Feb 18
Failure Detectors

Vijay Chidambaram

Recap: models

- Async model: any message or event can take arbitrary time, there are no physical clocks
- Sync model: Messages get delivered within a bounded time, physical clocks can be used
- Async models are more portable, and in practice, timing assumptions in sync models are wrong at least temporarily

Consensus and Atomic Broadcast

- These are two useful primitives
- Consensus: N nodes agree on value X, in the presence of failures
- Atomic Broadcast: N nodes receive the same items in the same order, in the presence of failures
- Consensus and Atomic Broadcast are equivalent problems
- Both impossible to solve under pure asynchrony
  - We will see this result later in class

Failure Detectors

- Consensus impossible in async model because we cannot differentiate a failed node and a slow node
- Propose to solve consensus in asynchronous model by introducing unreliable failure detectors
- Model: async + crash failures + unreliable failure detectors
  - Each process p in the system maintains a list of processes suspected to be crashed

Failure Detectors

- Defined in terms of abstract properties
- Any implementation that provides those properties is then okay
  - Good engineering practice!
- Completeness: every failure is eventually detected
- Accuracy: limits false alarms
- Reducibility: D is reducible to D′ if a dist algo can transform D to D′.

W - weakest failure detectors

- Completeness: There is a time after which every process that crashes is permanently suspected by some correct process.
- Accuracy. There is a time after which some correct process is never suspected by any correct process.
- Allows a process to be wrongly identified as crashed by all processes repeatedly
- Allows a process to be added and removed again and again by other processes in their crash list

W failure detector

- Why is this useful? Guarantees safety
- A consensus system built using W gets safety, but not liveness
- No wrong values are accepted, processes don't accept different values

Failure Detector

- Every process q periodically sends a “q-is-alive” message to all
- If a process p times-out on some process q, it adds q to its list of suspects.
- If p laters gets message from q:
  - It removes q from list
  - It increases time-out value
- This does not satisfy W in async system with unbounded timeouts since correct processes be wrongly suspected forever
- However, in a sync system, this does satisfy W

Weakest failure detector - W0

- W0 satisfies the properties of W, and no other properties
- W0 is necessary and sufficient for solving consensus in async systems

The Model

- Async model, reliable communication channels
- A process fails by stopping
- Once a process fails, it does not recover
- crashed() and correct() are the set of failed and active processes
- We assume that not all processes fail at the same time
- H(p, t) = the list of processes p thinks has failed at time t

Failure Detector (FD) Properties

- Strong Completeness: every crashed process is suspected by every correct process, eventually.
- Weak completeness: every crashed process is suspected by some correct process, eventually.
- Strong Accuracy: No process is suspected before it crashes
- Weak Accuracy: Some correct process is never suspected
- If we want strong/weak accuracy to only hold at some time points, we use eventual strong/weak accuracy: after some time point t, strong accuracy holds

Eventual Strong Accuracy

- Eventual Strong Accuracy. There is a time after which correct processes are not suspected by any correct process.
- Eventual Weak Accuracy. There is a time after which some correct process is never suspected by any correct process.

FD Classes

- Perfect (P): strong completeness + strong accuracy

Completeness	Accuracy			
	Strong	Weak	Eventual Strong	Eventual Weak
Strong	*Perfect* $\mathcal{P}$	*Strong* $\mathcal{S}$	*Eventually Perfect* $\diamond\mathcal{P}$	*Eventually Strong* $\diamond\mathcal{S}$
Weak	$\mathcal{Q}$	*Weak* $\mathcal{W}$	$\diamond\mathcal{Q}$	*Eventually Weak* $\diamond\mathcal{W}$

  FIG. 1. Eight classes of failure detectors defined in terms of accuracy and completeness.

