

Appendix E

Queuing Theory and Erlangs

Queuing theory deals with queues. A queue, in turn, is a line, as in “wait in line for your turn”. The word is used much more in England than in the U.S.; the British would rather say “Join the queue” than “Wait in line.”

In communications, queuing theory deals with a lineup of uses or users, waiting for a free line or other facility. For example, consider a company which has five incoming telephone lines for taking orders, and customers who call at random times during the day. How often will one of those customers get a busy signal because all five lines are busy?

The subject was first studied by A. K. Erlang, a Danish mathematician working for the Danish telephone company, in the early 1910’s. He was interested in calculating how many telephone lines (trunks) were required from his village to the outside world so as to allow villagers to call out without getting too many busy signals.

Erlang actually studied three possible situations:

- Someone calls, gets a busy, and hangs up, *or*
- Someone calls, gets a busy, hangs up, and then immediately tries again (and keeps trying), *or*
- Someone calls, and gets a message that says, “Please wait for an available line,” so he waits until a line frees up.

The simplest case is the first, and it is generally called an *Erlang B* problem.

The “please wait...” case is called *Erlang C*, and has an additional factor — calculating how long, on average, the caller will have to wait before being answered.

Of course, back in 1917, when Erlang published his work, there was no such thing as a recording of “please wait...”; he was thinking more of people in a store, waiting on line for a clerk. In general, these Erlang problems can apply to various situations — a line of people in a store, the number of trunk lines connecting cities or central offices, or the number of incoming lines into a company’s order department. For the rest of this appendix, we will consider the trunks and offices case.

Offered Traffic

The amount of traffic on a system is measured in units called *erlangs*. An erlang is a trunk line in full-time use for one hour. A few examples:

- One trunk line used for 30 minutes out of each hour is 0.5 erlang.
- Two trunk lines, each of which is used for 30 minutes out of each hour, is a total load of 1 erlang.

- Two trunk lines, one of which is used 20 minutes per hour (which is 0.33 erlang), and the other used 30 minutes per hour (which is 0.5 erlang) make up a total of 0.83 erlang.

The above examples describe erlangs in terms of *actual traffic* on a trunk. Another way is to think of erlangs as a need:

- If you have 60 calls per hour, and each call lasts 1 minute, you need to handle 1 erlang worth of calls.

This “need” is actually called the *offered traffic*. That is, this is the amount of traffic that the customers offer (or provide to) the system. The offered traffic in erlangs is calculated from

$$\text{erlangs} = \frac{\text{average length}}{\text{average spacing}}$$

In this case, the average length of a call is 1 minute, and the average spacing is also 1 minute (which is 60 calls per hour), so the offered traffic in this example is 1 erlang. (We must be careful to use the same units in both the numerator and denominator of this equation.)

If these calls were evenly spaced — one call every minute — and if each call was exactly one minute long, then one line used the entire 60 minutes of each hour — which is also 1 erlang — would be sufficient to handle the calls.

The problem is that, in real life, the calls will not be evenly spaced, and they will not all be exactly the same length. Sometimes a call will come in while the line is still busy with a previous call, and that customer will get a busy signal — the call is said to be *blocked*. Other times, the line may be free, but there are no calls.

To avoid blocked calls, more than one line may be needed, yet even then it may still occasionally happen that there are more simultaneous calls than there are lines. This was Erlang’s problem: Given a certain number of lines or trunks, and a certain amount of offered traffic, what is the probability that a call will be blocked?

DETOUR 

Erlang’s answer for this particular example: with one trunk, and 1 erlang of offered traffic, the probability of a blocked call is $\frac{1}{2}$. That is, 50% of the calls will be blocked, and 50% will get through. Since only $\frac{1}{2}$ of the calls get through, the trunk is only used $\frac{1}{2}$ of the time, so the actual load is $\frac{1}{2}$ erlang.

Financial people and efficiency experts are often concerned with *line utilization*. Since the trunk is only used $\frac{1}{2}$ of the time, the line utilization is 50%, which worries them. In an actual case, however, with calls ar-

riving at random intervals and random lengths, it is impossible to get 100% utilization.



When we look at offered traffic, we have to realize that it changes during the day. The telephone order desk of a company may be very busy between 9 A.M. and 11 A.M., and relatively light between 3 P.M. and 4 P.M. This difference has to be taken into account. Traffic calculations are usually based on *Busy Hour Traffic* or BHT — the amount of traffic during the busy times of the day.

