Empirical Methods for Call Admission Control

Dr. R. K. Mehra and Dr. M. Perloff
Scientific Systems Company Inc., Woburn, Massachusetts
rkm@ssci.com mike@ssci.com

Abstract

Effective Call Admission Control (CAC) methods are needed to provide Bandwidth on Demand and meet Quality of Service requirements. Current methods are often excessively strict, allowing available bandwidth to go unused, and depend on accurate estimation of traffic description parameters. We formulate and test a simple method for estimating Cell Loss Rate from discretized queue size measurements and using the estimate in Admission decisions. Simulation test results show that our estimation method will improve preformance of current methods by correcting for incorrect traffic parameter estimations.

1 Introduction

Two objectives of ATM switching are to provide "bandwidth on demand" - bandwidth to connections exactly when it is needed - and to provide Quality of Service - to meet specifications on delay, delay variance, and cell loss. Since new connections interfere with Quality of Service by increasing queueing delays and queue sizes, it is important to prevent the entry of new connections that could prevent the network from meeting quality of service requirements. Call Admission Control algorithms do this by determining the effect a new connection will have on current connections and allowing the new connection only if all service requirements - for both the current and new connections can be met. Overly strict admission criteria - e.g., admitting new connections only if the sum of their peak rate and those of current connections is less than switch bandwidth - easily meet QoS requirements, but - by reserving bandwidth instead of using switch queues for infrequent simultaneous bursts from several connections - deny service to new connections whose service requirements could easily be met. Inadequate admission criteria will provide a much higher flow of cells through the switch, but will cause excessive cell loss when simultaneous bursts from several admitted connections overflow switch queues. Optimized, adaptive CAC algorithms will allow switches to run at the highest utilization levels consistent with QoS requirements.

The most commonly used current CAC algorithm, Equivalent Capacity, provides a formula for approximating the bandwidth necessary to meet delay and loss requirements under certain simplifying assumptions on network traffic (e.g., that the traffic is Markov, and modelled as a fluid) (see [1] and [2]). It suffers from two important problems that diminish its effectiveness:

- It is a conservative approximation - so that, even when applied to traffic that meets the conditions on which the formula is based, it produces lower levels of utilization than is needed to meet QoS requirements. To rectify this, switches allow for an "overbooking factor", that forces the EC computation to use a higher bandwidth than is actually available and thereby admit more traffic. If the overbooking factor is set too high, or if the traffic characteristics change, too much traffic is admitted and cell loss standards are not met.
- The traffic description inputs required by the EC algorithm (Peak rate, mean rate, maximum burst size) are often not known for real-time video or voice traffic. Consequently, estimates are used in the EC computation. Once again, the estimates are frequently conservative, resulting in low utilization and inefficient operation.

To rectify these problems and improve switch utilization, we have developed empirical methods for estimating cell loss rate and doing the CAC computation. Empirical methods will increase the accuracy of CAC computations by making fewer simplifying assumptions on network traffic - since it is based on direct measurements and computational assumptions can be verified by statistical tests. Furthermore, empirical methods do not rely on a-priori estimates of traffic parameters and allow the network to adapt when traffic loads are smaller (or larger) than initially expected. Empirical CAC methods can be self contained, or can work as a correction to EC to increase switch utilization, or can be used to gather traffic statistics to support improved future admission decisions.

These new CAC methods were part of a larger ATM traffic management study including extended ABR methods, routing, and scheduling. The full report is found in [3].

2 Basic Method

We have focused on a simple empirical method for estimating cell loss rate for VBR Traffic on ATM switches. Our method uses simple, discrete measurements over discrete time steps to minimize storage and computation. We have tested it on a variety of simulated VBR traffic mixtures. We have also used it to adjust the overbooking factor and improve the performance of an Equivalent Capacity computation.

Our method consists of three main steps. First we periodically sample the queue size. We use the queue size data to obtain a histogram of the queue size samples and of the changes in queue size between measurements. These queue-change measurements are filtered (as explained below) and histogrammed. After we obtain a specified number of valid queue-change measurements, we do a computation to estimate the cell loss rate. A comparison of these cell loss rate estimates and the target cell loss rate is used to adjust the Equivalent Capacity overbooking factor so that cell loss service requirements are met. Consequently, the estimation procedure helps adjust the Equivalent Capacity algorithm to actual traffic conditions. To histogram the queue sizes and queue-changes, we use the following procedure:

- Q_n = Measured queue size at time n.
- B_n = Q_n div B, where "div" is integer division and B is the bin size.
- D_n = Q_n - Q_{n-1} is the measured queue-change at time $n > 0$ and
- D'_n = D_n div B.

