*Proc. Natl. Sci. Counc. ROC(A)*
Vol. 19, No. 6, 1995. pp. 614-631

# A Reservation-based Media Access Protocol on Multiple-bus Distributed Shared-Memory Multiprocessors

I-SHYAN HWANG

*Department of Electronic Engineering*
*Van-Nung Institute of Technology and Commerce*
*Chung-Li, Taiwan, R.O.C*

(Received May 18, 1994; Accepted March 20, 1995)

**ABSTRACT**

A popular architecture for multiprocessor system is a multiple-bus interconnection network. Multiple buses may increase capacity, but the system is still severely limited in size due to bus contention. A parallel I/O system and a decentralized crossbar switch system are proposed to support the operations of a *reservation* (control-bus based) TDMA-C protocol and increase the scalability of DSM multiprocessor multiple-bus interconnection networks. The function of the *control bus* is to broadcast reservation requests, and the broadcast capability is also used to support cache coherence level control signals such as invalidations. This *reservation* approach can achieve collisionless transmission and avoid or alleviate memory conflict before transmitting data through the data bus. A semi-Markov model is developed to reflect the impact of low level media access protocols on high level systems Trace-based discrete-event simulation has been used to evaluate the model The model is used to examine the system performance with varying system parameters

**Key Words:** DSM, multiple-bus interconnection network, *reservation* media access protocol, parallel I/O system, decentralized crossbar switch system, semi-Markov model

## I. Introduction

Distributed shared memory (DSM) systems have attracted much interest as a means of obtaining the topological advantages of distributed memory systems and the programming advantages of shared memory systems (Thakkar *et al.*, 1990; Minnich and Farber, 1990; Bisiani and Ravishankar, 1990; Wittie *et al.*, 1990). DSM systems have a global address space that is shared by all the processors (Fleisch and Popek, 1989; Tam *et al.*, 1990). A popular architecture for multiprocessor systems is a multiple-bus interconnection network (Feng, 1981; Lang and Alegre, 1982; Mudge and Al-Sadoun, 1985; Mudge *et al.*, 1986; 1987). The multiple-bus interconnection scheme allows easy incremental expansion of the number of nodes, and multiple buses may increase capacity, but the system is still severely limited in size due to bus contention (Mudge *et al.*, 1987; Wilson, 1987; James *et al.*, 1990; Gustavson, 1992; Gupta and Weber, 1992; Scott *et al.*, 1992; Tung, 1991; Bertoni *et al.*, 1992). The main reason is that the common bus tends to be the primary source for contention, and thus imposes a limit on the number of processors in the system. The key issue in designing multiple-bus interconnection networks is to support larger scale multiprocessor systems. This paper proposed a *reservation* (control-bus based) media access protocol TDMA-C (Bogineni *et al.*, 1993; Bogineni and Dowd, 1992; Hwang and Dowd, 1995; Dowd and Hwang, 1995), a parallel I/O system, and a decentralized crossbar switch system to support DSM multiprocessors through multiple-bus interconnection networks.

The multiple-bus system is proposed in this paper. One of the buses is the *control bus*, and the rest of buses are *data buses*. A node may transmit or receive on any data bus as well as the control bus. When a node needs to access a particular node's memory, each node-node pair is connected by several available data buses. The arbitration of a node-node pair is determined by going through the control bus. Access to the control bus is based on a static cyclic slot allocation scheme (Bogineni and Dowd, 1992). Each node is assigned one control slot per cycle, and all nodes have an opportunity to transmit a control packet or data packet during each cycle. Collisionless transmission is achieved by this approach through the use of status tables. Each node maintains two tables; a table to track the status of the

data bus in order to eliminate data bus collision, and a table to avoid destination conflict by tracking the status of the receiver of the *data bus* at each node. The purpose of the *control bus* is to broadcast reservation requests, and the broadcast capability is also used to support cache coherence level control signals such as invalidations (Chu and Dowd, 1992). The snooping-based protocol investigated in this paper employs a *write-invalidate* policy. Section II.4 The traffic waiting to be transmitted at a node can be of two types: *requests* and *responses*. A *request packet* is generated when the local node needs the target's data when Read Miss or Write Miss occurs. The *request packet* is formed as a control packet and is transmitted through the control bus in a preassigned control slot. A *response packet* can be either the requested data or the *write permission* in response to a request packet received by the local node. The *response packet* is formed as a control packet and data packet if it is the required data. The data packet can be transmitted immediately after the control packet transmits successfully. The *response packet* is formed as a control packet if it is the *write permission* and is transmitted through control bus in a preassigned control slot.

A parallel I/O system (Chen *et al.*, 1990; Hou *et al.*, 1992; Jain *et al.*, 1992) and a decentralized crossbar switch system are used to support the operations of the *reservation* media access protocol and increase the scalability of the multiple-bus interconnection networks. Each parallel I/O system has three I/O ports: one is connected to the *control bus*, and the others are connected to the rest of the buses called *data buses*, and an I/O controller. The function of two I/O ports connected to the *data bus* is to avoid a collision when a node transmits data and receives data simultaneously. It can achieve data processing parallelly, relax the high-speed switch and I/O interface technologies, and further alleviate the I/O bottleneck problem. The I/O controller deals with the data process (transmit data or receive data) on the data bus, and the control packet on the control bus in sequence.

All current approaches to high-performance switching fabrics employ a high degree of parallelism, distributed control, and the routing function (Turner and Wyatt, 1983; Vaidya and Pashan, 1988; Ahmadi and Denzel, 1989; Davie, 1993). High-performance switching fabrics are performed at the hardware level (Turner and Wyatt, 1983; Ahmadi and Denzel, 1989; Killat, 1987). A decentralized crossbar switch system supports the operations of the media access protocol in multiple-bus based interconnection networks. Crossbar-based switches have always been attractive because they are internally nonblocking and simple. This way, the matrix itself remains as simple as possible and can be realized economically with high-speed technology. By use of several matrices in parallel, even higher speeds can be achieved. In addition, this switch has the address decoder function distributed in the crosspoints of the matrix to achieve a *self-routing* crossbar switch.

This paper is organized as follows: Section II describes the multiple-bus system architecture in terms of the node architecture, system organization, decentralized crossbar switch system, parallel I/O system, media access protocol, and cache coherence protocol. Section III addresses the assumptions and operations of the multiple-bus system and develops the semi-Markov performance model. Section IV defines the performance metrics, verifies the model through a comparison with simulation, and compares the relative performance with varying system parameters.

## II. System Architecture

The multiple-bus system shown in Fig. 2 and the node architecture shown in Fig. 1, comprise a distributed memory MIMD system that appears to the application developer to be a shared memory MIMD system. The system is based on a multiple-bus structure with a static cyclic slot allocation scheme to support the communication requirement. The following section describes the proposed architecture, the media access protocol on which the cache coherence protocol is based, the parallel I/O system, and the operations of the decentralized crossbar switch system.

### 1. Node Organization

The multiple-bus system consists of $m$ identical nodes. Each node possesses a local processor, a memory management unit (MMU), and a parallel I/O system as shown in Fig. 1. A node has two levels of cache: the processor cache (PC), as with the Intel *i*860, and DEC Alpha (21064-AA), and an extended cache (EC) located in the physical memory of the node. The physical memory of a node is partitioned into EC and local (global) memory (LGM). LGM is mapped into the global physical address space of the system, and the EC is used to cache the global virtual memory. An objective of the proposed approach is to take advantage of recent advances in microprocessor design that have on-chip instruction and data processor cache with write-back (as with Intel *i*860) or write-through (as with DEC Alpha) coherence. This allows the EC to be constructed with the same low cost memory as in the LGM. Although a node is described as if one processor is contained per node, each node could be viewed as a cluster of processors as in DASH (Lenoski *et al.*, 1992). Further-

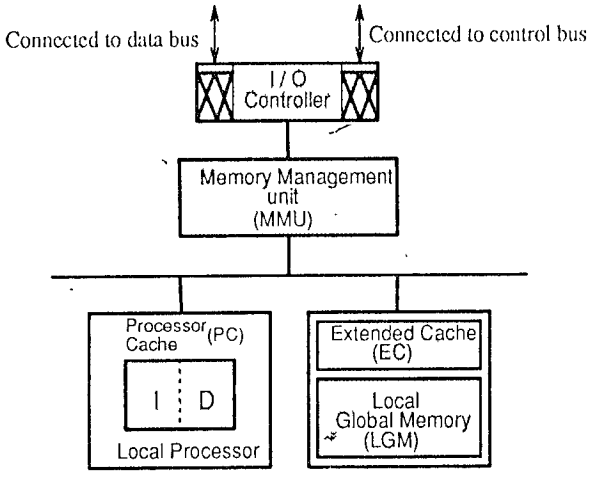


**Fig. 1.** Node organizarion of a multiple-bus system.

more, the performance model presented in Section III can support clustered nodes by viewing the requests to the EC as the aggregate of the individual processor requests within a cluster.

The LGM at each node is mapped into the global physical address space of the system. Each processor owns a portion of the physical global address space and manages a portion of the global virtual memory depending on the address allocation scheme. Copies of memory blocks from global memory are staged in the EC. EC consistency is maintained by a cache coherence protocol. The system is described from the view-point of a tagged *(source or local)* node, so the memory at a *(owner, target or dirty)* node is denoted by NLGM (non-local global memory).

When cache-able data required by a processor is not in its EC, a miss occurs that causes the MMU to issue a shared memory request for a block transfer. If the data is not resident in LGM, the MMU transmits a request to the block owner through the multiple-bus network. Support of cache coherence is an important issue for DSM systems and is described in Sec. II 3. The design of the memory organization and address allocation scheme (Bogineni and Dowd, 1992) was motivated by the goal of simplifying its task by restricting a single copy of a block in the global virtual memory and allowing easy/fast identification of block ownership.

The parallel I/O system contains three I/O ports: one for control packet transmitting/receiving on the control bus; one for transmitting data on the data bus; and one for receiving data on the data bus, respectively, and an I/O controller deals with data processing in sequence. The parallel I/O system, by incorporating the reservation media access protocol and decentralized crossbar switch system, can increase the scalability of the multiple-bus interconnection network. It can achieve data processing parallelly to relax the high-speed switch and I/O interface technologies, and further alleviate the I/O bottleneck problem (Chen *et al.*, 1990; Hou *et al.*, 1992).

## 2. System Organization

This section describes the architecture of the multiple-bus interconnection network and the operations of the decentralized crossbar switch system on multiple-bus interconnection networks.

