Fault-Tolerant Computing

Software Design Methods
About This Presentation

This presentation has been prepared for the graduate course ECE 257A (Fault-Tolerant Computing) by Behrooz Parhami, Professor of Electrical and Computer Engineering at University of California, Santa Barbara. The material contained herein can be used freely in classroom teaching or any other educational setting. Unauthorized uses are prohibited. © Behrooz Parhami

<table>
<thead>
<tr>
<th>Edition</th>
<th>Released</th>
<th>Revised</th>
</tr>
</thead>
<tbody>
<tr>
<td>First</td>
<td>Dec. 2006</td>
<td>Nov. 2007</td>
</tr>
</tbody>
</table>
Agreement and Adjudication
Nov. 2007 Agreement and Adjudication Slide 4

Software Specification

Abe, I just noticed – our marriage license has an expiration date! And it’s today! Do you know anything about this, Abe? . . . Abe? . . . Abe?

“Next case: the Internet economy versus millions of investors who should have known better.”

1. As Management Requested It
2. As Specified in the Project Request
3. As Designed By The Senior Analyst
4. As Produced By The Programmers
5. As Installed
6. What The User Wanted

“Drink 8 Glasses of Water Everyday”

“May I see the Manager? I have a small problem..."

“Would you like to make it a large, for only twenty-five cents?"

“It was the only mission statement everyone could agree on.”

© 2000 Randy Glasbergen. www.glasbergen.com

© 2002 Steve S.
Multilevel Model of Dependable Computing

Legend:
- Entry
- Deviation
- Remedy
- Tolerance

Level → Component Logic Information System Service Result

Ideal → Defective → Faulty → Erroneous → Malfunctioning → Degraded → Failed

Unimpaired → Low-Level Impaired → Mid-Level Impaired → High-Level Impaired
Introduction to Voting

Voting schemes and associated terminology in dependable computing were originally derived from concepts in sociopolitical elections.

With inputs drawn from a small set of integers, the similarity between the two domains is strong.

**Example:** Radar image analysis used to classify approaching aircraft type as civilian (0), fighter (1), bomber (2).

If three independent units arrive at the conclusions \( \langle 1, 1, 2 \rangle \), then the presence of a fighter plane may be assumed.

Option or candidate 1 “wins” a majority of the vote.

With a large or infinite input domain, voting takes on a new meaning.

**Example:** There is no strict majority when distance of an approaching aircraft, in km, is indicated as \( \langle 12.5, 12.6, 14.0 \rangle \), even though the good agreement between 12.5 and 12.6 could lead to a 12.55 km estimate.
A General Framework for Voting

Virtually all voting schemes of practical interest can be formulated in terms of the generalized weighted voting model, as follows:

Given $n$ input data objects $x_1, x_2, \ldots, x_n$ and associated nonnegative real votes $v_1, v_2, \ldots, v_n$, with $\sum v_i = V$, compute output $y$ and its vote $w$ such that $y$ is “supported by” a set of input objects with votes totaling $w$, where $w$ satisfies a condition associated with the voting subscheme.

Possible voting subschemes:

- **Unanimity** $w = V$
- **Majority** $w > V/2$
- **Supermajority** $w \geq 2V/3$
- **Byzantine** $w > 2V/3$
- **Plurality** $(w$ for $y) \geq (w$ for any $z \neq y)$
- **Threshold** $w > a$ preset lower bound

\[ \langle x_1, v_1 \rangle \rightarrow \langle x_2, v_2 \rangle \rightarrow \ldots \rightarrow \langle x_n, v_n \rangle \rightarrow \langle y, w \rangle \]
A Taxonomy of Voting Schemes

One can classify generalized weighted voting schemes into $2^4 = 16$ categories based on dichotomies associated with input data (the $x_i$s), output data ($y$), input votes (the $v_i$s), and output vote ($w$).

