Final Exam for Capacity Planning (CIS 4930/6930)

Spring 2001

>>> *SOLUTIONS* <<<

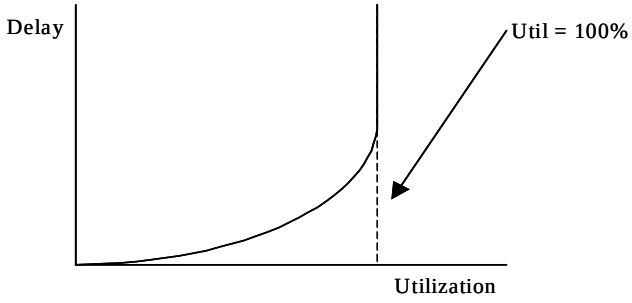
Welcome to the final exam in *Capacity Planning* (CIS 4930/6930). You have 120 minutes. Read each problem carefully. There are ten required problems (each worth 10 points) and one extra credit problem worth 10 points. You may have with you a calculator, pencils, erasers, blank paper, lucky rabbit's foot, and one 8.5 x 11 inch "formula sheet". On this formula sheet you may have anything you want (definitions, formulas, etc.) *handwritten by you.* You may use both sides. Computer generated text, photocopies, and scans are not allowed on this sheet. Please submit your formula sheet with your exam. Please start each numbered problem on a new sheet of paper and do not write on the back of the sheets (I really do not care about saving paper!). Submit everything in problem order. No sharing of calculators. Good luck and be sure to show your work!

Problem #1 (10 minutes)

a) What is capacity planning? Identify the key inputs, methods, and outputs of capacity planning.

Capacity planning is a process for determining the saturation point for computer systems. Capacity planning is also used to determine what can be done (i.e., alternatives) when saturation is reached. The inputs to capacity planning are workload evolution (the inputs to the system), system parameters (the system itself), and the desired service level (e.g., 99% of all transactions must complete in 2 seconds or less).

b) Sketch the delay (response time) behavior of a "typical system". The X-axis is offered load (utilization) and the Y-axis is response time. Carefully identify any asymptotes.



c) Why study performance? Give four reasons (we discussed at least six reasons in class).

1) Performance is a fundamental characteristic of a system. To fully understand system, need to understand performance
2) Cost of hardware may not be important, but being able to provide a guaranteed level of service is increasingly important. This is a capacity planning problem.
3) For leading edge applications, performance is always important
4) For small systems (e.g., embedded systems) performance is an issue due to limited space, power, etc.
5) Performance is subject to "market trick". Need to be knowledgeable to be a smart consumer.
6) Performance evaluation is a key "toolkit" for lots of CS research

Problem #2 (10 minutes)

Answer the following questions about arrival processes, workload generation, and queueing.

a) What is the *relationship* between a Poisson process and the exponential distribution (you do not need to prove, just state)?

The time between Poisson arrivals is exponentially distributed with the same rate parameter, λ.

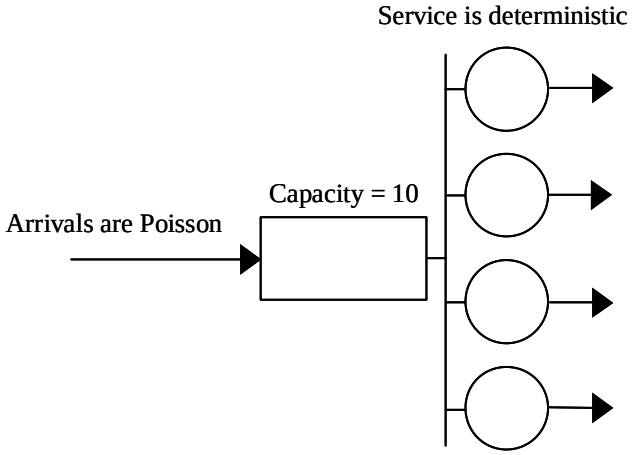
b) For Poisson arrivals of rate $\lambda = 5$ per second, what is the probability of have 6 arrivals in 2 seconds?

$$f(x) = \Pr[X = x] = \left(\frac{(\lambda t)^x}{x!} \right) e^{-\lambda t}, \quad \text{so...} \quad \Pr[X = 12] = \left(\frac{(5 \cdot 2)^6}{6!} \right) e^{-5 \cdot 2} = 0.063$$

c) Give four attributes of a good workload model.

