Impossibility of Distributed Consensus with One Faulty Process

Journal of the ACM 32(2):374-382, April 1985.
MJ Fischer, NA Lynch, MS Peterson.

Won the 2002 Dijkstra Award (for influential paper in distributed computing).

Very Weak Consensus

Assume that each process has an initial value $\in \{0, 1\}$.

If v is a Boolean value, then $\underline{v}$ stands for the complement value $1-v$.

Considers the problem of *very weak consensus*:

- *Weak Termination* Some correct process decides.
- *Agreement* No two correct processes decide on different values.
- *Very Weak Validity* There is a run in which the decision is 0 and a run in which the decision is 1.

... not all correct processes need decide and 0 and 1 are the possible outcomes.

Basics

- Let P be the set of processes
- Let M be a message buffer containing messages of the format $(p, q: m)$ where $p, q \in P$.

A process is an automaton that proceeds in steps, where a step is:

$\langle$send 0 or more messages; receive message or λ; change state$\rangle$

... the state change is determined solely by the current state and the messages received.

Basics (continued)

- A process receives a message from M; the exact message it delivers is chosen nondeterministically. Delivery is not immediate, and it isn't FIFO, but it can not be postponed forever.
- Each $p \in P$ has two distinct initial states σ_p^0 and σ_p^1. It also has two distinct sets of decision states D_p^0 and D_p^1. Being in one of these sets of states is stable.

Basics (continued)

- $C = (s, M)$ is a configuration of the systems:
  - $s(p)$ is the state of process p
  - M is the message buffer $\{ (p, q: m) \}$ (hence unique messages).
  - Initial configuration
    - $M = \{ \}$
    - $\forall p \in P: s(p) \in \{\sigma_p^0, \sigma_p^1\}$
- Event e of process p is $(p, q: m)$ or $(\lambda: p)$.
  - e is applicable to C if $e \in M$ or e is $(\lambda: p)$.

Basics (continued)

- $e(C)$ is the configuration resulting by applying e to C.

  Assuming e is of p, the state of p changes, e removed from M if it is not $(\lambda: p)$, and zero or more messages are added to M.

- A schedule S is (finite or infinite) sequence of events.

  If $S = e_1\ e_2\ e_3$ ... then $C^1 = e_1(C)$, $C^2 = e_2(C^1)$, $C^3 = e_3(C^2)$... thus creating a sequence of configurations $C\ C^1\ C^2\ C^3$ corresponding to S.

Basics (continued)

- If S is finite $e_1\ e_2\ \ldots\ e_k$ then $C^k = S(C)$. We say that C^k *follows* C.
- If S is infinite, then the corresponding sequence of configurations $C\ C^1\ C^2\ \ldots$ is called a *run* if:
  - C is an initial configuration
  - $\forall\ p \in P$: no non-λ event e of p is applicable to infinitely many C^i unless p has only a finite number of events in S.

    Hence, if p attempts to receive e from q infinitely often, then p will eventually receive e.

Basics (continued)

- If there are not an infinite number of events in p in the run, then we say that p is *faulty* in the run. This models p as having crashed. Otherwise, we say that p is *correct* in the run.
- For $v \in \{0, 1\}$, configuration C is *v-determined* if there is no schedule S and process p: $S(C) = (s, M)$ where $s(p) \in D_p^{\underline{v}}$.
  - A configuration is *determined* if it is 0-determined or 1-determined. If it is not determined, then it is *undetermined*.

