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

![Model of Process Execution Diagram]

![Recall Resource Manager Diagram]
Scheduler as CPU Res Mgr

Units of time for a time-multiplexed CPU

Scheduler Components

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
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

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
**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**
- New Process
- Ready List
- Scheduler
- CPU
- Done
- Preemption or voluntary yield
- "Ready"
- Resource Manager
- "Blocked"
- "Request"
- Allocate
- "Resources"
Selection Strategies

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

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

...
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

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

Simplified Model

- Simplified, but still provide analysis result
- Easy to analyze performance

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

\[
T_{\text{max}}(p_0) = \pi(p_0) = 350 \\
W(p_0) = 0
\]

\[
T_{\text{max}}(p_1) = (\pi(p_1) + T_{\text{max}}(p_0)) = 125 + 350 = 475 \\
W(p_1) = T_{\text{max}}(p_0) = 350
\]

\[
T_{\text{max}}(p_2) = (\pi(p_2) + T_{\text{max}}(p_1)) = 250 + 475 = 725 \\
W(p_2) = T_{\text{max}}(p_1) = 475
\]

First-Come-First-Served

\[
T_{\text{max}}(p_0) = \pi(p_0) = 350 \\
W(p_0) = 0
\]

\[
T_{\text{max}}(p_1) = (\pi(p_1) + T_{\text{max}}(p_0)) = 125 + 350 = 475 \\
W(p_1) = T_{\text{max}}(p_0) = 350
\]

\[
T_{\text{max}}(p_2) = (\pi(p_2) + T_{\text{max}}(p_1)) = 250 + 475 = 725 \\
W(p_2) = T_{\text{max}}(p_1) = 475
\]
### First-Come-First-Served

| \( p_0 \) | 0  | 350 |
| \( p_1 \) | 1  | 125 |
| \( p_2 \) | 2  | 475 |
| \( p_3 \) | 3  | 250 |
| \( p_4 \) | 4  | 75  |

\[
\begin{align*}
T_{\text{TRnd}}(p_0) &= t(p_0) = 350 \\
T_{\text{TRnd}}(p_1) &= (t(p_1) + T_{\text{TRnd}}(p_0)) = 125 + 350 = 475 \\
T_{\text{TRnd}}(p_2) &= (t(p_2) + T_{\text{TRnd}}(p_1)) = 475 + 475 = 950 \\
T_{\text{TRnd}}(p_3) &= (t(p_3) + T_{\text{TRnd}}(p_2)) = 250 + 950 = 1200 \\
T_{\text{TRnd}}(p_4) &= (t(p_4) + T_{\text{TRnd}}(p_3)) = 75 + 1200 = 1275
\end{align*}
\]

\[
\begin{align*}
W(p_0) &= 0 \\
W(p_1) &= T_{\text{TRnd}}(p_0) = 350 \\
W(p_2) &= T_{\text{TRnd}}(p_1) = 475 \\
W(p_3) &= T_{\text{TRnd}}(p_2) = 950 \\
W(p_4) &= T_{\text{TRnd}}(p_3) = 1200
\end{align*}
\]

\[
\text{W}_{\text{avg}} = \frac{0 + 350 + 475 + 950 + 1200}{5} = 2974/5 = 595
\]

### FCFS Average Wait Time

| \( p_0 \) | 0  | 350 |
| \( p_1 \) | 1  | 125 |
| \( p_2 \) | 2  | 475 |
| \( p_3 \) | 3  | 250 |
| \( p_4 \) | 4  | 75  |

\[
\begin{align*}
T_{\text{TRnd}}(p_0) &= t(p_0) = 350 \\
T_{\text{TRnd}}(p_1) &= (t(p_1) + T_{\text{TRnd}}(p_0)) = 125 + 350 = 475 \\
T_{\text{TRnd}}(p_2) &= (t(p_2) + T_{\text{TRnd}}(p_1)) = 475 + 475 = 950 \\
T_{\text{TRnd}}(p_3) &= (t(p_3) + T_{\text{TRnd}}(p_2)) = 250 + 950 = 1200 \\
T_{\text{TRnd}}(p_4) &= (t(p_4) + T_{\text{TRnd}}(p_3)) = 75 + 1200 = 1275
\end{align*}
\]

\[
\begin{align*}
W(p_0) &= 0 \\
W(p_1) &= T_{\text{TRnd}}(p_0) = 350 \\
W(p_2) &= T_{\text{TRnd}}(p_1) = 475 \\
W(p_3) &= T_{\text{TRnd}}(p_2) = 950 \\
W(p_4) &= T_{\text{TRnd}}(p_3) = 1200
\end{align*}
\]

\[
\text{W}_{\text{avg}} = \frac{0 + 350 + 475 + 950 + 1200}{5} = 2974/5 = 595
\]

