Failure Detectors

But first

- Next class is the last lecture
  - Current Schedule
    - Extensions that help with BFT Agreement

- Other Possibilities
  - Problems beyond agreement
  - Other ways to reason about protocols
  - New programming paradigms

Which do you prefer?

Where we are

RSM → Consensus/Agreement
  → Validity, agreement, termination

Fail-Stop

Asynch
\[ \times \]

Partially Synch
\[ \sqrt[\chi]{\text{if } n > 2f} \]

Sync
\[ \sqrt[\chi]{\text{if } n > f+1} \]

BFT (w/auth.)
\[ \times \]

\[ \sqrt[\chi]{\text{if } n > 3f+1} \]
Why gap? Hand-wavy claim

\[ \rightarrow \text{Partial synchrony allows distinguishing by failure & delay} \]

\[ \rightarrow \text{When?} \]

\[ \rightarrow \text{For how many nodes?} \]

\[ \rightarrow \ldots \]

Can we more precisely define what we need?

Why - Theory: Better understand the protocol

Practice: Maintaining/establishing assumptions come at a cost

\[ \rightarrow n > 2f \]

\[ \rightarrow \text{Partial synchrony} \]

\[ \rightarrow \text{Bounded message latency} \]

\[ \ldots \]

Useful to know what is required.

Failure Detectors
Modeling Notes

- \( F_0 \) Time \( \rightarrow \) Output
  can be anything

In this paper

Output = \( 2^n \): Set of processes suspected of failing

Other work

Output = \( \exists \) Green, Red \( \exists \) (Green \( \rightarrow \) No process is suspected of having failed)

= \( 2^n \): Quorum

= \( 2^n \)/Green/Red (sufficient for NBAC)

Can we more precisely define what we need?

1. Show consensus (or other problem) can be...
Solved if each process had access to F, assuming some failure model.

Chandra and Toueg:

Strong (S) or

\( W(S) \) w/ 2ftl processes suffice

2. Show that if \( \exists F' \) fddo to solve consensus then one can construct F from F'

Chandra, Hadzilacos & Toueg (CHT):

\( W(S) \) is weakest F.D. form

Solving consensus

[Roughly: Use F' to run a consensus protocol that tracks participants]

\( W \rightarrow \exists pETT \) that everyone agrees has not failed

\( \rightarrow \) Leader election?

Chandra & Toueg’s Failure Detectors
$\text{F}(\text{time} \rightarrow \text{set of suspect processes})$

Accuracy: "What correct processes can be suspected"
- **Strong**: No correct process is ever suspected
- **Weak**: $\exists p$ (Correct process $p$) that is never suspected.

- $\Diamond \text{Strong}$: Eventually no correct process is suspected
  $\exists t \text{ s.t. } \forall t' \geq t \Rightarrow \text{F}(t')$ only contains failed processes

- $\Diamond \text{Weak}$: $\exists p$ (Correct process $p$) that is
  eventually not suspected
  $\exists p \in \Pi, t : \forall t' \geq t \Rightarrow p \notin \text{F}(t')$

Completeness: "What faulty processes are suspected"
- **Strong**: Eventually, all failed processes are suspected by all processes
- Weak. Eventually, some correct process suspects each failed process.

Combine these two dimensions to come up with F.D.O.

<table>
<thead>
<tr>
<th>Comp</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strong</td>
<td>Weak</td>
</tr>
<tr>
<td>weak</td>
<td></td>
</tr>
</tbody>
</table>

Eight is too many objects.

Observe for
- Comp & accuracy
  \[ \text{Strong} \implies \text{weak} \]
- Accuracy
  \[ \diamond \text{Strong} \implies \diamond \text{weak} \]

Can build F.D. with strong completeness given F.D. with weak completeness.

