FTCS 93 June 1993

A Formally Verified Algorithm for Interactive Consistency Under a Hybrid Fault Model*

Patrick Lincoln and John Rushby

Computer Science Laboratory
SRI International
Menlo Park CA USA

Summary

- What we have done
  - Found a flaw in an algorithm for achieving Interactive Consistency under a hybrid fault model
  - Corrected it
  - Formally verified it

- Why we think it is interesting
  - Interactive consistency is an important problem
  - The hybrid fault model is very attractive
  - The algorithm is practical and useful
  - Illustrates fallibility of informal proofs
  - Demonstrates feasibility of mechanically-checked verification

Overview

- Context
- Interactive Consistency and Byzantine Agreement
- The classical Oral Messages (OM) algorithm for Byzantine Agreement
- The hybrid fault model
- The flawed algorithm
- The repaired algorithm
- The formal specification and verification
- Conclusions

Context: Fault-Tolerant Architectures for Flight Control

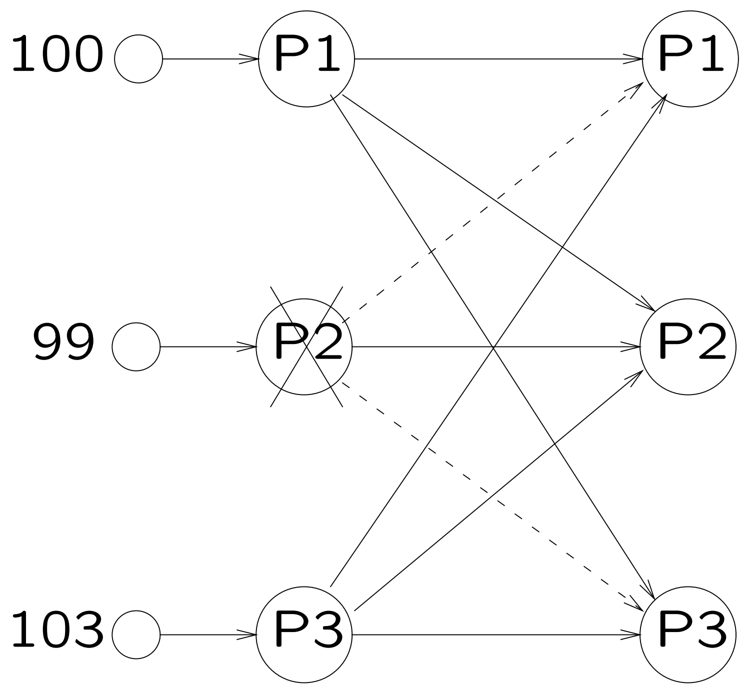
There are two main ways to organize the redundant computing channels of a fault-tolerant flight-control system

Asynchronous: run the channels fairly independently and use averaging and threshold voting to mask faults—difficult to predict behavior under all combinations of clock drift, sensor noise, channel failure

Synchronous: run the channels in lock-step, distribute sensor data to all channels, and use exact-match voting—behavior is predictable, but basic algorithms are difficult

Our interest: use of formal methods to develop and analyze algorithms and architectures for synchronous fault-tolerant systems

The Problem of Interactive Consistency
(aka. Source Congruence)



values received			mid-value
100	200	103	103
100	99	103	99
100	23	103	100

The Byzantine Generals Problem

- The real problem is *interactive consistency* (IC): every channel has a (set of) values that must be communicated reliably to every other channel

- The *Byzantine Generals* (BG) version is a little easier to describe: a Commanding General has an order that must be conveyed to a set of Lieutenant Generals

  (Interactive consistency is just iterated Byzantine Generals)

The Byzantine Generals Problem: Requirements

There are n generals, of whom as many as m can be faulty (including the Commander)

BG1: nonfaulty lieutenants agree on the value received from the Commander

BG2: if the Commander is nonfaulty, the value received by every nonfaulty lieutenant is the value he sent

