Midterm 1

Graded

Student

Jae Hong Lee

Total Points

96 / 108 pts

Question 1

Problem 1 - True/False

11 / 14 pts

(a)

+ 2 pts (a) Correct

✔ **− 1 pt** (a) Incorrect

+ 0 pts (a) Missing

(b)

✔ **+ 2 pts** (b) Correct

− 1 pt (b) Incorrect

+ 0 pts (b) Missing

(c)

✔ **+ 2 pts** (c) Correct

− 1 pt (c) Incorrect

+ 0 pts (c) Missing

(d)

✔ **+ 2 pts** (d) Correct

− 1 pt (d) Incorrect

+ 0 pts (d) Missing

(e)

✔ **+ 2 pts** (e) Correct

− 1 pt (e) Incorrect

+ 0 pts (e) Missing

(f)

✔ **+ 2 pts** (f) Correct

− 1 pt (f) Incorrect

+ 0 pts (f) Missing

(g)

+ 2 pts (g) Correct

✔ **− 1 pt** (g) Incorrect

+ 0 pts (g) Missing

(h)

+ 2 pts (h) Correct

✔ − 1 pt (h) Incorrect

+ 0 pts (h) Missing

(i)

✔ + 2 pts (i) Correct

− 1 pt (i) Incorrect

+ 0 pts (i) Missing

(j)

✔ + 2 pts (j) Correct

− 1 pt (j) Incorrect

+ 0 pts (j) Missing

Question 2

Problem 2 - Code Evaluation **25** / 25 pts

2.1 **a. Find the bug** **6** / 6 pts

- ✔ **+ 5 pts** both bugs identified
- **+ 3 pts** One bug identified
- ✔ **+ 1 pt** with two correct explanations
- **+ 0.5 pts** with one correct explanation
- **+ 0 pts** no bug identified / no sol

2.2 **b. Fix the bug** **6** / 6 pts

- ✔ **+ 6 pts** Both correct fixes
- **+ 3 pts** One correct fix
- **+ 0 pts** no fixes / no sol
- **+ 5 pts** partial credit (correct code wrong line numbers)
- **+ 5 pts** partial credit (minor syntax issue)
- **+ 2 pts** partial credit (minor syntax issue)
- **+ 2 pts** partial credit (correct code wrong line number)

2.3 **c. Server utilization** **6** / 6 pts

- ✔ **+ 6 pts** Correct graph choice + correct util calculation
- **+ 2 pts** Correct graph choice
- **+ 4 pts** Correct util calculation
- **+ 0 pts** incorrect / no sol
- **+ 1 pt** partial credit

2.4 **d. Server response time** **7** / 7 pts

- ✔ **+ 7 pts** Correct
- **+ 3 pts** Correctly identify M/G/1
- **+ 2 pts** correct sum queue
- **+ 2 pts** Correct parameters plugged into formula
- **+ 3 pts** Not using M/G/1 but correct formula
- **+ 1 pt** Correct final numerical solution
- **+ 0 pts** incorrect / no sol

Question 3

Problem 3 - CS350 Textbook **21** / 22 pts

3.1 **a. Expected response time** **5** / 5 pts

- ✔ **+ 5 pts** Correct
- **+ 2.5 pts** Correct formula setup
- **+ 1 pt** Correct final numerical solution
- **+ 2 pts** Partial correct formula setup
- **+ 2 pts** Correctly identify M/M/1
- **+ 0 pts** incorrect / no sol

3.2 **b. Limited queue response time** **4** / 5 pts

- **+ 5 pts** Correct
- ✔ **+ 2 pts** Correctly identify M/M/1/K
- ✔ **+ 2 pts** Correct formula setup
- **+ 1 pt** Partial formula setup
- **+ 1 pt** Correct final numerical solution
- **+ 0 pts** incorrect / no sol

