Fairness Analysis of Multiprocessor Bus Arbitration Protocols

Mary K. Vernon
Scott T. Leutenegger

Computer Sciences Technical Report #744

September 1988

Fairness Analysis of Multiprocessor Bus Arbitration Protocols

Mary K. Vernon
Scott T. Leutenegger

Computer Science Department
University of Wisconsin-Madison
1210 Dayton Street
Madison, WI 53706

Abstract

Efficiency and fairness are two important properties of the bus arbiter in a bus-based multiprocessor system. The *parallel contention arbiter* is a highly efficient arbiter that requires very few control lines on the bus. The popularity of this arbiter is demonstrated by its adoption in nearly all multiprocessor bus standards, including the Fastbus, Futurebus, NuBus, and Multibus II standards. We have used Generalized Timed Petri Nets to analyze the the fairness of the protocols used in these arbiters. We also compare the efficiency and fairness of these protocols with the central round-robin, central priority, and simple daisy chain arbiter.

The "fairness protocols" that have been implemented in the parallel contention arbiter are widely regarded as being perfectly "fair", but no quantitative studies of their fairness have been reported. Our results show that these protocols have some undesirable fairness properties at high bus load. We define a useful fairness measure to be the ratio of the bandwidths allocated to the devices that are treated most favorably and least favorably by the protocols. Under an important set of assumptions about the system workload, this ratio is as high as 2.0 for the existing fairness protocols. We propose a small modification to one of the protocols that dramatically improves the fairness of the algorithm. In the modified protocol, the ratio of the bandwidths asymptotically approaches 1.0 as the load on the bus increases. However, the bandwidths allocated to devices by the improved protocol still have a spread of up to 10-15% in a range of offered loads where the bus is just becoming saturated.

The relative bandwidth allocated to each processor translates directly to the relative speeds at which application processes run on the processor. Our analytical results do not argue strongly in favor of changing existing standards or implementations, since the unfairness only arises at peak bus loads. However, we also present a new protocol for the parallel contention arbiter that implements the round-robin algorithm of central round-robin arbiters. The new protocol is as efficient as the existing protocols, is simpler to implement, and is perfectly fair. It would thus be the preferred protocol when designing a new standard or a new system bus.

Index Terms: bus arbitration protocols, Markov models, multiprocessors, parallel contention arbiter, performance evaluation, Petri nets.

This research was partially supported by the National Science Foundation under Grant No. DCR8402680 and Presidential Young Investigator Award DCR-8451405, and by grants from IBM and Cray Research.

1. Introduction

A computer system bus is a passive path that supports communication among various components of the system. An *agent* is any device that can independently issue requests for control of the bus. Sharing of the bus among agents is governed by a bus arbitration protocol. The logic that implements the protocol is called the bus arbiter.

In a multiprocessor that employs shared buses to interconnect processors and shared memory, the arbiter must be efficient for a relatively large number of agents (i.e. typically 10-64 agents, including I/O devices). In many multiprocessor systems, the arbiter should also arbitrate fairly among a subset of the agents (i.e. the processors).

The methods that have been employed to date for multiprocessor bus arbitration are the daisy chain, the central arbiter, and a parallel contention arbiter invented at Computing Devices of Canada [UKPS66] and by D. M. Taub [Taub84]. The third method is the most popular, measured by the fact that it has been adopted in nearly all multiprocessor bus standards. This method can be as efficient as the central arbiter, yet it requires only $O(log_2N)$ control lines, where N is the number of agents. The parallel contention arbiter has a number of additional advantages which are discussed in section 2 of this paper.

The basic algorithm in the parallel contention arbiter efficiently determines which competing agent asserts the largest number on the control lines during an arbitration. In order to prevent errors in the operation of the arbiter, a fixed unique number is assigned to each agent. Thus, the basic algorithm implements strict priority service of agent requests, where priority is determined by the agent's assigned arbitration number. *Fairness protocols* have been designed to overcome the unfairness of the priority allocation algorithm by specifying when an agent is allowed to compete in an arbitration.

This paper examines in detail two distinct fairness protocols that have been specified for the parallel contention arbiter. Protocol I has been adopted in the Fastbus, NuBus, and Multibus II bus standards [Gust86, PWGr86, Intel]. Protocol II has been adopted in the Futurebus standard [Taub84]. We compare these fairness protocols with the round-robin protocol implemented by the central arbiter, and with the strict priority algorithm implemented by the daisy chain arbiter, the central arbiter, or the parallel contention arbiter.

We make the comparisons assuming the same technology is used to implement each of the arbiters, in order to factor out the influence of technology on the performance of the arbitration protocol. Thus, our results will not indicate how the performance of Fastbus, which is implemented in ECL, compares with the performance of Futurebus, which is implemented in TTL, but will instead indicate how the arbitration protocol adopted in Fastbus compares with the protocol adopted in Futurebus.

We have used Generalized Timed Petri Nets (GTPN) to model and analyze the protocols. A key assumption we make in the analysis is that processors do not perform useful execution while waiting for bus requests to be satisfied. Our results show that although the fairness protocols for the parallel contention arbiter are widely regarded as being perfectly fair, they do not allocate bus bandwidth equally among agents that have equal need for the bus at high bus load. In particular, for an important set of assumptions about the workload *up to 100% more bandwidth is allocated to the agent that is treated most favorably as compared with the the agent that is treated least favorably.* The relative bandwidth allocated to each processor in a multiprocessor translates directly to the relative speeds at which application processes run on the processors. We propose a small modification to Protocol II that dramatically improves the fairness of the protocol. However, the improved protocol still allocates up to 10-15% more bandwidth to the most-favorably treated agent. Finally, we present a new fairness protocol that we have proposed for the parallel contention arbiter [VeMa87]. This new protocol uses *fixed* arbitration numbers and implements precisely the central arbiter round-robin protocol. The new protocol retains all of the advantages of the parallel contention arbiter, is simpler to implement, and is perfectly fair in its allocation of bus bandwidth for equal-priority transactions.

Section 2 of this paper describes the arbitration methods and protocols studied, and reviews previous work in evaluating bus arbitration protocols. Section 3 discusses the system assumptions we have made in developing the models, gives a brief overview of the GTPN, and presents the protocol models used in our study. Section 4 contains the results of the analysis, and section 5 describes the proposed new protocol for the parallel contention arbiter. Section 6 contains the conclusions of this paper.

2. Background

In this section, we first describe the parallel contention arbiter and outline the existing priority and fairness protocols for this arbiter. We then briefly review the daisy chain and central round-robin arbiter protocols. In section 2.3 we review previous work in analyzing arbitration protocols.

In the descriptions of the bus arbiters, arbitration control lines are assumed to carry the wired-OR of all the driving signals [Gust84]. That is, the line is tied to a voltage source, and each agent either lets the line float (logical "0"), or forces it to another particular voltage level (logical "1"). Applying a "1" to the line is called *asserting* the line; allowing it to float back to "0" is called *releasing* it.

2.1. The Parallel Contention Arbiter

The parallel contention arbiter, illustrated in Figure 2.1, supports up to 2^k-1 agents with k (plus three or four) arbitration control lines. Each agent is assigned a unique k-bit arbitration number. At the start of an arbitration, each agent requesting ownership of the bus asserts its arbitration number on the arbitration lines. If an agent senses a "1" on a line it is not asserting, then it releases the lower-order bits of its arbitration number from all lower-order lines. If the line drops back to "0", the agent re-asserts the lower order bits of its arbitration number.

For example, consider the case where two agents, with arbitration numbers 110010 and 101110 respectively, are competing in the arbitration. Initially each agent asserts all bits of its arbitration number. Next, the first agent removes the three lowest order bits of its number, while the second agent removes all but the most significant bit of its number. The second agent takes no further action, but the first agent will re-assert the three lowest order bits of its number when it senses that the third and fourth most significant arbitration lines have been released.

After some period of time, the system reaches steady state, and the arbitration lines carry the maximum of the arbitration numbers of the competing agents. The agent whose assigned arbitration number equals the value on the control lines determines that it is the winner.

The parallel contention arbiter is fully distributed and highly efficient. Taub has shown that the maximum delay to reach steady state is $\frac{k}{2}$ end-to-end bus propagation delays, plus a small amount of delay for the logic [Taub84]. (His proof is based on a worst-case physical assignment of arbitration numbers along the bus; not all intermediate values are required to travel the entire length of the bus). In practical systems, k equals six or seven, and the delay for

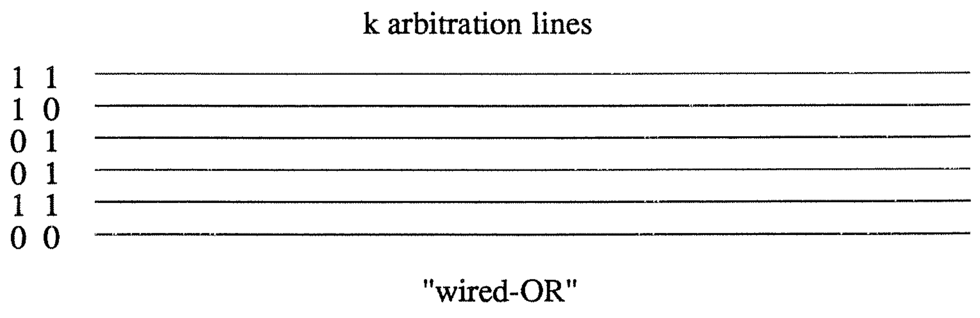


Figure 2.1: The Parallel Contention Arbiter

arbitration safely occurs within one hundred nanoseconds. Thus, the efficiency of the parallel contention arbiter is comparable to the central arbiter, yet the number of control lines grows only logarithmically with the number of agents on the bus.

There are two additional advantages of this arbiter. First, the state of the arbitration is available everywhere on the bus, which facilitates software initialization and failure diagnosis. Second, as will be discussed below, it is straightforward to integrate priority service of urgent requests with fair service of non-urgent requests in the arbiter.

2.2. Priority and Fairness Protocols for the Parallel Contention Arbiter

Fixed Priority Protocol

The parallel contention arbiter naturally implements a fixed priority protocol, in which an agent's priority is determined by its assigned arbitration number. When an arbitration takes place, the competing agent with the highest priority is elected. All other agents wait and compete in the next arbitration.

The fixed priority protocol is employed in systems where it is important to guarantee certain agents priority access to the bus. For example, the high-priority agents may be engaged in real-time data acquisition activities. However, in many multiprocessors it is desirable to arbitrate fairly among the processors. Thus, two "fairness protocols", outlined below, have been specified to overcome the unfairness of the priority protocol.

Fairness Protocols I and II

The fairness protocols maintain a small amount of state information that is used to decide whether an agent wanting control of the bus will compete in the next arbitration. Both protocols are based on the idea of *batching* requests.

In protocol I, the request(s) that arrive to an idle bus form a batch. An agent in the batch competes during each arbitration, until it is elected by the basic priority protocol. An agent that generates a new request while a batch is in progress must wait for the batch to end before competing for access. The end of the batch may be signaled by a logical "0" on the bus request line, or by an arbitration in which no agents compete. The second mechanism is known as a *fairness release cycle*. All requests that are waiting at the end of a batch form a new batch. This protocol was adopted in the Fastbus, NuBus, and Multibus II standards. A similar protocol was implemented in a daisy chain arbiter known as the *static batching daisy chain* [KlCa86].

In Protocol II, an agent with a request initially asserts the request line and competes in successive arbitrations until it is elected. At the *completion* of its bus transaction, the agent marks itself as inhibited from competing for bus ownership until a fairness release operation takes place. The Futurebus standard, which is the only specification of this protocol, does not specify whether a fairness release operation is triggered if there are no active requests, or only if all agents with outstanding requests are inhibited. This decision is left to the system implementor. The protocol is slightly more fair in the second case, so we assume that case in all experiments reported in section 4.

