Unreliable Failure Detectors for Reliable Distributed Systems
A different approach

- Augment the asynchronous model with an unreliable failure detector for crash failures
- Define failure detectors in terms of abstract properties, not specific implementations
- Identify classes of failure detectors that allow to solve Consensus
The Model

General
- asynchronous system
- processes fail by crashing
- a failed process does not recover

Failure Detectors
- outputs set of processes that it currently suspects to have crashed
- the set may be different for different 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 correct process is ever suspected

Weak Accuracy
Some correct process is never suspected
Accuracy

Strong Accuracy
No correct process is ever suspected

Weak Accuracy
Some correct process is never suspected

Eventual Strong Accuracy
There is a time after which no correct process is ever suspected

Eventual Weak Accuracy
There is a time after which some correct process is never suspected
# Failure detectors

<table>
<thead>
<tr>
<th>Completeness</th>
<th>Strong</th>
<th>Weak</th>
<th>Eventual strong</th>
<th>Eventual weak</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strong</td>
<td>Perfect $P$</td>
<td>Strong $S$</td>
<td>$\Diamond P$</td>
<td>$\Diamond S$</td>
</tr>
<tr>
<td>Weak</td>
<td>Quasi $Q$</td>
<td>Weak $W$</td>
<td>$\Diamond Q$</td>
<td>$\Diamond W$</td>
</tr>
</tbody>
</table>
Reducibility

$T_{D \rightarrow D'}$ transforms failure detector $D$ into failure detector $D'$

If we can transform $D$ into $D'$ then we say that $D$ is stronger than $D'$ and that $D'$ is reducible to $D$

If $D \geq D'$ and $D' \geq D$ then we say that $D$ and $D'$ are equivalent:

$D \equiv D'$
Simplify, Simplify!

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

\[ P \geq Q, \quad S \geq W, \quad \diamond P \geq \diamond Q, \quad \diamond S \geq \diamond W \]
Simplify, Simplify!

- All weakly complete failure detectors are reducible to strongly complete failure detectors
  \[ P \geq Q, \quad S \geq W, \quad \lozenge P \geq \lozenge Q, \quad \lozenge S \geq \lozenge W \]

- All strongly complete failure detectors are reducible to weakly complete failure detectors (!)
  \[ Q \geq P, \quad W \geq S, \quad \lozenge Q \geq \lozenge P, \quad \lozenge W \geq \lozenge S \]

Weakly and strongly complete failure detectors are equivalent!
From Weak Completeness to Strong Completeness

Every process $p$ executes the following:

$$\text{output}_p := 0$$

cobegin

Task 1: repeat forever

- $\{p \text{ queries its local failure detector module } D_p\}$
- $\text{suspects}_p := D_p$
- send $(p, \text{suspects}_p)$ to all

Task 2: when receive $(q, \text{suspects}_q)$ from some $q$

- $\text{output}_p := (\text{output}_p \cup \text{suspects}_p) - \{q\}$

coend
The Theorems

**Theorem 1** In an asynchronous system with $W$, consensus can be solved as long as $f \leq n - 1$
The Theorems

Theorem 1 In an asynchronous system with $W$, consensus can be solved as long as $f \leq n - 1$

Theorem 2 There is no $f$-resilient consensus protocol using $\lozenge P$ for $f \geq n/2$
The Theorems

Theorem 1  In an asynchronous system with $W$, consensus can be solved as long as $f \leq n - 1$

Theorem 2  There is no $f$-resilient consensus protocol using $\diamond P$ for $f \geq n/2$

Theorem 3  In asynchronous systems in which Processes can use $\diamond W$, consensus can be solved as long as $f < n/2$
The Theorems

**Theorem 1** In an asynchronous system with $W$, consensus can be solved as long as $f \leq n - 1$

**Theorem 2** There is no $f$-resilient consensus protocol using $\Diamond P$ for $f \geq n/2$

**Theorem 3** In asynchronous systems in which processes can use $\Diamond W$, consensus can be solved as long as $f < n/2$

**Theorem 4** A failure detector can solve consensus only if it satisfies weak completeness and eventual weak accuracy—i.e. $\Diamond W$ is the weakest failure detector that can solve consensus.
Solving consensus using $S$

$S$: Strong Completeness, Weak Accuracy

- at least some correct process $c$ is never suspected

- Each process $p$ has its own failure detector

- Input values are chosen from the set $\{0, 1\}$
Notation

We introduce the operators $\oplus$, $\star$, $\otimes$

They operate element-wise on vectors whose entries have values from the set $\{0, 1, \bot\}$

