COS418 Precept 4 - Distributed Snapshots

10/4/19

Chandy-Lamport Snapshot Algorithm

Starting the snapshot process on a server:

- Record its local state
- Send ***marker messages*** on all outbound interfaces

When you receive a ***marker message***:

- If you haven't started the snapshot process yet, record your local state and send ***marker messages*** on all outbound interfaces
- Start recording messages you receive on all *other* interfaces
- Stop recording messages you receive on *this* interface

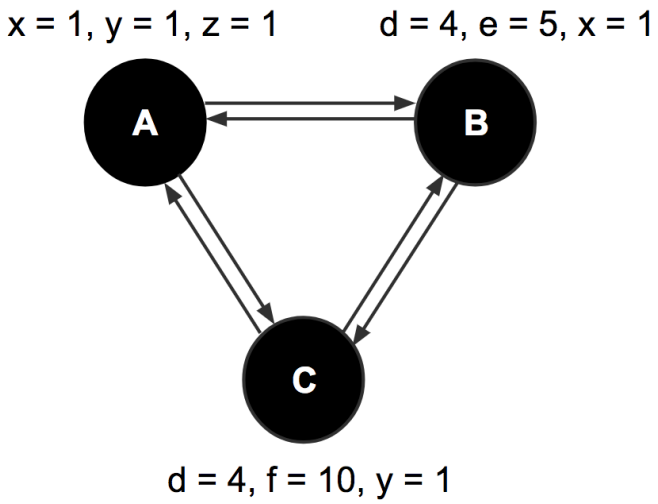
Terminate when all servers have received ***marker messages*** on all interfaces

Distributed Database Example

Below is a distributed database consisting of 3 servers, A, B, and C. Each server is responsible for storing a fraction of the data. Some keys are replicated on multiple servers for fault tolerance and availability. You are also given a sequence of events for this database.

The ***Set(key, value)*** message may be exchanged between servers if either:

- The client contacts the wrong server, in which case the contacted server will forward the request to the server that is responsible for the key of interest, or
- The client contacts the right server but the key is replicated, in which case the contacted server will forward the request to other replicas holding the same key



Local state:

A	x = __, y = __, z = __
B	d = __, e = __, x = __
C	d = __, f = __, y = __

Link state:

Link(BA)	
Link(CA)	
Link(AB)	
Link(CB)	
Link(AC)	
Link(BC)	

Run the Chandy-Lamport snapshot algorithm on the following sequence of events, recording the following:

1) The steps at which the snapshot process starts and finishes on each server (after it is initiated by node C).
2) The snapshot taken by the algorithm. Use the tables above to record local state on nodes and any in-flight messages over links.

B	`e = 10`
A → B	sends `Set(d, 8)`
A → C	sends `Set(d, 8)`
B	receives `Set(d, 8)` from A, `d = 8`
C	`y = 3`
C → A	sends `Set(y, 3)`
C	***starts snapshot***
C	receives `Set(d, 8)` from A, `d = 8`
A	receives `Set(y, 3)` from C, `y = 3`
B	receives `Marker` from C
B	`e = 4`
B → A	sends `Set(x, 4)`
A	receives `Marker` from C
A	receives `Marker` from B
A	receives `Set(x, 4)` from B, `x = 4`
C	receives `Marker` from B
C	receives `Marker` from A
B	receives `Marker` from A