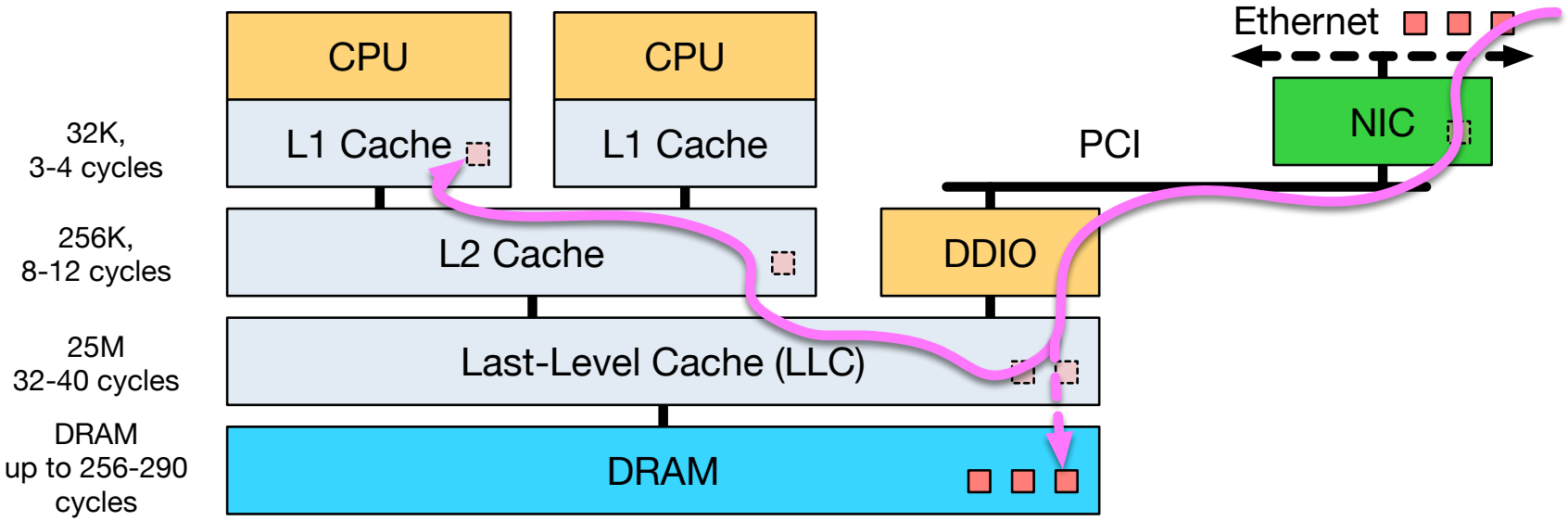
The Network Stack (1)

