A Robust and Efficient Algorithm for Evaluating Erlang B Formula

Sanzheng Qiao*
Department of Computing and Software
McMaster University
Hamilton, Ontario L8S 4L7 Canada

Liyuan Qiao
Customer Relationship Solutions, IBM Canada Ltd.,
3600 Steeles Ave. East, Markham, Ontario, Canada

October 17, 1998

1 Introduction

Call center and telephone system designers use the Erlang B traffic model to determine the number of central office (CO) trunks (lines) required based on estimated call traffic. The number of lines required is a function of busy hour traffic and the grade of service.

In this paper, we first introduce the formula involved in the Erlang B traffic model. We then analyze the formula for numerical evaluation. Finally, we present a robust and efficient algorithm for calculating the Erlang B formula. An on-line calculator is available on http://www.cas.mcmaster.ca/~qiao/.

2 Erlang's Formula

Let λ be the arrival rate. That is the mean number of arrivals per unit time, for example, the average number of calls per hour. Then λ^{-1} is the average time interval between two consecutive arrivals. Let μ be the service rate. That is the mean number of services per unit time. Then μ^{-1} is the average service time. For example, if the average call duration is six minutes (0.1 hour), then $\mu = 10$.

*This work was partly supported by the Natural Sciences and Engineering Research Council of Canada under grant OGP0046301

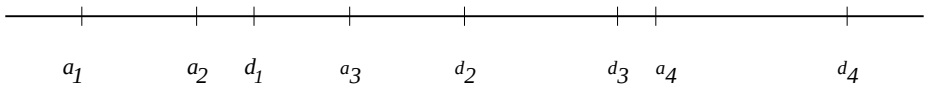

Figure 1: Arrivals and departures of four customers.

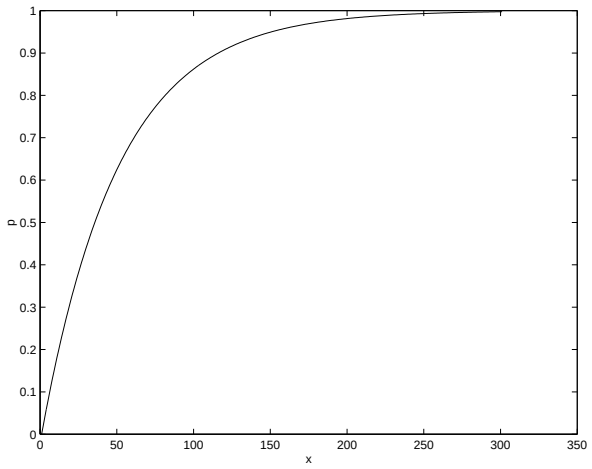

Figure 2: Exponential distribution.

Figure 1 depicts a discrete case. Four customers 1, 2, 3, 4 arrive at a_1, a_2, a_3, and a_4 respectively and depart at d_1, d_2, d_3, and d_4 respectively. The intervals, $I_i = a_{i+1} - a_i$, between two consecutive arrivals are called interarrivals. The arrival rate defined above is $\lambda = 1/E[I_i]$, where $E[I_i]$ is the expectation of the interarrivals I_i. The intervals, $S_1 = d_1 - a_1$, $S_2 = d_2 - d_1$, $S_3 = d_3 - d_2$, and $S_4 = d_4 - a_4$ are called service times. The service rate defined above is $\mu = 1/E[S_i]$.

Usually, we assume that the service time is a random variable and its distribution is exponential with parameter μ. Specifically,

$$\Pr(S_i < x) = 1 - e^{-\mu x}.$$

Figure 2 shows a typical curve of the probability that a service time is less than x.

Also, we usually assume that the interarrival is a random variable and its distribution is exponential with parameter λ: $\Pr(I_i < x) = 1 - e^{-\lambda x}$. Under this assumption, the arrival process is Poisson. That is the probability that the number of arrivals up to time t, denoted by $A(t)$, is n is given by

$$\Pr(A(t) = n) = e^{-\lambda t} \frac{(\lambda t)^n}{n!}.$$

In the example of Figure 1, $A(a_1) = 0$, $A(a_2) = 1$, $A(d_1) = 2$, etc. Figure 3 shows the

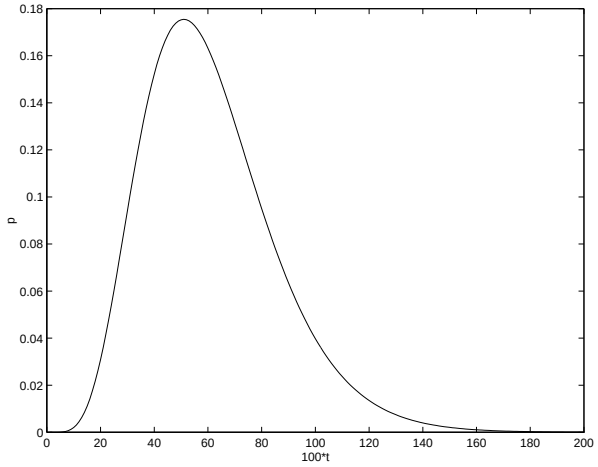

Figure 3: Poisson distribution where $\lambda = 10$ and $n = 5$.