Here are five attributes (need only four):
1) Parsimonious (few parameters)
2) Easily generated
3) Easily tunable
4) Tractable (if for analytical modeling)
5) Of course, accurate representation of real workload

d) Sketch an M/D/4/10 queue. Carefully label your sketch.



e) At 90% offered load (utilization), what is the mean queue length of a D/D/1 queue? What is the mean queue length of an M/M/1 queue (again, for 90% offered load)? Note that queue length does not include customers in service, only those in the "buffer".

For D/D/1 there is never a queue for a stable system (so $L_q = 0$). For M/M/1, $L_q = \frac{\rho}{1-\rho} - \rho$ = 8.1 customers.

Problem #3 (15 minutes)

Here are some measurements (e.g., of web server response time in milliseconds)...

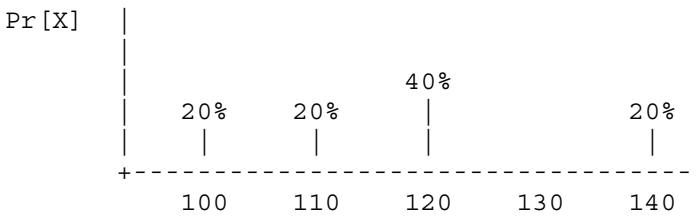
110, 120, 140, 120, 120, 100, 110, 100, 140, 120

a) Compute the mean and variance of this population of measurements

$$E[X] = \frac{1}{10}(110 + 120 + 140 + 120 + 120 + 100 + 110 + 100 + 140 + 120) = 118$$

$$\sigma^2 = \frac{1}{10}(110^2 + 120^2 + 140^2 + 120^2 + 120^2 + 100^2 + 110^2 + 100^2 + 140^2 + 120^2) - 118^2 = 176$$

b) Plot a histogram for the measurements.



c) Write a C function that will return a value from an empirical distribution based on the above measurements. About a half-dozen lines of code should do the trick.

```
#include <stdio.h>    // Needed for printf()
#include <stdlib.h>   // Needed for ran() and RAND_MAX

int emp(void);

void main(void)
{
  printf("%d \n", emp());
}
```

```
int emp(void)
{
  double z;         // unif(0,1)

  z = (double) rand() / RAND_MAX;

  if (z <= 0.20) return(100);
  if (z <= 0.40) return(110);
  if (z <= 0.80) return(120);
  return(140);
}
```

Problem #4 (15 minutes)

Answer the following questions about client/server and sockets.

a) What does the bind() sockets function do?

The bind function "gives a socket a name". The bind() function associates an IP address and port number with a socket (e.g., as previously created with a socket() call).

b) When we say that a sockets function "blocks", what do we mean? Give an example of a sockets function that blocks. How can blocking be (intelligently) handled in an application?

A function that blocks does not return until it has completed and hence stops or "blocks" the calling program from execution until it is completed. This blocking may take several seconds for a connect() or accept() function (two examples of blocking calls). The use of separate processes or threads to handle functions the blocks prevents the entire program from stopping execution due to a single blocking function call.

Problem #5 (10 minutes)

Answer the following questions about web servers and benchmarks.

a) Give the high-level steps (or flowchart) for how a web server works. Your description need only handle GET requests.

The basic steps of a web server are:
1) Wait for a connection
2) Make the connection (for an incoming connection request)
3) Receive an HTTP command and parse it
4) Respond with the requested file (the file name specified in the GET command)
5) Drop the connection

b) Identify two design and/or implementation "hot-spots" (key areas) that affect web server performance.

Spawning and killing threads or processes is expensive in time. File I/O (opening/closing files, reading from disk) is also expensive.

c) What is the purpose of a benchmark? Name and briefly describe two commercial or "well known" benchmarks or benchmark suites (other than SURGE).

The purpose of a benchmark is to offer comparative performance measures across systems. Two well known benchmarks are the SPEC benchmarks for processor performance and TPC for transaction performance.

d) What are the two key ideas embodied in the SURGE benchmark by Paul Barford.

Two keys ideas are 1) request rates are a function of user response delay (user response delay does not change), and 2) web objects are heavy tailed.

