Measurement Based Resource Allocation for Multimedia Applications*

Paul Barham, Simon Crosby, Tim Granger, Neil Stratford,[a]
Meriel Huggard, Fergal Toomey[b]

[a]University of Cambridge Computer Laboratory,
New Museums Site, Pembroke Street, Cambridge CB2 3QG, UK
[b]Dublin Institute for Advanced Studies,
10 Burlington Road, Dublin 4, Ireland.

Abstract

Modern networks are now capable of guaranteeing a consistent Quality of Service (QoS) to multimedia traffic streams. A number of major operating system vendors are also working hard to extend these guarantees into the end-system. In both cases, however, there remains the problem of determining a service rate sufficient to ensure the desired Quality of Service. Source modelling is not a sustainable approach in the network case and it is even less feasible to model the demands of multimedia applications. The ESPRIT Measure project is successfully using on-line measurement and estimation to perform resource allocation for bursty traffic in ATM networks. In this paper we consider the applicability of the same theory to resource allocation in a multimedia operating system which offers QoS guarantees to its applications.

1 INTRODUCTION

Multimedia applications are characterised by a need for guaranteed access to system resources in order to meet their Quality of Service (QoS) constraints. The spectrum of QoS requirements is an extremely broad one ranging from the loose guarantees required by text-processing applications to the comparatively strict guarantees required for real-time audio processing.

Multimedia traffic in modern high speed networks can already be provided with QoS guarantees bounding delay, loss and jitter. Unfortunately, these guarantees are of little use if they cannot be extended up through the operating system to the destination application. A number of major operating system vendors are therefore working hard to provide such guarantees, and there are now several research operating systems incorporating soft-real time scheduling techniques[1] – typically it is possible to provide a sustained rate of CPU allocation to each process. The Nemesis operating system[2] is unusual in that guaranteed resource levels are provided for *all* operating system resources including memory and I/O bandwidth.

As is the case with high speed networks, however, there remains the problem of determining the resource allocation(s) necessary to ensure a desired quality of service.

Resource allocation can be decomposed into two sub-problems: admission control and scheduling. In the first, the system must decide, dynamically, whether or not a new request for resources can be met given the current activity and set of existing guarantees in the system. In the second, the system must apportion its resources between the currently running applications in a way which ensures that each meets its QoS constraints. This split corresponds to a separation of mechanism and policy.

Resource allocation in multi-service networks has, in the past, been performed by modelling the characteristics of traffic sources and their multiplexing behaviour. Since new traffic classes are

*This work was funded by the European Commission ESPRIT Measure Project LTR 20113
Further author information: (authors listed alphabetically)
CUCL: Telephone: +44 1223 334416; Fax: +44 1223 334678; Email: Neil.Stratford@cl.cam.ac.uk
DIAS: Telehone: +353 1 6680748; Fax: +353 1 6680561; Email: huggard@stp.dias.ie

appearing every day, it is clearly desirable to be able to provide QoS guarantees without having to model the multiplexing behaviour of every new traffic source. This requirement has led to resource allocation schemes which bypass the requirement to model traffic.

The ESPRIT Measure project[3] is successfully using on-line measurement and estimation to perform resource allocation for bursty traffic in ATM networks. The approach is based upon the theory of Large Deviations,[4] which has been recognised as a valuable tool for reasoning about rare events in stochastic systems. Previous work by some of the authors has shown that the large deviations rate function, or entropy, of bursty traffic can be estimated from measurements of its activity. This entropy can be used to determine the "effective bandwidth" of the traffic and hence the determine the necessary resource allocation.

In an analogous manner, much of the previous work on real-time scheduling in operating systems has attempted to perform admission control by modelling the resource requirements for each application based on detailed knowledge of both its computational complexity and the data which it processes. The resource requirements are then used to compute a feasible schedule (if possible) for a real-time scheduler. We believe that it is even less desirable, in a software environment, to insist that each application provides an accurate model of its resource requirements, especially given the complexity of modern processors.

We argue that multimedia applications require resource guarantees which are of the same nature as those assigned to the media by other components of the system, typically these include bounds on delay, jitter and loss. An application may receive data from a network which offers QoS constraints which are at best statistical in nature; it is natural to carry the notion of statistical guarantees on performance up through the end-system operating system to the applications themselves.

In the current framework, we use measurements of application activity to obtain estimates of the *entropy* of the demand process for the system resources, made by each application. From the entropy we are able to determine precisely and directly the *rate* at which each application must be assigned use of the resource (e.g. the CPU) in order for its constraints on timeliness, delay, jitter and loss to be met.

The remainder of this paper is structured as follows: Section 2 presents an overview of Nemesis and contrasts our approach to that of other operating systems research. Section 3 then introduces the theoretical framework upon which our measurement-based estimation approach is based. To test the theoretical framework a series of experiments using Nemesis with both well characterised and more realistic applications was carried out. The results are presented in Sections 5 and 6. Finally, Section 7 describes our current work and highlights several remaining research issues.

2 THE NEMESIS OPERATING SYSTEM

Nemesis is the operating system developed at the University of Cambridge computer Laboratory as part of the ESPRIT Pegasus project. It was designed with the aim of supporting distributed multimedia applications which *process* time sensitive data, as opposed to merely moving it around. The operating system therefore provides Quality of Service guarantees for *all* system resources.

For these QoS guarantees to be of any use to an application, it is important that the performance of each application is not dependent on the QoS parameters of other processes which it must invoke to perform work on its behalf. There are two distinct approaches to this problem. One approach is to contrive mechanisms mechanisms which transfer resources between processes in the system every time they interact, or in the extreme case even cause threads to migrate between processes. The second approach is to construct the operating system so that the problem does not arise.