There is another unit for measuring traffic — the CCS. The first C means “one hundred” (the letter C stands for 100 in Roman numbers), and the remaining CS means call seconds. Hence one CCS is 100 seconds of call time. Since there are 3600 seconds in one hour (or 36 hundred seconds), there would be 36 CCS in an hour worth of traffic, which is an erlang. Thus

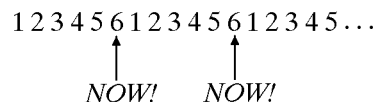
$$1 \text{ erlang} = 36 \text{ CCS}$$

CCS are sometimes used instead of erlangs, but we will stick with erlangs out of respect for A. K., the man who started it all.

Poisson Distribution

To develop his equation for the probability of a blocked call, A. K. Erlang had to look at the way incoming calls arrived, which is called a *Poisson probability distribution*. Let’s take a simple look at it.

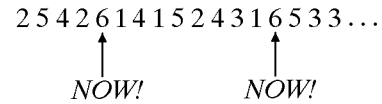
Suppose calls are spaced 6 seconds apart on average. If they were evenly spaced, a call would arrive every six seconds like clockwork. We could look at the second-hand on a watch, and just count off the seconds to predict when the next call will come:



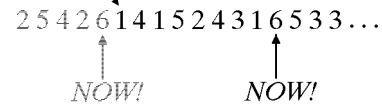
But let us try to simulate the actual random arrival of the calls by using dice. Instead of neatly counting off the seconds 1 2 3 4 5 ... by using a watch, let’s generate a random sequence of digits, like 2 5 4 2 6 1 4 1 5 ... by throwing a die once a second. Over the long term, we will get just as many 1’s, 2’s etc. as before, but now they are scrambled. And then let’s make believe that a call comes in each time we get the number 6, just like before.

We get a new series of numbers, where the 6’s appear more randomly, just like incoming calls would come it at more random times. Still, on the average,

there are just as many 6’s over a long period, so there is the same number of calls as before, and their average spacing is still the same six seconds:



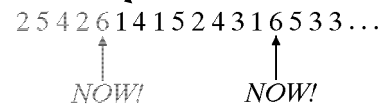
Let’s now take a look at what happens after the very first call, right here



What is the probability of another call right after that first one? That is, what is the probability of throwing another 6 on the dice just after the previous 6? That is,

- What is the probability of a call coming within 1 second after the previous call?
- The probability of throwing a 6 on a six-sided dice is $\frac{1}{6}$, so the probability of a call coming in 1 second after a previous call is 0.1667.

Let’s now take a look at the next second after that first call, right here



We now ask:

- What is the probability that there wasn’t a call during the first second, but there is a call during the next one? That is, what is the probability that a call came in not one, but *two* seconds after the previous one?

In math, the *joint probability* of two independent events occurring is the probability that they both happened, and it is the product of the two separate probabilities. In this case, the probability that there was *not* a call during the first second is $\frac{5}{6}$ (there was a 5 out of 6 chance of *not* throwing a 6 on the first toss), and the probability that there *was* a call during the next second is $\frac{1}{6}$ (i.e., the chance of throwing a 6 on the second toss is 1 out of 6), so the joint probability of both of these occurring is

$$\frac{5}{6} \times \frac{1}{6} = 0.1389$$

Our next question is:

- What is the probability that there wasn’t a call during the first second, and there was also not a call during the second second, but there is a call during the third one? That is, what is the probability that a call came in not one, not two, but *three* seconds after the previous one?

Now the probability that there was *not* a call during the first second is $\frac{5}{6}$, the probability that there was *not* a call during the second second is also $\frac{5}{6}$, and the probability that there *was* a call during the last second is $\frac{1}{6}$, so the joint probability of all three of these occurring is

$$\frac{5}{6} \times \frac{5}{6} \times \frac{1}{6} = 0.1157$$

We could continue like this to get the following table:

Time since previous call	Probability of a call
1 second	0.1667
2 seconds	0.1389
3 seconds	0.1157
4 seconds	0.0965
5 seconds	0.0804
6 seconds	0.0670
7 seconds	0.0558
8 seconds	0.0465
9 seconds	0.0388
	(and so on...)

Graphing the data in this table gives us Fig. E-1; this is the Poisson Probability Distribution, and you will probably recognize it as a decaying exponential.

Both the table and the graph tell us this: even though the average spacing between calls in this example is 6 seconds, there is still a good probability of two calls coming just a second or two apart. Even if the average length of a call were just one second, it would still happen about $\frac{1}{6}$ of the time that an incoming call comes in before the preceding one is finished. In other words, even if the average length of a call were substantially less than the spacing between calls, which would give you less than one erlang of offered traffic (since

$$\text{erlangs} = \frac{\text{average length}}{\text{average spacing}}$$

and the length in this example is smaller than the spacing so the fraction is smaller than 1), you would still need more than one trunk to avoid getting blocked calls.

