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	Partially Synch	Synch
✗	✓ if $n > 2f$	✓ if $n > f+1$

BFT (w/auth.)

Asynch	Partially Synch	Synch
✗	✓ if $n > 3f+1$	✓ if $n > f+1$

Why gap? Hand-wavy claim

↳ Partial synchrony allows distinguishing b/w failure & delay

↳ When?

→ For how many nodes?

→ ...

Can we more precisely define what we need?

Why – Theory: Better understand the protocol

Practice: Maintaining/establishing assumptions come at a cost

↳ $n > 2f$

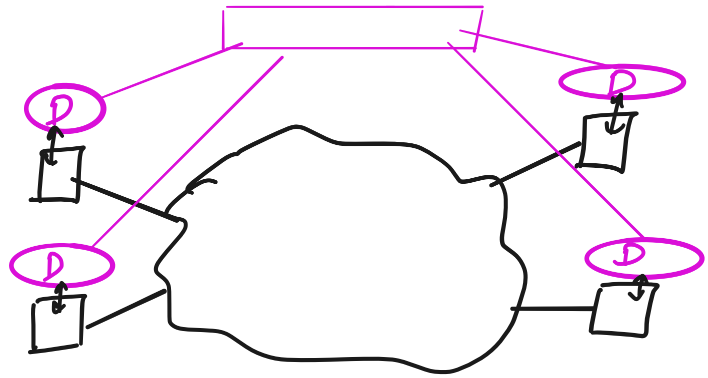
→ Partial synchrony

→ Bounded message latency

...

Useful to know what is required.

Failure Detectors



Modeling Notes

– F: Time $\longrightarrow$ Output

$(\mathbb{Z}^+)$ can be anything

In this paper

$\text{Output} = 2^{\Pi}$: Set of processes suspected of failing

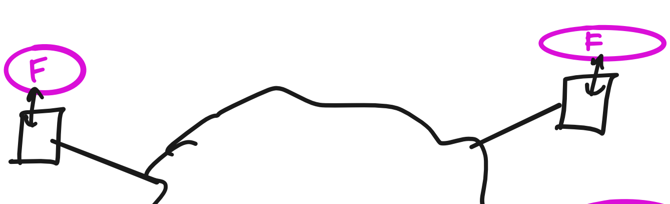
Other work

$\text{Output} = \{$Green, Red$\}$ [Green $\Rightarrow$ No process is suspected of having failed]

$= 2^{\Pi}$: Quorum

$= 2^{\Pi}$ / 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 ($\mathcal{S}$) or

$\Diamond W (\Omega)$ w/ $2f+1$

processes suffice

② Show that if $\exists F'$ f.d.s to solve consensus then one can construct F from F'

Chandra, Hadzilacos & Toueg (CHT):

$\Diamond W (\Omega)$ is weakest F.D. for solving consensus

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

$\Diamond W \rightarrow \exists p \in \Pi$ that everyone agrees has not failed

↳ Leader election?

Chandra & Toueg's Failure Detectors

F: time $\rightarrow$ Set of suspect processes

Accuracy: "What correct processes can be suspected"

- Strong: No correct process is ever suspected
- Weak: $\exists$ Correct process p that is never suspected.
- $\Diamond$ Strong: Eventually no correct process is suspected

  $\exists t$. s.t. $t' \geq t \Rightarrow F(t')$ only contains failed processes

- $\Diamond$ Weak: $\exists$ Correct process p that is eventually not suspected

  $$\exists p \in \Pi, t: \ t' \geq t \Rightarrow p \notin F(t')$$

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 8 F.D.s

	Accuracy			
Comp	Strong	Weak	◇Strong	◇weak
Strong	P			
weak				

Eight is too many objects.

Observe for

- Comp & accuracy

  Strong ⇒ weak

- Accuracy

  ◇strong ⇒ ◇weak

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

① 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]

$$\parallel$$

$\bigcup_{\text{correct } p}$ weak complete output

[a correct process eventually suspects any failed process]

② Tricky: Preserving accuracy.

ⓐ Strong ← No correct process is suspected

⇒ No processes FD output contains correct processes

⇒ Union does not contain correct p

ⓑ Weak ← ∃ correct p that is never suspected

[Same reasoning as above]

ⓒ ◇strong ∧ ◇weak: Trickier

[illegible] FD output might contain correct

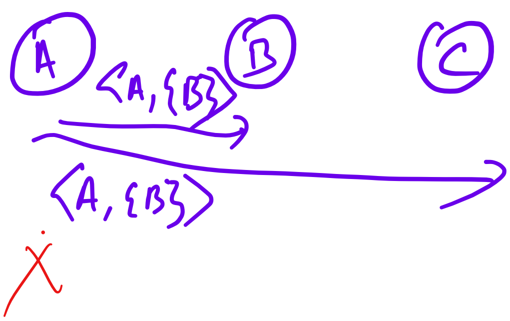
Why? FD output might contain
processes (or chosen correct process
p) initially

Need to get rid of it eventually.