In Nemesis this is achieved by using a *vertically integrated* structure (Figure 1) in which a minimal kernel is used simply to manage the hardware and to dispatch the CPU and other system resources according to a *rate based* schedule. The use of a *single address space* (but multiple protection domains) facilitates rapid context switching, data sharing and the implementation of

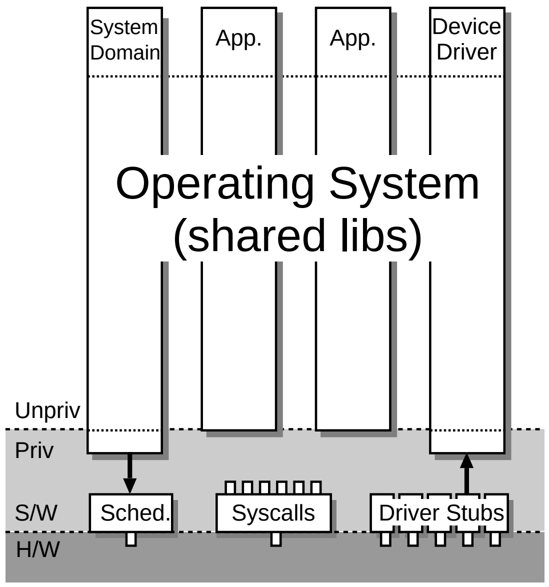

Figure 1. The vertically integrated structure of Nemesis

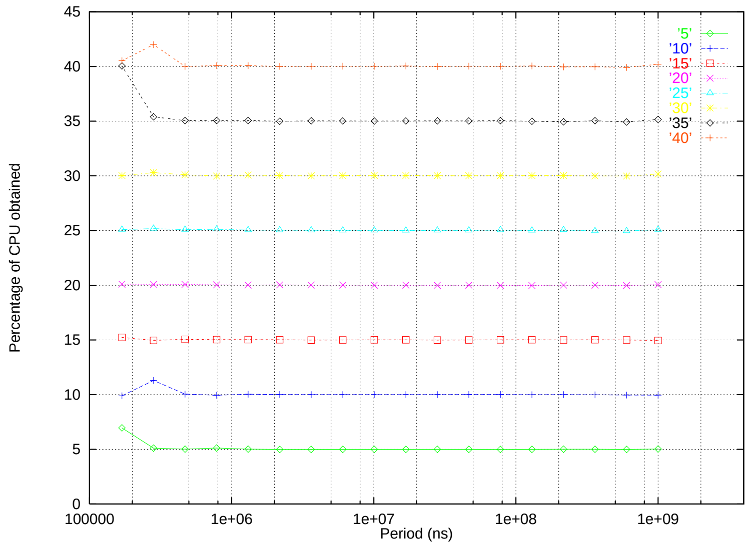

Figure 2. Guaranteed allocation of CPU rates by the Nemesis scheduler.

most traditional operating system functions as shared, stateless libraries which are executed by the applications themselves. All resources consumed by an application can thus be accounted to it, enabling the system to police resource usage and deliver meaningful QoS guarantees.

2.1 Scheduling

A Nemesis application specifies its CPU requirements as a tuple (p, s, x, l) which identifies a period p (in nanoseconds) within which the application must receive s ns of CPU. The parameter x indicates whether the application wishes to be allocated a fair share of any additional CPU remaining after all applications have been given their guaranteed resources, and l is a *latency hint* used to provide the scheduler with an idea as to how soon the domain should be rescheduled after unblocking.

The simplest kernel scheduler employs a variant of the EDF algorithm[5] where the deadlines are

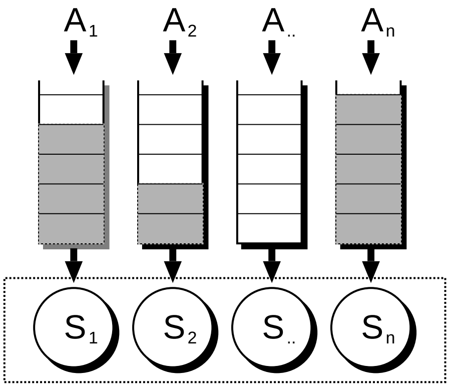

Figure 3. Nemesis domains as independent single-server queues.

derived from the QoS parameters of the domain and are purely internal to the scheduler. Figure 2 shows how accurately the scheduler assigns proportions s of the CPU resource to a domain for different values of the period p. Each line represents a different guaranteed proportion of the total resource, as indicated in the key. The scheduler can simultaneously support any reasonable number of such guarantees provided that $\sum_i \frac{s_i}{p_i} \leqslant 1$.

This leaves the problem of how the parameters p and s should be determined. Typically p results from an application imposed constraint: for example, a networked motion JPEG video decoder requires sufficient CPU in order decompress and display 25 frames per second without overflowing its buffer allocation – here p is related to the amount of buffering, latency and jitter which may be tolerated.

The determination of s is more problematic. Different hardware platforms will execute the same code in a different amount of time, and the problem is compounded by the fact that multimedia data streams are naturally bursty. Compressed video and audio, and even elastic data transfers exhibit enormous fluctuations in bit-rate. If the data which the applications consume is bursty, it seems natural to ask how this affects the resource requirements of the applications which process the data.

2.2 Queue-Server Paradigm

Unlike most contemporary operating systems, the scheduling regime used by Nemesis allows each *domain*† to be considered as though it executed on an independent CPU running at a rate proportional to its QoS guarantee. This, in turn, allows the application of queueing theory to each domain individually.

