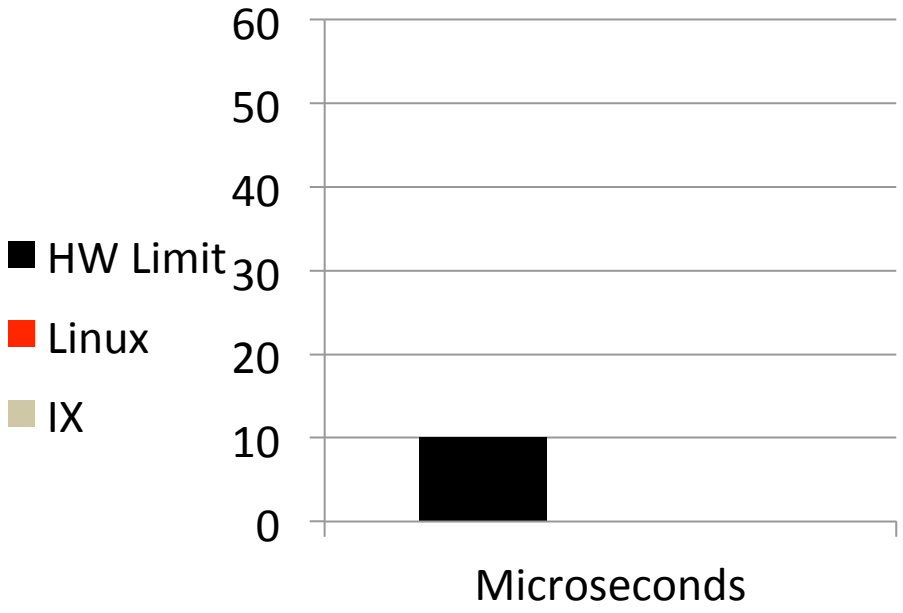
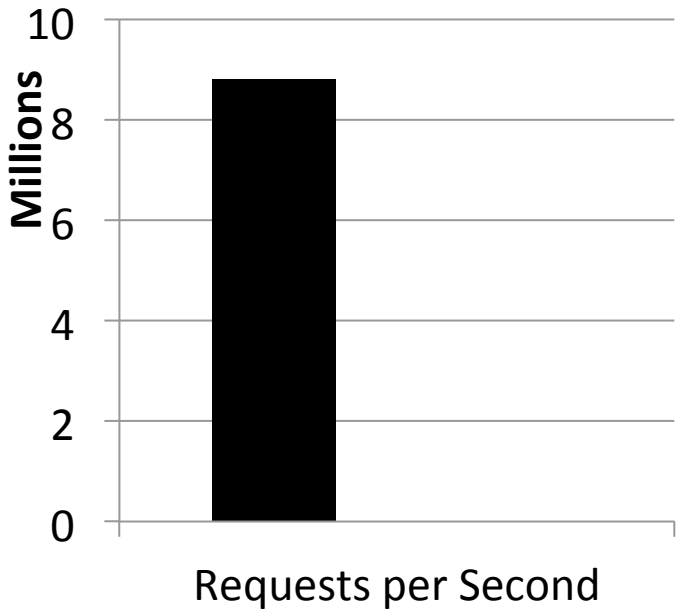
IX: A Protected Dataplane Operating System for High Throughput and Low Latency

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

HW is fast

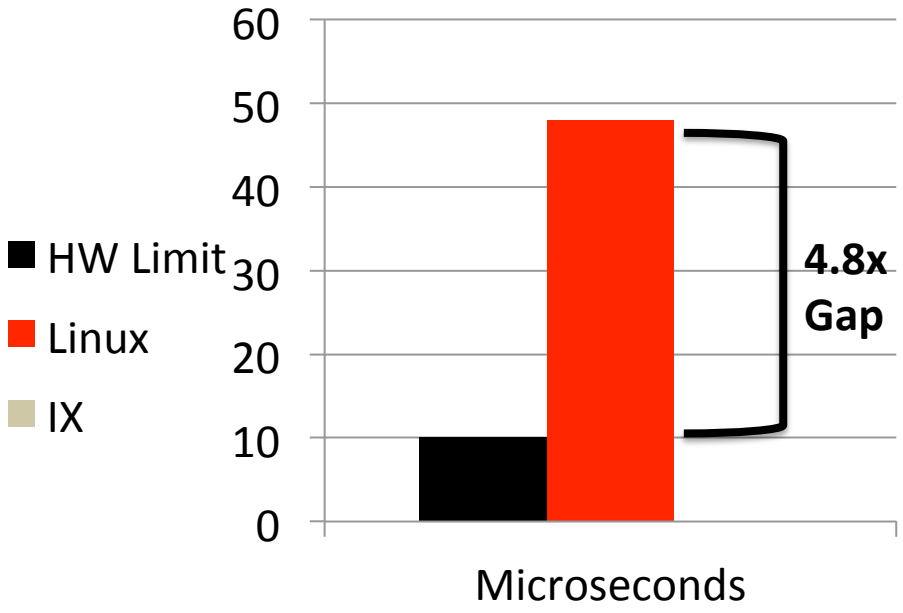
64-byte TCP Echo:

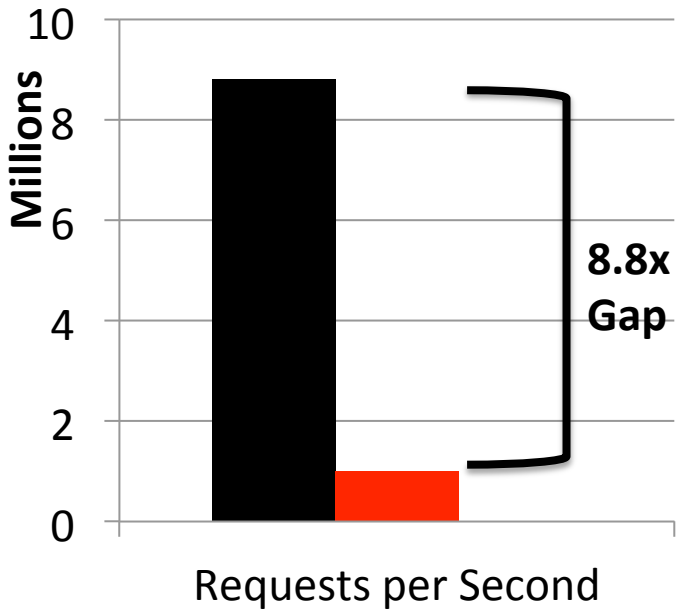




HW is fast, but SW is a Bottleneck

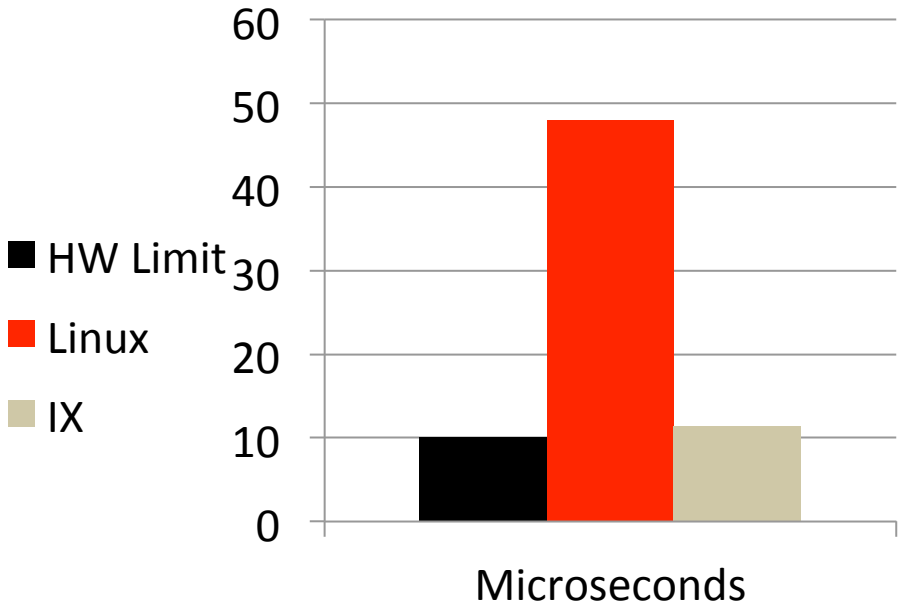
64-byte TCP Echo:

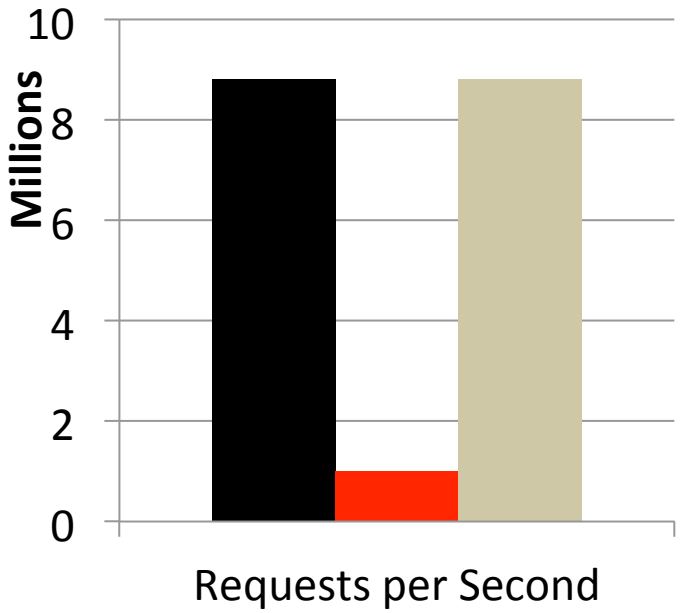




IX Closes the SW Performance Gap

64-byte TCP Echo:

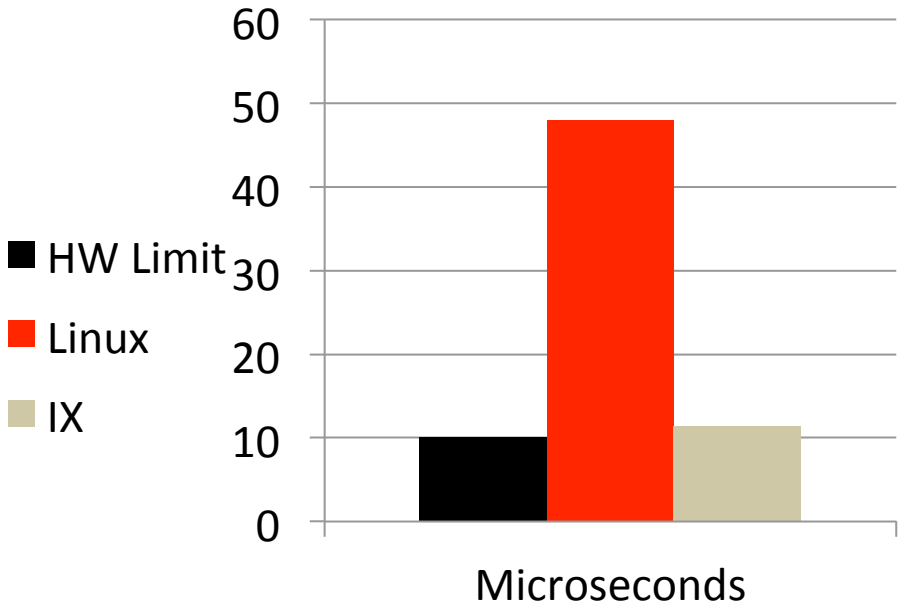


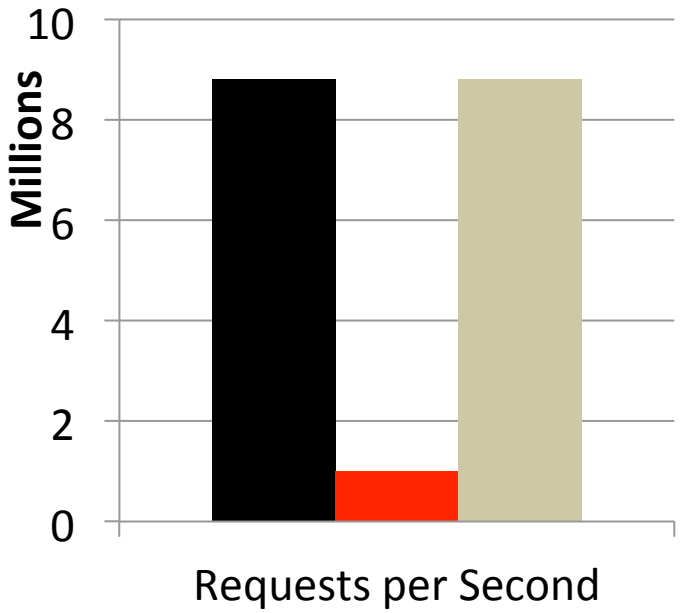


Two Contributions

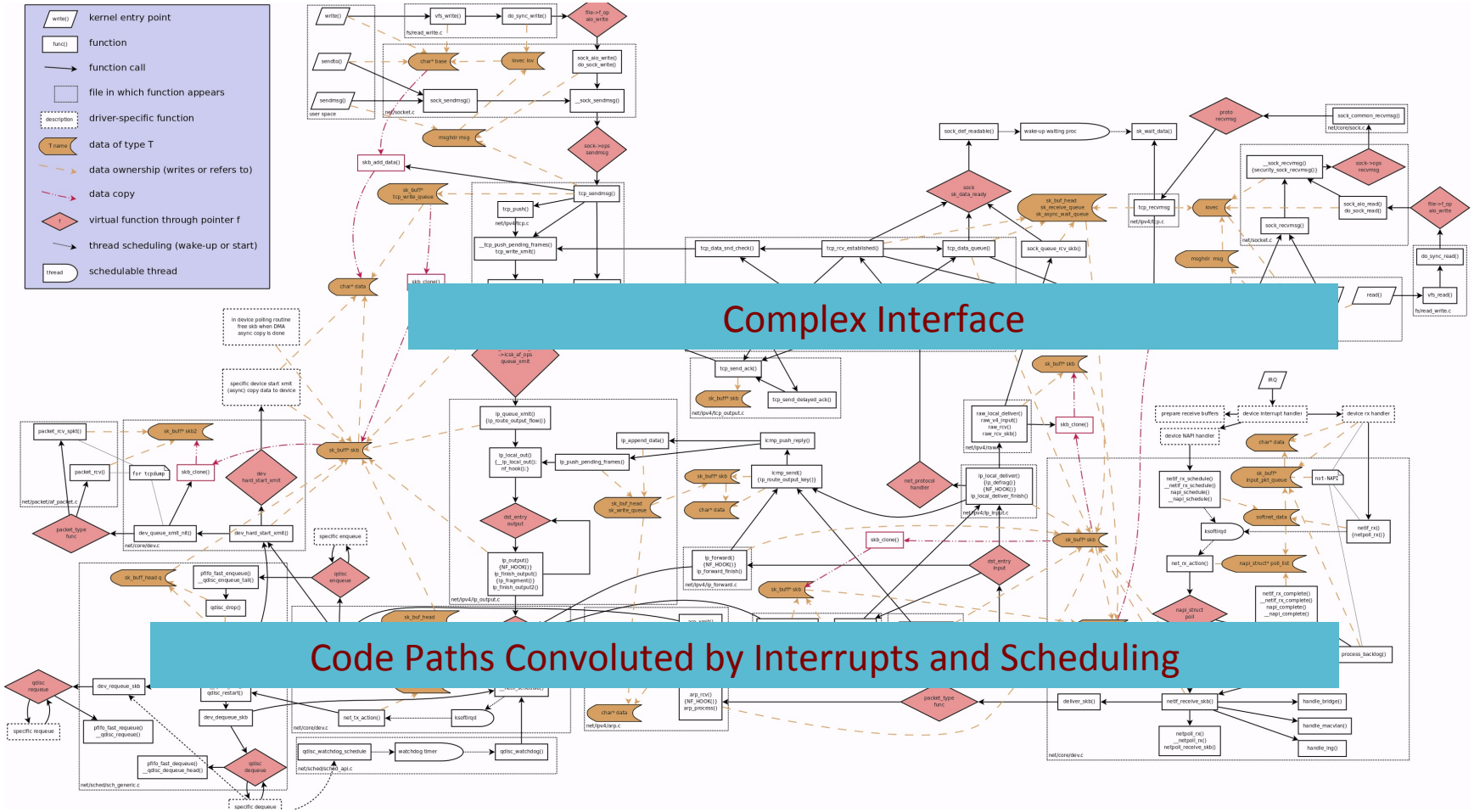
#1: Protection and direct HW access through virtualization

#2: Execution model for low latency and high throughput





Why is SW Slow?



Created by: Arnout Vandecappelle
http://www.linuxfoundation.org/collaborate/workgroups/networking/kernel_flow

Problem: 1980s Software Architecture

- Berkeley sockets, designed for CPU time sharing
- Today's large-scale datacenter workloads:

Hardware: Dense Multicore + 10 GbE (soon 40)

- API scalability critical!
- Gap between compute and RAM -> Cache behavior matters
- Packet inter-arrival times of 50 ns

Scale out access patterns

- Fan-in -> Large connection counts, high request rates
- Fan-out -> Tail latency matters!

