Chapter 7: Scheduling

Process Scheduler

- Why do we even need a process scheduler?
  - In simplest form, CPU must be *shared* by
    - OS
    - Application
  - In reality, [multiprogramming]
    - OS : many separate pieces (processes)
    - Many Applications

- Scheduling [Policy] addresses...
  - When to remove a process from CPU?
  - Which ready process to allocate the CPU to?
Model of Process Execution

Recall Resource Manager
Scheduler as CPU Res Mgr

- Ready List
- Ready to run
- Dispatch
- Release

Scheduler

Units of time for a time-multiplexed CPU

Scheduler Components

- Ready Process
- Process Descriptor
- Enqueue
- Ready List
- Dispatch
- Context Switch

Running Process

From Other States
Context Switch

- Processes are switched out using **Context Switching**
- Context Switch:
  - **Save** pertinent info for current process
    - PC, Register, Status, etc.
  - **Update** PC, Register, Status, etc.
    - with info for process selected to run

- **Switching User Process**
  - 2 Context switches (CTX)
    - Process 1 running
      - CTX
      - Dispatcher: selects next process
      - CTX
      - Process 2 running

Process Context
Invoking the Scheduler

- Need a mechanism to call the scheduler
- Voluntary call
  - Process blocks itself
  - Calls the scheduler
- Involuntary call
  - External force (interrupt) blocks the process
  - Calls the scheduler
Contemporary Scheduling

- Involuntary CPU sharing – timer interrupts
  - *Time quantum* determined by interval timer – usually fixed size for every process using the system
  - Sometimes called the *time slice length*

---

Choosing a Process to Run

- *Mechanism* never changes
- Strategy = *policy* the dispatcher uses to select a process from the ready list
- Different policies for different requirements
Policy Considerations

- Policy can control/influence:
  - CPU utilization
  - Average time a process waits for service
  - Average amount of time to complete a job

- Could strive for any of:
  - Equitability
  - Favor very short or long jobs
  - Meet priority requirements
  - Meet deadlines

Optimal Scheduling

- Suppose the scheduler knows each process $p_i$’s service time, $\tau(p_i)$ -- or it can estimate each $\tau(p_i)$:

- Policy can optimize on any criteria, e.g.,
  - CPU utilization
  - Waiting time
  - Deadline

- To find an optimal schedule:
  - Have a finite, fixed # of $p_i$
  - Know $\tau(p_i)$ for each $p_i$
  - Enumerate all schedules, then choose the best
However ...

- The $\tau(p_i)$ are almost certainly just estimates
- General algorithm to choose optimal schedule is $O(n^2)$
- Other processes may arrive while these processes are being serviced
- Usually, optimal schedule is only a theoretical benchmark – scheduling policies try to approximate an optimal schedule

---

Model of Process Execution

![Diagram of process execution model](image-url)
Selection Strategies

- Motivation
  - To “optimize” some aspect of system behavior

- Considerations
  - Priority of process
    - External: assigned
    - Internal: aging
  - Fairness: no starvation
  - Overall Resource Utilization

...
**Talking About Scheduling ...**

- Let $P = \{p_i \mid 0 \leq i < n\}$ = set of processes
- Let $S(p_i) \in \{\text{running, ready, blocked}\}$
- Let $\tau(p_i)$ = Time process needs to be in running state (the *service time*)
- Let $W(p_i) = $ Time $p_i$ is in ready state before first transition to running (*wait time*)
- Let $T_{TRnd}(p_i) = $ Time from $p_i$ first enter ready to last exit ready (*turnaround time*)
- Batch *Throughput rate* = inverse of avg $T_{TRnd}$
- Timesharing response time = $W(p_i)$

---

**Definition & Terms**

- **Time Quantum**
  - Amount of time between timer interrupts
  - Also called Time Slice

- **Service Time $\tau(P_i)$**
  - Amount of time process needs to be in Running state (acquired CPU) before it is completed

- **Wait Time $W(P_i)$**
  - Time a process spends waiting in the Ready state before its *first* transition to the Running state
Definition & Terms...

