Modelling Network Memory Servers with Parallel Processors, Break-downs and Repairs

O Gemikonakli, G Mapp, E Ever, and D Thakker
Middlesex University, London UK
o.gemikonakli, g.mapp, e.ever, d.thakker@mdx.ac.uk

Abstract

This paper presents an analytical method for the performability evaluation of a previously reported network memory server attached to a local area network. To increase the performance and availability of the proposed system, an additional server is added to the system. Such systems are prone to failures. With this in mind, a mathematical model has been developed to analyse the performability of the proposed system with break-downs and repairs. Mean queue lengths and the probability of job losses for the LAN feeding the Network Memory Server is calculated and presented.

1. Introduction

Increases in computing power and network speeds have enabled the development of new types of network services. A **Network Memory Server** (NMS) is one such service which is being developed here at Middlesex University [10]. The motivation for this effort is the observation that the cost of Dynamic Random Access Memory (DRAM) has been continuously falling over the last few years so building very large memory servers is no longer prohibitively expensive. In addition, increasing network speeds now make it possible to access data from the memory of another machine on a High-Speed Local Area Network or HSLAN faster than accessing that same data on the local hard disk.

These observations make it possible to build a new global storage model based on specialised servers. Though efforts to build such a system over wired networks have been reported [1] [2], a new application-space for this type of service is now being explored because in the near future, these systems will be needed to provide high-performance storage facilities for mobile devices such as cell-phones, PDAs and Thin Clients which tend to have minimal storage capacity. So at Middlesex University we are developing network memory servers as part of an overall architecture to provide central storage facilities for mobile environments [10].

The rise of wireless networking has been rapid and sustained. Bandwidth has also been increasing as well as the area of coverage which is set to continue with the deployment of WiMax. However, the deployment of network memory services in such an environment requires careful planning and analysis as the success of the proposed system will depend heavily on the characteristics of the network on which it is deployed. Hence, it is important to evaluate the performance of the proposed system with a view to investigating how the system would perform over a wide range of network and server characteristics. In particular, issues of system stability and optimum performance must be addressed to ensure the development of a successful design.

The performability of such a system has previously been reported [7]. However, in [7], a single NMS with no back-up, and no break-downs was considered. This was due to the state space explosion problem inherent to most analytical solutions to two-dimensional Markov processes. In this paper, a new solution for a mathematical model based on a three-dimensional Markov process used to evaluate the performance of two-processor network memory servers with break-downs, and repairs is presented. The model allows detailed analyses to be performed on the interaction between the server and the network. In particular, results are generated when the throughput of the network relative to the server is high as would be the case in Gigabit networks, and also when the throughput of the network is similar to that of the server. This condition is likely to be true in wireless networks which offer lower bandwidth and increased latency compared to wired systems.

Secondly, the model is also attempting to look at the effect of other LAN traffic on the performance of the Network Memory Server. In both wired and wireless cases, we are primarily interested in systems based on Ethernet technology. Hence requests from different clients form a distributed network queue which can be modelled as a large capacity queue with exponential service times.

Another key issue in the composition of network memory servers is that of client-caching. Since it will always be relatively cheaper to access local memory rather than going over a network – no matter how fast the network - client caching appears sensible. However, in environments such as mobile phones and PDAs, where memory is at a premium, selective caching strategies may be required. The proposed model helps to address this issue.

The effects of back-up, break-downs and repairs are also demonstrated.

