View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Washington University St. Louis: Open Scholarship

Washington University in St. Louis

Washington University Open Scholarship

All Computer Science and Engineering Research

Computer Science and Engineering

Report Number: WUCS-93-42

1993

A Proposed Bus Arbitration Scheme for Multimedia Workstations

Saied Hosseini Khayat and Andreas D. Bovopoulos

The integration of video and audio into computers requires the support of continuous streams at the hardware level. This paper proposes a bus bandwidth management and access arbitration scheme for a multimedia workstation. It is assumed that a multimedia workstation consists of several specialized processing modules which are linked by a packet-switched bus. Using the proposed scheme, the bus can support a mix of real-time continuous media streams and random transactions while fulfilling special requirements corresponding to each traffic type. Specifically, the bus provides very fast response to random transactions and serves continuous media streams in such a way... **Read complete abstract on page 2.**

Follow this and additional works at: https://openscholarship.wustl.edu/cse_research

Recommended Citation

Khayat, Saied Hosseini and Bovopoulos, Andreas D., "A Proposed Bus Arbitration Scheme for Multimedia Workstations" Report Number: WUCS-93-42 (1993). *All Computer Science and Engineering Research.* https://openscholarship.wustl.edu/cse_research/537

Department of Computer Science & Engineering - Washington University in St. Louis
Campus Box 1045 - St. Louis, MO - 63130 - ph: (314) 935-6160.

This technical report is available at Washington University Open Scholarship: https://openscholarship.wustl.edu/cse_research/537

A Proposed Bus Arbitration Scheme for Multimedia Workstations

Saied Hosseini Khayat and Andreas D. Bovopoulos

Complete Abstract:

The integration of video and audio into computers requires the support of continuous streams at the hardware level. This paper proposes a bus bandwidth management and access arbitration scheme for a multimedia workstation. It is assumed that a multimedia workstation consists of several specialized processing modules which are linked by a packet-switched bus. Using the proposed scheme, the bus can support a mix of real-time continuous media streams and random transactions while fulfilling special requirements corresponding to each traffic type. Specifically, the bus provides very fast response to random transactions and serves continuous media streams in such a way that no piece of data falls behind its deadline. Furthermore, the performance with respect to continuous media traffic is maintained independent of time variations of randomm traffic. Practical implementation guidelines are provided. Finally, the performance of the proposed scheme is compared with other possible approaches.

A Proposed Bus Arbitration Scheme for Multimedia Workstations

Saied Hosseini Khayat and Andreas D. Bovopoulos

WUCS-93-42

November 1993

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
St. Louis MO 63130-4899

This work supported by the National Science Foundation, Ascom Timeplex, Bell Communications Research, Bell Northern Research, Gold Star Communications, Italtel SIT, NEC, NTT and SynOptics.

A Proposed Bus Arbitration Scheme for Multimedia Workstations [*]

Saied Hosseini Khayat and Andreas D. Bovopoulos
Computer and Communication Research Center
Washington University in St. Louis

WUCS-93-42
September 1993

Abstract

The integration of video and audio into computers requires the support of continuous streams at the hardware level. This paper proposes a bus bandwidth management and access arbitration scheme for a multimedia workstation. It is assumed that a multimedia workstation consists of several specialized processing modules which are linked by a packet-switched bus. Using the proposed scheme, the bus can support a mix of real-time continuous media streams and random transactions while fulfilling special requirements corresponding to each traffic type. Specifically, the bus provides very fast response to random transactions and serves continuous media streams in such a way that no piece of data falls behind its deadline. Furthermore, the performance with respect to continuous media traffic is maintained independent of time variations of random traffic. Practical implementation guidelines are provided. Finally, the performance of the proposed scheme is compared with other possible approaches.

1. Introduction

The next generation of workstations will be capable of handling continuous media (real-time digital audio and video) as well as conventional media (text, still image and numerical data). Continuous media (CM) processing has special requirements among which is a huge appetite for bandwidth, processing power, and storage. It has become clear that conventional architectures cannot meet these requirements, mainly because of their centralized organization, which leads to processing and transfer bottlenecks. Therefore it is necessary

[*]This work supported by the National Science Foundation, Ascom Timeplex, Bell Communications Research, Bell Northern Research, Goldstar, Italtel SIT, NEC, NTT, and SynOptics.

to move to multiprocessing architectures which allow the distribution of workload among task-specialized modules [7, 2, 6]. As depicted in Figure 1, processing modules should be linked by a high speed multimedia interconnect. A broadband network (e.g. ATM Network) can be accessed via one module functioning as the network interface, or if necessary, via a hardware multiplexor/demultiplexor connected to those modules which need direct link to the network. This paper focuses on the multimedia interconnect and its bandwidth management.

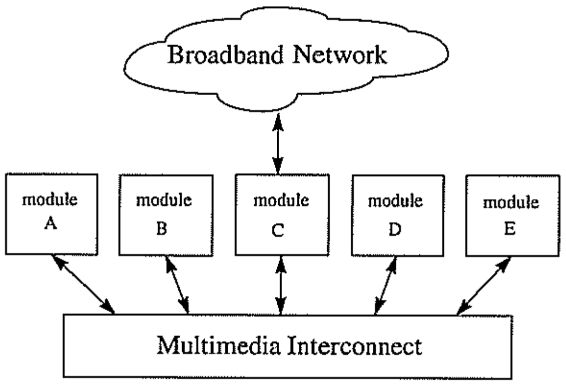

Figure 1: General Organization of a Multimedia Workstation

Ideally, a multimedia interconnect should satisfy several requirements:

1. It should provide the bandwidth necessary for CM and random traffic.
2. It must provide very fast response to random transcations normally associated with virtual memory page faults or cache coherence protocol (if there is a shared memory).
3. It must provide *constant throughput* for each CM stream. Desirably this should be in the form of *backlog-avoidance* in the sense that the transfer of each block of CM data must be completed before the next block arrives.
4. The performance with respect to CM streams should be de-coupled from the time variations of the random traffic.
5. The bandwidth management and access arbitration scheme should be implementable in very fast hardware.
6. The interconnect must be simple enough to be reliable and economical.

