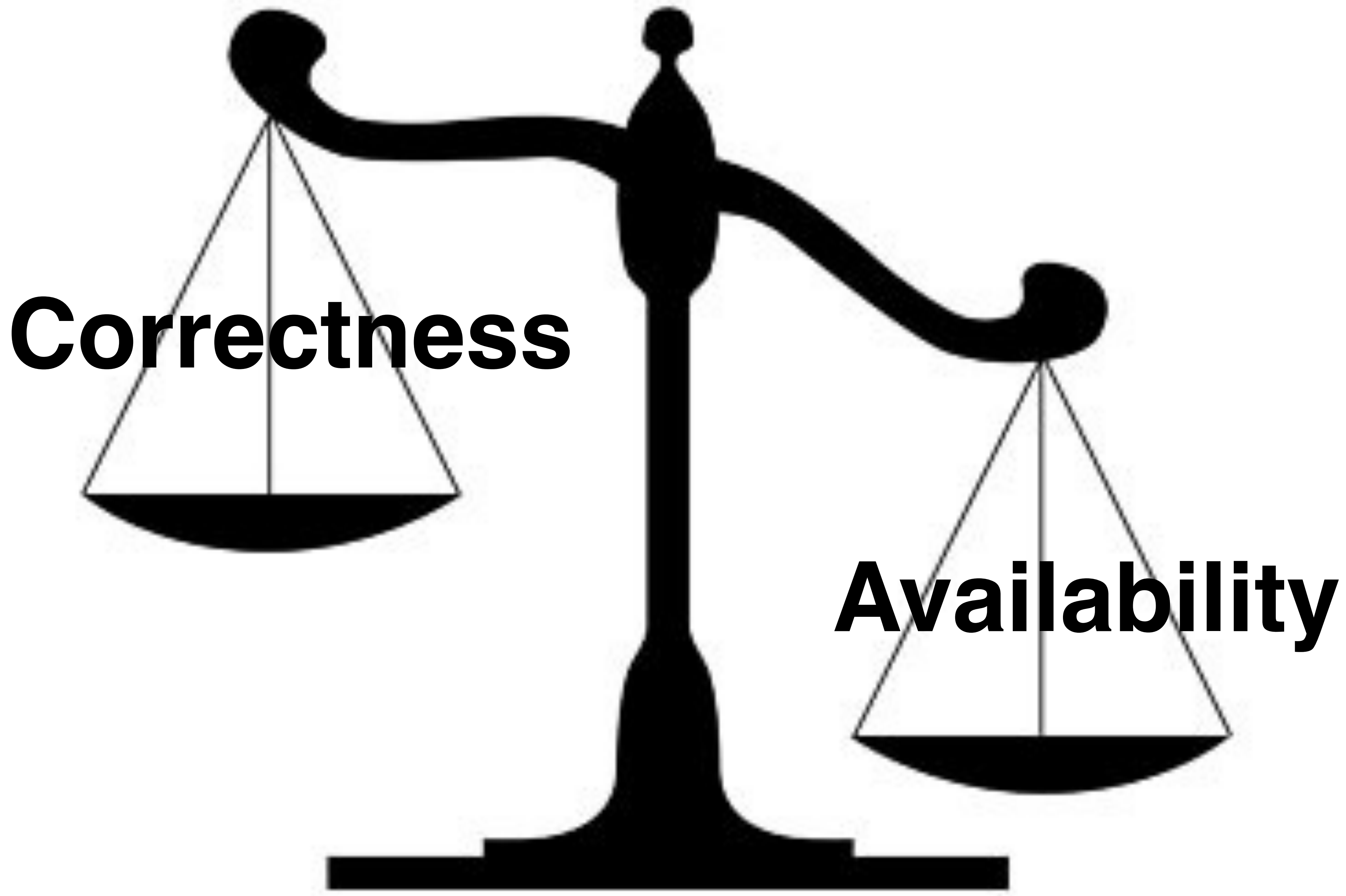
# How to Ensure System Correctness

Approach 2: Set long timeout intervals

- system correctness relies on timeout accuracy
- estimate the maximum delay of the communication channel
- examples: HDFS, Ceph, Yarn, etc
- **Our work** aims to improve this approach

# The Dilemma: Availability v.s. Correctness

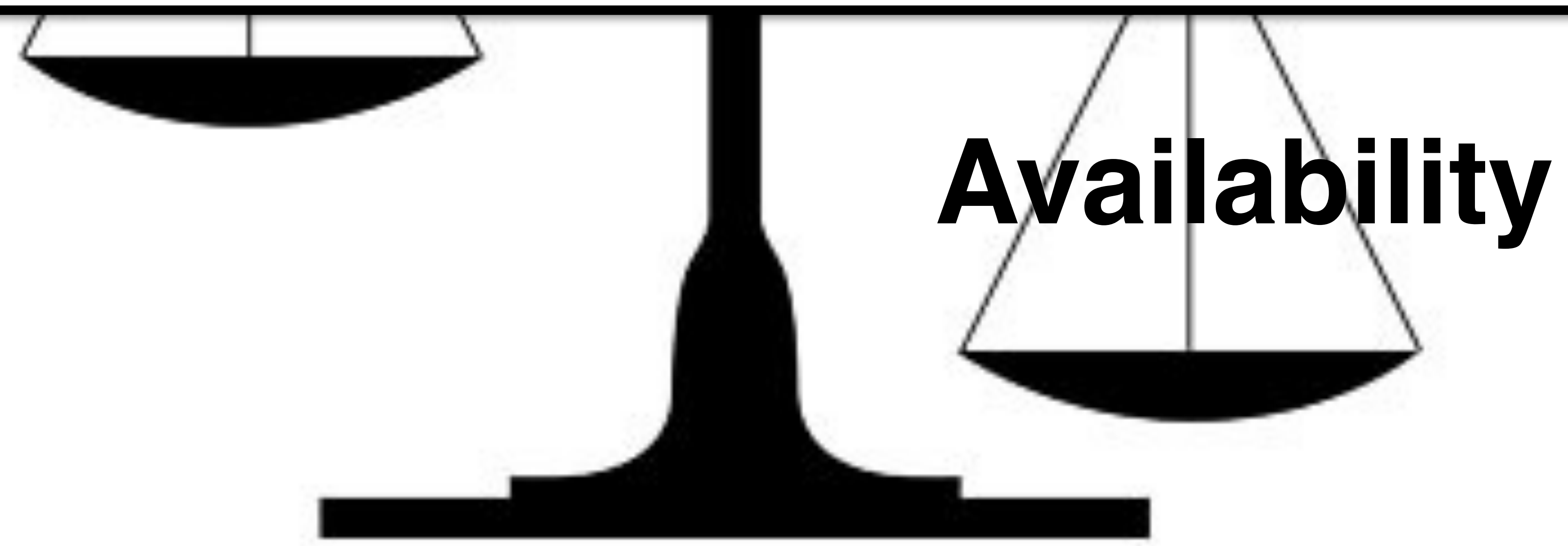
- **Correctness**: require **long timeout** to tolerate maximum delays
- **Availability**: prefer **short timeout** for fast failure detection



# The Dilemma: Availability v.s. Correctness

- Correctness: require long timeout to tolerate maximum delays
- Availa ... tion

**Can we shorten timeout intervals without sacrificing correctness?**



# Motivations

1. Long delays in OS and application
2. Their whitebox nature creates opportunities for better solutions

# Motivations

1. Long delays in OS and application
2. Their whitebox nature creates opportunities for better solutions

# Heartbeat Delay in Our Experiment

- Disk I/O: **10** seconds
- Packet processing: **2** seconds
- JVM garbage collection: **26** seconds
- Application specific delays: **several minutes**
  - HDFS: directories deletion before heartbeat sending
  - ZooKeeper: session close/expire flooding

# Heartbeat Delay Reported in Communities

**CEPH**-19335: MDS heartbeat timeout during rejoin, when working with large amount of

**HDFS**-611: Heartbeats times
from Datano
when there a

**ZOOKEEPER**-1049:
Session expire/close
ats

**HBASE**-3273: Set the ZK default timeout to 3 minutes

“Stack suggested that we increase the ZK timeout and proposed that we set it to **3 minutes**. This should cover most of the big GC pauses.”

