packets = request → avg size = 1.2 KB

1024 bytes

1 byte = 8 bits

server

Problem 1

Your job is to dimension a network switch for ultra-high-speed communication. According to your evaluations, the network packets that will be transmitted on this network are exponentially distributed in size with an average of about 1.2 KB (1 KB = 1024 bytes). The switch will need to handle an average traffic of 8200 packets per second. Initially, you decide to go cheap on the design on the switch and use commercial-off-the-shelf (COTS) Ethernet technology, where the maximum bit transmission speed is 100 Mbps (10^8 bits per second).

λ

μ

a) Packets will be held in memory by the switch while queued and while being processed. How much memory (in bytes) does the switch need to operate such that *most of the time* no packets are dropped due to memory exhaustion?

queue is full

b) If a timestamp is taken for each packet when they enter the system and another one when they begin service, what will be (on average) the difference between the two timestamps?

arr time.

start time.

$\bar{T}_w$?

c) The COTS Ethernet chip you want to use requires to be cooled down with a fan if for every 10 seconds of operation, the chip is busy for 7 or more seconds. Do you need to add a fan in your design?

d) What assumptions have you made to solve the system in all the steps above?

Your colleagues have reviewed your design and are bringing something you did not consider to your attention. That is, the switch is too slow and will be the bottleneck of the entire network. So here is your plan. You propose to use Gigabit Ethernet technology instead, making your switch capable of processing packets at 1 Gbps (i.e. 10^9 bits per second). Respond to your colleagues by computing the following:

$\bar{T}_q$

e) What will be the response time speedup compared to your previous proposal (i.e. standard Ethernet)?

f) Given how expensive are Gigabit Ethernet chips, can we lower the amount of memory to hold in-flight packets in the switch? And by how much?

bits → packets.

a). $\mu = 100\,Mbps = 10^8\,bits/s = \frac{10^8\,bits/s}{1.2KB\,(1024\,bytes)(8\,bits)} = 10172.5\,packets/s$

$\bar{T}_s = \frac{1}{\mu} = \frac{1}{10172.5} = 9.83 \times 10^{-5} s$

$\rho = \lambda \bar{T}_s = 8200\,(9.83 \times 10^{-5}) = 0.8061$

$q = \frac{\rho}{1-\rho} = \frac{0.8061}{1-0.8061} = 4.1573\,packets$

$4.1573\,(1.2kB) = 4.989 \approx 5\,kB$

b). $\bar{T}_w = \bar{T}_q - \bar{T}_s$

$\bar{T}_q = \frac{q}{\lambda} = \frac{4.1573}{8200} = 5.0699 \times 10^{-4} s$ (9.83×10^{-5})

$\bar{T}_w = 5.0699 \times 10^{-4} - 9.83 \times 10^{-5} = 4.0869 \times 10^{-4} s$

c). if $\rho \geq \frac{7}{10}$, then need.

$\rho = 0.8061 \geq \frac{7}{10}$, so need.

(d). We assume that it is a M/M/1 system as as having a Markovian arrival process (i.e., Poisson) and a Markovian (i.e., exponential) service discipline with 1 server.

(e). $\mu' = 1\,Gbps = 10^9\,bits/s = \frac{10^9\,bits/s}{1.2kB\,(1024\,bytes)(8\,bits)}$

$= 101725.3\,packets/s$

$\bar{T}_s' = \frac{1}{\mu'} = \frac{1}{101725.3} = 9.8304 \times 10^{-6}$

$\rho' = \lambda \bar{T}_s' = 8200\,(9.8304 \times 10^{-6}) = 0.080609$

$q' = \frac{\rho'}{1-\rho'} = \frac{0.080609}{1-0.080609} = 0.0877\,packets$

$\bar{T}_q' = \frac{q'}{\lambda} = \frac{0.0877}{8200} = 1.0692 \times 10^{-5} s$

Yes

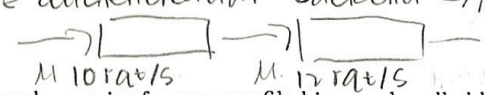
f). $0.080609\,(1.2kB) = 0.0967\,kB$

$5kB - 0.09671kB = 4.9033\,kB$

2 $\frac{\bar{T}_q}{\bar{T}_q'} = \frac{5.0699 \times 10^{-4}}{1.0692 \times 10^{-5}} = 47.4163$

M/M/1 ← authentication backend → M/M/1/K (K=5).

