

CS 262 Paper Assignment 1

The following two conditions are sufficient for a collection of state machines to perform identical computations:

Agreement: Every non-faulty state machine replica receives every request.

Order: Every non-faulty state machine replica processes the requests it receives in the same relative order.

If these two conditions are satisfied, it is clear that all functioning replicas will pass through an identical sequence of states, since the state of each replica only changes in response to incoming requests and since all commands are deterministic. These conditions can be weakened somewhat while still preserving replication; for example, the paper comments that read-only requests of fail-stop machines need not be replicated, since such a request does not cause state to change and since any machine will yield the correct answer. Furthermore, events that commute can be allowed to occur in any order on any given replica since they will produce the same final state regardless.

Now we can consider the communication requirements of various implementations that enforce the conditions above. To support agreement, several strategies can be pursued. It is perhaps most straightforward to allow all clients to communicate with all state machine replicas; in this case, the client plays the role of the transmitter. It is also possible for one of the state machine replicas to serve as the transmitter, in which case the client only communicates with that replica and the replica communicates with all other replicas. To allow the client to check that the message is passed along, the transmitter may broadcast the request back to the client as well as to the other replicas. In any case, the communication is accomplished with a so-called agreement protocol to guarantee that all non-faulty processors agree on the transmitter's value if it is functioning properly.

There are several ways to implement the order condition, and they have varying communication requirements. One approach uses logical clocks, which assign numbers to events. These numbers need not be related to physical time; they are simply required to satisfy certain conditions. In particular, the clock value must be incremented after each event within a process, and a process that receives a timestamped message must increment its clock value beyond the max of its current value and the timestamp. This implies a communication requirement —

all messages must include timestamps. It is also convenient to assume that the messages between pairs of processors are delivered in the order sent; this FIFO property can be implemented with sequence numbers. To guarantee that requests will be executed in the same order at all replicas, all clients must periodically make a request (possibly a null request) to the state machines; a request is then considered stable (and can be acted upon) when another request with a larger timestamp has been received from all non-faulty clients. Since logical clock values only increase and we assume messages are delivered in the order that they were sent, the collection of requests from clients with larger timestamps than a given request indicate that no request with a smaller timestamp is still on the way.

Another way to satisfy the order condition is to employ synchronized real-time clocks. As long as these clocks are synchronized to within an interval that is shorter than the minimum message delivery time, and as long as no client makes more than one request between successive clock ticks, properties O1 and O2 from the paper are upheld. In this case, all processors must communicate to keep their clocks synchronized. Clock synchronization protocols exist that make it possible to satisfy the condition that the synchronization window remains smaller than the message delivery delay. If the FIFO channel assumption is valid for the communication network, the state machines can operate without the delay inherent in the basic synchronized clock implementation.

Finally, the order condition can be implemented with replica-generated identifiers; this approach requires less communication than the previous two approaches — only the client and the state machine replicas must communicate. Unique identifiers for each request are computed in two phases: the state machine replicas first propose candidate unique identifiers, and then one of these identifiers is accepted by all. In order for O1 and O2 to hold, an additional communication restriction must be imposed on the clients — once a client starts to send a request to the state machine replicas, it must not perform any other communication until all of the replicas have accepted its request; otherwise, causality could be violated. This method also requires that the state machine replicas communicate candidate unique identifiers with an agreement protocol. If the processors can experience Byzantine failures, this system must be enhanced with timeouts and the ability for state machine replicas to broadcast timeout information to each other via an agreement protocol.