Hybrid Fault Algorithms

Patrick Lincoln, SRI International

In Association With:

J. Rushby SRI

N. Suri Allied Signal

C. Walter (then) Allied Signal

Sponsored by the NASA Langley Research Center under contract NAS1 18969

Application: Digital Flight Control

- **Extreme Reliability (10^{-9} system failures/hour)**
- **Synchronous Systems**
- **NASA RCP**

Byzantine Fault Tolerant Machine Designs

- SRI's SIFT (early 80's) was the first; used 70% CPU cycles on overhead for voting etc.
- Allied-Signal's MAFT (late 80's) used hardware assistance; numerous innovations
- Draper Lab's Asymmetric FTP: less hardware
- Others: TU Vienna MARS, AIPS, FTTP...

The Algorithms Used are Complex
Single Points of Failure

General Points About Verification

- **Informal Proofs are Unreliable:**
  In the past, have Found and Corrected Flaws in Published Algorithms and Proofs.
- **PVS is an Effective Tool:**
  Emphasis on Minimizing Human Effort
- **Proof Reuse Reduces Cost**
- **Still Expensive to Prove Anything Interesting**

General Points About Algorithms

- Slight Modifications to Traditional Algorithms
  - Can Provide Much Greater Reliability
  - Can Destroy Correctness

- Hybrid Approach Widely Applicable
  - Clock Synchronization
  - Byzantine Agreement
  - Diagnosis

Byzantine Agreement

Set of n Generals (Processors)

m of them are Traitors (faulty)

All Good Generals Must Agree on Plan of Attack
All Good Processors Must Agree on Sensor Value

Make NO Assumptions about Behavior of Traitors
Make NO Assumptions about Behavior of Faults

Assume One Commander, Others Are Lieutenants
Assume One Transmitter, Others Are Receivers

The Byzantine Generals Problem: Requirements

Agreement: Nonfaulty Lieutenants Agree on the Value Received from the Commander

Validity: If the Commander is Nonfaulty, the Value Received by Nonfaulty Lieutenants is the Value Actually Sent

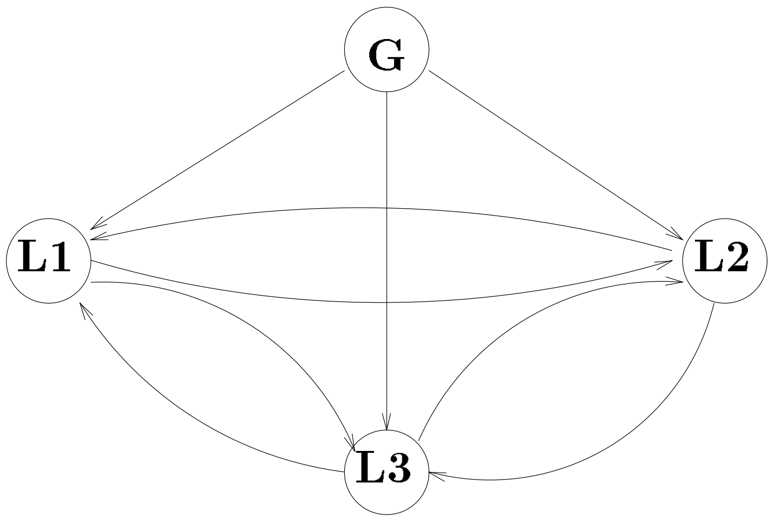
The Oral Messages Algorithm (OM) (Lamport, Shostak, Pease 1982)

- **The Commander Sends Value to Each Lieutenant**

- **If $r = 0$, Each Lieutenant Accepts A Value**

  Otherwise, each lieutenant plays the part of the general in $OM(r-1)$ to send value received to other lieutenants

- **Each Lieutenant Takes Majority Vote of the Values Received**



Properties of OM

- **For Agreement and Validity Requires $n > 3a$ (Best Possible Bound)**
- $r + 1$ **rounds of message exchange**
- **Exponential Number of Messages (Moses 93)**
- **Need Four Channels to Withstand a Single Fault**
- **Formally verified (Bevier & Young '92, Rushby '92, Shankar '93)**

Hybrid Fault Models

- **Extreme Positions:**
  - **Byzantine Approach: Components are Perfect or Failed in Unknown Manner**
  - **FMEA Approach: Components Fail in (many) Known Ways; Design Countermeasures for each (and combinations)**
- **Allied Signal's Hybrid Fault-Models:**
  Include Byzantine Case, Plus a Few Common Cases (Fail-Stop, Stuck-At, $\cdots$)

Formalizing Hybrid Fault Models: Motivation

- **Maximum Assurance**
- **Maximum Reliability**
- **Minimum Hardware**

Algorithms for Hybrid Fault Models

- **Extend Byzantine Fault-Tolerant Algorithms to Deal Better With NonByzantine Faults**

- **Survive a Few Bad Faults**
  or a Lot of Simple Faults with the Same Protocol

- **Allied Signal's Reliability Analysis:**
  This Provides Superior Reliability
  (McElvany-Hugue '93)

A Hybrid Fault Model

- **Allied Signal (Thambidurai and Park '88)**
  - **(a) Arbitrary (Byzantine, Asymmetric)**
  - **(s) Symmetric (Value, Stuck-At)**
  - **(c) Crash (Manifest, Stopped, Out-of-Bounds)**
  - **(g) Good (Nonfaulty, OK)**

Requirements for Hybrid Agreement

Agreement: All nonfaulty receivers agree on the value ascribed to the transmitter.

Validity: If receiver q is nonfaulty, the value it ascribes to the transmitter is

- The correct value if the transmitter is nonfaulty.
- The value actually sent to all receivers if the transmitter is symmetric-faulty.
- The distinguished value E if the transmitter is manifest-faulty.

Allied Signal's Algorithm Z
(Thambidurai and Park 1988)