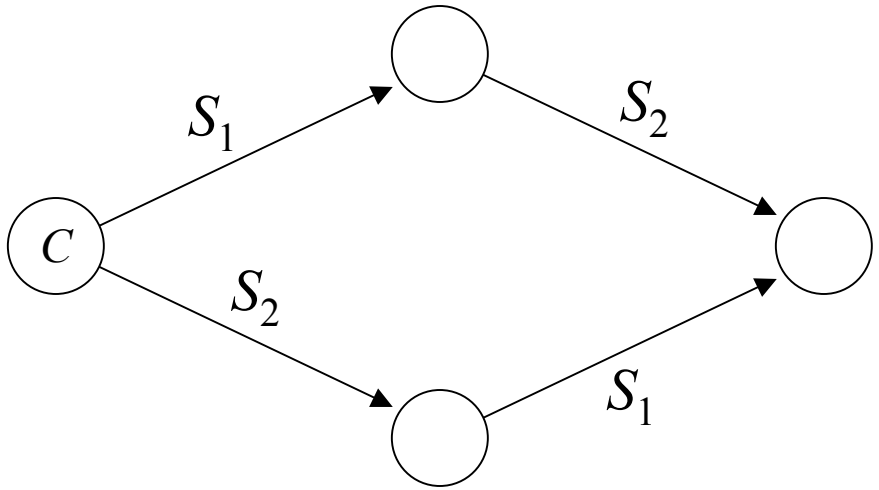
Restating the problem

- *Weak Termination* In any run R, $\exists\, p \in P$ correct in R and a configuration $C = (s, M)$ in R: $s(p) \in D_p^0 \cup D_p^1$.
- *Agreement* In any run R there is no configuration R and p, q correct in R: $s(p) \in D_p^0$ and $s(q) \in D_q^1$.
- *Very Weak Validity* For both $v \in \{0, 1\}$ there is a run R_v, a process p correct in R_v, and a configuration $C = (s, M)$ in R_v where $s(p) \in D_p^v$.

A Handy Lemma

- Let S_1 and S_2 be schedules from C such that no process has events in both S_1 and S_2. Then, $S_1(S_2(C)) = S_2(S_1(C))$.

  Proof by straightforward induction.



The theorem

Very Weak Consensus is not solvable if one or more processes may be faulty.

Proof: Assume that there is a protocol Π that solves Very Weak Consensus when at most one process can be faulty.

Proof (continued)

FACT 1 Π has at least one undetermined initial configuration.

FACT 2 Let C be an undetermined configuration and let e and event applicable to C.

- Let $CS = \{C'$: $C' = S(C)$ for some finite S not containing $e\}$
- Let $DS = \{D$: $\exists\ C'$ in CS: $D = e(C')\}$

... DS contains an undetermined configuration.

Proof (continued)

... assuming FACT 1 and FACT 2 are true, then we can derive a contradiction.

- Let C^0 be an undetermined initial configuration (FACT 1).
- Construct a run R starting with C^0 in which every corresponding configuration is undetermined (thus violating Weak Termination).

Proof (continued)

Algorithm for constructing R. Put the process identifiers P into a FIFO queue Q.

$C = C^0$
do forever
    p = first process identifier in Q
    if (M contains an event of p) e = oldest such event
    else $e = (\lambda: p)$
    extend run to undetermined configuration $D = e(S(C))$ for some S (FACT 2)
    move p to the end of Q

Note that all processes are correct in R and select events in M so that no event is applicable infinitely often.

Proof of FACT 1

FACT 1 Π has at least one undetermined initial configuration.