Currently there is no interconnect that fulfils these specifications. Most backplane buses currently in use (Multibus II, VMEbus, NuBus, Sbus, etc.) fail to satisfy even the first requirement. They cannot provide bandwidth beyond 100 MB/s [1], whereas a single stream of full-motion video with image resolution of 1024×1280 pixels and 24-bit color and 30 frames/sec requires about 120 MB/s . Image compression schemes (e.g. JPEG, MPEG) can reduce the rate by one or two orders of magnitude to alleviate remote transfer and storage difficulties. However, the transfer of uncompressed video is necessary within the computer. Using high speed interconnect—such as Apple's QuickRing and Sun's XDbus with 350 MB/s and 320 MB/s respectively—is not a complete solution. Fulfilling the other requirements mentioned above necessitates the introduction of innovative bandwidth management schemes which are absent from today's interconnects.

There are different candidate topologies for multimedia interconnect. This paper focuses on the *bus* topology. While the bus topology has certain drawbacks that make it inappropriate for large-scale applications, it offers advantages that make it a viable solution for small-scale purposes such as a workstation. A bus cannot support a large number of bus masters firstly due to capacitive loading effects that limit its maximum operating frequency and secondly because it is a shared medium whose throughput does not scale with the number of bus masters. For small-scale applications such as multimedia workstations, however, a bus offers acceptable throughput and speed at the lowest cost compared to other topologies.

The rest of this paper is organized as follows: Section 2 describes the underlying assumptions of this work. Section 3 discusses the drawbacks of two possible schemes. In section 4, first the proposed scheme is presented in detail, then two implementation methods are discussed, afterwards an example multimedia bus and traffic is described, and finally simulation results are explained.

2. Assumptions

In this work a packet-switched bus is considered. This type of bus can support *split transactions* [11] in the sense that a *read* operation is not atomic and is divided into a *request* and a subsequent *reply* operation. Between a request and its corresponding reply, other bus transactions are allowed to occur, thus improving bus utilization. It is assumed that all packets are of the same size called *cells*. A cell has a header part containing the destination address and other identification information, as well as a payload part carrying data. Therefore in a read operation, the request may be transferred as a single cell, while the reply may consist of many cells.

Transmission of a cell is atomic and takes one time slot of the bus. During each time slot, the bus master in the next time slot is determined. Thus arbitration totally overlaps transmission, making back-to-back cell transmission possible.

Each hardware module may generate two types of cell traffic: random and periodic. Periodic traffic occurs because of CM transactions (real-time video processing, playback, etc.) between different modules and is relatively prolonged in time. Periods are normally much longer than bus time slots. Every module is connected to the bus via a bus interface unit (BIU). Each BIU has some *cell buffer* space . Once a module gains access to the bus, it transmits one cell, releases it and starts another round of contention for another time slot. The arbitration unit determines which BIU uses the bus at any time slot. In the next sections a more detailed description of the BIU and the arbitration unit is provided.

Each continuous media stream is described by two parameters T and B, where T is the period and B is the maximum number of cells in a period. The CM source i *demands* B_i

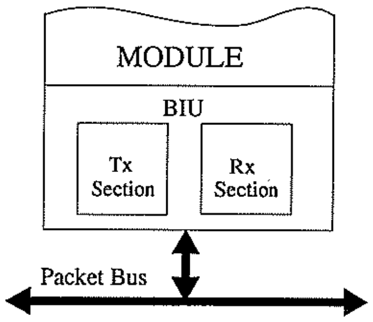

Figure 2: Bus-Module Interface

cells every T_i time slots. Although B_i and T_i do not completely characterize the continuous media stream, a complete characterization is unnecessary. Besides simplicity, the proposed characterization has two advantages. First, it is flexible. For example, a video source may select to send up to 1 MB every 1/30 second, 2 MB every 1/15 second, or even 1/10 MB every 1/300 second. Second, the characterization is applicable for both compressed and uncompressed video/audio.

3. Possible Approaches

This section discussees two possible arbitration schemes and identifies their strengths and weaknesses. In Section 4 a superior scheme is presented and discussed.

The first approach, which we refer to as *indiscriminate policy*, is to treat all cells in the same way. The buffer space in every BIU is arranged as a single *FIFO* queue. CM cells and random cells in each module are queued and served in their order of arrival. The contention among different BIU's is resolved by conventional arbitration schemes such as a daisy-chain or a distributed contention resolution circuit [11].

This scheme has the advantage of simplicity, but it suffers from severe drawbacks with respect to the third and fourth requirements mentioned in Section 1. First, there is no guarantee that CM cells are transferred within the time frame that they belong to. This leads to cell *loss* from the point of view of the real-time application. Second, there is no control on the ratio of lost to delivered CM cells because the user-initiated random traffic is under no constraint. Users can generate momentary bus overloads that result in unacceptable CM performance. It is possible to alleviate these problems by over-engineering the bus bandwidth, however this would be an expensive solution considering the large volume of expected CM traffic.

To preserve the time structure of CM streams it is necessary to discriminate between CM and random traffic. One natural way is to give priority to CM cells over random cells. A problem arises, however, when there are two or more independent CM streams. The time structure of each stream cannot be preserved unless proper scheduling among streams is enforced. The next policy and the proposed policy presented in the next Section address the need for scheduling and priority. The cell buffer space of each BIU will be be divided into two FIFO queues denoted by PQ and RQ for CM cells and random cells respectively.

The second approach is to give high priority to CM cells, thereby putting the random traffic in the background and fulfilling the fourth requirement mentioned in Section 1. To satisfy the third requirement the cells corresponding to each CM stream are dispersed in time. This allows the CM streams to get through without greatly disturbing one another. Specifically, source j in module i, described by (T_j^i, B_j^i), is allowed to place in buffer PQ$_i$ upto one cell every N_j^i time slots, where $B_j^i/T_j^i \leq 1/N_j^i$ and $\sum_{i,j} 1/N_j^i \leq 1$. A static priority scheme such as a daisy-chain is used to resolves the contention among PQ's for the bus. If they are all empty, the RQ buffers containing random cells are served. A second daisy-chain arbitration sheme can be used for contention resolution among RQ's.

The main drawback of this approach, which will be later referred to as the *dispersion policy*, concerns the second requirement stated in Section 1. Random cells, as shown in simulation results in Section 4.4, suffer from long delays especially when CM traffic load is significant. This is accounted for by the fact that random cells are served in the background with respect to CM cells. Another weakness of this approach is that the cell loss, i.e. the transfer of cells after their deadlines, is not eliminated and in fact it tends to happen periodically becuase of the contention of periodic streams.