- **Like OM**

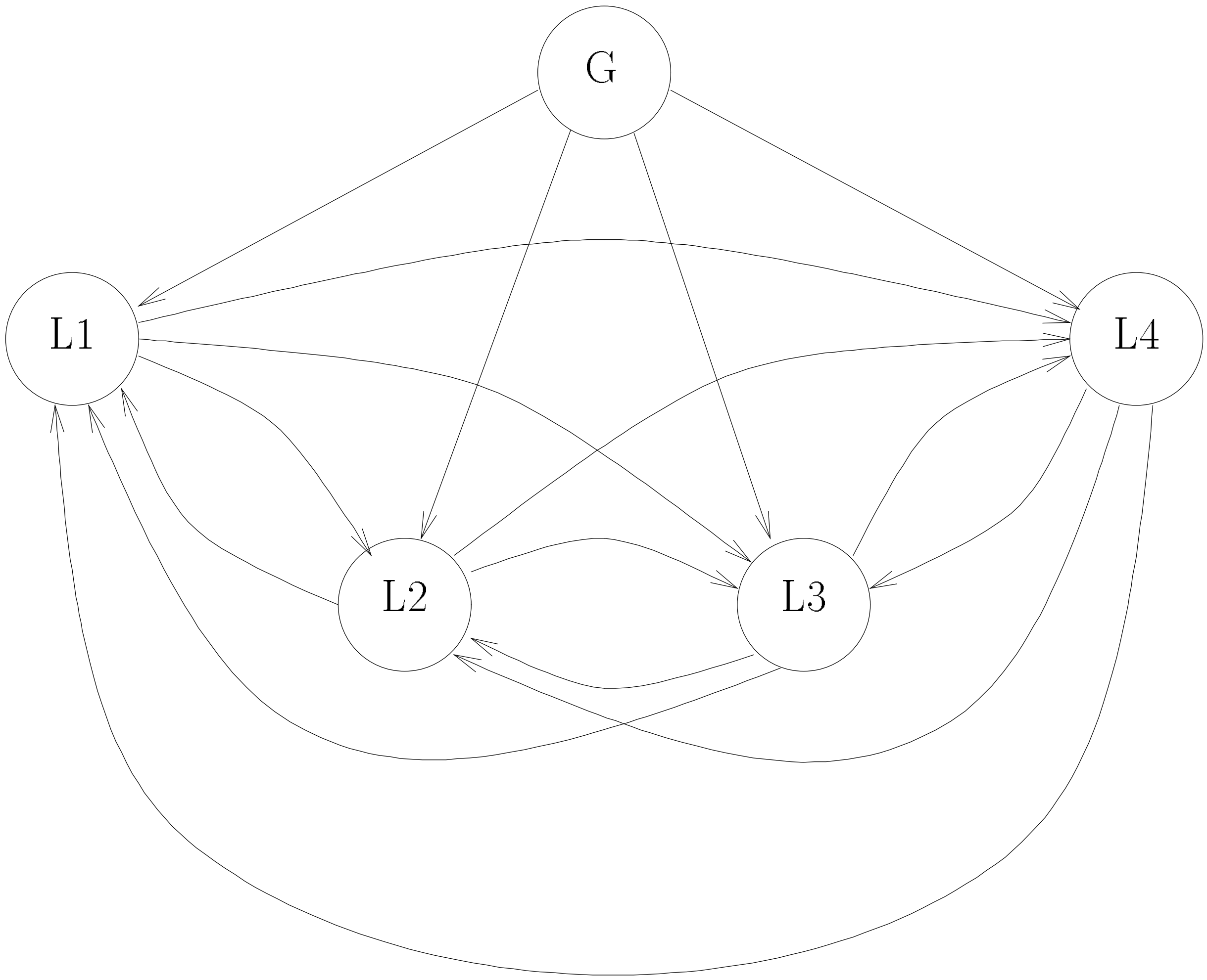
- **When Manifestly Bad Value Received, Record E**

- **Lieutenants Pass on E When Receive E**

- **Ignore E in majority vote**

Algorithm Z: Flawed

- $n = 5$, $r = 1$, **the Commander has a Crash Fault, one Lieutenant is a Traitor**
- **Good Lieutenants will believe whatever they receive from the Traitor: Protocol Fails**
- **MAFT Implementation Correct**



Importance of Validation

- **Lincoln & Rushby Specified Incorrect Algorithm**
- **Like Z, But Lieutenants Pass on RE "Reported Error"**
- **With Incorrect Axiom "Hybrid_Majority_Vote_1"**
  - **"Ignore Votes of Crashed Processors"**
  - **1-Round Case: OK**
  - **2-Round Case: Impossible**
  - **Should Have Been: "Ignore E Values"**
- **With Axiom, "Verified" Incorrect Algorithm**
- **Should Validate Axioms With Implementation**

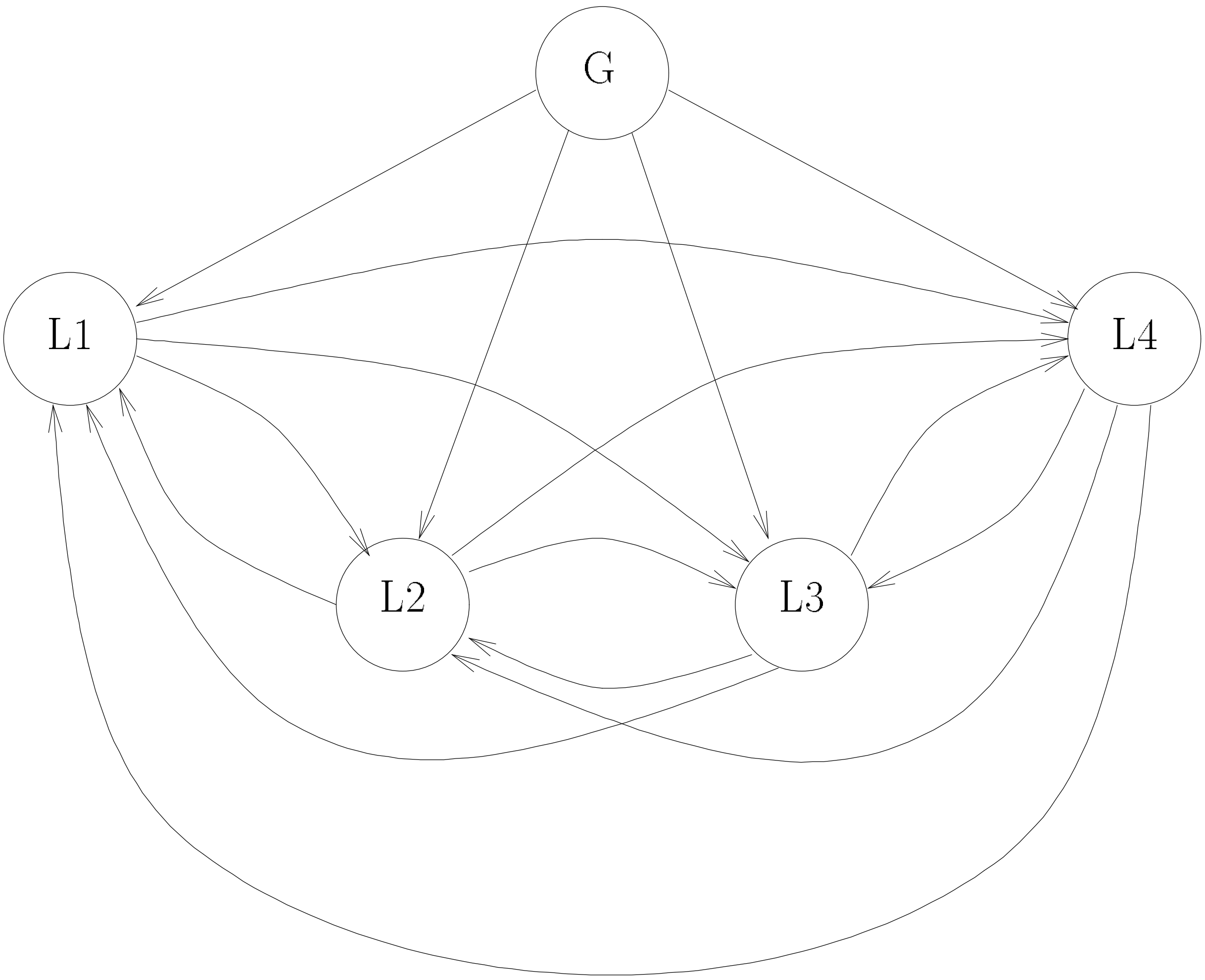
Algorithm OMH (Lincoln & Rushby '93)