Problem 2



At the Instakilo social network, queries for user profile bios are handled by a dedicated two-stage subsystem. Requests for bios first arrive at a user-facing authentication server. After authentication, they are sent to a backend server. Once they complete handling at the backend server, the response is provided to the users. Thanks to the well-behaved user-base all requests always pass authentication. The authentication server is capable of handling up to 10 requests per second. The backend server, instead, is capable of handling 12 requests per second. But unfortunately, a request at the backend requires a lot of memory and to prevent the server to fail due to memory exhaustion, the engineers have imposed a hard cap on the number of requests that can be admitted at the backend. The limit has been set to 5. Any request arriving while 5 are currently being handled at the backend is rejected and an error response is returned to the user.

a) What is the maximum traffic that the system can handle while still being able to reach steady-state?

b) From now on, consider an input traffic of 8 requests per second. What is the total average number of requests that is either queued or currently being processed in the entire system?

c) When a request is completed at the first stage and then rejected at the second stage, the computation time used to process the request at the first stage is wasted. Over an interval of time of 100 seconds, how much computation time is wasted on average in the system under analysis at steady-state?

d) What is the average response time for requests that are correctly handled (i.e., not rejected)?

e) What is the average response time for a generic request, rejected or not? Assume that the error response to the user is immediately produced if their request is rejected at the second stage.

a). The maximum traffic is 10 requests/s which is equal to the capability of the backend server, because backend server is the bottleneck

b). $\lambda = 8$ requests/s q?

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

$\rho = \lambda \bar{T}_s$

$\bar{T}_s = \frac{1}{\mu} = \frac{1}{10} = 0.1s.$ $\quad \bar{T}_s = \frac{1}{\mu} = \frac{1}{12} = 0.083s$

$\rho = \lambda \bar{T}_s = 8(0.1) = 0.8$ $\quad \rho = \lambda \bar{T}_s = 8(0.083) = 0.664 \neq 1.$

$q = \frac{\rho}{1-\rho} = \frac{0.8}{1-0.8} = 4$ requests $\quad \therefore q = \frac{\rho}{1-\rho} - \frac{(k+1)\cdot\rho^{k+1}}{(1-\rho^{k+1})}$

$= \frac{0.664}{1-0.664} - \frac{6(0.664)^6}{1-0.664^6} = 1.4138$ rqts.

total $q = 4 + 1.4138 = 5.4138$ rqts

c). $P(S_k) = \frac{(1-\rho)\rho^k}{(1-\rho^{k+1})} = \frac{(1-0.664)0.664^5}{(1-0.664^6)} = 0.0474$

$\lambda \cdot P(S_k) = 8(0.0474) = 0.3795$ dropped rqts/s

$0.3795(100) = 37.95$ dropped requests in 100s.

$37.95(\bar{T}_s) = 37.95(0.1) = 3.795s$

d). $\lambda = \lambda(1 - P(S_k)) = 8(1-0.0474) = 7.6208$ not rejected rqt/s

$\bar{T}_q = \frac{q}{\lambda} = \frac{4}{8} = 0.5s.$ $\quad \bar{T}_q = \frac{q}{\lambda} = \frac{1.4138}{7.6208} = 0.1855s$

total $\bar{T}_q = 0.5 + 0.1855 = 0.6855s$

e). $\bar{T}_q = P(S_k)\bar{T}_q + (1 - P(S_k))\bar{T}_q$

$= 0.0474(0.5) + (1-0.0474)(0.6855) = 0.6767s.$

Problem 3

Breaking news! Your multi-million grant to seed your *CheckMeOut* startup. The idea is the following. Your users upload a selfie and the clever CheckMeOut algorithm checks that everything looks in order. The modern-day version of a pocket mirror. But with an unforgiving computerized eye ready to (constructively) criticize you. For instance, forgot to shave? *CheckMeOut:* "Have you lost your razor?"; Pillow marks on your cheeks? *CheckMeOut:* "Them pillow marks make me yawn.". And so on.
With these funds at hand, now it is time to perform system tuning. After observing the system for a while you have noticed the average picture size submitted by the users is 3 MB (1 MB = 1024 KB; and 1 KB = 1024 bytes). Every day, you are receiving about 26,000 CheckMeOut requests that you process in order of arrival. Your want to know the following:

$\lambda = 26000 \text{ rqts/day}$.

a) How long should your system spend processing each individual request (on average) so that the system ends up handling, on average, about 20 requests—either being currently processed or queued?

$\frac{1}{\lambda} = 2.55$

b) Assume that you were able to tune your system so that processing each request takes about 2.5 s, what is the time that each user will end up waiting on average before receiving their CheckMeOut response?

c) How would the average response time change if you decided to switch to SJN policy instead of serving requests in their order of arrival? T_q

d) How would the average response time change if you determined that the processing time for each request follows a Normal distribution with mean 2.5 s and standard deviation 0.5 s? T_q M/G/1

μ σ 6.

