Paxos

CSEP590SG– University of Washington

Steve Gribble (gribble@cs.washington.edu)

[This material is cobbled together from various papers by Butler Lampson and Leslie Lamport.]

Context

- **Start with a (known) set of leaders and agents**
  - leaders can be agents, or leaders might not be agents
- **Goal of system is to pass a decree**
  - system proceeds through sequence of rounds until decree is passed
  - any leader can choose to begin a sequence for a new decree
    - and, multiple leaders can offer opinions on what value of decree is
  - termination: majority of agents agree on the same outcome of decree
- **A round:**
  - leader "proposes" value, agents may "accept" value
  - value is "chosen" as soon as majority of agents accept the same value in a round

Comments on context

- **Unlike BGP:**
  - decrees can be started at any time
    - in BGP, problem is phrased so that consensus problem has already begun
  - byzantine failures are not tolerated
    - all agents “believe” anything that any leader proposes
    - the consensus problem is about conflicting proposals, not untrustworthy participants
      - *[alternatively, about order of proposals: conflict is disagreement on which goes next]*
  - no assumptions about reliability of network
    - besides the fact that messages are never corrupted
    - messages can be dropped, reordered, delayed, duplicated, etc.

“Correctness”

- **Paxos is a protocol that:**
  - guarantees correctness under all circumstances
    - including # of simultaneous leaders, # and rate of leader/agent failure and recovery, and bad network juju
  - terminates under some circumstances
    - if a single leader runs by itself in a round for long enough time to talk to majority of agents twice
- **“correct” := safety + liveness**
  - safety [a.k.a. agreement + integrity]
    - only a single value that has been proposed may be chosen
    - only a single value is chosen
    - agent never learns that a value is chosen unless it has been
  - liveness [a.k.a termination]:
    - terminates under certain circumstances

Basic idea for single decree synod

- **Name rounds by (round #, leader name)**
  - thus, guarantee only a single leader per round
    - leader names must be unique and ordered
  - assume we are in round X
    - a round earlier than X may proceed/finish after X finishes
    - a round after X may "stomp all over" X
    - need to worry about both cases
- **In each round:**
  - leader first "**interrogates**" agents to figure out what decisions have been made in the past
  - *if hears back from a quorum of agents*, leader then "**proposes**" a value for the decree *consistent with what has happened in past*, else give up round
  - if majority of agents see and accept proposal, the value is chosen and the algorithm has "morally terminated"

Challenges faced by idea

- **Previous or future rounds may be temporally concurrent**
  - and hence agents may see proposals from many rounds at same time, and worse, those proposals may conflict
- **Leader may fail**
  - and hence not send proposals to enough agents
- **A leader or agent may wake up after a long slumber**
  - and not know what is going on, or what happened in the past
  - for example, a leader may wake up and not know that the algorithm has terminated! (i.e., that a value was chosen)

- ***Asynchronous system: failure and slumber indistinguishable***

Idea: use correctness conditions to deduce constraints on protocol

- **imagine only a single leader ever exists, it interrogates, then sends out its proposals, then dies.**
  - if a majority of agents hear proposal, the proposal must be chosen, according to termination condition
  - hence, an agent must accept first proposal that it hears
    - because it can't know if more proposals are coming, and it can't know whether or not other agents accept or not
- **safety condition: only a single value is every chosen**
  - thus, if in round M a proposal V is chosen…
    - then every higher-numbered proposal that is chosen has value V
    - but: a leader cannot predict whether a proposal will be chosen or not- it must assume that it might be chosen
      - thus, every higher-numbered proposal must have value V.

What this implies about leaders

- **During the interrogation phase, a leader must find out what proposals might have been chosen already**
  - if it is conceivable that a proposal might have been chosen in the past…
    - the leader must select the same value for its future proposals
  - using agent state, figure out rounds that are dead. if all dead, pick any value. of any non-dead, must pick that value.
- **Also, leader must prevent any "temporally concurrent" proposals from previous rounds from being chosen**
  - since their value might conflict
  - convention: later numbered rounds "squelch" earlier numbered rounds

What this implies about agents

- **If agent hears an interrogation in round M, it atomically:**
  - squelches any rounds earlier than M
    - what this means in practice is accepting "no" for that round, where "no" is a special value that says the agent believes the round should fail
    - majority of "no" votes means the round has failed
  - returns its history [what it accepted] for rounds earlier than M-1
- **Note that at this point, the history of all rounds earlier than M is fixed for that agent**
  - no future rounds can change the outcome of these earlier rounds, under any circumstances
  - history reported is always complete - leader gets all or none

A nice side-effect of majority

- **How does a leader know what past values might have been chosen?**
  - if a round is chosen, then a majority of agents accepted the value
  - any two majority sets share at least one agent
  - during interrogation, the leader self-imposes the requirement that it hears back from a majority of agents
    - if a value has been chosen in the past
      - then, at least one agent

        that the leader heard back from

        is an agent that accepted the chosen value

    [byzantine: need to hear back from majority of “good” agents, hence 3K+1, not 2K+1]

How to reason about the past

- **So, after interrogation:**
  - if leader doesn't hear back from majority, round dies [no action needed]
  - if leader hears back from majority, then:
    - if nobody in majority has accepted any proposals ever [everybody said "no" for all rounds], leader can propose any value it wants
    - if all earlier rounds are "dead"- provable that majority said no- leader can propose any value it wants
    - else, not provable that some earlier round was dead- leader must assume the value in that round was chosen by majority
      - leader figures out value of the highest numbered proposal that somebody has accepted in a non-dead round, and proposes that value

Why use highest numbered proposal?