### *A. The Multiple–Bus System*

The proposed architecture is shown in Fig. 2. It contains $m$ nodes, each having its own processor $P_i$, memory $M_i$, private cache, and parallel I/O system, $1 \le i \le m$. The control bus is defined as $b_0$; and $b_i$, $1 \le i \le (B-1)$, denotes a data bus. The purpose of the *control bus* is to broadcast reservation requests, and the broadcast capability is also used to support cache coherence level control signals such as invalidations. This *reservation* approach can achieve collisionless transmission and avoid or alleviate memory conflict before transmitting the data through the data bus. A node may transmit or receive on any data bus as well as the control bus in this configuration. When a node needs to access a particular node's memory, it has *B-1* buses from which to choose. Each node-node pair is connected by several redundant paths, which implies that the failure of one or more paths can, in principle, be tolerated at the cost of some degradation in system performance. The arbitration of node-node pair is determined by going through the control bus. In this paper, access to the control bus is based on a static cyclic slot allocation scheme (Bogineni and Dowd, 1992). Each node is assigned one control slot per cycle, and all nodes have the opportunity to transmit a control packet or data packet during each cycle. Collisionless transmission is achieved by this approach through the use of status

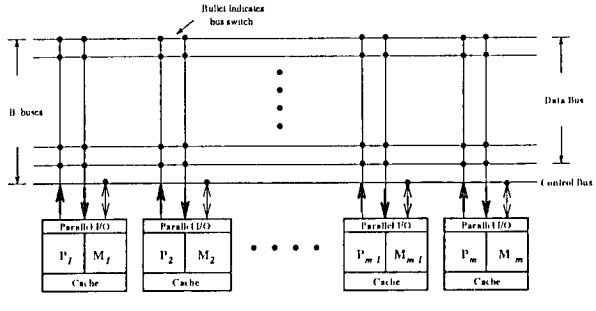


**Fig. 2.** A multiple-bus system.

tables discussed in Sec. II 3. This approach can avoid the memory interference or memory conflict (Mudge *et al.*, 1986, 1987) and increases the scalability of the multiprocessor systems without increasing the complexity of the multiple-bus interconnection networks.

*B. The Decentralized Crossbar Switch System*

Crossbar-based switches are internally nonblocking and simple (Ahmadi and Denzel, 1989). A decentralized crossbar-based switch system is shown in Fig. 3. The system distributes the control function to relieve the control overhead (Ahmadi and Denzel, 1989). The switch has the bus address decoder function distributed in the crosspoints of the matrix to achieve a *self-routing* crossbar switch.

The operations of this crossbar switch system are highly dependent on the media access protocol. If the entries for both target node and a data bus status tables are all 0, the control packet is formed with the source, target, selected data bus, and packet length identifiers. The control packet can be transmitted in preassigned slot. After the source node transmits the data packet at the end of its control slot, it sends the trigger signal to trigger the selected bus to let the source node's data packet pass through. The operation is in reverse direction for the destination node. The destination node waits to receive the coming data packet from the source node after sending the trigger signal to trigger the selected bus.

This approach achieves a high degree of parallelism, distributed control, and the routing function, since several nodes can transmit data at the same time through different buses, and each node can transmit and receive the data parallelly. Each crosspoint of the matrix has an address decoder to decode the selected data bus independently.

## 3. Media Access Protocol

This section describes the media access protocol being considered to transmit data packets between each node-node pair in this paper. TDMA-C (Dowd, 1991) is a reservation-based protocol with a time-division multiplexed control bus. The system considers $m$ nodes and $B$ buses numbered $\{b_0, b_1, ..., b_{B-1}\}$.

Each node has the capability of switching to any data bus. Data packet transmission and data packet reception begins when the data bus is set up between the source node and target node. The control packet has four integer fields for the reservation process of the access protocol $s$, $d$, $i$ and $T$: $s$, $1 \le s \le m$, identifies the source node address; $d$, $1 \le d \le m$, identifies the destination node; $i$, $1 \le i \le B-1$, identifies bus $c_i$ as the selected data bus; and $T$ indicates the data packet length. Additional fields are used for cache coherence operations. Time is normalized to the *control slot*, the time required for transmission of a control packet. $L$ is the propagation delay from the source node to destination node. A data packet is taken to be a positive integer $T$ times the length of a control packet.

Access to the control bus is based on a static cyclic slot allocation scheme. Each node is assigned one control slot per cycle, and all nodes have an opportunity to transmit a control and data packet during each cycle. The slot allocation is static and does not change with the load. This protocol has the advantage that the cycle length is proportional to the length of a control packet.

A control cycle consists of $m$ control slots as shown in Fig. 4. Every node has an assigned control slot which it used to reserve access on a data bus if backlogges. In Fig. 4, node $P$ transmits a control packet in control slot $T_p$. The control slot includes the time required for the source node to check the status tables, build and transmit the control packet. Collisionless transmission is achieved by this protocol through the use of status tables. Each node maintains two tables: a table to track the status of the data buses to eliminate data bus collision, and a table to avoid destination conflict by tracking the status of the receiver of *data bus* I/O at each node. This is why each node has a *control bus* I/O *parked* on the control bus: all trans-

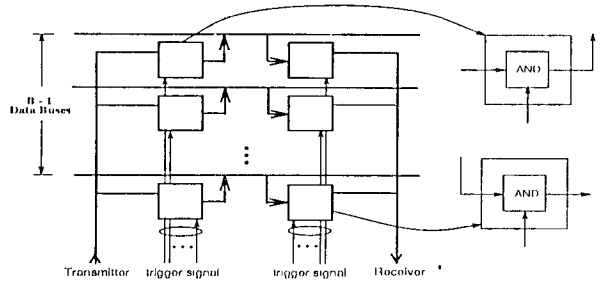


**Fig. 3.** A decentralized crossbar-based switch system.

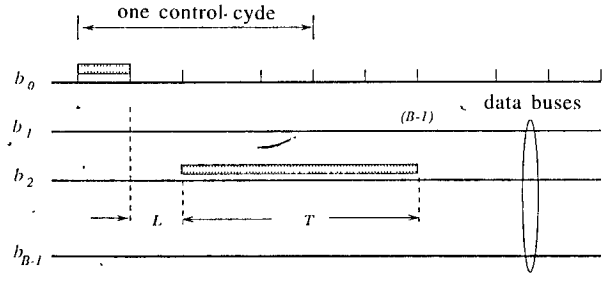


**Fig. 4.** Time-space diagram for the TDMA-C access protocol illustrating access to the control bus and transmission on the data buses.

mitted control packets are received by all nodes. *Control bus* I/O updates the two status tables at the end of each control slot after receiving and decoding a control packet. If node $m_j$ transmits a control packet targeting $m_i$ on data bus $c_k$, all nodes add $T$ against entry $i$ in their node status table and entry $k$ in the bus status table. The entries indicate the number of time slots during which the resources will be busy. All positive entries of each table are decremented at the end of every control time slot to update the remaining busy control slots.

A backlogged node checks its status tables at the beginning of its preallocated time slot. If the target node has a status table entry equal to 0, it is considered idle. If the target node is idle, the transmitter then checks for any available data bus. A data bus is considered idle if its status table entry is equal to 0. The control packet is then formed with the source, target, selected data bus, and packet length identifiers. If a node is not backlogged, its control slot remains idle during that cycle. In case the target is busy or an idle data bus is not available, the transmitter waits until the next cycle to attempt transmission.

## 4. Cache Coherence Protocol

This section describes the cache coherence protocol being studied in this paper. First, the definitions of the states of cache and memory blocks are presented, followed by a description of the protocol. There has been no attempt to reduce the size of the tables for cache coherence protocols since the improvements suggested in Gupta *et al.* (1990) O'Krafka and Newton (1990) which reduce storage requirements without a significant reduction in performance are applicable here.

### *A. Definitions*

The cache coherence protocols are sectored and have fine granularity of invalidation, at the line rather than the block level, which enable large block sizes but avoid *false sharing*. The unit of transfer is a block, made up of $k$ lines, but the unit of invalidation is a single line. This allows us to take advantage of spatial locality and alleviate the problem of *false sharing*. In the description to follow, $m_r$ denotes the *requesting node*, which originates the memory request, $m_o$ is the *owner node*, which is responsible for management of the requested block, and $m_d$ is the node which holds the *dirty copy* of the requested block. Write permission is only given on the specifically requested line and not on the whole block at once. This avoids invalidation of the entire block so other nodes which may have a copy of this block in their extended caches may use the remaining (valid) portion of the block.

### *B. Tag Organization*

This section defines the organization of the cache tag and owner memory entry. The set of states which a block residing in extended cache or memory may take is also defined.

**Cache Tag.** A *Cache Tag* (CT), shown in Fig. 5(a), is used to maintain details of blocks in EC. The subscript $l$ is used to denote location, which in this case could be $l \in \{r, d\}$, and $o$ is used to denote owner memory. For each block in EC, the following bit vectors are defined:

$V_l[i]$ - Valid Bit of line $i$, $1 \le i \le k$. Indicates if line $i$ within a block is valid at that node.

$M_l[i]$ - Modified Bit of line $i$, $1 \le i \le k$. Indicates if write permission of line $i$ within a block has been granted to that node.

The CT at each EC tracks the status of all resident blocks (and their lines). The status of each individual line is maintained to provide finer granularity of invalidation: an invalidation signal only causes the line modified at some other node to be marked invalid rather than the entire block. The CT may show a line within a block to be in one of the following three states:

**valid:** the block is resident in the EC, and the required cache line is valid and unmodified ($V_r[i]=1$ and $M_r[i]=0$).

**invalid:** the block is not resident, or the block is resident, but the required cache line is invalid ($V_r[i]=0$ and $M_r[i]=0$).

**dirty:** the block is resident, and the required cache line is valid and modified ($V_r[i]=1$ and $M_r[i]=1$). This state indicates that this copy is the only valid version of the line in the entire multiprocessor.

**Owner Memory Organization.** Each node is assigned a fraction of the global virtual address space to manage. A node maintains an *Owner Memory* (OM), as shown in Fig. 5(b), to track the state of all blocks under its control that are resident in the physical memory of the system. A block may be resident in many extended caches at different nodes but will reside in the global physical memory in a maximum of one location (in the

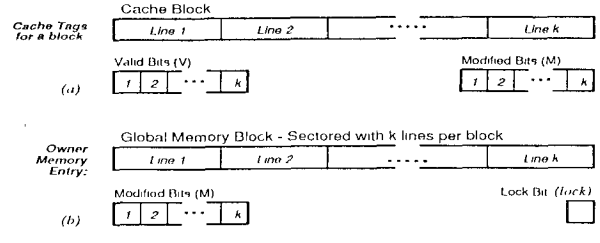


**Fig. 5.** Tag scheme for the sectored cache coherence protocol (a) Cache tag format for the extended cache, and (b) Owner memory format.

LGM of the owner) due to the memory allocation scheme (Hwang and Dowd, 1995).