The scheme proposed in the next section avoids these drawbacks.

4. Proposed Solution

4.1. Cyclic Dynamic Priority Policy

The essence of this policy is to put CM traffic in the background as long as a backlog does not occur. This way *backlog-avoidance* as well as minimal delay of random cells and efficient utilization of bandwidth is achieved.

The bus periodically repeats *service cycles*, each of which is N time slots in duration. A hardware unit called the *bus arbiter* (BA) executes a *service algorithm* during each service cycle. N is a system parameter—typically in the range of 10 to 100—chosen to optimize performance.

The idea is to spread all CM streams in time almost evenly and to serve them in such a way that a series of *shortly-spaced* deadlines, which are separated by N time slots, are met by each stream. If a certain condition (presented later) is satisfied, it is shown that all streams will achieve backlog-avoidance in the sense that every CM cell will get through within its time frame. In the meantime, during each service cycle, CM cells are delayed in favor of random cells as long as they can be sent before their short deadline.

Suppose that there are J modules connected to the bus, there are K_i active continuous media streams that originate from module i, and stream j in module i has demand (T_j^i, B_j^i), $j = 1, 2, \ldots, K_i$. There is a software module called the *bus manager* (BM) which assigns a number M_j^i to the CM stream described by (T_j^i, B_j^i) . M_j^i denotes the number of time slots reserved for this stream in every cycle and is the smallest integer satisfying the following condition:

$$\frac{B_j^i}{T_j^i} \leq \frac{M_j^i}{N} \ . \tag{1}$$

Each module has two FIFO queues in its BIU for outgoing cells. One, denoted by PQ (periodic queue), is dedicated to CM cells, and the other, denoted by RQ (random queue), holds random cells. In every service cycle, module i puts up to $\sum_{j=1}^{K_i} M_j^i$ cells into PQ$_i$, M_j^i of which are out of the j^{th} stream. The goal is to drain every PQ before the next service cycle starts. This and the backlog-avoidance condition that comes later ensure that for each source at least B_j^i CM cells get the chance of transmission every T_j^i time slots. Random cells generated by module i are buffered in RQ$_i$ as they arrive. The goal is to serve all RQ's as soon as possible without violating the former goal. Note that when an RQ becomes full, it does not lead to cell loss, but it results in a suspension of the processes which fill that queue. The bus manager computes

$$Q \stackrel{def}{=} \sum_{i=1}^{J} \sum_{j=1}^{K_i} M_j^i \ . \tag{2}$$

where Q is the number of time slots in a service cycle reserved for all CM streams and is updated when a CM session is admitted or terminated. Admission or rejection is done by the bus manager using this number and an *admission rule* that is presented later. Every time Q is updated, it is written into a special register in the bus arbiter. The bus arbiter continually executes the following service algorithm:

```
procedure service($N, Q$)
    while $1 = 1$ do
        $n = N$;
        $q = Q$;
        while $n > 0$ do
            if $n > q$ then
                if some RQ$_i$ non-empty then
                    serve one cell from a non-empty RQ$_i$;
                else
                    if some PQ$_i$ non-empty then
                        serve one cell from a non-empty PQ$_i$;
                        $q \leftarrow q - 1$ ;
                    fi
                fi
            else
                if some PQ$_i$ non-empty then
                    serve one cell from a non-empty PQ$_i$;
                    $q \leftarrow q - 1$ ;
                else
                    if some RQ$_i$ non-empty then
                        serve one cell from a non-empty RQ$_i$;
                    fi
                fi
            fi
            $n \leftarrow n - 1$ ;
        od
    od
endprocedure
```

It is seen that, given $Q \leq N$, the cycle time is divided into two intervals. The first is the interval during which $q < n$. In this interval priority is given to random traffic, and if all random queues (RQ's) are empty, the bus is granted to CM traffic. Note that in any case n is decremented by one at any time slot. The second interval starts when $q = n$. (Note that $q > n$ denotes a faulty condition that should not logically occur.) In this interval CM traffic gains high priority, and, in case there is no CM cell in any periodic queue (PQ's), random cells get service. This case may occur due to random fluctuations of CM traffic or slight over-allocations of bandwidth. It is interesting to note that the boundary between the two intervals is pushed toward the end of the cycle as CM cells are served in the first interval. For moderate amounts of CM and random load, CM traffic is almost *transparent* to random traffic. Therefore random cells suffer delay only due to the share of random traffic in the total load. This is an enormous improvement over the previous schemes. On the other hand, performance with respect to CM traffic is completely *independent* of random traffic because a sufficient number of time slots is reserved for CM traffic in every cycle. It is shown later that a mild condition on the demands of CM streams will result in a perfect backlog-avoidance.

Admission Rule – Admission of new continuous media sessions is the job of the bus manager, which is a software process. A request for a new CM session is done by sending the new demand (T_{new}, B_{new}) to the bus manager. The bus manager computes M_{new} using (1) and admits the session if the following condition holds:

$$M_{new} \leq N - \alpha - Q_{current} \; , \tag{3}$$

where $Q_{current}$ is the total number of time slots in a cycle reserved for currently admitted CM streams using (2) and α denotes the number of time slots reserved for random traffic. This number is set by the super-user and prevents the response time of the system from degrading to unacceptable limits.

Backlog-avoidance Condition – Backlog-avoidance for a CM source with demand (T, B) can be achieved if the source can actually send at least B cells in every period T. This allows loss-free transmission of the stream with no more than $2B$ cells of memory space. Backlog-avoidance is ensured if

$$B \leq M \; (\lceil \frac{T}{N} \rceil - 2) \; , \tag{4}$$

where N is the bus service cycle and M is found using (1). Given that $N < T$, condition (4) is based on the fact that for any choice of T and N, there are at least $\lceil T/N \rceil - 2$ intervals of length N totally overlapped by an interval of length T. When $N \ll T$ —usually true in practice—this condition is not very restrictive because it is very close to condition (1), which is already satisfied. Another way to secure backlog-avoidance is to choose T and N such that T is an integral multiple of N and make source periods run in synchronization with bus service cycles. This requirement is not hard to fulfil in practice. For example, if a bus service cycle is 10 μs, the continuous media application should choose its transmission period to be a multiple of 10 μs, which is not a major restriction.

