APPLICATION NOTE

Tuning the limit on incoming connections to a mail server

Peter Moylan

Glendale, NSW

peter@pmoylan.org http://www.pmoylan.org

December 2017

ABSTRACT: When setting up a mail server, how many simultaneous incoming mail sessions should we allow for? The tradition has been to set a limit by trial and error. In this note we use queueing theory to work out how many simultaneous connections we should allow for.

1. Introduction

A mail server, in common with many other server applications, needs to be able to handle multiple simultaneous connections from remote clients. Naturally, there has to be some limit on how many simultaneous sessions can be handled. A naïve approach would be to allow as many simultaneous as can be handled before running out of resources. That is a bad decision for several reasons. First, we might run out of resources after starting work on the latest connection, and in that case we waste resources by having to discard the work already done. Second, such an approach would make the server a "greedy" application that hogs resources that ought to be made available to other software running on the same system.

Third, and most importantly, allowing too many connections is simply being kind to attackers. Attacks on mail servers, mostly with the aim of guessing passwords by attempting to be authenticated over and over again, are unfortunately all too common. We can block some of those attacks with good firewall rules and various blacklisting approaches, but there will always be some attackers that get through. One way to limit the damage caused by those attackers is to have a tight limit on the number of permitted simultaneous client sessions.

That limit should be set based on the expected level of legitimate traffic, and most servers do set such a limit. (Connection attempts that go over the limit will get a "try again later" response, and the retries do mean that legitimate mail gets through with only a small delay.) The limit is usually set by guesswork, relying on the experience of the system manager. In this note we show that, by modelling the mail server as a queue, we can calculate what the limit should be.

This note was written for the benefit of users of the Weasel mail server [Weasel], but the conclusions are applicable to just about any mail server, and indeed to many non-mail server applications.

Although we use queueing theory in this paper, we simply quote some well-known results without derivation. The theory should therefore be understandable even to someone who knows nothing about queueing theory.

2. Theory

A *queue* is a structure where elements can be removed from the head, and new elements can be added at the tail. The length of the queue at any time is the initial length, plus the total number of arrivals, minus the total number of departures.

A multiserver queue can be thought of as several queues running in parallel. An extreme case is where there is an unlimited number of servers, so that new arrivals are dealt with as soon as they arrive. In that case there is no queueing in the popular sense of the word.

For our purposes, the interesting kind of queue is one where elements arrive and leave at random times. In particular, an M/M/∞ queue is one where, in any short time interval dt, the probability of an arrival is λdt and, for each element currently being served, the probability of service completion is μdt, where λ and μ are constants. The letters M in the notation indicate that both the arrival and departure processes are memoryless – that is, the arrival and service completion probabilities do not depend on what has happened in the past – and the ∞ indicates that there is no limit on the length of the queue.

The queue length varies randomly, but we can solve for the probabilities of each possible value of the length, and how these probabilities evolve with time. In particular, it is known [Kleinrock 1976] that the steady state probabilities follow a Poisson distribution

$$p_k = \frac{1}{k!} m^k e^{-m}$$

where p_k is the probability that the queue length is k, and

$$m = \frac{\lambda}{\mu}$$

is the mean value of the distribution. It is also known that the mean service time is $1/\mu$, a fact that lets us estimate the value of μ from server log files.

Solutions for the transient response are also known, but we will not need them for our present purposes. Still, it is of interest to know that $1/\mu$ is the time constant of the transients. That tells us, for example, how long it will take for a sudden load change to settle down to the new equilibrium.

Can the "incoming mail" part of a mail server be modelled by an M/M/∞ queue? The assumption of infinite queue capacity is not realistic, but we justify it as follows. Our goal is to work out what the capacity should be. To do this, we start with an infinite queue capacity, work out the queue length probabilities, and then truncate the model in such a way that we eliminate the lengths of low probability.