The entries of the owner memory tables for a particular block are defined as follows:

$M_o[i]$ - Modified Bit of line $i$, $1 \le i \le k$, indicates whether the owner has granted write permission of that line to some node.

*lock* - Identifies an operation is taking place on this block for the single dirty block version. All lines of that block share the one lock bit, and the *loc* field shows the location of the dirty block.

A memory block can be in one of the following states:

**uncached:** the memory block is not cached at any node and *lock*=0.

**shared:** the memory block is present in the extended caches of one or more nodes and $M[i]=0$ for all $1 \le i \le k$.

**dirty:** permission to modify some element of the block has been granted. The block is present in one cache in the modified state. The lines that have been granted write permission to the dirty node have been marked invalid at all other nodes which also hold a copy of this block in their extended caches.

**locked:** the memory block is currently being modified, and the lock bit is set.

The snooping-based protocol investigated in this paper employs a *write-invalidate* policy (Chu and Dowd, 1992). The action that takes place following a read miss and write miss depends on the media access protocol TDMA-C. Requests for blocks and permissions for writing onto blocks are transmitted on the control bus. Monitoring the control bus insures proper sequencing of requests and eliminates much of the overhead required to maintain coherence. Responses are sent back on the data buses after sending a control packet informing the requesting node on which data bus to expect the requested block.

The cache coherence protocol defines the action that has to be taken for the cases of a read hit, a read miss, a write hit, and a write miss. In the case of a virtual memory miss where the block is not in physical memory, it is brought from virtual memory directly to the owner. The following assumes that node $m_r$ requires line $i$ of a particular block:

**Read Hit:** The required line is present in EC and is valid.

**Read Miss:** The required line is not available in EC ($V_r[i]=0$). The action of $m_o$ depends on the state of the **block:**

**uncached block:** the requested block is sent to $m_r$ by $m_o$. The state of the block of the CT is set to valid, $V_r[j]=1$ and $M_r[j]=0$ for all $1 \le j \le k$, and the modified bits of the OM are all reset.

**shared block:** same as for uncached.

**dirty:** If $M_o[i]=1$, $m_d$ responds by sending a copy of the requested block to $m_r$. $m_d$ is the only node which has write permission to that block so that all of its lines are valid. In this case, $m_d$ will forward its valid lines of the block to $m_r$ and $m_o$. The valid bits are all set, and the modified bits are all reset for both CT entries.

**Write Hit:** The line is present in the EC in the dirty state.

**Write Miss:** The line is present in the EC in the valid or invalid state. The request is processed as for a read miss with the difference that the requesting node gets permission to write into the block. The dirty node responds on its own without a request from the owner and reset $V_d[i]$ and $M_d[i]$.

# III. Multiple-Bus System Model

This section defines the model used to evaluate the multiple-bus system performance. The model, based on a semi-Markov process, is used to determine the impact of the media access protocol incorporating the cache coherence protocol on system performance. The model predicts the behavior of a process that resides at each of the $m$ nodes.

## 1. Model Assumptions

The multiple-bus system is described in terms of normalized time. Time is normalized to the average time between successive EC accesses. Since the nodes in the system are assumed to possess an on-chip processor cache, the basic time unit could be viewed as the average time between processor cache misses. The memory access time of a block from LGM or NLGM is assumed to be $M$ time units. The time needed to transmit a block from a source to a target node, not including any queueing or overhead due to the media access protocol, is denoted as $T$.

The behavior of the system is described in terms of the state of the process executing at a local processor. The process is viewed as residing at the processor (extended cache hit), local memory (LGM hit), the communication network during transfer, or external access (NLGM hit). System operations will be characterized by the following assumptions:

$A_1$: *i.i.d.* behavior of the nodes.

$A_2$: Each processor submits a global memory request when an extended cache miss occurs with a hit ratio or $\alpha$.

$A_3$: The probability that more than one node requests arrival or release of a memory module or a data bus between time $t$ and time $t+\Delta t$ is $o(\Delta t)$.

$A_4$: Packet-switched system. A source node sends a request message (a single packet) to the target node requesting a copy of a particular memory block. The target node receives the packet, decodes the request, accesses its local memory to satisfy the global memory request, and then sends the requested data to the source node via a single packet along the multiple-bus network.

$A_5$: Random selection for service (RSS) queue discipline for network and memory access. When more than one request is queued for a resource, the requests are selected for service in random order independent of the time of arrival.

$A_6$: No context switching. A processor waits for the response to an external request rather than context switching.

$A_7$: The memory access time is $M$ time units. Access to memory is slotted on unit time. Time is slotted for network access based on the block transmission time $T$. (There is an initial synchronization delay for network access to align to the packet slot boundaries.)

$A_8$: Request and response packets have the same size.

The following section derives a performance model of the proposed system based on a semi-Markov model. A number of performance metrics can be derived from the analytical model. In particular, we are interested in memory module utilization, data bus utilization, and average transaction time per access. Table 1 summarizes the terms used in this paper.

## 2. The Semi-Markov Model

A semi-Markov process is developed to approximate the behavior of a node. The state diagram of the model is depicted in Fig. 6. Note that the self-loops in this model could be eliminated by specifying the average sojourn time as the mean of the appropriate distribution. However, the self-loops have been retained since we feel they aid in clarity and add little additional complexity.

**Table 1.** Summary of Notation

| | |
|---|---|
| $m$: | number of nodes in system |
| $m_i$: | denotes a specific node, where $0 \le i \le m-1$ |
| $B$: | number of buses in the multiple-bus network |
| $b_0$: | control bus in TDMA-C |
| $b_i$: | data bus $i$, $1 \le i \le (B-1)$ for TDMA-C |
| $M$: | the memory access time |
| $L$: | propagation delay from source node to destination node |
| $T$: | ratio of length of data packet to length of control packet |
| $S_{yz}$: | process state, where $y \in \mathcal{Y}$ and $z \in \mathcal{Z}$ |
| $\tau_{yz}$: | sojourn time in state $S_{yz}$ |
| $v_{yz}$: | limiting probability of being in state $S_{yz}$ in the embedded Markov chain |
| $P_{yz}$: | limiting probability of being in state $S_{yz}$ in the semi-Markov process |

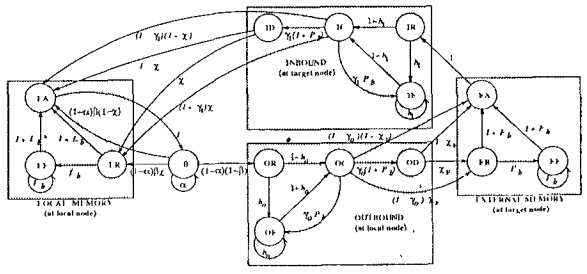

**Fig. 6.** State diagram of the semi-Markov model for the multiple-bus system.

### A. State Definitions

The states of the semi-Markov process define the behavior of a node in the system. In addition to $S_0$, the active state, $S_{yz}$ denotes a state with an average sojourn time of $\tau_{yz}$, where $y \in Y$ and $z \in Z$. The two sets, $Y$ and $Z$, are defined as follows. Let $Y=\{L,E,O,I\}$, where $L$, $E$, $O$, and $I$ denote local memory access, external memory access (either owner or dirty memory), outbound memory request via the network, and inbound memory response via the network, respectively. Let $Z=\{R,F,C,D\}$, with the elements defined as residual wait ($R$), full wait ($F$), access resource to control bus ($C$), and data buses ($D$).

As shown in Fig. 6, states often appear as triples: the access state, the full wait state, and the residual wait state. The residual wait state represents a synchronization delay: the delay until the beginning of the next access cycle of the resource. The full wait state represents queueing (with a RSS service discipline) for the resource, once the initial residual time has been met, until access is obtained.

The traffic is highly dependent on the actions of the cache coherence protocol. There are two types: *requests* and *responses*. A *request packet* is generated when the local node needs the target's data when Read Miss or Write Miss occurs. The *request packet* is formed as a control packet and is transmitted through the control bus in a preassigned control slot. A *response packet* can be either the requested data or the *write permission* in response to a request packet received by the local node. The *response packet* is formed as a control packet and a data packet if it is the required data. The data packet can be transmitted immediately after the control packet is transmitted successfully. The *response packet* is formed as a control packet if it is the *write permission* and is transmitted

**Table 2.** Transition Probabilities Based on Trace-Based Simulation

| | |
|---|---|
| $\alpha$: | hit ratio. This represents the portion of the actions in the cache coherence protocol: Read Hit and Write Hit. |
| $\beta$: | the probability that the request circulates in the local memory given the that cache miss occurs: $m_r=m_o$ for uncached and shared cases, $m_r=m_d$ for dirty case in Read Miss, and $m_i=m_o$ for uncached, $m_i=m_d$ for dirty case in Write Miss. |
| $\gamma_O$: | the probability that the local node sends both the control packet and data packet to respond to the external node request through outbound network $1-\gamma_O$ includes the actions: $m_r\neq m_o$ for uncached and shared cases, $m_r\neq m_d$ for dirty case in Read Miss, and shared case, $m_i\neq m_o$ for uncached case, and $m_r\neq m_o$ for dirty case in Write Miss. |
| $\gamma_I$: | the probability that the local node receives both the control packet and data packet from the external node through inbound network It includes the actions: $m_r\neq m_o$ for uncached and shared cases, and $m_r\neq m_d$ for dirty case in Read Miss. |

through the control bus in a preassigned control slot. Table 2 describes the detail actions of the cache coherence protocol. The state definitions are:

$S_0$: *Processor active (in cache).* The process enters $S_0$ with hit ratio $\alpha$ and remains there for a duration of time equivalent to the time between extended cache accesses. This state contains the actions of Read Hit and Write Miss. A global memory request (EC miss) is modeled by leaving $S_0$. The destination state depends on the state and location of the requested memory.

$S_{LR}$: *Local residual wait for memory.* The process enters $S_{LR}$ when the local request is blocked due to a currently busy local memory based on $A_2$. The request waits for the residual memory service time before re-attempting access to the local memory. The local request in $S_{LR}$ contends for access with (non-local) requests that are either in residual or full wait when the residual memory service time is completed. The local request moves to $S_{LA}$ if the request succeeds in accessing the local memory, or $S_{LF}$ (full wait) if it loses arbitration with a non-local request that was either in full or residual waiting.

$S_{LF}$: *Local full wait for memory.* The process enters $S_{LF}$ when LGM access is blocked at the beginning of a memory service cycle. This occurs when a newly released memory is requested simultaneously by the local node and at least one external request that was already queued for waiting access *and* the local node loses the arbitration.

$S_{LA}$: *local memory access.* The process enters $S_{LA}$ if the local memory is idle (or arbitration is won) and the data required by an EC miss is contained in local memory (LGM hit). The process remains in $S_{LA}$ for a duration equivalent to the service time of memory. From $S_{LA}$, the process returns to $S_0$ and resumes its active state.

