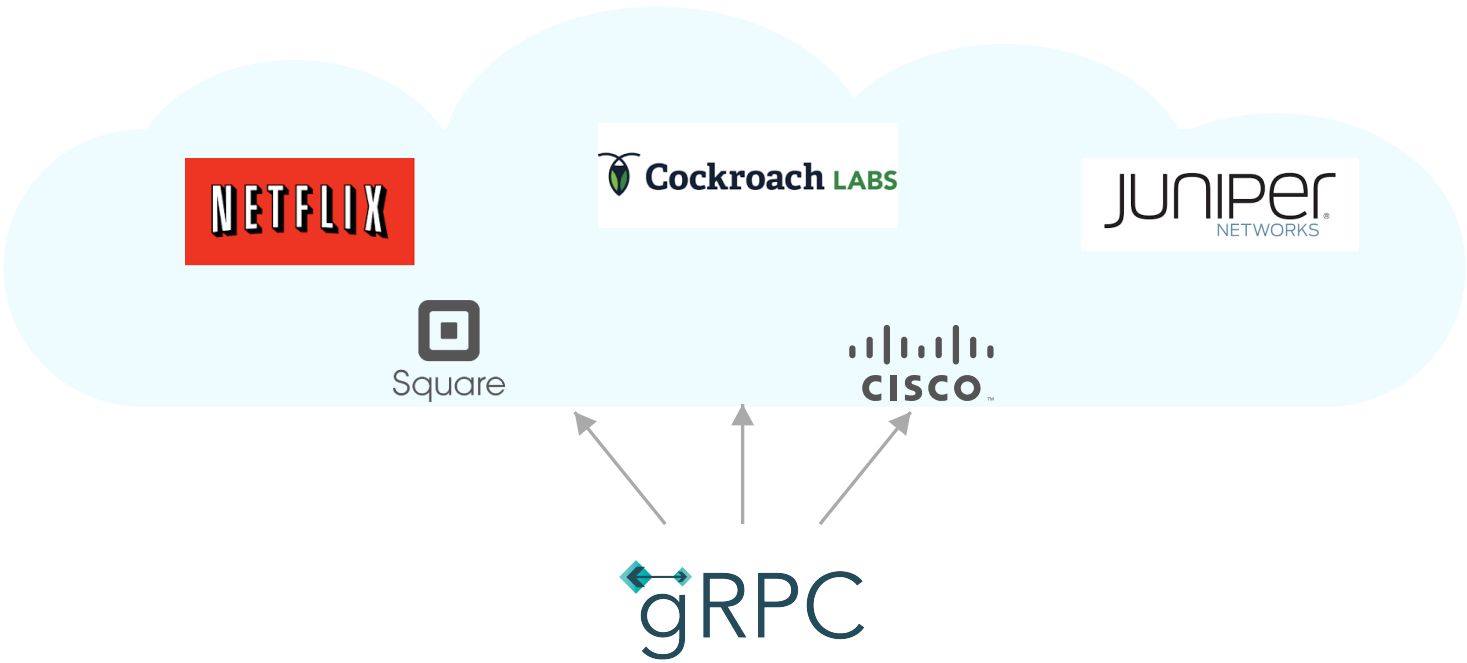
Rakaia: Scalable In-Kernel Scheduling for TCP-Based RPCs

Rui Yang Konstantinos Prasopoulos Edouard Bugnion

EPFL

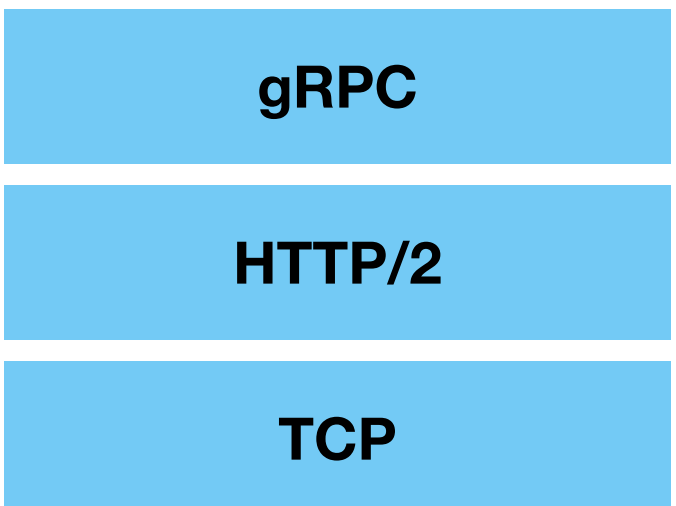
Netdev 2026

gRPC is the cornerstone for modern infrastructure



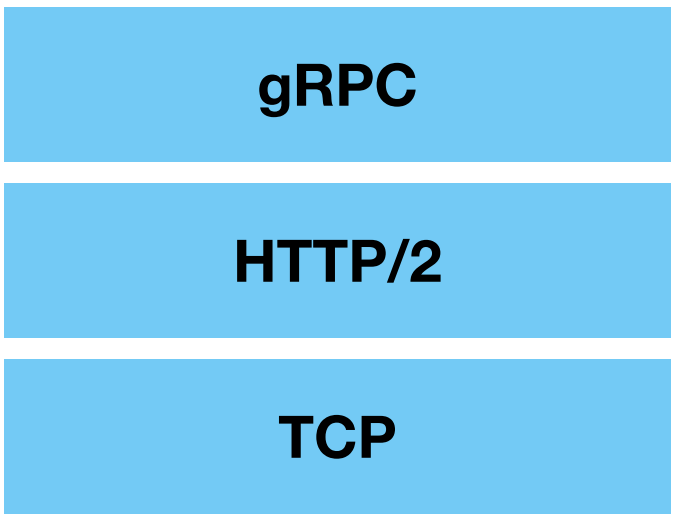
gRPC is the cornerstone for modern infrastructure

- Layered atop TCP and HTTP/2
  - compatibility and deployability
- Multi-language support
  - via generated Protobuf stubs and language libraries



Yet, we all know gRPC is “slow”[1][2]

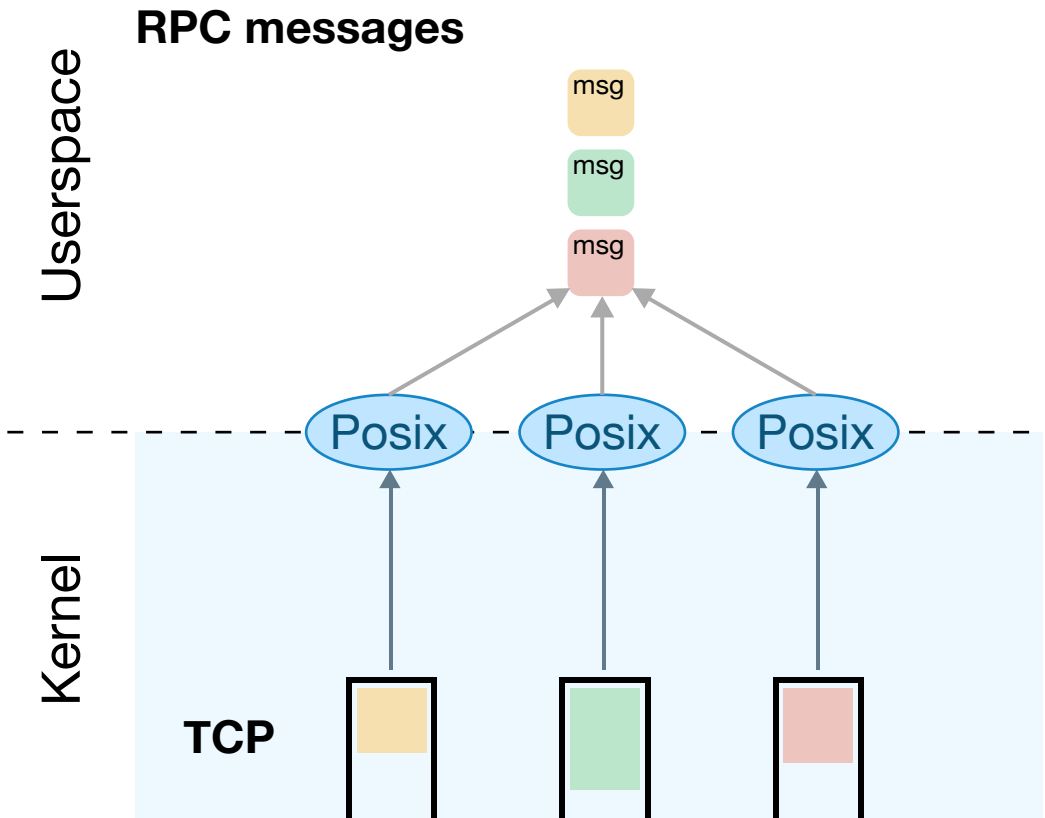
- Serialization overhead
  - Protobuf marshalling and unmarshalling
- Transport overhead
  - kernel TCP/IP and TLS
  - HTTP/2 framing and flow control