Make *no assumptions at all* about the behavior of faulty generals

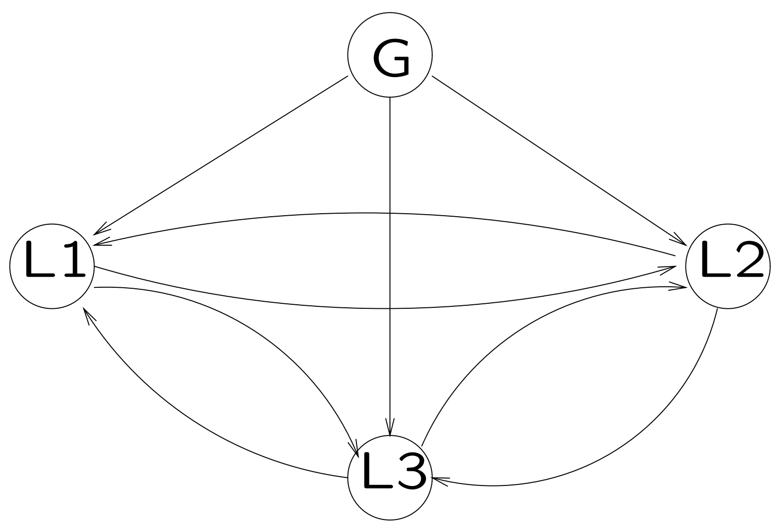
The Oral Messages Algorithm for BG

- Requires $n > 3m$
- And $m + 1$ rounds of message exchange
- So need four channels and two rounds to withstand a single fault