At a simplistic level, we can consider each application to behave as an independent statistical process generating arrivals of work into a notional queue which is then serviced at a constant rate S_i determined by its QoS guarantee. (Figure 3). Even using such a simple model, it is possible to make useful statements about the behaviour of applications, for example the theory of large deviations can be used to predict the probability of this queue exceeding a given length for any given service rate. For a number of applications, these fictitious queue lengths can be related fairly trivially to latency and buffering requirements.

In reality, however, things are not so simple. For example, in the case of an ATM network, this queue of work equates simply to the queue of cells in each switch port, arrivals to the queue can be measured trivially, and the service rate is a known constant. When we consider applications being scheduled on a CPU, we have no way of observing arrivals of work into the "queue" since,

†The Nemesis equivalent of a process.

in reality, there may not be a queue at all. We can, however, measure the service process and at very low loads these can be equated.

To further complicate matters, the amount of real "work" an application can perform in a fixed time period is greatly influenced by a number of uncontrollable effects such as the processor cache and the context-switch overheads. These must be factored into the resource allocation.

3 THEORETICAL FRAMEWORK

Our work is motivated by results from large deviation theory, which provide a framework within which the resource requirements of multimedia applications may be determined non-parametrically. For a complete discussion of the theory behind our approach the reader is directed to our earlier work.[4,6] Here we give only a brief summary.

Consider a multimedia application which is sensitive to the passage of time. We assume that there is a time constraint d associated with the application which indicates the maximum tolerable delay which it can sustain before the information or processing effort of the task at hand becomes worthless. For example, the decoding of a video frame for real time display may be considered a worthless task if it is delayed long beyond the due date of the frame. The frame would then be discarded by the application: we let c denote the maximum tolerable error rate arising in this way from violation of the delay constraint d.

The system may be thought of as a single-server queue, to which work is added as it becomes due, and from which work is removed at some constant rate r by the system resouce (e.g. CPU) for which the application is contending. Suppose that the work arriving at the queue is modelled as a stochastic process; then, for a wide range of possible stochastic models, the tail of the steady-state queue length distribution is found to be exponential:

$$\mathbb{P}(Q \geq q) \sim e^{-\delta(r)q}. \tag{1}$$

A precise statement of conditions under which this relation holds is given in our earlier work.[6] The decay parameter δ may be determined from the *scaled cumulant generating function* (CGF), or *entropy*, of the arriving work. Letting A_t be a random variable representing the quantity of work arriving at the queue in an interval of length t, the scaled CGF $\lambda : \mathbb{R} \to \mathbb{R}$ is defined by

$$\lambda(\theta) := \lim_{t\to\infty} \frac{1}{t} \log \mathbb{E}e^{\theta A_t}.$$

$\delta(r)$ is related to λ by the equation

$$\delta(r) = \sup\{\theta \geq 0 : \lambda(\theta) \leq \theta r\}. \tag{2}$$

Thus $\delta(r)$ may be found from measurements of the empirical CGF of the arriving work.

A simple method of estimating the CGF is to divide the time-axis into intervals of finite length T, measure the quantity of work $A_T(k)$ arriving in each such interval k, and then calculate

$$\hat{\lambda}_T(\theta) := \frac{1}{T} \log \frac{1}{N} \sum_{k=1}^{N} e^{\theta A_T(k)}. \tag{3}$$

Here, N represents the number of elapsed intervals of length T.

Given an estimate $\hat{\lambda}$ of the CGF we may use this to calculate an estimate $\hat{\delta}$ of the decay parameter appearing in (1). Our aim is to use these estimates to calculate a safe value for the service rate r which will ensure that the delay constraint is rarely violated. Work which becomes due when the queue length is already equal to q will suffer a delay of duration q/r. Therefore we must choose r such that

