

More techniques for localised failures

Riku Saikkonen

4th April 2007

*Based on sections 7.4–7.7 of
N. Santoro: Design and Analysis of
Distributed Algorithms, Wiley 2007.*

Contents

Ways of avoiding the Single-Fault Disaster theorem:

- synchronous systems (previous presentation)
- randomisation
- failure detectors
- pre-execution failures

And a slightly different topic:

- localised permanent link failures

Restrictions

Assumptions for all the node failure topics:

- connectivity, bidirectional links, unique IDs
- complete graph
- **at most f nodes can fail**, and only by crashing
- (asynchronous system)

Using randomisation

Uncertainty

Non-determinism \Rightarrow uncertain results
 \Rightarrow a probability distribution on executions

Types of randomised protocols:

Monte Carlo always terminates
correct result with high probability

Las Vegas always correct
terminates with high probability

Hybrid both with high probability

Example: Randomised asynchronous consensus

Consensus problem:

- nodes have initial values 0 or 1
- goal: all non-faulty nodes decide on a common value
- non-triviality: if all values are the same, select that one

Las Vegas protocol Rand-Omit (next slide):

- solves Consensus with up to $f < n/2$ crash failures
- additional restriction: Message Ordering

Algorithm Rand-Omit

pref \leftarrow initial value; $r \leftarrow 1$;

repeat

Send $\langle \text{VOTE}, r, \text{pref} \rangle$ to all.

Receive $n - f$ VOTE messages.

if all have the same value v

then found $\leftarrow v$ **else** found $\leftarrow ?$;

Send $\langle \text{RATIFY}, r, \text{found} \rangle$ to all.

Receive $n - f$ RATIFY messages.

if one or more have a value $w \neq ?$ **then**

pref $\leftarrow w$;

if all have the same w **and** not decided yet **then**

Decide on w .

else pref \leftarrow 0 or 1 randomly;

$r \leftarrow r + 1$

until one round after we made our decision

Algorithm Rand-Omit

pref \leftarrow initial value; $r \leftarrow 1$;

repeat

Send $\langle \text{VOTE}, r, \text{pref} \rangle$ to all.

stage 1 Receive $n - f$ VOTE messages.

if all have the same value v
then found $\leftarrow v$ **else** found $\leftarrow ?$;

Send $\langle \text{RATIFY}, r, \text{found} \rangle$ to all.

Receive $n - f$ RATIFY messages.

stage 2 **if** one or more have a value $w \neq ?$ **then**

pref $\leftarrow w$;

if all have the same w **and** not decided yet **then**

Decide on w .

else pref \leftarrow 0 or 1 randomly;

$r \leftarrow r + 1$

until one round after we made our decision

Algorithm Rand-Omit

pref \leftarrow initial value; $r \leftarrow 1$;

repeat

Send $\langle \text{VOTE}, r, \text{pref} \rangle$ to all.

stage 1 Receive $n - f$ VOTE messages.

if all have the same value v *or: $> n/2$ messages*

then found $\leftarrow v$ **else** found $\leftarrow ?$;

Send $\langle \text{RATIFY}, r, \text{found} \rangle$ to all.

Receive $n - f$ RATIFY messages.

stage 2 **if one or more** have a value $w \neq ?$ **then**

pref $\leftarrow w$;

if all have the same w **and not decided yet then** *or: $> f$*

Decide on w .

else pref \leftarrow 0 or 1 randomly;

$r \leftarrow r + 1$

until one round after we made our decision

Analysis of Rand-Omit

Lemma: If $\text{pref}_x(r) = v$ for every correct x , then all correct entities decide on v in that round r .

Lemma: In every round r , for all correct x , either $\text{found}_x(r) \in \{0, ?\}$ or $\text{found}_x(r) \in \{1, ?\}$.

Lemma: If x makes the first decision on v at round r , then all nonfaulty nodes decide v by round $r + 1$.

