Response times in Client-Server Systems

A.J. Field* , P.G. Harrison*, J. Parry†

Abstract. Response time is the key performance measure in on-line transaction processing systems and other client-server architectures. Not only is it important to achieve a low average value and correspondingly high throughput, but response time should also be fairly consistent in order to provide a good quality of service. We develop a new algorithm for computing the probability density function of response times in Markovian models of client-server systems. We model the clients and servers by central server queueing networks and obtain response time densities as simple functions of time under independence assumptions that are shown to hold asymptotically as network size increases. The communication network is modelled as a single server queue with mean service time determined by its operational characteristics. We consider an Ethernet and construct a new model, of interest in its own right, that captures details not modelled hitherto. This model is validated against simulation and shows good agreement up to moderate utilisations, the normal operating environment for Ethernets. The whole client-server model is implemented in the Metron Athene Client-Server capacity planning tool and sample runs are examined.

1. Introduction

The proliferation of client-server systems in business continues unabated, as applications are split into local tasks run on "client" workstations and resource-intensive computations run on a "server" mainframe. The architecture is diverse, typically supplied by several vendors using several operating systems. It is also complex with new levels of systems software such as the Distributed Computing Environment (DCE) superimposed on the traditional layers of operating system, database and application. Many of these systems are provided for non-IT staff, so inadequate performance is directly exposed.

Quality of service in client-server systems is typically measured by *response time*. This is the time elapsed between a client issuing a request to the server and the arrival at the client of the response (e.g. data); alternately, it is sometimes considered as the elapsed time between successive requests, i.e. including processing time and thinking time at the client. Not only is it important to achieve a low average response time and correspondingly high throughput, but response time should also be fairly consistent in order to provide a good quality of service. In other words, it is important to be able to estimate the probability distribution of response time and, indeed, 95th quantiles are specified in the TPC benchmarks.

We derive an approximate algorithm for computing response time probability density functions for client-server systems and demonstrate it through the capacity planning tool Athene Client-Server developed by Metron Technology Ltd, [5]. The clients and servers are represented by standard central server queueing networks and the communication network by a novel, customised model of the Ethernet [4]. This model captures details not modelled hitherto and is of interest in its own right, but we use it here to determine the service rate of a single server queue that we use to approximate the behaviour of the network in the wider model. A similar approach can be used for other types of network such as token rings. Under appropriate independence assumptions,

* Department of Computing, Imperial College, London

† Metron Technology Limited, Taunton

which are shown to hold asymptotically as network size increases, response time is a convolution which can be simplified and then inverted analytically to give the probability density directly as a function of time. Implementation is thereby easy compared with conventional methods that rely on numerical inversion of Laplace transforms.

The paper is set out as follows. In section 2, the model structure for client server systems is described and in the next two sections the sub-models corresponding to clients and servers (viz. central server queueing networks) and the Ethernet are presented in detail. Section 5 presents some numerical results as computed by Athene Client-Server and the paper concludes in section 6.

2. Client-server models

In the simplest of client-server systems, there are logically three sources of delay: the client (a sum of sub-delays comprising user "think time" at a terminal and further ones in the host computer system), the network (in two directions corresponding to a request and a reply) and the server. More generally, a client-server system consists of a more complex network of nodes linked by communication networks of various kinds. These nodes comprise the clients and various levels of server. If a client's request can be satisfied at a 'local' server, then only that one server and the network connecting to it are involved in the transaction. Otherwise the request must be forwarded, over another network, to a 'second level' server and possibly to higher level servers until the request can be satisfied. The topology of such systems can therefore be quite complex with the sharing of applications and databases, for example. Response time now becomes dependent on the level of server to which a transaction has to be sent and response time distribution or average response time is a weighted sum of the corresponding quantity conditioned on the transaction path required.

In order to calculate average response time for a given transaction path, we merely need to sum the average delays in each component in that path. In the simplest case this is just $m_c + m_{n1} + m_s + m_{n2}$ where m_c, m_{n1}, m_s, m_{n2} are the mean delays at the client, network (from client to server), server and network (from server to client) respectively. More generally, the sum will be

For response time distribution, we make the approximating assumption that each node and network yields an independent contribution to a transaction delay, with corresponding probability distribution functions $F_c(t), F_{s_i}(t), F_{n_j1}(t), F_{n_j2}(t)$. We denote the density (which will always exist) and Laplace-Stieltjes transform (LST) of a probability distribution function $F(t)$ by $f(t)$ and $F^*(s)$ respectively. Then the LST of response time distribution is approximated by

It therefore remains to compute the LSTs for each component and invert where possible to find the required probability density function of response time. To this end, in the next section we represent both client and server nodes by central-server queueing network models of the type shown in Figure 1. More generally, specific models should be tailored to the type of node in question, for example a parallel database server. However, the resulting LST would then be exceedingly complex, perhaps only available numerically and almost certainly requiring numerical inversion. We do not pursue this further in this paper.

Regarding network delays, we develop a detailed model of the ubiquitous Ethernet in section 4 and obtain a good approximation for its mean. At reasonably low utilisations, the ethernet can be adequately modelled as a single server queue and, using the value calculated for the mean delay to calibrate it, a highly efficient approximate algorithm has been implemented for transaction response time density. Greater accuracy can be obtained, particularly at higher loading, at increased computational cost by using the LST directly and inverting numerically. However, the normal operating load on an Ethernet must usually be kept at a low level to achieve adequate performance.

3 Central server models

We use a central server model of the type shown in Figure 1 to represent both client and server nodes; see [2] for example. This is a single class queueing network consisting of a CPU (the central server) and M devices parameterized as follows:

1) The average total CPU time of a task, T;

2) The average number of visits a task makes to each device i, v_i $(1 \le i \le M)$

3) The average service time (per visit) at each device i, $1/\mu_i$ $(1 \le i \le M)$

