IX: A Protected Dataplane Operating System for High Throughput and Low Latency

Adam Belay, George Prekas, Ana Klimovic, Samuel Grossman, Christos Kozyrakis, Edouard Bugnion

Environment in the Datacenter

- Many small packets
- Large numbers of connections
- Short response latencies required (SLAs)

Assumptions in Traditional Systems

- Few/single core
- Slow NICs/long periods between packets
- Priority is to maximize utilization

Problems with Regular Networking Stacks

- Context switching
- Prioritization of fine-grained scheduling
- Hardware assumptions (single/few core, high latency packets)

Solution 1: Tuning

- No need for a custom solution
- Only provides some incremental improvement

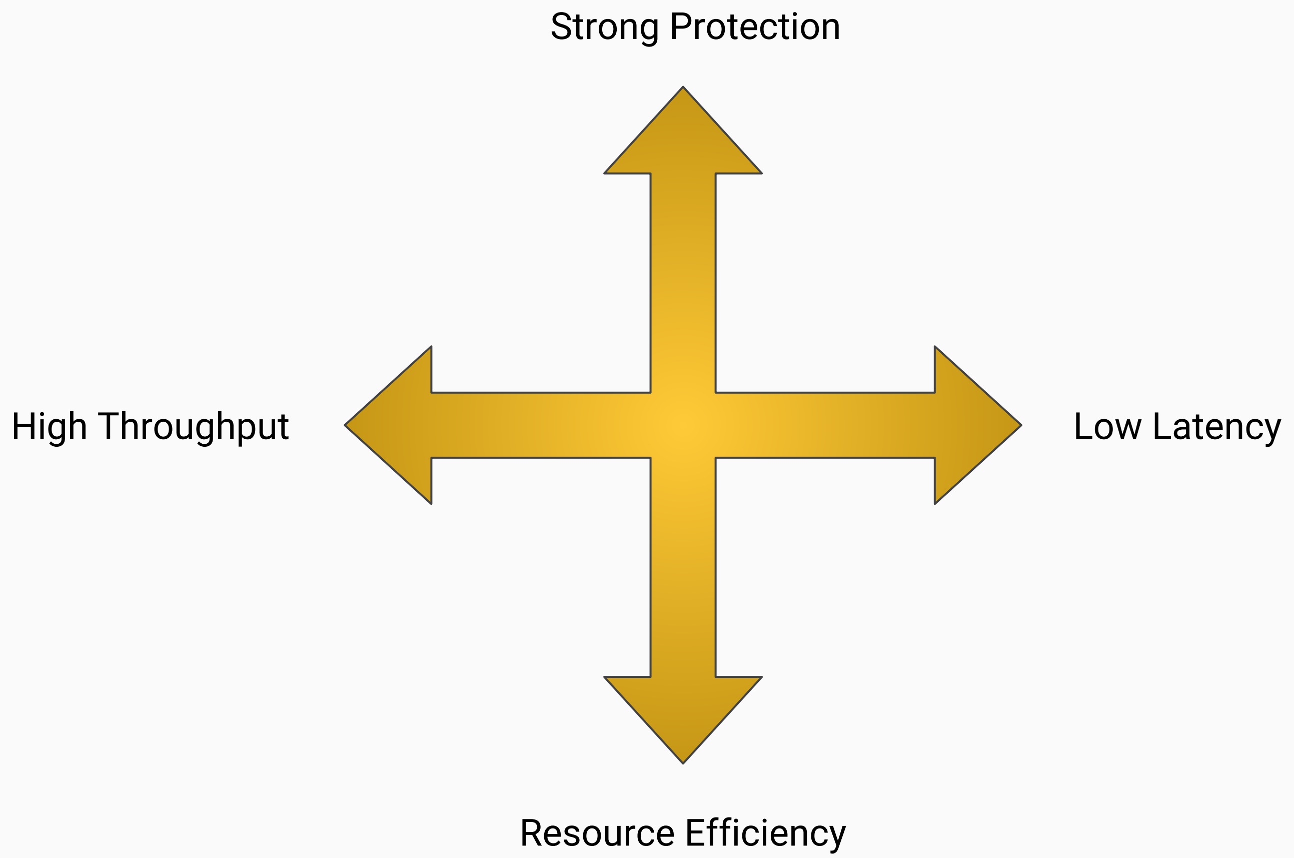
Solution 2: User-Space Networking

- Eliminate kernel transition costs
- No need for OS modifications
- Harder resource sharing, more batching
- Hard to enforce strong protections