Protocol II implements a batching algorithm similar to protocol I. No agent is served more than once in a batch. However, in protocol II, a new request that is generated during a batch is allowed to join the batch if the agent has not previously received service in the batch. Protocol II has been adopted in the Futurebus standard [Taub84].

The arbitration overhead for the fairness protocols is the same as for the basic parallel contention arbiter, except when requests are waiting at the end of a batch. In this case, a small amount of overhead is incurred for the waiting agents to detect the end of the batch and to assert the request line to form a new batch.

The fairness protocols for the parallel contention arbiter are widely regarded as being "fair". However, since agents within a batch are served in order of their arbitration numbers, there is a greater opportunity for agents with higher assigned numbers to generate requests for the next batch. The fairness of these protocols has not previously been examined in detail, and is the subject of the analysis in Section 4 of this paper.

A Protocol that Implements Both Priority and Fairness

Priority scheduling of urgent requests can easily be integrated with more fair scheduling of non-urgent requests in the parallel contention arbiter. In this case, agents follow one of the fairness protocols for non-priority requests, but ignore the protocol for priority requests. That is, priority requests compete in the every arbitration after they occur, until they are satisfied. Non-priority requests follow the assured access protocol which specifies when they are first allowed to compete in the arbitration. Furthermore, an extra line is typically provided on the bus, to be treated as the most significant bit of the arbitration number. Agents with priority requests assert this line during an arbitration; agents with non-priority requests do not. This guarantees that all priority requests are served before non-priority requests.

The integration of priority and fairness scheduling in the parallel contention arbiter is not analyzed in our study. We mention this feature for two reasons. First, it is yet another advantage of the parallel contention arbiter. Second, the protocol we propose in Section 5 uses the idea of providing an extra high-order bit in the arbitration number.

Round-Robin Protocol

The round-robin arbitration protocol is based conceptually on a circular ordering of the agents. An arbitration occurs only if there is at least one active request. Starting from the winner of the previous arbitration, the requesting agent "next" in the circular order is elected. Note that the round-robin scan does not occur continuously throughout time, but is instead triggered by one or more active requests.

The schemes proposed previously for implementing round-robin service in the parallel contention arbiter, are based on rotating the assigned arbitration numbers after each arbitration [Borr88]. However, these schemes are less robust and more complex to implement than the fairness protocols discussed above, which are based on fixed assigned arbitration numbers. Thus, the fairness protocols have been preferred in the multiprocessor bus standards that employ the parallel contention arbiter.

2.3. Other Multiprocessor Bus Arbiters

The parallel contention arbiter can be compared with the simple daisy chain arbiter and the central arbiter, which have also been implemented in multiprocessor systems and adopted in multiprocessor bus standards. Below we briefly review these arbiters and the protocols they implement.

The Daisy Chain Arbiter

The simple daisy chain arbiter [Baer80], illustrated in Figure 2.2, is a very inexpensive arbiter that is efficient if the number of agents on the bus is small. In the implementation, agents share a single bus *request* line. Any number of agents may assert the request line simultaneously. In response to a request, a grant signal is issued at one end of the bus grant line, which is chained through all the agents. Each agent must actively propagate the signal on the grant line to the next agent, until the signal reaches an agent that has issued a request. This agent is elected the next bus master. The daisy chain is reset at the start of the elected agent's bus tenure. The next arbitration takes place in the same manner, possibly in parallel with the bus transaction.

The protocol followed by each agent on the bus is simple, and only three control lines are required on the bus to implement the daisy chain arbiter. However, arbitration overhead increases linearly in the number of agents on the bus.

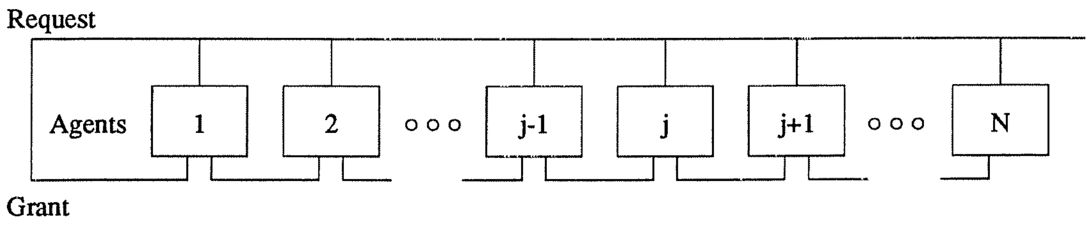


Figure 2.2: The Daisy Chain Arbiter

Like the parallel contention arbiter, the daisy chain arbiter naturally implements a fixed priority protocol, where in this case the agent's priority is determined by its physical location on the bus grant daisy chain. This is the protocol most commonly used in daisy chain arbiters.

The daisy chain arbiter can alternatively implement the round-robin protocol. In this case, instead of initiating the grant signal at a central location on one end of the chain, the agent that wins an arbitration initiates the bus grant signal in the next arbitration. This scheme, named the *rotating daisy chain* [BaAh81], has similar overhead to the standard daisy chain, and is rarely used in practice.

The Central Arbiter

In the central arbiter, illustrated in Figure 2.3, each agent has a private request and a private grant line to the arbiter. If any request line is asserted, the next bus master is computed by the central logic, and a grant signal is sent to the elected agent.

The central arbiter is efficient and can implement a variety of scheduling policies. Overhead for arbitration among a large number of agents is very low; however, the number of control lines and the number of inputs to the arbiter equals twice the number of agents on the bus. The central arbiter also has problems in terms of monitoring and

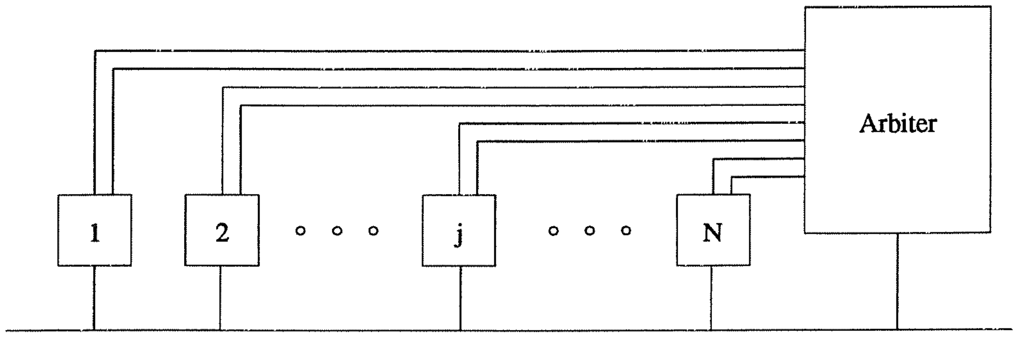


Figure 2.3: The Central Arbiter

initialization [Gust84].

The central arbiter is the most common and straightforward method of implementing the round-robin protocol. One possible circuit that implements this protocol is illustrated in Figure 2.4 for four agents.

The fixed priority protocol is implemented in the central arbiter by assigning fixed numbers to the agent request lines. The maximum of the active input requests is then elected by the central arbitration logic.

2.4. Previous Analyses of Bus Arbitration Protocols

Bain and Ahuja [BaAh81] have studied the performance and fairness of a static priority arbiter, a fixed time slice algorithm (i.e. time division multiplexing), a rotating daisy chain, a "least recently used" dynamic priority arbiter, and an ideal FCFS arbiter. They assume that all arbitration overhead is negligible. Thus, their rotating daisy chain model is an idealized model of the central round-robin arbiter. Their measures, computed using simulation, include the mean and the coefficient of variation of the waiting time over all requests, and over all agents. They assume a fixed bus access time of 400 nanoseconds (i.e. 4 cycles on a 10 MHz bus), and a uniformly distributed inter-request time with a fixed mean of 10 microseconds. They report performance and fairness as a function of the number of agents on the bus, and they include studies of the sensitivity of the results to variations in the bus access time. Our models can be run with their parameters, and where our results overlap with their study, they agree. Bain

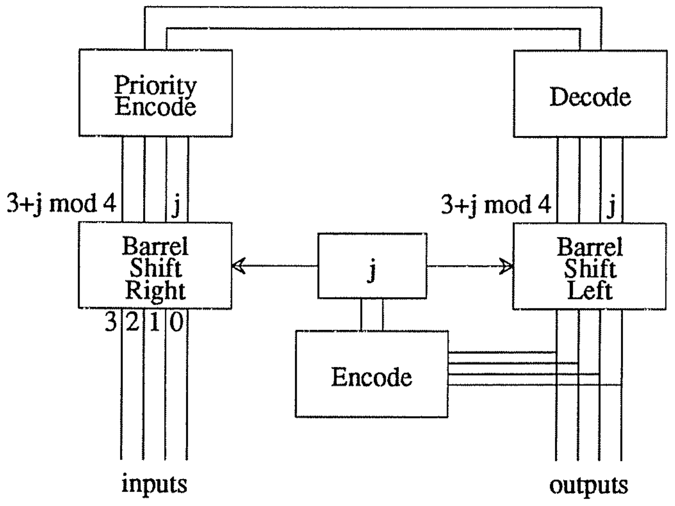


Figure 2.4: The Central Round-Robin Arbiter

and Ahuja do not consider the fairness protocols for the parallel contention arbiters, nor do they study the asymptotic performance of the arbiters under high system load. Furthermore, we have found the ratio of the bandwidths allocated to the most favorably and least favorably treated devices to be a more meaningful measure of fairness than the coefficient of variation of the request waiting times.

Sharma and Ahuja [ShAh81] have proposed a first-come first-serve bus arbiter based on a ticket assignment scheme. Arbitration for ticket numbers is done using an existing arbitration method, such as the static or rotating daisy chain. We have not considered this arbiter in this paper, since it is only a close approximation to the ideal FCFS algorithm if the time to arbitrate for ticket numbers is very short compared with bus service times.

Kleeman and Cantoni [KlCa86] have analyzed a static batching daisy chain arbiter using a finite-state, irreducible Markov Chain model. They give limiting results for light and heavy loading conditions, as well as performance results derived numerically for a few system parameters. The arbitration algorithm in their static batching daisy chain is similar to Protocol I for the parallel contention arbiter. They do not consider Protocol II. Their work provided the starting point for the study reported in this paper.

3. Arbiter Models

We have used Generalized Timed Petri Nets (GTPN) to model the most common multiprocessor bus arbitration protocols: fixed priority, round-robin, and fairness Protocols I and II. The advantages of using the GTPN are two-fold. First, the models are precisely defined and have a very compact and regular structure. Second, performance estimates can be computed from the models automatically using Markov Chain methods, as described in [HoVe85]. We have solved the models analytically for small systems, and have simulated them for larger systems. The results of the models are presented in section 4.

In section 3.1, we discuss the system assumptions made in our protocol models. In section 3.2 we discuss the input parameters and output performance measures of the models. We then provide a brief overview of the GTPN (Section 3.3), and present our models of Fairness Protocols I and II for the parallel contention arbiter (Sections 3.4 and 3.5). The models of the fixed priority and round-robin protocols have structure and complexity similar to the fairness protocol models, and are presented in the Appendix. The reader who is unfamiliar with the Petri net notation may skip sections 3.3 through 3.5.

3.1. System Assumptions

In this section, we outline the assumptions made in our protocol models. These assumptions are not the only assumptions that might be reasonable for a study of arbiter performance, but are an important set of assumptions for which the arbiter should be efficient and fair.

