Host Networking

CS 168 – Spring 2024 – Discussion 12

Agenda

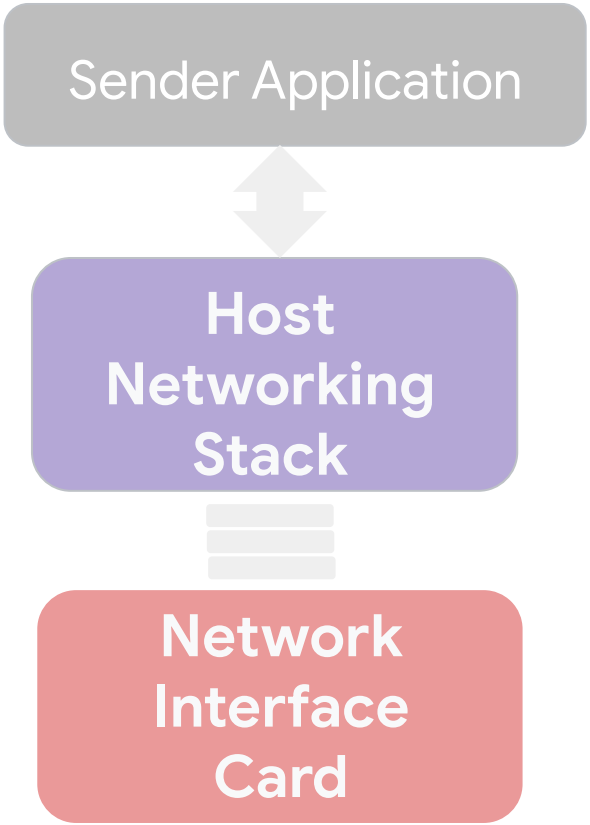
- Problem: datacenter requirements
- Host networking
  - Kernel Bypass
  - RDMA
  - Congestion Control
  - Load balancing
  - Traffic shaping
  - QoS

Problem: Datacenter Requirements

- Datacenter applications have extreme performance requirements
  - Ex. 100ms can cost lots of $$
- Cores are valuable
  - In cloud context, any cycle you waste could have been rented out!
- Kernel development is hard
  - If we are making changes to the networking stack, we might be stuck doing a lot of kernel development
  - Ideally, we could skip this entirely

What is Host Networking?

- Everything at the host that enables it to use the network
  - Linux networking stack, network driver, NIC, etc...



