# Study and Analysis of Snapshot algorithms (Chandy Mishra snapshot algorithm, Ho-Ramammorthy's 2 phase deadlock detection algorithm)

Amit Mali
Department Of Computer Science
Kent State University, Kent
amali@kent.edu

Abstract: In this paper we are going to study two snapshot algorithms, namely Chandy-Mishra Snapshot algorithm and Ho-Ramammorthy 2-phase deadlock detection algorithm. The paper is based on the practical implementation of the algorithms and comparison is made according the experiment results. These algorithms are implemented on java platform. The message complexity and time complexity are used to measure and compare the performance of the algorithms. In section III of the paper we will analyze the Chandy-Mishra snapshot algorithm. Section IV presents analysis of the Ho-Ramammorthy 2-phase deadlock detection algorithm. Section V talks about the implementation details in brief.In section VI the comparison of the performance of two algorithms is discussed. To conclude the paper we present the result of the comparison and suggest improvements over the implementation on a large scale distributed systems.

I. Introduction: In the introduction we will talk about some basic definitions related to the algorithms which will enable better understanding of the algorithms and the implementation.

Snapshot: Snapshot of an algorithm returns the state of the system and message queue. It basically returns a status message giving details about the current state of individual nodes in the system graph.

2-Phase snapshot: This term is used on Ho-Ramammorthy's 2 phase deadlock detection algorithm. The first phase is essentially same as a snapshot, in the second phase it reiterates through the process tables to ensure no false deadlocks are detected.

Snapshot and deadlock detection algorithms are not the core algorithms in regard to the level on which they operate. These algorithms operate on top of the basic algorithm adopted by the system. In this particular paper and the implementation of the algorithms the basic algorithm for the system is a flooding algorithm. The rules followed in the flooding algorithm can be listed as follows.

1. Each process forwards a message at least once.
2. Each process receives at least one

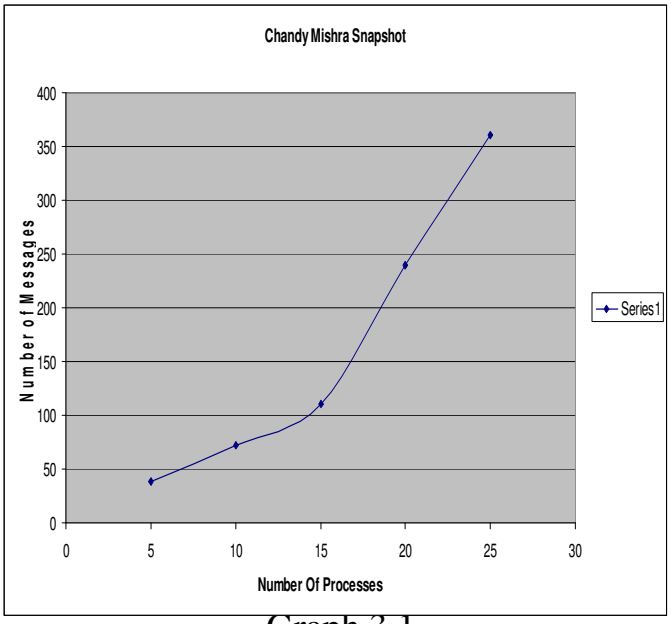
message.
3. When above two conditions are satisfied algorithm terminates.

Various data points are inserted in the implementation of the algorithms to keep track on the number of messages and the time taken till that particular execution point. This data is collected over several runs and performance comparisons are presented based on the readings.

III. Analysis of Chandy-Mishra snapshot algorithm:
The data for measuring the message and time complexity was gathered over 30 runs of the implementation. The average of all the collected values is used to plot the comparison graphs. The data is shown in Table 3.1.

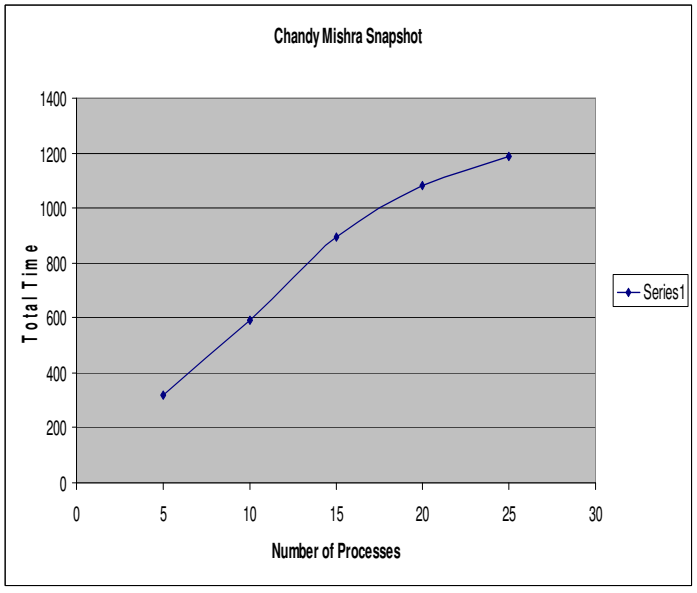
| Processes | Messages | Time |
|---|---|---|
| 25 | 360 | 1180 |
| 20 | 240 | 1080 |
| 15 | 110 | 894 |
| 10 | 72 | 594 |
| 5 | 38 | 318 |

Table 3.1




Graph 3.1
Processes Vs Messages

Graph 1.1 shows the exponential increase in the number of messages passes in the system as the number of processes increase.