The models hold for both synchronous and asynchronous buses. We assume that the time required for a bus transaction is fixed. This is true, for example, for cache block transfers between the processors and shared memory in a multiprocessor. We let the fixed bus transaction time define the unit of time in the models. All other model parameters and our performance measures are normalized to the unit of time for a bus transaction.

A key assumption in the models is that agents cannot perform productive work that will lead to the next bus request while waiting for a bus request to be satisfied. We assume that bus interrequest times are independent and variable, and model the interrequest time by a geometric random variable with a specified mean. The mean bus interrequest time is the important workload parameter in our models.

In each of the models, we assume that arbitration for the next winner begins at the start of a bus transaction if any requests are pending. In the current study, we assume the arbitration time for the central and parallel contention arbiters is equal to one-half of a bus transaction time. This estimate was arrived at by assuming bus speeds in the range of 10-40 MHz, and fixed bus transaction times on the order of four to eight bus cycles in duration. For longer bus transactions, the assumed overhead is too high; however, the overhead is only observed when a request arrives to an idle bus. For the case of very small bus transaction times, such as single-cycle bus operations, the assumed overhead is lower than the actual overhead. Since single-cycle bus operations only account for a fraction of the traffic in most real systems, we expect any errors in the overhead estimate to have a small effect on the performance estimates, and a negligible effect on the fairness results of our analysis.

For the daisy chain arbiter, we consider bus grant propagation delays of $\frac{1}{8}$, and $\frac{1}{4}$ of a bus transaction time, per station. The lower delay of $\frac{1}{8}$ of a bus transaction time allows us to make the most favorable comparison possible between this implementation of the protocols and the other implementations considered in our study.

3.2. Model Input Parameters and Output Measures

The input parameters of our arbiter models are 1) the number of agents, 2) the normalized mean interrequest time per agent, and 3) the overhead for releasing inhibited agents in the fairness protocols. The value assumed for the third parameter is zero unless otherwise indicated in section 4. Recall that the normalized mean interrequest time is given in units of the deterministic bus transaction time.

We define the offered *load* of an agent to be the reciprocal of its normalized mean interrequest time. That is, the offered load for an agent is the ratio of its bus access time to its mean interrequest time. We also define the *total offered load* to be the sum of the loads of all agents.

The performance measure we are interested in is the total bus throughput. Using measures normalized to the bus transaction time, the throughput measure represents the number of completed requests per bus transaction time. Thus, the normalized total bus throughput also equals the bus utilization.

The fairness measure we use is the ratio of the throughput for the agent that is treated most favorably to the throughput of the agent treated least favorably by the arbitration protocol. We have found this to be a more meaningful measure than the coefficient of variation of the agent throughputs or waiting times, used in [BaAh81]. We will also look at throughput curves for each agent to understand qualitatively how the agents are treated by the protocols.

All performance measures are obtained automatically from the analysis, using resource usage estimates computed by the GTPN analyzer.

3.3. Generalized Timed Petri Nets (GTPN)

In this section we provide a very brief review of the main features of the GTPN. For a detailed description of the GTPN, the reader is referred to [HoVe85]. We assume the reader is familiar with the Petri net notation [Pete81]. As mentioned above, a reader unfamiliar with this model notation may skip this and the next two subsections.

The GTPN is a Petri net which has the following attributes associated with each transition: 1) a deterministic firing duration which may depend on the state in which the transition starts firing, 2) a relative firing frequency which may depend on the state in which the transition starts firing, and 3) a special flag that specifies how next-state probabilities are to be calculated.

The state of the net is defined by the number of tokens in each place in the net, and the *remaining firing time* for each transition firing that is currently in progress. If any transitions are enabled in a given state of the net, then the set of possible next states is defined by all maximal sets of the enabled transitions that can start firing together. Once a transition starts firing, it cannot be pre-empted. If no transitions are enabled in a given state, the net will remain in this state for a duration equal to the minimum remaining firing time of all transitions that are currently firing.

For all nets in this study, the flag associated with each transition is "no", indicating that the next state probability for a maximal set of transitions that start firing together should be computed simply by taking the product of the transition firing frequencies, appropriately normalized over all possible next states.

The model also has a set of named *resources*, each of which is associated with any number of places and/or transitions in the net. A named resource is considered to be "in use" for each token residing in an associated place, and for each firing of an associated transition that is in progress. We compute the state space of the Markov Chain associated with a GTPN model, and derive steady state resource usage estimates from the solution of the Markov Chain, automatically.

3.4. GTPN Model of Protocol I

The GTPN model of Protocol I is shown in Figure 3.1 and Table 3.1 for two agents. The delay and relative frequency associated with each transition in the model are given in Table 3.1. Note that some frequency attributes contain logical expressions that define when the enabled transition is allowed to fire. The name of a place in a logical expression represents the number of tokens currently in the place. The name of a transition represents either "1" or "0", depending on whether or not the transition is currently firing.

Transitions T1-T4 model the behavior of agent 1, the agent with the higher assigned arbitration number. Transitions T5-T8 similarly model the behavior of the agent with the lower arbitration number. The model is extended to n agents in the obvious way. The place BS is an output of both T3 and T7. Thus, BS and OT are shared by all agents. The following describes the behavior of the subnet for agent 1.

Transitions T1 and T2 implement the geometric distribution of time until the agent issues a new request for the bus. With probability $1-p$, the agent spends a duration of *step-size* executing T1, and then returns to E1. With probability p, the agent issues a request. The mean bus interrequest time, measured in the units of a bus transaction time, is $\frac{1}{p} \times$ *step-size*. The reciprocal of this value is the offered load of the agent. Once the agent makes a bus request, the token moves to place R1.

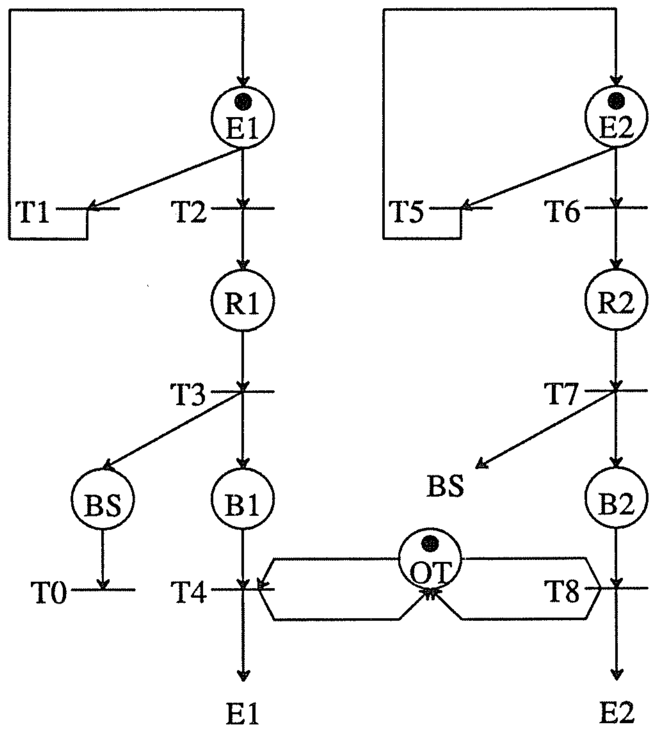


Figure 3.1: GTPN Model of Protocol I

Table 3.1: Transition Durations and Frequencies for the Model of Fairness Protocol I

Transition	Duration	Frequency
T0	*batch reset overhead*	(OAT = 0) × (B1 + B2 = 0)
T1	*step size*	1-p
T2	*step size*	p
T3	*arbitration time*	(BS + T0 = 0) × (T3 + T7 = 0)
T4	1.0	1.0
T5	*step size*	1-p
T6	*step size*	p
T7	*arbitration time*	(BS + T0 = 0) × (T3 + T7 = 0)
T8	1.0	(B1 = 0)

Transition T3 fires if the bus is idle or serving the last request in a batch (i.e., BS+T0=0), and no arbitration is currently in progress (i.e., T3+T7=0). In this case, the token moves to B1, signifying that the request is competing for its turn at the bus. Transition T4 models the use of the bus by agent 1. The delay for this transition is one time unit, defining the unit of a bus transaction time in the model.

In the model with n agents, all tokens in places R1–Rn are moved to B1–Bn at the same time (i.e. at the beginning of a batch). Requests in places B1–Bn are allowed access to the bus one at a time, in priority order. The token in OT insures that only one agent is granted access at a time. The transition frequency expressions for T4 and T8 implement the priority order (see Table 3.1).

Transition T0 implements the event of recognizing the end of a batch. When the last request in a batch begins service, OT=0 and B1+B2+...+Bn=0, and transition T0 fires. The delay for T0 models the overhead for releasing inhibited agents in the real system. Two values of this "batch reset overhead" are considered in our experiments: 1) the ideal case of zero, and 2) the more typical case of 0.5 units of time. When T0 completes, the next batch is allowed to start its first arbitration.

Note that the last agent served in a batch cannot submit a request in the next batch if two conditions are met. The first is that the *batch reset overhead* parameter is less than 1.0 plus *step size*. The second condition is that inhibited requests are waiting before the agent completes service. Similarly, the next-to-last agent served in a batch cannot join the next batch if *batch reset overhead* is less than *step size*, and inhibited requests are waiting when it completes service. Thus, depending on the values of the batch reset overhead parameter, *the two or four lowest priority agents may be blocked in every other batch during periods of high demand for the bus.*

The *step-size* parameter determines the discrete instants during a bus transaction at which new requests are issued for the bus. When solving the model analytically, we use a value of 0.5 for this parameter. In our model simulations, we use the limiting case of an exponentially distributed interrequest time with the appropriate mean.

To compute the normalized throughput for agent 1 for the model of protocol I, we associate a resource, *u1* with transition T4. The analyzer computes the steady-state utilization of *u1* (i.e. the steady-state fraction of time that t4 is firing). By Little's Result, the throughput for agent 1 is equal to this value, since the duration of T4 is 1.0. The measures for other agents are computed using the same technique. Total throughput is the sum of the throughputs for each agent. Throughput measures are calculated in a similar way for the other models.

It is similarly straightforward to compute mean bus waiting times from the GTPN models; however, the mean waiting time estimates do not provide additional insight beyond the throughput results reported in the next section.

3.5. Protocol II

The model of Protocol II is shown for two agents in Figure 3.2 and Table 3.2. Transitions T0 through T4 model the agent with the higher arbitration number; transitions T5 through T9 model the agent with the lower number. Again, this net is extended to *n* agents in the obvious way.

The following describes the subnet for agent 1. As in the model for protocol I, transitions T0 and T1 model the geometric distribution of time until the agent requests access to the bus. When the agent requests access, the token moves to R1.