### Shortest Job Next

| \( p_0 \) | 0  | 350 |
| \( p_1 \) | 1  | 125 |
| \( p_2 \) | 2  | 475 |
| \( p_3 \) | 3  | 250 |
| \( p_4 \) | 4  | 75  |

\[
\begin{align*}
T_{\text{TRnd}}(p_0) &= t(p_0) = 75 \\
T_{\text{TRnd}}(p_1) &= t(p_1) + t(p_0) = 125 + 75 = 200 \\
T_{\text{TRnd}}(p_2) &= t(p_2) = 75 \\
T_{\text{TRnd}}(p_3) &= t(p_3) = 75 \\
W(p_4) &= 0
\end{align*}
\]

\[
\begin{align*}
T_{\text{TRnd}}(p_0) &= t(p_0) = 75 \\
T_{\text{TRnd}}(p_1) &= t(p_1) + t(p_0) = 125 + 75 = 200 \\
T_{\text{TRnd}}(p_2) &= t(p_2) = 75 \\
T_{\text{TRnd}}(p_3) &= t(p_3) = 75 \\
W(p_4) &= 0
\end{align*}
\]
Shortest Job Next

<table>
<thead>
<tr>
<th>i</th>
<th>(p_i)</th>
<th>0</th>
<th>75</th>
<th>200</th>
<th>450</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>350</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>
</tr>
<tr>
<td>2</td>
<td>475</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>
</tr>
<tr>
<td>4</td>
<td>75</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ T_{\text{TRnd}}(p_i) = \sum_{j=1}^{i} t(p_j) \]

\[ W(p_i) = T_{\text{TRnd}}(p_i) \]

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

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

Deadline Scheduling

- Allocates service by deadline
- May not be feasible

Preemptive Schedulers

- New Process
- Ready List
- Scheduler
- CPU
- Done

- 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)

- W(p_0) = 0
Round Robin (TQ=50)

1. \( \pi(p_0) \)
   0 350
   1 125
   2 475
   3 250
   4 75

\[ W(p_0) = 0 \]
\[ W(p_1) = 50 \]

Round Robin (TQ=50)

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

\[ W(p_0) = 0 \]
\[ W(p_1) = 50 \]
\[ W(p_2) = 100 \]

Round Robin (TQ=50)

1. \( \pi(p_2) \)
   0 350
   1 125
   2 475
   3 250
   4 75

\[ W(p_0) = 0 \]
\[ W(p_1) = 50 \]
\[ W(p_2) = 100 \]
\[ W(p_3) = 150 \]

Round Robin (TQ=50)

1. \( \pi(p_3) \)
   0 350
   1 125
   2 475
   3 250
   4 75

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

<table>
<thead>
<tr>
<th></th>
<th>p_i</th>
<th>T_{\text{TRnd}}(p_i)</th>
<th>W(p_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>350</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>125</td>
<td></td>
<td>50</td>
</tr>
<tr>
<td>2</td>
<td>475</td>
<td></td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>250</td>
<td></td>
<td>150</td>
</tr>
<tr>
<td>4</td>
<td>75</td>
<td></td>
<td>200</td>
</tr>
</tbody>
</table>

### Round Robin (TQ=50)

<table>
<thead>
<tr>
<th></th>
<th>p_i</th>
<th>T_{\text{TRnd}}(p_i)</th>
<th>W(p_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>350</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>125</td>
<td></td>
<td>50</td>
</tr>
<tr>
<td>2</td>
<td>475</td>
<td></td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>250</td>
<td></td>
<td>150</td>
</tr>
<tr>
<td>4</td>
<td>75</td>
<td></td>
<td>200</td>
</tr>
</tbody>
</table>
**Round Robin (TQ=50)**

<table>
<thead>
<tr>
<th></th>
<th>( p_i )</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>

<table>
<thead>
<tr>
<th>( p_0 )</th>
<th>( p_1 )</th>
<th>( p_2 )</th>
<th>( p_3 )</th>
<th>( p_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>100</td>
<td>200</td>
<td>300</td>
<td>400</td>
</tr>
<tr>
<td>1</td>
<td>500</td>
<td>550</td>
<td>650</td>
<td>750</td>
</tr>
<tr>
<td>2</td>
<td>1000</td>
<td>1275</td>
<td>1535</td>
<td>1850</td>
</tr>
<tr>
<td>3</td>
<td>950</td>
<td>1140</td>
<td>1270</td>
<td>1510</td>
</tr>
<tr>
<td>4</td>
<td>475</td>
<td>565</td>
<td>635</td>
<td>790</td>
</tr>
</tbody>
</table>

\[ T_{\text{TR}}(p_0) = 1100 \]
\[ T_{\text{TR}}(p_1) = 550 \]
\[ T_{\text{TR}}(p_2) = 1275 \]
\[ T_{\text{TR}}(p_3) = 950 \]
\[ T_{\text{TR}}(p_4) = 475 \]

\[ W(p_0) = 0 \]
\[ W(p_1) = 50 \]
\[ W(p_2) = 100 \]
\[ W(p_3) = 150 \]
\[ W(p_4) = 200 \]

\[ T_{\text{TR avg}} = 1100 \]
\[ W_{\text{avg}} = 0 \]

\[ T_{\text{TR avg}} = 550 \]
\[ W_{\text{avg}} = 50 \]

\[ T_{\text{TR avg}} = 1275 \]
\[ W_{\text{avg}} = 100 \]

\[ T_{\text{TR avg}} = 950 \]
\[ W_{\text{avg}} = 150 \]

\[ T_{\text{TR avg}} = 475 \]
\[ W_{\text{avg}} = 200 \]

