Empirical Evaluation of Distributed Mutual Exclusion Algorithms

Shiwa S. Fu, Nian-Feng Tzeng, and **Zhiyuan Li**

Center for Advanced Computer Studies
University of Southwestern Louisiana
Lafayette, LA 70504

Abstract

In this paper, we evaluated various distributed mutual exclusion algorithms on the IBM SP2 machine and the Intel iPSC/860 system. The empirical results are compared in terms of such criteria as the number of message exchanges and the response time. Our results indicate that the Star algorithm [2] achieves the shortest response time in most cases among all the algorithms on a small to medium sized system, when processors request for the critical section many times before involving any barrier synchronization. On the other hand, if every processor enters the critical section only once before encountering a barrier, the improved Ring algorithm [4] is found to outperform others under a heavy load; but the Star algorithm and the CSL algorithm [3] prevail when the request rate becomes light. The best solution to mutual exclusion in distributed memory systems is determined by how participating sites generate their mutual exclusion requests.

1 Introduction

Mutual exclusion is achieved by a mechanism that ensures involved sites to get access to a designated section of code (called the *critical section*) in a mutually exclusive way. The focus of this paper is on evaluating current distributed mutual exclusion algorithms on two real machines, comparing their behaviors in terms of the mean number of messages exchanged per critical section entry (or NME for short) and the response time. Four algorithms are compared, including the Raymond algorithm [1], the Neilsen and Mizuno's algorithm [2] with star topology (called the Star algorithm), the improved Ring algorithm [4], and the Chang, Singhal, and Liu's algorithm (i.e., CSL in short) [3]. Our improved Ring algorithm is a variation of that described earlier in [5] but exhibits improved performance due to the elimination of unnecessary messages, as detailed in Sec. 2.4. In addition, we also introduce and evaluate a modified Raymond algorithm, which gives rise to better performance than the Raymond algorithm.

In order to observe the actual behaviors of these algorithms, we implemented them on the IBM's Scalable POWERparallel System 2 (SP2) and the Intel *iPSC*/860. We carried out our study on the IBM SP2 machine of size up to 64, and on the Intel *iPSC*/860 of size 16 (which is the largest subcube available to users). Our results on both distributed-memory machines suggest that the Neilsen and Mizuno's Star algorithm [2] outperforms all other algorithms with respect to the response time for most cases, when the critical section is requested by processors repeatedly and no barrier is involved in the meantime. This is due to the following two facts: (1) the Star algorithm has the lowest NME unless the request rate is extremely high, and (2) while all processors contend for the critical section initially, fewer and fewer processors experience contention gradually, as only one processor is permitted to enter/leave the critical section at a time in sequence and a processor does not generate another critical section request until its earlier request has been served. Consequently, the processors gradually "serialize" their generation of critical section requests and soon avoid most contention, as long as no barrier is involved in the meantime. If every site issues just one request to the critical section before involving in barrier synchronization, our improved Ring algorithm exhibits the best performance for a high request rate; but the Star and the CSL algorithms are superior to others for a low request rate.

A simulation study was performed previously by Johnson [6] to contrast mutual exclusion algorithms described in [1, 2, 3], mainly on the basis of NME. That simulation study examined a different construct of the Star algorithm [2], with its logical structure changed dynamically, as opposed to a fixed structure with a fixed site as the root in our implementation. When the root of the Star algorithm is fixed, it requires no more than 3 messages exchanged per critical section entry, exhibiting better performance than a construct with its structure changed dynamically. The Raymond algorithm studied in that simulation study is limited to its original form, but our work here introduces a modified version of the algorithm and compares the behaviors of both Raymond algorithms. We also present the results of our improved Ring algorithm. Our empirical evaluation gives rise to a different conclusion than the earlier simulation study performed in [6] does.

2 Distributed Mutual Exclusion Algorithms

Distributed mutual exclusion algorithms can be divided into two classes: (1) *permission-based* algorithms, where all involved sites vote to select one which receives the permission to enter the critical section, and (2) *token-based* algorithms, in which only the site with the token may enter the critical section. In general, a permission-based algorithm involves higher communication traffic overhead than a token-based algorithm. We therefore focus our attention to token-based algorithms in our empirical study. The algorithms under this study are reviewed briefly in sequence.

