CS241 Lawrence Angrave L27 – UDP & TCP packets. Scheduling II

Why might a process be placed on the ready queue?
What is 'wait time'? Total wait time, or the first waiting before it is scheduled the first time?
Write a formula for the wait time based on arrival time, execution time and completion time

Determine the scheduling sequence and calculate the average wait time of the following schedulers
In a tie-break schedule the earliest arriving job

Round robin (quanta = 10ms)

Process	Arrival Time(ms)	Execution Time(ms)
P1	0	30
P2	0	20
P3	0	20
P4	10	10

0..10	.. 20	..30	..40	..50	.. 60	.. 70	..80

Shortest Job First

Process	Arrival Time(ms)	Execution Time(ms)
P1	0	30
P2	0	20
P3	0	20
P4	10	10

0..10	20	30	40	50	60	70	80

First Come First Served (assume arrive in order P1,P2,P3)

Process	Arrival Time(ms)	Execution Time(ms)
P1	0	30
P2	0	20
P3	0	20
P4	10	10

0..10	20	30	40	50	60	70	80

Pre-emptive Shortest Job First (assume interrupted jobs are placed at the front of the queue)

Process	Arrival Time(ms)	Execution Time(ms)
P1	0	30
P2	0	20
P3	0	20
P4	10	10

0..10	20	30	40	50	60	70	80

Pre-emptive Priority (higher value = higher priority)

Process	Arrival (ms)	Execution (ms)	Priority
P1	0	30	1
P2	0	20	2
P3	0	20	3
P4	10	10	4

0..10	20	30	40	50	60	70	80

Which schedulers can suffer from starvation?
Which schedulers are appropriate for batch jobs?

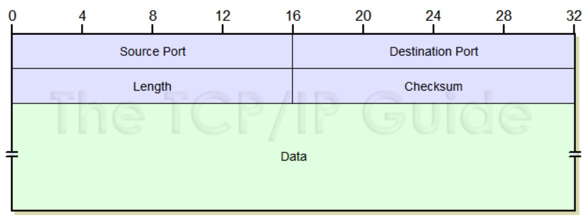
What scheduler does Linux use?

What is the convoy effect (poor I/O parallelism)?

What about threads? What does *nice* do?

How do you listen for IPv6 UDP packets?

```
// get host info, make socket, bind it to port 300
memset(&hints, 0, sizeof hints);
hints.ai_family = _____
hints.ai_socktype =  _____
hints.ai_flags =  ______;

getaddrinfo(______, "______", &hints, &res);

sockfd = socket(res->ai_family, res->ai_socktype, res->ai_protocol);

bind(sockfd, res->ai_addr, res->ai_addrlen);

// no need to accept(), just recvfrom():

struct sockaddr_storage addr;
fromlen = sizeof addr;

// ssize_t  recvfrom(int socket, void *buffer, size_t length,
   int flags,  struct sockaddr *address, socklen_t * address_len);

byte_count = recvfrom(sockfd, buf, sizeof(buf), 0, &addr, &fromlen);
```



UDP format from www.tcpipguide.com

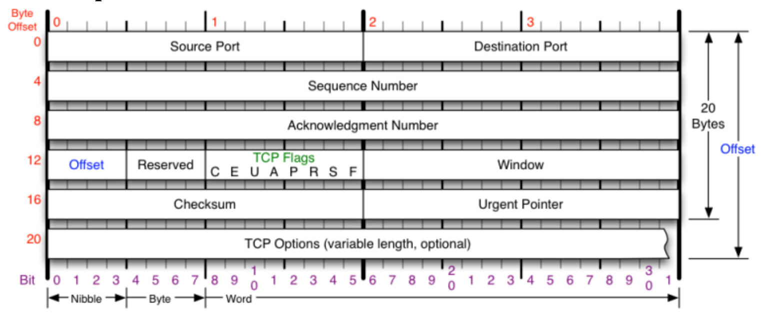
TCP Packets:

What is "SYN. SYK-ACK. ACK" ?

What is a SYN flood?