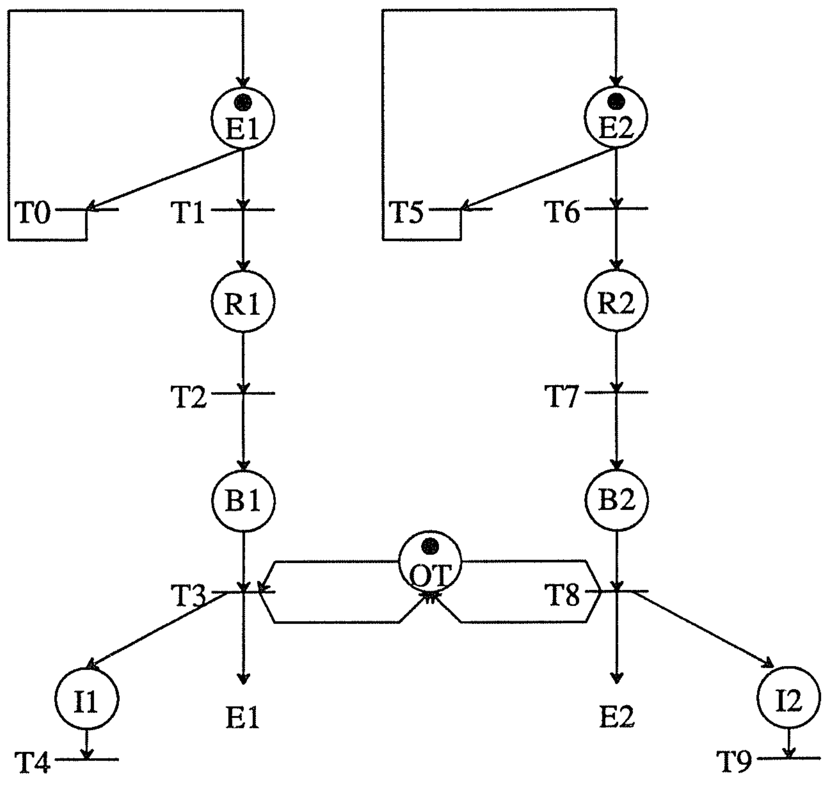


Figure 3.2: GTPN Model of Protocol II

Table 3.2: Transition Durations and Frequencies for the Model of Fairness Protocol II

Transition	Duration	Frequency
T0	*step size*	1-p
T1	*step size*	p
T2	*arbitration overhead*	$(I1 + T4 = 0) \times (T2 + T7 = 0) \times ((OT = 1) \text{ or } (B1 + B2 = 0))$
T3	1.0	1.0
T4	*batch reset overhead*	$(R1 + R2 > 0) \times (R1 \leq I1) \times (R2 \leq I2)$
		$\times (T2 + T7 + B1 + B2 = 0)$
T5	*step size*	1-p
T6	*step size*	p
T7	*arbitration overhead*	$(I2 + T9 = 0) \times (T2 + T7 = 0) \times ((OT = 1) \text{ or } (B1 + B2 = 0))$
T8	1.0	$(B1 = 0)$
T9	*batch reset overhead*	$(R1 + R2 > 0) \times (R1 \leq I1) \times (R2 \leq I2)$
		$\times (T2 + T7 + B1 + B2 = 0)$

Transition T2 only fires if three conditions are met: (1) the agent has not marked itself as inhibited due to previous service in the current batch (I1+T4=0), (2) there is no arbitration in progress (T2+T7=0), and (3) an arbitration may start (OT=1 or B1+B2=0). When transition T2 fires, the token moves to B1, signifying that the agent has participated in the arbitration and is competing to become the bus master. Tokens in places B1 - Bn are granted access one at a time in priority order, as in the model of Protocol I. As specified in the Futurebus protocol, an agent sets a flag indicating that it has received service in the current batch, at the *end* of its service. For agent 1, this is modeled by placing a token in I1 when transition T3 finishes firing. The token in I1 will inhibit transition T2 until the batch in progress at the time the agent inhibits itself ends.

A fairness release operation occurs when the following three conditions are met. First, there are requests waiting (R1+R2>0). Second, all of the waiting requests are inhibited (R1≤I1 and R2≤I2). Third, no agents are arbitrating or waiting for bus mastership (T2+T7+B1+B2=0). In this case, transition T4 fires, removing the inhibiting token in I1. Note that removal of the first condition also conforms to the Futurebus specification of the protocol. We have found that this has negligible effect on the results in Section 4.

4. Model Results

Some results of our models are presented in this section. We are interested in comparing the efficiency and fairness of the arbitration protocols for systems of five to sixty-four agents.

We can vary the total offered load on the bus either by increasing the number of agents, or by increasing the offered load per agent. We choose to hold the number of agents fixed and to vary the offered load (i.e. normalized agent interrequest times). This approach makes it easier to compare the relative agent throughputs as bus load increases. We demonstrate that we would reach the same conclusions by holding interrequest time fixed and varying the number of agents. (See Section 4.3). We solve the models analytically for five agents and simulate them for thirty agents. The results are plotted as a function of of total offered load so that curves obtained with different numbers of agents can be compared.

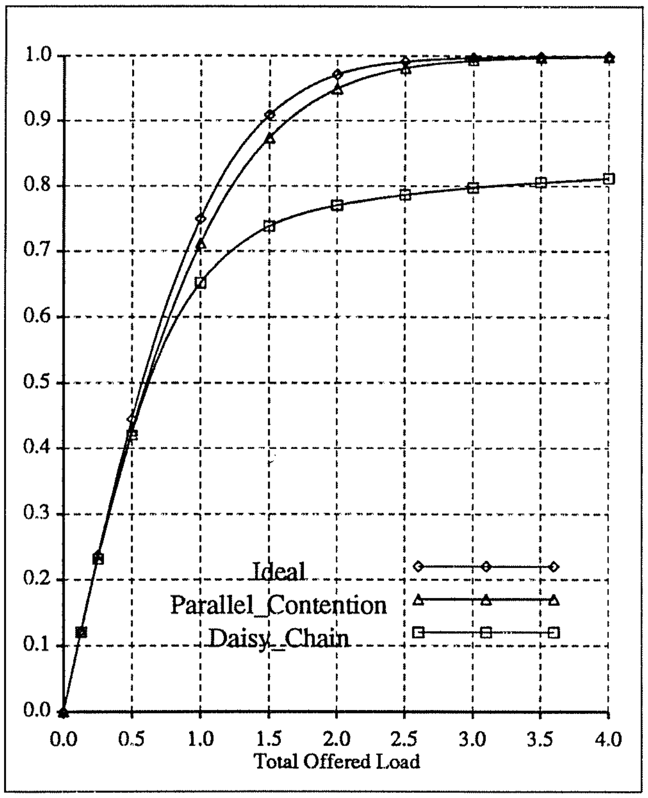
The *batch reset overhead* of the fairness protocols is assumed to be zero unless otherwise indicated.

4.1. Overall Efficiency of the Arbiters

We first examine the overall performance of the daisy chain, central, and parallel contention arbiters. Total bus throughput is the performance measure. The conclusions of this section are largely obvious, but to the authors' knowledge have not been previously quantified. Furthermore, the total throughput curves show bus utilization as a function of total offered load, since throughput is measured in requests per bus transaction time. This helps calibrate the independent variable, *total offered load*, in the model.

Total bus throughput is the same for all arbiters that have the same arbitration time. In Figure 4.1a, we have plotted the total throughput as a function of total offered load for three cases. The top curve is the "ideal" arbiter, in which arbitration takes place in zero time. The second curve is for the central arbiter and parallel contention arbiter, which each have arbitration time equal to 0.5 times a bus transaction time in our models. The lowest curve is for a daisy chain arbiter with a per agent bus grant propagation delay of 0.125 times a bus transaction time. The daisy chain curve assumes that arbitration is not overlapped with bus transactions. This assumption was motivated by what we considered to be a "typical" environment for the daisy chain arbiter. All three curves were obtained by solving the five-agent model analytically.

The efficiency of the central and parallel contention arbiters is very nearly equal to the efficiency of the ideal case. The parallel contention or central arbiter has a throughput at most 5% less (at total offered load of 1.0) than the



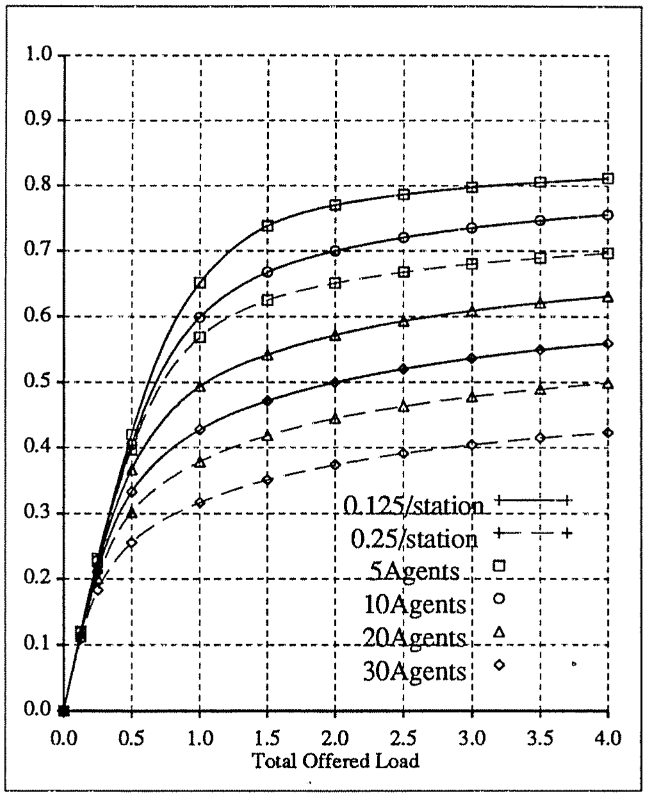


Figure 4.1a: Total Bus Throughput vs Offered Load

Figure 4.1b: Daisy Chain Throughput as a Function of Total Offered Load

throughput of the ideal arbiter, and operates and 100% efficiency in the limiting case. Thus, these arbiters are highly efficient. Note that the bus utilization equals nearly 45% at total offered load of 0.5, equals 70-75% at total offered load of 1.0, and equals approximately 90% at total offered load of 1.5. Thus, the interesting range of the offered load parameter is between 0.5 and 1.5.

Under the most optimistic assumptions, the five-agent daisy chain is competitive with the parallel contention or central arbiter at total offered load below 0.75 (steady state bus utilizations below 60%). However, in Figure 4.1b we investigate the performance of the daisy chain arbiter further for five, ten, twenty, and thirty agents. The solid curves in the figure assume a per agent bus grant propagation delay of 0.125 times a bus transaction time; the dashed curves assume a per agent overhead of 0.25 units. The dashed curve for ten agents is indistinguishable from the solid curve for twenty agents, and thus is not drawn. The curves show that the daisy chain arbiter is only competitive with the parallel contention or central arbiter for systems with fewer than 10 agents and total offered load below 0.5-0.7.

The curves in Figure 4.1b were obtained by solving the five-agent model, and letting each agent in the model represent $\frac{N}{5}$ of the agents in the real N-agent system. That is, the offered load and propagation delay of each agent in the model is the sum of the corresponding values for $\frac{N}{5}$ of the agents in the N-agent system. This approach is approximate, but we claim that it is sufficiently accurate for the conclusions of this subsection, due to the agreement between five-agent and thirty-agent results which is demonstrated in section 4.3.

4.2. Fairness of the Arbitration Protocols

In order to study the fairness of the arbitration protocols, we first examine the throughput of each agent on the bus as a function of total offered load, under the assumption that each agent has the same mean interrequest time. The throughputs are shown in Figure 4.2a for the fixed priority protocol.