If backlog-avoidance condition is ignored, then there is a possibility that at most $2M$ cells occasionally miss their deadlines. This may be still a reasonable performance for some applications because cells loss is upper-bounded by a known value.

Choice of Parameters – There are two parameters to be chosen, N and T. N is a system parameter under control of the super-user, and T is an application parameter selected by the user. For best results, their values should be chosen far apart.

Qualitatively speaking, for a fixed random and CM load, increasing the cycle time N increases the room for delay of CM cells in favor of random cells and helps CM traffic be absorbed in the background , thus reducing the delay of random traffic. It helps to imagine that N is infinitely large. This implies that all CM cells can be delayed forever; in this case obviously random cells will see no CM traffic and only experience the delay due to their own traffic. This is an extreme case which is not possible because CM cells have finite deadline. N can be at most equal to the smallest T. On the other hand, increasing N implies increasing all M's, which results in increased buffer space in the BIU's. Therefore delay is traded against buffer space. Another implication of large N is the difficulty to fulfil the backlog-avoidance condition and the larger number of lost cells when that condition is not met.

The choice of T has a similar effect as the choice of N. For example, for a given random and CM load, suppose that all T's are doubled. N can also be doubled as a consequence, resulting in improved delay performance for random cells. On the other hand, to keep the CM loads fixed, all B's must be doubled, resulting in an increased memory requirement.

B can also be a parameter of choice in some applications. In the case of uncompressed video/audio, a fixed number of cells β is generated in every frame, and B is normally set equal to β. For example, a high quality video may need $\beta = 10000$ cells every 1/70 sec. One may choose to have the best quality, in which case B is equal to β, or may settle for a lower quality by setting $B = 5000$ in order to reduce memory usage and also to increase the chance of the stream being accepted by the system. For a compressed video/audio stream, however, the number of cells generated per frame is a random variable with an upper bound β. B may be chosen to equal β, ensuring that quality is not lost, or it may be set to a value slightly lower than β , sacrificing some quality only occasionally. It may also be possible to take advantage of statistical pooling in the sense that all compressed CM sources set their B slightly lower their β values and share a pool of time slots equal to the sum of their M's. This allows more sessions to be admitted and less memory to be consumed, but the cost is additional complexity.

4.2. Implementation

The service algorithm can be realized with simple and fast hardware, thus fulfilling the fifth and sixth requirements mentioned in Section 1. For any module, there is one BIU connected to the bus (Figure 3). Assuming every module can be a sender of real-time video/audio, each BIU has two cell queues, RQ (random queue) and PQ (periodic queue), either of which can have cells to transmit at any time slot.

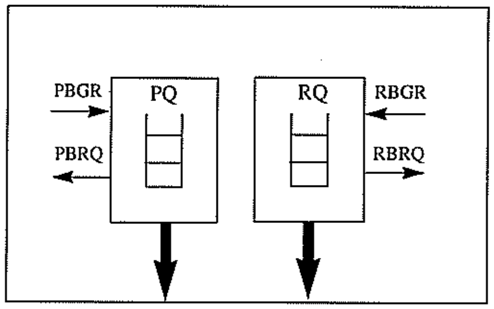

Figure 3: Transmit Section of a BIU

The bus arbiter should grant the bus to only one queue during any time slot according the service algorithm. There are two signal lines associated with every queue. BRQ (bus request) is an output signal which is active (low) when the queue is non-empty. BGR (bus grant) is an input signal issued by the arbiter and is active (low) when bus is granted. When BGR becomes active for a queue, BIU drains that queue by one cell, which is sent in the next time slot. BRQ and BGR have a prefix R (P) to show they are related to an RQ (PQ). The bus arbiter has two registers N and Q initially loaded by the bus manager, as well as two counters n and q. At the end of every cycle n and q are re-loaded from N and Q respectively. There are two methods of implementation.

Serial Implementation – The bus arbiter and BIU's each have two input signals RGI and PGI (GI for *grant in*) and two output signals RGO and PGO (GO for *grant out*). A daisy chain is formed as in Figure 4.

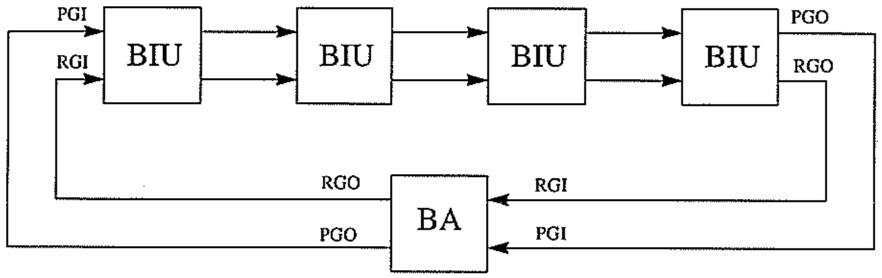

Figure 4: Serial Implementation (Double Daisy-chain)

Every BIU uses the following logic with respect to each of its queues.

$$\mathtt{GO} = \mathtt{GI} \vee \overline{\mathtt{BRQ}}$$

$$\mathtt{BGR} = \mathtt{GI} \vee \mathtt{BRQ}$$

The BIU blocks a RGI (PGI) signal from passing on to the next BIU, if its RQ (PQ) has a cell to send. In every time slot, the bus arbiter executes the following logic:

```
if $n > q$ then
    activate RGO;
    wait for signal to ripple through;
    if RGI active then
        activate PGO;
        wait for signal to ripple through;
        if PGI inactive then
            $q \leftarrow q - 1$ ;
        fi
    fi
else
    activate PGO;
    wait for signal to ripple through;
    if PGI inactive then
        $q \leftarrow q - 1$ ;
    else
        activate RGO;
    fi
fi
$n \leftarrow n - 1$ ;
```

This logic, being trivial to realize, works properly if a time slot is longer than two complete ripple-through times for grant signals. This sets a lower bound on the length of a time slot for a given number of modules—or equivalently an upper bound on the number of modules for a given time slot. If other considerations dictate a shorter time slot, parallel implementation must be chosen.