- We can transform a failure detector with weak completeness into one that satisfies strong completeness

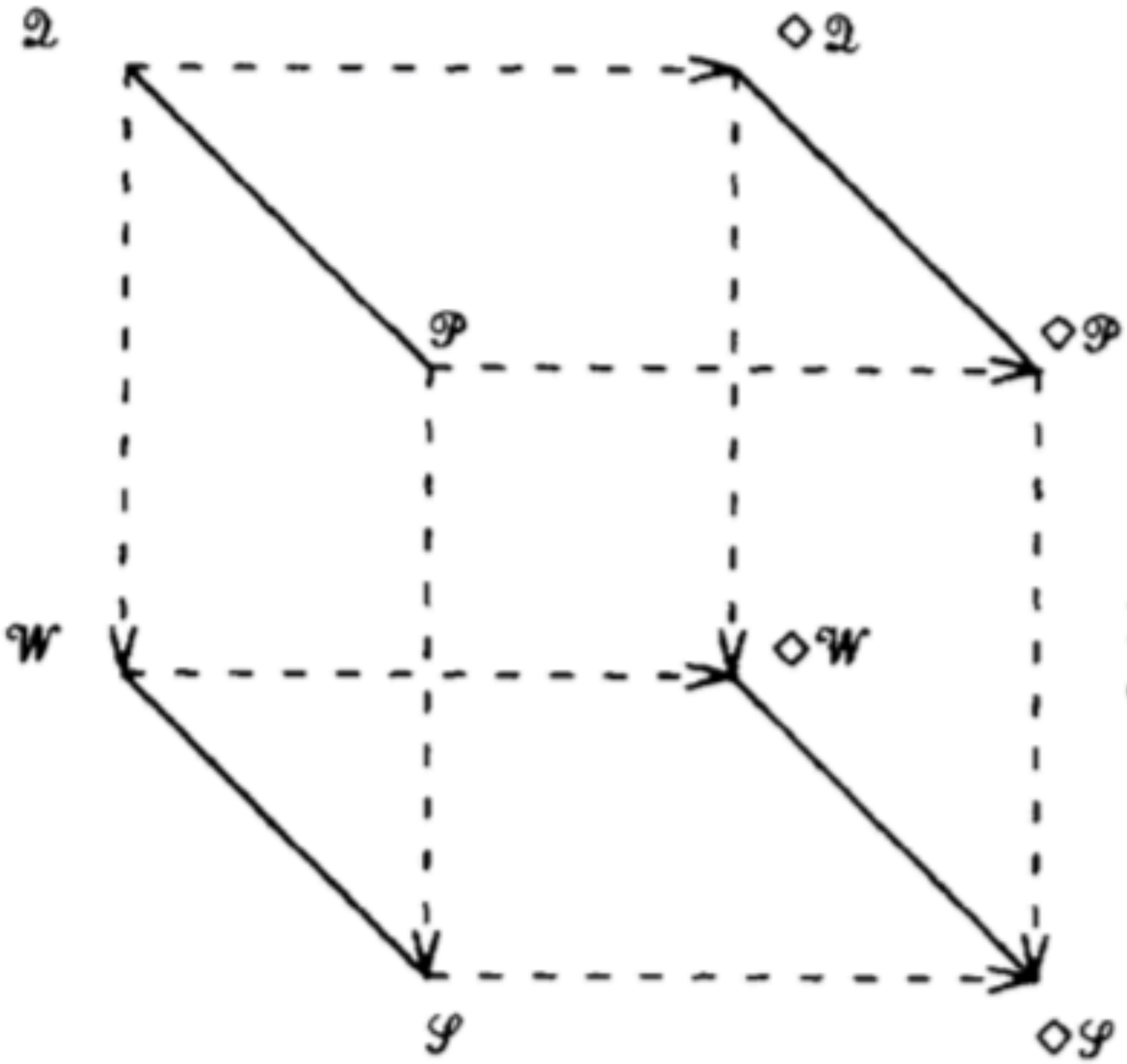

FIG. 8. Comparing the eight failure detector classes by reducibility.

