Understanding DPDK

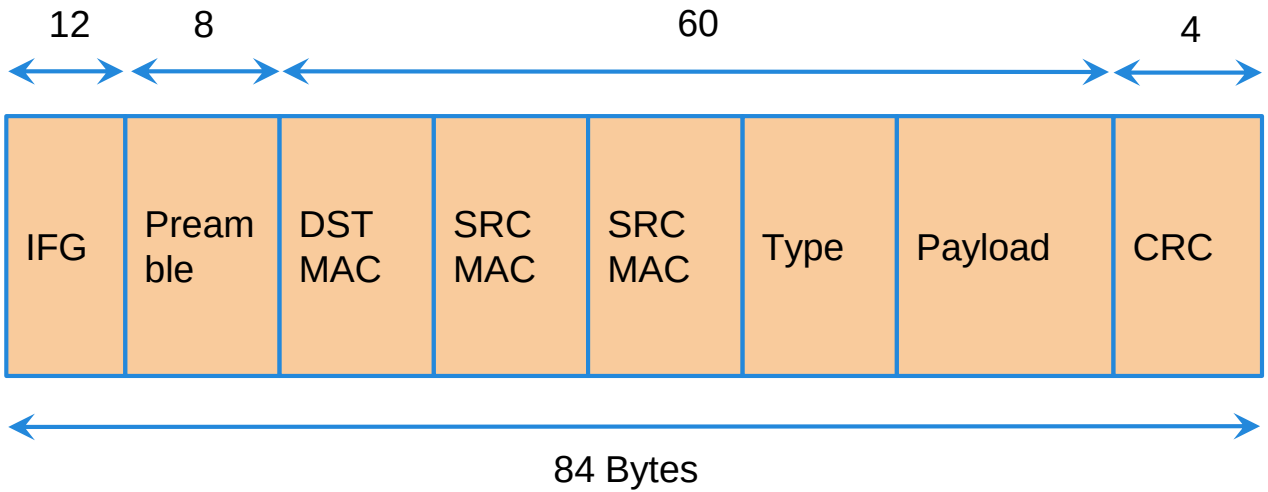
Description of techniques used to achieve high throughput on a commodity hardware

How fast SW has to work?

14.88 millions of 64 byte packets per second on 10G interface

1.8 GHz -> 1 cycle = 0,55 ns

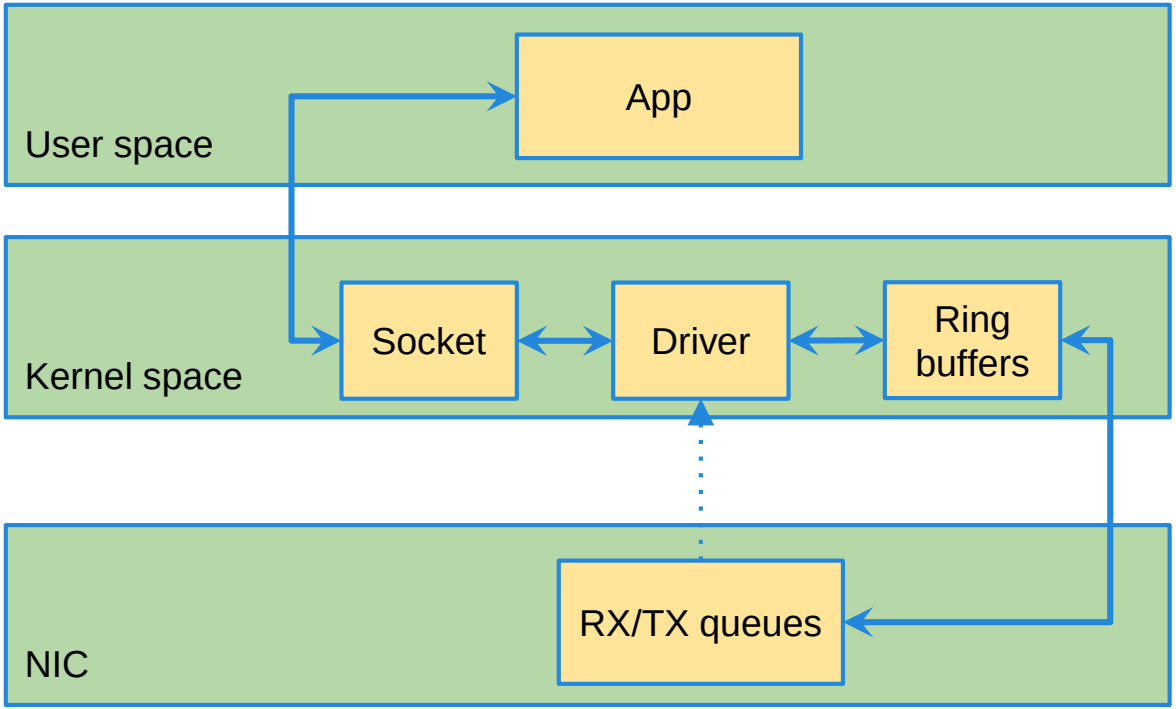
1 packet -> 67.2 ns = 120 clock cycles



Comparative speed values

CPU to memory speed = 6-8 GBytes/s
PCI-Express x16 speed = 5 GBytes/s
Access to RAM = 200 ns
Access to L3 cache = 4 ns
Context switch ~= 1000 ns (3.2 GHz)

Packet processing in Linux



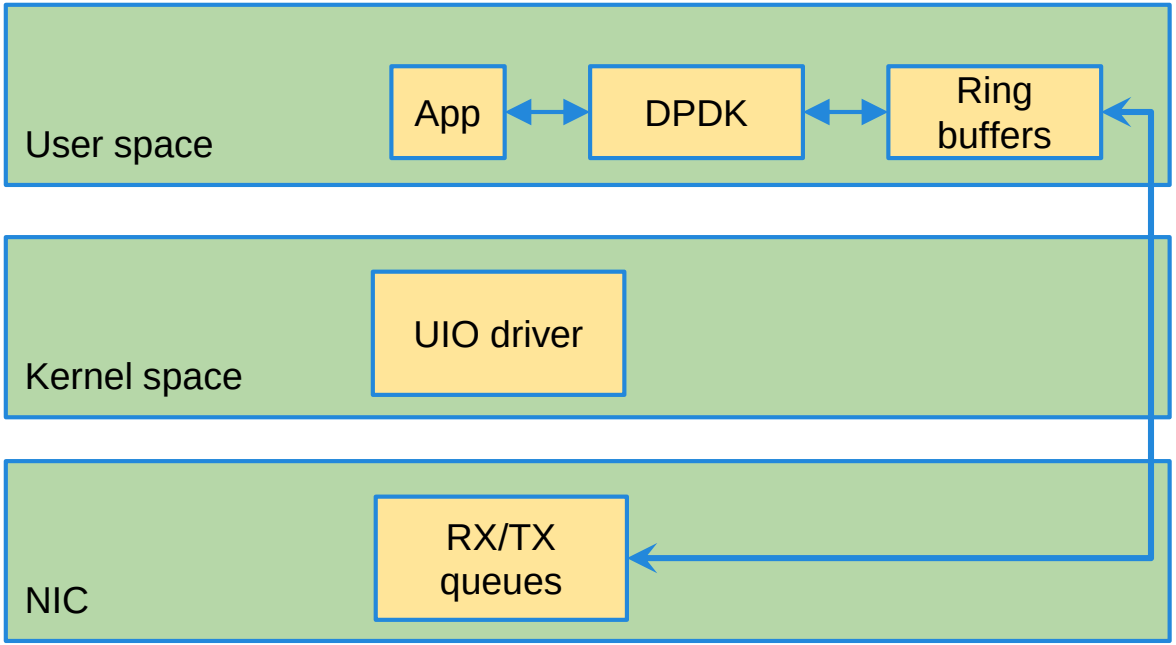
Linux kernel overhead

System calls
Context switching on blocking I/O
Data copying from kernel to user space
Interrupt handling in kernel

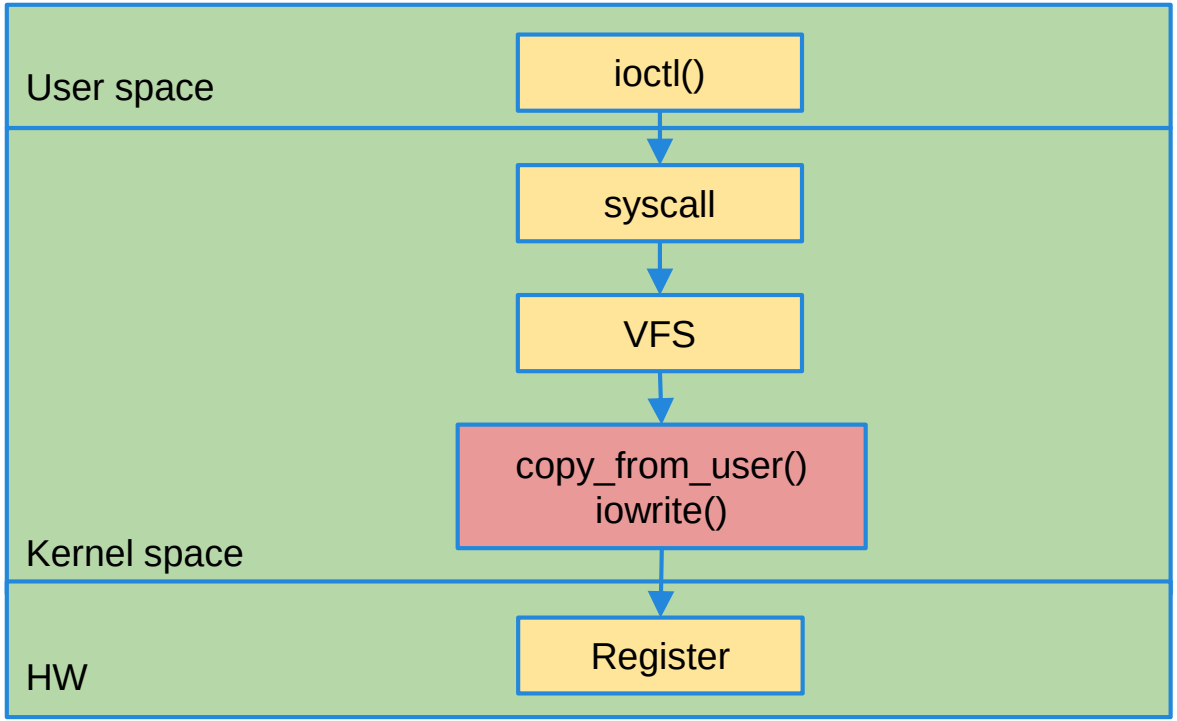
Expense of sendto

Function	Activity	Time (ns)
sendto	system call	96
sosend_dgram	lock sock_buff, alloc mbuf, copy in	137
udp_output	UDP header setup	57
ip_output	route lookup, ip header setup	198
ether_otput	MAC lookup, MAC header setup	162
ixgbe_xmit	device programming	220
Total		**950**