$$\mathbb{P}(Q \geq dr) \leq c. \tag{4}$$

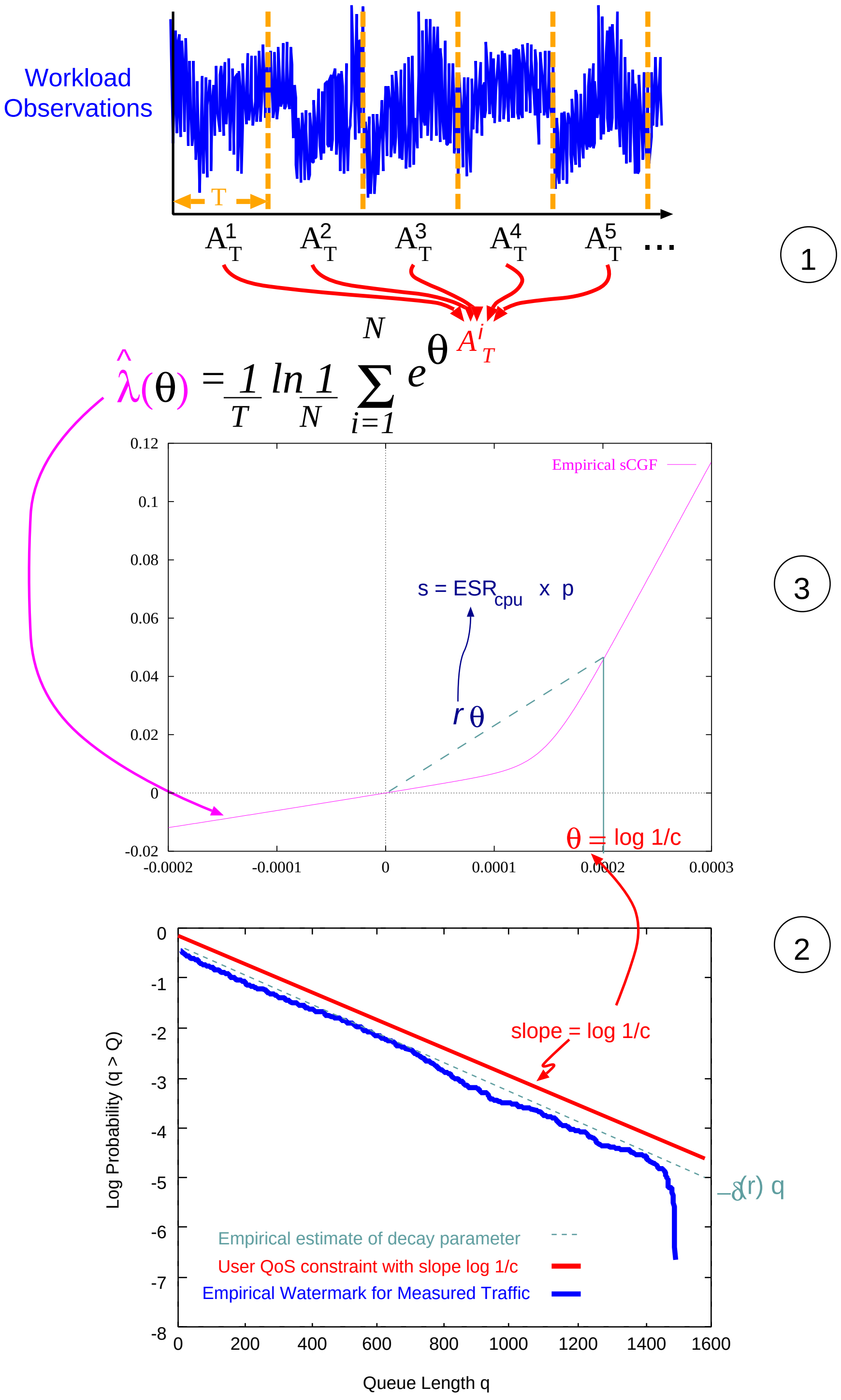

Figure 4. Obtaining the effective service rate $\widehat{ESR}$

We define the *effective service requirement* of the application to be the value of r calculated using (1) to satisfy (4):

$$\widehat{ESR}(d,c) := \min\{r : -\hat{\delta}(r)dr \leq \log c\}. \tag{5}$$

According to equation (2), $\widehat{ESR}(d,c)$ is the value of r which satisfies the equation

$$\lambda(\delta(r)) = -\frac{1}{d}\log c. \tag{6}$$

Recall that in Nemesis the CPU QoS parameters of an application are specified as a tuple (p,s,x,l). Given p, the quantum s may be determined directly from on-line estimation of effective service requirement:

$$s = \widehat{ESR}'_{\text{cpu}} \times p. \tag{7}$$

Here, $\widehat{ESR}'_{\text{cpu}}$ expresses a normalised proportion of CPU bandwidth between 0 and 1. Figure 4 shows the process of arriving at $\widehat{ESR}$ from a set of measurements of applcation activity.

In our previous work we have reported on a family of measurement-based CAC algorithms for ATM networks which use entropy estimates to determine the *effective bandwidth* of a traffic multiplex (effective bandwidth being equivalent to effective service rate in the present context).[7–9] The effective bandwidth of a traffic stream is a value between the mean and peak traffic rates which expresses the tradeoff between the performance impact of buffering (delay,jitter,loss) and the cost of transmission capacity. The properties of (3) as an estimator of the CGF have been studied in detail.[4,10] Application of this and other estimators to the CAC problem for ATM traffic has yielded good results.[3] The experiments discussed in the balance of this paper are designed in part to begin to test this theoretical framework with traffic measurements from Nemesis.

It is unfortunately not generally possible to observe directly the quantity of work that an application has to do. Instead of observing the arriving work A_t, we can however observe the amount of work actually done on behalf of the application by the CPU (or other system resource). The work done corresponds to the departure process from the single-server queue which represents the application, and it has been shown that, when the service rate is r, the CGF λ' of the departing work satisfies $\lambda'(\theta) = \lambda(\theta)$ for all $\theta \leq \delta(r)$, where $\delta(r)$ is the decay parameter obtained in the queue.[11] If we initially give the application more service than it needs to meet its QoS constraints, then the decay parameter $\delta(r)$ will be larger than its target value (where the target value is that which solves equation (6)). Therefore $\lambda'(\theta)$ will be equal to $\lambda(\theta)$ in the region of the target, so that $\widehat{ESR}(d,c)$ will be equal to the value of r which solves

$$\lambda'(\delta(r)) = -\frac{1}{d}\log c.$$

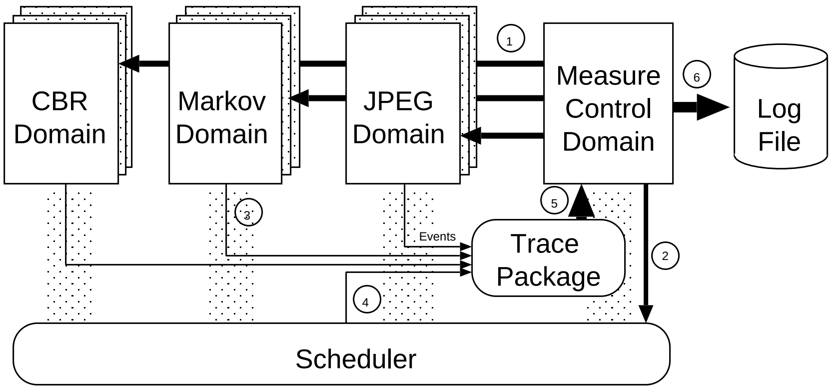

Figure 5. Instrumentation of Nemesis Domains

4 EXPERIMENTAL CONFIGURATION

To enable the collection of experimental data a tracing mechanism has been developed which allows events on a running Nemesis system to be recorded with minimal intrusion. This feeds data to an experimental harness which facilitates the collection and analysis of log-files and manages experiments using the entropy estimators.

