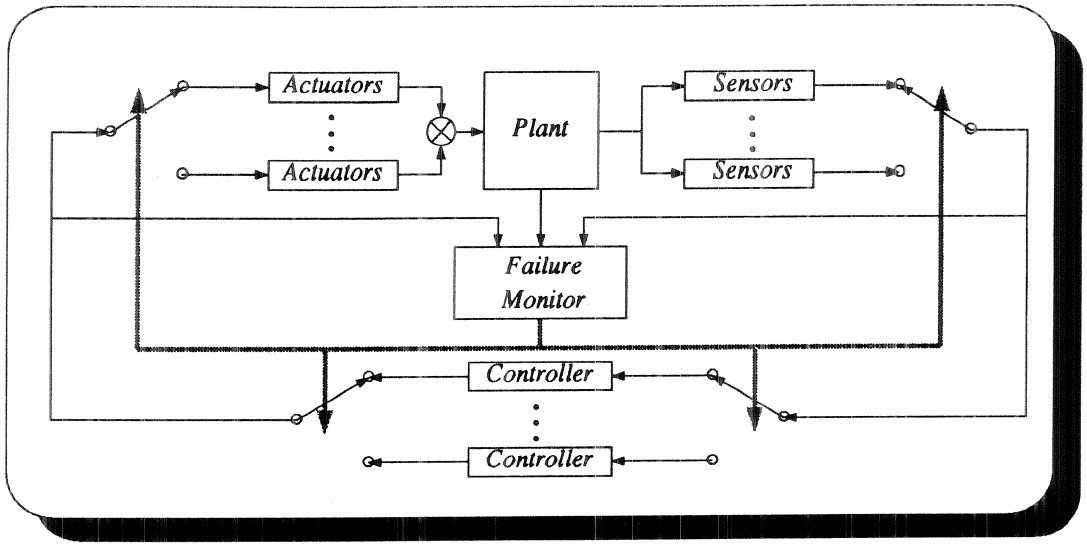
Fault-Tolerant Control System Design



N. Viswanadham
Indian Institute of Science
Bangalore, India

K. D. Minto
Control Systems Laboratory
GE-CRD, Schenectady, NY

January, 1990

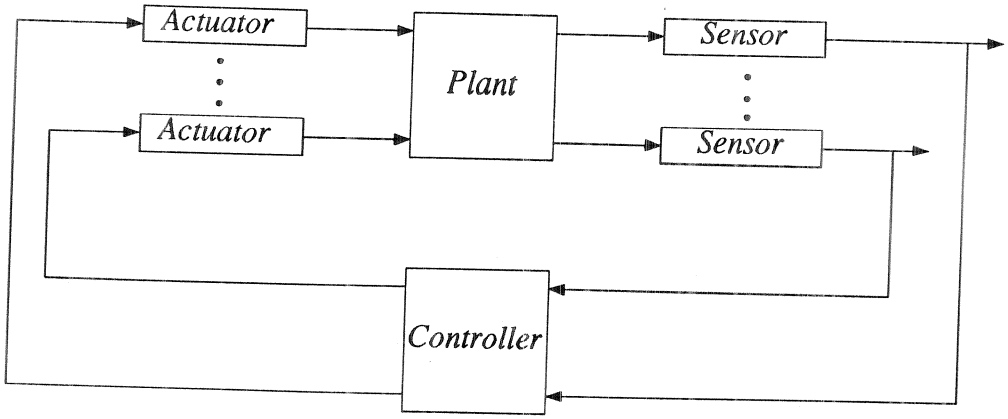
Fault-Tolerant Control System Design

Contents

1. Computer Control System Design

- Computer Control System
- Standard Design Methods: Deficiencies

Computer Control System



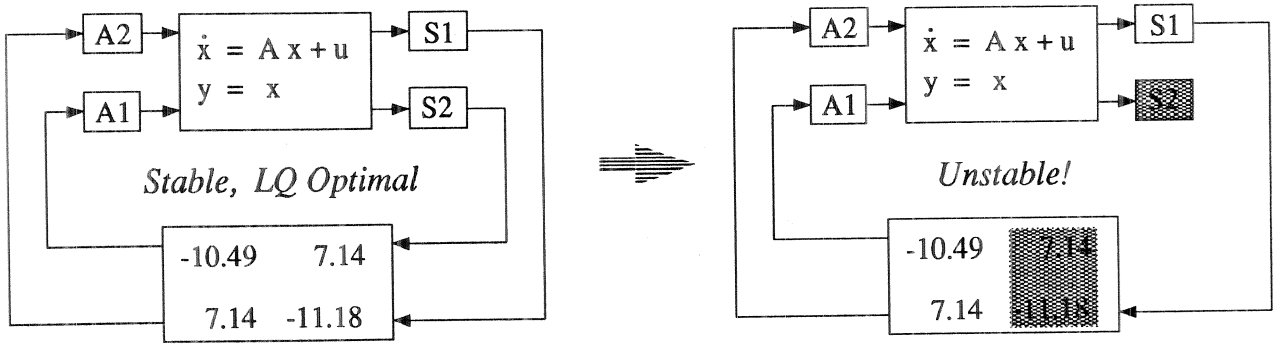
- Conventional design methods (H^∞/LQG/Multivariable Frequency Response, etc.) aim at **High Performance under failure-free conditions**

