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?

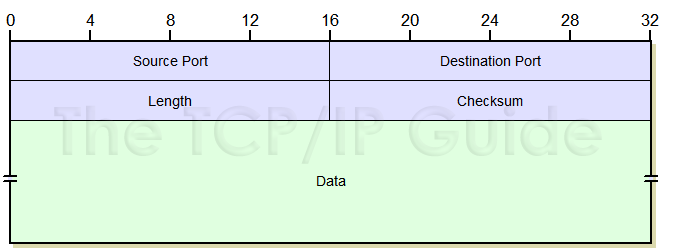
Which schedulers are appropriate for batch jobs?

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 = \_\_\_\_\_\_;

UDP format from www.tcpipguide.com

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);

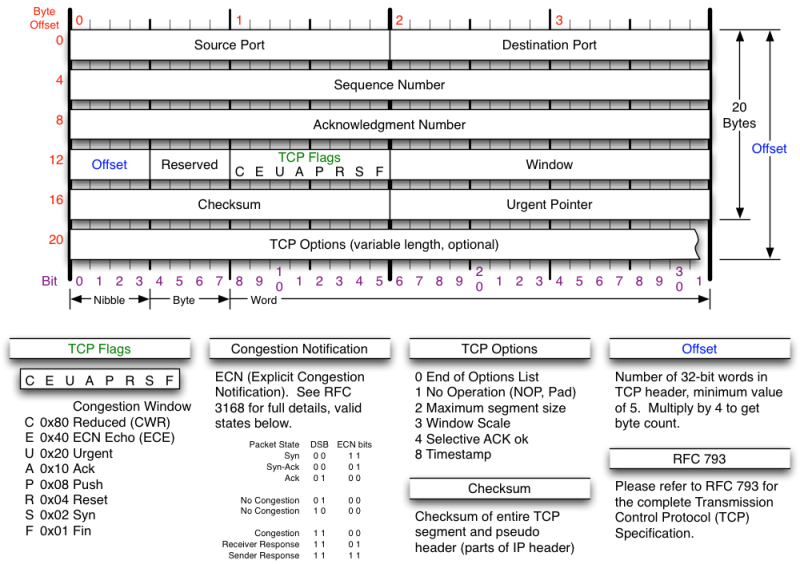
TCP Packets:

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?



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")