Quorum Based Distributed Mutual System

M.V.Ramana Murthy[*], Pradosh Patnaik[**], M.V.Vijaya Saradhi[***], V.Venkateswarlu[*]
Md. Ismail**, Syed Salahuddin*****
* Dept.of Mathematics & Computer Science, University College of Science, Osmania University, Hyderabad, India.

** Dept.of Computer Science, Aurora Pg College, Ramanthapur, Hyderabad, India.
*** Dept. of Computer Science & Engineering, Astra, Bandlaguda, Hyderabad, India.

Abstract
The main goal of a distributed computing system is to connect users and resources in a transparent, open, and scalable way. Ideally this arrangement is drastically more fault tolerant and more powerful than many combinations of stand-alone computer systems. Openness is the property of distributed systems such that each subsystem is continually open to interaction with other systems. An open system that scales has an advantage over a perfectly closed and self-contained system. Distributed programming typically falls into one of several basic architectures or categories: Client-server, 3-tier architecture, N-tier architecture, Distributed objects, loose coupling, or tight coupling. In Client-server architecture Smart client code contacts the server for data, then formats and displays it to the user. Input at the client is committed back to the server when it represents a permanent change. Three tier systems move the client intelligence to a middle tier so that stateless clients can be used. This simplifies application deployment. Most web applications are 3-Tier. N-Tier refers typically to web applications, which further forward their requests to other enterprise services. This type of application is the one most responsible for the success of application servers.

Key words:
DCE, Maekawa's Algorithm, Mutual exclusion, Lamport algorithm and the Ricart-Agrawal algorithm.

Introduction

A **Distributed System** [1],[2] consists of a collection of autonomous computers, connected through a network and distribution middleware, which enables computers to coordinate their activities and to share the resources of the system, so that users perceive the system as a single, integrated computing facility.

The **Distributed Computing Environment (DCE)** is a software system developed in the early 1990s by a consortium that included Apollo Computer, Helwet-Packard, IBM, Digital Equipment Corporation, and others. The DCE supplies a framework and toolkit for developing Client/Server applications. The framework includes a Remote Procedure Call (RPC) mechanism known asDCE/RPC, a naming (directory) service, a time service, an Authentication service, an authorization service and a Distributed File System (DFS) known as DCE/DFS Distributed computing never really caught on as much as had been hoped for in the late 1980s and early 1990s. The rise of the Internet, Java and web services stole much of its mindshare through the mid-to-late 1990s, and competing systems such as CORBA muddied the waters as well. Perhaps ironically, one of the major uses of DCE/RPC today are Microsoft's DCOM and ODBC systems, which use DCE/RPC (in MSRPC) as their network transport layer.

The main goal of a distributed computing system is to connect users and resources in a transparent, open, and scalable way. Ideally this arrangement is drastically more fault tolerant and more powerful than many combinations of stand-alone computer systems. Openness is the property of distributed systems such that each subsystem is continually open to interaction with other systems. An open system that scales has an advantage over a perfectly closed and self-contained system.

Distributed programming typically falls into one of several basic architectures or categories: Client-server, 3-tier architecture, N-tier architecture, Distributed objects, loose coupling, or tight coupling.

In Client-server architecture Smart client code contacts the server for data, then formats and displays it to the user. Input at the client is committed back to the server when it represents a permanent change.

Three tier systems move the client intelligence to a middle tier so that stateless clients can be used. This simplifies application deployment. Most web applications are 3-Tier.

N-Tier refers typically to web applications, which further forward their requests to other enterprise services. This type of application is the one most responsible for the success of application servers.

In Tightly coupled (clustered) architecture a set of highly integrated machines that run the same process in parallel, subdividing the task in parts that are made individually by each one, and then put back together to make the final result.

