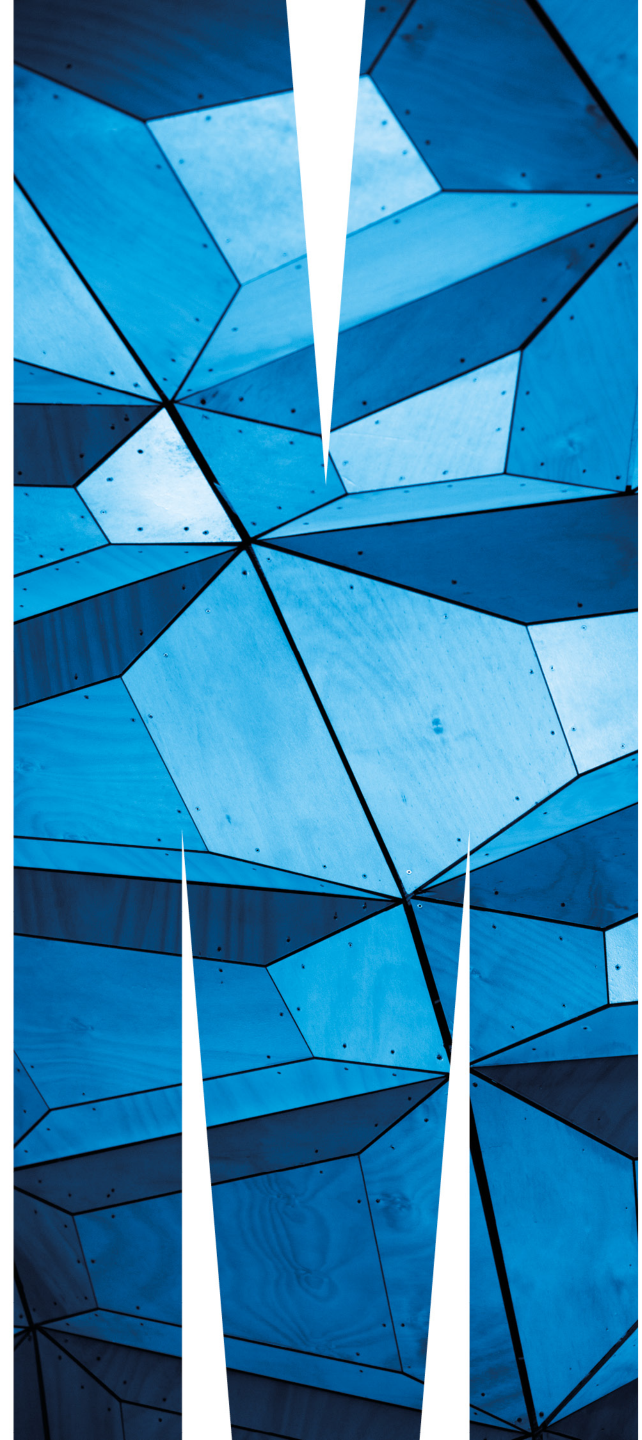
MONASH
INFORMATION
TECHNOLOGY

FIT2100 Semester 2 2017

Lecture 6: Uniprocessor Scheduling

(Reading: Stallings, Chapter 9)

Jojo Wong



Lecture 6: Learning Outcomes

- Upon the completion of this lecture, you should be able to:
  - Discuss the differences among long-, medium-, and short-term scheduling
  - Assess the performance of different scheduling policies
  - Understand the scheduling technique used in Unix

MONASH University

What is the aim of processor scheduling?

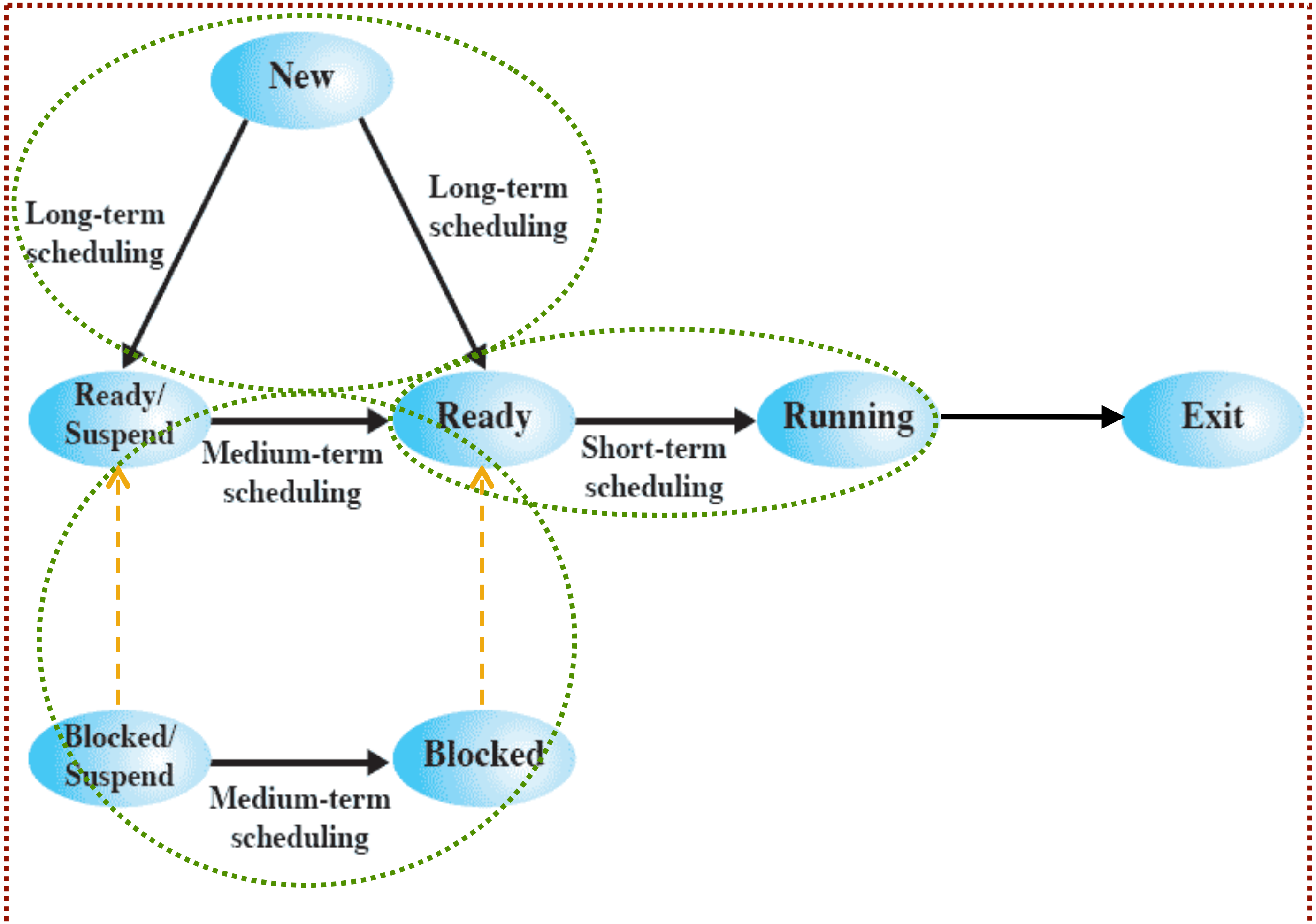
Processor Scheduling

- To assign processes to be executed by the processor in a way that meets system objectives, such as response time, throughput, and processor efficiency
- Broken down into three separate functions:

Types of Scheduling

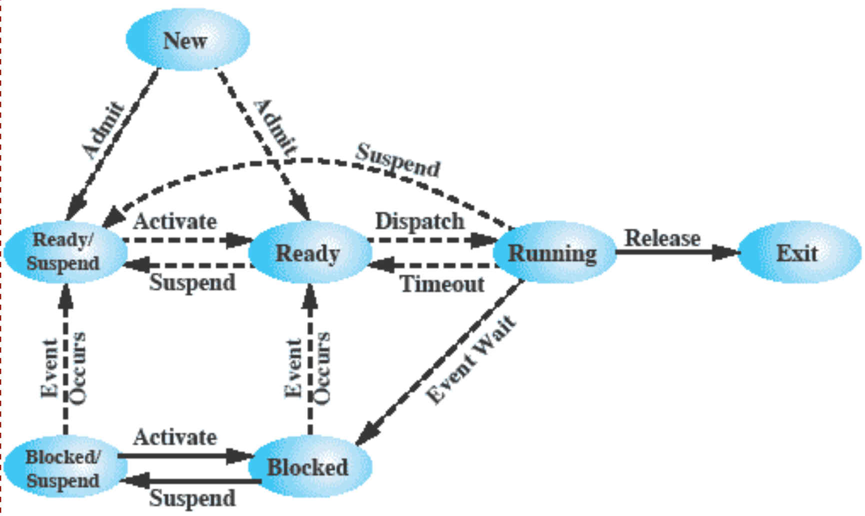
Long-term scheduling	The decision to add to the pool of processes to be executed
Medium-term scheduling	The decision to add to the number of processes that **are partially or fully in main memory**
Short-term scheduling	The decision as to **which available process** will be executed by the processor