Parallel Implementation – This implementation requires that all RBRQ and PBRQ lines be the input to the bus arbiter (Figure 5). A combinational logic circuit is used to generate all RBGR, PBGR signals in parallel.

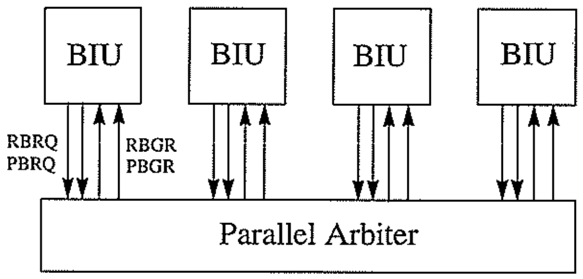

Figure 5: Parallel Implementation

This can be done in different ways. One realization is the following: The bus arbiter has three sub-units, I, II and III (Figure 6). Sub-units I and II are identical. Each consists of a priority encoder cascaded by a binary decoder. All RBRQ lines (PBRQ lines) are input to sub-unit I (II). A binary code corresponding to the active line with highest priority is generated at the output of priority encoder, which is the input to a binary decoder. The

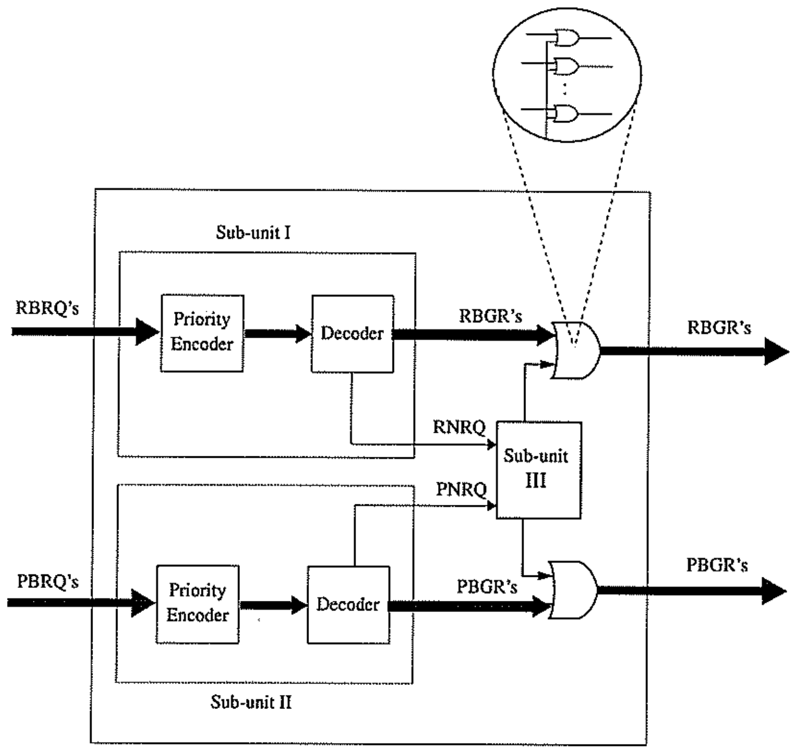

Figure 6: A Parallel Arbiter

outputs of the decoder correspond to BGR lines with only a single active line corresponding to the highest priority input. Another output RNRQ (PNRQ)—NRQ for *no request*—is also generated , which becomes active when there is no active BRQ. Sub-unit III, which contains two registers N and Q and two counters n and q, takes RNRQ and PNRQ inputs and generates two outputs based on the following logic. The outputs select one set of BGR lines generated by sub-unit I or II to appear at the output of bus arbiter and de-select the other set.

```
if $n > q$ then
    if RNRQ inactive then
        select sub-unit I ;
        de-select sub-unit II ;
    else
        if PNRQ inactive then
            select sub-unit II;
            de-select sub-unit I;
            $q \leftarrow q - 1$ ;
        fi
    fi
else
    if PNRQ inactive then
        select sub-unit II ;
        de-select sub-unit I ;
```

```
                $q \leftarrow q - 1$ ;
            else
                select sub-unit I;
                de-select sub-unit II;
            fi
        fi
        $n \leftarrow n - 1$ ;
```

Even a faster realization is possible if the above logic is totally built as a single two-level combinational logic circuit.

4.3. Application Example

First, a packet bus is described which serves both as a example of a high performance multimedia bus and as a model used in our simulations.

Multimedia Bus – This bus *, initially inspired by Sun's XDbus, has a 64-bit useful data path operating at 40 MHz. A bus clock cycle is 25 *ns* in which 8 bytes of data can be transferred. For the sake of efficiency there are two types of packet: *request* and *reply.* A request packet is of fixed size of 16 bytes, while a reply packet is $128 + 8$ bytes, 8 bytes of which is header.

The bus is logically divided into two separate buses, A-bus and B-bus. A-bus, being $64 + 4$ bits wide, is dedicated to reply packets. Data is carried over 64 lines, and header is carried at the same time over 4 dedicated lines. Thus the time overhead associated with headers is eliminated at the cost of 4 additional lines. B-bus is only 8 bits wide and carries only request packets. The above choices make it possible to transfer 128 bytes of useful data every 400 *ns.* Also note that it takes equal time to transfer a request or a reply and this can be done concurrently. Therefore a useful bandwidth of 320 Mbytes/sec is provided, fulfilling the first requirement of a multimedia interconnect.

Arbitration for B-bus is done using conventional schemes because requests are generated at random. CM traffic is absent from B-bus because it is reservation-based in contrast to request-reply-based. However A-bus carries a mix of random and CM traffic, and arbitration is done using the scheme proposed here. With 400 *ns* slot time, both serial and parallel arbiters work properly.

Example Traffic – Suppose, as an example, that there are 10 modules interconnected by the above multimedia bus.

- Module A,B : Processing units
- Module C,D : Image processing units
- Module E : Shared memory

*If advantageous, one may design a bus based on standard ATM cells in order to make the interconnect conformable to ATM networks.