- $v \star \bot = v$  $\bot \star v = v$
- $v \star v = \bot$  $\bot \star \bot = \bot$
- $v \otimes \bot = v$  $\bot \otimes v = v$
- $v \oplus v = v$  $\bot \oplus \bot = \bot$
- $v \otimes \bot = \bot$  $\bot \otimes v = \bot$
- $v \otimes v = v$  $\bot \otimes \bot = \bot$

Given two vectors $A$ and $B$, we write $A \leq B$ if $A[i] \neq \bot$ implies $B[i] \neq \bot$
Solving Consensus using any $D \in S$

1: $V_p := (\bot, \ldots, \bot, v_p, \bot, \ldots, \bot)$ \hspace{2cm} \{p’s estimate of the proposed values\}
2: $\Delta_p := (\bot, \ldots, \bot, v_p, \bot, \ldots, \bot)$ \hspace{2cm} \{asynchronous rounds $r_p$, $1 \leq r_p \leq n - 1$\}
3: {phase 1}
4: \textbf{for} $r_p := 1$ \textbf{to} $n - 1$
5: \hspace{1cm} \textbf{send} $(r_p, \Delta_p, p)$ \textbf{to} all
6: \hspace{1cm} \textbf{wait} until $[\forall q : \text{received} (r_p, \Delta_q, q) \text{ or } q \in D_p]$ \hspace{0.5cm} \{query the failure detector\}
7: $O_p := V_p$
8: $V_p := V_p \oplus (\oplus_q \text{ received } \Delta_q)$
9: $\Delta_p := V_p \star O_p$ \hspace{2cm} \{value is only echoed the first time it is seen\}
10: {phase 2}
11: \textbf{send} $(r_p, V_p, p)$ \textbf{to} all
12: \textbf{wait} until $[\forall q : \text{received} (r_p, V_q, q) \text{ or } q \in D_p]$ \hspace{2cm}
13: $V_p := \otimes_q \text{ received } V_q$ \hspace{2cm} \{computes the "intersection", including $V_p$\}
14: {phase 3}
15: \textbf{decide} on leftmost non-$\bot$ coordinate of $V_p$
A useful Lemma

1: \( V_p := (\bot, \ldots, \bot, v_p, \bot, \ldots, \bot) \)
   \{p's estimate of the proposed values\}
2: \( \Delta_p := (\bot, \ldots, \bot, v_p, \bot, \ldots, \bot) \)
3: \{phase 1\}
   \{asynchronous rounds \( r_p \) is \( r_p \leq n - 1 \)\}
4: \textbf{for} \( r_p := 1 \) \textbf{to} \( n-1 \)
5: \textbf{send} \((r_p, \Delta_p, p)\) \textbf{to} all
6: \textbf{wait until} \[ \forall q: \text{received} \((r_p, \Delta_q, q)\) \textbf{or} \ q \in D_p \]
7: \( O_p := V_p \)
8: \( V_p := V_p \oplus (\oplus q \text{ received } \Delta_q) \)
9: \( \Delta_p := V_p \otimes O_p \) \{value is only echoed first time it is seen\}
10: \{phase 2\}
11: \textbf{send} \((r_p, V_p, p)\) \textbf{to} all
12: \textbf{wait until} \[ \forall q: \text{received} \((r_p, V_q, q)\) \textbf{or} \ q \in D_p \]
13: \( V_p := \oplus q \text{ received } V_q \) \{computes the "intersection", including \( V_p \)\}
14: \{phase 3\}
15: decide on leftmost non- \( \bot \) coordinate of \( V_p \)

Lemma 1 After phase 1 is complete, \( V_c \leq V_p \) for all processes \( p \) that complete phase 1
A useful Lemma

Let $V_c$ be the first round when $c$ sees $v_i$.

**Lemma 1** After phase 1 is complete, $V_c \leq V_p$ for all processes $p$ that complete phase 1.

**Proof** We show that $V_c[i] = v_i \land v_i \neq \bot \Rightarrow \forall p : V_p[i] = v_i$.

Let $r$ be the first round when $c$ sees $v_i$.

- $r \leq n-2$
  - will send to all $\Delta_c$ with $v_i$ in round $r$.
- By weak accuracy, all correct processes receive $v_i$ by next round.