Scheduling: Process State Transitions

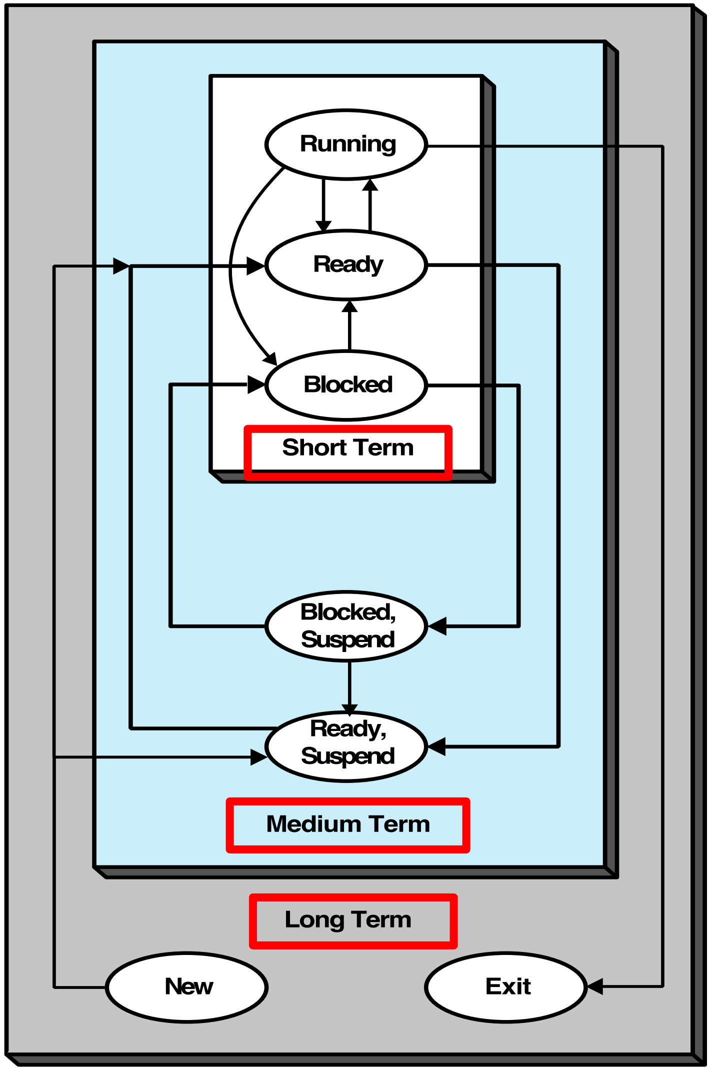


Scheduling: Levels

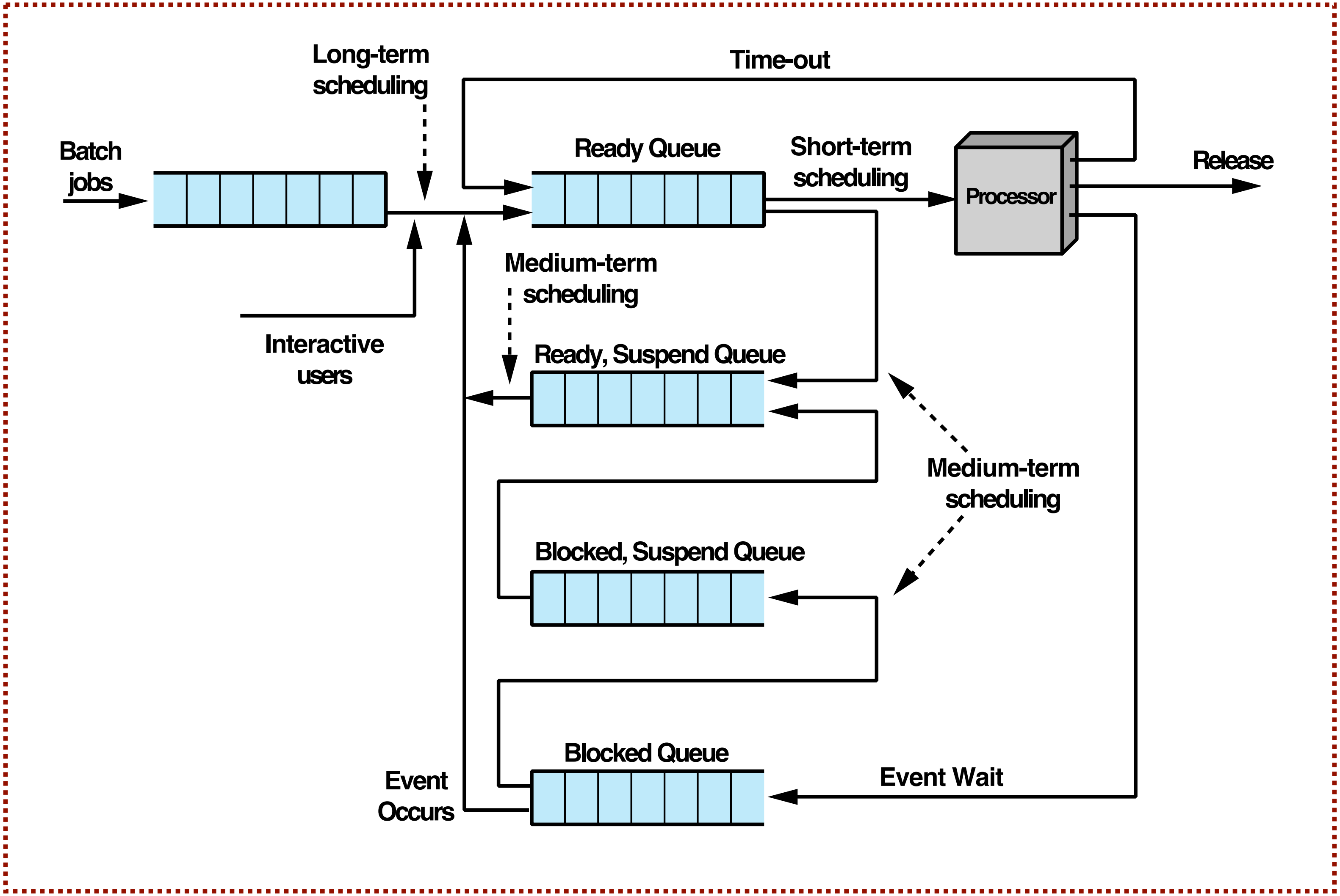
(Process State Diagram)



(b) With Two Suspend States



Scheduling: Queueing Diagram



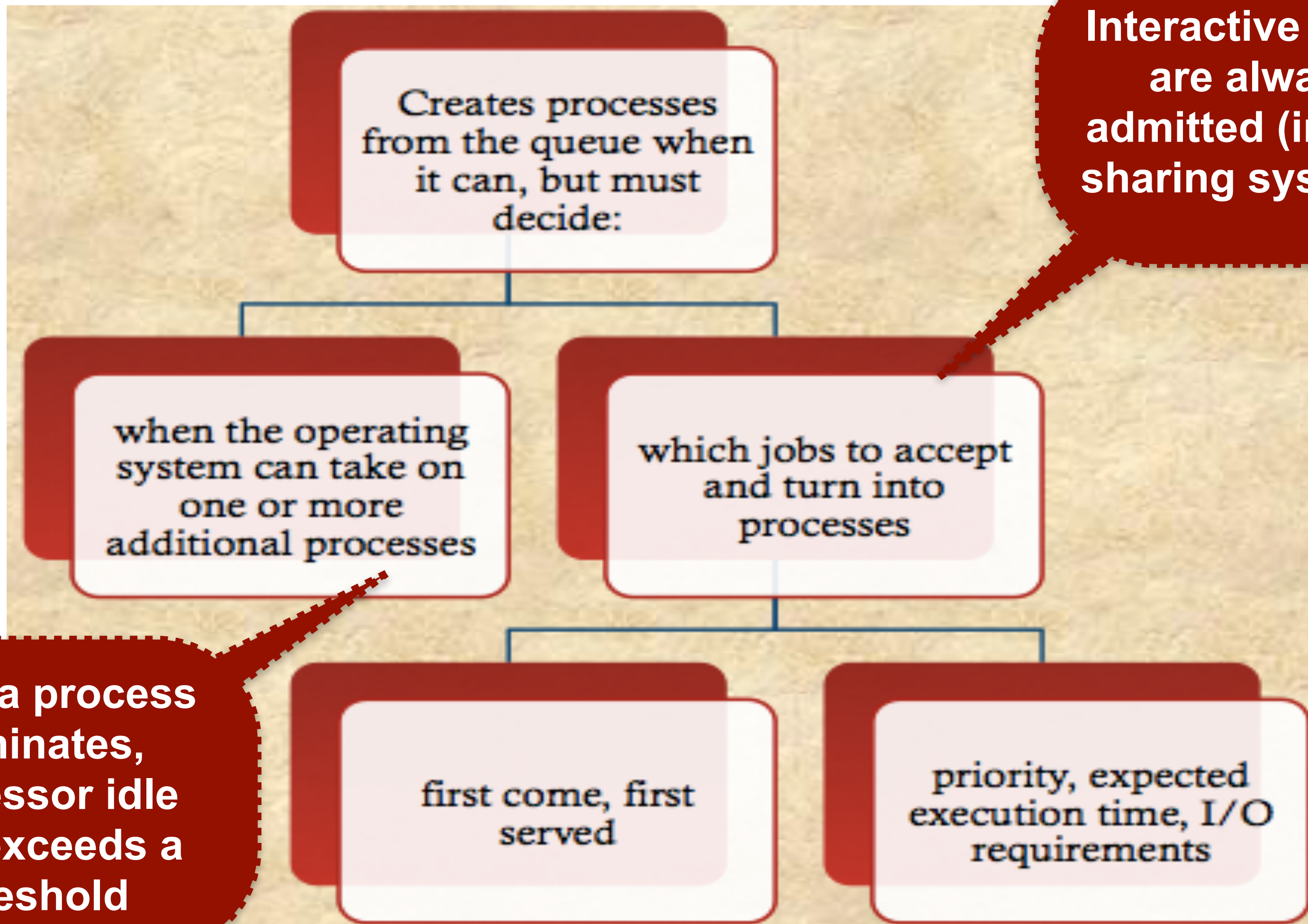
Monash University

What are the three different scheduling functions?