Lecture 5, Part 2: Network Stack Implementation

Prof. Robert N. M. Watson

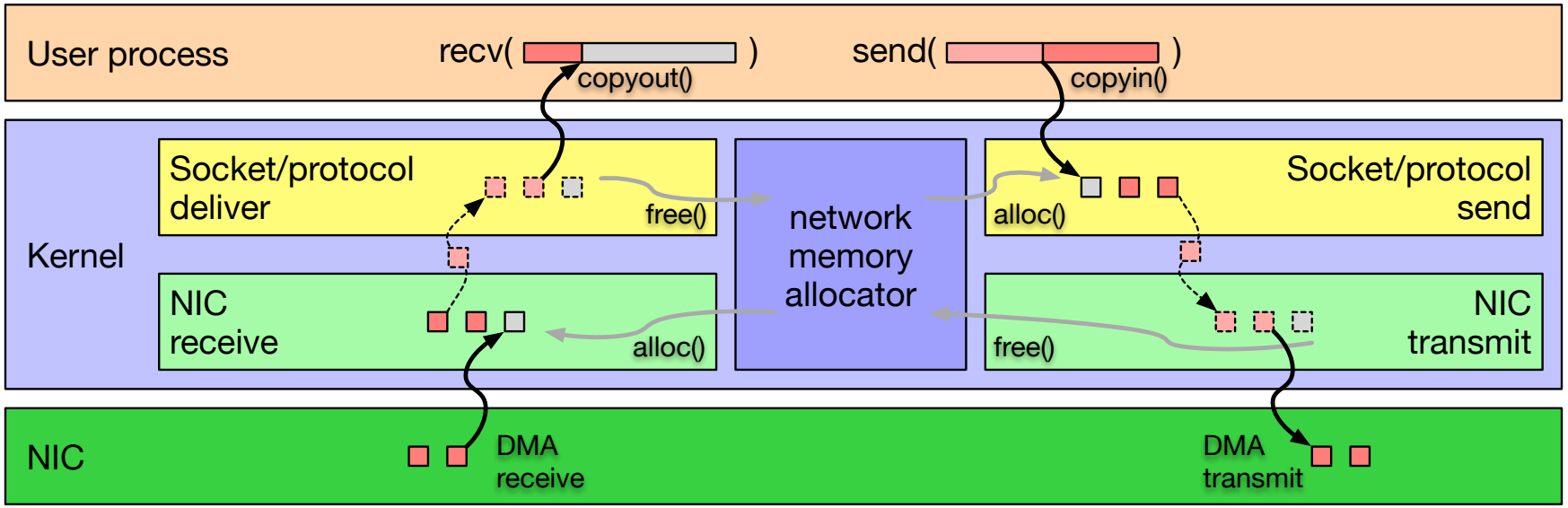
2022-2023

Memory flow in hardware



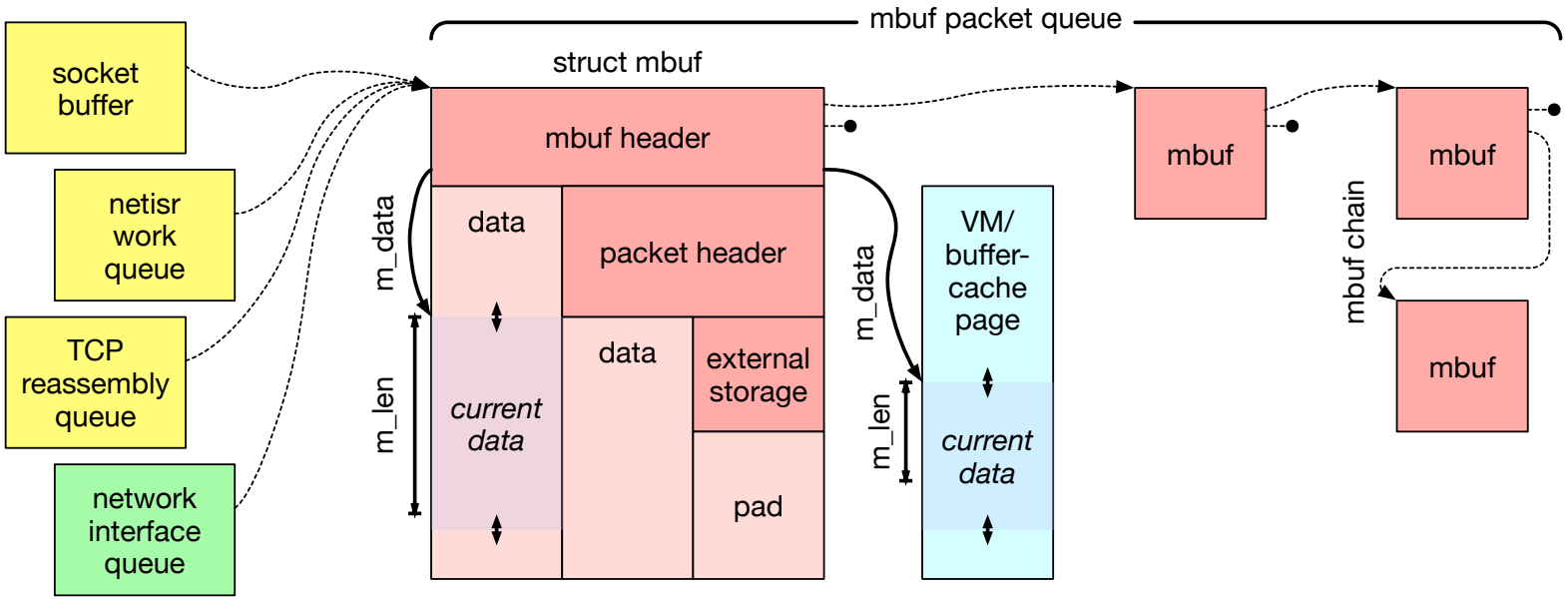
- Key idea: **follow the memory**
  - Historically, memory copying is avoided due to **instruction count**
  - Today, memory copying is avoided due to **cache footprint**
- Recent Intel CPUs push and pull DMA via the LLC ("DDIO")
  - If we differentiate 'send' and 'transmit', 'receive' vs. 'deliver', is this a good idea?
  - … it depends on the latency between DMA and processing