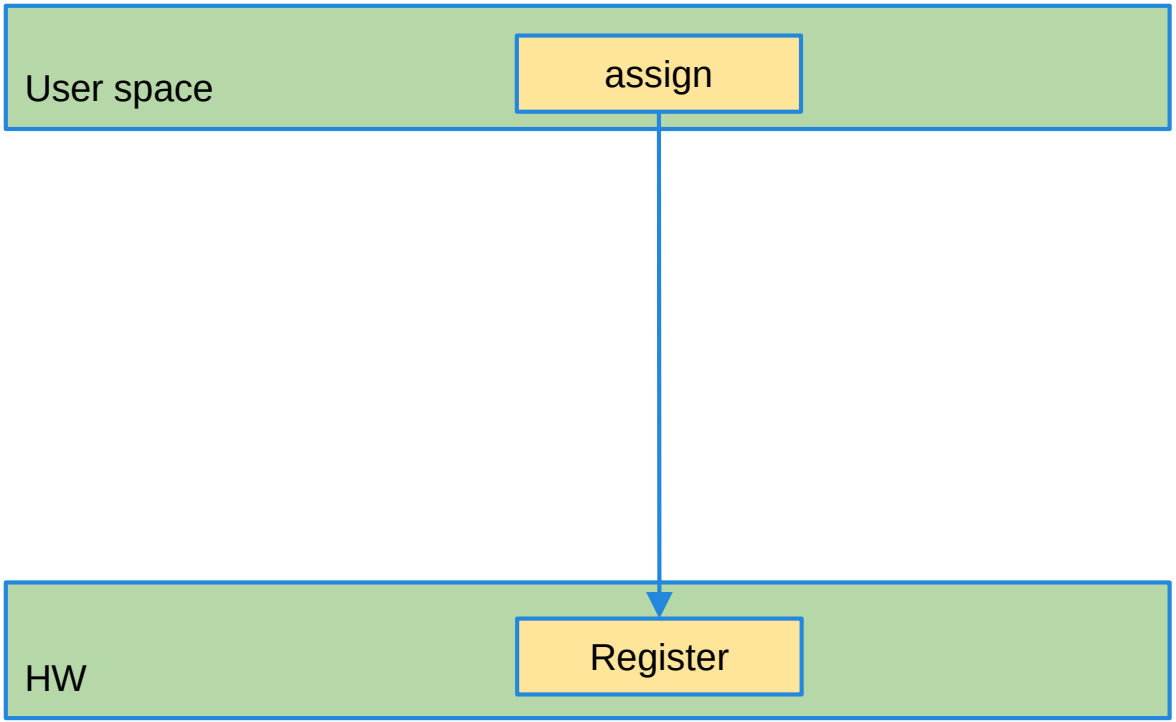
Packet processing with DPDK



Updating a register in Linux



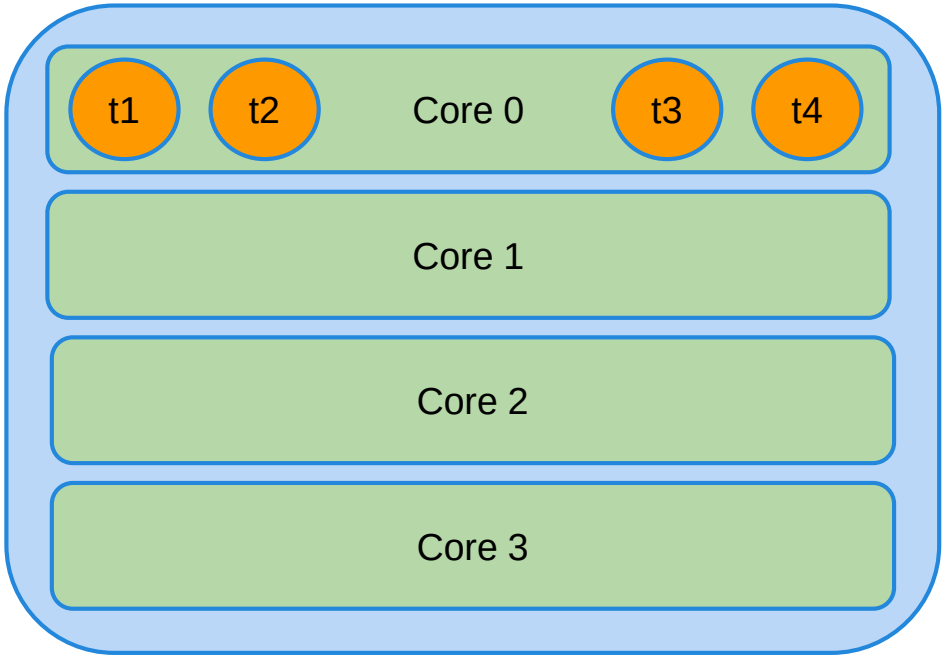
Updating a register with DPDK



What is used inside DPDK?

Processor affinity (separate cores)
Huge pages (no swap, TLB)
UIO (no copying from kernel)
Polling (no interrupts overhead)
Lockless synchronization (avoid waiting)
Batch packets handling
SSE, NUMA awareness

Linux default scheduling



How to isolate a core for a process

To diagnose use **top**

“top” , press “f” , press “j”

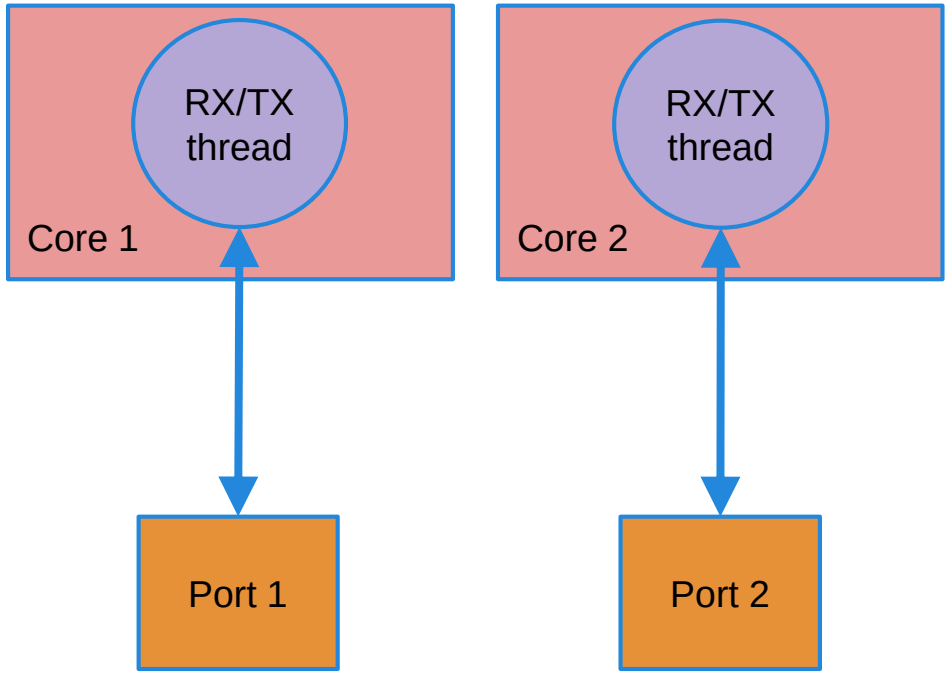
Before boot use **isolcpus**

“isolcpus=2,4,6”

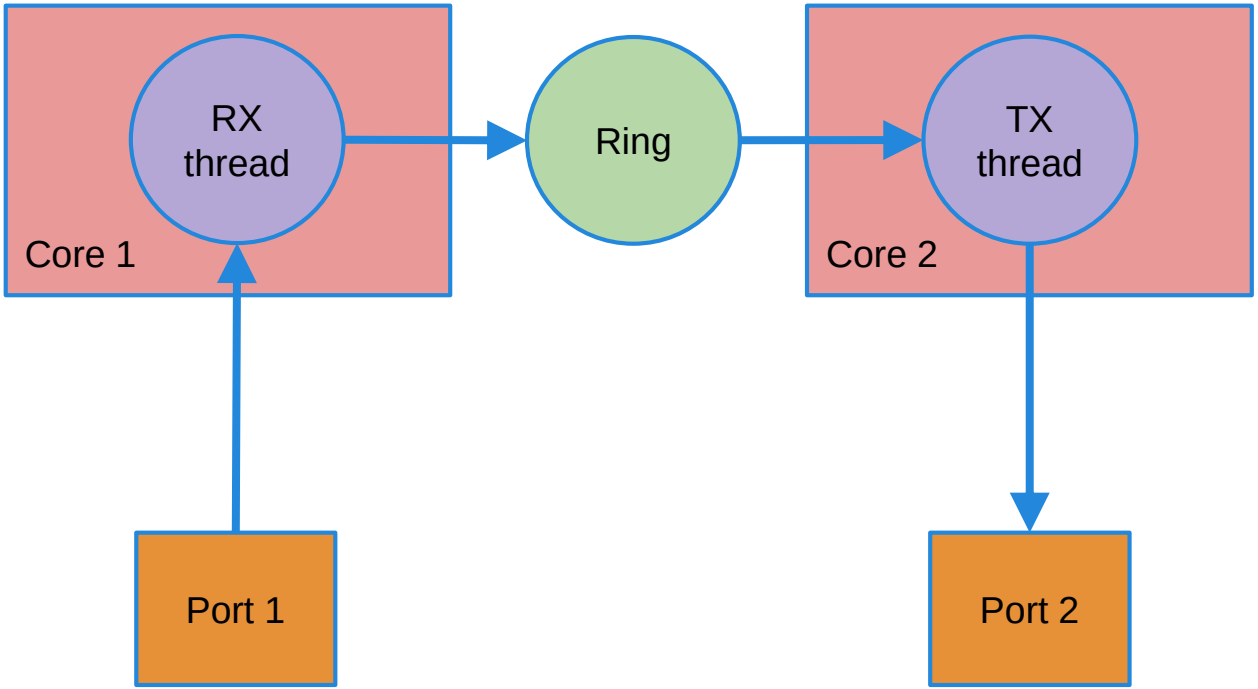
After boot - use **cpuset**

“cset shield -c 1-3”, “cset shield -k on”