Lemma: Let “success” = prefs of correct nodes identical. Then $\Pr[\text{success within } k \text{ rounds}] \geq 1 - (1 - 2^{-(n-f)})^k$.

\Rightarrow Rand-Omit terminates with probability 1.

Theorem (very non-trivial)

If $f = O(\sqrt{n})$, the expected number of rounds in Rand-Omit is constant (i.e., independent of n).

Reducing the number of rounds

Protocol Committee

$$f < n/3 \text{ (not } n/2)$$

- create $k = O(n^2)$ committees, each having $s = O(\log n)$ nodes as members
- select the members such that at most $O(n) = O(\sqrt{k})$ committees are faulty, i.e., have $> s/3$ faulty nodes
- each committee simulates one entity of Rand-Omit
- a nonfaulty committee must work together and use its own (common) random numbers
- $O(\sqrt{k})$ faulty committees, so the expected number of simulated Rand-Omit rounds is constant
- time for simulating one round is $O(\text{coin flips}) = O(\text{max. faulty members in a nonfaulty committee}) = O(s) = O(\log n)$

Failure detection

Using failure detection

The Single-Fault Disaster theorem requires that faults cannot be detected.

- a reliable **failure detector** would make the problem solvable
- ...but cannot be constructed in practice (except for synchronous systems)
- an unreliable failure detector is often good enough!

Using failure detection

The Single-Fault Disaster theorem requires that faults cannot be detected.

- a reliable **failure detector** would make the problem solvable
- ...but cannot be constructed in practice (except for synchronous systems)
- an unreliable failure detector is often good enough!

Failure detectors are distributed: each node **suspects** some of its possibly faulty neighbours.

- additional restriction here: IDs of neighbours known

Classification of unreliable failure detectors

Completeness property “can’t suspect nothing”

Strong completeness eventually every failed node is permanently suspected by **every** correct node

Weak completeness eventually every failed node is permanently suspected by **some** correct node

Accuracy property “can’t suspect everything”

Perpetual strong no node suspected before it crashes

Perpetual weak some correct node is never suspected

Eventual strong eventually **no** correct nodes are suspected

Eventual weak eventually **one** correct node is not suspected

The weakest useful failure detector

Weak completeness to strong completeness

Algorithm to transform weak D_x to strong D'_x in node x :

initialise: $D'_x \leftarrow \emptyset$

run repeatedly: Send $\langle x, D_x \rangle$ to all neighbours.

when receiving $\langle y, s \rangle$: $D'_x \leftarrow D'_x \cup s - \{y\}$

- preserves accuracy properties

Theorem

Weak completeness and eventual weak accuracy are sufficient for reaching consensus with $f < n/2$ crashes.

Pre-execution failures

Pre-execution failures are different

The Single-Fault Disaster theorem relies on choosing the failed node and the time of failure **during** the execution of the protocol.

New restriction: Partial Reliability

- no failures occur during the computation
- at most f nodes have crashed before the protocol starts
- but we still do not know which nodes have failed

Recap: Efficient election in a complete graph

The CompleteElect algorithm from a previous presentation:

CompleteElect

no failures, n nodes, k initiators

States: candidate (initial), captured, passive

Define: s_x = number of nodes that x has captured ("stage")

Basic algorithm:

- Candidate x sends $\langle \text{Capture}, s_x, \text{id}(x) \rangle$ to a neighbour y .
- If y is **passive**, the attack succeeds.
- If y is a **candidate**, the attack succeeds if $s_x > s_y$, or $s_x = s_y$ and $\text{id}(x) < \text{id}(y)$; otherwise x becomes passive.
- If y is **captured**: y sends $\langle \text{Warning}, s_x, \text{id}(x) \rangle$ to its owner (unless s_x is too small), which replies Yes or No; y will wait for this result before issuing another Warning.

Message complexity $O(n \log n)$, time $O(n)$.

