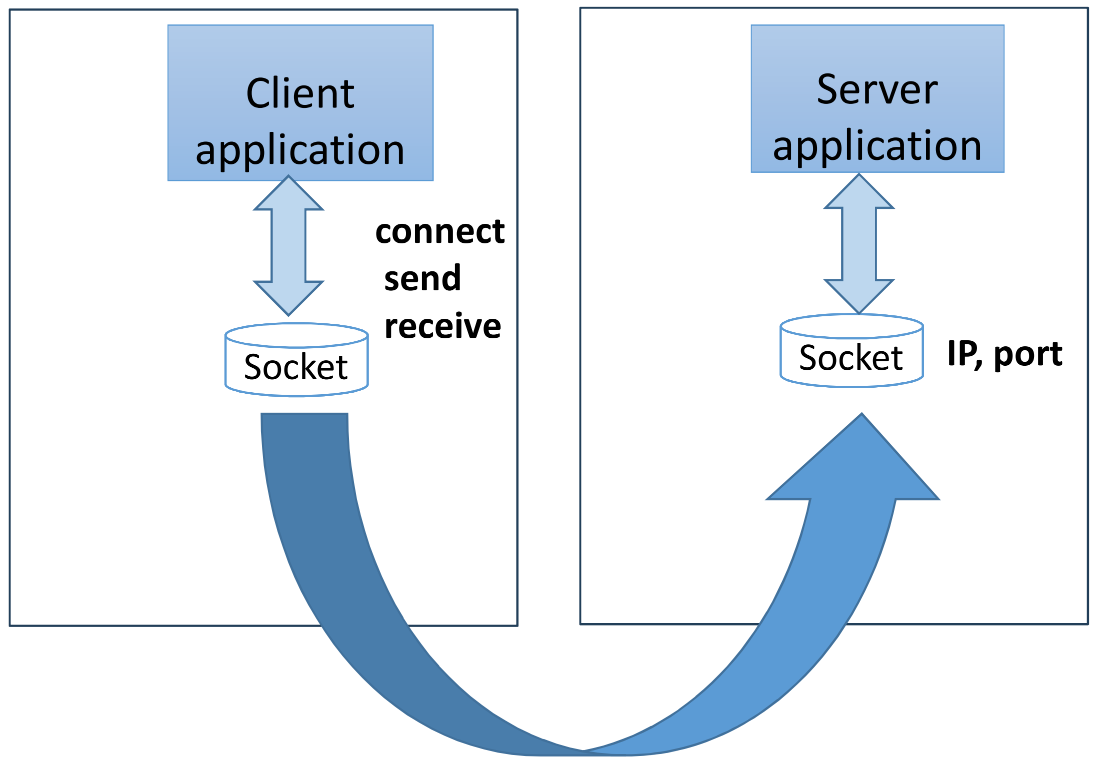
Network I/O subsystem

Mythili Vutukuru
CSE, IIT Bombay

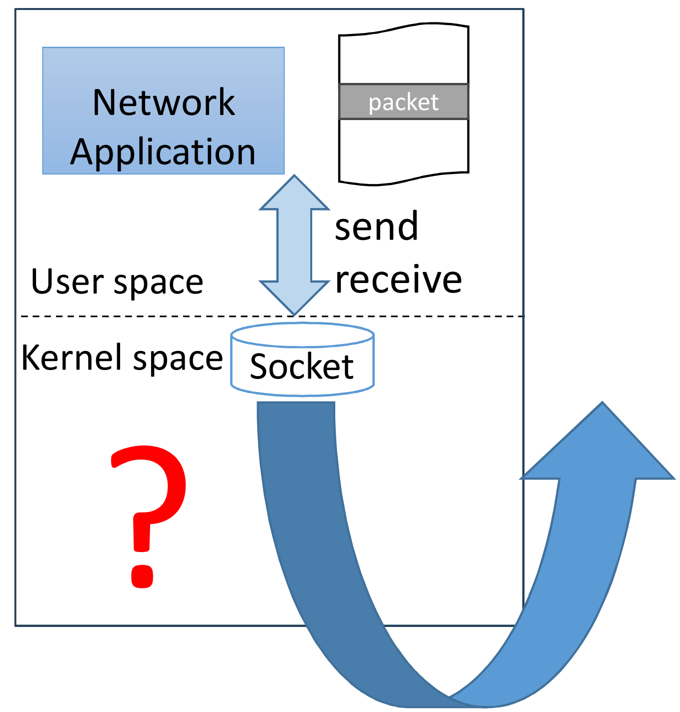
Networking applications

- Networking applications: web server, email client, browser etc..
- Exchange network packets via APIs like sockets
- Servers open sockets at well known IP address + port number
- Socket of web client connects to socket at web server, send and receive messages