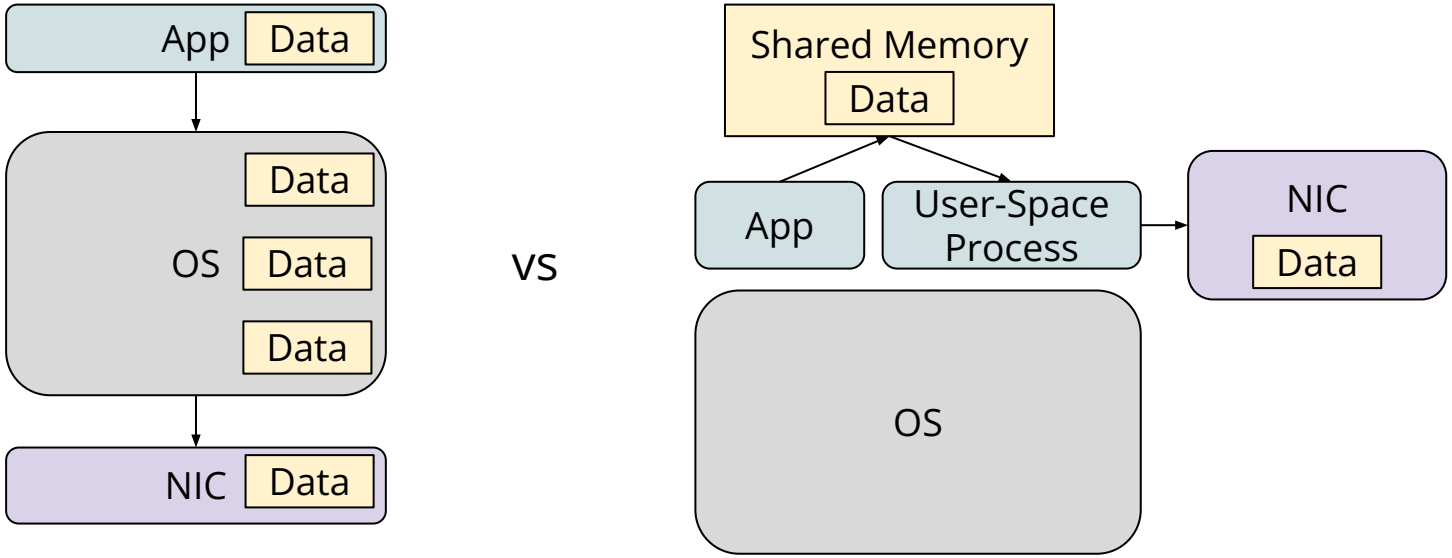
Advanced Host Networking

- Datacenter operators have the incentive to put a lot of effort into optimization
  - And have the means: total administrative control

- Some opportunities for optimization
  - Kernel bypass (avoid kernel development)
  - NIC offloads (save CPU cycles)
  - RDMA (save CPU cycles, and better performance)
  - Other features (better performance)
    - Congestion control
    - Load balancing
    - Traffic shaping
    - QoS

Kernel Bypass

- Run networking stack in user space rather than kernel space

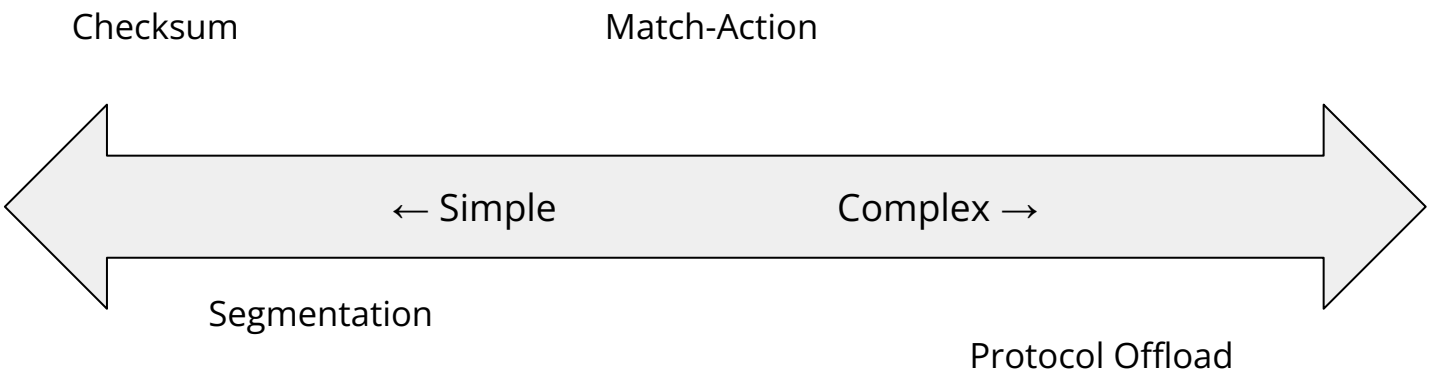


NIC Offloads

- What is offloading?
  - Implementing some tasks in hardware to free up CPU cycles
  - In our case, putting some networking tasks in the NIC instead of the kernel
- Why offload?
  - Save CPU cycles
  - Performance gains