Example: Election with Partial Reliability

Changes to CompleteElect:

$$f < \lceil n/2 \rceil + 1$$

- x sends Capture to $f + 1$ neighbours (not 1)
- if x receives Accept, send one new Capture (i.e., still $f + 1$ Captures pending)
- was: unsuccessful attack (Reject message) $\Rightarrow x$ passive; now, s_x may have increased from other Captures
 - x must reject Rejects if s_x has become too large
 - this is done by **settlement**: x sends a new Capture to y and waits for its reply, queuing all other messages
- Warning-waits and settlement work because **y must be nonfaulty** due to Partial Reliability
- settlements cannot create a deadlock (because of asymmetry in s_x and s_y)

Analysis of Election with Partial Reliability

Lemma: Every node x reaches $s_x > n/2$
or ceases to be a candidate.

Lemma: Let x be a candidate and s its final size. The total number of Capture messages from x is $\leq 2s + f$.
($f + 1$ initially, $\leq s - 1$ after Accepts, $\leq s$ replies to Rejects)

Lemma: $s_x \leq n/l$ if there are $l - 1$ candidates whose final size is not smaller than that of candidate x .

$\dots \Rightarrow$ Messages: $\leq n - 1 + 4 \cdot \sum_{j=1}^k (2(n/j) + f)$

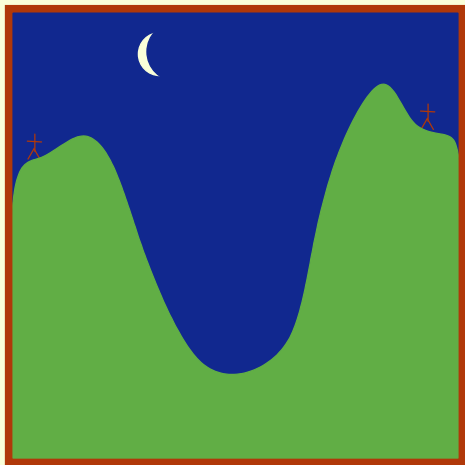
FT-CompleteElect is **worst-case optimal**:

Message complexity: $O(n \log k + kf)$

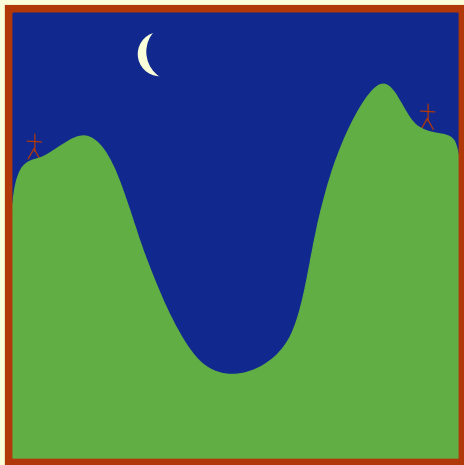
$\Omega(n \log k)$ for fault-tolerant election + $\Omega(kf)$ initial Captures

Localised link failures

A tale of two synchronous generals

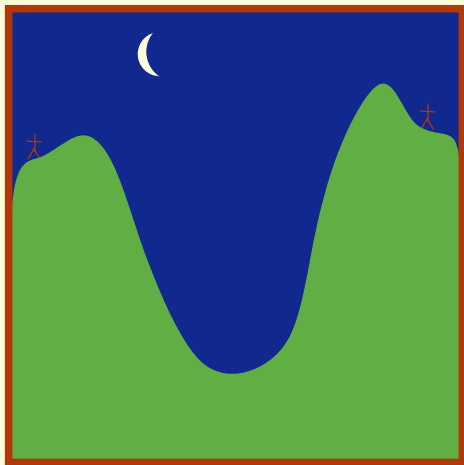


A tale of two synchronous generals



- **unsolvable** even if the system is synchronous
- nodes cannot achieve common knowledge if the only link **can** fail

A tale of two synchronous generals



- **unsolvable** even if the system is synchronous
- nodes cannot achieve common knowledge if the only link **can** fail
- broadcast not possible
⇒ common knowledge not possible
- **solution**: more links, i.e., better connectivity in the network

Assumptions

Restrictions in this section:

- fully synchronous
- at most F links can fail, and only permanently
- failure by **send-receive omissions**:
a failed link drops some of its messages
 - less restrictive than crashing, but happens to be easy to handle here

(Non-permanent link failures in next presentation.)