Memory flow in software



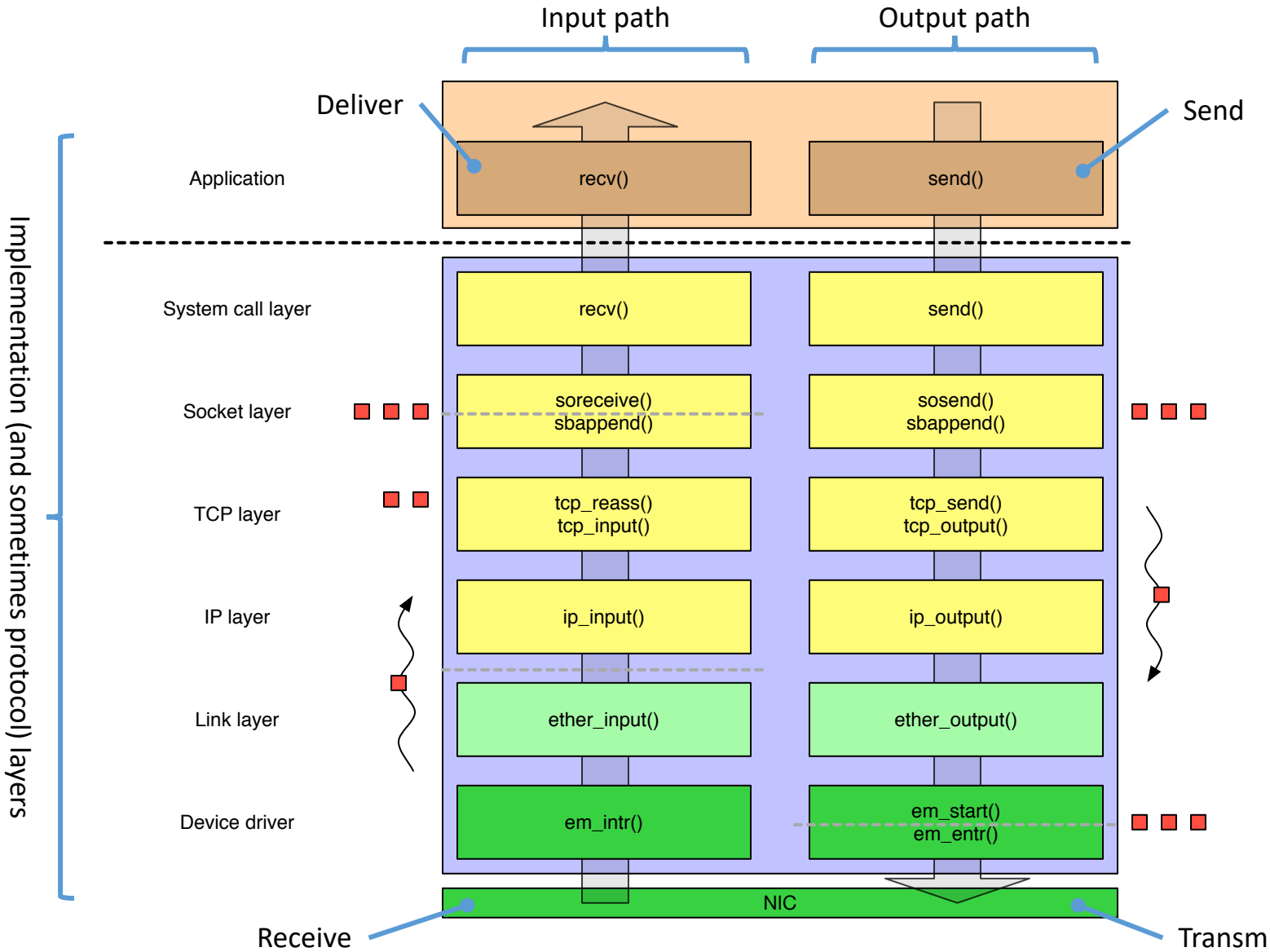
- Socket API implies **one software-driven copy** to/from user memory
  - Historically, zero-copy VM tricks for socket API ineffective
- Network buffers cycle through the slab allocator
  - Receive: allocate in NIC driver, free in socket layer
  - Transmit: allocate in socket layer, free in NIC driver
- **DMA performs second copy**; can affect cache/memory bandwidth
  - NB: what if packet-buffer working set is larger than the cache?