3.3 **c. Amount of busy time** **6** / 6 pts

- ✔ **+ 6 pts** Correct
- **+ 1 pt** Partial formula credit
- **+ 1 pt** correct setup using time (using hrs expression or 60 mins or 3600s, etc)
- **+ 0 pts** incorrect / no sol
- **+ 3 pts** correctly find new utilization or new probability (ie get 0.92)
- **+ 2 pts** correct numerical solution
- **– 1 pt** Minor math error but correct in general

3.4 **d. Doubling service time** **6** / 6 pts

- **+ 6 pts** Correct with explanations
- **+ 1 pt** Correct by guess (no explanation)
- ✔ **+ 3 pts** correct setup for new rejection rate
- ✔ **+ 3 pts** Correct t/f with reasonable explanation and calculations
- **+ 1 pt** correct numerical solution
- **+ 0 pts** incorrect / no sol

Question 4

Problem 4 - PetBuddy System **17** / 22 pts

4.1 **a. Average response time** **2** / 6 pts

+ **6 pts** Correct

+ **2 pts** correctly identify M/D/1

+ **2 pts** correct formula setup for q

✔ + **1 pt** correct formula setup for Tq

+ **1 pt** correct numerical solution

+ **0 pts** incorrect / no sol

✔ + **1 pt** partial credit

4.2 **b. Upgrade to M/M/2** **8** / 8 pts

✔ + **8 pts** Correct

✔ + **2 pts** correctly identify M/M/2

+ **2 pts** recognizing needing to find C / both servers are busy

+ **2 pts** correct setup for K

+ **1 pt** correct setup for C

+ **1 pt** correct numerical sol

+ **0 pts** incorrect / no sol

+ **1 pt** partial credit

4.3 **c. Speedup from upgrade** **7** / 8 pts

+ **8 pts** Correct

✔ + **2 pts** recognizing that C is needed (call or reuse from before)

✔ + **3 pts** correct formula setup for q

✔ + **2 pts** correct formula setup for Tq

+ **1 pt** correct numerical sol

+ **0 pts** incorrect / no sol

+ **1 pt** partial credit

Question 5

Problem 5 - ChatLPT System **22** / 25 pts

5.1 **a. System diagram and assumptions** **5** / 5 pts

✔ **+ 5 pts** Correct

+ 1 pt correct diagram

+ 4 pts 4 (out of 5) correct assumptions

+ 3 pts 3 correct assumptions

+ 2 pts 2 correct assumptions

+ 1 pt 1 correct assumption

+ 0 pts incorrect / no sol

5.2 **b. Bottleneck identification** **4** / 5 pts

+ 5 pts Correct

+ 3 pts All (3) correct setups for λ

✔ **+ 2.5 pts** 2 (out of 3) correct setups for λ

+ 2 pts 1 (out of 3) correct setups for λ

✔ **+ 1.5 pts** correct formula setup for ρ

+ 1 pt correct bottleneck choice

+ 0 pts incorrect / no sol

5.3 **c. End-to-end response time** **4** / 5 pts

+ 5 pts Correct

+ 3 pts 3 (out of 3) correct setup for q

+ 2 pts 2 (out of 3) correct setup for q

+ 1 pt 1 (out of 3) correct setup for q

✔ **+ 2 pts** procedure/setup correct but incorrect value(s)

✔ **+ 1 pt** correct formula for Q_{Total}

✔ **+ 1 pt** correct setup for T_q

+ 0 pts incorrect / no sol

5.4 **d. Speedup with no queuing** **5** / 6 pts

+ 6 pts Correct

✔ + 2 pts correct setup for w

+ 1 pt correct setup for T_w

+ 2 pts correct setup for T_s

+ 1 pt correct numerical solution

✔ + 3 pts correct setup for speedup

+ 1 pt Some attempt, but far from solution

+ 0 pts No solution

5.5 **e. System capacity** **4** / 4 pts

✔ + 4 pts Correct

+ 2 pts correct setup for inequality of ρ

+ 1 pt partial credit

+ 1 pt correct setup for λ

+ 1 pt correct numerical sol

+ 0 pts incorrect / no sol

✔ + 0 pts carry over error -> correct logic

Department of Computer Science BOSTON UNIVERSITY

CS-350 - Fundamentals of Computing Systems

Midterm Exam #1
Fall 2024

Name: Jue Hong Lee

BU Username: jhonglee BU ID: U20565203

NOTE: *Please use only the provided space and the included extra page to answer the questions. If you do use the extra pages, make sure you reference them properly in your solutions. The very last page can be detached for your convenience.*