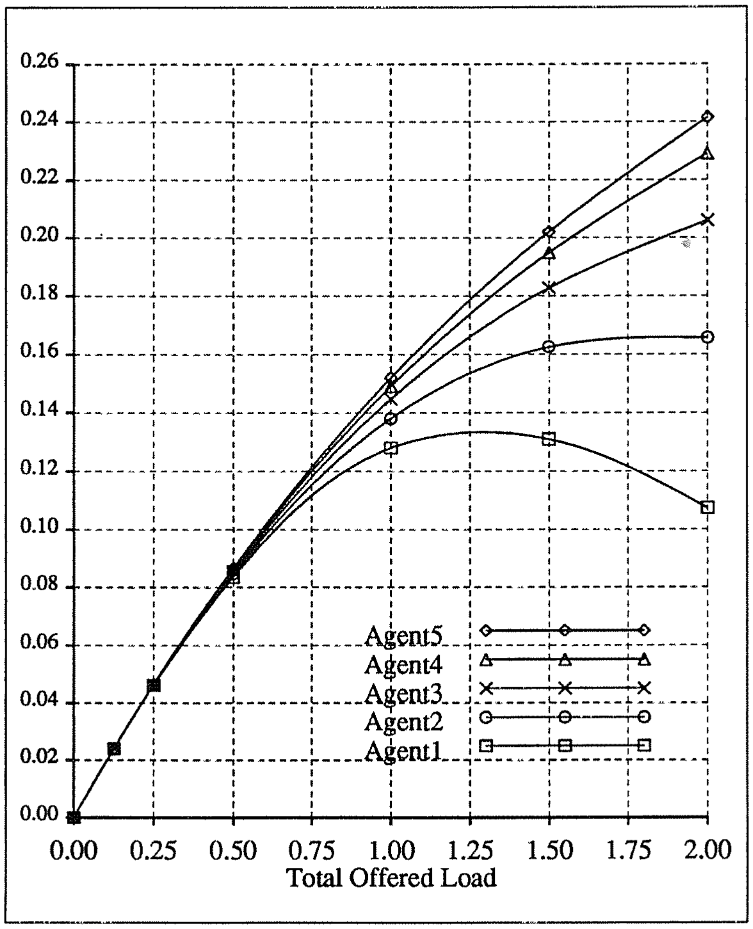


Figure 4.2a: Agent Throughputs for Fixed Priority Protocol

Figures 4.2b and c give the throughputs for two cases of fairness Protocol I. Figure 4.2b gives the results for "batch reset overhead" equal to 0.5; Figure 4.2c gives the results for zero overhead for the release of inhibited agents at the end of a batch. Figure 4.2d gives the agent throughputs for Protocol II. The value of the batch reset overhead does not have a noticeable effect on the relative throughputs under Protocol II. The throughputs of all agents are identical under the round-robin protocol, and for five agents these throughputs approach 0.2 at high load. Thus we have not plotted those curves. Note that at a total offered load of 2.0, the bus utilization is approximately 95% (see Figure 4.1a).

Surprisingly, the fixed priority protocol allocates nearly equal bandwidth at offered bus load below 0.5, or at bus utilizations below 40%. On the other hand, at bus utilizations above 60% the fixed priority protocol has a significant degree of inequity in the allocation of bus bandwidth compared with the fairness protocols. If fair allocation of bandwidth is desired and the system is expected to incur bus loads in this range, a fairness protocol should be implemented.

From the graphs in Figures 4.2b-d, we see that both fairness protocols are quite fair at total offered load below 0.75. Above 0.75, the throughputs of the agents are discernibly unequal. Agents with lower arbitration numbers are allocated less bandwidth because they are served later, on the average, within a batch, and because they don't generate a new request until the previous is satisfied. It appears that all three cases have roughly the same degree of fairness over the range of total offered load that one might expect during the normal operation of the system, although Protocol I with a batch reset overhead of 0.5 has slightly more diverging curves at the highest load shown.

In the next section we examine the asymptotic behavior of the throughput curves for the fairness protocols as we increase the total offered load.

4.3. Asymptotic Analysis of the Protocols I and II

Figures 4.3a-e contain the throughput curves for each agent as a function of total offered load, where the load is varied from 0 to 7.5. The point here is to understand the behavior of the protocol as demand on the bus increases. The graphs in Figures 4.3a and b are extensions of Figures 4.2a and b, and give the analytical results for Protocol I with five agents. Figure 4.3c contains the throughput curves obtained by simulating the Protocol I model with thirty agents and a batch reset overhead of 0. Figure 4.3d gives the analytical results for Protocol II with five agents, and Figure 4.3e contains the results obtained by simulating Protocol II with thirty agents.

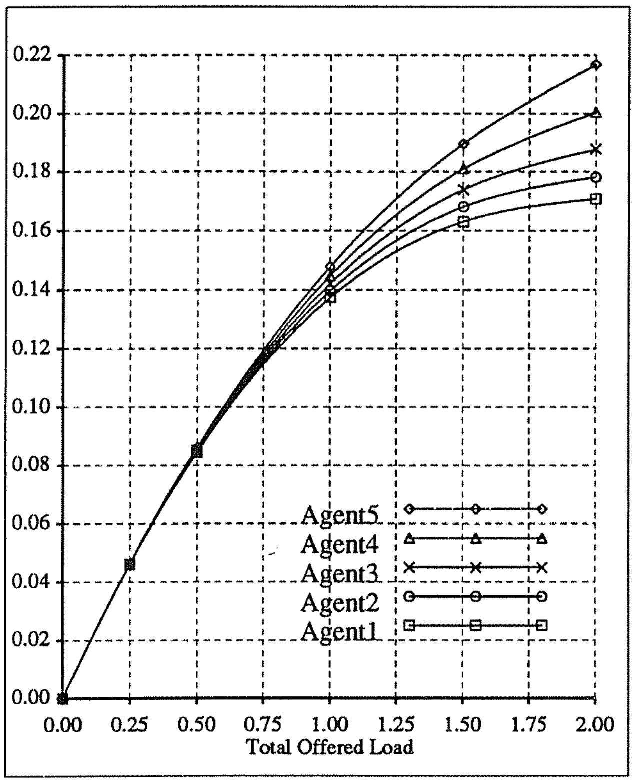


Figure 4.2b: Agent Throughputs for Protocol I (Batch Reset Overhead = 0.5)

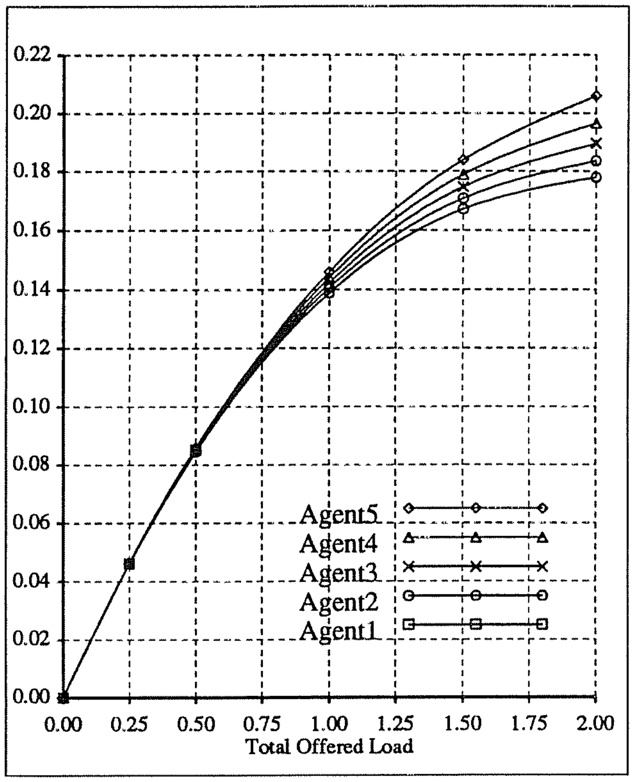


Figure 4.2c: Agent Throughputs for Protocol I (Batch Reset Overhead = 0)

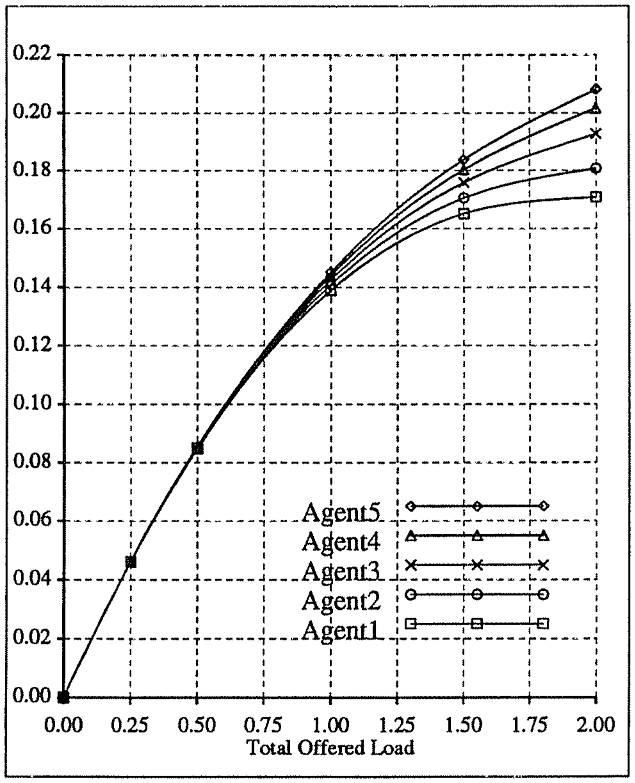
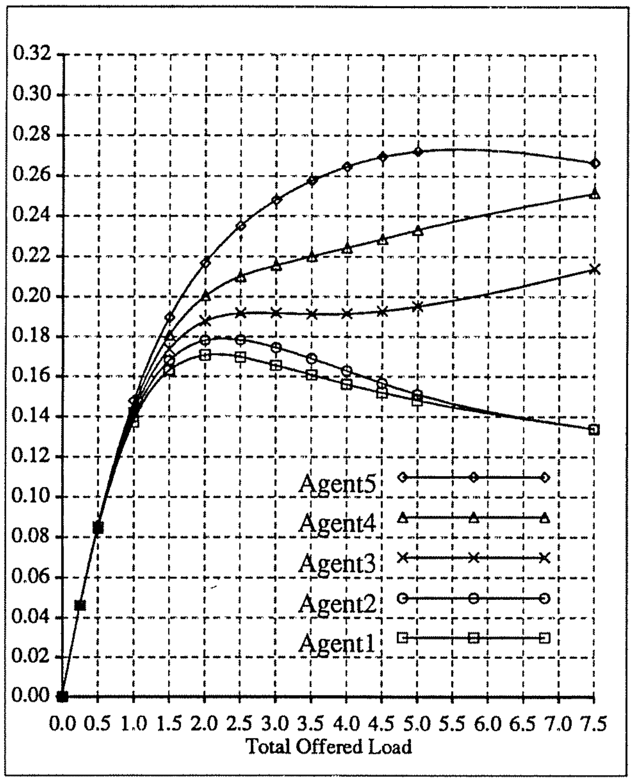


Figure 4.2d: Agent Throughputs for Protocol II



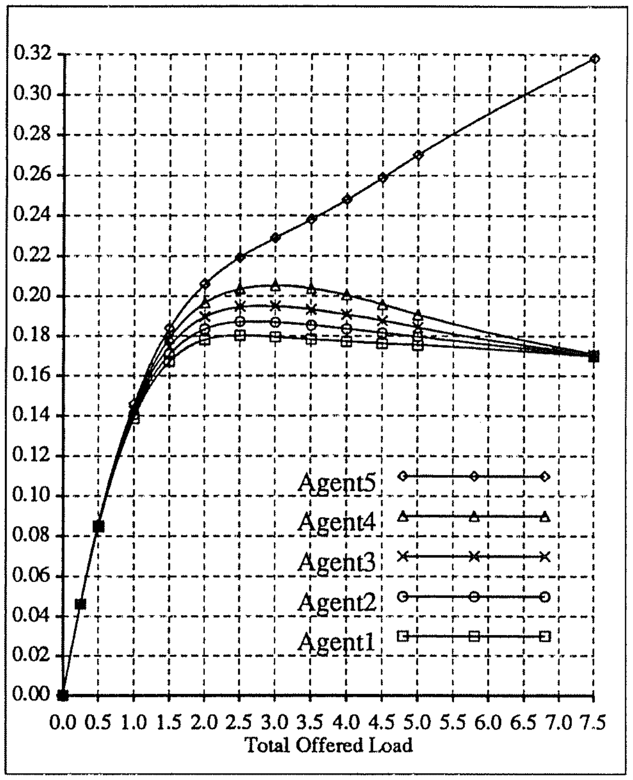


Figure 4.3a: Asymptotic Throughputs for Protocol I (Batch Reset Overhead = 0.5)