In Peer-to-peer architecture there is no special machine or machines that provide a service or manage the network resources. Instead all responsibilities are uniformly divided among all machines, known as peers. Peers can serve both as clients and servers **Mutual exclusion** (often

Manuscript received December 5, 2009
Manuscript revised December 20, 2009

abbreviated to mutex) algorithms are used in concurrent programming to avoid the simultaneous use of a common resource, such as a global variable. In concurrent programming a critical section is a piece of code that accesses a shared resource (data structure or device) that must not be concurrently accessed by more than one thread of execution.
A **critical section** will usually terminate in fixed time, and a thread, task or process will only have to wait a fixed time to enter it (i.e. bounded waiting). Some synchronization mechanism is required at the entry and exit of the critical section to ensure exclusive use, for example a semaphore.

Requirment of the Algorithm:

Maekawa's Algorithm is an algorithm for mutual exclusion on a distributed system. The basis of this algorithm is a quorum like approach where any one site needs only to seek permissions from a subset of other sites.
A site is any computing device which is running the Maekawa's Algorithm For any one request of the critical section, the requesting site is the site which is requesting entry into the critical section, The receiving site is every other site which is receiving the request from the requesting site. Timestamp refers to the local timestamp of the system according to its logical clock.
When a processor wishes to enter the critical section, it sends a vote request to every member of its voting district. When the processor receives replies from all the members of the district, it can enter the critical section. When a processor receives a vote request, it responds with a "YES" vote if it has not already cast its vote. When a processor exits the critical section, it informs the voting district, which can then vote for other candidates.
Deadlock can be avoided by using the following mechanism: when a processor makes a request, it assigns a (Lamport) timestamp to the request. The voters will prefer to vote for the earliest candidates. If a processor V has cast its ballot for processor B and then processor C, which has an earlier timestamp than B's, asks for V's vote, V will try to retrieve its vote from B with an INQUIRE message. If B has not yet received all the votes of its voting district, it will relinguish V's vote, which can then be given to C. Lamport timestamps impose a total order, so either the candidate with the lowest timestamp eventually gets all of the votes or the candidate with the lowest timestamp is blocked by a candidate that enters the critical section. In either case, some candidate enters the critical section, so deadlock is avoided.
There is one point at which we need to match up the messages with protocol invocations. Because the INQUIRE messages are generated by the voters asynchronously, a candidate might receive an INQUIRE message that was generated in response to a previous critical section request. The usual method for preventing ambiguous messages is to use sequence numbers. Since the candidate samples a timestamp for the algorithm, we use the timestamp as a sequence number to match INQUIRE messages with critical section request.
The performance of a mutual exclusion algorithm [3],[4] is measured by the number of messages exchanged per critical section execution and the delay between successive executions of the critical section. There is a message complexity and synchronization delay trade-off in mutual exclusion algorithms. The Lamport algorithm and the Ricart-Agrawal algorithm both have a synchronization delay of T (T is the average message delay), but their message complexity is O(N).
Maekawa's algorithm reduces the message complexity to O(sqrt{N})[5],[6],[7], however it increases the synchronization delay to 2T.

Implementation of Algorithm:

The first variant sends a message to the output channel; if the receiver node linked to the channel is blocked, this node is released.
The second variant sends message to all output channels in a set.
In both variants, the sender node is not blocked even if the receiver node linked to the corresponding channel is not ready to receive m. Consequently, output channel may hold an arbitrary number of messages.
receive: Receiving a Message.
The first variant returns a message from output channel; if output channel is empty, the current node is blocked until output channel becomes non-empty.
The second variant behaves like the first one except for the fact that the current node is blocked for at most time units. If then output channel is still empty, null is returned.
Which message is returned from c depends on the selector associated to output channel; by default, this is the oldest message held by output channel.
The first variant returns the index of a non-empty channel in the (non-empty) set; if all channels are empty, the current node is blocked until a channel becomes non-empty. This index may be used to reference the corresponding channel in the set.
The second variant behaves like the first one except for the fact that the current node is blocked for at most time units. If then all channels are still empty, -1 is returned.

Conclusions:

Maekawa's algorithm reduces the message complexity to O(sqrt{N}); however, it increases the synchronization

delay to 2T.. And this Algorithm is efficient and less Time complexity compare to other distributed mutual exclusion algorithms.