Figure 5 shows the experimental configuration used. The system records logs of events from a running system which can be used in off-line traffic characterisation analysis. The events provide information on work arrival rates, service rates and scheduler activity for a selection of experimental domains. The experiments are controlled by the `Measure` domain which parameterises the applications (1 in Figure 5) and scheduler(4). A trace package records events generated by the scheduler(4) and application code(3) in an in-memory buffer with space for about 10^6 events. At the conclusion of an experiment the results are written to disk for off-line analysis(6). Each event records its time of occurrence and type, the domain identifier, and a counter.

Currently, the only resource considered is the CPU; other resources such as disk, network bandwidth and memory will be considered in future work.

4.1 Artificial Workloads

To verify whether the queue-theoretic results of section 3 are applicable in the context of Nemesis, we initially constructed a series of artificial workloads with well known queueing properties:

- `while(1)`: makes a continuous, infinite demand, and is used for validating the tracing system and CPU scheduler. This also serves as a model for non-multimedia applications such as compilers, etc.
- `CBR`: demands service at a predetermined, constant rate. This also simulates applications processing uncompressed video/audio.
- `Markov`: exhibits a bursty demand process from a 2-state Markovian ON-OFF traffic generator. In the ON state it generates work at a constant rate, and in the OFF it is quiescent. The lengths of successive sojourns in the ON and OFF states are independent geometrically distributed random variables. This is the simplest domain capable of producing a bursty workload.

Within each domain the abstraction of a 'queue' of arriving work is maintained; work arrives at the queue according to the arrivals process defined by the type of domain (CBR, Markov, etc) and is serviced at a rate determined by the kernel CPU scheduler. Each unit is typically a millisecond of work, represented (after calibration) as a number of executions of a particular piece of code. Units

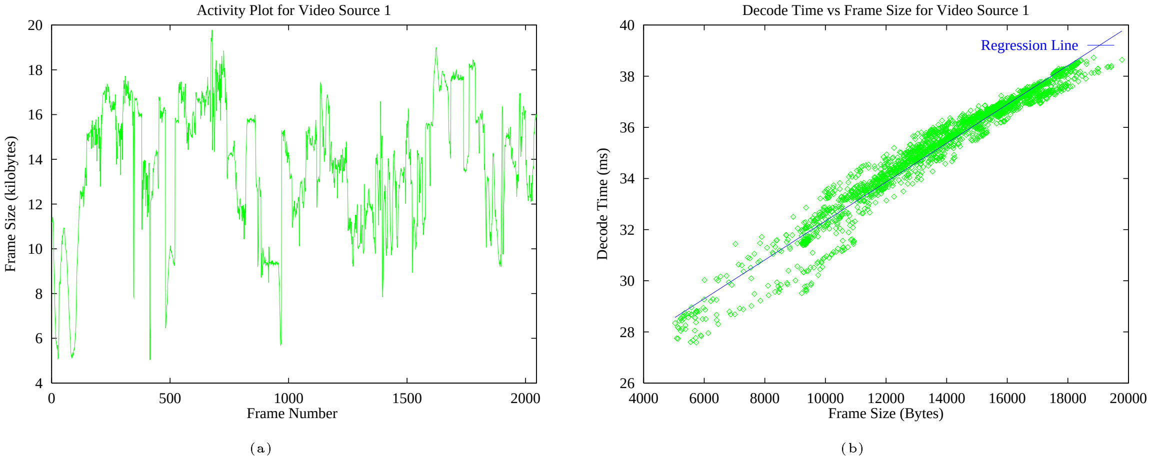

Figure 6. Statistics of an example Motion-JPEG trace. Graph (a) is an activity plot of frame sizes, (b) shows the linear relationship between frame size and decode time

of work are removed from a virtual queue by a thread which executes the appropriate number of iterations of the calibrated section of code.

Each domain was instrumented to keep track of a queue length which could be compared with the theoretical queue length distribution in a fixed rate single-server queueing system. Trace events are recorded when a domain is handed the CPU, and when work is added to or removed from the application queue.

4.2 More Realistic Workloads

Ultimately, we are interested in workloads from applications processing real continuous media data, such as motion JPEG decompression, feature tracking and MIDI file playout. Since the JPEG domain is of considerable interest we will concentrate on its behaviour in this paper. JPEG is a lossy image compression technique consisting of a Discrete Cosine Transform (DCT) and Entropy (in this case Huffman) encoding. Motion-JPEG is a compressed video format in which each frame is independently encoded thus allowing simple application degradation by simply discarding data.

In order that experiments may be repeated with a number of different services rates, we use a simple application to pre-record motion-JPEG PDUs from the (ATM) network device driver onto disk. A second domain is used to decode the JPEG data from an in-memory buffer with varying service rates. Video is stored in the format produced by a Fore Systems AVA-200, along with playout timestamps allowing the data to replayed to a decoder in a repeatable fashion. This also simplifies the queueing system by removing interaction with the device driver drivers.