Ultimately we want to histogram the B_n's and D'_n's and use them to estimate the cell loss rate. To do this, however, we must first discard unusable measurements and unbias the observed distributions. There are two cases in which observed measurements are not directly usable:

- The queue has changed from any value to a value in the highest bin. In this case the underlying arrival process may have been sufficient to cause the queue to attain a larger value than the highest bin size and was stopped by the limited queue size (and the resultant discards) - in which case the queue-change measurement is artificially low. Since the measurement may be inaccurate, it is discarded and not used in making the histogram or in subsequent computations. Although it may be possible to measure discards and compensate for this truncation error, discards are not currently measured and used anywhere in our CLR estimate algorithm.
- The queue has changed from any value to the a value in the lowest bin. In this case the underlying arrival process may have actually caused a more negative queue transition which was not observed since the initial queue value was not high enough - i.e., had the initial queue been higher, more cells would have been serviced, and the resulting queue dimunition would have been greater. Note that this case includes transitions from the lowest bin to the lowest bin -which turn out to be the most frequently observed transitions.

The first type of unusable measurement occurs only when the queue is very high and when there is likely to be cell loss. Since the objective is to keep cell loss low (so low that it cannot be sampled and estimated directly), this condition results in few measurement discards. The second condition occurs frequently, and if unrectified, would substantially increase the time required to obtain CLR estimates. Consequently we have implemented a method to compensate for the inaccuracy and make use of at least some of these measurements:

In the case where the queue changes from bin value B_n >= 1 to the lowest bin value (0), the number of idle slots in the measurement period tells the number of extra slots that could have been serviced had the initial queue been large enough. Consequently, we replace the observed (negative) queue change D_n = Q_{n+1}-Q_n with the more negative value Q_{n+1}-(Q_n+I_n), where
I_n = Number of idle slots in the measurement interval.
Since idle slots are not routinely monitored, we derive the number I_n from the arrivals and the measurement interval length. If
A_n = Arrivals in time interval n, and
S = Cell Slots in interval n (Interval length in cell service times)
then
Q_{n+1} = Q_n + A_n - (S - I_n)
since Q_n + A_n are the cells available for service and (S-I_n) were the actual number of cells serviced. Consequently, the adjusted measurement we use is:
Q_{n+1}-Q_n-I_n = A_n - S .
There is one more step before the queue change histogram can be computed. Even though the underlying arrival process may generate large upward queue-changes at any time, positive, usable queue-change transitions can only be observed from initial states (queue size bins) small enough that the queue change does not go to the largest bin. Similarly, even with the adjustment above, negative queue changes can only be observed from initial queue-sizes in the first bin or higher (not the lowest bin). This means that the observed queue-change frequencies are not accurate estimates of the actual underlying queue-change frequencies (assumed to be driven by the arrival process, independent of the current queue size), but rather of the

events
{Q_{n+1} - Q_n in Bin n} AND {Current queue size allows queue change to be observed}.
Under the assumption of independence, we must compensate for this bias by dividing by the probability that the queue-change can be observed: Pr $\{Queuesize >= B\}$, for negative queue changes, and: Pr $\{Queuesize < Bmax - Queue - change\}$, for positive queue changes. Notice that, without this correction, since the system starts with an empty queue, and, if the traffic load is less than capacity, must necessarily return to the empty queue state, the number of the negative transitions observed (or net expected negative transition) must equal the number of positive transitions (or the net expected upward transition) (respectively)- so that the resulting system, based on the directly observed queue-changes, would be unstable. The adjustment above requires queue size probabilities. We use the observed queue-size histogram to supply these probabilities.

