Protocol stacks and multicore scalability

The evolving hardware-software interface

or

Why we love and hate offload

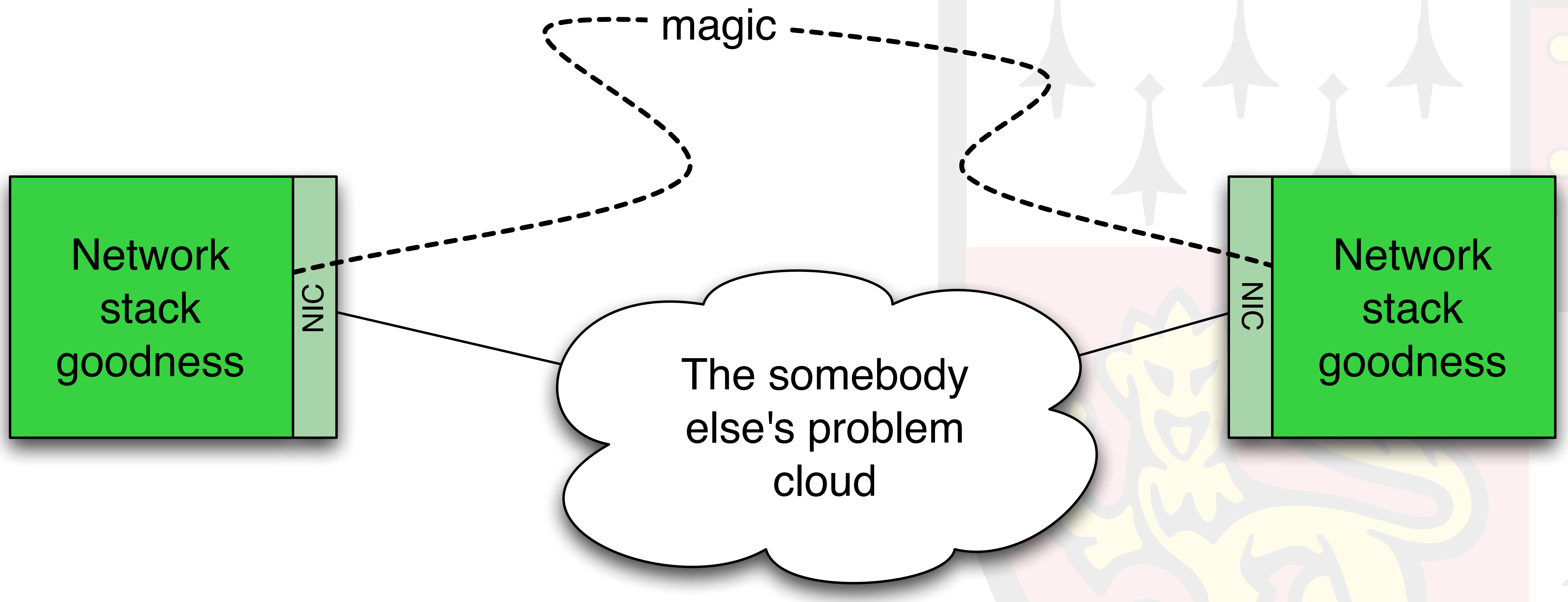
MSN 2010

Robert N. M. Watson
University of Cambridge

Portions of this work supported
by Juniper Networks, Inc.

UNIVERSITY OF
CAMBRIDGE

Idealised network for an OS developer



UNIVERSITY OF CAMBRIDGE

Things are getting a bit sticky at the end host*

* … and end host-like middle nodes: proxies, application firewalls, anti-spam, anti-virus, …

Packets-per-second (PPS) scales with bandwidth, but per-core limits reached

- ➪ Transition to multicore

Even today's bandwidth achieved only with protocol offload to the NIC

- ➪ But just specific protocols, workloads

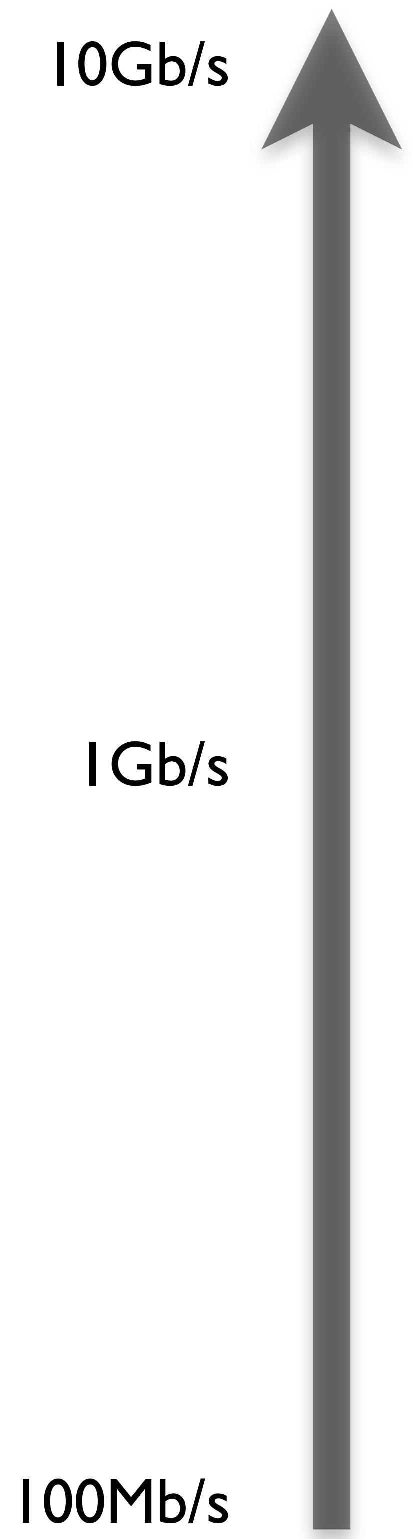
Contemporary network stack scalability themes

- Counting instructions → cache misses
- Lock contention → cache line contention
- Locking → finding parallelism opportunities
- Work ordering, classification, distribution
- NIC offload of even more protocol layers
- Vertical integrated work distribution/affinity

Why we love offload

Better performance,
no protocol changes*

* It sounds good so it must be true!