2.1 Raymond Algorithm

The Raymond algorithm [1] determines and maintains a *static* logical structure. The logical structure (for example, a spanning tree) is kept unchanged throughout its lifetime, but the directions of edges in the structure change dynamically as the token migrates among sites, in order to point toward the possible token holder. The directions of edges in the structure always point to the possible token

This material is based upon work supported in part by the NSF under Grants MIP-9201308 and CCR-9300075. We wish to thank Argonne National Laboratory and *ERC* for Computational Field Simulation, Mississippi State University, for allowing our access to the IBM SP2 machine and the Intel *iPSC*/860 system, respectively. The IBM SP2 machine belongs to the Argonne High-Performance Computing Research Facility, which is funded principally by the U.S. Department of Energy, Mathematical, Information and Computational Sciences Division (ER-31).

holder, making the token holder a sink node in the structure. Each site has a local queue to hold requests coming from its neighbors and itself, and has only one outstanding request at any given time, resulting in the local queue length no more than the node degree of the embedding structure.

Each site wishing to enter the critical section inserts its local request to the rear of its local queue, so that all requests appeared at that site in a first-come-first-served order. While it is possible to get better performance by inserting a locally generated request at the front of the local queue, referred to as the eager Raymond algorithm (because the local site is then allowed to enter the critical section immediately when the token reaches the site) [1], this tends to pose a concern on the fairness of requests and is not considered here.

2.2 Modified Raymond Algorithm

In the Raymond Algorithm described above, a token request always follows the token from an intermediate site whose local queue contains more than one element. This situation happens more frequently as the critical section request rate grows. We introduce a simple modification to lower communication traffic by eliminating the token request from a site whose local queue contains multiple elements. Instead of sending a separate token request, the site marks in the token message the situation that the token has to come back later on. A marked token causes an enqueueing operation at the receiving site, recording that the token will be sent back along the link from which it gets to the site. This combines the token message with a subsequent token request message at every site whose local queue length is greater than 1, effectively lowering mutual exclusion traffic and thus improving performance.

2.3 Neilsen and Mizuno's Star Algorithm

Instead of passing the token step by step through intermediate sites in the logic structure to the token requestor as in the Raymond algorithm [1], Neilsen and Mizuno proposed an algorithm where the token holder can send the token directly to the requesting site with one message [2]. This is made possible by attaching the requestor's ID in the request message so that the token holder knows, on receiving the message, who is the requestor.

One special case of this algorithm is that the logical structure can be a fixed star topology (called the Star algorithm). Under such a situation, any site ready to enter the critical section always sends a request message attached with its own ID directly to the root node. The root node makes it possible to establish a distributed waiting queue (of all requesting sites) by recording the site which has most recently requested the token (and is the tail site in the distributed waiting queue). When receiving a request message, the root forwards the message to the tail site (of the queue) and updates its record, unless the root itself holds the token. On receiving a request message, the token holder, if not in need of the token, forwards the privilege to the requestor directly using a token message. A very attractive property of the Star algorithm is that it always takes three (3) exchange messages for a requestor to get the token, if the root does not own the token, and only two (2) messages if the root holds the token.

2.4 Improved Ring Algorithm

The algorithm proposed in [5] establishes a static logical ring over all sites and allows the token to move along a fixed direction, in response to a token request traveling along an opposite direction. The logic ring and the direction of its links are all kept unchanged. When ready to enter the critical section, a site without the token, say S_w, must send a request messages to its successor, site $S_{(w+1) \bmod N}$, and then goes to WAIT state until it receives the token, where N is the system size. If $S_{(w+1) \bmod N}$ is not the token holder, it sends a request message to its successor, site $S_{(w+2) \bmod N}$. This process repeats until the site with the token, say S_h, receives a request message from its predecessor. All sites within S_w and S_h (along the direction of the request message traversals) are all at the *SUBS* (short for substitute) state. On receiving a request message, the token holder, if not in need of the token, forwards the privilege to its predecessor using a token message. The token is then forwarded by the SUBS sites in sequence to site S_w (along the reverse direction of the request message). If the number of SUBS sites is α, $0 \leq \alpha < N-1$, the total number of messages exchanged for S_w to get the privilege of entering the critical section equals $2\times(\alpha + 1)$.

