Distributed Algorithms
Lecture 12

Lecture by Dr. Gadi Taubenfeld
Typeset by Steven Karas

2017-06-21
Last edited 17:52:52 2017-06-28

Disclaimer These lecture notes are based on the lecture for the course Distributed Algorithms, taught by Dr. Gadi Taubenfeld at IDC Herzliyah in the spring semester of 2017. Sections may be based on the lecture slides and accompanying book written by Dr. Gadi Taubenfeld.

Agenda

- Snapshot algorithm

1 The snapshot problem

Presented by K. M. Chandy and L. Lamport in 1985, and Dijkstra (as a summary of their algorithm) in 1984.[1]

The snapshot problem is that of discovering the global state of a distributed system by collecting information from local links and vertices.

Applications Used for discovering global state, for example deadlock detection, backups, debugging, termination detection, etc.

For example, take a network of nodes with values, trying to find the GCD of all the values. Each node takes on the GCD of itself and all its neighbors. As such, each node cannot know when the network has finished.

Stable Property A stable property is a property that once it holds, it will hold in all possible future states (e.g. deadlock).

The snapshot algorithm needs to collect state information such that stable properties can be detected.

Snapshot state The snapshot state (S_{SS}) is the observed global state of the system somewhere between the initial (S_0) and terminal (S_1). The snapshot state is a possible state that follows S_0, and S_1 is a state that is possible after the snapshot state.

If a stable property holds in S_{SS}, then it must hold in S_1. Therefore, even if the actual execution of the system took a different path of state transitions from S_0 to S_1.

Consistent Cut A consistent cut is a snapshot of the global state such that it captures the state of all the processes and all inflight messages. An example of this is on slide 16.

1.1 Chandy and Lamport's Algorithm

Model

- Asynchronous network. Directed, well connected, finite graph with edges E between n processes.
- FIFO links without any failures.
- Global state is defined as the local state of each process and a list of the messages on each link.

[1] Copies of the papers are included in the lecture notes

We assume that each process, operation, and message is either colored white or red. Every operation is colored with the color of the process that ran it. Every message is colored with the color of the process that sent it.

The system starts with all processes colored white. The system at termination will be colored red.

A **marker** message recolors a process from white to red.

The Algorithm At the start, some process spontaneously turns red and sends marker messages on all its outgoing edges.

A white process that receives a marker message recolors itself red, records its current state, and sends marker messages on all its outgoing edges.

A red process that receives messages records them until it receives a marker message from that edge.

This continues until every process has received a marker message from all its incoming edges.

At the end, the S_{SS} is the state when each process became red, and the list of white messages received on each link by a red process.

1.1.1 Correctness

Definition: The snapshot state is S Take some possible execution of the system. Let S be the snapshot state.

In S, all nodes and messages are white. Starting from S, a node does not get any message before it turns red. If it does, the message should have been white. Thus, its local state when it changes to red is the same as in S.

All the white messages in S will be recorded.

Claims Every process turns color to red because the graph is strongly connected.

Every edge gets exactly one marker. A marker is sent only when a process turns red.

A process will know when the algorithm has completed in its local frame, because it gets markers from all its incoming edges.

Commutative Operations Two successive atomic operations from different processes commute unless the first sends a message that is accepted by the other.

By interchanging successive commutative operations, we can derive an execution that is equivalent to that without the interchange.

Recall that every operation is either white or red. A red message is never accepted by a white process. A red operation and a subsequent white one are from different processes. Therefore, a red operation and a subsequent white one commute.

If we sort the operations and the interleaving states such that all the white operations precede the red, then the system state at that moment is the snapshot state we recorded in the algorithm.

Properties

- Asynchronous - nothing is assumed about relative speed of processes
- Symmetric - no leader necessary
- Generic - no topology assumptions (beyond connectedness)
- Complexity - $O(E)$ without collecting the state
- One-shot - can not be repeated without modification (although possible)

2 Next week