- **Any proposal accepted by an agent in a non-dead round is OK**
  - thus, as long as it doesn't violate correctness, it is OK for the leader to use the highest numbered proposal from set of non-dead rounds
- **If the leader uses this, we can prove correctness**
  - using highest numbered proposal provides an "induction" across all rounds
  - Assume a value is chosen in round M
    - all "earlier" rounds are squelched
    - and thus, all "later" rounds will have same proposal
      - because no other value can ever be proposed
  - If a value has been proposed but not chosen…
    - concurrent proposals might be happening to different non-majority sets
    - leaders might discover any (or none) of these values during interrogation

Another way of thinking about it..

- **Assume there are 5 agents, and 2 leaders L1,L2**
- **Leader doesn't know whether a value is chosen**
  - manifestly, else it wouldn't be participating anymore
- **Assume leader L1 interrogates in round 3, and gets:**
  - round (1, L1): {1, -, -, no, 1}
  - round (1, L2): {2, -, -, no, no}
  - round (2, L1): {no, -, -, 1, 1}
  - round (3, L1): {1, -, -, 1, 1}
    - what is correct outcome?
- **How about:**
  - round (1, L1): {1, -, -, -, 1}
  - round (1, L2): {2, -, -, -, no}
  - round (2, L1): {no, -, -, -, 1}
  - round (3, L1): {1, -, -, -, no}
- **FORCED to choose latest (possibly) non-dead round value**

Another pop quiz…

	agent:	{a, b , c}
round 1:	vote 7	{7, no, no}
round 2:	vote 8	{8, no, no}
round 3:	vote 9:	{no, no, 9}

what are choices for leader in round 4, if:

- all a,b,c report?
- if a,b report?
- if a,c report?

More detail

- **A leader will look back through the history from interrogation, and:**
  - skip rounds that are "dead"
    - rounds with no value reported at all
    - rounds in which it can prove there is no majority, because it heard from enough "no" votes
  - once it hits a round that might not be dead
    - it picks the value reported from that round to propose in the future
    - because, it can't tell whether or not a majority accepted the value, so it must pessimistically assume that it did
  - if all previous rounds are dead
    - it picks any value that it likes

Another pop quiz…

	agent:	{a, b , c}
round 1:	vote 7	{7, no, no}
round 2:	vote 8	{8, no, no}
round 3:	vote 9:	{no, no, 9}

what are choices for leader in round 4, if:

- all a,b,c report? anything - all rounds dead
- if a,b report? must choose 8: r3 dead, can't tell r2
- if a,c report? must choose 9: can't rell r3 dead
- if b,c report? must choose 9: can't tell r3 dead

It turns out that…

- **If an agent wants, it can just report its latest accepted value, and that's good enough**
  - But this has implications. Consider the following two cases:

M	no	no	1	no	1
M+1	3	no	no	3	no
M+1	no	2	no	no	no
M+2	2	no	no	2	no

no	no	1	no	1
3	no	*(no)*	3	*(no)*
no	2	*(no)*	no	*(no)*
2	*(no)*	*(no)*	2	*(no)*

Full algorithm

- **Leader:**
  - pick a new round number greater than any other it has chosen
  - interrogate all agents for their status. if not get majority of agents responding, terminate round.
  - if majority responds:
    - pick value to preserve invariant that chosen is stable
    - command (a majority) of agents to accept value
- **If leader wants, it can then**
  - hear back from, or ask, agents to see if a majority did accept
  - and if so, publish the outcome

Full Algorithm

- **Agent:**
  - if hear a new interrogation for a new round:
    - mark “no” for earlier rounds for which it hasn’t accepted a value
    - report either
      - full history of previous rounds
      - or, latest round for which it accepted a value
  - if hear a proposal for a round:
    - if the round is marked “no” or already accepted, drop proposal
    - otherwise, accept proposal
- **If agent wants, can:**
  - broadcast or notify to leader once it accepts a proposal

Other optimizations

- **Stateless leaders**
  - before, a leader needed keep state to pick a higher round number. instead, can interrogate agents for their current highest round number
- **Multiple decrees**
  - if same leader across multiple decrees in common case, then leader doesn't need to query state except at very beginning
    - implies a running leader knows when a leader change occurs, I.e., some new mechanism enforces a single leader and notifies (old,new) when change occurs