- Turnaround Time $T (P_i)$
  - Amount of time between moment process first enters Ready state and the moment the process exits Running state for the last time (completed)

- Service time, Wait time & Turnaround time are measurable metrics used to compare scheduling algorithms

Simplified Model

- Simplified, but still provide analysis result
- Easy to analyze performance
Classes of Scheduling Algorithms

- 2 major classes
  - Non-preemptive
    - Run to completion
  - Preemptive
    - Process with highest priority always gets CPU

Recall: Several ways to establish priority

Nonpreemptive Schedulers

- Try to use the simplified scheduling model
- Only consider running and ready states
- Ignores time in blocked state:
  - “New process created when it enters ready state”
  - “Process is destroyed when it enters blocked state”
  - Really just looking at “small phases” of a process
First-Come-First-Served

\[ T_{\text{TRnd}}(p_0) = \tau(p_0) = 350 \]
\[ W(p_0) = 0 \]

\[ T_{\text{TRnd}}(p_1) = (\tau(p_1) + T_{\text{TRnd}}(p_0)) = 125 + 350 = 475 \]
\[ W(p_1) = T_{\text{TRnd}}(p_0) = 350 \]
First-Come-First-Served

<table>
<thead>
<tr>
<th>τ(p)</th>
<th>T_{TRnd}(p)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>350</td>
</tr>
<tr>
<td>1</td>
<td>125</td>
</tr>
<tr>
<td>2</td>
<td>475</td>
</tr>
<tr>
<td>3</td>
<td>250</td>
</tr>
<tr>
<td>4</td>
<td>75</td>
</tr>
</tbody>
</table>

W(p) = W(p) = 0

T_{TRnd}(p_0) = τ(p_0) = 350
T_{TRnd}(p_1) = (τ(p_1) + T_{TRnd}(p_0)) = 125 + 350 = 475
T_{TRnd}(p_2) = (τ(p_2) + T_{TRnd}(p_1)) = 475 + 475 = 950
T_{TRnd}(p_3) = (τ(p_3) + T_{TRnd}(p_2)) = 250 + 950 = 1200
First-Come-First-Served

\[ T_{TRnd}(p_i) = \tau(p_i) \]

\[
\begin{array}{c|c|c|c|c}
  i & \tau(p_i) & 0 & 350 & 1 & 125 & 2 & 475 & 3 & 250 & 4 & 75 & 1200 & 1275 \\
\hline
  P_0 & P_1 & P_2 & P_3 & P_4 & \\
\end{array}
\]

\[ W(p_0) = 0 \]
\[ W(p_1) = T_{TRnd}(p_0) = 350 \]
\[ W(p_2) = T_{TRnd}(p_1) = 475 \]
\[ W(p_3) = T_{TRnd}(p_2) = 950 \]
\[ W(p_4) = T_{TRnd}(p_3) = 1200 \]

\[ W_{avg} = \frac{0+350+475+950+1200}{5} = \frac{2974}{5} = 595 \]

---

FCFS Average Wait Time

\[ T_{TRnd}(p_i) = \tau(p_i) \]

\[
\begin{array}{c|c|c|c|c}
  i & \tau(p_i) & 0 & 350 & 1 & 125 & 2 & 475 & 3 & 250 & 4 & 75 & 0 & 350 & 475 & 900 & 1200 & 1275 \\
\hline
  P_0 & P_1 & P_2 & P_3 & P_4 & \\
\end{array}
\]

\[ W(p_0) = 0 \]
\[ W(p_1) = T_{TRnd}(p_0) = 350 \]
\[ W(p_2) = T_{TRnd}(p_1) = 475 \]
\[ W(p_3) = T_{TRnd}(p_2) = 950 \]
\[ W(p_4) = T_{TRnd}(p_3) = 1200 \]

\[ W_{avg} = \frac{0+350+475+950+1200}{5} = \frac{2974}{5} = 595 \]
Shortest Job Next

\[
\begin{array}{c|c|c|c}
   n & \tau(p_i) & 0 & 75 \\
   \hline
   0 & 350 & \n & 75 \\
   1 & 125 & \n & 75 \\
   2 & 475 & \n & 75 \\
   3 & 250 & \n & 75 \\
   4 & 75 & \n & 75 \\
\end{array}
\]