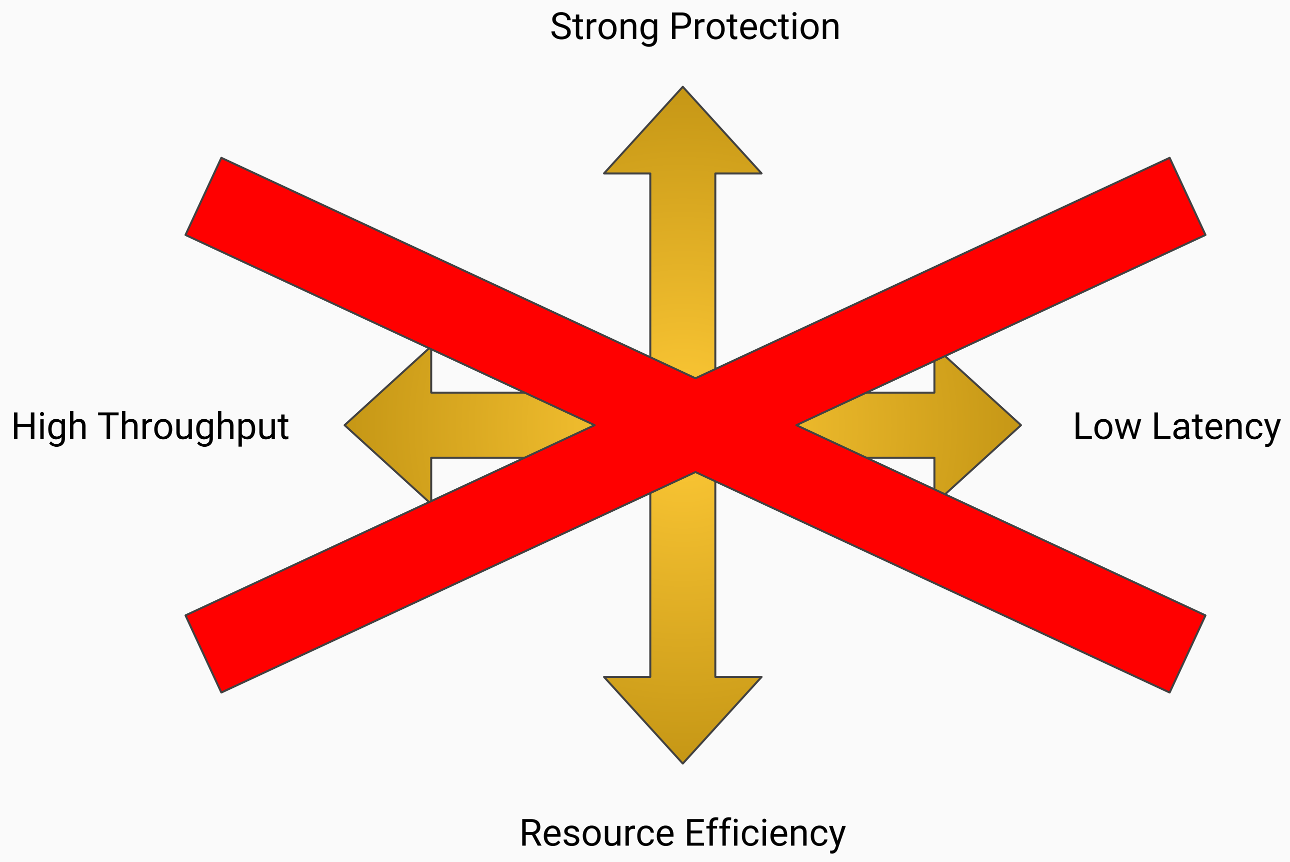
Solution 3: Alternative Protocols

- RDMA requests to dedicated middleboxes
- Requires speciality hardware and datacenter setup