The resulting algorithm for obtaining the queue-change probabilities is then:

- At each sampling time:
  - Obtain measurements of A_{n+1} and Q_{n+1}, where
  - A_{n+1} = number of arrivals between times n and n+1, and
  - Q_{n+1} = the queue size measured at time n+1.
  - Obtain the queue size histogram point h = (Q_{n+1} div B),
    where B is the bin size.
  - If $h > 0$ obtain the queue-change histogram point:
    d = ((Q_{n+1}-Q_n) div B). Record this value if $d > 0$ and $h < Bmax$. If $d < 0$ and h = 0 and the previous value of $h > 0$, replace d by
  - (A_{n+1} - S) and record the value of d. If d is recorded, increment the number, N, of valid queue-change histogram points. If no value is recorded, the queue-change measurement is not usable and is discarded.
- After the number, N, of valid queue-change measurements reaches a threshold, M:
  - Compute the unadjusted queue change frequencies, the number of values in each bin divided by M, the number of measurement points.
  - Compute the queue histogram frequencies - the number of values in each bin divided by the number of histogram measurements (which will generally be greater than M).
  - Adjust the queue-change histogram:
    * for each negative queue change bin, replace F_m (the observed frequency that the queue-change is in bin m) by $F_m/(f_1 + ... + f_{max})$ where f_i is the observed frequency that the queue size fell into bin i.
    * for each positive queue change F_m, replace F_m by $F_m/(f_0+...+f_k)$, where $k+m < max$, and f_0, ... f_k are as above.

The resulting adjusted queue-change histogram is used to compute the estimated long-term queue-histogram and the estimated CLR.

The CLR computation (from the queue-change histogram) is done in two main steps. First, we obtain an estimate of the queue histogram from the queue-change histogram and then we obtain a CLR estimate from the estimated queue histogram. This computation implicitly relies on a number of assumptions:

- the queue change process is Markov, i.e., queue changes depend only on the current state of the system. This assumption is justified if user traffic bursts begin and end independently of the state of the current system and independent of past history (as they do in Markov Modulated Poisson Processes (MMPP)), and if the sampling interval is long enough and the number of users large enough that a significant number of bursts end and begin during the sampling interval.
- the queue change probabilities are constant when the queue size is not in the lowest bin. In low traffic periods, when total traffic is substantially less than link bandwidth, additional user traffic bursts generate at most small, transient queues. In higher traffic periods, the generation or termination of traffic bursts is independent of queue sizes at a switch.

With these assumptions, we view the queue as a discrete state Markov Chain with states corresponding to the queue size bins and transition functions given by the queue-change histogram (computed above). To compute the estimated long-term queue histogram, start with any initial queue size histogram and apply the queue-change probabilities to obtain a new queue-size histogram:

- Set P'[n] = P[n] for each n, $n <= 0 <= M$

- for each n and m:
  - Compute: dP[m,n] = P[n] * q[m-n]
  - Decrease P'[n] by dP[m,n]
  - If $(0 < m + n < M)$ then increase P'[n+m] by dP[m,n]
  - If $(m + n <= 0)$ then increase P'[0] by dP[m,n]
  - If $(m + n >= M)$ increase P'[M] by dP[m,n]

where the qsize bins are numbered by n, $0 <= n <= M$, in order, and
the queue-change bins are numbered by m, $-M <= m <= M$, also in order, and
P[i] is the initial probability of the system being in state i (having a queue size in the i-th bin), and,
q[m] is the probability of a queue size change of m bins, and,
P'[i] is the new, computed probability of the system being in state i.

This computation is repeated, iteratively, until the distributions P' yield a stable CLR estimate. To compute the CLR for an estimated queue-size distribution, P, perform the following computations:

Find the average upward probability p_{up}:
p_{up} = q[1] + q[2] + ... + q[M]
CLR(P) = P[M]*p_{up}

Thus the CLR is estimated in the simplest possible way as the probability of moving upward from the highest queue-size bin. We experimented with more elaborate CLR estimates - e.g., estimating the cell loss from each transition and dividing the total cell loss by the average number of cells serviced in a measurement interval - but the simple estimate above provided a consistent, reliable, moderate overestimate of the observed CLR in nearly all of the test cases.

The overall computation is then:

- Start with P = P_0, where P_0[0] =1; P_0[n] = 0, for $0 < n <= M$.
- Compute P' from P according to the algorithm above.
- Compute CLR(P')
- if $abs(CLR(P') - CLR(P)) > 0.01 * CLR(P')$ then repeat the overall computation, replacing P with P'. Otherwise the computation is finished and the CLR estimate is CLR(P'). (To avoid some trivial but incorrect cases, we require at least 10 iterations and that CLR(P') be greater than 0. The number of iterations is also upper bounded at 50).

The parameters required by the algorithm are:

- Bin size B
- Number of bins M (so the maximum queue size is B*M)
- Measurement interval, S, (measured either in seconds or in cell slots)
- Maximum number of iterations in the CLR computation.