The mbuf abstraction

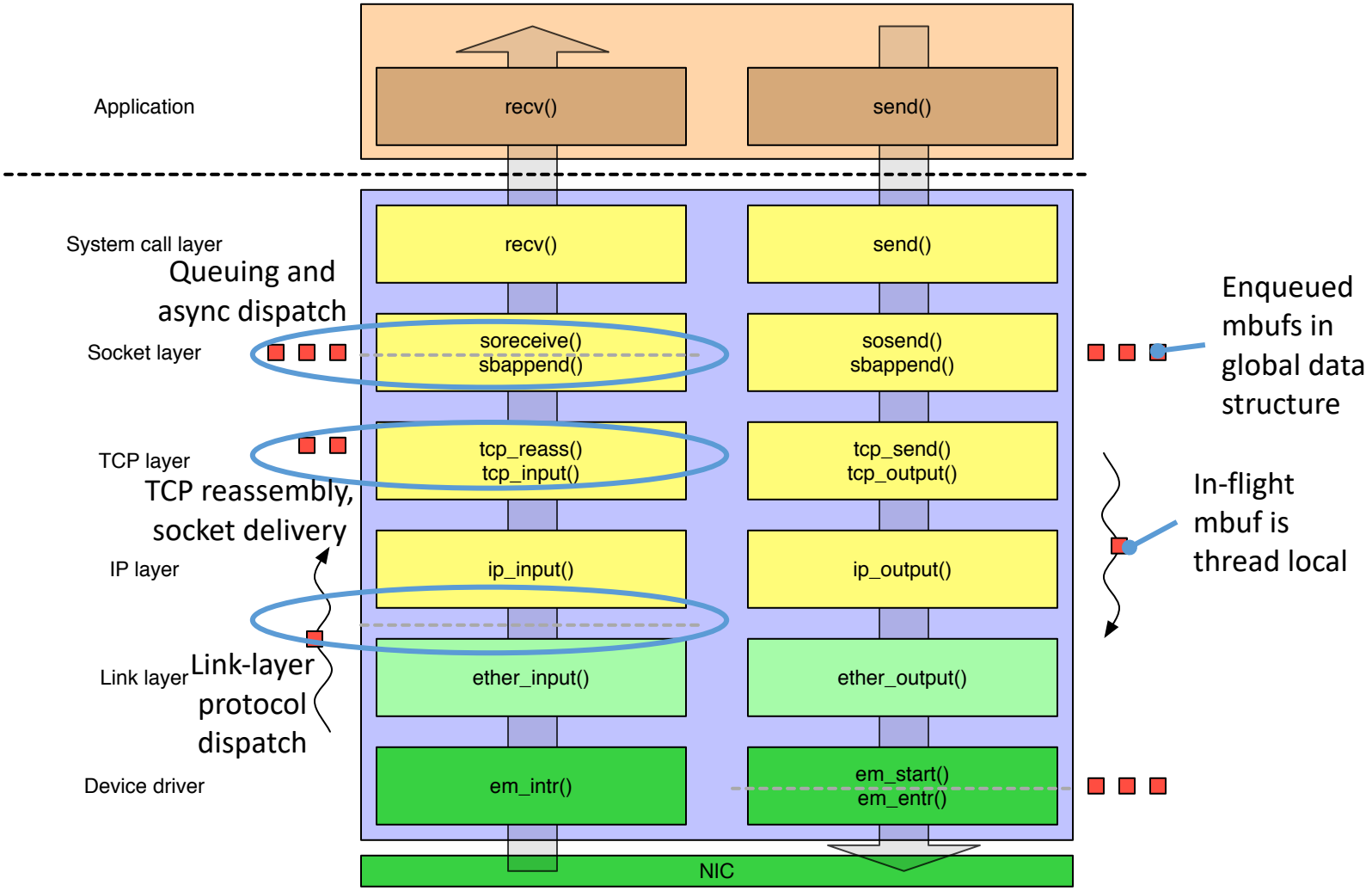


- Unit of **work allocation and distribution** throughout the stack
- mbuf chains represent in-flight packets, streams, etc.
  - Operations: alloc, free, prepend, append, truncate, enqueue, dequeue
