Study and Analysis of Chandy Misra Haas Distributed deadlock Detection

Purva Gawde

Department of Computer Science

Kent state University

pgawde@kent.edu

Abstract—in this paper, we are going to study Chandy Misra Haas distributed deadlock detection. The paper is based on the practical implementation of the algorithm and the comparison made based on the experimental results. These algorithms have been implemented on C++ format. The message complexity and time complexity have been used to measure the performance of the algorithm. In section II of the paper we are going to study the analysis of the algorithm when only one initiator is present. In section III of the paper we are going to study the analysis of the algorithm when each process acts as an initiator. Section 4 talks about the conclusion of this experiment in brief. To conclude the paper we present the result of the comparison and suggest some improvements.

I. Introduction

In the introduction we will talk about the basic concepts of the algorithm which will enable better understanding of algorithm and implementation.

Communication Model: Network of processes which communicate via messages. Controllers are implemented by processes and requests for resource allocation and cancellation and release must be implemented by processes.

A process can be either idle or active. Processes can wait for more than one process for resource allocation and process becomes active if it receives a message from any one of the process it's waiting for.

Properties of a query computation:

If a process is deadlocked when it initiates a query computation, it will receive a reply.

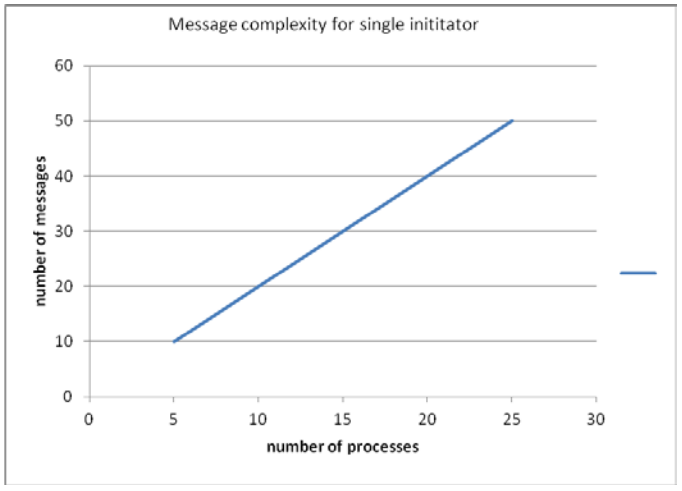
Several processes may initiate a query computation and same process may initiate a query computation several times.

Properties for processes:
Every process maintains four local variables:
Latest: largest sequence number in any query
Engager: it is the identity of the process which caused latest to be set to its current value.
Num: total number of query minus reply messages
When value of num for the initiator becomes zero then the deadlock is detected.
Wait: is true only when the process is idle.
Initial value of latest for each of the process is set to be zero and initial value for wait for all processes is false.

Various data points are inserted in Implementation to calculate the number of messages.

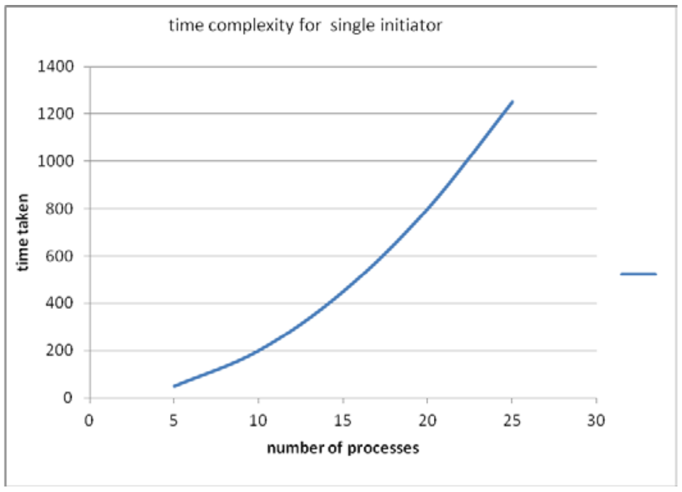
II. Single initiator

Analysis of the Chandy Misra Distributed deadlock detection when only single process in the deadlocked processes initiates the query:
The data for measuring the message and time complexity for the algorithm is gathered over 10 runs of the implementation.



Graph 2.1
Processes vs. messages

Message complexity:
The graph represents the number of processes vs. number of messages. Number of messages is counted till the deadlock which is initiated by a single process has been detected.
The graph shows as the number of processes increase the number of messages exchanged to detect a deadlock is increasing linearly. This happens because the same number of messages is exchanged each time query is initiated.