$r = n-1$
- $v_i$ has been forwarded $n-1$ times: every other process has seen $v_i$. 

```
1: \(V_p := (\bot, \ldots, \bot, v_p, \bot, \ldots, \bot)\)  
   \{p's estimate of the proposed values\}
2: \(\Delta_p := (\bot, \ldots, \bot, v_p, \bot, \ldots, \bot)\)
3: \{phase 1\}
   \{asynchronous rounds \(1 \leq r_p \leq n - 1\)\}
4: for \(r_p := 1\) to \(n-1\)
5: \(\text{send} (r_p, \Delta_p, p)\) to all
6: \(\text{wait until} \ [\forall q: \text{received} (r_p, \Delta_q, q) \text{ or } q \in D_p]\)
7: \(O_p := V_p\)
8: \(V_p := V_p \oplus (\oplus q \text{ received } \Delta_q)\)
9: \(\Delta_p := V_p \otimes O_p\)  \{value is only echoed first time it is seen\}
10: \{phase 2\}
11: \(\text{send} (r, V_p, p)\) to all
12: \(\text{wait until} \ [\forall q: \text{received} (r, V_q, q) \text{ or } q \in D_p]\)
13: \(V_p := \oplus q \text{ received } V_q\) \{computes the "intersection", including $V_p$\}
14: \{phase 3\}
15: \(\text{decide on leftmost non-} \bot \text{ coordinate of } V_p\)
```
Two additional cool lemmas

1: \( V_p := (\perp, \ldots, \perp, v_p, \perp, \ldots, \perp) \)
   \{p's estimate of the proposed values\}
2: \( \Delta_p := (\perp, \ldots, \perp, v_p, \perp, \ldots, \perp) \)
3: \{phase 1\}
   \{asynchronous rounds \( r_p \) \( 1 \leq r_p \leq n - 1 \)\}
4: \textbf{for} \( r_p := 1 \) \textbf{to} \( n-1 \)
5: \textbf{send} \( (r_p, \Delta_p, p) \) \textbf{to all}
6: \textbf{wait until} \[ \forall q: \text{received} (r_p, \Delta_q, q) \text{ or } q \in D_p \]
7: \( O_p := V_p \)
8: \( V_p := V_p \oplus (\oplus q \text{ received } \Delta_q) \)
9: \( \Delta_p := V_p \otimes O_p \) \{value is only echoed first time it is seen\}
10: \{phase 2\}
11: \textbf{send} \( (r_p, V_p, p) \) \textbf{to all}
12: \textbf{wait until} \[ \forall q: \text{received} (r_p, V_q, q) \text{ or } q \in D_p \]
13: \( V_p := \oplus q \text{ received } V_q \) \{computes the "intersection", including \( V_p \)\}
14: \{phase 3\}
15: decide on leftmost non- \( \perp \) coordinate of \( V_p \)

\textbf{Lemma 2} After phase 2 is complete, \( V_c = V_p \) for each \( p \) that completes phase 2

\textbf{Proof}

\( \text{All processes that completed phase 2 have received } V_c. \) Since \( V_c \) is the smallest \( V \text{-vector}, \)
\[ V_c[i] \neq \perp \Rightarrow V_p[i] \neq \perp \ \forall p \]

\( \text{By the definition of } \otimes \)
\[ V_c[i] = \perp \Rightarrow V_p[i] = \perp \ \forall p \]

\textbf{after phase 2} \vspace{1cm}

\textbf{Lemma 3} \( V_c \neq (\perp, \perp, \perp, \ldots, \perp) \)
Solving consensus

Theorem The protocol to the left satisfies Validity, Agreement, and Termination

Proof Left as an exercise

1: $V_p := (⊥, ..., ⊥, v_p, ⊥, ..., ⊥)$  
   {p's estimate of the proposed values}
2: $Δ_p := (⊥, ..., ⊥, v_p, ⊥, ..., ⊥)$
3: {phase 1}
   {asynchronous rounds $r_p, 1 ≤ r_p ≤ n - 1$}
4:   for $r_p := 1$ to $n-1$
5:     send $(r_p, Δ_p, p)$ to all
6:     wait until $[∀q: received (r_p, Δ_q, q) \text{ or } q ∈ D_p]$  
7:     $O_p := V_p$
8:     $V_p := V_p ⊕ (∇q \text{ received } Δ_q)$
9:     $Δ_p := V_p ⊕ O_p$  
   {value is only echoed first time it is seen}
10: {phase 2}
11:   send $(r_p, V_p, p)$ to all
12:   wait until $[∀q: received (r_p, V_q, q) \text{ or } q ∈ D_p]$  
13: $V_p := (∇q \text{ received } V_q$  
   {computes the "intersection", including $V_p$}
14: {phase 3}
15:   decide on leftmost non- ⊥ coordinate of $V_p$
A lower bound – I