- Several CAD packages are available

  — ISICLE, PRO-MATLAB, MATRIX$_x$, Control-C, ...

Standard Design Methods: Deficiencies

- Standard design methods do not incorporate reliability as a design specification
- Failures induce large structural changes in the system adversely affecting stability and performance
- Example



$$J = \int_0^\infty \left(x_1^2 + x_2^2 + u_1^2 + u_2^2\right) dt\,; \qquad A = \begin{bmatrix} 5 & -3 \\ -4 & 6 \end{bmatrix}$$

2. Failures and Consequences

- Failures in Computer Control Systems
- Computer Fault Taxonomy
  - Behavioural classification
  - Temporal classification
  - Failure characteristics
  - Software errors
- Cause-Consequence Analysis

Failures in Computer Control Systems-1

- The Computer Control System
  - — Sensors, Actuators, Computer System and interface
- Sensors: Speed, Temperature, Pressure,..
- Actuators: Motors, Valves, Switches,..
- These Components Fail i.e. "They do not behave as they should"
  - — Hard Failure
    - → Outputs Violate a Constraint : Range, rate of Change
  - — Soft Failure
    - → Outputs Drift Slowly towards the Limit
- Consequence of Failures
  - — Benign
  - — Loss of Performance
  - — Instability

Failures in Computer Control Systems-2

- The Computer System
  - — Hardware
    - → CPU, Memory, I / O,..
  - — Software (Control Law)
    - → Baseline Programs