\[T_{\text{trnd}}(p_4) = \tau(p_4) = 75\]

\[W(p_4) = 0\]

\[
\begin{array}{c|c|c|c}
   n & \tau(p_i) & 0 & 75 \\
   \hline
   0 & 350 & \n & 75 \\
   1 & 125 & \n & 75 \\
   2 & 475 & \n & 75 \\
   3 & 250 & \n & 75 \\
   4 & 75 & \n & 75 \\
\end{array}
\]

\[T_{\text{trnd}}(p_4) = \tau(p_4) = 75\]

\[W(p_4) = 0\]
Shortest Job Next

1  \( t(p_1) \)
0  350
1  125
2  475
3  250
4  75

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>75</th>
<th>200</th>
<th>450</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P1</td>
<td>P2</td>
<td>P3</td>
</tr>
</tbody>
</table>

\[ T_{\text{TRND}}(p_1) = t(p_1) + t(p_4) = 125 + 75 = 200 \]
\[ W(p_1) = 75 \]

\[ T_{\text{TRND}}(p_3) = t(p_3) + t(p_1) + t(p_4) = 250 + 125 + 75 = 450 \]
\[ T_{\text{TRND}}(p_4) = t(p_4) = 75 \]
\[ W(p_3) = 200 \]
\[ W(p_4) = 0 \]

\[ T_{\text{TRND}}(p_0) = t(p_0) + t(p_3) + t(p_1) + t(p_4) = 350 + 250 + 125 + 75 = 800 \]
\[ W(p_0) = 450 \]
\[ T_{\text{TRND}}(p_1) = t(p_1) + t(p_4) = 125 + 75 = 200 \]
\[ W(p_1) = 75 \]

\[ T_{\text{TRND}}(p_3) = t(p_3) + t(p_1) + t(p_4) = 250 + 125 + 75 = 450 \]
\[ W(p_3) = 200 \]
\[ W(p_4) = 0 \]
Shortest Job Next

- **Minimizes wait time**
- May starve large jobs
- Must know service times

### Example

<table>
<thead>
<tr>
<th>Job</th>
<th>( \tau(p_i) )</th>
<th>( W(p_i) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>350</td>
<td>450</td>
</tr>
<tr>
<td>1</td>
<td>125</td>
<td>75</td>
</tr>
<tr>
<td>2</td>
<td>475</td>
<td>800</td>
</tr>
<tr>
<td>3</td>
<td>250</td>
<td>200</td>
</tr>
<tr>
<td>4</td>
<td>75</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ W_{avg} = \frac{(450+75+800+200+0)}{5} = \frac{1525}{5} = 305 \]
Priority Scheduling

- Reflects importance of external use
- May cause starvation
- Can address starvation with aging

<table>
<thead>
<tr>
<th>i</th>
<th>τ(p_i)</th>
<th>Pri</th>
<th>Deadline</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>350</td>
<td>5</td>
<td>575</td>
</tr>
<tr>
<td>1</td>
<td>125</td>
<td>2</td>
<td>550</td>
</tr>
<tr>
<td>2</td>
<td>475</td>
<td>3</td>
<td>1050</td>
</tr>
<tr>
<td>3</td>
<td>250</td>
<td>1</td>
<td>(none)</td>
</tr>
<tr>
<td>4</td>
<td>75</td>
<td>4</td>
<td>200</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>p</th>
<th>0</th>
<th>250</th>
<th>375</th>
<th>850</th>
<th>925</th>
<th>1275</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>575</td>
</tr>
<tr>
<td>p_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>550</td>
</tr>
<tr>
<td>p_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1050</td>
</tr>
<tr>
<td>p_3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(none)</td>
</tr>
<tr>
<td>p_4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>200</td>
</tr>
</tbody>
</table>