[1] Kalia, Kaminsky, and Andersen, “Datacenter RPCs can be General and Fast,” NSDI 2019.
[2] Xue et al., “Fast Distributed Deep Learning over RDMA,” EuroSys 2019.

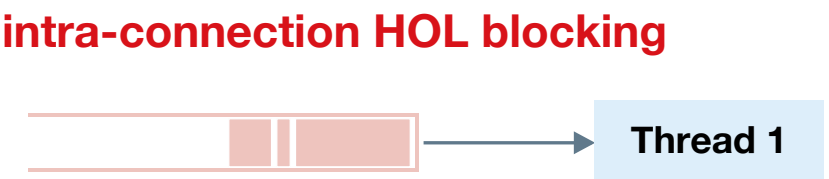
The hidden cost is its architecture design

- The POSIX API provides a stream abstraction
- RPC frameworks require a message abstraction



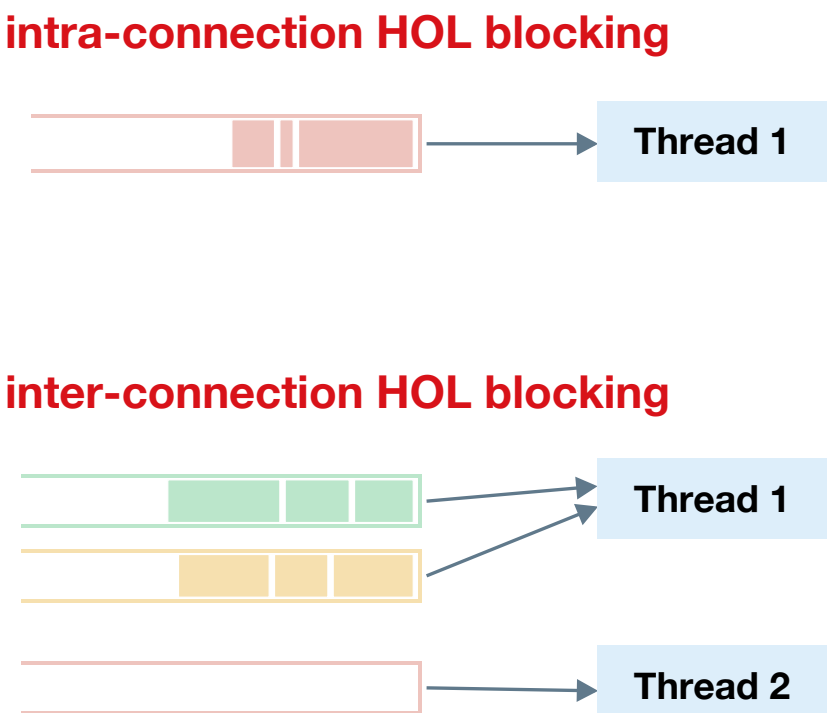
Streams create head-of-line (HOL) blocking

- Intra-connection HOL blocking
  - a slow RPC delays later RPCs on the same TCP stream



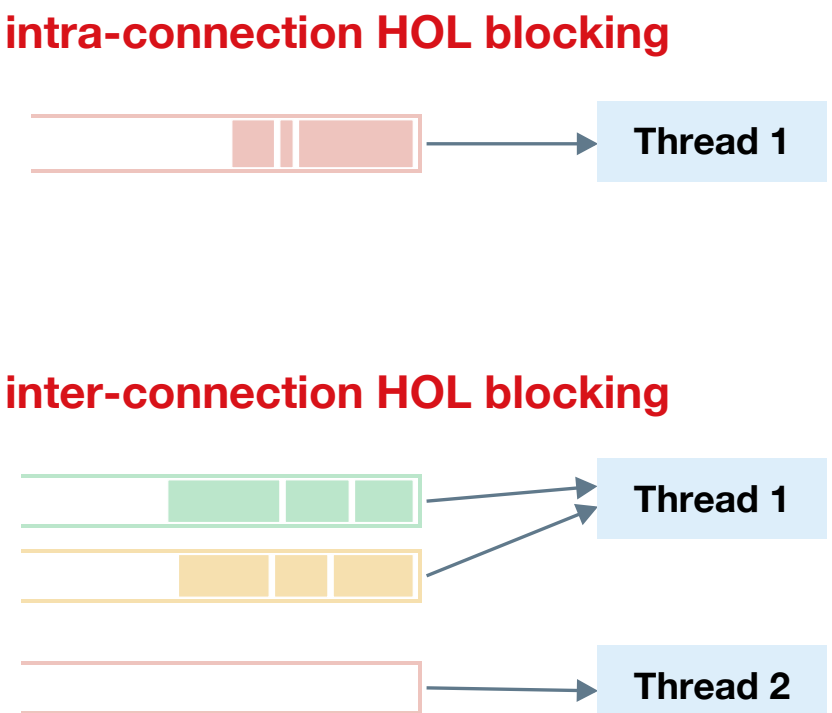
Streams create head-of-line (HOL) blocking

- Intra-connection HOL blocking
  - a slow RPC delays later RPCs on the same TCP stream
- Inter-connection HOL blocking
  - busy connections accumulate work while other cores sit idle



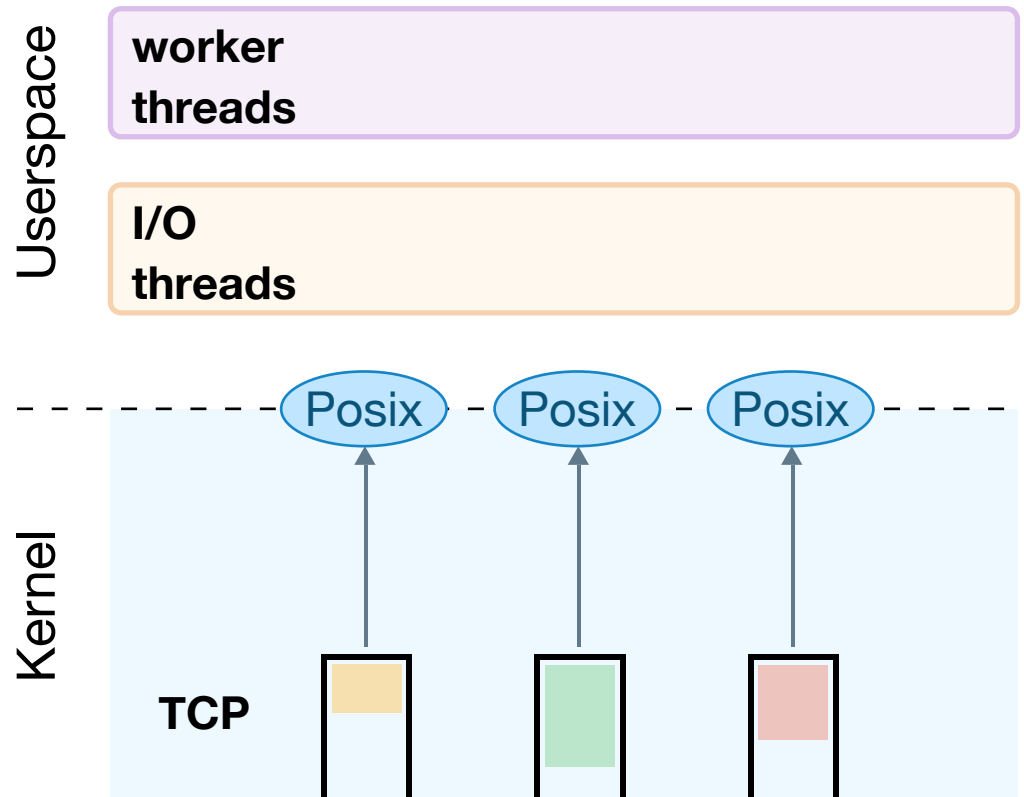
Streams create head-of-line (HOL) blocking

