Failure Detectors

Poster session: Next week, here

- No need to print poster: Just show it on your laptop, etc.
- Ideally: 1 slide/screen (WILL NOT LOOK AT more than 2)
  - What were you doing
  - What was the best/coolest thing you found.
  - THINGS YOU WANT YOUR CLASSMATES TO KNOW
- MAIN AUDIENCE: OTHERS IN THE CLASS!

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.)

✗	✓ 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

↳ $n > 2f$

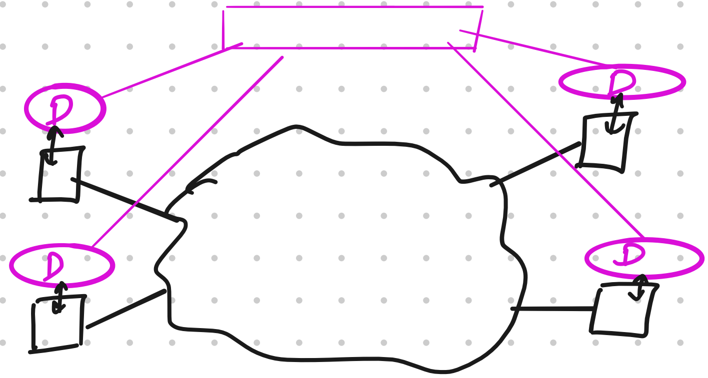
→ Partial synchrony

→ 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

Output = 2^Π : Set of processes suspected of failing

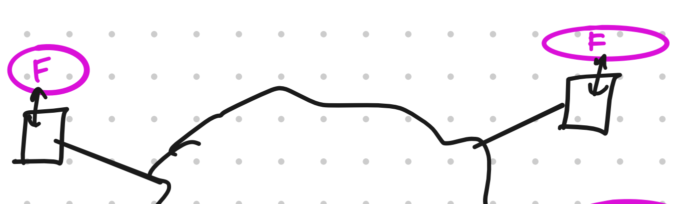
Other work

Output = {Green, Red} [Green ⇒ No process is suspected of having failed]

= 2^Π : Quorum

= 2^Π / 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

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

processes suffice

(2) Show that if $\exists F'$ f.d. 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 → 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

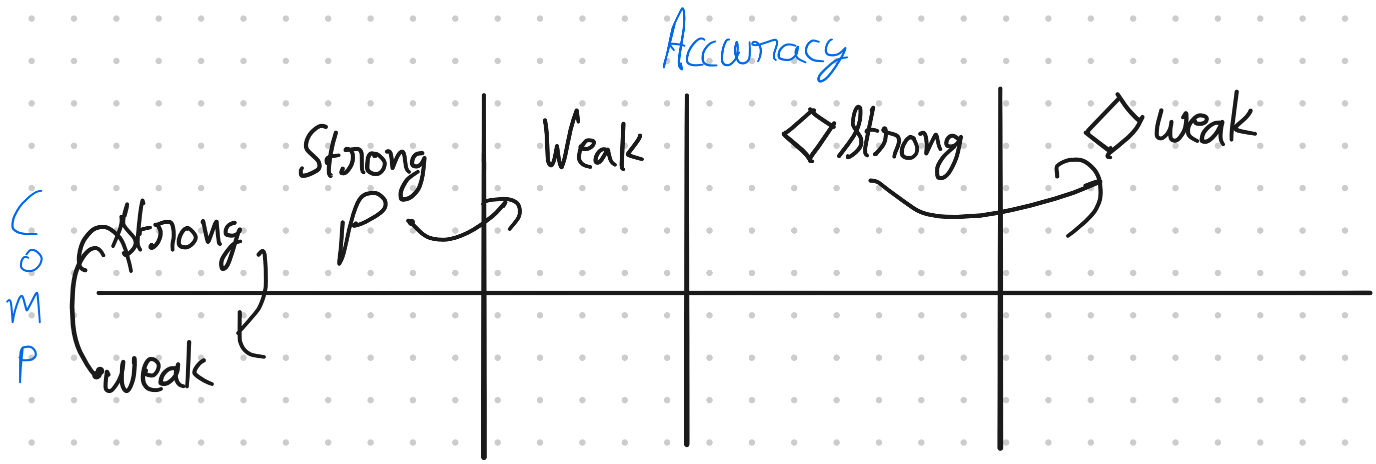
$$\exists\, p \in \Pi,\ t:\ t' \geq t \Rightarrow p \notin F(t')$$

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



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]

||

$\bigcup_{\text{correct } p}$ 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.

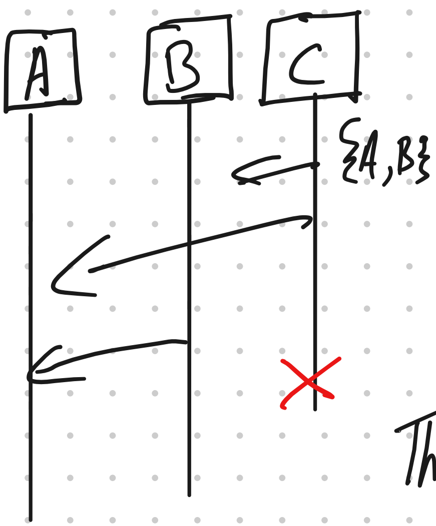
How?

