Network Performance, GZ02 and M033, 2010/2011
Answer ALL questions from part A and ANY TWO questions from part B
Marks for each part of each question are indicated in square brackets
Calculators are permitted

Part A

I want to design circuitry for a Internet router to inspect packet headers and count the total number of flows over some period of time. I expect to see many flows, but I am operating under very stringent memory restrictions and I only have enough space to store a few floating point numbers, so I cannot simply keep track of all the flow IDs in the packets I have seen. Instead, I will use the following approximate method:

Each time a packet arrives, I compute a hash U of its flow ID, which gives me a number uniformly distributed in the range $[0,1]$. Obviously, if two packets have the same flow ID, they have the same hash value. I then compute $X = -\log U$. I then find the minimum value M of all the values X I see in the measurement interval. I do this process simultaneously for nine different hash functions, giving me nine different minimums $M_1, \dots, M_9$. Finally I compute the average $L = (M_1 + \cdots + M_9)/9$.

1. Show that X has an Exponential distribution with mean 1. Supposing the packets in fact come from n different flow IDs, find the distribution of M and show your reasoning.

   [10 marks]

2. What is the mean and standard deviation of M? Show that the mean of L is $1/n$, and find its standard deviation.

   [10 marks]

3. Since the mean of L is $1/n$, I shall use $1/L$ as an estimate of n. I actually observed $L = 0.001$, which tells me $n \approx 1000$. Give a 95% confidence interval for n.

   [10 marks]

Part B

Questions 4 and 5 both concern a simple model for a web server. However, the two questions are independent, and you may answer one without answering the other.

The web server has a pool of threads to work on page requests, and a single-core CPU. Requests arrive at the web server and are stored until a thread becomes available. We will assume that the web server is heavily loaded, so that there are always requests waiting for a thread. There is a queue of threads waiting to execute, and requests that have newly been assigned a thread are put at the back of this queue. When the CPU starts executing the thread at the front of the queue, it first spends some time doing a context switch, and then it executes for some additional time, and then it moves on to the next thread in the queue. The CPU only works on one thread at a time. The thread may either complete (finish work on a request), or be blocked (waiting for external IO before it can continue working on the request). If the thread completes it is returned to the pool and is availabe to serve a new request. If the thread is blocked it waits some time for external IO, then it joins the back of the queue.

Let the constants describing this system be as follows:

- θ, the number of threads in the thread pool;
- c, the number of instructions per second that the CPU can execute;
- a, the average number of instructions per request (not including context switching);
- p, the probability that a thread completes when it is served rather than being blocked;
- a_0, the average number of instructions per context switch;
- b, the average duration for which a thread is blocked.

4. Consider the web server described above. Let the state be the pair (N, S), where N is the number of blocked threads, $S = 0$ if the CPU is context-switching, $S = 1$ if the CPU is working on a request, and $S = *$ if the CPU is idle. The state space diagram is as follows:

$$\begin{tikzcd}
(0,0) \arrow[dd] & (1,0) \arrow[l] \arrow[dd] & & (n,0) \arrow[l, "w"'] \arrow[dd, "x"'] & & (\theta\text{-}1,0) \arrow[l] \arrow[dd] & \\
 & & \cdots & & \cdots & & (\theta,*) \arrow[ul] \\
(0,1) \arrow[uur] & (1,1) \arrow[l] \arrow[uur] & & (n,1) \arrow[l, "y"] \arrow[uur, "z"'] & & (\theta\text{-}1,1) \arrow[l] \arrow[ur] &
\end{tikzcd}$$

(a) Find formulae for the transition rates w, x, y and z. Explain your reasoning carefully. State any assumptions you are making about the distributions of random variables.

(b) Assuming that N never hits 0 or θ, draw a state space diagram for S and calculate its equilibrium distribution.

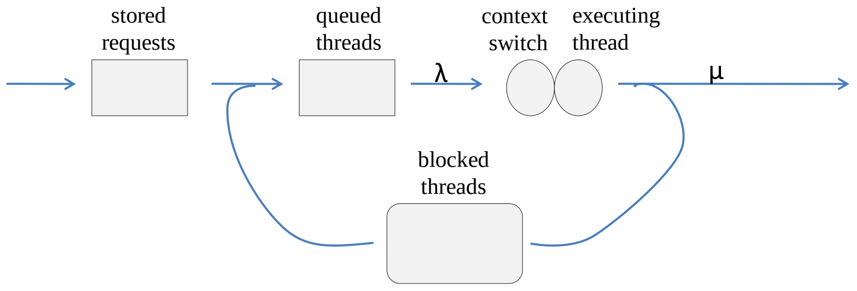
It is difficult to find an exact formula for the equilibrium distribution of the full state space diagram, so instead we will produce an approximate state space diagram for N and find its equilibrium distribution. Let π_0 be the equilibrium probability that $S = 0$, which you calculated in part (b). In the approximate state space diagram for N, let there be transitions from n to $n+1$ at rate $(1-\pi_0)z$, and from n to $n-1$ at rate $\pi_0 w + (1-\pi_0)y$, with an appropriate modification for $n = \theta$.

