Introduction to System Programming

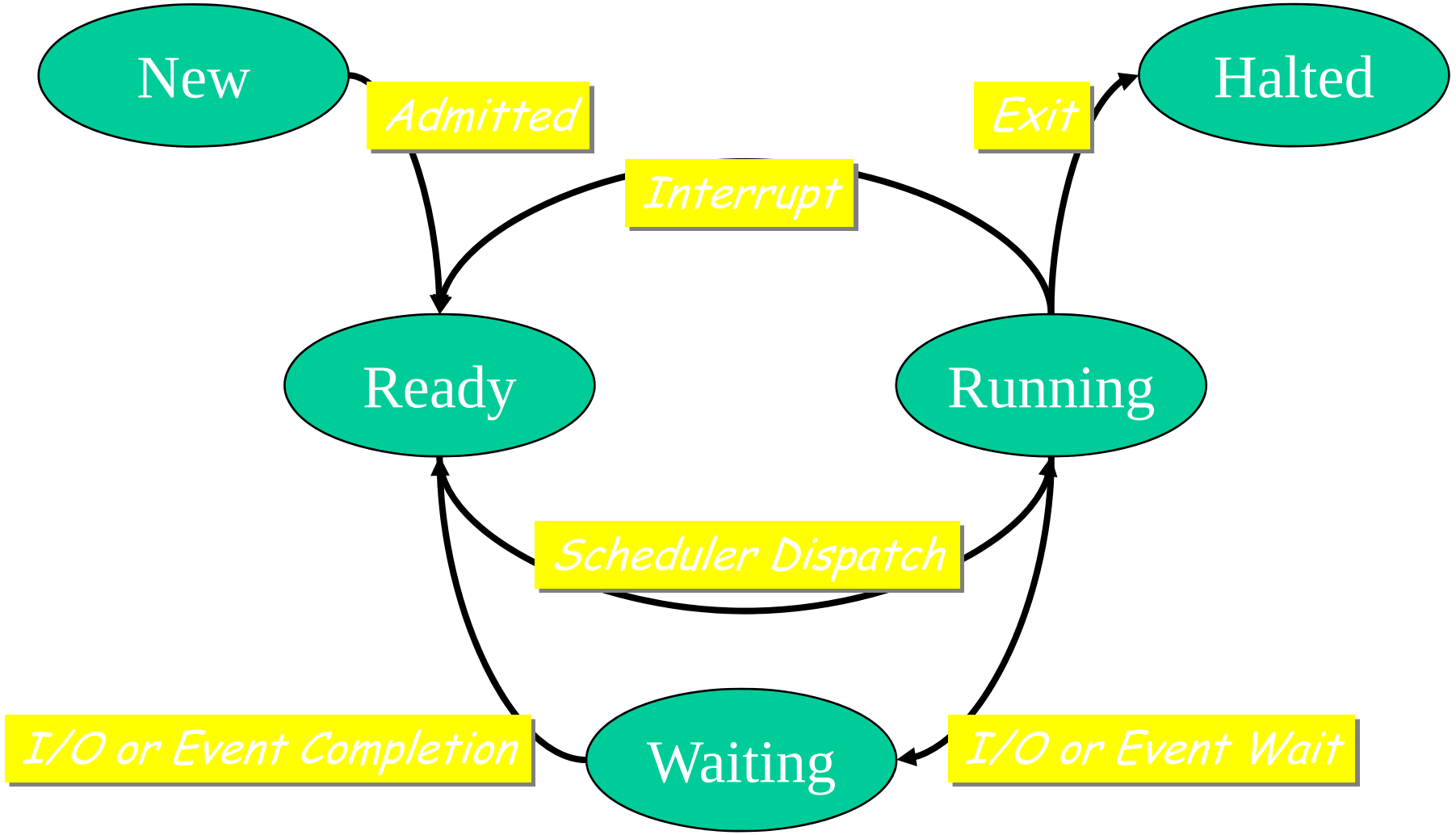
Process Scheduling

Minal Shah

Outline

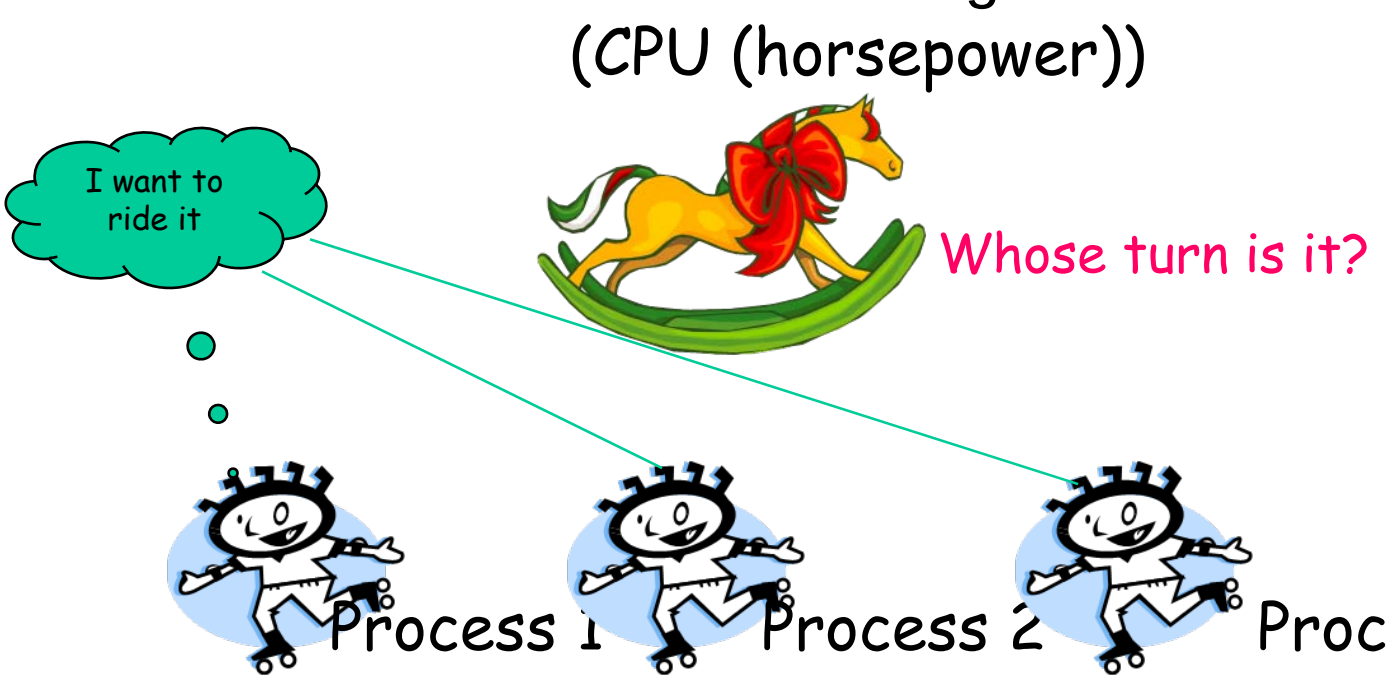
- Process Scheduling:
  - Types and Algorithms