2. The Network Memory Server

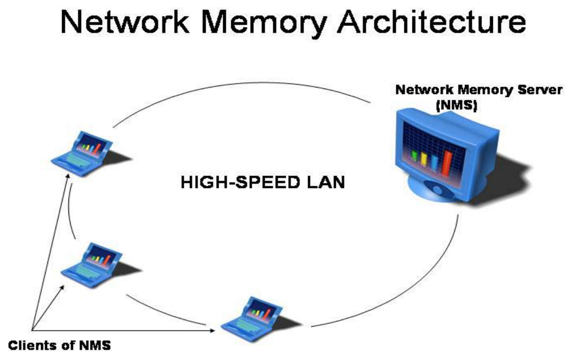

Figure 1. Network memory architecture for memory server

The Network Memory Architecture being deployed is shown in Figure 1. The main principle used in the design of the NMS was to keep the design as simple as possible because any unnecessary complicity would make this networked system difficult to debug as shown in similar architectures [9]. This led to several important characteristics of the system which can be stated as follows: Firstly the NMS is transparent to end-users of the machines in the system. Secondly, the NMS is stateless. It therefore treats requests independently and keeps no record of previous interactions with its clients. Thirdly the NMS deals only with blocks of data. It assumes nothing about how data is referenced and accessed by the relevant users of that data. It is therefore more basic than remote file systems such as NFS. However, this simplicity means that the NMS can be used by many applications including remote paging [5], storage for database applications [1] and better access models for datasets that do not fit well into a file-structured format such as multimedia data, etc.

The clients of the NMS are machines and applications that would like to make use of Network Storage. The NMS receives requests to create, read, write or delete blocks of data of various sizes. Each client is identified by a unique **clientid**. Each block of data is represented by a **blockid**, which is generated when the block of memory is assigned by the NMS. A **blockid** is a 64-bit number comprising a 48-bit index number and a 16-bit security tag. This tag allows the NMS to detect a modified or invalid blockid.

Requests from clients are therefore sent to the NMS using TCP/IP as it provides reliable communication between processes on different machines. Each client machine establishes TCP connections to the NMS when the driver is loaded. Thus the requests to the server along these connections form a distributed input queue. The system is a client/server interaction where the client waits for a reply from the server before proceeding. In addition, the server uses the socket abstraction to handle each connection. Each socket has a finite buffer associated with it. When this buffer is full, no more requests can be sent to the server (which can be modelled as the network no longer serving requests to the NMS) until those already in the queue have been processed. The TCP windowing mechanism prevents packets from being thrown away when the buffer is full. This means that it is safe to model the system as a finite queue whose exact length will vary according to size of the buffers allocated to sockets.

The design uses a multi-threaded approach where each connection, which represents a client, is managed by a thread. All requests to the server from that client are handled by that thread in a sequential fashion. Since we are assuming a single processor system, threads will have to be scheduled before they can service clients' requests. So in effect we have a distributed contention system which can be modelled as a single server with exponential service times.

The system has been engineered such that each client can send a packet containing multiple requests to the NMS. So for analytic purposes, it is possible to treat such a packet as one job – though made up of many small ones. It is worth pointing out that as far as the NMS is concerned, reading and writing involves the same basic operation namely: taking a request and finding the corresponding block. If the operation is to write then data is written from the incoming network buffer to that block in memory. If a read operation is requested then the data from the block is copied to an outgoing network buffer.

However, from a network traffic point of view, the situation is also interesting but different. In terms of volume, most of the traffic going to the server will consist of requests to write blocks. This is because the request headers of the NMS are quite small compared to the data being transferred, so write requests will cause a lot of data to be transferred from the client machines to the NMS. In the opposite direction, most of the traffic moving from the NMS to client machines will be due to read requests. This can be modelled as a feedback loop going from the NMS Server back to its clients.

Network memory servers hold data that would normally be held on hard disk such as the operating system of a machine, hence it is necessary to consider backup and repair provision for such environments. In addition, since the NMS is stateless and deals only in blocks using client calls, the use of parallel servers should be extremely desirable.

In order to facilitate the parallel operation of this service with backup and repair capability, it is necessary to enhance the memory architecture. A global memory storage unit needs to be added which is accessible by all the NMS servers using shared memory techniques. The global storage unit contains blocks of data as well as the supporting data structures including a hash table which stores the block management structure (bms) for a given block.

When a block needs to be created, a unique blockid is obtained, the memory for the block is acquired from the global storage unit and the bms structure is created and placed in the hash table. Locking is provided using distributed locking techniques.

3. The System, The Model, And The Solution

The system under consideration consists of a LAN, a number of clients, two network memory servers (i.e. main and back-up servers) and various other servers (e.g. file server, database server, e-mail server etc.). The servers both store data coming from clients to have duplicate copies for improved availability. The memory servers serve read requests when operative. All the devices attached to the LAN share a transmission medium and compete for LAN bandwidth using some form of access method (e.g. carrier-sense multiple access with collision detection). The traffic generated can be destined for any of the hosts attached to the LAN.