**HDFS**-9901: Move disk IO out of the heartbeat thread

“In extreme cases, the heartbeat thread hang more than **10 minutes** so the namenode marked the datanode as dead”

heartbeats get blocked
by disk in checkBlock()

over large regions”

# Delays in OS and Application Are Significant

Compared to default timeout, delays in OS and App are **significant**

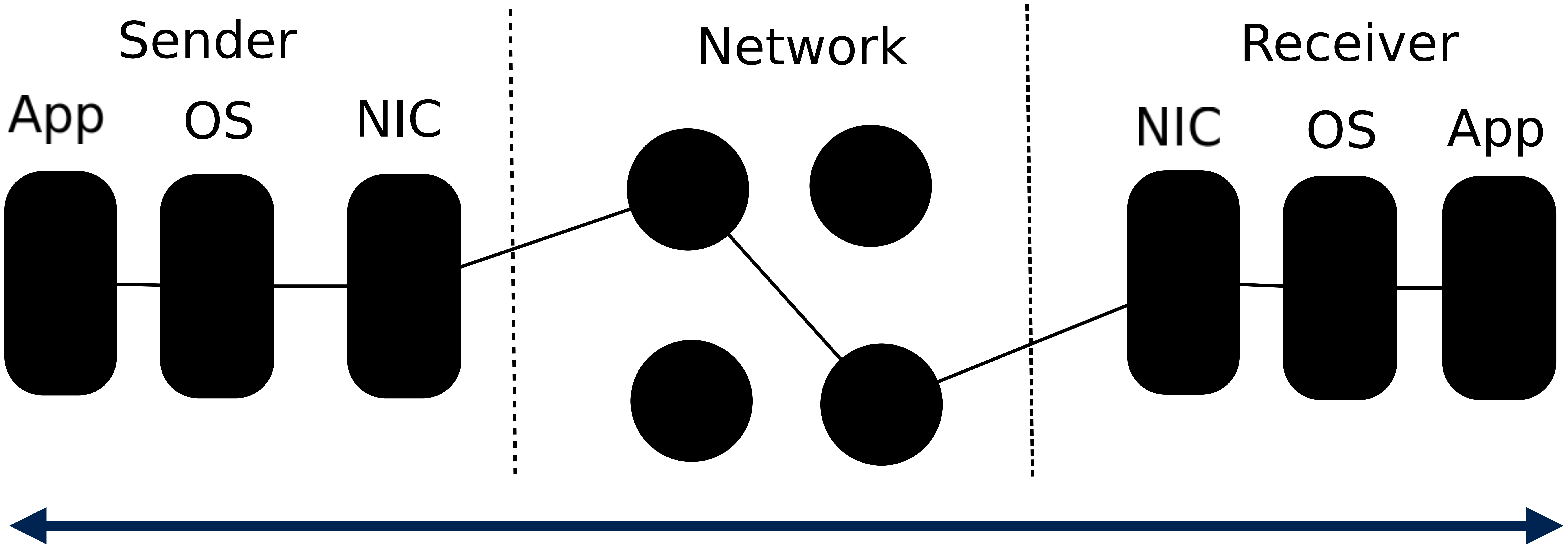
- HDFS: **30** seconds
- Ceph: **20** seconds
- ZooKeeper: **5** seconds

# Motivations

1. Long delays in OS and application
2. Their whitebox nature creates opportunities for better solutions

# Existing Timeout Views Channel as a Blackbox

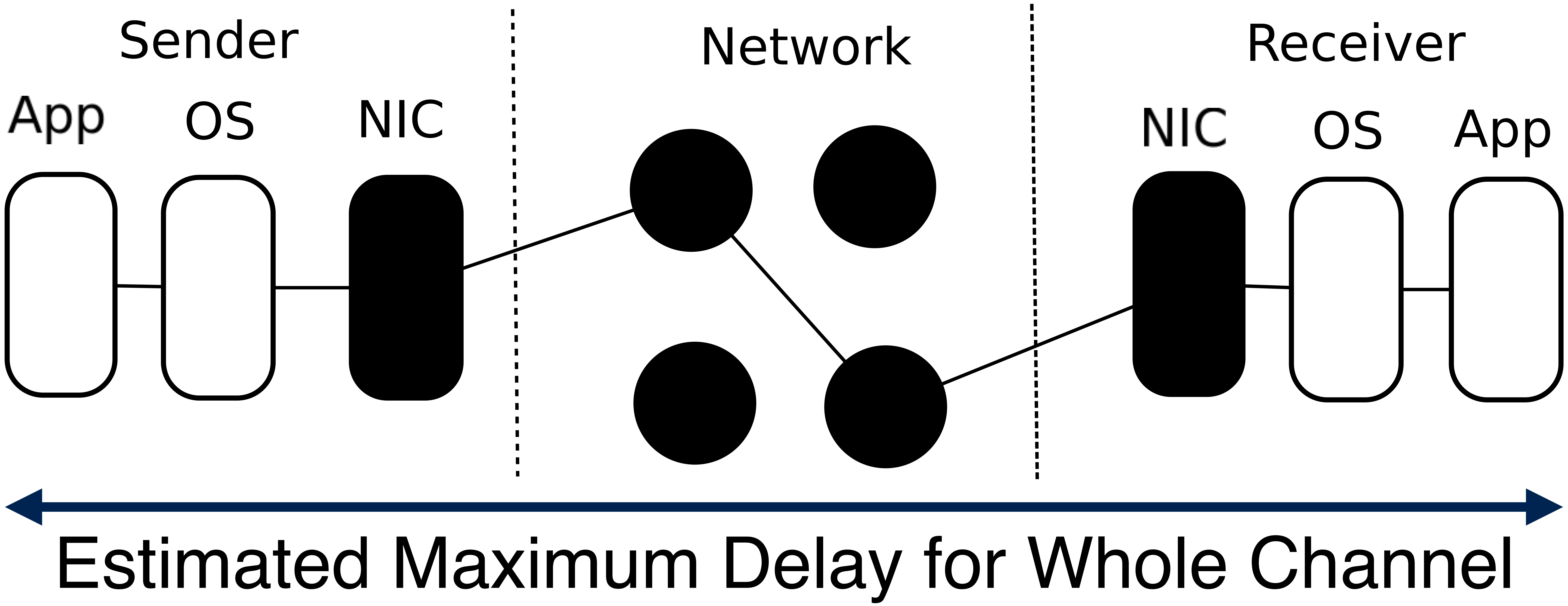
- **Blackbox**: only provides information when receiving a packet