Theorem  Consensus with $\Diamond P$ requires $f < n/2$
A lower bound – I

Theorem  Consensus with $\diamond P$ requires $f < n/2$

Proof

- Suppose $n$ is even, and a protocol exists that solves consensus when $f = n/2$
- Divide the set of processes in two sets of size $n/2$, $P_1$ and $P_2$
A lower bound – II

Consider three executions:

\[ P_1 \leftarrow 0; \ P_2 \leftarrow 0 \]

All processes in \( P_2 \) crash before they can propose

Detectors work perfectly
A lower bound – II

Consider three executions:

\[
P_1 \leftarrow 0; P_2 \leftarrow 0
\]
All processes in \( P_2 \) crash before they can propose

Detectors work perfectly

\[
P_1 \text{ decides } 0
\]

after \( t_1 \)
A lower bound – II

Consider three executions:

- $P_1 ← 0; P_2 ← 0$
  - All processes in $P_2$ crash before they can propose
  - Detectors work perfectly
  - $P_1$ decides 0 after $t_1$

- $P_1 ← 1; P_2 ← 1$
  - All processes in $P_1$ crash before they can propose
  - Detectors work perfectly
A lower bound – II

Consider three executions:

\[ P_1 \leftarrow 0; P_2 \leftarrow 0 \]
All processes in \( P_2 \) crash before they can propose
Detectors work perfectly

\( P_1 \) decides 0
after \( t_1 \)

\[ P_1 \leftarrow 1; P_2 \leftarrow 1 \]
All processes in \( P_1 \) crash before they can propose
Detectors work perfectly

\( P_2 \) decides 1
after \( t_2 \)
A lower bound - II

Consider three executions:

1. $P_1 \leftarrow 0; P_2 \leftarrow 0$
   - All processes in $P_2$ crash before they can propose
   - Detectors work perfectly

2. $P_1 \leftarrow 0; P_2 \leftarrow 1$
   - No process crashes
   - Detectors make mistakes:
     until $\max(t_1, t_2)$, $P_1$ believes $P_2$ crashed, and vice versa

3. $P_1 \leftarrow 1; P_2 \leftarrow 1$
   - All processes in $P_1$ crash before they can propose
   - Detectors work perfectly

$P_1$ decides 0 after $t_1$

$P_2$ decides 1 after $t_2$
A lower bound - II

Consider three executions:

1. $P_1 \leftarrow 0; P_2 \leftarrow 0$
   - All processes in $P_2$ crash before they can propose
   - Detectors work perfectly
   - $P_1$ decides 0 after $t_1$

2. $P_1 \leftarrow 0; P_2 \leftarrow 1$
   - No process crashes
   - Detectors make mistakes:
     until $\max(t_1, t_2)$, $P_1$ believes $P_2$ crashed, and vice versa
   - $P_1$ decides 0
   - $P_2$ decides 1

3. $P_1 \leftarrow 1; P_2 \leftarrow 1$
   - All processes in $P_1$ crash before they can propose
   - Detectors work perfectly
   - $P_2$ decides 1 after $t_2$
The case of the Rotating Coordinator

Solving consensus with $\diamond W$ (actually, $\diamond S$)

- Asynchronous rounds
- In round $r$, only messages timestamped $r$ are sent and processed (except for DECIDE messages)
- Each process $p$ has an opinion $v_p \in \{0, 1\}$
- Each opinion has a time of adoption $t_p$ (initially, $t_p = 0$)
- Each round has a coordinator $c$ such that $c_{id} = (r \mod n) + 1$
One round, four phases

Phase 1
Each process, including $c$, sends its opinion timestamped $r$ to $c$
One round, four phases

Phase 1
Each process, including $c$, sends its opinion timestamped $r$ to $c$

Phase 2
$c$ waits for first $\lfloor n/2 + 1 \rfloor$ opinions with timestamp $r$
c selects $v$, one of the most recently adopted opinions
$v$ becomes $c$’s suggestion for round $r$
c sends its suggestion to all
One round, four phases

