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

Sohail Ahmed Shaikh

January 14, 2019

1 Summary:

The goal of this paper is to clearly address the problem of synchronization and total ordering of events in a multi-process environment, in the absence of clocks that can keep physical time.
To do this, the paper gives an insight into the "happened before" relationship between events.
It describes this relationship in context of the partial ordering of events in a distributed multi-process environment.
Using this definition, the author attempts to extend it to define total ordering of events and how it can be used for synchronization problems.
In essence, the ordering is achieved by an algorithm which utilizes time-stamped message passing between all the processes of a system.
Finally, physical clocks are introduced in the system as an attempt to solve anomalous behavior due to the differences in the actual ordering and the order perceived by users.

2 Description:

In a distributed system the perception of time is difficult to comprehend because the system consists of a collection of processes that could be located on different systems and the only way of communication between them could be passing messages. Also, the message transmission delay in such a system is significant and cannot be ignored.

Thus, if events are occurring in a set of processes, it is not straightforward to determine which event occurred first.

The happened before relation ("$\rightarrow$") for events of two processes can be determined with the help of logical clocks which the author introduces in the absence of clocks that keep the physical time.
Using this concept, a set of processes can decide the clock of events that occur in each process, based on messages of an event that have a time-stamp assigned to them. All the processes coordinate with each other by exchanging these time-stamped event messages.
In this way, a partial ordering of events between processes in such a system can be achieved.

Now, this idea can be extrapolated to the whole system to achieve system wide synchronization between all the processes using only logical clocks. The author describes how total ordering can be achieved when a set of processes are competing for a resource.

The challenge in such a system is that a centralized scheduling scheme doesn't work because network

delays and out of order delivery of requests will make it difficult to ensure that a process which requested a resource first was granted it.

The author describes an algorithm that can be used to achieve this with the help of message passing and request queues. In short, when a process wants to request for a resource, it sends a message with a time-stamp to all the processes in the system as well as it's own request queue. The other processes that receive this message acknowledge this with a time-stamped acknowledgement message. A similar message passing and removal is done in the case when a resource has to be released. Based on the order of the time-stamps in the request queue of the processes, the resource can be acquired by a process.
This ensures a total ordering of events in a system using only logical clocks.

An anomalous behavior can occur if there are some external events transparent to the distributed system. This can make the ordering incorrect with logical clocks. To solve this problem the author introduces synchronization of physical clocks.

3 Strong points:

1) The concepts of logical clocks is quite novel as it enables partial ordering of events in the absence of a physical clock.
2) This further enables a distributed algorithm to achieve total ordering of events in a system without the requirement of a centralized synchronization process or central storage.
3) The approach is quite generic and could be applied to implement any desired synchronization in a multi-process environment.

4 Weak points:

1) This system has a huge overhead as all the processes must participate for synchronization. All the processes must know the entire global state which is counter-intuitive to a stateless distributed system and extremely expensive.
2) Most modern distributed systems are designed with commodity hardware in which components fail frequently. However, in this system, failure of a single process could halt the system.
3) A large scale distributed system processes may not be able to communicate directly with other processes which is one of the assumptions in this paper.
4) Also there is no mechanism for loss of packets and an assumption has been made that the messages will be delivered in order which may be unrealistic in modern scenarios.

5 Improvements:

The paper can be improved by accounting for some kind of fault tolerance when a process that is participating in synchronization crashes. So some improvements in reliability of the system can be definitely made. Also the same reliability guarantees could be provided for network failures.