$C \dashrightarrow C'$: C' is strictly weaker than C'

C —— C': C is equivalent to C'

Transforming WC to SC

- WC: Some process suspects every crashed process
- SC: Every process suspects every crashed process
- Transformation algo:
  - Each process p broadcasts suspicion of crashed process q
  - On receiving a message from p about q:
    - Add q to suspects list
    - Remove p from suspects list

FD Classes

- Now we have only four classes, all with Strong C:
  - Perfect: Strong A
  - Strong: Weak A (poor naming)
  - Eventually Perfect: Eventually Strong A
  - Eventually Strong: Eventually Weak A

Reliable Broadcast

- Guarantees:
  - all correct processes deliver the same set of messages,
  - all messages broadcast by correct processes are delivered,
  - no spurious messages are ever delivered.
- Properties:
  - Validity. If a correct process R-broadcasts a message m, then it eventually R-delivers m.
  - Agreement. If a correct process R-delivers a message m, then all correct processes eventually R-deliver m.
  - Uniform integrity. For any message m, every process R-delivers m at most once, and only if m was previously R-broadcast by sender(m).

Reliable Broadcast

- R-broadcast (m):
  - Send m to all (including sender p)
- R-deliver (m):
  - If getting m for the first time:
    - If sender(m) != p then send m to all
    - Deliver(m)

Consensus

- All correct processes propose a value and must agree on a proposed value
- Termination. Every correct process eventually decides some value.
- Uniform integrity. Every process decides at most once.
- Agreement. No two correct processes decide differently.
- Uniform validity. If a process decides v, then v was proposed by some process.

Solving Consensus using Strong FD

- FD has strong completeness but weak accuracy
  - every crashed process is suspected by every correct process, eventually.
  - Some correct process is never suspected
-

Every process p executes the following:

procedure *propose*(v_p)
    $V_p \leftarrow \langle \bot, \bot, \ldots, \bot \rangle$ {*p's estimate of the proposed values*}
    $V_p[p] \leftarrow v_p$
    $\Delta_p \leftarrow V_p$

    Phase 1: {*asynchronous rounds* r_p, $1 \le r_p \le n-1$}
        for $r_p \leftarrow 1$ to $n-1$
            send (r_p, Δ_p, p) to all
            wait until $[\forall q:$ received (r_p, Δ_q, q) **or** $q \in \mathcal{D}_p]$ {*query the failure detector*}
            $msgs_p[r_p] \leftarrow \{(r_p, \Delta_q, q) \mid \text{received } (r_p, \Delta_q, q)\}$
            $\Delta_p \leftarrow \langle \bot, \bot, \ldots, \bot \rangle$
            for $k \leftarrow 1$ to n
                if $V_p[k] = \bot$ **and** $\exists (r_p, \Delta_q, q) \in msgs_p[r_p]$ with $\Delta_q[k] \ne \bot$ **then**
                    $V_p[k] \leftarrow \Delta_q[k]$
                    $\Delta_p[k] \leftarrow \Delta_q[k]$

    Phase 2: send V_p to all
        wait until $[\forall q:$ received V_q **or** $q \in \mathcal{D}_p]$ {*query the failure detector*}
        $lastmsgs_p \leftarrow \{V_q \mid \text{received } V_q\}$
        for $k \leftarrow 1$ to n
            if $\exists V_q \in lastmsgs_p$ with $V_q[k] = \bot$ **then** $V_p[k] \leftarrow \bot$

    Phase 3: *decide*(first non-$\bot$ component of V_p)

FIG. 5. Solving Consensus using any $\mathcal{D} \in \mathcal{S}$.

Algo

- Phase 1:
  - processes execute n-1 async rounds
  - In each round, p broadcast their proposed values
  - In each around, p waits for all other processes not suspected to be crashed
- At the end of Phase 2, everyone agrees on a vector of proposed values
- Phase 3: decided based on first non-null component of vector

Results

- With Strong Failure Detector, we could solve consensus
  - Key is that one correct process is never suspected of being crashed
- We can also solve consensus with weaker failure detectors
  - A FD with weak completeness and eventual weak accuracy is the weakest FD that can be used
  - Weakest FD can solve consensus as long as less than half the processes are faulty

Why are failure detectors important?

- In previous algo, each process p waits for a message for all non-crashed processs
- The failure detector helps identify this set
- Without FD, process p may block forever waiting for process q
- Strong completeness of FD ensures that every crashed process is suspected eventually -> this eliminates the endless waiting