Problem #6 (10 minutes)

Answer the following questions about scaling.

a) State Amdahl's Law.

$Speed-up = \frac{q}{(1+\alpha\cdot(q-1))}$ where q is the number of processors and α is the percentage of the program that cannot be parallelized (i.e., must be run in serial).

b) Given an 8 processor system and an application in which 99% of the run-time can be parallelized, give the speed-up using Amdahl's Law. Repeat for a 16 processor system.

We have that 0.01 (1%) of the application must be executed in serial. So, speed-up = 8 / (1 + 0.01*(8-1)) = 7.48 for an 8 processor system and speed-up = 16 / (1 + 0.01*(16 - 1)) = 13.91.

c) Describe at a high level how the Akamai service works OR give two methods for building a web server cluster.

For an Akamized web site, the origin server serves the HTML, but the links for embedded images in the returned HTML file point to Akamai owned servers that are closer to the client that the origin server. Thus, the "heavy lifting" of delivering images and other embedded stuff is done by the Akamai servers reducing load on the origin site.

One method for building a web server cluster is to use a rotating DNS (i.e., resolving the web site name to multiple IP addresses, each IP address a mirrored server). Another method is to "spoof" address where a single dispatcher box receives all requests and then forwards them to servers of different IP addresses. The dispatcher box has a single IP address (the IP address for the web site) and maps (or spoofs) the IP addresses of the separate servers for each request.

Problem #7 (10 minutes)

Answer the following questions about modeling and random number generation.

a) What is a model?

"A model is a representation (physical, logical, or functional) that mimics another object under study." (Molloy, 1989)

b) Why build a model and not just experiment with the actual system of interest?

Building and measuring a model can be cheaper, easier, faster, and/or safer than building and measuring an actual system.

c) Give the four desired properties of a random number generator.

Desired properties are:
1) Numbers generated should be uniformly distributed and be independent.
2) Generator should be fast and not require much storage
3) Random numbers stream should be reproducible (e.g., for debugging purposes)
4) Generator should have provision to produce several separate (and independent) streams of random numbers.

Problem #8 (10 minutes)

Answer the following questions about simulation.

a) Give the flowchart or pseudocode for a discrete event simulation program.

```
Initialize global state variables
Initialize global time to zero
Obtain the first input event
While (not yet done)
{
  Remove next event from event list
  Advance time to time of this event
  Perform event routine for the event type of this event
  Update global variables
  Generate next event triggered by this event
}
Generate report
```

b) Describe an event list and identify what variables each event structure (i.e., the objects in the event list) must contain at a minimum. Explain what operations must be performable on the event list.

An event list is a linked list with event structures that contain, at a minimum, the event number and its time of occurrence. A simulation program must be able to remove the head event from the list and insert an event into an arbitrary location in the list based on a time ordering of the list.

Problem #9 (15 minutes)

On the last page of this exam is `f2.c`, a small SMPL simulation program. Give the output for this program.

```
time=0.00 event=1 cust_id=1
time=0.00 event=1 cust_id=2
time=0.00 event=2 cust_id=1
  Inside 'if' at time=0.00
time=0.00 event=2 cust_id=2
```

```
time=10.00 event=3 cust_id=1
time=10.00 event=2 cust_id=2
  Inside 'if' at time=10.00
time=18.00 event=1 cust_id=3
time=18.00 event=2 cust_id=3
time=20.00 event=3 cust_id=2
time=20.00 event=2 cust_id=3
  Inside 'if' at time=20.00
time=30.00 event=3 cust_id=3

**** Simulation Error at Time 30.000
     Empty Event List
```

Problem #10 (15 minutes)

Assume a system with three states, UP, DOWN, and DIAGNOSE (call them states 0, 1, and 2 respectively). Assume that the states are memoryless and transitions occur at embedded time instants every 1 minute (i.e., this is a discrete time system). The transition probabilities are: 0-to-0 = 0.80, 0-to-1 = 0.10, 0-to-2 = 0.10, 1-to-0 = 0.40, 1-to-1=0.50, 1-to-2 = 0.10, 2-to-1 = 0.80, 2-to-2 = 0.20. If an observer arrives at a random time, what is the probability that the system is in the DIAGNOSE state?

$$P = \begin{bmatrix} 0.80 & 0.10 & 0.10 \\ 0.40 & 0.50 & 0.10 \\ 0.00 & 0.80 & 0.20 \end{bmatrix}$$

We solve for $\pi 0$, $\pi 1$, and $\pi 2$ in (note substitution of $1 = \pi 0 + \pi 1 + \pi 2$ for the last equation to break linear dependence).

$$\pi 0 = 0.80\,\pi 0 + 0.40\,\pi 1 + 0.00\,\pi 2$$
$$\pi 1 = 0.10\,\pi 0 + 0.50\,\pi 1 + 0.80\,\pi 2$$
$$1 = 1.00\,\pi 0 + 1.00\,\pi 1 + 1.00\,\pi 2$$

Rewriting...