- Intra-connection HOL blocking
  - a slow RPC delays later RPCs on the same TCP stream
- Inter-connection HOL blocking
  - busy connections accumulate work while other cores sit idle
- RPC servers need message-level scheduling across connections and cores



The hidden cost is its architecture design

- The POSIX API provides a stream abstraction
- RPC frameworks require a message abstraction
- **gRPC Solution**: dedicated IO threads reconstruct messages and hand them to worker threads



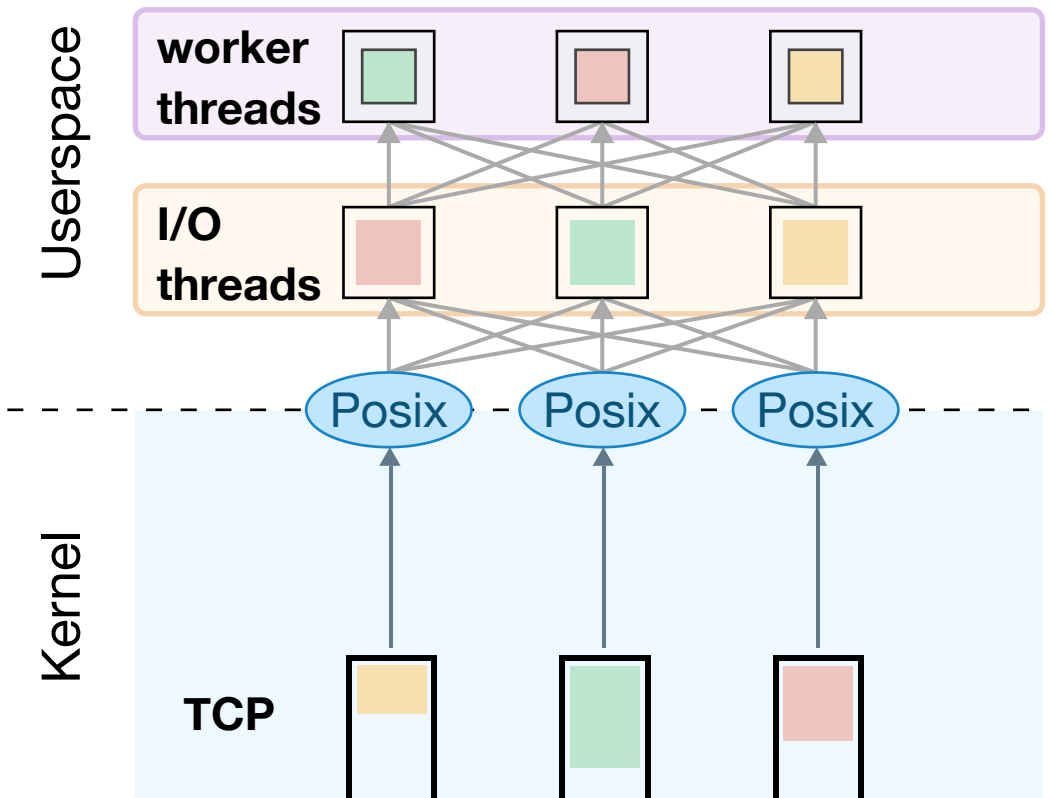
The hidden cost is its architecture design

Pros

- Eliminate both intra- and inter-connection HOL blocking

Cons

- This threading model brings overhead (e.g., context switching, locking, and scheduling)

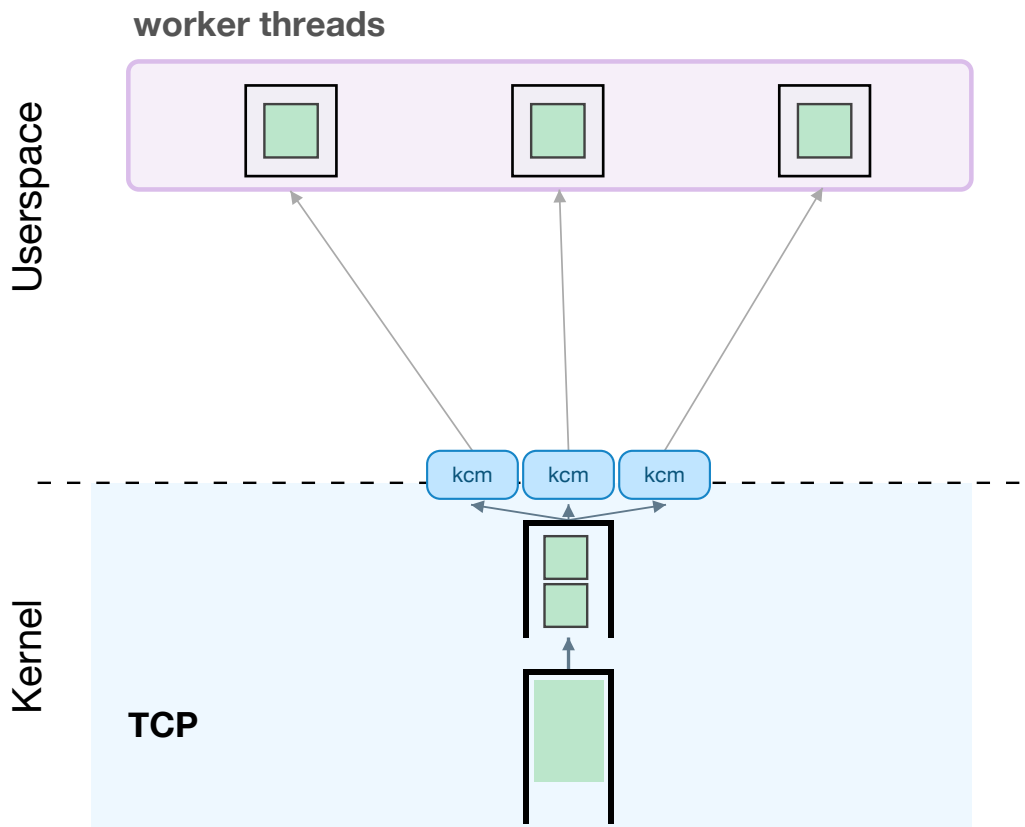


Can we keep kernel TCP/IP compatibility with HOL-free scheduling

and still achieve high-performance RPCs?