◇ Completeness is eventual
Can exchange messages and get FD output from other correct processes

Idea: Strong complete output
- All correct processes eventually suspect all failed processes

\[ \bigcup_{\text{correct}} \text{weak complete output} \]
- A correct process eventually suspects any failed process

② Tricky: Preserving accuracy.
⑥ Strong ← No correct process is suspected

⇒ No processes FD output contains correct processes
⇒ Union does not contain correct p

⑥ Weak ← \exists \text{correct } p \text{ that is never suspected}
- [Same reasoning as above]

② ⊘ strong & ⊘ weak: Tricky
- We suspect and want contain correct
Why? FD output might contain only dead processes (or chosen correct process \( p \)) initially.

Need to get rid of it eventually.

How?

\underline{Idea 1}:

\[ \text{out}[A] = \{ B \} \]

while true:

\[ \text{send} (\langle p, \text{fd output} \rangle) \]

on receive \( \langle p, f \rangle \)

\[ \text{out}[p] \leftarrow f \]

\[ \text{fd output} = \bigcup \text{out}[p] \]

This doesn't work. Why?

\underline{Idea 2}:

\begin{align*}
\text{Every process } p \text{ executes the following:} \\
\text{output}_p & \leftarrow \emptyset \\
\text{cobegin} \\
\text{|| Task 1: repeat forever} \\
\{ p \text{ queries its local failure detector module } \mathcal{D}_p \} \\
\text{suspects}_p & \leftarrow \mathcal{D}_p \\
\text{send} (p, \text{suspects}_p) \text{ to all} \\
\text{|| Task 2: when receive } (q, \text{suspects}_q) \text{ for some } q \\
\text{output}_p & \leftarrow (\text{output}_p \cup \text{suspects}_q) \setminus \{q\} \\
\text{coend}
\end{align*}

Why does this work?
Can focus on one or the other.

Two results:

1. $S \lor W \rightarrow$ Consensus with $f+1$ nodes
2. $\square S \lor \square W \rightarrow$ Consensus with $2f+1$ nodes.

Going to start with 2 ← Similar to what we have seen before.

Core idea:

1. Nodes take it in turn to be leader
   ∴ When a node becomes leader, it tries to replicate a value (proposal) to all correct nodes.
Leader counts how many times its value has been replicated & decides (commits) when the value is sufficiently replicated. Informs nodes about decisions.

2. Other processes move onto another leader if F.D. suspects current leader.

Timing might lead to a scenario where current leader is suspected after value is sufficiently replicated (or potentially committed).

Must ensure that a sufficiently replicated value is used for all future proposals.

→ Compare to Paxos.

P2. If a proposal with value \( v \) is chosen, then every higher-numbered proposal that is chosen has value \( v \).
Every process $p$ executes the following:

```
procedure propose($v_p$)
    $estimate_p ← v_p$  \{estimate$_p$ is $p$'s estimate of the decision value\}
    $state_p ←$ undecided
    $r_p ← 0$  \{$r_p$ is $p$'s current round number\}
    $ts_p ← 0$  \{$ts_p$ is the last round in which $p$ updated $estimate_p$, initially 0\}

{Rotate through coordinators until decision is reached}

while $state_p =$ undecided
    $r_p ← r_p + 1$
    $c_p ← (r_p \mod n) + 1$  \{$c_p$ is the current coordinator\}

Phase 1: {All processes $p$ send $estimate_p$ to the current coordinator}
    send $(p, r_p, estimate_p, ts_p)$ to $c_p$

Phase 2: {The current coordinator gathers $\lceil \frac{n+1}{2} \rceil$ estimates and proposes a new estimate}
    if $p = c_p$ then
        wait until [for $\lceil \frac{n+1}{2} \rceil$ processes $q$ : received $(q, r_p, estimate_q, ts_q)$ from $q$]
        $msg_c[p] ← \{(q, r_p, estimate_q, ts_q) | p$ received $(q, r_p, estimate_q, ts_q)$ from $q$\}
        $t ←$ largest $ts_q$ such that $(q, r_p, estimate_q, ts_q) ∈ msg_c[p]$
        $estimate_c ←$ select one $estimate_q$ such that $(q, r_p, estimate_q, t) ∈ msg_c[p]$  \{selects the largest $ts_q$\}
        send $(p, r_p, estimate_p) \text{ to } c_p$
    else
        \{send $(p, r_p, ack)$ to $c_p$\}

Phase 3: {All processes wait for the new estimate proposed by the current coordinator}
    wait until [received $(c_p, r_p, estimate_{c_p})$ from $c_p$ or $c_p ∈ \mathbb{B}_p$]  \{Query the failure detector\}
    if [received $(c_p, r_p, estimate_{c_p})$ from $c_p$] then
        $estimate_p ← estimate_{c_p}$
        $ts_p ← r_p$
        send $(p, r_p, ack)$ to $c_p$
    else
        send $(p, r_p, nack)$ to $c_p$  \{$p$ suspects that $c_p$ crashed\}

Phase 4: {The current coordinator waits for $\lceil \frac{n+1}{2} \rceil$ replies. If they indicate that $\lceil \frac{n+1}{2} \rceil$ processes adopted its estimate, the coordinator $R$-broadcasts a decide message}
    if $p = c_p$ then
        wait until [for $\lceil \frac{n+1}{2} \rceil$ processes $q$ : received $(q, r_p, ack)$ or $(q, r_p, nack)$]
        if [for $\lceil \frac{n+1}{2} \rceil$ processes $q$ : received $(q, r_p, ack)$] then
            $R$-broadcast($p, r_p, estimate_p, decide$)
    \{If $p$ $R$-delivers a decide message, $p$ decides accordingly\}

when $R$-deliver($q, r_q, estimate_q, decide$)
    if $status_p =$ undecided then
        decide($estimate_q$)
        $state_p ←$ decided
```