Problem #1:	/14
Problem #2:	/25
Problem #3:	/22
Problem #4:	/22
Problem #5:	/25
Total Score:	/108

Remarks:

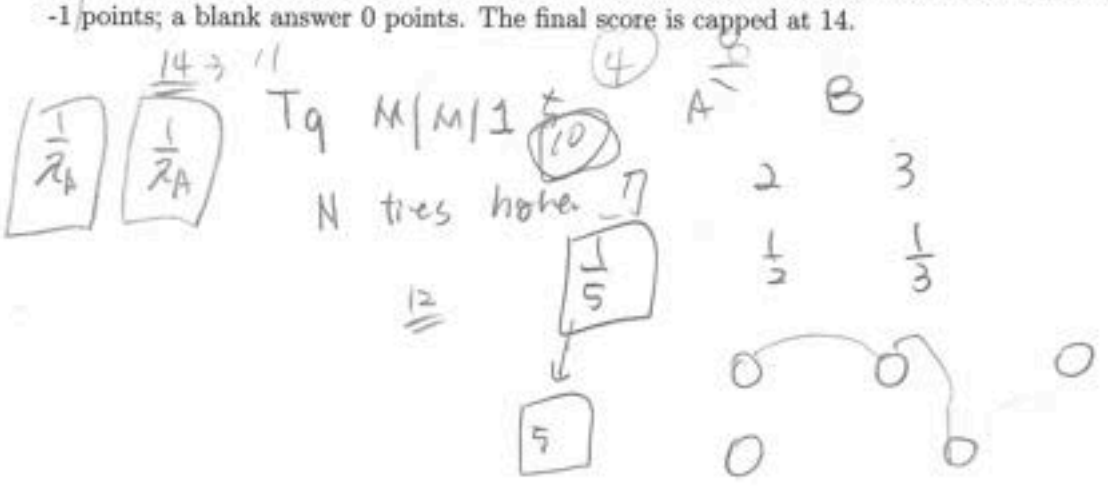
- This is a closed-book/notes exam.
- Basic calculators are allowed.
- You have 80 minutes to complete your exam.
- There are 108 total points.
- If your score is more than 100, it is capped at 100.
- Show your work for full marks.
- Problems and sub-problems weighted as shown.
- **Explain all your assumptions clearly.**

Problem 1

Label each of the statements below with either **True (T)** or **False (F)**:

	Statement	T/F
a.	If two systems are characterized by the same availability, then they can be considered more predictable that two systems with different availability.	T
b.	An M/M/1/K queuing system is always able to reach steady-state operation regardless of the traffic input rate λ.	T
c.	Increasing the multi-programming level (MPL) in a system always leads to an increase in the capacity of the system.	F
d.	Consider two different systems, namely A and B. Even if the capacity of B is twice that of A, it is still possible for B to have a throughput lower than A.	T
e.	If a process is blocked from the perspective of a given the resource, then this process is not currently contributing to the resource utilization.	T
f.	The `pthread_create(...)` function returns only when the thread spawned by the function terminates its execution and exits.	F
g.	The steady-state expected response time in an M/M/1 system is always N times higher compared to an N*M/M/1 system subject to the same input traffic rate λ and identical per-server mean service time T_s.	T
h.	Assume that Poisson-distributed events of type A and B arrive with a mean rate of λ_A and λ_B, respectively. Then, the mean time that elapses between any two events, regardless of their type, is $\frac{1}{\lambda_A+\lambda_B}$.	F
i.	If a system is operating close to its capacity, a very small input traffic increase might lead to a very large increase in the end-to-end response time.	T
j.	Just like an M/M/1 system, an M/D/1 system will be unable to reach steady state if the input traffic rate approaches the system's service rate.	T

Note: There are 10 questions. A correct answer will get you 2 points; an incorrect answer -1 points; a blank answer 0 points. The final score is capped at 14.



