CS 204: Scheduling

Jiasi Chen

Lectures: MWF 12:10-1pm

Humanities and Social Sciences 1403

http://www.cs.ucr.edu/~jiasi/teaching/cs204_spring17/

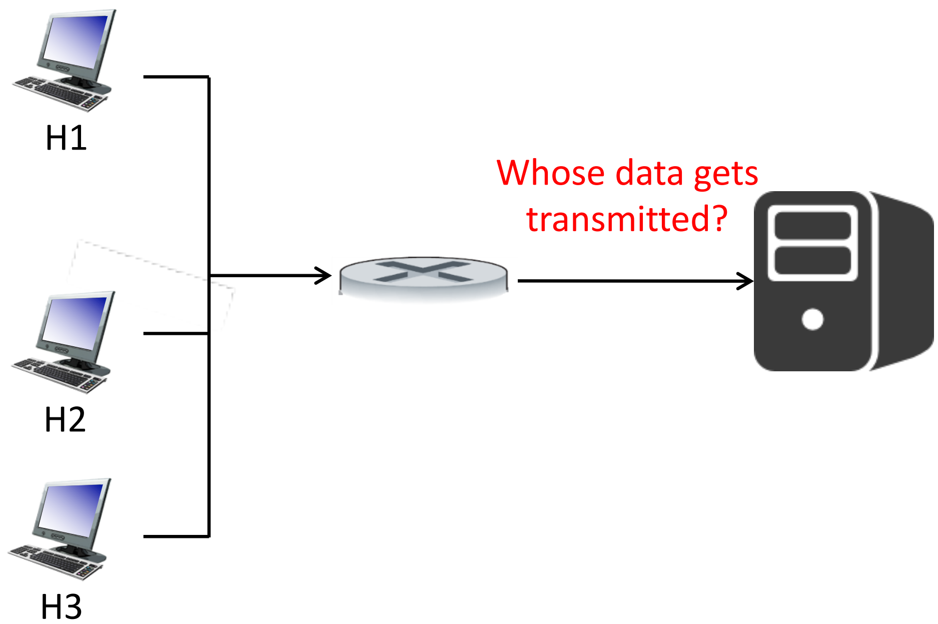
Overview

- What is scheduling?
  - Round robin
  - Weighted round robin
- Generalized processor sharing (GPS)
- Implementations of GPS
  - Deficit round robin
  - Weighted fair queuing
- Rate control with GPS: Token bucket

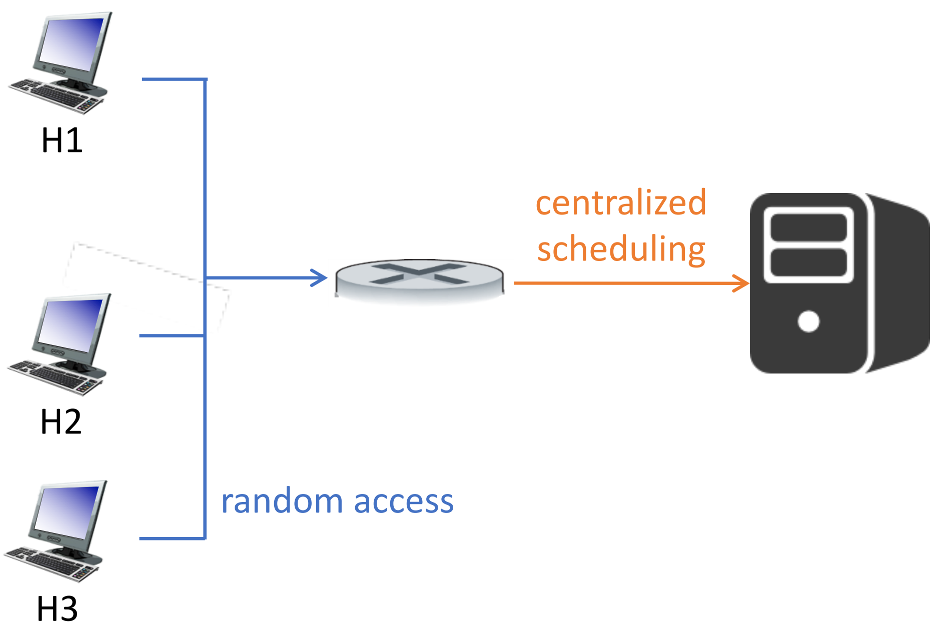
Q: How should a common resource be shared between multiple users?

What is scheduling?

- Send packets from multiple hosts
- Required features
  - Fair
  - Easy to implement
  - Scalable
  - Work-conserving
- Desirable features
  - Admission control
  - QoS for different types of traffic

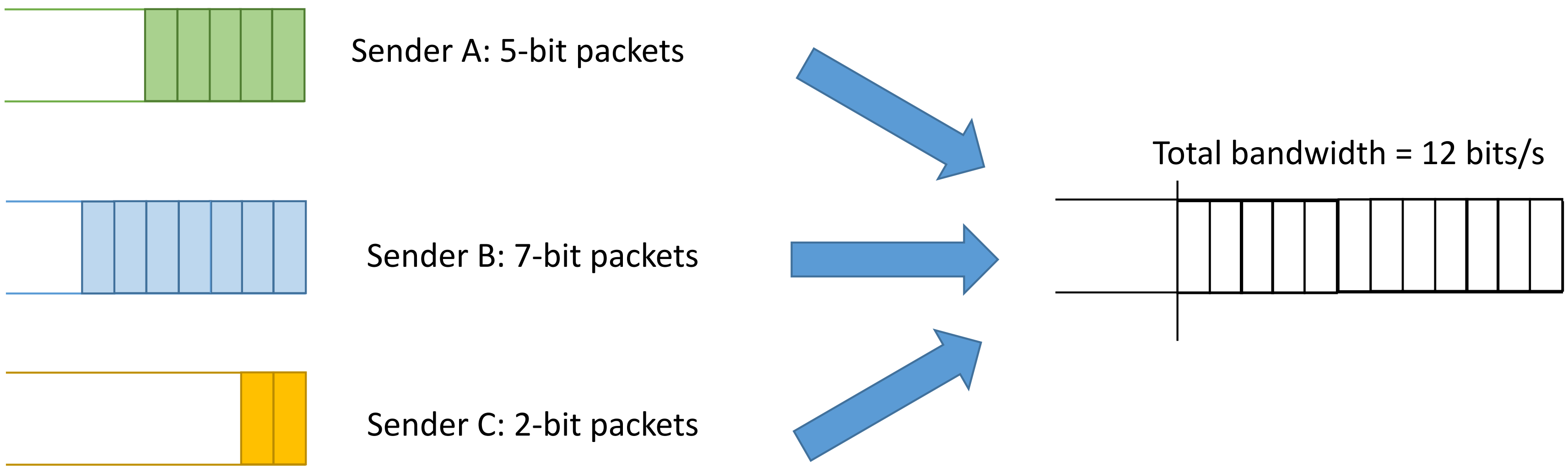


What is NOT scheduling?