4) The external arrival rate to the CPU, λ

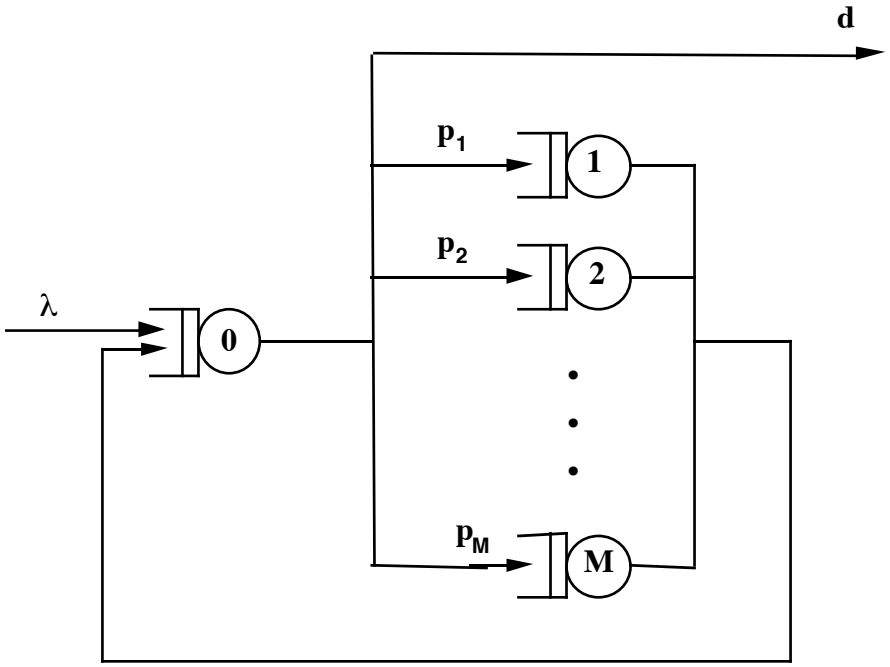

Figure 1. A central server queueing network

The CPU is labelled 0, and the M devices are labelled 1 to M. We therefore define the visit count and service rate of the CPU to be

$$v_0 = 1 + \sum_{i=1}^{M} v_i \quad \text{and} \quad \mu_0 = \frac{v_0}{T}$$

In addition, the routing probabilities shown in Figure 1 are defined to be $p_i = v_i/v_0$ and the probability of departing the network after visiting the CPU is $d = 1/v_0$.

Now, the response time of a task in a queueing network is the sum of its busy times at each of the nodes it visits. Hence, average response time is the sum of the average busy times, an easy calculation. We consider central server networks with several devices in which tasks make a large number of cycles before departing. We then make the approximating assumption that a task's busy times at each node are independent random variables. This appears a strong assumption which is clearly not valid in networks with only one device, for example. This is because the number of visits to the CPU must be the same as the number of visits to the device; hence the respective visit times are strongly correlated. However, as the number of devices M increases, the assumption holds asymptotically provided no small set of devices dominates, i.e. only a few of the visit counts are significant. This is formalised in the following:

Proposition 1

Let V_i denote the random variable for the number of visits a task makes to device i $(1 \le i \le M)$ and $V = V_1 + \ldots + V_M$ denote the number of cycles made by a task. Then

$$\mathrm{P}(V_1 = n_1, \mathsf{K}, V_M = n_M \mid V = n) \approx \prod_{i=1}^{M} \mathrm{P}(V_i = n_i \mid V = n)$$

as $n \to \infty$ provided no device *i* dominates.

Proof

The left hand side of the equation is

$$L = \frac{n!}{\prod_{i=1}^{M} n_i!} \prod_{i=1}^{M} p_i^{n_i}$$

and the right hand side can be written $R = L \times U$ where

$$U = (n!)^{M-1} \prod_{i=1}^{M} \frac{(1-p_i)^{n-n_i}}{(n-n_i)!}$$

We therefore need to show that $U \to 1$ as $n \to \infty$. Now write $p_i = k_i / n$ where $\sum_{i=1}^{M} k_i = n$ and each k_i is small compared to n under the hypothesis that no device *i* dominates. Applying Stirling's formula to the factorial terms, as $n \to \infty$,

$$U \approx \left(\frac{n}{e}\right)^{(M-1)n} \prod_{i=1}^{M} \frac{(1-k_i/n)^{n-n_i}}{(n/e)^{n-n_i}(1-n_i/n)^{n-n_i}}$$

Thus, as $n \to \infty$, for n_i small in comparison to *n*,

$$U \approx \prod_{i=1}^{M} \frac{e^{-k_i}}{e^{-n_i}} = \frac{e^{-n}}{e^{-n}} = 1$$

However, if any n_i is large enough to be comparable with *n*, both the probabilities *L* and *R* are neglibible.

QED

We now need to compute the busy time distribution at each node, $B_i(t)$ at node *i* say. Let the *j*th waiting time of a given task at node *i* be denoted by W_j and let the task make *V* visits to node *i*. Then the LST of the busy time distribution is

$$B^*(s) = E[e^{-s(W_1+\mathsf{K}+W_V)}] = E[E[e^{-s(W_1+\mathsf{K}+W_V)} \mid V]] = E[(E[e^{-sW_1}])^V] = G_V(W^*(s))$$

where G_V is the probability generating function of the random variable *V* and $W^*(s)$ is the LST of the waiting time distribution at node *i*. In an open Markovian network, the waiting time density at node *i* is $W_i^*(s) = (\mu_i - \lambda_i)/(s + \mu_i - \lambda_i)$ where $\lambda_i = \lambda v_i$ for $(0 \le i \le M)$.

For the CPU, the number of visits made by a task is geometric so that

$$G_{V0}(z) = \sum_{n=0}^{\infty} d(1-d)^n z^{n+1} = \frac{zd}{1-z(1-d)}$$

and hence, after simple algebra,

$$B_0^*(s) = (\mu_0 - \lambda_0)/(v_0 s + \mu_0 - \lambda_0)$$

For the devices, to find the probability that a task makes *v* visits, we condition on the number of cycles *n*, i.e. visits made to the CPU. Hence we find for device *i* $(1 \le i \le M)$

$$G_{Vi}(z) = \sum_{v=0}^{\infty} \sum_{n=v}^{\infty} d(1-d)^n \binom{n}{v} q_i^{\,v} z^v (1-q_i)^{n-v}$$

where $q_i = p_i/(1-d)$. Changing the order of summation (with $n \ge v$) and rearranging, we obtain

$$G_{Vi}(z) = \sum_{n=0}^{\infty} d(1-d)^n (1-q_i^{\,n}) \sum_{v=0}^{n} \binom{n}{v} \left(\frac{q_i z}{1-q_i}\right)^v = \frac{d}{d + p_i(1-z)}$$

Substituting $W_i^*(s)$ for z then yields

$$B_i^*(s) = \frac{s + \mu_i - \lambda_i}{(v_i + 1)s + \mu_i - \lambda_i} = \frac{1}{v_i + 1} + \frac{(\mu_i - \lambda_i)v_i/(v_i + 1)}{(v_i + 1)s + \mu_i - \lambda_i}$$

Under our independence of busy times assumption, the LST of the response time distribution for the central server queueing network is now

$$R^*(s) = \prod_{i=0}^{M} B_i^*(s)$$

This can be inverted analytically, using the method described below, to give the density $r(t)$ of response time. However, to illustrate the method, we make the further simplifying assumption that $d << p_i$ for all i. This is frequently reasonable; for example if $d = 1/M^2$ and all the p_i are equal, then each p_i is of the order $1/M$. When the assumption does not hold, the problem of LST inversion becomes a collection of inversions of the type we now consider when it does hold.