In this discussion, we assumed that the incoming calls were randomly spaced, but were all of the same length. That is obviously not going to happen, and A. K. Erlang looked at four different possibilities for the call length:

- All calls could be the same length, namely the average length, *or*
- The call lengths might be uniformly distributed over some range. For example, one 1-second call, one 2-second

call, one 3-second call, and so on, up to one 11-second call, would all average to 6 seconds average length, *or*

- The call lengths might have a Gaussian or normal distribution; this is also sometimes known as the Bell curve. That is, there might be a fairly large range of lengths, but most calls would be near the average of 6 seconds, while relatively few would be very short or very long, *or*
- The call lengths might have the same Poisson distribution as the spacing between calls. That is, there might be a large number of very short calls, and a very small number of very long calls, with a curve somewhat like Fig. E-1.

Erlang found that call lengths in the real world followed the Poisson distribution, and also that how the lengths were distributed did make a difference in his results. But he also found that the difference was small and not worth worrying about.

Erlang derived the following formula for the fraction of calls blocked:

$$P_{\text{blocking}} = \frac{\frac{E^n}{n!}}{\frac{E^n}{n!} + \frac{E^{n-1}}{(n-1)!} + \dots + \frac{E^2}{2!} + E + 1}$$

where

P_{blocking} is the probability of blocking (for example, 0.15 would mean 15% of the calls are blocked), for the Erlang-B case,

E is the number of erlangs, and

n is the number of trunks or circuits.

Unfortunately, this equation is difficult to evaluate for large values of n ; even using a computer, you must

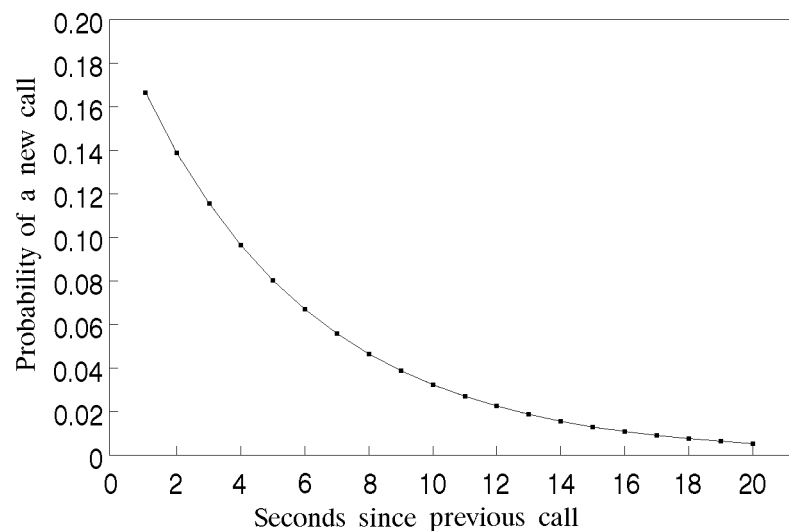


Fig. E-1. Poisson probability of an incoming call

program it carefully to avoid overflows (cases where a number is bigger than what the computer can safely handle without error), and so most people prefer to use a table, rather than the equation itself.

Table E-1 gives the number of erlangs of offered traffic that can be handled by a given number of trunk lines, at a given percentage of blocked calls, for the Erlang-B case. For example, the top line of the table tells us that one trunk can only handle 0.01 erlangs if the blocking percentage is 1%. Even if you are willing to tolerate 20% blocked calls, one trunk can still only handle an offered load of 0.25 erlangs (of which 20% will be blocked!).

Let's return to our earlier example where we had calls which averaged 6 seconds long, and which came 6 seconds apart, giving us 1 erlang. Suppose we wanted 1% or fewer blocked calls — how many trunks do we need?

Looking at the 1% block column of the table, we see that four trunks could handle 0.87 erlang, which is not quite enough. Five trunks, on the other hand, can handle 1.36 erlangs, so that would be sufficient to handle the 1 erlang of offered traffic.

What percentage of blocked calls is acceptable? That is a business decision, which depends on the cost of lost calls compared to the cost of providing more trunks or lines. For example, an order department for a mail-order merchant that sells cheap trinkets might tolerate a higher percentage of blocked calls, because the loss of a few sales is small compared to the cost of hiring more people and getting more phone lines. On the other hand, a company selling expensive cameras might not want to lose a few high-ticket sales.

