CS241 #28 — Scheduling Algorithms. Networking (UDP, IPv6, Building a better HTTP server)

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(=duration) 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)	Burst Time(ms)	Wait 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)	Burst Time(ms)	Wait 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)	Burst Time(ms)	Wait 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 T	Burst T	Wait T
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	Burst	Priority	Wait
P1	0	30	2	
P2	0	20	4	
P3	0	20	1	
P4	10	10	3	

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)?

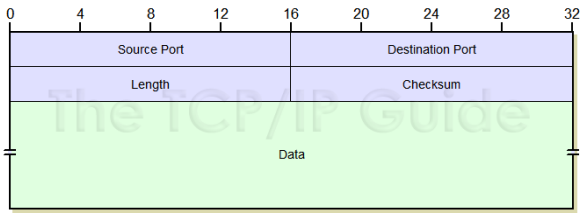
What about threads? What does *nice* do?

Webserver challenge : How do I make a web server that can serve different files?

Parse a string "GET /mypage.pdf HTTP/1.0"

```
char method[16],url[2048],protocol[32];
sscanf( buffer, "%15s %2047s %31s", method, url, protocol);
sprintf(filename,"/var/www/mysite/%s", url);
// Todo: use realpath() and validate directory is a subdirectory

int fd = open( filename, O_RDONLY);
struct stat file_stat;
fstat(fd, &file_stat);
off_t len = file_stat.st_size;
void * buf = mmap(NULL, len, PROT_READ, MAP_SHARED,fd,0);
// Todo: write headers including MIME-TYPE and size
write( client, buf, len);
```

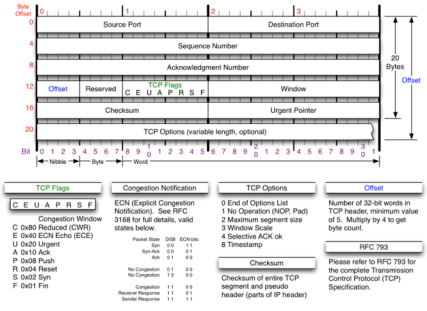
Example: How do you **listen** for IPv6 **UDP** packets?



UDP format from www.tcpipguide.com

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

ok= getaddrinfo(_______, _______, &hints, &res);

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

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

struct sockaddr_storage addr;
fromlen = sizeof addr;

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

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

TCP Packets: What is "SYN. SYK-ACK. ACK" ?



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

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?

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