An experimental configuration for the evaluation of CAC algorithms

Andrew Moore and Simon Crosby
University of Cambridge Computer Laboratory,
New Museum Site, Pembroke Street
Cambridge, CB2 3QG, UK
{andrew.moore,simon.crosby}@cl.cam.ac.uk

Abstract

Interest in Connection Admission Control (CAC) algorithms stems from the need for a network user and a network provider to forge an agreement on the Quality of Service (QoS) for a new network connection. Traditional evaluation of CAC algorithms has been through simulation studies. We present an alternative approach: an evaluation environment for CAC algorithms that is based around an experimental test-rig. This paper presents the architecture of the test-rig and an evaluation of its performance.

1 Introduction

Connection Admission Control (CAC) denotes the set of actions taken by the network during the connection set-up in order to accept or reject an ATM connection. A connection request is only accepted if sufficient resources are available to carry the new connection through the network at its requested Quality of Service (QoS) while maintaining the agreed QoS of existing connections. It can be seen that while attempting to balance the requirements of the "user" (achieve the desired QoS,) and the requirements of the "network" (do not violate the QoS guarantees made to existing connections,) the algorithm controlling the decision made during the CAC will control the policy of the network.

During the development of CAC algorithms, substantial effort has been invested in modelling and experimenting using simulators such as **ns** [1]. In these simulators, models are made of all aspects of the situation including traffic and network behaviour. However, modelling alone does not satisfactorily assess the behaviour of real CAC algorithms implemented in real situations. Most common simulators, such as **ns**, are discrete event simulators not constrained by CPU or memory usage in the same way an actual switch implementation would be. Additionally, common modelling techniques involve the use of simulated sources of traffic – both well understood and easily generated. However, due to the variety of sources and the continual development of new network users, such simulated sources of traffic are not adequately representative. In a test environment that uses real rather than simulated traffic sources, testing CAC algorithms against the latest traffic and network behaviour is simplified. While we do not advocate the wholesale replacement of simulators with evaluation systems based upon implementation, both have an important role to play.

Section 2 gives a brief summary of how existing test implementations have either not been suited for use in the evaluation of CAC algorithms, or that their limited availability has restricted their usefulness. The architecture of the test environment is discussed in Section 4. Additionally, the components and the operation of the test-rig are also covered. Finally, an evaluation of the test-rig is given in Section 5; this evaluation covers the test-rig's per-connection and repeatability performance.

2 Previous work

Test environments that are able to assess a CAC algorithm require the ability to dynamically generate traffic as the connection load varies, the ability to extract information to be used by the CAC algorithm (as required) and to extract information on the current traffic allowing us to assess the behaviour of the CAC algorithm.

Assessment of network behaviour is a common requirement of network management, as a result systems such as Chen et al. [2] and Lazar et al. [3] are designed to allow only the dynamic collection of measurements. In contrast, several systems [4, 5] were built to control and assess the behaviour of complete networks and from these systems, CAC evaluation environments are more plausible. Aneroussis et al. [4] document the implementation of a novel CAC approach and so it seems a reasonable conclusion that the system could be used to assess different CAC algorithms.

More recently from this same research group, Lazar & Nandikesan [6] describe a test environment built around the Hewlett-Packard Broadband Series Test System (HPBSTS). While it seems likely that this system could be used to evaluate CAC algorithms the parts of the system are based around a limited release of proprietary information; that, combined with the high cost of the HPBSTS itself, makes it difficult to recreate this test environment.

From this brief review, it can be seen that a system to allow the assessment of a wide range of CAC algorithms and one that is able to use a wide variety of network traffic and network conditions is not readily available. Firstly we will discuss the CAC algorithm used in this evaluation.

3 Simple Threshold CAC algorithm

A threshold based CAC mechanism is one that allows a new connection to be admitted into the network if the measured traffic level is equal to or below a predefined level or threshold, the actual process for calculating the threshold is unimportant at this stage.

The threshold based CAC algorithm in operation is shown in Figure 4(a). Connection A requests a connection into the network. The CAC makes a current bandwidth sample; the value is below the pre-calculated threshold. The CAC can admit the new connection A into the network. Now new connection B attempts to connect to the network. The CAC makes another sample of the current bandwidth; the value now is above the pre-calculated threshold. Because the value is above the pre-calculated threshold the CAC rejects the new connection B, not allowing it into the network. The following Section describes the architecture of the test-rig we built.

4 CAC Evaluation Environment

The CAC test-rig consists of a combination of hardware of the ATM switch and ATM interface cards, as well as software to generate new connections, perform CAC operations, obtain measurements from the ATM switch, generate traffic sources and control the generation of traffic sources. Figure 1 shows the implementation architecture adopted to evaluate CAC algorithms. Also specific components of the CAC test-rig are discussed along with an outline of the test-rig's operation.

4.1 ATM switch

The ATM switch component must control where cell loss will occur and allow variables such as buffer size and buffer service rate to be controlled. In addition to being a controllable buffer, the ATM switch also makes measurements of line utilisation, cell arrivals and cells departures. Using these measurements, the CLR of the line and indeed the utilisation and CLR per connection can be determined.

Our implementation, based around a commercially available ATM switch, a Fore ASX-200WG, is shown in Figure 2. To ensure cell loss occurs in the controlled buffer, the service rate is reduced by $1/D$, where D is a chosen integer. In Figure 2 each link is labelled with its transmission-rate relative to the full line rate of 1. The transmission rate of each traffic source is scaled by a factor of $1/D$. The rate of the interface between the input port A and the buffer for output port B is at the full capacity of the switch. The speed of the output port, B, is scaled by the same quantity as the traffic sources, $1/D$.