The LAN and the memory server can be thought as two tandem queues each serving arriving jobs. A model representing the NMSs together with the underlying network is shown in Figure 2. Here, L_j is much larger than L_i. L_i represents the queuing capacity of the proposed NMSs. Hence, this is a blocking network. The LAN and the memory server operate at mean service rates of μ_1 and μ_2 respectively. The mean service times $1/\mu_1$ and $1/\mu_2$ are assumed to follow exponential distributions as discussed above. The LAN serves the NMSs at a mean rate of $\mu_1(1-\theta_1)$, where $0 \leq \theta_1 \leq 1$, and the remaining traffic is directed to other hosts attached to the LAN. Each NMS writes data at a mean rate of $\mu_2(1-\theta_2)$ and data is read on request and fed back to the LAN at a mean rate of $k\mu_2\theta_2$, where $0 < \theta_2 < 1$ and k is the number of operative NMSs. σ_1 is the mean arrival rate of jobs generated by sources other than the NMSs (which may be destined for the NMS), creating traffic on the LAN.

The LAN is assumed to be highly reliable. The memory servers are prone to failures. ξ and η are the failure and break-down rates respectively.

When the common queue for the NMSs is full, or both servers are broken, the LAN stops serving this queue, and resumes service when spaces are free in the queue and at least one of the servers is ready to accept more jobs. This network is an extended version of the tandem network studied by Konheim and Reiser [8] with the addition of a second server running in parallel, break-downs and repairs.

3.1. Three Dimensional Markov Representation of the Proposed Model

Figure 2 shows the system under consideration. In this system, L_j and L_i represent the queuing capacities of the first and second stages respectively. L_j and L_i are both finite and L_j can be a lot larger than L_i. Jobs arrive at stage 1 at a mean arrival rate of σ_1, following a Poisson distribution. The mean service rates are μ_1 and μ_2 per server for stages one and two respectively.

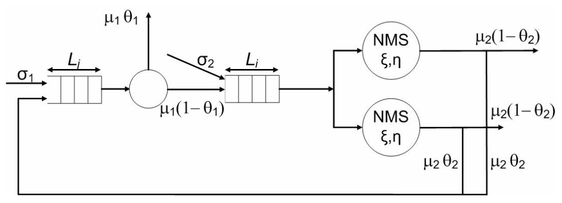

Figure 2. The system considered

The first stage is a LAN modelled as a single server with a queuing capacity L_j. Network memory servers of the second stage are homogeneous with mean breakdown rate ξ, and mean repair rate η. The first stage is assumed to be highly reliable. θ_1 and θ_2 represent the fraction of serviced jobs leaving the system after stage 1 and fed back to first stage respectively. The total number of servers at stage 2 is two, k is the number of the operative servers and i represents the number of jobs, both at stage two at time t. When both servers at stage 2 are broken, or the queue is full, stage 1 stops sending jobs to stage 2.

The proposed method models the system shown in Figure 2 as a three-dimensional Markov process and solves the resultant model using a spectral expansion solution together with an iterative process. Then, the steady state probabilities of the three-dimensional Markov chain are calculated. Consider a discrete time, two-dimensional Markov process on a finite or semi-infinite lattice strip. The Markov Process can be defined as $X=\{I_n, J_n; n=0, 1, \ldots\}$ with a state space of $(\{0, 1, 2, \ldots, L_j\} \text{x} \{0, 1, 2, \ldots, L_i\})$. Then, $i=0, 1, 2, \ldots, L_j$, and $j=0, 1, 2, \ldots, L_i$ can be used to represent all possible states, (i,j) on the lattice strip. This system can be solved using existing approaches where most popular ones are explained in [3] and [4]. The limiting factor here is the size of I_n, i.e. L_i. When more than one homogeneous servers with breakdowns are considered for stage 2, the size of I_n becomes $(K+1)L_i$, where K is the total number of servers. Clearly, when $K>1$, only small queuing capacities for stage 2 can be assumed to avoid the state space explosion problem. In practice however, larger queuing capacities exist. By modeling such networks using three dimensional processes, I_n becomes independent of K. The Markov process can now be defined as $Y=\{I_n, J_n, P_n; n=0, 1, \ldots\}$ with a state space of $(\{0, 1, 2, \ldots, L_i\} \text{x} \{0, 1, 2, \ldots, L_j\} \text{x} \{0, 1, 2, \ldots, K\})$. This can be considered as $(K+1)$ two-dimensional processes, X, let's say X_k, where $k = 0, 1, 2$. Each X_k can be considered an independent two-dimensional Markov process. Defining the sum of the steady state probabilities of these two-dimensional processes lead to the following sums:

$$S_k = \sum_{i=0}^{L_i} \sum_{j=0}^{L_j} p_{i,j,k} \text{ and } \sum_{k=0}^{2} S_k = 1 \text{ , } k=0, 1, 2 \tag{1}$$

It is important to calculate S_k for all k. For the system considered on each lattice representing X_k, while one-step downward transition rates are a function of μ_1, one step upward transition rates are a function of $\mu_2\theta_2$ and σ_1. Lateral transitions are determined by $\mu_1(1-\theta_1)$, σ_2 and μ_2. Transitions between X_k,s occur with rates ξ, and η, and each S_k can be expressed as follows:

$$S_k = \left[k! \sum_{l=0}^{2} \left(\frac{(\eta/\xi)^{l-k}}{l!} \right) \right]^{-1}, k = 0,1,2. \tag{2}$$

Let's define matrices $\mathbf{u}_k$s for all i, j and k, as $\mathbf{u}_k = \lfloor p_{i,j,k} \rfloor$ Then, for $0<j<L_j$, $0<i<L_i$, $k=1$ and $\sigma_2=0$, $p_{i,j,1}$s can be calculated as

$$\begin{aligned} p_{i,j,1} &= [2\xi p_{i,j,2} + \eta p_{i,j,0} \\ &+ \sigma_1 p_{i,j-1,1} + \mu_2\theta_2 p_{i+1,j-1,1} \\ &+ \mu_2(1-\theta_2) p_{i+1,j,1} \\ &+ \mu_1\theta_1 p_{i,j+1,1} + \mu_1(1-\theta_1) p_{i-1,j+1,1}] \\ &/[\sigma_1 + \mu_1 + \mu_2 + \xi + \eta] \end{aligned} \tag{3}$$

For each $\mathbf{u}_k$, $0\leq k\leq 2$, the transition matrices are defined in terms of σ_1, μ_1, μ_2, θ_1, and θ_2. From this, the approximate $p_{i,j,k}$s can be obtained using a spectral expansion solution [3] for each k. Since the steady state probabilities for lattice k are initially calculated independent of lattices k-1 and k+1 (as appropriate), it is important to use a technique to compensate for the unaccounted effects of the latter two lattices on lattice k. This can be achieved through the use of the balance equation in (3) and the ones derived from that. If $t_{x,,y}$ represents transitions from lattice x to lattice y where $x \neq y$ and x, y = 0, 1, 2, then, the transitions can be summarized as follows:

$$\begin{tikzcd}\left[\mathbf{u}_0\right] \arrow[r, shift left, "t_{0,1}"] & \left[\mathbf{u}_1\right] \arrow[l, shift left, "t_{1,0}"] \arrow[r, shift left, "t_{1,2}"] & \left[\mathbf{u}_2\right]. \arrow[l, shift left, "t_{2,1}"]\end{tikzcd}$$

Since (3) is for $0<j<L_j$, $0<i<L_i$, $k=1$ only, it is important to obtain balance equations for $k=1$ with $j=0$ and L_j, $i=0$ and L_i, as well as for all possible combinations of $k=0$ & 2 for all i and j. These can all be deduced from (3). All approximate steady state probabilities, $\mathbf{u}_k$, can be calculated for k operative servers, where k = 0, 1, 2 as long as there is a spectral expansion solution. These probabilities are approximate because lateral transitions have not been taken into account. Once the approximate steady state probabilities are calculated, the balance equation for $0<j<L_j$, $0<i<L_i$, $0<k<2$, given in (3) can be used together with all other balance equations derived from this to calculate the steady state probabilities more accurately. Then, an iterative procedure can be followed to accurately calculate $p_{i,j,k}$s. The procedure can be given as follows:

1. $\mathbf{u}_k$s are calculated for k = 0, 1, 2 using a spectral expansion solution together with (2).
2. The balance equation given in (3) together with the derived ones are used to calculate the correct steady state probabilities.
3. Mean queue length is calculated for the queuing system considered.
4. Steps 2 and 3 are repeated until the mean queue length converges sufficiently.