Figure 4.3b: Asymptotic Throughputs for Protocol I (Batch Reset Overhead = 0)

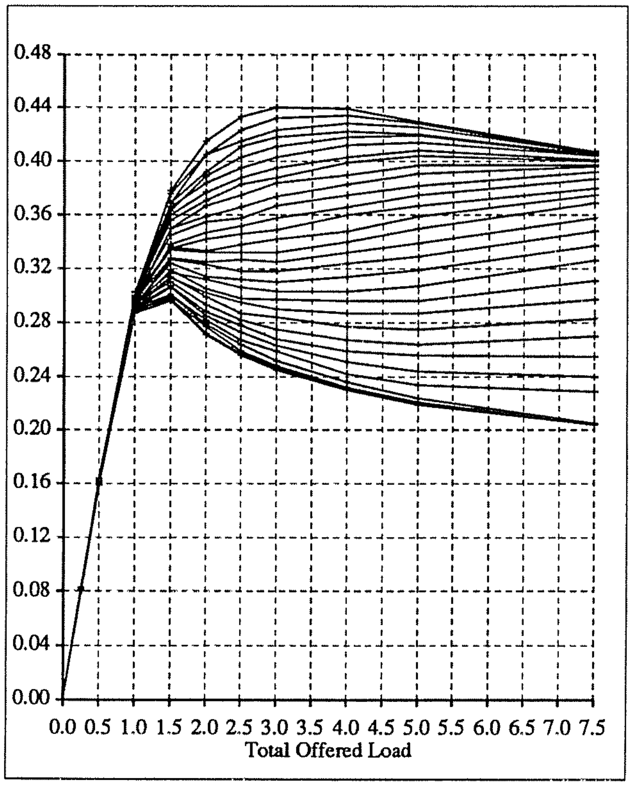


Figure 4.3c: Asymptotic Throughputs for Protocol I - Simulation with 30 Agents (Batch Reset Overhead = 0)

The first observation we make from the graphs is that the simulation results for thirty agents show the same general behavior of the protocols as the analytical results for five agents. This is also true for the other experiments reported in this section. Thus, we conclude that the total offered load is the primary determinant of arbiter fairness, and can be varied by decreasing the interrequest time instead of increasing the number of agents in the model.

The second observation we make from the graphs is that the asymptotic fairness of Protocol II is actually quite different than the asymptotic fairness of Protocol I. In Protocol I, the amount of bandwidth allocated to each device is a continuum, with roughly half of the agents receiving progressively more than $\frac{1}{n}$ of the total available bandwidth, and the other half receiving progressively less than $\frac{1}{n}$ of the total bandwidth. (Note that when the batch reset overhead is 0.5 (0), the two (four) agents with the lowest assigned arbitration numbers miss out in every other batch, as pointed out in section 3.4.) In contrast, all agents in Protocol II except the two agents with lowest assigned arbitration numbers, receive roughly $\frac{1}{n}$ of the total available bandwidth as load increases. The reason that two of the agents are treated unfairly, and a simple correction to the protocol to solve this problem are discussed next.

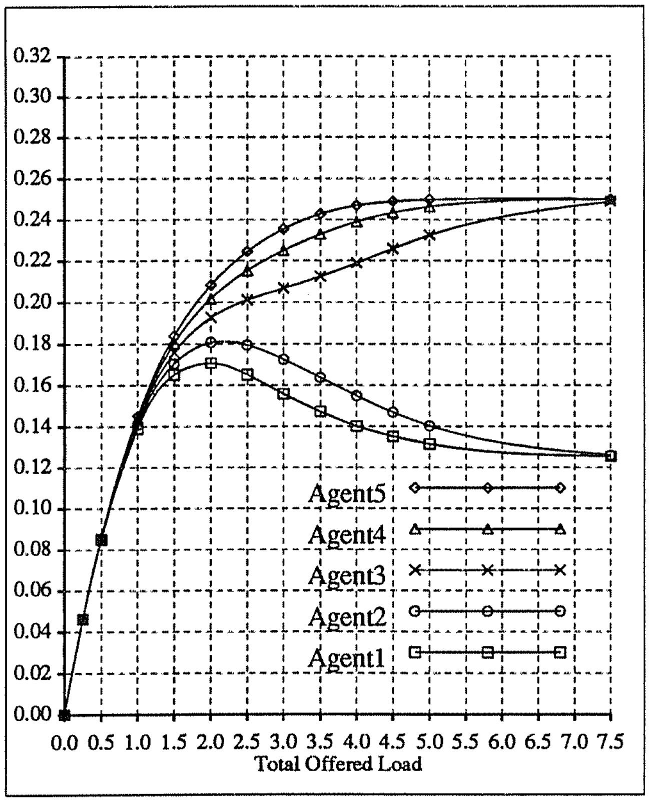


Figure 4.3d: Asymptotic Throughputs for Protocol II

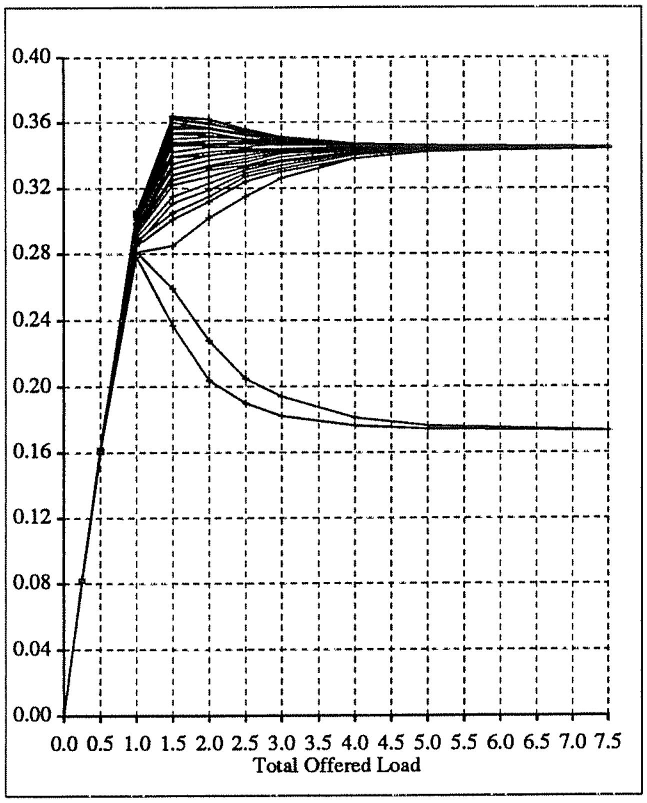
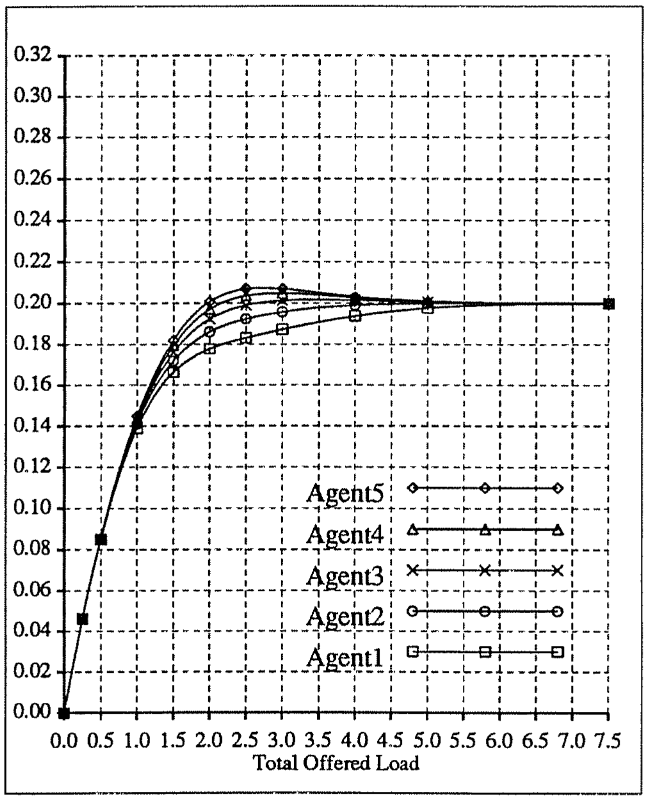


Figure 4.3e: Asymptotic Throughputs for Protocol II Simulation with 30 Agents

4.4. Proposed Modification to Fairness Protocol II

The results of Section 4.3 indicate an unexpected source of unfairness in protocol II. The cause of the unfairness is the rule that an agent marks itself as inhibited at the *end* of its bus transaction. Since arbitration for the next elected agent generally takes place concurrently with the bus transaction of a previously elected agent, a fairness release operation can take place concurrently with the bus transaction of the lowest-priority agent in a batch. This agent is then inhibited at the end of its transaction, and is counted as the first agent to be served in the new batch, rather than the last agent to be served in the old batch. Agents with the two lowest assigned arbitration numbers may thus only be able to compete in every other batch during periods of very high demand on the bus.

We propose a simple modification to protocol II, which dramatically improves the asymptotic fairness of this protocol. The new rule in our *Modified Protocol II* is that an agent marks itself as inhibited at the *beginning* of its bus transaction (or possibly when it wins an arbitration), before a new arbitration or fairness release operation is allowed to take place. If a fairness release operation is overlapped with the agent's bus transaction, it will become uninhibited



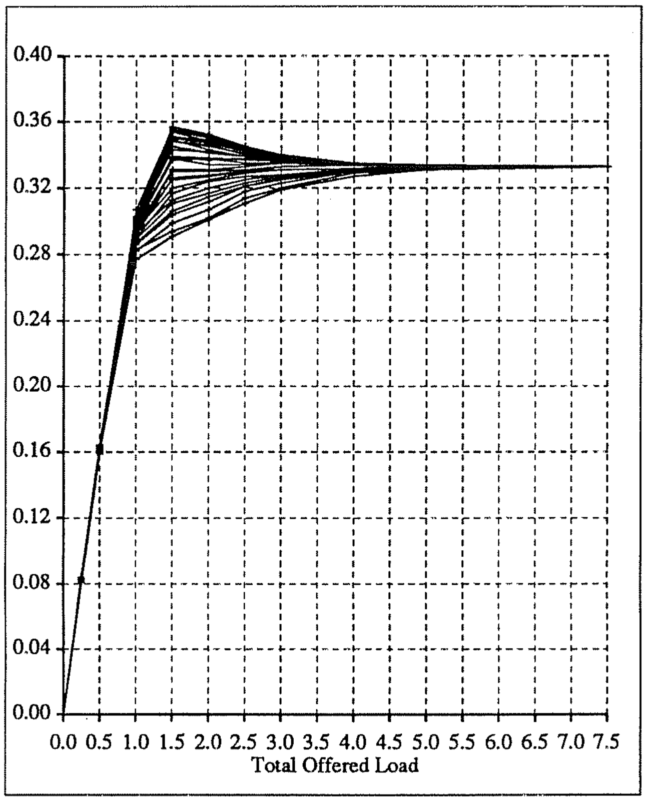


Figure 4.4a: Agent Throughputs for Modified Protocol II

Figure 4.4b: Agent Throughputs for Modified Protocol II

during its bus tenure. The bus transaction then counts as the last transaction in the old batch, as should be the case.