Full TCP, iSCSI, RDMA, ... offload

MultiQ: RSS, CAMs, MIPS, …

IP fragmentation/TSO/LRO

Checksum offload, VLAN en/decap

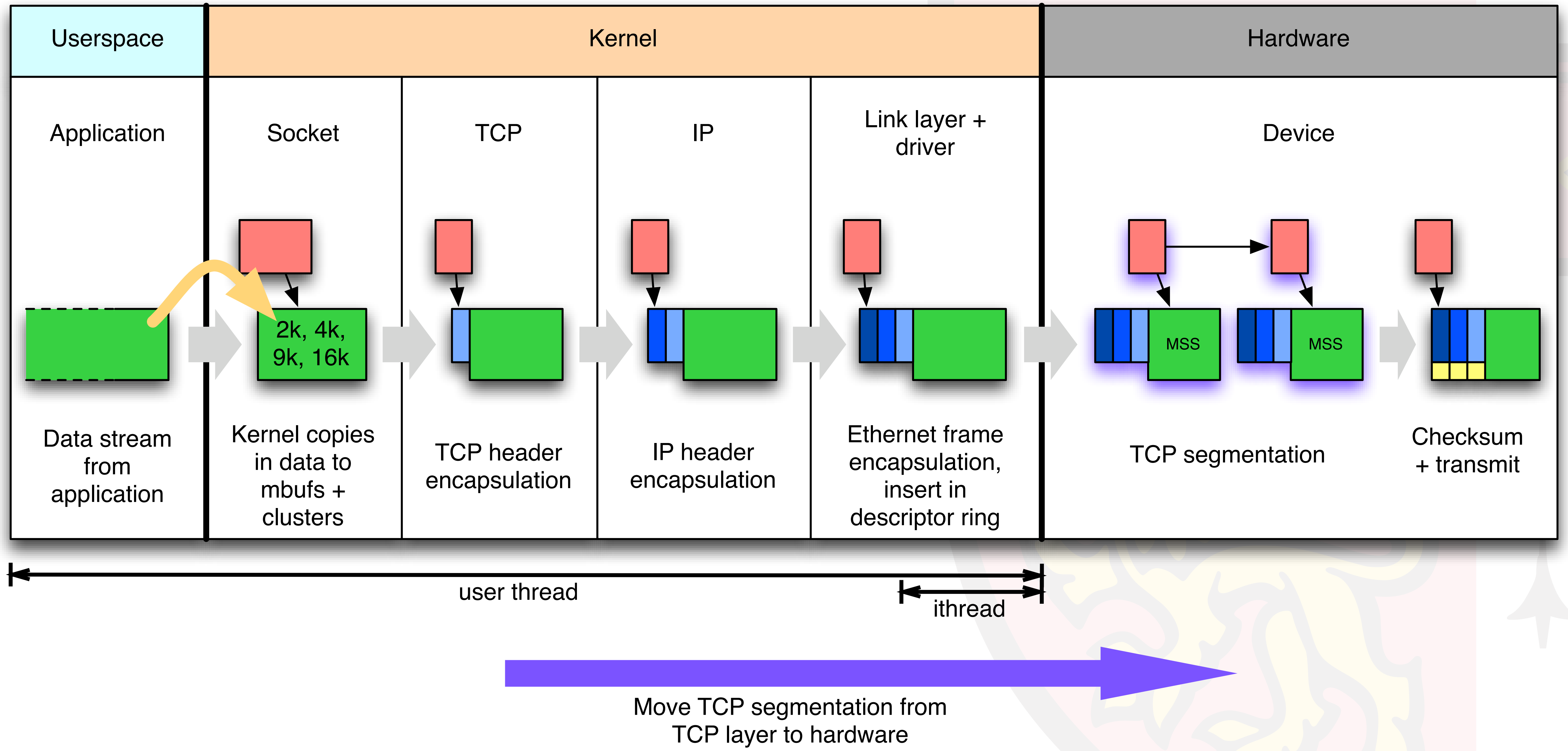
Interrupt moderation

PIO → DMA rings

UNIVERSITY OF CAMBRIDGE

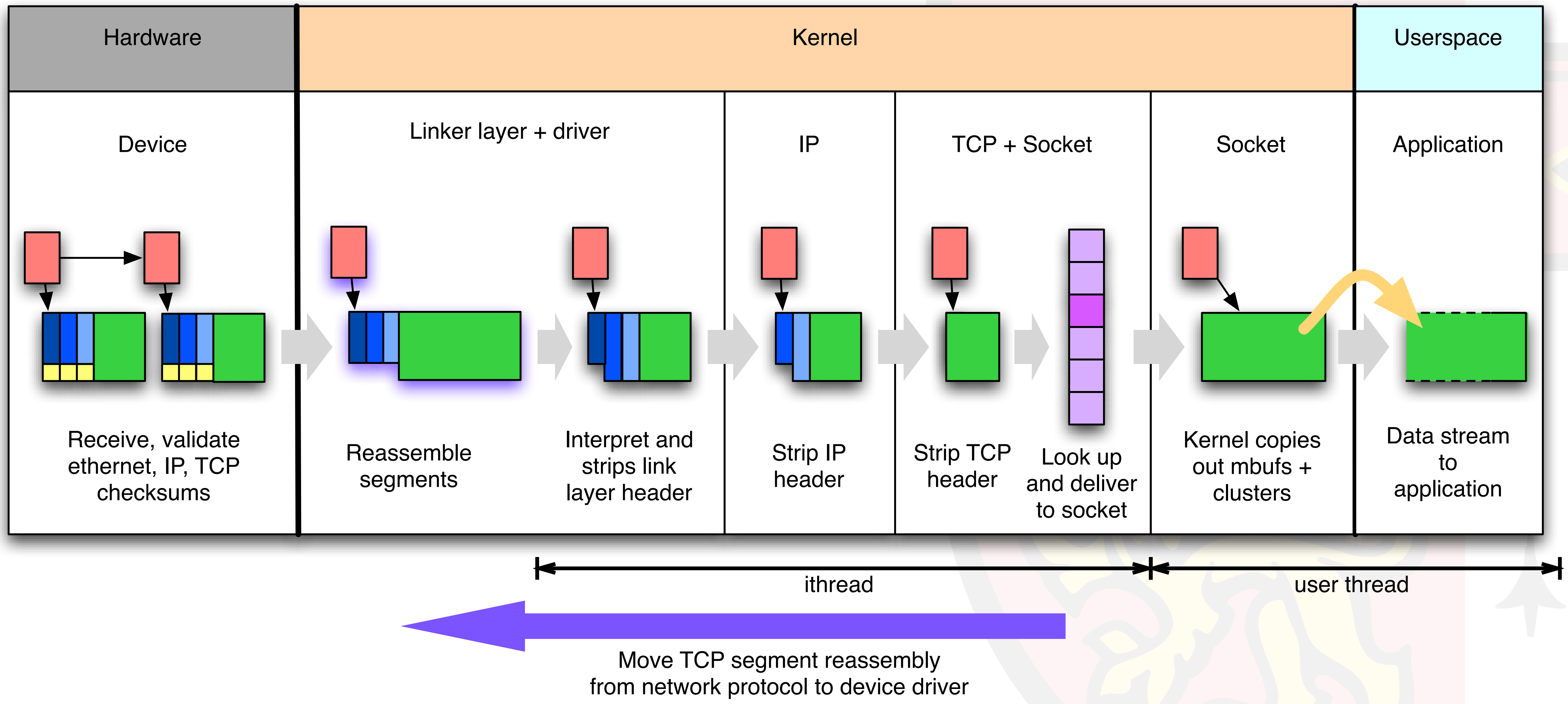
Reducing effective PPS with offload

TCP Segmentation Offload (TSO)

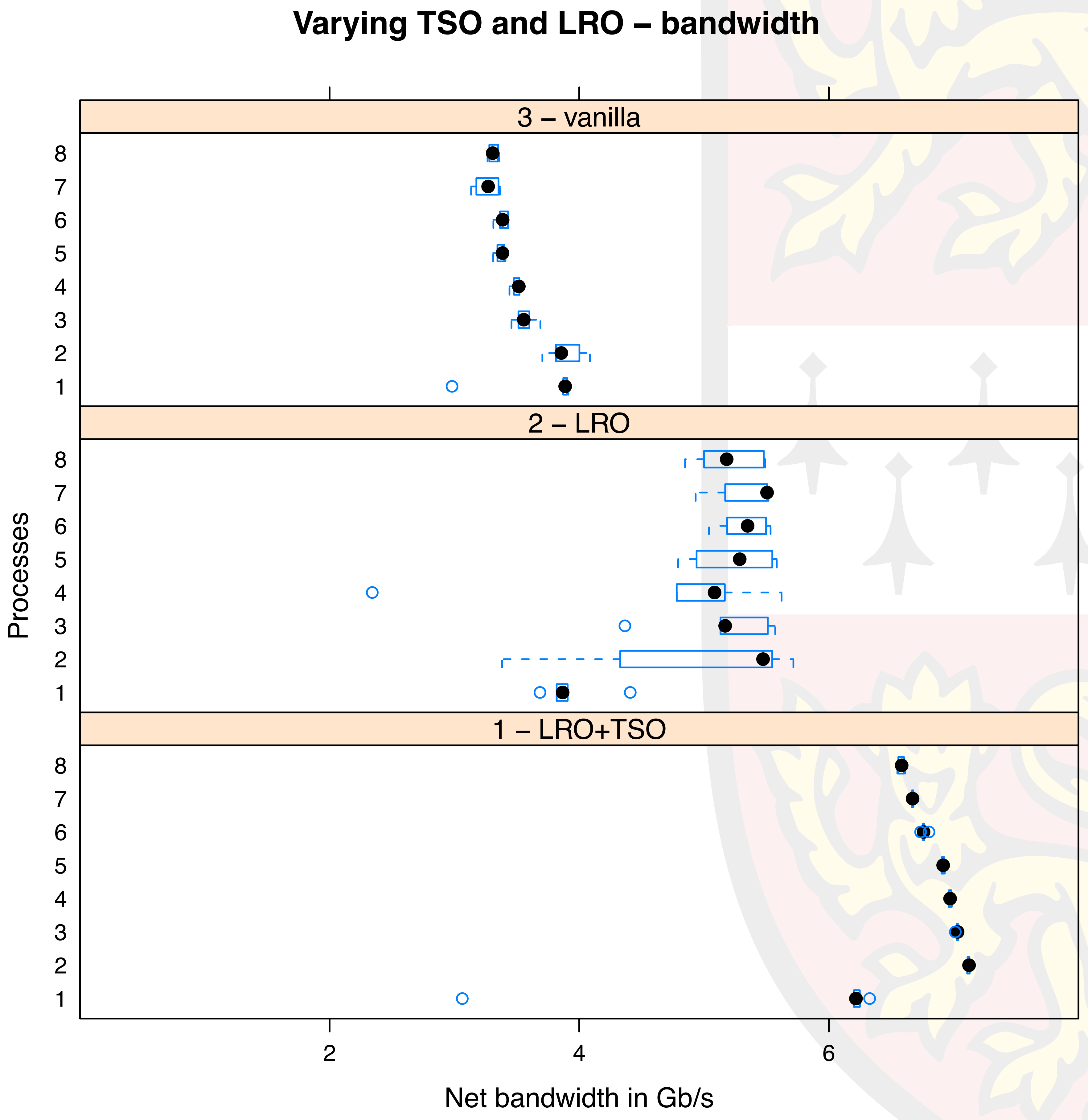


Reduce effective PPS to improve OS performance

Large Receive Offload (LRO)*


Hardware
Kernel
Userspace
Device
Linker layer + driver
IP
TCP + Socket
Socket
Application
Receive, validate ethernet, IP, TCP checksums
Reassemble segments
Interpret and strips link layer header
Strip IP header
Strip TCP header
Look up and deliver to socket
Kernel copies out mbufs + clusters
Data stream to application
ithread
user thread
Move TCP segment reassembly from network protocol to device driver

* Interestingly, LRO is often done in software



TSO and LRO off from now on

UNIVERSITY OF CAMBRIDGE

What about the wire protocol?

