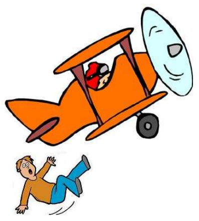
Computer Networks
M/M/1 with finite queue

Saad Mneimneh
Computer Science
Hunter College of CUNY
New York



Sorry, I have to drop you!

1 Introduction

We now re-examine the M/M/1 queing system under the more realistic assumption of a finite queue size. Therefore, we assume that the queue can hold up to m packets. If a packet arrives and the queue is full, the packet is simply dropped from the system. The analysis of M/M/1 under this setting will give us some insight into the behavior of networks in general, and an idea for a congestion control algorithm used by TCP.

2 The Markov chain

For this new setting, we have a finite Markov chain with a set of states $\{0, 1, 2, \ldots m\}$ and transional probabilities as illustrated in Figure 1.

As before, we need to find the steady state probability p_n of being in state n (system has n packets). The steady state equations are as follows (refer to Figure 1):

$$p_0 = p_0(1 - \lambda\delta) + p_1\mu\delta + o(\delta)$$

$$p_m = p_{m-1}\mu\delta + p_m(1 - \mu\delta) + o(\delta)$$

$$p_n = p_{n-1}\lambda\delta + p_n(1 - \lambda\delta - \mu\delta) + p_{n+1}\mu\delta + o(\delta) \quad \forall n \neq 0, m$$

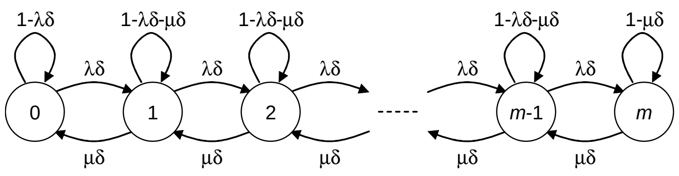


Figure 1: Markov chain for finite M/M/1

$$\sum_{n=0}^{m} p_n = 1$$

Since $\lim_{\delta\to 0} o(\delta)/\delta = 0$, taking the limit as δ goes to 0 gives:

$$\frac{p_n}{p_{n-1}} = \frac{\lambda}{\mu} = \rho$$

Therefore,

$$\sum_{n=0}^{m} p_n = 1 \Rightarrow \sum_{n=0}^{m} \rho^n p_0 = 1 \Rightarrow p_0 = \frac{1}{\sum_{n=0}^{m} \rho^n} = \frac{\rho - 1}{\rho^{m+1} - 1}$$

$$p_m = \rho^m p_0 = \frac{\rho^{m+1} - \rho^m}{\rho^{m+1} - 1}$$

$$(\text{because } \rho \neq 1 \Rightarrow \sum_{n=0}^{m} \rho^n = \frac{\rho^{m+1} - 1}{\rho - 1})$$

If $\rho = 1$, $\sum_{n=0}^{m} \rho^n = m + 1$ and $p_n = \frac{1}{m+1} \quad \forall n$ (all states are equally likely). Note that p_m is the probability of dropping a packet (because of the PASTA property of Poisson arrivals). When $m \to \infty$ and $\rho < 1$, $p_m \to 0$ (never drop, inifite queue). When $\rho \to \infty$, $p_m \to 1$ (always drop).

Now we can compute N, the expected number of packets in the system.

$$N = \sum_{n=0}^{m} p_n n = \sum_{n=0}^{m} \rho^n p_0 n = \frac{\rho - 1}{\rho^{m+1} - 1} \rho \sum_{n=0}^{m} n \rho^{n-1}$$

$$N = \frac{\rho - 1}{\rho^{m+1} - 1} \rho \frac{\partial}{\partial \rho} \sum_{n=0}^{m} \rho^n$$

$$N = \frac{(m+1)\rho^{m+1}}{\rho^{m+1} - 1} + \frac{\rho}{1 - \rho}$$

If $\rho = 1$, $p_n = \frac{1}{m+1} \quad \forall n$. Then,

$$N = \sum_{n=0}^{m} \frac{1}{m+1} n = \frac{m}{2}$$

When $m \to \infty$ and $\rho < 1$, $N \to \frac{\rho}{1-\rho}$ (as before). When $\rho \to \infty$, $N \to m+1-1 = m$ (queue always full).

Packets that are not dropped arrive according to a Poisson process with rate $\lambda(1-p_m)$ (obtained by a splitting of the original Poisson process as seen below).