Problem 2

ChatGPT generated the following incomplete code of the `handle_connection(...)` implementation for a server that uses a worker child thread to process transactions via a FIFO queue shared with the parent thread. Elements in the queue are added and popped via the `add_to_queue(...)` and `get_from_queue(...)` functions, respectively. When handling a request, the worker thread busywaits for the length of the request, and then responds to the client with the same request ID.

```
1  struct request {
2    uint64_t req_id; /* Integer request ID set by the client */
3    struct timespec req_timestamp; /* time when request sent by client */
4    struct timespec req_length; /* time length of the request */
5  };
6  struct request_meta {
7    struct request request;
8    struct timespec receipt_timestamp; /* time when request enqueued */
9    struct timespec start_timestamp; /* time when request starts service */
10   struct timespec completion_timestamp; /* time when request completed */
11 };
12 struct worker_params {
13   int conn_socket;
14   struct queue * the_queue;
15 };
16 int worker_main (void * arg) { /* Main logic of the worker thread */
17   struct timespec now;
18   struct worker_params * params = (struct worker_params *)arg;
19   while (/* Worker termination condition -- CODE OMITTED */) {
20     struct request_meta req;
21     struct response resp;
22     req = get_from_queue(params->the_queue);
23     clock_gettime(CLOCK_MONOTONIC, &req.start_timestamp);
24     busywait_timespec(req.completion_timestamp);
25     clock_gettime(CLOCK_MONOTONIC, &req.request.req_length);
26     resp.req_id = req.request.req_id;
27     send(params->conn_socket, &resp, sizeof(struct response), 0);
28     printout_completed_request(req);
29     dump_queue_status(params->the_queue);
30   }
31   return EXIT_SUCCESS;
32 }
33 void handle_connection(int conn_socket) { /* Main connection handling logic */
34   struct queue * the_queue;
35   int in_bytes, worker_id;
36   struct request_meta * req = (struct request_meta *)malloc(sizeof(struct request_meta));
37   the_queue = initialize_queue(QUEUE_SIZE);
38   worker_id = start_worker(conn_socket, the_queue);
39   do {
40     in_bytes = recv(conn_socket, &req->request, sizeof(struct request), 0);
41     clock_gettime(CLOCK_MONOTONIC, &req->receipt_timestamp);
42     if (in_bytes > 0) {
43       add_to_queue(*req, the_queue);
44     }
45   } while (in_bytes > 0);
46   /* Terminate worker and shutdown socket -- CODE OMITTED */
47 }
```

Listing 1: Handle connection and worker implementation

(a) [6 points] The code seems to compile without problems, but when you execute it and try to compute the statistics for your EVAL assignment from it output, something seems very off. Sometimes, the server just stalls forever while processing some requests. Moreover, when attempting to plot the service time distribution, the resulting distribution is unrecognizable. Identify the cause of the bug. You can directly refer to the line number of the code line(s) that are problematic. Do your best to explain the behavior of (1) long stalls upon processing some requests, and (2) unexpected service time distribution. (1) (2)

problem:
1) stall forever
2) server distribution uniform

1) In the handle connection function, the worker_main function is not properly called so the worker threads might not be starts property.

Also Even if the worker_main is called properly in then handle_connection, in the line code 24, the busy wait time have to be the time length from the client requested, but instead of "req.request.req_timestamp" it used "req.completion_timestamp so it might took long stalls upon processing some requests.

