Evaluation of Overflow Probabilities in Resource Management

Dinesh Chandra Verma and Domenico Ferrari
The Tenet Group
University of California at Berkeley and
International Computer Science Institute

TR-91-051

October 1991

Abstract

In a number of network and database management applications, we need to evaluate an overflow probability, which is an upper bound on the probability that the capacity of a server will be exceeded. The problem can be essentially reduced to evaluating the probability that the sum of $N$ independent random variables exceed a given threshold. Evaluation of this probability by brute-force enumeration requires exponential time, so attempts have been made to approximate the overflow probability by using Chernoff bounds. This paper presents a simple scheme that can be used to evaluate the overflow probability with a higher degree of accuracy and lower computational efforts than the Chernoff bound approach.

This research was supported by the National Science Foundation and the Defense Advanced Research Projects Agency (DARPA) under Cooperative Agreement NCR-8919038 with the Corporation for National Research Initiatives, by AT&T Bell Laboratories, Hitachi Ltd., Hitachi America Ltd., the University of California under a MICRO grant, and the International Computer Science Institute. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing official policies, either expressed or implied, of the U.S. Government or any of the sponsoring organizations.
1 Introduction

In a number of situations dealing with network or database resource allocation, we need to compute an overflow probability. We define the computation of an overflow probability as the following problem:

Given \( N \) independent variables \( X_1, X_2, \ldots, X_N, X_i \) taking the value \( w_i \) (\( 0 \leq w_i \leq 1 \)) with probability \( p_i \), and the value 0 with probability \( (1 - p_i) \), compute the probability that \( X_1 + X_2 + \ldots + X_N > 1 \).

A situation where we need to compute such an overflow probability arises in the context of real-time communication in a packet-switched computer network [Ferrari 90]. Real-time communication is defined as the problem of providing performance guarantees to individual connections. Guarantees are made regarding upper bounds on packet delays and loss-rates by reserving network resources (like bandwidth, computational power and buffer space) on the basis of traffic characteristics declared at connection establishment time. Communicating applications declare their traffic characterization by specifying three parameters for each connection: \( x_{min} \), the minimum spacing between packets; \( x_{ave} \), the average spacing between packets over an averaging interval, and the maximum service time \( t \) of a packet at a node in the network. If a node is traversed by \( N \) real-time connections, the variable \( X_i \) is associated with the \( i^{th} \) real-time connection; the value of \( p_i \) is given by the ratio of \( x_{min} \) to \( x_{ave} \), and the value of \( w_i \) is given by the ratio of \( t \) to \( x_{min} \). A probabilistic delay bound can be offered to a connection by (among the tests that are needed) computing the overflow probability, as defined above, and verifying that it is less than a performance threshold specified by the application.

A very similar situation arises in the case of admission control for ATM networks. ATM, or the asynchronous transfer mode, is regarded as the most promising method for achieving a Broadband Integrated Digital Services Network, offering a range of communication services, such as voice, data and video services. A number of call admission schemes, which determine the criterion for call acceptance in such a network have been proposed ([Gallassi 90], [Esaki 90], [Todorova 90] among others). In some of these schemes, a call is characterized by two parameters, its peak bandwidth and its average bandwidth. Calls are accepted at a switch subject to the constraint that the probability of cell loss at that switch is less than a threshold specified by the ATM network manager. The computation of the call loss probability is identical to computing the overflow probability when variable \( X_i \) is associated with the \( i^{th} \) call in the network; the value of \( p_i \) is given by the ratio of the average bandwidth to the peak bandwidth of a call; and the value of \( w_i \) is given by the ratio of the peak bandwidth to the link or line speed of the ATM network.

Another situation requiring the computation of the overflow probability arises in the area of database management [Cohrs 90]. A database manager may wish to compute the probability that it is unable to satisfy the requests of its clients, and ensure that this probability is less than a predetermined threshold. This is another computation of an overflow probability, where the variable \( X_i \) is associated with the \( i^{th} \) database client; the value of \( p_i \) is given by the probability that this client would be requesting the databases’s services; and the value of \( w_i \) is the fraction of the capacity of
the database that the client’s request would require.

Given that the problem arises frequently in computer science and computer network applications, we expected that a suitable method to compute the overflow probability would be well-known and widely used. We were surprised to find that many of the people in the network community were using brute-force enumeration to determine this overflow probability, or else complex Chernoff bounds [Chernoff 52] to obtain an estimate of the overflow probability. The approach that we will be proposing would be much easier to compute, and clearly better than the Chernoff bounds approach. The approach is so simple that we decided it was too trivial for publication when we first discovered it in the context of real-time communication. However, the use of the more inaccurate and complex approximations in [Miyao 91] convinced us that presenting this work might be useful for the telecommunications and computer science community.

