Tolerating Transient and Intermittent Failures

Sylvie Delaet and Sebastien Tixeuil

Journal of Parallel and Distributed Computing 62, p. 961-981, 2002
Presented by: Taylor Johnson
ECE598NV, Fall 2009

December 8, 2009

Motivation

Consensus (synchronous)

- Every process has an input and all non-faulty ones must decide on a common value in finite time

in spite of failures	processes (at least)	rounds
f **crash failures**	$f+1$	$f+1$
t **Byzantine failures**	$3t+1$	$t+1$

- Fault model of crash assumes *permanent failure*

Failure occurrence models

Models

- Single: p_i: `...GGGGBGGGGG...`
- Permanent: p_i: `...GGGGGGBBBBBBBBBBB...`

Failure occurrence models

Models

- Single: p_i: `...GGGGBGGGGG...`
- Permanent: p_i: `...GGGGGGBBBBBBBBBB...`
- Question: at what round did p_i become faulty?

Failure occurrence models

Models

- Single: p_i: . . . GGGGBGGGGG . . .
- Permanent: p_i: . . . GGGGGGBBBBBBBBBBB . . .
- Question: at what round did p_i become faulty? Recall lower-bound proof on number of rounds for synchronous consensus with crash failures: $f + 1$ round for a *failure sparse* execution, e.g., $n = 3$, $f = 2$ case: p_1: GGB, p_2: GBB, p_3: GGG
- Transient: . . . BBBBBBGGGGGGGGG . . .
- Intermittent: . . . GGGBBBGGBBGBGGGGBBBB . . .

Methods for fault-tolerance

Methods

- Robustness (masking): always guarantee correct behavior, normally for a small number of failures which may occur frequently (permanent and intermittent)
- Self-stabilization (non-masking): accept temporary violation of correctness, normally when all components behave correctly most of the time (transient)

Census problem

- Informal: eventually the system reaches a global state where each processor knows the identifiers of all processors in the network and their relative distances to it
- Potential uses: leader election (e.g., minimum identifier elected), counting processors, spanning tree
- A configuration c satisfies Census if and only if for any p_i in P, the set of processors, i knows the identifier and distance of all other elements relative to itself in c
- Results: $O(N \times (\log_2 k + \delta_i))$ (for $\Omega(N \times \log_2 N)$) bits per node, where N is number of processors in the system, k is number of possible identifiers, and δ_i is input degree of p_i with $O(D)$ stabilization time, where D is diameter of network
- Comparison: [T. Masuzawa, "A fault-tolerant and self-stabilizing protocol for the topology problem", SSS1995] requires $\Omega(N^2)$ bits at each processor

Assumptions and model

- Message passing, strongly connected network, all processors know their unique identifiers, all processors know their indegree δ_i (links numbered $1, \ldots, \delta_i$), and know from which link a message arrives
- Input neighbors known, not output (e.g., satellites)
- Fair message loss: when infinitely messages are sent by *o* (origin), infinitely messages are received by *d* (destination)
- Finite duplication: every messages sent by *o* may be received by *d* a finite number of times (unknown bound)
- Reordering: messages sent by *o* may be received by *d* in a different order than sent
- Any message not lost is eventually received: if *o* continuously sends the same message to *d*, then eventually it is received
- Each can happen in both the stabilizing phase and stabilized phase

Algorithm preliminaries

- Local memory: $(i_1; i_2; \ldots; i_k)$ of $< identifier, colors >$ pairs where *colors* is an array of booleans of size δ_i, the indegree used to indicate if a link was used for an incoming processor
- Example: local memory at i is: $((j, [100]; q, [010]; t, [001]), (z, [111]))$ where $\delta_i = 3$ and j, q, t are at distance 1 from i and z is at distance 2 from i
- Messages: contents are a list $(i_1; i_2; \ldots; i_k)$
- Example: $((i)(j; q; t)(z))$

Algorithm preliminaries

- Local memory: $(i_1; i_2; \ldots; i_k)$ of $< \textit{identifier}, \textit{colors} >$ pairs where *colors* is an array of booleans of size δ_i, the indegree used to indicate if a link was used for an incoming processor
- Example: local memory at i is: $((j, [100]; q, [010]; t, [001]), (z, [111]))$ where $\delta_i = 3$ and j, q, t are at distance 1 from i and z is at distance 2 from i
- Messages: contents are a list $(i_1; i_2; \ldots; i_k)$
- Example: $((i)(j; q; t)(z))$
- i sent message (since it's first), i believes j, q, t are each direct ancestors of i and z is at distance 2
- Notation: distance from i to j denoted as $d(i, j)$ which is the minimal edges from i to j which is always defined due to strong connectivity; not assuming bidirectional, so $d(i, j) \neq d(j, i)$ possible