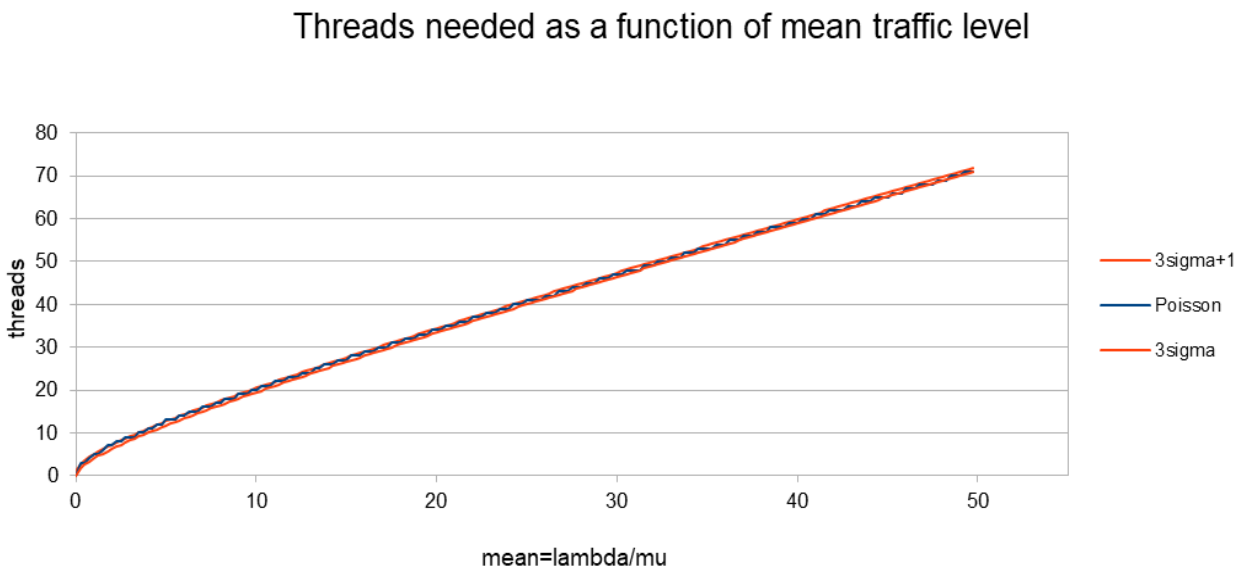
That leaves the question of memoryless arrival and departure processes. We do not know of any practical studies of the statistical distribution of message arrival times at a mail server, or of the distribution of processing times. For now, however, it seems reasonable to act as if those arrivals and departures have the right randomness properties. We will in any case add a safety factor to the calculated queue capacity, so any modelling errors should not be too serious.

When the mean of a Poisson process is large, a Poisson distribution is approximately equal to a Gaussian distribution. A common way of specifying the width of a Gaussian distribution are the "three-sigma bounds". About 99.7% of the values drawn from such a distribution fall within three standard deviations from the mean. For our purposes, that suggests where to put the limit on allowable connections. Indeed, with that limit an arriving client has a better than 99.8% chance of being accepted, because we are only chopping off the upper tail of the distribution, not both tails.

NOTE: if there is a 0.2% chance of meeting a "too many connections" limit, that does *not* imply that 0.2% of the mail is lost. The rejection is of the "try again later" variety. In those cases, the "rejected" mail will still arrive within the next few minutes.

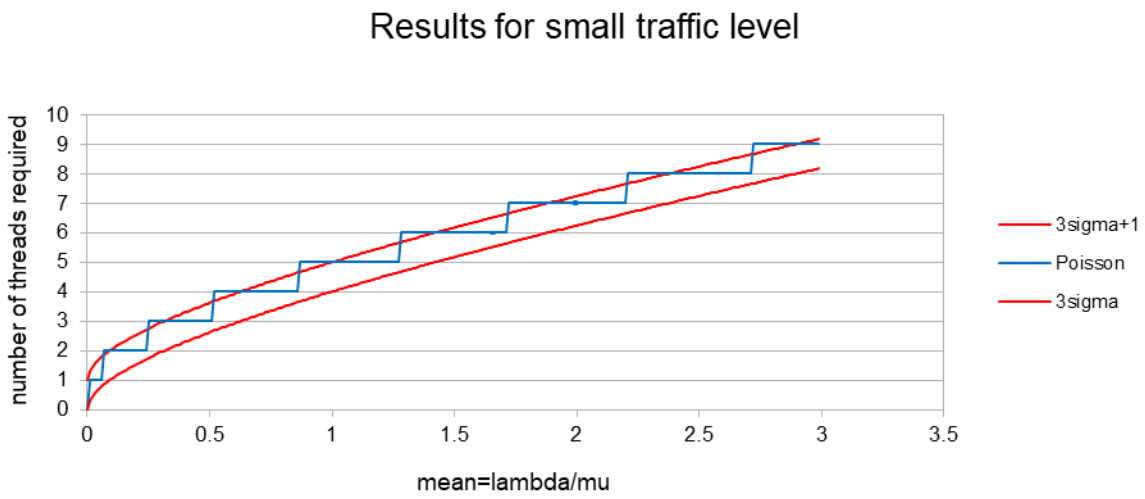
But what if the mean is so small that the Poisson distribution does not look much like a Gaussian distribution? We can handle that case – and, in fact, all cases – with a simple calculation. For a given mean, calculate the probability that the queue size is k, for $k = 0,1,2, \ldots$. Add these probabilities to get a cumulative probability, and stop the calculation when the cumulative probability is 0.998 or higher. That says what our connection limit should be for that mean. Repeat the calculation for a range of means, and prepare a table or graph to show the results.

