The Paxos algorithm and consensus variants

Fernando Pedone
University of Lugano (USI)
Switzerland

Highlights

- Consensus with leader election
  - ✦ Paxos
- Weaker forms of consensus
  - ✦ k-set agreement
  - ✦ Randomized algorithms

Asynchronous system with leader election

- Paxos
  - ✦ Four process “roles”
    - Proposers
    - Acceptors, a quorum Q_a of acceptors, where $|Q_a|$ is a majority
    - Learners
    - Coordinator/Leader

Asynchronous system with leader election

- Paxos
  - ✦ Leader election oracle
    - Each process p has access to a leader election oracle, which outputs at p a process denoted $leader_p$ such that there is (a) a correct process l and (b) a time after which, for every p $leader_p = l$

Paxos

Variables

c-rnd : highest-numbered round the process has started
c-val : value the process has picked for round c-rnd
rnd : highest-numbered round the acceptor has participated, initially 0
v -rnd : highest-numbered round the acceptor has cast a vote, initially 0
v-val : value voted by the acceptor in round v-rnd, initially null

Phase 1A
Proposer

To propose value v:
    increase c-rnd to an arbitrary unique value
    send (PHASE 1A, c-rnd) to acceptors

Phase 1B
Acceptor

upon receiving (PHASE 1A, c-rnd) from proposer
    if c-rnd > rnd **then**
        rnd ← c-rnd
        send (PHASE 1B, rnd, v-rnd, v-val) to proposer

Paxos

Phase 2A Proposer

upon receiving (PHASE 1B, rnd, v-rnd, v-val) from Q_a such that c-rnd = rnd
    k ← largest v-rnd value received
    V ← set of (v-rnd,v-val) received with v-rnd = k
    if k = 0 **then** let c-val be v
    else c-val ← the only v-val in V
    send (PHASE 2A, c-rnd, c-val) to acceptors