- Packet format remains the same
- Transmit/receive code essentially identical
- Just shifted segmentation/reassembly
- Effective ACK behaviour has changed!
    - ACK every 6-8 segments instead of every 2 segments!

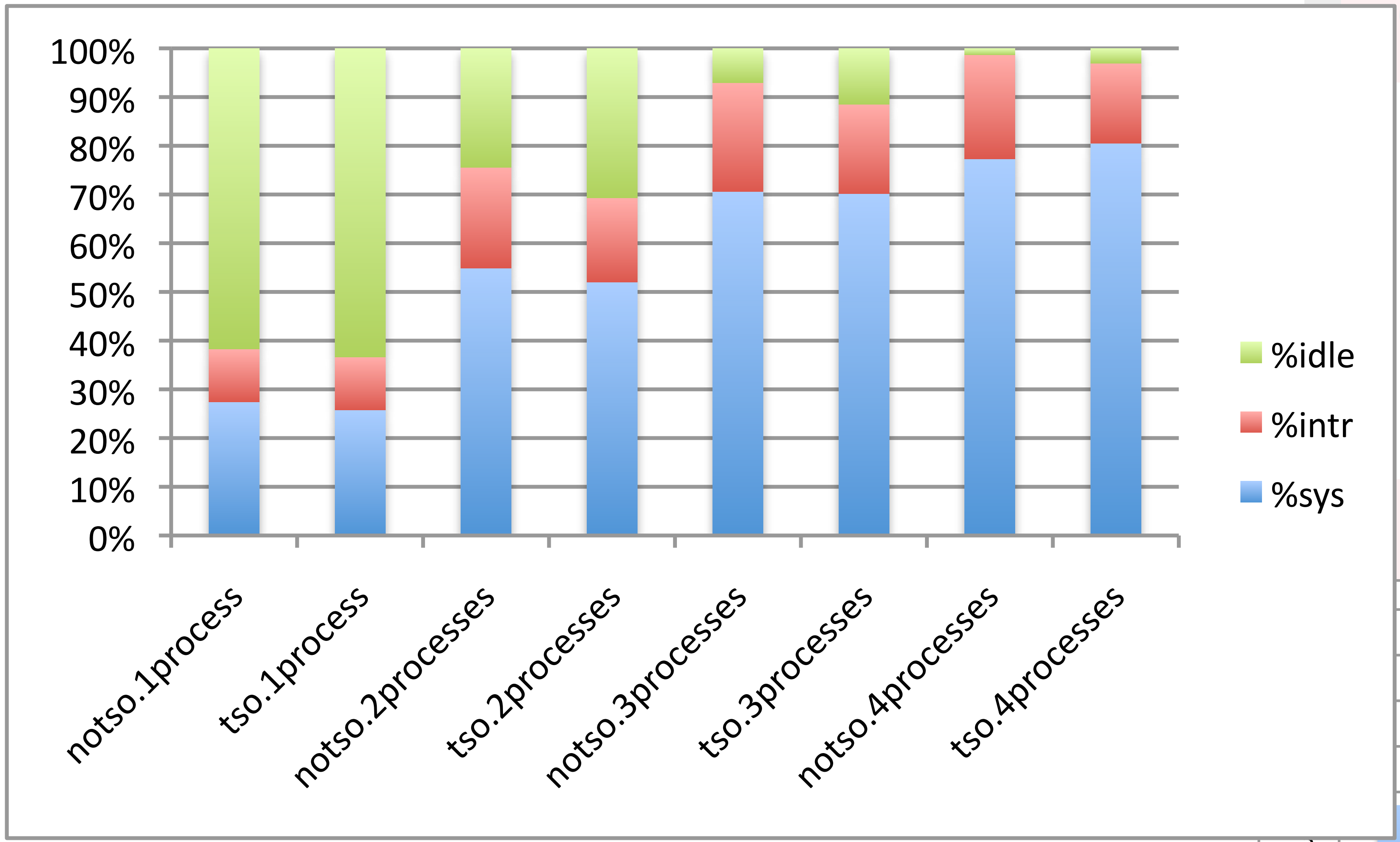
UNIVERSITY OF CAMBRIDGE

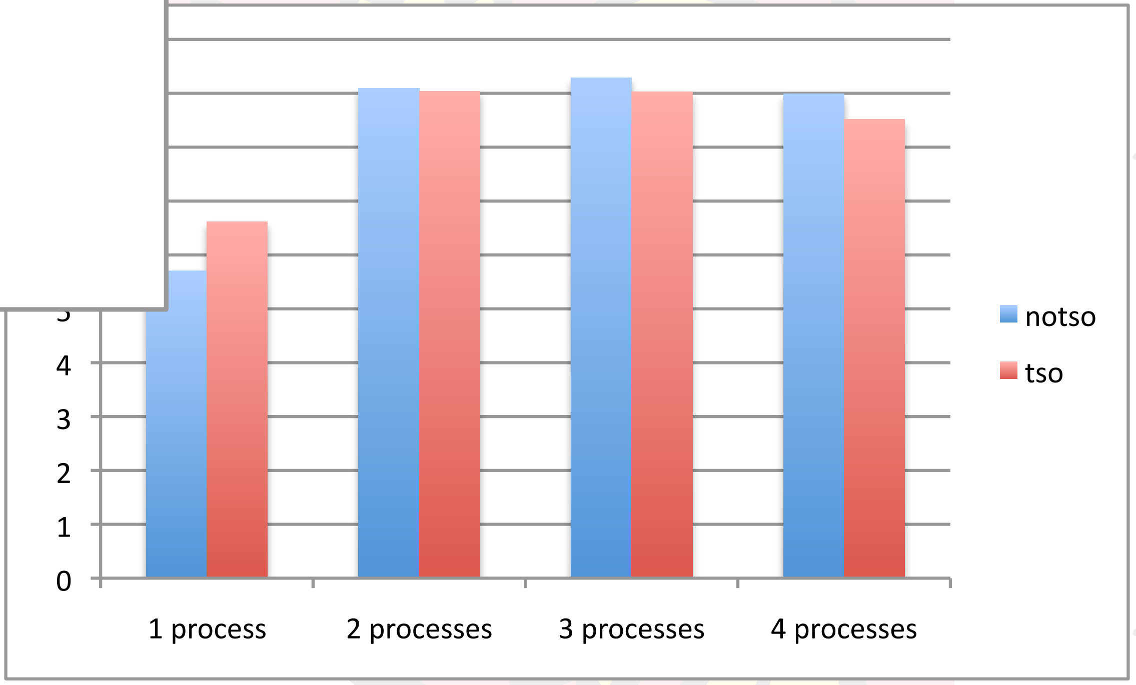
Managing contention *and* the search for parallelism*