We have introduced a variation of the Ring algorithm to achieve improved performance recently [4]. This improvement results from the following two facts: (1) it is possible that SUBS sites may also wish to enter the critical section in the meanwhile, and such a site is allowed to do so by holding the token until it finishes with the critical section after it receives the token message from its successor, and (2) a SUBS site does not have to issue any message to its successor for requesting the token, because the token will traverse the site on its way to S_w (i.e., the token requestor). In the best scenario, as many as $(\alpha + 1)$ sites (i.e., those α SUBS sites and S_w) may enter the critical section in sequence with $2\times(\alpha + 1)$ messages exchanged, giving rise to NME = 2. This attractive property occurs under a heavy request load for the critical section. However, the NME could be as large as $2N$ for a system with N sites in the worst case, as the token message might travel one revolution (in response to the request message) to serve a requestor under a very light request load. Our experimental study considered this variant ring algorithm, referred to as the improved Ring algorithm [4], for it results in better performance than that proposed in [5].

2.5 Chang, Singhal, and Liu Algorithm

Chang, Singhal, and Liu's algorithm [3] maintains a list which links all requesting sites (i.e., a distributed queue), such that each requesting site records (using variable **Next**) only the identifier of its next requesting site, thereby simplifying the data structure of token message [3]. The logical structure in the CSL algorithm is a star topology initially, and it changes dynamically as the algorithm proceeds. A site is the tail in the distributed queue, if it is waiting for the token and its **Next** is **NIL**. If its **Next** is not **NIL**, its successor site in the distributed queue is pointed by **Next**. As a result, when a request message arrives at a site which is the tail in the distributed queue, the site simply sets its **Next** to the requesting site. If a request message arrives at a site which neither holds nor is requesting the token, or which is requesting the token but its **Next** is not **NIL**, the request message is forwarded to the possible token holder (pointed by variable **NewRoot**) to form a distributed queue; **NewRoot** is then set to point to the current requestor because it will eventually becomes the new token holder. On sending the token message to the 'next' site, the variable **NewRoot** is piggybacked in the token message so that the 'next' site can update its **NewRoot** accordingly.

The NME complexity of this algorithm depends on the height of the logical tree, and it is $O(\log N)$ per critical section entry, where N is the system size. Because its

logical structure changes dynamically, the CSL algorithm fails to exhibit as good performance as the Star algorithm, where the structure is kept unchanged.

2.6 Summary

A summary of the four algorithms considered in this study is provided in Table 1, where the NME complexity of the Raymond algorithm is $O(\log N)$, so are the modified Raymond algorithm and the CSL algorithm; it is $O(N)$ for the improved Ring algorithm, and about 3 or less for the Star algorithm. The token message and the request message used to communicate among sites are of fixed sizes for all these algorithms. Only the CSL algorithm maintains a *dynamic* logical structure, while others construct *static* logical structures throughout lifetime. The Raymond and the modified Raymond algorithms require a local queue in each site to hold the requests (which are no more than the node degree of the logical structure), but all other three algorithms require no local queue at any site.

Table 1. Comparison of various algorithms

	(Modified) Raymod	Improved Ring	CSL	Star
NME	$O(\log N)$	$O(N)$	$O(\log N)$	≤ 3
Local queue	$\leq$ degree of structure	no	no	no
Token size	fixed	fixed	fixed	fixed
Req msg size	fixed	fixed	fixed	fixed
structure	static	static	dynamic	static

3 Performance Evaluation

3.1 Platforms and Implementation Details

Our experiment was conducted on both the IBM SP2 machine at Argonne National Laboratory and the Intel *iPSC*/860 system at Mississippi State University. The SP2 involves 128 nodes, which are standard POWER2 Architecture RISC System/6000 processors with speed of 66.7 MHz. The latency for sending a message from one node to another involves (1) software overhead to initiate a send operation, and (2) hardware overhead of $46+0.035m$ μs [7], where m is the message size in bytes. According to our measurement on the IBM SP2 at ANL, it takes about 110 μs to send a one-byte message to another node in the system. The Intel *iPSC*/860 is a distributed memory system, with its nodes interconnected according to the hypercube topology. Each cube node is a 40 MHz *i*860 processor with 8 MBytes of memory. The time taken to deliver a message can be expressed by $95+0.394m+10.3d$ μs for m bytes over distance d [8], with the distance between two neighboring nodes being 1. Our measurement reveals that it takes roughly 150 μs to transfer a one-byte message between two nodes of 1 unit distance.