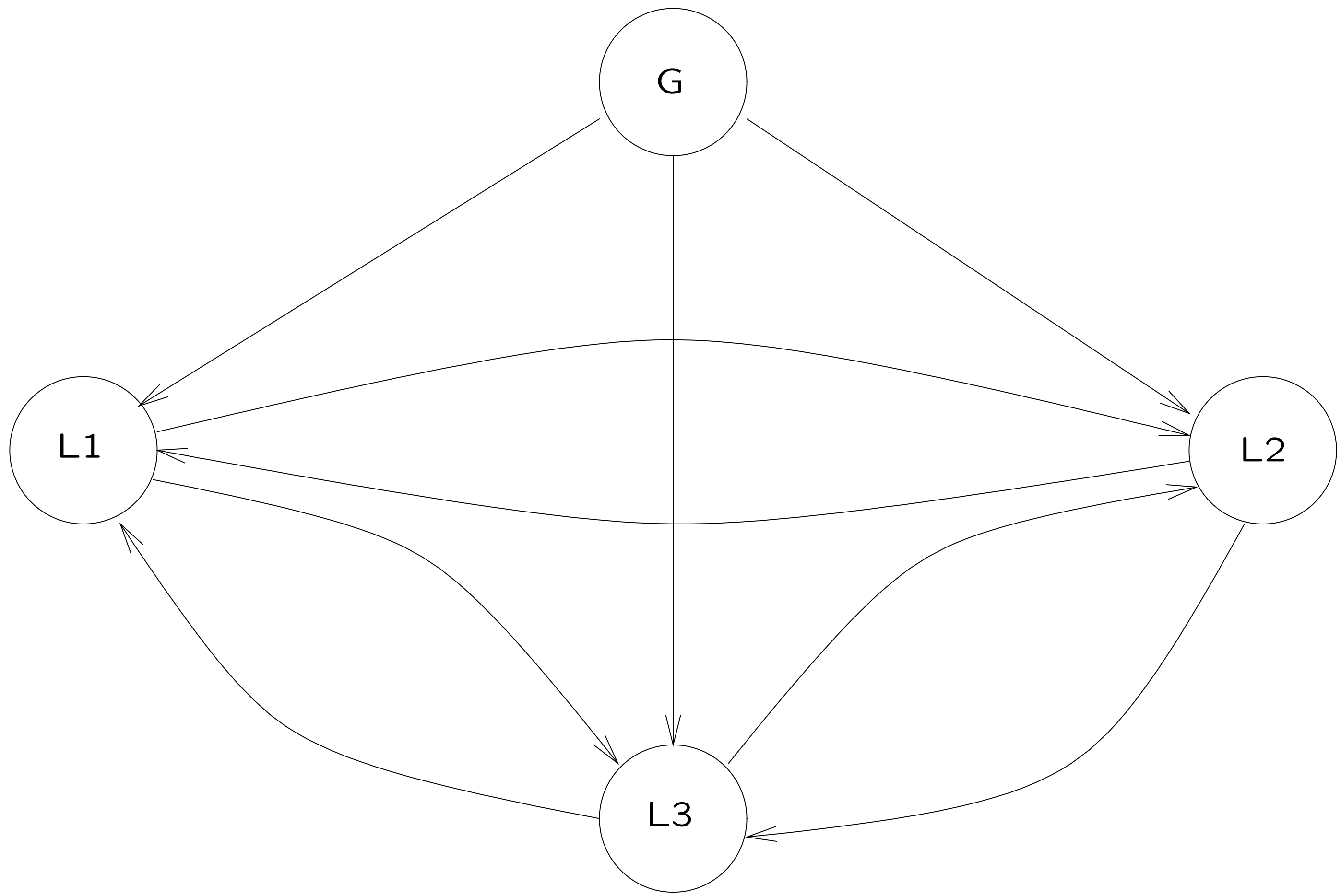
The Oral Messages Algorithm

- The Commander sends value to each lieutenant
- If $m = 0$, each lieutenant accepts the value he receives

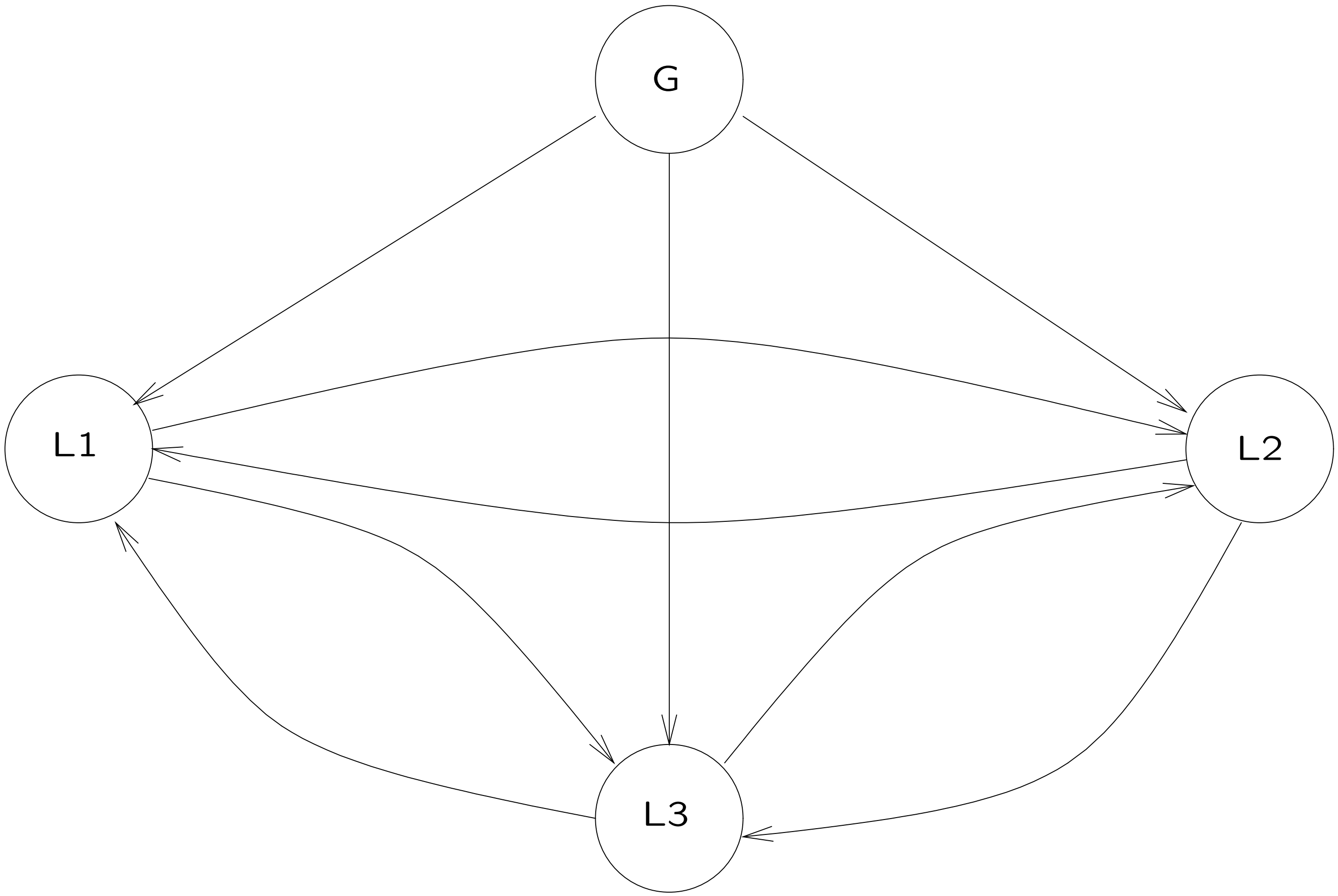
  Otherwise, each lieutenant takes the part of the general in $OM(m-1)$ to send value received to all other lieutenants
- Each lieutenant takes majority vote of the values received directly from Commander and via the other lieutenants