For experimental purposes, JPEG frames arrive in the decoder domain's work queue at a fixed rate. "Frames" of work are removed from the queue when they have been successfully decoded for display. It is the variable CPU demands of this decoder domain which are of interest. Figure 6 shows the statistics of a small (30MB) section of Motion-JPEG video. It can be seen that decode times for compressed frames exhibit similar variations to the bandwidth observed on the ATM network. Determination of a guaranteed CPU service rate so as to bound the probability of delay exceeding a specified threshold is thus analogous to the allocation of network bandwidth so as to bound the probability of buffer overflow.

5 VALIDATION OF THE QUEUE-SERVER PARADIGM

In this section we present results obtained from the application of our theoretical framework to measured application activities on Nemesis. Our work is still at an early stage – we are currently

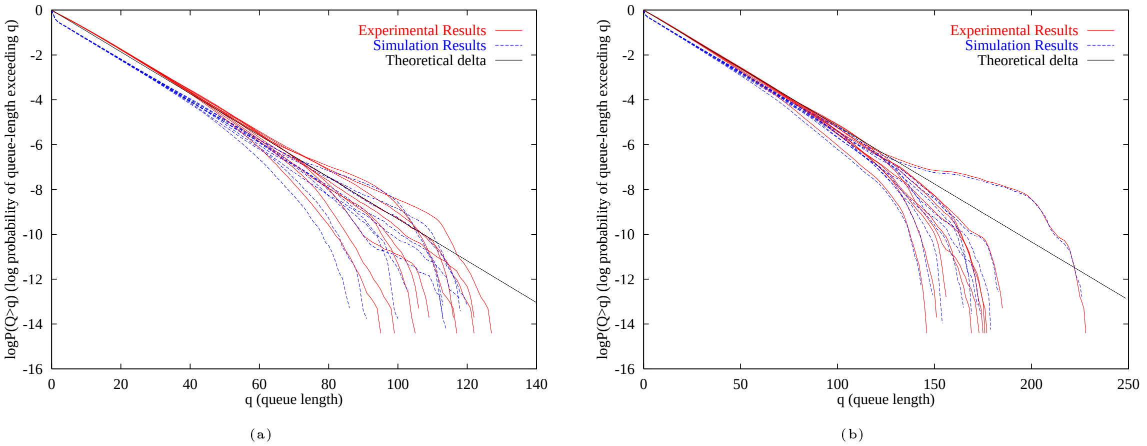

Figure 7. Performance comparison of the theoretical and simulated queue-server paradigm using a watermark plot of the simulated queue, Nemesis domains and theoretical $\delta(r)$ and Markovian traffic of load (a) 0.75, and (b) 0.85

implementing an on-line resource allocation framework which uses entropy estimates directly. The results here are based on off-line analysis of traces of application activity – by using the observations of application activity to set system parameters we can then repeat the experiment and observe whether the predicted system performance results. Our ultimate goal is to derive, for given application period p, a slice s which will ensure that it meets its QoS constraints whilst allowing the operating system to operate at maximum efficiency. We succeed in this if we can obtain an empirical slope $\hat{\delta(r)}$ which satisfies the constraint

$$-\hat{\delta}(r) \leq \log c. \tag{8}$$

For given r we can observe the frequency with which the constraint q is exceeded. If a watermark plot of $\log \mathbb{P}(Q > q)$ vs q falls below a line of slope $-\delta(r)$, then the allocation of a rate r is sufficient to meet the application constraints (q, c). Depending on the type of system and application task the load s/p will vary: if a large amount of computation must be performed each period, the load s/p will be close to 1. Our principal aim in the remainder of this paper is to investigate whether the theory of Section 3 can be applied to Nemesis. If it does, then we will have demonstrated that the Nemeisis "firewalls" which guarantee application resources are valid, and also that results from Large Deviations theory can be used to develop a measurement based resource allocation scheme.

5.1 Single Domains

Experiments were conducted using two-state Markov domains with *guaranteed* allocations of CPU time chosen so as to produce the effect of a single server queue at 75%, 85% and 95% loads. Traces were obtained for a number of random number generator seed values.

Identical sample paths were passed through an off-line simulator of a single-server queueing system to produce watermark plots. These were compared with watermark plots derived from the Nemesis scheduler traces. For each of the ten sample paths used, we estimate $\mathbb{P}(Q > q)$ by the normalised frequency with which Q exceeds a given length q during the course of the experiment and plot the logarithm of this value against q. Figure 7(a) and (b) show that the empirical watermarks obtained from the simulated queue agree closely with those obtained from the Nemesis traces and the asymptotic slopes of both are close to the calculated value. The graphs show the simulated values, the empirical values (from Nemesis scheduler traces) and a theoretical line of slope $-\delta$ derived for this traffic using Large Deviations Theory. The figure shows that the empirical watermarks

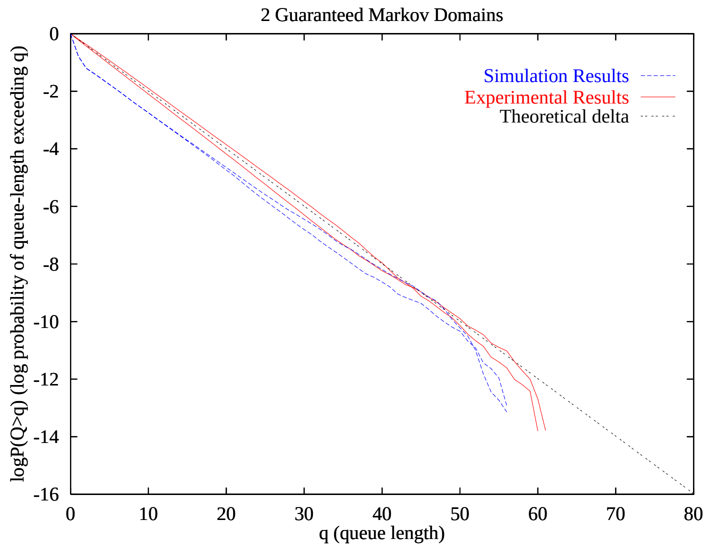

Figure 8. Watermarks and theoretical $\delta(r)$ for 2 Markov domains at a load of 0.99