Scheduling Environments



Scheduling

- Deciding which process/thread should occupy a resource (CPU, disk, etc.)
- An OS must allocate resources amongst competing processes.
- The resource provided by a processor is execution time
  - The resource is allocated by means of a schedule
  - means of allocation is scheduling.



Scheduling

- The aim of processor scheduling is to assign processes to be executed by the processor over time,
  - in a way that meets system objectives, such as response time, throughput, and processor efficiency.
  - Fundamentally, scheduling is a matter of managing queues to minimize queuing delay and to optimize performance in a queuing environment.
- The objective of scheduling function is
  - Share time fairly among processes
  - Prevent starvation of a process
  - Use the processor efficiently
  - Have low overhead
  - Prioritise processes when necessary (e.g. real time deadlines)

Scheduling Criteria

Criteria	Maximize	Minimize
CPU utilization – keep the CPU as busy as possible.	✓	
Throughput – # of processes that complete their execution per time unit.	✓	
Turnaround time – time to execute a process from submission to completion.		✓
Waiting time – amount of time a process has been waiting in the ready queue.		✓
Response time – time it takes from when a request was submitted until the first response is produced, not output (for time-sharing environment).		✓

When to Schedule?

- When is scheduling needed?
  - Process creation
  - Process exit
  - Process blocks
  - Higher priority process becomes ready
  - I/O Interrupt (resource becomes available)
- What are the two types of schedulers?
  - Non-preemptive
    - Execute until yield, exit, or block.
  - Preemptive
    - Highest priority process ALWAYS executing
    - Reschedule immediately on events (interrupt if necessary).

Scheduling Goals

All Systems

- Fairness
- Policy Enforcement
- Balance

Batch Systems

- Throughput
- Turnaround time
- CPU utilization

Interactive Systems

- Response time
- Proportionality
- User experience

Real-time Systems

- Meeting deadlines
- Predictability
- Stability

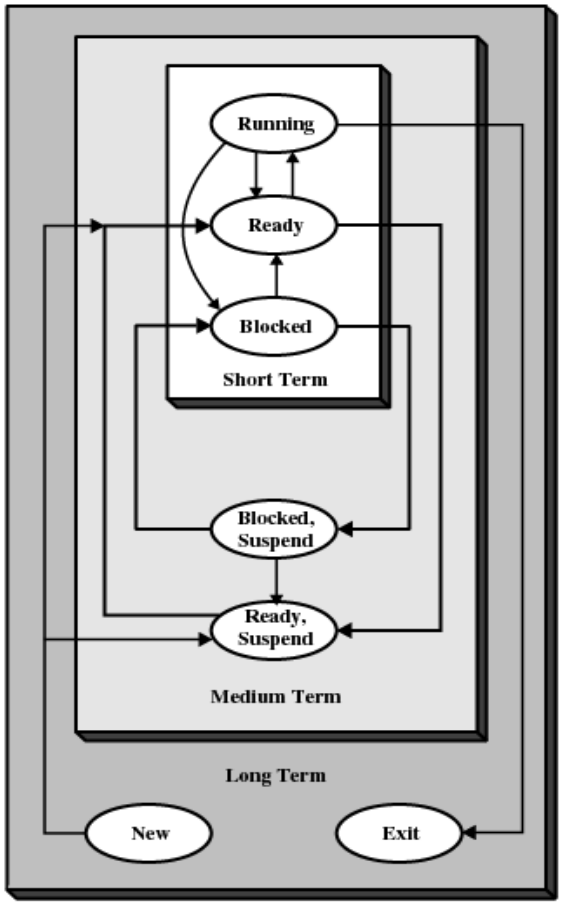


Figure 9.2 Levels of Scheduling

Interdependent Scheduling Criteria

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.

Interdependent Scheduling Criteria cont.

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.

