CLOUD COMPUTING CONCEPTS

with Indranil Gupta (Indy)

LEADER ELECTION

Lecture A

THE ELECTION PROBLEM

Why Election?

- Example 1: Your Bank account details are replicated at a few servers, but one of these servers is responsible for receiving all reads and writes, i.e., it is the leader among the replicas
  - What if there are two leaders per customer?
  - What if servers disagree about who the leader is?
  - What if the leader crashes?

  Each of the above scenarios leads to Inconsistency

More motivating examples

- Example 2: (A few lectures ago) In the sequencer-based algorithm for total ordering of multicasts, the "sequencer" = leader

- Example 3: Group of NTP servers: who is the root server?

- Other systems that need leader election: Apache Zookeeper, Google's Chubby
- Leader is useful for coordination among distributed servers

Leader Election Problem

- In a group of processes, elect a *Leader* to undertake special tasks
  - And *let everyone know* in the group about this Leader
- What happens when a leader fails (crashes)
  - Some process detects this (using a Failure Detector!)
  - Then what?
- Focus of this lecture: Election algorithm. Its goal:

  1. Elect one leader only among the non-faulty processes
  2. All non-faulty processes agree on who is the leader

System Model

- *N* processes.
- Each process has a unique id.
- Messages are eventually delivered.
- Failures may occur during the election protocol.

Calling for an Election

- Any process can call for an election.
- A process can call for at most one election at a time.
- Multiple processes are allowed to call an election simultaneously.
    - All of them together must yield only a single leader
- The result of an election should not depend on which process calls for it.

Election Problem, Formally

- A run of the election algorithm must always guarantee at the end:
  - **Safety**: For all non-faulty processes *p*: (*p*' s elected = (q: a particular non-faulty process with the best attribute value) or Null)
  - **Liveness**: For all election runs: (election run terminates)

    & for all non-faulty processes *p*: *p*' s elected is not Null
- At the end of the election protocol, the non-faulty process with the best (highest) election attribute value is elected.
  - Common attribute : leader has highest id
  - Other attribute examples: leader has highest IP address, or fastest cpu, or most disk space, or most number of files, etc.

Next

- A classical leader election algorithm.