What happens inside the kernel?

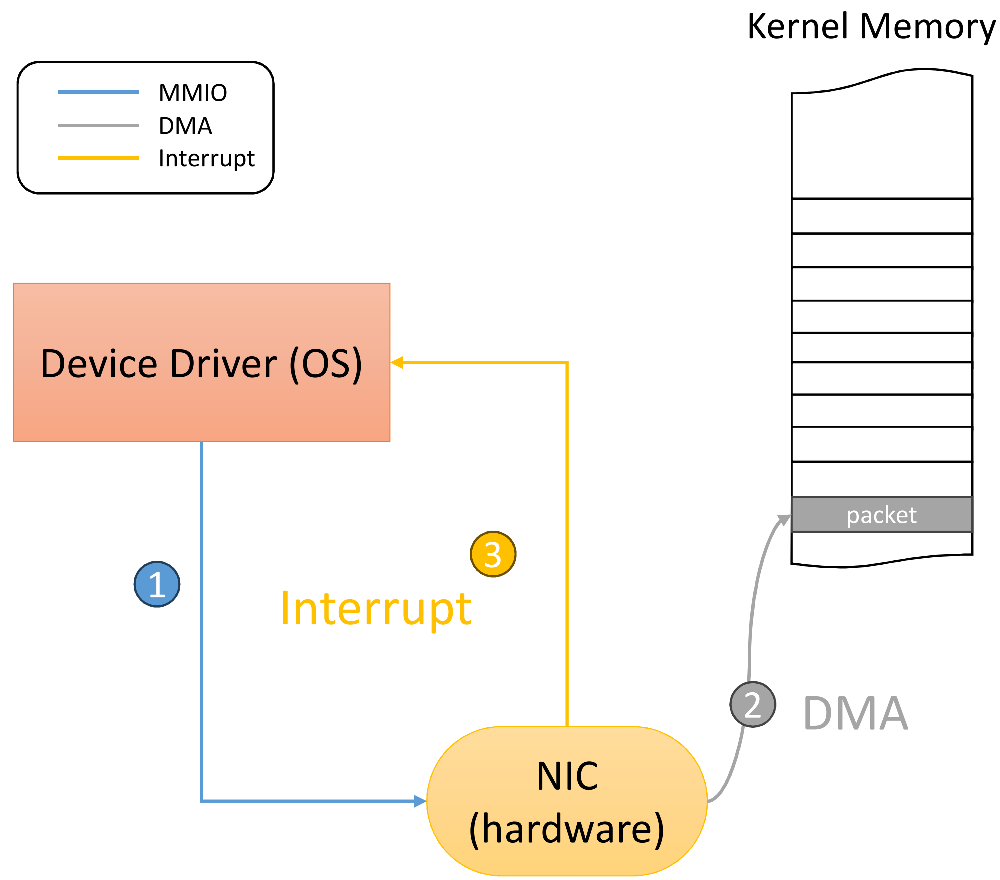
- What happens when you send and receive data through a socket?
- The story of what happens over the network will be covered in your networking course
- What happens at the sender and receiver end host kernel network stack?
- Many recent advances, to help kernel keep up with increasing network speeds