- Module F : Monitor & keyboard
- Module G : High speed network interface
- Module H : Video storage unit
- Module I : Data storage unit
- Module J : Video camera unit

Assume the following scenario. There are two compressed video streams arriving at module G from the network. One stream is destined for module H. The second stream goes to module C, for say image decompression and enhancement, and then goes to module F. Another compressed video stream comes out of module J headed for module G to go to a remote user and is simultaneously received by module F to be displayed. Suppose the demand of each compressed video is $(250000, 4608)$. This means that each stream is guaranteed to use 4608 time slots every 0.1 second, which corresponds to about 6MB/s. This is equivalent to a video image of resolution 1024×1280 and 24 bits/pixel with an average compression ratio of 1/20. Choosing a 0.1 second period allows for the time variation of the stream to be smoothed out. The demand of decompressed video stream is $(250000, 92160)$, which corresponds to 92160 cells every 0.1 second, or about 118 MB/s. There will be 4 streams to be transferred over the bus, 3 compressed and 1 uncompressed video.

Let the bus service cycle be 50 . Applying condition (1) yields

$$M_{compressed} = 1$$

$$M_{uncompressed} = 19$$

Therefore module F is allowed to use 2 time slots (for two streams), module C 19 slots, and module J 1 slot every bus service cycle. Overall there are 22 time slots reserved for CM streams ($Q = 22$) . It is easily checked that condition (4) already holds. Thus at most %44 of the bandwidth is used up by continuous media, the rest is for random traffic. Random traffic mainly arises due to virtual memory page faults, shared memory usage, data and control messages passed between modules, and non-CM network activities such as image retrieval, remote transfer of position of a pointer in tele-presentation, or multimedia email.

4.4. Simulation Results

Simulation was done as an attempt to visualize the performance of the proposed scheme as well as candidate schemes. Throughout this study a Bernoulli trial model was adopted to simulate the arrival of random cells, in which module j generates at any time slot a random cell with probability p_j. This model is not realistic, but it served as a common reference input to different schemes in order to make the performance comparison possible.

CM traffic was chosen to be deterministic and resembled closely those streams described above, with the exception that compressed video streams generate at every period a fixed number of cells equal to their maximum demand. This is an extreme case, and the results should be interpreted as worst case. The following definitions have been used throughout:

$$\mathit{CM\ load} = \sum_{\mathit{all\ streams}} B_i/T_i \ ,$$

$$\mathit{Random\ load} = \sum_{\mathit{all\ modules}} p_j \ .$$

The only performance criterion under consideration has been the delay of random cells. Another performance criterion could be the loss (i.e. after-deadline transfer) of CM cells, but since the condition (4) was satisfied by every CM stream, CM loss could not occur and was not considered.

A comparison of the indiscriminate policy, the dispersion policy and the cyclic dynamic policy is shown in Figures 7 ,8 and 9. Figures 7 and 8 show the mean and standard deviation of delay of random cells for a fixed value of CM load at 0.5 . It is seen that delay is the lowest for the cyclic dynamic priority policy (with service cycle N equal to 40) especially at heavy random traffic. Figure 9 is a histogram comparing the distribution of delay for the three schemes when random load is fixed at 0.5 and cycle N fixed at 20. Performance of cyclic dynamic priority is shown in the next three figures. Figure 10 is the histogram of delay for a fixed $N = 40$ and different random loads. Figure 11 plots mean delay versus service cycle N with random load as parameter. CM load is fixed at 0.5 for these plots. It is shown that increasing N decreases the delay of random traffic. Figure 12 shows the mean delay in the presence and absence of CM traffic. It is seen that in the absence of CM traffic, the mean delay is a constant independent of N, and when CM traffic is present, as N is increased, the mean delay asymptotically approaches the value of the mean delay in the absence of CM traffic. This can be interpreted as the absorption of CM traffic in the background. It is seen that in the above example, the service cycle N may be chosen equal to 60 in order to achieve the best performance.

5. Conclusion

This paper proposed an arbitration scheme for a packet-switched bus that allows a mix of continuous media stream and random transactions to be supported on the bus. The scheme offers perfect service of continuous media streams in the sense that no piece of data is sent after its associated deadline, and it provides almost minimal delay for random packets by effectively putting the continuous media traffic in the background. Implementation of this scheme was also shown to be practical and simple.

Acknowledgement – The authors have enjoyed fruitful discussions with Dr. Jonathan S. Turner, H. Saidi and R. Gopalakrishnan.

References

[1] Borrill, P.L., "32-bit buses—An objective comparison," *Proc. Buscon 1986 West,* San Jose, California.

[2] Bovopoulos, Andreas D., Gopalakrishnan, R., and Hosseini Khayat S., "SYMPHONY: A Hardware, Operating System, and Protocol Processing Architecture for Distibuted Multimedia Applications," Technical Report WUCS-93-06, March 1993.

[3] Cole, Bernard, "The Technology Framework," *IEEE Spectrum,* March 1993.

[4] Cram, R.M., "Microcomputer Busses," Academic Press Inc., 1991.

[5] Giacomo, Joseph Di, "Digital Bus Handbook," MacGraw-Hill, 1990.

[6] Gopalakrishnan, R. and Bovopoulos, A. D., "A Protocol Processing Architecture for Networked Multimedia Computers," *Operating Systems Review,* Vol. 27, No. 3, July 1993.

[7] Hayter, M.,and McAuley, D., "The Desk Area Network," *Operating Sytems Review,* October 1991.

[8] Lyle, James D., "SBus, Information, Applications and Experience," Springer-Verlag, New York Inc., 1992.

[9] Patterson, David A., Hennessy, John L., "Computer Architecture, A Quantitative Approach," Morgan Kaufmann Publishers, Inc., 1990

[10] "SPARCcenter 2000, Architecture and Implementation," Sun Microsystems, Inc., Technical white paper, Draft Edition of 11/18/92.

[11] Ward, Stephen A., Halstead, Robert H., "Computation Structures," The MIT Press, 1990.