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>Exact/Inexact</td>
</tr>
<tr>
<td>Vote</td>
<td>Oblivious/Adaptive</td>
</tr>
</tbody>
</table>

First entry in each box is the simpler of the two terms such as “threshold voting” stands for 8 different methods.

Input objects inflexible, or representing flexible “neighborhoods”

Input weights set at design time, or allowed to change dynamically (adjustable/variable)

Total support from a subset of inputs (quorum), or shades of support from all inputs

Support exceeds a lower bound, or is max over all possible outputs
Generalized Median Voting

To find the median of a set of numbers, repeatedly remove largest and smallest numbers, until only one or two remain.

If we replace “largest and smallest numbers” by “the two inputs that are furthest apart,” we can use an arbitrary distance metric in our screening.

A distance metric is any metric (mapping of pairs of inputs into real values) that satisfies the three conditions:

- **Isolation**: \( d(x, y) = 0 \) iff \( x = y \)
- **Symmetry**: \( d(x, y) = d(y, x) \)
- **Triangle inequality**: \( d(x, y) + d(y, z) \geq d(x, z) \)

For example, the Hamming distance satisfies these conditions.
Approximate Voting

The notion of an input object “supporting” a particular output (akin to a hypothesis supporting an end result or conclusion) allows us to treat approximate and exact voting in the same way.

**Example 1:** Input objects are points in the 2D space and the level of “support” between them is a function of their Euclidean distance.

**Example 2:** Input objects are conclusions of character recognizers as to the identity of a character, with varying degrees of mutual support.

![Diagram illustrating median selection from the maximal mutually supportive set of inputs.](image-url)
Approval Voting

Approval voting was introduced to prevent the splitting of votes among several highly qualified candidates from leading to the election of a less qualified candidate in plurality voting.

In approval voting, a voter divides the set of candidates into two subsets of “qualified” and “not qualified” and indicates approval of the first subset.

In the context of computing, approval voting is useful when a question has multiple answers or when the solution process is imprecise or fuzzy.

Example question: What is a safe setting for a particular parameter in a process control system?

When the set of approved values constitute a continuous interval of real values, we have “interval” inputs and “interval” voting.
Interval Voting

Inputs to the voting process are intervals, representing approved values.

How should the voting result be derived from the input intervals?

Various combining rules can be envisaged.

If there is overlap among all intervals, then the decision is simple.

Depending on context, it may make sense to consider greater levels of approval near the middle of each interval or to associate negative approval levels outside the approved intervals.
The Impossibility of Perfect Voting

Properties of an ideal voting scheme:

1. No big brother
   (voters free to express preferences)
2. Independence of irrelevant alternatives
   (preference for one candidate over
   another is independent of all others)
3. Involvement
   (every outcome is possible)
4. No dictatorship or antidictatorship
   (outcome not always conforming to,
   or opposite of, one voter’s view)

Arrow’s Theorem:
No voting scheme exists that satisfies all four conditions

True majority voting scheme:
Each voter orders all the candidates; no circular preference allowed
Choose a candidate who beats every other one in pairwise competitions
(both simple majority and plurality rules fail to choose a candidate)
Distributed Voting: The Agreement Problem

**Problem:** Derive a highly reliable value from multiple computation results or stored data replicas at multiple sites.

**Key challenge:** Exchange data among nodes so that all healthy nodes end up with the same set of values; this guarantees that running the same decision process on the healthy nodes produces the same result.

Errors are possible in both data values and in their transmission between sites.