2) Also in the code line 25, After the busy wait, worker main calls the clock get time and save it to &req.request.req_length instead of client's request, it should be saved to "&req.completion Timestap" to properly find distribution

(b) [6 points] Provide a fix for the code. For each problematic line identified above, write down how the line should read instead. Make sure to indicate which line (identified by its line number) in the original code your line will replace.

for line 24 busywait_timespec (req.request.req_timestamp)

for line 25 clock_gettime (CLOCK_MONOTONIC, &req.completion_timestamp)

&req.completion_timestamp)

(c) **[6 points]** After running a long experiment and post-processing the server's output, you have produced the following plots for the request inter-arrival times (Figure 1) and service times (Figure 2). Apart from the distributions of the experimental data, some theoretical distributions have been added to the plots for your convenience:

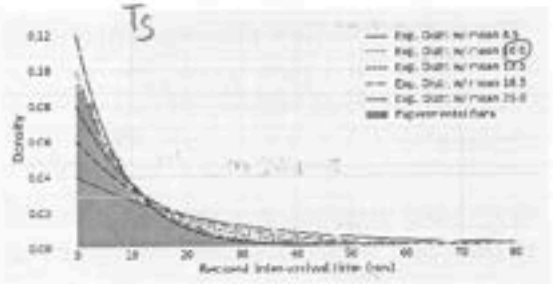


Figure 1: Experimental and theoretical inter-arrival time densities.

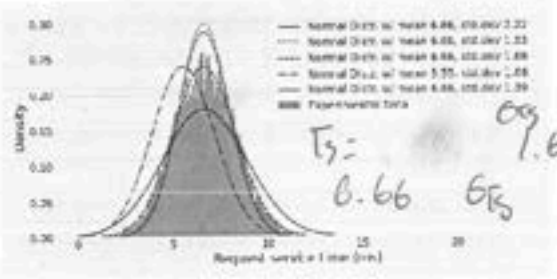


Figure 2: Experimental and theoretical service time densities.

What is the expected utilization of the server at the end of the experiment?

From the figure 1, the closest inter arrival times is mean of 10 which
From the figure 2, the closest distribution of service time is mean 6.66 and std dev 1.66
the inter arrival rate is mean 10 so the $\lambda = \frac{1}{10}$ req/sec
and we know the mean service time is mean 6.66 sec/req
the utilization is $\rho \times T_s = \frac{1}{10} \times 6.66 = 0.666$

$\rho = 0.666$ $\quad \rho = \lambda \cdot T_s$

(d) **[7 points]** Considering the same experimental data provided in Figure 1 and Figure 2, what is the expected response time for a generic request arriving at the server during the experiment?

M/G/1 $q = \frac{\rho^2 A}{(1-\rho)} + \rho$ where $A = \frac{1}{2}\left[1 + \left(\frac{\sigma_s}{T_s}\right)^2\right]$

plug in T_s and σ_s A value is $\frac{1}{2}\left[1 + \left(\frac{1.66}{6.66}\right)^2\right] = \boxed{0.531}$

so plug in to find the average $q = \frac{(0.666)^2 \times 0.531}{(1-0.666)} + 0.666 = \frac{0.23555}{0.334}$

$+ 0.666 = \boxed{1.371}$

so the average response time is $q \div \lambda = T_q$

$T_q = 1.371 \div 0.1 = \boxed{13.71 \text{ sec}}$

Problem 3

availability reliability $e^{-t/2}$

As the midterm date approaches, the CS350 online textbook begins to see more traffic, which could pose a problem to the stability of the server it runs on. Suppose incoming requests follow a Poisson distribution, have exponentially distributed service time, and are immediately placed in a queue to be serviced. The server will then process a request by loading the content of the textbook. When loading is done, the request is complete and can leave the system. On average, it takes 120 ms to load the textbook. We want to ensure that the textbook server won't crash so close to the midterm!

M/M/ FIFO

$T_S = 120ms$ → milliseconds

(a) [5 points] Consider that, on average, students place requests at a rate of 8 requests per second. What is the average amount of time that a generic request will spend in the system?