Conventional Wisdom

- Bypass the kernel
  - Move TCP to user-space (Onload, mTCP, Sandstorm)
  - Move TCP to hardware (TOE)
- Avoid the connection scalability bottleneck
  - Use datagrams instead of connections (DIY congestion management)
  - Use proxies at the expense of latency
- Replace classic Ethernet
  - Use a lossless fabric (Infiniband)
  - Offload memory access (rDMA)

- **Common thread: Give up on systems software**

Our Approach

- Bypass the kernel
  - Move TCP to user space (Onload, mTCP, Sandstor
  - Move TCP to hardware (TOE)
- Avoid the connection scalability bottle
  - Use datagrams instead of connections (DIY congestior
  - Use proxies at the expense of latency
- Replace classic Ethernet
  - Use a lossless fabric (Infiniband)
  - Offload memory access (rDMA)

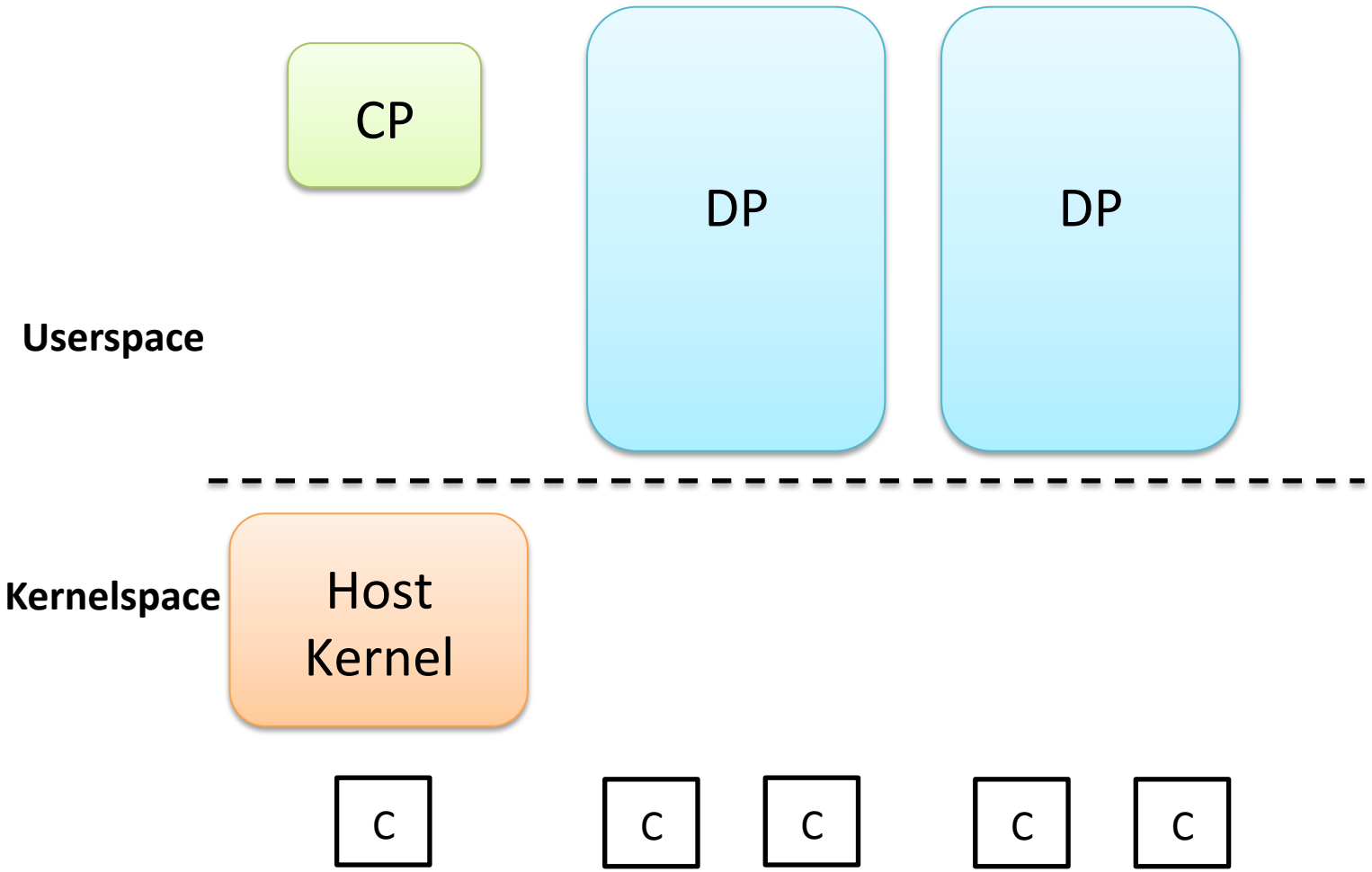
Robust Protection Between App and Netstack