Uplink?
Downlink?

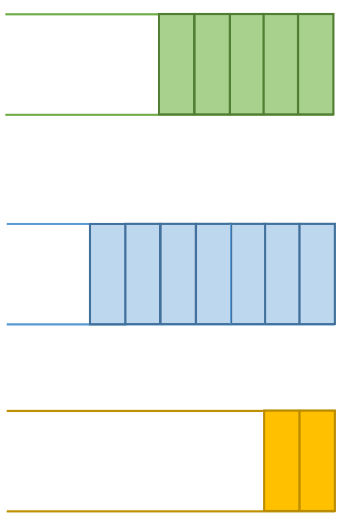
Toy example



Sender A: 5-bit packets
Sender B: 7-bit packets
Sender C: 2-bit packets
Total bandwidth = 12 bits/s

Fairness

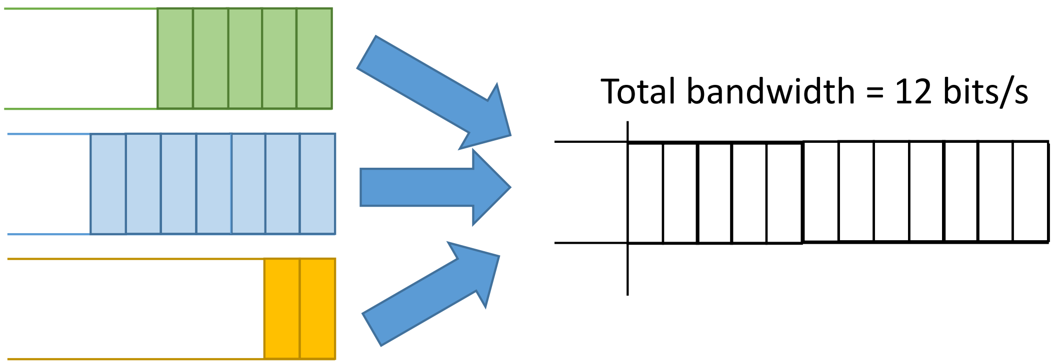
- One measure of fairness: max-min fair
  - Maximize the minimum rate across all users



	Max-min fair	NOT max-min fair
Sender A: 5-bit packets	5	4
Sender B: 7-bit packets	5	6
Sender C: 2-bit packets	2	2

How to Calculate Max-Min Fairness?

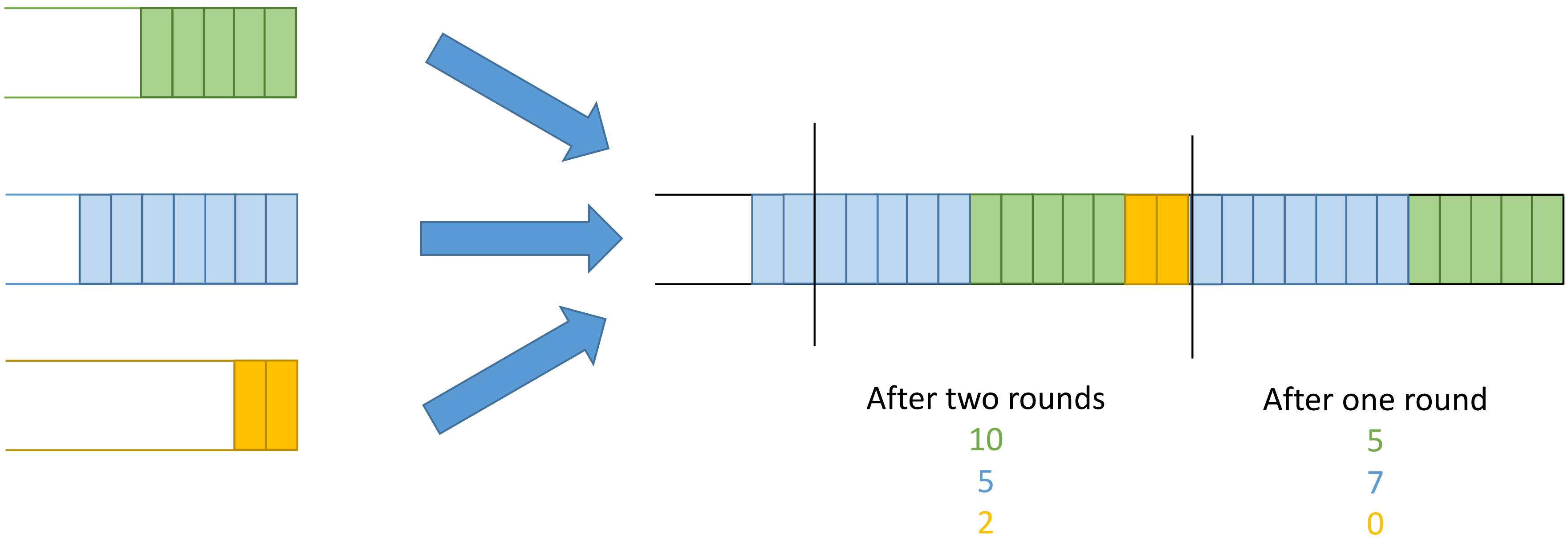
1. Compute the fair share of each unsatisfied flow
2. Assign the fair share to each unsatisfied flow
3. Take back any over-assignments, and repeat from Step 1



1. 12 bits / 3 users = 4 bits/user
2. Assign 4 4 4
3. Take back 2 → 4 4 2
1. 2 bits / 2 remaining users = 1 bit/user
2. Assign 5 5 2