Estimated Maximum Delay for Whole Channel

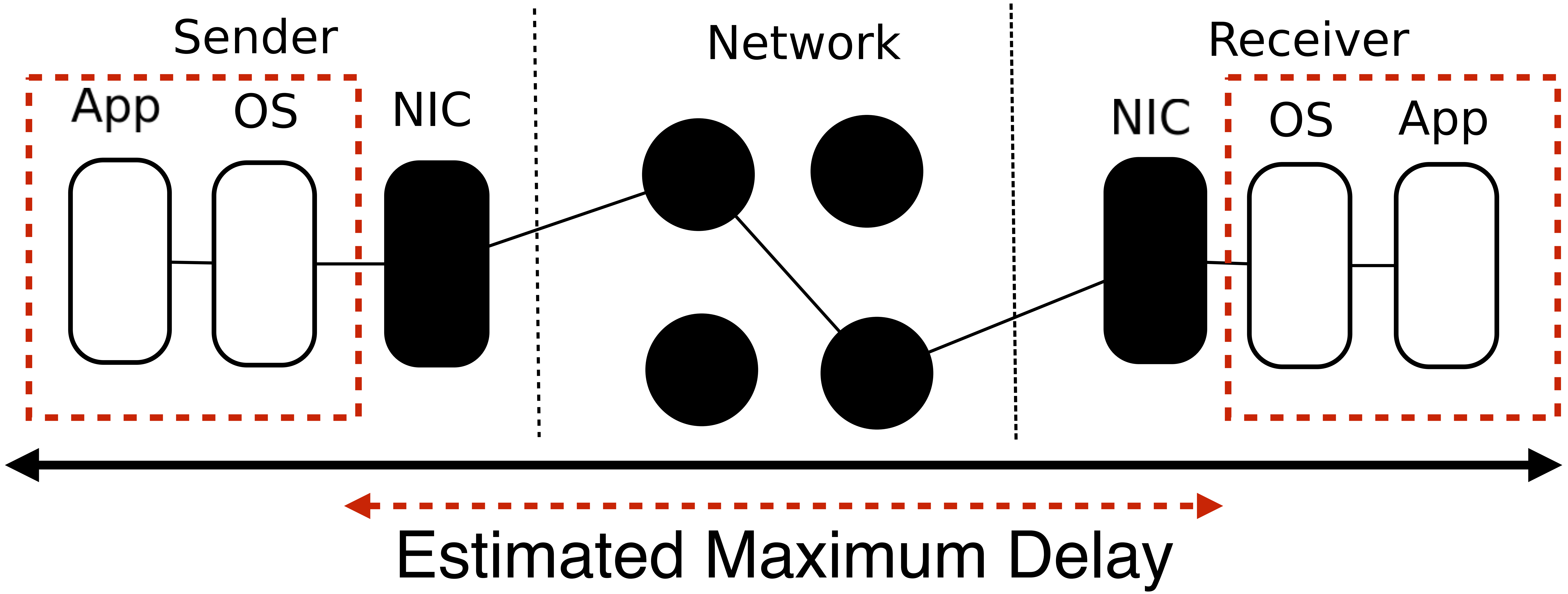
# Whitebox Nature of OS and Application

- **Whitebox**: can provide information such as packet pending/drop



# Whitebox Nature of OS and Application

- **Whitebox**: can provide information such as packet pending/drop
- Can we utilize whitebox nature to design better solution?



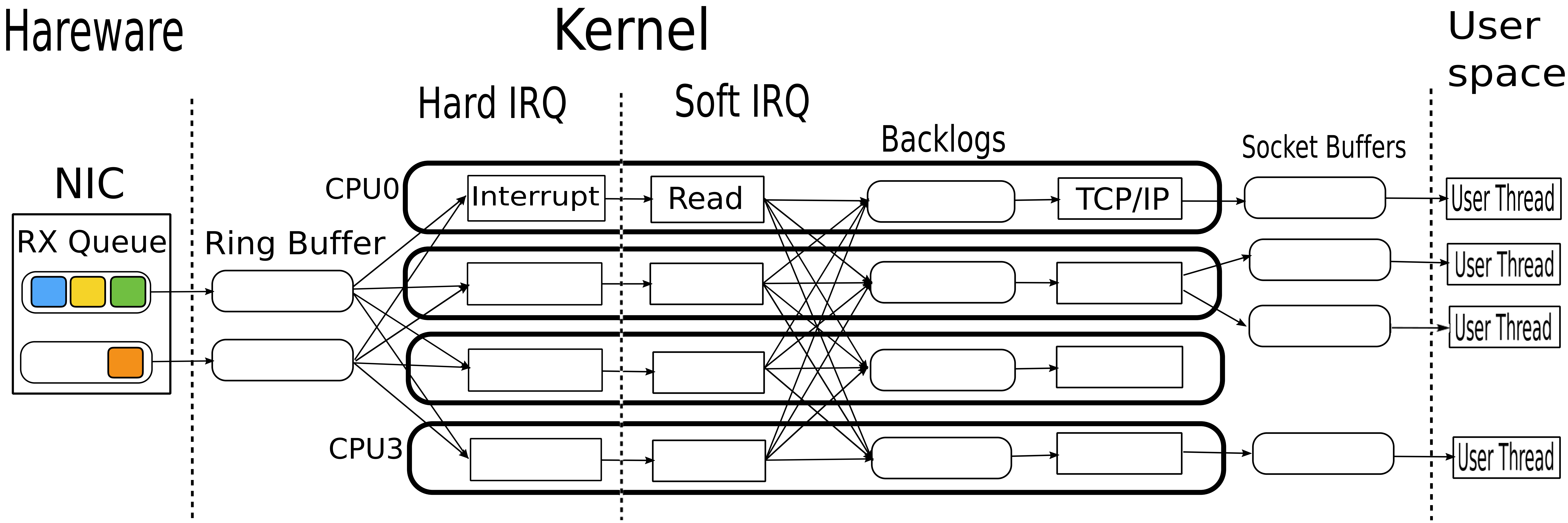
# Overview of SafeTimer

- **Goal**: if the receiver reports timeout, the sender must have failed
- **Assumptions** of SafeTimer
  - Delays in whitebox can be arbitrarily long
  - SafeTimer relies on existing protocol for blackbox
- **Solutions**
  - **Receiver**: check pending/dropped heartbeats when timeout occurs
  - **Sender**: blocks sender when heartbeat sending is slow