$\lambda = 8$ req/sec what is T_q ?

$T_S = 0.12$ sec/req

$\rho = \lambda \cdot T_S = 0.96$

$q = \frac{0.96}{1-0.96} = \frac{0.96}{0.04} = 24$

$T_q = 24 \div 8 = 3$ sec

M/M/1 system
- FIFO
- steady state

T_q

3 sec

(b) [5 points] The staff decides that this is too long a wait time for each student, and choose to modify the server to limit the queue to 16 requests in the system at a time. Compute the speedup of the new system (considering only properly served requests i.e., requests that don't get rejected) compared to the old system.

T_{old} / T_{new}

$k = 16$ M/M/1/16 system Need T_q for this system

Speed up = $\frac{T_{old}}{T_{new}}$ $\lambda = 8$ req/sec, $T_S = 0.12$ sec/req does not change

$\rho = 0.96$ $\rho \neq 1$ so

$q = \frac{\rho}{1-\rho} - \frac{(k+1)\rho^{k+1}}{1-\rho^{k+1}} = \frac{0.96}{1-0.96} - \frac{(17)(0.96)^{17}}{1-(0.96)^{17}} = 24 - \frac{0.492917}{0.5004}$

$= 24 - 16.972 = 17.0276$

$T_q = q \div \lambda = 17.0276 \div 8 = 0.897845$

Speed up = $\frac{T_{old}}{T_{new}} = \frac{3}{0.897845} = 3.41510$

Speed up is 3.41510

almost 3.4 times faster

(c) **[6 points]** Over the span of an hour, what is the expected amount of time during which the NEW server is busy serving a request?

M | M | 1 | K

Expected amount of time

h = 1 hr = 60 min = 3600 sec

1 − (the probability of the system has 0 request)

$= P(S_0) =$ since $\rho \neq 1$ $\frac{(1-\rho)\rho^i}{1-\rho^{K+1}} = \frac{1-\rho}{1-\rho^{K+1}} = \frac{0.96}{1-(0.96)^{167}} = \frac{0.96}{0.50041}$

$= 1.9184$

poisson $h = 3600\,sec$ $\lambda h = 28800$

$f(x) = \frac{28800^x}{x!} e^{-28800}$

$x = 0$ No packet

probability of No packet $P(S_0) = \frac{0.04}{0.5004} = 0.0799936$

$P(S_0) = \frac{28800^0}{0} e^{-28800} = \frac{1}{=}$

$P(S_0) = \frac{60}{3600} 1 - e^{-\frac{0}{3600}} = 0$

55.20 minutes

0.92 hours

(d) **[6 points]** True/False: If the average service time of the system doubles, will the number of requests that get rejected also be exactly double compared to what it was before? Motivate your answer.

False

T_s: T_s is doubled utilization will be down

P

When the T_s is doubled that means

probability of rejection $= P(S_k) = \frac{(1-\rho)\rho^k}{1-\rho^{k+1}}$ $\rho \to 2\rho$ Now $\frac{(1-2\rho)(2\rho)^k}{1-(2\rho)^{k+1}}$

lets say $k=2$ $\frac{(1-\rho)\rho^2}{1-\rho^3} \neq \frac{(1-2\rho)4\rho}{1-(8\rho^3)} = \frac{(1-2\rho)4\rho}{1-8\rho^3}$

So the rejection rate will be Not exactly the double

Problem 4

You are a member of an elite computer engineering team behind a growing social networking platform PetBuddy. You manage the system which uploads and processes images of users pets'. Your current system allows users to upload pet pictures of a fixed size (exactly 250×250 pixels). The system that processes these profile pictures is very predictable, and you find that uploading one profile picture always takes 0.1 second. On average, the PetBuddy system receives 5 requests every second to upload a new pet picture.

(a) [6 points] What is the average response time a user will see when uploading a picture of their pet?

M/M/1

upload

$\lambda = 5$ req/sec $T_s = 0.1$ sec

process

$T_s =$

T_q ?