Connection Scalability

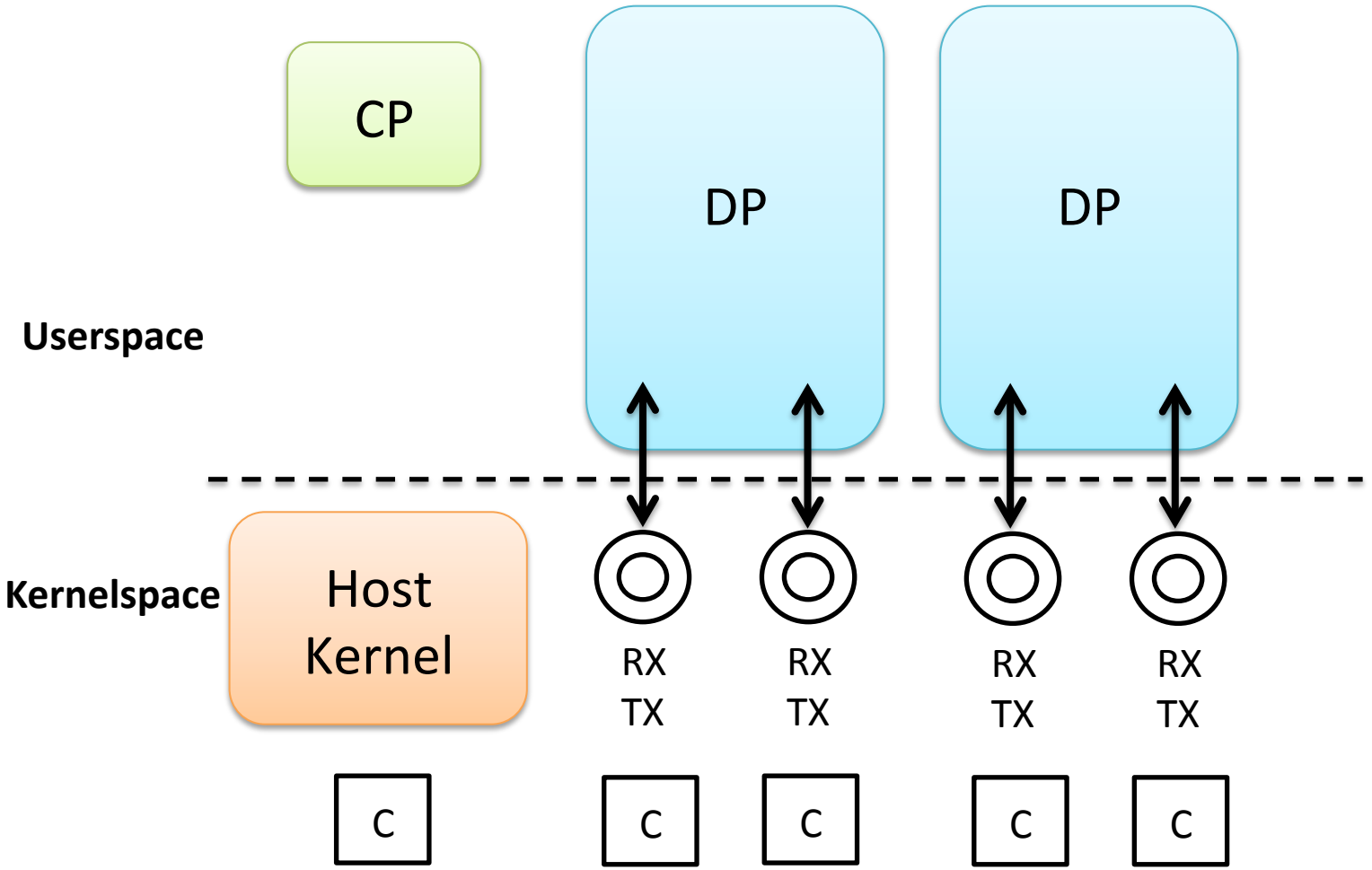
Commodity 10Gb Ethernet

- **Tackle the problem head on…**

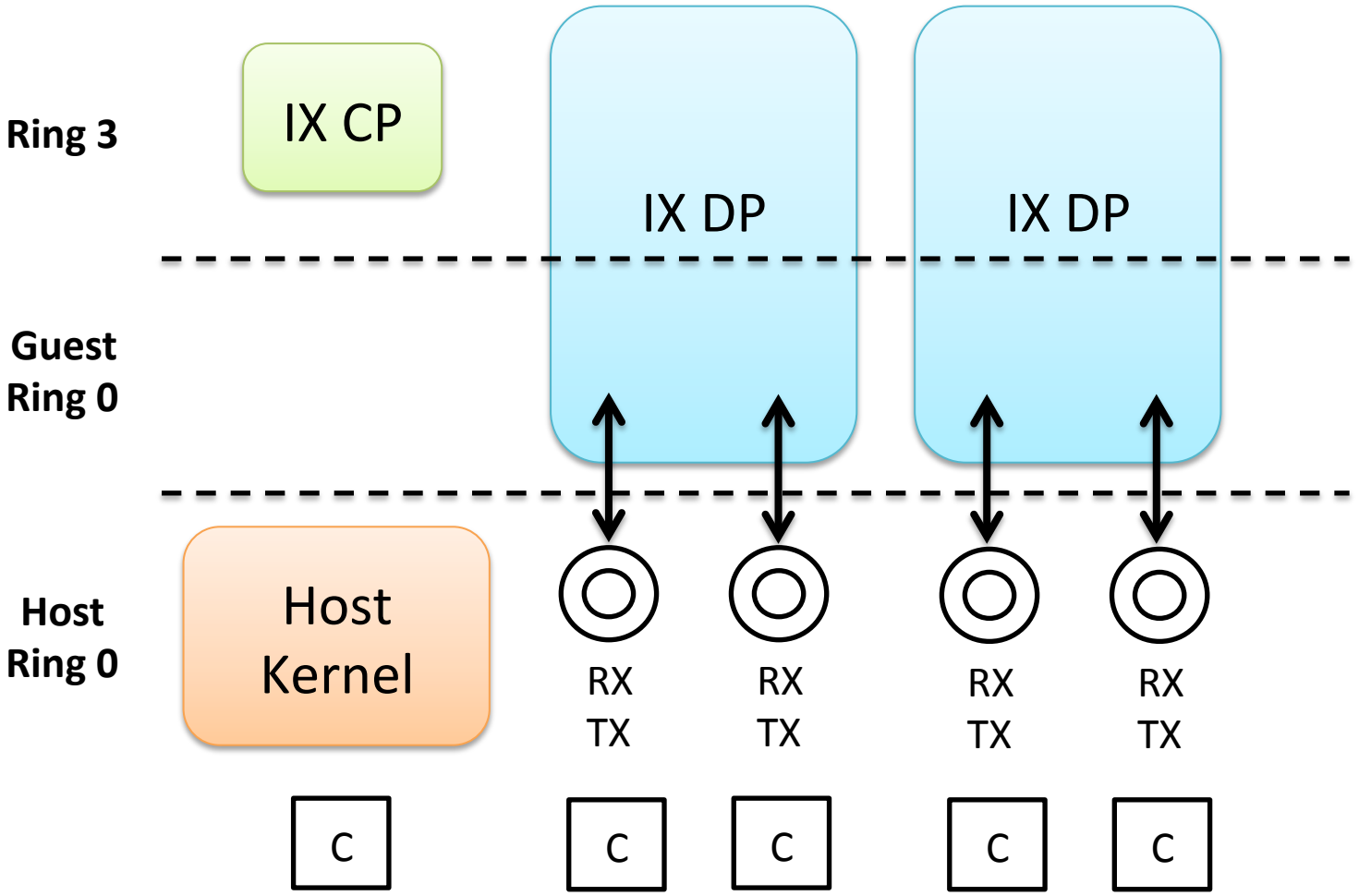
Separation of Control and Data Plane



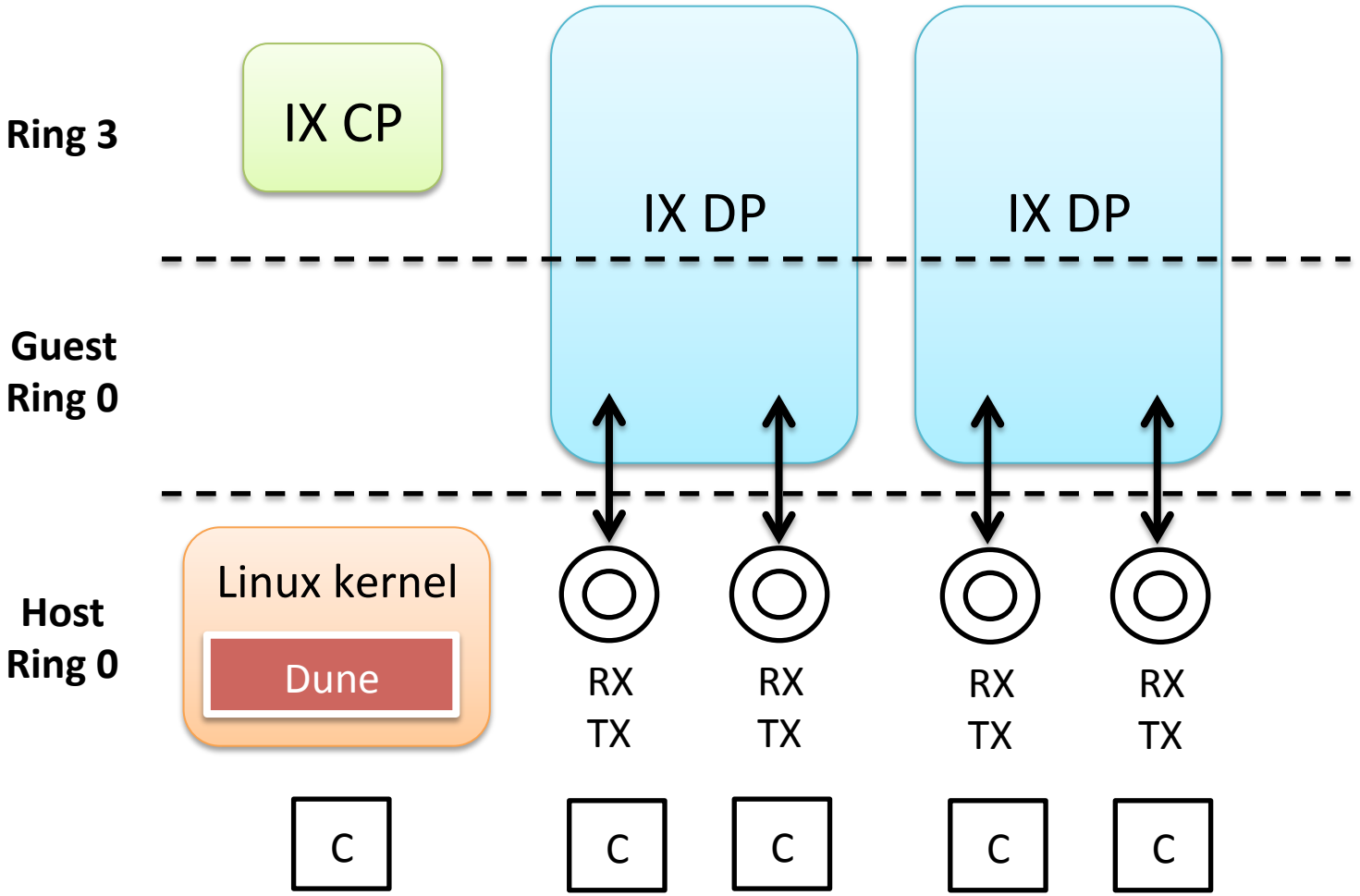
Separation of Control and Data Plane