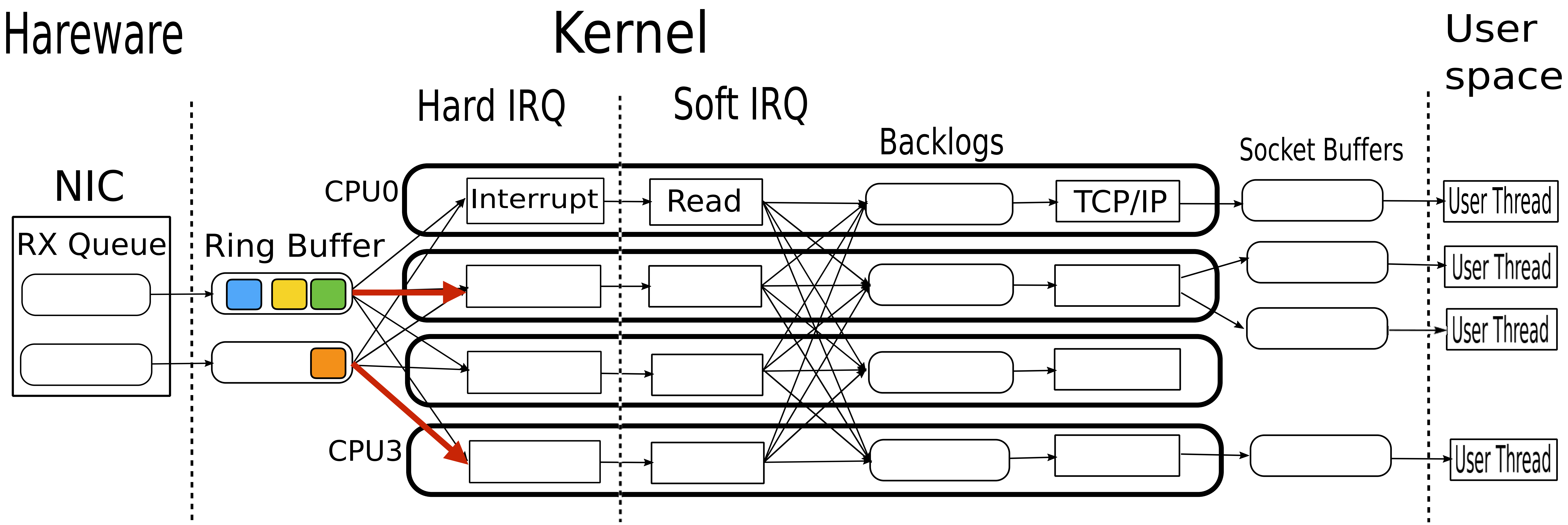
# Overview of SafeTimer

- **Goal**: if the receiver reports timeout, the sender must have failed
- **Assumptions** of SafeTimer
  - Delays in whitebox can be arbitrarily long
  - SafeTimer relies on existing protocol for blackbox
- **Solutions**
  - **Receiver**: check pending/dropped heartbeats when timeout occurs
  - **Sender**: blocks sender when heartbeat sending is slow

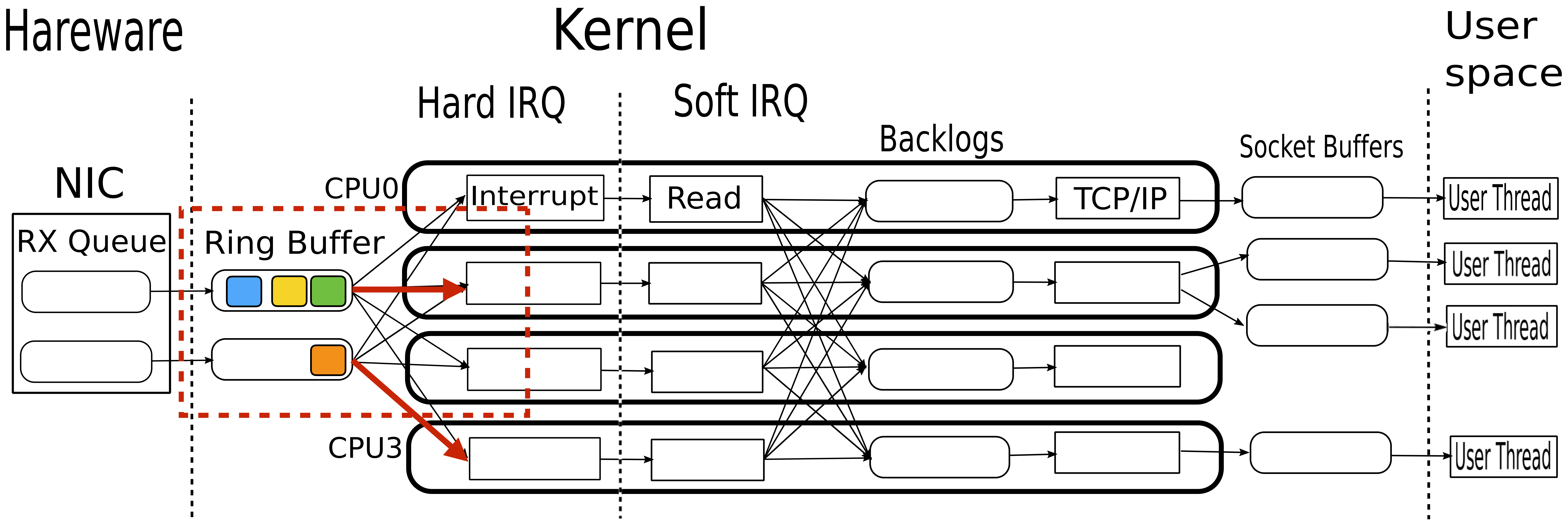
# Background: Concurrent Packet Processing