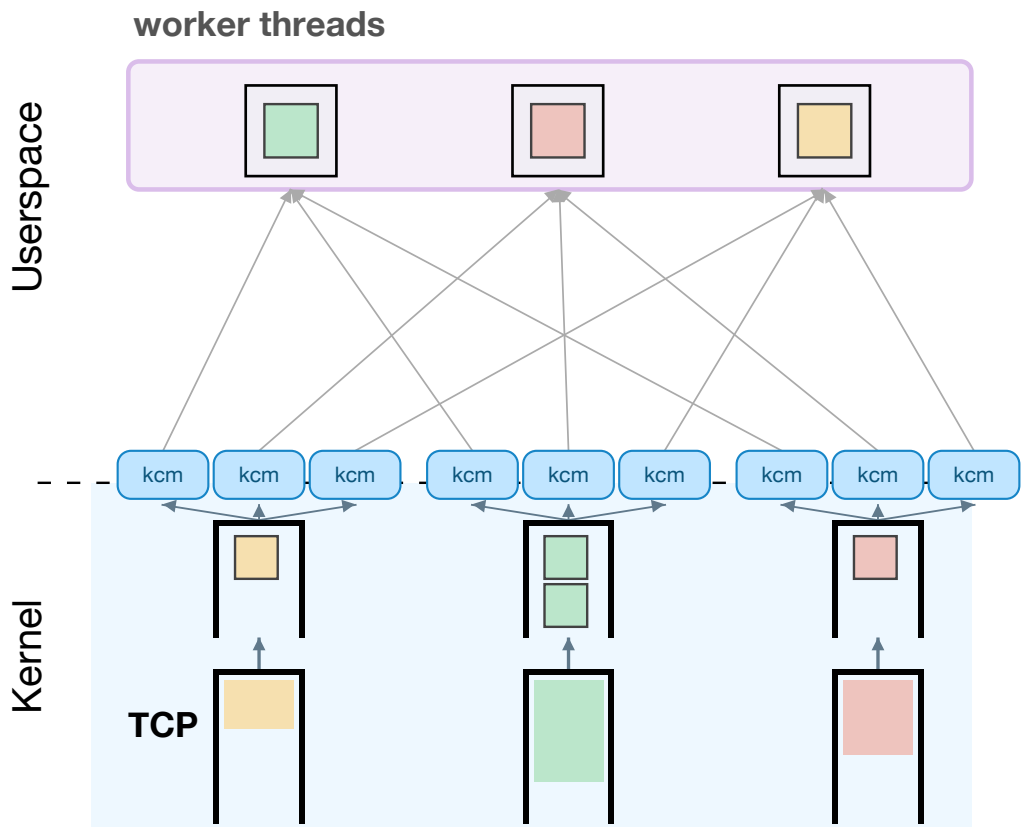
KCM points in the right direction

- Introduced into the kernel in 2015
- Parsing RPC/ULP messages in the kernel directly
  - eliminate userspace IO threads' parsing
- Every worker thread opens a KCM socket per connection
  - allow concurrent processing of RPCs from the same TCP stream



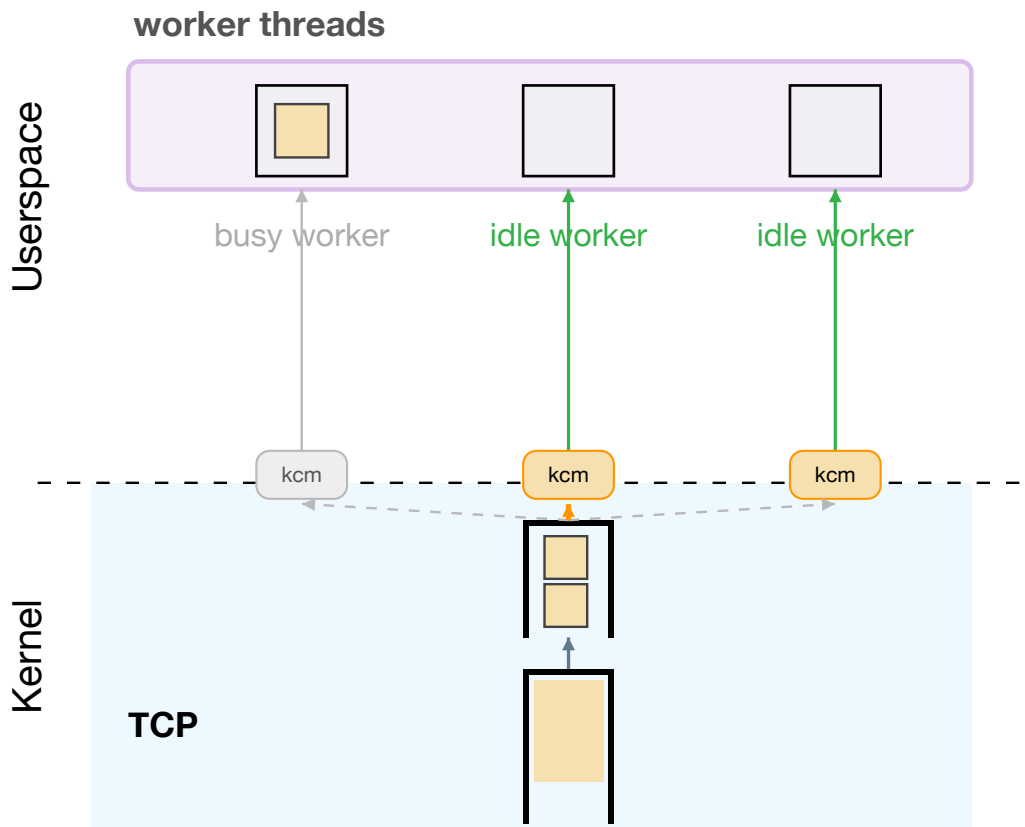
KCM Issue 1: socket scalability

- Each TCP connection attaches to KCM
- To let every worker receive from every connection:
  - fan out one KCM socket per worker
- Socket state grows with workers × connections
  - quadratic growth in #sockets.



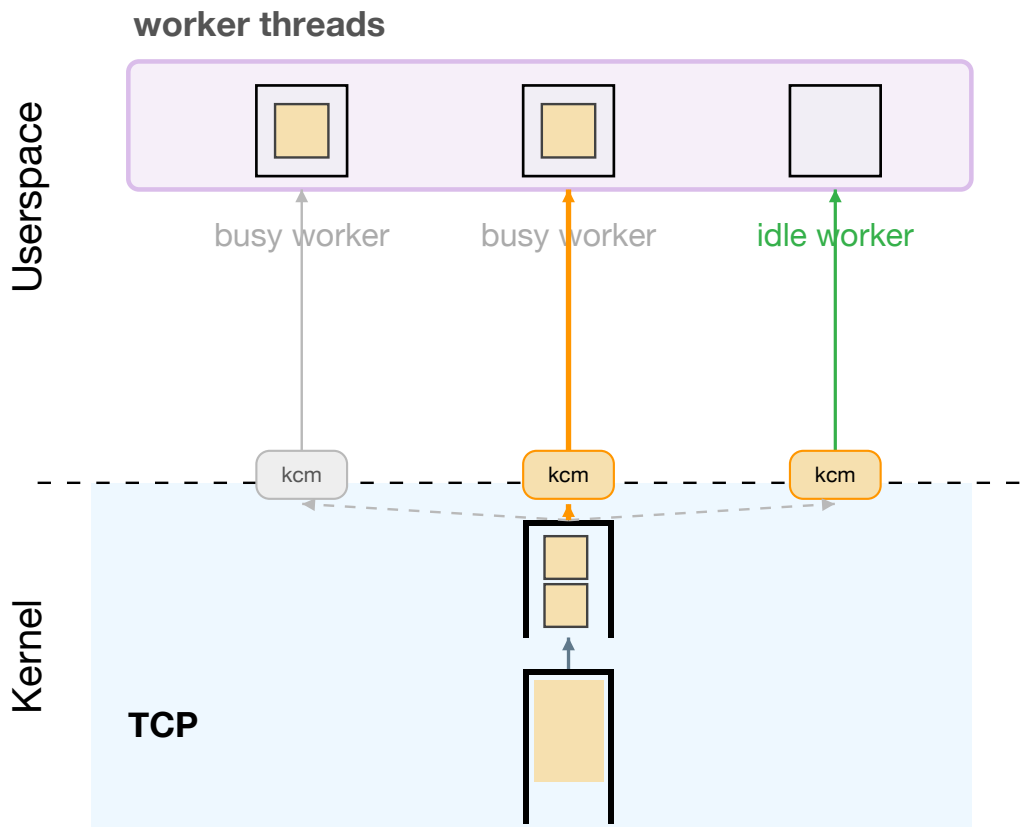
KCM Issue 2: connection-oriented scheduling

