Efficient Mutual Exclusion in Peer-to-Peer Systems

Moosa Muhammad, Adeep S. Cheema, and Indranil Gupta

Abstract— **Due to the recent surge in the area of Grid computing, there is an urgency to find efficient ways of protecting consistent and concurrent access to shared resources. Traditional peer-to-peer (p2p) applications such as Kazaa and Gnutella have been primarily used for sharing read-only files (such as mpegs and mp3s). This paper introduces two novel protocols, the End-to-End and Non End-to-End, for achieving mutual exclusion efficiently in dynamic p2p systems. The protocols are layered atop a distributed hash table (DHT), making them scalable and fault-tolerant. The burden of controlling access to the critical section is also evenly distributed among all the nodes in the network, making the protocols more distributed and easily adaptable to growing networks. Since the protocols are designed independent of any specific DHT implementation, they can be incorporated with any generic p2p DHT, depending on the application requirements. We present experiments comparing our implementations with existing mutual exclusion algorithms. The significant reduction in overall message overhead and better load-balancing mechanisms makes the proposed protocols very attractive in being used for current and future p2p and Grid applications.**

Index Terms— **Distributed algorithms, Distributed computing, Resource management, Token networks**

I. Introduction

THE problem of mutual exclusion can be described as a collection of asynchronous processes, each alternately executing a critical and a non-critical section that must be synchronized so that no two processes ever execute their critical sections concurrently. It was first described and solved by Dijkstra in [1].

Even though mutual exclusion is a well-studied problem in distributed systems, it is not possible to directly adapt the proposed solutions into the p2p domain. This dilemma is caused by the differences in the underlying system models, one of which being the absence of a centralized index server to keep track of membership and to ensure consistency. Also, classical distributed algorithms use several rounds of all-to-all communication which is not scalable.

Resources that are being shared can be either computational resources or data, and access to them should be controlled in an efficient and completely decentralized manner. Since each resource can have multiple replicas, the problem in question becomes even more challenging. Access to that resource is controlled by its set of replicas.

The proposed mutual exclusion protocols combine token and quorum-based approaches, to provide efficient and reliable access to shared resources in dynamic p2p systems. The algorithms demonstrate good load-balancing characteristics and low message overhead, when acquiring access to a resource. The use of these quorum sets allow the protocols to scale well with system size since a large number of messages will be intercepted before reaching the replicas.

II. System Model

The protocols presented in this paper are based on the following system model representing a dynamic p2p DHT:

- The basic entities in the system are called nodes (or peers).
- Each virtual resource (e.g., a file or a computational resource) corresponds to a set of nodes (i.e., replicas) that are responsible for granting access to that resource.
- A node can access a resource if and only if each responsible replica grants access to it.
- Nodes are connected over a p2p DHT that allows any node to route a message to any other node.
- The replicas for a resource are always available, but their internal states may be randomly reset due to a crash-recovery failure. They rejoin the network with the same nodeId.
- The number of clients is unpredictable and can be very large. Clients are not malicious.
- There may be high churn in the system – nodes may enter and leave the system anytime.
- Clients and replicas communicate via messages across unreliable channels. Messages can be replicated, lost, but never forged.

Manuscript received June 3, 2005. This work was supported in part by the NSF CAREER grant CNS-0448246 and in part by NSF ITR grant CMS-0427089.

Moosa Muhammad was with the Computer Science Department, University of Illinois at Urbana-Champaign, USA. He is now with Motorola, Inc. (e-mail: mmuhamma@motorola.com).

Adeep S. Cheema was with the Computer Science Department, University of Illinois at Urbana-Champaign, USA. He is now with Microsoft, Inc. (e-mail: adeepc@microsoft.com).

Indranil Gupta is with the Computer Science Department, University of Illinois at Urbana-Champaign, USA (e-mail: indy@cs.uiuc.edu).

