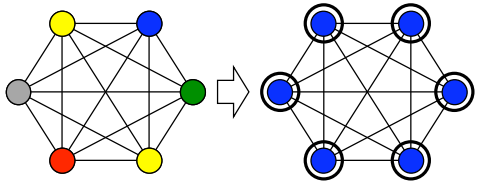
The consensus problem

The consensus problem

- Informal definition
  - Each process starts with an initial value
  - Each one is supposed to output the same value
  - A validity condition relates outputs to inputs



The consensus problem

- Failure model
  - Crash-stop failure model
    - Correct processes never crash
    - Faulty processes are not correct
- Formal definition(s)
  - Two primitives
    - propose(v), and
    - decide(v), where $v \in V$

The consensus problem

- Formal definition(s)
  - Consensus properties
    - Agreement: No two correct processes decide on different values.
    - Termination: All correct processes eventually decide.
    - Validity: If all processes start with the same initial value v, then v is the only possible decision value.
    - Strong validity: Any decision value is the initial value of some process.

The consensus problem

- Formal definition(s)
  - Uniform consensus
    - Uniform agreement: No two processes decide on different values.
    - Termination: All correct processes eventually decide.
    - Strong validity: Any decision value is the initial value of some process.

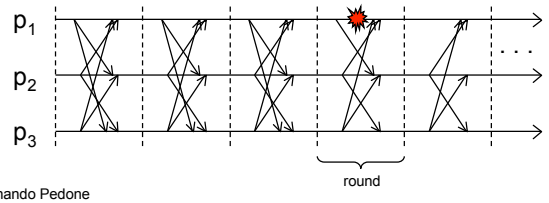
The consensus problem

- Synchronous systems
  - System model
    - The execution begins with all processes in arbitrary start states, and all channels empty. Then, in lock-step mode, processes do:
    - Apply a message-generation function to the current state to generate messages and send them to all outgoing neighbors.
    - Apply a state-transition function to the current state and incoming messages to obtain the new state.

round

The consensus problem

- Synchronous systems
  - System model (cont'd)
    - If a process crashes in the middle of a message-sending step, only a subset of the messages may be delivered. At most f processes crash.



The consensus problem

- The algorithm (UA, T, V)
  - Each process maintains a variable W containing a subset of V. Initially, process p_i's variable W has only p_i's initial value. For each of f+1 rounds, each process broadcasts W, then adds all the elements of the received sets to W.
    After f+1 rounds, p_i decides on the unique element of W; otherwise, p_i decides on the default value v_0.

The consensus problem

– Correctness

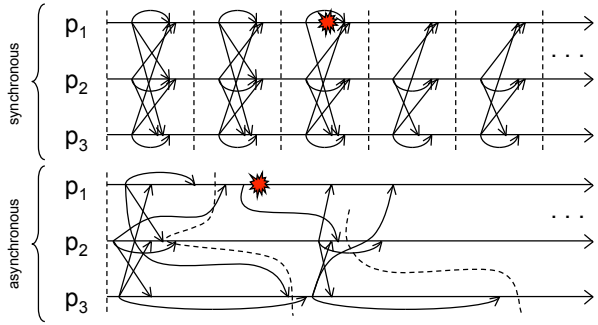
- Lemma 1: If no process fails during a particular round r, $1 \leq r \leq f+1$, then $W_i(r) = W_j(r)$ for all p_i and p_j that are active after r rounds.
- Lemma 2: Suppose that $W_i(r) = W_j(r)$ for all p_i and p_j that are active after r rounds. Then, for any round r', $r \leq r' \leq f+1$, the same holds, that is, $W_i(r) = W_j(r)$ for all p_i and p_j that are active after r' rounds.
- Lemma 3: If processes p_i and p_j are active after f+1 rounds, then $W_i(r) = W_j(r)$ at the end of round f+1.

The consensus problem

- Asynchronous systems
  – System model
  - All processes begin in arbitrary start states, and all channels empty. Then, processes do:
  - Apply a message-generation function to the current state to generate messages and send them to all outgoing neighbors.
  - Wait for n-f messages and apply a state-transition function to the current state and incoming messages to obtain the new state.

The consensus problem

– Synchronous vs. asynchronous



The consensus problem

– Fundamental result:

No algorithm can solve consensus in an asynchronous system despite a single crash.

FLP impossibility result (after Fischer, Lynch, and Paterson, 1985)

Synchronous assumptions are too strong…
Asynchronous assumptions don't allow a solution…
…is there hope?

The consensus problem

- Solving consensus by…
  - …weakening the problem definition
  - …strengthening the model assumptions
  - …doing both

The consensus problem

- k-Agreement
  - Properties
    - Agreement: There is a subset W of V, |W| = k, such that all decision values are in W.
    - Termination: All correct processes eventually decide.
    - Validity: Any decision value is the initial value of some process.

