ISSN: 2277-9655
Impact Factor: 1.852

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Algorithm for Ordering of Message in Mobile Computing Environment

Mr. Jayanta Loha
Assistant Professor, Department of Computer Science and Application, St. Joseph's College, Darjeeling, India
jayanta.sjc@gmail.com

Abstract

Causal message ordering is required for several distributed applications. In order to preserve causal ordering, only direct dependency information between messages, with respect to the destination process (es), need be sent with each message. By eliminating other kinds of control information from the messages, the communication overheads can be significantly reduced. This paper presents an algorithm that uses this knowledge to efficiently enforce causal ordering of messages. The proposed algorithm does not require any prior knowledge of the network communication pattern .The algorithm can be employed in a variety o f distributed computing environments. Its energy efficiency and low bandwidth requirement make it especially suitable for mobile computing systems.

Keywords: Causal message ordering, MSS, MH, Done relation, Hand-off.

Introduction

A mobile computing system is a distributed system consisting of a number of mobile and fixed processing units. The fixed units referred to as a Mobile Support Stations (MSSs), can communicate with each other through a fixed wire line network [8]. The geographical area of the mobile computing system is divided into regions called cells with an MSS in each cell. The mobile hosts (MHs) in a cell can open a wireless communication channel with the MSS in the cell and communicate with all the other processors in the system through this MSS. Alternatively an MH can be connected through an access point of the fixed wire line network [9] for communication purposes. An MH can move out of one cell and into another cell. In such a case the MSSs of the old cell has to hand over the responsibilities for the MHs communication to the MSS of the new cell. This process is referred to as hand-off [1, 5].

A distributed application executing in a mobile computing environment consist of a collection of processors such that one or more processors may be running on each processor. The processors communicate with each other through asynchronous message passing with message propagation time being finite but arbitrary. The execution of a process consists of three types of events: message send, message delivery and internal events. Internal events represent local computations at the processes. In the absence of a global clock, determining the relative order of occurrence of events on diff-erent processes is non-trivial.

Controlling the executions of a distributed application such that all events are totally ordered is expensive. A less severe form of ordering of message transmission and reception, called causal ordering, is sufficient for a variety of applications like management of replicated data, observation of a distributed system, resource allocation, multimedia systems and teleconferencing [10, 3, 6].

Hence, there is a need for an implementation of causal ordering of messages that has low communication, computation and memory overheads [2, 4]. This paper presents a low overhead algorithm for causal ordering of messages. While the proposed algorithm is suitable for a variety of distributed systems, its efficiency makes it especially attractive for mobile computing systems.

System Model

The application under consideration is composed of N processors. These processes collective execute a distributed computation. There exists a logical communication channel between each pair of processes. A process can send a message to either one process or a group of processes. The group of processes to which a process sends multicast messages need not be fixed, i.e. a process P_i may send one multicast message to a group of processes G1 and later another multicast message to a different process G2. Thus, dynamic multicast groups are

ISSN: 2277-9655
Impact Factor: 1.852

allowed where in a process can do a multicast to any group of processes without having form groups a priori.

However, two events occur at two different processes cannot be determined solely on the basis of the local time of occurrence. However information about the order of occurrence of events can be gathered based on the causal dependencies between them. Such dependencies can be expressed using the *done* relation (→) between events. The *done* relation between events has been defined in as:

- $a \rightarrow b$, if a and b are events in the same process and a occurred before b.
- $a \rightarrow b$, if a is the event of sending a message M in a process and b is the event of delivery of the same message to another process.
- If $a \rightarrow b$ and $b \rightarrow c$, then $a \rightarrow c$ (i.e."→"relation is transitive).

If $a \not\rightarrow b$ and $b \not\rightarrow a$ then a and b are said to be concurrent and represented as $a \,||\, b$.

Events a, b and c mentioned above can be message SEND, message DELIVER or internal events of the processes. Causal ordering of messages specifies the relative order in which two messages can be delivered to application process.

Basic Idea of Algorithm

(M′)→SEND(M″) and SEND(M″)→SEND(M) [7]. Thus, when M reaches P_i it can be delivered to P_i only after M′ has been delivered to P_i. Until then M is buffered at P_i. If causal ordering of messages is enforced between every pair of immediate causal predecessor and successor messages, then causal ordering among all messages will be automatically ensured due to the transitivity of the *done* relation.

Now the basic idea of underlying the algorithm is shown in Figure below.

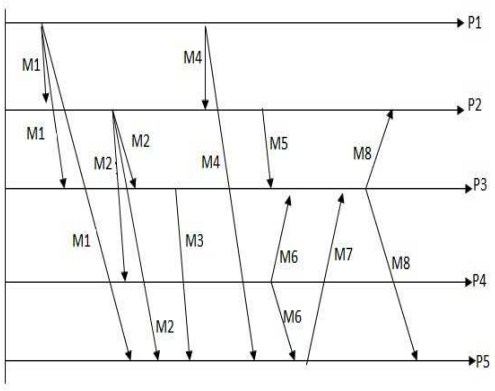

Fig. 1 Illustrate the basic idea of the algorithm

Therefore two kinds of information need to be sent with each message: (i) direct predecessor messages with respect to each destination process and (ii) sender's knowledge of the most recent mutually concurrent message from the other processes delivered to the sender.

Algorithm for Causal Ordering

Causal ordering of messages is implemented by the underlying system by executing the following protocol at the time of send and reception of a message M at P_i.

Message Send: P_i sends M to *destination* (M)

1. for all $P_j \in$ *destination* (M): SEND the unique identifier for message M of P_i's knowledge of the direct processor with respect to message sent to P_k.
2. For all $P_j \in$ *destination* (M): Future message sent to P_j should deliver after M.