Phase 1
Each process, including $c$, sends its opinion timestamped $r$ to $c$

Phase 2
$c$ waits for first $|n/2 + 1|$ opinions with timestamp $r$
$c$ selects $v$, one of the most recently adopted opinions
$v$ becomes $c$’s suggestion for round $r$
$c$ sends its suggestion to all

Phase 3
Each $p$ waits for a suggestion, or for failure detector to signal $c$ is faulty
If a suggestion is received, it is adopted: $v_p := v ; t_p := r ;$ ACK to $c$
Otherwise, NACK to $c$
One round, four phases

Phase 1
Each process, including \( c \), sends its opinion timestamped \( r \) to \( c \)

Phase 2
\( c \) waits for first \( \lfloor n/2 + 1 \rfloor \) opinions with timestamp \( r \)
\( c \) selects \( v \), one of the most recently adopted opinions
\( v \) becomes \( c \)'s suggestion for round \( r \)
\( c \) sends its suggestion to all

Phase 3
Each \( p \) waits for a suggestion, or for failure detector to signal \( c \) is faulty
If a suggestion is received, it is adopted: \( v_p := v ; t_p := r ; \) ACK to \( c \)
Otherwise, NACK to \( c \)

Phase 4
\( c \) waits for first \( \lfloor n/2 + 1 \rfloor \) responses
if all ACKs, then \( c \) decides on \( v \) and sends DECIDE to all
if \( p \) receives DECIDE, then \( p \) decides on \( v \)
Consensus using $\diamond S$

$v_p := \text{input bit}; \ r := 0; \ t_p := 0; \ state_p := \text{undecided}$

while $p$ undecided do

\begin{align*}
  r &:= r + 1 \\
  c &:= (r \mod n) + 1
\end{align*}

{phase 1: all processes send opinion to current coordinator}

$p$ sends $(p, r, v_p, t_p)$ to $c$

{phase 2: current coordinator gather a majority of opinions}

$c$ waits for first $\lceil n/2+1 \rceil$ opinions $(q, r, v_q, t_q)$

$c$ selects among them the value $v_q$ with the largest $t_q$

$c$ sends $(c, r, v_q)$ to all

{phase 3: all processes wait for new suggestions from the current coordinator}

$p$ waits until suggestion $(c, r, v)$ arrives or $c \in \diamond S_p$

if suggestion is received then \{ $v_p := v; \ t_p := r; \ p$ sends $(r, \text{ACK})$ to $c$ \}

else $p$ sends $(r, \text{NACK})$ to $c$

{phase 4: coordinator waits for majority of replies. If majority adopted the coordinator’s suggestion, then coordinator sends request to decide}

$c$ waits for first $\lceil n/2+1 \rceil$ $(r, \text{ACK})$ or $(r, \text{NACK})$

if $c$ receives $\lceil n/2+1 \rceil$ ACKs, then $c$ sends $(r, \text{DECIDE}, v)$ to all

when $p$ delivers $(r, \text{DECIDE}, v)$ then \{ $p$ decides $v$ ; $state_p := \text{decided}$ \}
Validity

\[ v_p := \text{input bit; } r := 0; \ t_p := 0; \text{state}_p := \text{undecided} \]

while \( p \) undecided do
\[ r := r + 1 \]
\[ c := (r \mod n) + 1 \]
{phase 1: all processes send their opinion to current coordinator}
\[ p \text{ sends } (p, r, v_p, t_p) \text{ to } c \]
{phase 2: current coordinator gather a majority of opinions}
\[ c \text{ waits for first } \lceil n/2+1 \rceil \text{ opinions (q, r, v_q, t_q)} \]
\[ c \text{ selects among them the value } v_q \text{ with largest } t_q \]
\[ c \text{ sends } (c, r, v_q) \text{ to all} \]
{phase 3: all processes wait for new suggestions from the current coordinator}
\[ p \text{ waits until suggestion } (c, r, v) \text{ arrives or } c \in \text{state}_p \]
if the suggestion is received then
\[ \{v_p := v; \ t_p := r; \text{p sends } (r, \text{ACK}) \text{ to } c \} \]
else \( p \) sends \( (r, \text{NACK}) \text{ to } c \)
{phase 4: coordinator waits for majority of replies. If majority adopted the coordinator's suggestion, then coordinator sends request to decide}
\[ c \text{ waits for first } \lceil n/2+1 \rceil \text{ (r, ACK) or (r, NACK)} \]
if \( c \) receives \( \lceil n/2+1 \rceil \text{ ACKs}, \) then
\[ c \text{ sends } (r, \text{DECIDE, } v) \text{ to all} \]
when \( p \) delivers \( (r, \text{DECIDE, } v) \) then
\[ \{p \text{ decides } v; \text{state}_p := \text{decided}\} \]