Traffic arriving at the buffer will queue in the output buffer of port B. The switch has a switching capacity of D times the input and output transmission rates, and is therefore effectively non-blocking. In this way we achieve a controllable buffer the parameters for which we can set (queue length) and about which measurements can be taken (cell loss, cell delay). We are able to use the switch buffer configuration, such as traffic classes, intelligent discard policies and scheduling systems to be constructed, however at this stage our work has been with a single FIFO cell buffer aggregating the traffic of all connections present.

The ATM switch also makes measurements, counting the cells moving into and out of the buffer in a given period of time. From these measurements the line utilisation and cell loss can be calculated; thus the utilisation and cell loss can be calculated on a per-connection basis. These counts of cells traversing the system are transmitted to an external *measurement controller*, so as to reduce the work-load on the ATM switch itself.

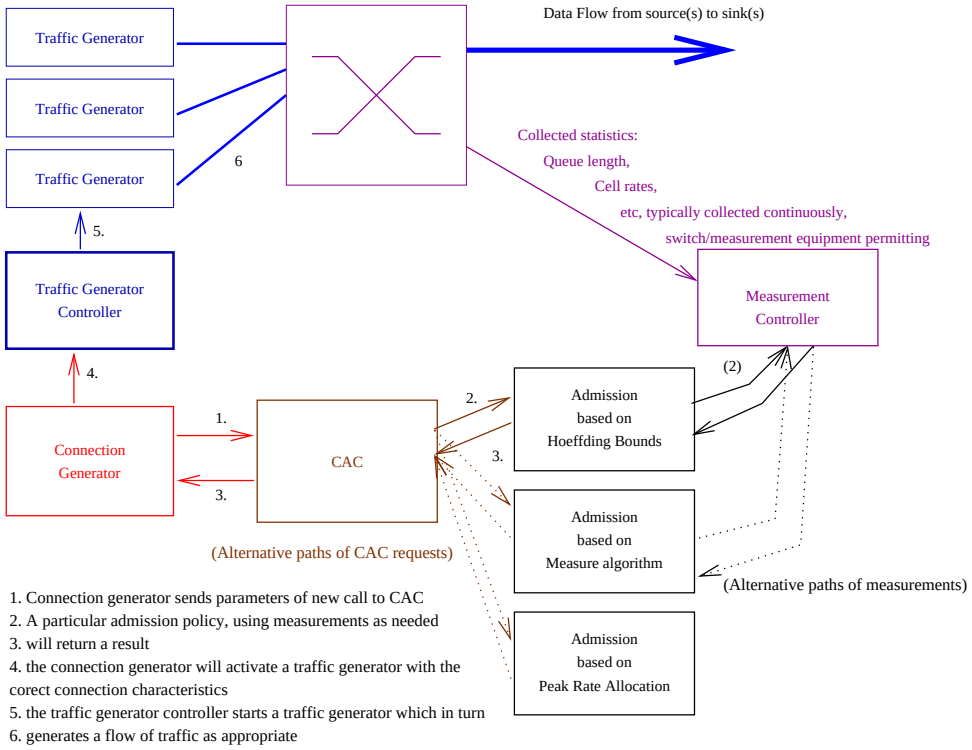


Figure 1: Architecture for the implementation of a test environment to evaluate CAC mechanisms.

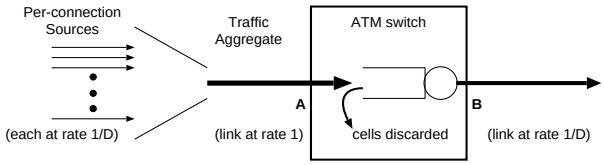


Figure 2: Topology of ATM switch.

4.2 Measurement controller

The measurement controller, a process running under Unix, obtains from the switch the measurements that may be required as input into a CAC algorithm. Additionally, the measurement controller collects not only input for the CAC algorithm, but also measures QoS experienced by the traffic such as CLR, queue length distributions, inter-cell loss times and other comparative measurements. The measurement controller tasks are to match the asynchronous measurement requests (from the CAC algorithm) to the synchronous methods in which measurements must be taken and to interface between the simple inter-machine protocol used by the switch with a standard RPC interface that is used between the measurement server and the CAC system.

4.3 Traffic Generator

The generator illustrated in Figure 3 can be used to transmit a multiplexed stream of cells from theoretical generators creating cells in real-time and from generators reading from pre-generated ATM cell lists. Traces of real ATM traffic, such as video streams or IP on ATM, are a useful set of test traffic to be carried by connections. In addition to being able to deliver cell streams consisting of all required traffic types, this physical generator can be dynamically controlled, able to stop and start individual traffic sources using a purpose built RPC mechanism.