Priorities

- Scheduler will always choose a process of higher priority over one of lower priority
- Have multiple ready queues to represent each level of priority

Starvation

- Problem:
  - Lower-priority may suffer starvation if there is a steady supply of high priority processes.
- Solution
  - Allow a process to change its priority based on its age or execution history

Preemptive vs. Non-preemptive

- Scheduling that only takes place due to I/O or process termination is non-preemptive.
- Preemptive scheduling allows the operating system to interrupt the currently running process and move it to the ready state.
  - new process arrives
  - clock interrupt
- Preemptive scheduling:
  - incurs greater overhead (context switch)
  - provides better service to the total population of processes
  - may prevent one process from monopolizing the processor

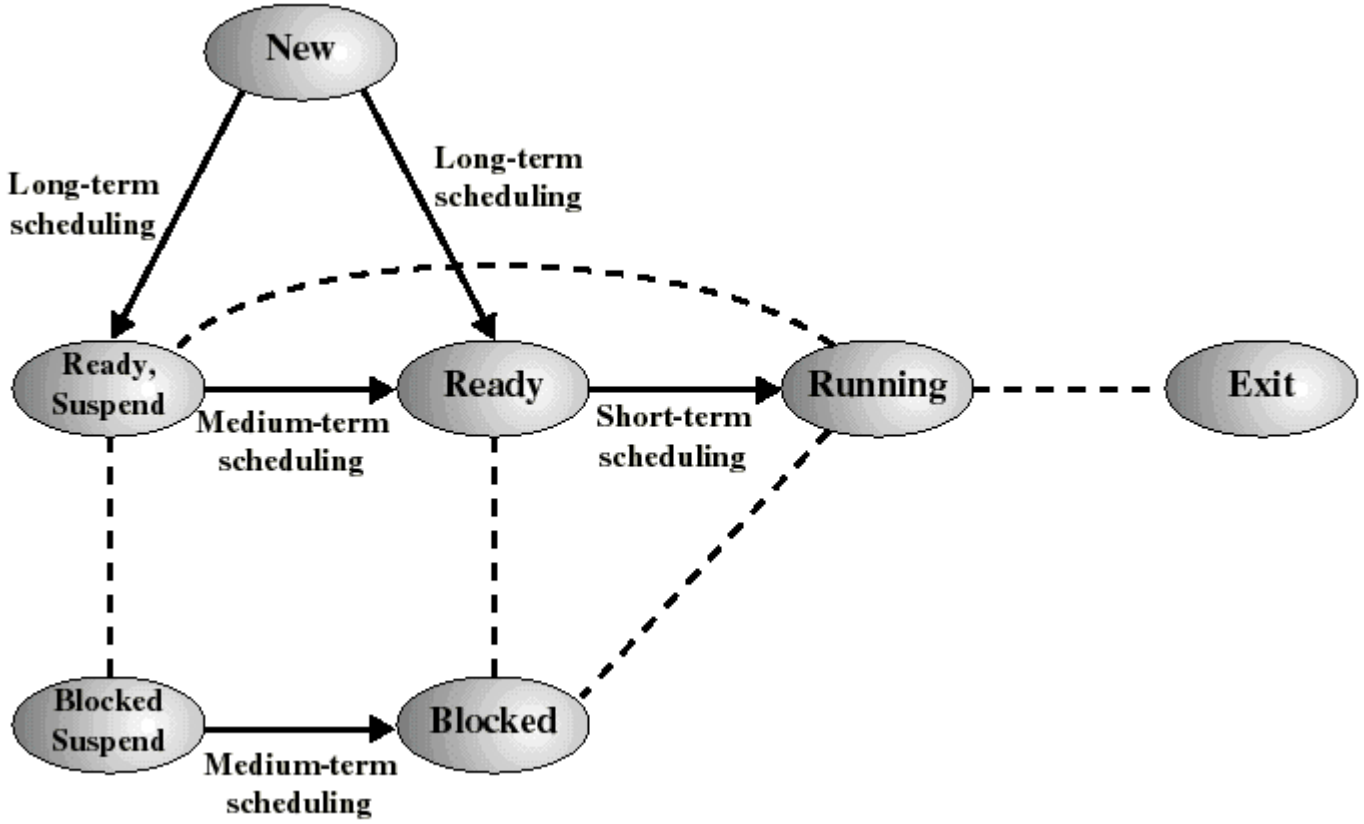
Response Time

- User productivity tends to improve with a more rapid response time.
  - Especially true for expert users
  - Becomes very noticeable as response time drops below 1 second
- User time or “think time” – Time user spends thinking about the response.
- System time – Time system takes to generate its response.
  - Short response times are important
  - User response time tends to decrease as system response time decreases
  - If the system gets too slow to a user, they may slow down or abort the operation
- Turnaround time (TAT) – total time that an item spends in the system.

Types of Scheduling

- **Long-term scheduling** is performed when a new process is created.
  - This is a decision whether to add a new process to the set of processes that are currently active.
- **Medium-term scheduling** is a part of the swapping function.
  - This is a decision whether to add a process to those that are at least partially in main memory and therefore available for execution.
- **Short-term scheduling** is the actual decision of which ready process to execute next.
- **I/O scheduling** is the decision as to which process's pending I/O request shall be handled by the available I/O device.

Classification of Scheduling Activity



- Long-term: which process to admit
- Medium-term: which process to swap in or out
- Short-term: which ready process to execute next

Long-Term Scheduling

- Determines which programs are admitted to the system for processing
  - May be first-come-first-served
  - Or according to criteria such as priority, I/O requirements or expected execution time
- Controls the degree of multiprogramming
- More processes, smaller percentage of time each process is executed

Medium-Term Scheduling

- The medium-term scheduler is executed somewhat more frequently.
- Part of the swapping function
- Swapping-in decisions are based on the need to manage the degree of multiprogramming
- On a system that does not use virtual memory, memory management is also an issue.
- Thus, the swapping-in decision will consider the memory requirements of the swapped-out processes.

Short-Term Scheduling

- Known as the dispatcher executes most frequently and makes the fine-grained decision of which process to execute next.
- Invoked when an event occurs that may lead to the blocking of the current process or that may provide an opportunity to preempt a currently running process in favor of another.
- Examples of such events include
  - Clock interrupts
  - I/O interrupts
  - Operating system calls
  - Signals
- Main objective is to allocate processor time to optimize certain aspects of system behaviour.
- A set of criteria is needed to evaluate the scheduling policy.