That result is shown in the following graph. Also shown, in red, is the 3-sigma limit that is obtained if we assume a Gaussian distribution. It is not surprising that that two graphs match for large values of the mean. What *is* surprising is that they still come close to matching for small values of the mean.



The match is not perfect, and this becomes clearer in our second graph, which zooms in on the portion with small mean. It can be seen that the graph lies roughly between the 3-sigma limit and one more than the 3-sigma limit. In fact, we can deduce a better result. The 3-sigma values are real numbers, but the k values have to be integers. If we round up the 3-sigma values to the next integer, then it is easy to see from the graph that the blue stepped line lies precisely between the rounded-up values of the 3-sigma values. That is, we can calculate the

number of threads as if we had a Gaussian distribution, even in the case where the true distribution is not Gaussian.



Thus, it seems reasonable to design for a capacity

$$\frac{\lambda}{\mu} + 3\sqrt{\frac{\lambda}{\mu}} + 1$$

Note that the values of λ and μ can often be found easily by looking at mail log data.

3. Worked examples

Consider a small mail server, with 80 local users who each receive, on average, 100 mail messages per day. (This is a rather high load per user, at first sight, but we have to include spam in the calculation, unless there is some upstream server that is blocking the spam.) Whether those users are in different mail domains or in the same domain is irrelevant to the calculation. The arrival rate is 8000 messages/day, which works out to be about λ=0.09 messages/second. Greater precision than this is not justified, given the approximations that have to be made.

The departure rate has to be calculated on the time that one message reception takes, from the initial connection from the sending machine to the closing of the session. This includes things like blacklist checking. To be conservative, let us take that value to be 10 seconds, which might be typical when the hardware is slow or when many mail messages contain large attachments. The reciprocal of this is $\mu=0.1\ \text{sec}^{-1}$.

That means that the mean queue size is $\lambda/\mu = 0.9$. That means that on average there is just under one incoming connection active. To allow for random arrivals, however, we should design for

$$\frac{\lambda}{\mu} + 3\sqrt{\frac{\lambda}{\mu}} + 1 = 4.75$$

The answer has to be an integer, so we round this up to 5.

Now consider a busier server, with 5000 users, each of whom receive 50 messages/day, each of which takes about 3 seconds to receive. This gives us λ=250000 messages/day=2.89 messages/sec. We also have μ=0.33 sec^{-1}, so λ/μ=8.77. Now we have to allow for

$$\frac{\lambda}{\mu} + 3\sqrt{\frac{\lambda}{\mu}} + 1 = 8.77 + 8.88 + 1 = 18.65$$

simultaneous connections, or 19 after rounding up.

4. Conclusions

We have shown a simple way to calculate how many simultaneous SMTP connections must be allowed for, given known traffic levels that can be extracted from log data.

The answers obtained in the examples are surprisingly low. Most people, working from guesswork and intuition, would probably set the limits much higher. This is possibly a panic reaction to what happens when the server is under attack and quickly hits the "too many users" limit. Increasing the quota for that reason, however, would be a bad mistake, because it would only give more resources to the attacker. It is better to focus on ways to block attackers, with updated firewall rules for example.

Although we have looked only at incoming SMTP connections to a mail server, it should be clear that the same analysis can be applied to other kinds of server connections, where the intention is to serve multiple simultaneous clients.

References

[Kleinrock 1976] L. Kleinrock, *Queueing Theory*, Wiley, New York, 1976.

[Weasel] P. Moylan, Weasel mail server, http://www.pmoylan.org/pages/os2/Weasel.html.