# Background: Concurrent Packet Processing

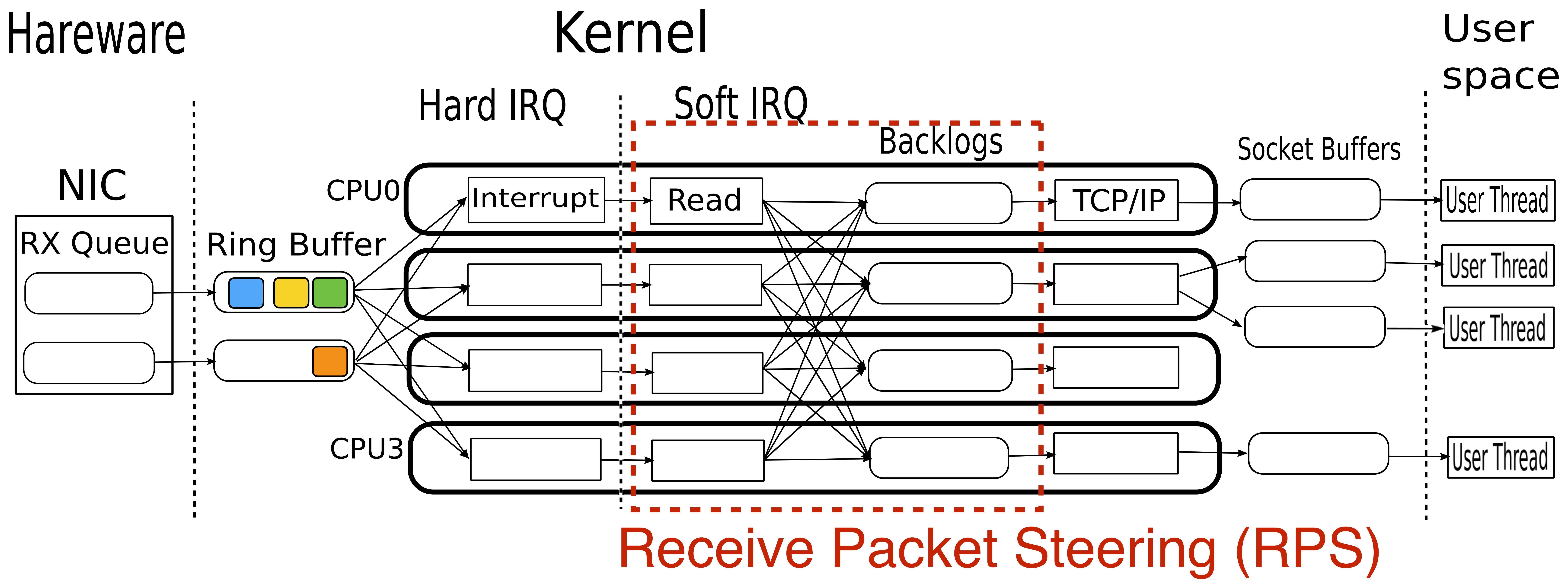


# Background: Concurrent Packet Processing

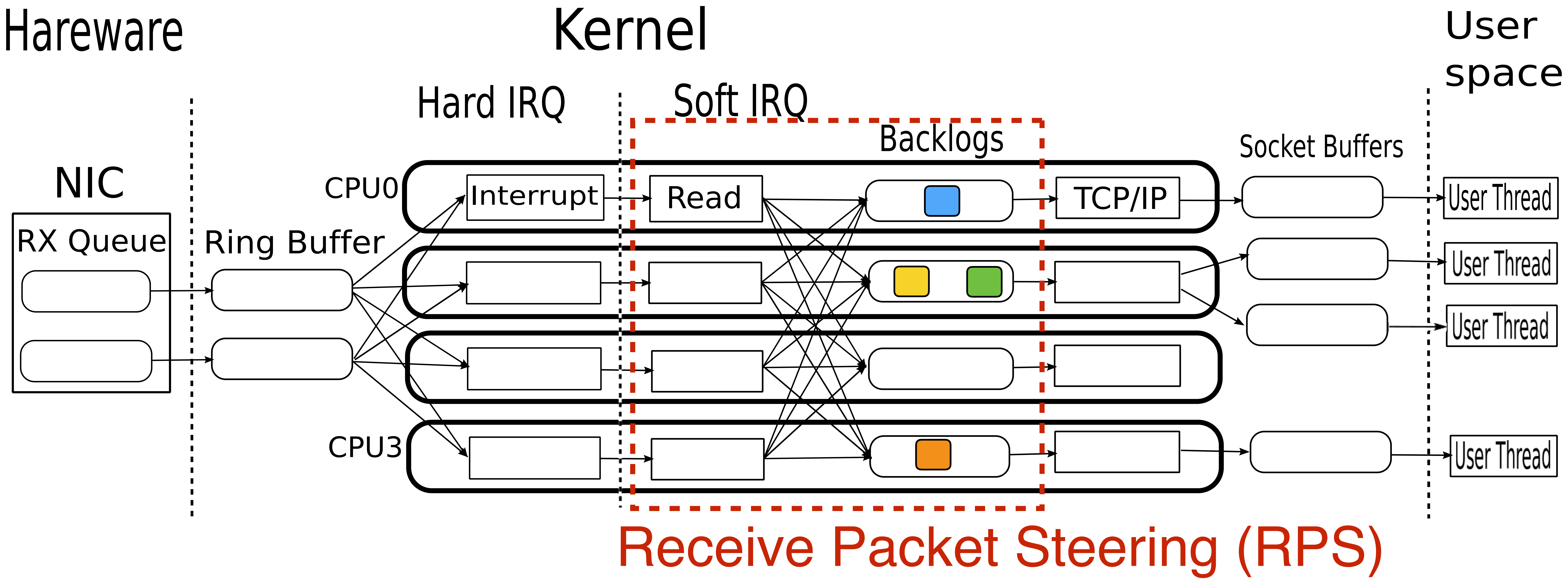



Receive Side Scaling (RSS)

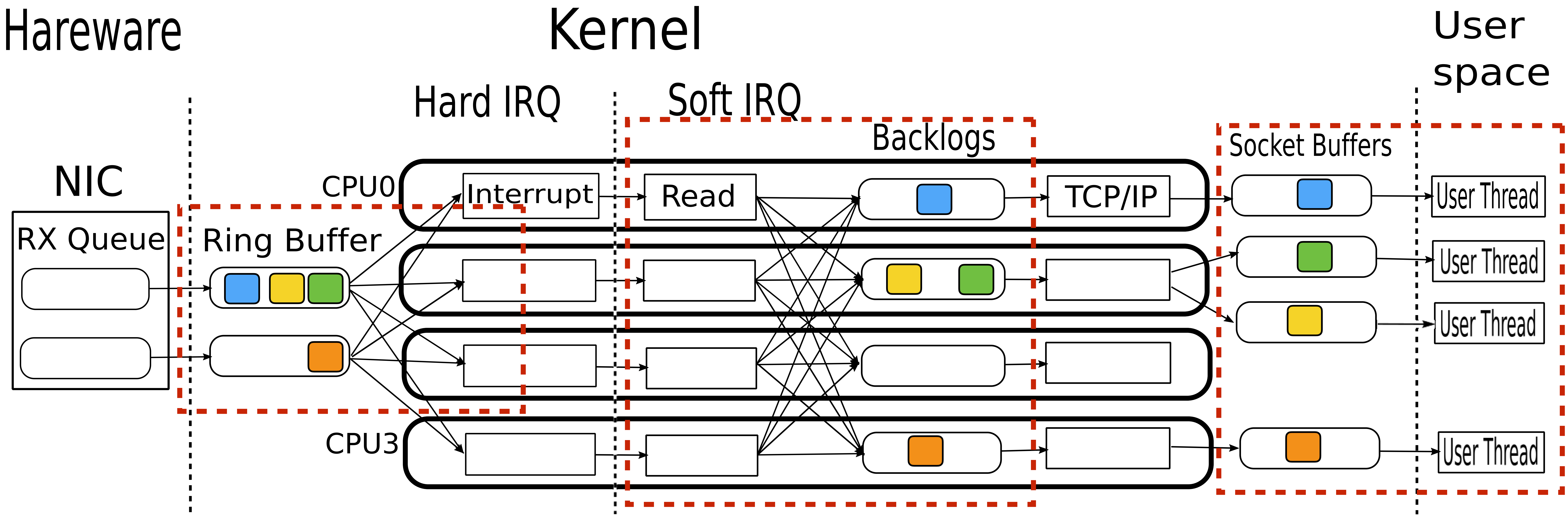
# Background: Concurrent Packet Processing



# Background: Concurrent Packet Processing

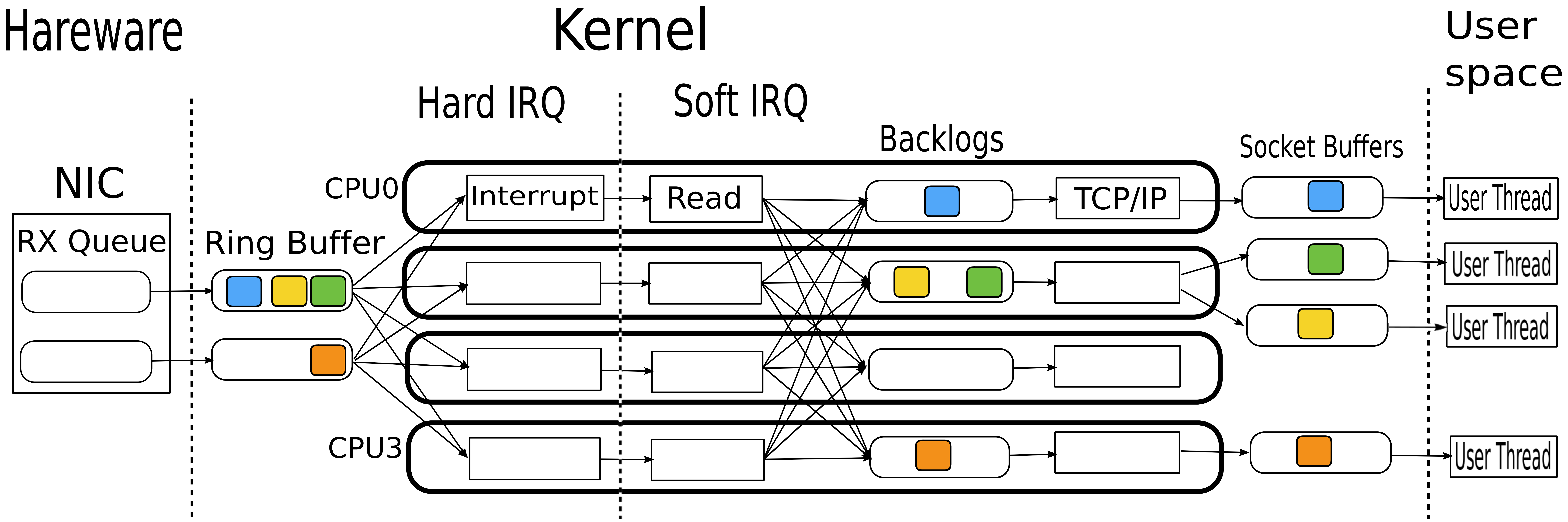


# Challenge: How to Check Pending Heartbeats?




- Multiple concurrent pipelines
- Packet Reordering

# Challenge: How to Check Pending Heartbeats?




Pause all threads and check all buffers?