Short-Term Scheduling Criteria: User vs System

- We can differentiate between user and system criteria
- User-oriented relate to the behavior of the system as perceived by the individual user or process.
  - response time in an interactive system which is the elapsed time between the submission of a request until the response begins to appear as output.
- System-oriented criteria is the focused on effective and efficient utilization of the processor
  - An example is throughput, i.e. the rate at which processes are completed. Thus, throughput is of concern to a system administrator but not to the user population.

Scheduling Algorithms

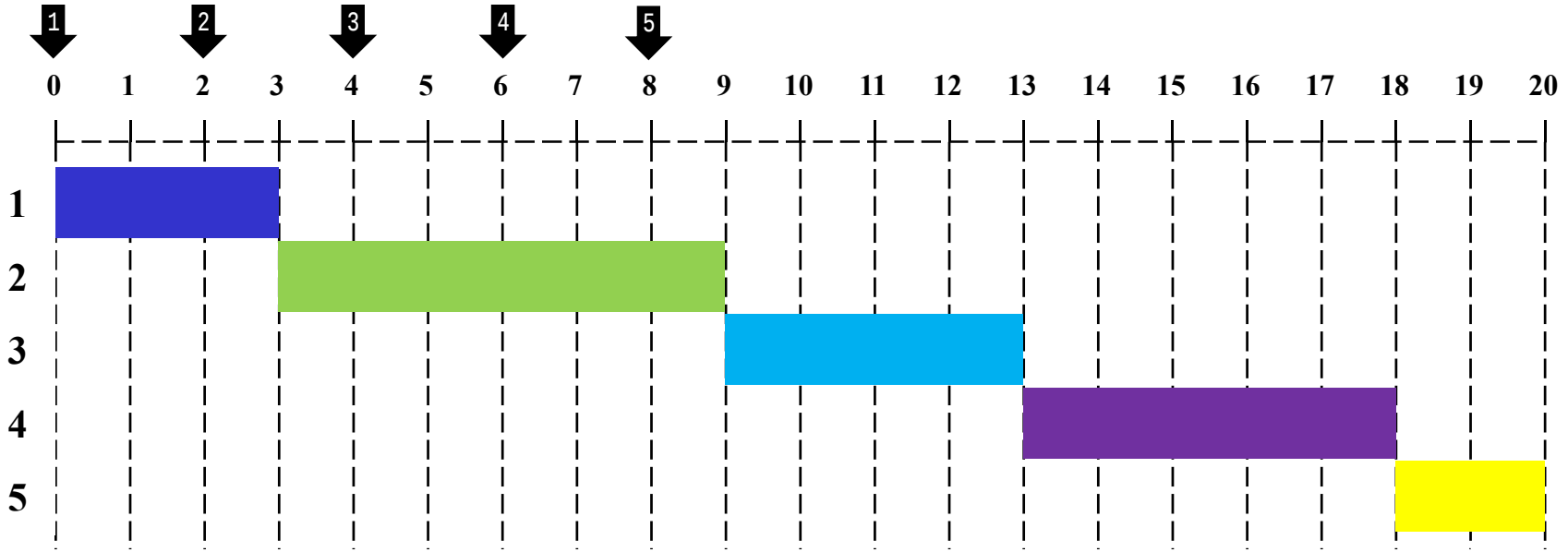
- First Come First Served (FCFS)
- Round Robin (RR) – time slicing.
- Shortest Process Next (SPN)
- Shortest Remaining Time (SRT)
- Highest Response Ratio Next (HRRN)
- Feedback

First-Come-First-Served (FCFS)

- Simplest scheduling algorithm - process joins the Ready queue upon arrival, oldest process in the Ready queue is selected (non-preemptive).
- A short process may have to wait a very long time before it can execute, thus favors CPU-bound processes over I/O-bound processes - I/O processes have to wait until CPU-bound process completes.
- FCFS, although not attractive alternative for a uni-processor system, when combined with a priority scheme may prove to be an effective scheduler.

First-Come-First-Served (FCFS)

Process	Arrival	Service
1	0	3
2	2	6
3	4	4
4	6	5
5	8	2



First-Come-First-Served (FCFS)

Process	Arrival Time (T_0) ms	Next burst(Δ t) ms	Finish Time(T_1) ms	Turnaround Time TAT = T_1-T_0	Waiting Time WT = TAT - Δt
1	0	3	3	3	0
2	2	6	9	7	1
3	4	4	13	9	5
4	6	5	18	12	7
5	8	2	20	12	10
				43	23

- Average Turnaround time = 43/5 = 8.6
- Average waiting time = 23/5 = 4.6

(FCFS) Disadvantage:

1) Order is important.
2) Suppose one CPU bound process and many I/O bound processes.