$S_{OR}$: *Outbound residual wait for control slot.* The process enters $S_{OR}$ to synchronize the preassigned control slot. The processor waits for the residual control cycle before attempting control bus access. The process then enters $S_{OC}$ if the processor succeeds in accessing the control slot; or it enters $S_{OF}$ if the request is blocked.

$S_{OF}$: *Outbound full wait for control slot.* The process enters $S_{OF}$ when a local request loses the arbitration to send the control packet. The process reattempts access at the end of the full control cycle and remains in $S_{OF}$ if unsuccessful or moves to $S_{OC}$ if it gets arbitration.

$S_{OC}$: *Outbound ready to transmit control packet.* The process enters $S_{OC}$ if the local request gets arbitration to send the control packet to a preassigned slot. Based on the cache coherence protocol, there are two cases: one is to transmit the data packet and the control packet to respond the external node requests with probability $\gamma_o$; the other is to transmit the control packet to request external node with probability $1-\gamma_o$. There is no collision in the latter case since the control bus is based on a static cyclic allocation scheme. The process enters $S_{ER}$ if the target node is busy or enters $S_{EA}$ if the target node is available. For the first case, the process enters $S_{OF}$ with blocking probability $P_b$; it enters $S_{OD}$ if the target node receiver is available, and at least one data bus is available.

$S_{OD}$: *Outbound data transmission.* The process enters $S_{ER}$ if the target node is busy; or it enters $S_{EA}$ if the target node is available.

$S_{ER}$: *External residual wait for memory.* Similar to $S_{LR}$, but the request is targeted to a non-local memory.

$S_{EF}$: *External full wait for memory.* Similar to $S_{LF}$, but the request is targeted to a non-local memory.

$S_{EA}$: *External memory access.* The process enters $S_{EA}$ if the target non-local memory is acquired after the request is successfully passed by the access protocol through the network. The process remains in $S_{EA}$ for a duration equivalent to the service time of memory. From $S_{EA}$, the process returns to $S_0$ through the network and the local MMU.

$S_{IR}$: *Inbound residual wait for control slot.* This state is similar to $S_{OR}$.

$S_{IF}$: *Inbound full wait-for control slot.* This state is similar to $S_{OF}$.

$S_{IC}$: *Inbound ready to transmit control packet.* This state is similar to $S_{OC}$. There are two cases: one is to transmit data packet and control packet to respond to the local node request with probability $\gamma_I$, and the process enters $S_{IF}$ with blocking probability $P_b$ or enters $S_{ID}$ if the local node receiver

is available and at least one data bus is available. The other case is where one of the external nodes transmits the control packet to a request local node with probability $1-\gamma_I$; the process enters $S_{LR}$ if the local node is busy or enters $S_{LA}$ if the local node is available.

$S_{ID}$: *Inbound full wait for data transmission.* This state is similar to $S_{OD}$.

The embedded Markov chain can be solved and the limiting probabilities $v_0$, $v_{yz}$ for $y \in Y$ and $z \in Z$ can be derived as functions of the transition probabilities. The limiting probabilities are obtained by solving the set of equations from

$$\overline{V} = \overline{V}\ \overline{P}$$

and

$$v_0 + \sum_{y \in Y, z \in Z} v_{yz} = 1,$$

where $\overline{V} = [v_0, v_{LR}, v_{LF}, v_{LA}, v_{ER}, v_{EF}, v_{EA}, v_{OR}, v_{OF}, v_{OC}, v_{OD}, v_{IR}, v_{IF}, v_{IC}, v_{ID}]$, and the transition probability matrix $\overline{P} =$

$$\begin{bmatrix}
\alpha & \bar{\alpha}\beta_\chi & 0 & \bar{\alpha}\beta\overline{\chi} & 0 & 0 & 0 & \bar{\alpha}\overline{\beta} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & L_b & \overline{L_b} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & L_b & \overline{L_b} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & E_b & \overline{E_b} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & E_b & \overline{E_b} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & h_O & \overline{h_O} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & h_O & \overline{h_O} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \overline{\gamma_O}\chi_E & 0 & \overline{\gamma_O\chi_E} & 0 & \gamma_O P_b & 0 & \gamma_O\overline{P_b} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \chi_E & 0 & \overline{\chi_E} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & h_I & \overline{h_I} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & h_I & \overline{h_I} & 0 \\
0 & \overline{\gamma_I}\chi & 0 & \overline{\gamma_I}\overline{\chi} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \gamma_I P_b & 0 & \gamma_I\overline{P_b} \\
0 & \chi & 0 & \overline{\chi} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

$1-\alpha=\overline{\alpha}$, $1-\beta=\overline{\beta}$, $1-\chi=\overline{\chi}$, $1-\chi_E=\overline{\chi_E}$, $1-L_b=\overline{L_b}$, $1-E_b=\overline{E_b}$, $1-h_O=\overline{h_O}$, $1-h_I=\overline{h_I}$, $1-\gamma_O=\overline{\gamma_O}$, $1-\gamma_I=\overline{\gamma_I}$, $1-P_b=\overline{P_b}$.

The limiting probabilities can be expressed as a function of $v_0$ from the state diagram and the transition matrix as follows.

For local memory access:

$$v_{LA} = (1-\alpha)v_0, \quad v_{LR} = (1-\alpha)\gamma v_0,$$

$$v_{LF} = \frac{L_b}{1-L_b}(1-\alpha)\gamma v_0.$$

For outbound bus access (from local node):

$$v_{OR} = (1-\alpha)(1-\beta)v_0,$$

$$v_{OF} = \left[\frac{1}{(1-h_O)(1-\gamma_O P_b)} - 1\right](1-\alpha)(1-\beta)v_0.$$

$$v_{OC} = \frac{1}{1-\gamma_O P_b}(1-\alpha)(1-\beta)v_0,$$

$$v_{OD} = \frac{\gamma_O(1-P_b)}{1-\gamma_O P_b}(1-\alpha)(1-\beta)v_0.$$

For external memory access:

$$v_{EA} = (1-\alpha)(1-\beta)v_0, \quad v_{ER} = \chi_E(1-\alpha)(1-\beta)v_0,$$

$$v_{EF} = \chi_E \frac{E_b}{1-E_b}(1-\alpha)(1-\beta)v_0.$$

For inbound bus access:

$$v_{IR} = (1-\alpha)(1-\beta)v_0,$$

$$v_{IF} = \left[\frac{1}{(1-h_I)(1-\gamma_I P_b)} - 1\right](1-\alpha)(1-\beta)v_0,$$

$$v_{IC} = \frac{1}{1-\gamma_I P_b}(1-\alpha)(1-\beta)v_0,$$

$$v_{ID}=\frac{\gamma_I(1-P_b)}{1-\gamma_I P_b}(1-\alpha)(1-\beta)v_0,$$

and

$$v_0=[1+(1-\alpha)[1+\frac{\chi}{1-L_b}+(1-\beta)[3+\frac{\chi_E}{1-E_b}$$

$$+\frac{1+\gamma_O-h_O\gamma_O}{(1-h_O)(1-\gamma_O P_b)}+\frac{1+\gamma_I-h_I\gamma_I}{(1-h_I)(1-\gamma_I P_b)}]]]^{-1}$$

It can be seen from the above equations that a set of simultaneous nonlinear equations must be solved. The nonlinearity is introduced because the transition probabilities are defined as functions of limiting probabilities while the limiting probabilities are defined as functions of transition probabilities. An iterative algorithm (Hostetter *et al.*, 1991) can be used to solve for $P_0$, $P_{yz}$ $y\in Y_I$ and $z\in Z$. The algorithm to compute the results is as follows: (1) Input the number of nodes and buses. (2) Input the average sojourn time in each state. (3) Choose initial values for the limiting probabilities such that $0<P_0<1$. $0<P_{yz}<1$ for all $y\in Y$ and $z\in Z$, (4) Calculate the transition probabilities. (5) Calculate the limiting probabilities based on transition probabilities. (6) Repeat steps 4-5 until $P_0+\Sigma_{y\in Y}\,{}_{z\in Z}P_{yz}$ converges to 1.0.

The average sojourn times are summarized as follows: $\tau_0=1$, $\tau_{yz}=M$ for all $y\in\{L, E\}$ and $z\in\{F, A\}$, $\tau_{LR}=\tau_{ER}=\frac{M}{2}$, and $\tau_{OF}=\tau_{IF}=m-1$, $\tau_{OR}=\tau_{IR}=\frac{m}{2}$, $\tau_{OC}=\tau_{IC}=1$, $\tau_{OD}=\tau_{ID}=T+L$.

The limiting probability of being in state $S_{yz}$, denoted as $P_{yz}$, can be expressed as:

$$P_{yz}=v_{yz}\tau_{yz}/\sum_{i\in Y,\,j\in Z}v_{ij}\tau_{ij}$$

for some $y\in Y$ and $z\in Z$ where $v_{yz}$ is the limiting probabilities of being in state $S_{yz}$ of the embedded Markov chain. The semi-Markov process considered in this paper has an irreducible embedded Markov chain with ergodic states so the above equation is always applicable. The derivation of the transition probabilities may be found in (Hwang and Dowd, 1995; Dowd and Hwang, 1995), and in Appendix A.

# IV. Performance Analysis

The model developed in the previous sections is now used to analyze the behavior of the multiple-bus system. The analytical model is validated through a comparison with simulation. The performance of the system is analyzed in terms of the following metrics: average transaction time per access, memory module utilization, and data bus utilization.