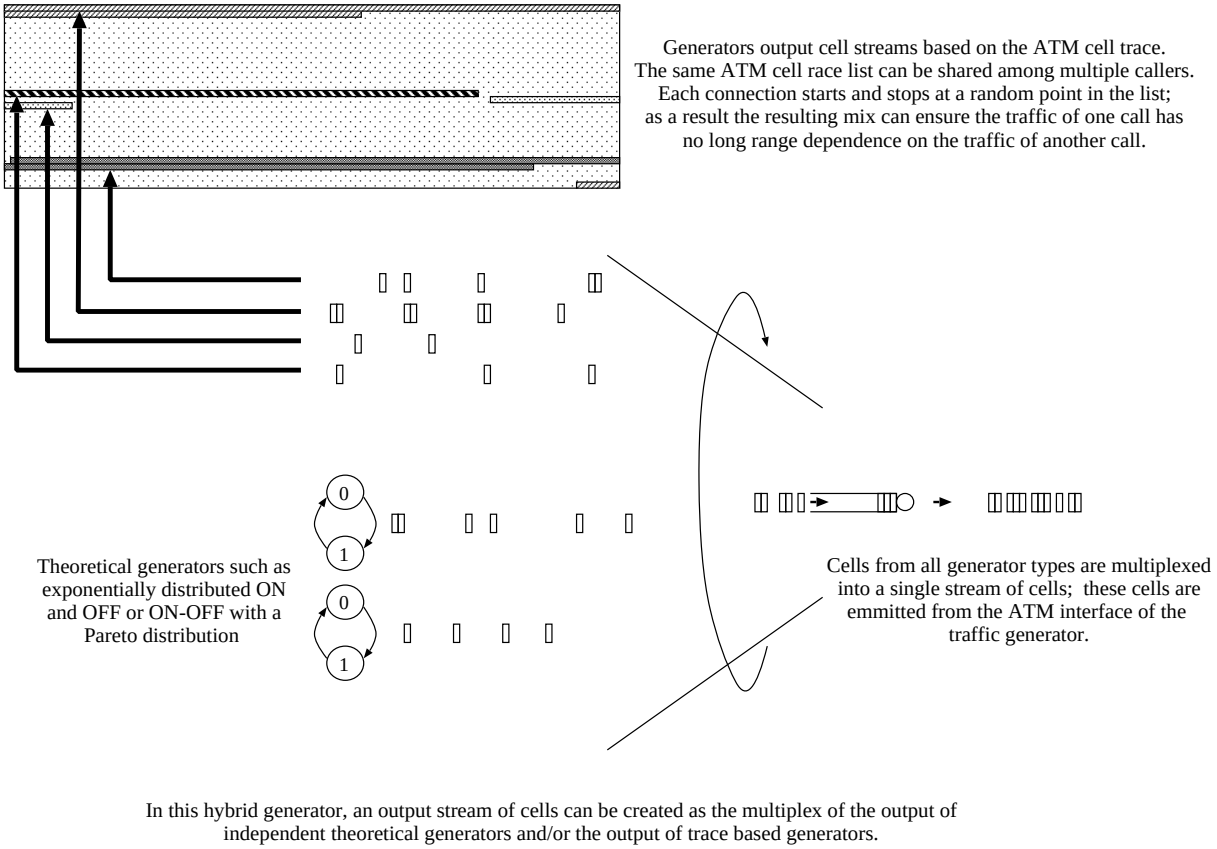


Figure 3: Hybrid physical generator able to generate cells from theoretical traffic sources operating in real-time and from ATM cell traces loaded into memory.

The whole physical generator runs on a PC that is running the Nemesis operating system [7] which allows the construction of complex, time-critical tasks (the real-time creation of traffic traces) and the timely operations of device-drivers. In addition to allowing a purpose built device driver for the ATM interface, using Nemesis means that guarantees of timeliness can be made to the RPC based control mechanism to ensure time-bounded actions and replies.

With an individual traffic generator representing each connection in progress, the physical generator is capable of saturating the ATM transmission link should this be required and the ability to combine a virtually unlimited number of traffic types of both the theoretical generator or those based upon ATM cell traces gives us unrivalled flexibility in experimentation.

4.4 Traffic Generator controller

The traffic generator controller, a process running under Unix, will instruct the traffic generator to start and stop individual traffic sources representing each connection as these connections are set-up and pulled-down. The traffic generator controller, like the measurement controller, gives an interface between the standard RPC based interface that is used by the CAC system and the purpose-built inter-machine protocol used by the traffic controller. In later revisions of the physical traffic generator it is expected to be able to dispense with the traffic generator controller altogether.

4.5 Connection Generation

The connection generator, a process running under Unix, will initiate new connection attempts into the CAC test-rig. Connections entering the system can be described by the arrival rate of new

connections, the connection holding time and the traffic each new connection will carry. The parameters of connection inter-arrival rate and connection holding time can have specified values or have a range of values based on a distribution – for example the period over which a connection will be in progress could have an exponential distribution with a given mean. The value for traffic type would typically be specified for a set of connections. The three parameters can also be specified in a file; in this way logs of connection events can be processed to produce a set of connection arrival-rates, durations and traffic types. Finally, the connection generator is able to generate new connections of more than one traffic type simultaneously.

4.6 CAC and admission policy

The CAC component forms the core of the CAC test-rig. The CAC component has the capability to change the CAC admission policy as required. Only one policy is in place during any experiment however consecutive experiments can operate with only the admission policy itself or the control parameters of any particular policy being changed.

During the generation of new connections, the traffic type and the parameters that describe traffic that the connection will carry are declared to the CAC algorithm. The parameters of each new connection can be specified in any of the TM 4.0 parameter formats [8]. Each new connection presents its parameters to the CAC system and requests a connection to be set up across the switch.

Each admission policy obtains the required measurements from the measurement controller as part of that particular CAC's decision process. Each admission policy can obtain the measurements of the type and format it requires, in the case of a simple threshold CAC algorithm measurements are of instantaneous line utilisation while for Peak Rate allocation no measurements are required for the CAC algorithm.