  - Internal or external data buffer (e.g., VM page)
  - Reflects bi-modal packet-size distribution (e.g., TCP ACKs vs data)
- Similar structures in other OSes – e.g., skbuff in Linux

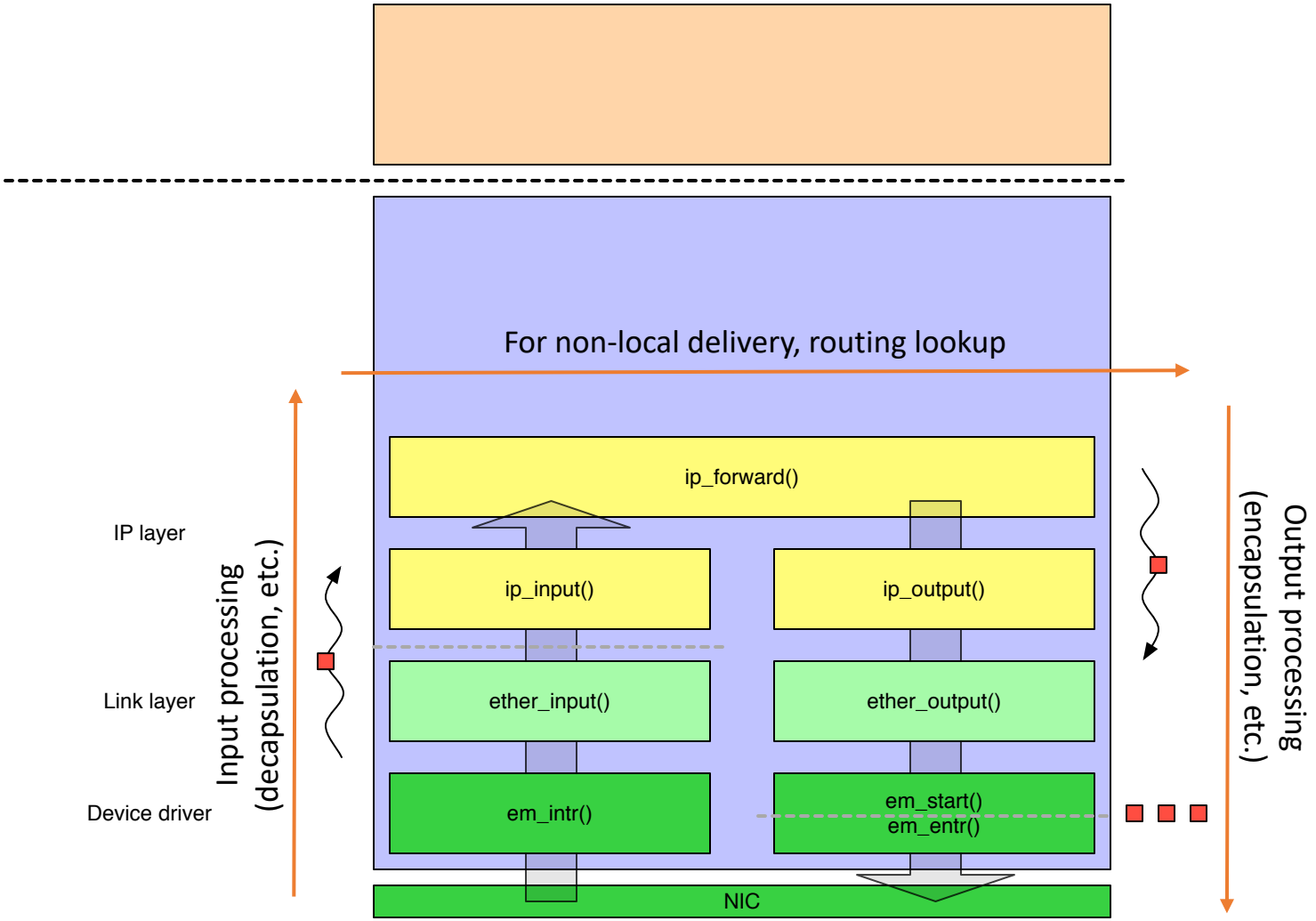
Send/receive paths in the network stack (1/2)