## 1. Performance Metrics

This section defines the performance metrics and can be derived from the model as follows. The memory module utilization per node is defined as

$$u_M=P_{LA}+P_{EA}.$$

The total memory utilization would then be $U_M=mu_M$. The total data bus utilization is

$$U_B=m(P_{OD}+P_{ID}),$$

and the utilization per data bus is

$$u_B=\frac{m}{B-1}(P_{OD}+P_{ID}).$$

Let $t_{avg}$ denote the average transaction time per access, the weighted sum of an EC hit and miss. There are three components: an EC hit with $\alpha$, the time for accessing its local memory with probability $(1-\alpha)\beta$, and the time for accessing the target memory with probability $(1-\alpha)(1-\beta)$. The time for accessing local memory ($T_{LM}$) and external memory ($T_{EM}$) can be obtained by applying Little's Law (Bogineni and Dowd, 1992) and includes both the waiting time and the transmission time. Let $T_{LM}$, $E[N_{LM}]$, and $\Lambda_{LM}$ denote the average time for accessing local memory, the average requests waiting at the local node's queue, and the throughput of the local memory, respectively. Then, $T_{LM}=\frac{E[N_{LM}]}{\Lambda_{LM}}$, where $\Lambda_{LM}=\frac{P_{LA}}{M}$, and $E[N_{LM}]=1+P_{EF}+P_{ER}$. Therefore,

$$T_{LM}=\frac{M(1+P_{EF}+P_{ER})}{P_{LA}}.$$

A similar argument can be applied to the time it takes to access the external memory. In this case, the process is as follows: the local node sends out the control packet through the control bus to inform the target node. The target responds in accordance with the information in the control packet sent by the local node. The average time for accessing the target memory can be broken down into four parts: $T_{OB}$, sending the control packet to target node through control bus (outbound network); $T_{TM}$, accessing target memory; $T_{IB}$, returning the data packet and the control packet through data buses and control bus (inbound network),

respectively; and $T_{LM}$, returning to local memory. $T_{OB}=\frac{E[N_{OB}]}{\Lambda_{OB}}$, where $E[N_{OB}]=1+P_{IF}+P_{IR}$, and $\Lambda_{OB}=(1-\gamma_O)\frac{P_{OC}}{\tau_{OC}}$. $T_{TM}=\frac{E[N_{TM}]}{\Lambda_{TM}}$, where $E[N_{TM}]=1+P_{LF}+P_{LR}+\frac{m-2}{m-1}(P_{EF}+P_{ER})$, and $\Lambda_{TM}=\frac{P_{EA}}{M}$. $T_{IB}=\frac{E[N_{IB}]}{\Lambda_{IB}}+T+L$, where $E[N_{IB}]=1+P_{OF}+P_{OR}+\frac{m-2}{m-1}(P_{IF}+P_{IR})$, $\Lambda_{IB}=\gamma_I(1-P_b)\frac{P_{ID}}{\tau_{ID}}$, $T$ is the data transmission time. Then,

$$T_{EM}=T+L+\frac{\tau_{OC}(1+P_{IF}+P_{IB})}{(1-\gamma_O)P_{OC}}$$

$$+\frac{M(1+P_{LF}+P_{LR}+\frac{m-2}{m-1}(P_{EF}+P_{ER}))}{P_{EA}}$$

$$+\frac{\tau_{IC}(1+P_{OF}+P_{OR}+\frac{m-2}{m-1}(P_{IF}+P_{IR}))}{\gamma_I(1-P_b)P_{IC}}$$

$$+\frac{M(1+P_{EF}+P_{ER})}{P_{LA}}. \quad (1)$$

So, $t_{avg}$ can be expressed as:

$$t_{avg}=\alpha+(1-\alpha)\beta T_{LM}+(1-\alpha)(1-\beta)T_{EM}. \quad (2)$$

## 2. Validation of the Model

This section compares the values predicted by the analytical model with simulation results based on the assumptions of Section III.1. The simulator is based on a stochastic self-driven discrete event model, written in the *C* programming language with a *C*-based library of routines that provide discrete-event and random variate facilities. A trace-based simulation was performed using the *fft64* trace, which features heavy sharing between all processors (the shared data is often referenced (Chaiken *et al.*, 1990). It is intended to stress the multiple-bus system with reservation media access protocol to give the results for a worst case scenario.

The 64 processor multiple-bus system with a 192K EC was run to assess performance. Additional assumptions are: (1) all nodes operate independently, (2) infinite queuing for each transmitter buffer, (3) a 32-bit wide bus connects the processor and EC, (4) a node will generate a new memory request immediately after the previous one is fulfilled, (5) a line is 32 bytes long, (6) the time latency of an EC cache hit is the time needed to transfer a line from EC to PC, and (7) the size of data packets on the network is fixed and equal to the block size, $\in \{1, 2, 4, 8, 16\}$ lines.

Unless otherwise noted, the following sojourn times are used in the remainder of this paper: $L$=0, propagation delay =0, $\tau_0$=1.0, which represents the average inter-arrival time between two requests to EC, the memory access time $M$=4/line, and the bus access times $T\in\{5, 10, 50\}$/line for transmission rates $\in\{100, 50, 10\}$ Mbps. The memory access time depends on the block size, memory transfer width (bytes/transfer), and the memory transfer speed (processor cycles/transfer). The bus access time is dependent on the packet size (EC block size) and data rate of the communication bus.

Steady state limiting probabilities and transaction times were measured. Simulation convergence was obtained through the replication/deletion method (Law and Kelton, 1991) with 99% confidence and less 1% variation from the mean. Four transition probabilities, $\alpha$, $\beta$, $\gamma_O$, $\gamma_I$ for different transmission rates measured from simulation with different block sizes (numbers of line sizes) and buses are shown in Table 3, and Table 4, Table 5.

Figure 7 compares the analytical model with the *fft64* trace-based simulation. The graphs in Fig. 7 show the average transaction time per access, $t_{avg}$, and the

**Table 3.** Transition Probabilities Measured from the *fft*64 Trace-Simulation for Bus Transmission Rate 100 Mbps

| block sizes | buses | $\alpha$ | $\beta$ | $\gamma_O$ | $\gamma_I$ |
|---|---|---|---|---|---|
| 1 | 2 | 0.689759 | 0.015413 | 0.258264 | 0.989871 |
| | 4 | 0.689759 | 0.015413 | 0.258264 | 0 989871 |
| | 8 | 0.689759 | 0.015413 | 0.258264 | 0.989871 |
| | 16 | 0.689759 | 0.015413 | 0.258264 | 0 989871 |
| | 32 | 0.689759 | 0.015413 | 0.258264 | 0.989871 |
| 2 | 2 | 0.731285 | 0.015700 | 0.289039 | 0.990862 |
| | 4 | 0.730695 | 0.015945 | 0.291342 | 0.990575 |
| | 8 | 0.730695 | 0.015945 | 0.291342 | 0.990575 |
| | 16 | 0.730695 | 0.015945 | 0.291342 | 0.990575 |
| | 32 | 0.730695 | 0.015945 | 0.291342 | 0.990575 |
| 4 | 2 | 0.746390 | 0 015866 | 0.299945 | 0.991086 |
| | 4 | 0.743152 | 0.015880 | 0 300447 | 0.990991 |
| | 8 | 0.742126 | 0.015963 | 0 302492 | 0.990856 |
| | 16 | 0.742126 | 0.015963 | 0.302492 | 0.990856 |
| | 32 | 0.742126 | 0.015963 | 0.302492 | 0.990856 |
| 8 | 2 | 0.748071 | 0.015975 | 0.303184 | 0.991039 |
| | 4 | 0 740034 | 0.015935 | 0 303757 | 0.991033 |
| | 8 | 0.738955 | 0.015721 | 0 303764 | 0.991027 |
| | 16 | 0.738955 | 0.015721 | 0.303764 | 0.991027 |
| | 32 | 0.738955 | 0.015721 | 0.303764 | 0.991027 |
| 16 | 2 | 0.836895 | 0.013374 | 0.413994 | 0.996429 |
| | 4 | 0.841568 | 0.015357 | 0.417773 | 0.996171 |
| | 8 | 0 841638 | 0.015890 | 0.417773 | 0.995948 |
| | 16 | 0.841638 | 0.015890 | 0.417773 | 0.995948 |
| | 32 | 0.841638 | 0.015890 | 0.417773 | 0.995948 |

**Table 4.** Transition Probabilities Measured from the *fft64* Trace-Simulation for Bus Transmission Rate 50 Mbps

| block sizes | buses | $\alpha$ | $\beta$ | $\gamma_0$ | $\gamma_1$ |
|---|---|---|---|---|---|
| 1 | 2 | 0.691233 | 0.015317 | 0.258793 | 0.990216 |
| | 4 | 0.691233 | 0.015317 | 0.258793 | 0.990216 |
| | 8 | 0.691233 | 0.015317 | 0.258793 | 0 990216 |
| | 16 | 0.691233 | 0.015317 | 0.258793 | 0 990216 |
| | 32 | 0.691233 | 0.015317 | 0.258793 | 0.990216 |
| 2 | 2 | 0.734553 | 0.015569 | 0.289001 | 0.990722 |
| | 4 | 0.729466 | 0.016024 | 0.289355 | 0.990449 |
| | 8 | 0.729466 | 0.016024 | 0.289355 | 0.990449 |
| | 16 | 0.729466 | 0.016024 | 0.289355 | 0.990449 |
| | 32 | 0.729466 | 0.016024 | 0.259355 | 0.990449 |
| 4 | 2 | 0.750436 | 0.015438 | 0.298418 | 0 991216 |
| | 4 | 0.743078 | 0.015565 | 0.300562 | 0.991169 |
| | 8 | 0.741916 | 0 015758 | 0.300660 | 0.991095 |
| | 16 | 0.741916 | 0.015758 | 0.300660 | 0.991095 |
| | 32 | 0 741916 | 0.015758 | 0.300660 | 0.991095 |
| 8 | 2 | 0.741833 | 0.014760 | 0 300585 | 0.991872 |
| | 4 | 0.740943 | 0.015837 | 0.304183 | 0.991047 |
| | 8 | 0.740600 | 0 015902 | 0.304250 | 0.990936 |
| | 16 | 0 740600 | 0 015902 | 0.304250 | 0.990936 |
| | 32 | 0 740600 | 0 015902 | 0 304250 | 0.990936 |
| 16 | 2 | 0.827273 | 0.026477 | 0.392341 | 0.992657 |
| | 4 | 0.841081 | 0.015577 | 0.414072 | 0.995921 |
| | 8 | 0.841272 | 0.015459 | 0.416265 | 0 996353 |
| | 16 | 0.841272 | 0.015459 | 0.416265 | 0 996353 |
| | 32 | 0.841272 | 0.015459 | 0 416265 | 0.996353 |