- The yellow TCP connection is the only active connection



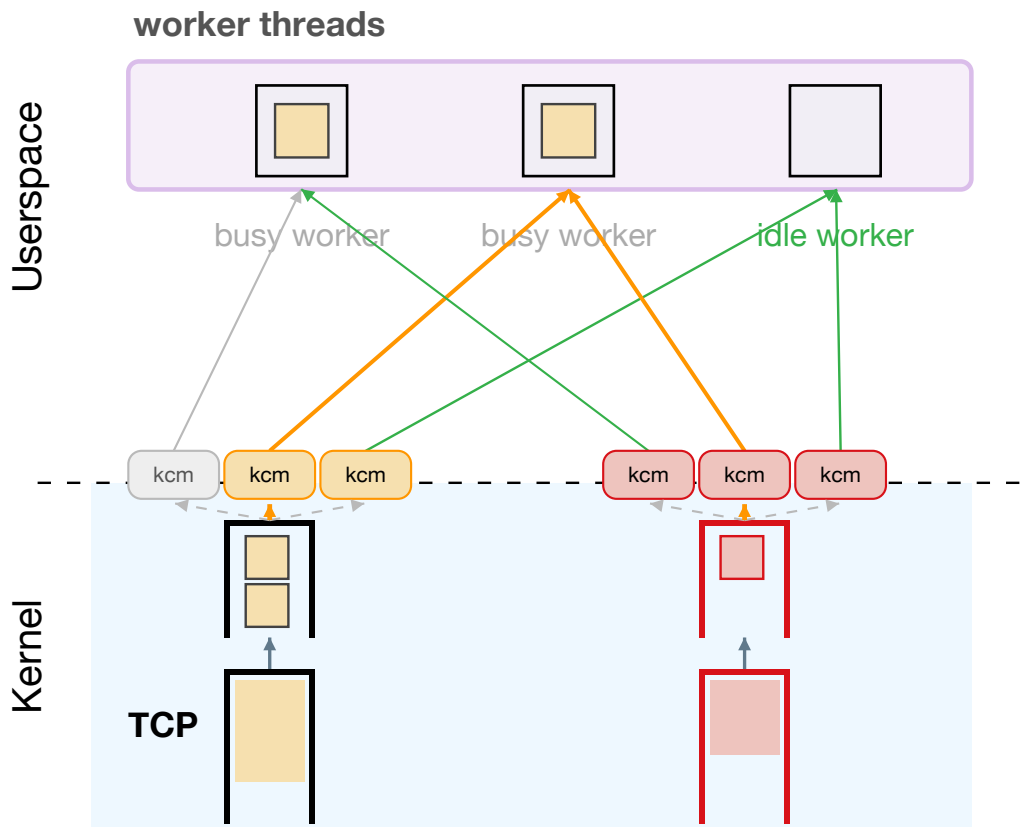
KCM Issue 2: connection-oriented scheduling

- The yellow TCP connection is the only active connection
- It chooses the KCM socket for the middle worker
  - messages are only queued at idle KCM sockets
- With one connection, this is still work-conserving



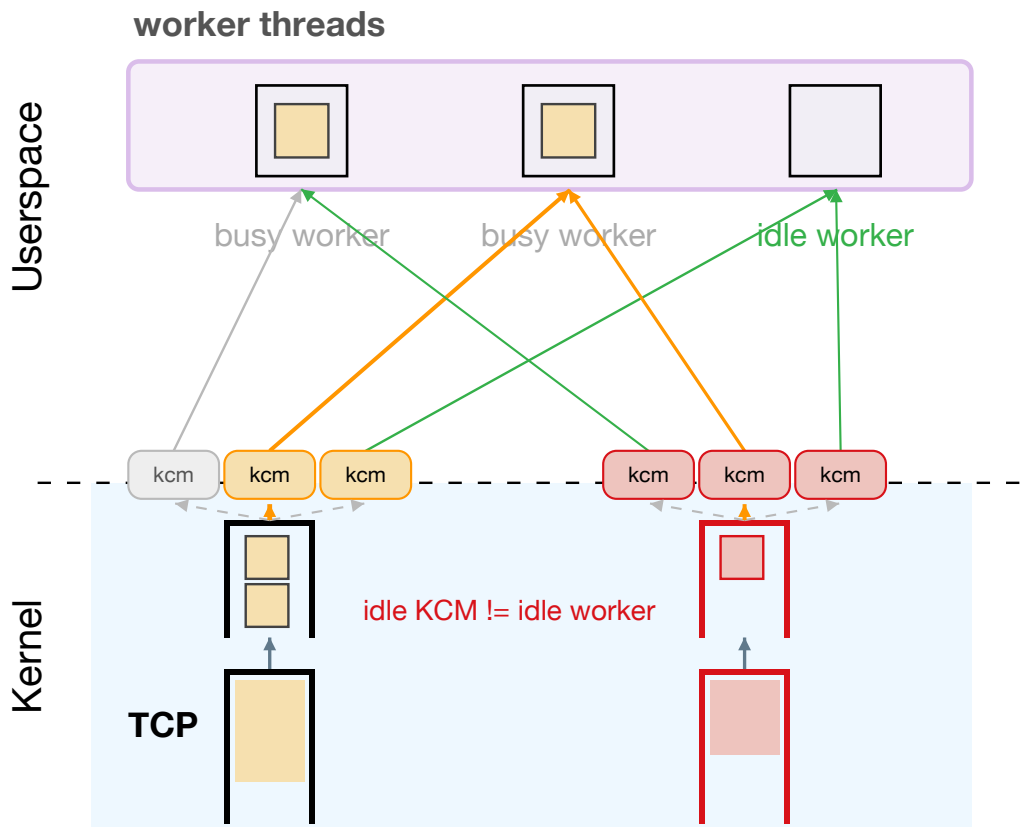
KCM Issue 2: connection-oriented scheduling

- A second TCP connection arrives
- It also chooses a KCM socket for the middle worker
  - scheduling is local to each connection
- Now both connections target the same worker
  - even though one worker thread is idle