      Set of Continuously Executed Software Modules
    - → Auxiliary Programs

      Software Executed Occasionally
- These Components also Fail

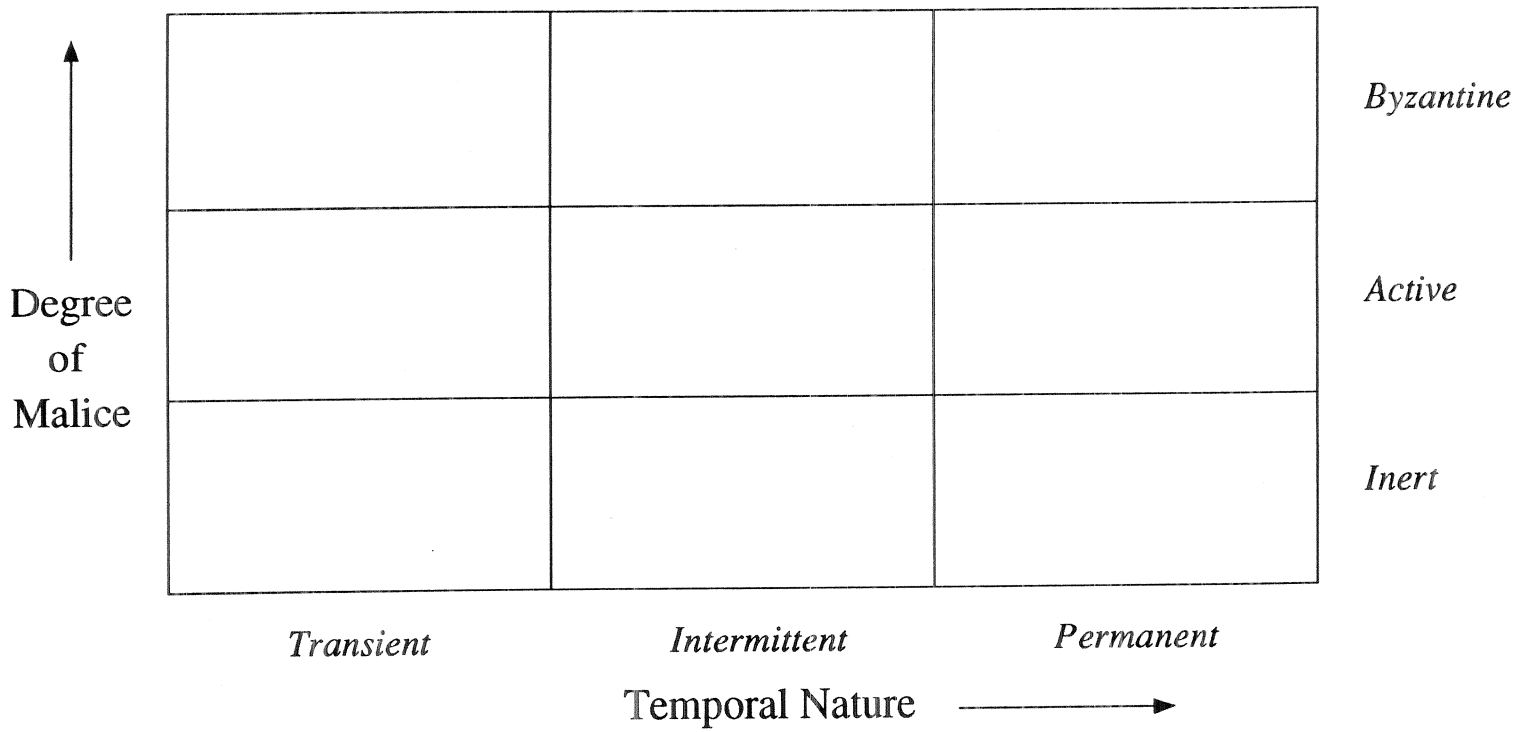
Computer Fault Taxonomy–1

- Behavioural Classification
  - — Inert fault
    - → Processor simply halts
      Causes include: loss of power, spurious jumps to wrong code segments, bus errors
  - — Timing fault
    - → Processor completes execution but too late
  - — Byzantine fault (Rare events in control applications)
    - → Processor has arbitrary and malicious behaviour
    - → Stops and restarts at a future time
    - → Sends conflicting information to different destinations

Computer Fault Taxonomy–2

- Temporal Classification
  - — Permanent fault (5–10%)
    - → Malfunction occurs and remains in the system
      - Bit in a memory stuck a fixed value
      - Open and short circuits
  - — Intermittent fault (5–10%)
    - → Malfunction appears and disappears repetitively
    - → A function of system loading and environmental stress
  - — Transient fault (80–90%)
    - → Fault appears, persists for a finite time, then vanishes
      - Noise spikes, power surges, EMI
    - → Not symptomatic of faulty hardware, but due to spurious external signals

Computer Fault Taxonomy — 3



Computer Failure Characteristics

- Activity: Latent or Active
  - — Active faults generate failures which can be detected whereas latent faults are present but do not cause errors
- Duration: Transient or Permanent
- Perception: Symmetric or Asymmetric
  - — Symmetric faults are seen identically by all good subsystems, while asymmetric faults are not
- Cause: Random or Generic
  - — Random faults are typically caused by environment while Generic faults are "designed in"
- Intent: Benign or Malicious
- Count: Single or Multiple
- Time (multiple faults): Coincident or Distinct
- Cause (multiple faults): Independent or Common-Mode

Software Errors (Bugs)

- Most frequently, man-made design faults
  - — User's intent may be compromised
- Software Failure Modes:
  - — Poor quality of fabrication
    - (i) typographical error eluding compiler checks
    - (ii) inclusion of wrong versions of subroutines
    - (iii) program incompatibility with OS and hardware
  - — Design errors
    - (i) Series expansion does not converge for certain input values
    - (ii) Forgeting to clear all registers when returning from a subroutine
    - (iii) Interchanging THEN ELSE branches in an IF statement
  - — Overload:
    - (i) Fast inputing of data at terminals
    - (ii) Number of terminals connected approaching maximum
- No faults due to wear out

Cause-Consequence Analysis

- Analysis should precede design
- Useful Failure Analysis Techniques:
  - FMEA
  - Event Trees
  - Fault Trees
  - Cause-Consequence Diagrams