Example: Oral Messages With Faulty Commander



Example: Oral Messages With A Faulty Lieutenant



Properties of the Oral Messages Algorithm

- Optimal in number of processors required ($n = 3m + 1$) to withstand given number (m) of faults

- Suboptimal in terms of number of rounds (no early stopping) and number of messages exchanged (exponential in m)

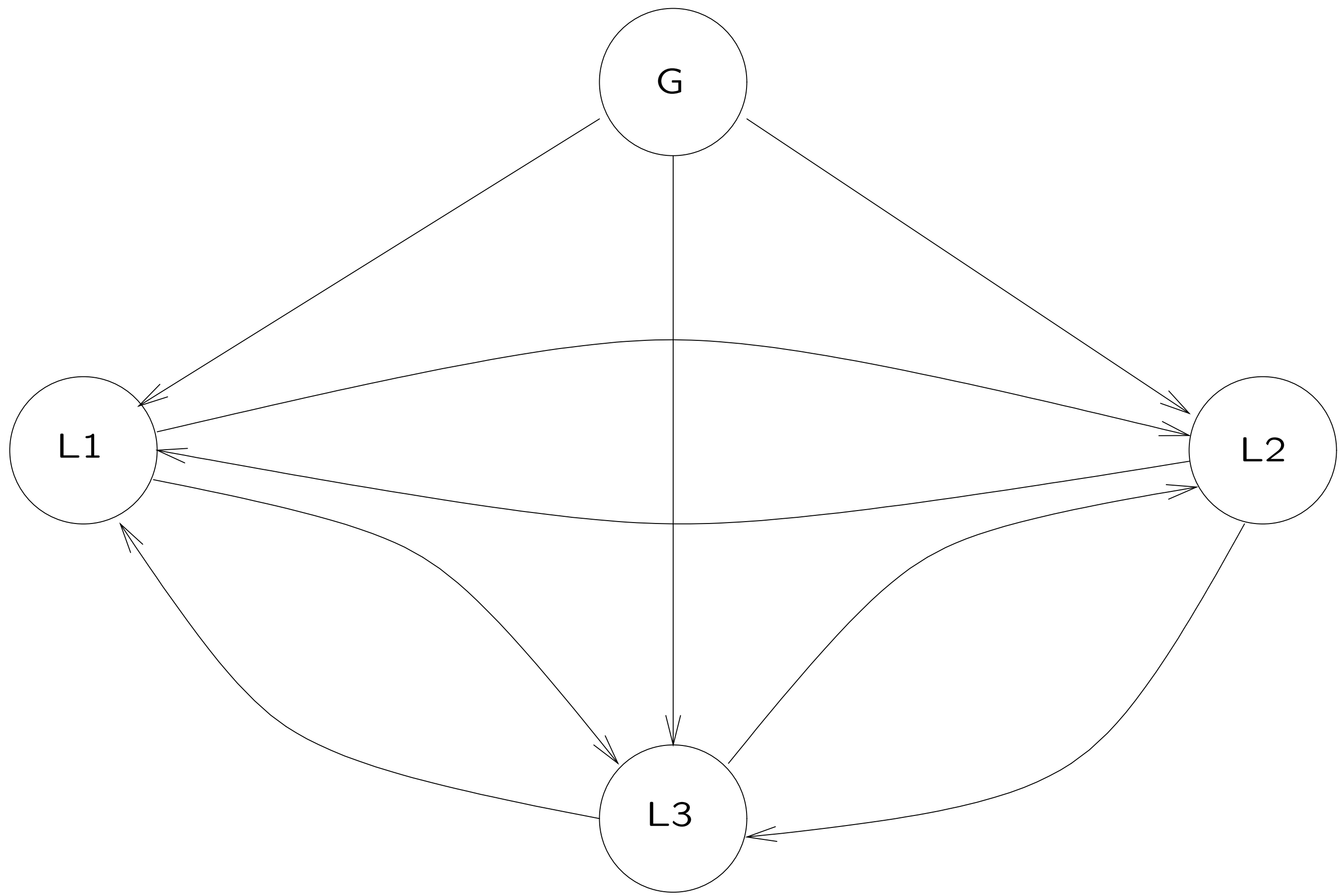
  Adequate for cases of practical interest ($m \leq 2$, $n \leq 7$)

