Kernel-Managed User Buffers In Homa

John Ousterhout
Stanford University

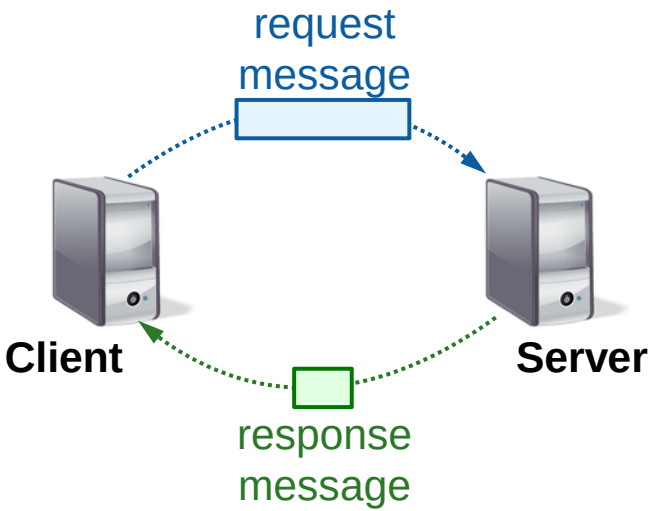
Introduction

- **Homa based on messages, not streams**
  - Good for latency, challenging for throughput
  - Traditional buffer management approach defeats pipelining
- **New approach for Homa:**
  - Kernel allocates buffers from client-supplied pool
- **Improved large-message throughput by 70% (25 Gbps network)**

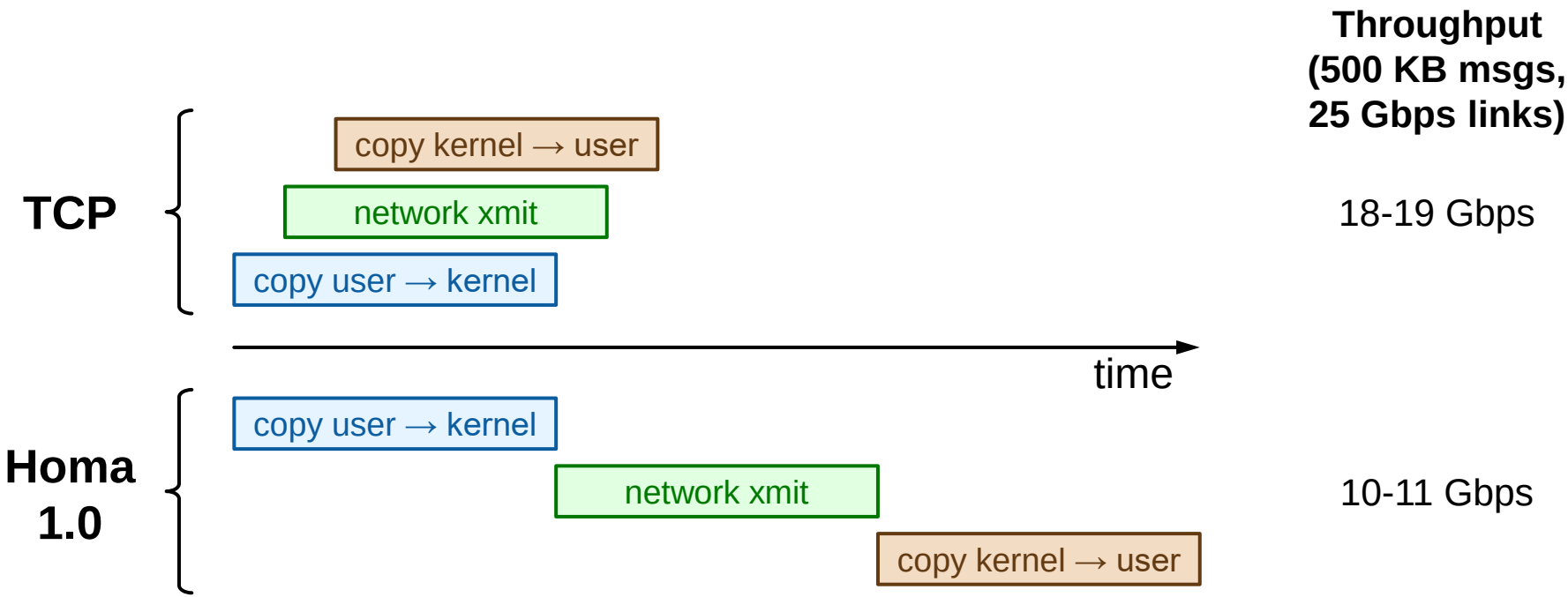
Homa Overview

Clean-slate redesign of network transport for datacenters:

- **Message-oriented (RPCs)**
- **Connectionless: one socket per application**
- **SRPT: prioritizes short messages**
- **Novel congestion control uses switch priority queues**
- **Benefit: 7–83x reduction in tail latency compared to TCP**

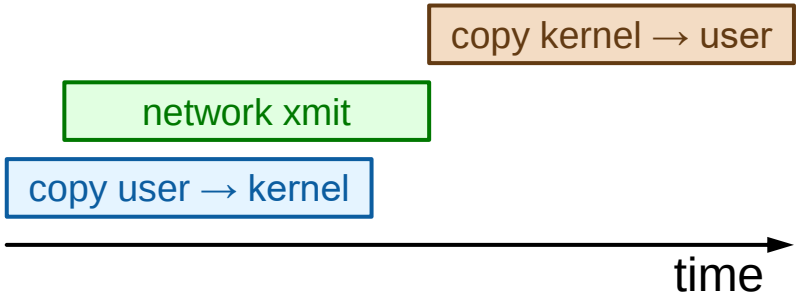


Pipelining Harder for Messages



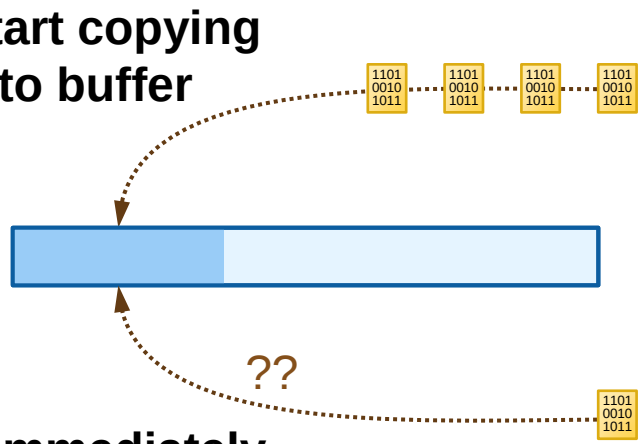
At higher network speeds, copy costs dominate

Sender-Side Not Too Hard



- **Main challenge: synchronization**
  - Must not hold RPC lock while copying
  - Without lock, RPC could be deleted while copy in progress
  - RCU not practical: time constants too long