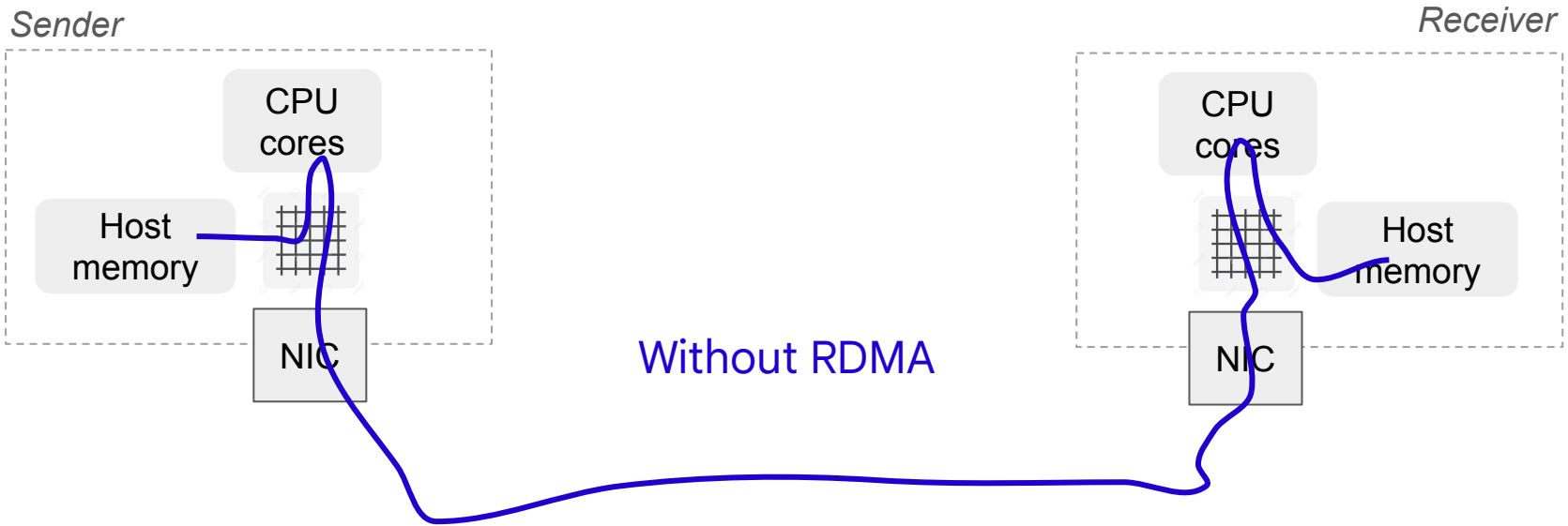
NIC Offloads

- Range from simple (checksum computation) to advanced (protocol offload)



RDMA

- Remote Direct Memory Access
- Removes CPU from transfers (almost entirely)



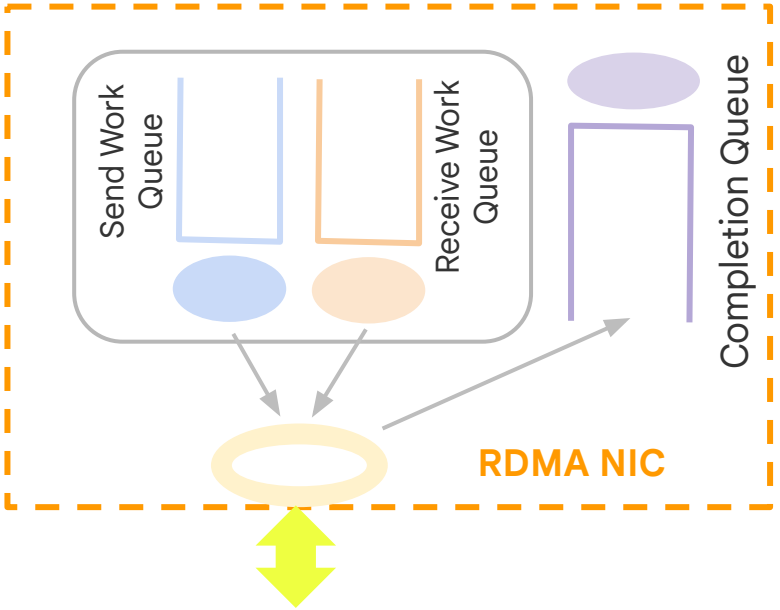
RDMA

- Remote Direct Memory Access
- Removes CPU from transfers (almost entirely)

With RDMA

RDMA - How?