Edge connectivity

Edge connectivity, $c_{\text{edge}}(G)$

For graph G , $c_{\text{edge}}(G) = k$ if there are k (but not $k + 1$) edge-disjoint paths between all pairs of nodes.

The graph G is k -edge-connected, if $c_{\text{edge}}(G) \geq k$.

Common knowledge cannot be achieved (in all possible networks) unless $c_{\text{edge}}(G) \geq F + 1$.

Computing with faulty links

If the network is $F + 1$ -edge-connected, consensus and most computations can be done, even in an asynchronous system:

- because broadcasting can be done, e.g., with protocol Flood
- Flood is independent of F
- even with faulty links, Flood is optimal in time $O(\text{diam}(G'))$ and message complexity $\leq 2 \cdot m(G)$ (assuming no knowledge of the topology of the network)

Example: Broadcasting in a complete graph

Without failures, broadcasting is trivial: $n - 1$ messages.

If $F < n - 1$ of the $n(n - 1)/2$ links can fail:

- Flood works, but uses $(n - 1)^2$ messages
- the following protocol uses only $(F + 1)(n - 1)$ messages to broadcast the information i

Protocol TwoSteps

1. x sends $\langle \text{Info}, i \rangle$ to $F + 1$ neighbours
2. each y that receives it sends $\langle \text{Echo}, i \rangle$ to all its neighbours

Example: Simple election in a complete graph

A simple strategy for Election is to use a fault-tolerant broadcasting protocol:

FT-BcastElect

1. Each node x broadcasts $\text{id}(x)$.
2. When all IDs have been received, x becomes the leader iff its ID is the smallest.

The cost depends on the broadcast protocol:
using TwoSteps, $n(F + 1)(n - 1)$ messages are used.

Example: More efficient election in a complete graph

Changes to CompleteElect: works if $F \leq (n - 6) / 2$

- x sends Capture to rF neighbours (in stage 1)
or $(r - 1)F$ neighbours (stage > 1)
- **no waiting** after Warning messages (and no settlement)
- stage s_x increases only when $(r - 1)F$ **Accept messages** have arrived from the current stage
- Capture messages are sent only at the start of a new stage
- termination: if $s_x = (n + 2) / 2F$, then x becomes leader and broadcasts this using TwoSteps

The selectable parameter r gives a messages/time tradeoff:

	<i>Time</i>	<i>Messages</i>
<i>any r:</i>	$O\left(\frac{n}{(r-1)F}\right)$	$O\left(nrF + \frac{nr}{(r-1)} \log \frac{n}{(r-1)F}\right)$
$r = 2:$	$O(n/F)$	$O(nF + n \log(n/F))$

More on complete graphs

Open problem:

Is it possible to elect a leader using $O(nF)$ messages if $F < n - 1$ links can fail?

A much larger total number of failures can also be tolerated: if at most $f < n/2$ incident links at each node may fail ($F < (n^2 - 2n)/2$), consensus can be achieved in $O(n^2)$ messages.

Summary

Ways to work around the Single-Fault Disaster problem:

- **randomisation** works, but gives up certainty
- **failure detection** is a good solution for many computations?
- **pre-execution failures** help, but are not very realistic

Permanent link failures:

- are not a difficult problem if the **edge connectivity** can be increased (i.e., more hardware costs)
- but is the model that only F links can ever fail realistic enough?