Round robin

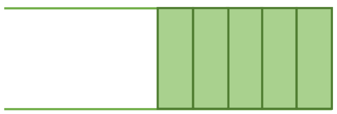
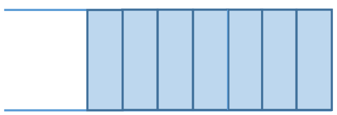
- Choose 1 packet from each subqueue



Is this max-min fair? (5,5,2)

Weighted round robin

- Choose certain # packets from each subqueue
  - # of packets sent = weight / packet size

	Weights	# packets	# packets (normalized *7)	Number of bits sent
	5	5/5 = 1 packet	7 packets	7 packets * 5 bits/packet = 35 bits
	5	5/7 packet	5 packets	5 packets * 7 bits/packet = 35 bits
	2	2/2 = 1 packet	7 packets	7 packets * 2 bits/packet = 14 bits

Is this max-min fair? (5,5,2)

Need to know packet size!

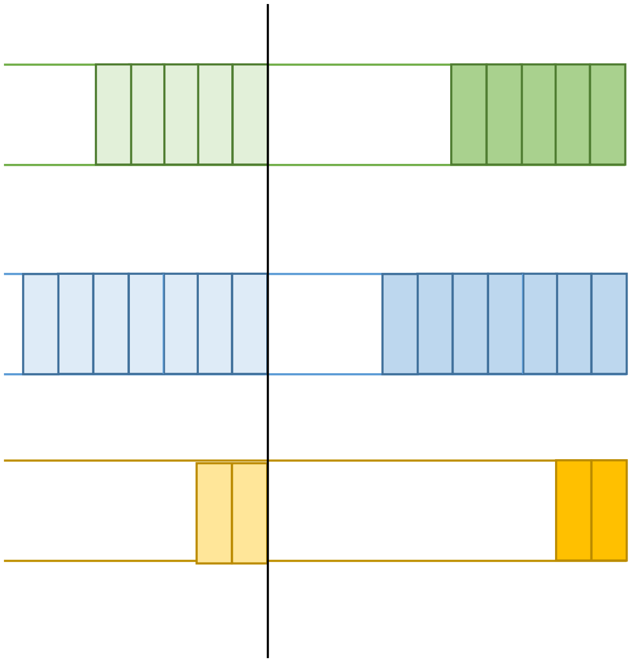
Overview

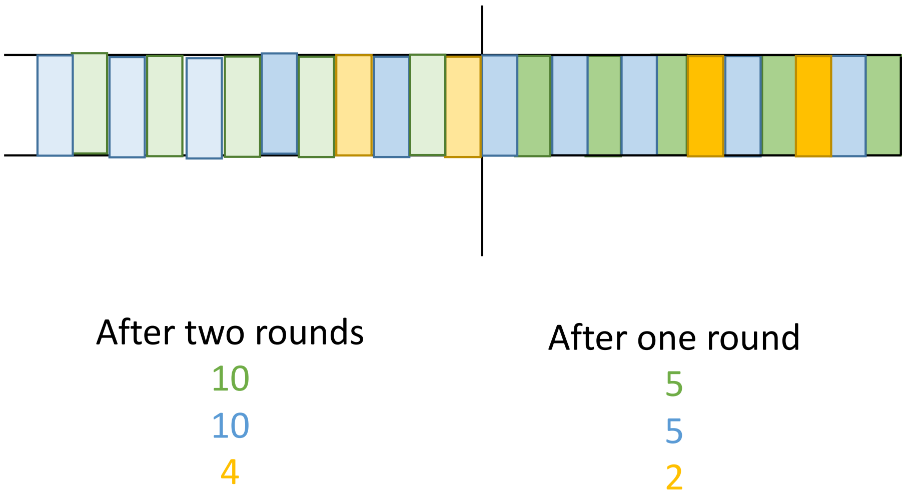
- What is scheduling?
  - Round robin
  - Weighted round robin
- Generalized processor sharing (GPS)
- Implementations of GPS
  - Deficit round robin
  - Weighted fair queuing
- Rate control with GPS: Leaky bucket

Q: How should a common resource be shared between multiple users?

Bit-by-bit Round Robin

- Will scheduling on a finer granularity do the trick?



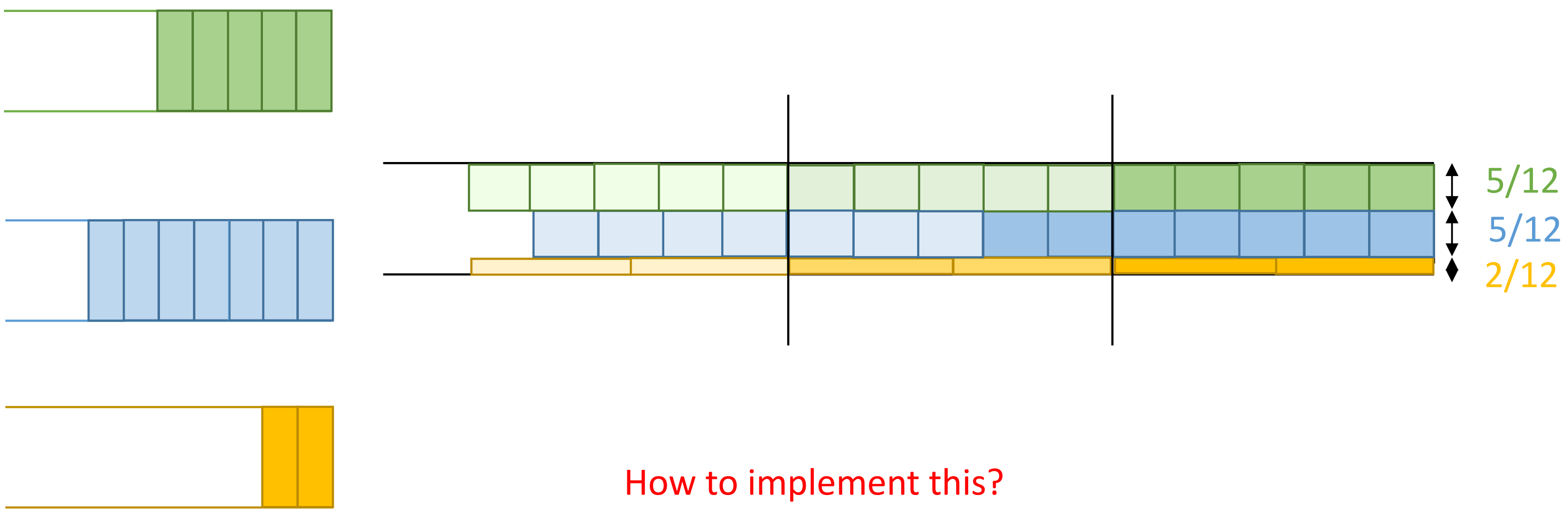


Is this max-min fair? (5,5,2)√

Is this implementable?