The Lamport algorithm and the Ricart-Agrawal algorithm both have a synchronization delay of T (T is the average message delay), but their message complexity is O(N). The token-based Suzuki-Kasamis Broad Cast Algorithm has message complexity as N. The other token based Raymonds Tree-based Algorithm has the average case complexity as 2LogN.

Finally Maekawa's Algorithm is easy to implement and comparatively less complexity with respective to other algorithms.

References

[1] "Distributed Systems Concepts and Design", G COULOURIS, J DOLLIMORE, T KINDBERG

[2] "Distributed Operating Systems" ANDREW S. TANENBAUM, , Prentice Hall, 1995

[3] A Toolkit for the Simulation of Distributed Algorithms in Java Technical Report 97-36, Research Institute for Symbolic Computation (RISC-Linz), Johannes Kepler University, Linz, Austria, November 1997.

[4] A Fair Distributed Mutual Exclusion Algorithm, IEEE TRANSACTIONS ON PARALLELAND DISTRIBUTED SYSTEMS, VOL. 11, NO. 6, JUNE 2000

[5] A SQRT(N) Algorithm for Mutual Exclusion in Decentralized Systems ACM Transactions on Computer Systems, Vol. 3, No. 2, May 1985, Pages 145-159.

[6] Performance of the Network Intrusion Detection Systems M.V.Ramana Murthy et.al, Int. Jr.Computer Science & NetworkSecurities,Oct, 2009, Vol. 9 No. 10, pp. 198-202

[7] Journal of Universal Computer Science, vol. 12, no. 2 (2006), 140-159 An O(SQRT(N) Distributed Mutual Exclusion Algorithm Using Queue Migration1.

Mangipudi Venkata Ramana Murthy is currently Professor in Faculty of Mathematics and Computer Science in University College of Science, Osmania University. He has received his Ph.D in Computational Fluid Mechanics in 1986 from Osmania University. He is actively involved in research and successfully supervised 22 students for their Doctorial work in the areas of Computer Science and applied Mathematics. The research areas include Artificial Neural Net works, Net work securities , Digital Image processing of Computer Science besides this he also contributed to Fluid Mechanics of Applied Mathematics. He has several research publications to his credit which has international repute such as IEEE, ATTI DELLA FOUNDZIONE, ASME , JFMR, IJHMT

Pradosh Chandra Pattnaik is currently working as Associate Professor in the Department of Computer Science at Aurora's PG College, Hyderabad,India ,where he teaches several Courses in the area of Computer Science.He is Currently Pursuing the PhD degree in Computer Science at Dravidian University,Kuppam, India.His main research interests are Distributed Systems,Design Patterns,Object Oriented Design Analysis and Software Engineering

M.Vijaya Saradhi is currently working as Associate Professor in the Department of Computer Science and Engineering at Aurora's Scientific,Technological and Research Academy, Hyderabad,India, where he teaches Several Courses in the area of Computer Science.He is Currently Pursuing the PhD degree in Computer Science at Osmania University,Hyderabad, India.His main research interests are Software Metrics,Distributed Systems,Design Patterns,Object Oriented Design Measurements and Empirical Software Engineering

Dr V.Venkateswarlu has obtained Ph.D from the Osmania University in the area of digital image processing in the year 2001. My research interest includes Artificial Intelligence and expert systems, network securities besides digital image processing. My prime research includes security applications in distributed computing. Presently I am as a Sr. database administrator.

Md. Ismail is currently working as Associate Professor in the Department of Computer Science at Aurora's PG College, Hyderabad,India ,where he teaches several Courses in the area of Computer Science.He is Currently Pursuing the PhD degree in Computer Science at Dravidian University,Kuppam, India.His main research interests are Computer Architectutre and Design, Computer Networks, Distributed Systems and Network Security.

Syed Salahuddin is Currently working as Lecturer in Mathematics College of Science, Al-Jouf University, Jouf, Saudi Arabia, He has received his Ph.D in Linear and non-Linear paramateric programming problems with bounded variables from Osmania University, Hyderabad, India.