- **Like OM, but**
- **Add $m+1$ kinds of error value: E, RE, R^2E etc.**
- **If receive R^iE, pass on as $R^{i+1}E$**
- **Ignore E in majority vote**
- **If vote yields R^jE, selected value is $R^{j-1}E$**
- **Formally Specified and Verified in PVS**
  (Lincoln & Rushby '93)
  Validated Axioms With MJRTY Implementation

Doubt Anyone Could Get This Right
Without Formal Verification

Algorithm OMH

$n = 5, r = 1$



Algorithm OMH

- **If $r \geq a$, then OMH(r) Tolerates**

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

If $r = a$, and $s = c = 0$, Then $n > 3a$
(Same as Traditional Bound)

- **(a) Arbitrary (Byzantine, Asymmetric)**
- **(s) Symmetric (Value, Stuck-At)**
- **(c) Crash (Manifest, Stopped, Out-of-Bounds)**
- **(g) Good (Nonfaulty, OK)**

Fault Combinations Tolerated

OM

Number of Faults		
Arbitrary (byzantine)	**Symmetric (value)**	**Crash (manifest)**
1	**0**	**0**
0	**1**	**0**
0	**0**	**1**

OMH

Number of Faults		
Arbitrary (byzantine)	**Symmetric (value)**	**Crash (manifest)**
1	**1**	**0**
1	**0**	**2**
0	**2**	**0**
0	**1**	**2**
0	**0**	**5**

Achieved Reliability: 6 Processor OMH

- **McElvany-Hugue's Reliability Analysis**
- **Reliability Goal: 10^{-9} System Failures/Hour**
  - **6 Processor OM: 1.5×10^{-7}**
  - **6 Processor OMH: 1.5×10^{-11}**

Formal Specification of OMH in PVS

$\mathrm{omh}[m : \mathrm{nat},\ \ n : \mathrm{posnat},\ \ T : \textsc{type},\ \ \mathrm{error} : T,\ \ \mathrm{R}, \mathrm{UnR} : [T \rightarrow T]] : \textsc{theory}$

$\textsc{begin}$

$\quad\textsc{assuming}\quad \mathrm{act\_ax} : \textsc{assumption}\ (\forall\ (t :\ T) :\ \mathrm{R}(t)\ \neq\ \mathrm{error})$

$\qquad\qquad\qquad \mathrm{unact\_ax} : \textsc{assumption}\ (\forall\ (t :\ T) :\ \mathrm{UnR}(\mathrm{R}(t))\ =\ t)$

$\quad\textsc{endassuming}$

$\mathrm{rounds} :\ \textsc{type}\ \ =\ \mathrm{upto}[m]$

$t :\ \textsc{var}\ T$

$\mathrm{fcu} :\ \textsc{type}\ \ =\ \mathrm{below}[n]$

$\mathrm{fcuset} :\ \textsc{type}\ \ =\ \mathrm{setof}[\mathrm{fcu}]$

$\mathrm{fcuvector} :\ \textsc{type}\ \ =\ [\mathrm{fcu} \rightarrow\ T]$

$G, p, q, z :\ \textsc{var}\ \mathrm{fcu}$

$v, v_1, v_2 :\ \textsc{var}\ \mathrm{fcuvector}$

$\mathrm{caucus} :\ \textsc{var}\ \mathrm{fcuset}$

$r :\ \textsc{var}\ \mathrm{rounds}$

$\textsc{importing}\ \mathrm{finite\_cardinality}[\mathrm{fcu}, n, \mathrm{identity}[\mathrm{fcu}]],$

$\qquad\qquad \mathrm{filters}[\mathrm{fcu}],$

$\qquad\qquad \mathrm{card\_set}[\mathrm{fcu}, n, \mathrm{identity}[\mathrm{fcu}]],$

$\qquad\qquad \mathrm{hybrid\_mjrty}[T, n, \mathrm{error}]$

$\mathrm{statuses} :\ \textsc{type}\ \ =\ \{\mathrm{arbitrary}, \mathrm{symmetric}, \mathrm{manifest}, \mathrm{nonfaulty}\}$

$\mathrm{status} :\ [\mathrm{fcu} \rightarrow\ \mathrm{statuses}]$

$a(z) :\ \mathrm{bool}\ =\ \mathrm{arbitrary}(\mathrm{status}(z))$

$s(z) :\ \mathrm{bool}\ =\ \mathrm{symmetric}(\mathrm{status}(z))$

$c(z) :\ \mathrm{bool}\ =\ \mathrm{manifest}(\mathrm{status}(z))$

$g(z) :\ \mathrm{bool}\ =\ \mathrm{nonfaulty}(\mathrm{status}(z))$

$\mathrm{as}(\mathrm{caucus}) :\ \mathrm{fcuset}\ =\ \mathrm{filter}(\mathrm{caucus}, a)$

$\mathrm{ss}(\mathrm{caucus}) :\ \mathrm{fcuset}\ =\ \mathrm{filter}(\mathrm{caucus}, s)$

$\mathrm{cs}(\mathrm{caucus}) :\ \mathrm{fcuset}\ =\ \mathrm{filter}(\mathrm{caucus}, c)$

$\mathrm{gs}(\mathrm{caucus}) :\ \mathrm{fcuset}\ =\ \mathrm{filter}(\mathrm{caucus}, g)$

$\mathrm{fincard\_all} : \textsc{lemma}$

$\quad |\mathrm{caucus}|\ =\ |\mathrm{as}(\mathrm{caucus})|\ +\ |\mathrm{ss}(\mathrm{caucus})|\ +\ |\mathrm{cs}(\mathrm{caucus})|\ +\ |\mathrm{gs}(\mathrm{caucus})|$

$\mathrm{send} :\ [T, \mathrm{fcu}, \mathrm{fcu} \rightarrow\ T]$

$\mathrm{send1} :\ \textsc{axiom}\ g(p)\ \supset\ \mathrm{send}(t, p, q)\ =\ t$

$\mathrm{send2} :\ \textsc{axiom}\ c(p)\ \supset\ \mathrm{send}(t, p, q)\ =\ \mathrm{error}$

$\mathrm{send3} :\ \textsc{axiom}\ s(p)\ \supset\ \mathrm{send}(t, p, q)\ =\ \mathrm{send}(t, p, z)$