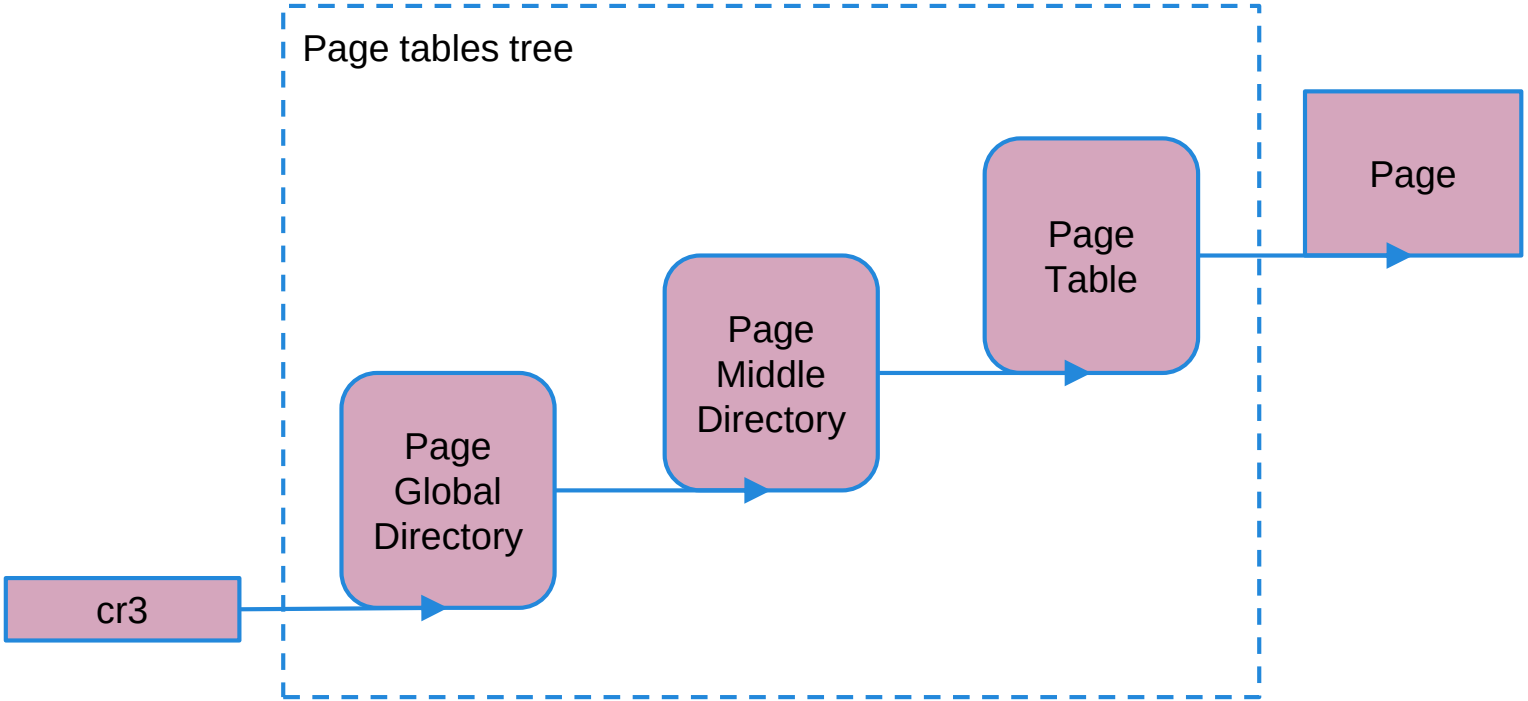
Run-to-completion model



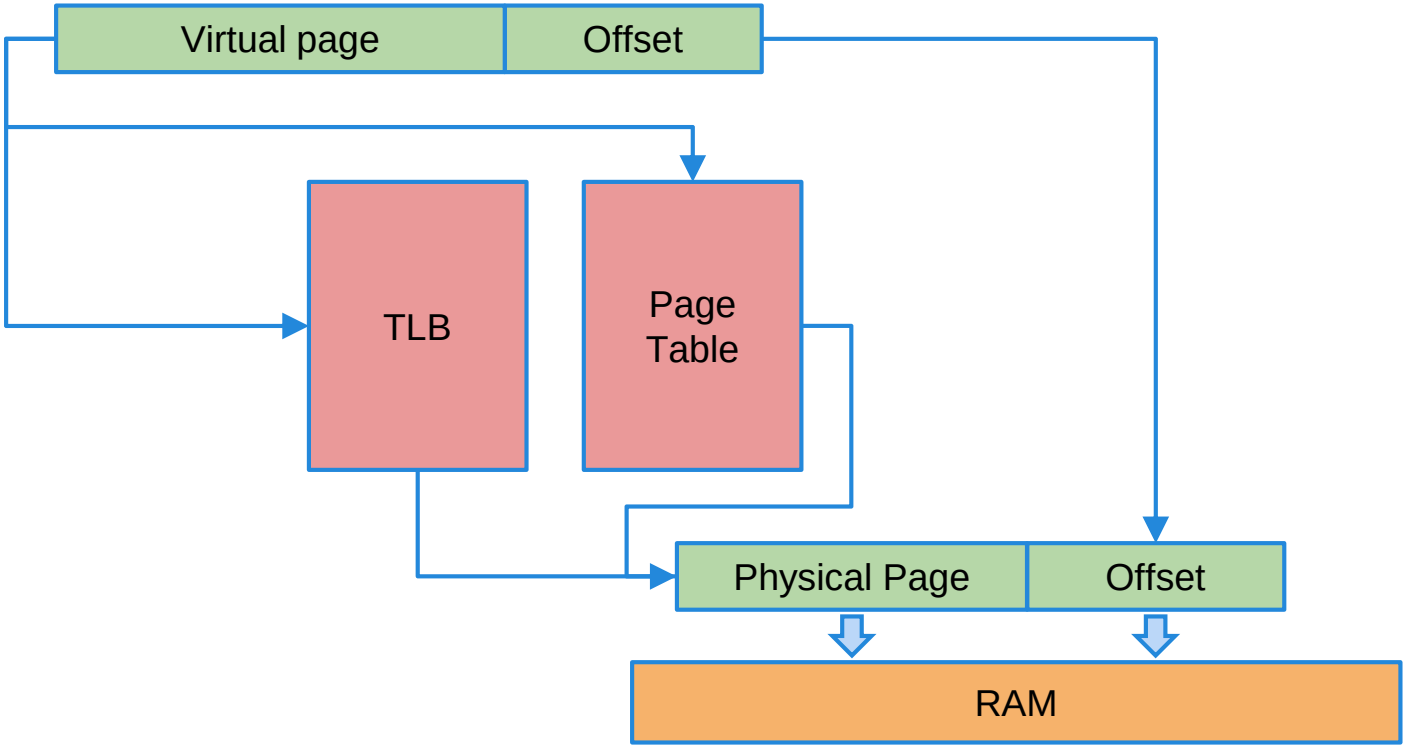
Pipeline model



Linux paging model



TLB


Virtual page
Offset
TLB
Page Table
Physical Page
Offset
RAM

TLB characteristics

$ cpuid | grep -i tlb

size: 12–4,096 entries

hit time: 0.5–1 clock cycle

miss penalty: 10–100 clock cycles

miss rate: 0.01–1%

It is very expensive resource!

Solution - Hugepages

Benefit: optimized TLB usage, no swap

Hugepage size = 2M

Usage:

mount hugetlbfs /mnt/huge

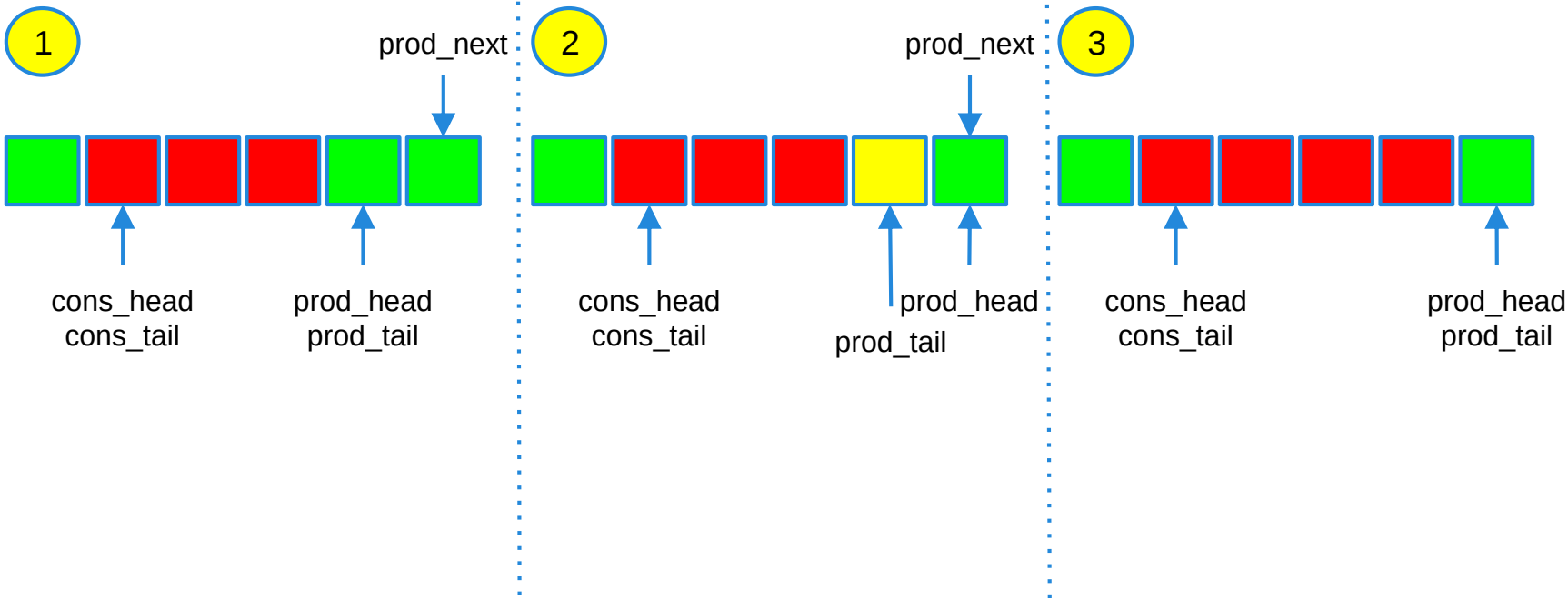
mmap

Library - libhugetlbfs

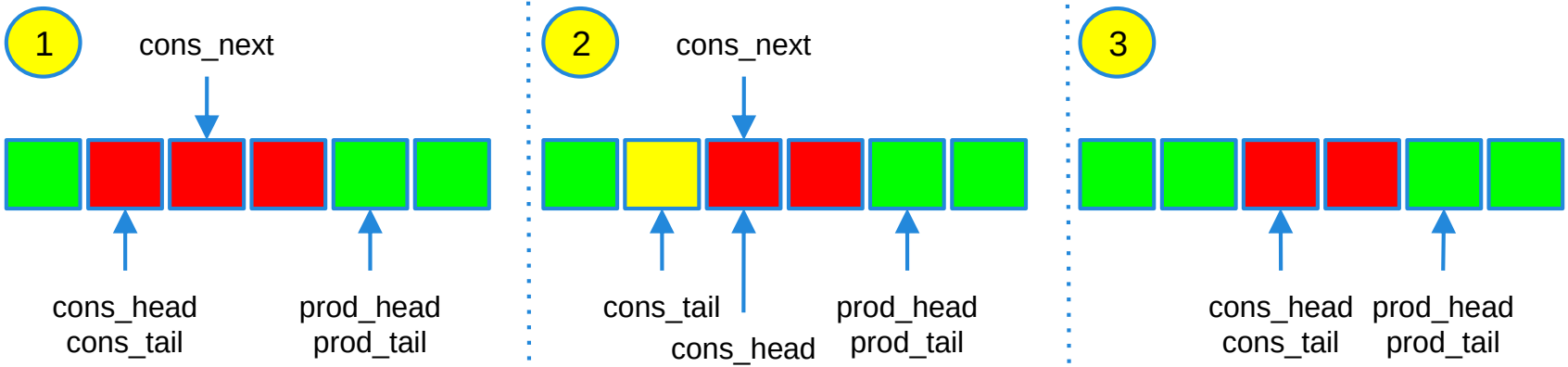
Lockless ring design

Writer can preempt writer and reader
Reader can not preempt writer
Reader and writer can work simultaneously on different cores
Barrier
CAS operation
Bulk queue/dequeue