Outside end host: switching, routing, congestion control

Device drivers

- Device driver manages interaction between NIC (network interface card) and software
- Configures NIC via memory mapped I/O (MMIO)
- NIC performs Direct Memory Access (DMA) of network packets into kernel memory
- NIC raises interrupt to indicate reception of packets
- We will discuss only RX path here



Interrupt handling

- How are interrupts handled?
  - CPU is running process P and interrupt arrives
  - CPU saves context of P, runs OS code to handle interrupt in kernel mode
  - Restore context of P, resume P in user mode
- Interrupt handling code is part of OS device drivers
- Network device drivers handle interrupts from NICs

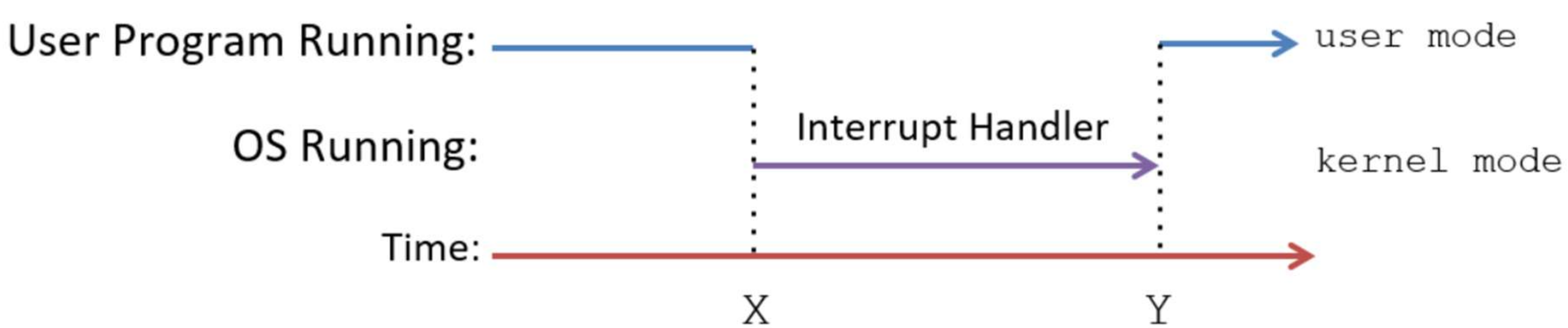


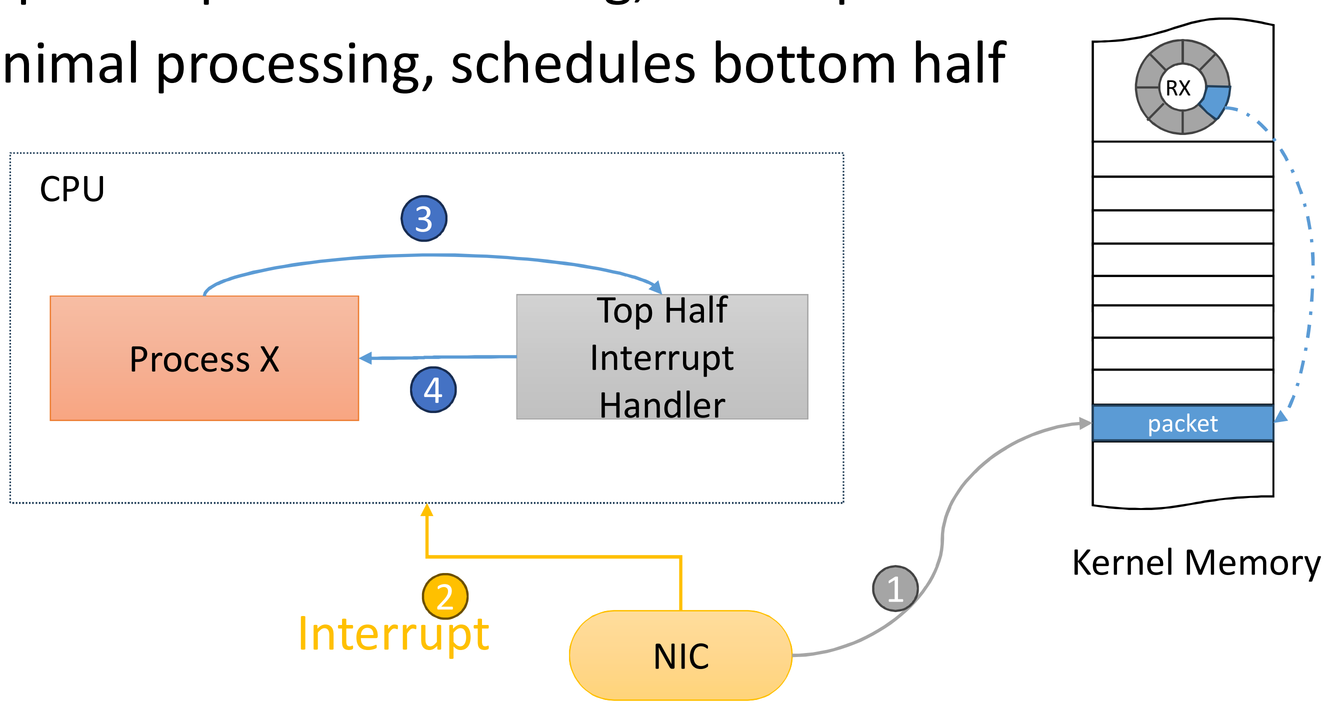
Image credit: Dive into Systems, by Mathews, Newhall, Webb

Network Interrupt handling

- Interrupt handling from NIC involves lot of work
  - Processing information about the network, congestion control, …
- To avoid excessive disruption to interrupted process, NIC interrupt handling split into two parts
- Top half interrupt handler acknowledges interrupt, does minimal processing, disables future interrupts
- Top half schedules a kernel process for full interrupt handling, called bottom half interrupt handler
- Bottom half processes all packets received so far, re-enables interrupts

