Name: Prashanth Mallyampatti
Unity ID: pmallya
Student ID: 200250501

CSC724-Advanced Distributed Systems
Paper Review

Time, Clocks, and the Ordering of Events in a Distributed System

Summary:

This paper is about how to order events in a distributed system, and mechanisms to synchronize processes with some assumptions.

The problem addressed by this paper is that, how to maintain and predict the order of events occurring in distributed systems. That is, there should be a strong guarantee that if process 'a' occurs before process 'b', the ordering state is preserved across the entire system. Usual ordering is using the conventional physical time which require real clocks. Real clocks further impose out of sync problems as no two clocks will run exactly at the same rate. So how to define a convention which tells which event has occurred first. Since a distributed system is a decentralized system, having a centralized clock contradicts its true nature, and may also cause fault tolerance issues. Hence the notion of physical time solely cannot be taken into consideration for ordering of events in a distributed system.

To solve this problem, Leslie Lamport came up with the term "happened before", which is a relative term used to order events. Lamport proposes solution to this in four phases:

1. Partial Ordering: Here a simple "happened before" relation- if 'a' event comes before 'b' event in the same process, and 'a' is sending a message to 'b' and 'b' is a recipient (both events of different processes), then a→b. This relation imposes a simple constraint on 'a' and 'b' which satisfy the condition of simple ordering.
2. Logical Clocks: Extending the partial ordering, logical number $C_i<a>$(considered as logical time), which is usually generated using counters, is assigned to an event. This paper defines this logical number by the message timestamps set at the sender and receiver end. By adding a Clock Condition: For any events a,b: if a→b then $C<a> < C<b>$, total ordering of events has been proposed.
3. Total Ordering: For total ordering a '=>' relation is defined, which according to the paper satisfies the Clock condition, as well as if $C<a> = C<b>$, then an arbitrary $<$ total ordering of process is chosen between processes in a mutual exclusion problem. The algorithm discussed in this paper, considers a request queue for each process, which follows a set of rules to acquire or release resources in order to achieve total ordering.
4. Physical Clocks: Even though total ordering of events is achieved, an anomalous behavior-when external messages take part in ordering has been observed. To solve this, Lamport proposes the use of physical clocks and introduces strong clock condition so as to account for the information exchanged by users outside the system. Formal proofs for clock synchronization & error margins, physical clock conditions have been defined so as to achieve proper ordering of events on top of logical clocks.

Strong Points:

1. Ordering without physical clocks, at least for a system with no external events is more elegant way to solve the problem.
2. The algorithm is simple and addresses major problems involved in ordering of events, synchronization which is crucial in financial applications.

Weak Points:

1. A node when goes down and reboots again, there is no scope of resynchronization in this proposed algorithm.
2. The algorithm also assumes that the every process messages to every other process, which introduces huge traffic.
3. The total ordering solution proposed starts with an assumption of processes being mutually exclusive. Thus deadlock problem isn't discussed.