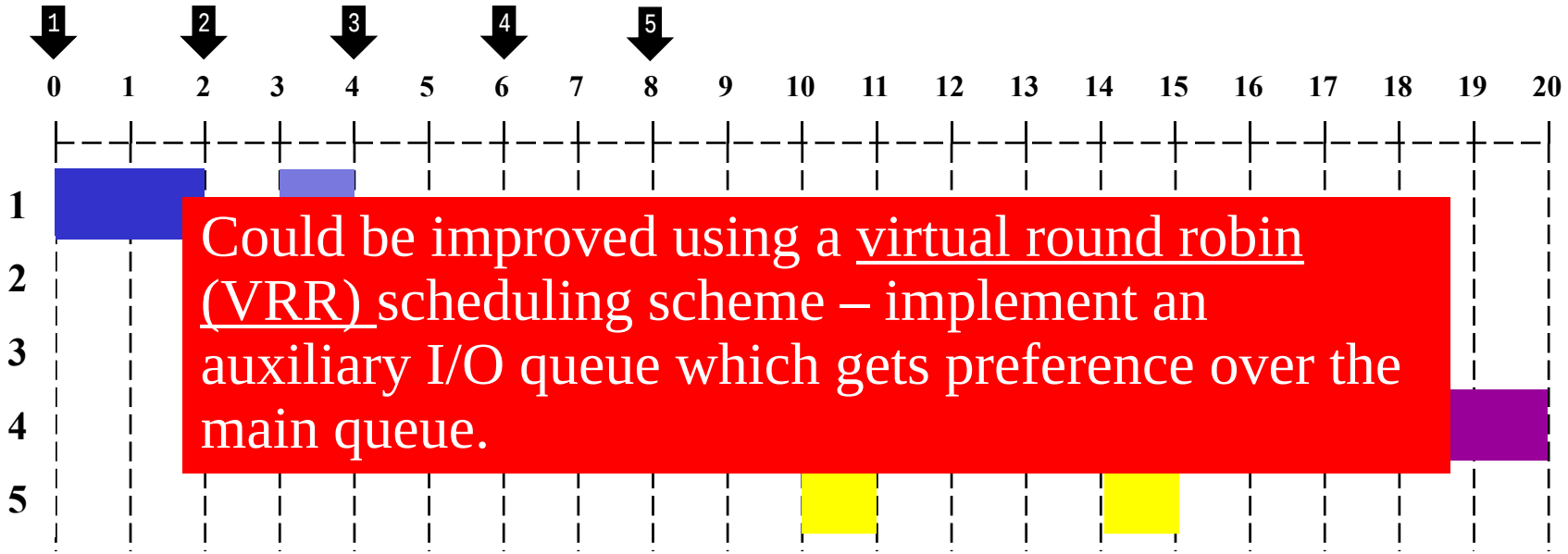
- CPU bound process will get CPU and other I/O processes will be in ready queue.
- By that time I/O is idle.
- CPU bound moves to I/O. Meanwhile all I/O processes will execute quickly and come to I/O queue. Therefore, CPU is idle

Round Robin (RR)

- Uses FCFS w/preemption - based on a time quantum coming from a clock (time slicing).
  - Short time quantum: processes move relatively quickly through the system (with more overhead).
  - The time quantum should be slightly greater than the time required for a typical interaction.
- Particularly effective in GP time-sharing or transaction processing system.
- Generally favors processor bound processes as I/O bound give up their time slice while process bound use their complete time quantum.

Round Robin (RR)

Process	Arrival	Service
1	0	3
2	2	6
3	4	4
4	6	5
5	8	2



Could be improved using a virtual round robin (VRR) scheduling scheme – implement an auxiliary I/O queue which gets preference over the main queue.

Round Robin (RR) time slice = 4ms, context switching time = 1 ms

Process	Arrival Time (T_0) ms	Next burst(Δ t) ms	Finish Time(T_1) ms	Turnaround Time TAT = T_1-T_0	Waiting Time WT = TAT - Δt
1	0	3	3	3	0
2	2	6	27	25	19
3	4	4	14	10	6
4	6	5	19	13	8
5	8	2	24	16	14
				67	47

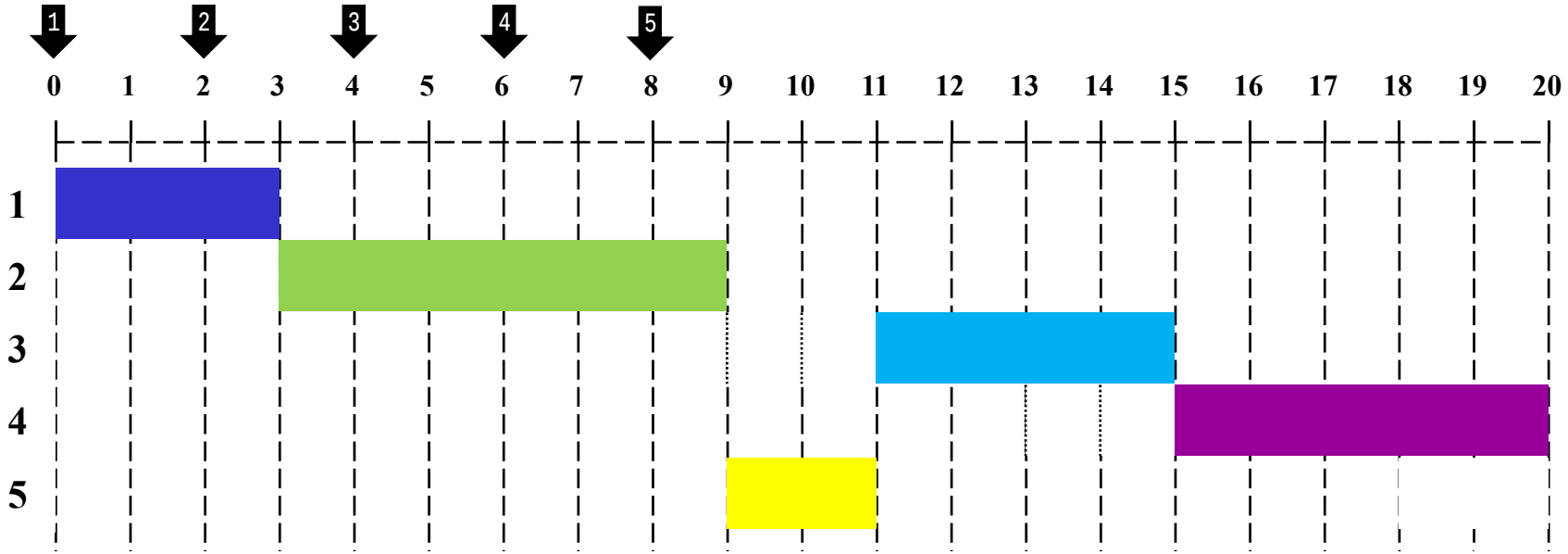
- Average Turnaround time = 67/5 = 13.4
- Average waiting time = 47/5 = 9.4

Shortest Process Next (SPN)

- Non-preemptive policy - Process with shortest expected processing time is selected next.
- Short process jumps ahead of longer processes.
- May be impossible to know or at least estimate the required processing time of a process.
  - Batch jobs require a programmer's estimate. (If estimate is substantially off, system may abort job.)
  - In a production environment, the same jobs run frequently and statistics may be gathered.
  - In an interactive environment, the operating system may keep a running average of each "burst" for each process.
- SPN could result in starvation of longer processes if there is a steady supply of short processes.
- Not suitable for time-sharing or transaction processing environment because of lack of preemption.

