1 Administrative Information

This project is due Monday, Feb. 27 at the start of class. Each student is expected to do their own work and their own debugging.

You should use your account on eve.albany.edu to do this project. Project specific queries should be first be directed to the TA, Bill Rennie, rennie@cs.albany.edu, who will keep a FAQ. To save time for yourself and the TA, please check the FAQ before asking a question.

Questions or problem reports about eve should should be directed to helpdesk@albany.edu.

2 Project Overview

Recall the web server example in Topic 5, in this project we examine at what is going on internally to our web server. We are going to look at the actual configuration we are modeling and then develop both a simplified configuration (in order to attempt a queuing theory analysis) and a discrete event simulation model.

2.1 The Modeled System

In this project we seek to estimate the behavior of an operating system on a busy simple computer using both queuing theory and simulation. The web server has the following components:

1. disk(s) — modeled as I/O server
2. processor(s) — called as CPU server and
3. a network card — modeled as a network server.

Some options to improve performance are being considered, and your boss has asked you to help select a cost effective solution for improving the system performance.

Your server has 120 users, each user generating a hit on average once every 250 seconds (exponentially distributed). This computer has two resources an I/O server and a CPU (assume out-bound network traffic is handled using
a very fast network card, which is omitted from the model for simplicity). On the computer there is room for up
to $N = 24$ server hits (due to memory constraints). Each server hit causes the creation of an associated process.
When a server hit finishes the web page is queued for transmission via the network (in the network card) and the
process terminates. The server has been modified from the version where it ran out of memory, to “drop” hits,
that is if servicing a hit would force $n > N$ then the server discards the request (and waits for the user to try
again).

Of the current $n < N$ server hit generated processes, $k = \frac{n}{25}$ processes are dynamic XML search engine requests
and require a larger ratio of computation to I/O utilization, while the other $n - k = \frac{24n}{25}$ server hits are direct
lookups requiring a higher I/O to CPU utilization. Users are willing to wait for search engine results but want
fast direct lookups. The company is looking into making a server for mobile clients, to make the data more
accessible to users at remote locations, and are interested in predicting the power consumption under various
load conditions.

For simulations you are to model a closed system as shown in Figure 1. For analytical approximations you are

**An Open Queueing System with Feedback**

![An Open Queueing System with Feedback](image)

**Figure 1: The System being modeled**

to model an open system as shown in Figure 2.1.

The parameters for the system are given in Table 1.