Generalized processor sharing

- Send infinitesimal amount of bits each round
- Idealized version of bit-by-bit round robin
- Not implementable, but achieves perfect fairness



How to implement this?

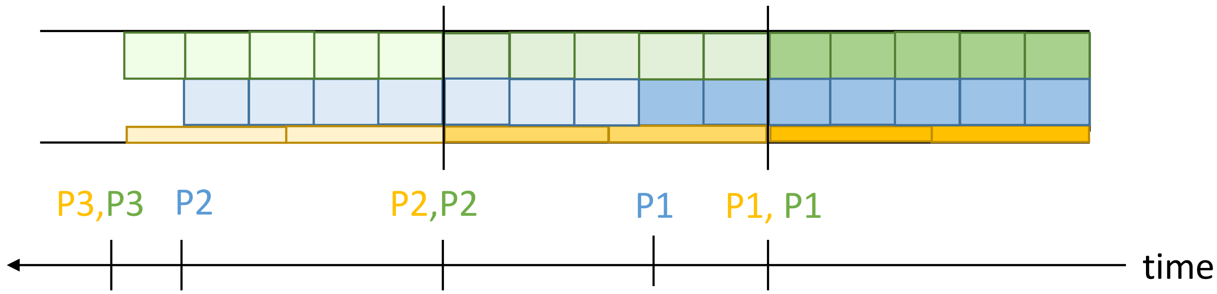
Overview

- What is scheduling?
  - FIFO
  - Round robin
  - Weighted round robin
- Generalized processor sharing (GPS)
- Implementations of GPS
  - Deficit round robin
  - Weighted fair queuing
- Rate control with GPS: Leaky bucket

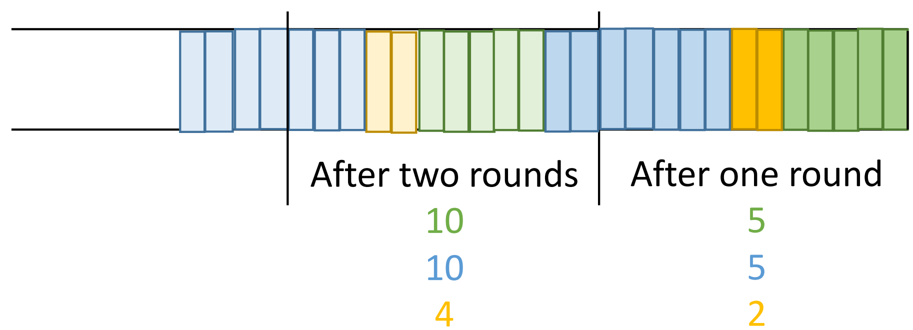
Q: How should a common resource be shared between multiple users?

Weighted fair queuing

- See which packets would finish first under GPS



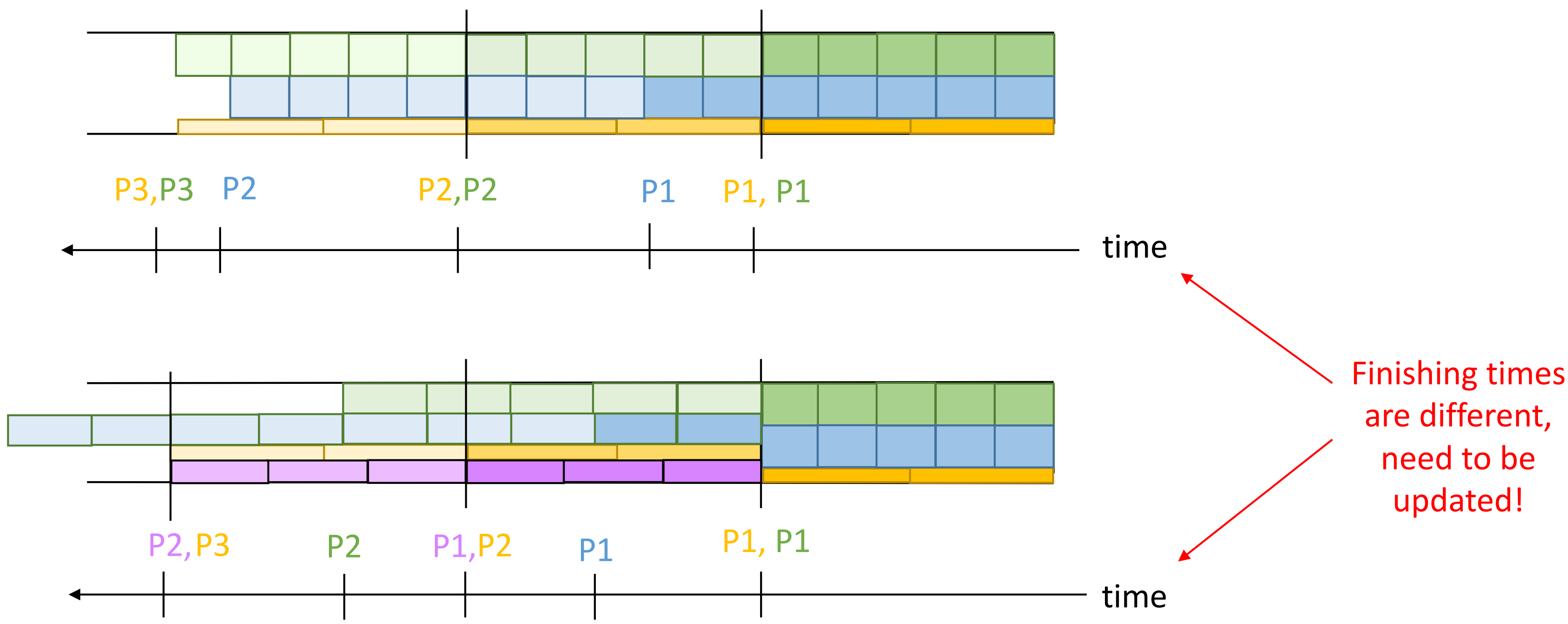
- Send packets in that order



O(log(n)) complexity
Why?