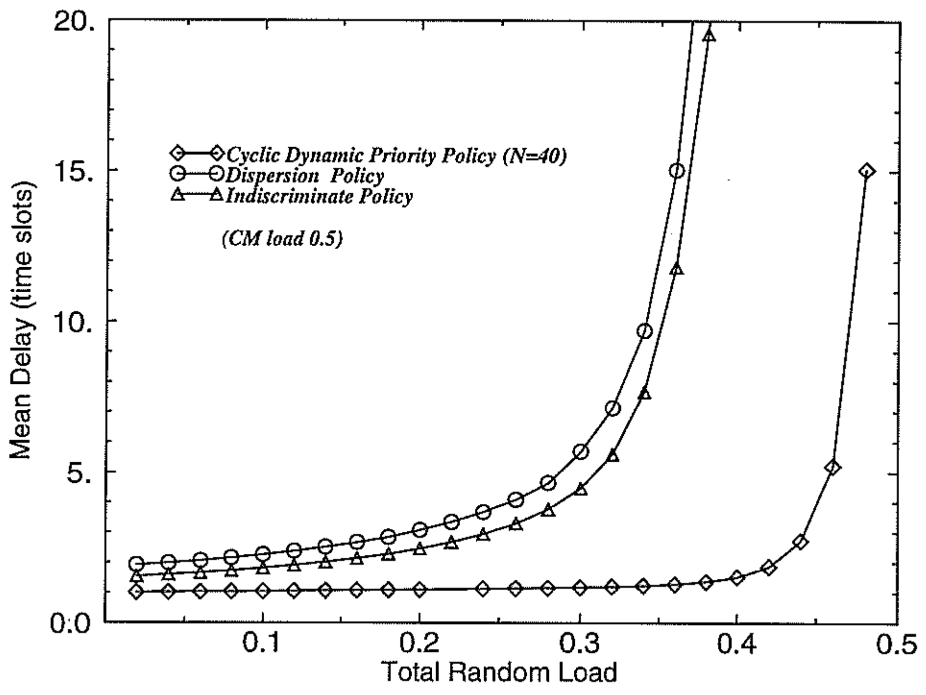

Figure 7: Comparison of Mean Delay for Different Schemes

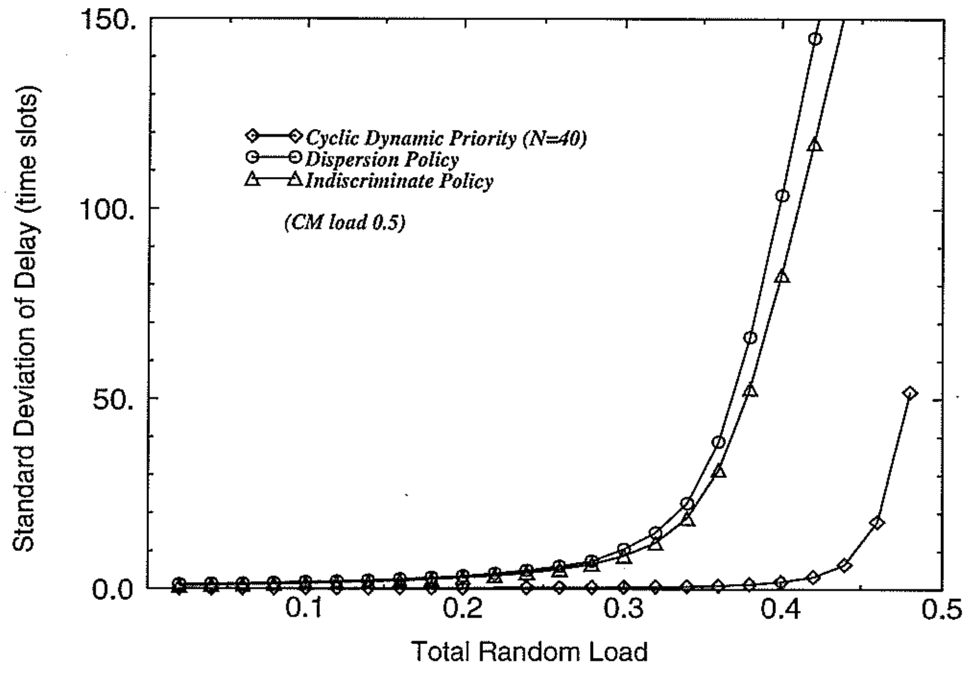

Figure 8: Comparison of Standard Deviation of Delay for Different Schemes

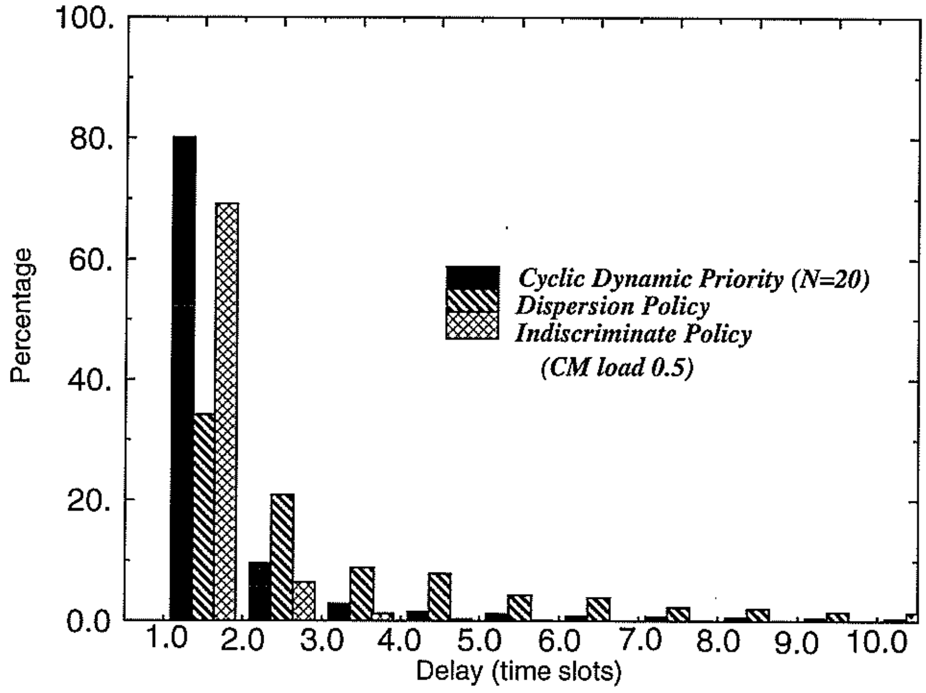

Figure 9: Distribution of Delay for Different Schemes