**Table 5.** Transition Probabilities Measured from the *fft64* Trace-Simulation for Bus Transmission Rate 10 Mbps

| block sizes | buses | $\alpha$ | $\beta$ | $\gamma_0$ | $\gamma_1$ |
|---|---|---|---|---|---|
| 1 | 2 | 0.696314 | 0.015927 | 0.252792 | 0.989693 |
| | 4 | 0.696314 | 0.015927 | 0.252792 | 0.989693 |
| | 8 | 0.696314 | 0.015927 | 0.252792 | 0.989693 |
| | 16 | 0.696314 | 0.015927 | 0.252792 | 0.989693 |
| | 32 | 0.696314 | 0.015927 | 0.252792 | 0.989693 |
| 2 | 2 | 0.728091 | 0.015160 | 0.285242 | 0.990994 |
| | 4 | 0 729739 | 0 015883 | 0.288229 | 0.990627 |
| | 8 | 0.729739 | 0.015883 | 0.288229 | 0.990627 |
| | 16 | 0.729739 | 0.015883 | 0.288229 | 0.990627 |
| | 32 | 0.729739 | 0.015883 | 0.288229 | 0.990627 |
| 4 | 2 | 0.748139 | 0.013994 | 0.296477 | 0.992060 |
| | 4 | 0.748217 | 0 015720 | 0.301380 | 0.991163 |
| | 8 | 0.748343 | 0.015844 | 0.300757 | 0.990939 |
| | 16 | 0.748343 | 0.015844 | 0.300757 | 0.990939 |
| | 32 | 0.748343 | 0 015844 | 0.300757 | 0.990939 |
| 8 | 2 | 0.742280 | 0.013513 | 0.302892 | 0.992443 |
| | 4 | 0.742864 | 0.015718 | 0.305301 | 0.991285 |
| | 8 | 0.743802 | 0.015677 | 0 304067 | 0.991213 |
| | 16 | 0.743802 | 0.015677 | 0.304067 | 0.991213 |
| | 32 | 0.743802 | 0.015677 | 0.304067 | 0.991213 |
| 16 | 2 | 0.829172 | 0 031960 | 0.396925 | 0.996375 |
| | 4 | 0.840299 | 0.015400 | 0.414366 | 0.996225 |
| | 8 | 0.840787 | 0.015320 | 0.415592 | 0 996127 |
| | 16 | 0 841255 | 0.015261 | 0.416615 | 0.996022 |
| | 32 | 0.841255 | 0.015261 | 0.416615 | 0.996022 |

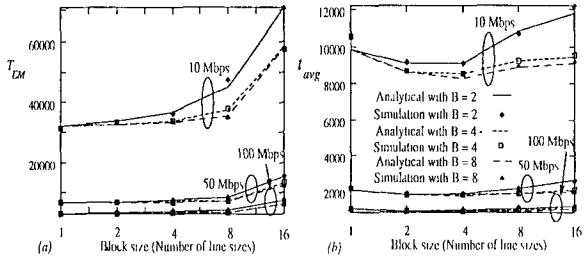


**Fig. 7.** Comparison of the analytic model to *fft64* trace-based simulation with varying block sizes$\in \{1, 2, 4, 8, 16\}$ times of line size. Points and lines represent simulation and analytic model values, respectively. (a) $T_{EM}$, (b) $t_{avg}$.

access external memory given cache miss, $T_{EM}$, with varying block sizes. Points and lines represent simulation and analytical model values, respectively.

The maximum deviation between the analytical and simulation model is shown in Fig. 7 to be less than 3.5% and 7.5% for $T_{EM}$ and $t_{avg}$ at block size = 8 times the line size and block size = 1 times the line size, respectively. The results show a high degree of correlation between the simulation and analytical models. The graphs show how system performance is impacted by variations in buses and block sizes. In general, the graphs show that $T_{EM}$ increases when the block size increases due to the problem of *false sharing* (Gupta and Weber, 1992; Chu and Dowd, 1992; O'Krafka and Newton, 1990), or because the number of buses is decreased, and that $t_{avg}$ depends on the block size and the number of buses. When the block size =4 times the line size, $t_{avg}$ is minimized for three different transmission rates. The results imply that the block size =4 times the line size is the best choice to get better performance in this multiple-bus computer system.

Based on the simulation and analytical results, there are no deviations for $T_{EM}$ and $t_{avg}$ in the 64 node system when the number of buses is greater than 8. This shows that the multiple-bus system with a reservation media access protocol can achieve high scalability of the system compared with (Lang and Alegre, 1982). The following sections investigate the performance of the multiple-bus system by varying the parameters based on the *fft64* trace in Section IV.2. Since some metrics have the same trend, we will show a typical metric and explain the other metrics. Unless otherwise stated, block size =4 time the lines with transmission rate 100Mbps is used in the following sections.

## 3. Variation in Hit Ratio $\alpha$

This section investigates the impact of the hit ratio $\alpha$ on system performance and fixes the other parameters shown in Table 3. Figure 8 shows the performance of the multiple-bus system by varying the hit ratio $0.8 \leq \alpha \leq 1.0$ and the number of buses. The metrics show that $t_{avg}$, $u_B$, and $U_M$ are all decreased when $\alpha$ is increased. Figure 8(a) shows the average transaction time per access, $t_{avg}$. $t_{avg}$ is decreased by 8.2% when the number of buses increases from 4 to 8 at $\alpha$=0.8, and decreased by 4.6% when the number of buses increases from 2 to 4 at $\alpha$=0.8, respectively. $T_{EM}$ has the same trend as $t_{avg}$. Figure 8(b) shows the utilization per data bus, $u_B$. The data bus utilization is relaxed by 52% when the number of buses increases from 4 to 8 at $\alpha$=0.8, and by 30% when the number of buses increases from 2 to 4 at $\alpha$=0.8. Figure 8(c) shows the total memory utilization, $U_M$. The total memory utilization increases 6.5% when the number of buses increases from 2 to 4 at $\alpha$=0.8, and 12.5% when the number of buses increases from 4 to 8 at $\alpha$=0.8.

## 4. Variation in $\beta$

This section discusses the impact on system performance of varying the conditional probability that requested data in the external node given the cache miss occurs, $\beta$.

Figure 9 shows the performance of the multiple-bus system by varying $\beta$ and the number of buses. The direct impact on system performance is $U_M$ and $u_M$. $U_M$ or $u_M$ increases when $\beta$ is increased, and the rest of the metrics decrease when $\beta$ is increased. Figure 9(a) shows that the average transaction time per access, $t_{avg}$, is an inverse linear proportional relationship by varying $\beta$ from 0.0 to 1.0. $t_{avg}$ is decreased 9% when the number of buses is changed from 2 to 4 at $\beta$=0, and 2% when the number of buses 4 to 8 at $\beta$=0. $T_{EM}$ shows the same result as $t_{avg}$. Figure 9(b) shows the utilization per data bus, $u_B$, with varying $\beta$. $u_B$ is decreased when $\beta$ increases. $u_B$ decreases 36% when the number of buses is changed from 2 to 4 at $\beta$=0 and 57% when the number of buses is changed from 4 to 8 at $\beta$=0, respectively.

Figure 9(c) shows the total memory utilization, $U_M$, by varying $\beta$. When $\beta$ increases, the total memory utilization, $U_M$, increases. $U_M$ is sensitive to the variation of $\beta$ when $\beta$ approaches 1.0. $U_M$ is 48 when $\beta$=1.0. The memory module utilization per node, $U_M$, has the same trend as $U_M$. $U_M$ is 0.69 when $\beta$=1.0.

## 5. Variation in $\gamma_O$ and $\gamma_I$

This section investigates the effect of the variation in $\gamma_O$ and $\gamma_I$ on the multiple-bus system performance. Figure 10 shows the multiple-bus system performance when $\gamma_O$ is varied. Figure 10(a) shows the average time per access, $t_{avg}$, when $\gamma_O$ is varied. $t_{avg}$ is increased when $\gamma_O$ increases since the higher $\gamma_O$ is, the less possibility there is that the local request can be sent out. $t_{avg}$ decreases 10% when the number of buses increases from 2 to 4 at $\beta$=0, and decreases 4% when the number of buses increases from 4 to 8 at $\beta$=0, respectively. $T_{EM}$ also shows the same results as $t_{avg}$. Figure 10(b) shows the utilization per data bus, $u_B$. The higher $\gamma_O$ is, the more data traffic there is on the bus. On the contrary, $U_M$ in Fig. 10(c) shows a different trend: The more time is spent on a bus, the less time is spent on memory, relatively.

Figure 11 shows the multiple-bus system performance when $\gamma_I$ is varied. Figure 11 shows results different from those in Fig. 10 for $T_{EM}$, $t_{avg}$, and $U_M$.

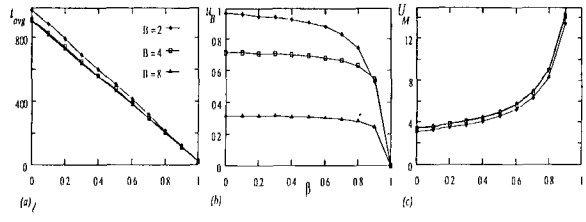

**Fig. 9.** The performance of the multiple-bus system while varying the $0.0 \leq \beta \leq 1.0$ and the number of buses. (a) The average transaction time per access, $t_{avg}$. (b) The utilization per data bus, $u_B$. (c) The total memory utilization, $U_M$.

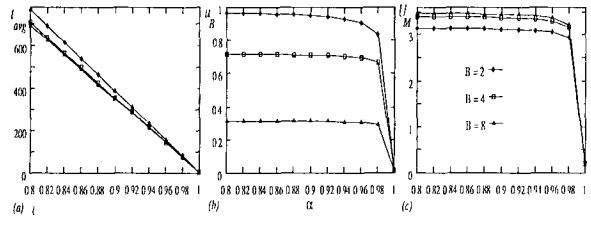

**Fig. 8.** The performance of the multiple-bus system while varying the hit ratio $0.8 \leq \alpha \leq 1.0$ and the number of buses. (a) The average transaction time per access, $t_{avg}$ (b) The utilization per data bus, $u_B$. (c) The total memory utilization, $U_M$.

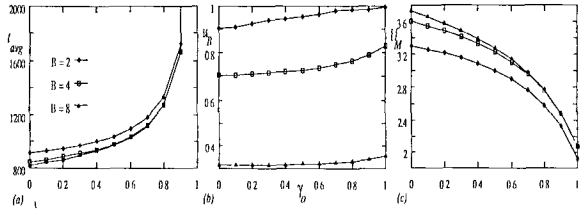

**Fig. 10.** The performance of the multiple-bus system while varying the $0.0 \leq \gamma_O \leq 1.0$. (a) The average transaction time per access, $t_{avg}$. (b) The utilization per data bus, $u_B$. (c) The total memory utilization, $U_M$

The higher $\gamma_l$ is , the greater is the possibility that the data can be sent back. Figure 11(a) shows the average transaction time per access, $t_{avg}$, when $\gamma_l$ is varied. When $\gamma_l$=0, $t_{avg}$ approaches infinity, and there is no possibility that the data can be sent back. Figure 11(b) shows the same results as Fig. 10(b). The more traffic there is on the bus, the higher is bus utilization, $u_B$. Figure 11(c) shows results similar to those in Fig. 10(c).

## 6. Variation in Memory Service Time and Data Bus Service Time

This section shows the impact of varying memory service time and data bus service time on multiple-bus system performance. Figure 12 shows the performance of the multiple-bus system while varying the memory service time $\in \{0.5, 1.0, 2.0\}M$. Figure 13 shows the performance of the multiple-bus system while varying the data bus service time $\in \{0.5, 1.0, 2.0\}T$. Figures (a), (b), and (c) show the average transaction time per access, $t_{avg}$, the utilization per data bus, $u_B$, and the memory module utilization per node, $U_M$, respectively.