read/recvmsg APIs Prohibit Pipelining

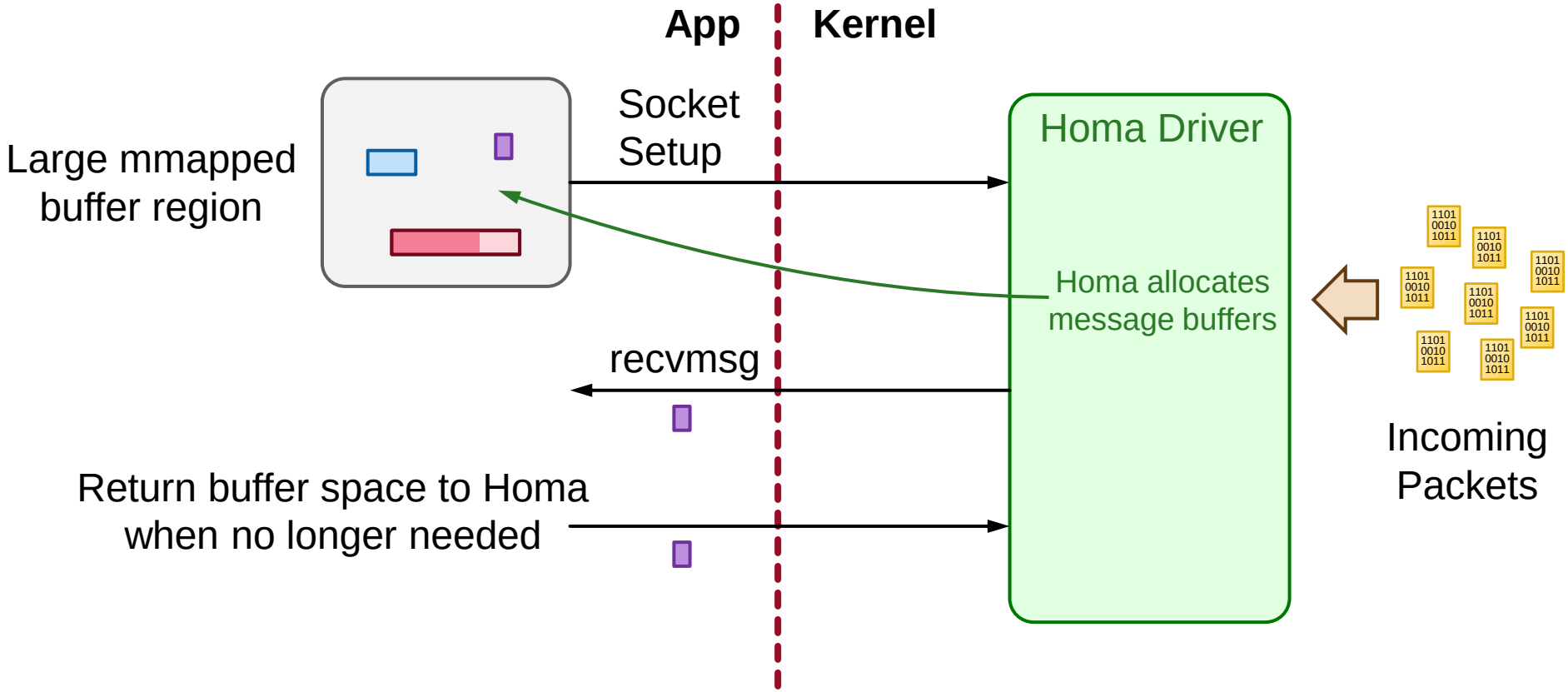


Cannot start copying until entire message received

Pipelining Requires New API

- **Homa must have buffer space for multiple incoming messages**
- **App no longer specifies the buffer when calling recvmsg**
  - Buffer is returned as result, not passed as parameter
  - Homa chooses which buffer to return

Basic Flow of Buffers

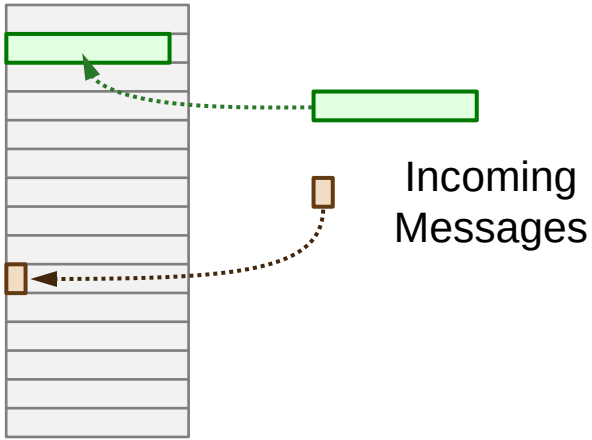


Challenges

- How to structure the buffer region?
- How to reclaim unused buffer space?
- Need high throughput for buffer allocation
- Cache/memory efficiency

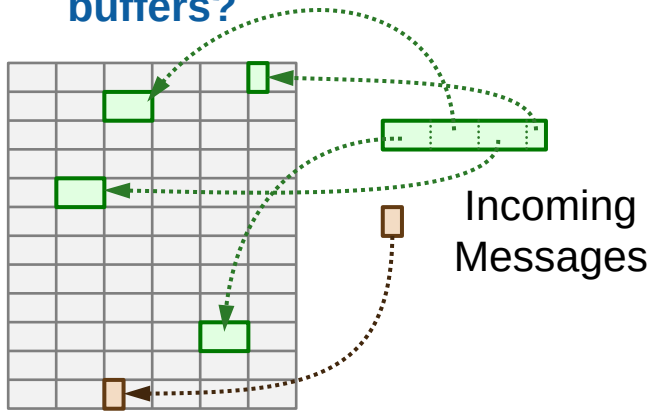
Alternatives for Buffer Structure

Array of full-size message buffers (1 MB)?