4.7 Cell time-frame scaling

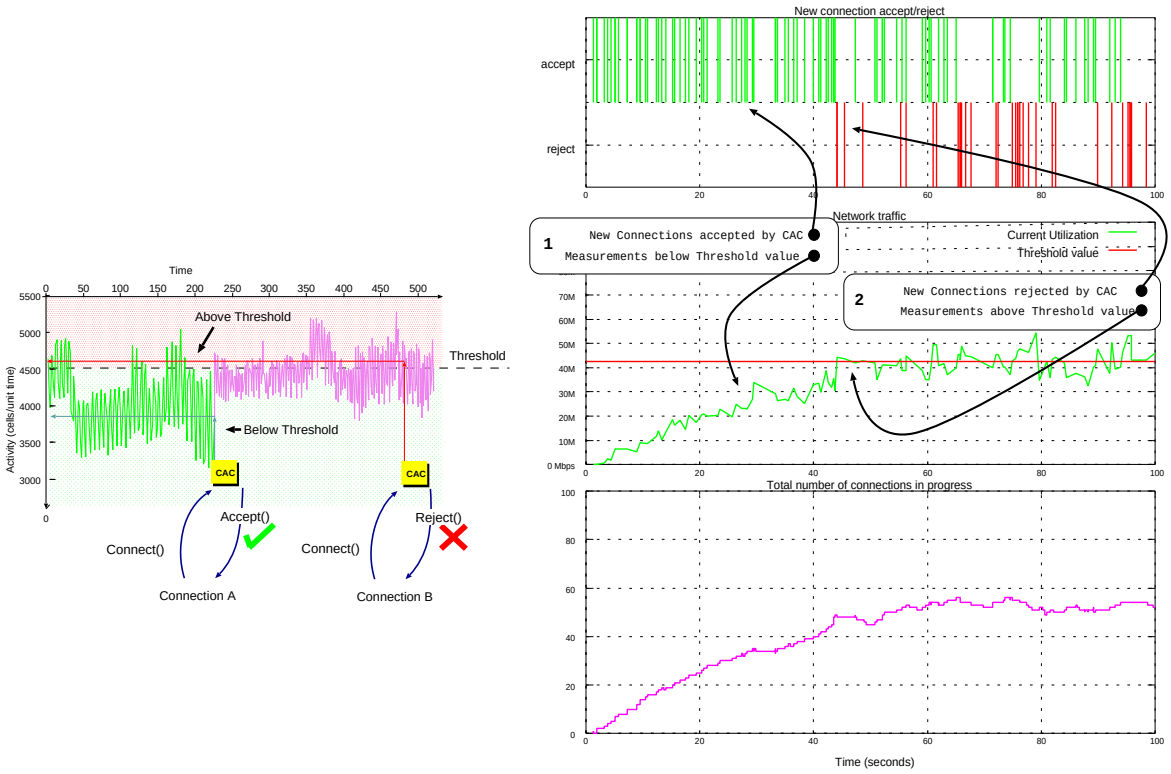
Section 4.1 discusses, the ATM switch used in the CAC test-rig. In that section it is noted that the rates of traffic sources are scaled by a factor D, this factor is a multiplier on the time between cells. As a result, the passage of time on the network, and hence the passage of time in the experiment as a whole, has been slowed down by the factor D. Throughout this document all times stated for kit performance, connection setup, connection holdings periods, measurement period, and any other time frame in the experiment are given in unscaled time; that is time that has not been multiplied by D which makes our experimental results directly comparable with measurements made on other systems.

4.8 Test-rig operation

The CAC system works as follows: a connection generator is responsible for 'generating' according to some distribution or from a previously collected trace of measured arrivals. New connections may be of multiple types, and each connection may, according to a random distribution, determine its connection type and any set of parameters such as: Sustained Cell Rate (SCR) or Peak Cell Rate (PCR), which it is required to present. New connections, once generated, present to the CAC decision system their parameters following being generated. The currently loaded CAC algorithm, using measurements from the switch, makes a decision as to whether or not to admit the connection. Only one CAC algorithm operates in any one experiment.

If a connection is admitted, the CAC algorithm will reply to the connection generator accepting the connection. The connection generator then instructs the traffic generator controller to 'set-up' a new traffic source with the appropriate parameters for a connection of this type. The traffic carried by this connection might be on-off, some other analytical model, or trace driven. The traffic generator controller then starts the new connection by instructing the physical generator to create and start a traffic source with the correct parameters. The cells of this new connection will then enter the multiplex of streams of cells that the physical generator is transmitting into the switch. When each new connection is created, apart from its traffic type and arrival time, a new connection will have associated with it a lifetime, or connection holding time. This connection lifetime, like the arrival time, can be drawn from a theoretical distribution or a trace driven set of values. Once the connection holding time is reached the connections' traffic source is stopped and that connection is 'cleared down'.

It is important to emphasise that in this set-up there is no real ATM signalling, all VP/VC pathways that will be required have been setup as permanent circuits prior to an experiment. The processes running off-switch assume the full load of the 'signalling' and therefore it is possible to emulate the arrival of connections at rates far higher than could be sustained by any real ATM signalling implementation.



(a) Threshold CAC mechanism in action

(b) The first 100 seconds of operation

Figure 4: Simple Threshold CAC Algorithm.