Once the correct state probabilities are obtained, various performability measures can be calculated.

The state of the system at time t can be described by a pair of integer valued random variables, $I(t)$ and $J(t)$ specifying the number of jobs present at time t for the NMSs and the LAN respectively. Hence, $I(t)$ = 0, 1, …, L_i and $J(t)$ = 0, 1, … , L_j.

Here, lateral transitions of process X can be defined by matrix A showing instantaneous transition rates from state i to state l with zeros on the main diagonal. Let's define matrices B and C as transition matrices for one-step upward and one-step downward transitions respectively. When the transition rate matrices depend on j for $j \geq M$, where M is a threshold having an integer value, the process X evolves with the following instantaneous transitions:

A_j: Purely lateral transition rate, from state (i, j) to state (l, j), (i=0,1,…, L_i, l=0,1,…, L_j; $i \neq l$; j=0,1, … L_j), caused by a change in the state of the queue length of the NMSs (i.e. a job served by the NMSs and leaves the system (writing is complete)).

B_j: One-step upward transition rate, from state (i, j) to state $(l, j+1)$, (i=0,1,…, L_j, l=0,1,…, L_j, and j=0,1,… L_j), caused by a job arrival into the queue which may also be a job transfer from the main/back-up NMS to the LAN (i.e. reading from the NMS).

C_j: One-step downward transition rate, from state (i, j) to state $(l, j-1)$, (i=0,1,…, L_j, k=0,1,…, L_j, and j=0,1,… L_j), caused by the departure of a serviced job from the LAN.

Therefore, the transition matrices A, A_j, B, B_j, C, and C_j, for this model can be expressed as follows:

$$A = \begin{bmatrix} 0 & \sigma_2 & . & . & 0 \\ \min(i,k)\mu_2(1-\theta_2) & 0 & . & . & 0 \\ 0 & \min(i,k)\mu_2(1-\theta_2) & . & . & . \\ . & . & . & . & \sigma_2 \\ 0 & 0 & . & \min(i,k)\mu_2(1-\theta_2) & 0 \end{bmatrix}$$

where i = 1,2,…, L_i and j=i-1 and A=A_j.

$$B = B_j = \begin{bmatrix} \sigma_1 & 0 & . & . & 0 \\ \min(i,k)\mu_2\theta_2 & \sigma_1 & . & . & 0 \\ 0 & \min(i,k)\mu_2\theta_2 & . & . & . \\ . & . & . & \sigma_1 & . \\ 0 & 0 & . & \min(i,k)\mu_2\theta_2 & \sigma_1 \end{bmatrix}$$

and

$$C = C_j = \begin{bmatrix} \mu_1\theta_1 & \mu_1(1-\theta_1) & . & . & 0 \\ 0 & \mu_1\theta_1 & . & . & 0 \\ 0 & 0 & . & . & . \\ . & . & . & . & \mu_1(1-\theta_1) \\ 0 & 0 & . & 0 & \mu_1\theta_1 \end{bmatrix}$$

with $C_0 = 0$, and threshold M=1.

4. Numerical Results And Discussions

We used the model and the solution presented in the previous section to show the validity of the model and evaluate the performance of the proposed system for specific cases. First, we assumed that the service rates of the LAN and the NMSs are equal (i.e. μ_1=μ_2=2), and then we obtained results for the case where μ_1>μ_2 for a specific case (i.e. μ_1=5μ_2). Break down and repair rates have been taken as ξ=0.001 and η=0.5 respectively.

Since the LAN serves other hosts as well as the NMS, θ_1 becomes an important parameter too. We assumed a finite set of values for θ_1. θ_2 shows the share of read and write operations carried out by the NMS. It should be noted that, a *read* operation may result in further *write* operation (e.g. reading a record, editing and re-writing). Hence, various values have been considered for θ_2 as well. Figures 3-6 show the mean queue length (MQL) of the first stage (i.e. the LAN) as a function of σ_1, and for various μ_1, μ_2, θ_1, θ_2, and L_i values. L_j=1000 ξ=0.001, η=0.5 is used for all calculations.