The Tradeoffs



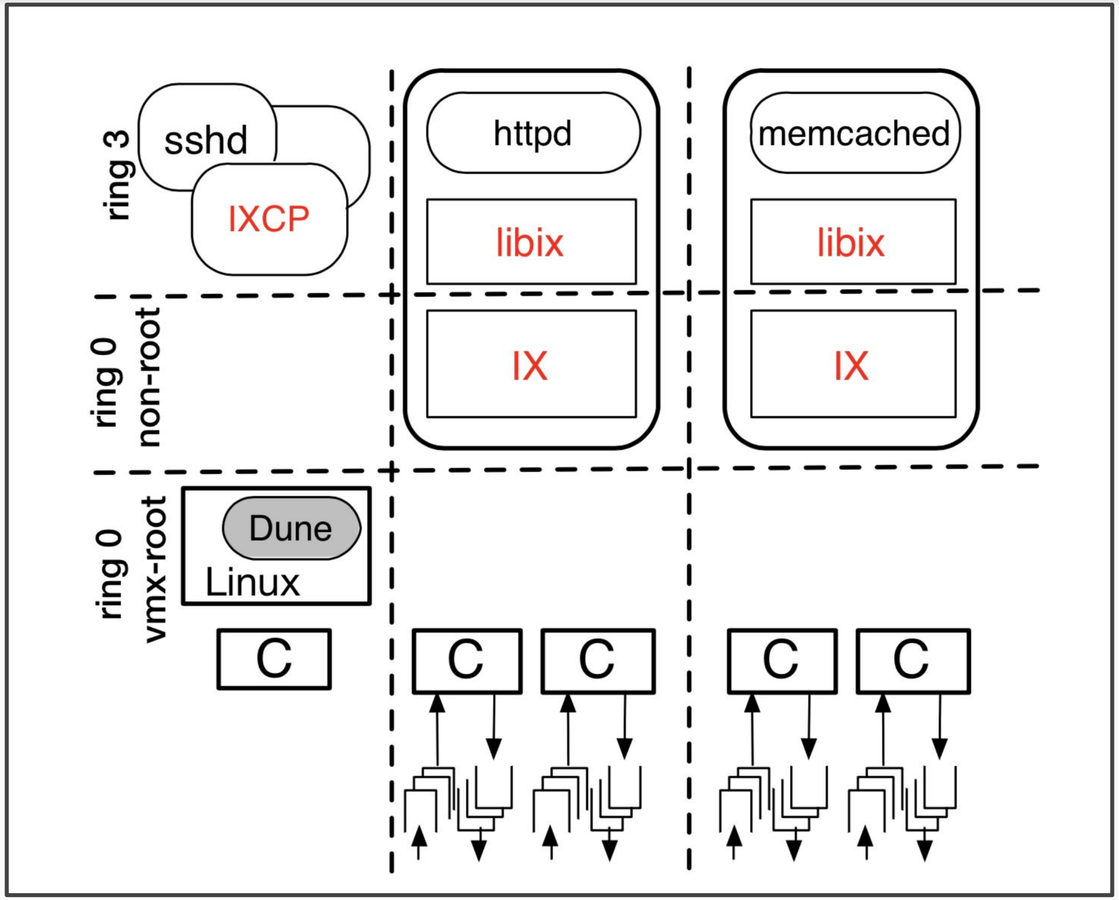
The IX Solution



IX: Dataplane OS

- **Idea: apply the high-speed processing approaches of user-space networking stacks with the isolation and protection of an OS**
- Virtualized dataplane OS provides protection and efficient scheduling
- Host controlplane OS coarsely allocates resources for dataplane

IX: Full-System Architecture



IX: Dataplane Architecture

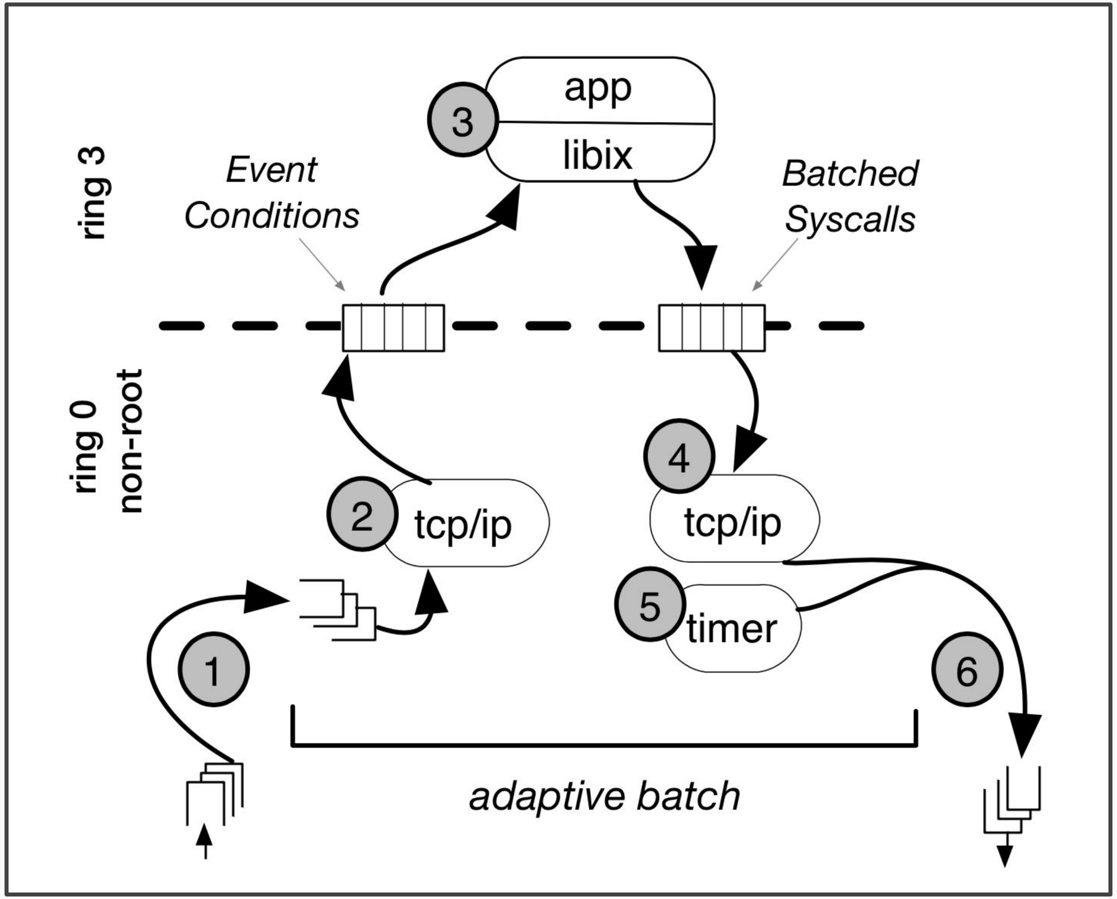
Elastic Threads:

- Interact with dataplane for network I/O
  - Issue batched syscalls
  - Consume event conditions
  - Receive payloads
- One per hardware thread

Background Threads:

- Everything else
- Can issue blocking syscalls
- Can arbitrarily multiplex cores

Adaptive Batching and Run to Completion



IX Features: Adaptive Batching

- Small batches (2--64 packets per batch)
- Amortize context switching costs
- Only batch if congested, otherwise don’t wait

IX Features: Locality

- Dedicated hardware threads/queues per dataplane
- Packets run to completion on core
- Reduces cache misses, critical for low tail latency

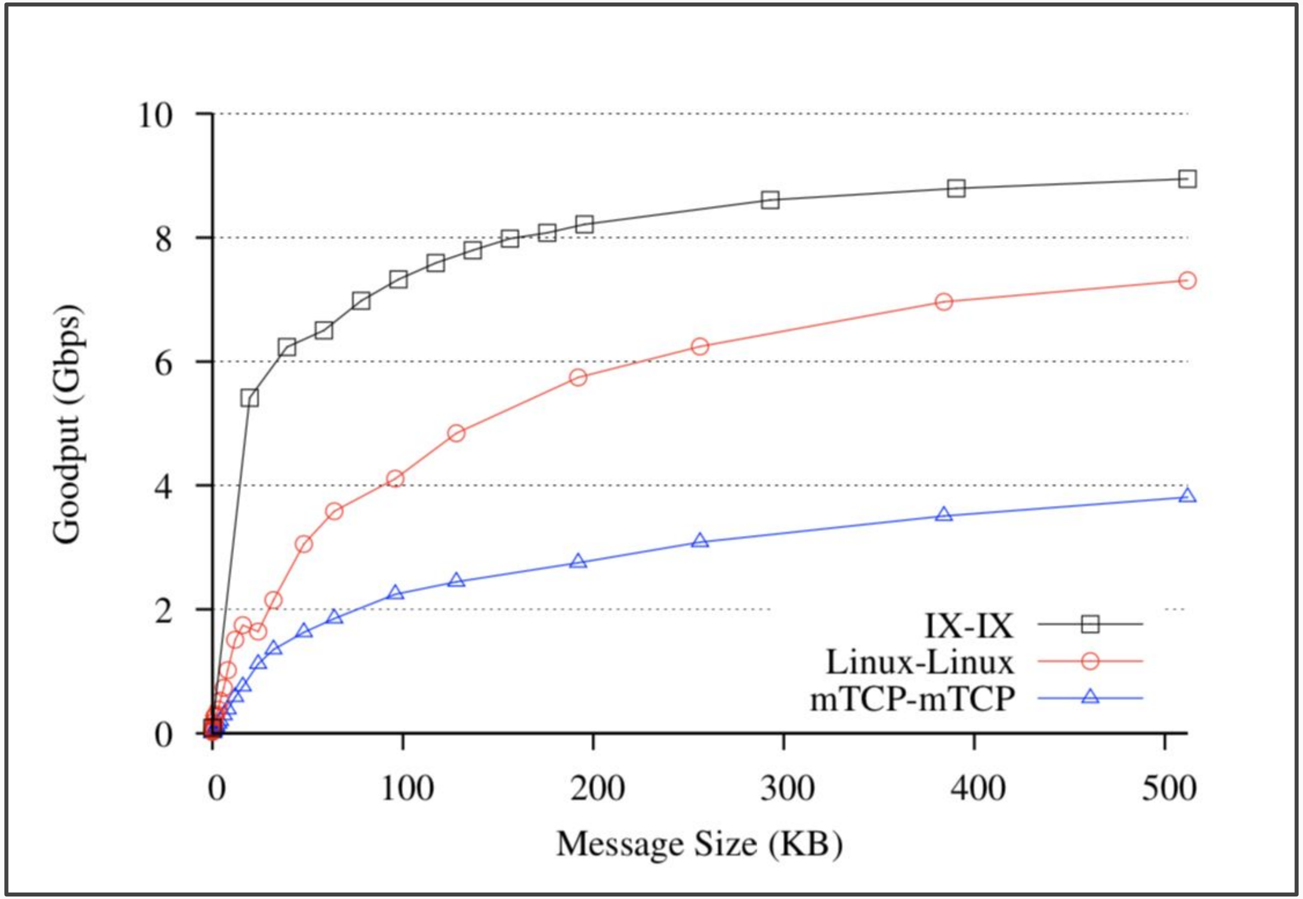
IX Features: No Synchronization

- Lock and coherency free
- Each elastic thread has own resource pools
- Threads work on disjoint sets
- Also critical for low latency and high throughput