\( T_{TRnd}(p_0) = \tau(p_0)+\tau(p_4)+\tau(p_2)+\tau(p_1) = 350+75+475+125+250 = 1275 \)
\( W(p_0) = 925 \)
\( T_{TRnd}(p_1) = \tau(p_1)+\tau(p_3) = 125+250 = 375 \)
\( W(p_1) = 250 \)
\( T_{TRnd}(p_2) = \tau(p_2)+\tau(p_1)+\tau(p_3) = 475+125+250 = 850 \)
\( W(p_2) = 375 \)
\( T_{TRnd}(p_3) = \tau(p_3) = 250 \)
\( W(p_3) = 0 \)
\( T_{TRnd}(p_4) = \tau(p_3)+\tau(p_2)+\tau(p_1)+\tau(p_3) = 75+475+125+250 = 925 \)
\( W(p_4) = 850 \)

\( W_{avg} = (925+250+375+0+850)/5 = 2400/5 = 480 \)

Deadline Scheduling

- Allocates service by deadline
- May not be feasible

<table>
<thead>
<tr>
<th>i</th>
<th>τ(p_i)</th>
<th>Deadline</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>350</td>
<td>575</td>
</tr>
<tr>
<td>1</td>
<td>125</td>
<td>550</td>
</tr>
<tr>
<td>2</td>
<td>475</td>
<td>1050</td>
</tr>
<tr>
<td>3</td>
<td>250</td>
<td>(none)</td>
</tr>
<tr>
<td>4</td>
<td>75</td>
<td>200</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>p</th>
<th>0</th>
<th>200</th>
<th>550</th>
<th>575</th>
<th>1050</th>
<th>1275</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>575</td>
</tr>
<tr>
<td>p_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1050</td>
</tr>
<tr>
<td>p_3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(none)</td>
</tr>
<tr>
<td>p_4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>200</td>
</tr>
</tbody>
</table>

\( W_{avg} = (925+250+375+0+850)/5 = 2400/5 = 480 \)
Preemptive Schedulers

- Highest priority process is guaranteed to be running at all times
  - Or at least at the beginning of a time slice
- Dominant form of contemporary scheduling
- But complex to build & analyze

Round Robin (TQ=50)

\[
\begin{array}{c|c|c|c|c}
\text{Process} & \tau(p) & T & B & P_0 \\
0 & 350 & 125 & 475 & 250 \newline & 75 & \text{W}(p_0) = 0
\end{array}
\]
### Round Robin (TQ=50)

<table>
<thead>
<tr>
<th>Time (t)</th>
<th>Process (p)</th>
<th>Ready Queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>p0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>p1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>p2</td>
<td>p0</td>
</tr>
<tr>
<td>3</td>
<td>p3</td>
<td>p1</td>
</tr>
<tr>
<td>4</td>
<td>p4</td>
<td>p2</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
W(p_0) &= 0 \\
W(p_1) &= 50
\end{align*}
\]

### Round Robin (TQ=50)

<table>
<thead>
<tr>
<th>Time (t)</th>
<th>Process (p)</th>
<th>Ready Queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>p0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>p1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>p2</td>
<td>p0</td>
</tr>
<tr>
<td>3</td>
<td>p3</td>
<td>p1</td>
</tr>
<tr>
<td>4</td>
<td>p4</td>
<td>p2</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
W(p_0) &= 0 \\
W(p_1) &= 50 \\
W(p_2) &= 100
\end{align*}
\]
### Round Robin (TQ=50)

<table>
<thead>
<tr>
<th>i</th>
<th>τ(pᵢ)</th>
<th>0</th>
<th>100</th>
<th>200</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>350</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>125</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>475</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>250</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>75</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- \( W(p₀) = 0 \)
- \( W(p₁) = 50 \)
- \( W(p₂) = 100 \)
- \( W(p₃) = 150 \)

### Round Robin (TQ=50)

<table>
<thead>
<tr>
<th>i</th>
<th>τ(pᵢ)</th>
<th>0</th>
<th>100</th>
<th>200</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>350</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>125</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>475</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>250</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>75</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- \( W(p₀) = 0 \)
- \( W(p₁) = 50 \)
- \( W(p₂) = 100 \)
- \( W(p₃) = 150 \)
- \( W(p₄) = 200 \)
Round Robin (TQ=50)