Legend for Figures 3-8 is as follows: ____: θ_1=0, _ _: θ_1=0.75, --: θ_1=0.25, +: θ_2=0.75, *: θ_2=0.5, and ○: θ_2=0.25:

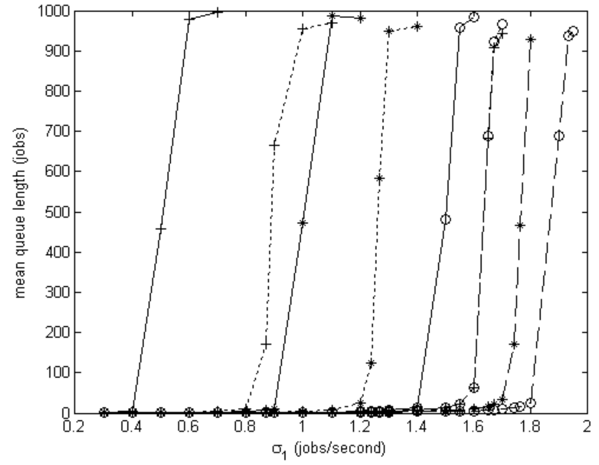

Figure 3. MQL for the LAN, for μ_1=μ_2 = 2, and L_i=10

In Figures 3 and 4, μ_1=μ_2=2, and μ_1=10, μ_2=2 respectively. L_j=10 for both cases. MQLs are shown for various combinations of θ_1 and θ_2. Results clearly show that the MQL closely relates to the combination of θ_1 and θ_2 values and not just one of these parameters. θ_1=0.75 (25% of jobs processed by the LAN are transferred to the NMSs) and θ_2=0.25 (25% of NMS activities are for reading) gives the best MQL performance for both μ_1=μ_2=2 and μ_1=10, μ_2=2. In the latter case, for most combinations of θ_1 and θ_2 the MQL performance changes so abruptly that it was not possible to get meaningful results for relatively large mean arrival rates.

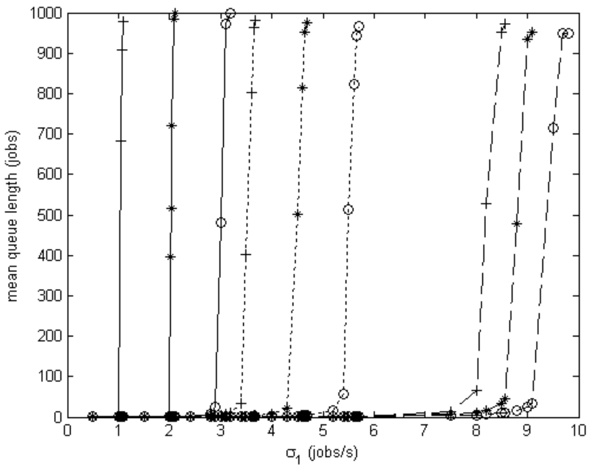

Figure 4. MQL for the LAN, for μ_1=10, μ_2 = 2, and L_i=10

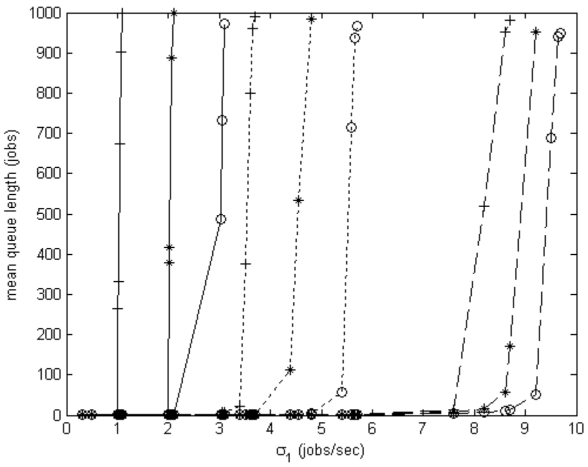

Figure 5. MQL for the LAN, for μ_1=10, μ_2 = 2, and L_i=40

For Figure 5 the only difference compared to Figure 4 is the queuing capacity of the NMSs. Here, L_2=40 is assumed. Best performances are achieved for θ_1=0.75, θ_2=0.25 as before. Again, in the latter case, for most combinations of θ_1 and θ_2 the MQL performance changes abruptly. Figure 6 compares the MQL performances of single-and-two-server systems for θ_2=0.75 and various θ_1 values. For a loaded network, for large θ_1, the presence of a back-up server becomes less significant.