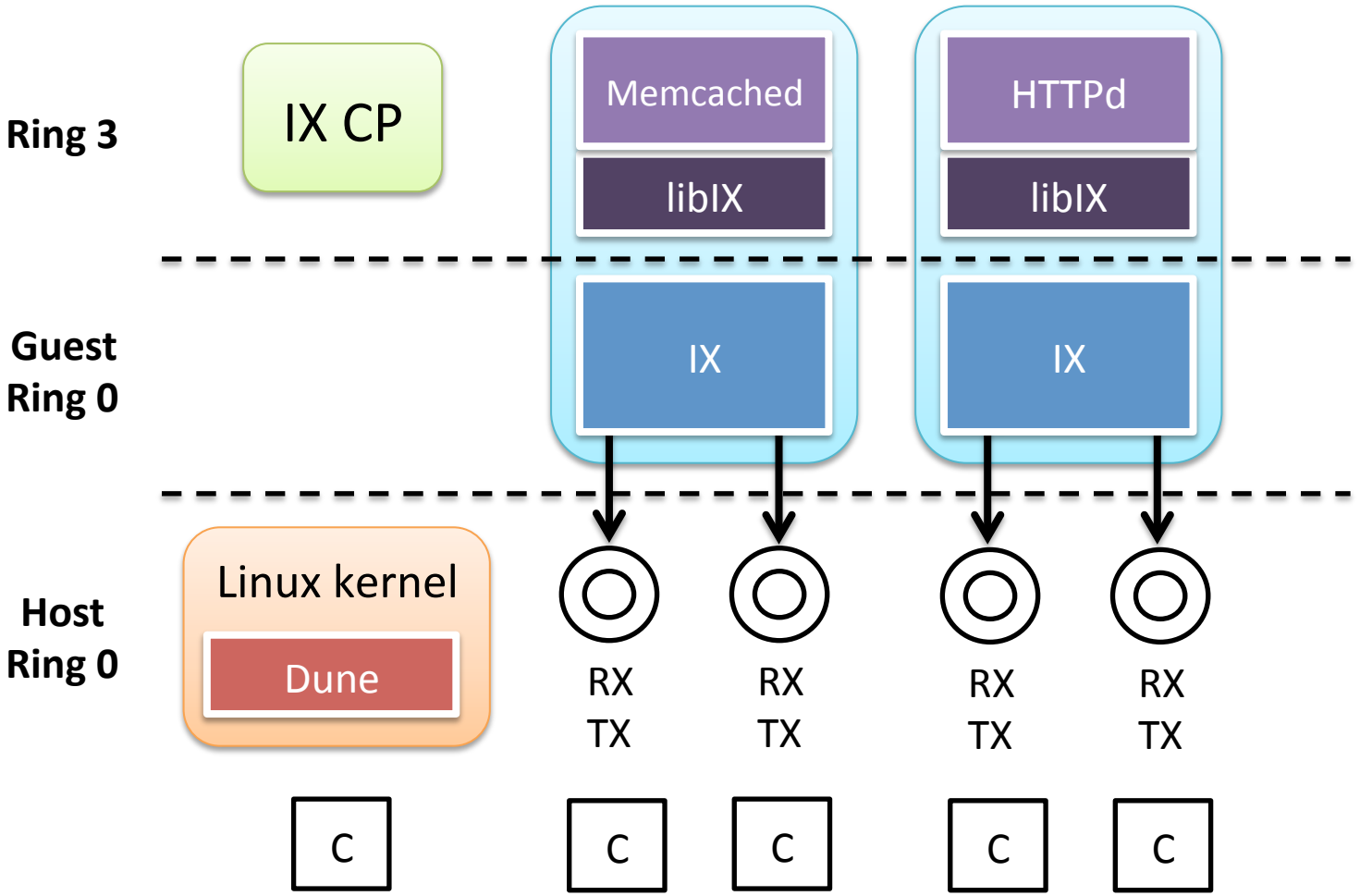
Separation of Control and Data Plane



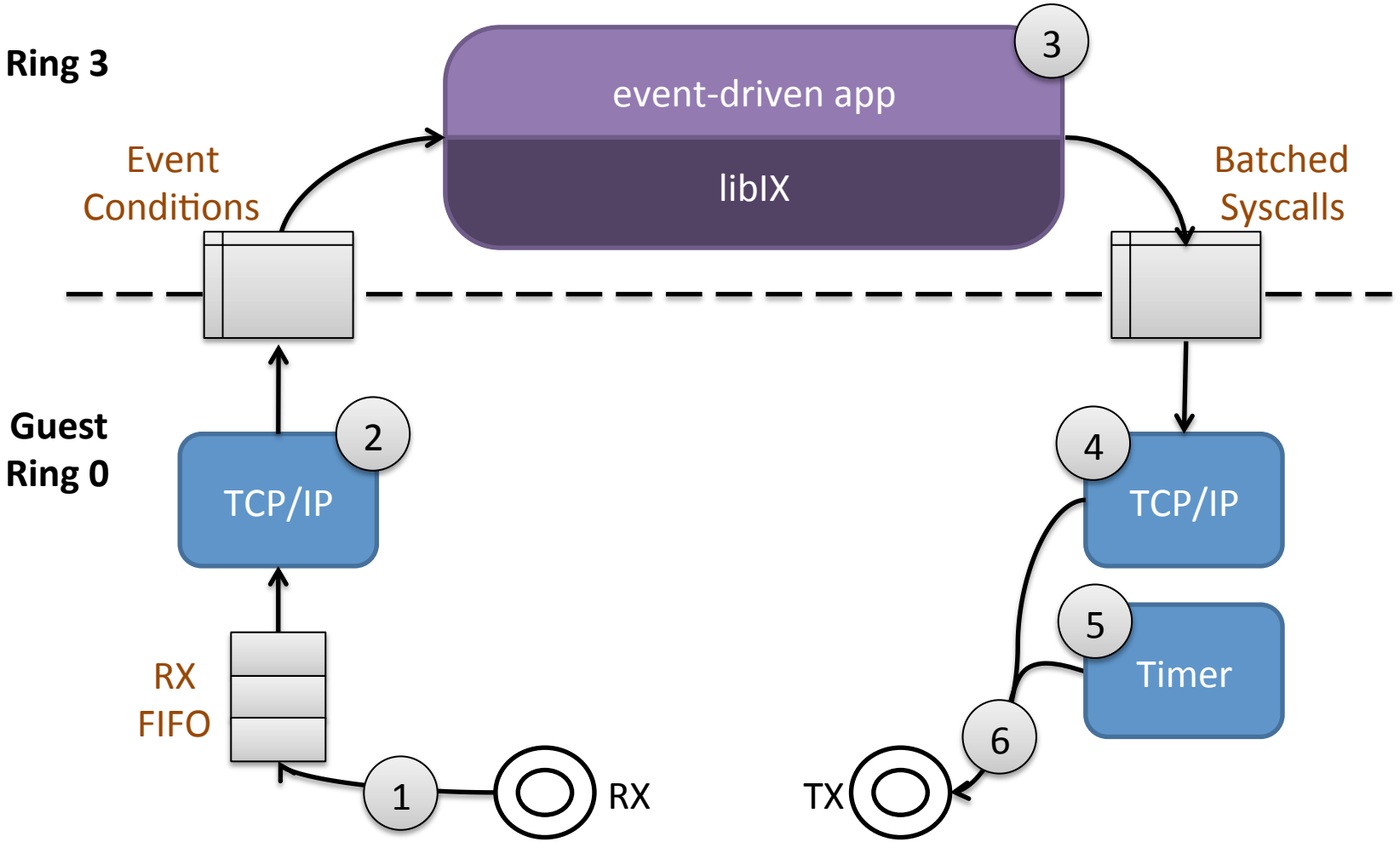
Separation of Control and Data Plane



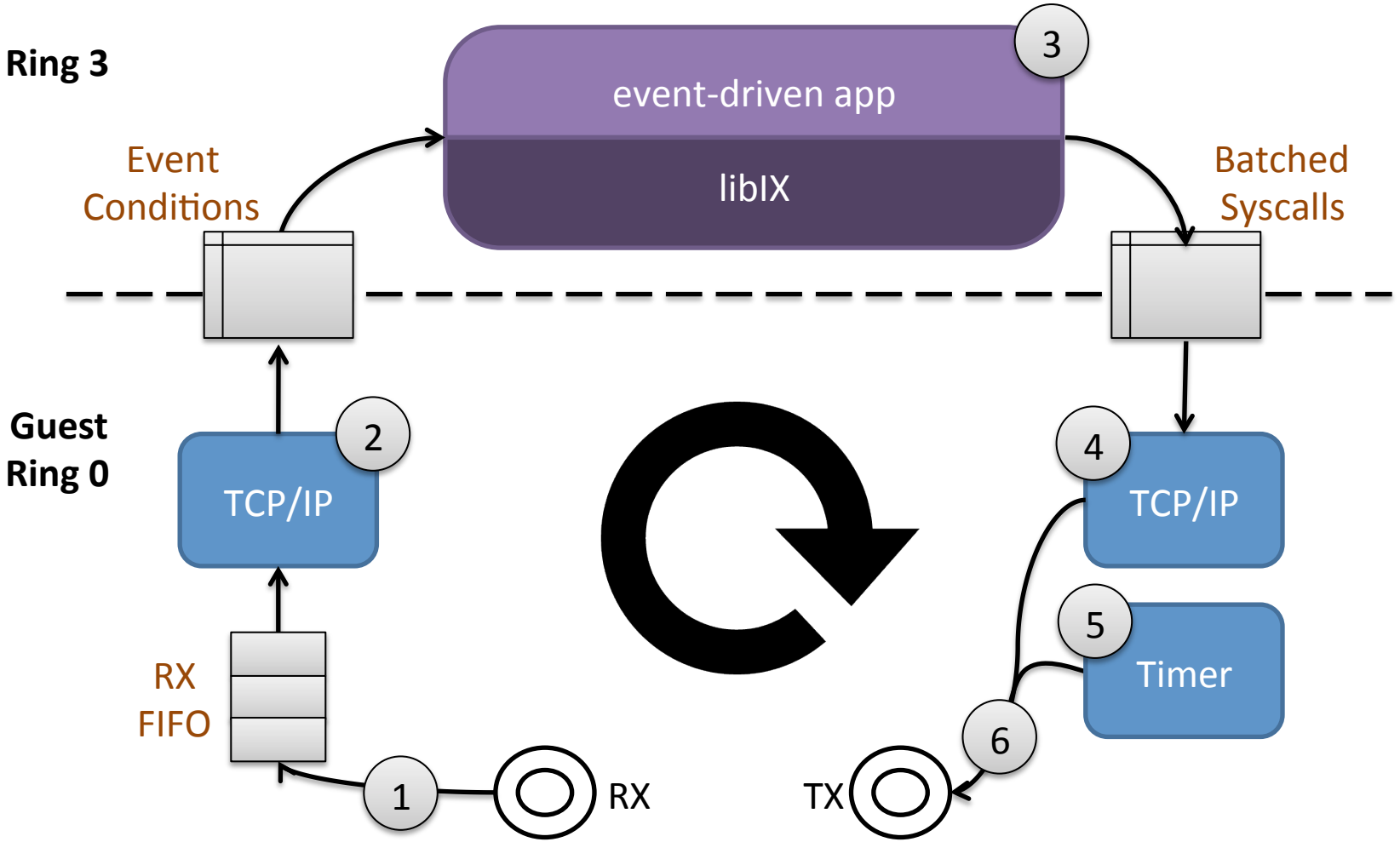
Separation of Control and Data Plane



The IX Execution Pipeline