III. RELATED WORK

Distributed mutual exclusion protocols tend to fall into two categories as detailed in survey papers [7] and [12]. These include token based protocols [6] and quorum based protocols [8] [5], which intersect at completely centralized exclusion. This study proposes a hybrid between these two sets of protocols, with a competitive level of performance and adherence to general guidelines established for such protocols.

Prior research conducted in efficiently routing messages to the nodes holding a particular resource include, the Chord [11] and the Pastry [9] protocols. Both are examples of structured p2p DHTs and choose their neighbors intelligently to lower the latency and message cost of routing to just O(log n). Sigma [3] is the only currently existing protocol for providing mutual exclusion in dynamic p2p systems. It is implemented inside a p2p DHT and adopts queuing and cooperation between clients and replicas, to enforce a quorum consensus.

IV. PROPOSED PROTOCOLS

The Sigma protocol is a step in the right direction, but it does not fully utilize the decentralized nature of p2p domain. It relies on the replica to maintain the queue of requests for the resource, leading to a less fault tolerant system due to a central point of failure and increased load on a few set of nodes. The two protocols below differ in terms of how well they conform to the End-to-End argument [10].

A. End-to-End Mutual Exclusion Protocol

In order to achieve better load-balancing characteristics, this protocol maintains the queue of requests at the node that is currently in the critical section, instead of at the replica (as was the case in the Sigma protocol). It defines the quorum set for gaining mutually exclusive access to a responsible replica R to be every node in the path from the requesting node to R. Since all the nodes within the quorum set maintain information regarding the current owner of the replica, requests for a resource that is already being held by another node can be satisfied by any of them.

The description of this protocol is as follows:

1. When a node wants mutually exclusive access to a particular resource, it sends requests to all the responsible replicas of that resource, using the underlying DHT routing mechanism.
2. Intermediate nodes lookup their replica list for the intended resource id. If found, the REQUEST message is stopped being forwarded and instead an ENQUEUE message is sent directly to the node currently accessing the resource in context. Otherwise, the REQUEST message continues to be routed.
3. Upon receiving the REQUEST message at the target node, that replica will check if it has voted already or not. A RESPONSE message is routed directly to the original sender of the REQUEST message. The RESPONSE message contains the id and timestamp of the replica's owner (i.e., the node this replica has voted for). If the replica has not voted before, this information would be that of the requesting node.
4. Whenever a requesting node receives a RESPONSE message, it checks to determine if it has received a majority of replica votes. If so, it sends an IAMWINNER message to all the replicas, declaring itself as the new owner of the resource. *Next* and *Previous* node pointers (part of the replica list entry) are updated as this message is routed to all the replicas. If nobody has accumulated enough votes in a round, the requesters send out a YIELD (i.e., RELEASE + REQUEST) message to each of the replicas that voted for it.
5. When an ENQUEUE message reaches its destination (i.e., reaches the owner of the resource that is being requested), the owner adds the requester's id to its request queue.
6. In order to release a resource, a RELEASE message is routed beginning at the current owner of the resource and is targeted for all the responsible replicas for that resource. The intermediate nodes that this message traverses through depend on the *Next* pointers of each node's replica list entry. Clean-up occurs as this message is being routed to the replicas. Once the message reaches its target replica, owner information of that replica is reset.
7. When the owner of the resource is done using that resource, it leaves the critical section and sends a TOKEN message to the node next in line (by removing the head of the queue). The TOKEN message that is sent contains the remainder of the request queue.