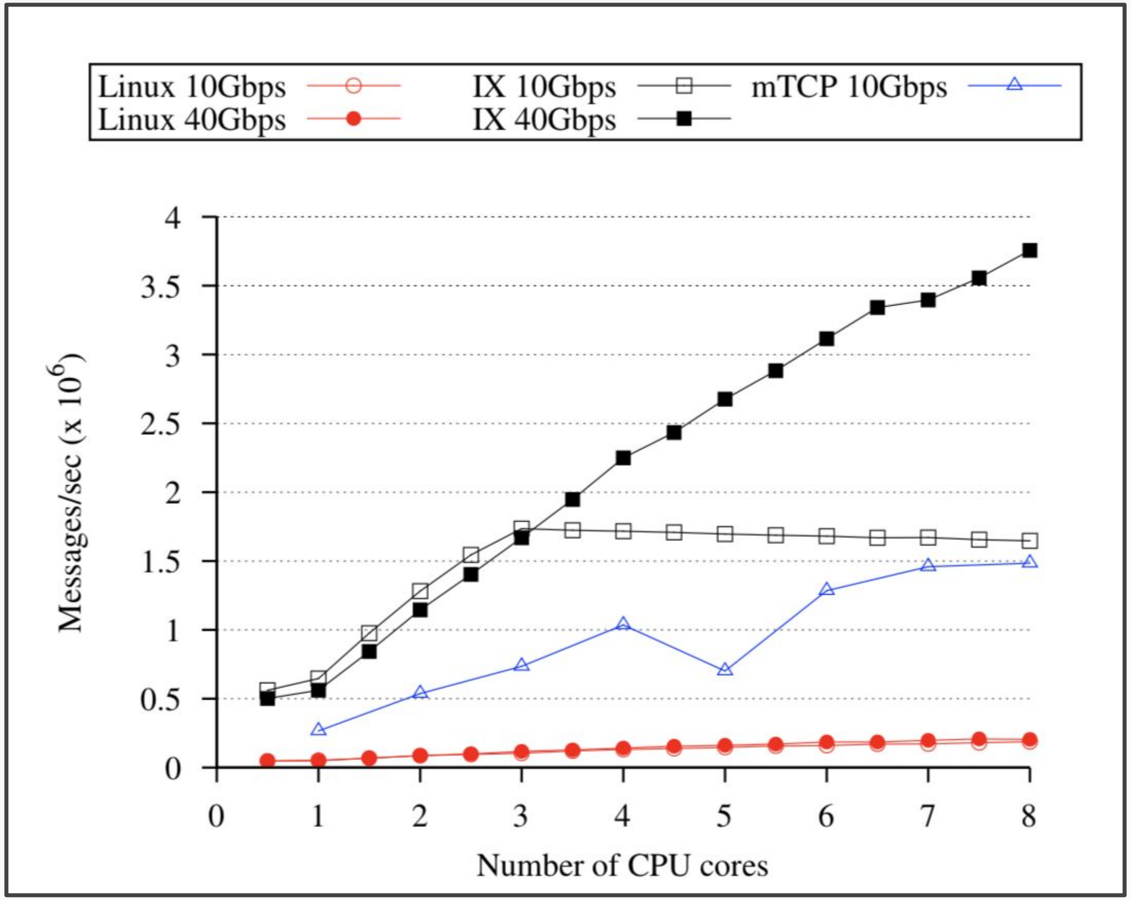
IX Features: Events and Zero Copy

- Notifications are done via polling rather than interrupts
- Data placed as pointers to shared buffers on buffer ring
- With dataplane isolation, data corruption leads only to bad messages