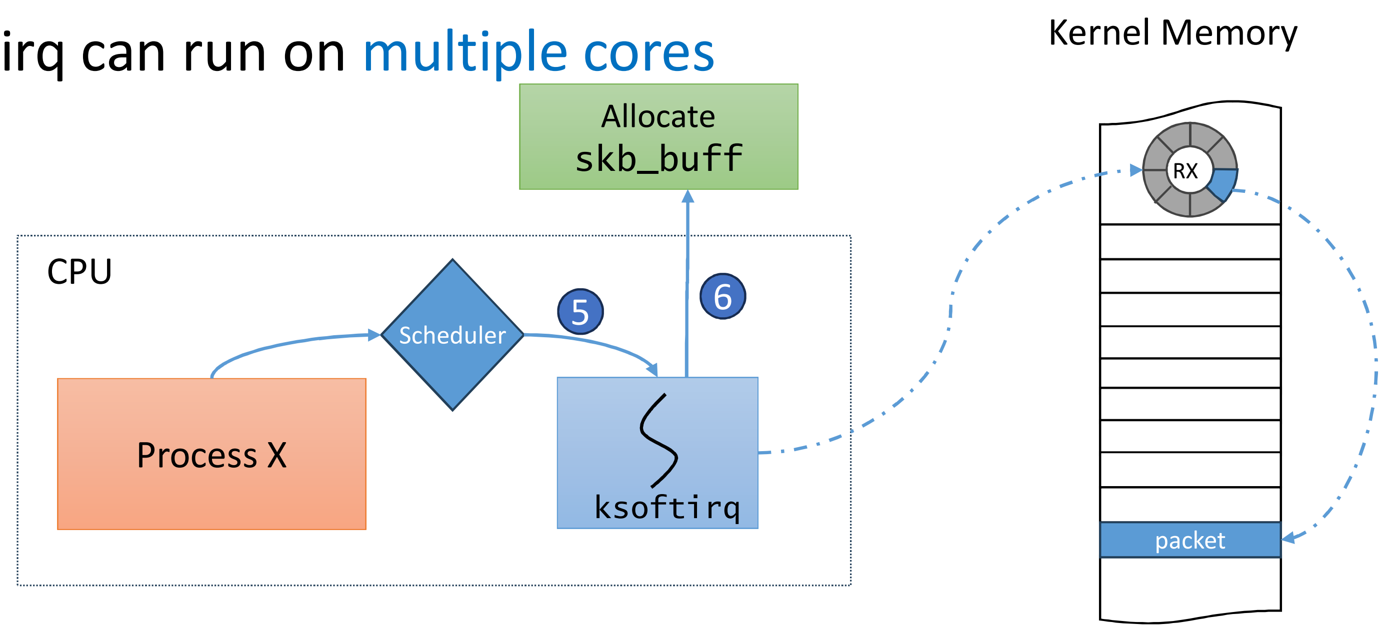
Device driver rings

- NIC and kernel exchange information about packets via TX/RX "rings"
- RX ring: circular array containing pointers to received packets
- NIC does DMA, updates pointer in RX ring, interrupts
- Top half does minimal processing, schedules bottom half