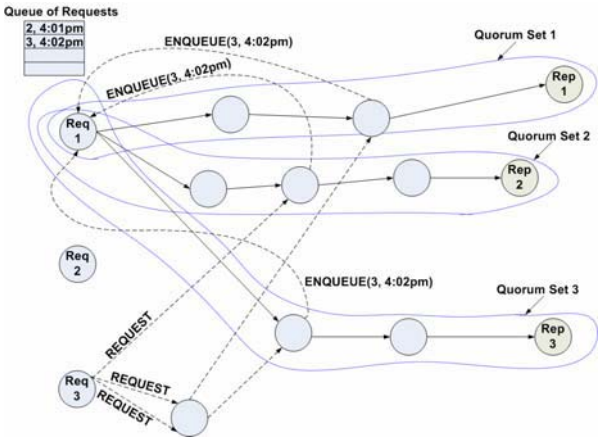

Fig. 1. A snapshot of the E2E Protocol, where Req1 is currently in the critical section and Req2's request to use the same resource is queued up at Req1. When Req3 sends out a request for the same resource, its request is intercepted by quorum set nodes (before reaching the replica), and therefore ENQUEUE messages are sent to Req1 and it adds Req3 to its queue. The three quorum sets that exist in this scenario are marked.

B. Non End-to-End Mutual Exclusion Protocol

The fundamental idea behind this protocol is to maintain a partial queue of requests at all the nodes in the quorum set

rather than a complete queue at only the accessing node. Information maintained in these partial queues can be consolidated when the current node exits the critical section, to determine the next node in line to use this resource.

The message overhead is low, so is the burden of achieving mutual exclusion on the replicas (just like the E2E protocol). Furthermore, this protocol is more distributed and fault-tolerant, since the queue of requests previously being stored at one node is now split among the quorum set.

The description of this protocol is as follows:

1. When a node wants mutually exclusive access to a particular resource, it sends a REQUEST message to all the responsible replicas of that resource.
2. Intermediate nodes lookup their replica list for the intended replica id. If found then the request is queued locally on the node and a GRANTREQ message is sent to the requester, containing the current holder of the replica. Otherwise, their replica list is updated and the REQUEST message continues to be routed.
3. Upon receiving the REQUEST message at the target node (i.e., a replica), the replica will check if it has already granted access to the required resource or not. If it has, the request is queued on the replica itself. In either case, a GRANTREQ message is routed directly to the requester, specifying the holder of the replica (which will be the requester if the replica has not voted before).
4. In order to release a resource, RELEASE messages are routed to all the responsible replicas for that resource. All quorum set members update their replica list by deleting the respective entry from it and forwarding the message together with a list of requests (sorted by timestamp), seen so far by any of the nodes prior to them in the sequence from the holder to the replica. When this message reaches its target replica, it contains every request seen so far for that replica, and is merged with the local queue. The next owner of this replica is then determined from this newly assembled queue and GRANTREQ message is sent directly to that node.
5. When all GRANTREQ messages reach their destination, the requester checks if it has attained majority of the replica votes, to enter the critical section. If no requester receives majority of the votes, all of them propagate a YIELD message to the replicas. Otherwise only one of the requesters will receive the majority votes and can therefore access the resource.
6. The YIELD operation reflects the collaborative nature of this protocol and is used to reshuffle the queue. The fact that nobody wins indicates that contention has occurred. The YIELD message allows all requesters to try to acquire the resource again after a random time period. Typically, this self-stabilization process will quickly settle, as verified in [4].

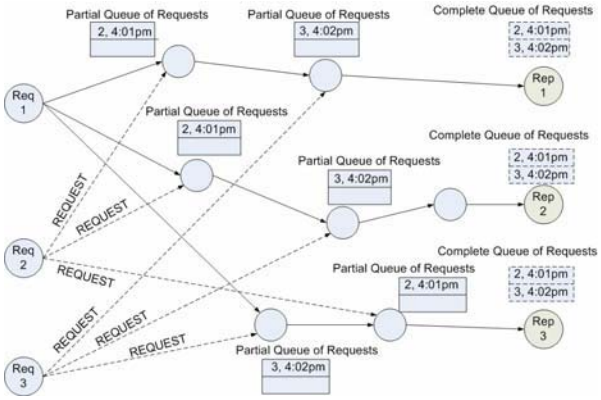

Fig. 2. A snapshot of the Non E2E Protocol, where Req1 is currently in the critical section. Req2 and Req3 have issued REQUEST messages to also try to gain access to the same resource. Their requests are queued by the intermediate nodes of the respective quorum sets. When Req1 has finished using the resource and leaves the critical section, the partial queues will be merged (within each quorum set) to form the complete queue of requests at the replicas.