Table E-1 only goes up to a 20% blocking rate on the assumption that most people would not want more than 20% of their calls to be blocked. There are, of course, exceptions — someone running a complaint department might not mind having 50% or even 80% of their callers getting a busy!

Utilization Factor

Note, by the way, that the Erlang-B results are biased to favor big spenders. For example, at a 1% blocking factor, three trunks can only handle 0.46 erlangs. Increasing the number of trunks by a factor of 10, from 3 to 30,

increases the possible load by a factor of 44, from 0.46 erlangs to 20.34 erlangs.

Another way to look at this is from an efficiency viewpoint. The *utilization factor* describes how busy the trunks (or lines or people) are. The equation is

$$\text{U.F. \%} = \frac{\text{erlangs offered} \times \text{fraction unblocked}}{\text{number of trunks}} \times 100$$

For example, 3 trunks could handle 3 erlangs if they were used 100% of the time. If they are only handling 0.46 erlang of which 1% is blocked (meaning that 99% or 0.99 of these are not blocked), then they are only used

TABLE E-1. ERLANG B TRAFFIC TABLE					
Trunks	1% block	3% block	5% block	10% block	20% block
1	0.01	0.03	0.05	0.11	0.25
2	0.15	0.28	0.38	0.60	1.00
3	0.46	0.72	0.90	1.27	1.93
4	0.87	1.26	1.52	2.05	2.95
5	1.36	1.88	2.22	2.88	4.01
6	1.91	2.54	2.96	3.76	5.11
7	2.50	3.25	3.74	4.67	6.23
8	3.13	3.99	4.54	5.60	7.37
9	3.78	4.75	5.37	6.55	8.52
10	4.46	5.53	6.22	7.51	9.69
11	5.16	6.33	7.08	8.49	10.86
12	5.88	7.14	7.95	9.47	12.04
13	6.61	7.97	8.83	10.47	13.22
14	7.35	8.80	9.73	11.47	14.41
15	8.11	9.65	10.63	12.48	15.61
16	8.87	10.51	11.54	13.50	16.81
17	9.65	11.37	12.46	14.52	18.01
18	10.44	12.24	13.39	15.55	19.22
19	11.23	13.12	14.31	16.58	20.42
20	12.03	14.00	15.25	17.61	21.64
21	12.84	14.89	16.19	18.65	22.85
22	13.65	15.78	17.13	19.69	24.06
23	14.47	16.68	18.08	20.74	25.28
24	15.29	17.58	19.03	21.78	26.50
25	16.12	18.48	19.99	22.83	27.72
26	16.96	19.39	20.94	23.88	28.94
27	17.80	20.30	21.90	24.94	30.16
28	18.64	21.22	22.87	25.99	31.39
29	19.49	22.14	23.83	27.05	32.61
30	20.34	23.06	24.80	28.11	33.84
31	21.19	23.99	25.77	29.17	35.07
32	22.05	24.91	26.75	30.24	36.30
33	22.91	25.84	27.72	31.30	37.52

$$\frac{0.46 \times 0.99}{3} \times 100 = 15 \% \text{ of the time.}$$

Thus the utilization factor is 15%. If this involves having three operators answering three lines, then they will only be busy 15% of the time; most of the time they will have nothing to do.

On the other hand, 30 trunks could handle 20.34 erlangs at 1% blocking, so the utilization factor is

$$\frac{20.34 \times 0.99}{30} \times 100 = 67\%$$

In this case, the 30 operators would be busy 67% of the time.

Efficiency experts often look at the utilization factor to judge how efficient an operation is, but this must be done very carefully. Look at this last example again: 30 operators handling 20.34 erlangs of offered traffic. But looking further up and to the right in the table, we see that 19 operators could also handle 20.42 erlangs of offered traffic, though at 20% of blocked calls. With 20% of blocked calls, 80% are not blocked, so the utilization factor is

$$\frac{20.42 \times 0.80}{19} \times 100 = 86\%$$

On the surface, this looks wonderful — only 19 operators instead of 30, and they are busy 86% of the time instead of only 67% — enough to make any bean counter proud. But the problem is that 20% of the customers are now being turned away because of busy circuits.

Basic program

The following program shows how to calculate the percentage of blocked calls with a simple computer program in Basic.

```

10 INPUT "How many trunks"; TRUNKS
20 INPUT "How many erlangs"; ERLANGS
30 NUM = 1: DENOM = 1
40 FOR I = TRUNKS TO 1 STEP -1
50 NUM = NUM * ERLANGS / I
60 DENOM = 1 + (DENOM * ERLANGS / I)
70 NEXT I
80 BLOCKED = NUM / DENOM
90 PRINT BLOCKED * 100; "% blocked."

```