The logging system in the CAC test-kit gives sufficient information that real-time graphs can be produced displaying information: the current line utilisation, the current connections in progress and information about new connection acceptance or rejection. Figure 4(b) shows the plot of 100 seconds of time from the start of an experiment. The x axis in all cases is time, shown in seconds since the start of the experiment. The top graph in each set shows the connection arrival process. For each connection which arrives a vertical bar is drawn. In the event that an arriving connection was accepted by the CAC algorithm, a vertical bar is drawn extending upwards. If the connection is rejected, then the vertical bar extends downwards. In the leftmost part of Figure 4(b), where the time is less than 45 seconds, no new connections have been rejected because measurements of the instantaneous line utilisation are less than or equal to the thresholding value. Whereas, after 45 seconds, sufficient traffic is now in the system for the instantaneous line utilisation to be above the thresholding value and as a result for connections to be rejected.

The second graph from the top is a display of the traffic dynamics in the switch, measured in real time. The thresholding value in use by the CAC algorithm is shown along-with the current measure of instantaneous line utilisation. The third graph of Figure 4(b) shows the number of connections in progress in the system, over time. This rapidly climbs, as new connections enter the system at a greater rate than they clear down. This is because in the empty system (at time zero) no connections are rejected. Once

rejections occur, the number of connections in progress stabilises but displays the expected variation due to statistical fluctuations. In this experiment connections carried a theoretical traffic type with a 10Mbps PCR and a 1Mbps SCR. A thresholding value of 42.6 Mbps was used and the Mean Connection Admission Rate (MCAR) was 10 connections per second with an Mean Connection Hold Time (MCHT) of 10 seconds. These output graphs are regularly used to check that the environment is operating correctly and to allow simple comparisons of consecutive experiments; in addition to generic parameters of utilisation and connection behaviour, algorithm specific information, such as threshold values, can also be output.

5 Test-rig evaluation

During the construction of the CAC test-rig, effort was made to ensure the goals of performance and repeatability. The test-rig was required to achieve high rates of new-connection setup: the rate at which new connections could be attempted and started when the connection was admitted into the system. Section 5.2 covers the connection setup performance of the test-rig. The test-rig must also allow high repeatability of experiments; ideally, consecutive experiments with no parameter changes should yield identical results. Section 5.3 reports results conducted to assess the repeatability of experiments on the CAC test-rig.

Aside from repeatability and high speed performance the test-rig needs to be flexible supporting a wide variety of connection methods, traffic sources, CAC algorithms and measurement methods. While these are not listed here, the modular design of the test-rig has allowed extensions for each of these aspects of its operation. However, firstly there are issues of stability and accuracy, particularly when dealing with a system that may possess start-up transients such as the test-rig. Section 5.1 discusses the issues of experiment run times, experiment stability and start-up transient detection.

5.1 Run length and initial stability

For experiments made using our CAC evaluation rig, there is an initial, "start-up," transient before the system stabilises and returns consistent results. Figure 4(b) shows clearly how an experiment has an initial transition period before its operation has settled, in this case there is a slow ramp-up to a value of connections in progress that is then held relatively constant. Such start up transients are quite common to steady-state simulation work and as a result we are able to draw from work in that field. Pawlikowski [9] gives a number of methods for determining at what point an experiment has become stable. A combination of several of these algorithms were used: waiting for the longest cycle in the system to have been executed 3 or 4 times and calculations based upon stabilisation of the variance of several key experiment measures (commonly the number of connections in progress, the global cell loss ratio and the mean line utilisation). At this time we determine the point of stability in off-line processing; once detected, data collected up until this point is discarded. Work is in progress to incorporate this off-line processing into the running system; in this way a running system can determine how much longer it needs to run for before it has passed the initial start-up transient and can collect data from the stable experiment. The contribution of the transient period to the the length of an experiment is small, the main factor in the length of experiments is the need to collect a representative samples of loss-events for the CLR.

For most experiments conducted, we have algorithms with a target CLR of 1×10^{-3}. In order to ensure a large sample of "loss-events" we run the experiments for a minimum of 1×10^8 cells: 1×10^5 loss events. The selection of this figure has been based upon our experience with CLR measurements for previous experiments [10] and practical limits on the run time for each experiment. Currently, for a typical experiment transferring 1×10^8 cells, the run period of this experiment is approximately 2 hours. The length of the experiment is due to combination of the number of cells required and the scaling factor D (discussed in Section 4.1) in use. To increase the number of loss-events recorded, and still use the same scaling factor D, we would need to run the experiment for longer periods of time – a magnitude increase in the number of loss events, would require experiments running for 20 hours. It was concluded that experiments of 1×10^8 cells (typically a 2 hour duration) was a satisfactory trade-off between the number of loss events and the running time of a typical experiment.

5.2 Performance

When a connection is entered into the system an assumption is that there is a negligible amount of time between when the new connection has been generated and, assuming acceptance, when cells transmitted by the corresponding traffic generator will start entering the data stream. This assumption is not valid in anything other than a theoretical test structure. However, it is important to quantify and where possible overcome such a delay between a new connection entering the system and cells being produced by the system so as to minimise the impact of experimental effects being introduced into the evaluation experiments. In this way theoretical results and experimental results can be compared more closely.

The performance goal in the construction of the CAC test-rig was to reduce the new connection generation, new connection test and new connection start-up delays to a minimum; the smaller the delay achieved the closer to the values used in a naïve simulation. When compared to the real-world implementation, aiming for a minimal delay could be seen as unnecessary – several authors [11, 12] note that in commercial ATM switch systems the connection setup process for a new connection can take 20–200ms. Such a quoted value for the delay does not include the additional time required for the end-system to become active. In our work to reduce the delay in the CAC test-rig, this delay period runs from the generation of the new connection request through to the moment cells are emitted from the ATM interface of the traffic generator.