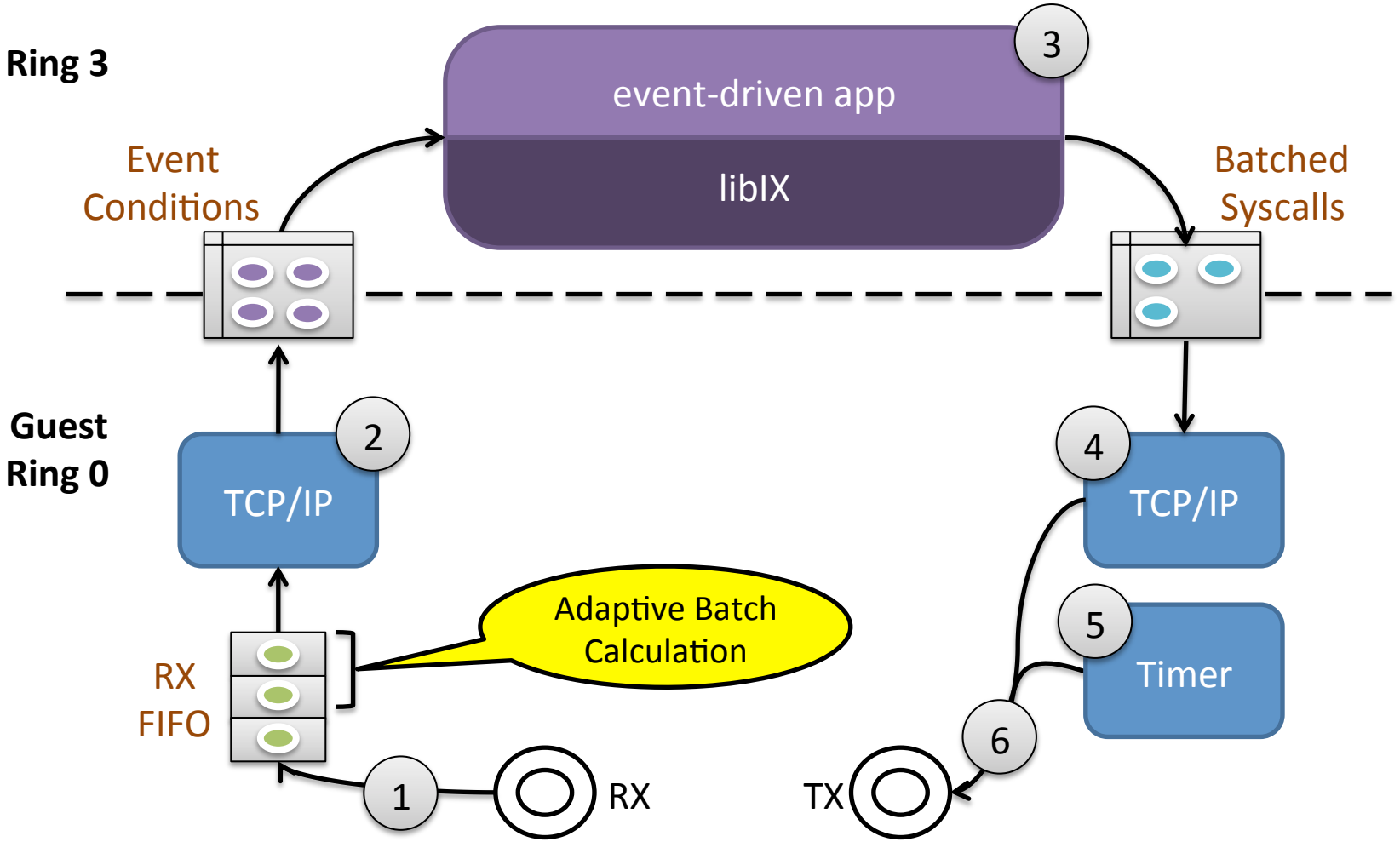
Design (1): Run to Completion



Improves Data-Cache Locality

Removes Scheduling Unpredictably

Design (2): Adaptive Batching



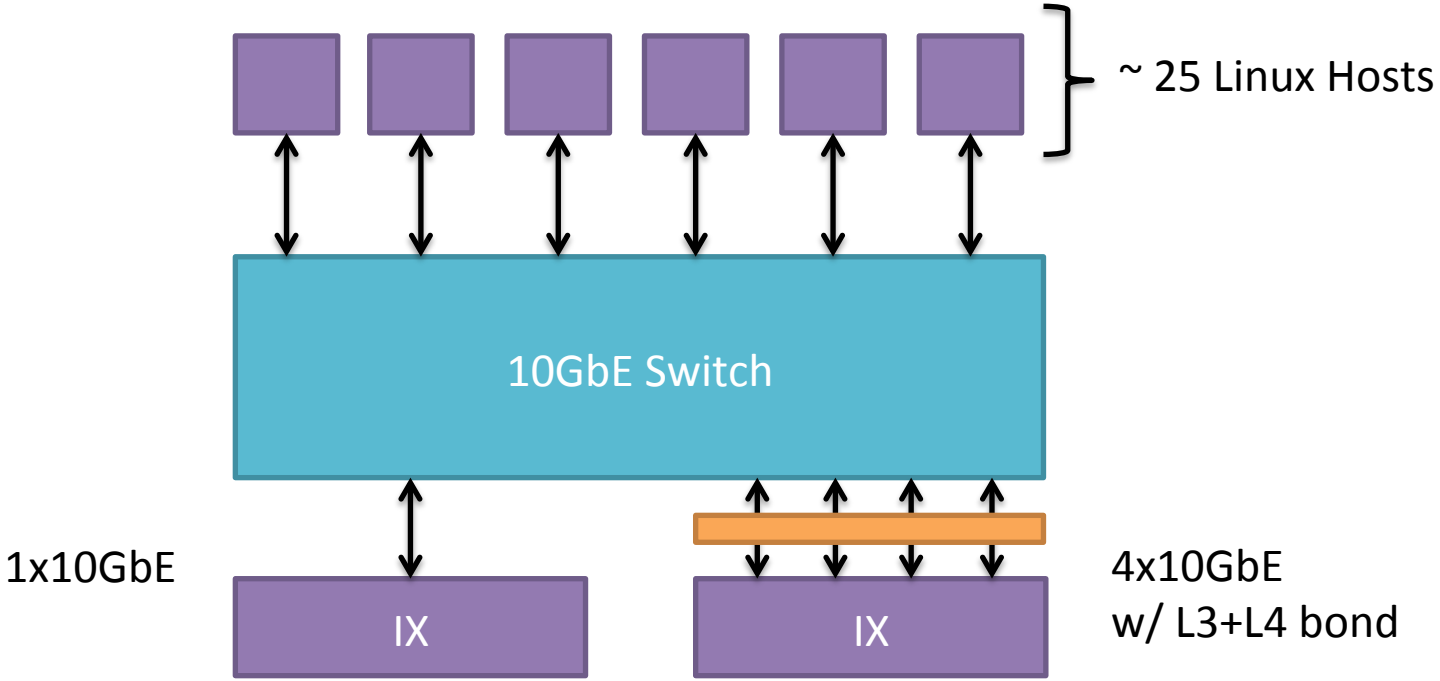
Improves Instruction-Cache Locality and Prefetching