From now on, consider the system tuned according to what mentioned in Question b). That includes exponentially distributed service times, with the server employing a FIFO dispath policy. $\frac{1}{\lambda} = 2.5$ s.

e) What is the slowdown of the system due to queuing?

f) What is the maximum number of requests that the system can sustain on a daily basis?

g) How much memory will the requests in the system occupy on average, considering that (1) queued requests occupy as much memory as the size of the input image; and (2) requests currently being processed occupy 3× as much until their processing is completed.

The federal agency providing the funds got back to you. They say that you must support way more users per day or the deal is off! Overnight, you devise the following optimization. You organize your system in two stages. At the first stage, pictures are compressed down to 450 KB. After compression, they are sent to the main processing server for CheckMeOut caption generation. This way, the final processing stage can take as little as 1.1 s. Now you worry about the following:

h) How long should compression take so that the system can sustain 70,000 requests per day?

i) Assume that compression takes 0.9 s, what is going to be the average time it takes your whole system (compression+processing) to take a new request in and produce the corresponding output when your system reaches 70,000 requests per day?

Problem 3

a). $\lambda = 26000$ requests/day

$= \frac{26000 \text{ rqt/day}}{24h(60min)(60s)} = 0.3 \text{ rqt/s}$

$q = 20$ requests $\rightarrow 20 = \frac{\rho}{1-\rho}$ $\therefore \rho = \frac{20}{21} = 0.9524$

$\rho = \lambda \bar{T}_s \rightarrow 0.9524 = 0.3 \bar{T}_s$ $\therefore \boxed{\bar{T}_s = 3.1746 s}$.

b). $\bar{T}_s = 2.5s$

$\rho = \lambda \bar{T}_s = 0.3(2.5) = 0.75$

$q = \frac{\rho}{1-\rho} = \frac{0.75}{1-0.75} = 3$ requests

$\bar{T}_q = \frac{q}{\lambda} = \frac{3}{0.3} = \boxed{10s}$.

c). The average response time Tq will be smaller because by using SJN, we pop the shorter LEN requests first which means we execute them first.

d). M/G/1 $\mu_{\bar{T}_s} = 2.5s$ $\sigma_{\bar{T}_s} = 0.5s$

$A = \frac{1}{2}(1 + (\frac{0.5}{2.5})^2) = 0.52$

$q = \frac{\rho^2 \cdot A}{1-\rho} + \rho$

$\rho = \lambda \bar{T}_s = 0.3(2.5) = 0.75$

$\therefore q = \frac{0.75^2 (0.52)}{1-0.25} + 0.75 = 1.92$ requests

$\bar{T}_q = \frac{q}{\lambda} = \frac{1.92}{0.3} = \boxed{6.4s}$

e). slowdown $= \frac{\bar{T}_q}{\bar{T}_s} = \frac{1}{1-\rho} = \frac{1}{1-0.75} = 4$

f). throughput $= \frac{1}{\bar{T}_s} = \frac{1}{2.5} = 0.4$ requests/s.

$= 0.4(60s)(60min)(24h) = \boxed{34560 \text{ rqts/day}}$

(1).

g). $W(3GB) = 2.25(3GB) = 6.75\ GB$

$W = q - \rho = 3 - 0.75 = 2.25$ requests

(2).

$q(3)(3GB) = 3 \cdot (3)(3GB) = 27\ GB$

$6.75GB + 27GB = \boxed{33.75\ GB}$

h). $\lambda = 70000$ requests / day

$= \frac{70000\ rqts/day}{(24h)(60min)(60s)} = 0.8102\ rqts/s.$

$\lambda = \mu = \frac{1}{T_s} \rightarrow 0.8102 = \frac{1}{T_s} \quad \therefore T_s = \boxed{1.2343s}$

i). $T_s = 0.9s \quad \lambda = 70000\ rqts/day = 0.8102\ rqts/s \quad T_q?$

$T_q = \frac{q}{\lambda}$

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

$\rho = \lambda T_s = 0.8102(0.9) = 0.72918$

$q = \frac{\rho}{1-\rho} = \frac{0.72918}{1-0.72918} = 2.6925$ requests

$T_q = \frac{q}{\lambda} = \frac{2.6925}{0.8102} = 3.3232s \leftarrow$ compression.

$T_s = 1.1s \quad \lambda = 0.8102\ rqts/s \quad T_q?$

$\rho = \lambda T_s = 0.8102(1.1) = 0.89122$

$q = \frac{\rho}{1-\rho} = \frac{0.89122}{1-0.89122} = 8.1929\ rqts$

$T_q = \frac{q}{\lambda} = \frac{8.1929}{0.8102} = 10.1122s. \leftarrow$ processing

total $T_q = 3.3232 + 10.1122 = 13.4354s$