- The value decided upon must have been suggested by the coordinator in some round
- A coordinator suggests a value only by selecting it among the participants' opinions
- From the algorithm, it is clear that each opinion correspond to a value proposed by some process
Agreement

\[ v_p := \text{input bit}; \ r := 0; \ t_p := 0; \ \text{state}_p := \text{undecided} \]
while \( p \) undecided do
\[ r := r + 1 \]
\[ c := (r \mod n) + 1 \]
{phase 1: all processes send their opinion to current coordinator} \( p \text{ sends } (p, r, v_p, t_p) \) to \( c \)
{phase 2: current coordinator gather a majority of opinions}
\( c \text{ waits for first } \lceil n/2+1 \rceil \text{ opinions } (q, r, v_q, t_q) \)
\( c \text{ selects among them the value } v_q \text{ with largest } t_q \)
\( c \text{ sends } (c, r, v_q) \) to all
{phase 3: all processes wait for new suggestions from the current coordinator}
\( p \text{ waits until suggestion } (c, r, v) \text{ arrives or } c \in \text{Sp} \)
if the suggestion is received then
\[ \{v_p := v; \ t_p := r; \ p \text{ sends } (r, \text{ACK}) \text{ to } c \} \]
else \( p \text{ sends } (r, \text{NACK}) \) to \( c \)
{phase 4: coordinator waits for majority of replies. If majority adopted the coordinator's suggestion, then coordinator sends request to decide} \( c \text{ waits for first } \lceil n/2+1 \rceil \text{ (r, ACK) or (r, NACK)} \)
if \( c \) receives \( \lceil n/2+1 \rceil \text{ ACKs}, \text{then} \)
\( c \text{ sends } (r, \text{DECIDE, v}) \) to all
when \( p \) delivers \( (r, \text{DECIDE, v}) \) then
\[ \{p \text{ decides } v; \text{state}_p := \text{decided}\} \]

**Strong Agreement** All processes that decide, decide the same value

**Proof**

- Trivially true if no process decides
- If some process decides, it has delivered (\( -, \text{DECIDE, -} \)) from a coordinator
- The coordinator has received a majority of (\( -, \text{ACK} \))
- Let \( r \) be the earliest round in which a majority of (\( -, \text{ACK} \)) have been sent to the coordinator \( c \) of \( r \)
- Let \( v_c \) be the value suggested by \( c \) in Phase 2 of round \( r \)
- Enter the Locking Lemma!
The Locking Lemma - I

\[ v_p := \text{input bit}; \quad r := 0; \quad t_p := 0; \quad \text{state}_p := \text{undecided} \]

while \( p \text{ undecided} \) do
  \( r := r + 1 \)
  \( c := (r \mod n) + 1 \)
  {\text{phase 1: all processes send their opinion to current coordinator}}
  \( p \text{ sends } (p, r, v_p, t_p) \text{ to } c \)
  {\text{phase 2: current coordinator gather a majority of opinions}}
  \( c \text{ waits for first } \left\lceil \frac{n}{2} + 1 \right\rceil \text{ opinions (q, r, v_q, t_q)} \)
  \( c \text{ selects among them the value } v_q \text{ with largest } t_q \)
  \( c \text{ sends } (c, r, v_q) \text{ to all} \)
  {\text{phase 3: all processes wait for new suggestions from the current coordinator}}
  \( p \text{ waits until suggestion } (c, r, v) \text{ arrives or } c \in \delta S_p \)
  if the suggestion is received then
    \{\text{send reply } (p, v, t_p := r; p \text{ sends } (r, \text{ACK}) \text{ to } c \text{}} \}
  \text{else } p \text{ sends } (r, \text{NACK}) \text{ to } c \}
  {\text{phase 4: coordinator waits for majority of replies. If majority adopted}}
  \text{the coordinator's suggestion, then coordinator sends request to decide}}
  \( c \text{ waits for first } \left\lceil \frac{n}{2} + 1 \right\rceil \text{ (r, ACK) or (r, NACK)} \)
  if \( c \text{ receives } \left\lceil \frac{n}{2} + 1 \right\rceil \text{ ACKs, then} \)
  \( c \text{ sends } (r, \text{DECIDE}, v) \text{ to all} \)
  when \( p \text{ delivers } (r, \text{DECIDE}, v) \text{ then} \)
  \( \{p \text{ decides } v \text{; state}_p := \text{decided}\} \)