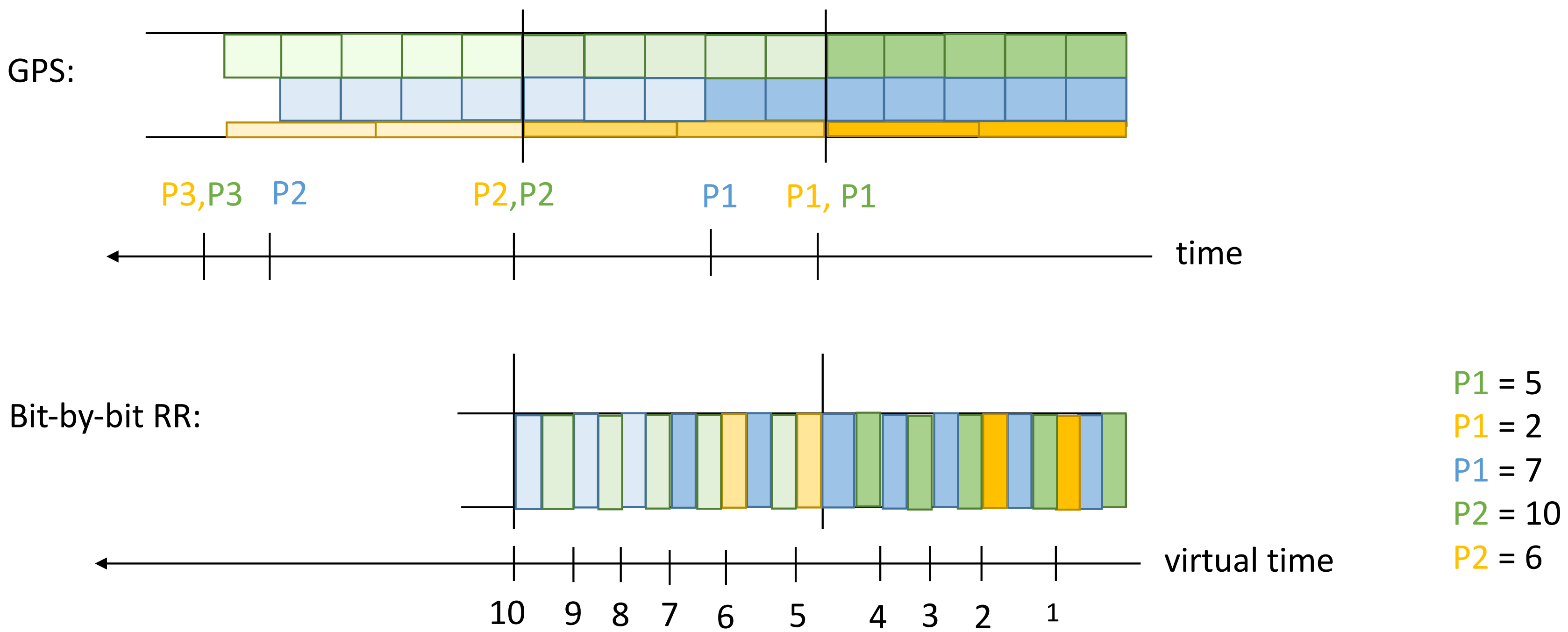
Weighted fair queuing (2)

- Problem: what if new user enters the system?



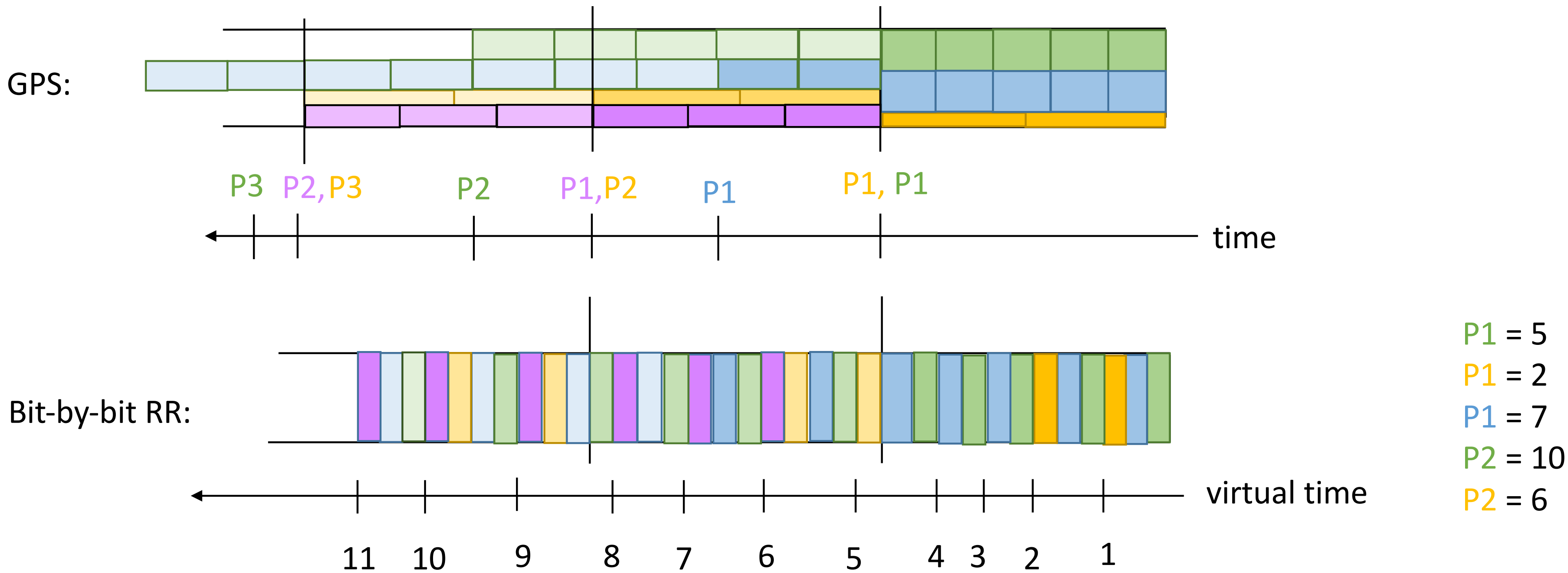
Weighted fair queuing (3)