See the Paper for more Details

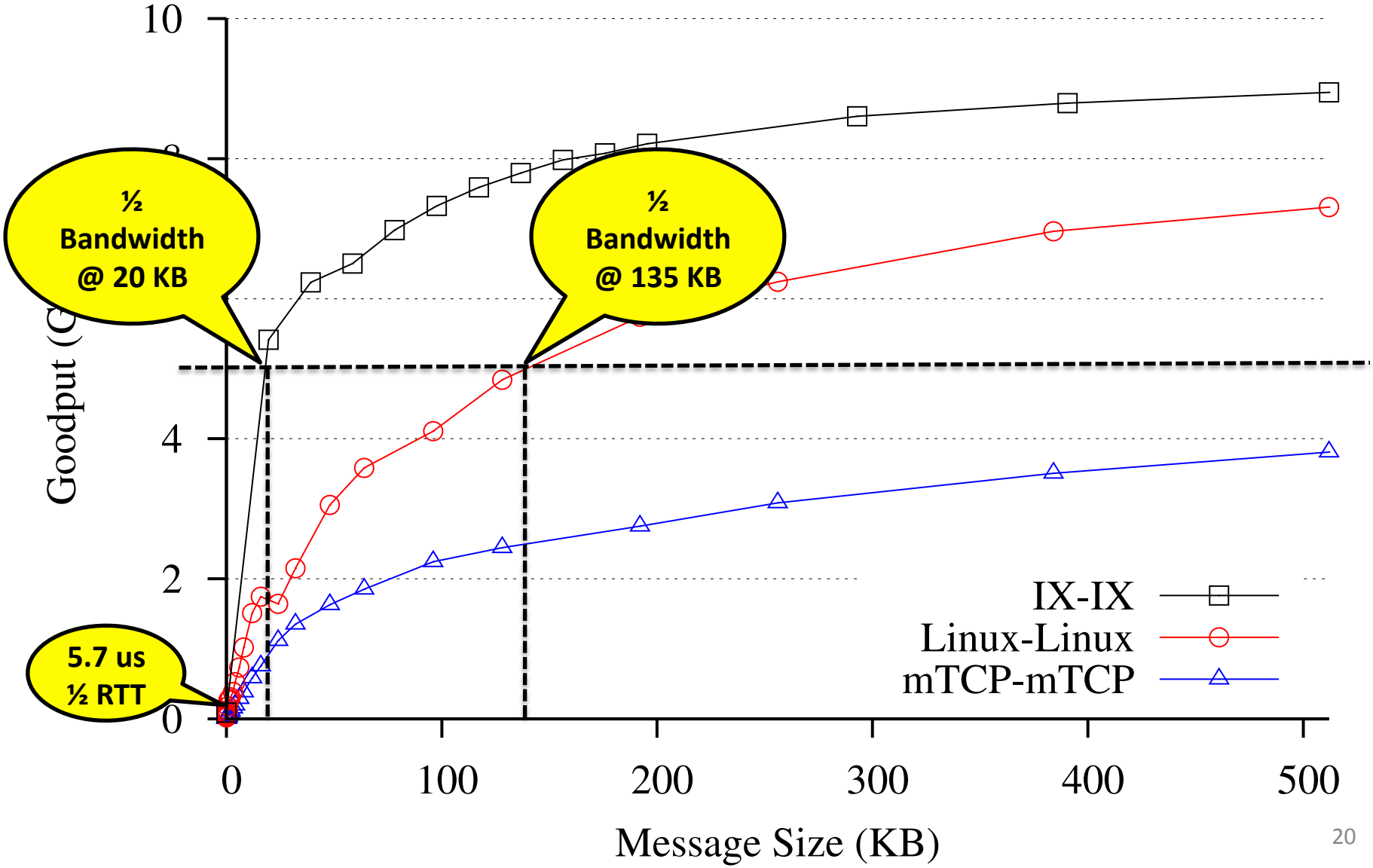
- Design (3): Flow consistent hashing
  - Synchronization & coherence free operation
- Design (4): Native zero-copy API
  - Flow control exposed to application
- Libix: Libevent-like event-based programming
- IX prototype implementation
  - Dune, DPDK, LWIP, ~40K SLOC of kernel code

Evaluation

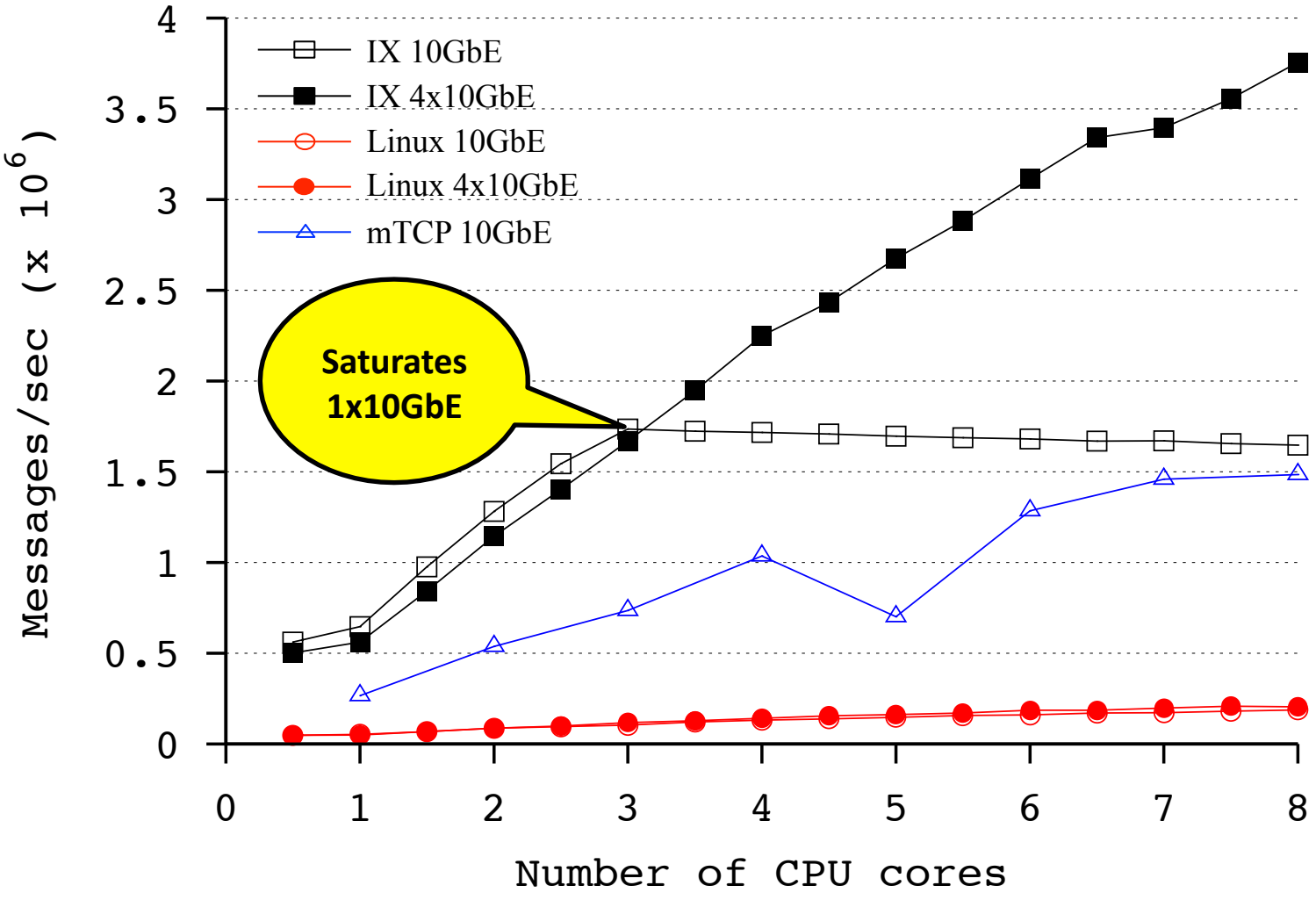
- Comparison IX to Linux and mTCP [NSDI '14]
- TCP microbenchmarks and Memcached