Algorithm overview

Steps

- Step 1: check message coherence: if a node receives a message that does not start with a singleton list, message is known erroneous and ignored
- example: $((j; q; t)(k)(m; y)(p; z))$ is ignored since $(j; q; t)$ not singleton
- Step 2: check local memory coherence: a node is incoherent if there exists at least one pair $< identifier, color >$ such that $colors = [00 \ldots 0]$, meaning some information in the local memory was not obtained from an input channel
  - Regularly: if a problem is detected upon time-out, then the local memory is reinitialized
  - On receive: else if a problem is detected upon message receipt, the local memory is completely replaced by the message information

Algorithm overview (cont)

Steps

- Step 3: trust most recent information: messages have more reliable information than local memory since they are more recent, so replace all local memory obtained from the incoming channel
- Example: $m = ((j)(k; l)(m)(p; q; r; i))$ received by i through link 1 and $\delta_i = 2$ means that
  - j is a direct ancestor of i
  - k and l are distance 2 of i and may send through j
  - m is at distance 3
  - p, q, r at distance 4 of i and j obtained this information through m

Algorithm overview (cont)

Step 3 Example

- $m = ((j)(k; l)(m)(p; q; r; i))$ received by i on link 1, $\delta_i = 2$
- Memory: $((j, [10]; q, [01])(k, [10]; e, [01]; w, [01])$ $(m, [01]; y, [11])(p, [10]; z, [01]; h, [10]))$
- On receipt of m: (1) local memory cleared from previous link 1 information, (2) incoming message colored by number of the link (each id $\alpha \in m$ becomes a pair $(\alpha, [10])$), and (3) local memory updated as:
- $((q, [01])(e, [01]; w, [01])(m, [01]; y, [01])(z, [01]))$
- $\leftarrow ((j, [10])(k, [10]; l, [10])(m, [10])$ $(p, [10]; q, [10]; r, [10]; i, [10]))$
- $\rightarrow ((j, [10]; q, [01])(k, [10]; e, [0, 1], w, [01]; l, [10])$ $(m, [11]; y, [01])(p, [10]; z, [01]; q, [10]; r, [10]))$

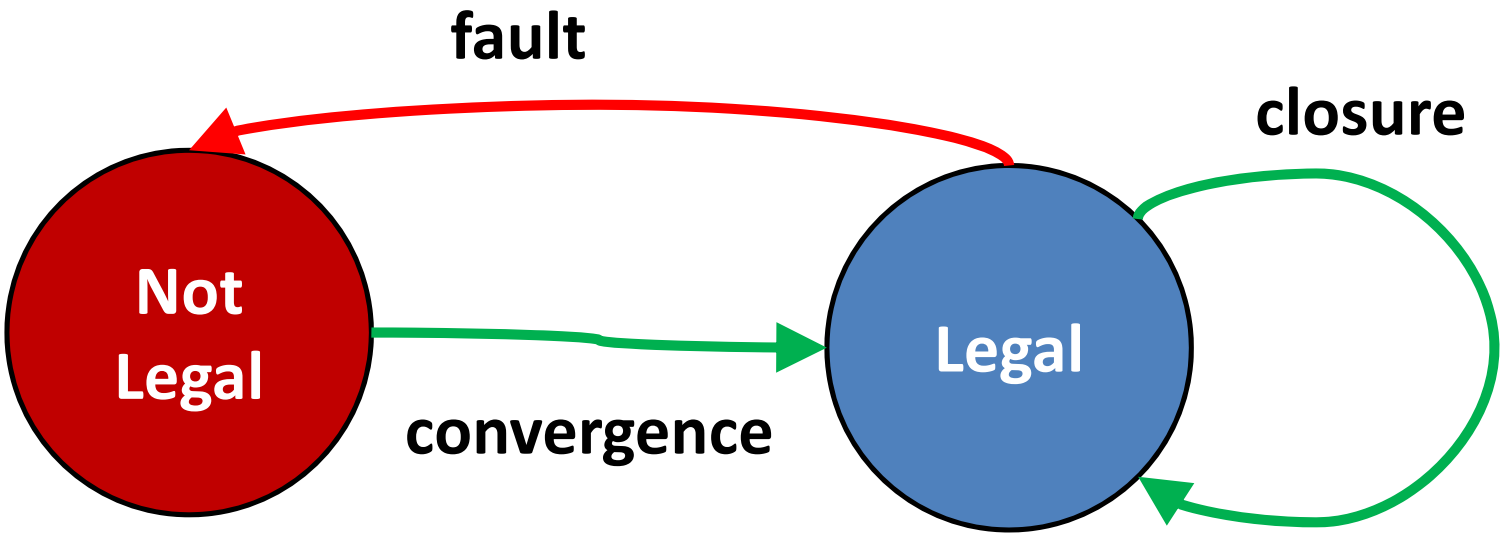
Correctness intuition

- Fair loss is compensated by infinite spontaneous retransmission by processors of their current knowledge
- Finite duplication tolerance due to the algorithm being *idempotent* in the sense that, if a processor receives the same message twice from the same incoming link, the second message does not modify knowledge of that processor
- Desequencing can be considered as a change in the relative speed of two message towards complete knowledge of the network: each message independently becomes more accurate and complete, so that relative order is insignificant