The static logical structure (i.e., a ring) for the improved Ring algorithm is embedded in a straightforward way in which site S_x is connected to site $S_{(x+1) \bmod N}$, where x is the logical ID of a particular node and N is the allocated system size. The token is assigned to an arbitrary site initially. For the (modified) Raymond algorithms, a binary spanning tree is embedded across involved nodes. A star topology is embedded for both the CSL algorithm and the Star algorithm after the token is assigned to an arbitrary site as the root. The Star algorithm in our implementation follows that described in [8] with a slight modification to maintain the fixed star structure throughout its lifetime. In our experiment, every involved site produces mutual exclusion requests in a random fashion governed by a random number generator with a mean value of Γ (which is the mean time between releasing the critical section and requesting it again by a typical site). The time taken by a site to execute the critical section is specified by ξ. The performance measures of interest include (1) NME and (2) the response time, which is the average time from requesting the token to finishing the critical section by a site. A small NME value indicates light traffic overhead, while a short response time reflects assigning the critical section to waiting sites effectively. All sites execute the same code and stop execution after any one of them reaches 10000 critical section entries. The results illustrated in Figs. 1-2 are averaged values over all the involved sites.

3.2 Empirical Results and Discussion

A program code typically consists of the critical section part and the non-critical section part. Both parts are likely to have some read/write operations, and the critical section normally contains a few instructions. In a distributed memory system (like SP2 or *iPSC*/860), a read from or write to a remote memory location takes at least one message sending time, which is about 110 μs on SP2 and about 150 μs on *iPSC*/860. For convenience, the critical section duration (ξ) in our experiment is set to multiples of 100 μs.

In Fig. 1, the NME and the response time versus Γ obtained on SP2 under system size $N = 16$ are illustrated for these algorithms, where ξ is chosen 100 μs and Γ ranges from 300 μs up to 18000 μs. The improved Ring algorithm has the lowest NME value (of 2) under a heavy request rate (i.e., $\Gamma \leq 1000$ μs), but its NME grows quickly as Γ increases. This is because most intermediate sites are in the WAIT state under a heavy request rate, but fewer sites are in the WAIT state as the request rate decreases. The NME values of the both Raymond algorithm grow as Γ increases, but they flat off after $\Gamma > 6000$ μs (or 12000 μs). The modified version has a smaller NME than the original Raymond algorithm for any Γ. When compared with the improved Ring algorithm under a heavy request rate (say $\Gamma \leq 3000$ μs), the modified Raymond algorithm yields worse performance on both NME and the response time. The reason is that each site in the improved Ring algorithm then requires as few as two (2) messages (one request message and one token message) to accomplish one mutual exclusion entry, whereas each site in the tree structure of the modified Raymond algorithm requires more messages to do so, because the token message in the tree structure travels (in response to a request message) along each of the $(N-1)$ edges exactly twice in order to bring the token to all N sites in the tree.

It is observed from Fig. 1 that NME for the CSL algorithm increases slightly at the beginning but keeps flat afterward (i.e., $\Gamma > 3000$ μs). Although both the Raymond and the CSL algorithms have the same NME complexity of $O(\log N)$, the CSL algorithm yields better performance in regard to NME than the modified Raymond algorithm under a moderate to light request load, indicating that the dynamic logical structure used in the CSL algorithm can effectively adjust and evolve according to the occurrence of critical section requests, while the token message in the static structure used in the modified Raymond algorithm must move step by step through intermediate sites, which may not request the critical section in the meantime. However, the situation reverses under a heavy request load

because most of the intermediate sites are then ready for the critical section, while sites in the CSL algorithm still need to forward request messages to form a distributed queue. The Star algorithm keeps a fixed NME value (of about 3) throughout the simulated Γ values, as expected.

