CS425 / CSE424 / ECE428 — Distributed Systems — Fall 2011

Paxos

Some material derived from slides by Leslie Lamport

Problem

- Achieve consensus
  - In an asynchronous network
  - Non-Byzantine failures
- Safety requirements
  - Only a proposed value may be chosen
  - Only a single value is chosen
  - Process never learns of a chosen value unless it has *actually* been chosen
- Also, liveness

Actors

- Proposer
  - Proposes values
- Acceptor
  - Accepts (or rejects) values
- Learner
  - Finds out what value has been chosen
- Typically a process acts as one or more of these (often all 3)

Choosing a Value

- Acceptor:
  - Accepts or rejects (ignores) proposals
- Consistency guarantee
  - Value is chosen if and only if a majority of acceptors accepted this value
- Simple (common!) case: one proposer
  - Must accept proposal, o/w no progress!

P1: Acceptor must accept the first proposal it receives.

More proposers

- What if there are two proposers?
  - Proposer 1 sends value v to N/2 acceptors
  - Proposer 2 sends value v' to the other N/2 acceptors
- P1 means that each acceptor accepts the corresponding proposal
  - No majority, therefore deadlock
- Solutions?

Multiple Acceptances

- Acceptors must accept multiple proposals
- Therefore, multiple proposals may be chosen (accepted by a majority). (Why?)
- Therefore, must ensure safety

P2. If a proposal with value v is chosen, then every higher-numbered proposal that is chosen has value v.

- Note: every proposal must have unique number

P2. If a proposal with value v is chosen, then every higher-numbered proposal that is chosen has value v.

- A chosen proposal is accepted by at least one acceptor, so:

*P2^a. If a proposal with value v is chosen, then every higher-numbered proposal **accepted** by any acceptor has value v.*

(P2^a => P2)

Multiple Proposals

P1: Acceptor must accept the first proposal it receives.

P2^a. If a proposal with value v is chosen, then every higher-numbered proposal accepted by any acceptor has value v.

- Asynchronous network
  - Acceptor c does not hear some chosen proposal v
  - Proposer p makes a new proposal v' to c
  - By P1, c must accept. By P2, v' = v.

*P2^b. If a proposal with value v is chosen, then every higher-numbered proposal **issued** by any proposer has value v.*

(P2^b => P2^a => P2)

Invariant

*P2^b. If a proposal with value v is chosen, then every higher-numbered proposal **issued** by any proposer has value v.*

- A chosen proposal is accepted by majority of acceptors

P2^c. For any v and n, if a proposal with value v and number n is issued, then there is a set S consisting of a majority of acceptors such that either:

(a) no acceptor in S has accepted any proposal numbered less than n, or

(b) v is the value of the highest-numbered proposal among all proposals numbered less than n accepted by acceptors in S.

How to satisfy invariant?

- Proposer must know what proposals have been accepted by a majority of acceptors.
  - Ask acceptors about what they have accepted
- But, remember, asynchronous
  - New proposals may be made after reply is sent
- E.g.:
  - p asks c about accepted proposals
  - c replies with empty set {}
  - p' proposes (n,v) to c
  - c accepts
  - p proposes (m,v') with m>n to c, violating $P2^c$

Solution

- Don't accept proposal from p'!
  - p asks c about accepted proposals with numbers less than m *(prepare* request)
  - c replies with empty set {}, **and promises not to accept proposals with numbers less than m**
  - p' proposes (n,v) to c, with n < m
  - c **rejects (ignores) p's proposal**
  - p proposes (m,v') with m>n to c

P1^a. An acceptor can accept a proposal numbered n if and only if it has not responded to a prepare request with a number > n.

Paxos algorithm

- Phase 1 (prepare):
  - A proposer selects a proposal number n and sends a *prepare request* with number n to majority of acceptors.
  - If an acceptor receives a prepare request with number n greater than that of any prepare request it saw, it responses YES to that request with a promise not to accept any more proposals numbered less than n and include the highest-numbered proposal (if any) that it has accepted.

Paxos algorithm

- Phase 2 (accept):
  - If the proposer receives a response YES to its prepare requests from a majority of acceptors, then it sends an *accept request* to each of those acceptors for a proposal numbered n with a values *v* which is the value of the highest-numbered proposal among the responses.
  - If an acceptor receives an accept request for a proposal numbered n, it accepts the proposal unless it has already responded to a prepare request having a number greater than n.

Definition of chosen

- A value is chosen at proposal number n iff majority of acceptor accept that value in phase 2 of the proposal number.

Paxos's properties

- P1: Any proposal number is unique.
- P2: Any two set of acceptors have at least one acceptor in common.
- P3: the value sent out in phase 2 is the value of the highest-numbered proposal of all the responses in phase 1.

Interpretation of P3

#	value	pool of acceptors				
2	α	a_1	a_2	a_3	**a_4**	
5	β	a_1	a_2	**a_3**		a_5
14	α		**a_2**		a_4	**a_5**
27	β	**a_1**		**a_3**	**a_4**	
29	β		**a_2**	a_3	a_4	

Proof of safety

- Claim: if a value v is chosen at proposal number n, any value that is sent out in phase 2 of any later prososal numbers must be also v.
- Proof (by contradiction): Let m is the first proposal number that is later than n and in phase 2, the value sent out is not v.

Proof

#	value	pool of acceptors
n	v	…
		a
n+1	v	…
		…
m-1	v	…
		a
m	v’	…

the highest # chosen in phase 2 ≥ the highest # that a accept ≥ n

Learning a chosen value

- There are some options:
  - Each acceptor, whenever it accepts a proposal, informs all the learners.
  - Acceptors informs a distinguished learner (usually the proposer) and let the distinguished learner broadcast the result.

Tunable knobs

- Acceptors have many options to response:
  - Prepare request: No/Yes
  - Accept request: No/Yes if it didn’ t promise not to do so
- Back off time after abandon a proposal: exponential back-off/pre-assigned values
- Should we wait for nodes to online in each phase?

Applications

- Chubby lock service.
- Petal: Distributed virtual disks.
- Frangipani: A scalable distributed file system.