For $d << p_i$ for all $i>0$, i.e. $v_i >> 0$, we find that $B_i^*(s) \approx \frac{\mu_i - \lambda_i}{(v_i + 1)s + \mu_i - \lambda_i}$

and hence that

$$R^*(s) = K \prod_{i=o}^{M} \left(\frac{1}{s + a_i} \right) \qquad (1)$$

where $a_i = (\mu_i - \lambda_i)/(v_i + 1)$ for $i > 0$, $a_0 = (\mu_0 - \lambda_0)/v_0$ and $K = \prod_{i=0}^{M} a_i$.

The problem has now been reduced to finding an expression for $R^*(s)$ which can be straightforwardly inverted, for example by using the method of partial fractions. If all the a_i's are distinct, we obtain:

$$R^*(s) = K \sum_{i=0}^{M} \frac{r_i}{(s + a_i)}$$

where the factors r_i are given by

$$r_i = \frac{(-1)^M}{\prod_{j=0, j\neq i}^{M} (a_i - a_j)}$$

This can be proved either by standard linear algebra, or (more straightforwardly) by induction on M. Using the facts that $R^*(0) = 1$ and $R^{*\prime}(0) = -\sum_{i=0}^{M} 1/a_i$ (from equation (1)), we find that

$$\sum_{i=0}^{M} \frac{r_i}{a_i} = \frac{1}{K} \quad \text{and} \quad \sum_{i=0}^{M} \frac{r_i}{a_i^2} = \frac{1}{K} \sum_{i=0}^{M} \frac{1}{a_i}$$

We have now decomposed $R^*(s)$ into a linear combination of Laplace transforms of exponential densities, and thus, since the Laplace transformation is a linear operator, we can invert it to give the density function r(t) as

$$r(t) = K \sum_{i=0}^{n} r_i e^{-a_i t}$$

Finally in this section we note that a transaction path consists of a client node, a number of server nodes and corresponding network nodes which we shall represent as M/M/1 queues. We assume that all of these components are independent and hence that the LST of system response time distribution is a product of products of the form of $R^*(s)$ in equation (1). This is just another expression of the same form and hence is invertible in the same way to give response time density as a function of time.

4 The Ethernet and its modelling

The methodology described in section 2 applies to all types of client-server network; it is necessary to determine only the mean latency of the network so that the appropriate service rate can be chosen for the M/M/1 queue used to represent it. If the basis of the contention for the network is queueing, then the service rate must be chosen so that the resulting waiting time matches the mean latency. If the basis of contention is sharing, a delay ('infinite') server can be used instead whereupon service time will be the same as latency.

We choose the Ethernet for our network because of its predominance as a local area network (LAN). From its origins at Xerox in the mid-seventies, Ethernet [4] has become the most successful LAN technology. Recent developments such as Fast Ethernet and GigaBit Ethernet offer a relatively easy way to improve the bandwidth of an existing Ethernet to cope with the ever increasing demand.

Boggs, Mogul and Kent [1] give a detailed account of the many attempts which have been made to model Ethernets. As they point out, many models which have been developed do not model the correct Medium Access Control layer. There are many models which aim to find the throughput for various values of the 'offered load', but yield no information on the average packet delay. Many models make unrealistic assumptions such as constant message lengths. The model presented here is based on simple probability theory, gives the average packet delay and any assumptions made are explicitly stated.

4.1 Modus operandi

Upper layer protocols are not considered here: this discussion centres primarily on the Medium Access Control layer. In the case of an Ethernet, the process by which a message gets access to the wire is called Carrier Sense Multiple Access with Collision Detection (CSMA/CD). The classic analogy of this process is a dinner party conversation. Everybody around the dinner table must listen for a period of quiet before speaking. Once a space occurs, each person has an equal chance to say something. If two people start talking at once, they both realise and both try to speak again a short time later.

The following algorithm is followed each time a message requires access to the medium:

1) If the medium is idle, **transmit**. Otherwise go to step 2.
2) If the medium is busy, continue to listen until the channel is idle, then **transmit**.
3) If a collision is detected during transmission, transmit a brief **jamming signal** to ensure that all stations know there has been a collision. After transmitting this signal, wait (back off) a random amount of time, then attempt to transmit again – step 1.

Collisions are integral to the correct functioning of an Ethernet - many efficient Ethernets have a large number of collisions per hour. However, as an Ethernet becomes busier and busier there will be a point at which messages spend more time colliding and backing off than actually transmitting, and performance degrades rapidly.

The above algorithm is '1-persistent'. This means that once the medium is sensed idle a transmission is always attempted (i.e., is attempted with probability 1). Other MAC algorithms are 'p-persistent' – on sensing the medium idle, transmission is attempted with probability p. These tend to have worse performance characteristics at low loads, but offer a better performance at high loads. The IEEE 802.3 standard is for 1-persistent CSMA/CD.

There is always an inter-frame gap of 9.6 microseconds between successive data transmissions. This figure arose originally because Ethernet cards took this long to switch from listening mode to transmit mode. This is done much faster by modern Ethernet cards, but the figure has remained. Each station waits for the network to be idle for 9.6μS before transmitting.

Imagine now the scenario of two stations A and B that try to access the medium at the same time. The station A starts transmission, because the network has been idle for 9.6μS. It starts transmission to the station C.

A ===‡ B C D

B has a message ready, and has also seen that the network has been idle for 9.6uS, so begins transmission to station D.

A==========‡ fl=B=‡ C D

The messages clash:

A ========== fl***‡ ======B=========‡ C D

A short time later B senses a clash and starts transmission of the jamming signal

A =====fl**********fl!!!!!!!!!B!!!!!!!!‡======C========‡ D

The jamming signal is **4 bytes** long. Once B has transmitted the jamming signal it stops transmission. When A detects the collision it also transmits a jamming signal and stops transmission. Both stations then back off before attempting to transmit again. The jamming signal causes all stations to realise that a collision has occurred, and any station which detects a collision while receiving a message will discard that message as invalid.

Immediately after a collision, after a station has stopped transmitting the jamming signal, the station backs off for a time determined by the truncated binary exponential backoff algorithm (not to be confused with the exponential distribution) before attempting to retransmit. This is described below. The 'slot time' referred to is the time taken to transmit the smallest possible message (64 octets) = 51.2 microseconds for a 10Mbit Ethernet.