**An Open Queueing System**

![An Open Queueing System](image)
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
<th>Derivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha_{CPU}$</td>
<td>Weighting for CPU History Aging</td>
<td>Tuning Parameter</td>
</tr>
<tr>
<td>$H_D$</td>
<td>Number of dropped server hits</td>
<td>Measured</td>
</tr>
<tr>
<td>$k$</td>
<td>Mean number of Search Engine Jobs In System</td>
<td>$25$ jobs</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>Arrival Rate</td>
<td>Computed $\frac{1}{\lambda} = \frac{12}{25}$</td>
</tr>
<tr>
<td>$n$</td>
<td>Number of Server Hits being Processed</td>
<td>Measured</td>
</tr>
<tr>
<td>$N$</td>
<td>Max Number of Server Hits In System at a time</td>
<td>$24$</td>
</tr>
<tr>
<td>$P_{b, CPU}$</td>
<td>Power consumption of CPU when busy</td>
<td>$4W$</td>
</tr>
<tr>
<td>$P_{i, CPU}$</td>
<td>Power consumption of CPU when idle</td>
<td>$1W$</td>
</tr>
<tr>
<td>$P_{b, IO}$</td>
<td>Power consumption of I/O Device when busy</td>
<td>$0.625W$</td>
</tr>
<tr>
<td>$P_{i, IO}$</td>
<td>Power consumption of I/O Device when idle</td>
<td>$0.015W$</td>
</tr>
<tr>
<td>$P_{b, NET}$</td>
<td>Power consumption of Network Device when busy</td>
<td>$1.5W$</td>
</tr>
<tr>
<td>$P_{i, NET}$</td>
<td>Power consumption of Network Device when idle</td>
<td>$1W$</td>
</tr>
<tr>
<td>$\rho_{CPU}$</td>
<td>CPU Utilization</td>
<td>Measured</td>
</tr>
<tr>
<td>$\rho_{IO}$</td>
<td>I/O Server Utilization</td>
<td>Measured</td>
</tr>
<tr>
<td>$\rho_{NET}$</td>
<td>NET Server Utilization</td>
<td>Measured</td>
</tr>
<tr>
<td>$s_{CPU}$</td>
<td>CPU Service Time for Search Engine Jobs</td>
<td>$37.6$ sec. exponentially distributed</td>
</tr>
<tr>
<td>$s_{IO}$</td>
<td>I/O Service Time</td>
<td>$0.1$ sec. exponentially distributed</td>
</tr>
<tr>
<td>$s_{NE}$</td>
<td>Network Service Time</td>
<td>$2.01$ sec. exponentially distributed</td>
</tr>
<tr>
<td>$t_{CPU}$</td>
<td>Time in CPU Queue</td>
<td>Measured</td>
</tr>
<tr>
<td>$t_{IO}$</td>
<td>Time in I/O Queue</td>
<td>Measured</td>
</tr>
<tr>
<td>$t_{NET}$</td>
<td>Time in Network Queue</td>
<td>Measured</td>
</tr>
<tr>
<td>$t_l$</td>
<td>Mean Turnaround Time for Direct Lookup Hits</td>
<td>Measured</td>
</tr>
<tr>
<td>$t_S$</td>
<td>Mean Turnaround Time for Search Engine Hits</td>
<td>Measured</td>
</tr>
<tr>
<td>$T_{cs}$</td>
<td>context switch duration</td>
<td>$10 \mu$sec. constant</td>
</tr>
<tr>
<td>$T_Q$</td>
<td>quantum size only (for RR)</td>
<td>Tuning Parameter</td>
</tr>
<tr>
<td>$T_{IIO}$</td>
<td>Direct Lookup Job’s mean time till next I/O</td>
<td>$0.04$ sec. exponentially distributed</td>
</tr>
<tr>
<td>$T_{SIO}$</td>
<td>Search Engine Job’s mean time till next I/O</td>
<td>$0.76$ sec. exponentially distributed</td>
</tr>
<tr>
<td>$\tau$</td>
<td>Mean Interarrival Time</td>
<td>$\frac{12}{N}$ sec.</td>
</tr>
<tr>
<td>$\tau_{IO}$</td>
<td>Mean I/O Interarrival Time</td>
<td>Measured</td>
</tr>
<tr>
<td>$w_{CPU}$</td>
<td>CPU Queue Length</td>
<td>Measured</td>
</tr>
<tr>
<td>$w_{IO}$</td>
<td>I/O Queue Length</td>
<td>Measured</td>
</tr>
<tr>
<td>$w_{NET}$</td>
<td>I/O Queue Length</td>
<td>Measured</td>
</tr>
</tbody>
</table>

Table 1: Parameters Used in Project 1
2.2 A Sample discrete event simulation

The instructor has implemented a simple discrete event simulator of the package in C. You may use this package as a basis for your discrete event simulation, or if you like you may implement your own. Be careful to to adhere to the command line interface specifications given.

The source code is available for download from the lab related stuff directory. The source code is driver.c and the makefile is makedriver. To build your own copy, you should copy both driver.c and makedriver to your personal working directory and issue the command:

```
make -f makedriver
```

2.3 Goals of the Project

This project involves the following:

1. Computing an analytical approximation of the performance using queuing theory (See the lecture notes). You will be permitted to make some simplifying assumptions (but not too many, or your approximation will be weak).

2. Extending the existing simulator (or writing your own) to support both direct lookup server hits and search engine hits.

3. Measuring power consumption of the simulated system (in Watt hours).

4. Simulating the system with assorted queuing disciplines for $q_{CPU}$, treating queuing discipline as a performance tuning parameter. You should implement FIFO and one of the following based on the first letter of your family name:
   - Round Robin (RR) – A through H
   - Shortest Remaining Time (SRT) — I through Q
   - Earliest Deadline First (EDF) — R through Z

Some of the queueing discipline specific details follow:

(a) Time Sliced Round Robin (RR) — If a job voluntarily leaves the CPU to perform an I/O service, it should be given a full time slice during its next CPU service.