---

**Round Robin (TQ=50)**

<table>
<thead>
<tr>
<th></th>
<th>( p_i )</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>

<table>
<thead>
<tr>
<th>( p_0 )</th>
<th>( p_1 )</th>
<th>( p_2 )</th>
<th>( p_3 )</th>
<th>( p_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>120</td>
<td>240</td>
<td>360</td>
<td>480</td>
</tr>
<tr>
<td>1</td>
<td>350</td>
<td>520</td>
<td>670</td>
<td>790</td>
</tr>
<tr>
<td>2</td>
<td>475</td>
<td>660</td>
<td>790</td>
<td>910</td>
</tr>
<tr>
<td>3</td>
<td>140</td>
<td>1535</td>
<td>1640</td>
<td>1750</td>
</tr>
<tr>
<td>4</td>
<td>1320</td>
<td>1565</td>
<td>1790</td>
<td>2020</td>
</tr>
</tbody>
</table>

\[ T_{\text{TR}}(p_0) = 1320 \]
\[ T_{\text{TR}}(p_1) = 1350 \]
\[ T_{\text{TR}}(p_2) = 1565 \]
\[ T_{\text{TR}}(p_3) = 1640 \]
\[ T_{\text{TR}}(p_4) = 1790 \]

\[ W(p_0) = 0 \]
\[ W(p_1) = 60 \]
\[ W(p_2) = 120 \]
\[ W(p_3) = 180 \]
\[ W(p_4) = 240 \]

\[ T_{\text{TR avg}} = 1320+660+1350+1400+1565)/5 = 5220/5 = 1044 \]
\[ W_{\text{avg}} = (0+60+120+180+240)/5 = 600/5 = 120 \]
Multi-Level Queues

Preemption or voluntary yield

New Process

Ready List

Scheduler

CPU

Done

Ready List

• 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 Scheduler
- Process Scheduler

Dark square contains fixed, maximum number of processes

**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:
\[ WT_i = \frac{(F_i - A_i)}{\text{(Service-time)}_i} \]

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

<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

- **Number of processes in ready queue before this event**
- **Current event**: A=Arrival S=Schedule F=Finish H=In Hold
- **Reduction in run time that each process in ready queue experiences since last event occurred**
- **Remaining run time of each process in ready queue**

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>0.3</td>
<td>1 0.3</td>
</tr>
<tr>
<td>10.2</td>
<td>2 A,S</td>
<td>1</td>
<td>0.1</td>
<td>1 0.1</td>
</tr>
<tr>
<td>10.4</td>
<td>1 F</td>
<td>2</td>
<td>0.4</td>
<td>1 0.4</td>
</tr>
<tr>
<td>10.5</td>
<td>3 A,S</td>
<td>3</td>
<td>0.1</td>
<td>1 0.1</td>
</tr>
<tr>
<td>10.65</td>
<td>3 F</td>
<td>3</td>
<td>0.3</td>
<td>1 0.3</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 0.225</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4 0.275</td>
</tr>
<tr>
<td>11.1</td>
<td>5 F</td>
<td>3</td>
<td>0.1</td>
<td>2 0.125</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4 0.175</td>
</tr>
<tr>
<td>11.35</td>
<td>2 F</td>
<td>2</td>
<td>0.125</td>
<td>4 0.05</td>
</tr>
<tr>
<td>11.40</td>
<td>4 F</td>
<td>1</td>
<td>0.05</td>
<td>4 0.05</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 100 K Main Memory
5 Tape Drives 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 # 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>70</td>
<td>3</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>1.2</td>
<td>2 A,S</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0.3</td>
</tr>
<tr>
<td>1.3</td>
<td>3 A,H</td>
<td>2.05</td>
<td>2</td>
<td>1</td>
<td>0.25</td>
</tr>
<tr>
<td>1.4</td>
<td>4 A,S</td>
<td>0.05</td>
<td>0</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>1.7</td>
<td>5 A,H</td>
<td>0.1</td>
<td>0</td>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>2.0</td>
<td>1 F</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 # 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>
</tr>
<tr>
<td>4.05</td>
<td>2 F</td>
<td>3</td>
<td>0.65</td>
<td>50</td>
<td>1</td>
</tr>
<tr>
<td>5.1</td>
<td>6 F</td>
<td>3</td>
<td>0.35</td>
<td>30</td>
<td>2</td>
</tr>
<tr>
<td>6.6</td>
<td>4 F</td>
<td>2</td>
<td>0.75</td>
<td>50</td>
<td>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>
</tr>
<tr>
<td>7.5</td>
<td>5 F</td>
<td>1</td>
<td>0.1</td>
<td>100</td>
<td>5</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>$T_i$</th>
<th>$W_{Ti}$</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>

$T = 3.99 \quad WT = 4.35$

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>23.95</td>
</tr>
<tr>
<td></td>
<td>26.11</td>
</tr>
</tbody>
</table>