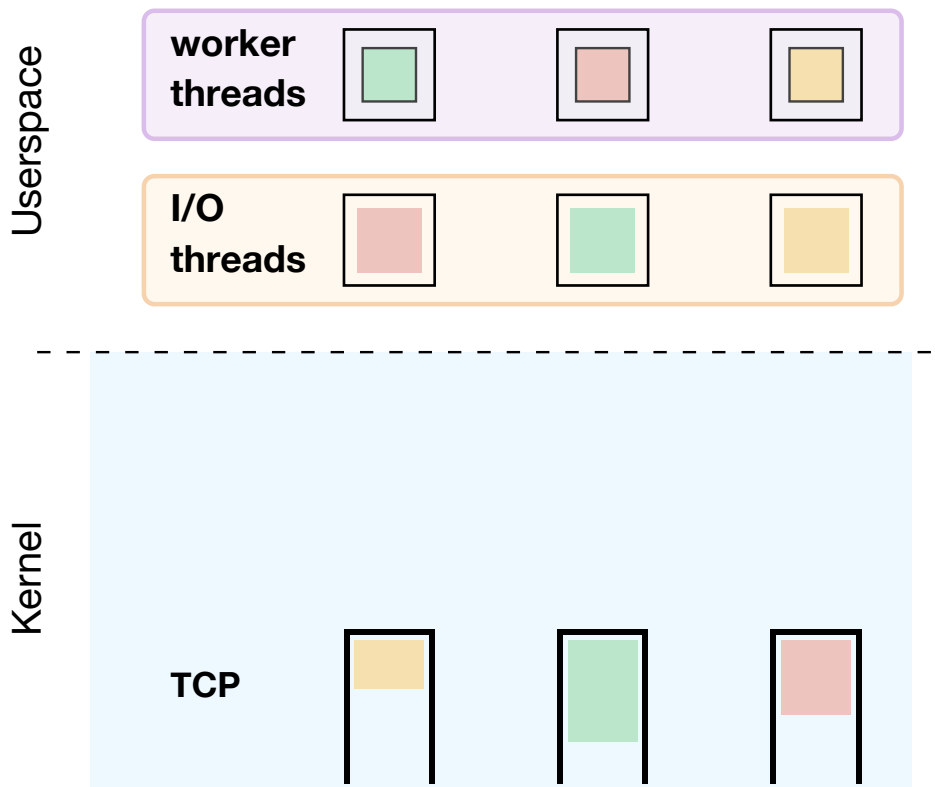


KCM Issue 2: connection-oriented scheduling

- A KCM socket can be idle, even when its worker thread is busy
- Result: not work-conserving across connections

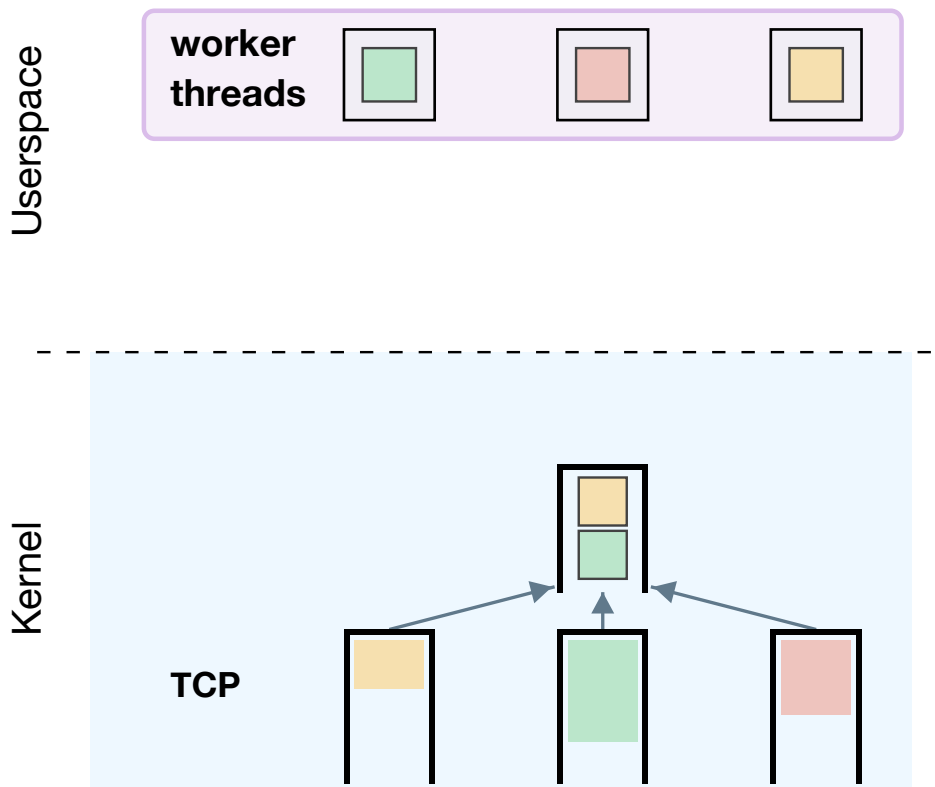


Rakaia design



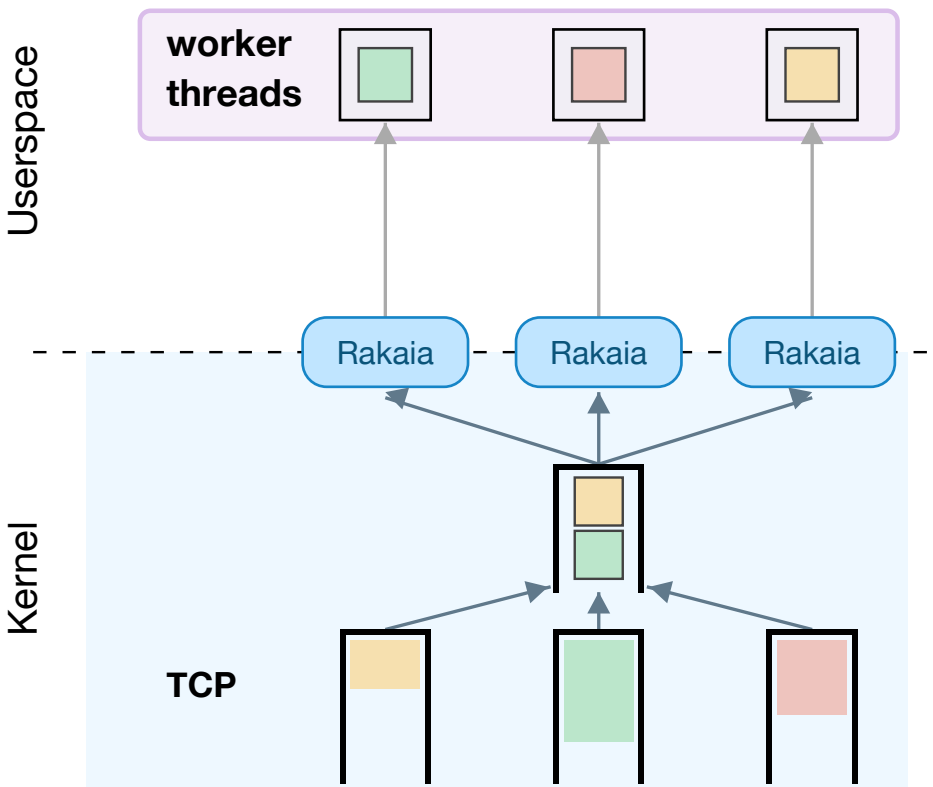
Rakaia design

- Message parsing should be done in the same context that processes TCP packets.
  - kick IO threads out of the picture
  - message-oriented API (e.g., Homa)