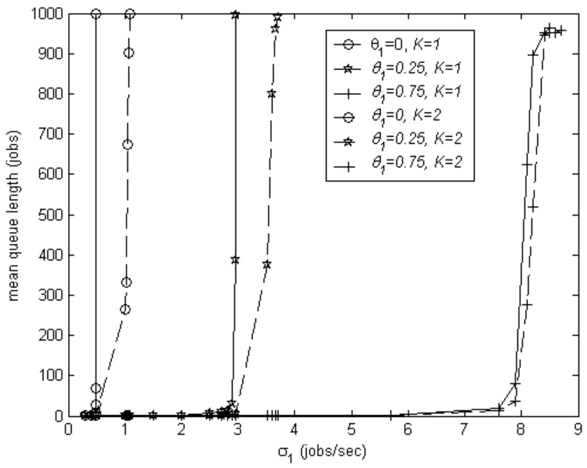

Figure 6. MQL for the LAN, for K=1 & 2, μ_1=10, μ_2 = 2, θ_2=0.75, and L_i=40

Finally, we computed the probability that the jobs are lost at first stage when μ_1=10, μ_2 = 2, various θ_1 values, and θ_2=0.75. Results are presented in Figure 7. As shown on the diagram, for this performance measure, for large θ_1, a back-up server does not have a significant contribution.

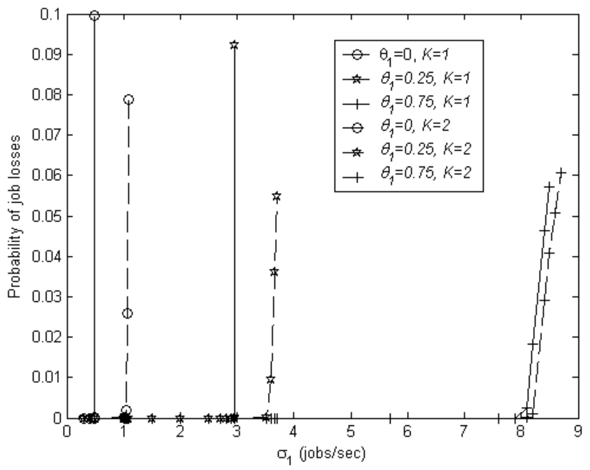

Figure 7. The probability of job losses for the LAN

5. Practical Implications From The Results

It is worth pointing out that servers such as the NMS tend to be latency sensitive. By this we mean that the clients of the NMS will react badly to large delays in the system. This is because in most cases NMS clients will be blocked, unable to proceed unless the requested block is retrieved. In this regard, the mean queue lengths are of particular interest. All the results suggest that system performance is affected by high values of θ_1, (i.e. the network memory server is competing with other traffic for the network hence it is underutilised) while high values of θ_2 (there is a lot more reading than writing). The high θ_1 values can be overcome by network provisioning in wired environments. The high θ_2 values mean that there must be an attempt to minimize excessive reading from the NMS. This would suggest that there is a need to look at intelligent cache strategies for read operations at the client end [11].

The results were less severe when the relative speed of the network compared to the server is large (μ_1=10, μ_2=2). However, when they are similar (μ_1=μ_2=2), the MQLs are much longer. Such an observation has clear implications in trying to support wireless systems which tend to have lower bandwidths. The results show that intelligent client caching as well as network features such as QoS support will be needed to provide good storage support for mobile environments [12]. Queuing capacity of the NMS plays a critical role in system performance. So much that, for loaded networks, having a back-up server can easily loose its significance.

The results clearly highlight the fact that parallel operation with backup and repair capabilities does improve the general performance of the system. This was observed more prominently for lower values of θ_1 and emphasizes the need to ensure that θ_1 remains relatively small in such environments.