- Queue pairs
  - Send and receive queues with Work Queue Elements (WQEs) in them
- WQEs
  - Pointer to memory to transfer / receive data in
- Completion Queue and Completion Queue Elements (CQEs)
  - Signals about whether transfers have completed or not



(Some) Advanced Features

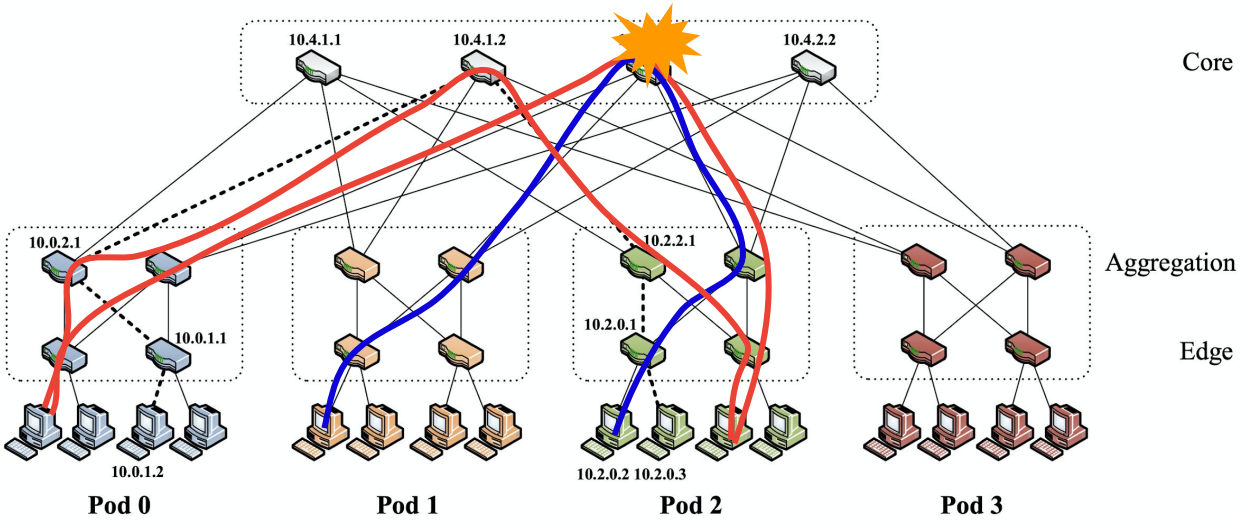
Advanced Features - Congestion Control

- Problem:
  - TCP is **buffer-filling**
  - Loss is a coarse signal
- Solution: use delay as a signal
  - Challenging to get a correct measurement
  - Why is this approach better in a DC?
- Note: this is not the only way to do advanced CC
  - Very active area of research!

```
if RTT < Target
    increase cwnd
   (Additively)
else
    decrease cwnd
   (Multiplicatively)
```