$\mathrm{send\_lemma} :\ \textsc{lemma}\ \neg\ a(p)\ \supset\ \mathrm{send}(t, p, q)\ =\ \mathrm{send}(t, p, z)$

$\mathrm{HMajority}(\mathrm{caucus}, v) :\ T\ =\ \mathrm{proj\_1}(\mathrm{hybrid\_mjrty}(\mathrm{caucus}, v, n))$

$\mathrm{HMajority1} : \textsc{lemma}$

$\quad |\mathrm{gs}(\mathrm{caucus})|\ >\ |\mathrm{as}(\mathrm{caucus})|\ +\ |\mathrm{ss}(\mathrm{caucus})|$

$\qquad \wedge\ (\forall\ p :\ g(p)\ \wedge\ p \in \mathrm{caucus}\ \supset\ v(p)\ =\ t)$

$\qquad\ \ \wedge\ t\ \neq\ \mathrm{error}\ \wedge\ (\forall\ p :\ c(p)\ \wedge\ p \in \mathrm{caucus}\ \supset\ v(p)\ =\ \mathrm{error})$

$\qquad \supset\ \mathrm{HMajority}(\mathrm{caucus}, v)\ =\ t$

HMajority2 : LEMMA
$(\forall\ p:\ p \in \text{caucus} \supset v_1(p) = v_2(p))$
$\supset \text{HMajority}(\text{caucus}, v_1) = \text{HMajority}(\text{caucus}, v_2)$

$\text{OMH}(G, r, t, \text{caucus})$: RECURSIVE fcuvector =
IF $r = 0$
THEN $(\lambda\ p:\ \text{send}(t, G, p))$
ELSE $(\lambda\ p :$ IF $p = G$
THEN $\text{send}(t, G, p)$
ELSE $\text{UnR}(\text{HMajority}(\text{caucus} - \{G\},$
$(\lambda\ q : \text{OMH}(q, r-1, \text{R}(\text{send}(t, G, q)), \text{caucus} - \{G\})(p))))$
ENDIF)
ENDIF
MEASURE $(\lambda\ G, r, t, \text{caucus} \rightarrow \text{nat}:\ r)$

$\text{Validity\_Prop}(r) : \text{bool} =$
$\neg\, a(q) \wedge p \in \text{caucus} \wedge q \in \text{caucus}$
$\wedge\ |\text{caucus}| > 2 \times (|\text{as}(\text{caucus})| + |\text{ss}(\text{caucus})|) + |\text{cs}(\text{caucus})| + r$
$\supset \text{OMH}(q, r, t, \text{caucus})(p) = \text{send}(t, q, p)$

Validity : LEMMA $\text{Validity\_Prop}(r)$

Validity_Final : THEOREM
$g(p) \wedge \neg\, a(G) \wedge 2 \times |a| + 2 \times |s| + |c| + m < n$
$\supset \text{OMH}(G, r, t, \text{fullset}[\text{fcu}])(p) = \text{send}(t, G, p)$

$\text{Agreement\_Prop}(r) : \text{bool} =$
$g(p) \wedge g(q) \wedge p \in \text{caucus} \wedge q \in \text{caucus} \wedge z \in \text{caucus}$
$\wedge\ |\text{caucus}| > 2 \times (|\text{as}(\text{caucus})| + |\text{ss}(\text{caucus})|) + |\text{cs}(\text{caucus})| + r$
$\wedge\ r \geq |\text{as}(\text{caucus})|$
$\supset \text{OMH}(z, r, t, \text{caucus})(p) = \text{OMH}(z, r, t, \text{caucus})(q)$

Agreement : LEMMA $\text{Agreement\_Prop}(r)$

Agreement_Final : THEOREM
$g(p) \wedge g(q) \wedge |a| \leq m \wedge 2 \times |a| + 2 \times |s| + |c| + m < n$
$\supset \text{OMH}(G, m, t, \text{fullset}[\text{fcu}])(p) = \text{OMH}(G, m, t, \text{fullset}[\text{fcu}])(q)$
END omh

Axioms (Assumptions)

$$\text{send1}: \;\; \text{AXIOM}\;\; g(p) \;\supset\; \text{send}(t,p,q) \;=\; t$$
$$\text{send2}: \;\; \text{AXIOM}\;\; c(p) \;\supset\; \text{send}(t,p,q) \;=\; \text{error}$$
$$\text{send3}: \;\; \text{AXIOM}\;\; s(p) \;\supset\; \text{send}(t,p,q) \;=\; \text{send}(t,p,z)$$

$$\text{send\_lemma}: \;\; \text{LEMMA}$$
$$\neg\, a(p) \;\supset\; \text{send}(t,p,q) \;=\; \text{send}(t,p,z)$$

Theorem 1

$$\begin{array}{l}\text{Validity\_Final} : \text{THEOREM} \\ (g(p) \ \wedge \ \neg \ a(G) \\ \ \ \wedge \ 2 \ \times \ |a| \ + \ 2 \ \times \ |s| \ + \ |c| \ + \ r \ < \ n) \\ mbox \ \supset \ \text{OMH}(G, r, t, \text{fullset}[\text{fcu}])(p) \ = \ \text{send}(t, G, p)\end{array}$$

Algorithm OMH(r) **satisfies Validity if there are more than** $2(a+s)+c+r$ **processors.**

Validity: If The Commander Is not Byzantine-Faulty, The Value Received By Nonfaulty Lieutenants Is The Value Actually Sent

Style of Proof
Bevier & Young, Rushby, Shankar

- **Inductive Property Definition**
- **Induction Property Assertion (Lemma)**
- **Theorem in Final Form**

$$
\begin{array}{l}
\text{Validity\_Prop}(r) : \text{bool} \;\; = \\
\neg\; a(q) \;\; \wedge \;\; p \in \text{caucus} \;\; \wedge \;\; q \in \text{caucus} \\
\quad \wedge \;\; |\text{caucus}| \;\; > \;\; 2 \;\; \times \;\; (|\text{as}(\text{caucus})| \;\; + \;\; |\text{ss}(\text{caucus})|) \\
\qquad\qquad\qquad\qquad + \;\; |\text{cs}(\text{caucus})| \;\; + \;\; r \\
\quad \supset \;\; \text{OMH}(q, r, t, \text{caucus})(p) \;\; = \;\; \text{send}(t, q, p)
\end{array}
$$

$$
\text{Validity} : \;\; \textsc{lemma} \;\; \text{Validity\_Prop}(r)
$$

$$
\begin{array}{l}
\text{Validity\_Final} : \textsc{theorem} \\
\;(g(p) \;\; \wedge \;\; \neg \; a(G) \\
\quad \wedge \;\; 2 \;\; \times \;\; |a| \;\; + \;\; 2 \;\; \times \;\; |s| \;\; + \;\; |c| \;\; + \;\; r \;\; < \;\; n) \\
\quad \supset \;\; \text{OMH}(G, r, t, \text{fullset}[\text{fcu}])(p) \;\; = \;\; \text{send}(t, G, p)
\end{array}
$$

While ReRunning Proof, We Noticed

Lemma **If $c = 0$ (No Crash Faults), OMH $\approx$ OM**

OM Is Better Than Earlier Claimed:

Corollary 1 **For any r, Algorithm OM(r) satisfies Validity if there are more than $2(a + s) + r$ processors.**

Corollary 2 **For any r, Algorithm OM(r) satisfies Agreement if there are more than $2(a+s)+r$ processors and $r \geq a$.**

OMH = OM + Diagnosis of Manifest Faults

More Corollaries

More Theorems Derived from our New Improved Understanding

Corollary 3: No Good Processors Are Confused By Crash Faults: If $a = s = 0$ (No Byzantine, No Symmetric Faults), OMH is optimal.