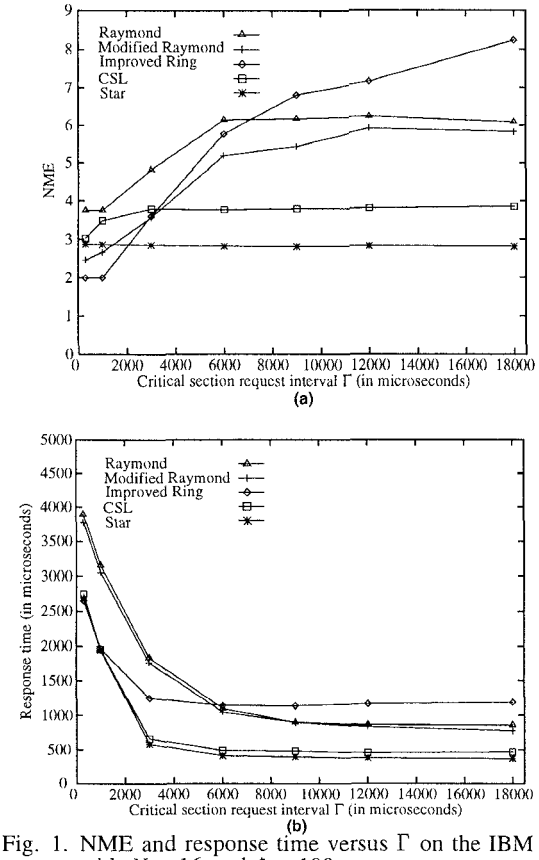

Fig. 1. NME and response time versus Γ on the IBM SP2 with $N = 16$ and $\xi = 100$ μs.

The response time is dictated by Γ as shown in Fig. 1(b). Under a heavy request rate, many sites wait to enter the critical section and a long waiting queue results, thus prolonging the response time. The response time curves of all these algorithms decrease gradually before flatting off as Γ increases. The Star algorithm has the best performance in regard to the response time under almost all the Γ values (except for $\Gamma = 100$ μs). The curve of the CSL algorithm stays close to that of the Star algorithm throughout the range of Γ examined. The improved Ring algorithm leads to the shortest response time when $\Gamma = 100$ μs (due to its low NME), but it has the longest response time as Γ goes beyond 6000 μs, since the token message must then go through many intermediate sites to reach the requestor. The gap between the Star algorithm (or the CSL algorithm) and the improved Ring algorithm (and the modified Raymond algorithm) tends to be large. Because the user cares more about the response time and the Star algorithm always gives rise to the smallest response time, the Star algorithm is considered the most favorable choice on SP2.

The NME and the response time versus Γ under these algorithms on *iPSC*/860 with $N = 16$ are shown in Fig. 2, where ξ and the Γ range are selected as above. Again, the improved Ring algorithm presents the smallest NME value for a high request rate, but its NME increases consistently and much faster than any other algorithm if Γ grows. As might be expected, our modified Raymond algorithm always yields better performance than the original Raymond algorithm. The improved Ring algorithm is superior to the modified Raymond algorithm under a heavy request rate (say $\Gamma < 2000$ μs). The Star algorithm exhibits better performance than the CSL algorithm throughout the Γ range considered. In fact, the Star algorithm always leads to the shortest response time among all algorithms, despite it has a slightly larger NME value than the improved Ring algorithm or the modified Raymond algorithm when the request rate is very high. When comparing Fig. 1(b) and Fig. 2(b), we find that the response time of a given case is more on *iPSC*/860 than on SP2, since communication takes longer on *iPSC*/860.