<u>Idea 1</u>

while true:
    send ($\langle p,$ fd output$\rangle$)

on receive $\langle p, f \rangle$
    out $[p] \leftarrow f$

fd output $= \bigcup$ 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

Send FD output

|| *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) ⟶ Consensus with f+1 nodes

② $\Diamond$S (or $\Diamond$W) ⟶ 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 (& 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 ← ⟨⊥, ⊥, ..., ⊥⟩                                    {p's estimate of the proposed values}
  V_p[p] ← v_p
  Δ_p ← V_p

  Phase 1: {asynchronous rounds r_p, 1 ≤ r_p ≤ n − 1}
        for r_p ← 1 to n − 1
          send (r_p, Δ_p, p) to all
          wait until [∀q : received (r_p, Δ_q, q) or q ∈ 𝒟_p]      {query the failure detector}
          msgs_p[r_p] ← {(r_p, Δ_q, q) | received (r_p, Δ_q, q)}
          Δ_p ← ⟨⊥, ⊥, ..., ⊥⟩
          for k ← 1 to n
            if V_p[k] = ⊥ and ∃(r_p, Δ_q, q) ∈ msgs_p[r_p] with Δ_q[k] ≠ ⊥ then
                V_p[k] ← Δ_q[k]
                Δ_p[k] ← Δ_q[k]
```

Gather & Relay Proposals

```
  Phase 2: send V_p to all
        wait until [∀q : received V_q or q ∈ 𝒟_p]                  {query the failure detector}
        lastmsgs_p ← {V_q | received V_q}
        for k ← 1 to n
          if ∃V_q ∈ lastmsgs_p with V_q[k] = ⊥ then V_p[k] ← ⊥
```

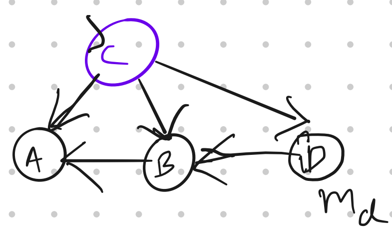
Send vector of all proposals

```
  Phase 3: decide( first non-⊥ 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.

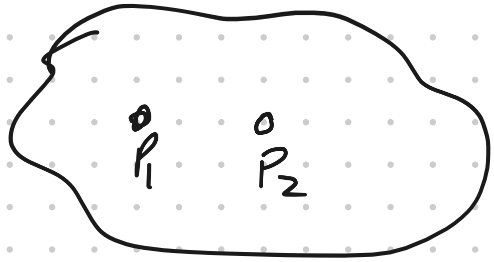
Pigeon

→ Provide information about failure certainty (Warnings vs facts)

→ Use additional sources of information to improve predictions] Core Benefit!

Strong Accuracy Weak Complete

A



Output

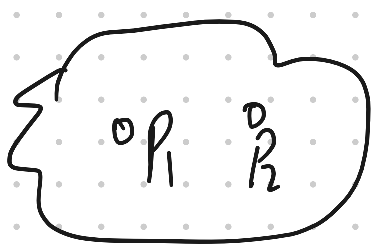
$\hookrightarrow$ $fd(x) = \{p_1, p_2\}$

- Can p_1 be correct?
- Can p_1 have failed?

$SA/WC \xrightarrow{\text{Union}} SA/SC$

WA WC [Sp. Process c]

A



1. Can p_1 be correct?

Yes

2. Can p_1 be failed? Yes

(2) Can P_1 be faulty?

(3) Can $P_1 = C$?

$\Diamond SA \quad WC \longrightarrow \Diamond SA \quad SC$

A: $\{ \circ P_1 \quad \circ P_2 \}$

Q1: Can P_1 be correct?

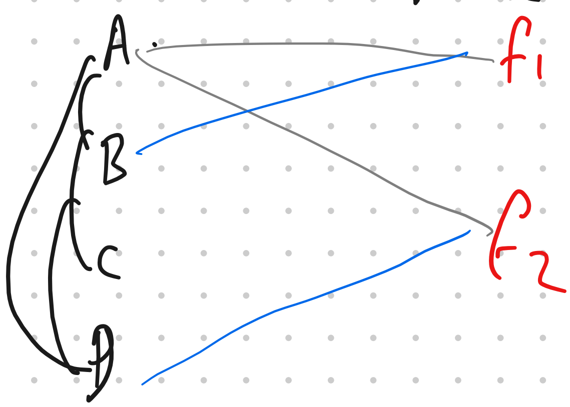
Q2: Can P_1 be faulty?

No correct process is suspected

Do not know

Weak Complete

Correct Failed



Eventually

Strong Complete