- Solution: use "virtual time" = # of bit-by-bit round robin rounds



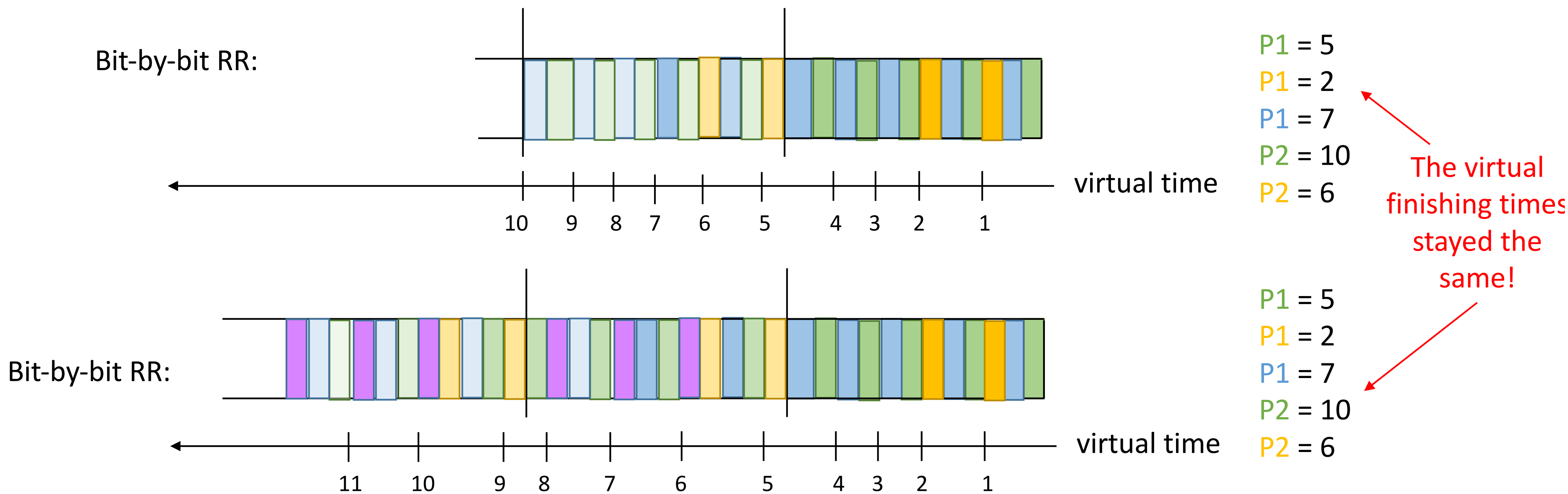
Weighted fair queuing (4)

- Let’s calculate the virtual finishing time with the new user



Weighted fair queuing (5)

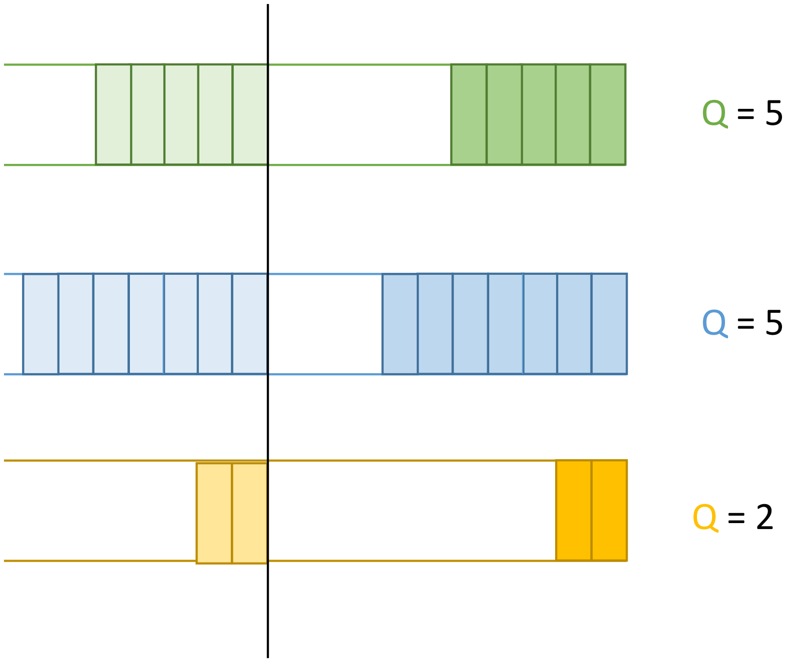
- Even if new user joins, the "virtual time" stays the same



Note: BBRR virtual time is shown here for ease of exposition; WFQ virtual time is slightly different. See Parekh-Gallager for details.