V. HANDLING FAILURES

The protocols described above work well under failure-free environments. The types of failures that can occur include: (1) Failure of node currently present in the critical section (2) Failure of node(s) maintaining the queue of requests (3) Failure of intermediate nodes of the quorum set. For a detailed analysis of handling failures, refer to [13].

VI. EXPERIMENTAL RESULTS

The E2E, Non E2E, and Sigma mutual exclusion protocols have been implemented over the FreePastry [2] implementation of the Pastry routing substrate. The test-bench code starts off by creating a set of nodes, arranged according to the Pastry network topology. The simulation then runs for a certain number of rounds and derives the experimental results, by averaging over ten such runs. In the simulation, the concept of a round is introduced in order to describe the amount of work done by the nodes in the network, which includes a certain number of requests for resources being issued by a set of randomly chosen nodes. Also in each round, a node releases one of its already held resources, with a 50% probability. This means that the average holding time of a critical section is 2 rounds.

A. Scalability

One of our initial goals behind the proposed protocols was to come up with an efficient and scalable method for achieving mutual exclusion. Figure 3 presents a comparison of the three protocols. Even though all the plots are linear with respect to number of nodes, the two proposed protocols require 1.5 to 2.5 times less messages than the Sigma protocol, which is a big improvement. The number of resources, rounds, and requests per round are held constant at 65, 20, and 20 respectively.

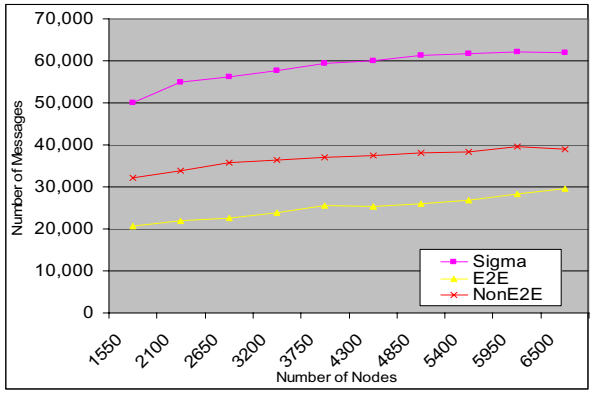

Fig. 3. A lower increase in bandwidth consumption, as the network grows

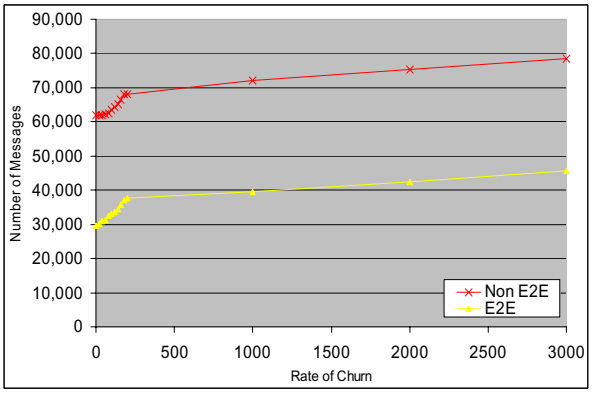

Fig. 5. Effect of rate of churn on message overhead

B. *Overall Load Distribution*

The Cumulative Distribution Function plots below demonstrate the even load distribution of the E2E protocol. Similar characteristics were observed for the Non E2E protocol [13]. They differ in the number of replicas that are responsible for granting access to a particular resource. The term load refers to the number of messages that a node has to process.