1) After the **first** collision, a station waits either 0 or 1 slot times. The probability of each is _.
2) After the **second**, the station waits one of (0,1,2,3) slot times with probability _.

.
.
.

3) After the **tenth**, the station waits one of (0, ..., $2^{10}-1$) slot times, with probability $1/2^{10}$
4) ... 15) Subsequently, the backoff time is chosen as if it were the tenth collision.

Thus the maximum back-off time picked subsequent to a collision is 2^{10} slot times = 51.2 milliseconds. After fifteen collisions (sixteen transmission attempts) the message is 'dropped' and this situation is handled by the higher level protocol. The time to process the exception is typically up to 100 - 1000 times longer than the transmission time.

4.2 An Ethernet model

The following assumptions are made in the model described here.

1) The offered arrivals have an exponentially distributed inter-arrival time (Poisson arrival process).
2) The combined arrivals (offered arrivals plus retries) also have an exponentially distributed inter-arrival time.
3) All stations are separated by a constant distance (star formation).
4) Backoff times are truncated binary exponential.
5) A message fails if the 16^{th} attempt to acquire the network fails.

We use the following notation and parameter values in the numerical results of section 4.3.

Ethernet speed = v = 10 MegaBits/Second

Propagation time (maximum possible) = Δ =25.6 μS

Preamble time (8 Bytes) = T_{PRE} = 2.6 μS

Jamming time (4 Bytes) = T_{JAM} =1.3 μS

Interframe gap = T_{IFG} =9.6 μS

Slot time (64 Bytes) = T_{SLOT} =51.2 μS

Minimum message time (64 Bytes) = T_{MIN}=51.2 μS

Maximum message time (1518 Bytes) = T_{MAX}= 1214 μS

4.2.1 A model with constant message lengths

To simplify the presentation, we first consider an Ethernet that carries messages of constant length, before generalising to arbitrary message lengths in the next subsection. Message transmission time, which excludes time due to collisions and backoffs, is denoted by T. The offered arrival rate, which excludes arrivals due to retries, is denoted λ. The utilisation – the probability of observing the Ethernet transmitting a successful message – is therefore given by $\rho = \lambda T$. The total arrival rate, which includes retries, is denoted by Λ, which we calculate below.

The probability of a message colliding is dependent both on the current backoff state of the message and the backoff state of any message it has previously collided with. For example, take two messages which collide together on their first attempt. Each choose a backoff time of either 0 or 1 slot times. If they choose the same time they will collide again, so the probability of another collision is at least a half.

We make the following three simplifying assumptions:

1) All collisions involve only two messages.
2) If two messages choose the same backoff time after a collision, they will collide again.
3) If two messages choose different backoff times after a collision, the probability of a collision is the same as that of the first attempt, with a modified network utilisation term (described below).

The first assumption is expected to give good results for low utilisations (the usual operating domain of an Ethernet) but may under-estimate the collision rate when the network is busier. Note as any message's backoff state becomes large, the probability of it colliding at the next attempt should tend to the probability of a collision at its first attempt, simply because it will have waited a long time before attempting to transmit again.

In this model, each incoming message has a *state number*, with state number 1 meaning the message is making its first attempt, etc. Denote the probability of failure at the nth attempt as p_n. Then the total arrival rate is given by

$$\Lambda = \lambda(1 + p_1 + p_1 p_2 + \ldots + \prod_{j=1}^{15} p_j).$$

The upper limit, 15, of the product reflects that each message tries up to sixteen times before giving up. The probability q_j of an arbitrary incoming message being in state j is given by

$$q_j = \frac{\lambda}{\Lambda} \prod_{k=1}^{j-1} p_k, \qquad 1 \le j \le 16$$

where the empty product is defined to be 1.

We first consider a message which is making its first attempt at transmission. The message suffers a collision if either

1) the network is busy and there is at least one other arrival during the transmission of that message or during the original message's initialisation phase (one propagation time, Δ) immediately preceding it; or

2) the network is idle and at least one other message arrives within two propagation times (one before and one after the arrival instant).

In the steady state, the probability that the network is busy on arrival is given by ρ by the random observer property of Poisson processes. Since the total arrival process is assumed to be Poisson, p_1, the probability of a collision at the first attempt, is given by

$$p_1 = (1-\rho)(1-e^{-2\lambda\Delta}) + \rho(1-e^{-\lambda(T+\Delta)})$$

Here, Δ is the propagation time (assumed constant) between stations – the time taken for a signal to traverse the Ethernet from one station to another. This equation should be compared to Kleinrock's collision probability for CSMA [3], where message transmission times are not taken into account.

If a message suffers a collision, it must have collided with at least one other message; by the first assumption above, *exactly* one other message. If both messages choose the same backoff time, they will collide again (assumption two). If they choose different backoff times, the probability of successful transmission is assumed to be the same as at the first attempt (assumption three). Thus the probability of failure at the *i*th attempt is

$$p_i = \sum_{j=1}^{16} q_j (P_{i-1,j} + (1 - P_{i-1,)\,j}) p_{1i}), \qquad 2 \le i \le 16$$

where P_{ij} is the probability of a direct clash between a message that was previously in state *i* and one which was previously in state *j*,

$$P_{ij} = (0.5)^{\max(\min(i,10),\min(j,10))}$$

i.e., the probability that the messages choose the same backoff time. Recalling assumption (3),

$$p_{1i} = (1-\rho_i)(1-e^{-2\lambda\Delta}) + \rho_i(1-e^{-\lambda(T+\Delta)})$$

where ρ_i is the probability that a message sees the network busy on its *i*th attempt ($2 \le i \le 16$). Now, if the preceding backoff time was large in comparison with message transmission time, ρ_i is well approximated by ρ. However, if the backoff time, *B* say, was small, the network would only be busy if another message arrived during that backoff time. Hence a better approximation would be $\rho_i = \lambda B$. We combine these approximations by defining

$$\rho_i = \sum_{j=0}^{2^i-1} 2^{-i} \operatorname{Min}[\lambda j T_{SLOT}, \rho]$$

The above equations for $\{p_i\}$ are solved iteratively with the starting values $p_i=\rho$ for all *i*. The iteration was always found to be convergent for $0<\rho<1$ in a wide range of tests.