Long-Term Scheduling

- Determines which programs are admitted to the system for processing
- Controls the degree of multiprogramming
  - The more processes that are created, the smaller the percentage of time that each process can be executed
  - May limit to provide satisfactory service to the current set of processes

Long-Term Scheduling



Interactive users are always admitted (in time sharing systems)

When a process terminates, processor idle time exceeds a threshold

Medium-Term Scheduling

- Part of the swapping function
- Swapping-in decisions are based on the need to manage the degree of multiprogramming
  - Considers the memory requirements of the swapped-out processes

Memory management can be an issue

Short-Term Scheduling

- Known as the **dispatcher** or **CPU scheduler**
- Executes most frequently
- Makes the fine-grained decision of which process to execute next
- Invoked whenever an event occurs
  - May lead to blocking of the current process or
  - May provide an opportunity to preempt a currently running process in favour of another

Short-Term Scheduling

- Known as the **dispatcher** or **CPU scheduler**
- Executes most frequently
- Makes the fine-grained decision of which process to execute next
- Invoked whenever an event occurs
  - May lead to blocking of the current process or
  - May provide an opportunity to preempt a currently running process in favour of another

Clock interrupts
I/O interrupts
System calls
Signals (e.g. semaphores)

Short-Term Scheduling: Criteria

- Main objective is to allocate processor time to optimise certain aspects of system behaviour
- A set of criteria is needed to evaluate the scheduling policy

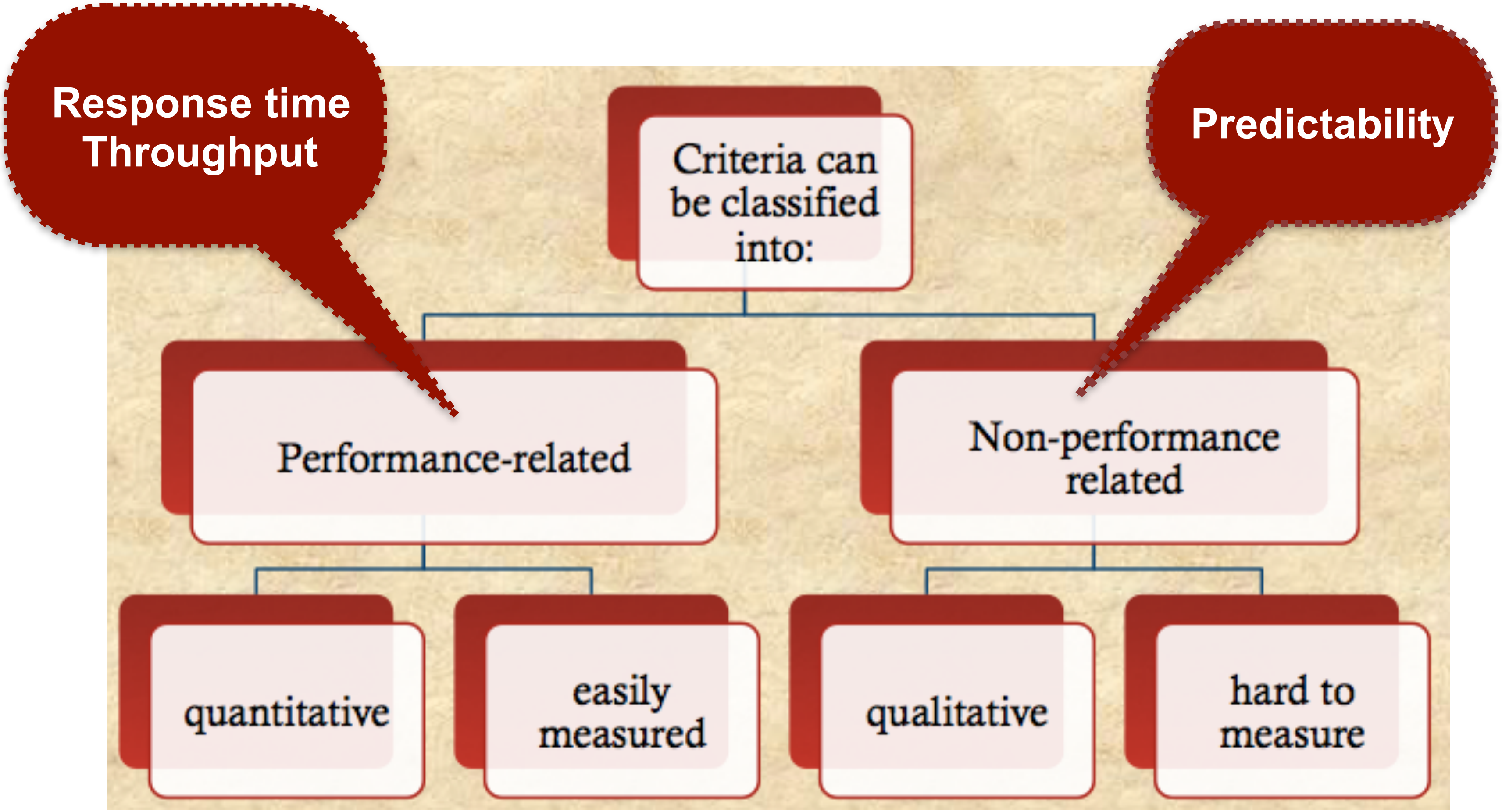
User-oriented Criteria

- Relate to the behaviour of the system as perceived by the individual user or process (e.g. **response time** in an interactive system)
- Important on virtually all systems

System-oriented Criteria

- Focus in on effective and efficient utilisation of the processor — rate at which processes are completed (i.e. **throughput**)
- Generally of minor importance on single-user systems

Short-Term Scheduling Criteria: Performance



Scheduling Criteria: Summary

User Oriented, Performance Related

Turnaround time This is the interval of time between the submission of a process and its completion. Includes actual execution time plus time spent waiting for resources, including the processor. This is an appropriate measure for a batch job.

Response time For an interactive process, this is the time from the submission of a request until the response begins to be received. Often a process can begin producing some output to the user while continuing to process the request. Thus, this is a better measure than turnaround time from the user's point of view. The scheduling discipline should attempt to achieve low response time and to maximize the number of interactive users receiving acceptable response time.

Deadlines When process completion deadlines can be specified, the scheduling discipline should subordinate other goals to that of maximizing the percentage of deadlines met.

User Oriented, Other

Predictability A given job should run in about the same amount of time and at about the same cost regardless of the load on the system. A wide variation in response time or turnaround time is distracting to users. It may signal a wide swing in system workloads or the need for system tuning to cure instabilities.

System Oriented, Performance Related

Throughput The scheduling policy should attempt to maximize the number of processes completed per unit of time. This is a measure of how much work is being performed. This clearly depends on the average length of a process but is also influenced by the scheduling policy, which may affect utilization.

Processor utilization This is the percentage of time that the processor is busy. For an expensive shared system, this is a significant criterion. In single-user systems and in some other systems, such as real-time systems, this criterion is less important than some of the others.