Shortest Process Next (SPN)

Process	Arrival	Service
1	0	3
2	2	6
3	4	4
4	6	5
5	8	2



Shortest Process/Job Next (SPN)
Shortest Process/Job First(SJF)

Process	Arrival Time (T_0) ms	Next burst(Δ t) ms	Finish Time(T_1) ms	Turnaround Time TAT = T_1-T_0	Waiting Time WT = TAT - Δt
1	0	3	3	3	0
2	2	6	9	7	1
3	4	4	15	11	7
4	6	5	20	14	9
5	8	2	11	3	1
				38	18

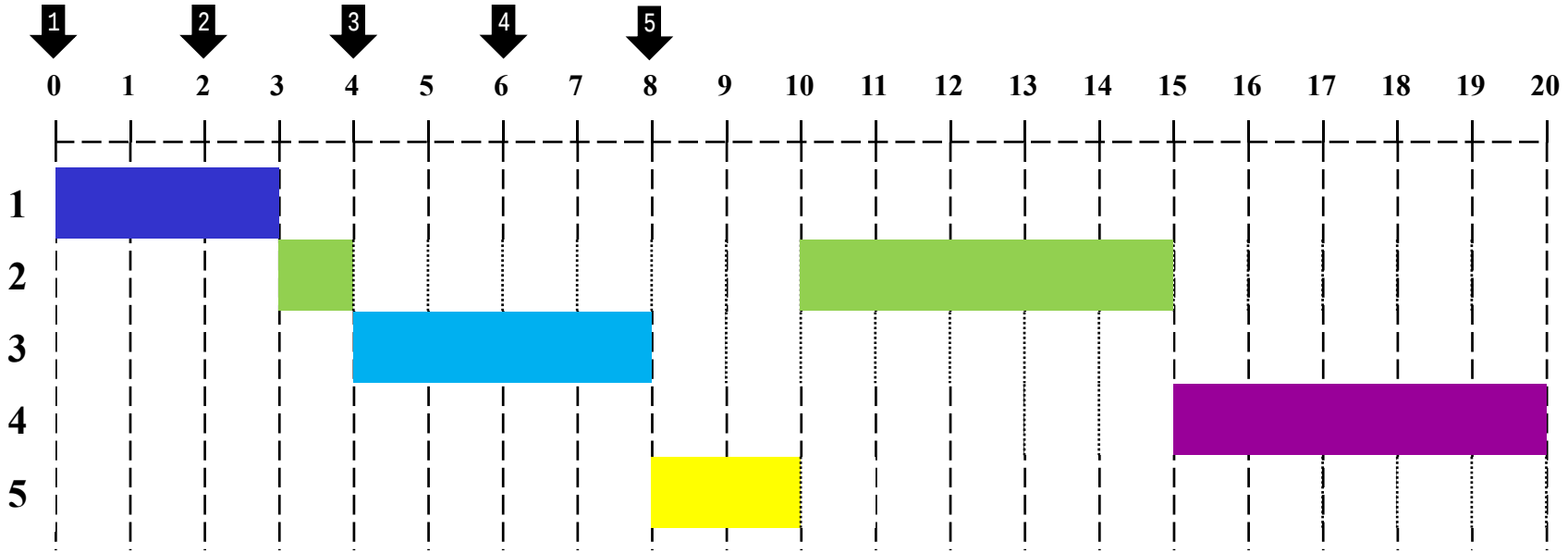
- Average Turnaround time = 38/5 = 7.6
- Average waiting time = 18/5 = 3.6

Shortest Remaining Time (SRT)

- The shortest remaining time (SRT) policy is a preemptive version of SPN.
- Must estimate expected remaining processing time
- When a new process joins the ready queue, it may have a shorter remaining time and preempts the current process.
- SRT does not bias in favor of long processes (as does FCFS)
- Unlike RR, no additional interrupts are generated reducing overhead.
- Superior turnaround performance to SPN, because a short job is given immediate preference to a running longer job.

Shortest Remaining Time (SRT)

Process	Arrival	Service
1	0	3
2	2	6
3	4	4
4	6	5
5	8	2



Shortest Remaining Time (SRT)

Process	Arrival Time (T_0) ms	Next burst(Δ t) ms	Finish Time(T_1) ms	Turnaround Time TAT = T_1-T_0	Waiting Time WT = TAT - Δt
1	0	3	3	3	0
2	2	6	15	13	7
3	4	4	8	4	0
4	6	5	20	14	9
5	8	2	10	2	0
				36	16