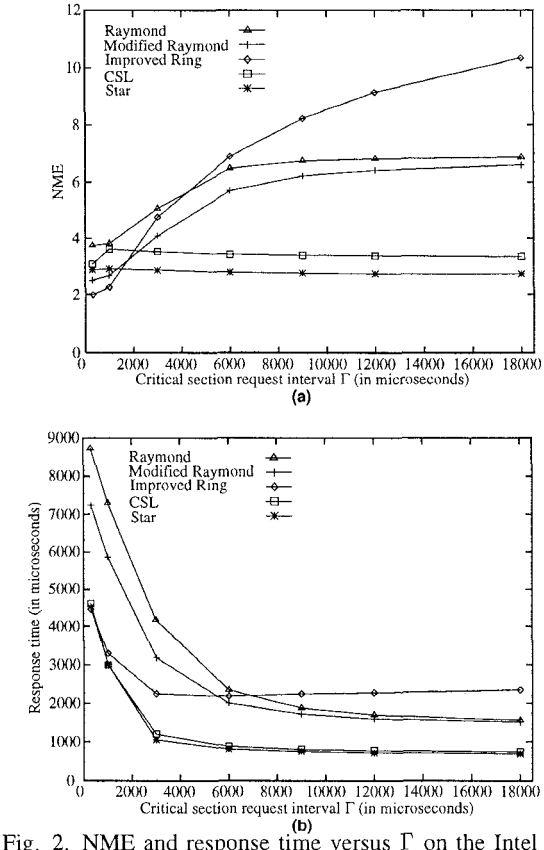

Fig. 2. NME and response time versus Γ on the Intel *iPSC*/860 with $N = 16$ and $\xi = 100$ μs.

3.3 Performance Under Different Numbers of Critical Section Entries

The previous section displays the results of various mutual exclusion algorithms under the situation that (1) every involved site produces a critical section request randomly after its earlier request gets served, and (2) every site makes a large number of critical section requests before encountering a barrier synchronization, if any. This situation exists in real application, like Traveling Salesman Problem developed at Rice University [10]. However, there are other situations where a participating site makes a few requests to the critical section before involving one barrier synchronization. An example of these situations can be found in such a benchmark code as Integer Sort (IS) in the NAS benchmark suite [9]; Specifically, a processor in IS requests the critical section exactly once before facing a barrier. Under these situations, all involved

sites after the barrier issue their critical section requests within a short period of time, causing high contention. The mutual exclusion algorithms under these situations are investigated here to unveil their behaviors.

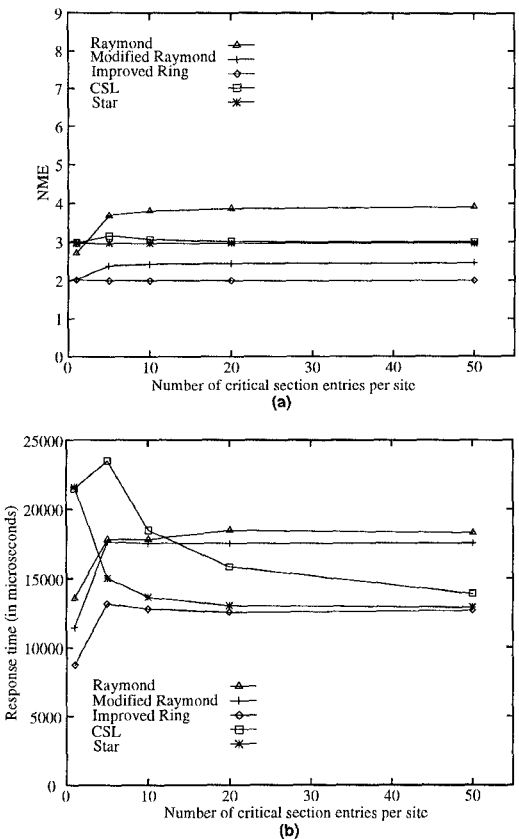

Fig. 3. NME and response time versus number of critical section entries achieved by each individual processor on SP2 with $N = 64$, under $\Gamma = 100$ μs and $\xi = 100$ μs.