Message Reception: P_i receives message from Pj

1. Wait for messages sent to Pi constraining M must be delivered prior to M′s delivery.
2. Delivery of M to Pi.
3. Delivery of message M from j to i.

To ensure causal ordering, a message M needs to carry i

Comparison with Related Work

The communication overhead of previous algorithms to implement causal ordering is high, because at least an N*X*N integer matrix [6] or N – 1 vector clock (each with N integer components) [11] are sent with every massage.

These matrices and vector sets contain information about the Direct as well as transitive causal predecessors of a message. In the proposed algorithm, a message carries information only about its direct predecessors with respect to each destination process. Hence, the communication overheads are low. However, such high overheads are incurred only when there is a high degree of concurrency in the message communication pattern, there exist concurrent messages sent by each process in the system that are direct causal predecessors of M with respect to each of its destinations. Usually, the degree of concurrency is much lower and the overheads are smaller.

The computation overhead at processes, for maintaining causal ordering are low, because processes have to perform only a small number of simple operation like integer comparisons and set operations at the time of sending and delivering messages. Moreover , if a process P_i multicasts a message M to a set of processes *destination*(M), all the computation overheads for

ISSN: 2277-9655
Impact Factor: 1.852

message send are incurred only once for the entire multicast, regardless of the cardinality of *destination* (M).

The proposed algorithm has lower communication overheads than existing algorithms. However there are situation in the generalized multicasting case where dependence information that is over and above the minimum required to maintain causal ordering is sent with message. Thus, if a message delivery is more than one message hop away from P_i. Delivered, does not have information about it.

Casually ordered multicast message may be sent by an MH or an MSS. If an MH needs to multicast a message, the responsibility is handled by the MSS of the cell in which the MH lies at the time of sending the message. This is a conservation of the limited energy supply of the MH.

The mobility of nodes can lead to complications in causal multicasting of messages. An MH that is a destination of a message and moves from one cell to another may not have the message delivered to it even through the MSSs in its previous and current cell receive the message. Also if a destination MH moves into a cell whose MSS does not receive a copy of the message, the MH will not have the message delivered to it. Therefore sending the multicast message only to the MSSs of the cells in which the target MHs are currently present may not work. So the message should be sent to all the MSSs.

Conclusion

Causal ordering of messages is required in a variety of distributed application. Previous algorithms to implement causal ordering have high communication overheads. This is because each computation message carries information about direct as well as transitive dependencies between messages. Here causal ordering needs to carry information for each message only about (i) its direct predecessor messages with respect to each destination process and (ii) most recent mutually concurrent messages delivered to its sender. It has low communication and computation overheads. It can handle dynamically changing multicast communication groups.

The proposed algorithm also adapts to the volume of traffic in the system. Simulation experiments showed that the higher the number of messages sent concurrently in the immediate past of a message, the more control information the message has to carry. On the other hand, if concurrency is low, the control information carried by the message is smaller. Generally, there is a positive correlation between traffic intensity and the level of concurrency: high traffic corresponds to greater concurrency and low traffic corresponds to low concurrency.

The efficiency of the algorithm makes it especially suitable for mobile computing environments. The low memory and communication overheads of the implementation satisfy the low energy consumption and low available bandwidth constraints of mobile computing systems. The low algorithm can be used to implement casually ordered multicasting of messages in mobile computing environments such that exactly one copy of a message is delivered to every destination node.

References

[1] A. Acharya and B. R. Badrinath. Delivering Multicast Messages in Network with Mobile Hosts. In Proceeding of the 13th International N Conference on Distributed Computing Systems, pages 292-299, IEEE, 1993.

[2] S. Alagar and S. Venkatesan. Causally Ordered Message delivery in Mobile Systems. In Proceeding of the Workshop on Mobile Computing Systems and Applications, pages 169-174, Santa Cruz, December 1994.

[3] R. Baldoni. A. Mostefaoui and M. Raynal. Efficient Causally Ordered Communications for Multimedia Real-Time Applications. In Proceedings of the 4th International Symposium on High Performance Distributed Computing, pages 140-147, Washington, D. C., August 1995.

[4] K. Birman. A. Schiper and P. Stephenson. Light weight Causal and Atomic Broadcast. ACM Transactions on Computer systems, 9(3):272- 314, 1991.

[5] Ravi Prakash, Michel Raynal and Mukesh Singhal. An Adaptive Causal Ordering Algorithm Suited to Mobile Computing Environments.

[6] M.Raynal, A.Schiper and S.Toueg. The causal ordering abstraction and a simple way to implement it. Information Processing Letters, 39(6): 343-350,1991.

[7] A.Mostefaoui and M.Raynal. Causal Multi casts in Overlapping Groups:Towards a 4th Low Cost Approach. In Proceedings of the IEEE International Conference on Future Trendsin Distributed Computing Systems, pages136-142, Lisbon, September1993.

[8] T.Imielinski and B.R.Badrinath,Mobile Wireless Computing. Communication of the ACM,37(10):19-28,1994. M.Callendar.International Standards for Personal Communications.In proceeding of the 39th IEEE vehicularTechnology Conference,pages 722-728,1989.

[9] F. Adelstein and M. Singhal. Real-Time Causal Message Ordering in Multimedia Systems.I Proceedings of the 15th International Conference on

ISSN: 2277-9655
Impact Factor: 1.852

Distributed Computing Systems, pages 36-43, June 1995.

[10] A. Schiper, J. Eggli, and A. Sandoz. A New Algorithm to Implement Causal Ordering.In Proceedings of the 3rd International Workshop on Distributed Algorithms, LNCS-392,pages 219-232, Berlin, 1989. **Springer**.