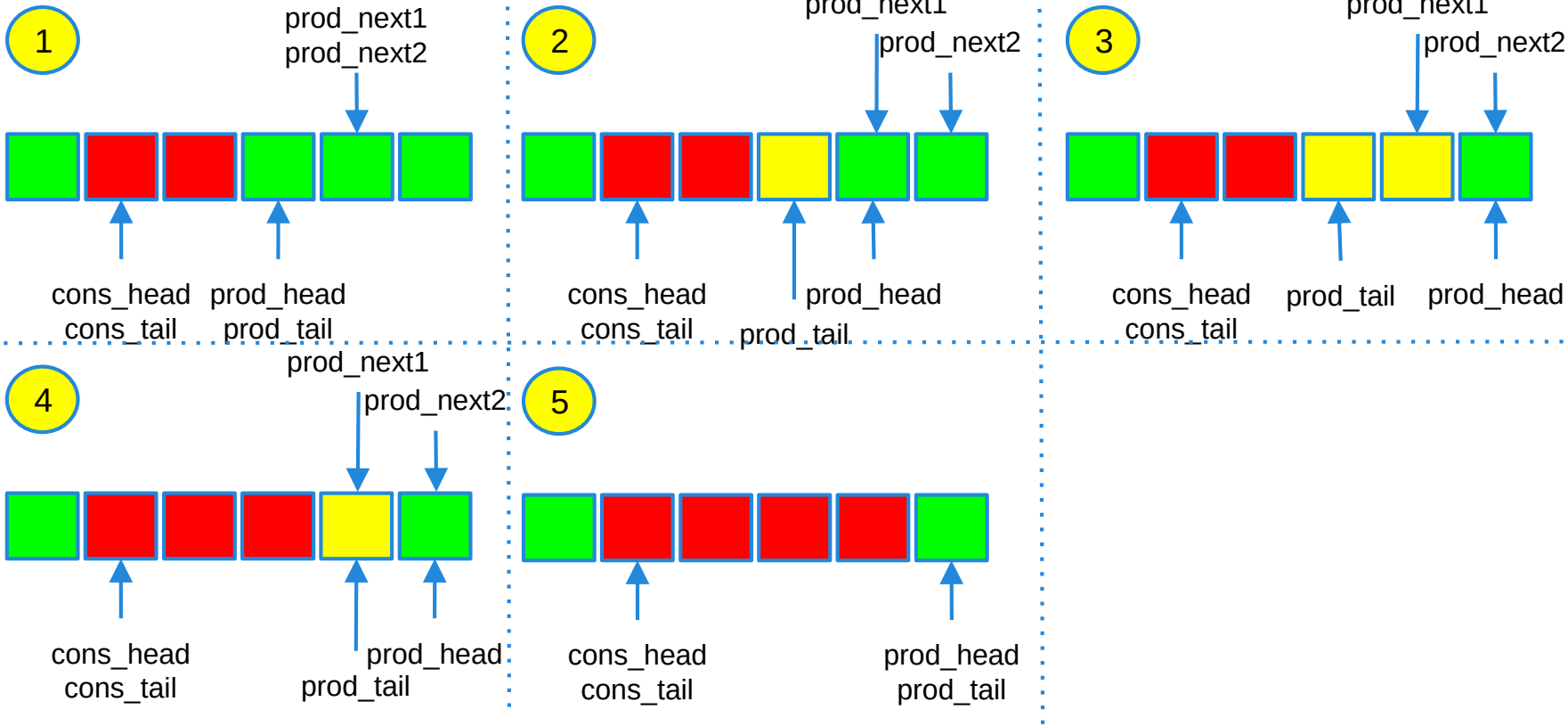
Lockless ring (Single Producer)



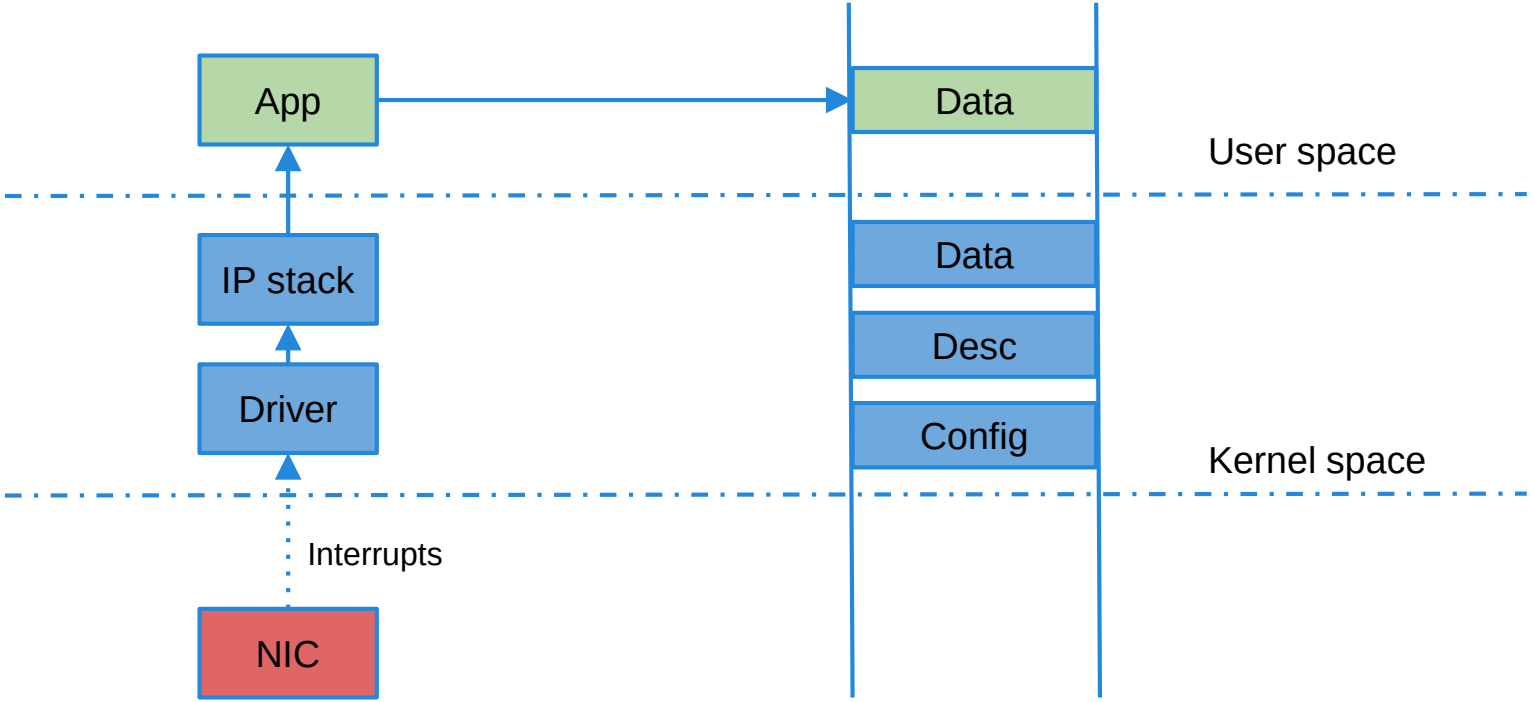
Lockless ring (Single Consumer)



Lockless ring (Multiple Producers)