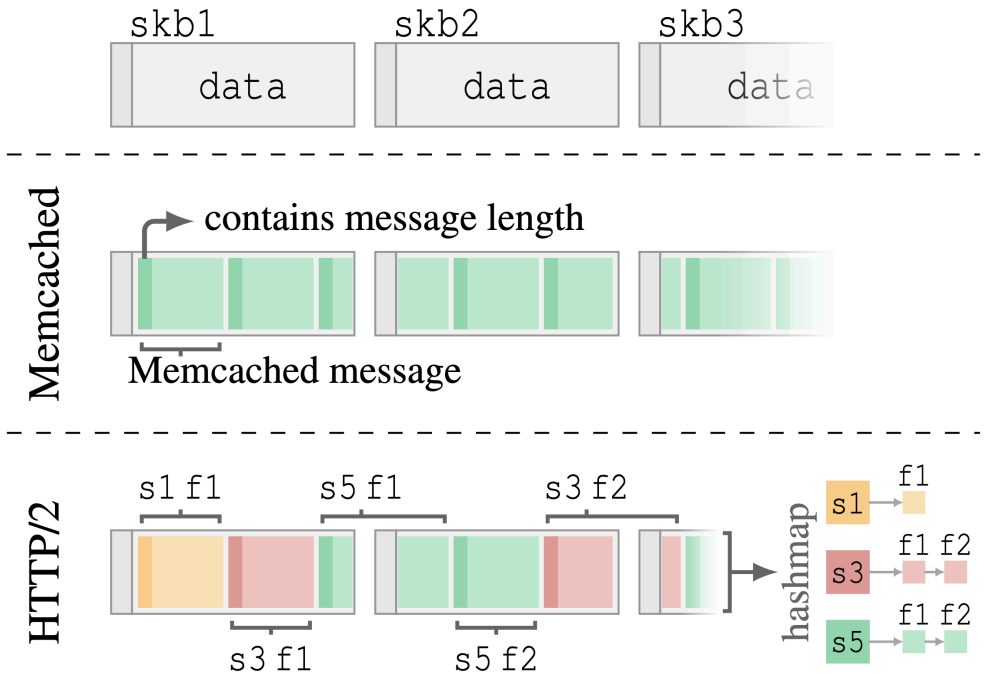
Rakaia design

- Message parsing should be done in the same context that processes TCP packets.
  - kick IO threads out of the picture
  - message-oriented API (e.g., Homa)
- Connection abstraction should be hidden from the application-facing API.
  - allow message-oriented scheduling



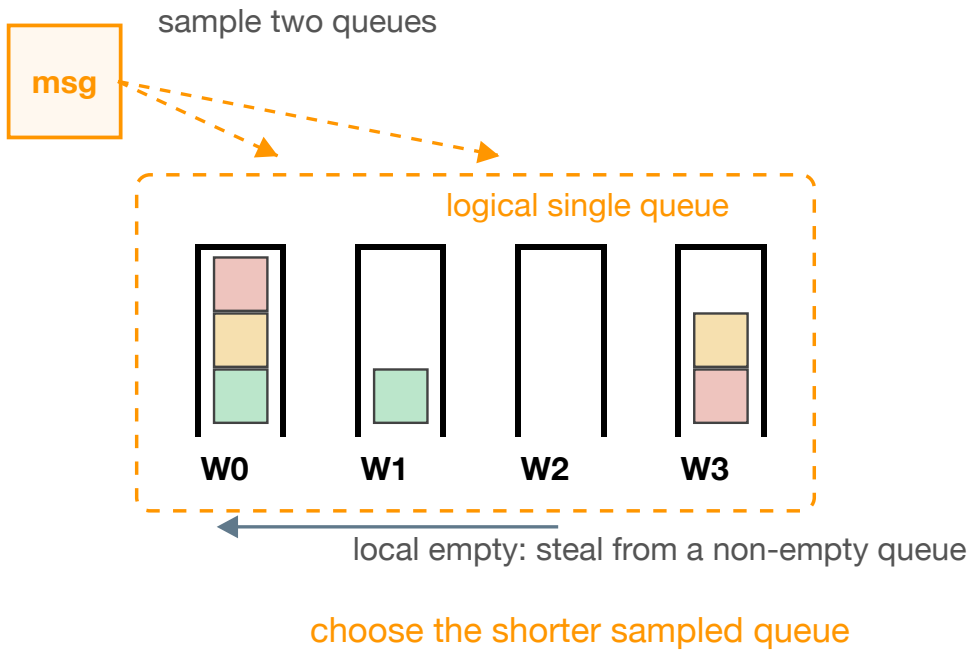
Extending strparser

- default strparser for simple ULP over TCP
  - e.g., Memcached
- extended strparser for RPC over HTTP/2
  - parse HTTP/2 frames and RPC messages



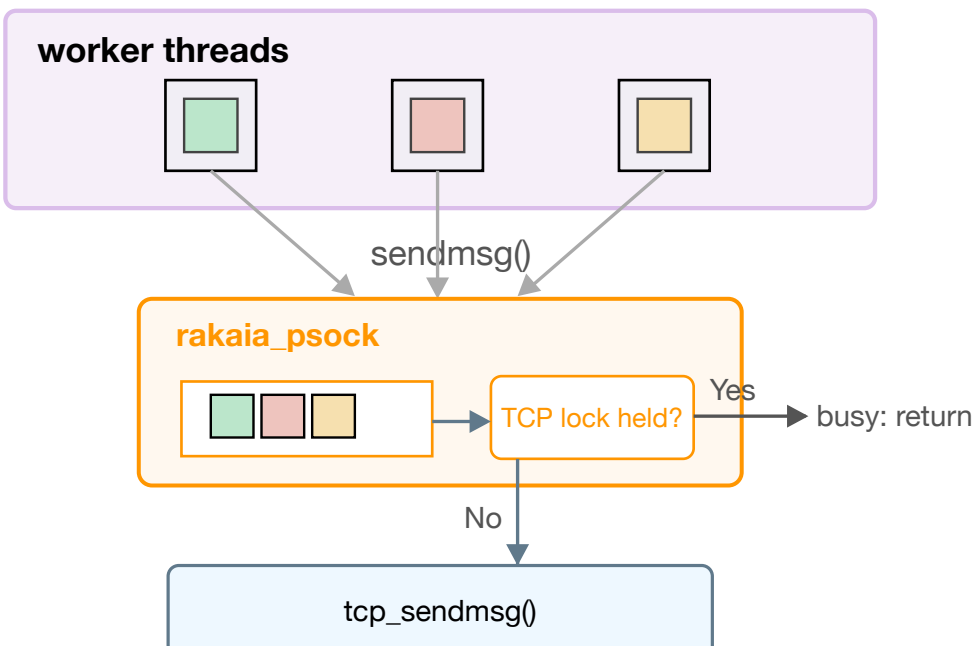
Logical single queue

- One logical message queue
  - physically: one FIFO per worker socket
- Best of two for message placement
  - sample two queues, pick the shorter one
- Idle workers get direct handoff
- Empty workers steal available work



TX delegation

- Responses still go out through the original TCP connection
- Each connection has a small virtual queue
  - senders enqueue without taking TCP lock
- If TCP lock is held, the sender returns
- Otherwise, it becomes the delegate and drains to TCP



kTLS integration

- **Issue**: kTLS decrypts in recv(), but Rakaia parses messages in the TCP data path
- **Temporary solution**: dump Rakaia rx path into a kernel workqueue, which runs in the same context as kTLS decryption processing
- **Drawback**: introduces performance degradation

Implementation and Evaluation

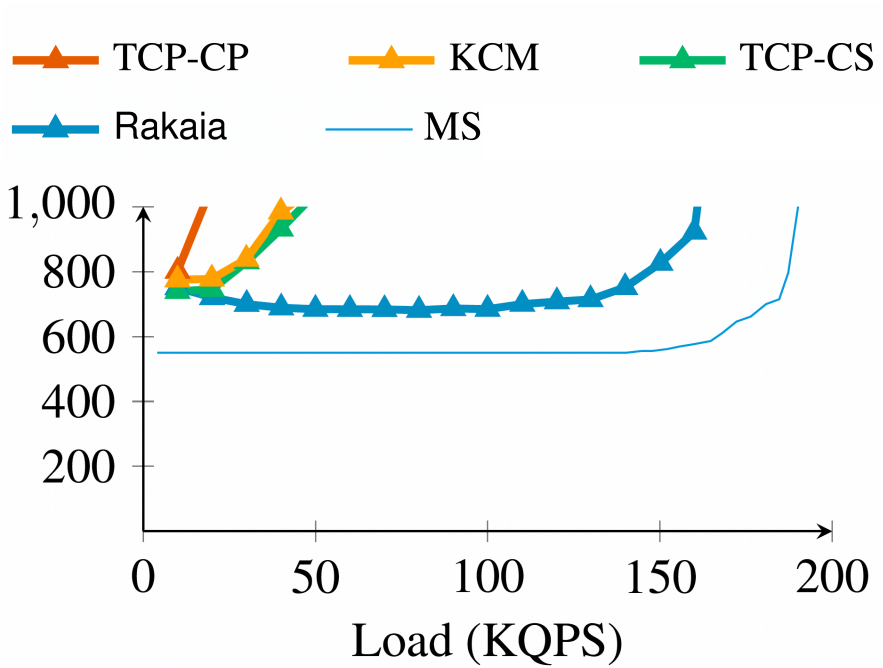
Implementation

- Linux 6.8 kernel module
- 3k-line C code and 60-line kernel patch
- extends Linux strparser for RPC framing
- compatible with default TCP/IP stack and kTLS