Agreement algorithms generally use multiple rounds of communication, with values held at each site compared and filtered, until the set of values held at all sites converge to the same set.
Byzantine Failures in Distributed Voting

Three sites are to collect three versions of some parameter and arrive at consistent voting results

Assume median voting
The Interactive Consistency Algorithm

ICA(0) [no failure]
1. The transmitter sends its value to all other \( n - 1 \) nodes
2. Each node uses the value received from the transmitter, or a default value \( \Phi \) if it received no value

ICA(\( f \)), \( f > 0 \) [\( f \) failures]
1. The transmitter sends its value to all other \( n - 1 \) nodes
2. Let \( v_i \) be the value received by node \( i \) from the transmitter, or a default value \( \Phi \) if it received no value; node \( i \) then becomes the transmitter in its own version of ICA(\( f - 1 \)), sending its value to \( n - 2 \) nodes
3. For each node \( i \), let \( v_{i,j} \) be the value it received from node \( j \), or a default value \( \Phi \) if it received no value from node \( j \). Node \( i \) then uses the value \( \text{majority}(v_{i,1}, v_{i,2}, \ldots, v_{i,i-1}, v_{i,i+1}, \ldots, v_{i,n}) \)

\( O(n^{f+1}) \) messages needed, in \( f + 1 \) rounds, to tolerate \( f \) Byzantine failures
Building upon Consensus Protocols

The same messages are delivered in the same order to all participating nodes.

All healthy participants arrive at vectors with a majority of elements correct.

All healthy participants arrive at vectors with correct value for every healthy node.

Agreeing on one of two values, 0 or 1.

If source and destination are healthy, message is eventually delivered unmodified.

Message from a good node is eventually delivered to all good nodes unmodified.

Correctness and Performance of ICA

**Theorem 1:** With ICA\(f\), all nonfailed nodes will agree on a common value, provided that \(n \geq 3f + 1\) (proof is by induction on \(f\))

ICA works correctly, but it needs an exponential number of messages:
\[
(n-1) + (n-1)(n-2) + (n-1)(n-2)(n-3) + \ldots + (n-1)(n-2)\ldots(n-m)
\]

More efficient agreement algorithms exist, but they are more difficult to describe or to prove correct; \(f + 1\) rounds of message exchange is the least possible, so some algorithms trade off rounds for \# of messages

**Theorem 2:** In a network \(G\) with \(f\) failed nodes, agreement is possible only if the connectivity is at least \(2f + 1\)

![Diagram](image)
The Two Generals Problem

Two generals lead divisions of an army camped on the mountains on the two sides of an enemy-occupied valley.

The two divisions can only communicate via messengers.

We need a scheme for the generals to agree on a common attack time, given that attack by only one division would be disastrous.

Messengers are totally reliable, but may need an arbitrary amount of time to cross the valley (they may even be captured and never arrive).

G1 decides on \( T \), sends a messenger to tell G2.

G2 acknowledges receipt of the attack time \( T \).

G2, unsure whether G1 got the ack (without which he would not attack), will need an ack of the ack!

This can go on forever, without either being sure.
The Byzantine Generals Problem

A general and $n - 1$ lieutenants lead $n$ divisions of the Byzantine army camped on the outskirts of an enemy city.

The $n$ divisions can only communicate via messengers.

We need a scheme for the generals to agree on a common plan of action (attack or retreat), even if some of the generals are traitors who will do anything to prevent loyal generals from reaching agreement.

The problem is nontrivial even if messengers are totally reliable.

With unreliable messengers, the problem is very complex.
Byzantine Generals with Reliable Messengers

With $f$ Byzantine failures, $\geq 3f + 1$ nodes needed to reach agreement
Byzantine Resiliency

To tolerate $f$ Byzantine failures:

We need $3f + 1$ or more FCRs (fault containment regions)

FCRs must be interconnected via at least $2f + 1$ disjoint paths

Inputs must be exchanged in at least $f + 1$ rounds

Corollary 1: Simple 3-way majority voting is not Byzantine resilient

Corollary 2: Because we need $2f + 1$ good nodes out of a total of $3f + 1$ nodes, a fraction $(2f + 1)/(3f + 1) = 2/3 + 1/(9f + 3)$ of the nodes must be healthy

This is greater than a supermajority ($2/3$) requirement