A simple change is required in the net of Figure 3.2 in order to model the Modified Protocol II. In this model, agent 1 places a token in AI1 when transition T2 completes, rather than when transition T3 completes. An analogous change is required in the net for each agent.

We have analyzed the model for the modified protocol, and the asymptotic curves are given in Figures 4.4a-b. Figure 4.4a gives the analytic results for five agents; Figure 4.4b gives the simulation results for thirty agents. We see that as load increases, the modified protocol is perfectly fair; however for total offered load of 2.0 to 5.0, the protocol still allocates bandwidth unequally.

4.5. Summary and Comparison of the Arbitration Protocols

In order to compare the fairness of the arbitration protocols more precisely, we use the ratio of the throughput achieved by agent *n* to the throughput achieved by agent 1, where *n* is the number of agents on the bus. This ratio is

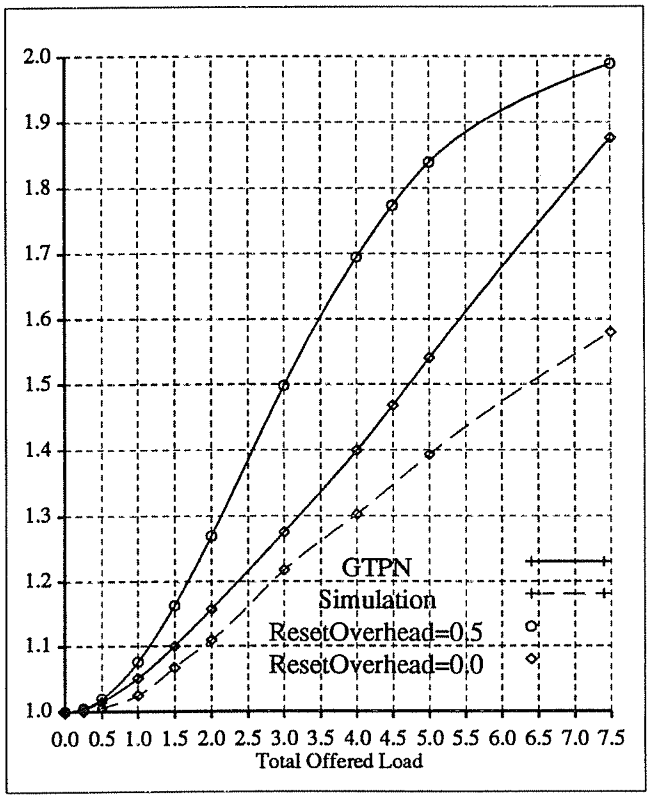


Figure 4.5a: Throughput Ratio vs Total Offered Load (Protocol I)

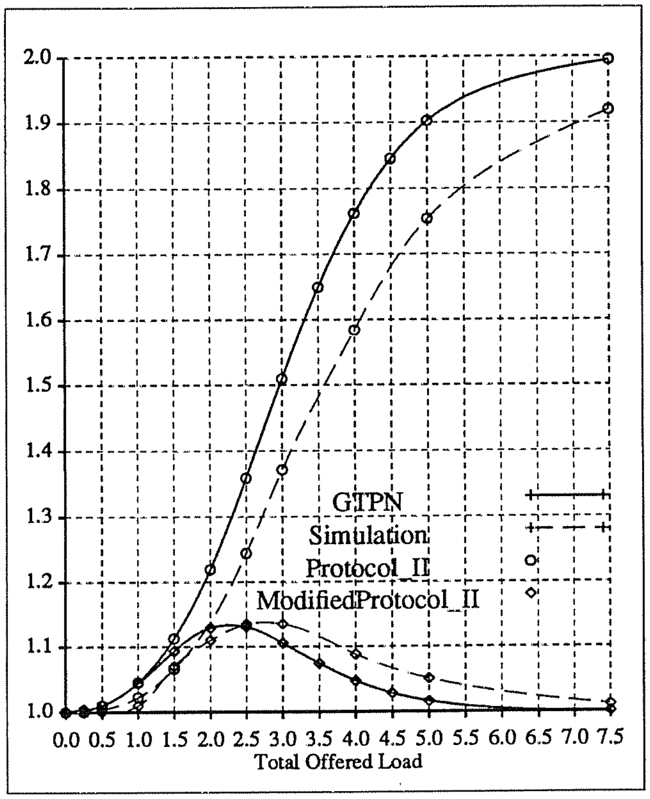


Figure 4.5b: Throughput Ratio vs. Total Offered Load (Protocol II and Modified Protocol II)

plotted as a function of total offered load in Figures 4.5a-c. In all cases, the ratio asymptotically approaches 2.0 as load increases.

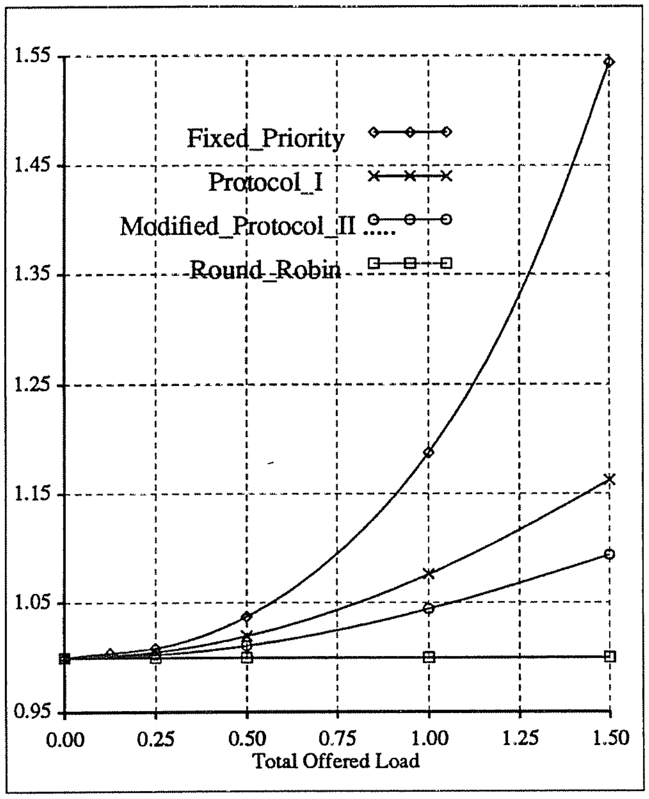
In Figure 4.5b, The top two curves are for Protocol II, and the bottom two curves are for Modified Protocol II. The dashed curves are obtained by simulating the five-agent model with exponential interrequest times. It's clear that the modification to Protocol II significantly improves its asymptotic fairness characteristics.

Figures 4.5c and 4.5d summarize the throughput ratios for the fixed priority protocol (top curve) Protocol I with batch reset overhead of 0.5 (next curve) Modified Protocol II (next curve), and the perfectly fair round-robin protocol (curve at 1.0). We can see that the existing fairness protocols for the parallel contention arbiter allocate bandwidth unequally to agents with equal need for the bus at loads which are very high, but which may occur during periods of peak usage during system operation. At these loads, the most favorably treated agent may receive 5-15% more bandwidth than the least favorably treated agent. In the next section we present a new protocol for the parallel contention arbiter that is based on fixed arbitration numbers, but which implements the perfectly fair round-robin protocol.

5. A New "Fairness" Protocol: Round-Robin

The most obvious way to implement round-robin scheduling in the parallel contention arbiter is to use a dynamic assignment of arbitration numbers for each agent. However, more careful thought reveals that a simple protocol which uses fixed arbitration numbers is possible. To see this, note that if an agent with assigned number j is the winner of a given arbitration, then the round-robin algorithm will scan agents with assigned numbers $j-1$ through 1 and then agents with numbers N through j in the next arbitration. Our round-robin protocol is based on the important observation that the parallel contention algorithm will implement this round-robin scan if we provide a mechanism to specify that agents with arbitration numbers $j-1$ through 1 have absolute priority over agents with arbitration numbers N through j, and then let the assigned numbers determine the ordering within each class.

The simplest implementation of this round-robin protocol requires an extra bit in the arbitration numbers (i.e. an extra line on the bus). We call this bit the *round-robin bit*. The round-robin bit is treated as the most significant bit of the agent's number during an arbitration. Each agent records the arbitration number of the winning agent at the end of an arbitration, excluding the round-robin bit. If the agent's arbitration number is less than this value and the agent competes in the next arbitration, the agent asserts the round-robin line.



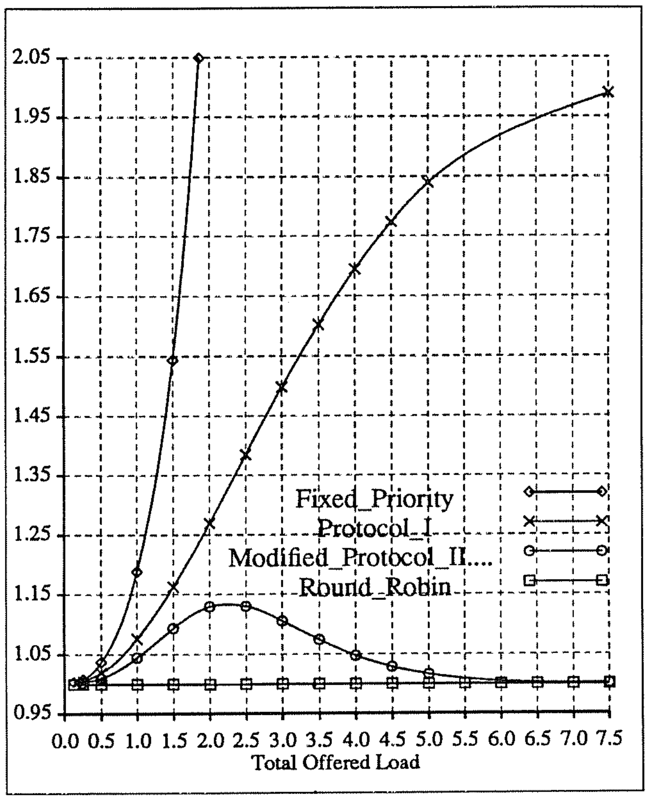


Figure 4.5c: Throughput Ratio vs Total Offered Load (Four Protocols)

Figure 4.5d: Asymptotic Throughput Ratios

The logic needed to implement this protocol primarily consists of a register to store the winning identity, and a comparator to determine if the agent's assigned arbitration number is smaller than the recorded value. The output of the comparator is input to the round-robin priority bit of the agent's arbitration number. Note that this logic replaces the logic required to implement either of the assured access protocols described in Section 2.2.

6. Conclusions

In this paper we have quantified the efficiency and fairness of the most common multiprocessor bus arbitration protocols. We have shown that the daisy chain arbiter is reasonably efficient for five to ten agents on the bus, assuming bus utilizations do not exceed 60%. We have also shown that the parallel contention and central arbiters have nearly optimal efficiency for as many agents as one might consider attaching to the bus and over all possible values of total offered load.

In terms of fairness, the fixed priority protocol allocates bus bandwidth surprisingly equitably if bus utilization is below 40%. The two distinct "fairness protocols" adopted for the basic parallel contention arbiter in the Futurebus, Fastbus, NuBus, and Multibus II standards, allocate bus bandwidth very fairly at bus utilizations below 50% and reasonably fairly at utilizations between 50 and 60%. We found agent throughput ratios to be the most useful measure of arbiter fairness. At utilizations above 60%, the bus allocation is substantially unequal in existing fairness protocols, and in the limit of extremely high load, the most favorably treated agent is allocated twice the bus bandwidth of the least favorably treated device in both fairness protocols.