(c) What is the equilibrium distribution for N? What is the fraction of time that the CPU spends idle? How big should θ be, to ensure that the CPU rarely goes idle?

[35 marks]

5. (a) Explain briefly what is meant by the Erlang link model. State Little's law. Using Little's law, find a formula for the average number of active calls on an Erlang link.

The web server described at the beginning of Section B may be represented schematically as follows. Here λ is the rate at which threads start being processed, and μ is the overall rate at which requests are completed.

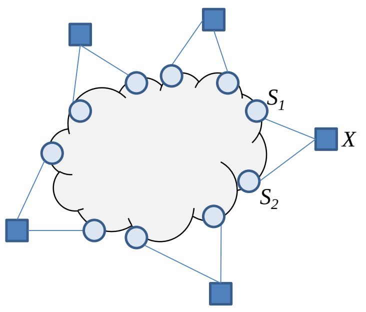


(b) Let i be the fraction of time the CPU spends idle. Explain why $\lambda(a_0 + ap) = c(1-i)$. Compute μ, the overall rate at which the web server can handle requests.

(c) Let n be the average number of threads queued up and waiting to execute. Calculate the average queueing delay in terms of n. (It is possible to find a formula for n in terms of the system parameters, but for this question you should just take n as given.) Hence calculate the average time it takes for a request to be completely served, once it has been assigned to a thread.

[35 marks]

6. The diagram below shows the structure of a dual-homed telephone network. Each exchange (X) is connected to two core switches (S_1 and S_2). A call from X to another exchange may go through either S_1 or S_2, and it may go through either core switch for the destination exchange, giving a total of four possible paths. Each core switch can handle a maximum of C concurrent calls.



When a call arrives to the network, it tries one of the four paths at random. If both of the core switches on that path have space free, the call uses that path. Otherwise it picks one of the remaining three paths at random, and so on, until it has either succeeded or been blocked on all four paths.

Let B be the probability that a given core switch is full. Let λ be the rate at which calls arrive at each exchange, and let m be the average call duration.

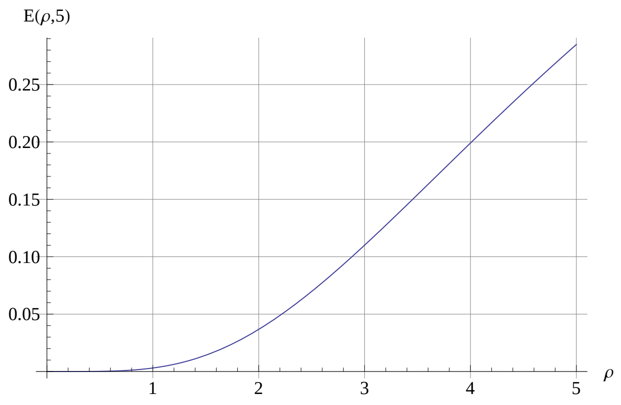
(a) Find an expression for the overall call blocking probability, in terms of B.

(b) Show that B satisfies the fixed point equation

$$B = E\left(\frac{3\lambda}{2m}(1+B^2-B^3), C\right),$$

where you should define the function E.

(c) Compute B numerically, for $C=5$, $\lambda=3$ and $m=1$. Explain your method carefully. You may find helpful this plot of $E(\rho,5)$ as a function of ρ.



[35 marks]

7. (a) The TCP drift model is

$$\frac{dw_t}{dt} \approx \frac{1}{\mathrm{RTT}} - p\frac{w_t^2}{2\mathrm{RTT}}$$

and the TCP throughput equation is $x \approx \sqrt{2}/(\mathrm{RTT}\sqrt{p})$. Define the terms in these equations, and explain briefly how they are derived.

With conventional congestion control, a flow's window size is cut by half whenever a packet is dropped, and packets are dropped when the queue is full. It has been suggested that it would be better if the queue gave more fine-grained feedback than just 'full' versus 'not full'. In this question, we consider the following scheme:

Fix some queue threshold q_0. Suppose a packet arrives and finds the queue size to be Q. If $Q \leq q_0$ then simply forward the packet and do nothing. If $Q > q_0$ then mark the packet with value $M = Q - q_0$. When a source receives an unmarked packet, it increases its window size just as with regular TCP. When a source receives a packet marked with value M, it decreases its window size w by $\beta w M$ where β is some predefined constant. In this way, the more congested the link, the greater the backoff.

(b) Derive a drift model for this congestion control scheme, using an $M/M/1$ model for the queue. Explain your reasoning carefully. Show that the throughput formula is

$$x = \frac{1}{\mathrm{RTT}}\sqrt{\frac{1-p^{1/q_0}}{\beta p^{1+1/q_0}}}.$$

Hint. In answering this question, you may find useful the following results about an $M/M/1$ queue with traffic intensity ρ. Let Q be the equilibrium queue size. Then

$$\mathbb{P}(Q > q_0) = \rho^{q_0} \quad \text{and} \quad \mathbb{E}(Q - q_0 \,|\, Q > q_0) = \frac{\rho}{1-\rho}.$$

[35 marks]

END OF PAPER