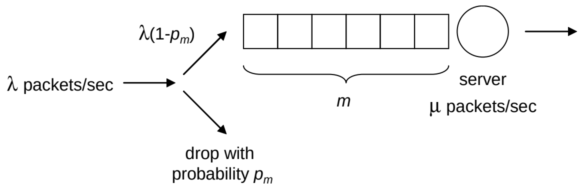


Figure 2: Splitting of a Poisson process

Therefore, by Little's theorem

$$T = \frac{N}{\lambda(1-p_m)}$$

$$T = \frac{1}{\mu}\left[\frac{(m+1)(1-\rho)\rho^m + \rho^{m+1} - 1}{(\rho^m - 1)(1-\rho)}\right]$$

If $\rho = 1$, $p_m = \frac{1}{m+1}$, $N = \frac{m}{2}$, and $T = \frac{m+1}{2\mu}$. When $m \to \infty$ and $\rho < 1$, $T = \frac{1}{\mu - \lambda}$ (as before). When $\rho \to \infty$, $T = \frac{m}{\mu}$.

3 Throughput, delay, and power

By Little's theorem, the expected number of packets being served (throughput) is

$$\frac{\lambda(1-p_m)}{\mu} = \rho(1-p_m) \leq 1$$

When $\rho > 1$, however, the throughput cannot be more than $(1-p_m)$, because $(1-p_m)$ is the fraction of packets that are not dropped. Therefore,

$$\text{throughput} = \min(\rho(1-p_m), 1-p_m)$$

Compare this to the infinite queue case where $p_m = 0$ and the throughput is given by $\min(\rho, 1)$.

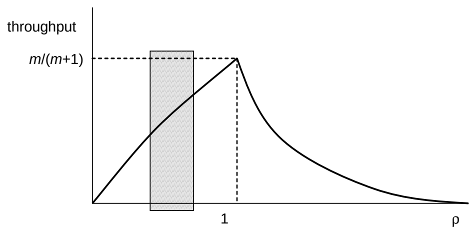


Figure 3: Throughput curve for finite M/M/1

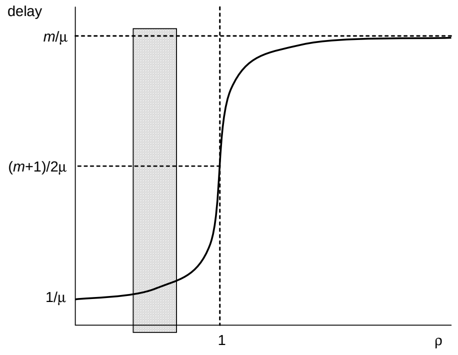


Figure 4: Delay curve for finite M/M/1

The dashed region shows desired operating region for power=throughput/delay.

4 A flavor of congestion control

To summarize what we have seen so far (refer to the throughput and delay curves above):

- Throughput increases with ρ until it reaches a maximum and then starts decreasing rapidly (this is when we have considerably high droping probability)
- Delay start increasing considerably when throughput approaches 1
- Best operating point is to keep throughput around 50%

Therefore, a possible congestion control algorithm would be the following:

- Upon detection of a loss (a sign that we have approached a throughput close to 1), decrease the rate (reduce window size) by a factor of 2
- Increase rate slowly until detecting another loss

We are going to see later that TCP employs basically a similar congestion control algorithm.

5 A note on Poisson arrivals

Is it always justifiable to assume Poisson arrivals? To answer this question, consider the following example of two queues in tandem.

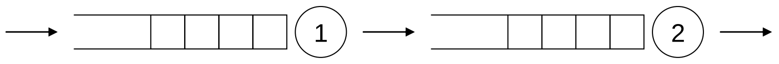


Figure 5: Two queues in tandem

If packets arrive according to a Poisson process (arrival), and packet lengths are exponentially distributed (service), the first queue can be modelled as an M/M/1 queuing system. However, the second queue cannot! For instance, the inter-arrival times at the second queue are strongly correlated with packet lengths (service). In particular, the inter-arrival time of two packets at the second queue is greather than or equal to the transmission time of the second packet at the first queue. Therefore, long packets typically wait less time at the second queue.

If the second server were to receive a substantial amount of additional external Poisson traffic, the dependence of inter-arrival times and service times would be weakened. Therefore, it is often appropriate to adopt an M/M/1 model (Kleinrock independence approximation) for each server when:

- we have Poisson arrivals at entry points
- packet lengths are nearly exponentially distributed
- the network is densely connected
- we have moderate to heavy traffic loads

Under these assumptions, if link (i, j) has a total rate λ_{ij}, then (from M/M/1):

$$N_{ij} = \frac{\lambda_{ij}}{\mu_{ij} - \lambda_{ij}}$$

The average delay per packet in the entire system (Little's theorem) is (ignoring propagation delay)

$$T = \frac{\sum_{i,j} N_{ij}}{\gamma}$$

where γ is the total arrival rate in the system. If the propagation delay d_{ij} for link (i, j) is taken into account, then:

$$T = \frac{\sum_{i,j}(N_{ij} + \lambda_{ij} d_{ij})}{\gamma}$$

References

Dimitri Bertsekas and Robert Gallager, Data Networks