1 \tau(p_i) 
0 350 
1 125 
2 475 
3 250 
4 75 

\begin{tabular}{c|cccc}
 & P_0 & P_1 & P_2 & P_3 \\
0 & 0 & 100 & 200 & 300 \\
\end{tabular}

\begin{align*}
W(p_0) &= 0 \\
W(p_1) &= 50 \\
W(p_2) &= 100 \\
W(p_3) &= 150 \\
W(p_4) &= 200 \\
\end{align*}

T_{TR}(p_4) = 475
### Round Robin (TQ=50)

<table>
<thead>
<tr>
<th>i</th>
<th>τ(p&lt;sub&gt;i&lt;/sub&gt;)</th>
<th>0</th>
<th>100</th>
<th>200</th>
<th>300</th>
<th>400</th>
<th>475</th>
<th>550</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>350</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>125</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>475</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>250</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>75</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
T_{\text{TRnd}}(p_0) = 550
\]

\[
W(p_0) = 0
\]

\[
T_{\text{TRnd}}(p_1) = 550
\]

\[
W(p_1) = 50
\]

\[
T_{\text{TRnd}}(p_2) = 950
\]

\[
W(p_2) = 100
\]

\[
T_{\text{TRnd}}(p_3) = 475
\]

\[
W(p_3) = 150
\]

\[
T_{\text{TRnd}}(p_4) = 475
\]

\[
W(p_4) = 200
\]
**Round Robin (TQ=50)**

| \( \tau(p_i) \) | 0  | 350 | 1  | 125 | 2  | 475 | 3  | 250 | 4  | 75 |

<table>
<thead>
<tr>
<th>0</th>
<th>100</th>
<th>200</th>
<th>300</th>
<th>400</th>
<th>475</th>
<th>550</th>
<th>650</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p_0 )</td>
<td>( p_1 )</td>
<td>( p_2 )</td>
<td>( p_3 )</td>
<td>( p_4 )</td>
<td>( p_0 )</td>
<td>( p_1 )</td>
<td>( p_2 )</td>
</tr>
</tbody>
</table>

**Round Robin (TQ=50)**

| \( \tau(p_i) \) | 0  | 350 | 1  | 125 | 2  | 475 | 3  | 250 | 4  | 75 |

<table>
<thead>
<tr>
<th>0</th>
<th>100</th>
<th>200</th>
<th>300</th>
<th>400</th>
<th>475</th>
<th>550</th>
<th>650</th>
<th>750</th>
<th>850</th>
<th>950</th>
<th>1050</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p_0 )</td>
<td>( p_1 )</td>
<td>( p_2 )</td>
<td>( p_3 )</td>
<td>( p_4 )</td>
<td>( p_0 )</td>
<td>( p_1 )</td>
<td>( p_2 )</td>
<td>( p_3 )</td>
<td>( p_4 )</td>
<td>( p_0 )</td>
<td>( p_1 )</td>
</tr>
</tbody>
</table>

- **\( T_{TR_{ad}}(p_0) = 1100 \)**
- **\( T_{TR_{ad}}(p_1) = 550 \)**
- **\( T_{TR_{ad}}(p_2) = 1275 \)**
- **\( T_{TR_{ad}}(p_3) = 950 \)**
- **\( T_{TR_{ad}}(p_4) = 475 \)**

- **\( W(p_0) = 0 \)**
- **\( W(p_1) = 50 \)**
- **\( W(p_2) = 100 \)**
- **\( W(p_3) = 150 \)**
- **\( W(p_4) = 200 \)**
### Round Robin (TQ=50)

- **Equitable**
- **Most widely-used**
- Fits naturally with interval timer

<table>
<thead>
<tr>
<th>i</th>
<th>$\tau(p_i)$</th>
<th>$W(p_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>350</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>125</td>
<td>50</td>
</tr>
<tr>
<td>2</td>
<td>475</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>250</td>
<td>150</td>
</tr>
<tr>
<td>4</td>
<td>75</td>
<td>200</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$T_{TRnd}(p_i)$</th>
<th>$W(p_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{TRnd}(p_0)$</td>
<td>$1100$</td>
</tr>
<tr>
<td>$T_{TRnd}(p_1)$</td>
<td>$550$</td>
</tr>
<tr>
<td>$T_{TRnd}(p_2)$</td>
<td>$1275$</td>
</tr>
<tr>
<td>$T_{TRnd}(p_3)$</td>
<td>$950$</td>
</tr>
<tr>
<td>$T_{TRnd}(p_4)$</td>
<td>$475$</td>
</tr>
</tbody>
</table>