Memory inefficient, e.g. big burst of small messages

Array of packet buffers?

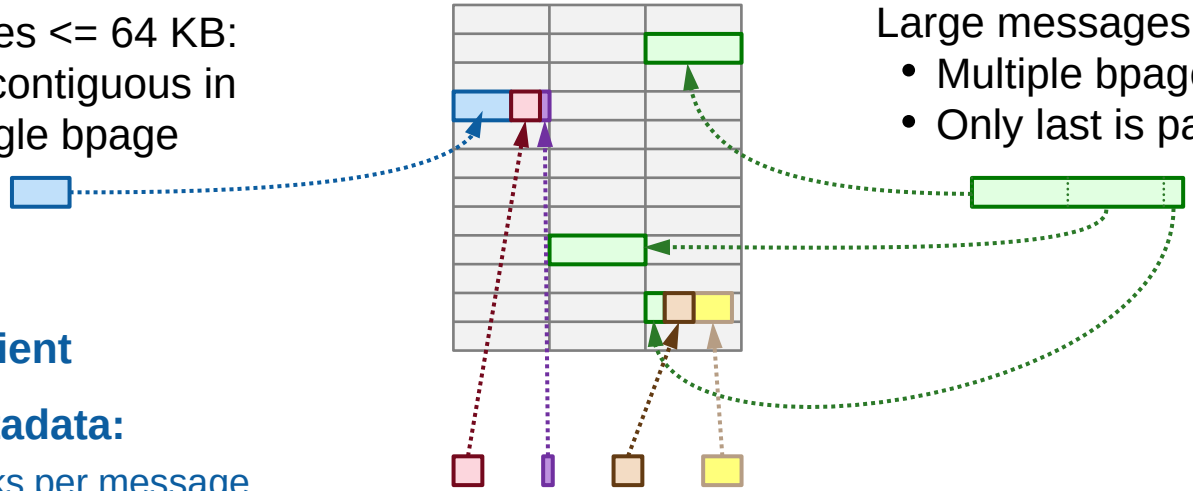


High overhead for metadata

Homa Choice: Bpages

Divide buffer region into **bpages** (64 KB)

Messages <= 64 KB: always contiguous in a single bpage



Large messages:
- Multiple bpages
- Only last is partial

Pack small messages into bpages

- ✓ **Memory efficient**
- ✓ **Reduced metadata:**
  - <= 16 chunks per message
- ✓ **Message headers always contiguous**

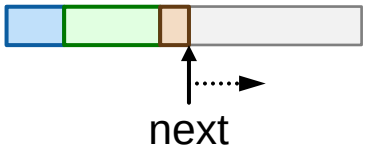
Buffer Reclamation

- **recvmsg returns pointers to message fragments**
- **Reference count per bpage: # pointers outstanding**
- **Application must eventually return pointers to Homa**
  - Arguments to recvmsg
  - Homa decrements reference count(s)
- **Homa recycles bpages when reference counts zero**

Optimizations

Per-core fragment pages:

- **For allocating small chunks**
- **No need for locking, no cache coherency**
- **Bump-a-pointer allocation**



  - Get new page if not enough space
- **Lease-based: reclaim if idle**

Memory/cache efficiency:

- **Buffer regions typically large (64 MB?)**
  - To handle worst-case scenarios
- **Homa prefers first bpages in region**
  - Later pages may never be mapped
  - Simplest case: only 2 bpages used

Buffer Exhaustion

What if buffer region fills up?

- **Allocate all bpages for a message when first packet arrives**
  - Prevents deadlock
- **If insufficient space for entire message:**
  - Queue message
  - Don’t hold any bpages
  - Discard incoming data packets
- **When bpages become available:**
  - Give to shortest queued message

Performance

Throughput (500 KB messages, 25 Gbps network):

- **Homa 1.0:** **10-11 Gbps**
- **Homa (new buffer mechanism):** **17-19 Gbps**
- **TCP:** **18-19 Gbps**

Remaining Issues

- **How large must buffer regions be? (currently 64 MB)**
- **Is 64 KB large enough for bpages?**

Conclusions

- **Linux has structured itself around TCP's stream-based model**
- **Message-based transport introduces conflicting needs**
- **One example: buffer management vs. throughput**
- **Solution: a new API for buffer management for Homa**