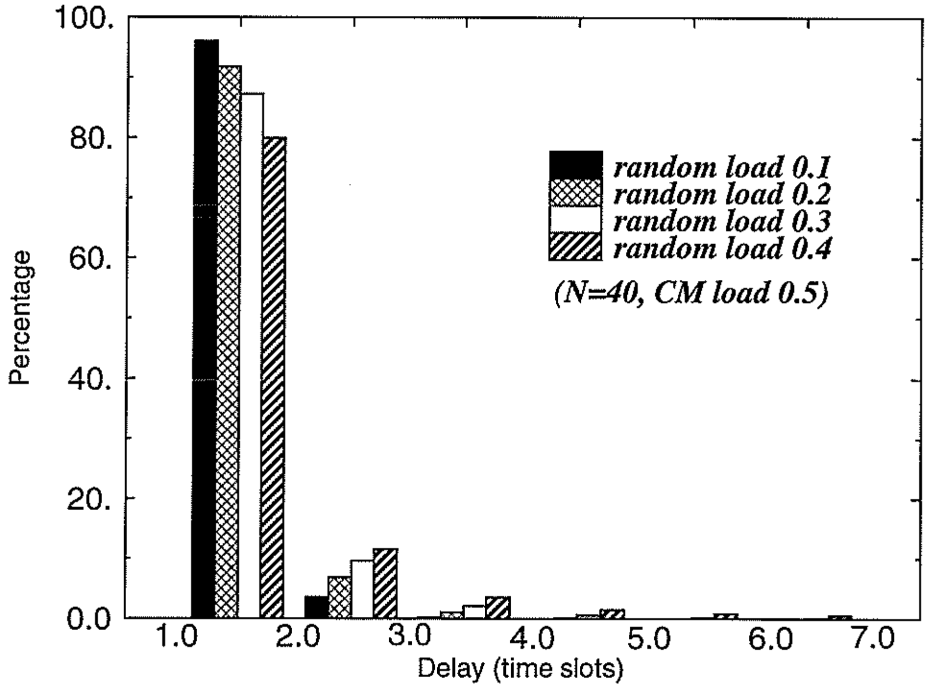

Figure 10: Distribution of Delay for the Proposed Scheme

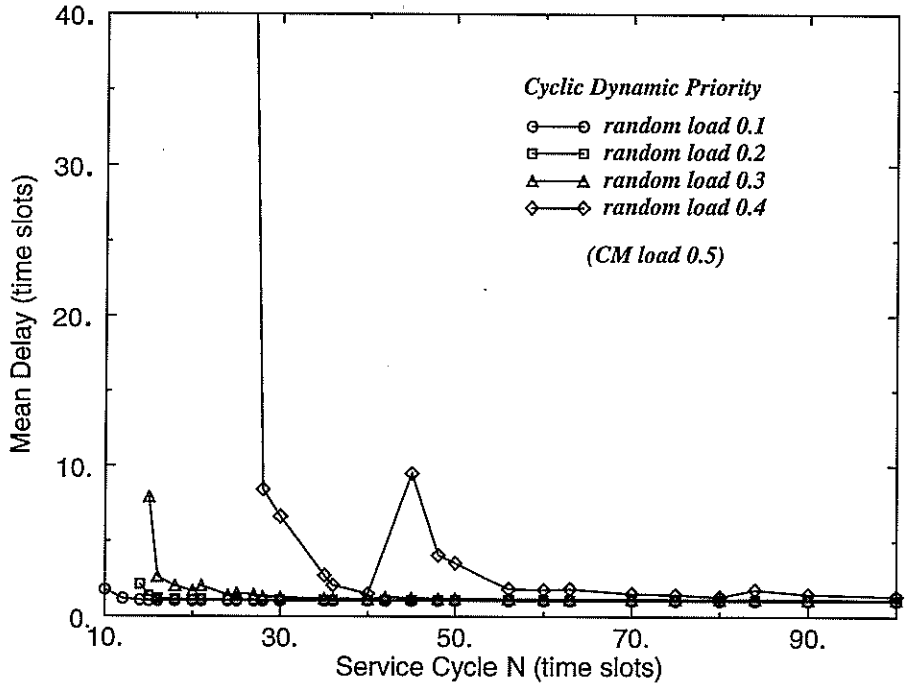

Figure 11: Mean Delay for the Proposed Scheme

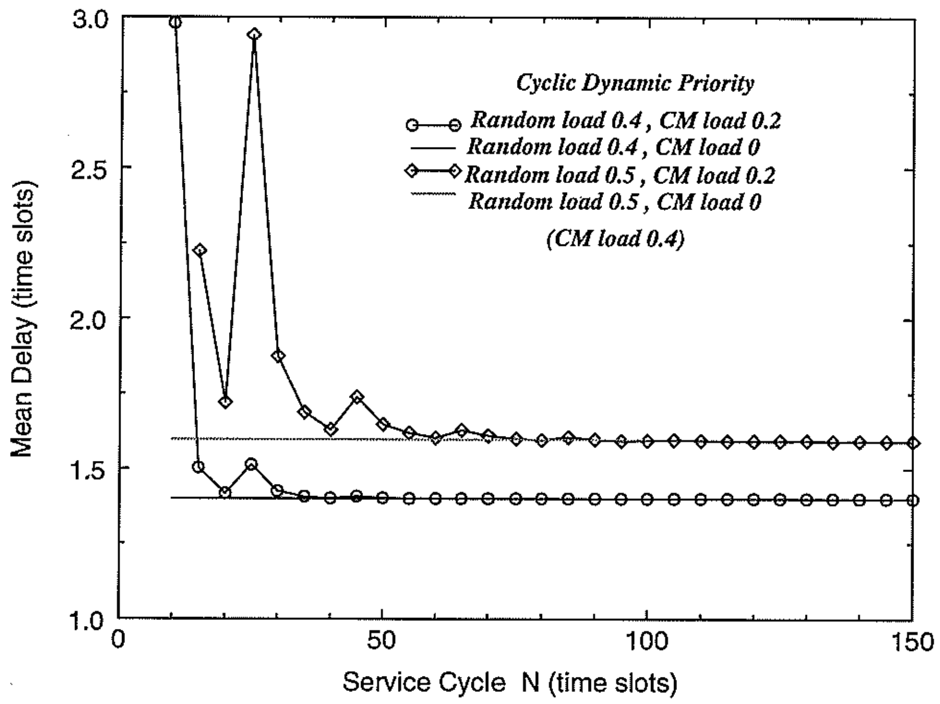

Figure 12: Mean Delay for the Proposed Scheme