Poisson distribution when $\lambda = 10$ and $n = 5$.

Furthermore, we suppose that the number of services, or the number of lines, is c and the capacity of the system, or the number of customers can be in the queue, is also c. In summary, we have a system in which

- Arrival process is Poisson distribution with parameter λ;
- Service-time process is exponential with parameter μ;
- Number of services (lines) is c;
- Capacity is also c.

This kind of system, using Kendall's notation [2], is denoted by M/M/c/c, Denoting the ratio of the arrival rate and service rate $\rho = \lambda/\mu$, Erlang's formula

$$p_c = \frac{\rho^c/c!}{\sum_{i=0}^{c} \rho^i/i!}$$

gives the probability that a new customer is turned away or the probability that all lines are busy in an M/M/c/c system. Surprisingly, the above Erlang's formula is also valid for general service-time process [2].

For example, suppose we expect 2,000 calls/hour, the arrival rate $\lambda = 2,000$. If the average call duration is six minutes, or 0.1 hour, then the service rate $\mu = 10$. Thus $\rho = 2000/10 = 200$. This ratio is also called BHT (busy hour traffic), since it is the product of the number of calls per hour and the average duration of each call. If the system capacity

$c = 245$, then Erlang's formula gives $p_c = 2.27 \times 10^{-4}$. That is the probability that a new call arrives, finds all lines busy, and is turned away is once every $p_c^{-1} = 4,400$ calls. Since the arrival rate is 2,000, that means the probability of getting a busy signal is once every two hours and 12 minutes. This probability is also called service grade.

3 Evaluating Erlang's Formula

The straightforward evaluation of Erlang's formula

$$p_c = \frac{\rho^c/c!}{\sum_{i=0}^{c} \rho^i/i!}$$

causes two problems: unnecessary overflow and inefficiency. In the IEEE floating point standard [3], the largest number can be presented in double precision is about 10^{308}. The factorial $c!$ grows very fast as c increases. In particular, 170! is about 7.26×10^{306} and 171! overflows to $+\infty$. Also, ρ^c grows very fast as c increases. For example, if $\rho = 100$, then ρ^c overflows to $+\infty$ when $c > 154$. As we know, ∞/∞ results NaN (Not a Number). To circumvent these problems, we reformulate Erlang's formula as following:

$$p_c = \frac{1}{1 + \sum_{i=1}^{c} \left(\frac{c}{\rho}\right)\left(\frac{c-1}{\rho}\right)\cdots\left(\frac{c-i+1}{\rho}\right)}.$$

This avoids unnecessary overflow. However, the ith product term of the summation in the above formula requires i divisions and $i-1$ multiplications. This yields about $c^2/2$ divisions and multiplications. Applying Horner's rule [1], we rewrite the summation

$$\frac{c}{\rho} + \frac{c}{\rho}\cdot\frac{c-1}{\rho} + \cdots + \frac{c}{\rho}\cdot\frac{c-1}{\rho}\cdots\frac{1}{\rho} = \frac{c}{\rho}\left(\cdots\left(1 + \frac{2}{\rho}\left(1 + \frac{1}{\rho}\right)\right)\cdots\right).$$

Implementing the above reformulated summation requires only about c divisions, multiplications, and additions. To conclude this section, we present the following robust and efficient algorithm for evaluating Erlang's formula.

Algorithm 1 (Erlang) *Given ρ and c, this algorithm evaluates Erlang's formula.*

/* Assert that both ρ and c are nonnegative and c is an integer. */
if $\rho = 0$, return $p_c = 0.0$;
$s = 0.0$;
for $i = 1 : c$
    $s = (1.0 + s) * (i/\rho)$;
end
Return $p_c = 1.0/(1.0 + s)$.

4 Finding the Number of Lines

In applications, we are often given the ratio ρ, i.e., BHT, and a desired grade of service, which is the probability p that all lines are busy, and we want to find the minimum number of lines (c) to satisfy p. In other words, find the smallest c such that $p_c \leq p$. In this section, we describe an algorithm for finding the number of lines using the algorithm for evaluating Erlang's formula presented in the previous section.

The basic idea of our algorithm is as follows. We first show that for a fixed ρ, the probability p_c is monotonically decreasing and approaches to zero as c increases. Then we propose to use bisection method to find the smallest integer c such that p_c is less than or equal to the desired probability. We discuss two issues in the bisection method: the initial interval and the termination criteria.