- But treats all faults as "worst case": makes no special provision for "simple" faults

  Withstands *fewer* simple faults than less sophisticated algorithms

  5 and 6 channels provide no benefit compared to 4

Example: Oral Messages With Two Crashed Lieutenants



Fault Models: Extreme positions

- Byzantine approach: components are either working correctly or have failed in some unknown manner
  - Cannot be defeated by unanticipated fault mode
  - But can be defeated by moderate number of “simple” faults
- FMEA approach: components can fail in (many) known ways; design countermeasures for each one (and their combinations)
  - May be defeated by unanticipated fault mode
  - But can be optimized to maximize resilience to certain faults

Hybrid Fault Models

- Goal is to maximize both the *modes* of fault that can be tolerated, and the *number*
- Include the arbitrary (Byzantine) mode
  - So cannot be defeated by unanticipated *kind* of fault
- Plus a couple of common, simpler fault modes
  - To maximize the *number* of faults (of those kinds) that can be tolerated

A Hybrid Fault Model

- Thambidurai and Park (then of Allied Signal) introduced a hybrid fault model with three fault classes:
  - Arbitrary (Byzantine) (asymmetric malicious)
  - Symmetric (symmetric malicious)
  - Manifest (crash) (benign)

- They exhibited an m-round Interactive Consistency Algorithm that can tolerate a arbitrary, s symmetric and c manifest faults simultaneously, provided $a \leq m$ and

$$n > 2a + 2s + c + m$$

(classically, $a = m$, and $s = c = 0$, so $n > 3m$ as usual)

Benefits of Hybrid Algorithm

- Consider 6 channels,
- $OM(1)$ can withstand a single Byzantine fault
- Hybrid algorithm can withstand the following combinations

Number of Faults		
Arbitrary (a)	Symmetric (s)	Manifest (c)
1	1	0
1	0	2
0	2	0
0	1	2
0	0	4

Hybrid Version of OM Algorithm (Algorithm Z)

- Whenever a detectably bad (or no) value is received, replace it by distinguished value E (for error)