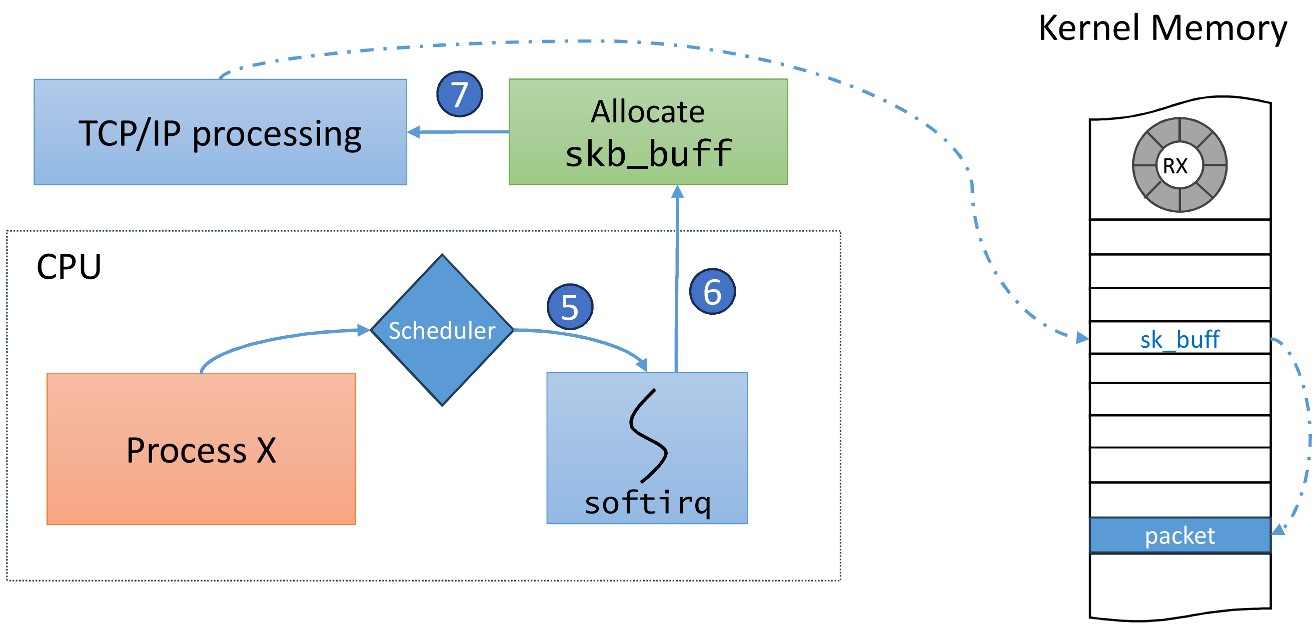
Bottom half interrupt handler

- Bottom half or ksoftirq process scheduled when CPU is free
- Processes all packets collected in the RX ring since the last round
    - Allocates socket buffer (sk_buff) structure for each packet
    - Socket buffer contains pointer to different fields (headers) in the packet
- Interrupt + ksoftirq can run on multiple cores