The dropping probability – the probability that a message fails at the 16th attempt – is given by

$$p_{DROP} = \prod_{i=1}^{16} p_i$$

An additional definition is now needed: that of the *average residual wait time*, R_i, which is the average time between a message in backoff state *i* arriving at the network and the discovery of a collision, given that the message actually has a collision. This is given by

$$R_i = \Delta + \frac{T}{2} w_i$$

where w_i is the *waiting probability* for backoff state *i*, i.e. the probability that a message, on suffering a collision, had seen the network busy on its arrival. This is given by

$$w_1 = \frac{\rho\left(1 - e^{-\Lambda(T+\Delta)}\right)}{p_1}$$

$$w_i = \frac{\sum_{j=1}^{16} q_j \rho_i \left[P_{i-1,j} + (1 - P_{i-1,j})\left(1 - e^{-\Lambda(T+\Delta)}\right)\right]}{p_i} \qquad 2 \le i \le 16$$

The average time for a message to successfully transmit is then given by:

$$T_{TOTAL} = T + T_{IFG} + \rho T/2 + \sum_{i=1}^{16} \left(\frac{\left(\prod_{j=1}^{i-1} p_j\right)(1 - p_i)}{1 - p_{DROP}} \right) \left((i-1)(T_{IFG} + T_{JAM}) + \sum_{j=1}^{i-1} \left(R_j + \left(\frac{2^{\min(j,10)} - 1}{2} \right) T_{SLOT} \right) \right) \tag{2}$$

(2)

The term $\rho T/2$ arises because the successful transmission attempt may have to wait for the network to become idle before transmitting.

4.2.2 General Message Length Distribution

It is well documented that the shape of the message length distribution has a significant effect on the throughput of an Ethernet; see [1, 6] for example. Let the message propagation time random variable (not including any back offs, jams, etc.) be denoted by X. This quantity is directly related to the message length L via $X=L/v$, where v is the Ethernet's speed. Let the message propagation time probability density function be $p(t)$. Now, when observing the network, there is a greater chance of seeing longer messages because they spend a longer amount of time on the network. This means the probability density of *observed* message propagation times, $b(t)$ say, has to be weighted by the message propagation time itself; this is just the familiar argument used to find (non-rigorously) the observed renewal period probability density in renewal theory. Thus we find:

$$b(t) = \frac{p(t)tdt}{\int_{y=0}^{\infty} p(t)tdt} = \frac{1}{T} p(t)t$$

where $T = E[X]$ is the average propagation time. The probability of failure at the first attempt is now

$$p_1 = (1 - \rho)(1 - e^{-2\Lambda\Delta}) + \rho \int_0^{\infty} b(t)(1 - e^{-\Lambda(t+\Delta)})\, dt$$

and similarly p_i is given by the same formula with ρ replaced by ρ_i which is now defined by

$$\rho_i = \sum_{j=0}^{2^i - 1} 2^{-i} \operatorname{Min}[\lambda \int_0^{\infty} \operatorname{Min}[jT_{SLOT}, t]\, p(t)dt, \rho]$$

$$= \sum_{j=0}^{2^i - 1} 2^{-i} \operatorname{Min}\left[\lambda \left[\int_0^{jT_{SLOT}} tp(t)dt + jT_{SLOT} \int_{jT_{SLOT}}^{\infty} p(t)dt \right], \rho \right]$$

In practice, the integrals above will be bounded due to the minimum and maximum message lengths as defined in the Ethernet specification. The waiting probabilities w_i are now given by:

$$w_1 = \frac{\rho \int_0^{\infty} b(t)\left(1 - e^{-\Lambda(t+\Delta)}\right)dt}{p_1}$$

$$w_i = \frac{\sum_{j=1}^{16} q_j \rho_i \left[P_{i-1,j} + (1 - P_{i-1,j}) \int_0^{\infty} b(t)\left(1 - e^{-\Lambda(t+\Delta)}\right)dt \right]}{p_i} \qquad \text{for } 1<i<16$$

Now let T_{WAIT} denote the average time a message spends waiting before attempting to transmit, if on arrival it finds the network busy. Then

$$T_{WAIT} = \tfrac{1}{2}\int_0^{\infty} tb(t)dt = \int_0^{\infty} \frac{t^2}{2T} p(t)\,dt$$

(This is just the mean residual life in renewal theory.) The average residual wait time is now $R_i = \Delta + w_i T_{WAIT}$. These expressions can now be substituted into equation (2) for the average total transmission time in the previous section. As expected, if we take $p(t) = \delta(t-T)$, the equation is exactly the same.

Measurements of real Ethernets have shown that the most common message lengths are either minimum size or maximum size – typically the small messages are acknowledgements or requests, while the large messages are chunks of data. Thus a *bimodal* probability distribution is an appropriate model. Let α be the fraction of messages which are short; this quantity may be measured by any network 'sniffer'. Then the propagation time probability density function is equal to a weighted sum of two delta functions:

$$p(t) = \alpha\delta(t - T_{short}) + (1-\alpha)\delta(t - T_{long})$$

All the expressions above can be computed very simply using this. For example, the quantity T_{WAIT} is:

$$T_{WAIT} = \frac{1}{2T}\left(\alpha T_{short}^2 + (1-\alpha) T_{long}^2\right)$$

4.3 Numerical results

The graphs below show the total message transmission time as a function of Ethernet utilisation as predicted by the analytical model, in comparison with simulation results. Over 10000 message transmissions were simulated for each point on the graphs giving narrow confidence bands at the 95% confidence level. The simulation models the Ethernet's modus operandi exactly as described in section 4.1 and so its output results from an entirely different basis compared to that of the analytical predictions. Consequently, good agreement instils confidence in the models.

For constant message lengths, we obtained the following graphs.