For any fixed ρ, the probability p_c is a monotonically decreasing function of c as shown below.

$$\begin{aligned} p_{c+1} &= \frac{\rho}{c+1} \cdot \frac{\rho^c/c!}{1+\sum_{i=1}^{c+1} \rho^i/i!} \\ &< \frac{\rho^c/c!}{\frac{c+1}{\rho}\sum_{i=1}^{c+1} \rho^i/i!} \\ &= \frac{\rho^c/c!}{\frac{c+1}{\rho}\sum_{i=0}^{c} \rho^{i+1}/(i+1)!} \\ &= \frac{\rho^c/c!}{\sum_{i=0}^{c}((c+1)/(i+1))(\rho^i/i!)} \\ &< \frac{\rho^c/c!}{\sum_{i=0}^{c} \rho^i/i!} = p_c \end{aligned} \tag{1}$$

It is obvious that $p_c = 1$ when $c = 0$ and 0 when $c = +\infty$. Thus given any p ($0 \leq p < 1$), there is a smallest integer c such that $p_c \leq p$ and the solution is unique. Figure 4 shows some curves of p_c for various values of ρ.

Because of the above properties of p_c as a function of c, we propose to use bisection method [1] to find the number of lines. For the bisection method, we must address two issues: initial interval and stopping criteria.

For a given p, we first locate an initial interval $[l,\ r]$ such that $p_l > p$ and $p_r \leq p$ where l and r are integers. We start with $l = 0$ and $r = \lceil \rho \rceil$ (the smallest integer greater than or equal to ρ). If $p_r > p$, then the solution lies between r and $+\infty$. We have found that

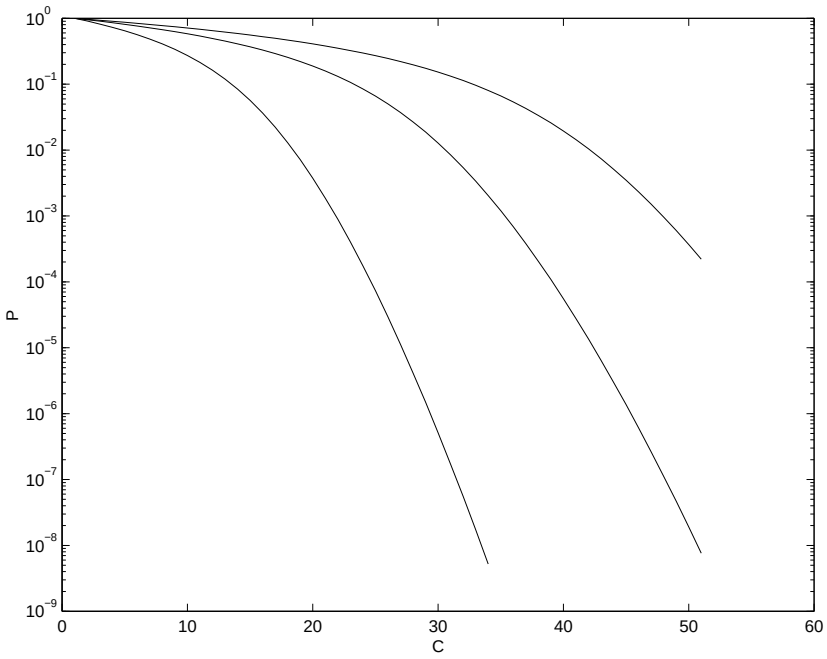

Figure 4: Probability p versus the number of lines c for various values of ρ. The top curve $\rho = 30$, middle $\rho = 20$, and bottom $\rho = 10$.

usually when $r = \lceil \rho \rceil + 32$, p_r is sufficiently small. So, we shift the interval by setting $l = r$ and $r = r + 32$. This procedure is repeated until we find l and r such that $p_l > p$ and $p_r \leq p$. After the initial interval is found, we apply the bisection method until the length of the interval reduces to 1. The loop invariance in the bisection method is $p_l > p$, $p_r \leq p$, and $(r - l) > 1$. Thus we return r as the number of lines when the loop terminates.

Algorithm 2 (FindLine) *Given a probability p and the ratio ρ, this algorithm computes the smallest integer c such that $p_c \leq p$. This algorithm uses function Erlang presented in the previous section.*

```
/* Assert that p is between 0 and 1. */
/* Assert that ρ is greater than a small positive tolerance. */
/* Start with interval: */
l = 0; r = ⌈ρ⌉;
/* Evaluate at the right endpoint. */
fR = Erlang(ρ, r);
/* Find an initial interval to include c. */
while fR > p
      l = r; r = r + 32;
      fR = Erlang(ρ, r);
end
/* Find c using bisection method. */
while (r − l) > 1
      mid = ⌈(l + r)/2⌉;
      fMid = Erlang(ρ, mid);
      if fMid > p
            l = mid;
      else
            r = mid;
      end
end
Return r.
```

After an initial interval is found, this algorithm takes at most five iterations to find the solution. Each iteration invokes only one function evaluation.

References

[1] Forsythe, G.E., M.A. Malcolm, and C.B. Moler, *Computer Methods for Mathematical Computations*, Prentice-Hall, 1977.

[2] Gross, D. and C.M. Harris, *Fundamentals of Queueing Theory*, John Wiley & Sons, 1974.

[3] IEEE, "IEEE Standard for Binary Floating-Point Arithmetic," *SIGPLAN Notices* 22:2, 9–25, 1985.