Termination? Eventually correct $p$, who no one suspects will become leader.

1. $S\text{onw}$ → Consensus with fit nodes

Why gap? $\exists$ correct process $p$ that is never suspected

→ Have correct process relay
Challenge: Don't know identity of correct p ahead of time

Agreeing on correct p equivalent to consensus

So instead rely on multiple rounds of communication

Every process \( p \) executes the following:

procedure propose(\( V_p \))
\[
V_p = \{1, 2, \ldots, n\} \\
V_p[p] = V_p \\
\Delta_p = V_p
\]

\{p's estimate of the proposed values\}

Phase 1: \{asynchronous rounds \( r_p, 1 \leq r_p \leq n-1 \)\}

for \( r_p \leftarrow 1 \) to \( n-1 \)

send \( (r_p, \Delta_p, p) \) to all

wait until \( [V_q : received (r_p, \Delta_q, q) or q \in \subseteq_p] \)

\( masg_p[p] = \{(r_p, \Delta_q, q) | received (r_p, \Delta_q, q)\} \)

\( \Delta_p = \{1, 2, \ldots, \Delta_q[p]\} \)

for \( k = 1 \) to \( n \)
if \( V_p[k] = \perp \) and \( \exists (r_p, \Delta_q, q) \in masg_p[p] \) with \( \Delta_q[k] \neq \perp \) then

\( V_p[k] = \Delta_q[k] \)

\( \Delta_p[k] = \Delta_q[k] \)

Phase 2: send \( V_p \) to all

wait until \( [V_q : received V_q or q \in \subseteq_p] \)

\( lastmsg_p = \{V_q | received V_q\} \)

for \( k = 1 \) to \( n \)
if \( \exists V_q \in lastmsg_p \) with \( V_q[k] = \perp \) then \( V_p[k] = \perp \)

Phase 3: decide\{first non-\perp component of \( V_p \}\)

Failure Detectors in practice

- Desirable but

- Accuracy is hard

\( \Rightarrow \) Weak: How to ensure that some process \( p \) is never suspected?

\( \Rightarrow \) How to choose \( p \)?

Fig. 5. Solving Consensus using any \( \subseteq \in \mathcal{I} \).
Strong: No correct process is even suspected

Delay vs. failure

Prion work (Falom, others)

Kill suspected nodes

 Likely to violate any assumptions about # of process failures.

Pigeon

Provide information about failure certainty (Warnings vs. facts)

Use additional sources of information to improve predictions