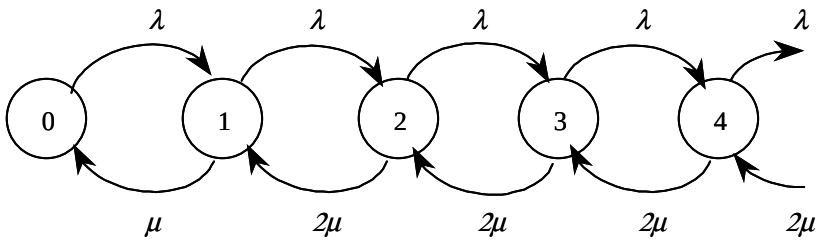
$$0 = -0.20\,\pi 0 + 0.40\pi 1 + 0.00\,\pi 2$$
$$0 = 0.10\,\pi 0 + -0.50\,\pi 1 + 0.80\,\pi 2$$
$$1 = 1.00\,\pi 0 + 1.00\,\pi 1 + 1.00\,\pi 2$$

Which solves to $\pi 0 = 16/27$, $\pi 1 = 8/27$, $\pi 2 = 1/9$. So, the probability that the system is in the DIAGNOSE state (state 2) in steady state (a random observer) is 1/9.

Extra Credit

Derive L for an M/M/2 queue. Full credit will be given for reaching a point at which Mathcad, or other symbolic math solver, can takeover.

The Markov chain (for a service rate of μ for each service center and $\lambda < 2\mu$ for stability) is…



We can write…

$$\pi_n = \begin{cases} \frac{(2\cdot\rho)^n}{n!}\pi_0 & \text{n} = 1,2 \\ \frac{(2\cdot\rho)^n}{2!\cdot 2^{n-2}}\pi_0 & \text{n} > 2 \end{cases}$$

We then solve for π_0 as…

$$\pi_0 = \frac{1}{\sum_{n=0}^{1}\frac{(2\cdot\rho)^n}{n!} + \frac{(2\cdot\rho)^2}{2!\cdot(1-\rho)}}$$

And, then to solve for L as (see also pages 234 of Lilja for this)…

$$L = \sum_{n=0}^{\infty} n\cdot\pi_n = \frac{(2\cdot\rho)^2\cdot\rho\cdot\pi_0}{2!\cdot(1-\rho)^2} + 2\cdot\rho$$

```
//============================================================ file = f2.c =====
//= Sample SMPL program for final exam (Spring 2001)                         =
//=============================================================================
#include <stdio.h>      // Needed for printf()
#include "smpl.h"       // Needed for SMPL

void main(void)
{
  int  cust_id;        // Customer id
  int  event;          // Event id
  int  server = 1;     // Server handle

  smpl(0, "f1.c");
  server = facility("server", 1);

  schedule(1, 0.0, cust_id = 1);
  schedule(1, 0.0, cust_id = 2);

  while (1)
  {
    cause(&event,&cust_id);

    switch(event)
    {
      case 1:
        printf("time=%4.2f event=%d cust_id=%d \n", time(), event, cust_id);
        schedule(2, 0.0, cust_id);
        break;

      case 2:
        printf("time=%4.2f event=%d cust_id=%d \n", time(), event, cust_id);
        if (request(server, cust_id, 1) == 0)
        {
          schedule(3, 10.0, cust_id);
          printf("  Inside 'if' at time=%4.2f \n", time());
        }
        break;

      case 3:
        printf("time=%4.2f event=%d cust_id=%d \n", time(), event, cust_id);
        release(server, cust_id);
        if (cust_id == 1) schedule(1, 8.0, cust_id = 3);
        break;
    }
  }
}
```