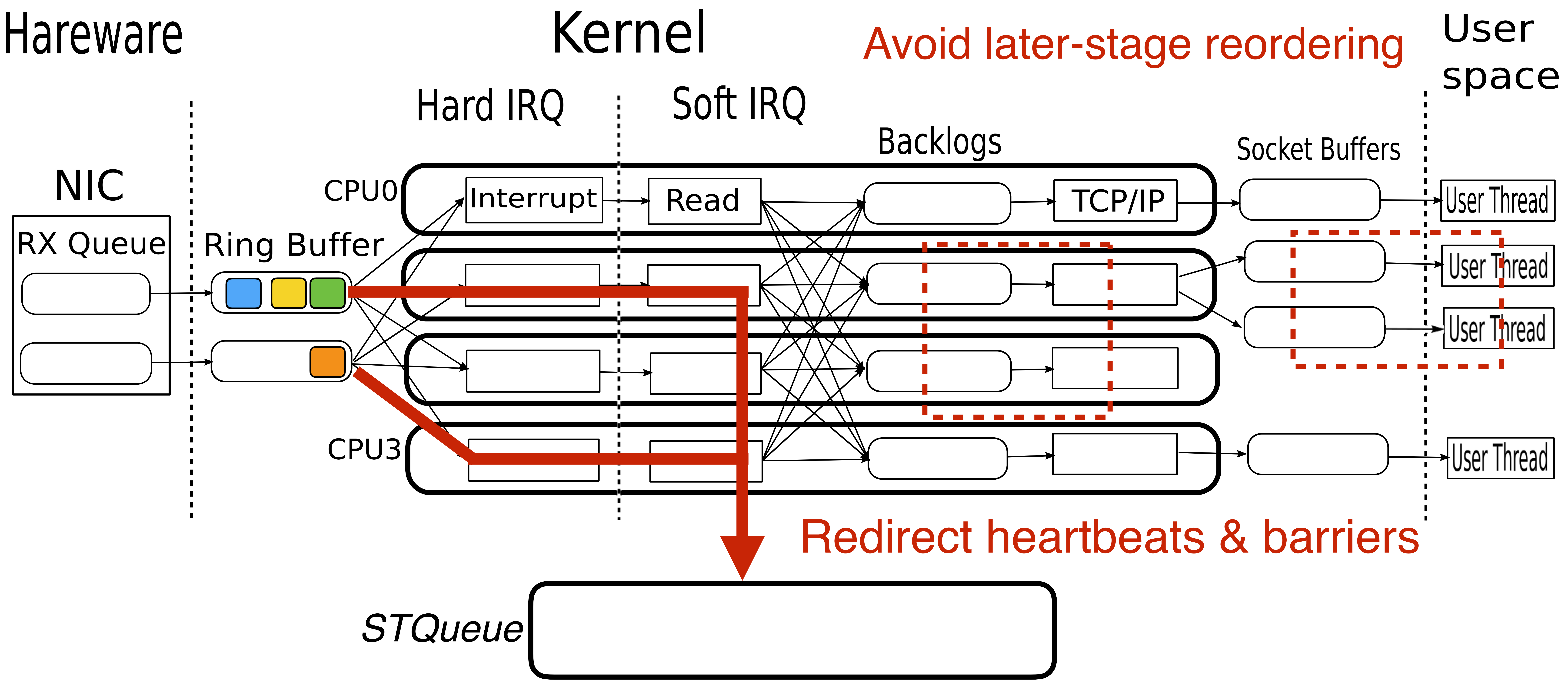
# SafeTimer's Solution: Barrier Mechanism

- Receiver sends barrier packets to **itself** when timeout
- Force heartbeats and barriers to be executed in **FIFO** order