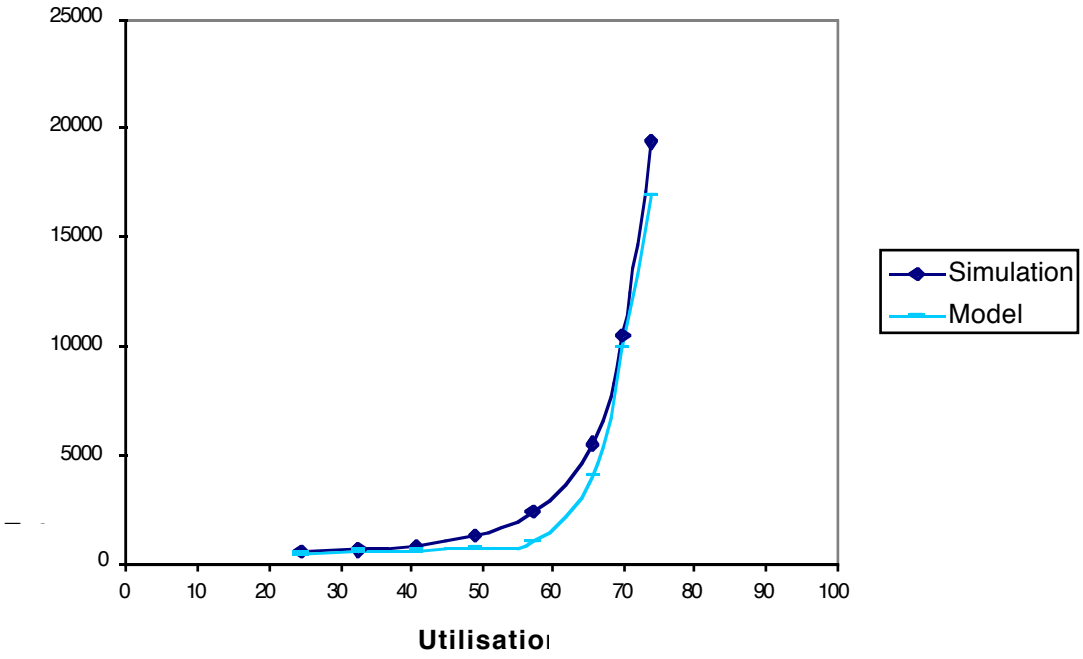

Figure 2. Transmission time versus utilisation

Generally, the analytical results compare favourably to the simulation results. In particular, the point at which performance starts to degrade seriously is clearly identified, at a utilisation of about 60%, which agrees closely with the simulation. This is the measure of most importance to capacity planners when designing their client-server system.

The next graph shows the dropping probabilities plotted against utilisation; recall that a message is dropped when it suffers 16 consecutive collisions. Again, it is clear that the network becomes saturated at utilisations of just over 60%. This is confirmed by simulation and is entirely consistent with the folk-lore surrounding the Ethernet.

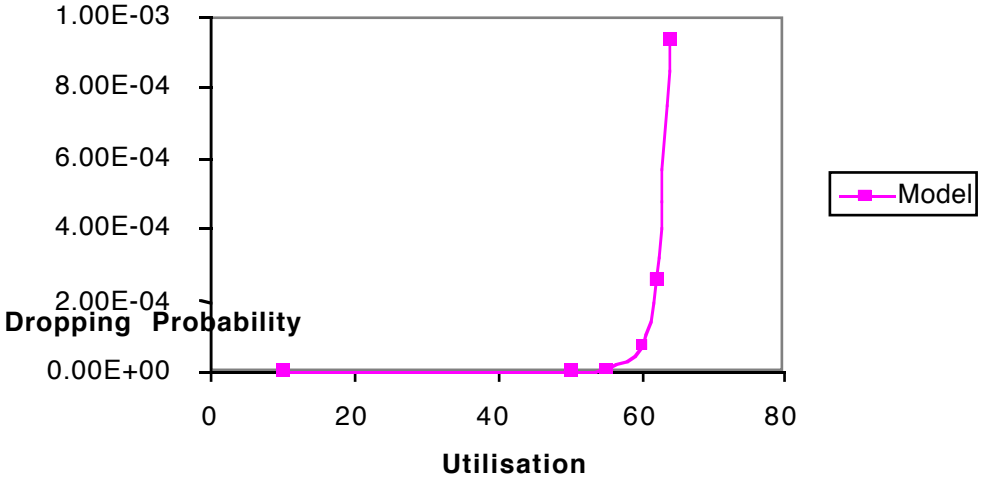

Figure 3. Dropping Probability vs Utilisation

The behaviour of the bimodal model parameterisation for T_{short} corresponding to 64 bytes and T_{long} corresponding to 1518 bytes and for various values of α is now

examined. We define the response overhead as the average time for a message transmission minus that message's propagation time. This is the overhead time due to collisions before successful transmission, and is therefore independent of the length of the message. The overhead time per message is plotted against utilisation for various α. In the case of 100% short messages, the analytical model agrees well with simulation at low utilisations, but the discrepancy increases nearer to the saturation point. The position of this point again appears to be accurately predicted, the network saturating at a lower utilisation between 25 and30%. This is due to an increased number of arrivals.

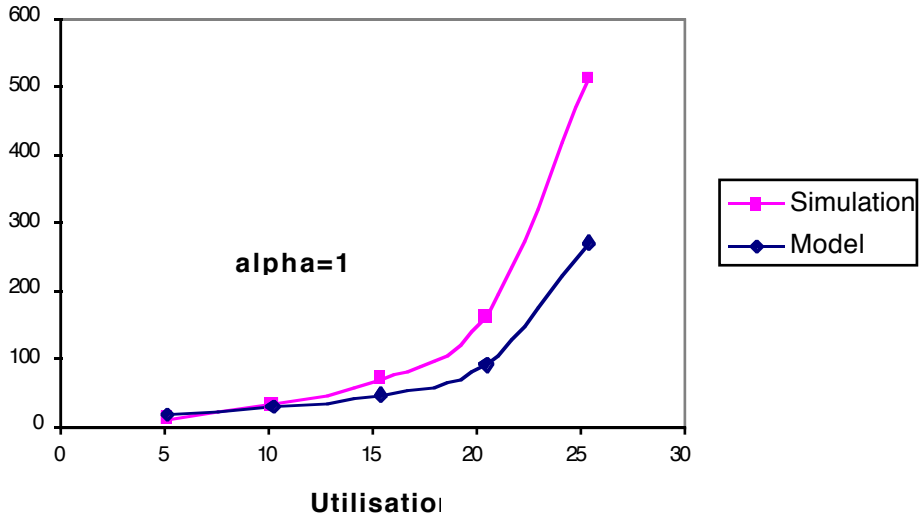

Figure 4. Response overhead with all short messages

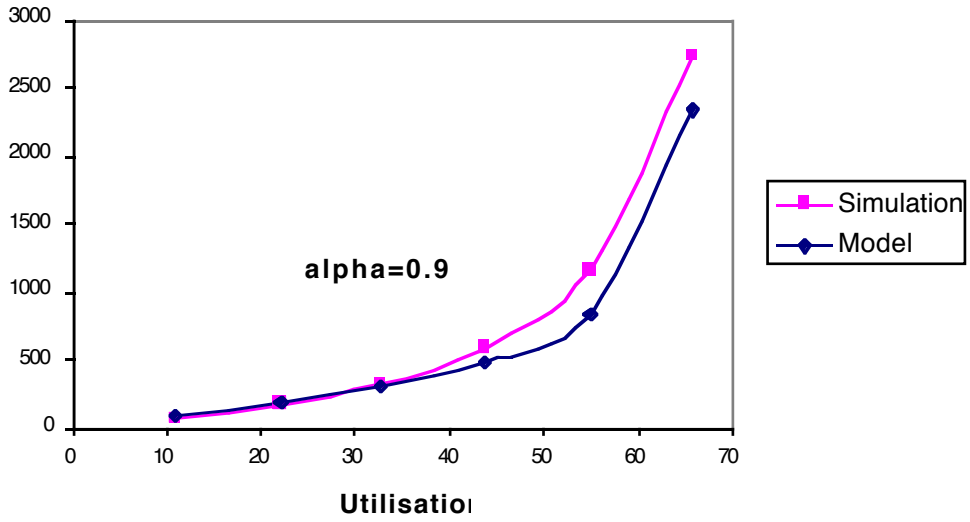

Figure 5. Response overhead with 90% short messages

With only a small proportion of long messages, the agreement between the analytical model and simulation is greatly improved and both models show that the Ethernet saturates at utilisation between 50 and 60%.