$T_{TRnd-avg} = \frac{(1100+550+1275+950+475)}{5} = 4350/5 = 870$

$W_{avg} = \frac{(0+50+100+150+200)}{5} = 500/5 = 100$

### RR with Overhead=10 (TQ=50)

- Overhead must be considered

<table>
<thead>
<tr>
<th>i</th>
<th>$\tau(p_i)$</th>
<th>$W(p_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>350</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>125</td>
<td>60</td>
</tr>
<tr>
<td>2</td>
<td>475</td>
<td>120</td>
</tr>
<tr>
<td>3</td>
<td>250</td>
<td>180</td>
</tr>
<tr>
<td>4</td>
<td>75</td>
<td>240</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$T_{TRnd}(p_i)$</th>
<th>$W(p_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{TRnd}(p_0)$</td>
<td>$1320$</td>
</tr>
<tr>
<td>$T_{TRnd}(p_1)$</td>
<td>$660$</td>
</tr>
<tr>
<td>$T_{TRnd}(p_2)$</td>
<td>$1535$</td>
</tr>
<tr>
<td>$T_{TRnd}(p_3)$</td>
<td>$1140$</td>
</tr>
<tr>
<td>$T_{TRnd}(p_4)$</td>
<td>$565$</td>
</tr>
</tbody>
</table>

$T_{TRnd-avg} = \frac{(1320+660+1535+1140+565)}{5} = 5220/5 = 1044$

$W_{avg} = \frac{(0+60+120+180+240)}{5} = 600/5 = 120$
Multi-Level Queues

- Preemption or voluntary yield

New Process

- Ready List_0
- Ready List_1
- Ready List_2
- Ready List_3

Scheduler

CPU

Done

- All processes at level i run before any process at level j
- At a level, use another policy, e.g. RR

Contemporary Scheduling

- Involuntary CPU sharing -- timer interrupts
  - *Time quantum* determined by interval timer -- usually fixed for every process using the system
  - Sometimes called the *time slice length*

- Priority-based process (job) selection
  - Select the highest priority process
  - Priority reflects policy

- With *preemption*

- Usually a variant of *Multi-Level Queues*
**BSD 4.4 Scheduling**

- Involuntary CPU Sharing
- Preemptive algorithms
- 32 Multi-Level Queues
  - Queues 0-7 are reserved for system functions
  - Queues 8-31 are for user space functions
  - `nice` influences (but does not dictate) queue level

**UNIX Scheduler**

The UNIX scheduler is based on a multilevel queue structure

*Taken from Modern Operating Systems, 2nd Ed, Tanenbaum, 2001*
Process Life Cycle

Job and Process Scheduler

**Job Scheduler**
- Controls when jobs will be allowed to contend the CPU
- Most popular techniques
  - **FIFO**  First in, first out
  - **SJF**   Shortest job first

**Process Scheduler**
- Controls when individual jobs (processes) will actually get the CPU
- Only interesting in multi-programming
- Most popular technique is **Round Robin**
  - Give each process one time slice in turn until complete
Turnaround and Weighted Turnaround Time

Let:

- \( N \) be number of jobs
- \( A_i \) be arrival time of \( i \)-th job
- \( F_i \) be finish time of \( i \)-th job

- Turnaround time for \( i \)-th job:
  \[ T_i = F_i - A_i \]
- Average turnaround time for \( i \)-th job:
  \[ T = \frac{\sum T_i}{N} \]
- Weighted turnaround time for \( i \)-th job:
  \[ WT_i = \frac{(F_i - A_i)}{(Service-time)_i} \]
- Average Weighted Turnaround time:
  \[ WT = \frac{\sum WT_i}{N} \]

Processor Sharing (PS)