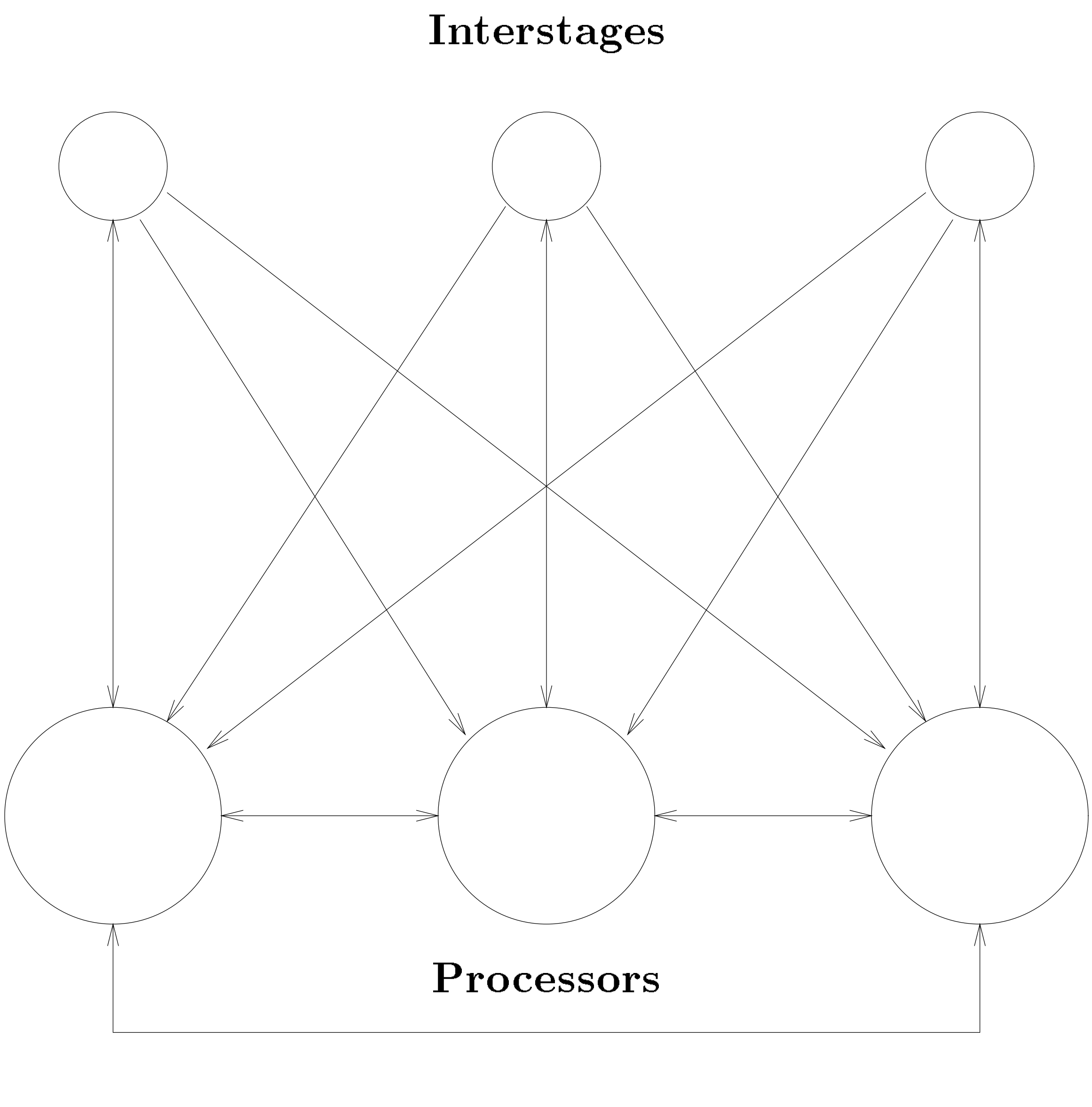
Corollary 4: Error Values (RE, R^{2}E, $\cdots$) Can Overlap With Real Values.

Corollary 5: (Slightly Modified) OMH(r) Allows Implementations With Only $r + 1$ Error Values.

FTP Architecture

- **Less Total Hardware**
- **Only need 3 voters to mask 1 Byzantine fault (!)**
- **Technology used in AIPS, FTTP, etc.**

FTP Architecture



Algorithm OM-FTP

- Lala 1986
- Similar to one-round OM with symmetric voters
- Formally specified in EHDM
  (Harper, Alger, & Lala '91)
  No proofs
- Formally specified in PVS
  (Lincoln & Rushby '94)
  Full formal proofs completed in PVS
  Couple of days

Hybrid Faults in FTP

- Integrate OMH and OM-FTP
- Resist more (simple) faults with less hardware

Algorithm OMH-FTP
(Lincoln & Rushby 1994)

- Combine OMH and OM-FTP
- Voters pass on E as RE
- Interstages pass on E as E
- Voters use value reflected from interstage, NOT original value

Formal Verification in PVS

- Proof Reuse
- Copy-and-Edit Proof of OM-FTP
- Grab parts of Proof of OMH
- M-x Edit Proof
- Rerunning Edited Proofs
- Couple of Days Effort

Copy-and-Edit a Proof

- **Emacs Interface to Edit Proof Commands**
- **Kill and Yank**
- **Mix-And-Match Partial Proofs**
- **Rerun Until Proof Fails**

Authenticated Agreement

- **Use Cryptographically Secure Signatures**

- **Can Tolerate Many More Faults**

However

- **Questionable Cryptographic Assumptions (Key Distribution, Complexity Questions)**

- **Traitor May Simply GUESS Generals Key Possible, Though Very Unlikely**

Agreement Using Signed Messages (SM) (Lamport, Shostak, & Pease 1982)

- $r + 1$ **Rounds Of Message Exchange**
- **If 1 Properly Signed Message At End, Adopt It Else General Is Faulty, Adopt Default**
- **Requires Only** $n \geq r$ **For Agreement and Validity (Best Possible Bound)**
- **If Signatures Broken, Protocol Fails Horribly**