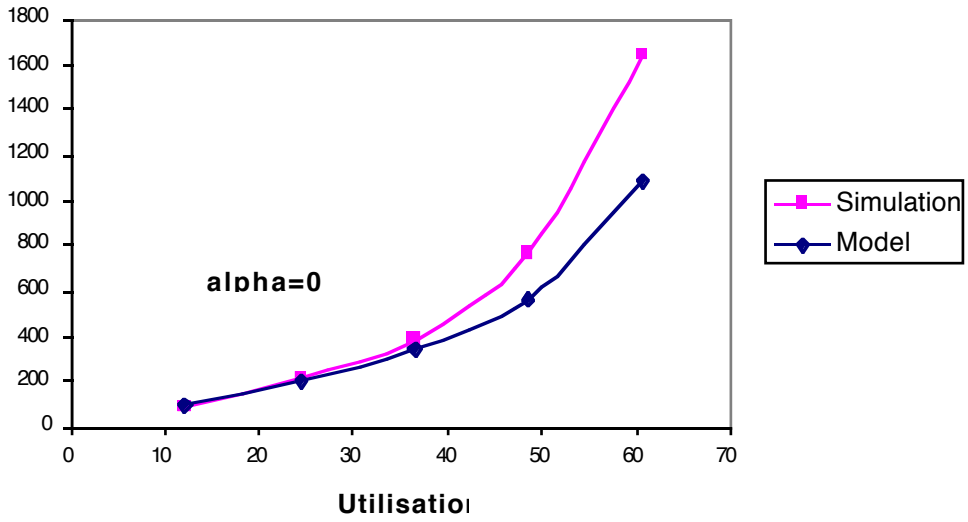

Figure 6. Response overhead with all long messages

The saturation point does not change very much as the proportion of long messages increases and with 100% long messages it is still between 50 and 60%. Note the larger scale in Figure 6. These graphs are very similar to those of Tobagi and Hunt [6].

4.4 Efficiency of the analytical model

The number of iterations needed by the model is a crucial parameter of the algorithm's performance. As expected the number of iterations necessary increases as the utilisation increases. However, to achieve results with a relative error of one part in a thousand, at most twelve iterations of the 16+1 equations for p_i must be performed. On a fast machine, this takes milliseconds. This should be compared with the simulation, which takes up to half an hour for highly utilised Ethernets. Simulation, whilst being slightly more accurate, is machine intensive: for typical capacity planning questions, the analytical model can provide very similar answers in a fraction of the time.

4.5 Parameterisation of an M/M/1 queue

We now wish to model the Ethernet as an M/M/1 queue in order to incorporate it into our response time model of section 3. Given the mean waiting time (total transmission time) at the Ethernet we have two choices:

- To consider the Ethernet as a queueing system where messages have to spend some time waiting before they transmit;
- To consider it as a shared resource where each message receives service continuously from the instant of its arrival, at a reducing rate as the number of messages there increases.

The operation of the Ethernet is closer to the first alternative, which is the one we choose.

If the service rate of the queue is μ, the mean waiting time is $1/(\mu-\lambda)$ where λ is as given in the Ethernet model. Hence we solve the Ethernet model for T_{TOTAL} and then match this with $1/(\mu-\lambda)$ to give $\mu = \lambda + 1/T_{TOTAL}$. This is the service rate we use in the response time model, which is now complete.

5 The Athene Client-Server tool

Athene Client-Server is a capacity planning tool from Metron Technology Ltd, which contains the models outlined in this paper. It contains a flexible topology editor to link clients, networks and servers. An example three-tier topology is shown in Figure 7. This model has ten NT clients communicating with an application server over an Ethernet. The application server calls a large back-end database. Data for modelling the clients and servers can be taken from NT and UNIX. Workloads are linked across machines over networks. Evaluation of the underlying queueing network is done near-instantaneously, on-line, as and when the parameters of the model are changed.

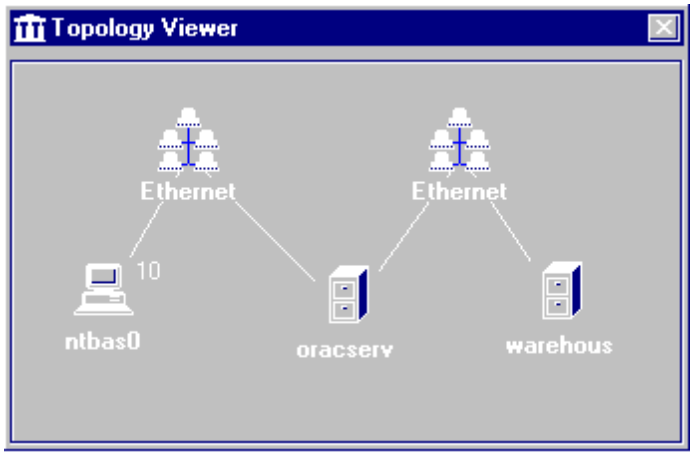

Figure 7. Network specification window

Response times of distributed workloads can be broken down into constituent parts, and even by device. This information is displayed as a bar chart in a new window, as shown in Figure 8 for example.