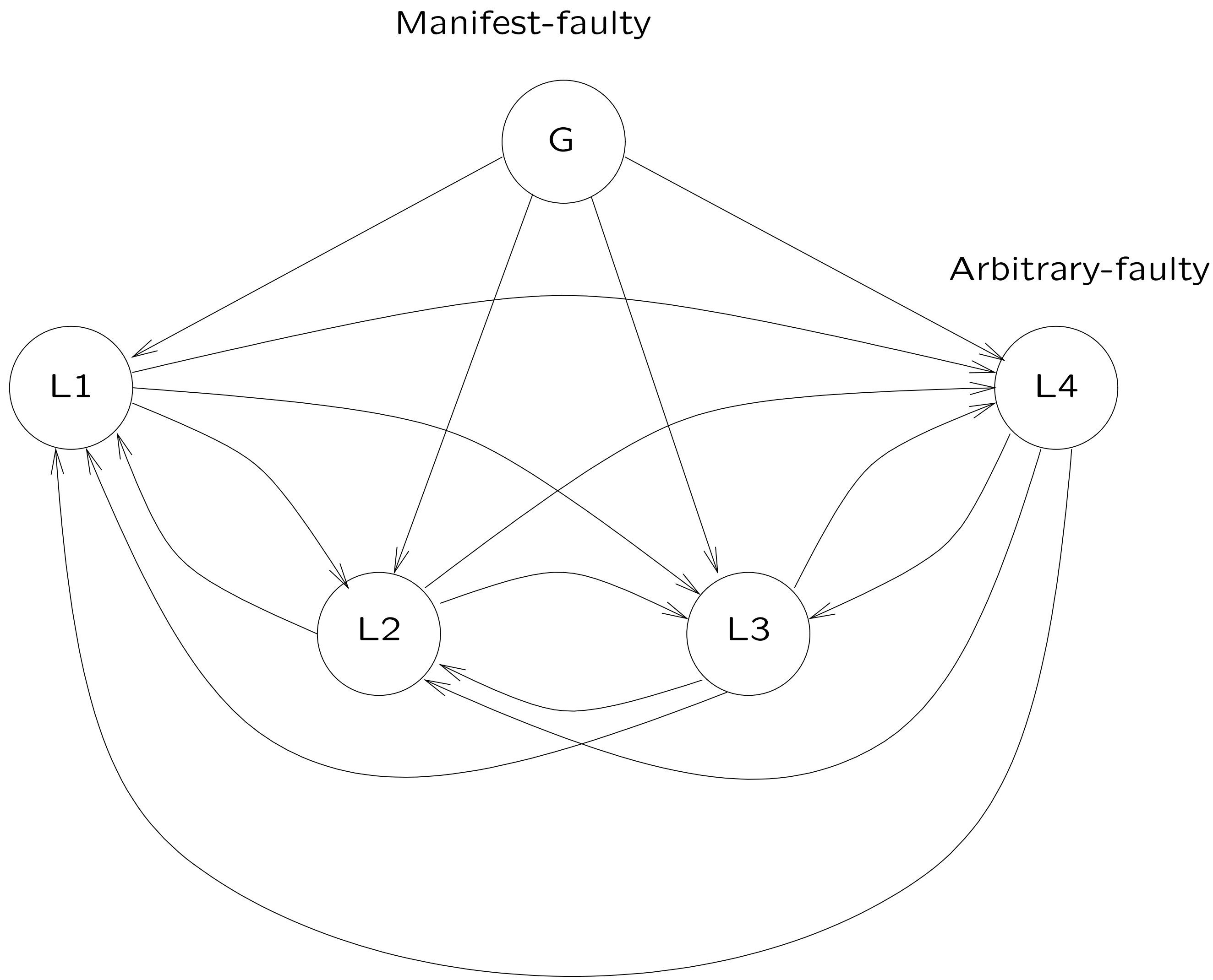
- Ignore E when constructing majority vote