The delays in the pathway between the generation of a new connection request and the emission of cells into the ATM switch take several forms: firstly there is time taken in the execution of code on the various machines that the CAC test-rig runs, secondly there are delays related to the communications between components of the CAC test-rig and finally there are delays in the physical generator that will cause a delay between the starting of traffic generators and the emission of cells from the ATM interface and into the ATM switch.

While we have not given a breakdown of the typical delays, places in which time was spent included: the execution of code on the various systems that make up the CAC test-rig, a compound of the many instances of inter-machine communications and the finally the network interface transmission system of the traffic generator.

Following improvement and optimisation, experiments on the final test-rig established that delays between a new connection arrival and the start of transmission of its cells from the corresponding generator has an upper bound of 8.38 milliseconds. The statistics and distribution of this delay are shown in Table 1 and Figure 5 respectively.

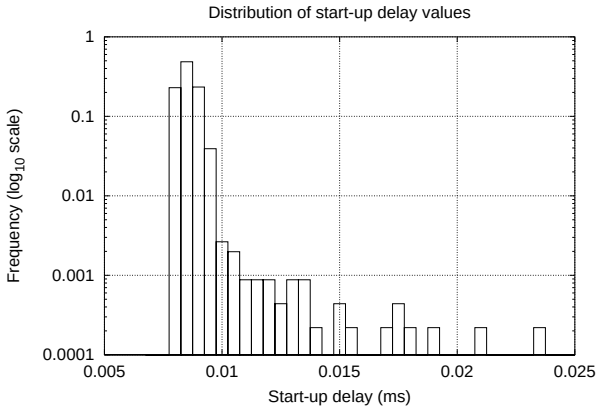


Figure 5: Distribution of start-up delay values.

The immediate effect is on consecutive connection attempts, connections may be delayed if attempted within less than 8.38 milliseconds of each other. This implies that, for the worst case, connections cannot be attempted and started at a rate any faster than ~ 119.3 connections per second. As the experiment

Mean	Var	Std. Dev.	95 % Conf. Int.
8.8048E-03	1.6678E-6	1.2914E-3	3.7556E-05

Table 1: Statistical properties of a set of start-up delay values.

has an exponentially distributed connection arrival rate with a mean of 10 connections per second, we can predict, with 95% confidence, that 0.27% of connections may be affected. However, the actual impact is less than this because this full delay impact is incurred only on new connections that are accepted into the system.

For new connections that are rejected from the system, the delay is substantially smaller. When compared with the earlier stated values for commercial connection setup of 20–200ms [11, 12] – a connection setup period of less than 9ms is quite acceptable. In real terms this means that, in the worst – hypothetical case, for an experiment of 6000 connections, less than 18 of those connections will be affected by system delays where the delaying of one connection will delay the next connection entering the system.

5.3 Repeatability

In order to reliably compare and contrast different CAC algorithms under a range of connection loads repeated experiments need to give high repeatability of results. Running consecutive experiments with no changes in parameters should reveal as near to identical results as is possible. This section reports on results appraising the repeatability of the CAC test-rig. Firstly, several sets of repeated experiments are contrasted with experiments run with a variety of random seeds. These random seeds form the inputs to the connection arrival and connection holding time distributions; both distributions are random with a negative exponential distribution and each is independent of the other. These seed values are also used in the creation of traffic generators on the physical generator. Each traffic generator uses these random seeds to seed the random number generators that will give distributions of cell burst length and inter burst time.

The objective was to establish that variations in results for experiments repeated with identical parameters and variations in results for experiments repeated with different random number seeds give a similar degree of variation in the results. The second round of repeatability experiments were run with identical sets of parameters. 100 experiment runs were performed and a statistical evaluation of the accuracy of the repeated results is given.

The first repeatability tests were made using with connections carrying a theoretical traffic source. The MCAR of these connections was 10 connections per second and the MCHT was 10 seconds per connection. The CAC algorithm was the Simple Threshold algorithm. Using a threshold of 42.6 Mbps, new connection attempts were rejected if the link utilisation was above 42.6 Mbps. The results obtained for mean line utilisation (MLU) and for the CLR from a buffer of 100 cells in length were compared for each experiment.

The statistical properties of the two sets of experiments, ten experiments using the same seed and ten experiments with different seeds for the random number generator, are documented in Table 2. The variation between experiment runs occurring in experiments without any change in parameters is slightly smaller but of the same order of magnitude as the variation between experiment runs where the seed of the random number generators has been altered for each experiment run. This implies that the variation in an individual experiment caused by the differences between consecutive runs is almost as large as the 'random' variation of changing the seeding of the random number generators. In addition, these experiments show that, for mean line utilisation at least, the selection of one particular set of random seeds for the generators does not artificially constrain the range results obtainable. The next set of results to compare were the CLR values obtained.

Statistical properties of the CLR experiments (for a buffer length of 100 cells) are documented in Table 2. Once again, similar to the results for mean line utilisation, the results indicate that variation of results for CLR are slightly smaller for experiments repeated with the same set of random seed values

	Mean	Var	Std. Dev.	95 % Conf. Int.
Mean line utilisation results				
varied seed	4.278841E-01	1.935880E-06	1.391359E-03	1.938881E-03
same seeds	4.268741E-01	8.443734E-07	9.188979E-04	1.280499E-03
CLR results				
varied seed	5.347374E-04	4.618412E-09	6.795890E-05	9.470178E-05
same seed	5.712818E-04	5.356902E-10	2.314498E-05	3.225289E-05

Table 2: Statistics for the values of mean line utilisation and CLR of a 100 cell buffer for experiments repeated with and without variations in the set of seed values.