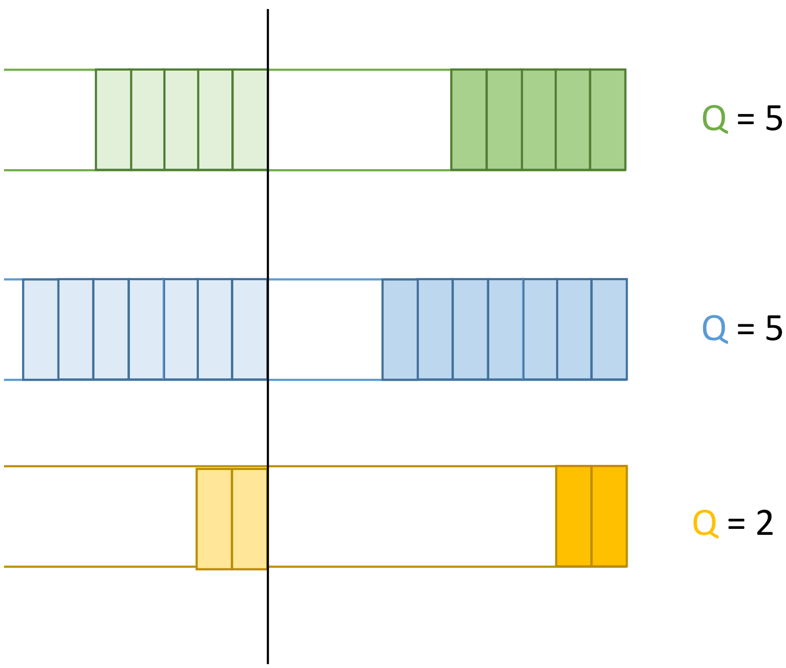
Deficit round robin

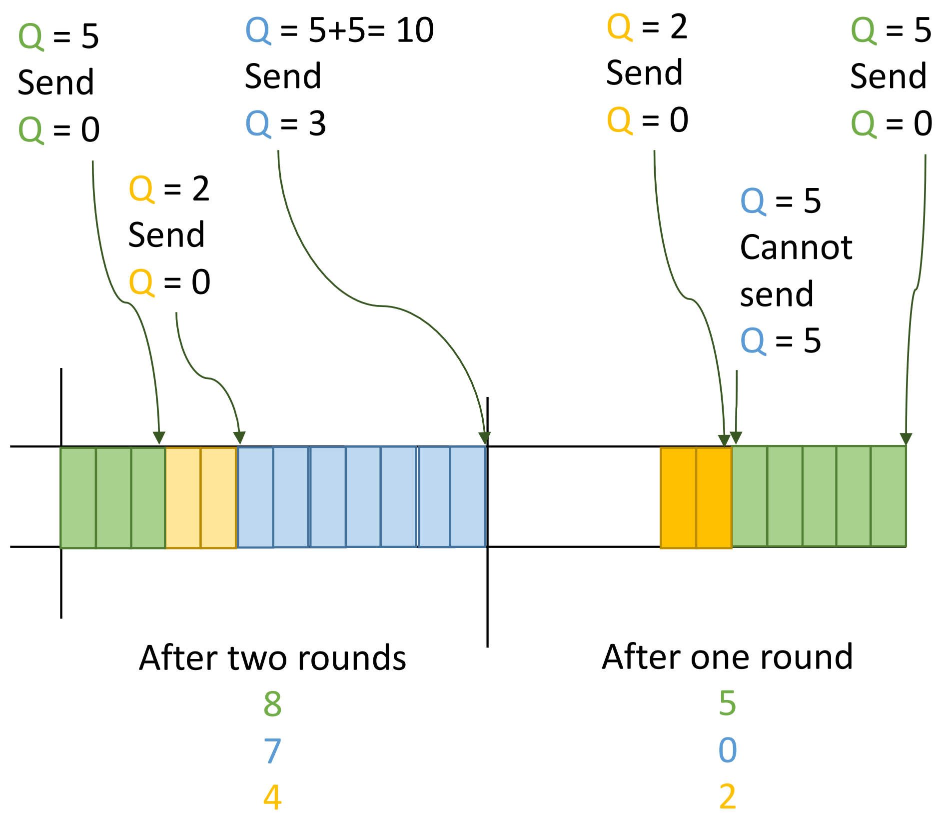
- That seems confusing… is there a simpler way?



- Define a quantum Q for each queue
- Can only send if have enough “credits” in Q
- Otherwise, save “credits” for next iteration

Deficit round robin





Is this max-min fair? (5,5,2)

O(1), but less accurate

Overview

- What is scheduling?
  - FIFO
  - Round robin
  - Weighted round robin
- Generalized processor sharing (GPS)
- Implementations of GPS
  - Deficit round robin
  - Weighted fair queuing
- Rate control with GPS: Token bucket

Q: How should a common resource be shared between multiple users?