**Locking Lemma**  For all rounds \( r' \):
\[ r' \geq r \text{ if a coordinator } c' \text{ sends } v_{c'}, \]
then \( v_{c'} = v_c \)

**Proof**

\( \triangleright \text{Trivially holds for } r' = r \)
\( \triangleright \text{Assume it holds for all } r' : r \leq r' < k \)
\( \triangleright \text{Let } c_k \text{ be the coordinator for round } k \)
\( \triangleright \text{If } c_k \text{ suggests } v_{c_k}, \text{ it must have} \)
\( \text{received opinions from a majority of processes} \)
\( \triangleright \text{There exists some } p \text{ that sent an ACK} \)
\( \text{in Phase 3 of round } r \text{ and whose} \)
\( \text{opinion has been received by } c_k \)
\( \triangleright \text{Consider the time of adoption } t_p \)
\( \triangleright \text{In Phase 3 of round } r, t_p = r \)
\( \triangleright \text{In Phase 2 of round } k, t_p \geq r \)
\( \triangleright \text{For any } t_q \text{ collected in round } k, t_q < k \)
The Locking Lemma – II

\( v_p := \text{input bit}; \ r := 0; \ t_p := 0; \ \text{state}_p := \text{undecided} \)

while \( p \) undecided do

\( r := r+1 \)

\( c := (r \mod n) + 1 \)

{phase 1: all processes send their opinion to current coordinator}

\( p \) sends \( (p, r, v_p, t_p) \) to \( c \)

{phase 2: current coordinator gather a majority of opinions}

\( c \) waits for first \( \lceil n/2+1 \rceil \) opinions \( (q, r, v_q, t_q) \)

\( c \) selects among them the value \( v_q \) with largest \( t_q \)

\( c \) sends \( (c, r, v_q) \) to all

{phase 3: all processes wait for new suggestions from the current coordinator}

\( p \) waits until suggestion \( (c, r, v) \) arrives or \( c \in \text{S}_p \)

if the suggestion is received then

\( \{v_p := v; \ t_p := r; \ p \) sends \( (r, \text{ACK}) \) to \( c \} \)

else \( p \) sends \( (r, \text{NACK}) \) to \( c \)

{phase 4: coordinator waits for majority of replies. If majority adopted the coordinator’s suggestion, then coordinator sends request to decide}

\( c \) waits for first \( \lceil n/2+1 \rceil \) \( (r, \text{ACK}) \) or \( (r, \text{NACK}) \)

if \( c \) receives \( \lceil n/2+1 \rceil \) ACKs, then \n
\( c \) sends \( (r, \text{DECIDE}, v) \) to all

when \( p \) delivers \( (r, \text{DECIDE}, v) \) then

\( \{p \) decides \( v \); \ \text{state}_p := \text{decided}\}

ości

Consider \( t, \) the largest time of adoption collected by \( c_k. \) Clearly, \( r \leq t < k \)

\( c_k \) adopted its suggestion from \( q, \) where \( q \) is the process that sent \( (q, k, v_q, t) \)

The coordinator of round \( t \) sent its suggestion in Phase 2 of round \( t, \) where \( r \leq t < k \)

By the Induction Hypothesis, that coordinator sent \( v_c! \)

Then, \( c_k \) sets \( v_{c_k} \) to \( v_c \)

Been there, done that?
Agreement

\( v_p := \text{input bit}; \ r := 0; \ t_p := 0; \ \text{state}_p := \text{undecided} \)

while \( p \) undecided do
\( \ r := r+1 \)
\( \ c := (r \mod n) + 1 \)

{phase 1: all processes send their opinion to current coordinator}
\( \ p \) sends \( (p, r, v_p, t_p) \) to \( c \)

{phase 2: current coordinator gather a majority of opinions}
\( c \) waits for first \( \lceil n/2+1 \rceil \) opinions \( (q, r, v_q, t_q) \)
\( c \) selects among them the value \( v_q \) with largest \( t_q \)
\( c \) sends \( (c, r, v_q) \) to all