Mean	Var	Std. Dev.	95 % Conf. Int.
5.477709E-01	3.384366E-05	5.817530E-03	1.160403E-03

Table 3: Statistical information on the 100 mean line utilisation results shown in Figure 6(a).

than those repeated with different random seed values although having a similar magnitude. Additionally, like the mean line utilisation results, this implies that using one particular set of random numbers will not artificially constrain the variation in results and that the variation is almost as large in consecutive runs for the case where the set of random numbers is kept the same as it is for the case where the set of random numbers is varied.

Once the variation between experiments runs without varying any parameters was shown to cause as much variation in the results from experiments as those experiments where the set of random number seeds was changed, the exact variation needed to be established with a larger set of repeat experiments of identical parameters.

In order to establish a more representative and comprehensive sample, 100 experiments, involving a mixture of different types of traffic streams, were run. In all possible ways input parameters were held as constants throughout successive experiments on the test-rig.

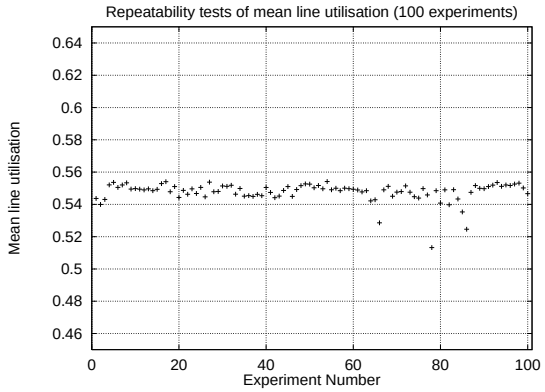
The experiments themselves involved a mixture of connections of two different traffic types. Two different traffic sources carrying video data were used, one video source, with a PCR of 10Mbps and an SCR of 1Mbps was carried on connections that had an MCAR of 5 connections per second and an MCHT of 10 seconds per connection; the other traffic type, with a PCR of 5Mbps and an SCR of 2Mbps, was being carried on connections that had an MCAR of 5 connections per second and an MCHT of 5 seconds per connection. The threshold value used in the CAC algorithm was 61Mbps.

During evaluation of CAC algorithms, the overall CLR and mean line utilisation of an experiment are significant comparison criteria, as a result it was these results that were commonly compared between experiment runs. Figure 6(a) shows the mean line utilisation values for the batch of 100 identical experiments. A statistical summary of this collection of results is in Table 3 and the distribution of the results is shown in Figure 6(b). In comparison, Figure 7(a) shows the CLR values for the batch of 100 identical experiments. The statistics of this collection of results is in Table 4 and the distribution of the results is shown in Figure 7(b).

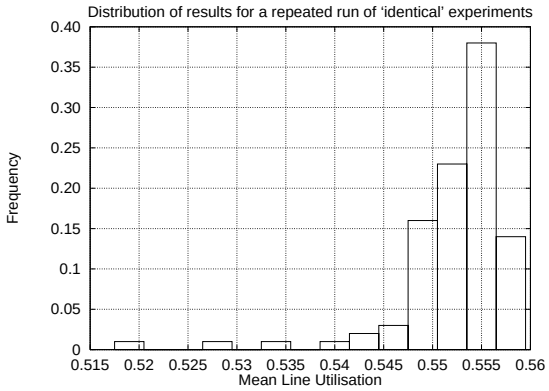
It is clear that even for experiments with a narrow distribution of mean line utilisation, the values for cell loss ratio have a much wider distribution. This will mean that with a 95% confidence the link utilisation value will have an error of $\pm 0.21\%$. With a 95% confidence, the CLR results will give

Mean	Var	Std. Dev.	95 % Conf. Int.
1.238580E-03	9.047702E-08	3.007940E-04	5.969022E-05

Table 4: Statistical information on the 100 CLR results shown in Figure 7(a).

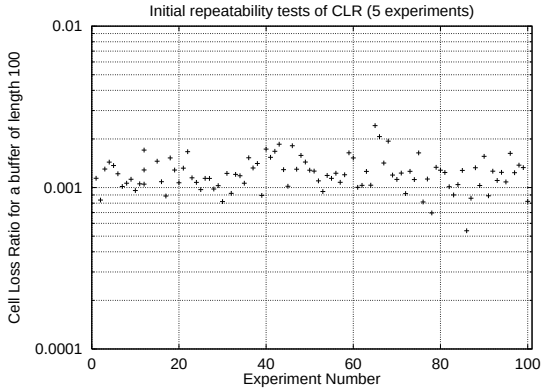


(a) MLU values

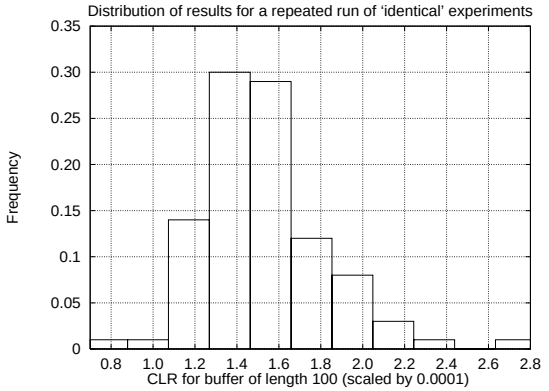


(b) Distribution of MLU values

Figure 6: Repeatability test results.