In our testing, the value of B above did not generally affect the results significantly. The most important parameter is S, which must be chosen large enough that successive measuring intervals are uncorrelated (but which cannot be chosen so large that substantial movements in the queue are missed).

Finally we propose a specific method for using the computed CLR above in admission tests. We use the estimated CLR to adjust the overbooking factor in an EC computation. This provides a method for obtaining and using empirical measurements to compensate for excessively conservative EC estimates and to compensate for low EC estimates caused by over-estimation of traffic parameters. It also allows the CLR computations to be carried out in the background, independent of specific traffic admission decisions and eliminates the need to carry state variables necessary for adjusting the CLR estimate calculation to determine the effects of adding new traffic. The main features of this algorithm are:

- the CLR is estimated periodically according to the procedure above.
- if the CLR differs significantly from the target CLR, the overbooking factor is adjusted incrementally (to either inhibit or encourage subsequent admissions).
- if there is an excessively long period in which the queue stays near zero (in the lowest bin), the overbooking factor is increased. (A step like this is necessary, since, if the system is very underloaded, there will be no queueing, and, since queue transitions from the lowest bin to the lowest bin are not counted or used in the CLR computation, there will be no CLR estimate obtained to use in the two steps above).

The precise steps are:

- when a CLR estimate, c, is obtained:

- Exponentially average the CLR estimate with the previous CLR estimate to obtain a new, averaged value of c. This averaging allows us to use previous estimates to obtain a CLR estimate value that uses previous data while still allowing the estimate to adjust to the addition or subtraction of traffic connections.
- if ($c < p/F$) increase the overbooking factor by a factor of 1+G.
- if ($c > Fp$) decrease the overbooking factor by a factor of 1-G , where:
- p is the target probability of loss,
- and F and G are given.
- If there is a sustained idle period (20 CLR estimation periods with no usable data points), either increase the overbooking factor by a factor of (1+G) (if the CLR estimate is low enough according to the test above) or decrease the current CLR estimate by a factor of (1-G).
- In any case, do not allow the overbooking factor to exit a predefined range.

3 Testing

To test the CLR estimation procedure, we applied it to a number of queue traces generated from VBR traffic. In generating the traces we used packet level simulation - where each burst from the VBR traffic source was signalled by one packet. At packet arrival, we computed the change in the queue and arrival growth rates from the new burst and scheduled an event for the end of burst (at which the queue and arrival growth rates were once again recomputed). At periodic intervals when the queue length was sampled for CLR estimation, the queue length, arrivals, and discards were computed from the growth rates. To confirm this approximation method (which ignores very fast queue variations coming from the exact pattern of arrivals), we compared the results from several cases with exact cell-by-cell simulations. In each case, the queue traces were visually indistinguishable, differing by at most a few cells. The discard and CLR computations from the packet level simulation were also very close to the values obtained from the cell-by-cell simulation. After defining a number of VBR connections similar to those used in the literature for testing CAC algorithms we carried out the following steps:

- For each VBR connection type, we tried to establish the CLR and its statistics by conducting 20 or more simulation runs with different random number seeds and computing the CLR for each run. These runs enabled us to obtain an average CLR, a variance, and an approximate confidence interval. Packet level simulation speeded up run times and enabled these estimates to be obtained quickly.
- We then generated a number of additional traces of switch activity from the VBR connection and generated a number of independent estimates of the CLR (with the algorithm above) from each trace.
- We determined the average of the CLR estimates (from the step above) and their spread to characterize the performance of the CLR estimation procedure. An ideal estimation procedure will have an average close the average we obtained in the first step above and will have all estimates close to the average, and will prefer errors in which the CLR is overestimated to errors in which the CLR is underestimated.

Name	Idle(ms)	Active (ms)	PCR (cps)	MCR (cps)	Average Burst (cells)
VBR3	117	13	27307	2730	355
VBR6	104	26	13654	2730	355
VBR8	78	52	6827	2730	355
VBR3a	117	13	81921	8192	1065

Table 1: CLR Estimation Testing: Traffic Models

Name	Mixture 1	Mixture 2	Mixture 3
VBR3	120	60	30
VBR6	0	60	0
VBR8	0	0	30
VBR3a	0	0	20

Table 2: Test Traffic Mixtures. Table entries in each column are the number of connections of each type in the mixture.