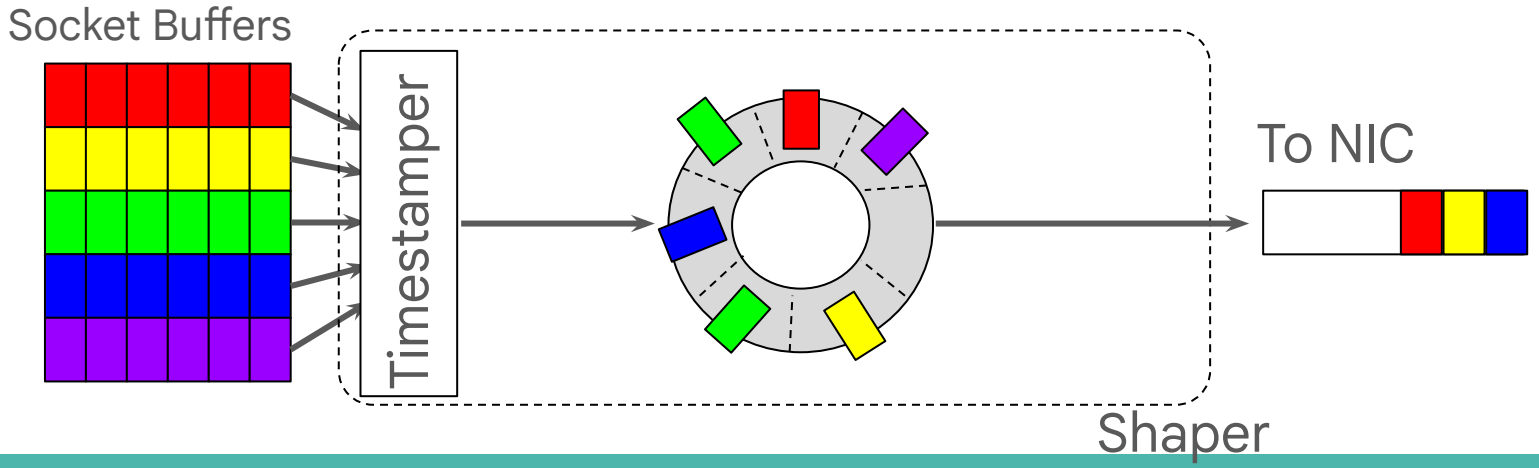
Advanced Features - Load Balancing

- Problem: ECMP can have collisions that cause hotspots
- Solution: repath based on congestion signals
  - Note: ECMP could place it badly again – repeat as necessary



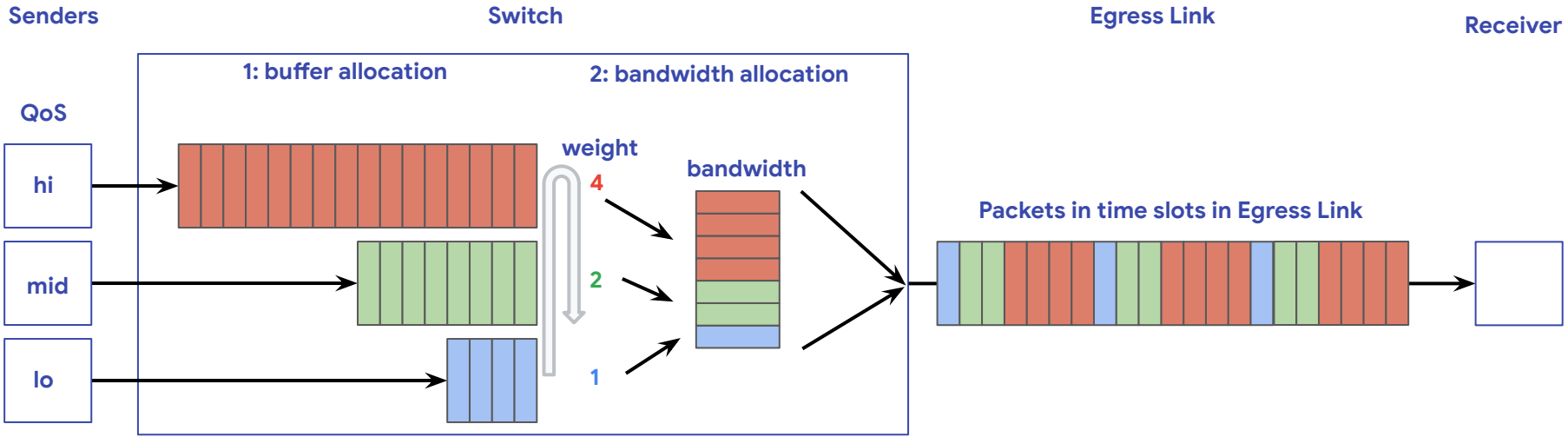
Advanced Features - Traffic Shaping

- Problem: Need to share bandwidth according to policy
  - Note: congestion control alone cannot do this
- Solution:
  - Classify traffic and rate limit it
  - Use a time wheel because managing multiple queues is difficult