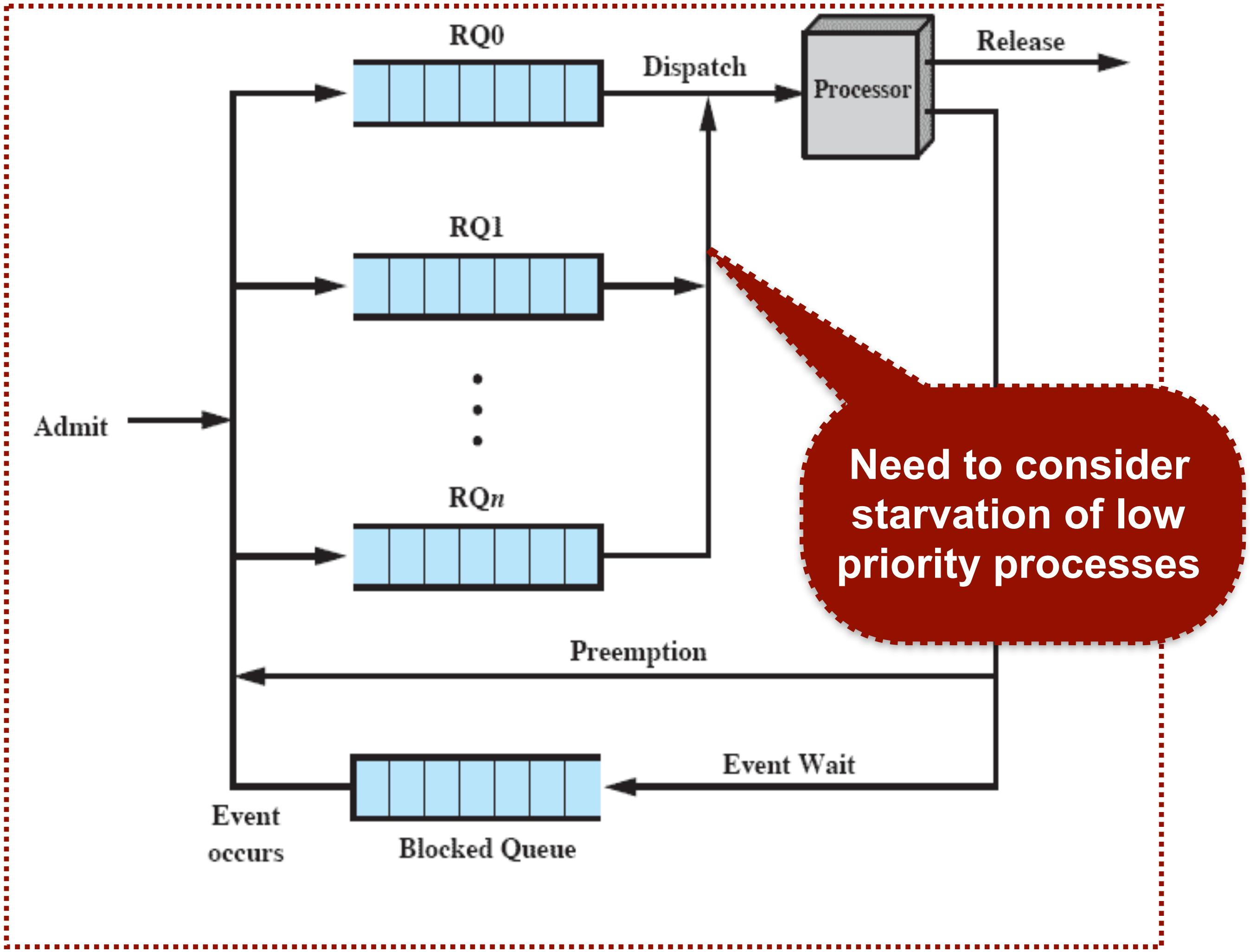
System Oriented, Other

Fairness In the absence of guidance from the user or other system-supplied guidance, processes should be treated the same, and no process should suffer starvation.

Enforcing priorities When processes are assigned priorities, the scheduling policy should favor higher-priority processes.

Balancing resources The scheduling policy should keep the resources of the system busy. Processes that will underutilize stressed resources should be favored. This criterion also involves medium-term and long-term scheduling.

Priority Queueing



What are the different scheduling policies?

Scheduling Policies

- **FCFS** First Come First Served (**FIFO** First-In First-Out)
- **RR** Round Robin
- **SPN** Shortest Process Next (**SJF** Short Job First)
- **SRT** Shortest Remaining Time
- **HRRN** Highest Response Ratio Next
- Feedback Scheduling

Selection Function

- Determines which process, among ready processes, is selected next for execution
- May be based on priority, resource requirements, or the execution characteristics of the process
- If based on **execution characteristics** then important quantities are:
  - **`w`** = time spent in system so far, waiting
  - **`e`** = time spent in execution so far
  - **`s`** = total service time required by the process, including **`e`**

Must be estimated or supplied by the user

Decision Mode

- Specifies the instants in time at which the selection function is exercise

- Two categories
  - **Non-preemptive**
  - **Preemptive**

Non-Preemptive vs Preemptive

Non-Preemptive

- Once a process is in the running state, it will continue until it terminates or blocks itself for I/O

Preemptive

- Currently running process may be *interrupted* and moved to Ready state by the OS
- Preemption may occur — when new process arrives, on an interrupt, or periodically (time-out)

Process Scheduling: Example

Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

First-Come-First-Served: FCFS

- Simplest scheduling policy
- Also known as first-in-first-out (FIFO) or a strict queuing scheme
- When the current process ceases to execute, the longest process in the Ready queue is selected

Process that has waited for long — choose that process that has the largest or max `w`

- Performs much better for long processes than short ones

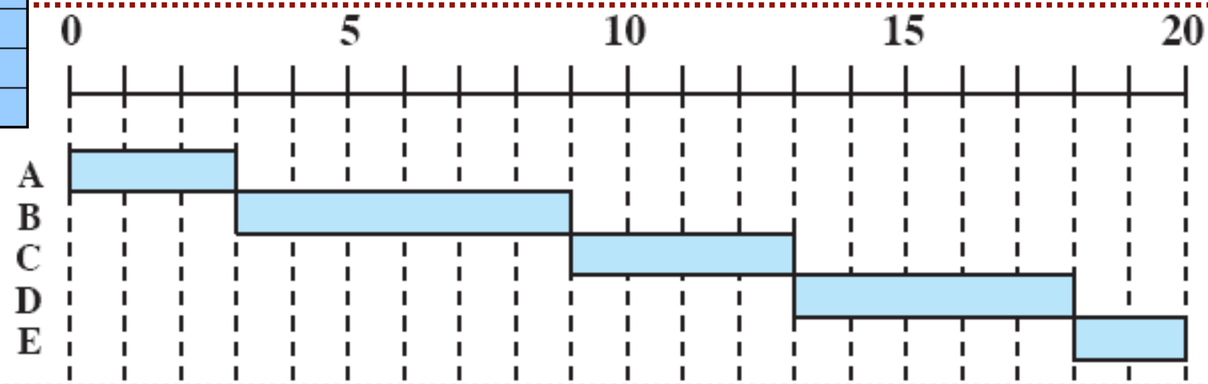
Due to non-preemptive

- Tends to favour processor-bound processes over I/O-bound processes

First-Come-First-Served: FCFS

Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

First-Come-First Served (FCFS)



FCFS						
Finish Time	A = 3	B = 9	C = 13	D = 18	E = 20	Mean
Turnaround Time (*Tr*) = Completion_time – Arrival_time	A = 3-0=3	B= 9-2=7	C= 13-4=9	D= 18-6=12	E= 20-8=12	8.60
ServiceTime (Ts)	3	6	4	5	2	4
Turnaround Time / ServiceTime = *Tr/ Ts*	1.00	1.17	2.25	2.40	6.00	2.56

Round Robin: RR

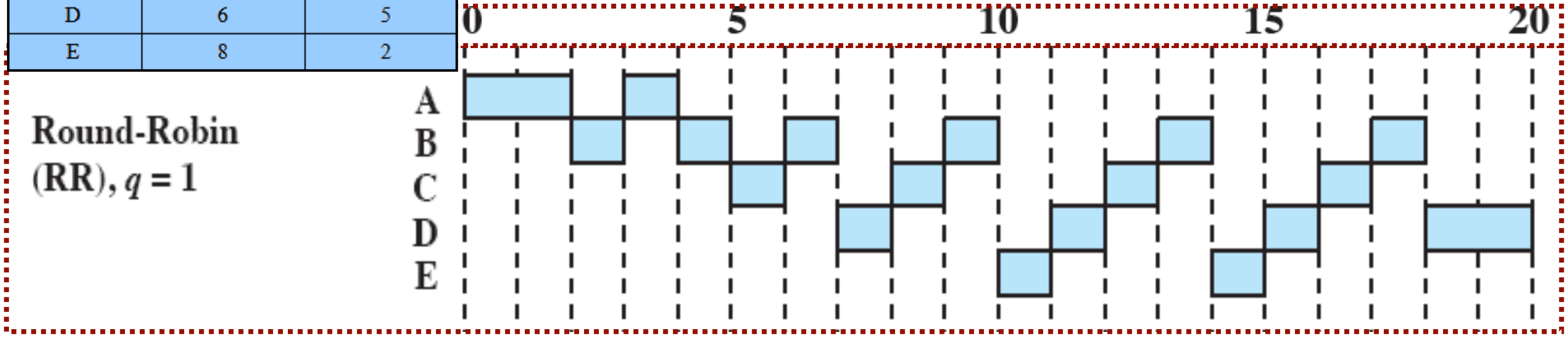
- Uses **preemption** based on a clock
- Also known as **time slicing** — each process is given a slice of time before being preempted
- Principal design issue is the length of the time quantum, or slice, to be used?