Performance guarantees with GPS

- GPS implementations give us an average throughput

Rate of user k $$\rho_k = \frac{w_k C}{\sum_j w_j}$$

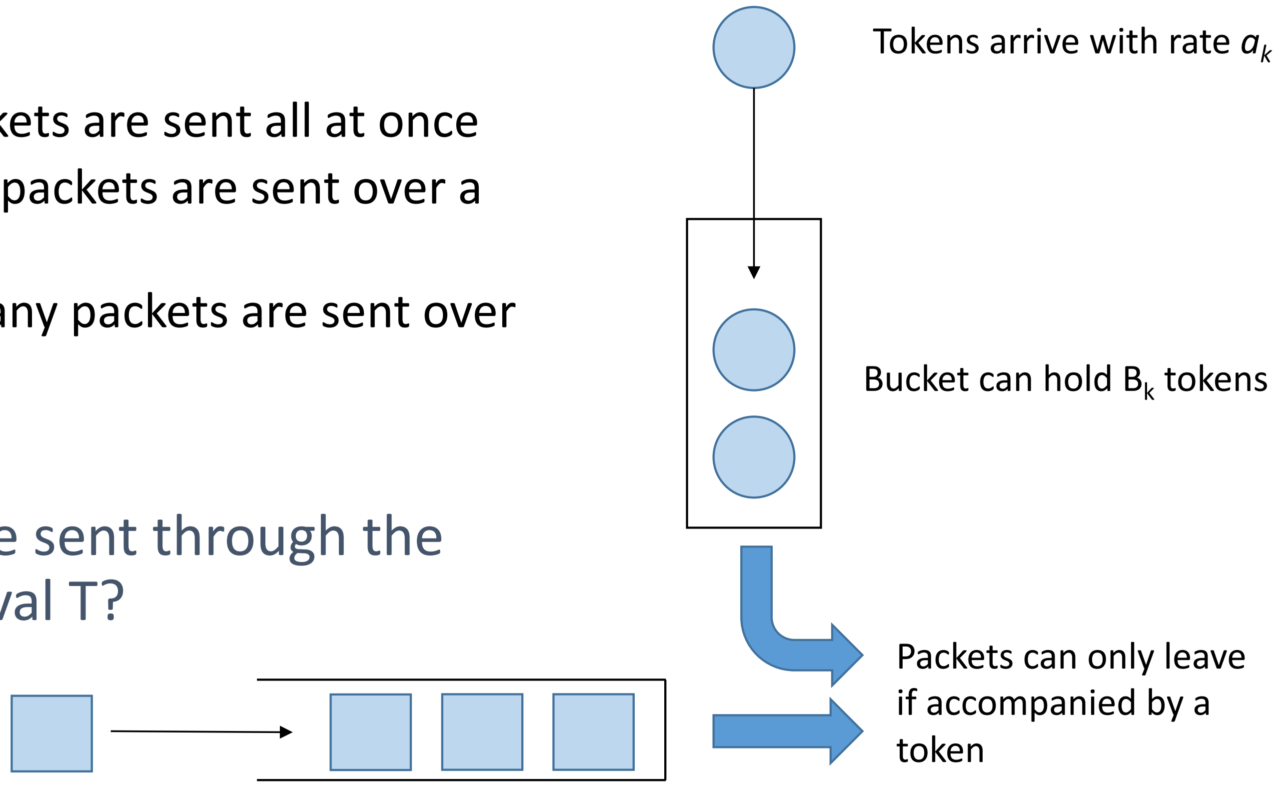
e.g., $w_1 = 5$, $w_2 = 5$, $w_3 = 2$

- What about delay guarantee?

Token Bucket

- Packet metrics
  - **Burst**: how many packets are sent all at once
  - **Peak rate**: how many packets are sent over a short period of time
  - **Average rate**: how many packets are sent over a long period of time

- How many packets are sent through the network in time interval T?
  - $a_k T + B_k$



GPS + Token Bucket

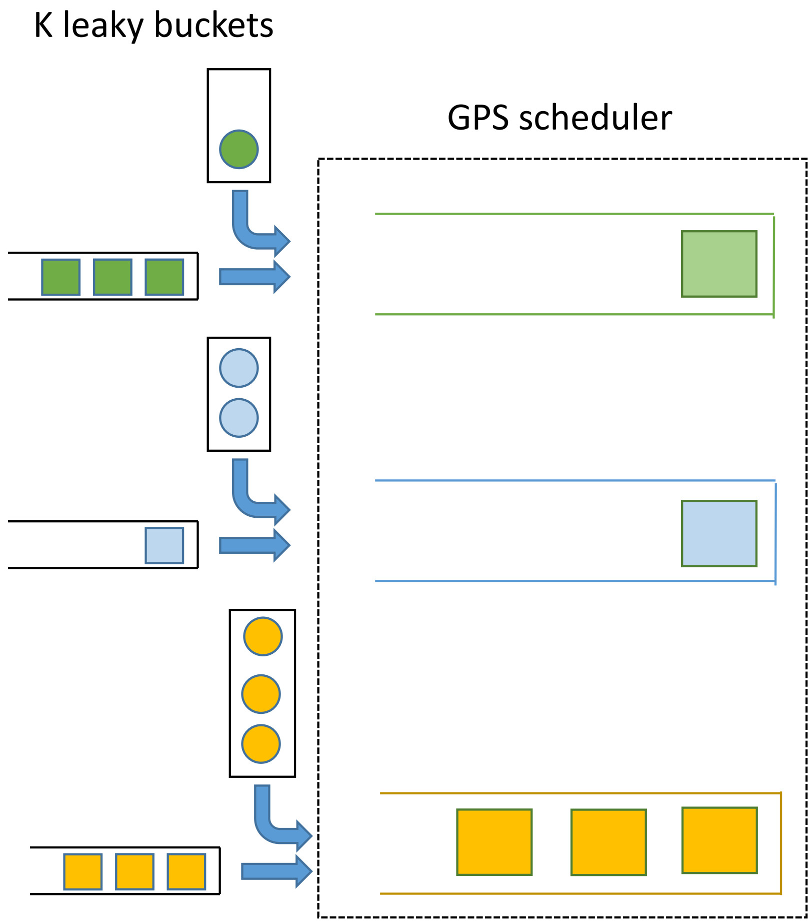
- Assume $\rho_k > a_k$. Then max delay experienced by a bit:

$$\frac{B_k}{\rho_k}$$

B_k Token bucket size

ρ_k GPS bandwidth

- Why?
  1. At most B_k packets waiting in GPS scheduler
  2. Delay = (B_k pkts) / (ρ_k pkts/s)



Implementation

- Different scheduling schemes available in Linux

sch_api.c	47423 bytes	sch_htb.c	43567 bytes
sch_atm.c	19306 bytes	sch_ingress.c	4512 bytes
sch_blackhole.c	1118 bytes	sch_mq.c	5960 bytes
sch_cbq.c	42206 bytes	sch_mqprio.c	11017 bytes
sch_choke.c	14276 bytes	sch_multiq.c	9197 bytes
sch_codel.c	8570 bytes	sch_netem.c	27979 bytes
sch_drr.c	11531 bytes	sch_pie.c	16002 bytes
sch_dsmark.c	11485 bytes	sch_plug.c	6729 bytes
sch_fifo.c	4373 bytes	sch_prio.c	8091 bytes
sch_fq.c	22457 bytes	sch_qfq.c	42960 bytes
sch_fq_codel.c	19635 bytes	sch_red.c	8651 bytes
sch_generic.c	24823 bytes	sch_sfb.c	16872 bytes
sch_gred.c	13937 bytes	sch_sfq.c	22441 bytes
sch_hfsc.c	40530 bytes	sch_tbf.c	14151 bytes
sch_hhf.c	21818 bytes	sch_teql.c	12301 bytes

Implementation

- `tc` traffic control tool

```
jc@jc-ideapad500s:~$ tc qdisc show dev wlp2s0
qdisc mq 0: root qdisc pfifo_fast 0: parent :1 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
qdisc pfifo_fast 0: parent :2 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1qdisc pfifo_fast 0: parent :3
bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
qdisc pfifo_fast 0: parent :4 bands 3 priomap  1 2 2 2 1 2 0 0 1 1 1 1 1 1 1 1
```

Add token bucket

```
jc@jc-ideapad500s:~$ tc class show dev wlp2s0
class htb 1:10 parent 1:1 prio 0 rate 1Mbit ceil 1Mbit burst 1600b cburst 1600
class htb 1:1 root rate 1Mbit ceil 1Mbit burst 1600b cburst 1600b
```

Sources

- *An Introduction to Computer Networks*, Peter Dordal
  http://intronetworks.cs.luc.edu/current/html/queuing.html

- A. Parekh and R. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case", *ToN* 1993.