- Average Turnaround time = 36/5 = 7.2
- Average waiting time = 16/5 = 3.2

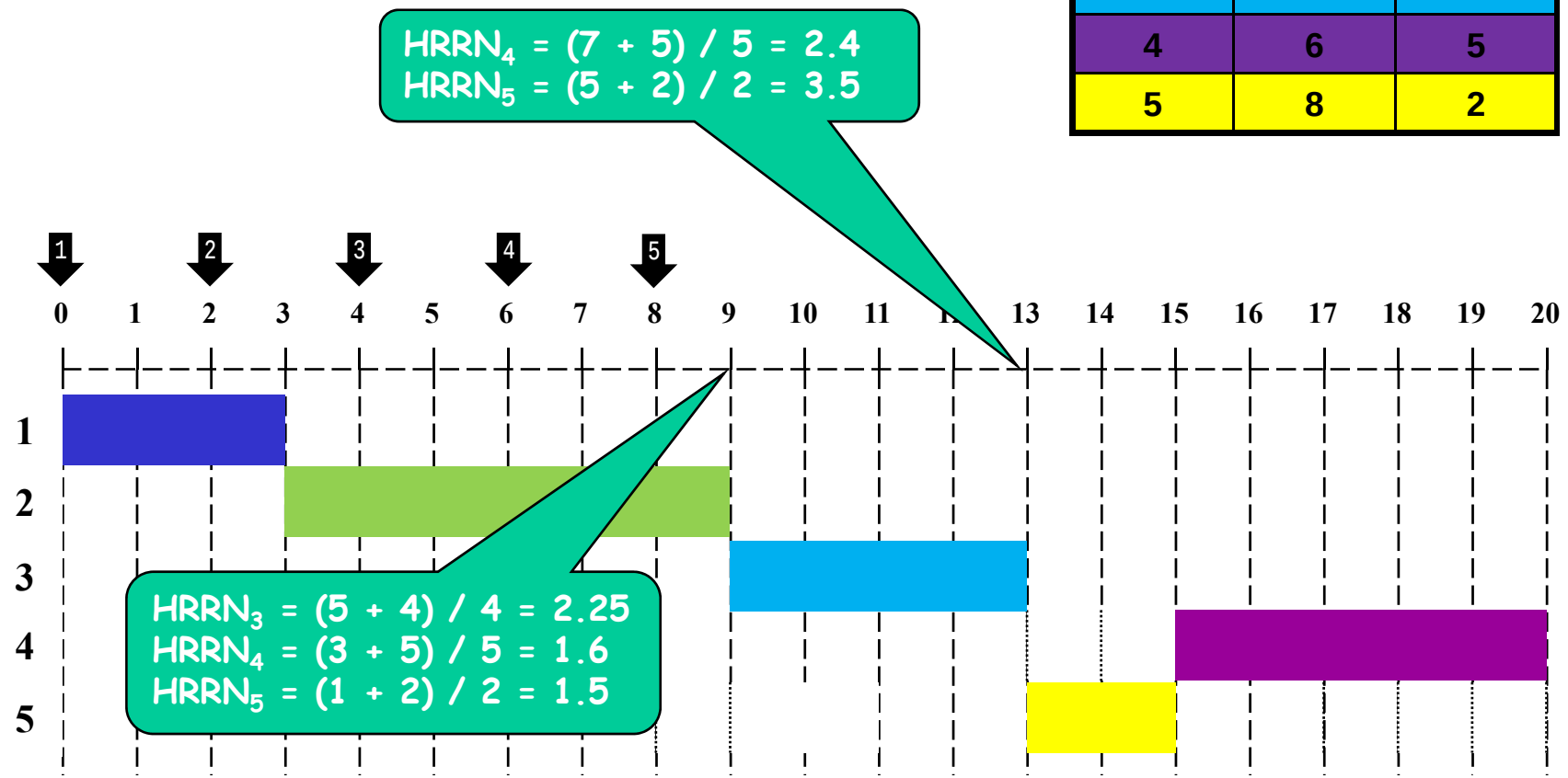
Highest Response Ratio Next (HRRN)

- Non-preemptive, similar to Shortest Job Next (SJN) but without starvation.
- Attractive approach to scheduling because it accounts for the age of a process.
- While shorter jobs are favored (a smaller denominator yields a larger ratio), aging without service increases the ratio so that a longer process will eventually get past competing shorter jobs.
- Choose next process with the greatest ratio

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

Highest Response Ratio Next (HRRN)

Process	Arrival	Service
1	0	3
2	2	6
3	4	4
4	6	5
5	8	2



Highest Response Ratio Next (HRRN)

Process	Arrival Time (T_0) ms	Next burst(Δ t) ms	Finish Time(T_1) ms	Turnaround Time TAT = T_1-T_0	Waiting Time WT = TAT - Δt
1	0	3	3	3	0
2	2	6	9	7	1
3	4	4	13	9	5
4	6	5	20	14	9
5	8	2	15	7	5
				40	20

- Average Turnaround time = 40/5 = 8
- Average waiting time = 20/5 = 4

Feedback Scheduling

- If we have no indication of the relative length of various processes, then SPN, SRT, and HRRN cannot be effectively used.
- Feedback scheduling is done on a preemptive basis with a dynamic priority mechanism.
- Penalize jobs that have been running longer: putting them into lower priority ready queues
- Goals of Multi-Level Feedback Queuing:
  - Optimize turnaround time - focus is not on time remaining, but on time spent in execution so far.
  - Be responsive to interactive users.

Feedback Scheduling

- MLFQ varies the priority of a job based on its observed behavior.
  - Give preference to shorter jobs by penalizing jobs that have been running longer.
  - Give preference to I/O bound processes.
  - Separate processes into categories based on their processor needs.

Multi-level Feedback Queue

- Possible MLFQ scheduling algorithm:
  - New jobs are placed at the highest priority (top) queue.
  - If a job uses up an entire time slice while running, its priority is reduced (i.e., it moves down one queue).
  - If a job gives up the CPU before the time slice is up, it stays at the same priority level.
  - Within each queue, a simple FCFS mechanism is used except once in the lowest-priority queue, which is treated in a RR fashion.
  - After some time period S, move all the jobs in the system to the topmost queue.

Multi-level Feedback Queue

- Results
  - As longer processes drift downward, shorter processes are favored.
  - Can make turnaround time for longer processes intolerable.
  - To avoid starvation, preemption time for lower-priority processes is usually longer.
  - A process is demoted to the next lower-priority queue each time it returns to the ready queue.