Proof preliminaries

- A computation e satisfies Census if and only if every configuration c in e satisfies Census
- A set of configurations $B \subset \mathcal{C}$ is *closed* if for any $b \in B$ any possible computation of the system whose b is an initial configuration only contains configurations in B
- A set of configurations $B_2 \subset \mathcal{C}$ is an *attractor* for a set of configurations $B_1 \subset \mathcal{C}$ if for any $b \in B_1$ and any possible computation of the system whose initial configuration is b, the computation contains a configuration of B_2
- System if self-stabilizing for a specification A if there exists a non-empty set of configurations $L \subset \mathcal{C}$ such that (1) closure: any computation whose initial configuration is in L satisfies A, and (2) convergence: L is an attractor for C

Self-stabilizing algorithms



Self-stabilizing census

Want: Independent of initial configuration of network channels (non-infinitely full though) and local memories, every processor ends up with a local memory that reflects the contents of the network, even with the unreliable communication media

Proof steps

- Closure: show that under normal operation (no faults) the system remains legitimate
- Measure on messages in network and local memory: either distance between current form of message and canonical form (optimal knowledge of network), or between current local memory and canonical form (local optimal knowledge): gives weight of a configuration, where weight 0 is *legitimate configuration*
- Convergence: Show that after a set of sends and receives, the weight of a configuration decreases
- Convergence: Induct to show this phenomenon appears infinitely often and that the weight of a configuration reaches 0, that is, a configuration where each message is correct (optimal) and where each local memory is correct

Closure preliminaries

- Define: age of i denoted by χ_i is the greatest distance $d(j, i)$ for any j in the graph, so the diameter is: $\max_i \chi_i = D$
- Define: the canonical form of a message circulating on a link between j and i is the list of lists starting with the singleton list (j) followed by the χ_j lists of ancestors of j at distance between 1 and χ_j
- Define: *Llc* to denote a list of lists of pairs $< identifier, colors >$
- Define: the canonical form of node i's local memory is the list of lists of pairs *Llc* of the χ_i lists of pairs $< identifier, colors >$ such that: *identifiers*(*Llc*) is the list of the χ_i lists on ancestors of i at distance 1 to χ_i, and if a shortest path from j to i passes through p^{th} input channel of i, then *colors* associated with j in *Llc* has $colors[p] = 1$

Closure

Proposition

The canonical form of node i's local memory and that of its incoming and outgoing channels are coherent

Proof.

- If local memory canonical, then send trivially produces canonical message
- If message canonical, then wind up with canonical memory (recall example replacing memory on receipt of a message)

■

Closure (cont)

Corollary

The set of legitimate configurations is closed.

Proof.

Starting from a configuration where every message and every local memory is canonical, none of the local memories are modified (recall example replacing memory), and none of the emitted messages are non-canonical. ■

Configuration weight

- Measure on messages and local memories: integer of $D + 2$ base-3 digits where D is graph diameter
- Define: Let m be a message with canonical message m_c. The weight of m is the integer written using $D + 2$ base-3 digits whose α^{th} digit is:
  - 0 if $m[\alpha] = m_c[\alpha]$
  - 1 if $\neg(m[\alpha] \subseteq m_c[\alpha])$
  - 2 if $m[\alpha] \neq m_c[\alpha]$
- Biggest weight (every part of message wrong): $3^{D+2} - 1$
- Optimal weight (every path of message right): 0 when $m = m_c$
- Example: $m_c = ((j)(k; e; w)(q)(i; p))$

Convergence base case

Lemma

In any configuration, only messages of weight lower than 3^{D+1} *may be sent*

Convergence inductive case

Lemma

Assume $\alpha \geq 1$. *The set of configurations whose weight is strictly lower than* $3^{\alpha-1}$ *is an attractor for the set of configurations whose weight is strictly lower than* 3^{α}

Proposition

The set of configurations whose weight is 0 *is an attractor for the set of all possible configurations*

Self-stabilizing

Theorem

The algorithm presented to solve Census is self-stabilizing

Thank you and questions