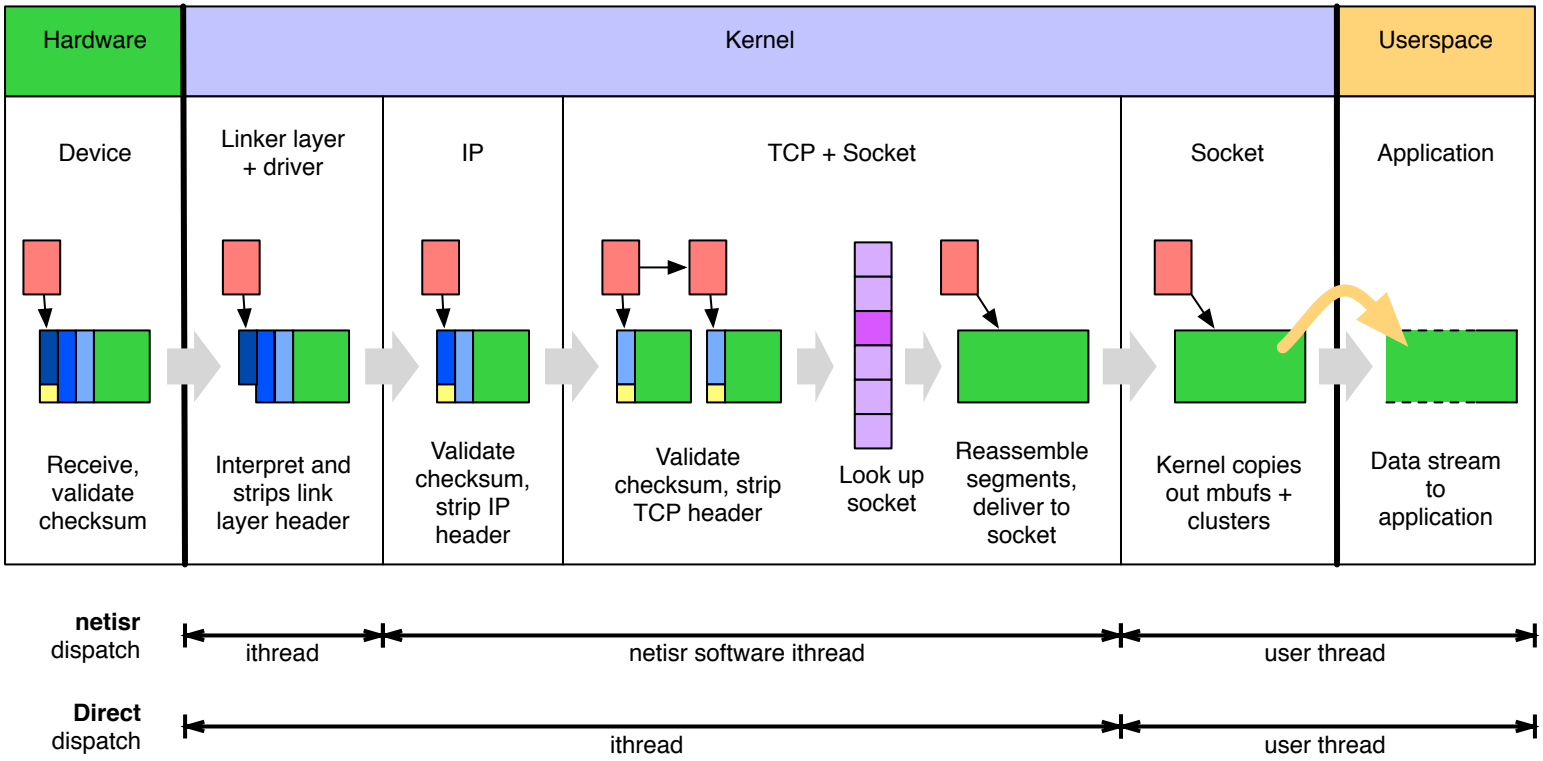
Send/receive paths in the network stack (2/2)