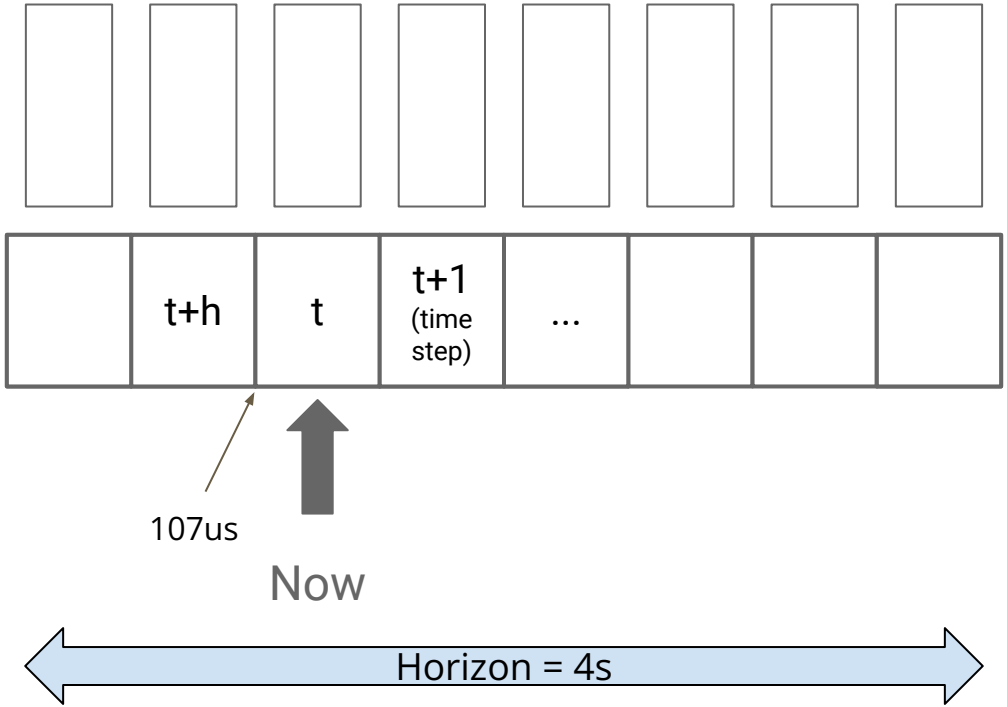
Advanced Features - QoS

- Problem: Need a way to express how much of a resource a flow should get
- Solution: priority classes and enforcement in the queue