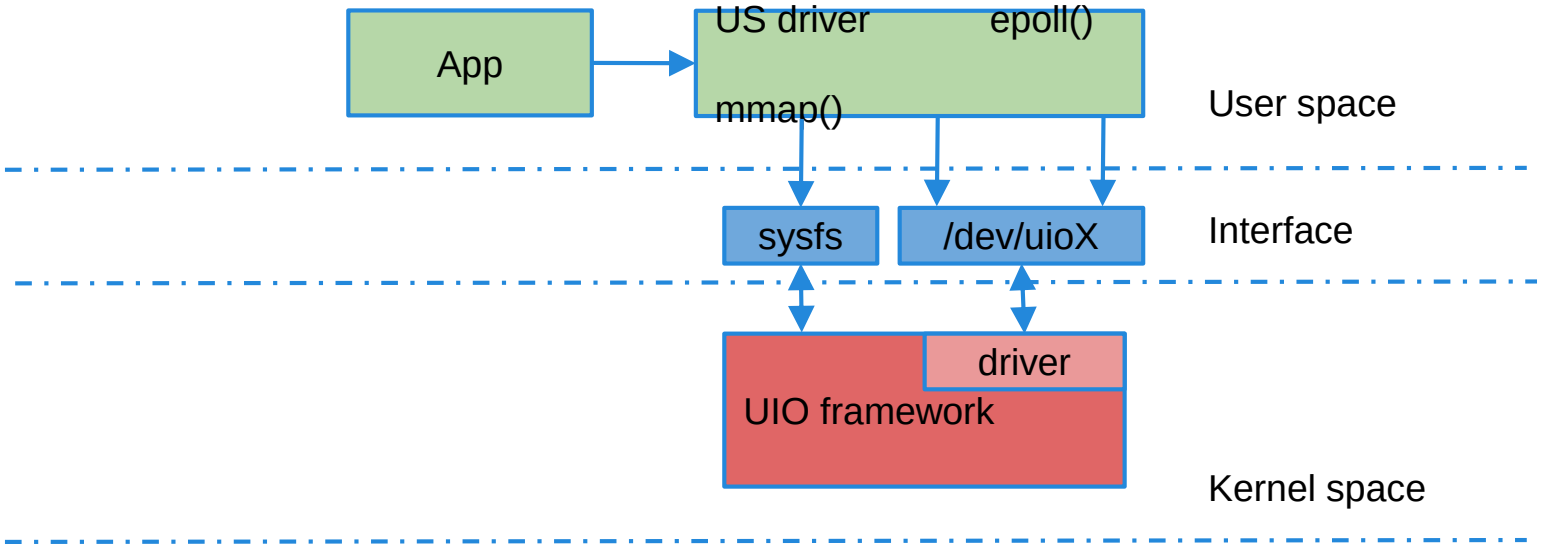
Kernel space network driver



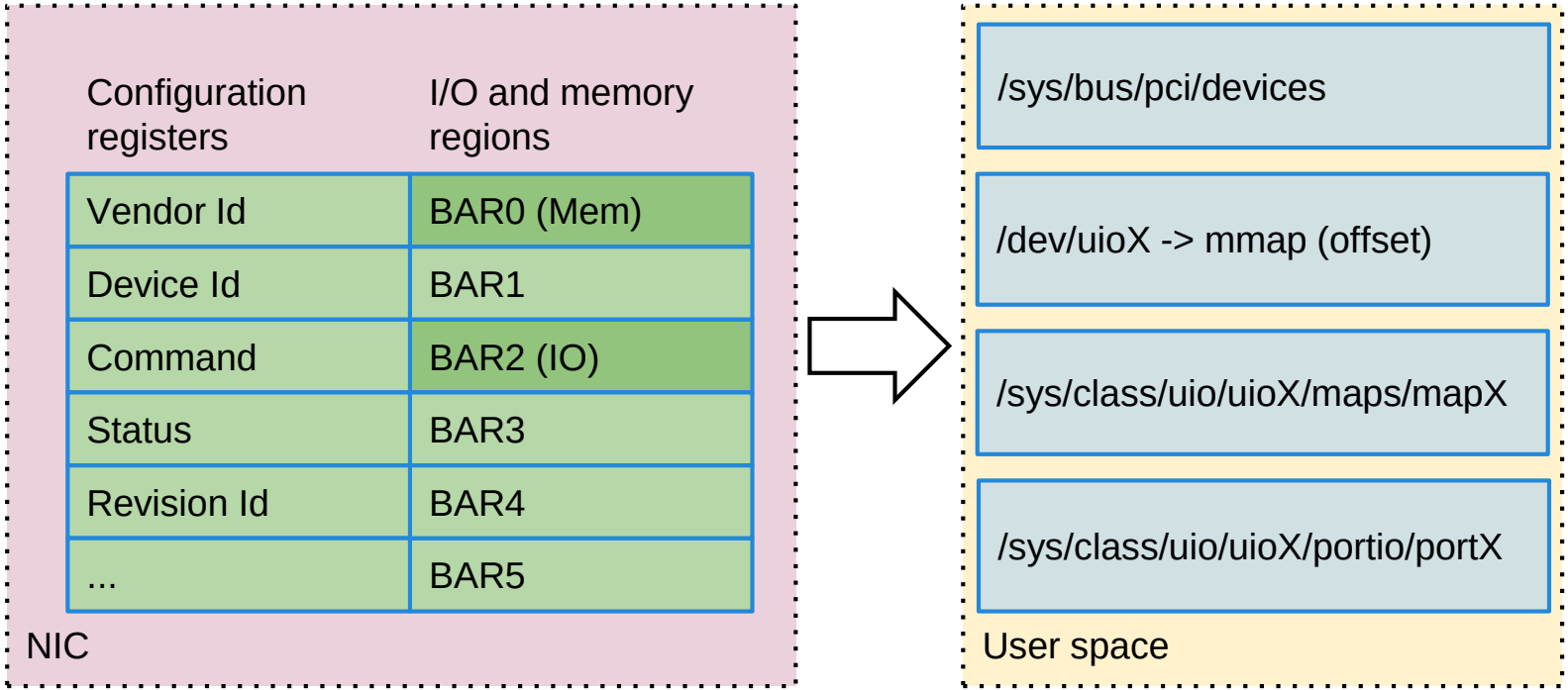
UIO

“The most important devices can’t be handled in user space, including, but not limited to, network interfaces and block devices.” - LDD3

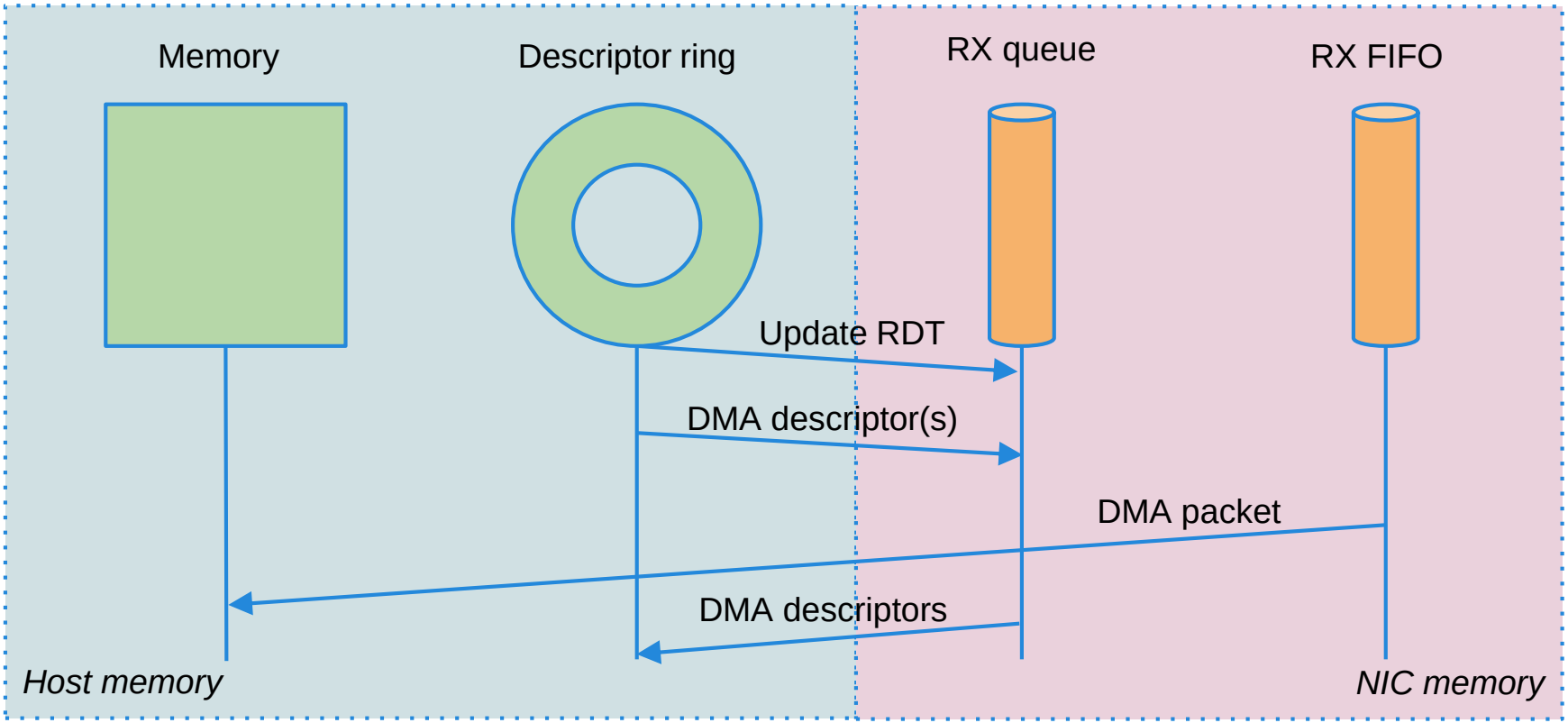
UIO


App
US driver
epoll()
mmap()
User space
sysfs
/dev/uioX
Interface
driver
UIO framework
Kernel space

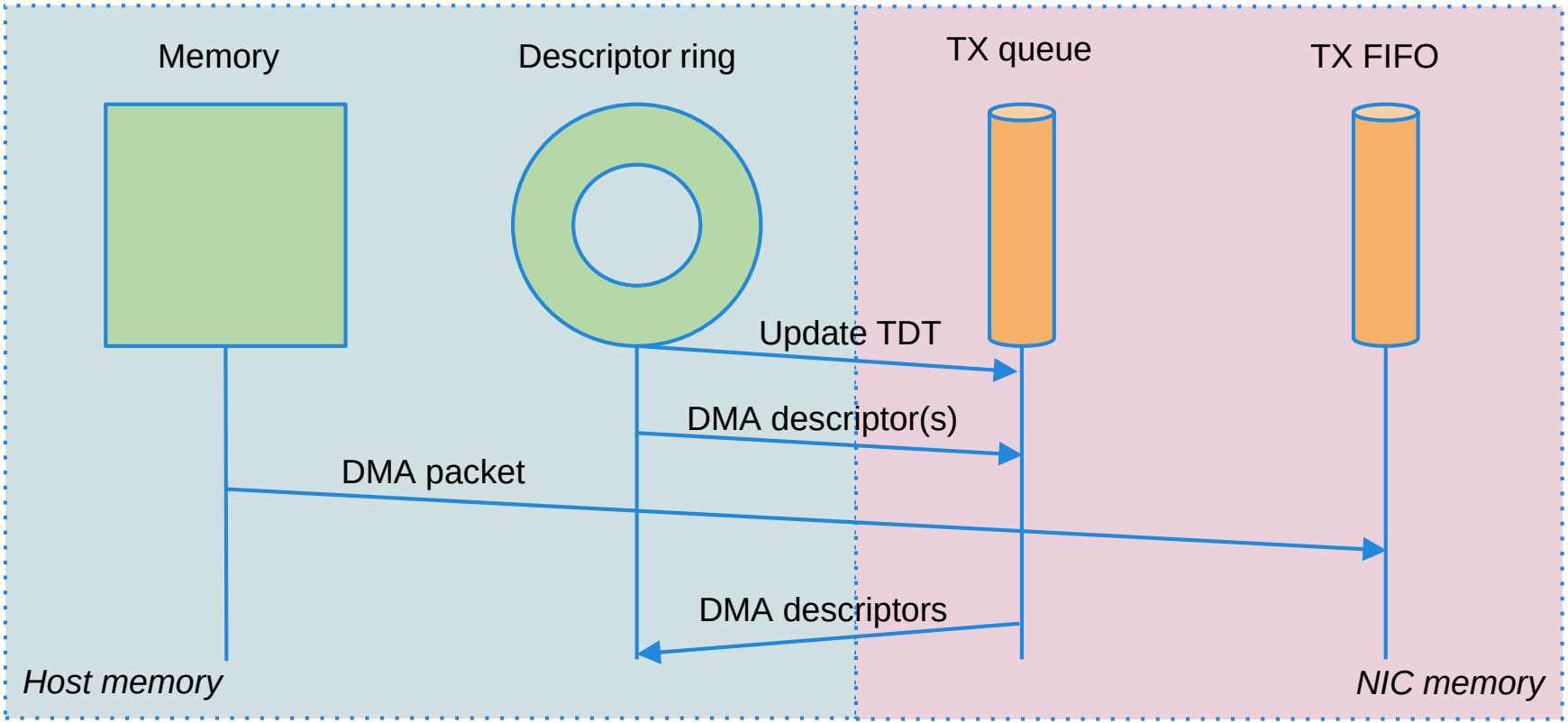
Access to device from user space



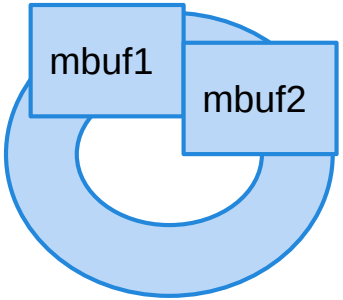
DMA RX

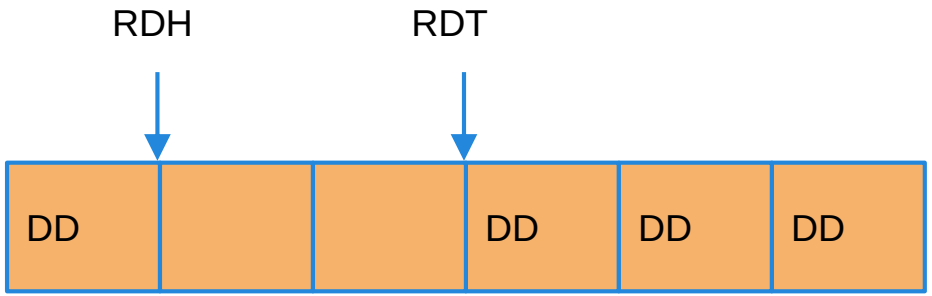

Memory
Descriptor ring
RX queue
RX FIFO
Update RDT
DMA descriptor(s)
DMA packet
DMA descriptors
Host memory
NIC memory