6. Conclusions

In this paper an analytical model for a local area network with a main and a back-up network memory server alongside various other servers is presented. A three-dimensional Markov model incorporating a spectral expansion approach is used for the approximate solution of the proposed model for evaluating performance measures such as mean queue length and probability that jobs are lost. Results are presented and discussed.

In terms of service rates, two cases are considered. The first case assumes equal service rates for both the underlying network and the NMS. This better reflects the case of wireless systems where, compared to wired systems, the bandwidth is scarce. The second case considers an underlying network which is much faster than the NMSs in serving jobs; a possible case where fast LANs can be employed. Various queuing capacities have also been considered. The results indicate the importance of the need for intelligent client caching as well as network features such as QoS support to provide good storage support for mobile environments. For wired networks, although "throwing in extra bandwidth" will have a significant effect on system performance, parameters such as θ_1 and θ_2 also need to be monitored. The contribution of a back-up server will be higher if the NMS queuing capacity is increased. Analytical solutions suffer from state space explosion problem and hence, for most solutions, L_j can not be increased significantly.

The proposed NMS is prone to failures, just like any other server. The model assumes break-downs and repairs of the NMSs considerably extending the method presented in [3] and [7] and using this together with a three dimensional model. Without a third dimension, L_j will be further restricted. The third dimension increases the state space by 300% (or K+1 times for a K-server system). The system can further be developed to incorporate rebooting and reconfiguration delays and performance measures can be computed following the approach of [3], [7], [8], and [13] together with the model proposed in this paper.

7. References

[1] Breuer P. *The Enhanced Network Block Device Linux Kernel Module,* December 2003, http://www.it.u3m.es/ptb/nbd

[2] Flouris M. and E. Markatos "The Network Ramdisk: Using Remote Memory on Heterogeneous NOWs". In *Cluster Computing, 2(4):* pp. *281-293, 1999.*

[3] Chakka, R., "Spectral expansion solution for some finite capacity queues". In *Annals of Operations Research* 79,1998, pp.27-44.

[4] Ciardo G. and E. Smirni, "ETAQA: An Efficient Technique for the Analysis of QBD-processes by Aggregation,". In *Performance Evaluation*, 1999,36-37: pp.71-93,.

[5] Dwarkadas S., N. Hardevellas, L. Kontothanassis, R. Nikhil and R. Stets, "cashmere-VLM Remote Memory Paging for Software Distributed Shared Memory". In *Proceedings, 13th International Parallel Processing Symposium and 10th Symposium on Parallel and Distributed Processing*, IEEE Computer Society Press, April 1999, pp. 153-159.

[6] Gemikonakli O., T. V. Do., R. Chakka, and E. Ever, (2005), "Numerical Solution to the Performability of a Multiprocessor System with Reconfiguration and Rebooting Delays", In *Proceedings of ECMS 2005*, Riga, Latvia, June 2005, pp. 766-773

[7] Gemikonakli O, G. Mapp, D. Thakker and E. Ever "Modelling and Performability Analysis of Network Memory Servers". In 39th Annual Simulation Symposium, Huntsville, Alabama, USA, 2nd - 6th April 2006, pp.127-134.

[8] Konheim, A.G., and M. Reiser, , "A queueing model with finite waiting room and blocking", *Journal of the ACM*, 1976, pp.328-341.

[9] Kubiatowicz J, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W.Weimar, C. Wells and B. Zhao *OceanStore: An Architecture for Global-Scale Persistent Storage*. In *the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems* (ASPLOS 2000) November 2000.

[10] Mapp G.E, D. Silcott and D. Thakker, "Network Memory Servers: An idea whose time has come". *Multi-Service Networks (MSN) Cosener's House*, Abingdon, July 2004.

[11] Thakker D., "Intelligent client caching for Network Memory Servers". PhD proposal. School of Computing Science, Middlesex University.

[12] Shaikh F., *Quality of Service Issues in Wireless Applications.* PhD proposal. School of Computing Science, Middlesex University.

[13] Trivedi, K. S., A.S.Sathaye, , O.C. Ibe, and R.C. Howe, (1990). "Should I add a processor?" In *Proceedings of 23rd Annual Hawaii International Conference on System Sciences*, IEEE Computer Society Press, (Los Alamos, CA), pp.214-221.