Circumventing Impossibility

Failure Detectors

Circumventing Impossibility

- Use timing assumptions
  - Δ, Φ
- Timing assumptions can be difficult to work with
- What about time-free algorithms?

Time-Free Algorithms

- Describe an algorithm using a *failure detector* abstraction [Chandra & Toueg 96]
- By abstracting away time we
  - get simpler algorithms
  - could specify (minimum) conditions under which certain problems are solvable in a simpler way
  - characterize systems based on their ability to implement certain types of failure detectors

Environment Model

- Asynchronous message-passing system
  - fully connected
- Crash failures
- n process $P_1,\ldots,P_n$
- Reliable links between each pair of correct processes
  - Can implement Reliable Broadcast

The Failure Detector Abstraction

- Each process has local failure detector oracle
  - typically outputs list of processes suspected to have crashed at any given time
- In each execution step, a process
  - receives a message (if there is one ready)
  - queries its failure detector oracle
  - makes a transition to a new state
  - may send messages to other processes

Specifying Failure Detectors

- *Accuracy*: which processes are not suspected and when
- *Completeness*: which processes are suspected and when

- Why do we need both?
  - A trivial failure detector will satisfy any accuracy (completeness) property by never (always) suspecting any (all) processes

Completeness

- *Strong Completeness*: Eventually every process that crashes is permanently suspected by *every* correct process
- *Weak Completeness*: Eventually every process that crashes is permanently suspected by *some* correct process

Accuracy

- *Strong Accuracy*: No process is suspected before it crashes
- *Weak Accuracy*: Some correct process is never suspected
- *Eventual Strong Accuracy*: Eventually correct processes are not suspected by any correct process
- *Eventual Weak Accuracy*: Eventually some correct process is never suspected by any correct process

Failure Detector Classes

Completeness	Accuracy: Strong	Accuracy: Weak	Accuracy: Eventual Strong	Accuracy: Eventual Weak
Strong	Perfect P	Strong S	Eventually Perfect ◊P	Eventually Strong ◊S
Weak	Q	Weak W	◊Q	Eventually Weak ◊W

Reducibility of failure detectors

- If a f. d. D' can be implemented using a f. d. D, then
  - D ≥ D', or D' is *reducible* to D, or D' is *weaker* than D
- If D ≥ D' and D'≥ D ➔ D and D' are equiv.

Reducibility of classes of failure detectors

- *C* and *C*' are classes of f. d. and $\forall D \in C$, $\exists D' \in C'$: $D \geq D'$ ➔ $C \geq C'$
  - $C \geq C'$, or *C*' is weaker than *C*
- If $C \geq C'$ and $C' \geq C$ ➔ *C* and *C'* are equiv.
- If $C \geq C'$, then any problem solvable using a $D' \in C'$ is also solvable using some $D \in C$

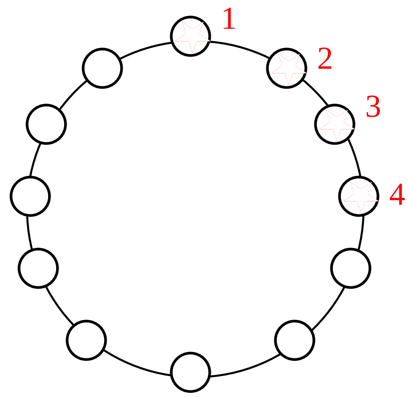
- $P \geq Q$, $S \geq W$, $\Diamond P \geq \Diamond Q$, $\Diamond S \geq \Diamond W$
- $P > S$, $\Diamond P > \Diamond S$, $P > \Diamond P$, $S > \Diamond S$, $P > \Diamond S$

Weak Completeness ➔ Strong Completeness

- Let D be a failure detector satisfying Weak Completeness
- Construct a failure detector D' satisfying Strong Completeness
- Conclusion: $D \geq D'$
- Theorem: $Q \geq P$, $W \geq S$, $\Diamond Q \geq \Diamond P$, $\Diamond W \geq S$
- Corollary: $Q = P$, $W = S$, $\Diamond Q = \Diamond P$, $\Diamond W = \Diamond S$