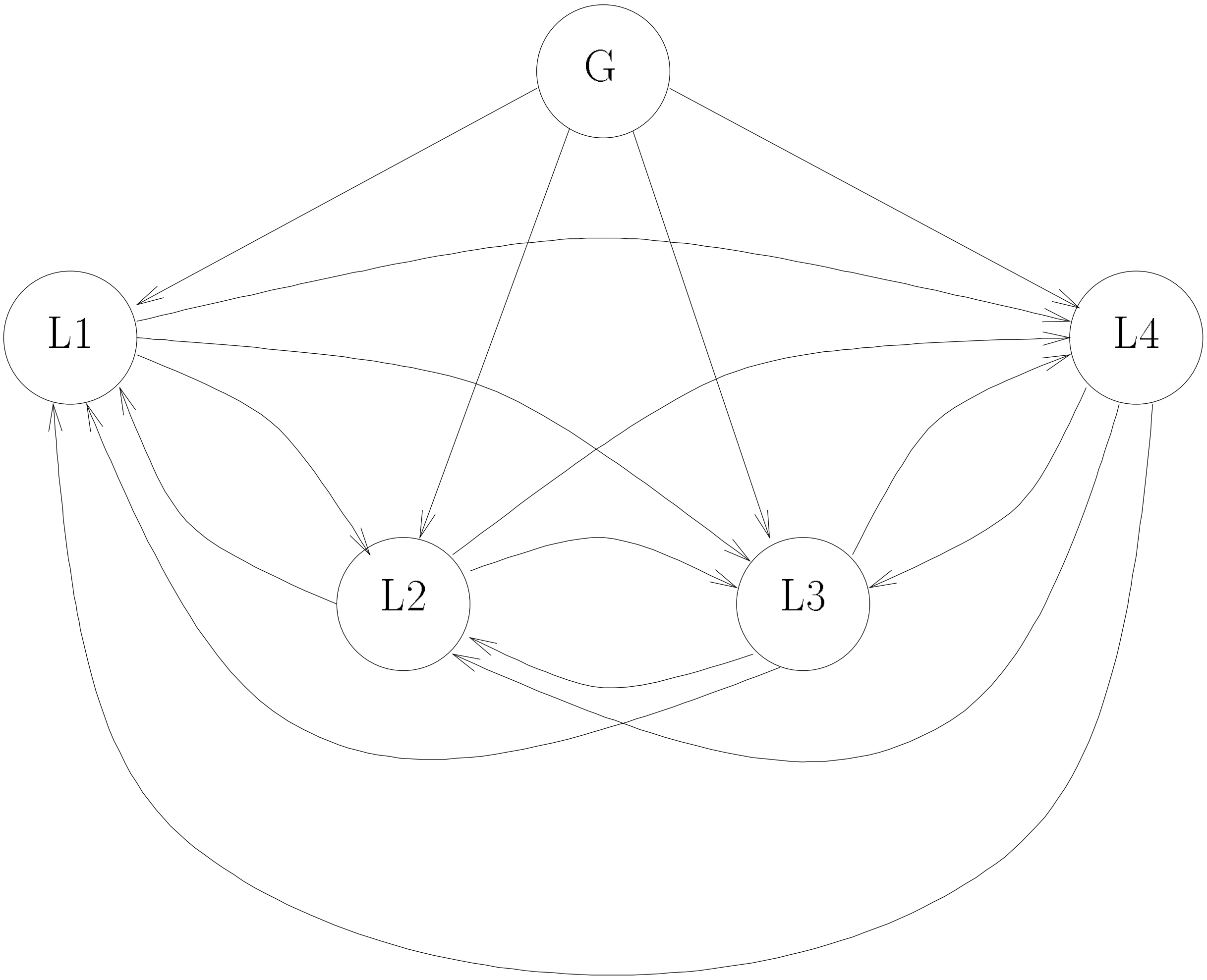
  One Traitor with Key breaks Agreement and Validity for Any Number of Rounds, Any Number of Processors

Use Signatures In OMH (OMHA)
(Gong, Lincoln, Rushby 1995)

- **Same as OMH, But at Every Step Check Signatures**
- **Symmetric, Arbitrary Lieutenants $\Rightarrow$ Manifest**
- **Except: Bad Lieutenants Could Send $R(E)$**
- **Worst Case With Secure Signatures Same As OMH (Worse Than SM)**
- **If Signatures Broken, Same Tolerance as OMH (Much Better Than SM)**