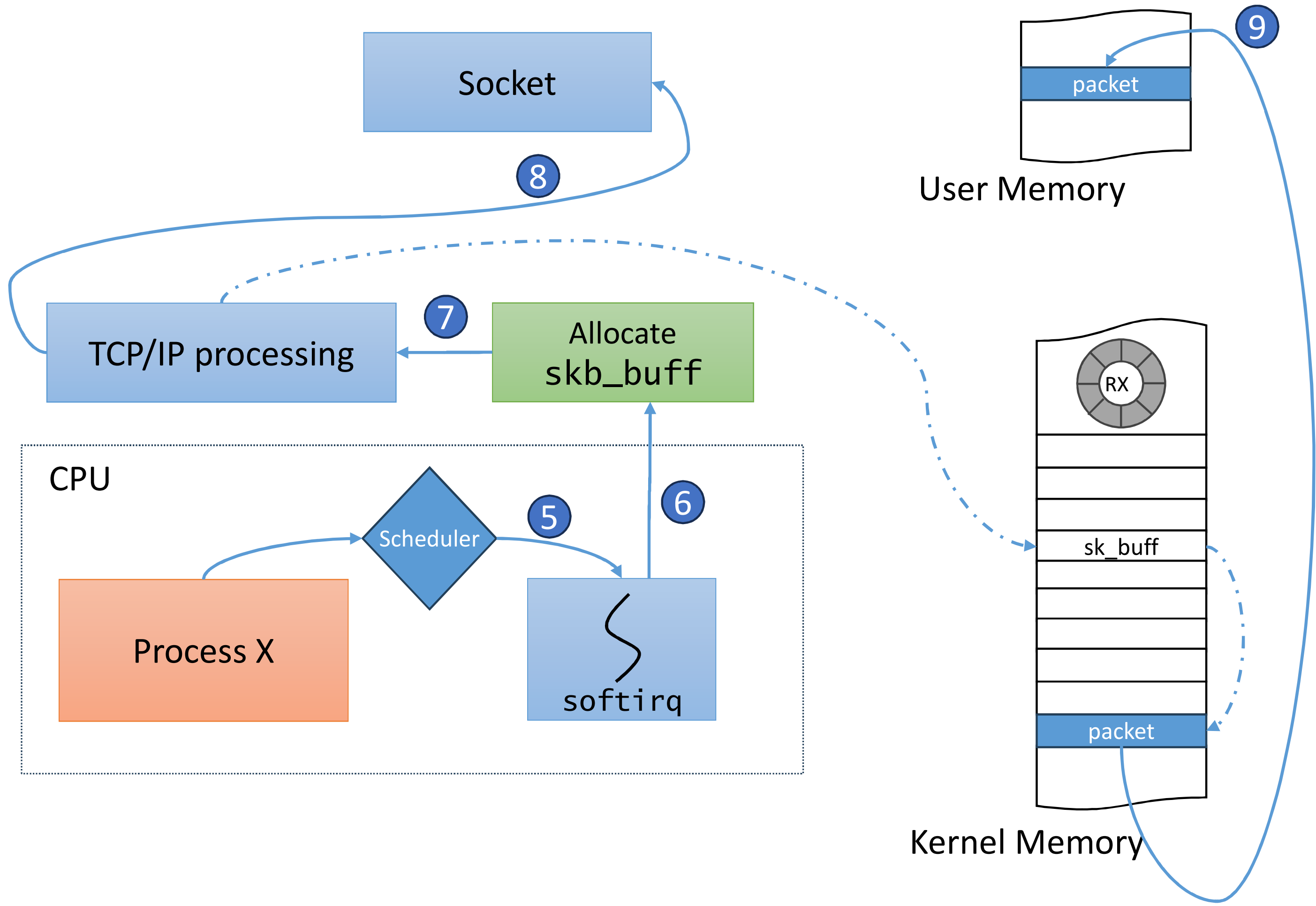
Network layer processing

- Bottom half interrupt handler performs all the network processing
  - Parsing and checking packet headers in sk_buff structure
  - IP routing, TCP reliability and congestion control algorithms (you will learn more about this in the networking course)



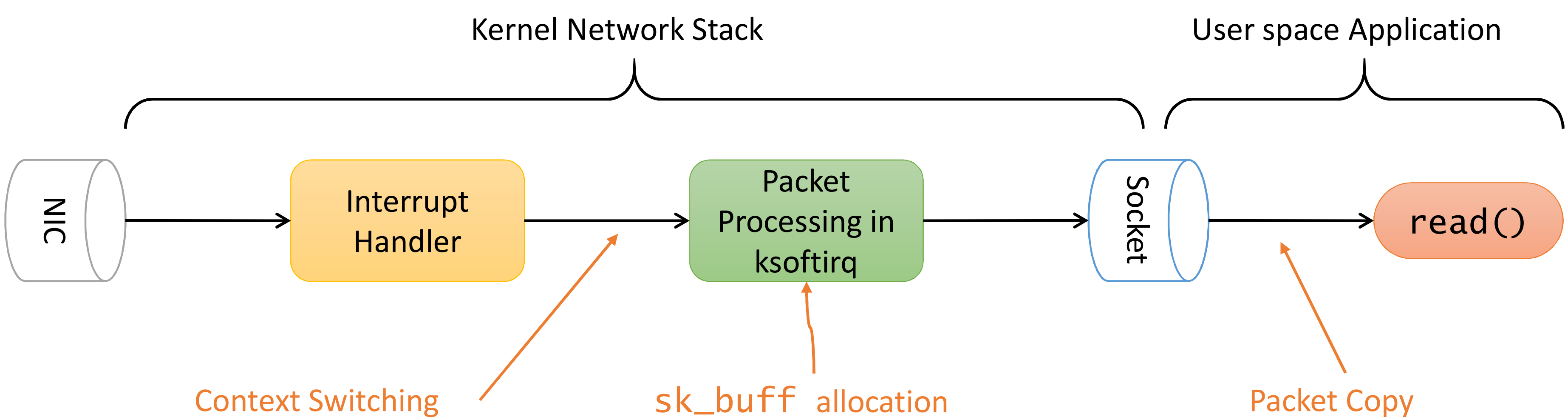
Packet copy to sockets

- Packet headers (port number) used to map received packet to socket
- On read from application, packet payload copied from kernel memory to user memory



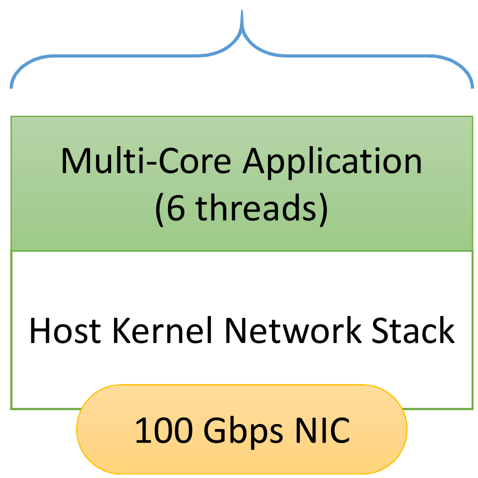
Overheads of the Linux network stack

- Interrupt handling, transition across user and kernel mode
- Context switching from application to ksoftirq
- Packet copy from kernel to user space