Graph 3.2
Processes Vs Time

Graph 3.2 shows the increase in the time taken by the algorithm as the number of processes increase.

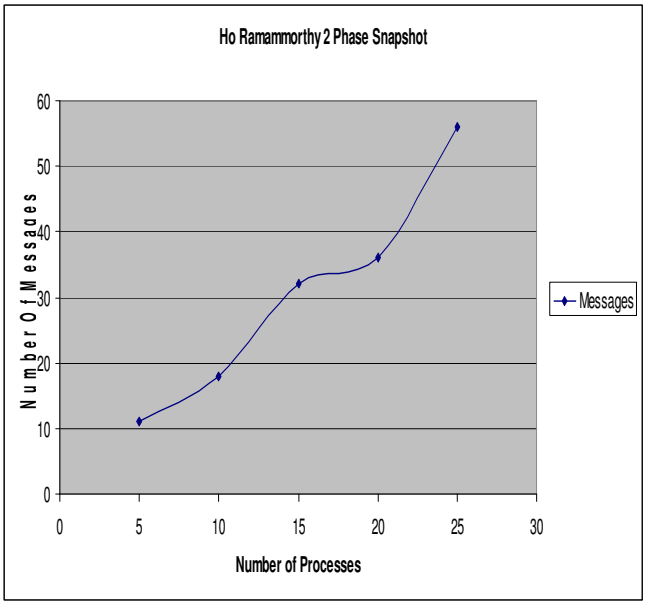
Interesting observations are made after number of processes is more than 15. The degree of increase in the time is reduced to some extent.

Section IV. Analysis of Ho-Ramammorthy's 2-phase deadlock detection:

The used data to analyze the behavior of the algorithm is presented in Table 4.1

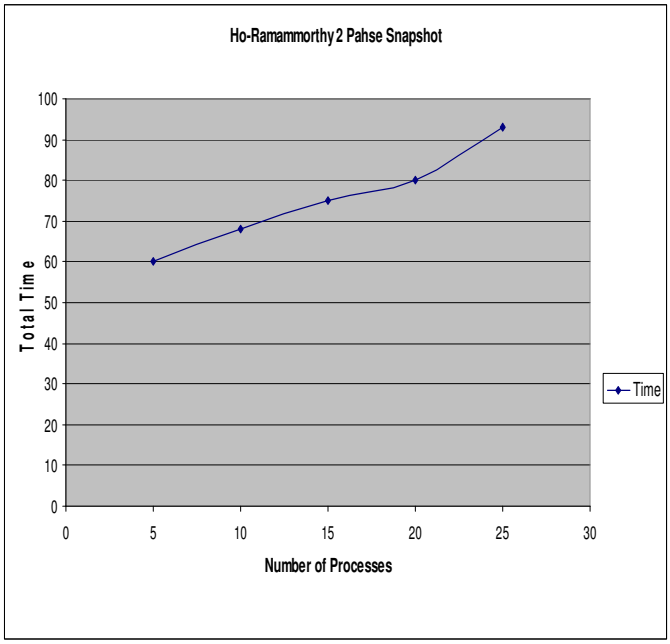
| Processes | Messages | Time |
|---|---|---|
| 25 | 56 | 93 |
| 20 | 36 | 80 |
| 15 | 32 | 75 |
| 10 | 18 | 68 |
| 5 | 11 | 60 |

Table 4.1




Graph 4.1
Processes Vs Messages

Graph 4.1 shows the time complexity of Ho-Ramammorthy's algorithm. Graph depicts a normal behavior. It shows gradual increase in number of messages as the number of processes increase.