Time quantum should be slightly greater than the time required for a typical interaction or process function

- Particularly effective in a general-purpose time-sharing system or transaction processing system
- Drawback: its relative treatment of processor-bound and I/O-bound processes

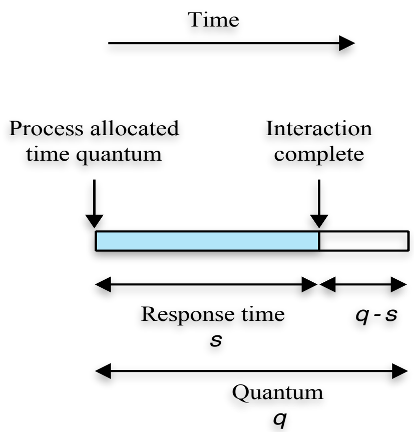
Round Robin: RR

Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

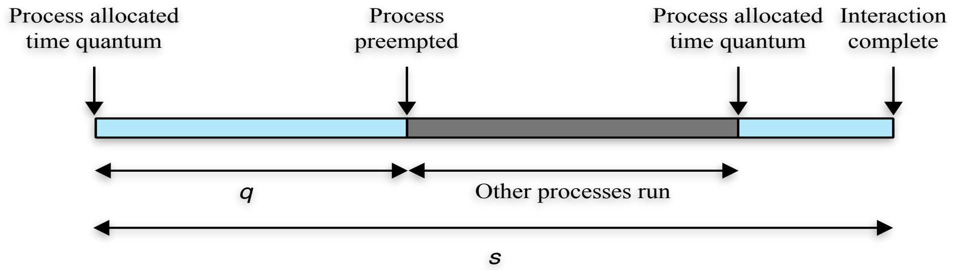


RR $q = 1$ (quantum q)						
Finish Time	4	18	17	20	15	Mean
Turnaround Time (T_r)	4	16	13	14	7	10.80
T_r/T_s	1.33	2.67	3.25	2.80	3.50	2.71
RR $q = 4$						
Finish Time	3	17	11	20	19	Mean
Turnaround Time (T_r)	3	15	7	14	11	10.00
T_r/T_s	1.00	2.5	1.75	2.80	5.50	2.71

Preemption Time Quantum: Effect

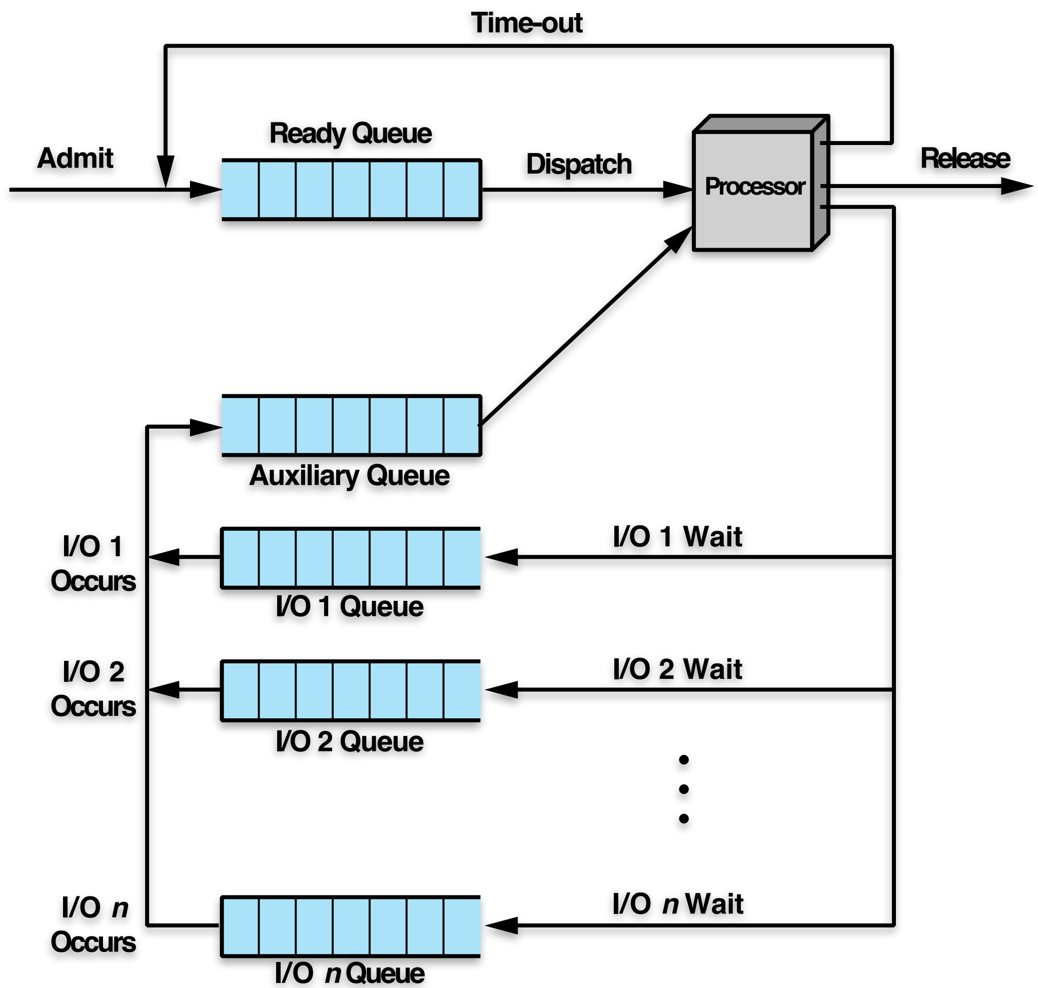


(a) Time quantum greater than typical interaction



(b) Time quantum less than typical interaction

Virtual Round Robin: VRR



Shortest Process Next: SPN

- **Non-preemptive** policy in which the process with the shortest expected processing time is selected next
- A short process will jump to the head of the queue
- Possibility of **starvation** for longer processes

- Difficulty: need to know, or at least estimate, the required processing time of each process
- If the programmer's estimate is substantially under the actual running time — the system may abort the job

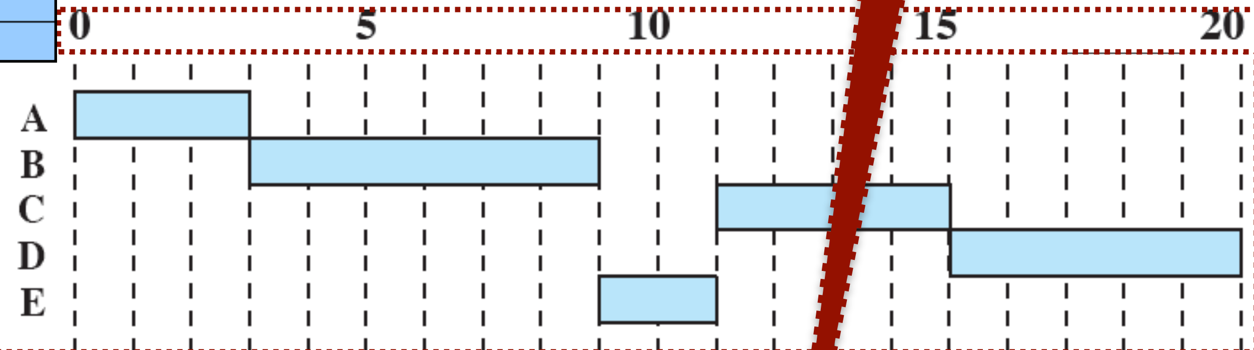
When there is a steady supply of short jobs

Shortest Process Next: SPN

1.50 compared to 6.0 for FCFS

Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

Shortest Process Next (SPN)



SPN						
Finish Time	3	9	15	20	11	Mean
Turnaround Time (*Tr*)	3	7	11	14	3	7.60
Tr/ *Ts*	1.00	1.17	2.75	2.80	1.50	1.84

Shortest Remaining Time: SRT

- **Preemptive version** of SPN (Shortest Process Next)
- Scheduler always chooses the process that has the shortest expected remaining processing time
- Risk of starvation of longer processes

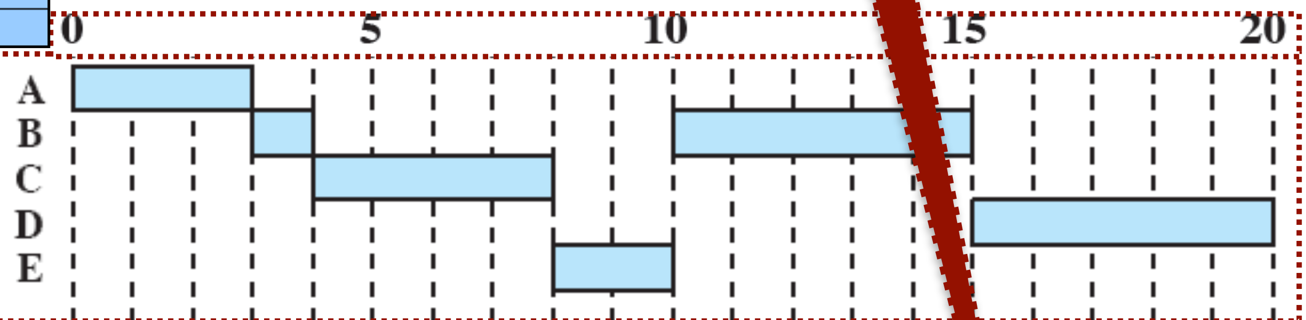
- Should give superior turnaround time performance to SPN — because a short job is given immediate preference to a running longer job

Shortest Remaining Time: SRT

Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

1.59 compared to 1.84 for SPN

Shortest Remaining Time (SRT)



SRT						
Finish Time	3	15	8	20	10	Mean
Turnaround Time (*Tr*)	3	13	4	14	2	7.20
Tr/ *Ts*	1.00	2.17	1.00	2.80	1.00	1.59