Workloads

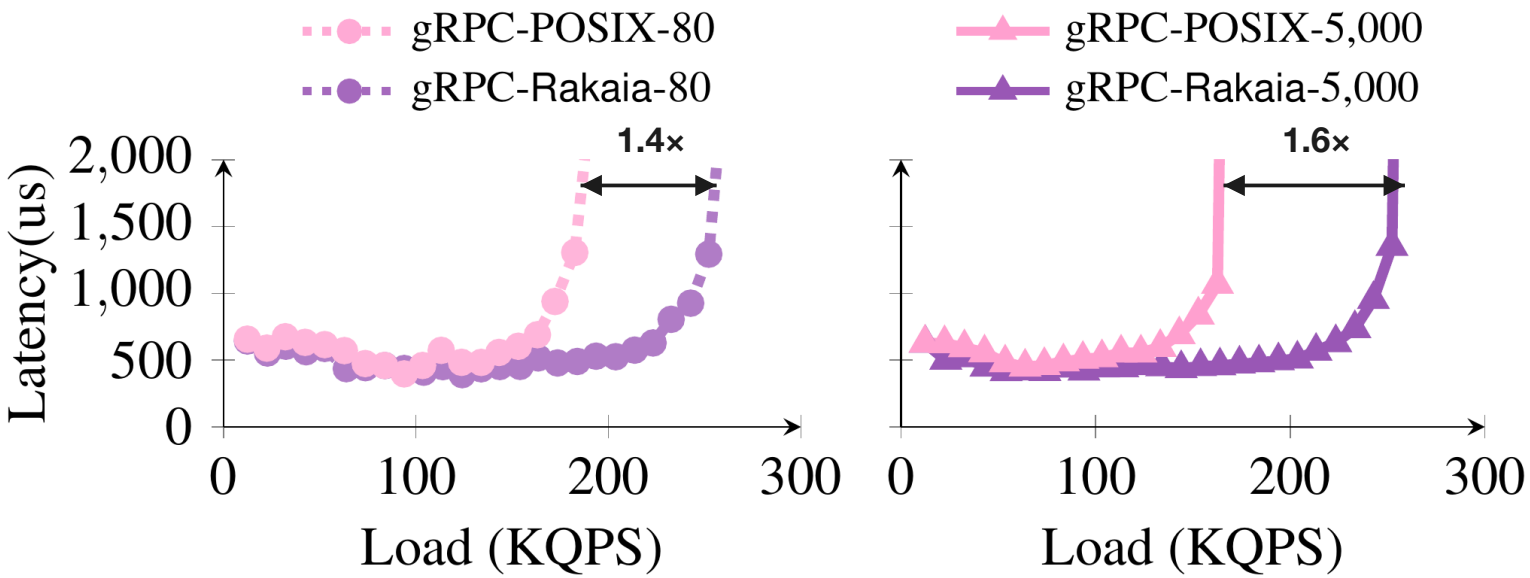
- plain-TCP RPCs
- gRPC-Go and gRPC-C++
- both synthetic and real applications

Plain-TCP RPCs



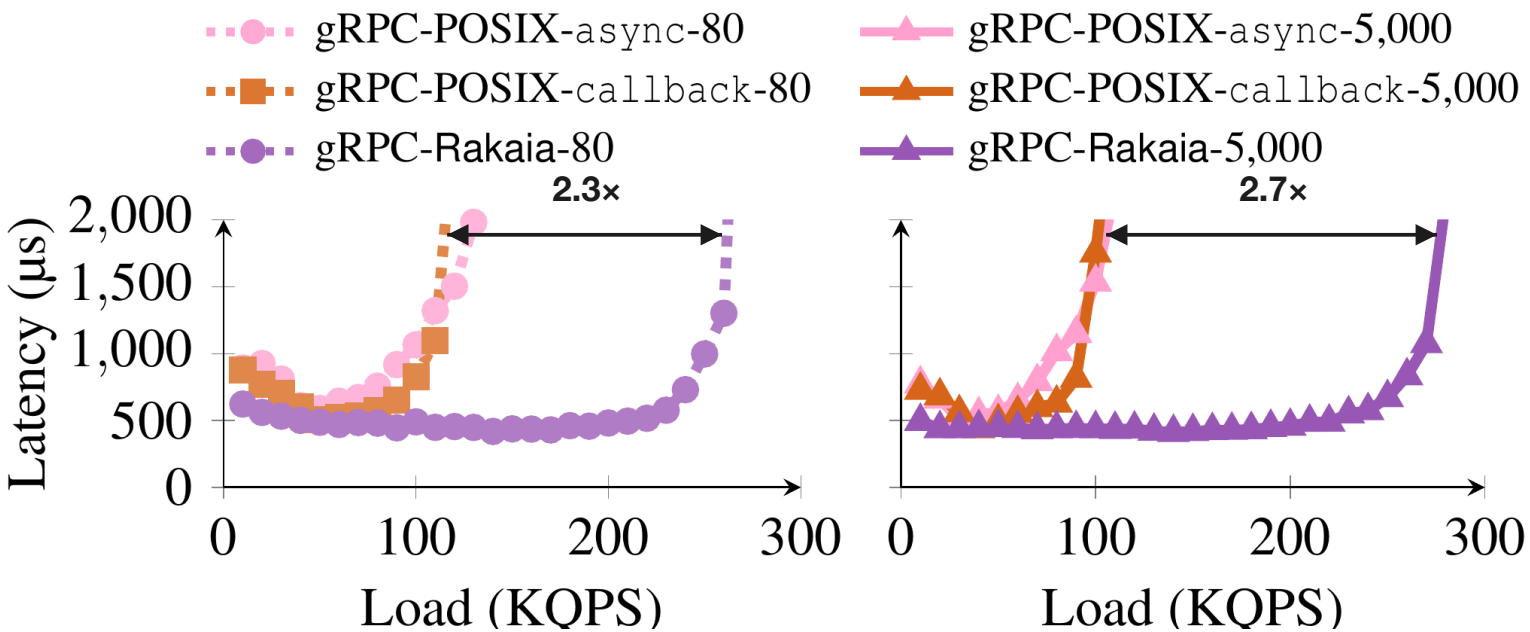
100µs service time, 20 connections, bimodal workload.

gRPC-Go comparison



With 5k connections, Rakaia only needs 33 goroutines, while gRPC-Go requires 17k goroutines.

gRPC-C++ comparison



Rakaia improves gRPC-C++ by 2.3× at 80 connections, and 2.7× at 5k connections

Conclusion

- Problem: Mismatch between stream-oriented TCP and message-oriented RPCs.
- Design: **Rakaia** parses TCP streams into messages and schedules them from the kernel.
- Result: Greatly accelerates both plain-TCP RPCs and gRPC (both C++ and Go).

Code artifact

github.com/epfl-dcsl

/rakaia-osdi26-artifact