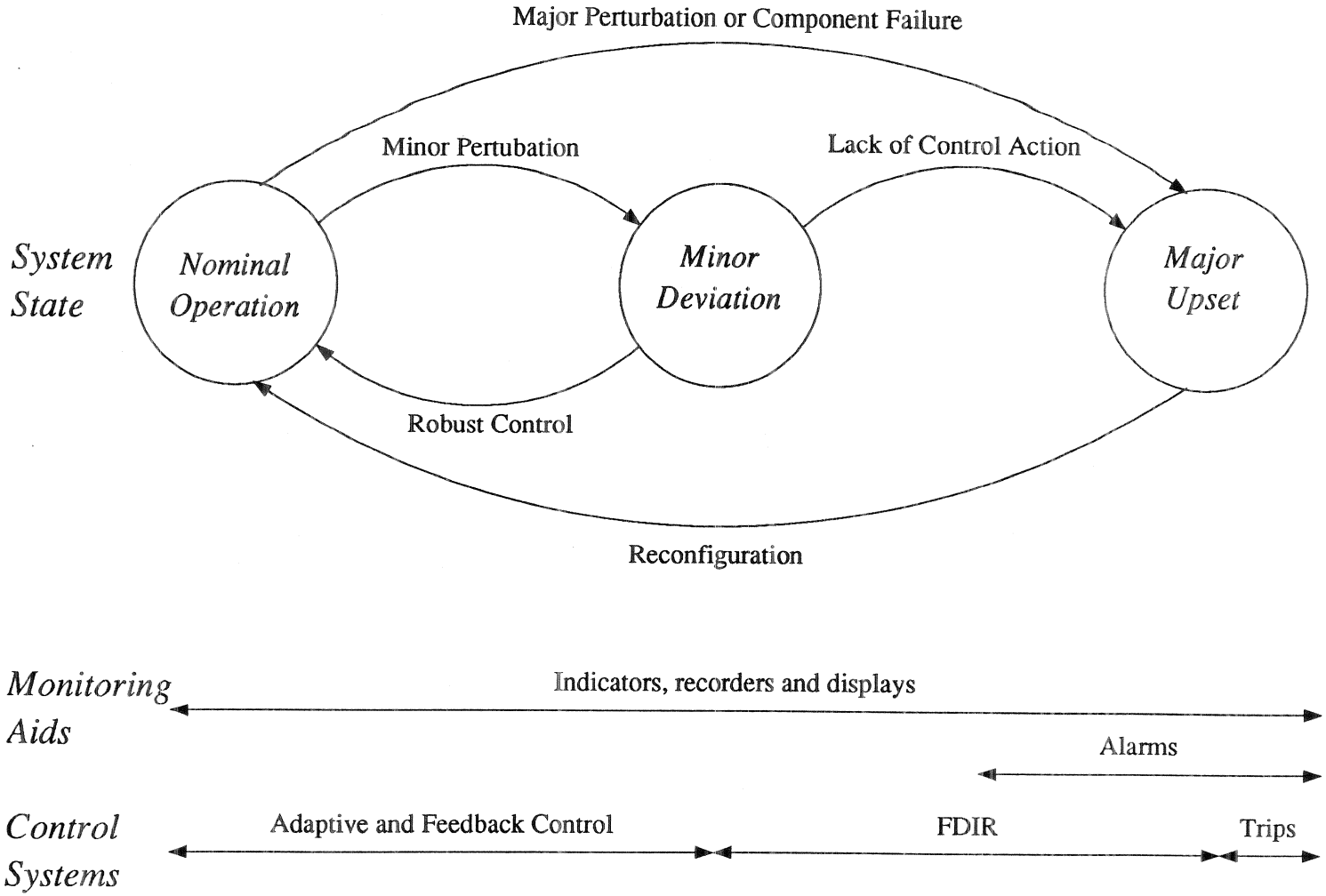
3. FTCS Definition and FTC Schemes

- Some Definitions
- Fault Detection, Isolation and Reconfiguration (FDIR): Operational Role
- FTCS: Definition
- FT Control Schemes
  - — Fault Masking Schemes
  - — Reconfiguration Schemes

Some definitions

- Faults: Departure from acceptable behaviour
  - Hard failure
  - Soft or deviational failure
- Detection: Binary decision
  - Component good
  - Fault occurance
- Isolation or Diagnosis:
  - Determine the source of failure
- Reconfiguration or Recovery:
  - Remove failed element from the system and reassign its function to other elements
- Fault Estimation:
  - Determine the extent of damage

FDIR: Operational Role



FTCS: Definition

"A *fault-tolerant control system* is one that has inherent capability to automatically adapt to failures of its own components, including sensors, actuators, computer hardware and software, so as to maintain *stability* and specified *steady-state* and *transient* response characteristics."