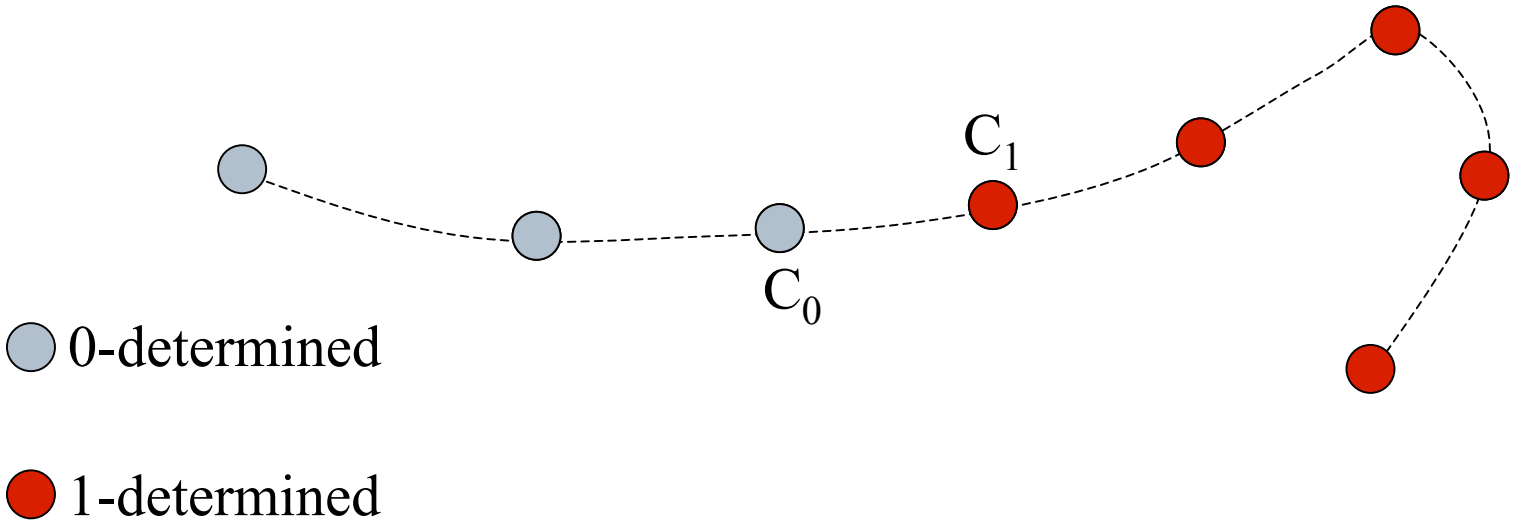
Assume that all initial configuration of Π are determined.

From *Very Weak Validity*, there is at least one initial configuration that is 0-determined and one that is 1-determined.

Say that two configurations are *neighbors* if they differ in only one initial value.

Proof of FACT 1 (continued)

There must be two initial configurations C_0 and C_1 that are neighbors (with p being the process with different initial values) such that C_v is v-determined.



Proof of FACT 1 (continued)

Let C_0 F^1 F^2 ... be a run which contains no events of p (hence p fails in the first state). From *Very Weak Validity*, there is such a run in which some process q decides. Since C_0 is 0-determined, q decides 0. Let this happen in F^i.

Note that F^1 F^2 ... is also applicable to C_1 since C_0 and C_1 are identical except for the state of p, and F^1 F^2 ... contains no events of p. As before, some process q' decides 1.

... but, being deterministic, q still decides 0, violating *agreement*.

Proof of FACT 2

FACT 2 Let C be an undetermined configuration and let e and event applicable to C.

- Let $CS = \{C' : C' = S(C)$ for some finite S not containing $e\}$
- Let $DS = \{D : \exists\ C'$ in $CS : D = e(C')\}$

... DS contains an undetermined configuration.

Assume that all $D \in DS$ are determined.

Proof of FACT 2 (continued)

- By definition, e is applicable to all $C' \in CS$.

  so, $e(C')$ is well defined and is in DS.

  CLAIM 1: There are both 0- and 1-determined configurations in DS.

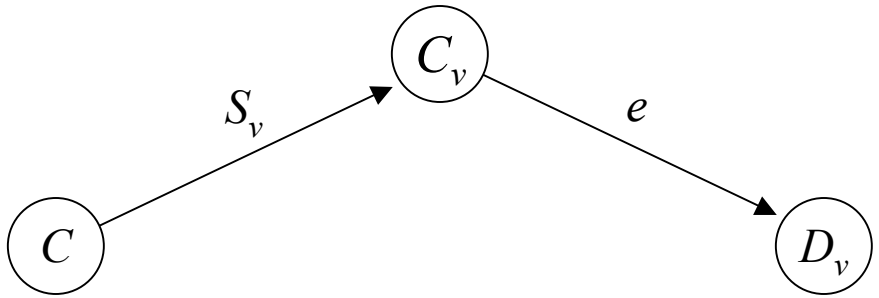
  Proof C is undetermined, and so by *weak termination* there is:

  - a schedule S_0: $S_0(C) = C_0$ that is 0-determined
  - a schedule S_1: $S_1(C) = C_1$ that is 1-determined

Proof of FACT 2 CLAIM 1 (continued)

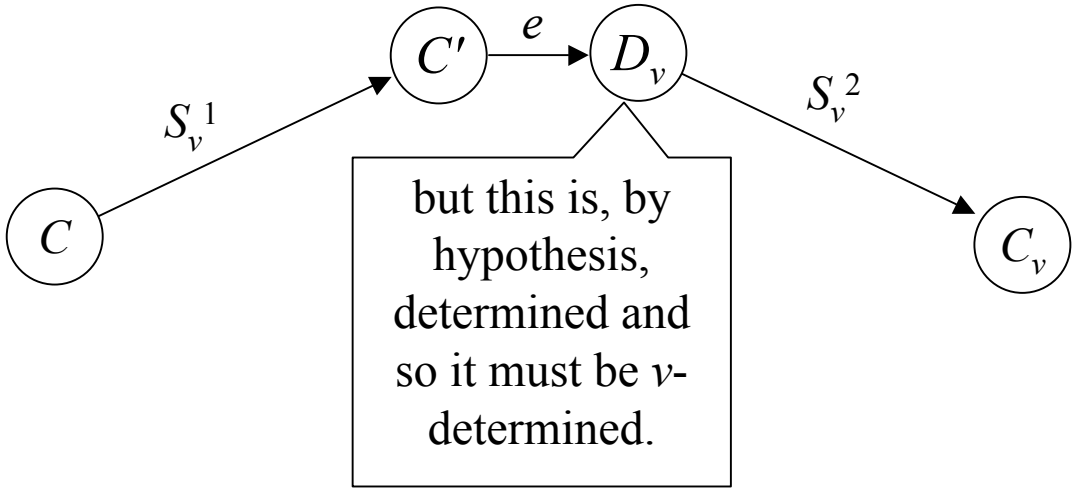
Consider two cases:

- If $C_v \in CS$ then e is not in S_v. Let $D_v = e(C_v)$. D_v is thus in DS and is v-determined.



Proof of FACT 2 CLAIM 1 (continued)

- If $C_v \notin CS$ then S_v contains e. Let $D_v = e(C_v)$. D_v is thus in DS and is v-determined.



Proof of FACT 2 CLAIM 2

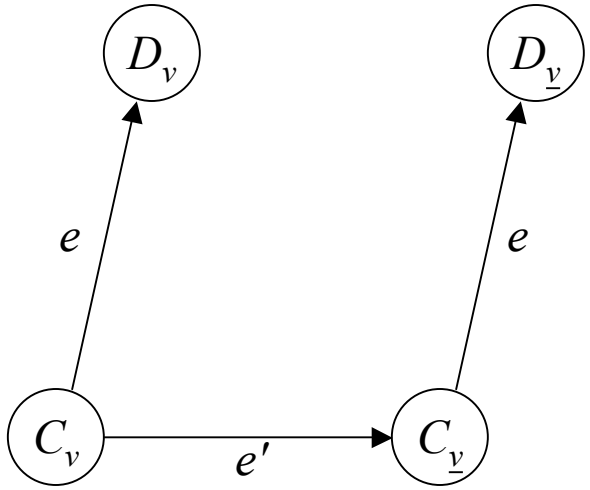
Define two configurations E_1 and E_2 to be adjacent if $\exists\ e'$: $E_1 = e'(E_2)$ or $E_2 = e'(E_1)$

CLAIM 2: There are two adjacent configuration C_0 and $C_1 \in CS$: $D_v = e(C_v)$ is v-determined.

Proof: Construct a graph with nodes representing configurations in CS and edges indicating adjacent configurations. Label each node with v where $e(C') \in DS$ is v-determined.