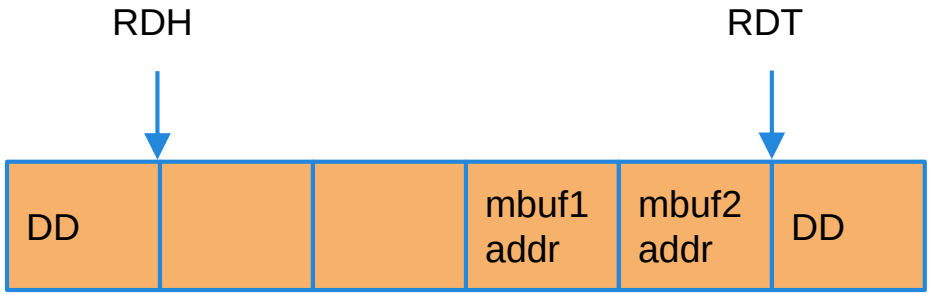
DMA TX


Memory
Descriptor ring
TX queue
TX FIFO
Update TDT
DMA descriptor(s)
DMA packet
DMA descriptors
Host memory
NIC memory

Receive from SW side

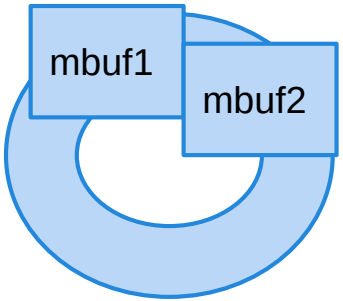


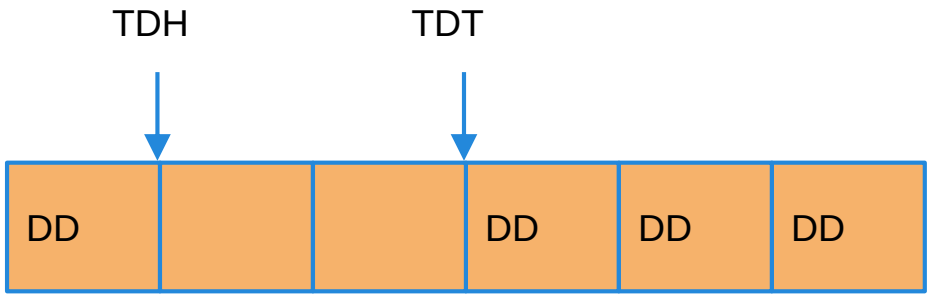


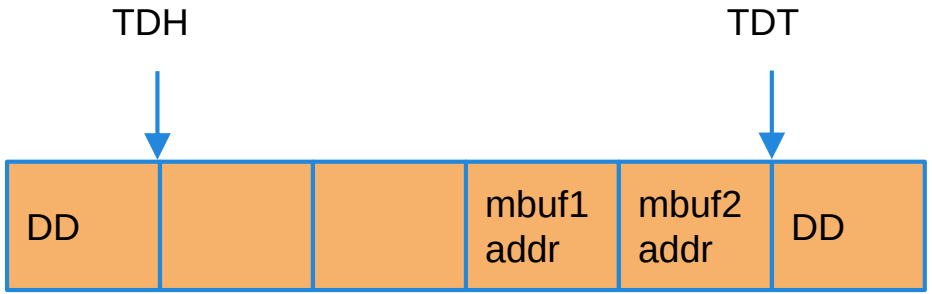


RDH = 1
RDT = 5
RDBA = 0
RDLEN = 6

Transmit from SW side

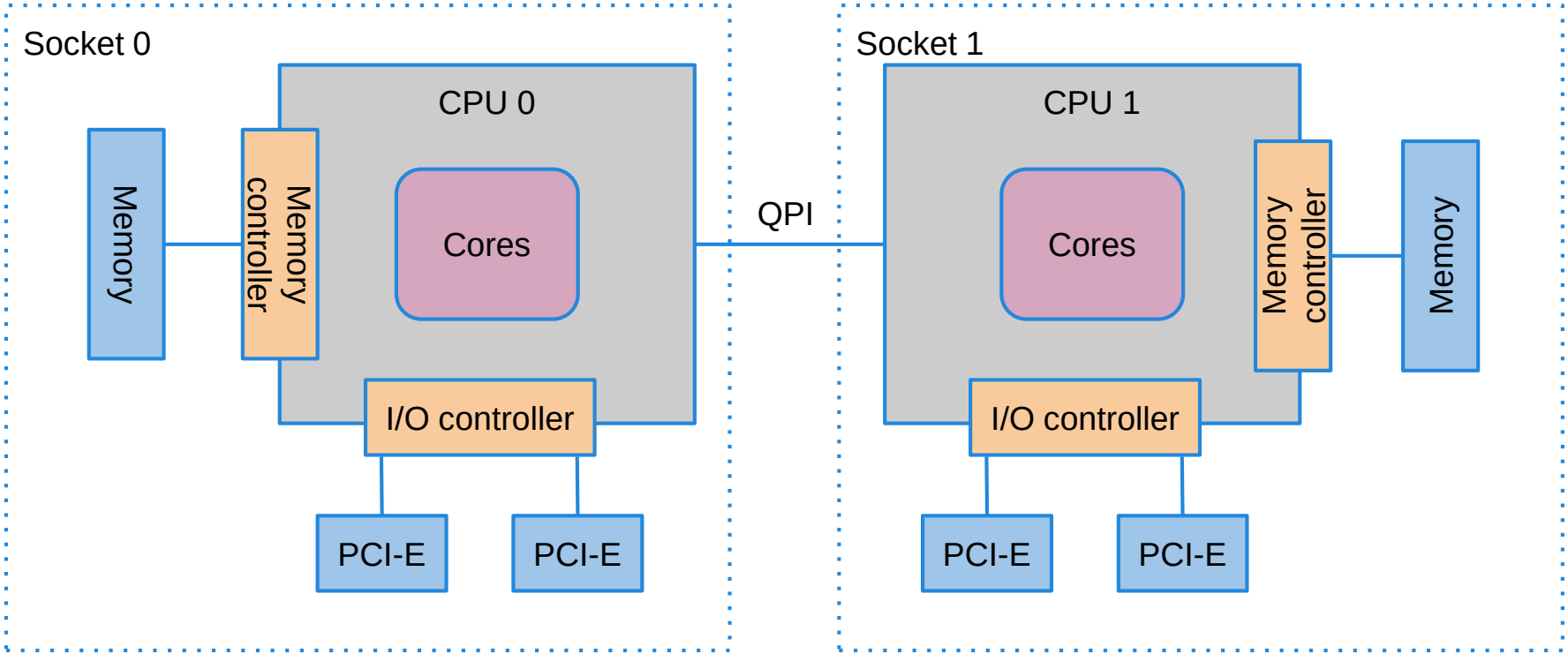




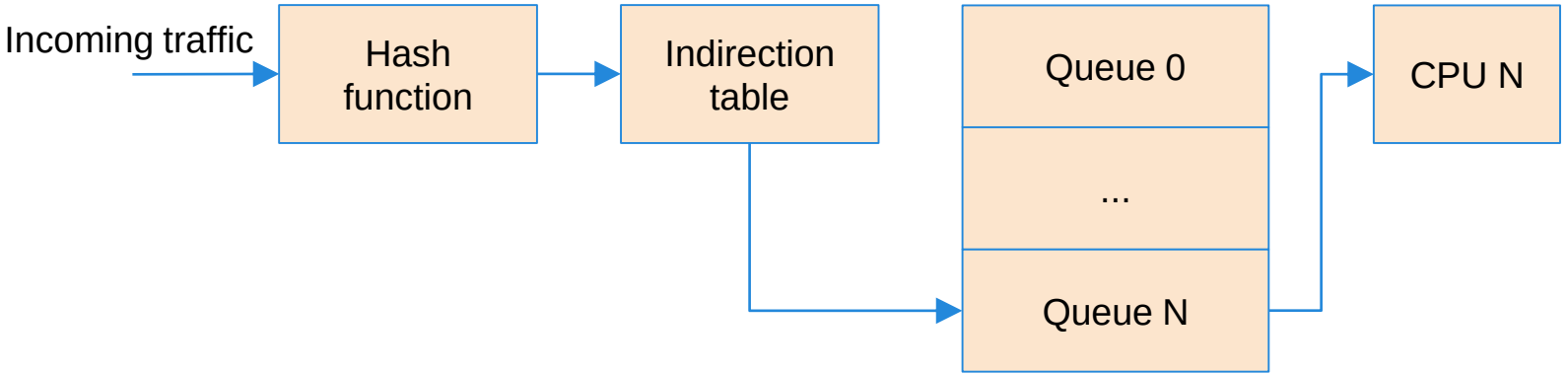


TDH = 1
TDT = 5
TDBA = 0
TDLEN = 6

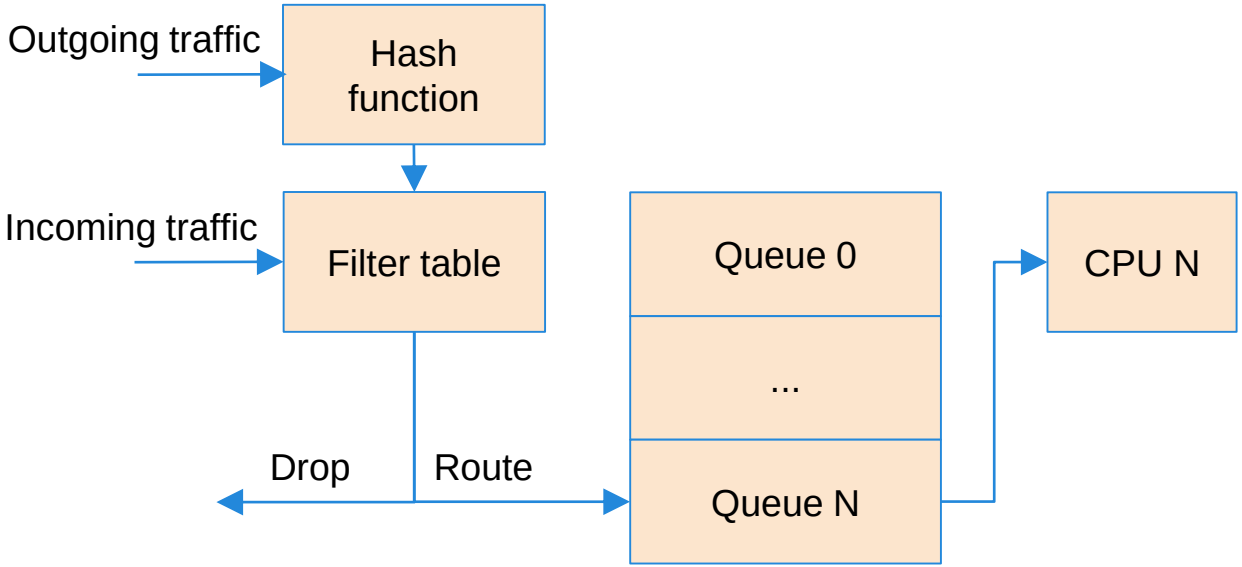
NUMA


Socket 0
CPU 0
Memory
Memory controller
Cores
I/O controller
PCI-E
PCI-E
QPI
Socket 1
CPU 1
Cores
Memory controller
Memory
I/O controller
PCI-E
PCI-E