(b) Shortest Remaining time (SRT) — In this system, we estimate how much CPU time is used by each individual process before its next I/O. Consider a particular job, say the $j$th job, it will iterate through the CPU and I/O loop many times. The $i$th iteration through the CPU of this, the $j$th job can have its CPU service time estimated as follows:

$$s_{E}(j, i + 1) = \alpha_{CPU}s_{CPU}(j, i) + (1 - \alpha_{CPU})s_{E}(j, i)$$

where $s_{CPU}$ is the job’s time till next I/O, which is exponentially distributed, the mean dependent on the request type. This follows the approach in the lecture notes.

(c) Earliest Deadline First (EDF) — Assume soft real time scheduling, with preemption. Be sure to remove jobs exceeding their deadline from the system when a deadline violation is detected. The deadline for direct lookup requests are 8 seconds after arrival, for search engine request 80 seconds after arrival. Be sure to track the number of deadline violations.

Suppose that your boss is willing to spend $6000 and can have one of the following done:
1. You can upgrade to a CPU which runs faster, and services jobs in 66% of the time that the original CPU required (it also uses 15% more power, i.e. increase $P_{i, IO}$ and $P_{b, IO}$ when applying this option). Recall that context switching becomes 66% faster with this option.

2. You can upgrade to a faster I/O server with service times 80% of the original I/O server required (it needs 10% more power, i.e. increase $P_{i, IO}$ and $P_{b, IO}$ when applying this option).

3. You can upgrade your network card (i.e. Network server) with service times 70% of the original I/O server required. The new card draws 20% more power, so increase $P_{i, NET}$ and $P_{b, NET}$

If you can show that none of the single upgrades work, you can get authorization to do 2 upgrades for 8,000, which two upgrades would you need? Changing queuing discipline is considered free (but permanent).

**For the Analytical Model only** you will need to estimate both the mean estimated cpu service time, $s_{CPU}$, and the mean interarrival time for I/O requests, $\tau_{IO}$. The following estimates are given:

\[
s_{CPU} = \frac{24s_{1CPU} + s_{SCPU}}{25} = 2.0\text{sec.} \quad (2)
\]

\[
\tau_{IO} = \frac{24T_{1IO} + T_{SIO}}{25\rho_{CPU}} = \frac{0.1\text{sec.}}{\rho_{CPU}} \quad (3)
\]

## 3 Hard Copy Deliverables

Your report should have the following sections:

1. Graphs (from the analytical approximation):
   - (a) $t_{cpu}$ as a function of $\tau$.
   - (b) $t_{io}$ as a function of $\tau$.
   - (c) $t_{net}$ as a function of $\tau$.

Recall that since the components in the system we are have fixed speeds, the mean service times and number of servers of each type can be treated as constant for each plot. Plots should be done for estimating the performance of the system to model the effects of faster components as well. Don’t forget that if you speed up the CPU the frequency of I/O instructions may need to be recalculated.

2. Your boss wants to know how much power these devices consume over a 4 hour interval (one of the customers asked for it, and he wants to know the battery/fuel cell requirements to see if it is feasible). For each solution considered, simulate 4 hours of running the web server (14400 seconds) and make a plot with one histogram of the power consumed per schedule/hardware configuration considered.

3. Compare and contrast the simulated performance with the results obtained from the analytical model. In particular measure how close the FCFS queuing theory model is to the simulation results of each of the various scheduling disciplines.

4. Compare and contrast the simulated throughput of different scheduling algorithms. Which scheduling algorithm/tuning parameter combination gives:
   - (a) the greatest throughput,
   - (b) the minimum response time,
   - (c) the minimum power consumption
   - (d) Schedule specific measures:
i. Earliest Deadline First — the number of deadline violations, assuming immediate preemption of the CPU for higher priority jobs. Assume that the deadline for direct lookup requests is 8 seconds and that for search engine requests have a deadline of 80 seconds.

ii. Round Robin — find a good $T_Q$ (use binary search for about 10 iterations).

iii. SRT — Find a good $\alpha$ value (using binary search for about 10 iterations).

4 Electronic Deliverables

Electronic submission should be done using the submission script described on the FAQ page. Any binaries or scripts you create should compile and run on eve.albany.edu. The following items should be submitted electronically:

1. A file named `README` which contains:
   (a) A list of files submitted
   (b) How to compile any executables you might have
   (c) How to run your simulations

2. The source code for your queuing theory analysis (If you cannot get it into ASCII text format, please print this out and submit as a hard copy).

3. The makefile for your simulation

4. Any data files or scripts needed to run your simulation

5. Source Code For your Simulation