- The graph is connected (all have a path back to the initial state).
- By CLAIM 1, there is a node in the graph labeled 0 and a node labeled 1.

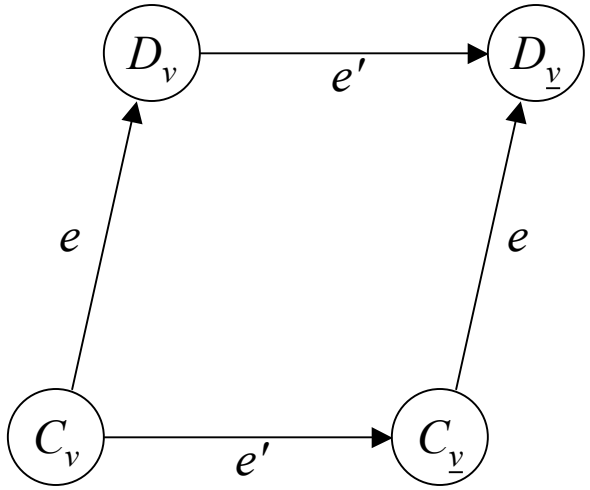
Proof of FACT 2 (continued)



assume e is of process p.

Proof of FACT 2 (continued)

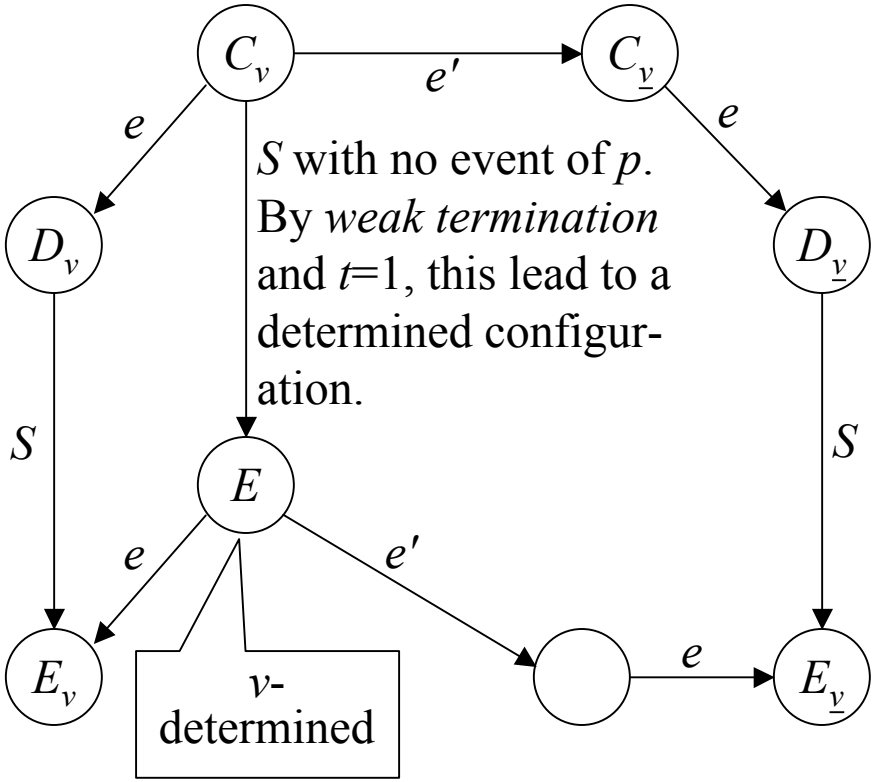
apply the handy lemma



assume e is of process p.
assume e' is not of process p.

Proof of FACT 2 (continued)

- If e and e' are both of p, then:



What does this mean?

- Common interpretation is *can't distinguish between a slow process and a crashed one* which is close.
- If *eventually there is some correct process that is not mistaken for a crashed process* (perhaps because some process, at some time in the indefinite future, is not too slow), then we can solve consensus.