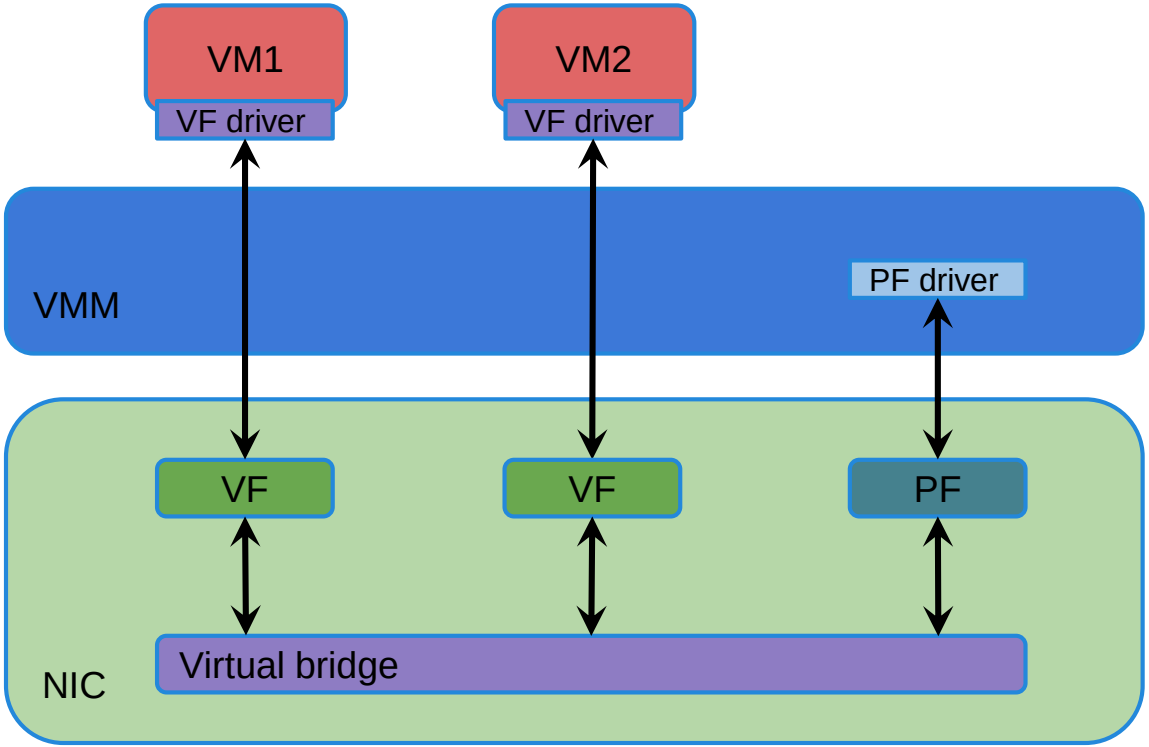
RSS (Receive Side Scaling)