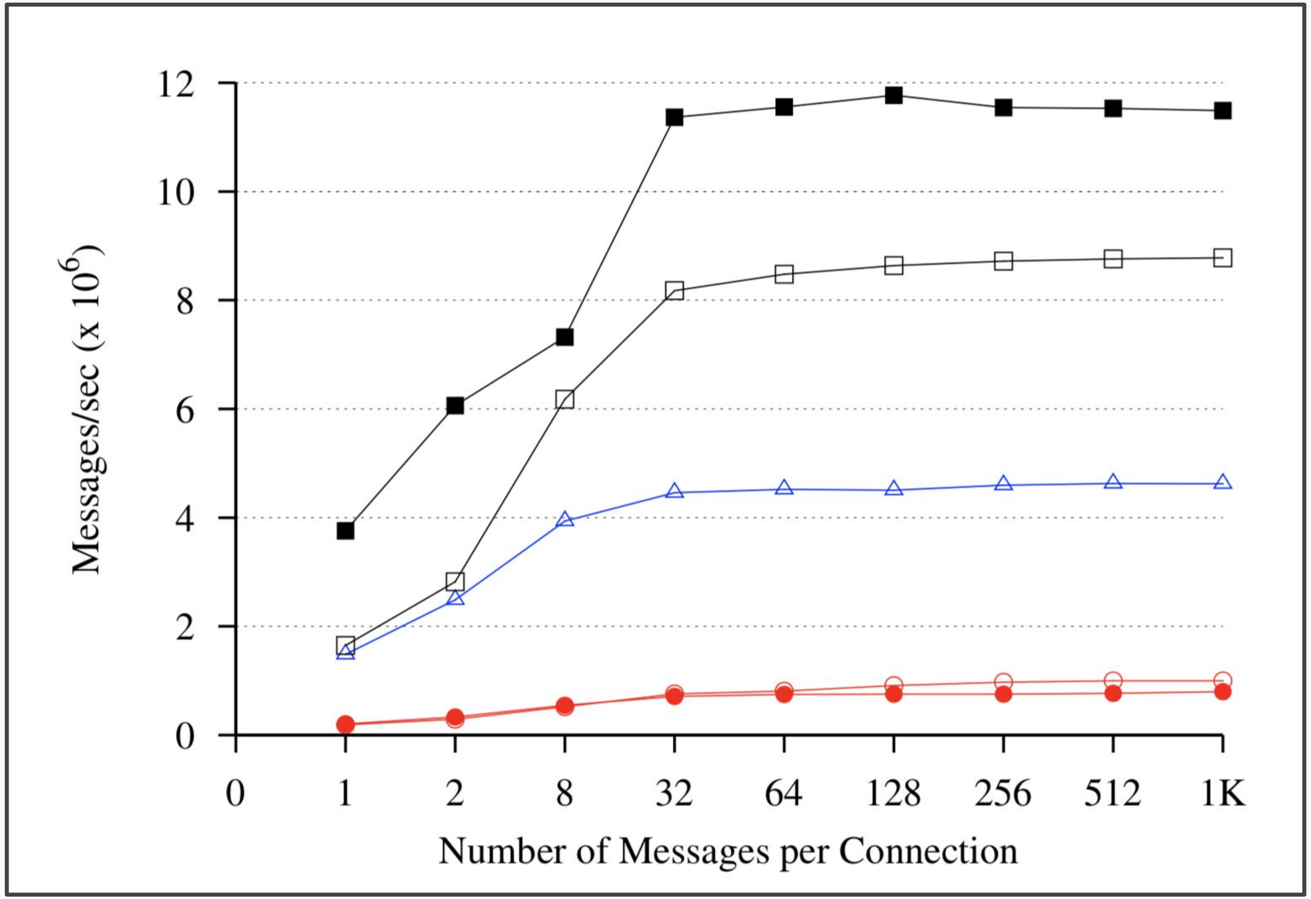
Evaluation: Throughput



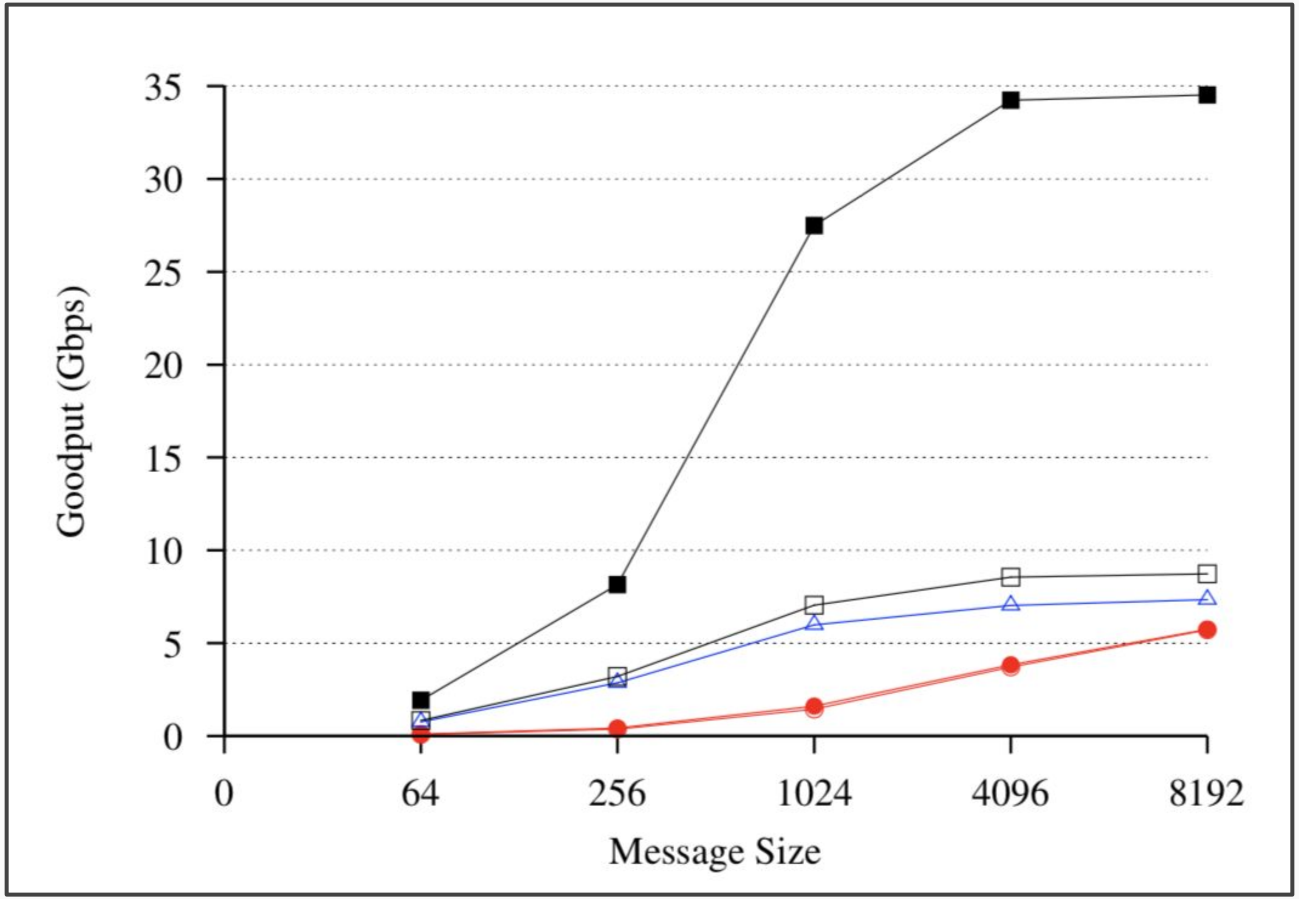
Evaluation: Scalability



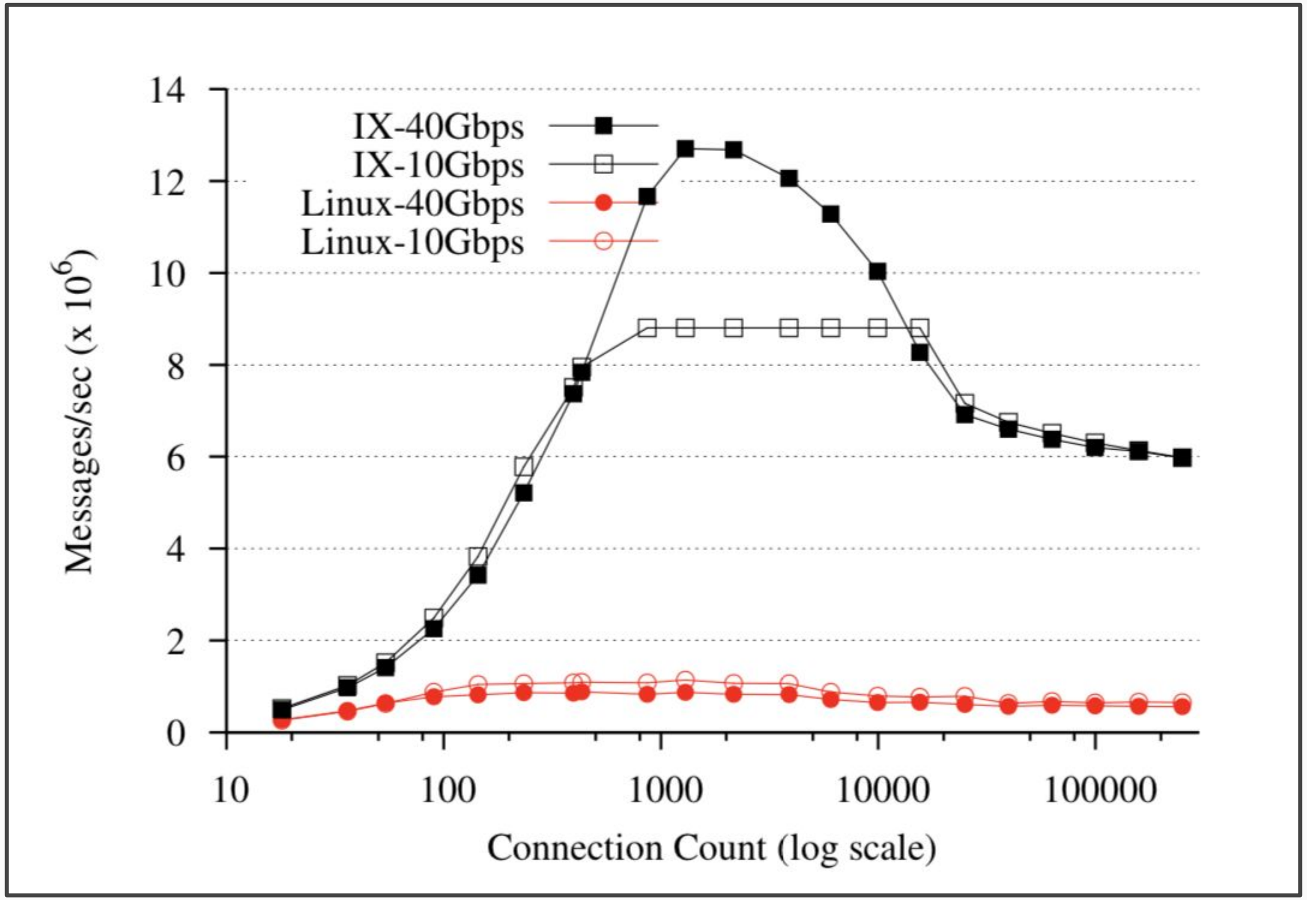
Evaluation: Scalability



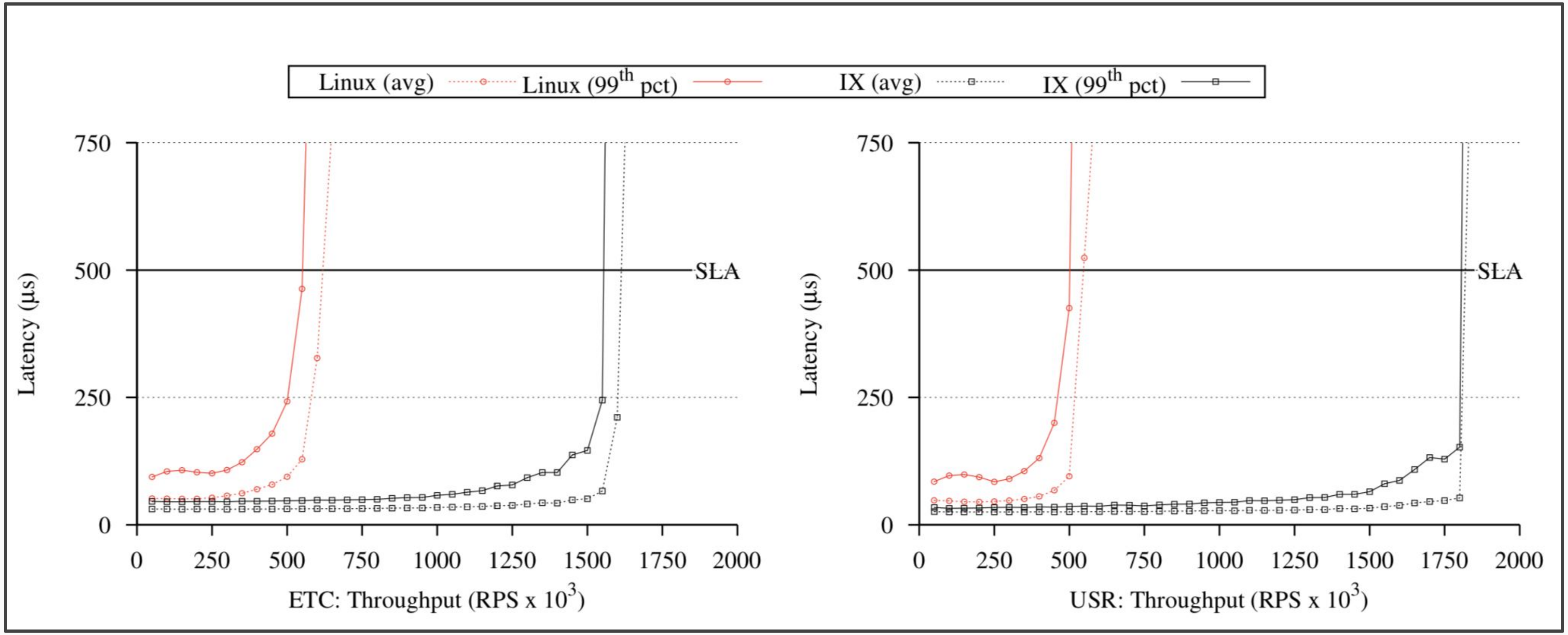
Evaluation: Throughput



Evaluation: Scalability



Evaluation: Latency



Related Work

- Arrakis
  - Uses Barrelfish as controlplane OS
  - Uses SR-IOV to give hardware resources directly to applications, IOMMU
- Exokernels
  - More general purpose

Summary

- **Low Latency** through locality, zero copy, asynchrony
- **High Throughput** through adaptive batching, dedicated cores/queues
- **Resource Efficiency** by coarse allocation + application scheduling
- **Strong Protection** by virtualization

Discussion

Questions

- What did you not like?
- Weaknesses?
- Practical?
- How could we apply these principles to other spaces?

Extra Stuff

Weaknesses

- Could do the same in user space
  - On order of 1 L3 miss reduced latency, less protection
  - Too dismissive?
-

Related Project: Dune

- Module to provide applications access to privileged hardware features
- Allows access through an API in a kernel module rather than trap/emulate

Other Optimizations

- Only uses huge pages (2MB versus 4KB)
- Dataplane manages its own TLB, address mappings