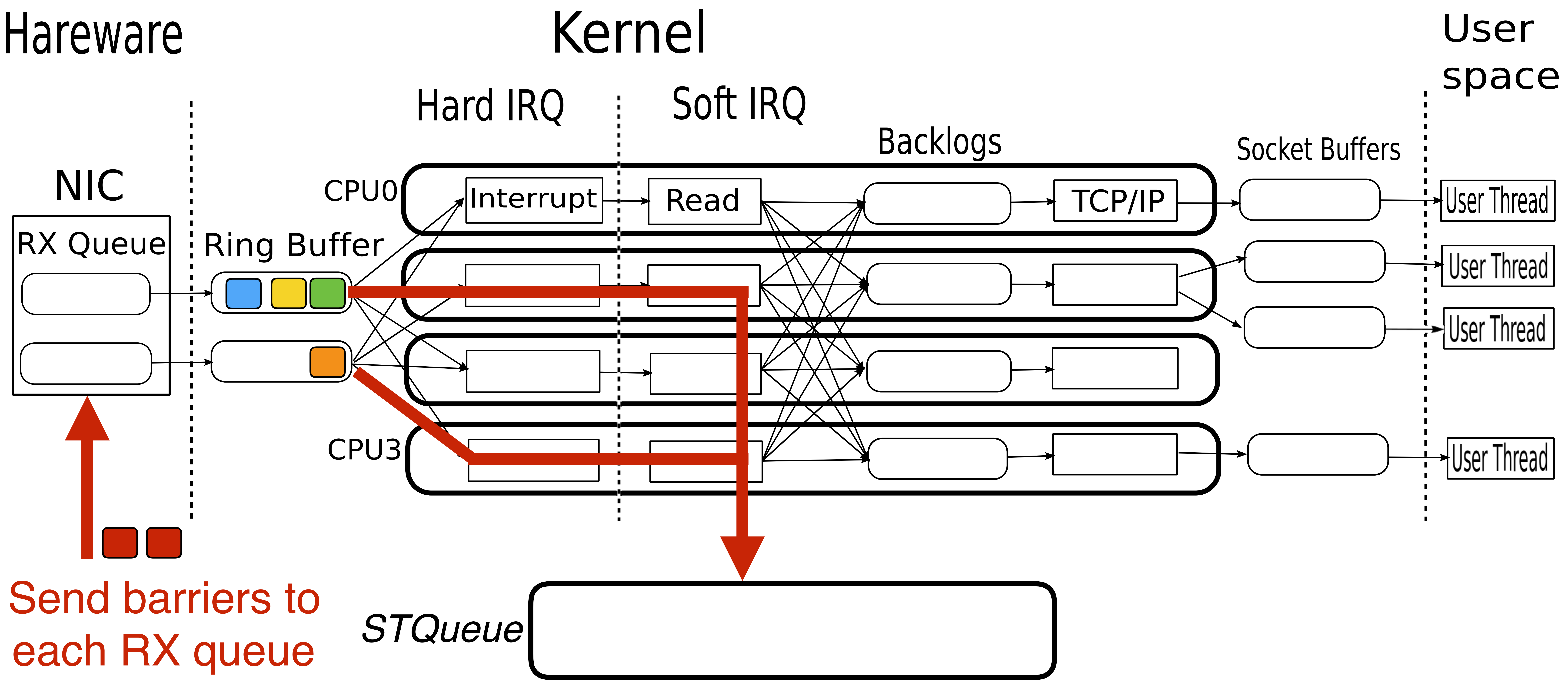
When barriers are processed =>

Heartbeats arrived before timeout must have been processed

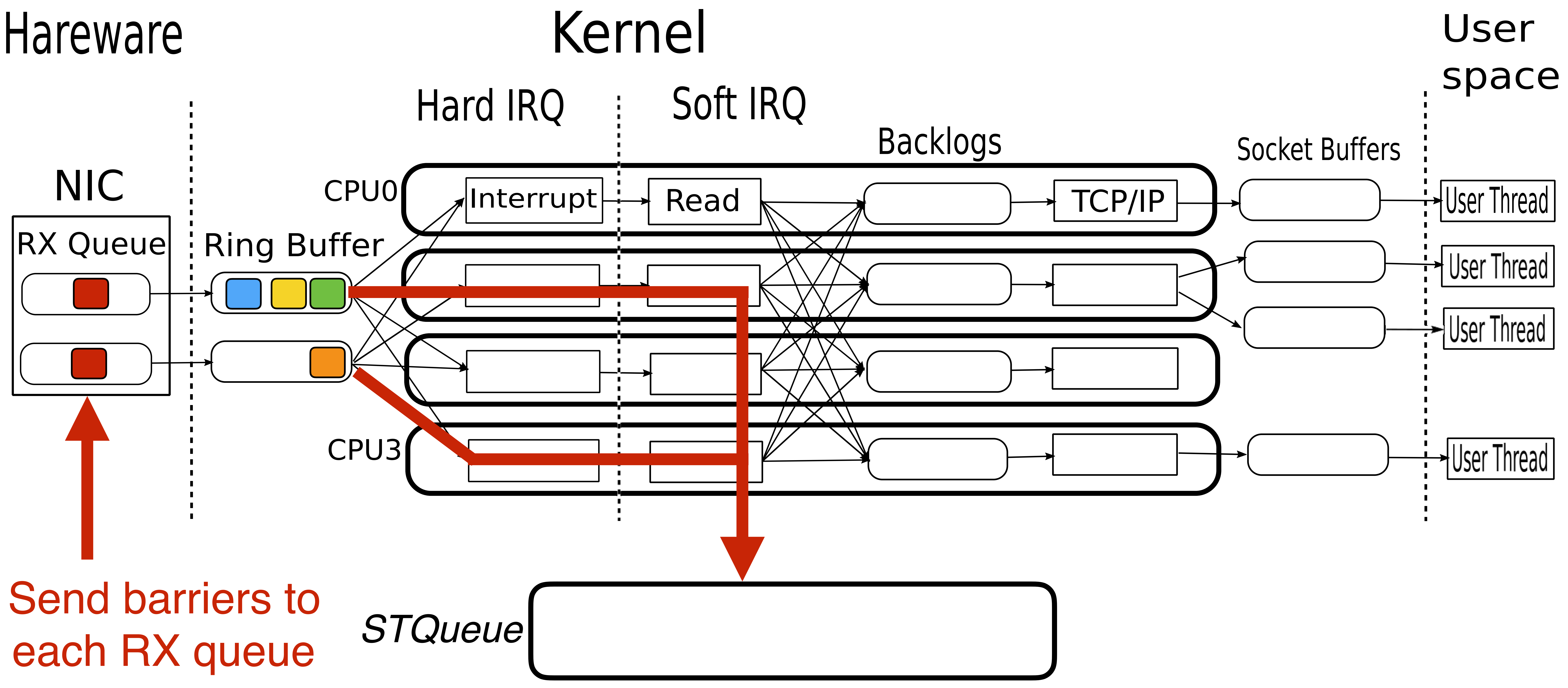
# Preserve Per-Ring FIFO Order



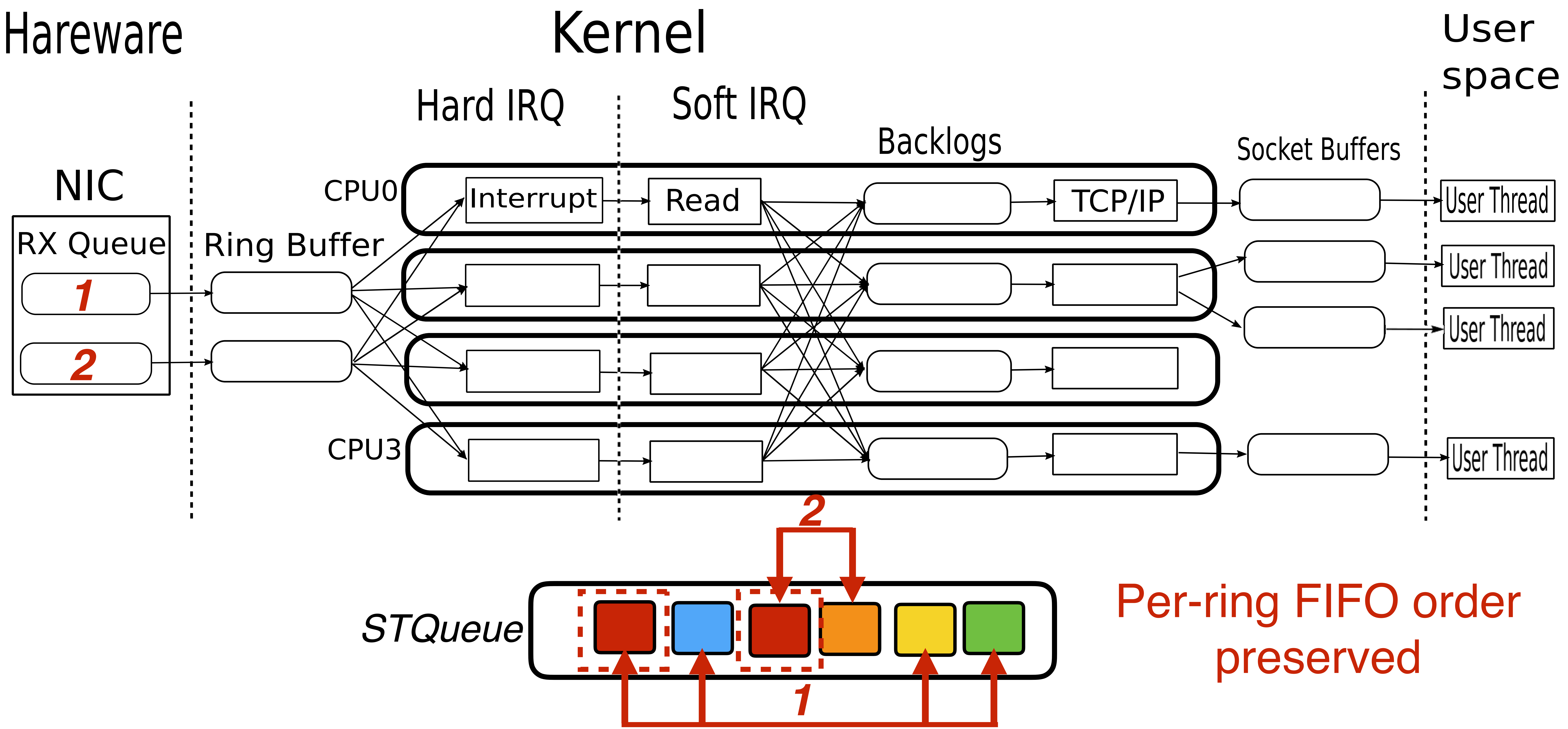
# Send Barriers to Flush Heartbeats



# Send Barriers to Flush Heartbeats



# When Barriers Processed, Heartbeat Processed




Per-ring FIFO order preserved

# Overview of SafeTimer

- **Goal**: if the receiver reports timeout, the sender must have failed
- **Assumptions** of SafeTimer
  - Delays in whitebox can be arbitrarily long
  - SafeTimer relies on existing protocol for blackbox
- **Solutions**
  - **Receiver**: check pending/dropped heartbeats when timeout occurs
  - **Sender**: blocks sender when heartbeat sending is slow

# Problems in Existing Killing Mechanism

- Killing a slow sender is not a new idea, but
- Killing operation itself can be **delayed**
- **Sender alive** for arbitrarily long after receiver reports failure