We discovered that the asymptotic behavior of the fairness protocol adopted in Futurebus is quite different than the asymptotic behavior of the fairness protocol adopted in the other standards. In particular, a small modification to the Futurebus protocol proposed in section 4.4 makes the protocol perfectly fair in the limit of extremely high offered load.

Our analytical results do not argue strongly in favor of changing existing standards or implementations, since unfairness only arises at peak bus loads. However, we presented a new protocol for the parallel contention arbiter in section 5 that implements the round-robin protocol of central round-robin arbiters. The new protocol is as efficient as the existing protocols, is simpler to implement, and is perfectly fair. It thus provides an attractive alternative when designing a new standard or a new system bus.

Acknowledgements

We would like to thank David Gustavson, Robert Downing, John Theus, Rae McLellan, and Jim Goodman for their helpful suggestions.

References

[Baer80] Baer, J.-L., *Computer Systems Architecture*, Computer Science Press, 1980.

[BaAh81] Bain, W. L., and S. R. Ahuja, "Performance Analysis of High-Speed Digital Buses for Multiprocessing Systems," *Proc. 8th Int'l. Symp. on Computer Architecture*, Minneapolis, Minn., May 12-14, 1981, pp. 107-133.

[Borr88] Borrill, Paul, private communication, June 1988.

[Gust84] Gustavson, D. B., "Computer Buses - A Tutorial," *IEEE Micro*, Vol. 4, No. 4, August 1984, pp. 7-22.

[Gust86] Gustavson, D. B., "Introduction to the Fastbus," *Microprocessors and Microsystems*, Vol. 10, No. 2, March 1986, pp. 77-85.

[Intel] Intel Corporation, *Multibus II System Specification.*

[KlCa86] Kleeman, L., and A. Cantoni, "The Analysis and Performance of Batching Arbiters," *Proc. of Performance '86 and ACM SIGMETRICS 1986 Joint Conf. on Computer Performance Modelling, Measurement and Evaluation*, Raleigh, NC, May 27-30, 1986, pp. 35-43.

[PWGr86] P1196 Working Group, *NuBus - A Simple 32-Bit Backplane Bus P1196 Specification*, Draft 2.0, IEEE Microprocessor Standards Committee, December 15, 1986.

[ShAh81] Sharma, D. K., and S. R. Ahuja, "A First-Come-First-Serve Bus Allocation Scheme Using Ticket Assignments," *The Bell System Technical Journal*, Vol. 60, No. 7, September 1981, pp. 1257-1269.

[Taub84] Taub, D. M., "Arbitration and Control Acquisition in the Proposed IEEE 896 Futurebus," *IEEE Micro*, Vol. 4, No. 4, August 1984, pp. 28-41.

[Taub87] Taub, D. M., "Improved Control Acquisition Scheme for the IEEE 896 Futurebus," *IEEE Micro*, Vol. 7, No. 3, June 1984, pp. 52-62.

[UKPS66] UK Patent Specification No. 1099575, Jan. 17, 1966, (no inventor specified).

[VeMa88] Vernon, M. K., and U. Manber, "Distributed First-Come First-Serve and Round Robin Protocols and Their Application to Multiprocessor Bus Arbitration," *15th Annual Int'l. Symp. on Computer Architecture*, Honolulu, Hawaii, May30-June 2, 1988.

Appendix: GTPN Models of Fixed Priority and Round Robin Protocols

The model for the daisy chain arbiter fixed priority protocol is shown in Figure A.1 and Table A.1. Transitions T0-T3 model the first agent on the chain, transitions T4-T7 model the second agent, and the model is extended to more agents in the obvious way. Assume the second agent makes a request and moves a token to R2. The bus grant signal propagates from the head of the chain (ST) through transitions T3 and T7 to G2. The token in G2 allows T6 to fire and the agent is serviced. T6 returns the grant token to ST. Note that the duration of T3 is half that of T7. If another agent is added, the grant signal transition T11 would have a duration of "station delay" units. Performance measures for the daisy chain model are obtained as in the fairness protocol models.

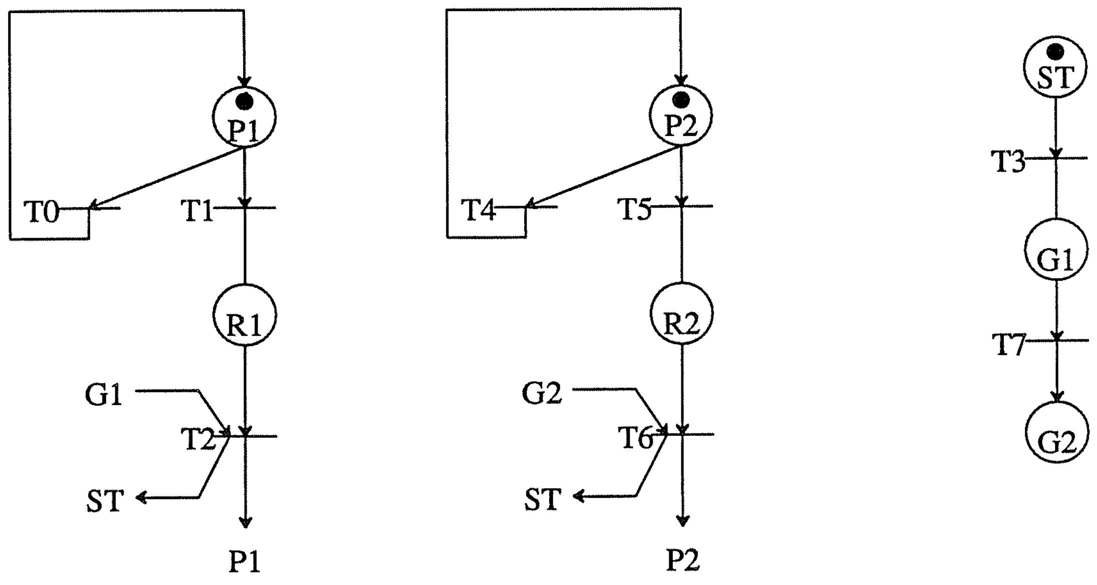


Figure A.1: Daisy Chain Net

Table A.1: Transition Delays and Frequencies for Daisy Chain Net

Transition	Duration	Frequency
T0	*step size*	1 - p
T1	*step size*	p
T2	1.0	1
T3	*station delay/2*	((R1 = 1) or (R2 = 1))
T4	*step size*	1 - p
T5	*step size*	p
T6	1.0	1
T7	*station delay*	((R1 = 1) or (R2 = 1))

The model of the central or parallel contention arbiter fixed priority protocol can be constructed from the daisy chain arbiter model by setting the grant propagation delay ("station delay") to 0. A more efficient model is obtained by deleting the transitions that implement these delays (i.e. T3 and T7 in Figure A.1), and implementing the priority ordering in the frequency expressions for the bus service transitions (i.e. T2 and T6 in Figure A.1).

The model of the round robin mechanism is shown in Figure A.2 and Table A.2. In this model, transitions T2-T6 model the first agent and T7-T11 model the second agent. Unlike the previous models there are no priorities associated with each agent. Transitions T2 and T3 model the geometric interrequest time for agent 1. The loop including T6 and T11 is used to decide which agent gets access next. When agent 1 requests it moves its token to R1. If there is a token in N1 than the request is allowed to move to B1 and get service. When it finishes service, the token is returned to P1, but the token from N1 is now moved to N2 to give agent 2 a chance to use the bus. If agent 2 has not requested service and agent 1 requests again, T11 fires, moving the token to N1 so that T4 can fire again. In this way, each agent uses the bus if it needs it every time the token cycles through the T6/T11 loop. Places ID and IU are used to tell when the bus is idle. When both agents are in P1/P2 or T2/T7 the bus is idle and T0 will fire. Now when a token moves to B1 or B2 it must first wait for the non-overlapped arbitration time to occur which is modeled by T1.

The round robin model is extended to three agents by adding another set of places and transitions like T1-T5 plus adding another place N3 with its input arc coming from N2 and its output arc going to N1. In this case the frequency expression for T11 would be ((R2 = 0) × ((R1 = 1) or (R3 = 1))). Performance measures are obtained from the model as before.

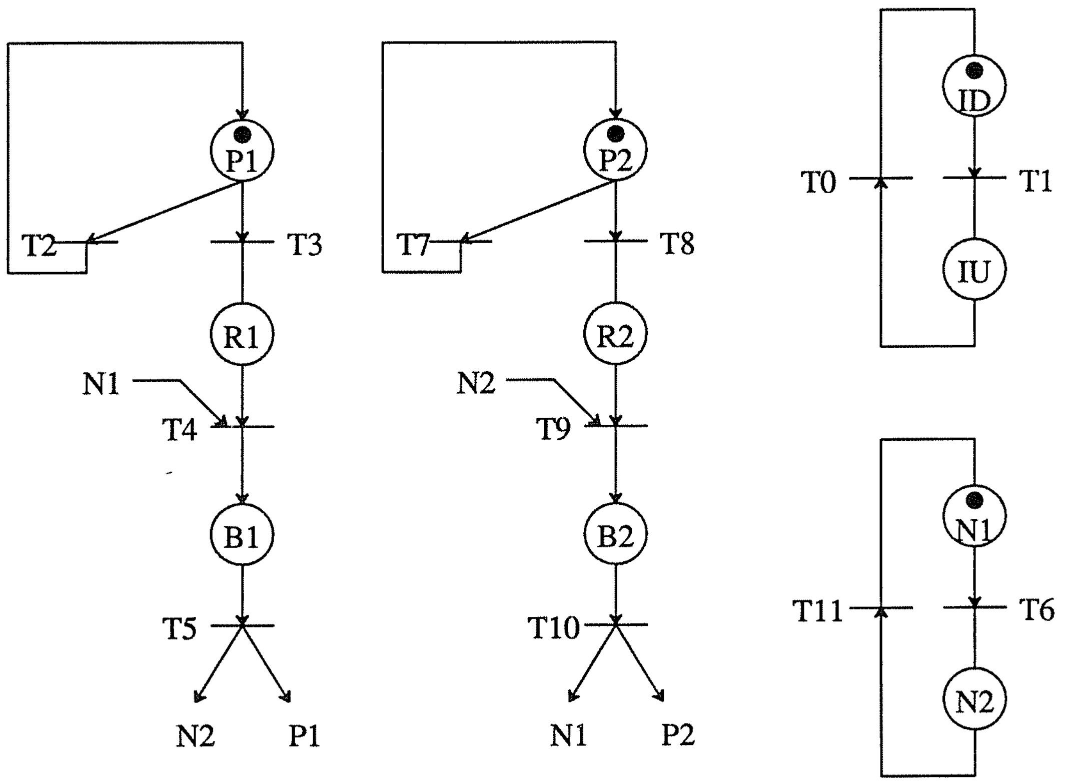


Figure B.1: Round Robin Net

Table B.1: Transition Delays and Frequencies for Round Robin Net

Transition	Duration	Frequency
T0	0	((T3 = 0) × (T8 = 0) × (R1 = 0) × (R2 = 0) × (T4 = 0) ×
		(T9 = 0) × (B1 = 0) × (B2 = 0) × (T5 = 0) × (T10 = 0))
T1	*arbitration time*	(B1 = 0) or (B2 = 0)
T2	*step size*	1 - p
T3	*step size*	p
T4	0	1
T5	1.0	(ID = 0) × (T1 = 0)
T6	0	(R1 = 0) × (R2 = 1)
T7	*step size*	1 - p
T8	*step size*	p
T9	0	1
T10	1.0	(ID = 0) × (T1 = 0)
T11	0	(R2 = 0) × (R1 = 1)