It Would Be Nice to Disallow $R(E)$

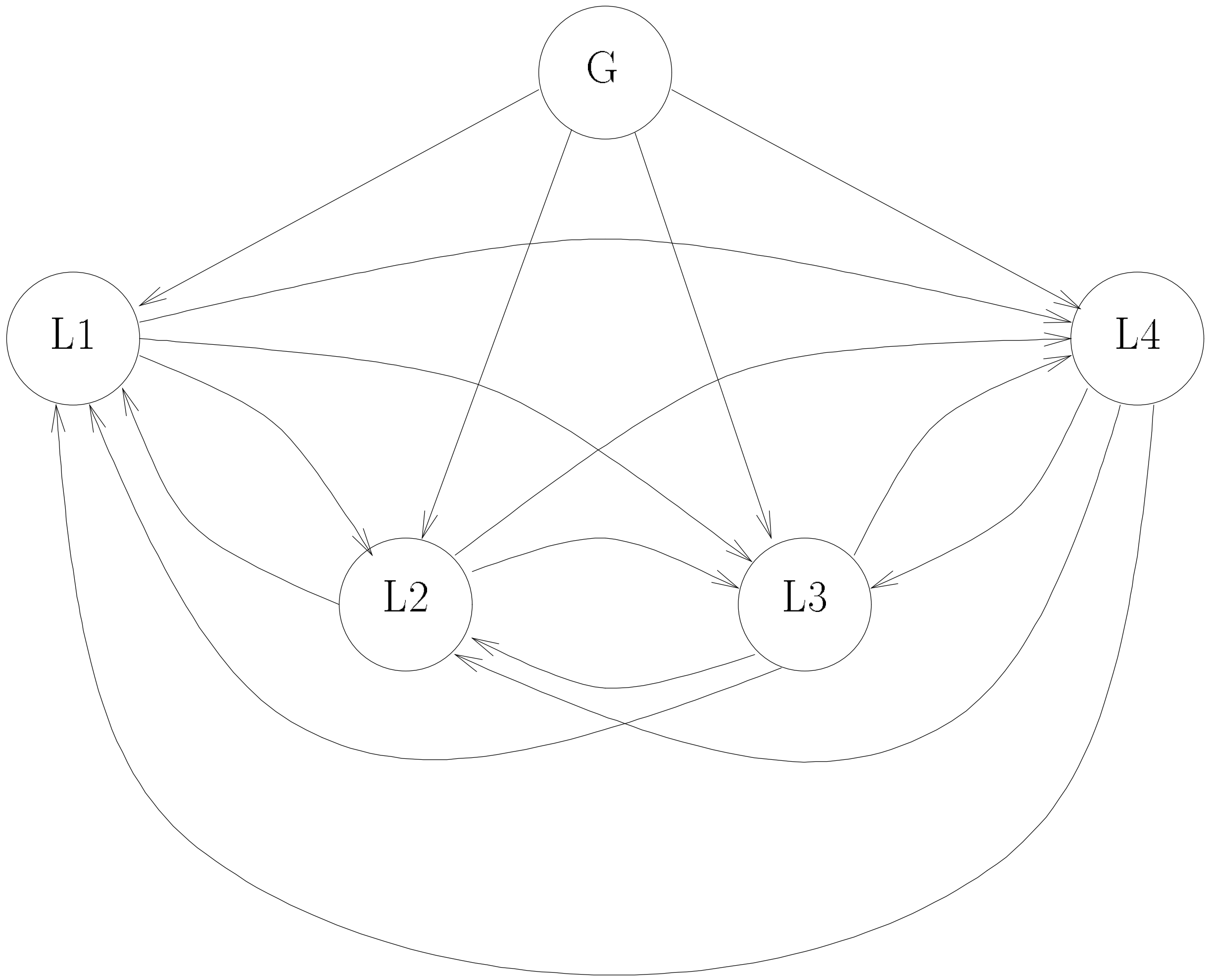
Algorithm OMHA: Signatures Secure



Algorithm OMHA: Broken Signatures

- $n = 5$, $r = 1$, **the Commander has a Crash Fault, one Lieutenant is a Traitor with Crypto Key**

- **Good Lieutenants Not Fooled: Protocol Succeeds**



Use Signatures In Z (ZA)
(Gong, Lincoln, Rushby 1995)

- **Same as Z, But at Every Step Check Signatures**

- **Symmetric, Arbitrary Lieutenants $\Rightarrow$ Manifest**

- **Bad Lieutenants Cannot Send R(E)**

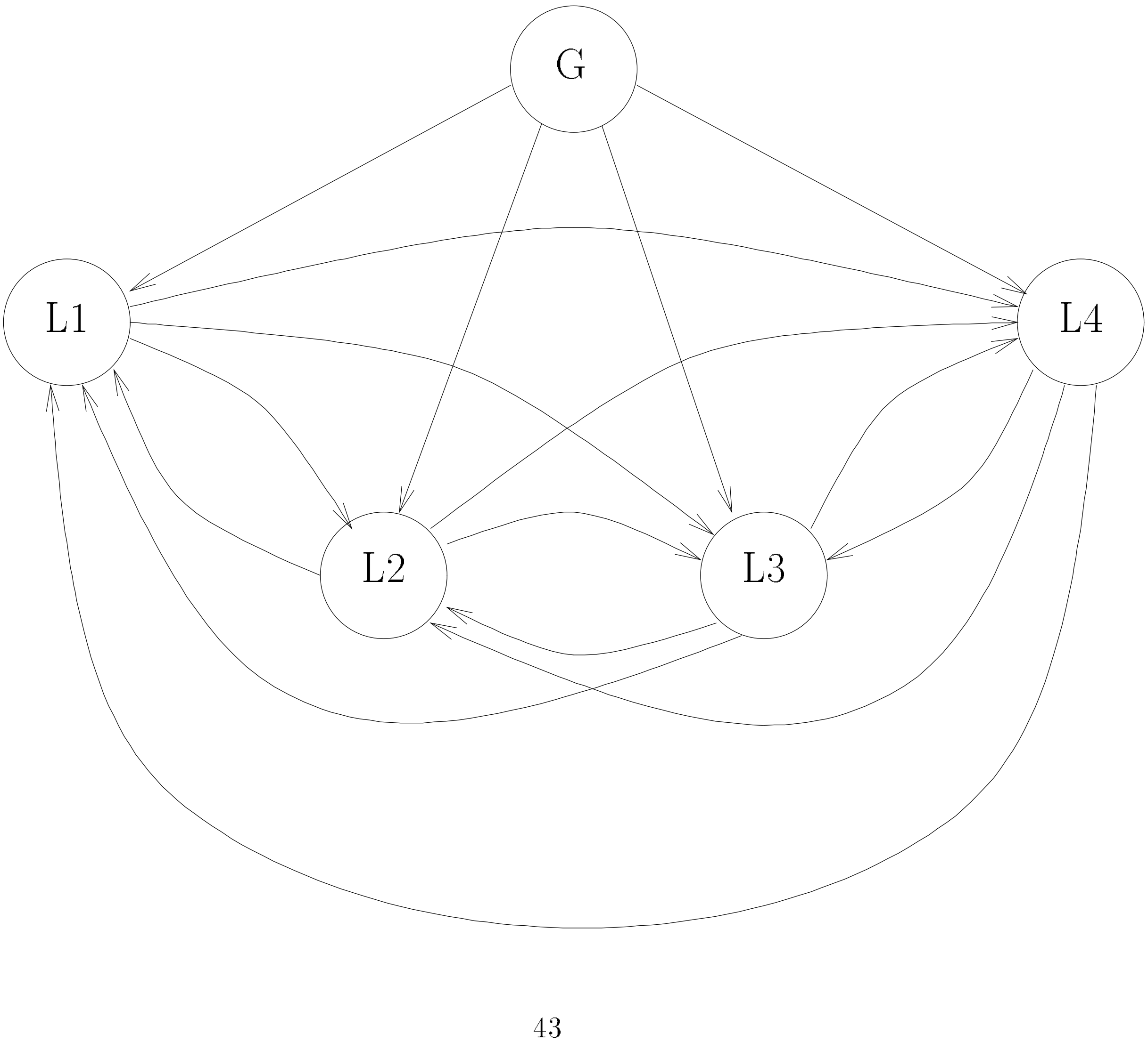
- **With Secure Signatures, ZA is Optimal (Same as SM, Better than OMH)**

- **If Signatures Broken, Same Tolerance as Z (Better Than SM, Worse Than OMH)**

Algorithm ZA: Broken Signatures

- $n = 5$, $r = 1$, **the Commander has a Crash Fault, one Lieutenant is a Traitor with Crypto Key**

- **Good Lieutenants will believe whatever they receive from the Traitor: Protocol Fails**



Symbolic Fault Injection (Gong, Lincoln, Rushby 1995)

- **Using Dill's Murϕ Tool (Stanford)**

- **Specified OM, OMH, OMHA, SM, Z, ZA**

- **Complete, Exhaustive State Exploration**
  5 Processors, 4 Values

- **All Above Bounds Correct**
  Rediscovered Flaw in Z

Symbolic Fault Injection: Beyond The Worst Case Bound

$>$ 20,000 Fault Configurations (Most Beyond Above Bounds)

Clock Synchronization

- **All Above Protocols Require Synchronization**
- **Byzantine Processors May Lie About Clock Value**
- **Assume Bounded Clock Drift For Good Clocks**
- **Goal: Maintain Bounded Clock Skew Within δ**

Interactive Convergence Algorithm: ICA (Lamport & Melliar-Smith '85)