* Again, try not to change the protocol…

UNIVERSITY OF CAMBRIDGE

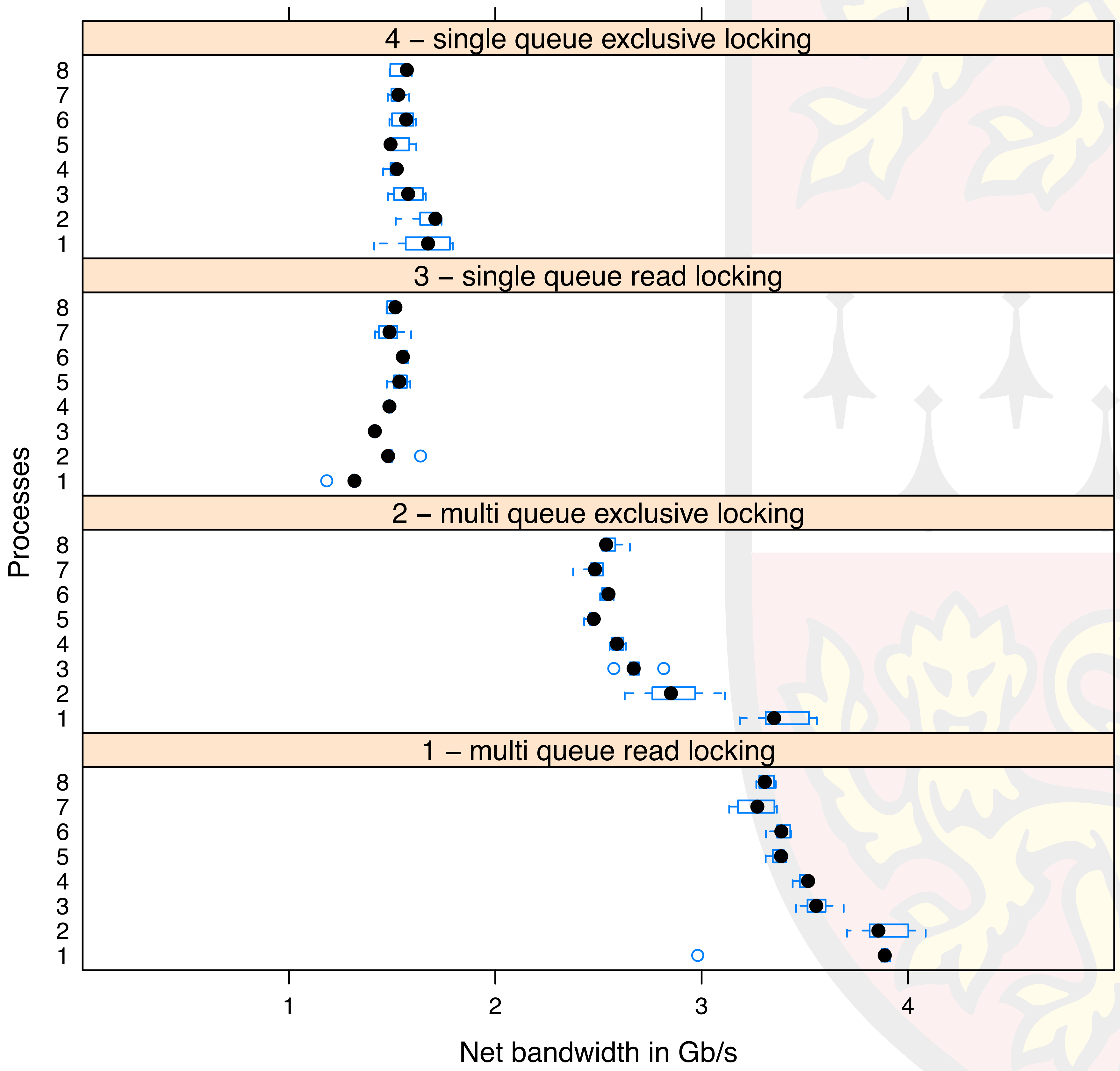
Lock contention





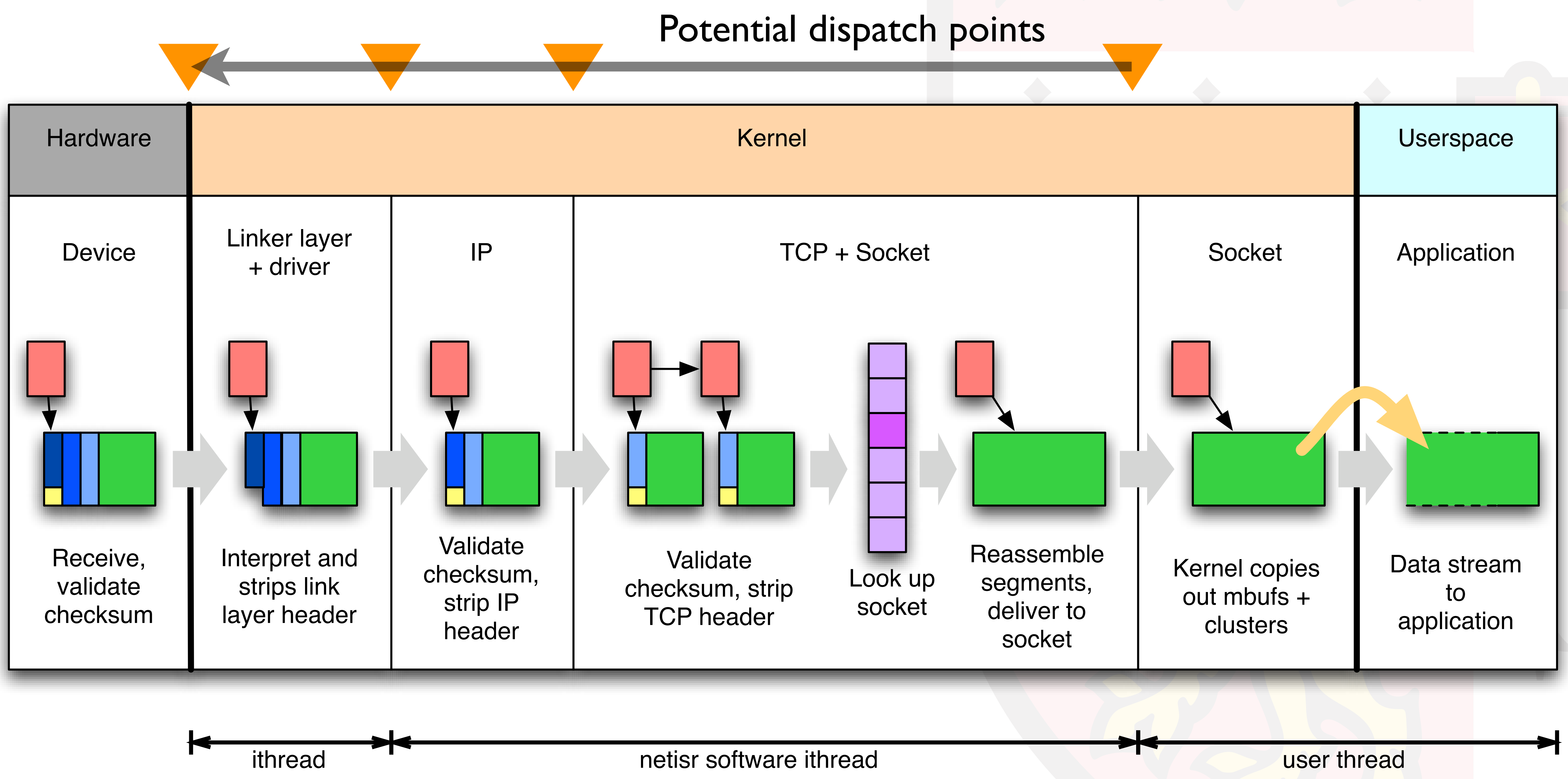
UNIVERSITY OF CAMBRIDGE

Varying locking strategy – bandwidth



UNIVERSITY OF CAMBRIDGE

TCP input path



Potential dispatch points
Hardware
Kernel
Userspace
Device
Linker layer + driver
IP
TCP + Socket
Socket
Application
Receive, validate checksum
Interpret and strips link layer header
Validate checksum, strip IP header