For a representative sample of the test results, the two tables (Tables 1 and 2) give statistics for the four types of VBR connections and three mixtures of these connections. Each VBR connection type (whose statistics are given in the table - average idle period, average burst period, peak cell rate, average cell rate, and average burst size) occupies (on the average) about 1/150 of the bandwidth of the switch used in testing (which had a capacity of 409600 cells per second). Each of the mixtures represents about 80% loading on the switch. By the procedure outlined above, we estimate the following cell loss rates and standard deviations for each of mixtures (CLR's and their standard deviations are given as multiples of 10^{-4}):

- Mixture 1: CLR = 3.8, SD = 2.0 with a 5000 cell

switch queue.

- Mixture 2: CLR = 17.5, SD = 3.7 with a 2000 cell switch queue.
- Mixture 3: CLR = 46, SD = 10.5 with a 5000 cell switch queue.

Although these test loads and cell loss rates are higher than QoS requirements would normally permit, cell loss rates in the range of 10^{-4} can be estimated with simulation runs consisting of tens of millions of cells; about the highest that can be done with acceptable run times. Finally we generated additional, independent, traffic and queue traces for each of the mixtures and applied the CLR estimation procedure above to these traces. We compared the average of the CLR estimates with the empirical CLR that we obtained directly from simulation (above) and noted the low and high estimates. The most desirable procedure would have an average close to the actual CLR, estimated above, with a narrow spread and more overestimation errors than underestimation errors (since overestimation leads only to conservative admission procedures, while underestimation can lead to unacceptable service). The specific procedure tested here was designed to give a moderate overestimate of the CLR, so it should have an average value in excess of the empirical CLR and few, if any, severe underestimates.

The results of the test procedure above are shown in Table 3. The table shows the average estimated CLR, the number of estimates, and the lowest and highest estimates. For mixtures 1,2, and 3, the queue was sampled every 10 ms., and a new CLR estimate was obtained either every 30 seconds or after 30 acceptable queue-change measurements were obtained (whichever came last). In general, the results shown in Table 3 indicate that the CLR estimation procedure works well for the tested traffic mixtures: most readings are high, but of the same order of magnitude as the simulation CLR values (above); few are more than one standard deviation lower than the simulation CLR values. Average values for the CLR estimates are within the range 1 to 2 times the actual (simulation) values.

4 Conclusions

The sections above described a simple, new procedure for CLR estimation from queue monitoring. Our CLR estimation procedure is capable of estimating the CLR from queue data only (no actual cell loss data is required) and, consequently, is capable of estimating the low CLR's required for VBR QoS requirements from queue data in periods in which no actual losses occur. The method above gives fairly reliable CLR overestimates on a variety of test traffic mixtures after about 30 seconds of sampling time. We also tested more precise methods for estimating the CLR from the observed queue-change data; the average estimate was closer to the actual value (here derived from simulation), but there were more unsatisfactorily low estimates. The test results show that our CLR estimation method should improve performance of current EC CAC by correcting for incorrect traffic parameter estimates used for setting up ATM traffic contracts. The procedure developed here cannot use queue data from periods in which the queue is low (or zero) and, consequently, may not be able to generate CLR estimates for long time periods. New procedures that would enable this data to be used could decrease the time required to obtain an estimate.

Traffic Mixture	Average CLR Estimate	Data Points	Low	High
Mixture 1	7.95	79	1.88	18.51
Mixture 2	38.5	66	15.2	70.9
Mixture 3	47.1	79	26.73	84.5

Table 3: CLR Estimation Tests - Summary Results

5 Acknowledgment

This work was performed with support under contract F30602-96-C0156 from the Air Force Research Laboratory.

References

[1] Raif O. Onvural, Asynchronous Transfer Mode Networks, Performance Issues
Artech House, 1995, pp148-158

[2] G. Guerin H. Ahmadi M. Naghshineh, Equivalent Capacity and its Applications to Bandwidth Allocation in High-Speed Networks
IEEE Journal on Selected Areas in Communications, vol 9, page 968-981, 1991

[3] R. K. Mehra M. Perloff D.N. Greve J. D. B. Cabrera R. Jain T. Basar and R. Srikant
Adaptive Intelligent Scheduling and Traffic Management for ATM Networks,
Final Technical Report, AFRL-IF-RS-TR-1999-220, Air Force Research Laboratory, Information Directorate, Rome New York