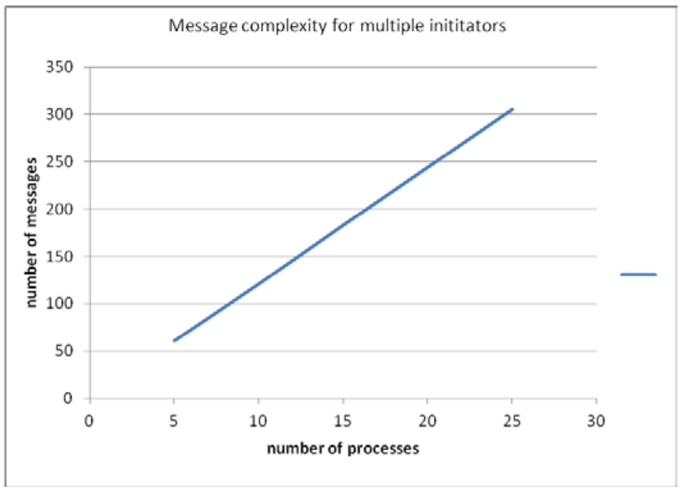


Graph 2.2
Processes vs. time

Time complexity:
The graph represents the Time taken to detect the deadlock when single initiator initializes a query.
Time taken is calculated in terms of number of steps taken to detect a deadlock.
The graph shows that as the number of processes increase, time taken to detect a deadlock is increasing linearly. This happens because equal number of steps is taken each time a process initiates the query.

III. All initiators

All the processes involved in the deadlock initiate the query. Processes are waiting for the next process for resources in a circular manner. Each one of these processes initiates the query. A process can initiate the query if deadlock for the previous process is already detected.
These are the assumptions made while calculating the message and time complexity.

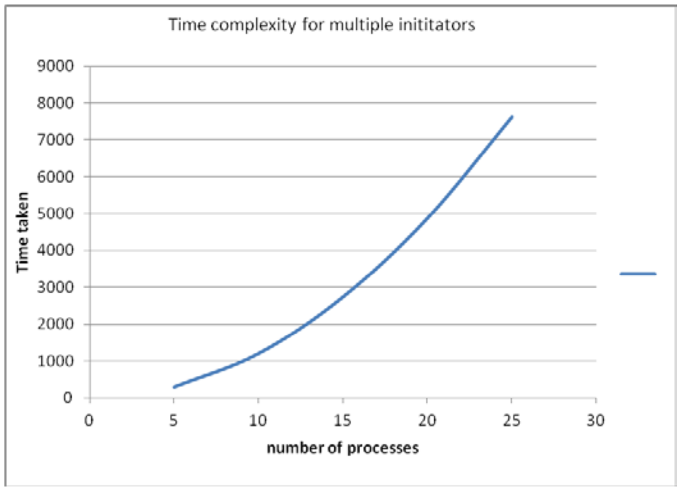


Graph3.1
Processes vs. messages

Message complexity:
This graph represents the number of messages exchanged till the deadlock is detected when each one of these processes initiate the query. We can see from the graph that number of messages exchanged increase significantly as number of processes increase. This happens because same set of messages are repeated for each of these processes when it initiates the deadlock by sending a query message.

Time complexity:

The graph represents the time required to detect the deadlock vs. the number of messages. Here all the processes act as an initiator. Hence time required to detect the deadlock for each one of these processes is measured in terms of number of steps. It is evident from the graph that time taken to detect all the deadlocks increases significantly as the number of messages. Same number of steps is repeating for each of the process initiating the deadlock.



Graph 3.4
Processes vs. time

IV. CONCLUSION

As seen in the performance graph of the algorithm
For both the scenarios it can be concluded that:
As the number of processes increase the number of messages increase to detect a deadlock.
But if the number of initiators increases, the number of messages exchanged and time taken to detect the deadlock of the algorithm increase significantly.

References:

1.Distributed Deadlock Detection-K Mani Chandy and Jaydev Mishra.

2.CHANDY, K.M., AND MISRA, J. A distributed algorithm for detecting resource deadlocks in distributed systems. In *Proc. A CM SIGA CT-SIGOPS Syrup. Principles of Distributed Computing* (Ottawa, Canada, August 18-20, 1982), ACM, New York, 1982, pp. 157-164.

3. CHANDY, K.M., AND MISRA, J. A distributed algorithm for detecting resource deadlocks in distributed systems. In *Proc A CM SIGA CT- SIGOPS Syrup. Principles of Distributed Computing* (Ottawa, Canada, August 18-20, 1982), ACM, New York, 1982, pp. 157-164