Validate checksum, strip TCP header
Look up socket
Reassemble segments, deliver to socket
Kernel copies out mbufs + clusters
Data stream to application
ithread
netisr software ithread
user thread

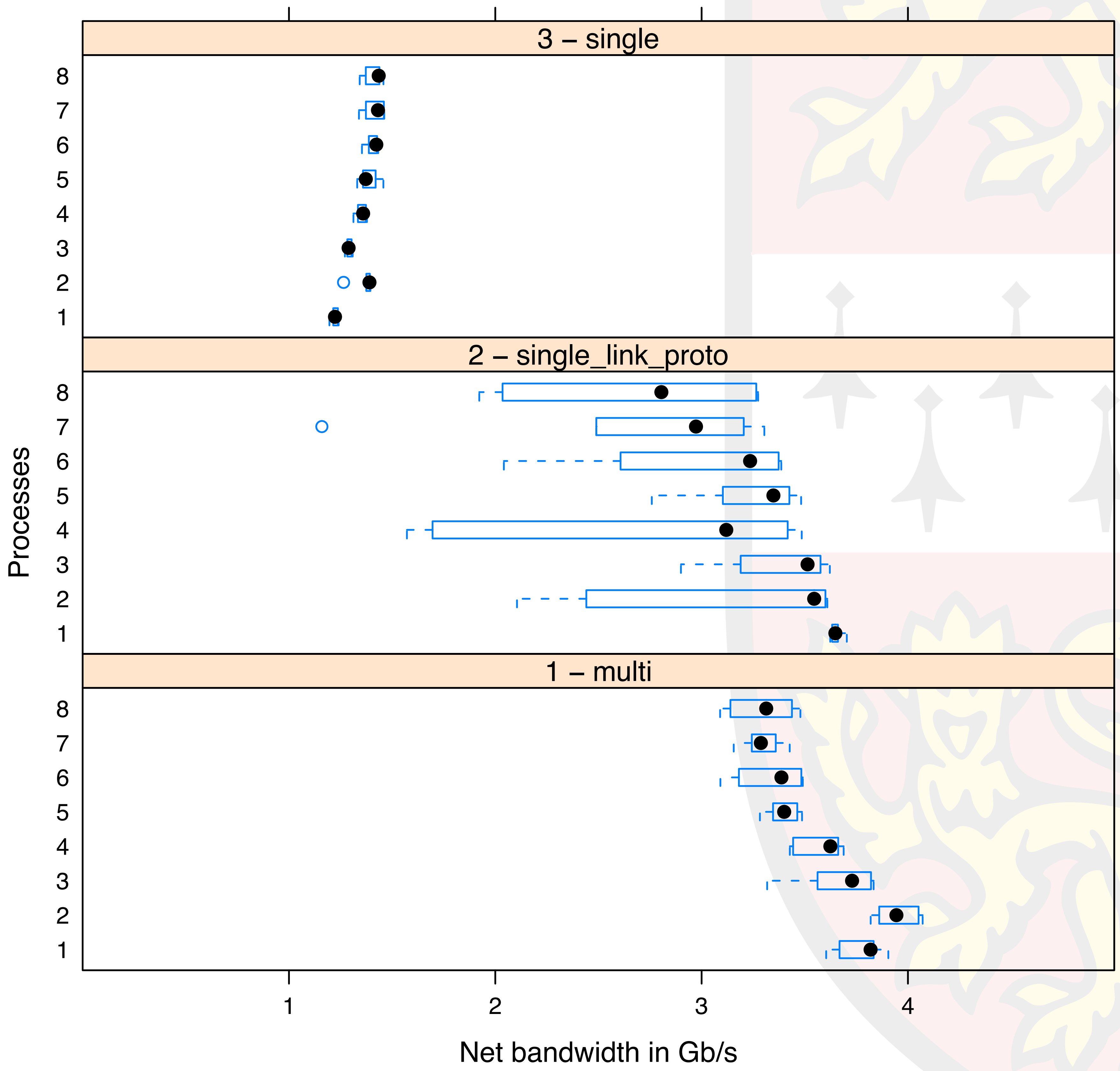
UNIVERSITY OF CAMBRIDGE

Work distribution

- Parallelism implies work distribution
  - Must keep work ordered
  - Establish flow-CPU affinity
- Microsoft Receive-Side Steering (RSS)
- More fine-grained solutions (CAMs, etc)