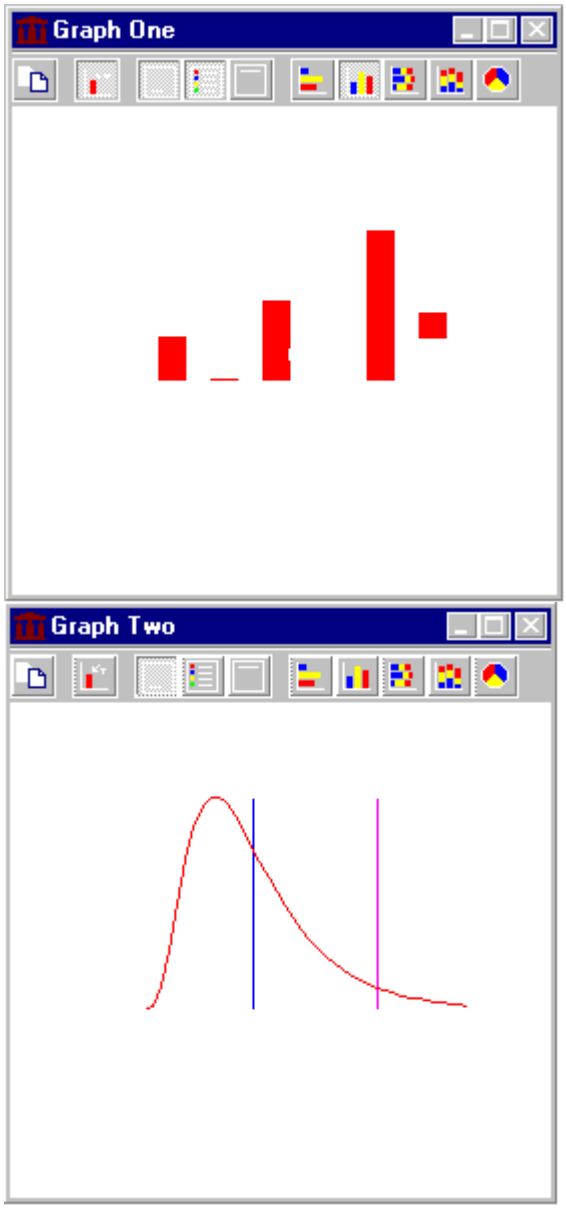

Figure 8. Constituents of response time and its probability density

The response time density calculation is implemented as described in section 3. The Ethernet contribution to response time is often negligible, as in this instance, whereupon its representation as an M/M/1 queue does not introduce a significant error. What is important is to be able to predict when the network will saturate, and we have seen that our model finds the corresponding level of utilisation accurately. The charts in Figure 8 change in real time as the user varies the parameters of the model, for example the number of clients, the rate at which transactions are generated by each client, Ethernet speed and the mean service time at the database.

'What-if' questions the model caters for include how the performance will change at increasing transaction rates, both on individual machines and across the enterprise, with greater numbers of clients, higher utilisation of Ethernets, etc. Long term forecasting, which can facilitate the maintenance of service levels, is also possible; a typical display is shown in Figure 9.

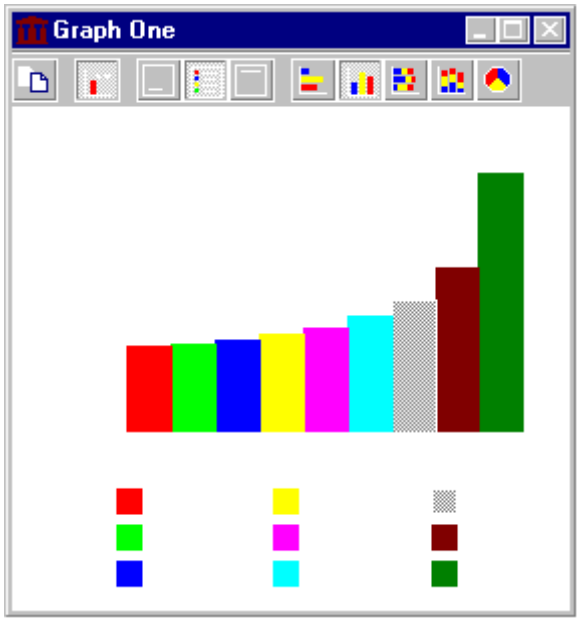

Figure 9. Long term forecasting

Here the growth in the usage of Oracle 80 over a period of eight months has been anticipated by installation management. It is required to find out whether system performance will remain adequate in terms of response time, which is characterised by its mean value. In this case we see after about six months (June) there is some cause for concern and an upgrade may be needed.

6 Conclusion

We have developed a simple but effective and efficient model for predicting mean response times and response time densities in client-server systems. We have also demonstrated its use by incorporating it into a capacity planning tool for client-server installation management. This software is in production and can provide a significant, live tool demonstration.

In terms of the mathematics involved, we have found a new approximate result for response time density in central server networks that holds under appropriate independence assumptions. Although these assumptions do not hold in practice, we have shown that they do asymptotically as network size increases and should not lead to undue inaccuracy for moderately sized networks. In addition, we have derived a new model of the Ethernet, which is of interest in its own right and produces accurate results at low utilisations and predicts well the maximum utilisation possible before performance degrades sharply. We used this model to parameterise the Ethernet as a single server Markovian queue in order to incorporate it into the above response time model. Again, at low utilisation, this does not introduce a significant error.

The main limitation of the model is its reliance on exponential service times. We believe that the Ethernet sub-model will generalise straightforwardly to approximate response time densities for arbitrary message length distributions. The same cannot be said for the response times in central server systems. However, we do have the property that the numbers of visits made to each node in these queueing networks are (asymptotically) geometric and independent. Hence an exponential form for the time spent by a task at each node is not an unreasonable approximation. Finally, we note that in the central server model, we assumed that any given task makes an independent choice of which device to visit next every time it leaves the CPU. In reality, it will often be that each task always visits the *same* device on every cycle, the average number of visits (by *all* tasks) to each device being the same by virtue of the distribution of tasks in

terms of which device they do choose. In fact, this problem is rather simpler to solve than ours and is an obvious extension of the model.

7 Acknowledgements

The authors would like to thank Mike Garth and Adam Grummitt for helpful comments on the manuscript. The work was supported by the Department of Trade and Industry's SPUR initiative.

8 References

[1] Boggs, Mogul & Kent, 'Measured Capacity of an Ethernet: Myths and Reality', WRL Research Report 88/4, (1988)

[2] Harrison P G, Patel N M; "Performance Modelling of Communication Networks and Computer Architectures", Addison-Wesley, 1993.

[3] Kleinrock, 'Queueing Systems, Volume Two: Computer Applications', Wiley, (1976)

[4] Metcalfe & Boggs, 'Ethernet: Distributed Packet Switching for Local Computer Networks', Communications of the CSM, 19, 5, 395, (1976)

[5] Metron; "Athene / Client-Server User Guide", Metron Technology Limited, Taunton, UK, 1995.

[6] Tobagi & Hunt, 'Performance Analysis of Carrier Sense Multiple Access with Collision Detection', Computer Networks, 4, 245, (1980)