$\rho = \lambda \times T_s = 0.5$

$q = \frac{0.5}{1-0.5} = 1$

$T_q = 1 \times 5 = 0.2$

0.2 sec

(b) **[8 points]** Your team is considering an upgrade to the PetBuddy system (called PetBuddy-Unlimited) that allows users to upload pet pictures of any size. To do this, they have purchased an additional image-processing server that is identical to the old one. In total, the PetBuddy-Unlimited system will have two servers which can process requests in parallel from a single shared queue. Because images sizes are not fixed anymore, your team has observed that it takes each server 0.15 seconds to service each upload request, and that these service times are now exponentially distributed.

If each server in the PetBuddy-Unlimited system, when being actively utilized, consumes 20 Watts of power, what is the probability that at a random point in time the system will be consuming 40 Watts of power?

M/M/2 active 20 x 0.2045 = probability

$\lambda = 2.5$ req/sec, $T_s = 0.15$, $T_s = 0.15$

asks C which is when both servers are utilized at the same time

$C = \frac{1-K}{1-\rho K}$ $K = 1 - \frac{\frac{(NP)^N}{N!}}{\sum_{i=0}^{N} \frac{(NP)^i}{i!}} = 1 - \frac{0.28125}{2.03125} = 0.8615$ K

$\rho = 0.375$ $i=0 = 1$

$i=1 = \frac{(2 \times 0.375)^1}{1} = 0.75$

$i=2 \quad \frac{(0.75)^2}{2 \times 1} = 0.28125$

$C = \frac{1-0.8615}{1-(0.375)(0.8615)} = \frac{0.13846}{0.67693375} = 0.2045$

C = 0.2045

(c) **[8 points]** If your team expects the arrival rate of requests to remain the same as in the old system, what is the speedup of PetBuddy-Unlimited in comparison to the old PetBuddy, as perceived its the users?

$\lambda = 5$ req find T_q for the new system

Speed up

$\frac{T_{old}}{T_{new}}$

$q = C \times \frac{\rho}{1-\rho} + N\rho$ (2x)

$= 0.2045 \times \frac{0.375}{1-0.375} + 2 \times 0.375 = \frac{0.0767}{0.625} + 0.75$

$\frac{1}{0.625} \quad \frac{1}{0.375}$

$= 0.87927$

$T_q = 0.87927 \div 5 = 0.17454$ sec

Speed up $= \frac{0.2}{0.17454} = 1.1458$

New system 14% faster

Problem 5

$\lambda = 12$ req/sec

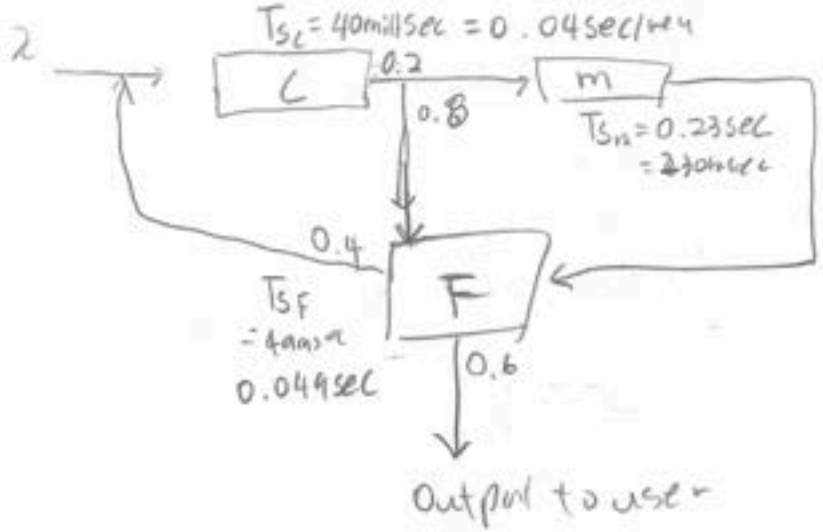
The ChatLPT is a web-based chatbot providing life pro tips. The service is quite popular, receiving on average 12 prompts every second. When a new prompt is submitted to ChatLPT, it *generally* undergoes three processing stages. First, the input is looked up in a cache at the cache machine (C), which takes about 40 milliseconds. After C, two things can happen: (i) with probability 20%, a new LPT needs to be generated, (ii) otherwise LPT generation can be skipped and some previously generated LPT is taken from the cache and forwarded to the last stage (F). In case (i), the main large language model (M) is queried for a candidate reply, which takes on average 230 milliseconds. After (M) or in case (ii) above, the model output is analyzed by a content filter (F) model which ensures that no inappropriate content is returned to the user. Querying the F model takes about 49 milliseconds. If content moderation is successful, the response can be returned to the user and the original request leaves ChatLPT. Conversely, if inappropriate content is detected, the original query is internally modified and resubmitted to the first stage (C) as if it was a new request. This is necessary because it seems that the main model produces inappropriate responses about 40% of the time. All sub-systems are implemented using single-CPU machines.