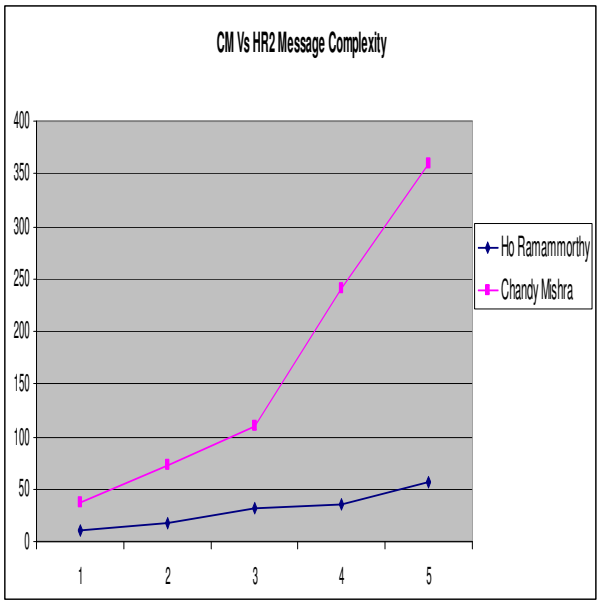



Graph 4.2
Processes Vs Time

Graph 4.2 shows time complexity of Ho-Ramammorthy's 2-phase snapshot algorithm. Graph depicts normal behavior, gradual increase in time taken as the number of processes increase.

Section V. comparing both the algorithms on calculated Time and message complexity:

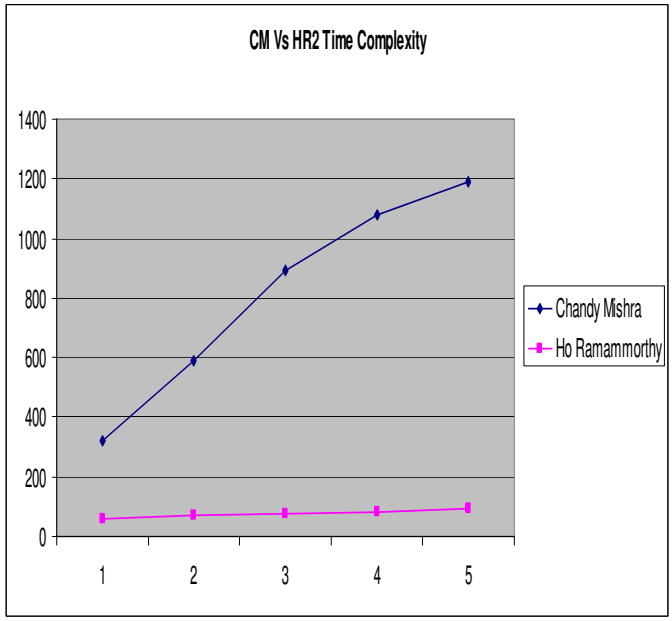
Message complexity:
As we have calculated the Time and Message complexity of the algorithms we can compare the performance of the algorithms.




Graph 5.1
Message complexity

Graph 5.1 depicts the performance difference in context of Message complexity. Ho-Ramammorthy's 2-phase deadlock detection algorithm has better message complexity than Chandy-Mishra snapshot algorithm.

Time complexity:




Graph 5.2
Time Complexity

As observed in Graph 5.2 Time complexity of Chandy-Mishra's snapshot algorithm is better than Ho-Ramammorthy's 2-phase deadlock detection.

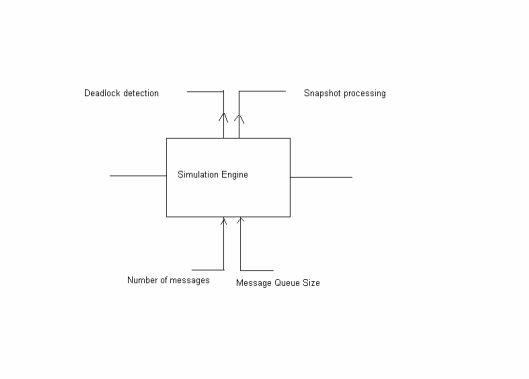
VI. Implementation details:

The algorithm implementations are on Java platform. The implementation can be divided into following parts.

1. Main simulation engine
2. Message forwarding and Queue implementation modules.

We will briefly describe the simulation engine for both the algorithms here. Simulation engine simulates the topology as a graph. It is responsible for taking

snapshots (Chandy Mishra) and detects deadlocks (Ho-Ramammorthy). Simulation engine implements the message forwarding and false deadlock detection methods.




VII. Conclusion:

As seen in the performance comparison graphs of both the algorithms following conclusions can be listed.

1. Message Complexity of Ho-Ramammorthy 2 phase snapshot is better than Chandy Mishra snapshot algorithm.
2. Time complexity of Ho-Ramammorthy 2 phase deadlock detection is better than Chandy Mishra snapshot algorithm.
3. Possibility of false deadlock detection in the implementation of Ho-Ramammorthy's 2-phase deadlock detection remains.
4. Algorithm implementations need to be tested in real distributed systems.

References:


- Distributed Deadlock Detection- K Mani Chandy and Jaydev Mishra.
- 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.
- LAMPORT, L. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM 21,* 7 (July 1978), 558-565.
- 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.