Both figures show the same results for $t_{avg}$ but show different trends for $u_B$ and $U_M$. The more service time there is in memory, the greater is the memory module utilization; the more service time there is in data bus, the greater is utilization of the data bus.

## 7. The impact of Propagation Delay, *L*

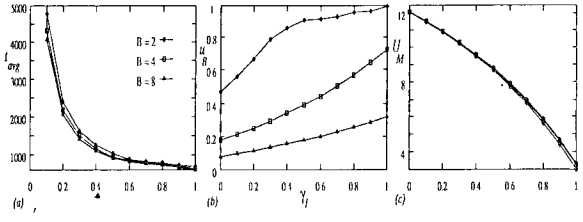

**Fig. 11.** The performance of the multiple-bus system while varying the $0.0 \leq \gamma_l \leq 1.0$. (a) The average transaction time per access, $t_{avg}$ (b) The utilization per data bus, $u_B$. (c) The total memory utilization, $U_M$.

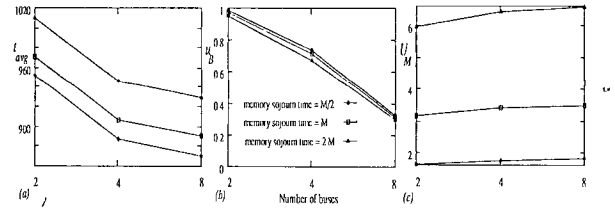

**Fig. 12.** The performance of the multiple-bus system while varying the memory service time$\in \{0.5, 1.0, 2.0\}M$ and the number of buses. (a) The average transaction time per access, $t_{avg}$. (b) The utilization per data bus, $u_B$. (c) The memory module utilization per node, $U_M$

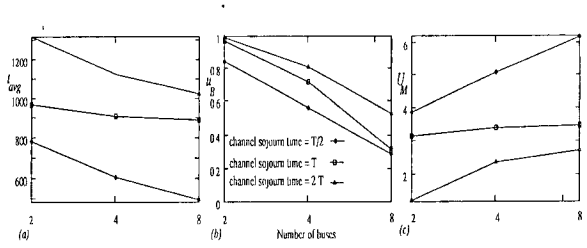

**Fig. 13.** The performance of the multiple-bus system while varying the data bus service time$\in \{0.5, 1.0, 2.0\}T$ and the number of buses. (a) The average transaction time per access, $t_{avg}$. (b) The utilization per data bus, $u_B$. (c) The memory module utilization per node, $U_M$.

This section discusses the impacts of propagation delay, $L$. All the performance metrics can be obtained by replacing the sojourn time $T+L$ instead of $L$. This shows the significant advantage of the semi-Markov model, which increases flexibility without increasing model complexity. The results show the same trend as shown before. The only differences in performance metrics are: $t_{avg}$ and $u_B$ are increased, and $U_M$ is decreased since the sojourn time in transmission state is increased.

# V. Conclusion

This paper has considered a reservation media access protocol to support DSM multiprocessor multiple-bus interconnection networks. A media access control (TDMA-C) and a *snooping* cache coherence protocol are incorporated to provide access to a distributed arbitration of the multiple-bus system. A parallel I/O system and a decentralized crossbar switch system are proposed to support the operations of media access protocol. Trace-based discrete-event simulation has been used to evaluate the performance. A semi-Markov model has been developed to study the performance of the multiple-bus system. The performance model was developed to reflect the impact of low-level media access issues on high-level distributed shared memory system performance, enabling the model to predict the performance impact with a media access control protocol and system parameters in the underlying multiple-bus system. The results show that $T_{EM}$ increases when the block size increases due to the problem of *false sharing*, and that $t_{avg}$ is minimized when the block size =4 times of the size for three different transmission rates, thus enhancing the scalability of the multiple-bus system. The performance was also compared for variation in the number of buses , the block size, memory service time, bus

service time, and the others parameters, $\alpha$, $\beta$, $\gamma_O$, and $\gamma_I$.

## A. Transition Probability Derivations

The derivation of the transition probabilities of Fig. 6 is based on the behavior of one specific node, referred to as the *local node*. The terms *external, target (owner, dirty)* are used to refer to any node in the system other than the local node.

Assumption $A_2$ states that the probability of more than one request arriving at the same time and the probability of multiple requests arriving at the instant of a release of memory or channel is negligible. This implies that the transition probabilities $p[0, LF]$, $p[0,OF]$, $p[OD,EF]$, $p[EA,IF]$ and $p[ID,LF]$ are negligible, where $p[ij,kl]$ denotes the transition probabilities between states $S_{ij}$ and $S_{kl}$.

Transition probabilities $p[0,OC]$ and $p[EA,IC]$ depend on the media access protocol. Based on assumptions $A_2$ and $A_6$, nodes access the channels based on time slotted to packet transmission boundaries so $p[0,OC]=0$ and $p[EA,IC]=0$.

There are two categories of transition probabilities in this paper: one category of transition probabilities shown in Table 2 depends on a trace-based simulation, such as $\alpha$, $\beta$, and $\gamma_X$, $X\in\{O,I\}$; the other category of transition probabilities between the states of the semi-Markov model can be expressed in terms of the limiting probabilities of the states $X_b$, $X\in\{L, E\}$, $h_X$, $X\in\{O, I\}$, $\chi$, $\chi_E$, and $P_b$.

Let $L_b$ denote the probability of losing arbitration for the local memory when other requests are currently waiting. The probability that one of the other $(m-1)$ external nodes has been prevented from making an external access to the LGM at the local node is $\frac{1}{m-1}[P_{EF}+P_{ER}]$ since an external request is equally likely to target any other node with equal probability. Considering all $(m-1)$ external nodes, $P_{EF}+P_{ER}$ is the average number of requests from external nodes waiting to access local memory. The probability of a local node losing arbitration for its LGM is:

$$1-L_b=\frac{1}{1+P_{EF}+P_{ER}}. \tag{3}$$

Let $E_b$ denote the probability of losing arbitration for the external node when other requests are currently waiting:

$$1-E_b=\frac{1}{1+P_{LF}+P_{LR}+\frac{m-2}{m-1}(P_{EF}+P_{ER})}, \tag{4}$$

where $P_{LF}+P_{LR}$ represents the likelihood that the processor of the target node is waiting to access its local memory, and $\frac{P_{EF}+P_{ER}}{m-1}$ is the average number of requests of one of the other $(m-2)$ external nodes waiting for the target memory.

If $\chi$ denotes the probability that the local memory is already busy when the local processor tries to access its LGM, then $\chi=P_{EA}$ since the probability that a request from one of the $(m-1)$ external nodes is being processed at the local node is $\frac{P_{EA}}{m-1}$:

$$\chi=P_{EA}. \tag{5}$$

Let $\chi_E$ denote the probability that the target NLGM is already busy when a request from the local processor arrives at the target node:

$$\chi_E=P_{LA}+\frac{m-2}{m-1}P_{EA}, \tag{6}$$

where $P_{LA}$ is the probability that the external processor is accessing its local memory, and $\frac{P_{EA}}{m-1}$ is the probability that one of the other $(m-2)$ external nodes is currently accessing the target memory.

**TDMA-C Transition Probabilities:** Let $h_O$ denote the probability that the local request loses access to the control slot (outbound) either from residual or full wait. Access can only be gained from the full or residual wait states. Let $\phi_O$ be the event such that the local request wins the arbitration to the control slot (outbound): $P(\phi_O)=\frac{1}{1+P_{IR}+P_{IF}}\frac{P_{IR}+P_{IF}}{m-1}$ is the probability that a response packet to one of the $m-1$ other external nodes is waiting for transmission at the local node. Since $h_O=1-P(\phi_O)$,

$$h_O=1-\frac{1}{1+P_{IR}+P_{IF}}. \tag{7}$$

Let $h_I$ denote the probability that the response packet generated at an external node that is targeted on the local node is prevented from accessing the control slot (inbound) due to either full or residual waiting. Let $\phi_I$ denote the event such that the response packet generated at the external node that is targeted on the local node wins arbitration to the control slot (outbound). Thus, $P(\phi_I)=\frac{1}{1+P_{OR}+P_{OF}+\frac{m-2}{m-1}(P_{IR}+P_{IF})}$, where $P_{OR}+P_{OF}$ represents the probability that the external node has a request packet waiting for transmission, and $\frac{P_{IR}+P_{IF}}{m-1}$ is the probability that there is a response packet

generated at the external node targeted on one of the other $m-2$ external nodes. Now $h_I$ can be expressed as $h_I=1-P(\phi_I)$, or

$$h_I = 1 - \frac{1}{1 + P_{OR} + P_{OF} + \frac{m-2}{m-1}(P_{IR} + P_{IF})}, \quad (8)$$

where $P_{OR}+P_{OF}$ represents the probability that the external node has a request packet waiting for transmission, and $\frac{P_{IR}+P_{IF}}{m-1}$ is the probability that a response packet to one of the other $(m-2)$ external nodes has caused the target node to enter the backoff state (inbound).

The probability of the transmitter being prevented from transmitting a control packet, $P_b$, is derived from the semi-Markov model (Mudge and Al-Sadoun, 1985; Bogineni and Dowd, 1992). The probability of total blocking depends on three factors: the probability that the target node is currently receiving a data packet on a data channel reserved in a previous control cycle by a source node other than the node being considered; the probability that the target node is busy receiving a data packet reserved by a different node during the current control cycle; and the availability of a data channel.

The probability that the target node is busy receiving a data packet from a node on a data channel reserved in the previous control cycle is computed as follows. The probability of leaving state $S_i$ is $\xi_i=\frac{P_i}{\tau_i}$ as shown in Mudge and Al-Sadaun (1995). $P_{OD}$ and $P_{ID}$ are the probabilities that a node is currently transmitting a data packet to one of the other $(m-1)$ nodes. $\xi_{OD}$ and $\xi_{ID}$ are the probabilities that a node completes data packet transmission during the current control slot. $(P_{OD}-\xi_{OD})$ and $(P_{ID}-\xi_{ID})$ are the probabilities that transmission has not been completed, so the probability that a node is transmitting a data packet to the target node is $\frac{1}{m-1}(P_{OD}-\xi_{OD}+P_{ID}-\xi_{ID})$.