from the simulated queue agree closely with those from the Nemesis traces and the asymptotic slopes of both are close to the calculated value. In examining the variation between simulated and experimental distributions, it must be remembered that there is a statistical variation within each group. Our results show that at higher loads there is some variance in the slopes of the watermark plots but its magnitude is roughly the same in both the simulated and experimental cases. The sudden tail-off at the lower end of the plots is due to the finite run-length of the experiments.

The sCGFs of the software virtual queue and the Nemesis scheduler were compared to test whether the two traces are distinguishable on the scale of large-deviations. We found the two systems to be identical from this point of view – the difference between the experimental and the simulated sCGFs in all cases was smaller than the statistical variation due to different sample paths. From these results we conclude that, for this well understood traffic at least, the queue-server paradigm holds, and that our measurement based resource allocation regime is applicable.

5.2 Multiple Domains

Figure 8 presents watermark plots for the workload (in ms) of two Markov domains running simultaneously on a Nemesis system. The domains had identical parameters, apart from the random number seed, and used guaranteed time only. Also shown on the plots are the straight line slopes $\delta(r)$ predicted by Large Deviation Theory. Comparison of the plot for each of the domains with results from off-line simulations of each domain in isolation shows the simulated watermarks to be virtually identical to the measured values shown, indicating that under Nemesis each domain is effectively isolated from the rest of the system, and confirming the "firewall" performance of Figure 2[‡]. These results confirm the unique ability of Nemesis to provide multiple domains with guaranteed resources to within an extremely fine tolerance. In summary, our queue-paradigm is broadly applicable in a system which schedules multiple domains, however the problem of cache and TLB miss effects can be expected to affect performance significantly.

5.3 Real Applications: the JPEG Domain

Figure 9(a) shows the empirical values of the watermark for the motion JPEG domain. Whereas in the previous experiments, the lengths of virtual queues have been used to measure the observed

[‡]This experiment was repeated using configurations of two guaranteed service Markov domains with one guaranteed continuous demand domain, and one guaranteed Markov domain with one guaranteed JPEG domain. In all cases the domains were found to act independently of each other.

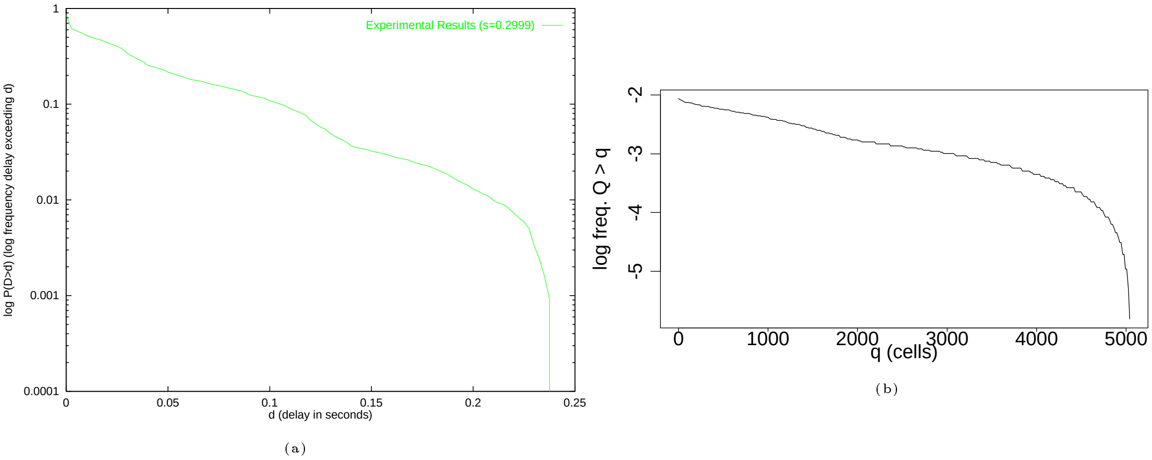

Figure 9. (a) Watermark plot for motion-JPEG decoder (b) Network queue length

QoS of experimental domains, for the JPEG decoder we use the PDU-decode latencies as "queue lengths". The log-linear decay rate is clearly visible, as it is when we observe a measurement of the cell-level process for the same traffic at an ATM switch buffer, as shown in Figure 9(b). We are currently applying the Measure estimators to the workload process of the JPEG domain, but note that here we face a problem: it is difficult to identify a real "queue" within the application, and moreover the service demands per PDU are not entirely accurately predicted by the line of regression in Figure 6(b). Finally, since PDUs are of variable sizes and bring different amounts of work to the queue, it is not easy to discard work latency from the queue in fixed amounts. This complicates the process considerably.