In Section 2, we explain our approach and show that it produces an upper bound on the probability of overflow. In the next section, we present a simple set of algorithms in C-like pseudo-code that can be used to evaluate the overflow probability. Then, in Section 4, we compare it to the best possible Chernoff bound that can be used, and study how accurate our approach is. We will present the algorithms in the context of real-time communication, that is, evaluate the probability of overflow when some number of real-time connections are established at a node. However, the work can be used in ATM admission control and other areas when an overflow probability needs to be computed.

2 The Approach

We consider the case of a node where connections are established. For the $i^{th}$ connection at the node, we define the variable $X_i$ as described in Section 1. Let us also define the variable $Y_N = X_1 + \ldots + X_N$. We are required to find the probability that $Y_N$ exceeds 1. Since connections can be added or deleted from the node and $Pr(Y_{N+1} > 1)$ depends on $Pr(Y_N > x)$ at values of $x$ other than 1, it is more efficient to maintain the whole distribution of $Y_N$.

Assuming that we define the tail distribution function $f_N(x) = Pr(Y_N \geq x)$, the plot of $f_N(x)$ would look like a series of steps as shown in Figure 1. The dotted lines show where the steps begin and end. Each step begins at the point where $x$ equals the sum of $w_i$ for some subset of $N$ variables $X_1 \ldots X_N$. In general, the (horizontal) width of the steps can be arbitrary and very small. In the worst case, there may be as many as $2^N$ such combinations, and therefore, $2^N$ steps in this function. Maintaining the distribution when adding or deleting any connection will take $O(2^N)$ time. The brute-force computation of the overflow probability $f(1)$ also takes $O(2^N)$ time. Thus, maintaining the distribution is no more expensive than computing the value of $f(1)$.

Figure 2 shows an approximation of the function $f_N(x)$ if we restrict ourselves to some fixed number of constant-width steps. The approximation uses the value of $f_N(x)$ at the beginning of a step as an upper bound on $f_N(x)$ throughout that step. In Figure 2, the constant-width steps are shown by
means of the dotted lines, the actual function \( f_N(x) \) by means of a solid line, and the bound on \( f_N(x) \) by means of a thick hatched line. It is clear that if all the actual function steps (that is, the vertical dotted lines in Figure 1) started at a location that was also the start of a constant width approximated step (the vertical dotted lines in Figure 2), the approximation would have identical to the actual probability distribution, that is, the solid and the thick hatched lines in Figure 2 would have been the same.

Suppose that each constant-width step in Figure 2 is \( \delta \) units wide and each \( w_j \) is an integral multiple of \( \delta \). Each of the dotted steps in Figure 1 corresponds to a some set \( C \) of connections, and starts at \( x = \sum_{i \in C} w_i \). This is an integral multiple of \( \delta \), and thus every actual function step would start
at a location that is also the start of a constant width step. In this case, we get the exact probability distribution. Clearly, as \( \delta \) becomes smaller, the approximation approaches the actual distribution, even if one or more \( w_i \) is not an integral multiple of \( \delta \).

The idea behind our fast approximation is to maintain a constant number of steps. Let us define the approximate function \( g(N,x) \) as the value of this bound on \( f_x(x) \), that is, \( g(N,x) \geq Pr(X_1 + \ldots + X_N \geq 1) \). Let us also define \( g(N,x) \) in the following manner:

\[
g(0,x) = \begin{cases} 1 & x \leq 0 \\ 0 & x > 0 \end{cases}
\]

(1)

\[
g(N,x) = \begin{cases} 1 & x \leq 0 \\ 0 & x > 0 \end{cases}
\]

(2)

and assuming that each of the steps in our approximation (shown in Figure 2) is \( \delta \) units long, we need to define \( g(N,x) \) only at the values of \( x \) that are integral multiples of \( \delta \). When we add a new connection, we have to define the new bounds at integral multiples of \( \delta \). At \( x = \delta m \), we define:

\[
g(N+1, \delta m) = (1 - p_{N+1}) g(N, \delta m) + p_{N+1} g(N, \delta (m - \lceil w_{N+1}/\delta \rceil)) .
\]

(3)

Similarly, let us delete a connection by the relationship:

\[
g(N, \delta m) = \left( \frac{1}{1 - p_{N+1}} \right) [g(N+1, \delta m) - p_{N+1} g(N, \delta (m - \lceil w_{N+1}/\delta \rceil))].
\]

(4)

Expanding the second factor on the right hand side repeatedly, and recalling that \( g(N, x) \) for negative or zero \( x \) is 1, we obtain that a relation equivalent to (4) is:

\[
g(N, \delta m) = \sum_{k=0}^{\lceil m/k \rceil} \frac{(-1)^k \delta^{k+1}}{(1 - p_{N+1})^{k+1} + 1} g(N+1, \delta (m - k\delta)),
\]

(5)

where \( h \) is defined as \( \lceil w_{N+1}/\delta \rceil \).