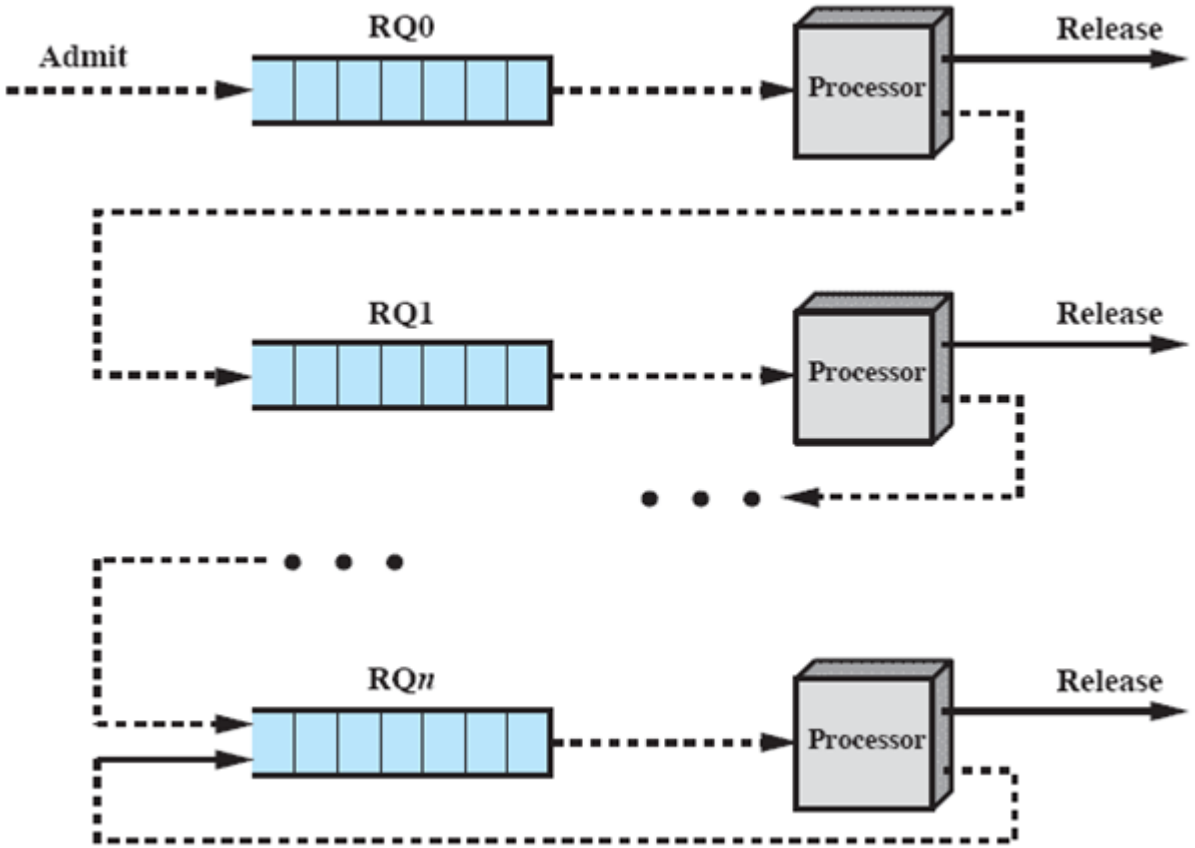
Feedback Algorithm

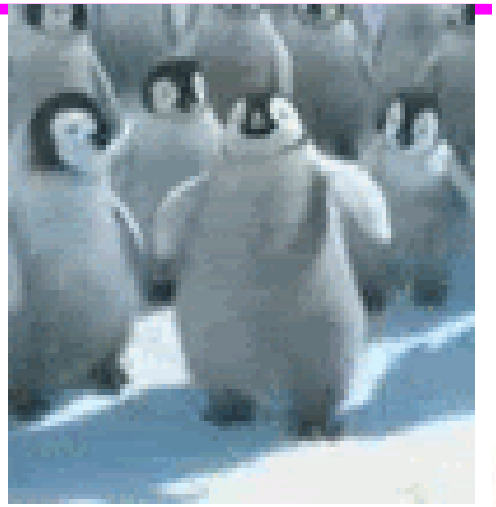
1. A new process is inserted at the end of the top-level FIFO queue.
2. At some stage the process reaches the head of the queue and is assigned the CPU.
3. If the process is completed within the time quantum of the given queue, it leaves the system.
4. If the process voluntarily relinquishes control of the CPU, it leaves the queuing network, and when the process becomes ready again it is inserted at the tail of the same queue which it relinquished earlier.
5. If the process uses all the quantum time, it is pre-empted and inserted at the end of the next lower level queue. This next lower level queue will have a time quantum which is more than that of the previous higher level queue.

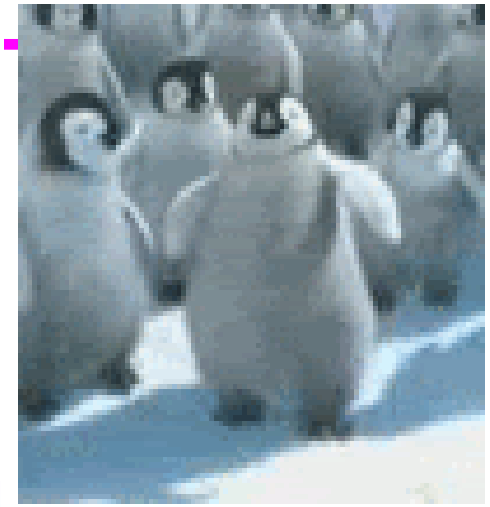
Feedback Algorithm

6. This scheme will continue until the process completes or it reaches the base level queue.
   - At the base level queue the processes circulate in round robin fashion until they complete and leave the system. Processes in the base level queue can also be scheduled on a first come first served basis.[4]
   - Optionally, if a process blocks for I/O, it is 'promoted' one level, and placed at the end of the next-higher queue. This allows I/O bound processes to be favored by the scheduler and allows processes to 'escape' the base level queue.

Feedback Scheduling







Minal Shah