6 MEASUREMENT-BASED RESOURCE ALLOCATION

Figure 10 presents the results of a set of experiments in which we applied the on-line measurement and estimation technique described above, to Nemesis. Here we use the Markov domain, and specify maximum tolerance of latency for any item of work added to the queue. Any item of work on the queue whose latency exceeds the (user specified) bound is discarded. The ratio of the work lost as a result of missing the latency constraint to the total work is termed the "Work Loss Ratio". Our experiments aim to find, for this domain, the effective service rate required to meet the constraint. The figure plots two sets of empirically derived results, and a line which reflects the WLR predicted for the Markov source process, derived analytically for each value of the service rate. Each family of curves was obtained from use of the same set of random number seeds in independent experiments. The lower family of curves were obtained by restricting the service rate for the Markov domain and measuring the WLR experienced. The upper family of curves was obtained from a "calibration" trace in which CPU work done by the domain was measured whilst the domain was running with no restriction on its service rate (i.e. it was given full access to the CPU). Applying the Large Deviations Theory to each trace yielded the estimated effective service rate, for a range of WLR values.

It is clear that the estimated service rate requirement, predicted on the basis of observation of the CPU used by the domain, agrees closely with the theoretically predicted value, but is consistently higher by around 1%. This is to be expected, given that we are not only measuring the work done due to the Markov arrivals process, but also the overheads of generating the arrivals, thread scheduling, context switches, and other operating system overhead. In the area of interest, where the WLR is under 0.01, the observed loss is slightly lower than predicted but also agrees closely with the theoretical results. If we factor in the overheads of running on a real OS, rather than a theoretical simulation, accurate and consistent estimation of required service rate to satisfy a given QoS contract can be obtained using this method.

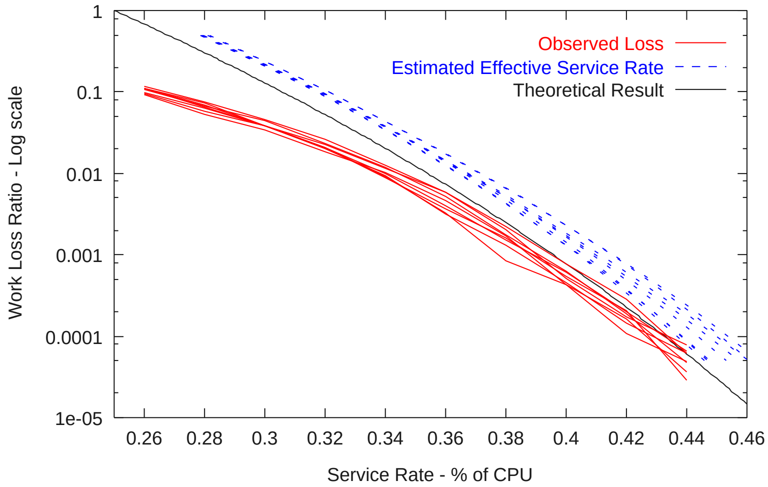

Figure 10. Application of entropy estimates to resource allocation on Nemesis

7 CONCLUSIONS

This paper has introduced a novel approach to the problem of resource allocation for multimedia applications. The use of Large Deviations theory in measurement based admission control for multi-service networks is now well established. Measurement based CAC algorithms have been implemented which are both computationally inexpensive and near optimal in performance.

Our approach extends this theory into the end system, based on the use of the unique Nemesis multimedia operating system which is capable of handing out explicit guarantees of resources to applications. Our early results show that for applications whose workload is well understood our theoretical queue-server paradigm and measurement based approach perform excellently.

We have demonstrated that a measurement based approach to resource allocation can successfully determine the service rate necessary to sustain the Quality of Service requirements of real multimedia applications. It is clear that these techniques are beneficial for calibration of multimedia application requirements. The rôle of admission control in a multi-service operating system is still an area of active research: many issues remain to be addressed, including the inter-dependence of applications sharing a cache and TLB.

REFERENCES

1. M. B. Jones, D. Rosu, and M.-C. Rosu, "An Overview of the Rialto Real-Time Architecture," in *Proceedings of the Seventh ACM SIGOPS European Workshop*, September 1996.
2. I. Leslie, D. McAuley, R. Black, T. Roscoe, P. Barham, D. Evers, R. Fairbairns, and E. Hyden, "The Design and Implementation of an Operating System to Support Distributed Multimedia Applications," *IEEE Journal on Selected Areas in Communication* **14**, pp. 1280–1297, September 1996.
3. S. Crosby, I. Leslie, J. Lewis, R. Russell, F. Toomey, and B. McGurk, "Practical Connection Admission Control for ATM Networks Based on On-line Measurements," in *IEEE ATM '97*, IEEE, (Lisbon), June 1997.
4. N. Duffield, J. Lewis, N. O'Connell, R. Russell, and F. Toomey, "Entropy of ATM traffic streams," *IEEE Journal on Selected Areas in Communications, Special issue on advances in the fundamentals of networking – part 1* **13**, August 1995.
5. C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment," *Journal of the Association for Computing Machinery* **20**, pp. 46–61, January 1973.
6. J. Lewis and R. Russell, "An Introduction to Large Deviations. (Tutorial)," in *Performance '96*, IFIP WG 7.3, (EPFL, Lausanne, Switzerland), October 1996.
7. J. Y. Hui, "Resource Allocation for Broadband Networks," *IEEE JSAC* **6**, pp. 1598–1608, Dec. 1988.
8. F. Kelly, "Effective Bandwidths at Multi-Class Queues," *Queueing Systems* **9**, pp. 5–16, 1991.

9. H. Ahmadi and R. Guérin, "Bandwidth allocation in high speed networks based on the concept of equivalent capacity," in *Broadband Technologies*, Proceedings Seventh ITC Specialist Seminar, (New Jersey), Oct. 1990.
10. A. Ganesh, "Bias Correction in Effective Bandwidth Estimation." 1996 Preprint.
11. N. O'Connell, "Large Deviations and Queueing Networks," Tech. Rep. DIAS-STP-94-13, Dublin Institute for Advanced Studies, 1994.