The consensus problem

- Randomized algorithm
  - Stronger than the asynchronous model
    - Processes can made random choices
  - Weak termination
    - Correct processes decide at time t with probability at least p(t)

The consensus problem

- Failure detectors
  - Prevent processes from blocking forever
  - General idea:

    wait for message m from p_i

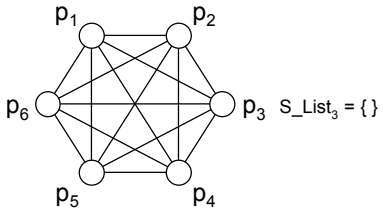
    is replaced by

    wait for m from p_i or suspect p_i

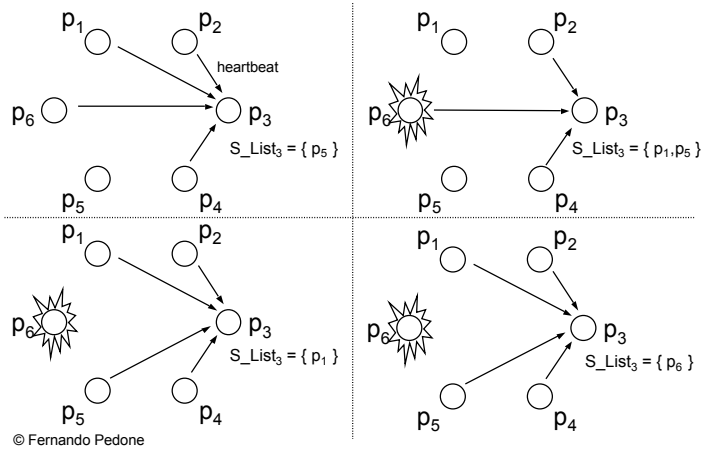
The consensus problem

– Informally

- Completeness
  Faulty processes should be suspected
- Accuracy
  Correct processes should not be suspected



The consensus problem



The consensus problem

– 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.

The consensus problem

– Accuracy

- Strong accuracy
  No process is suspected before it crashes.
- Weak accuracy
  Some correct process is never suspected.
- Eventual strong accuracy
  Eventually correct processes are not suspected.
- Eventual weak accuracy
  Eventually some correct process is not suspected.

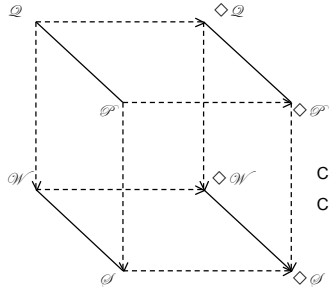
The consensus problem

– Classes of failure detectors

Completeness	Accuracy			
	Strong	*Weak*	*Eventual Strong*	*Eventual Weak*
Strong	Perfect, $\mathcal{P}$	Strong, $\mathcal{S}$	Eventually Perfect, $\diamond\mathcal{P}$	Eventually Strong, $\diamond\mathcal{S}$
Weak	$\mathcal{Q}$	Weak, $\mathcal{W}$	$\diamond\mathcal{Q}$	Eventually Weak, $\diamond\mathcal{W}$

The consensus problem

– Classes of failure detectors (cont'd)



C --> C' : C' is strictly weaker than C'

C — C' : C' is equivalent to C'

The consensus problem

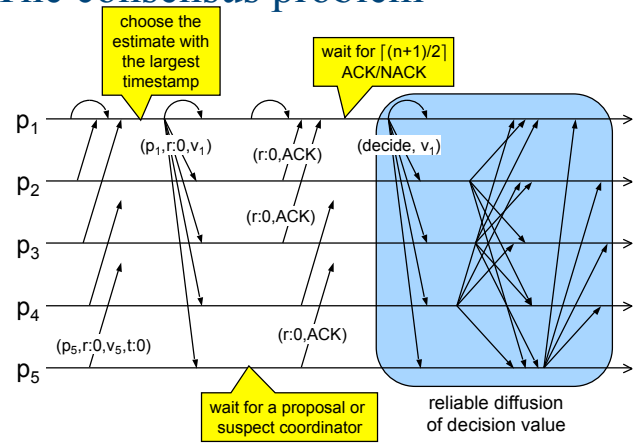
– Solving consensus with $\diamond\mathcal{S}$

```
var     v ← input
        r ← 0
        t ← 0

while undecided do
        … (next) …

upon receiving (decide, w)
        p sends (decide, w) to all
        p decides on w and halts
```


The consensus problem

```
while undecided do
        r ← r + 1
        c ← (r mod n) + 1

phase 1, phase 2:
        send (p,r,v,t) to c

        c waits for first ⌈(n+1)/2⌉ estimates
        c chooses the estimate w with the largest timestamp
        c proposes that value as (c,r,w)

phase 3:
        p waits for a proposal or suspects c
        if proposal (c,r,w) is received then
                v ← w; t ← r
                send (r, ACK) to c
        else
                send (r, NACK) to c

phase 4:
        c waits for ⌈(n+1)/2⌉ ACK/NACK messages
        if there are ⌈(n+1)/2⌉ ACK messages then
                c sends (decide, w) to all
```

The consensus problem



The consensus problem