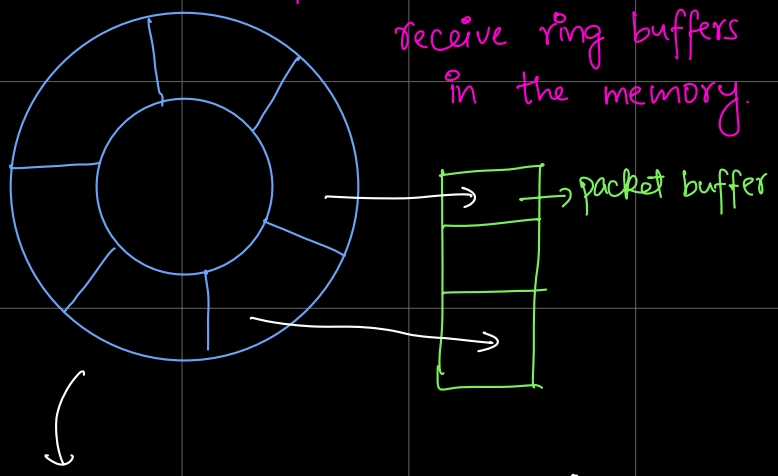
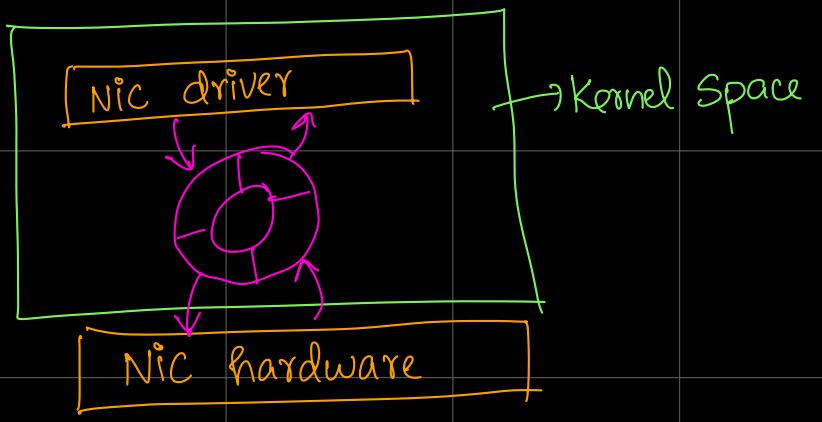
Packet arrive at the NIC. It is stored in the Hardware Receive Queue.

→ Here match the destination MAC address.
→ And verify the frame checksum (FCS)



This circular Queue (ring) is shared between the NIC and the driver.



- NIC copies the packets to the memory location pointed by the ring slot using DMA
  → CPU time is not spent in copying.

- NIC then generates an interuppt to inform that the packet is now available in the memory.

Interrupt processing in linux

- Top half
  - does minimal processing required
- Bottom Half
  - major heavylifting

→ So that we don't keep user processes away for too long.

TOP HALF PROCESSING

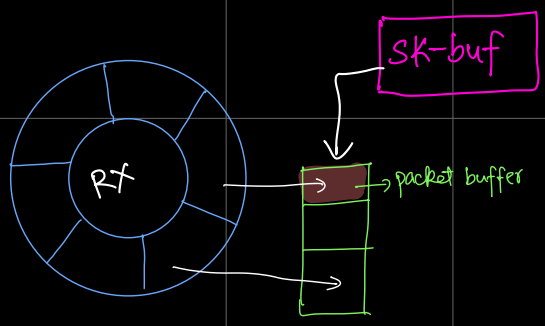
→ Interrupt generated by NIC
Switch from user space to kernel space.
Look IDT and access it using interrupt vector index.
→ Call the corresponding handler.
↓
(ISR: Interrupt Service Routine)

This ISR
→ Acknowledges the interrupt and schedules the bottom half processing

→ Switch back to user space.

Bottom Half Processing:

In the kernel space:
NIC driver dynamically allocates an SK-buf object.
→ in memory data structure, which contains pointers to packet headers and payloads.



NIC driver processing:
- Dynamically allocate an SK-buf
- Update SK-buf with packet metadata.
- Remove ethernet header
- Pass SK-buf to network stack

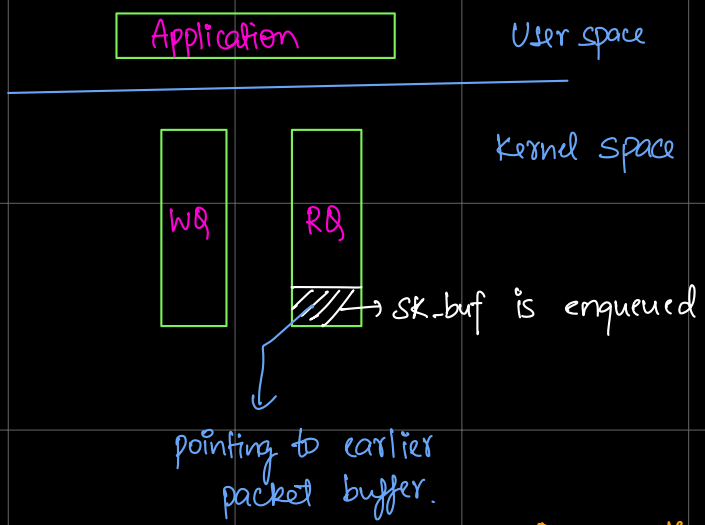
→ Call above layer handler.

for all packets in buffer.

In above layers handlers:

→ Matching of IP / Socket
→ Verify checksum
→ Remove headers

Application binding to a TCP socket
↳ Read / Write socket queues are assigned.



On read() system call: → Dequeue the packet from socket recv queue.
↳ Copy packet to user space buffer.
↳ Release the SK_buf.

Packet processing overheads in the kernel

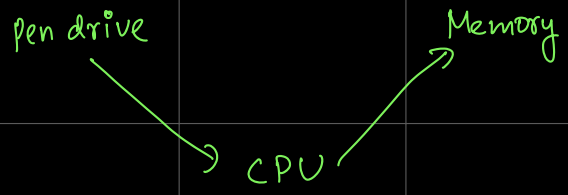
- Too many context switches!!
  - Pollutes CPU cache
- Per-packet interrupt overhead
- Dynamic allocation of sk-buff
- Packet copy between kernel and user space
- Shared data structures

Cannot achieve line-rate for recent high speed NICs!! (40Gbps/100Gbps)

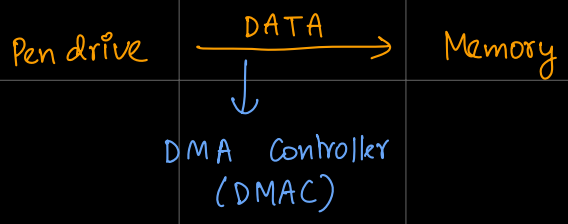
18

Direct Memory Access (DMA)

Naively, if we want to copy from pen-drive to memory.



DMA



We have a disk controller, which is used to communicate to the disk, external storage.

→ So basically CPU will ask DMAC to transfer x bytes of data from pen-drive to memory.

→ DMAC will now ask disk controller to read data from disk into its local buffer & then copy this data into RAM.
↳ disk controller's

→ When the complete data (x bytes) are transferred, DMAC will send an interrupt to the CPU.