We now have to prove that \( g(N, \delta m) \) as defined by equations (1)-(3) provides an upper bound on \( f_x(x) \), when \( x \) is in the interval \([\delta m, \delta m + \delta]\). This property has to be shown since performance guarantees offered to connections may be violated if we use a value of the overflow probability which is lower than the actual overflow probability. The informal proof is offered by Figures 1 and 2, while the formal proof can be done easily by induction on the number of connections being established.

Theorem 1: \( g(N,x) \) as defined by equations (1)-(3) serves as an upper bound on \( f_x(x) \) for \( x \) in the range \([\delta m, \delta m + \delta]\).
Proof: For the case when $N=1$:

$$f_N(x) = \begin{cases} 1 & x \leq 0 \\ 0 & 0 < x < w, \end{cases} \quad (6)$$

and $g(1, \delta m)$ is described by:

$$g(1, \delta m) = \begin{cases} 1 & m \leq 0 \\ 0 & 0 < m \leq \lceil w/\delta \rceil \\ \lceil w/\delta \rceil < m \end{cases} \quad (7)$$

This clearly shows that $g(1, x)$ is an upper bound on $f_1(x)$. We assume that the theorem is true for some value of $N$, and show that it also holds for $N+1$.

$$f_{N+1}(\delta m) = (1 - p_{N+1}) f_N(\delta m) + p_{N+1} f_N(\delta (m - \lceil w_{N+1}/\delta \rceil)), \quad (8)$$

and by the induction hypothesis $f_N(x) \leq g(N, x)$, therefore:

$$f_{N+1}(\delta m) \leq (1 - p_{N+1}) g(N, \delta m) + p_{N+1} g(N, \delta (m - \lceil w_{N+1}/\delta \rceil)) = g(N+1, \delta m) \quad (9)$$

Since $f_{N+1}(x) \leq f_{N+1}(\delta m)$ for any $x$ in the interval $[\delta m, \delta m + \delta]$, the theorem is valid for $N+1$, and by induction valid for any $N$.

Theorem 2: The deletion of connections according to equation (4) is safe, in the sense that the bound $g(N, x)$ still remains an upper bound on the actual overflow probability.

Proof: Equation (4) is equivalent to equation (3) and obtained only by algebraic manipulation.

3 The Algorithms

In this section, we explicitly give the algorithms that need to be executed whenever a connection needs to be added to or deleted from a node.

The algorithm attempts to compute the array prob_tab which is an array of size NUM_ELEM. The value of $\delta$ is $1.0/(\text{NUM_ELEM}-1)$. The probability table is initialized by the routine prob_tab_init.

The probability that a non-negative random variable will exceed zero is always one. Thus, the first entry of prob_tab is initialized to one, while the other entries are initialized to zero.

Whenever a new connection is to be added, we can check what the new probability of over-
prob_tab_init(prob_tab)
float prob_tab[];
{
    int i;
    prob_tab[0] = 1.0;
    for(i=1;i<NUM_ELEM;i++) {
        prob_tab[i] = 0.0;
    }
}

flow is going to be by means of a few simple additions. Given the probability of activity of the new connection prob, and the weight of the new connection wt, the routine new_prob returns the value of the new overflow probability.

new_prob(prob, wt)
float prob, wt;
{
    int skip = [wt/\delta];
    int i;
    float answer;

    answer = prob*prob_tab[NUM_ELEM-skip]
            + (1.0 - prob) * prob_tab[NUM_ELEM];

    return(answer);
}

If a connection passes all the tests required for real-time communication (see [Ferrari 90] for details) and is to be added to the set of existing connections, the routine add_connection is invoked. It also uses the same parameters of prob and wt. The differentiation between new_prob and add_connection allows us to test if a connection can be safely added without updating prob_tab. Thus, the admission control tests can be made very rapidly.

add_connection (prob, wt)
float prob, wt;
{
    int skip = [wt/\delta];
    int i;

    for(i=NUM_ELEM;i>skip;i--)
        prob_tab[i] = prob*prob_tab[i-skip]
            + (1.0 - prob) * prob_tab[i];
    }
    for(i=skip;i>=0;i--)
        prob_tab[i] = prob + (1.0 - prob) *prob_tab[i];
}
When we need to delete a connection, we would need a temporary working array of the same size as prob_tab. The temporary array is used to store the previous value of prob_tab as we update the new values. The procedure is described by the routine delete_connection.

```c
void delete_connection(float prob, float wt)
{
    int skip = (int)(wt / 8);
    float tmp_arr[NUM_ELEM+1];
    for(i=0; i<skip; i++) {
        tmp_arr[i] = (prob_tab[i] - prob) / (1.0 - prob);
    }
    for(i=skip; i<NUM_ELEM; i++) {
        prod = 1.0 / (1.0 - prob);
        tmp_arr[i] = 0.0;
        for(j=0; j<i/skip; j++) {
            tmp_arr[i] += prod * prob_tab[i];
        }
    }
    printf("Now copy the temporary array as the new prob_tab \n/\nfor(i=0; i<NUM_ELEM; i++) {
    prob_tab[i] = tmp_arr[i];
}
*/
```

In the next section, we compare this simple bound against the approximation scheme proposed in [Miyao 91].