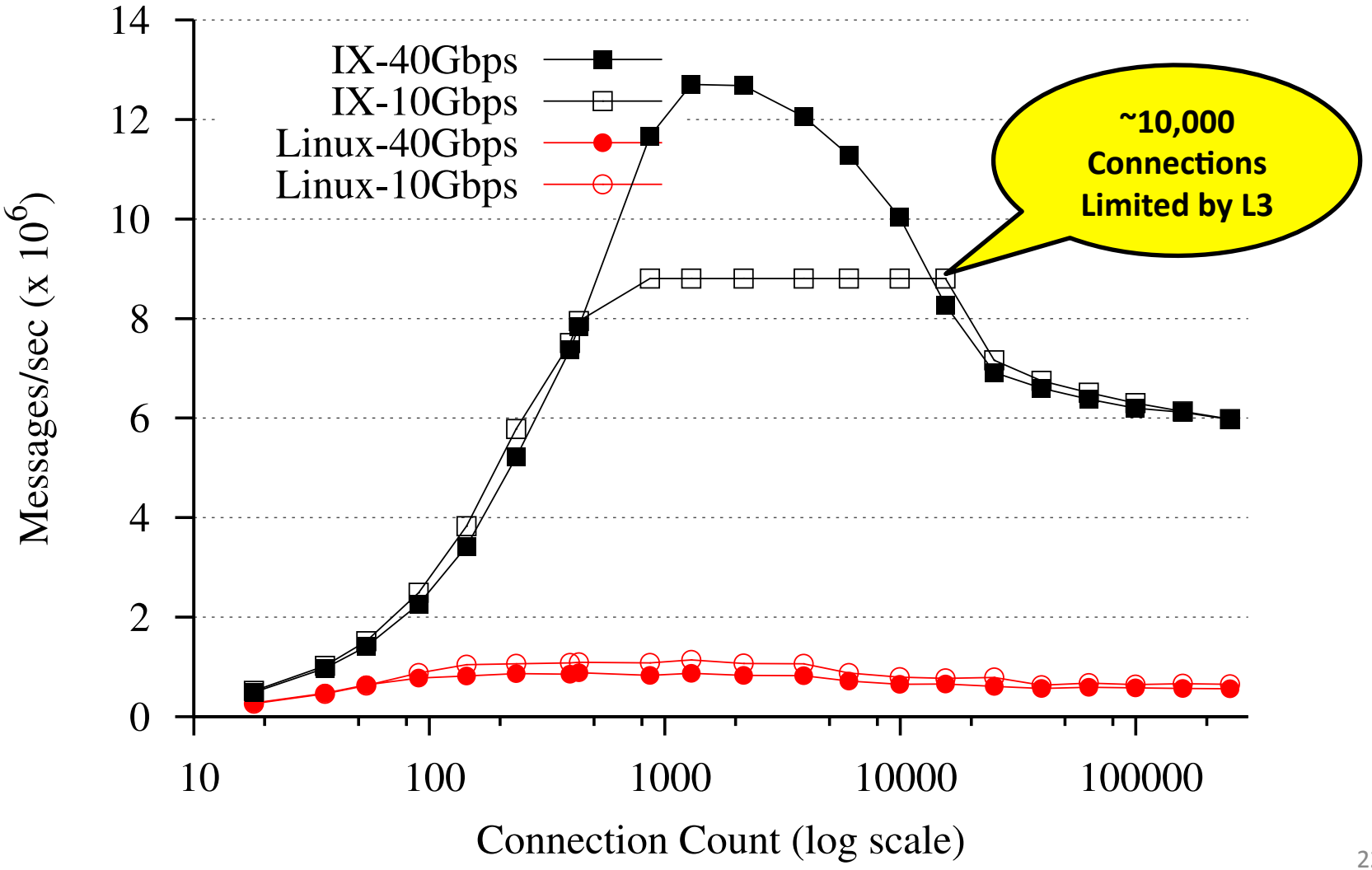
TCP Netpipe


10
4
2
0
Goodput (G
½ Bandwidth @ 20 KB
½ Bandwidth @ 135 KB
5.7 us ½ RTT
IX-IX
Linux-Linux
mTCP-mTCP
0
100
200
300
400
500
Message Size (KB)

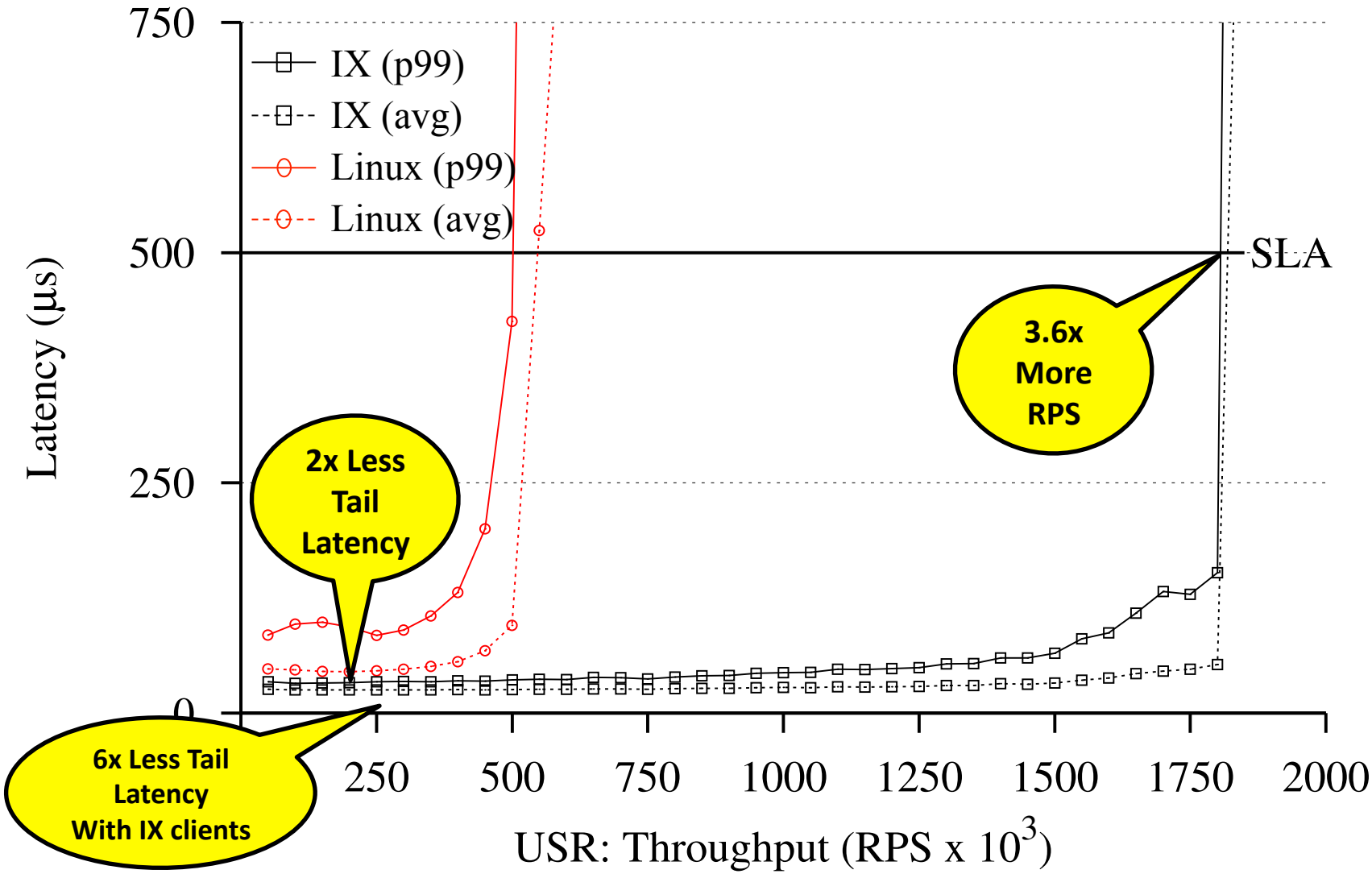
TCP Echo: Multicore Scalability for Short Connections



Connection Scalability



Memcached over TCP



IX Conclusion

- A protected dataplane OS for datacenter applications with an event-driven model and demanding connection scalability requirements
- Efficient access to HW, without sacrificing security, through virtualization
- High throughput and low latency enabled by a dataplane execution model