  => Accuracy will be **violated**

# Utilizing the Idea of Output Commit

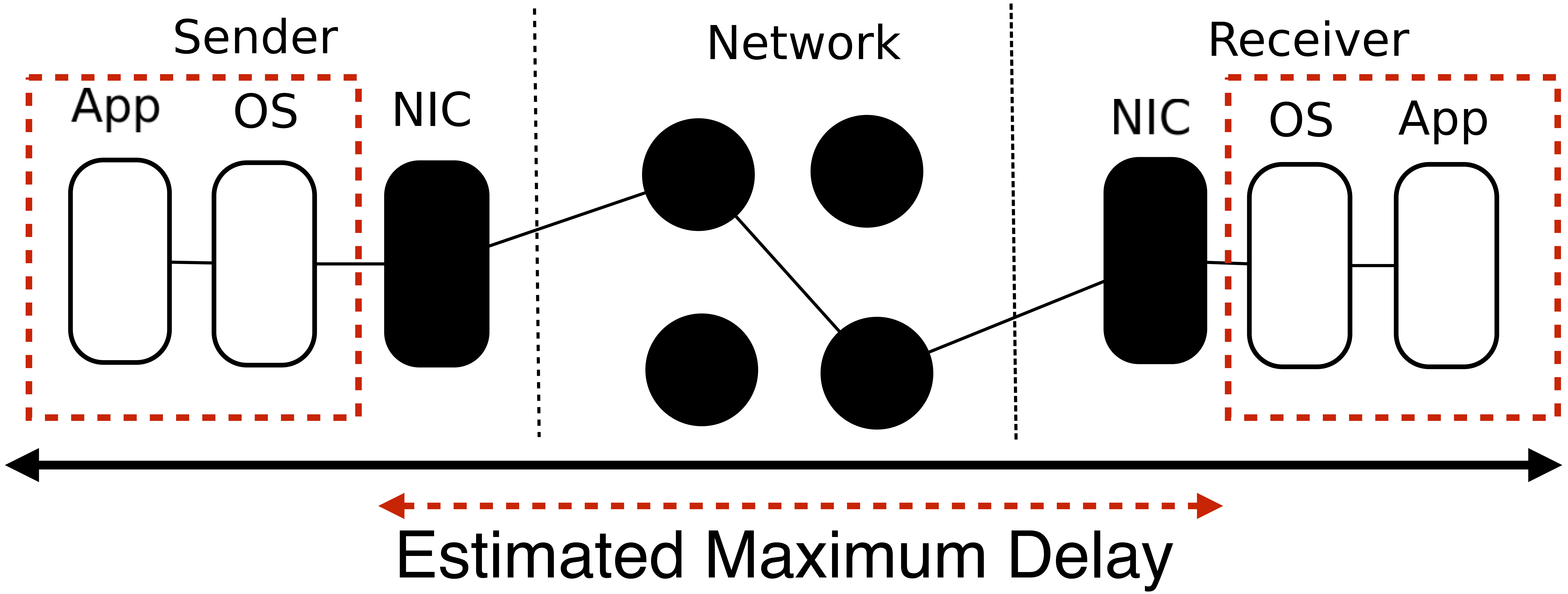
- A slow sender may continue processing
- As long as other nodes do not observe the effects, the slow sender is **indistinguishable** from a **failed** sender [Edmund, OSDI'06]

# Block Sender When It Is Slow

- Maintain a timestamp $t_{valid}$ before which sending is valid
- Extend $t_{valid}$ when sender sends heartbeats successfully
  - The definition of "success" depends on the blackbox protocol
- SafeTimer blocks sending if current time > $t_{valid}$

# No Need to Include Maximal Delay For Whitebox

- Receiver doesn’t report failure if heartbeats arrived before timeout
- Sender is blocked when sender is slow



# Implementation Overview

- Re-direct heartbeats and barriers to *STQueue*
- Send barriers to a specific RX Queue
- Force barriers to go through NIC
- Fetch real-time drop count
- Detect heartbeat sending completion
- Block slow sender

# Evaluation Overview

- Can SafeTimer achieve accuracy despite long delays in whitebox?
- What is the overhead of SafeTimer?

# Evaluation: Accuracy

- Methodology:
  - inject delay/drop at different layers
  - compare with vanilla timeout implementation
- Result:
  - SafeTimer can correctly prevent false timeout report
  - vanilla implementation violates accuracy

# Accuracy: Heartbeats Delayed/Dropped on Receiver

Sender is still alive!

| Node | Instrument Position | Injected Event | SafeTimer | Vanilla |
|---|---|---|---|---|
| Receiver | System call (recv) | Delay | No timeout | Timeout |
| Receiver | Socket (sock_queue_rcv_skb) | Delay/Drop | No timeout | Timeout |
| Receiver | NFQueue (nfqnl_enqueue_packet) | Delay/Drop | No timeout | N/A |
| Receiver | IP (ip_rcv) | Delay | No timeout | Timeout |
| Receiver | RPS (enqueue_to_backlog) | Delay/Drop | No timeout | Timeout |
| Receiver | Ethernet (napi_gro_receive) | Delay | No timeout | Timeout |
| Sender | System call (send) | Delay | Blocked | Alive |
| Sender | Socket (sock_sendmsg) | Delay | Blocked | Alive |
| Sender | IP (ip_output) | Delay/Drop | Blocked | Alive. Can observe drop. |
| Sender | Ethernet (dev_queue_xmit) | Delay | Blocked | Alive |

# Accuracy: Heartbeats Delayed/Dropped on Sender

| Node | Instrument Position | Injected Event | SafeTimer | Vanilla |
|---|---|---|---|---|
| Receiver | System call (recv) | Delay | No timeout | Timeout |
| Receiver | Socket (sock_queue_rcv_skb) | Delay/Drop | No timeout | Timeout |
| Receiver | NFQueue (nfqnl_enqueue_packet) | Delay/Drop | No timeout | N/A |
| Receiver | IP (ip_rcv) | Delay | No timeout | Timeout |
| Receiver | RPS (enqueue_to_backlog) | Delay/Drop | No timeout | Timeout |
| Receiver | Ethernet (napi_gro_receive) | Delay | No timeout | Timeout |
| Sender | System call (send) | Delay | Blocked | Alive |
| Sender | Socket (sock_sendmsg) | Delay | Blocked | Alive |
| Sender | IP (ip_output) | Delay/Drop | Blocked | Alive. Can observe drop. |
| Sender | Ethernet (dev_queue_xmit) | Delay | Blocked | Alive |

Receiver has reported timeout! ✓ ✗

# Evaluation: Performance Overhead

- Ping-Pong micro benchmark
    - **small** overhead (up to 2.7%) for small packets
    - **negligible** overhead for large packets
- Benchmarks for HDFS and Ceph
    - DFSIO and RADOS Bench
    - **negligible** overhead

# Related Work

- Synchronous systems: HDFS, Ceph, etc.
- Asynchronous systems: Spanner, ZooKeeper, etc.
- Failure detection without timeout:
    - Falcon and its following works [SOSP’11, NSDI’13, EuroSys’15]
    - Work if whole channel is a whitebox
    - Use timeout as a backup

# Related Work

- Real-time OS
    - Support: real-time scheduling; prioritized interrupts and threads, etc.
    - Guidelines: implement functions in low layers; pin memory; avoid disk I/Os, etc.
    - Still cannot provide hard real-time guarantees

# Summary

- SafeTimer achieves accurate timeout detection despite arbitrary processing delays
- Users can set shorter timeout intervals without sacrificing accuracy
- The overhead of SafeTimer is small

# The End

Questions?