Highest Response Ratio Next: HRRN

non-preemptive

- Chooses next process with the greatest ratio
- Attractive because it accounts for the age of the process

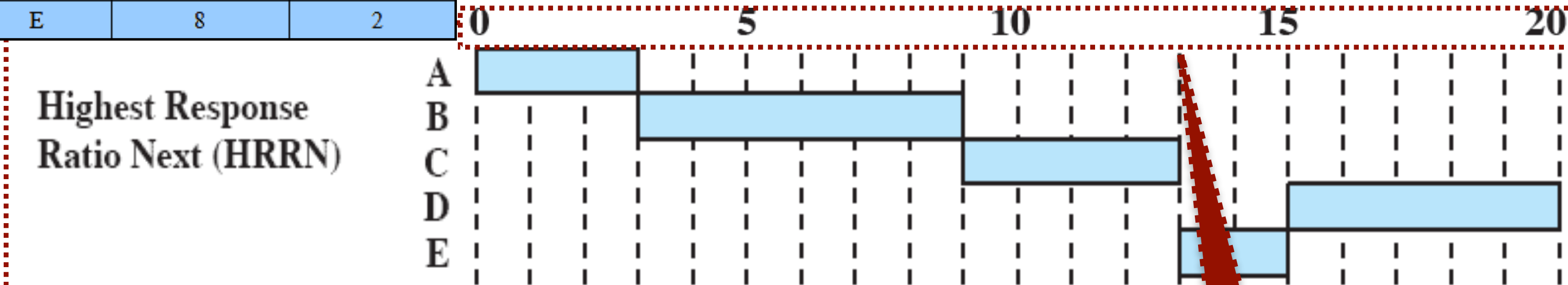
- While shorter jobs are favoured, ageing without service increases the ratio so that a longer process will eventually get past competing shorter jobs

$$Ratio = \frac{time\ spent\ waiting + expected\ service\ time}{expected\ service\ time}$$ **= max(w+s)/s**

Highest Response Ratio Next: HRRN

Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

$$Ratio = \frac{time\ spent\ waiting + expected\ service\ time}{expected\ service\ time}$$

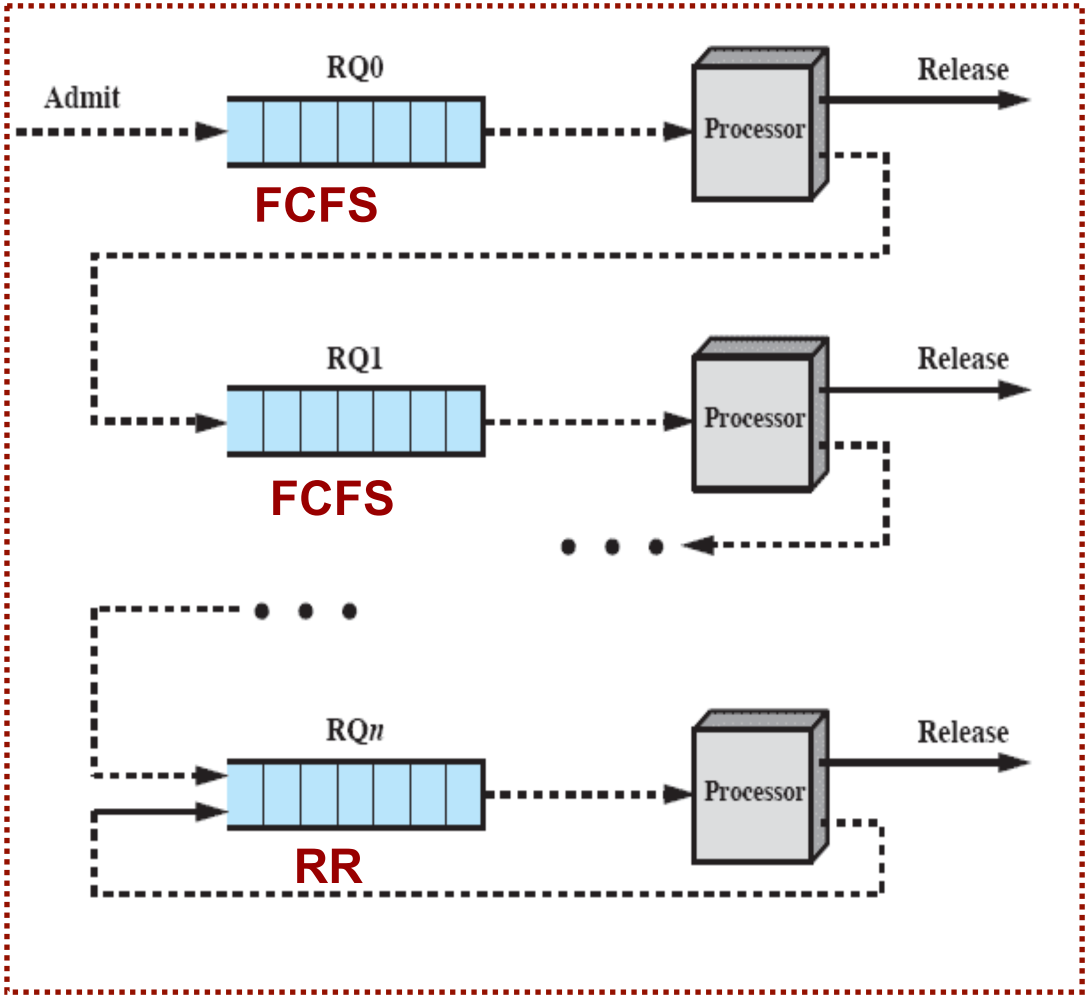


At t = 13, for D: (w+s)/s = [(13-6)+5]/5 = 2.4; for E = [(13-8)+2]/2 = 3.5

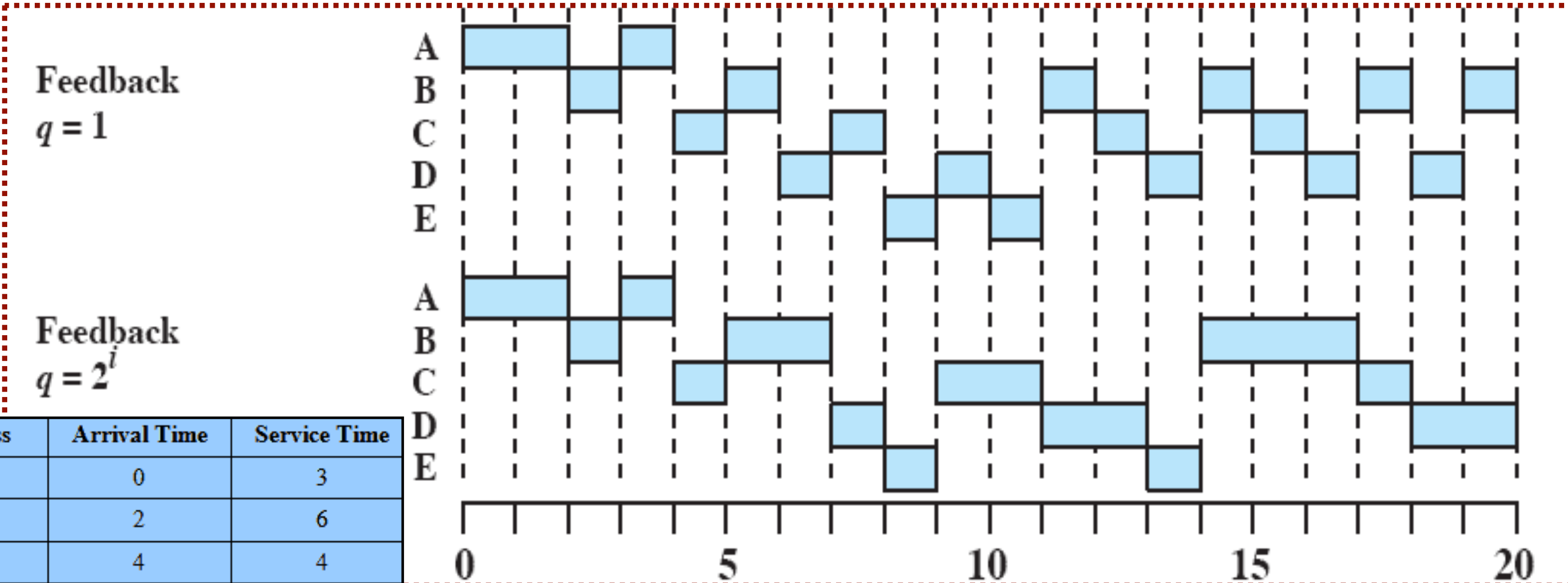
HRRN						
Finish Time	3	9	13	20	15	Mean
Turnaround Time (*Tr*)	3	7	9	14	7	8.00
Tr/ *Ts*	1.00	1.17	2.25	2.80	3.5	2.14

Feedback Scheduling

- Scheduling is done on a preemptive basis
- Dynamic priority mechanism is used
- Focus on the time spend on CPU rather than predicting how much CPU time a process will use