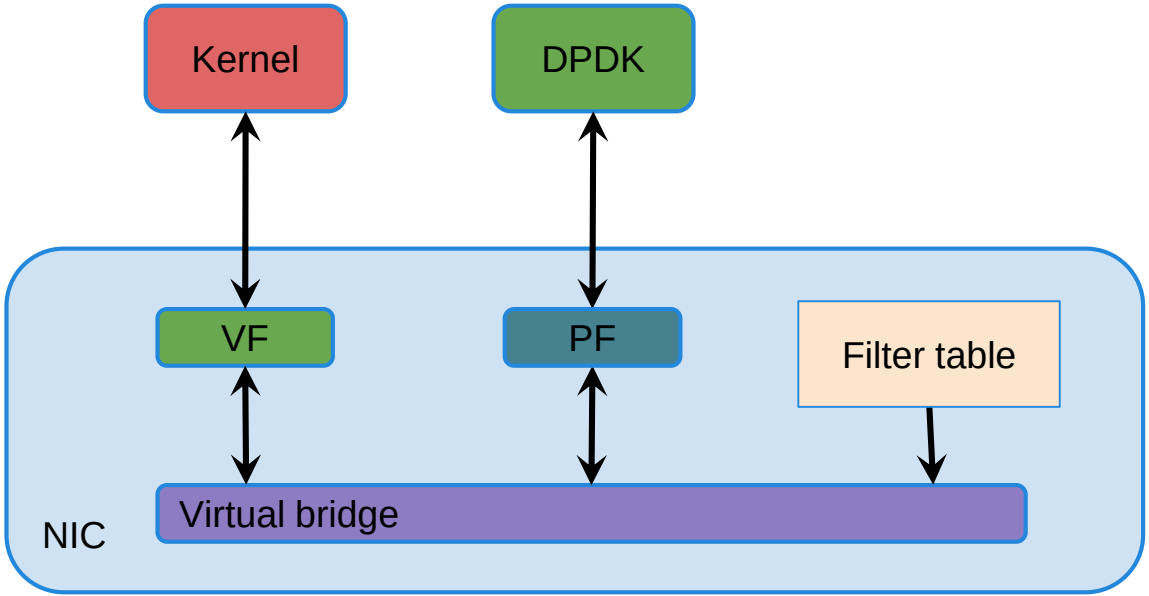
Flow director



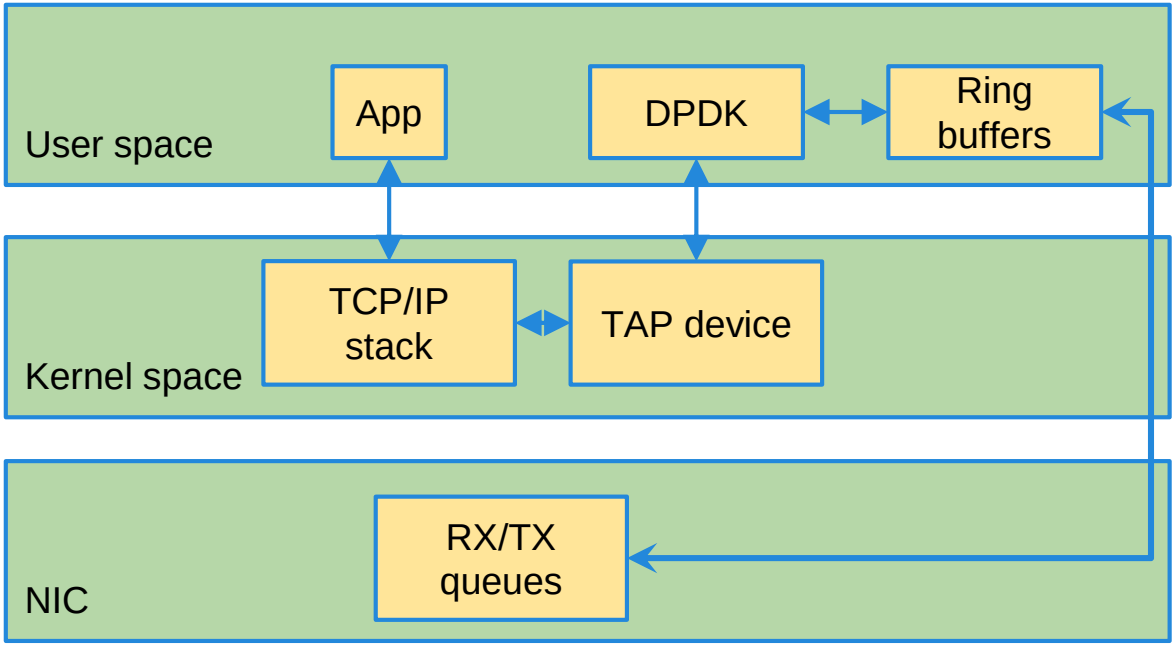
Virtualization - SR-IOV



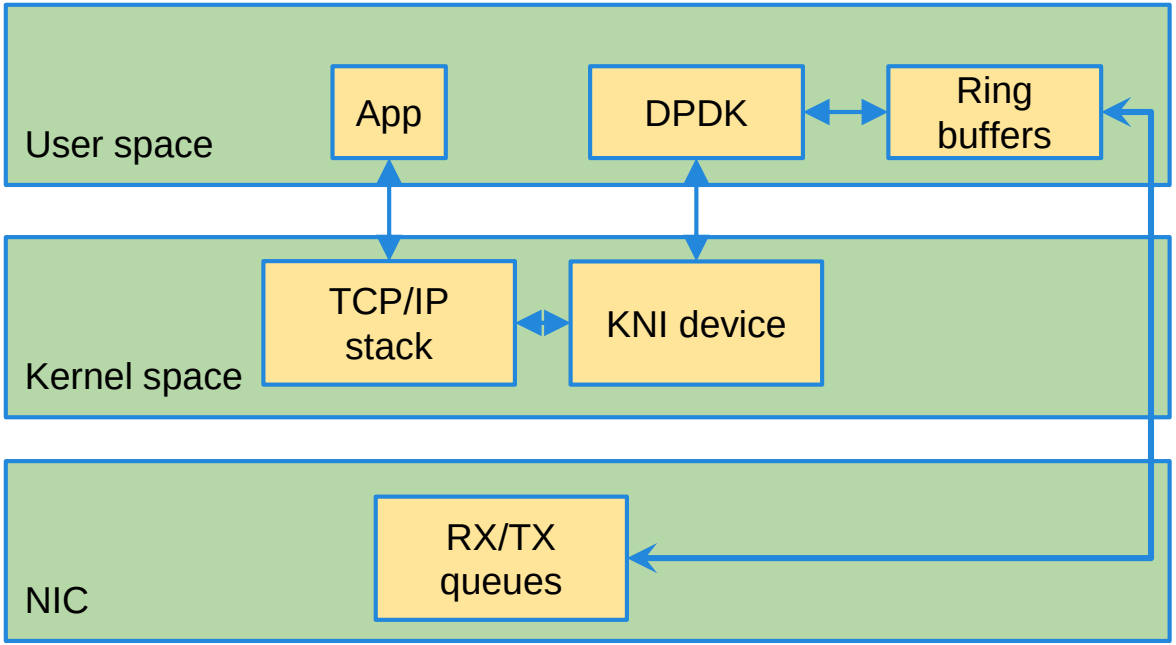
Slow path using bifurcated driver



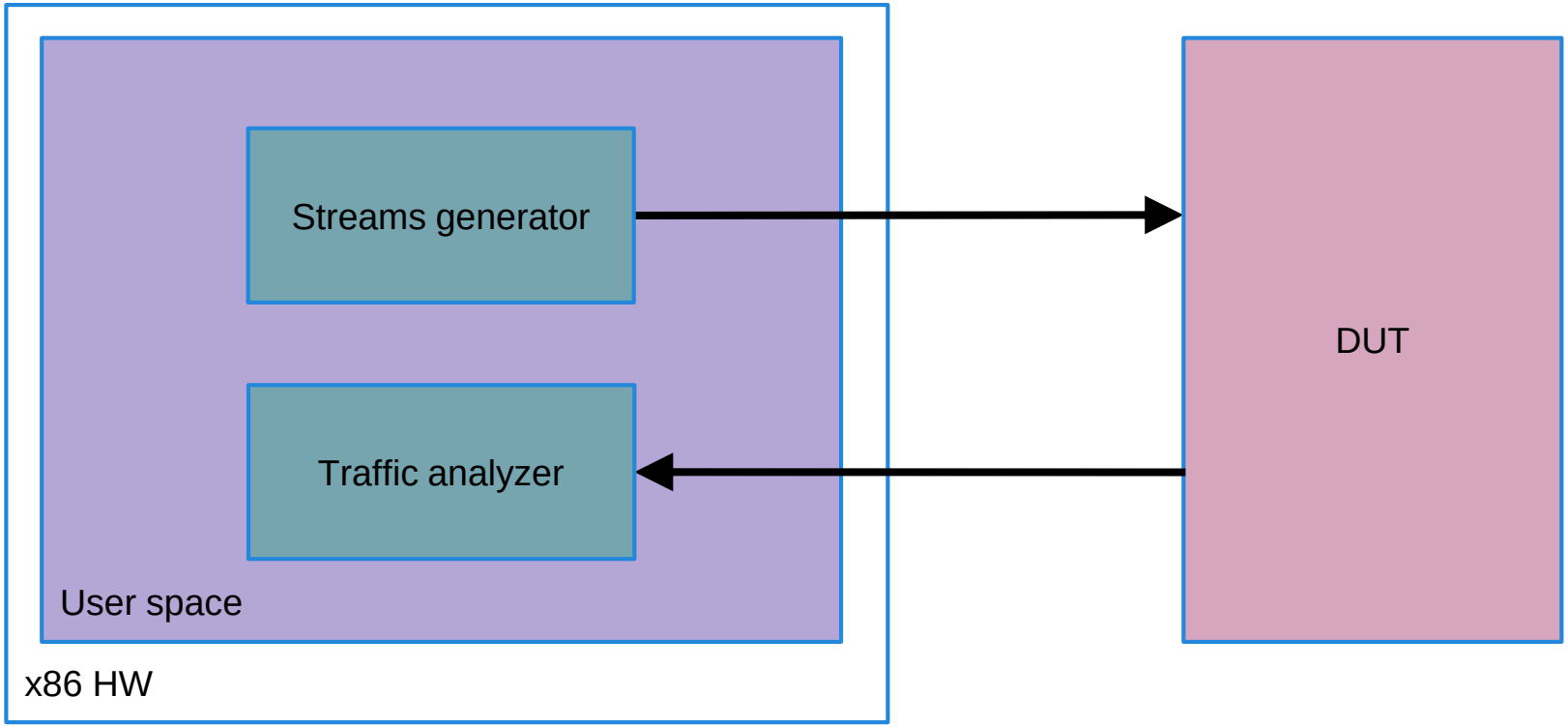
Slow path using TAP



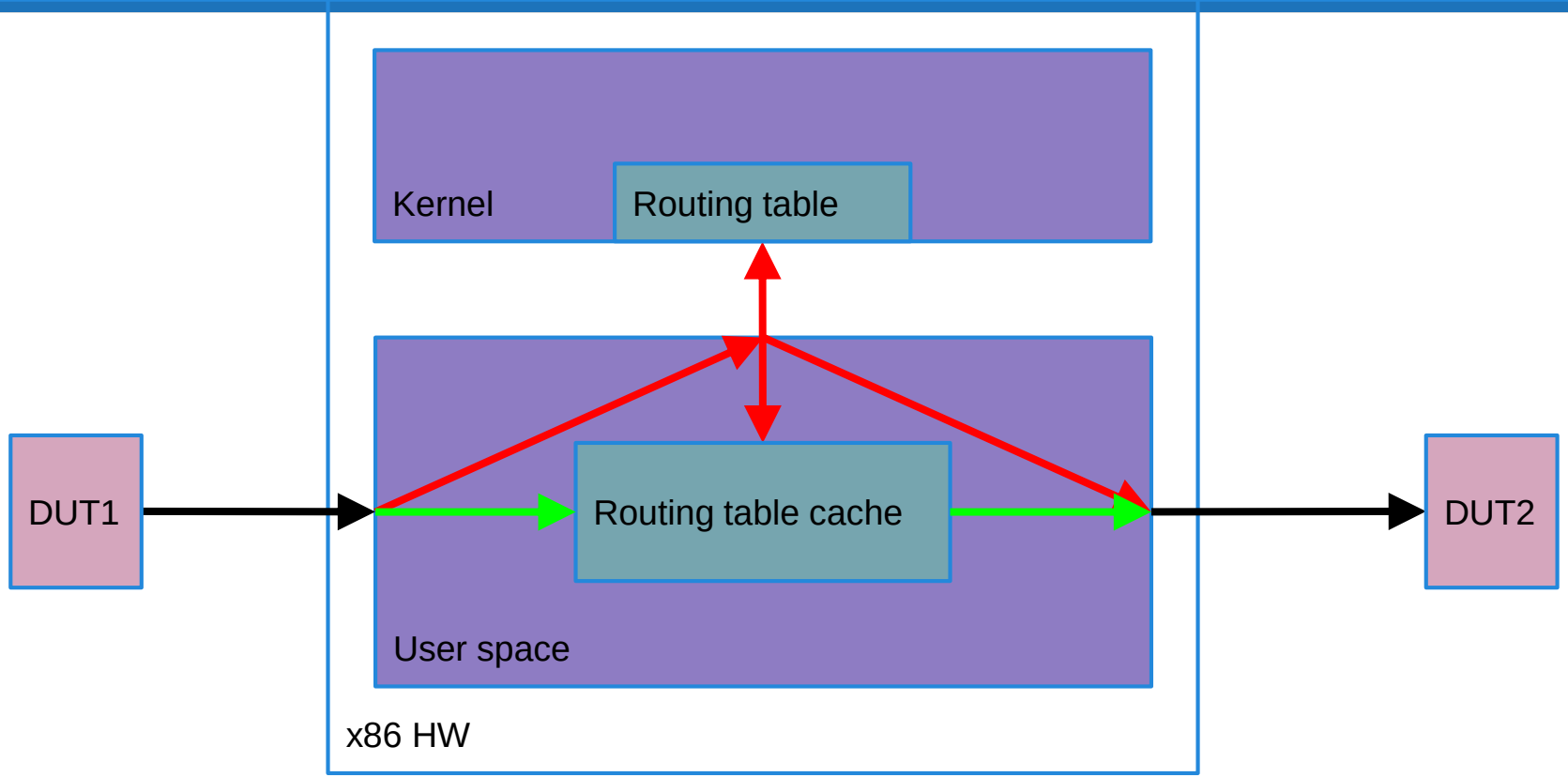
Slow path using KNI



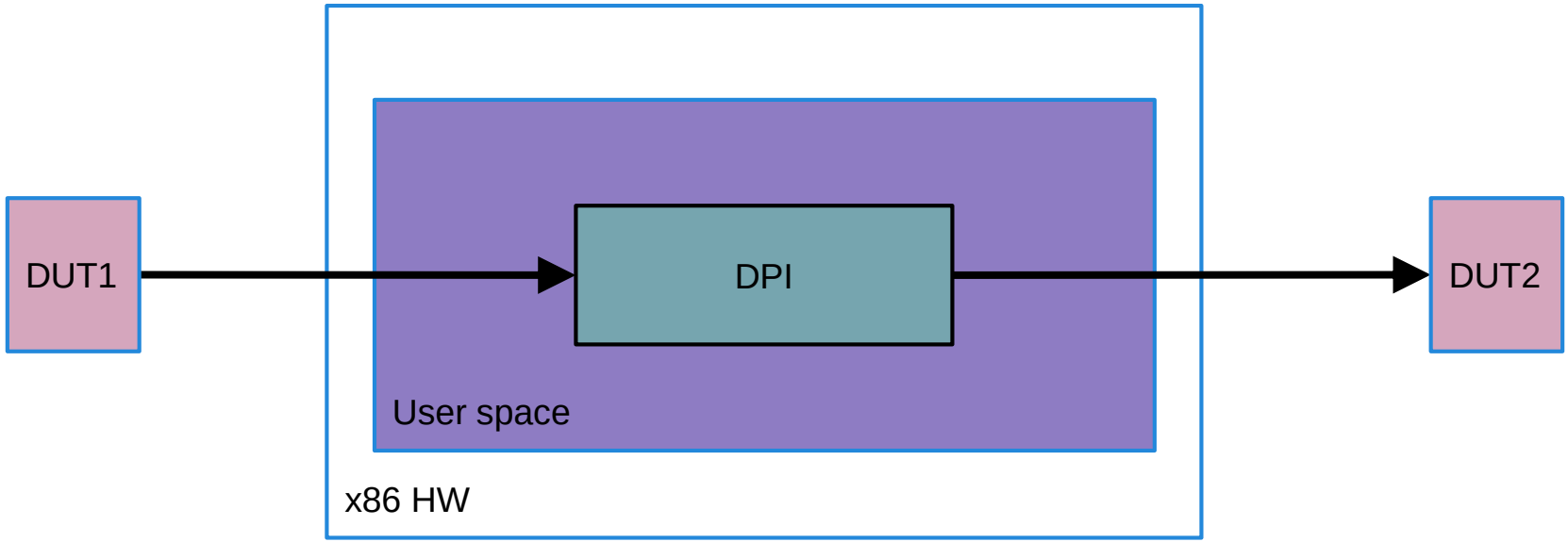
Application 1 - Traffic generator



Application 2 - Router



Application 3 - Middlebox



References

Device Drivers in User Space
Userspace I/O drivers in a realtime context
The Userspace I/O HOWTO
The anatomy of a PCI/PCI Express kernel driver
From Intel® Data Plane Development Kit to Wind River Network Acceleration Platform
DPDK Design Tips (Part 1 - RSS)
Getting the Best of Both Worlds with Queue Splitting (Bifurcated Driver)
Design considerations for efficient network applications with Intel® multi-core processor-based systems on Linux
Introduction to Intel Ethernet Flow Director

My blog

Learning Network Programming