Need for alternate fast network I/O tchniques

Max throughput possible is ~15 Gbps [1]



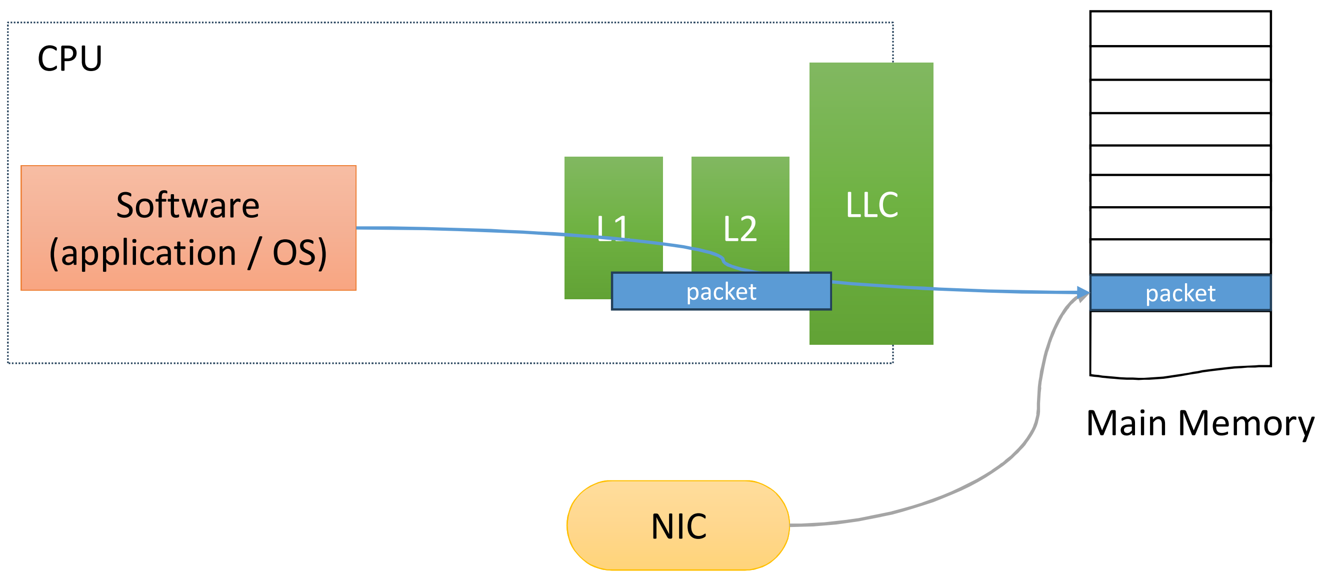
Throughput << NIC Line Rate

- Multi-threaded applications cannot easily achieve line rate in modern high-speed NICs, especially with small-sized packets
- Techniques to improve processing speed include kernel bypass techniques (directly DMA packets into user space) and using polling-mode device drivers
- Possible to process 100s of Gbps easily in software using such techniques

[1] T. Høiland-Jørgensen et al., "The eXpress data path: fast programmable packet processing in the operating system kernel"

Another problem: memory access bottleneck

- Memory wall: DRAM speeds have not increased as much as CPU or network hardware
- On high speed network links, only few nanoseconds budget per packet, but accessing main memory takes hundreds of nanosec



Direct Cache Access / DDIO

- Direct Cache Access (Intel's DDIO): NIC writes packet directly into CPU caches, and does not DMA into main memory
- User/kernel software can access packet quickly from cache
- Leads to much faster network packet processing