What is a Failure Detector?

- FD outputs set of processes that it currently suspects to have crashed
  - Each process is associated with a FD, and the set of crashed processes may be different for each process
  - Assume a fail-stop model

- Simple implementations
  - Ping test: send a ping; wait for ack
  - Heartbeat: each process sends “I am alive” msg

- FD is a distributed oracle that provides hints about the operational status of a process
  - Hints may be incorrect, and may be updated over time

Unreliable FD can be used to solve consensus problem
Classification of FD

- **Completeness**
  - There is a time after which every crashed process is permanently suspected by a correct process

- **Strong Completeness**
  - Eventually every crashed process is suspected by every correct process

- **Weak Completeness**
  - Eventually every crashed process is suspected by some (at least one) correct process
Classification of FD: Completeness

Strong Completeness

Weak Completeness
Classification of FD

- Accuracy
  - There is a time after which a correct process is never suspected by any correct process

- Perpetual Accuracy
  - Strong Accuracy: correct processes are never suspected by any correct process
  - Weak Accuracy: some correct process is never suspected by any correct process

- Eventual 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.
## Classification of FD

<table>
<thead>
<tr>
<th>Completeness</th>
<th>Accuracy</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Strong</td>
<td>Perfect $\mathcal{P}$</td>
<td>Strong $\mathcal{S}$</td>
<td>Eventually Perfect $\diamond \mathcal{P}$</td>
</tr>
<tr>
<td>Weak</td>
<td>$\mathcal{Q}$</td>
<td>$\mathcal{W}$</td>
<td>$\diamond \mathcal{Q}$</td>
</tr>
</tbody>
</table>

8 possible classes of Failure Detectors
Reducibility of FDs

- A Failure detector $D$ is reducible to another failure detector $D'$ if there exist a reduction algorithm $T_{D \rightarrow D'}$ that transforms $D$ to $D'$
  - $D'$ is Weaker than $D$ (i.e.) $D \subseteq D'$, or
  - $D$ is stronger than $D'$ (i.e.) $D \geq D'$

If FD $D$ is available, but the algorithm $A$ requires $D'$, then reduce from $D$ to $D'$
Every process $p$ executes the following:

$$output_p \leftarrow \emptyset$$

cobegin

|| Task 1: repeat forever

$$\{ p \text{ queries its local failure detector module } D_p \}$$
$$suspects_p \leftarrow D_p$$
$$\text{send } (p, suspects_p) \text{ to all}$$

|| Task 2: when receive $(q, suspects_q)$ for some $q$

$$output_p \leftarrow (output_p \cup suspects_q) - \{q\} \quad \{output_p \text{ emulates } D'_p\}$$

coend

Fig. 3. $T_{D \rightarrow D'}$: From Weak Completeness to Strong Completeness.

The reduction strengthens completeness while preserving accuracy $\Rightarrow$ can focus only on 4 classes of FD.
Reductions

• All weakly complete failure detectors are reducible to strongly complete failure detectors
  \[ P \geq V, \ldots \]

• All strongly complete failure detectors are reducible to weakly complete failure detectors

• Weakly and Strongly complete FDs are equivalent
Comparing the FDs

- Perfect (P) is reducible to Strong (S)
- Eventually perfect reducible to eventually strong

Important Classes of FD:
- Strong failure detectors (S)
- Eventually strong failure detectors (◊S)
Solving Consensus with S

• S: Strong Completeness, Weak Accuracy
  – At least some correct process is never suspected

• Three phases in the algorithm
  – Phase 1:
    • Executes n-1 asynchronous rounds
      – Broadcast their proposed value
      – Wait for message from other processes not in \( D_p \)
      – Process does not wait for q if q gets added to \( D_p \)
      – Update a missing value
  – Phase 2:
    • broadcast estimate of proposed values
    • Wait for estimate from other processes
    • In the end, correct processes agree on a vector based on proposed values
  – Phase 3:
    • Decide on the first non-trivial component of the agreed vector

• The solution tolerates any number of process failures
Every process \( p \) executes the following:

**procedure** propose\((i_p)\)

\( V_p \leftarrow \langle \bot, \bot, \ldots, \bot \rangle \) \( \{ \text{p's estimate of the proposed values} \} \)
\( V_p[p] \leftarrow i_p; \)
\( \Delta_p \leftarrow V_p \)

**Phase 1:** \( \{ \text{Execute round } r_p, 1 \leq r_p \leq n - 1 \} \)

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

\( p \) sends \((r_p, \Delta_p, p)\) to all other processes

wait until \([\forall q: \text{received } (r_p, \Delta_q, q) \text{ or } q \in D_p]\) \( \{ \text{query the failure detector} \} \)

\( msg_s_p[r_p] \leftarrow \{(r_p, \Delta_q, q) | \text{received}(r_p, \Delta_q, q)\} \)
\( \Delta_p \leftarrow \langle \bot, \bot, \ldots, \bot \rangle \)

for \( k \leftarrow 1 \) to \( n \)

if \((V_p[k] = \bot \text{ and } \exists (r_p, \Delta_q, q) \in msg_s_p(r_p) \text{ with } \Delta_q[k] \neq \bot)\)
then \( V_p[k] \leftarrow \Delta_q[k] \)
\( \Delta_p[k] \leftarrow \Delta_q[k] \)

**Phase 2:** \( p \) sends \( V_p \) to all processes

wait until \([\forall q: \text{received } V_q \text{ or } q \in D_p]\) \( \{ \text{query the failure detector} \} \)

\( lastmsgs_p \leftarrow \{V_q | \text{received } V_q\} \)

for \( k \leftarrow 1 \) to \( n \)

if \( \exists V_q \in lastmsgs_p \text{ with } V_q[k] = \bot \)
then \( V_p[k] \leftarrow \bot \)

**Phase 3:** decide on the first non-\( \bot \) element of \( V_p \)
Solving Consensus with <>S

• All correct processes may be added to lists of suspects at any time; but after a time a correct process is permanently removed from suspect list
  – At any time, processes cannot determine if a process is correct, or if a correct process will not be suspected after time t

• Needs $f < \frac{n}{2}$ ➞ any process must hear from a majority
Solving Consensus with $<>S$

- Uses a rotating coordinator paradigm
- Processes go through a sequence of asynchronous rounds, divided into 4 phases
- Phases (1 – 4)
  - All processes send (round, preference, timestamp) to the coordinator
  - Coordinator waits for majority $\Rightarrow$ sets its own estimate with largest timestamp message $\Rightarrow$ sends (round, estimate) to all processes
  - Each process waits for new proposal, or the FD to suspect coordinator
    - If estimate received, then adopt it and send ack to coordinator
    - Else, send a NACK to coordinator
  - Coordinator waits for ACK/NACK from majority $\Rightarrow$ if majority ACK received, then announce current estimate as the decision