Solving Consensus using ◊S : Rotating Coordinator Algorithms

Work for up to $f < n/2$ crashes



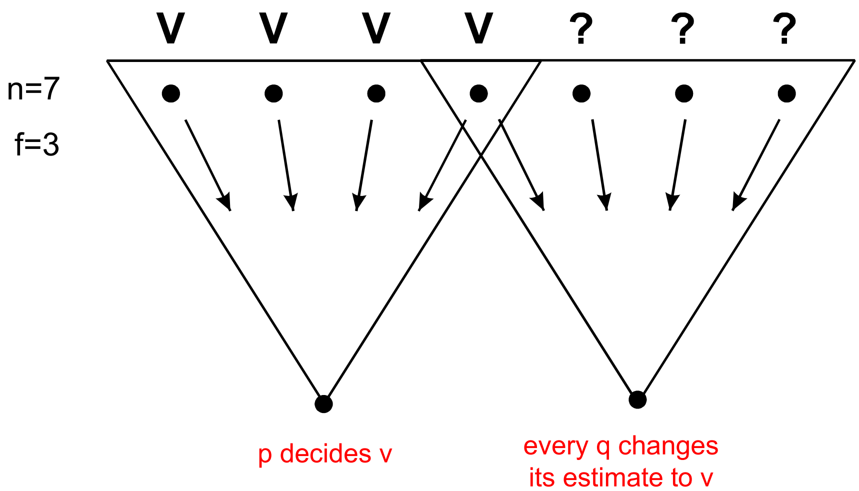
- Processes are numbered 1, 2, …, n
- They execute asynchronous *rounds*
- In round r , the *coordinator* is process (r mod n) + 1
- In round r , the coordinator:
  - tries to impose its estimate as the consensus value
  - succeeds if does not crash and it is not suspected by ◊S

Consensus using ◊S : Phase k; p = k mod n

- q sends (q, k, estimate, $r_{estimate}$) to p
  - p awaits a majority of round k estimates and chooses the estimate EST with the highest $r_{estimate}$
- p broadcasts (p,EST,k)
- q waits to hear (p,EST,k), or FD suspects p
  - if q hears from p, then q responds with (ack, k) and estimate := EST; $r_{estimate}$:= k
  - else q sends (nack, k)
- p awaits a majority of round k ack/nack's
  - If acks, then p R-broadcasts (decide v)

Why does it work?

Agreement:



Agreement

- Lemma 1: Let r be the smallest round such that c = r mod n decides. Let est_c be the estimate sent by c prior to deciding. Then

  For all rounds $r' \geq r$: If c' = r' mod n sends $est_{c'}$, then $est_{c'} = est_c$

Proof: By induction on r'. Trivial for r'=r. Assume for $r \leq r' < k$. Prove for r'=k. Since a majority responds to c with ack, one of the responding processes p sent its estimate to c_k. Hence $ts_p \geq r$. Since c_k selects maximum, it will select the estimate of q with timestamp t, $r \leq t < k$. By the induction hypothesis $est_q = est_c$.

Consensus using ◊S

- Agreement is immediate from Lemma 1
- Termination:
  - No process blocks indefinitely in any one of the wait statements. Why?
  - Eventually, all processes do not suspect a single correct process. Once it becomes a coordinator, everybody decides
- Validity: obvious

Consensus using ◊S

- Theorem: Consensus among $n>2t$ is solvable using ◊S
- Corollary: Consensus among $n>2t$ is solvable using ◊W
  - Why?
  - because ◊W ≥ ◊S
- The bound on the number of processes is tight (partitioning argument)

Implementation with partial synchrony

- Assume Δ is unknown
- Each process p holds a local estimate Δ_p
- Broadcast "I am alive" to all every time unit
- If p does not receive "I am alive" from q during the last Δ_p time units → suspect q
- If p receives a message from q and q is suspected → stop suspecting q, $\Delta_p := \Delta_p + 1$

$\Diamond P$

To which class belongs the failure detector implemented above?