How?

<u>Idea 1</u>

out[A] = {B}




```
while true:
    send (⟨p, fd output⟩)

on receive ⟨p, f⟩
    out[p] ← f

fd output = ∪ out[p]
```

This doesn't work. Why?

<u>Idea 2:</u>

Every process p executes the following:

$output_p \leftarrow \emptyset$

cobegin
|| *Task 1:* **repeat forever**
  {*p queries its local failure detector module* $\mathcal{D}_p$}
  $suspects_p \leftarrow \mathcal{D}_p$
  send $(p, suspects_p)$ to all

|| *Task 2:* **when** receive $(q, suspects_q)$ for some q
  $output_p \leftarrow (output_p \cup suspects_q) - \{q\}$
coend

Why does this work?

Accuracy

Comp	Strong	Weak	$\Diamond$ Strong	$\Diamond$ weak
Strong	P	S	$\Diamond$P	$\Diamond$S
weak	Q	W	$\Diamond$Q	$\Diamond$W

Can focus on one or the other.

Two results

① S (or W) $\longrightarrow$ Consensus with f+1 nodes

② $\Diamond$S (or $\Diamond$W) $\longrightarrow$ Consensus with 2f+1 nodes.

Going to start with ② $\longleftarrow$ 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 (& 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.

Use similar idea ← Quorum Intersection

to decide proposal.

Every process p executes the following:

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

  {*Rotate through coordinators until decision is reached*}

  while $state_p = undecided$
    $r_p \leftarrow r_p + 1$
    $c_p \leftarrow (r_p \bmod n) + 1$ {*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]
        $msgs_p[r_p] \leftarrow \{(q, r_p, estimate_q, ts_q) \mid p$ received $(q, r_p, estimate_q, ts_q)$ from $q\}$
        $t \leftarrow$ largest ts_q such that $(q, r_p, estimate_q, ts_q) \in msgs_p[r_p]$
        $estimate_p \leftarrow$ select one $estimate_q$ such that $(q, r_p, estimate_q, t) \in msgs_p[r_p]$
        send $(p, r_p, estimate_p)$ to all

Phase 1

    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** $c_p \in \mathcal{D}_p$]{*Query the failure detector*}
      if [received $(c_p, r_p, estimate_{c_p})$ from c_p] **then** {*p received $estimate_{c_p}$ from c_p*}
        $estimate_p \leftarrow estimate_{c_p}$
        $ts_p \leftarrow r_p$
        send (p, r_p, ack) to c_p
      else send $(p, r_p, nack)$ to c_p {*p suspects that c_p crashed*}

Phase 2

    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$(p, r_p, estimate_p, decide)$

Count & Commit.

{*If p R-delivers a decide message, p decides accordingly*}

when *R-deliver*$(q, r_q, estimate_q, decide)$
  if $state_p = undecided$ **then**
    decide$(estimate_q)$
    $state_p \leftarrow decided$

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

① S (or W) → Consensus with f+1 nodes

Why gap: ∃ correct process p that is never suspected

↳ Have correct process relay

information from other processes

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

Phase 1: {*asynchronous rounds* r_p, $1 \leq r_p \leq n-1$}
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*}
$msgs_p[r_p] \leftarrow \{(r_p, \Delta_q, q) \mid$ received $(r_p, \Delta_q, q)\}$
$\Delta_p \leftarrow \langle \perp, \perp, \ldots, \perp \rangle$
for $k \leftarrow 1$ **to** n
if $V_p[k] = \perp$ **and** $\exists (r_p, \Delta_q, q) \in msgs_p[r_p]$ with $\Delta_q[k] \neq \perp$ **then**
$V_p[k] \leftarrow \Delta_q[k]$
$\Delta_p[k] \leftarrow \Delta_q[k]$

Gather & Relay Proposals

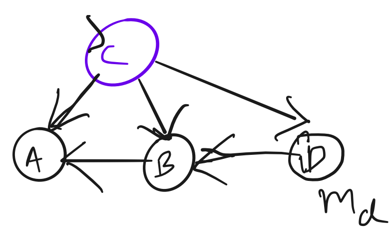
Phase 2: send V_p to all
wait until $[\forall q:$ received V_q **or** $q \in \mathcal{D}_p]$ {*query the failure detector*}
$lastmsgs_p \leftarrow \{V_q \mid$ received $V_q\}$
for $k \leftarrow 1$ **to** n
if $\exists V_q \in lastmsgs_p$ with $V_q[k] = \perp$ **then** $V_p[k] \leftarrow \perp$

Send vector of all proposals

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

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

Failure Detectors in practice



- Desirable but
  - Accuracy is hard

    ↳ Weak: How to ensure that some process p is never suspected?

    ↳ How to choose p?

- Strong: No correct process is ever suspected
  - Delay vs failure
  - Prior work (Falcon, 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] Core Benefit!