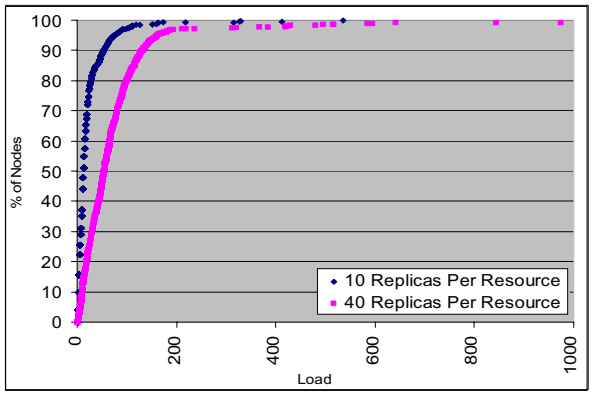

Fig. 4. By increasing the number of responsible replicas for a resource, the load on each node increases. This behavior is expected since the number of request/response messages exchanged between the replicas and the requesters would now be greater

C. *High Churn Rate*

Rate of churn is defined as the sum of the number of nodes that are in transition (i.e., are either leaving or joining the system per round). In our simulation, n/2 randomly chosen nodes are failed and n/2 new nodes are added to the system, where n is the desired churn rate. Requesters are not churned in our experiments.

Figure 5 shows the effect of increasing the rate of churn from 0 to 3000, on the overall message overhead. Both E2E and Non E2E protocols experience a slightly higher increase in message overhead in the beginning, but then increase at a much lower rate.

VII. CONCLUSION

In this work, a couple of protocols for achieving mutual exclusion in a p2p system were presented, one of which conformed more to the End-to-End argument. Both protocols can be easily configured to run on any DHT, therefore providing a truly generalized solution to the addressed problem. The purpose behind presenting two different approaches to the same problem was to present the inherent tradeoff that exists between the amount of bandwidth consumed and better load balancing and fault-tolerance level required. The proposed protocols, through their truly novel quorum and token-based schemes, were able to keep the load on the replicas relatively low even in the presence of a growing network and high churn rates.

REFERENCES

[1] E. W. Dijkstra, Solution of a Problem in Concurrent Programming Control, Communications ACM 8, 9 (Sept. 1965), 569.
[2] FreePastry. http://freepastry.rice.edu/FreePastry.
[3] S. Lin, S., Q. Lian, M. Chen, and Z. Zhang, A Practical Distributed Mutual Exclusion Protocol in Dynamic Peer-to-Peer Systems, International Workshop on P2P Systems (IPTPS), 2004.
[4] S. Lin, S., Q. Lian, M. Chen, and Z. Zhang, A Practical Distributed Mutual Exclusion Protocol in Dynamic P2P Systems, Microsoft Research, Technical Report MSR-TR-2004-13.
[5] M. Maekawa, A √n Algorithm for Mutual Exclusion in Decentralized Systems, ACM Transactions on Computer Systems (TOCS). 1985.
[6] K. Raymond, A Tree-Based Algorithm for Distributed Mutual Exclusion, TOCS. 1989.
[7] M. Raynal, A Simple Taxonomy for Distributed Mutual Exclusion Algorithms, ACM SIGOPS Operating Systems Review. 1991.
[8] G. Ricart, and A. K. Agrawala, An Optimal Algorithm for Mutual Exclusion in Computer Networks, Communications of the ACM. 1981.
[9] A. Rowstron, and P. Druschel, Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems, Middleware, 2001.
[10] J. H. Saltzer, D. P. Reed, and D. D. Clark, End-to-End Arguments in System Design, TOCS 2, 4, November, 1984, 277-288.
[11] I. Stoica, R. Morris, D. Karger, M. F.rans Kaashoek, and H. Balakrishnan, Chord: A Scalable Peer-to-Peer Lookup Service For Internet Applications, SIGCOMM, 2001.
[12] M. G. Velazquez, A Survey of Distributed Mutual Exclusion Algorithms, Technical Report CS-93-116, Colorado State University, 1993.
[13] M. Muhammad, A. Cheema, and I. Gupta. Efficient Mutual Exclusion in Peer-to Peer Systems, Computer Science Report No. UIUCDCS-R-2005-2622 (Engr. No. UILU-ENG-2005-1813), University of Illinois at Urbana-Champaign, August 2005.