4 Comparison with Other Approximation Methods

Our approximation method is only one of the many possible ones that can be used to obtain an upper bound on the overflow probability. A number of other upper bounds can be found in literature ([Bennett 62], [Hoefding 63], [Fuk 71], and [Miyao 91]). All of these bounds are based on a Chernoff-type bound [Chernoff 52] which states that for any random variable $Y$:

$$Pr(Y > x + \mu) \leq E(e^{x^2}) e^{-x^2}$$

(10)

where $\mu$ is the expected value of random variable $Y$ and $x$ is any non-negative constant. Taking into account the fact that in this case $Y$ is a sum of $N$ independent variables ($Y = X_1 + \ldots + X_N$), we obtain
that

\[ Pr\left(Y > x + \sum_{i=2}^{N} p_i w_i\right) \leq \left( \prod_{i=1}^{N} \left( p_i e^{e w_i} + (1 - p_i) \right) \right) e^{-x}, \tag{11} \]

where the relevant values of \( \mu \) and \( E(e^{e Y}) \) have been substituted. Since the expected value of \( Y \) is the same as the expected utilization of a node, and thus strictly less than 1, equation (11) can be used to obtain a bound on the value of the overflow probability.

The different approximations in literature ([Bennett 62], [Hoeffding 63], [Fuk 71], and [Miyao 91]) are refinements of this bound, and are based on the selection of different values of \( s \). [Miyao 91] applies the Chernoff bound approach to the problem of admission control in ATM networks, and obtains the value of \( s \) which minimizes the value of the Chernoff bound by solving a transcendental minimization equation numerically.

Instead of implementing the numerical method for solving a complex equation, we tried to compare our approximation method by using the same workload that was used in [Miyao 91]. The workload consisted of a mix of two types of connections, one with a \( p_1 \) of 0.5 and a \( w_1 \) of 0.05, and the other one consisted of connections with a \( p_1 \) of 0.1 and a \( w_1 \) of 0.05. In [Miyao 91], the number of accepted connections using the approximate bounds differed by more than 10% (for example, one could only accept 58 of the first type of connection instead of 70 connections that they would have accepted using the exact computation).

How does that compare with our approximation method? Suppose we use 21 bins; then \( \delta \) (the width of an approximation step) is 0.05, the same as the value of \( w_1 \), and we obtain the accurate value of the overflow probability. This would have been true also for other values of \( \delta \) which have been selected to make \( w_1 \) integral multiples of \( \delta \), e.g. \( \delta = 0.01 \) or 0.025 (corresponding to \( \text{NUM}_E = 101 \) or \( \text{NUM}_E = 51 \)). Thus, there would be no decrease in the number of accepted connections using our approximation. If there are only a few types of connections, as assumed in [Gallassi 90] or [Miyao 91], then the choice of a good value of \( \delta \) is very easy to make.

How inaccurate are we in the cases where the types of connections are not known in advance, where \( w_1 \) could be random and not an exact multiple of \( \delta \), and how does this inaccuracy depends on the number of approximation steps (\( \text{NUM}_E \))? In order to answer this question, we present the value of the approximate overflow probability computed for different values of \( \text{NUM}_E \). Ten connections with \( w_1 \) and \( p_1 \) uniformly distributed between 0 and 1 were considered. The value of the exact overflow probability was 0.02061. We show the approximate values of the overflow probability obtained for different values of \( \text{NUM}_E \) in Figure 3.

It is obvious from Figure 3 that the use of a larger value of \( \text{NUM}_E \) results in a more accurate bound. We found that about 1000 bins are adequate to obtain a reasonably accurate value
of the overflow probability in most of our simulations of ATM call admission policies, as well as in the context of real-time communication.

5 Conclusions

In this paper, we have presented a method for evaluating overflow probability, a problem that arises in a number of resource management situations, and especially in the case of admission control in guaranteed performance networks or in ATM networks. The algorithm is simple and efficient, provides provably upper bounds on the overflow probability, and is more accurate than Chernoff type bounds. We hope that this simple algorithm will be useful to the telecommunication and computer science research community and to network managers who need to evaluate a similar overflow probability.

6 Acknowledgments

The authors would like to thank Y. Miyao for suggesting that this work needs to be published for the telecommunication community.

7 References