- **Broadcast Current Clock Value**

- **Compute Egocentric Mean**
  - **Ignore Values Too Far Off**

    $n > 3a$

- **Algorithm Correct**

- **All But One Lemma Incorrect**

- **Proof of Main Theorem Incorrect**

- **Corrected Versions Formally Verified in EHDM (Rushby & VonHenke 93) Couple of Months Effort**

- **Young Formally Verified in NQTHM**
  Noted Perfect Clocks Excluded By Rushby
  $<$ should be $\leq$ in Axiom

Other Clock Synchronization Algorithms

- **Fred Schneider's General Treatment**

- **Natarajan Shankar Formally Specified and Verified in EHDM**
  Corrected Several Small Flaws

- **Verified That ICA Satisfies Corrected Properties**

- **Paul Miner Found Better, Weaker Assumption Set**
  Example Lundelius-Lynch Algorithm

Hybrid Interactive Convergence: ICAH (Rushby 94)

- **Ignore E Values In Egocentric Mean**

$$n > 3a + 2s + c$$

- **Extend Hybrid Fault Model to Link Faults:**
  - **Model Links Separate From Processors**
  - **Links Only Introduce Manifestly Bad Values**
  - **Two Links Between Each Pair (In, Out)**
  - **Asymmetric: Good or Manifest Value**

$$n > 3a + 2s + c + l$$

- **Formally Specified and Verified in EHDM (Rushby 94) Couple of Days Effort**

Diagnosis

- **Key Part of FDIR:**
  Detection, Identification, and Reconfiguration

- **Off-Line: After/Between Missions**
  - **Requires Record Keeping/Testing Harnesses**
  - **Few Additional Mechanisms In-Flight**
  - **May Miss Intermittent Faults**
  - **Use Test Injection Sequences**
- **On-Line: During Mission**
  - **Requires Additional Safety-Critical Mechanisms**
  - **Consumes Mission Resources**
  - **Survives More Faults During Mission**

Benefits of On-Line Diagnosis

OMH Masks Crashes Already: Why Diagnose?

- **Diagnose ALL Symmetric Faults**
  Greater Reliability

- **Diagnose SOME Byzantine Faults**
  Greater Reliability

- **Manifest Faults May Become Worse**

- **Stop Progression:**
  Manifest $\Rightarrow$ Symmetric $\Rightarrow$ Byzantine

- **Restart Crashed Processor**

On-Line Diagnosis

- Easy Without Byzantine Faults

- All Faults Are Symmetric
  Private Accusations Correct

- Impossible To Perfectly Diagnose Byzantine Faults

Desirable Properties

Correctness (Fairness, Soundness):
Every Processor Diagnosed Faulty IS Faulty

Completeness:
Every Faulty Processor is Diagnosed Faulty

On-Line Byzantine Diagnosis

- **PMC, Comparison Testing Inappropriate**

- **Shin & Ramanathan '87 (Requires Authentication)**

- **Ramarao & Adams '88 (Optimal, NP-hard)**
  Extreme Cost

- **Walter '90, Walter, Suri & Hugue '94: Spectrum of Algorithms**
  PP, PLP, DD, HD

Algorithm PP: Walter '90

- Assume Symmetric Faults

- Correctness and Completeness

- Formally Specified and Verified in PVS
  (Lincoln 95)
  Couple Weeks Effort

Algorithm PP

- **Monitor All Incoming Messages**
- **For Incorrect Messages, Record Accusation**
- **At End of Period, Broadcast Accusations**
- **If** $\geq \lceil N/2 \rceil$ **Accusations, Declare Faulty**
  - **Accuse All Non-Accusing Processors**
- **If** $< \lceil N/2 \rceil$ **Accusations, Declare NonFaulty**
  - **Accuse All Accusing Processors**
- **Iterate**

Algorithm PLP: Walter '90

- PP with Link Faults
- Correctness and Partial Completeness
- Formally Specified and Verified in PVS (Lincoln 95)
- Heavy Reuse of Lemmas/Proofs from PP Couple Days Effort

Algorithm PLP

- Like PP
- Except, two kinds of Error:
  Symmetric/Asymmetric
- Based on Kind of Erroneous Message
- Asymmetric Errors Assumed to Be Manifest
  as Asymmetric-Looking Values

Algorithm DD: Walter, Suri, & Hugue '94

- True Byzantine Fault Diagnosis
  - Local Detection
  - Byzantine Agreement on Detection: OMH
  - Combine into Diagnosis

- Correctness and Partial Completeness

- Formally Specified and Verified in PVS
  (Lincoln 95)

- Heavy Reuse of Lemmas/Proofs
  from OMH & PLP
  Couple Hours Effort

Algorithm DD

- **Monitor All Incoming Messages**
- **For Incorrect Messages, Record Accusation**
- **Use OMH to Agree on Accusations**
- **If** $\geq \lceil N/2 \rceil$ **Accusations, Declare Faulty**
- **If** $< \lceil N/2 \rceil$ **Accusations, Declare NonFaulty**
- **Iterate**

Algorithm HD: Walter, Suri, & Hugue '90

- True Byzantine Fault Diagnosis
- DD with Penalties, Accumulation, Decay
- Correctness and Partial Completeness
- Formally Specified and Verified in PVS
  (Special Case) (Lincoln 95)
- Heavy Reuse of Lemmas/Proofs From DD
  Couple Hours Effort

Algorithm HD

- **Like DD**
- **Except, Different Penalty Weights for Different Errors**
- **Decay of Accusations**
- **Cumulative Penalty $> K$, Declare Faulty**

Benefits of Formal Verification

- **Identify Exact Assumptions**
- **Explore Alternatives**
- **Extremely High Assurance**
  - **Diagnosis Is Safety-Critical**
- **Documentation**

Important Properties of User

- **Mathematical Sophistication**
- **Application-Area Knowledge**
- **Experience**

Important Properties of Proof Checker

- Cite Previous Lemmas and Theorems
- Library Mechanism
- Reuse Parts of Proofs
- Fast Cycle Time (Human-Computer-Human)
- Understandable Output in the Loop
- Heavy Controllable Automation
  (Rewriting, Decision Procedures)
- Exploration Tools
  (Murϕ)

Bottom Line

Exploration of Algorithms, Assumptions

- **Assurance: Formal Verification**

- **Little Hardware: FTP**

- **Cheap Reliability: Hybrid Algorithms**
  - **Byzantine Case Ensures Coverage**
  - **Simple Cases Handled Cheaper**
  - **More Reliability With Same Hardware**
  - **Widely Applicable Paradigm**
    Clocks, Agreement, Diagnosis

"The virtue of a logical proof is not that it compels belief but that it suggests doubts" (Lakatos)

Summary

- **Good Hybrid Fault-Tolerant Algorithms**
- **Informal Proofs are Unreliable**
- **Real Theorem Proving Still Expensive**
- **Interesting Examples Now Feasible**
- **Human-Computer Interaction A Key Element**
- **Proof Reuse Lowers Cost of Formal Verification**