- Published in SRDS 1988, with detailed proof of correctness

- Has a bug

- Found by us while preparing to formally specify and verify the algorithm

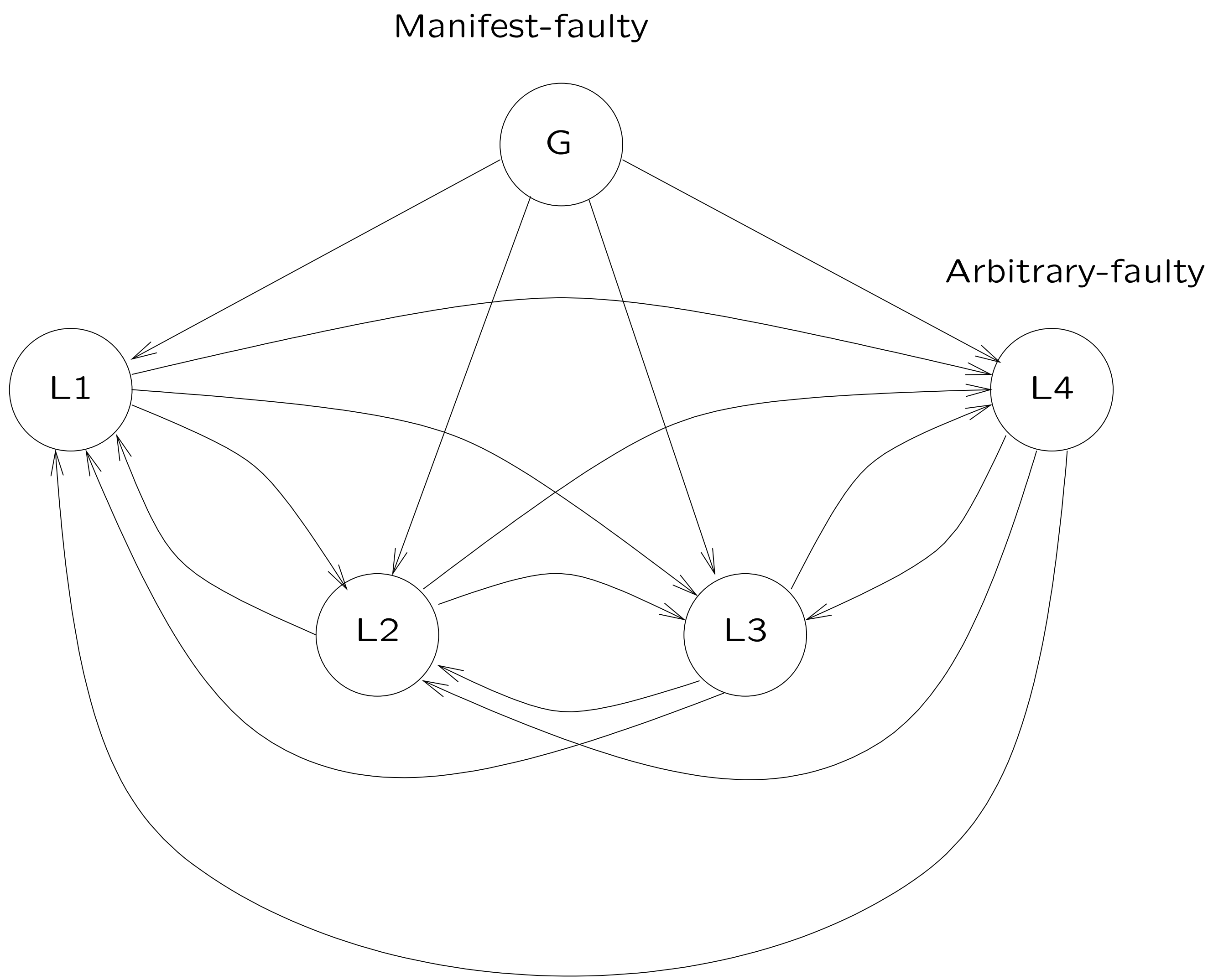
The Flaw in Algorithm Z



Repaired Hybrid Version of OM

- Need two distinguished values: E and RE (reported error)
- Detectably bad or missing values from the Commander are noted and passed on as RE
- Only E ignored in majority vote; if RE wins vote, reported as E
- Verified by hand (by us)
- Also has a bug (though correct for $m = 1$)

The Repaired (Though Still Incorrect) Algorithm



Correct Hybrid Version of OM (Algorithm OMH)

Found flaw in "repaired" algorithm while attempting formal verification

Discipline of formal methods eventually led us to discover a correct algorithm:

- Need m levels of reported error: RE, $\mathrm{R}^2\mathrm{E}$ etc.
  (and let E be $\mathrm{R}^0\mathrm{E}$ for consistency)
- If receive $\mathrm{R}^i\mathrm{E}$ from (recursive) Commander, pass on as $\mathrm{R}^{i+1}\mathrm{E}$
- Ignore E in majority vote
- If vote yields $\mathrm{R}^j\mathrm{E}$, selected value is $\mathrm{R}^{j-1}\mathrm{E}$
- It doesn't matter if $\mathrm{R}^i\mathrm{E}$, $i \geq 1$ is an ordinary value!
  (But E must be distinct)

Formal Methods

Formal Specification: Use of notations derived from formal logic to describe

- *Assumptions* about the world in which a system will operate
- *Requirements* that the system is to achieve
- A *design* to accomplish those requirements

Formal Verification: Use of methods from formal logic to

- *Analyze* specifications for certain forms of consistency, and completeness
- *Test* specifications by posing challenges
- *Prove* that the design will satisfy the requirements, given the assumptions
- *Prove* that a more detailed design *implements* a more abstract one

Formal Specification and Verification of OMH

- Formally specified OMH (as a recursive function)
- And formally specified the requirements BG1 and BG2 (modified for the hybrid case)
- Developed a mechanically-checked proof that OMH satisfies those requirements
- Performed by Pat Lincoln using PVS
- Took about two weeks to develop correct algorithm and attendant formalization and mechanically-checked proofs
- Interactive construction and checking of the proof takes about two hours (and a few minutes to rerun)
- Details of the formal verification described elsewhere (CAV93)
- Seriously doubt could get this right without mechanized assistance