The probability that the target node receiver is busy, $P_{TB}$, is the probability of the target node receiving a packet from the other $(m-2)$ nodes that are currently transmitting data packet:

$$P_{TB} = \frac{m-2}{m-1}(P_{OD} - \xi_{OD} + P_{ID} - \xi_{ID}).$$

$P_T$ is the probability that the target node is available to receive a packet from the source node in the current control cycle (Bogineni and Dowd, 1992). This probability is conditional on the availability of a data channel. A node may transmit to any of the other $(m-1)$ nodes in the system. State $S_{OD}$ and $S_{ID}$ represent data packet transmission by a node and are entered from $S_{OC}$ and $S_{IC}$, respectively, with a probability $\gamma_O(1-P_b)\xi_{OD}$ and $\gamma_I(1-P_b)\xi_{OI}$, respectively. The probability that a node is currently transmitting a packet to a particular target node is given by:

$$P_T = \frac{1-P_b}{m-1}(\gamma_O\xi_{OD} + \gamma_I\xi_{ID}).$$

The probability that a source node will not transmit to a particular target node is $(1-P_T)$, and the probability that none of the $(m-1)$ nodes will transmit to the target is $(1-P_T)^{(m-1)}$. The probability that at least one out of $(m-1)$ nodes will transmit a packet to the particular target node is approximately $p=1-(1-P_T)^{(m-1)}$. The expected number of nodes that have a packet to be transmitted to the particular target node is $(m-1)P_T$. The probability, $P_{NA}$, that the target node is available to receive the packet being transmitted from the source nodes is $P_{NA}=\frac{p}{(m-1)P_T}$.

The last factor is the availability of a data channel to the source node for data transmission. The probability of a channel being available is computed conditional on the availability of the target node and the number of idle channels. Let $X(j)$ denote the probability that a channel is available to the source node for packet transmission conditional on the probability that there are $j$ idle channels in the network, and $Z(j)$ is the probability that there are $j$ idle channels in the network. Using the theorem of total probability, the conditional probability of a channel being available to the source node, $P_{CA}$, is

$$P_{CA} = \sum_{j=1}^{C-1} X(j)Z(j).$$

If there are $i$ source nodes with data packets to be transmitted, and there are $j$ idle data channels in the network, then only $\frac{\min(i,j)}{i}$ find idle channels. The probability that the source node (one of the $i$ nodes) will find an idle channel is $\frac{\min(i,j)}{i}$. The probability that $(i-1)$ other nodes out of $(m-2)$ source nodes will find idle target nodes is $\binom{m-2}{i-1}p^{(i-1)}(1-p)^{(m-i-1)}$, so the total probability of a channel being available to the source node is:

$$X(j) = \sum_{i=1}^{m-1} \frac{\min(i,j)}{i}\binom{m-2}{i-1}p^{(i-1)}(1-p)^{(m-i-1)}.$$

The probability that $j$ out of the $(C-1)$ data channels are busy is given by the binomial distribution with parameter $q$:

$$Z(j)=\binom{C-1}{j} q^{(C-j-1)}(1-q)^{j},$$

where $q$ is the probability of a channel being busy and is given by $q=\frac{m-2}{C-1}(P_{OD}-\xi_{OD}+P_{ID}-\xi_{ID})$.

The total blocking probability is then computed as

$$P_b=P_{TB}+(1-P_{TB})P_{CA}(1-P_{NA})+(1-P_{TB})(1-P_{CA}).$$

## References


Ahmadi, H. and W. E. Denzel (1989) A survey of modern high-performance switching techniques. *IEEE Journal on Selected Areas of Communications*, **7**, 1091-1103.

Vaidya, A. K. and M. A. Pashan (1988) Technology advances in wideband packet switching, *in Proc. GLOBECOM'88*, (Hollywood, FL), 668-671.

Bertoni, J. -L. Baer, and W. -H. Wang (1992) Scaling shared-bus multiprocessors with multiple buses and shared caches. A performance study. *Microprocessors and Microsystems*. 339-350.

Bisiani, R. and M. Ravishankar (1990) Plus: A distributed shared-memory system. in Proc. 17th *International Symposium Computer Architecture*, (Seattle, Washington), 115-124.

Bogineni, K. and P. W. Dowd (1992) A collisionless multiple access protocol for a wavelength division multiplexed star-coupled configuration: Architecture and performancea nalysis. *IEEE Journal on Lightwave Technology*, **10**, 1688-1699.

Bogineni, K., K. M Sivalingam, and P. W. Dowd (1993) Low complexity multiple access protocols for wavelength-division multiplexed photonic networks *IEEE Journal on Selected Areas of Communications*, **11**, 590-604.

Bogineni, K. and P W. Dowd (1992) Performance analysis of two addreess allocation schemes for an optically interconnected distributed shared memory system, in *Proc. 6th International Parallel Processing Symposium*, 562-566.

Chaiken, D., C. Fields, K. Kurihara, and A. Agarwal (1990) Directory-based cache coherence in large-scale multiprocessors *IEEE Computer*, 49-58.

Chen, P., G. Gibson, R Katz, and D. Patterson (1990) An evaluation of redundant arrays of disks using an Amdahl 5890. in Proceedings of the 1990 ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems, 74-85.

Chu, J. and P. W. Dowd (1992) Coherence for photonically shcalable sared memory systems. Tech. Rep. TR-08-92-09, State University of New York at Buffalo, Department of Electrical and Computer Engineering.

Davie, B. S. (1993) The architecture and implementation of a high-speed host interface. *IEEE Journal on Selected Areas of Communications*, **11**, 228-239.

Dowd, P. W. and I -S. Hwang (1995) Memory and network architecture interaction in an optically interconnected distributed shared memory system, *Journal of Parallel and Distributed Computing*, **25**, 144-161.

Dowd, P W (1991) Random access protocols for high speed interprocessor communication based on a passive star topology *IEEE Journal on Lightwave Technology*, **9**, 799-808.

Feng, T. (1981) A survey of interconnection networks. *Computer*, 12-27.

Fleish, B. D and G. J Popek (1989) Mirage: A coherent distributed shared memory design. *Operating Systems Review*, **23**, 211-223.

Gustavson, D. (1992) The scalable coherent interface and related standards projects. *IEEE Micro*, 10-22

Gupta, A. and W -D Weber (1992) Cache invalidation patterns in shared-memory multiprocessors. *IEEE Transactions on Computers*, **C-41**, 794-810.

Gupta, A , W. -D. Weber, and T Mowry (1990) Reducing memory and traffic requirements for scalable directory-based cache coherence schemes, in International Conference on Parallel Processing, 312-I-321.

Hostetter, G , M. Santina, and P. D'Carpio-Montalvo (1991) *Analytical, Numerical and Computational Methods for Science and Engineering*. New York, USA Prentice Hall.

Hou, R. Y., G. R. Ganger, Y. N. Patt, and C. E. Gimarc (1992) Issues and problems in the I/O subsystem, part I - The magnetic disk, in Proceedings of the Twenty-Fifth Annual Hawaii International Conference on System Sciences, 48-57.

Hwang, I. -S. and P W. Dowd (1995) Media access protocol impact on a wdma passive star photonic network with distributed shared memory system. *Journal of Information Science and Engineering*, **11**, 345-369.

Jain, R., K. Somalwar, J. Werth, and J. Browne (1992) Scheduling parallel i/o operations in multiple bus system. *Journal of Parallel and Distributed Computing*, 352-362,

James, D., A Laundrie, S. Gjessing, and G. S. Sohi (1990) Distributed-directory scheme: Scalable coherent interface *IEEE Computer*, 74-77

Killat, U. (1987) Asynchrone zeitvielfachubermittung fur breitbandnetze. *Nachrichtentech. Z.*, **40**(8), 572-577.

Lang, M. V. T. and I. Alegre (1982) Bandwidth of crossbar and multipe-bus connections for multiprocessors. *IEEE Transactions on Computers*, **C-31**, 1227-1233.

Law, A. M. and W. D Kelton (1991) *Simulation Modeling and Analysis*, New York, USA McGraw Hill.

Lenoski, D , J Laudon, T. Joe, D Nakashira, L. Stevens,A. Gupta, and J. Hennessy (1992) The DASH prototype· Implementation and performance, in Proc. 19th International Symposium Computer Architecture, 92-103.

Minnich, R. G. and D. J. Farber (1990) Reducing host load, network load, and latency in a distributed shared memory. in 10th International Conf. Distributed Computer Systems, (Paris, France), 468-475.

Mudge, T. N. and H. B. Al-Sadoun (1985) A semi-markov model for the performance of multiple-bus systems. *IEEE Transactions on Computers*, **C-34**, 934-942.

Mudge, T., J. Hayes, G. Buzzard, and D. Winsor (1986) Analysis of multiple-bus interconnection networks. *Journal of Parallel andı Distributed Computing*, 328-343

Mudge, T., J. Hayes, and D. Winsor (1987) Multiple bus architectures *Computer*, 42-48.

O'Krafka, B. W. and A R. Newton (1990) An empirical evaluation of two memory-efficient directory methods, in Proc. 17th International Symposium Computer Architecture, (Seattle, Washington), 138-147.

Scott, S., J. Goodman, and M. K Vernon (1992) Performance of SCI Ring. In Proc 19th International Symposium Computer Architecture, (Gold Coast, Australia), 403-414.

Turner, J. S. and L F. Wyatt (1983) A packet network architecture for integrated services, in *Proc. GLOBECOM'83*, (San Diego, CA), 2.1.1-2.1 6

Tam, M., J. M. Smith, and D J. Farber (1990) A taxonomy-based

comparison of several distributed shared memory systems, *Operation Systems Review*, **24**, 40-67

Thakkar, S. , M. Dubois, A. T. Laundrie, and G. S. Sohi (1990) Scalable shared-memory multiprocessor architectures, *IEEE Computer*, 71-74.

Tung, C. -H. (1991) A scalable multibus multiprocessor architecture. In Proc. of the International Conference on Information Engineering - ICIE'91, **1**, 1-10.

Wilson, A. W. (1987) Hierarchical cache/bus architecture for shared memory multiprocessors. In Proc. 14th International Symposium on Computer Architecture, (Pittsburgh, Pennsylvania), 244-252.

Wittie, L., G. Hermannsson, and A. Li (1990) Scalable eagerly shared distributed memory. Tech. Rep. 90-48, Department of Computer Science, State University of New York at Stony Brook.

# 一種隱藏式媒體存取法應用於多重匯流排分散－共享記憶體系統

黃依賢

萬能工商專科學校電子科

## 摘　要

多重匯流排連接網路是常被使用於多重微處理器系統的一種結構。多重匯流排固然可增加系統的容量，但當使用者增加時會受制於匯流排的取得。本文提出一個平行輸入/ 輸出系統和一個分散式，橫條式開關系統，其用來幫助此隱藏式（控制匯流排）媒體存取法的操作，TDMA-C，以及增加多重匯流排連接網路的使用容量。本文建立一個半－馬可夫（semi–Markov）數學模式，一方面經實際模擬證實其準確性外，並改變其他參數而預估其系統特性。