Feedback Performance

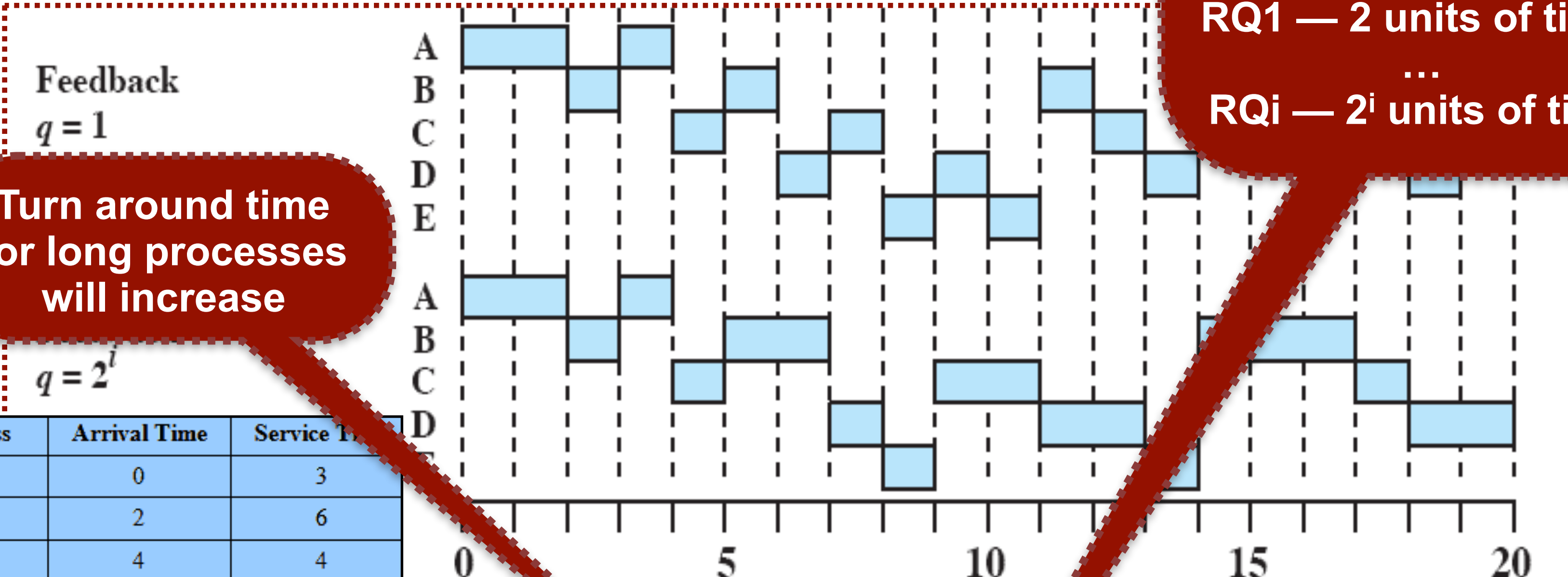


Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

FB $q = 1$						
Finish Time	4	20	16	19	11	Mean
Turnaround Time (*Tr*)	4	18	12	13	3	10.00
Tr/ *Ts*	1.33	3.00	3.00	2.60	1.5	2.29
FB $q = 2^i$						
Finish Time	4	17	18	20	14	Mean
Turnaround Time (*Tr*)	4	15	14	14	6	10.60
Tr/ *Ts*	1.33	2.50	3.50	2.80	3.00	2.63

Feedback Performance

RQ0 — 1 unit of time,
RQ1 — 2 units of time,
…
RQi — 2^i units of time



Turn around time for long processes will increase

Process	Arrival Time	Service Time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

FB $q = 1$						
Finish Time	4	20	16	19	11	Mean
Turnaround Time (Tr)	4	18	12	13	3	10.00
Tr/Ts	1.33	3.00	3.00	2.60	1.5	2.29
FB $q = 2^i$						
Finish Time	4	17	18	20	14	Mean
Turnaround Time (Tr)	4	15	14	14	6	10.60
Tr/Ts	1.33	2.50	3.50	2.80	3.00	2.63

Scheduling Policies: Comparison (1)

Process	A	B	C	D	E	
Arrival Time	0	2	4	6	8	
Service Time (T_s)	3	6	4	5	2	Mean
FCFS						
Finish Time	3	9	13	18	20	
Turnaround Time (T_r)	3	7	9	12	12	8.60
T_r/T_s	1.00	1.17	2.25	2.40	6.00	2.56
RR $q = 1$						
Finish Time	4	18	17	20	15	
Turnaround Time (T_r)	4	16	13	14	7	10.80
T_r/T_s	1.33	2.67	3.25	2.80	3.50	2.71
RR $q = 4$						
Finish Time	3	17	11	20	19	
Turnaround Time (T_r)	3	15	7	14	11	10.00
T_r/T_s	1.00	2.5	1.75	2.80	5.50	2.71
SPN						
Finish Time	3	9	15	20	11	
Turnaround Time (T_r)	3	7	11	14	3	7.60
T_r/T_s	1.00	1.17	2.75	2.80	1.50	1.84

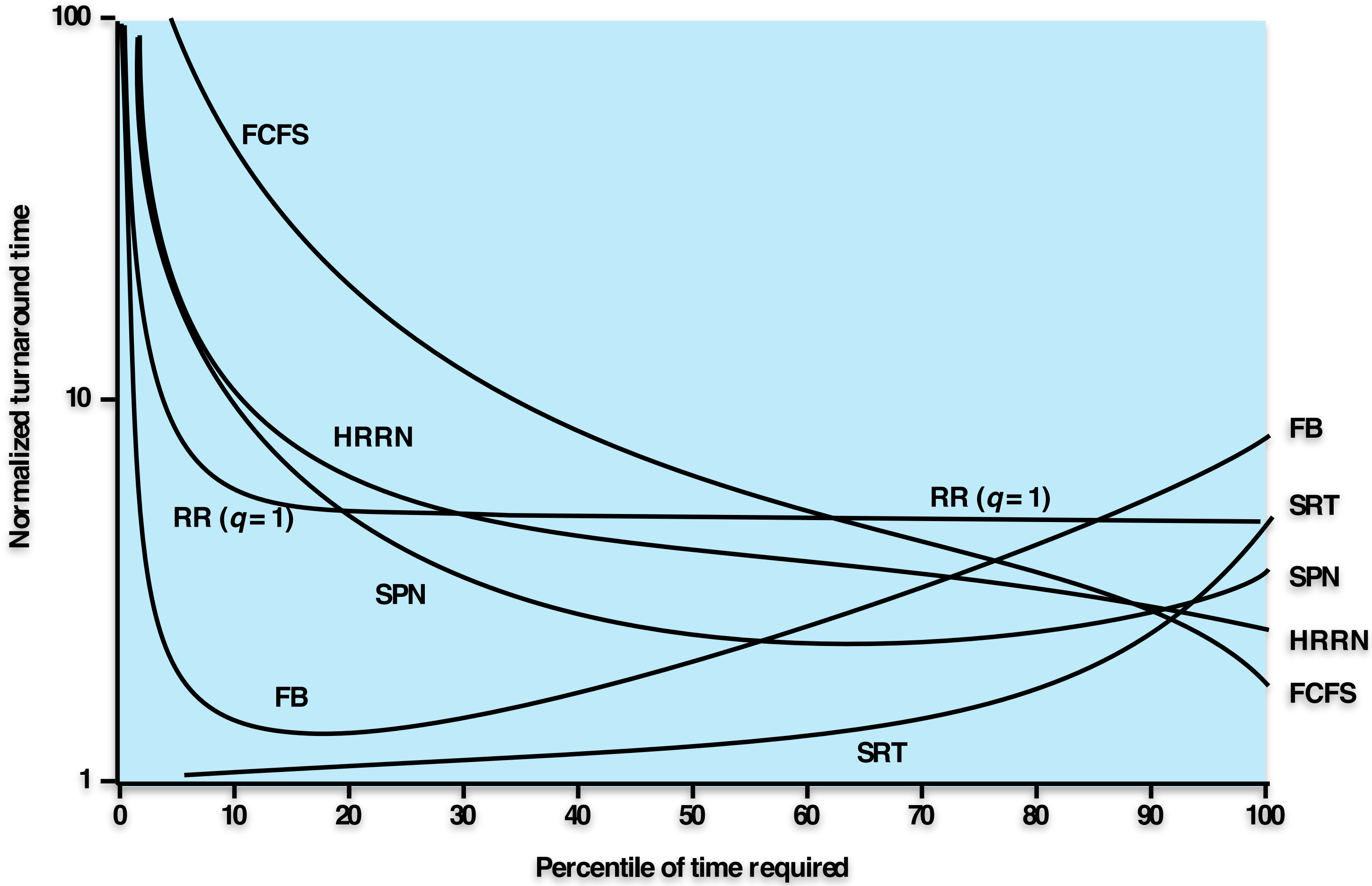
Scheduling Policies: Comparison (2)

SRT						
Finish Time	3	15	8	20	10	
Turnaround Time (T_r)	3	13	4	14	2	7.20
T_r/T_s	1.00	2.17	1.00	2.80	1.00	1.59
HRRN						
Finish Time	3	9	13	20	15	
Turnaround Time (T_r)	3	7	9	14	7	8.00
T_r/T_s	1.00	1.17	2.25	2.80	3.5	2.14
FB $q = 1$						
Finish Time	4	20	16	19	11	
Turnaround Time (T_r)	4	18	12	13	3	10.00
T_r/T_s	1.33	3.00	3.00	2.60	1.5	2.29
FB $q = 2^i$						
Finish Time	4	17	18	20	14	
Turnaround Time (T_r)	4	15	14	14	6	10.60
T_r/T_s	1.33	2.50	3.50	2.80	3.00	2.63