Forwarding path in the network stack

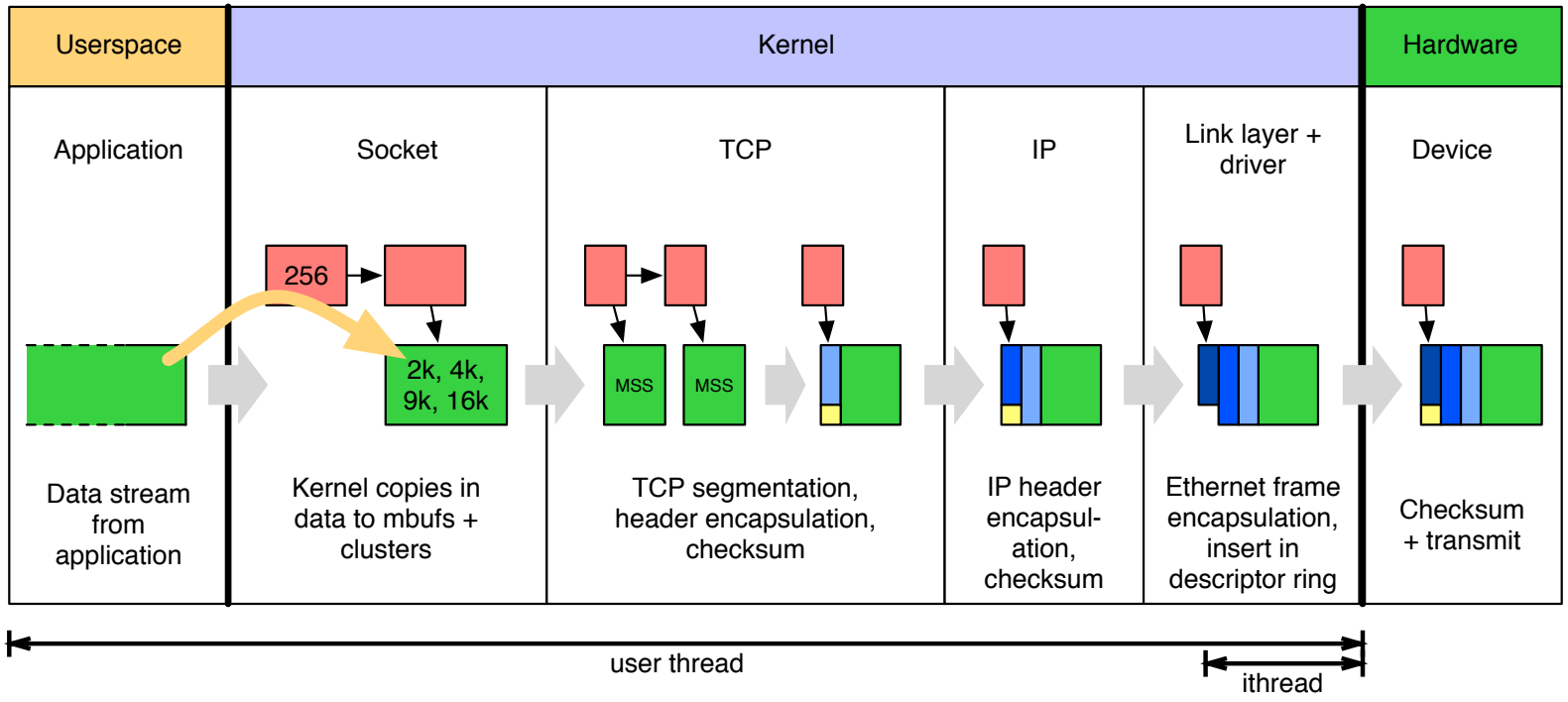


Work dispatch: input path



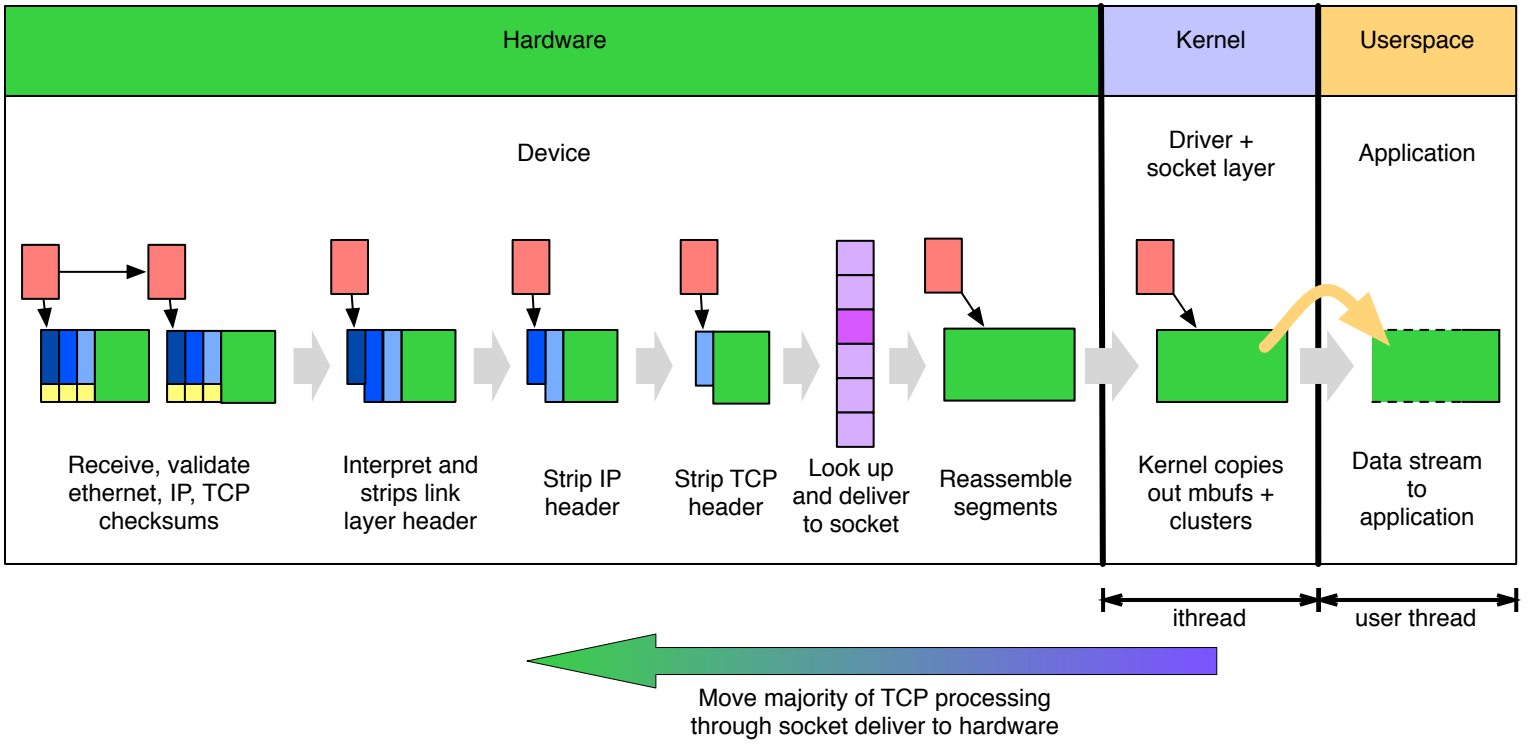
- **Deferred dispatch**: ithread → netisr thread → user thread
- **Direct dispatch**: ithread → user thread
  - Pros: reduced latency, better cache locality, drop early on overload
  - Cons: reduced parallelism and work placement opportunities

Work dispatch: output path



- Fewer deferred dispatch opportunities implemented
  - (Deferred dispatch on device-driver handoff in new `iflib` KPIs)
- Gradual shift of work from software to hardware
  - Checksum calculation, segmentation, …

Work dispatch: TOE input path



- Kernel provides socket buffers and resource allocation
- Remainder, including state, retransmissions, etc., in NIC
- But: two network stacks? Less flexible/updateable structure?
  - Better with an explicit HW/SW architecture – e.g., Microsoft Chimney