(a) CLR values



(b) Distribution of CLR values

Figure 7: Repeatability test results.

experimental results with an error of $\pm 4.6\%$.

We repeated a number of identical experiments to appraise the accuracy of the test-rig. It is difficult to know if the the results we obtained are accurate enough as test-rigs are not common and those that do exist are not easily comparable. Additionally, there are few avenues for improving these results without substantial changes to the experiment parameters or the architecture in use. We are confident that our CAC test-rig offers a unique insight into MBAC behaviour that cannot easily be explored with theoretical abstraction or simulation; previous work [10] found significant divergence common among theoretical models. Our test-rig displays an excellent faithfulness to real implementation and it is precisely this fidelity that makes comparison with simulations so difficult.

6 Conclusions

Interest in CAC algorithms stems from the need for a network user and a network provider to establish an agreement on the QoS for a connection the user wishes to have admitted into the network. Interest in an evaluation environment for CAC algorithms has grown from the failure of simulators to encompass all aspects of evaluation. In addition to an introductory discussion on how simulators are unable to

emulate all aspects of a real switch implementation, nor the traffic or networks conditions that the CAC may be placed under, we have outlined in Section 2 how existing test environments have either not been suited for use in the evaluation of CAC algorithms, or that their limited availability has restricted their usefulness thereby offering a place for a suitable CAC evaluation environment. Section 4 then outlines the architecture of our test environment, its components and its operation. Finally, Section 5 reports an evaluation of the test-rig, particularly the connection setup performance and repeatability performance.

This paper has presented an evaluation environment capable of evaluating a CAC algorithm including comparing that algorithm with several other CAC algorithms. In this way we have shown that test-rig evaluations have an important role to play in the assessment of CAC algorithms, not necessarily as a replacement for simulators but as an equally important method of approach. Whereas the use of simulators offers a widely available technique for the testing of CACs, the environment we have described here allows comparison of CAC implementations constrained in the same manner that real rather than simulated implementations are constrained.

Thanks

Thanks to Tim Granger, Neil Stratford and Ian Leslie for their valuable input over the course of design and construction of the test-rig.

A debt is owed to developers of the Nemesis operating system, and Austin Donnelly in particular, for making possible the construction of robust and highly adaptable traffic generators, my thanks to them.

Our thanks to Neil Stratford, Richard Mortier and Ralphe Neill for their helpful comments on drafts of this paper and to the anonymous reviewers for their valuable feedback.

References

[1] Sandeep Bajaj, Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, Padma Haldar, Mark Handley, Ahmed Helmy, John Heidemann, Polly Huang, Satish Kumar, Steven McCanne, Reza Rejaie, Puneet Sharma, Kannan Varadhan, Ya Xu, Haobo Yu, and Daniel Zappala. Improving simulation for network research. Technical Report 99-702, University of Southern California, March 1999.

[2] T.M. Chen, S.S. Liu, M.J. Procanik, D.C. Wang, and D.D. Casey. INQUIRE: A software approach to monitoring QoS in ATM networks. *IEEE Network*, pages 32–37, March/April 1998.

[3] A.A. Lazar, G. Pacifici, and J.S. White. Real-time traffic measurements on MAGNET II. *IEEE Journal on Selected Areas in Communications*, April 1990.

[4] N.G. Anerousis, A.A. Lazar, and D.E. Pendarakis. Taming Xunet III. Technical Report 481-97-15, Center for Telecommunications Research, Columbia University, 1995. ftp://ftp.ctr.columbia.edu/CTR-Research/comet/public/papers/95/ANE95b.ps.gz.

[5] S. Mazumdar and A.A. Lazar. Objective-driven monitoring for broadband networks. *IEEE Transactions on Data and Knowledge Engineering*, 8(3):391–402, June 1996.

[6] A.A. Lazar and M. Nandikesan. A Real-Time Traffic Generation and QOS Monitoring System. Technical Report 481-97-15, Center for Telecommunications Research, Columbia University, August 1997. ftp://ftp.ctr.columbia.edu/CTR-Research/comet/public/papers/97/tg.ps.gz.

[7] Ian Leslie, Derek McAuley, Richard Black, Timothy Roscoe, Paul Barham, David Evers, Robin Fairbairns, and Eoin Hyden. The design and implementation of an operating system to support distributed multimedia applications. *IEEE Journal on Selected Areas in Communication*, 1996.

[8] ATM Forum. *Traffic Management Specification, Version 4*. ATM Forum/95-0013R8, October 1995.

[9] Krzysztof Pawlikowski. Steady-state simulation of queueing processes: A survey of problems and solutions. *ACM Computer Surveys*, 22(2):123–170, June 1990.

[10] Andrew Moore and Simon Crosby. Experimental results from a practical implementation of a Measurement Based CAC algorithm. BTL Final Report – Contract ML704589, May 1998.

[11] Abdella Battou. Connections Establishment Latency: Measured Results. *ATM-Forum T1A1.3/96-071*, October 1996.

[12] Douglas Niehaus, Abdella Battou, Andrew McFarland, Basil Decina, Henry Dardy, Vinai Sirkay, and Bill Edwards. Performance Benchmarking of ATM Networks. *IEEE Communications*, 35(8):134–143, August 1997.