“Theoretical” Scheduling Algorithm

- Limit of RR as time quantum goes to zero.
- Like giving each CPU cycle to a different process, in round robin fashion.
- \( N \) processes scheduled by PS
  - Each job runs on dedicated \( N \)-fold slower CPU.
  - Thus, READY = RUNNING.
- CPU Time “shared” equally among processes
### Scheduling Example 2

**Assume:**

- Multiprogramming
- FIFO Job Scheduling
- Processor Sharing Process Scheduling

#### Job Arrival Times and Run Times

<table>
<thead>
<tr>
<th>Job</th>
<th>Arrives</th>
<th>Run Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10.0</td>
<td>0.3</td>
</tr>
<tr>
<td>2</td>
<td>10.2</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>10.4</td>
<td>0.1</td>
</tr>
<tr>
<td>4</td>
<td>10.5</td>
<td>0.4</td>
</tr>
<tr>
<td>5</td>
<td>10.8</td>
<td>0.1</td>
</tr>
</tbody>
</table>

### Definitions

- **Current event:** 
  - A=Arrival
  - S=Schedule
  - F=Finish
  - H=In HoldQ

- **Reduction in run time** that each process in ready queue experiences since last event occurred

- **Remaining run time** of each process in ready queue

- **Time**
- **Event**
- **# Jobs**
- **Headway**
- **Time Left**
## Example 2 Continued

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th># Jobs</th>
<th>Headway</th>
<th>Time Left</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0</td>
<td>1 A,S</td>
<td>1</td>
<td>0.2</td>
<td>1</td>
</tr>
<tr>
<td>10.2</td>
<td>2 A,S</td>
<td>1</td>
<td>0.2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>10.4</td>
<td>1 F</td>
<td>2</td>
<td>0.1</td>
<td>2</td>
</tr>
<tr>
<td>10.5</td>
<td>4 A,S</td>
<td>2</td>
<td>0.05</td>
<td>2</td>
</tr>
<tr>
<td>10.65</td>
<td>3 F</td>
<td>3</td>
<td>0.05</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
</tbody>
</table>

## Example 2 Continued...

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th># Jobs</th>
<th>Headway</th>
<th>Time Left</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.8</td>
<td>5 A,S</td>
<td>2</td>
<td>0.075</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.225</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.275</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td>11.1</td>
<td>5 F</td>
<td>3</td>
<td>0.1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.125</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.175</td>
</tr>
<tr>
<td>11.35</td>
<td>2 F</td>
<td>2</td>
<td>0.125</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.05</td>
</tr>
<tr>
<td>11.40</td>
<td>4 F</td>
<td>1</td>
<td>0.05</td>
<td>4</td>
</tr>
</tbody>
</table>
T and W for Example 2

<table>
<thead>
<tr>
<th>Job</th>
<th>Run</th>
<th>Start</th>
<th>Finish</th>
<th>Ti</th>
<th>WTi</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.3</td>
<td>10.0</td>
<td>10.4</td>
<td>0.4</td>
<td>1.33</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>10.2</td>
<td>11.35</td>
<td>1.15</td>
<td>2.3</td>
</tr>
<tr>
<td>3</td>
<td>0.1</td>
<td>10.4</td>
<td>10.65</td>
<td>0.25</td>
<td>2.5</td>
</tr>
<tr>
<td>4</td>
<td>0.4</td>
<td>10.5</td>
<td>11.4</td>
<td>0.9</td>
<td>2.25</td>
</tr>
<tr>
<td>5</td>
<td>0.1</td>
<td>10.8</td>
<td>11.1</td>
<td>0.3</td>
<td>3.0</td>
</tr>
</tbody>
</table>

\[
T = 0.6 \quad WT = 2.276
\]

Check:
Because CPU was never idle, 1.4 + 10.0 must equal time of last event (11.4)

---

Scheduling Example 4

Assume:
FIFO Job Scheduling
5 Tape Drives
100 K Main Memory
Processor Sharing Process Scheduling