Worksheet

Worksheet - Timing Wheel

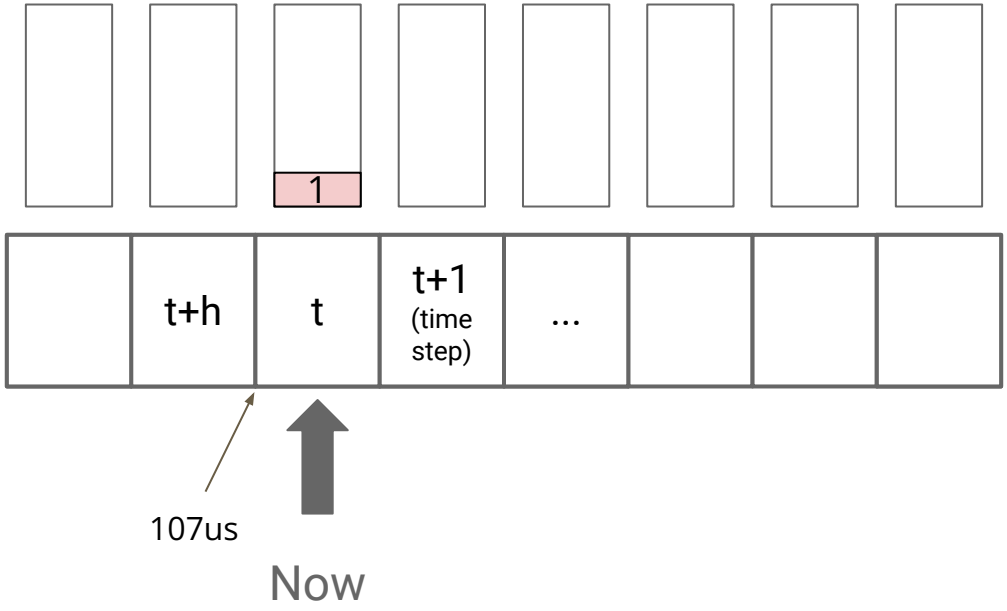


Worksheet - Timing Wheel Q4

Time step = 8us

A's Packet: 1

Timestamp = 112us

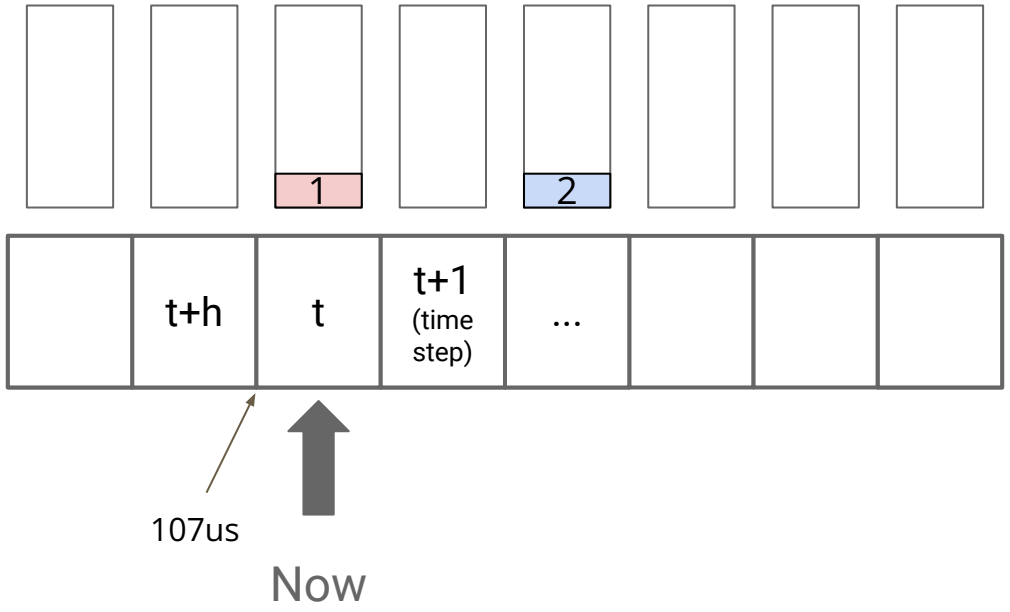


Worksheet - Timing Wheel Q5

Time step = 8us

A's Packet: 1
Timestamp = 112us

B's Packet: 2
Timestamp = 130us



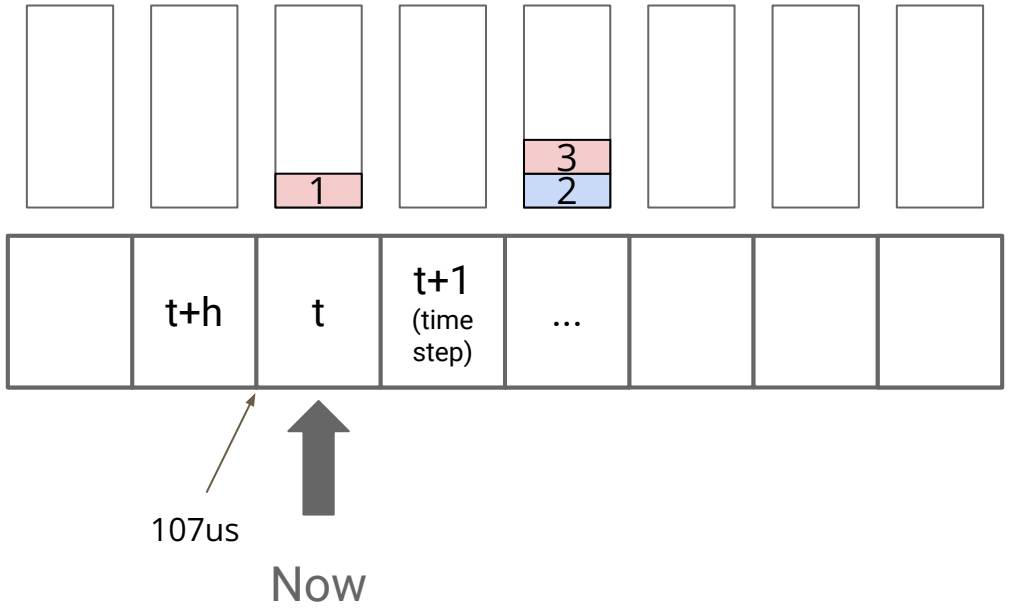
Worksheet - Timing Wheel Q6

Time step = 8us

A's Packet: 1
Timestamp = 112us

B's Packet: 2
Timestamp = 130us

A's Packet: 3
Timestamp = 124us



Worksheet - Timing Wheel Q7

Time step = 8us

A's Packet: 1
Timestamp = 112us

B's Packet: 2
Timestamp = 130us

A's Packet: 3
Timestamp = 124us