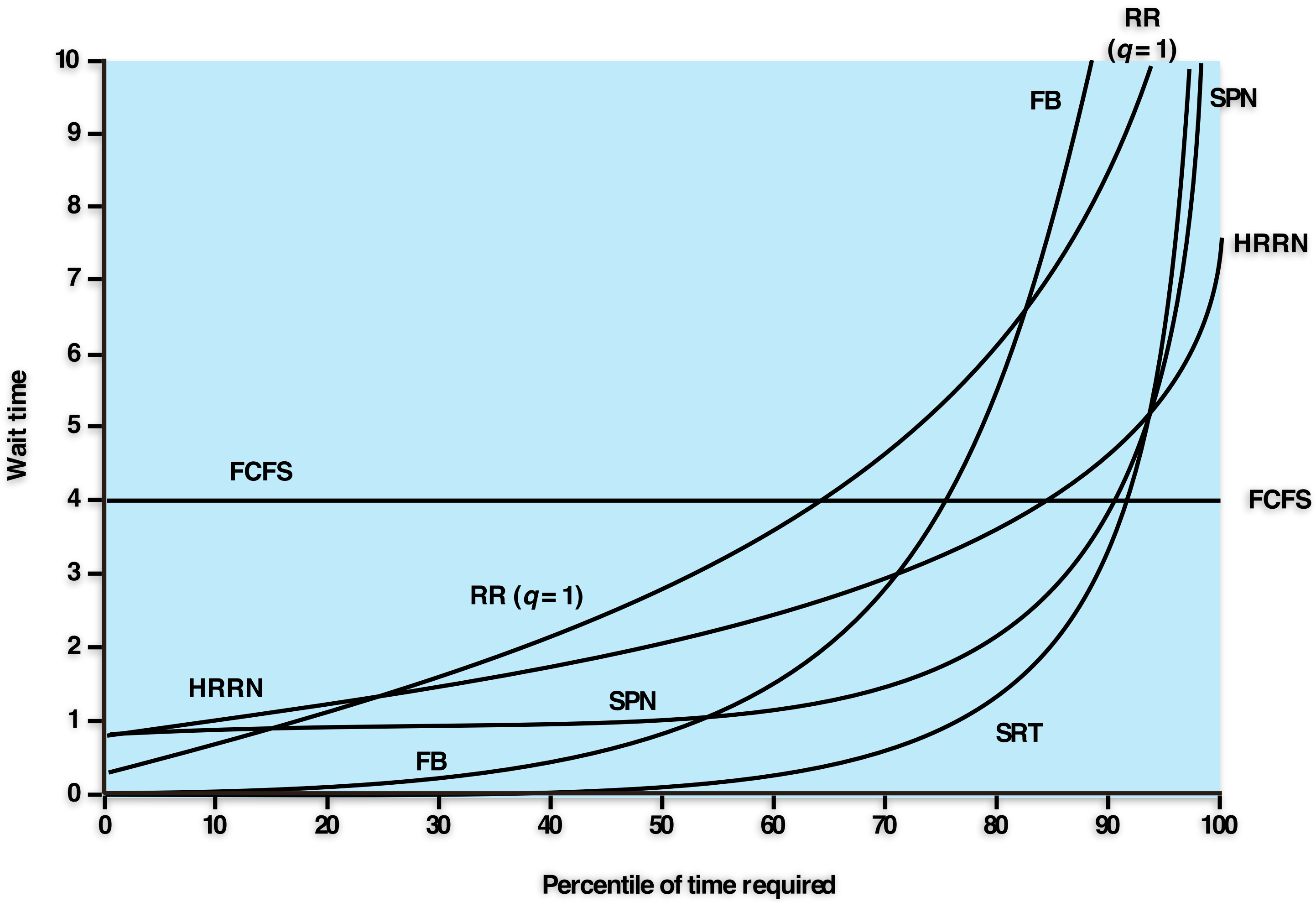
Scheduling Policies: Characteristics

	FCFS	Round robin	SPN	SRT	HRRN	Feedback
Selection function	max[w]	constant	min[s]	min[s – e]	$\max\left(\frac{w+s}{s}\right)$	(see text)
Decision mode	Non-preemptive	Preemptive (at time quantum)	Non-preemptive	Preemptive (at arrival)	Non-preemptive	Preemptive (at time quantum)
Throughput	Not emphasized	May be low if quantum is too small	High	High	High	Not emphasized
Response time	May be high, especially if there is a large variance in process execution times	Provides good response time for short processes	Provides good response time for short processes	Provides good response time	Provides good response time	Not emphasized
Overhead	Minimum	Minimum	Can be high	Can be high	Can be high	Can be high
Effect on processes	Penalizes short processes; penalizes I/O bound processes	Fair treatment	Penalizes long processes	Penalizes long processes	Good balance	May favor I/O bound processes
Starvation	No	No	Possible	Possible	No	Possible

Simulation Results: Normalised Turnaround Time



Simulation Results: Waiting Time



Fair Share Scheduling: FSS

- Scheduling decisions are based on the **process sets**

- Each user is assigned a **share** of the processor

- Objective*:* to monitor usage to give fewer resources to users who have had more than their fair share and more to those who have had less than their fair share

scheduling decisions are based on execution history and priority basis

Fair Share Scheduling: FSS

$$CPU_j(i) = \frac{CPU_j(i-1)}{2}$$

$$GCPU_k(i) = \frac{GCPU_k(i-1)}{2}$$

$$P_j(i) = Base_j + \frac{CPU_j(i)}{2} + \frac{GCPU_k(i)}{4 \times W_k}$$

where

$CPU_j(i)$ = measure of processor utilization by process j through interval i

$GCPU_k(i)$ = measure of processor utilization of group k through interval i

$P_j(i)$ = priority of process j at beginning of interval i; lower values equal higher priorities

$Base_j$ = base priority of process j

W_k = weighting assigned to group k, with the constraint that $0 < W_k \leq 1$ and $\sum_k W_k = 1$

Fair Share Scheduling: Three Processes

Time	Process A Priority	Process A Process CPU count	Process A Group CPU count	Process B Priority	Process B Process CPU count	Process B Group CPU count	Process C Priority	Process C Process CPU count	Process C Group CPU count
0	60	0 1 2 • • 60	0 1 2 • • 60	60	0	0	60	0	0
1	90	30	30	60	0 1 2 • • 60	0 1 2 • • 60	60	0	0 1 2 • • 60
2	74	15 16 17 • • 75	15 16 17 • • 75	90	30	30	75	0	30
3	96	37	37	74	15	15 16 17 • • 75	67	0 1 2 • • 60	15 16 17 • • 75
4	78	18 19 20 • • 78	18 19 20 • • 78	81	7	37	93	30	37
5	98	39	39	70	3	18	76	15	18

Group 1 (Process A) — Group 2 (Process B, Process C)

Colored rectangle represents executing process

priority recalculated once per second

Unix Scheduling

Unix Scheduling

- Primarily targeted at the time-sharing interactive environment
- Designed to provide good response time for interactive users while ensuring that low-priority background jobs do not starve
- Employs **multilevel feedback using round robin** within each of the priority queues
- Makes use of one-second preemption
- Priority is based on process type and execution history

Unix Scheduling: Formula

$$CPU_j(i) = \frac{CPU_j(i-1)}{2}$$

$$P_j(i) = Base_j + \frac{CPU_j(i)}{2} + nice_j$$

where

$CPU_j(i)$ = measure of processor utilization by process j through interval i

$P_j(i)$ = priority of process j at beginning of interval i; lower values equal higher priorities

$Base_j$ = base priority of process j

$nice_j$ = user-controllable adjustment factor

priority recalculated once per second

Unix Scheduling: Process Bands

- Used to optimise access to block devices and to allow the operating system to respond quickly to system calls
- In decreasing order of priority, the bands are:

Unix Scheduling: Example

Time	Process A		Process B		Process C	
	Priority	CPU count	Priority	CPU count	Priority	CPU count
0	60	0 1 2 • • 60	60	0	60	0
1	75	30	60	0 1 2 • • 60	60	0
2	67	15	75	30	60	0 1 2 • • 60
3	63	7 8 9 • • 67	67	15	75	30
4	76	33	63	7 8 9 • • 67	67	15
5	68	16	76	33	63	7

Colored rectangle represents executing process

Summary of Lecture 6

- OS must make three types of scheduling decisions with respect to the execution of processes:
  - **Long-term** — determines when new processes are admitted to the system.
  - **Medium-term** — part of the swapping function and determines when a program is brought into main memory so that it may be executed.
  - **Short-term** — determines which ready process will be executed next by the processor.
- From a user's point of view, response time is generally the most important characteristic of a system; from a system point of view, throughput or processor utilisation is important.
- Scheduling algorithms: **FCFS, Round Robin, SPN, SRT, HRRN, Feedback, Fair Share**