<table>
<thead>
<tr>
<th>Job</th>
<th>Arrives</th>
<th>Run Time</th>
<th>Memory</th>
<th>Tapes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0</td>
<td>0.5</td>
<td>30</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1.2</td>
<td>1.0</td>
<td>50</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1.3</td>
<td>1.5</td>
<td>50</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1.4</td>
<td>2.0</td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1.7</td>
<td>0.5</td>
<td>30</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>2.1</td>
<td>1.0</td>
<td>30</td>
<td>2</td>
</tr>
</tbody>
</table>
### Example 4 Continued

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th># Jobs</th>
<th>HWay</th>
<th>MM</th>
<th>Tapes</th>
<th>Time Left</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>1 A,S</td>
<td>1</td>
<td>0.2</td>
<td>70</td>
<td>3</td>
<td>1.0</td>
</tr>
<tr>
<td>1.2</td>
<td>2 A,S</td>
<td>1</td>
<td>0.2</td>
<td>20</td>
<td>2</td>
<td>1.0</td>
</tr>
<tr>
<td>1.3</td>
<td>3 A,H</td>
<td>2</td>
<td>0.05</td>
<td>20</td>
<td>2</td>
<td>0.25</td>
</tr>
<tr>
<td>1.4</td>
<td>4 A,S</td>
<td>2</td>
<td>0.05</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>1.7</td>
<td>5 A,H</td>
<td>3</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
</tr>
<tr>
<td>2.0</td>
<td>1 F</td>
<td>3</td>
<td>0.1</td>
<td>30</td>
<td>2</td>
<td>0.7</td>
</tr>
</tbody>
</table>

### Example 4 Continued ...

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th># Jobs</th>
<th>HWay</th>
<th>MM</th>
<th>Tapes</th>
<th>Time Left</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>6 A,S</td>
<td>2</td>
<td>0.05</td>
<td>0</td>
<td>0</td>
<td>2.0</td>
</tr>
<tr>
<td>4.05</td>
<td>2 F</td>
<td>3</td>
<td>0.65</td>
<td>50</td>
<td>1</td>
<td>1.75</td>
</tr>
<tr>
<td></td>
<td>3 S</td>
<td>3</td>
<td>0.65</td>
<td>50</td>
<td>1</td>
<td>1.75</td>
</tr>
<tr>
<td>5.1</td>
<td>6 F</td>
<td>3</td>
<td>0.35</td>
<td>30</td>
<td>2</td>
<td>0.75</td>
</tr>
<tr>
<td>6.6</td>
<td>4 F</td>
<td>2</td>
<td>0.75</td>
<td>50</td>
<td>4</td>
<td>0.4</td>
</tr>
<tr>
<td>7.4</td>
<td>3 F</td>
<td>2</td>
<td>0.4</td>
<td>70</td>
<td>2</td>
<td>0.1</td>
</tr>
<tr>
<td>7.5</td>
<td>5 F</td>
<td>1</td>
<td>0.1</td>
<td>100</td>
<td>5</td>
<td>0.01</td>
</tr>
</tbody>
</table>
### T and W for Example 4

<table>
<thead>
<tr>
<th>Job</th>
<th>Run</th>
<th>Arrives</th>
<th>Finish</th>
<th>Ti</th>
<th>WTi</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5</td>
<td>1.0</td>
<td>2.0</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>1.2</td>
<td>4.05</td>
<td>2.85</td>
<td>2.85</td>
</tr>
<tr>
<td>3</td>
<td>1.5</td>
<td>1.3</td>
<td>7.4</td>
<td>6.1</td>
<td>4.06</td>
</tr>
<tr>
<td>4</td>
<td>2.0</td>
<td>1.4</td>
<td>6.6</td>
<td>5.2</td>
<td>2.6</td>
</tr>
<tr>
<td>5</td>
<td>0.5</td>
<td>1.7</td>
<td>7.5</td>
<td>5.8</td>
<td>11.6</td>
</tr>
<tr>
<td>6</td>
<td>2.1</td>
<td>2.1</td>
<td>5.1</td>
<td>3.0</td>
<td>3.0</td>
</tr>
</tbody>
</table>

\[
\begin{array}{cc}
T & = 3.99 \\
WT & = 4.35
\end{array}
\]