The empirical study under these situations was conducted on the IBM SP2 machine with N equal to 64. Again, every involved site issues mutual exclusion requests randomly with a mean value of Γ. We plot NME and response time versus number of critical section entries in Fig. 3, where Γ and ξ are both set to 100 μs. When the number of critical section entry is 1, the improved Ring algorithm is observed to yield the shortest response time and the smallest NME, because each site in the ring structure sends one message to its successor and all the sites are ready for the critical section under this Γ value. In contrast, every site in the Star algorithm sends its critical section request message to the root at about the same time (under this Γ value), causing serious contention and thus exhibiting the worst response time. Similarly, the CSL algorithm also suffers from high contention and results in a long response time.

If each site enters the critical section multiple times before encountering a barrier during the course of program execution, the degree of contention subsides gradually. This is because a site normally does not produce another request for the critical section while waiting for the earlier request to be served, and the sites enter/leave the critical section one at a time. While all sites start generating requests for the critical section within a short time period initially, they generate their second critical section requests over a much longer period of time, reducing the degree of contention. All sites eventually "serialize" their generation of requests for the critical section and almost avoid producing critical section requests at the same time. This phenomenon makes the Star algorithm quickly become the most efficient algorithm, as can be observed in Fig. 3(b). These results reveal that the Star algorithm is preferred if each site requires to enter the critical section a number of times before encountering any barrier synchronization.

4 Conclusions

In this paper, we have investigated empirically various distributed mutual exclusion algorithms, including our proposed modified Raymond algorithm and improved Ring algorithm [4]. Our investigation was conducted on the IBM SP2 machine and the Intel *iPSC/860* system. The performance measures of interest are NME and the response time. If sites produce requests to the critical section repetitively many times before involving any barrier synchronization, the Star algorithm [2] achieves the best performance from the response time standpoint under almost all request loads, closely followed by the CSL algorithm [3]. On the other hand, the improved Ring algorithm [4] is found to yield the smallest NME under a heavy request load, but the Star algorithm leads to the lowest NME under a moderate to light request rate.

If every site enters the critical section only once before facing a barrier, different conclusions result, depending on the request rate. Under a heavy request load, our improved Ring algorithm seems to be most attractive, as it results in the smallest NME and the shortest response time. For a light request load, however, CSL and the Star algorithm prevail. As a result, the best algorithm for mutual exclusion in distributed memory systems depends on how involved sites produce mutual exclusion requests.

References

[1] K. Raymond, "A Tree Based Algorithm for Distributed Mutual Exclusion," *ACM Trans. on Computer Systems*, vol. 7, No. 1, pp. 61–77, February 1989.

[2] M. L. Neilsen and M. Mizuno, "A Dag-Based Algorithm for Distributed Mutual Exclusion," *Proc. 11th Int. Conf. Distributed Computer Systems*, pp. 354–360, May 1991.

[3] Y. I. Chang, M. Singhal, and M. T. Liu, "An Improved O(log N) Mutual Exclusion Algorithm for Distributed Systems," *Proc. 1990 Int. Conf. Parallel Processing*, vol. III, pp. 295–302, August 1990.

[4] S. S. Fu and N.-F. Tzeng, "Efficient Token-Based Approach to Mutual Exclusion in Distributed Memory Systems," Tech. Rep. TR-95-8-1, CACS, Univ. Southwestern Louisiana, Lafayette, LA 70504, July 1995.

[5] A. J. Martin, "Distributed Mutual Exclusion on a Ring of Processes," *Science of Computer Programming*, vol. 5, pp. 265–276, February 1985.

[6] T. Johnson, "A Performance Comparison of Fast Distributed Mutual Exclusion Algorithms," *Proc. 9th Int. Parallel Processing Symp.*, pp. 258–264, April 1995.

[7] Z. Xu and K. Hwang, "Modeling Communication Overhead: MPI and MPL Performance on the IBM SP2," *IEEE Parallel and Distributed Technology*, pp. 9–23, Spring 1996.

[8] S. Venugopal *et al.*, "Performance of Distributed Sparse Cholesky Factorization with Pre-scheduling," *Proc. Supercomputing '92*, pp. 52–61, Nov. 1992.

[9] D. Bailey *et al.*, "The NAS Parallel Benchmarks," Tech. Rep. TR RNR-91–002, NASA Ames, August 1991.

[10] P. Keleher, *Lazy Release Consistency for Distributed Shared Memory*, Ph.D thesis, Rice University, Jan. 1995.