⚠ MTCP watch out! ⚠ The Toeplitz catastrophe

Varying dispatch strategy – bandwidth



Why we hate offload

"Layering violations" are not invisible

- Hardware bugs harder to work around
- Instrumentation below socket layer affected
  - BPF, firewalls, traffic management, etc.
  - Interface migration more difficult
- All your protocols were not created equal
- Not all TOEs equal: SYN, TIMEWAIT, etc.

UNIVERSITY OF CAMBRIDGE

Protocol implications

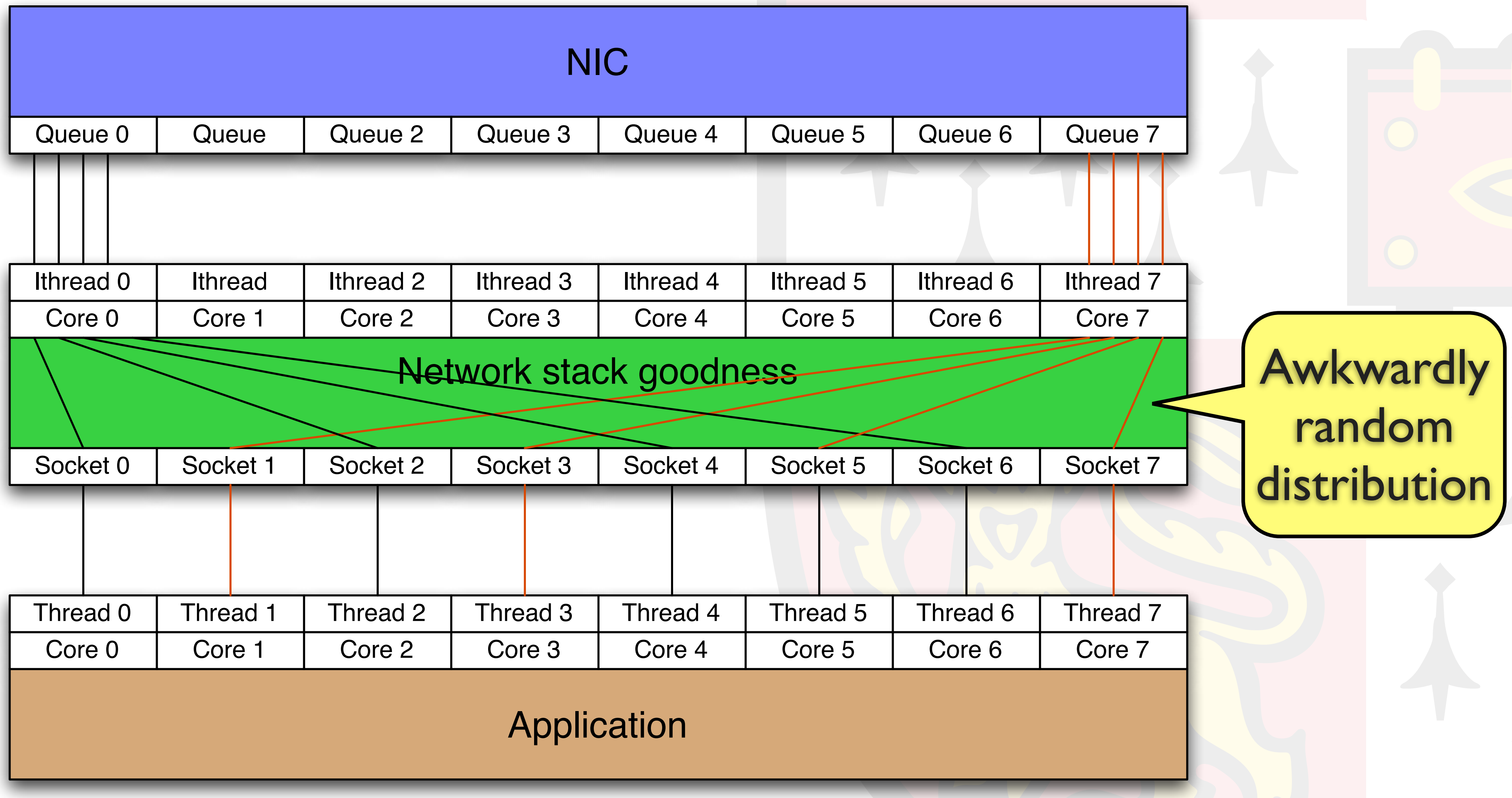
- Unsupported protocols and workloads see:
  - Internet-wide PMTU applied to PCI
  - Limited or no checksum offload
  - Ineffectual NIC-side load balancing
- Another nail in "deploy a new protocol" coffin? (e.g., SCTP, even multi-path TCP)
- Ideas about improving protocol design?