{phase 3: all processes wait for new suggestions from the current coordinator}
\( p \) waits until suggestion \( (c, r, v) \) arrives or \( c \in \circ S_p \)
if the suggestion is received then
\( \{ v_p := v; \ t_p := r; \ p \ \text{sends} \ (r, \text{ACK}) \ \text{to} \ c \} \)
else \( p \ \text{sends} \ (r, \text{NACK}) \ \text{to} \ c \)

{phase 4: coordinator waits for majority of replies. If majority adopted the coordinator's suggestion, then coordinator sends request to decide}
\( c \) waits for first \( \lceil n/2+1 \rceil \) \( (r, \text{ACK}) \) or \( (r, \text{NACK}) \)
if \( c \) receives \( \lceil n/2+1 \rceil \text{ ACKs} \), then
\( c \) sends \( (r, \text{DECIDE}, v) \) to all

when \( p \) delivers \( (r, \text{DECIDE}, v) \) then
\( \{ p \ \text{decides} \ v \ ; \ \text{state}_p := \text{decided} \} \)

All processes that decide, decide \( v_c \)

Proof

\( \square \) Suppose \( p \) delivers \( (r^*, \text{DECIDE}, v_{c^*}) \)

\( \square \) The coordinator \( c^* \) for round \( r^* \) has sent \( (r^*, \text{DECIDE}, v_{c^*}) \) in Phase 4 of round \( r^* \)

\( \square \) To do so \( c^* \) must have received a majority of \( (r^*,\text{ACK}) \) in Phase 4 of \( r^* \)

\( \square \) \( r \) is the earliest round in which a majority of \( (r, \text{ACK}) \) have been sent to a round's coordinator

\( \square \) Clearly, \( r \leq r^* \)

\( \square \) By the locking Lemma, \( c' \) must have suggested the locked value: \( v_{c^*} = v_c \)
Termination

\[ v_p := \text{input bit}; \ r := 0; \ t_p := 0; \ \text{state}_p := \text{undecided} \]

while \( p \) undecided do
\[ r := r+1 \]
\[ c := (r \mod n) + 1 \]

\{phase 1: all processes send their opinion to current coordinator\}
\[ p \text{ sends } (p, r, v_p, t_p) \text{ to } c \]

\{phase 2: current coordinator gather a majority of opinions\}
\[ c \text{ waits for first } \left\lceil \frac{n}{2} + 1 \right\rceil \text{ opinions } (q, r, v_q, t_q) \]
\[ c \text{ selects among them the value } v_q \text{ with largest } t_q \]
\[ c \text{ sends } (c, r, v_q) \text{ to all} \]

\{phase 3: all processes wait for new suggestions from the current coordinator\}
\[ p \text{ waits until suggestion } (c, r, v) \text{ arrives or } c \in \overset{p}{\circ}S_p \]

if the suggestion is received then
\[ \{v_p := v; \ t_p := r; \ p \text{ sends } (r, \text{ACK}) \text{ to } c \} \]

else \( p \) sends \( (r, \text{NACK}) \text{ to } c \)

\{phase 4: coordinator waits for majority of replies. If majority adopted the coordinator’s suggestion, then coordinator sends request to decide\}
\[ c \text{ waits for first } \left\lceil \frac{n}{2} + 1 \right\rceil \text{ } (r, \text{ACK}) \text{ or } (r, \text{NACK}) \]

if \( c \) receives \( \left\lceil \frac{n}{2} + 1 \right\rceil \text{ ACKs} \), then
\[ c \text{ sends } (r, \text{DECIDE}, v) \text{ to all} \]
when \( p \) delivers \( (r, \text{DECIDE}, v) \) then
\[ \{p \text{ decides } v; \ \text{state}_p := \text{decided} \} \]

\[ \bowtie \text{No correct process is blocked forever at a wait statement} \]

\[ \bowtie \text{By eventual weak accuracy, there is a correct process } c \text{ and a time } t \]
\[ \bowtie \text{such that no process suspects } c \]
\[ \text{after } t \]

\[ \bowtie \text{There is a round } r \text{ such that:} \]
\[ \square \text{all correct processes reach } r \]
\[ \square \text{after time } t \text{ (no one suspects } c) \]
\[ \square c \text{ is the coordinator for round } r \]

\[ \bowtie \text{If some correct process decides, eventually all do on the same value} \]
\[ \text{by Agreement} \]