What is the sequence number and what is it used for? What is its initial value & why?

I see the port number but where is the machine's IP address?



TCP Flags

C E U A P R S F

Congestion Window
C 0x80 Reduced (CWR)
E 0x40 ECN Echo (ECE)
U 0x20 Urgent
A 0x10 Ack
P 0x08 Push
R 0x04 Reset
S 0x02 Syn
F 0x01 Fin

Congestion Notification

ECN (Explicit Congestion Notification). See RFC 3168 for full details, valid states below.

Packet State	DSB	ECN bits
Syn	0 0	1 1
Syn-Ack	0 0	0 1
Ack	0 1	0 0
No Congestion	0 1	0 0
No Congestion	1 0	0 0
Congestion	1 1	0 0
Receiver Response	1 1	0 1
Sender Response	1 1	1 1

TCP Options

0 End of Options List
1 No Operation (NOP, Pad)
2 Maximum segment size
3 Window Scale
4 Selective ACK ok
8 Timestamp

Checksum

Checksum of entire TCP segment and pseudo header (parts of IP header)

Offset

Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.

RFC 793

Please refer to RFC 793 for the complete Transmission Control Protocol (TCP) Specification.

Source: http://nmap.org/book/tcpip-ref.html

Congestion control? Receive Window? Lost packet retransmission? Packet re-ordering? Secure?

Round Robin

Process	Arrival Time(ms)	Execution Time(ms)
P1	0	30
P2	0	20
P3	0	20
P4	10	10

0..10	20	30	40	50	60	70	80
P1	P2	P3	P4	P1	P2	P3	P1

Wait = End-Arrival-Execution
50 +40 + 50 + 40 = 160ms. Average Wait = 40 ms

Shortest Job First

Process	Arrival Time(ms)	Execution Time(ms)
P1	0	30
P2	0	20
P3	0	20
P4	10	10

0..10	20	30	40	50	60	70	80
P2	P2	P4	P3	P3	**P1**	P1	P1

Total Wait = 50 + 30 + 0 + 10 = 90 ms. Average wait = 90/4 = 22.5 ms

First Come First Served (assume arrive in order P1,P2,P3)

Process	Arrival Time(ms)	Execution Time(ms)
P1	0	30
P2	0	20
P3	0	20
P4	10	10

0..10	20	30	40	50	60	70	80
P1	P1	P1	P2	P2	P3	P3	**P4**

Total Wait = 0 + 30 + 50 + 60 = 140 ms. Average wait = 35 ms

Pre-emptive Shortest Job First

Process	Arrival Time(ms)	Execution Time(ms)
P1	0	30
P2	0	20
P3	0	20
P4	10	10

0..10	20	30	40	50	60	70	80
P2	**P4**	P2	P3	P3	P1	P1	P1

Total Wait = 50 + 10 + 30 + 0 = 90 ms. Average wait = 22.5 ms

Pre-emptive Priority (higher value = higher priority)

Process	Arrival (ms)	Execution (ms)	Priority
P1	0	30	1
P2	0	20	2
P3	0	20	3
P4	10	10	4

0..10	20	30	40	50	60	70	80
P3	**P4**	**P3**	P2	P2	P1	P1	P1

Total Wait = 50 + 30 + 10 + 0 = 90 ms. Average wait = 22.5 ms

Which scheduler has poor I/O parallelism (suffers from the "Convoy Effect")?
FCFS (Processes that could be using I/O have to queue behind long-running CPU job). Note, you could also make a similar argument for non-premptive SJF.

Which schedulers can suffer from starvation?
Pre-emptive SJF (long jobs may never be scheduled); Pre-emptive priority (low priority jobs may never be scheduled)
Which schedulers are appropriate for batch jobs? Ans: Depends on your requirements!
What scheduler does Linux use? What about threads? What does *nice* do?

Completely Fair Scheduler ("Stride scheduler"; inspired from similar network flow scheduling – gives additional time to processes that are in the waiting state more often than the executing state "If you only took small sips in the recent past, you can take longer drink now")