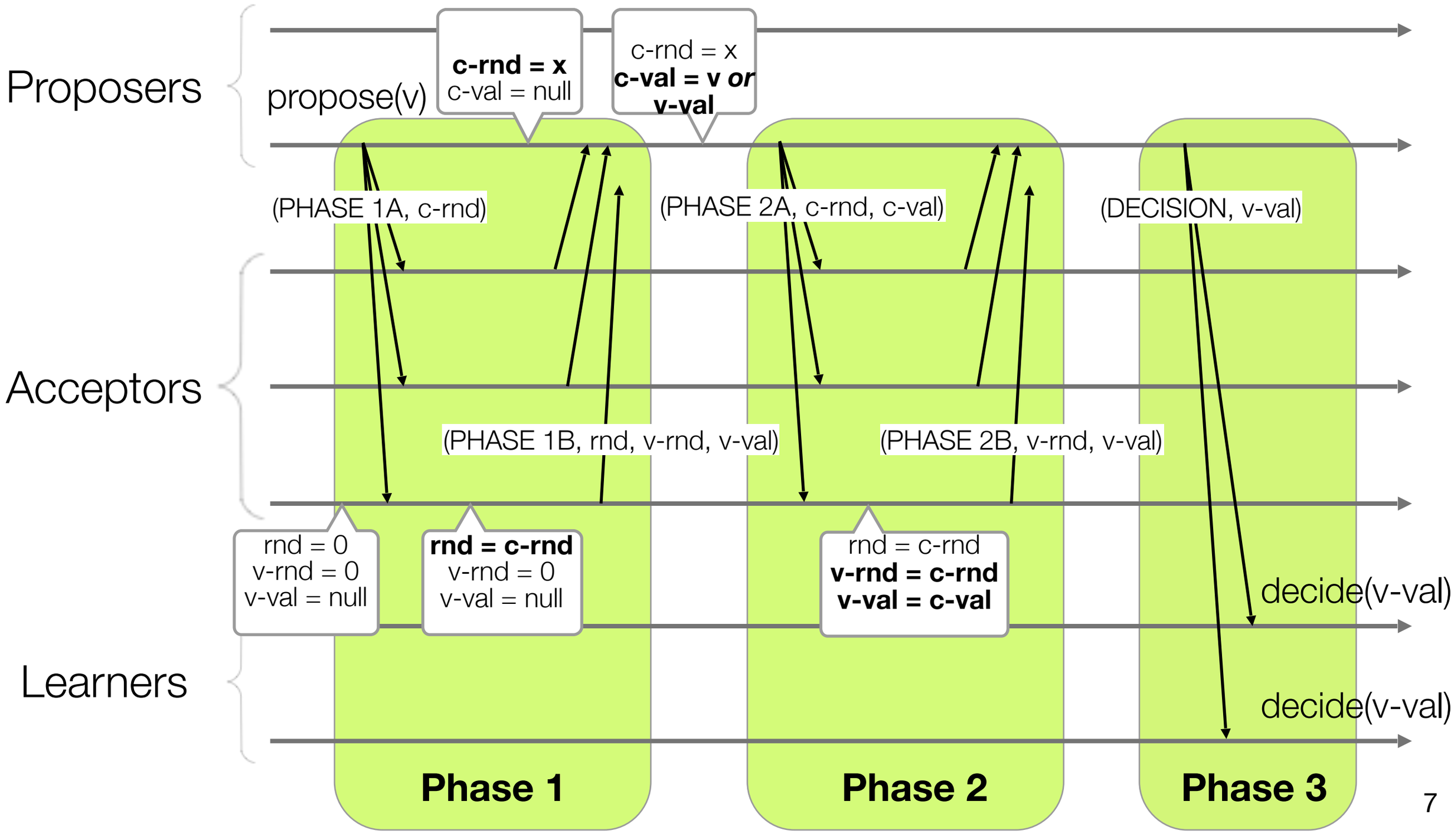
Phase 2B Acceptor

upon receiving (PHASE 2A, c-rnd, c-val) from proposer
    if c-rnd ≥ rnd **then**
        v-rnd ← c-rnd
        v-val ← c-val
        send (PHASE 2B, v-rnd, v-val) to proposer

Phase 3 Proposer

upon receiving (PHASE 2B, v-rnd, v-val) from Q_a
    if for all received messages: v-rnd = c-rnd **then**
        send (DECISION, v-val) to learners

Paxos


Proposers
Acceptors
Learners
propose(v)
c-rnd = x
c-val = null
c-rnd = x
c-val = v or
v-val
(PHASE 1A, c-rnd)
(PHASE 2A, c-rnd, c-val)
(DECISION, v-val)
(PHASE 1B, rnd, v-rnd, v-val)
(PHASE 2B, v-rnd, v-val)
rnd = 0
v-rnd = 0
v-val = null
rnd = c-rnd
v-rnd = 0
v-val = null
rnd = c-rnd
v-rnd = c-rnd
v-val = c-val
decide(v-val)
decide(v-val)
Phase 1
Phase 2
Phase 3

Paxos

- Correctness (intuition behind safety)
  - ✦ Assume learner L decides v, after receiving v from proposer P
  - ✦ Thus, P received (PHASE 2B, v-rnd=x, v-val=v) from Q_a
  - ✦ Let P' be another proposer that sends (PHASE 1A, c-rnd') to a quorum Q_a', so that c-rnd' > rnd for each acceptor in Q_a'
  - ✦ There is at least one acceptor A in $Q_a \cap Q_a$', and so A sends (PHASE 1B, rnd, v-rnd, v-val) to P', where v-rnd = x and v-val = v
  - ✦ It follows that P' picks v for its proposed value

Paxos

- Correctness (intuition behind liveness)
  - ✦ It turns out that the algorithm as presented does not guarantee liveness
  - ✦ Two or more proposers may execute forever alternating executions of Phase 1, without managing to execute Phases 1 and 2 in sequence
- ✦ To ensure liveness, elect a leader among the proposers
- ✦ Process that are not the leader must forward their value to the leader
- ✦ Upon the crash of a leader, another process is elected

Paxos

- Extensions and optimizations
  - ✦ Proposers send their proposal to the leader
  - ✦ The leader can execute Phase 1 before a value is proposed
  - ✦ Acceptors can send Phase 2B directly to the learners
  - ✦ Latency of Paxos (best case)
    - Two communication steps for the leader
    - Three communication steps for all other proposers

Weakening the problem definition

- k-set agreement
  - ✦ Agreement: At most k different values are decided.
  - ✦ Termination: All correct processes eventually decide.
  - ✦ Validity: A decided value is a proposed value.

Weakening the problem definition

- k-set agreement
  - ✦ Trivial solution if f < k
    - The first f+1 processes send their initial value to all
    - A process decides on the first value it receives

Weaker problem and stronger model

- Randomized algorithms
  - ✦ Stronger than the asynchronous model
    - Processes can make random choices
  - ✦ Weak termination
    - Correct processes decide at time t with probability at least p(t)

Ben-Or's algorithm (inspired by)

- Randomized binary consensus algorithm
  - n = 2*f + 1 processes, where f faulty
  - coin() : fair coin, returns 0 or 1

Initialization

estimate ← process p's proposed value
decided ← false
r ← 0

while true

… (next slide) …

Ben-Or's algorithm (inspired by)

```
while true do
    First phase
    send (FIRST, r, estimate) to all
    wait until received (FIRST, r, v) from n-f processes
    if all values received are v then
        estimate ← v
    else
        estimate ← ⊥
    Second phase
    send (SECOND, r, estimate) to all
    wait until received (SECOND, r, v) from n-f processes
    if not decided and (received v from f+1 processes) then
        decide v
        decided ← true
    if there is v ≠ ⊥ s.t. received (SECOND, r, v) then
        estimate ← v
    else
        estimate ← coin()
    r ← r + 1
```