$T_{SC} = 40$ miles

(a) [5 points] Provide a diagram of the system which shows the inter-connections between the C, M, and F machines. Describe the queuing models used for each machine, the known parameters, and the aossumptions you will employ to solve the system.

$\lambda = 12$ req/sec

All M/M/1 model



- infinite Queue for each models
- poisson distributed arrival rate
- exponential service rate
- FIFO
- steady state

(b) **[5 points]** Identify the resource that constitutes the system bottleneck. Show your reasoning.

$0.08 + 0.32$

$\lambda_c = \lambda + \lambda_c \times 0.2 \times 0.4 + \lambda_c \times 0.8 \times 0.4$

$\lambda_c = \lambda + 0.42\lambda_c$

$0.62\lambda_c = \lambda$

m is the bottle neck

$\lambda_m = 0.2 \times \lambda_c$

$\lambda_F = 0.8 \times \lambda_c$

$\lambda = 12$ everyset

$\lambda_c = 20$

$\lambda_m = 4$

$\lambda_F = 16$

$P_c = 20 \times 0.04 = 0.8$

$P_m = 4 \times 0.23 = 0.92$

$P_F = 16 \times 0.049 = 0.784$

(c) **[5 points]** What is the response time that should be expected by a generic user while the system is at steady state?

T_q ?

$q_c = \frac{0.8}{1-0.8} = \frac{0.8}{0.2} = 4$

$q_m = \frac{0.92}{1-0.92} = \frac{0.92}{0.08} = 11.5$

$q_F = \frac{0.784}{1-0.784} = \frac{0.784}{0.216} = 3.629$

$q_{total} = 19.129$

$q_{total} \setminus \lambda = 19.129 \div 12$

$T_q = 1.5945\,sec$

(d) [6 points] Consider a user that has been able to have their query processed without suffering any queuing delay. How much faster will the system appear to this user compared to the average user using the system at steady state?

$\frac{T_q}{T_{ss}} = \frac{T_s}{T_q}$

T_q for

Speed up $= \frac{1.594}{0.2086} =$ 17.6414

17.6 times faster

q - W/12 = T_s with no delay

19.129 - 16.625 = 2.504 ÷ 12 = 0.208

$W_c = \frac{(0.8)^2}{0.2} = 3.2$

$W_m = \frac{(0.92)^2}{1-0.92^2} \frac{(0.72)^2}{0.08} = 10.58$

$W_F = \frac{(0.784)^2}{1-0.784} \frac{0.614}{0.216} = 2.845$ 16.62

(e) [4 points] What is maximum throughput that can be sustained by the system without losing its ability to reach steady state?

13.043 req/sec

λ or μ the bottle neck is m

when P_m closes to 1

1 ÷ 0.23 = 4.347

λ_m has to be 4.347

λ_c has to be 21.739

λ has to be 13.04