Structural problems

- Replicated implementation and maintenance responsibility
- Difficult field upgrade
- Host vs. NIC interop problems
- Composability problem for virtualisation
- Encodes flow affinity policies in hardware

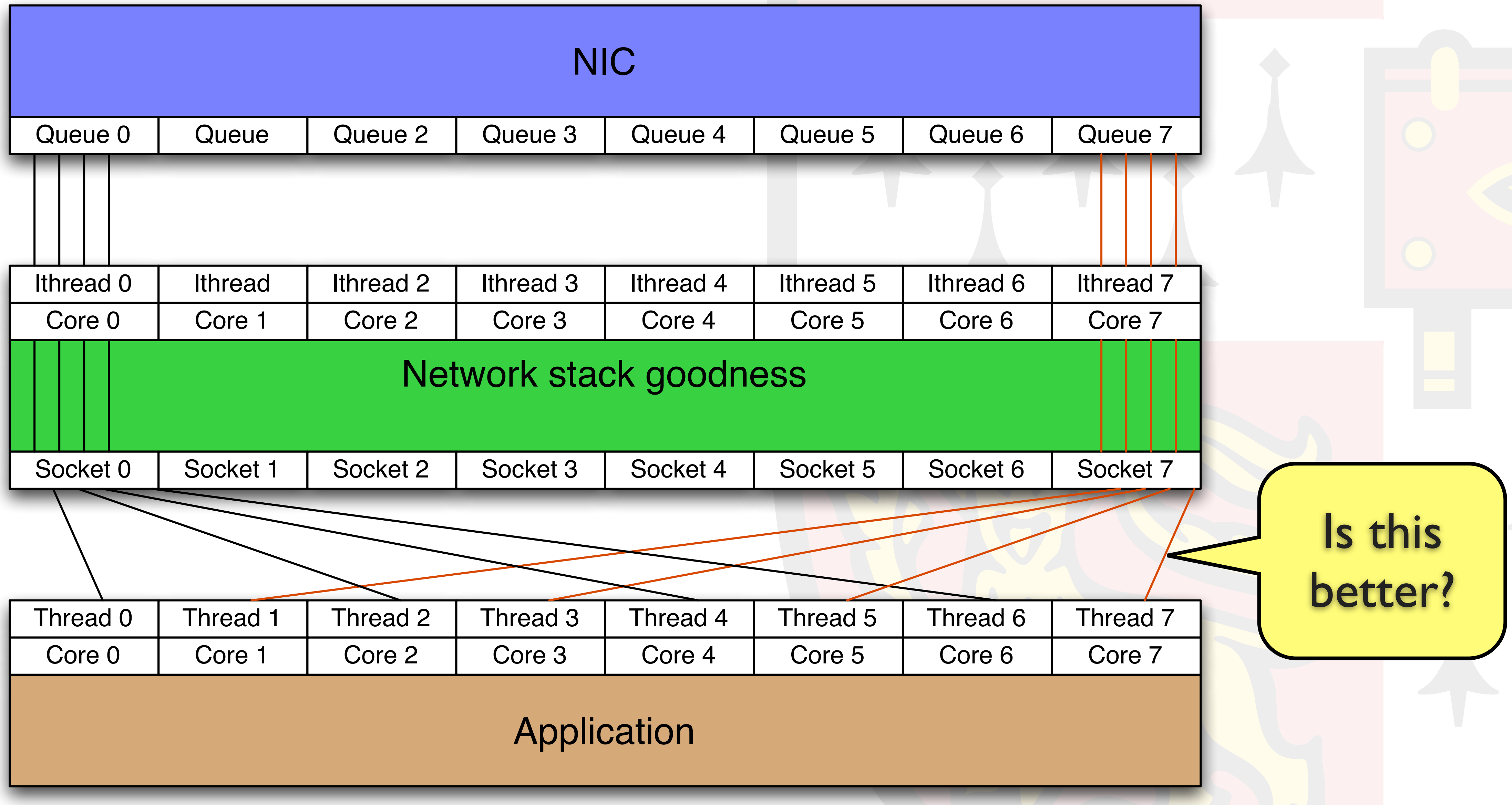
The vertical affinity problem

Hardware-only RSS



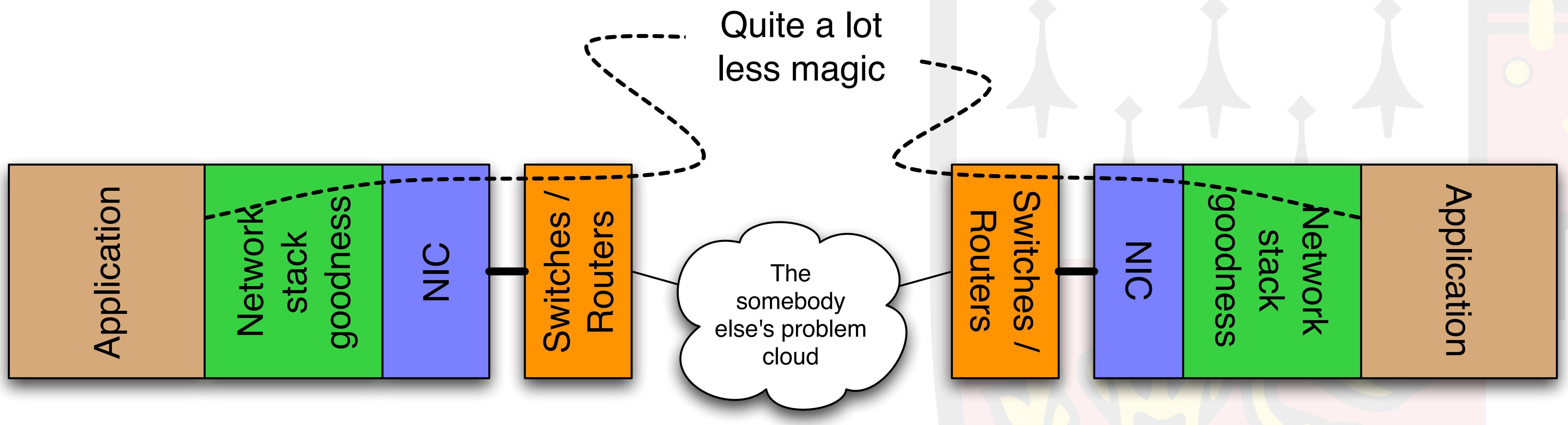
UNIVERSITY OF CAMBRIDGE

OS-aligned RSS



- Applications can express execution affinity
- How to align with network stack and network interface affinity?
- Sockets API inadequate; easy to imagine simple extensions but are they sufficient?
- How to deal with hardware vs. software policy mismatches?

Reality for an OS developer



UNIVERSITY OF CAMBRIDGE

Key research areas

- Explore programmability, debuggability, and traceability of heterogenous network stack
- Security implications of intelligent devices, diverse/new execution substrates, and single intermediate format
- Protocol impact: "end-to-end" endpoints shifting even further

Q&A