Fault-Tolerant Control Schemes–1

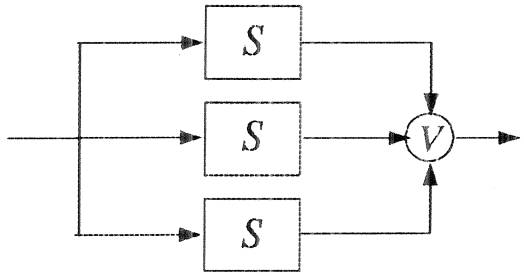
- Approach is not to eliminate faults, but to design for a high degree of fault-tolerance
- Schemes use redundant elements
- Candidate Approaches:
  - Fault-Masking Schemes (FMS)
  - Reconfiguration Schemes (RCS)
  - Hierarchical Schemes

Fault Masking Schemes

- These schemes hide the effects of failures
- No switching of redundant elements required
- Candidate Approaches:
  - Multi-Controller Configuration
  - Simultaneous Stabilization

Reconfiguration Schemes

- Use FDIR methods to isolate failed components and switch in the spares
- Candidate Approaches:
  - — System-Level TMR
  - — Component-Level TMR
  - — Controller Switching



TMR → "Triple Modular Redundancy"

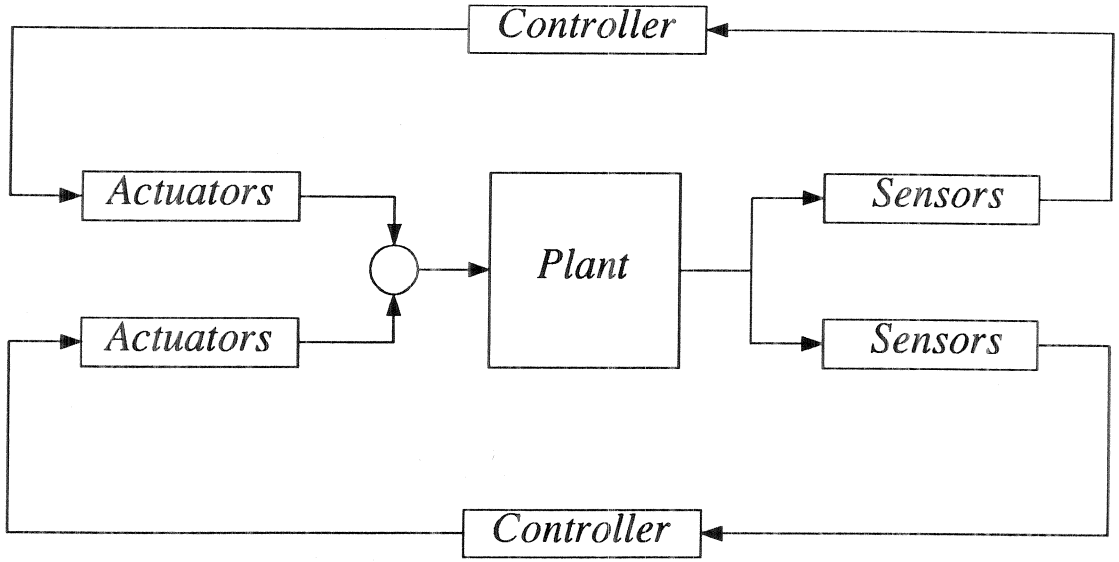
Hierarchical Schemes

- A two-layer scheme combining both fault-masking and reconfiguration
  - — First layer (real time)

    Fault-masking feedback controller
  - — Second layer (slower time-scale)

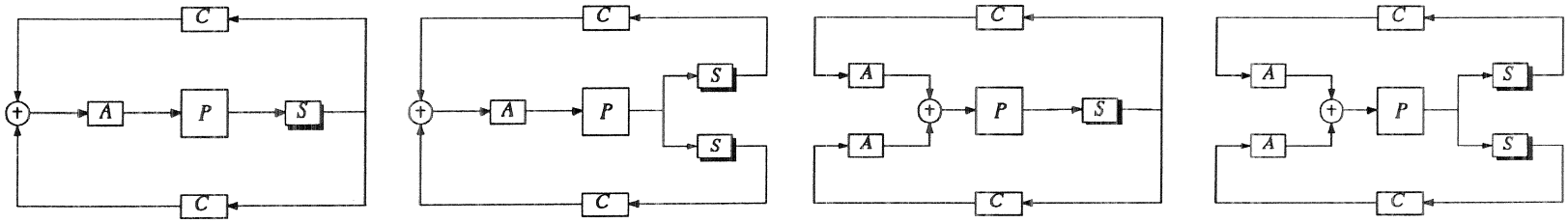
    A reconfiguration scheme comprising of:
    (i) Fault-detection and isolation
    (ii) Plant status estimation
    (iii) Reconfiguration

FMS: Multi-Controller Configuration-1



- Guaranteed stability for either loop failure, but performance may be degraded
- Equivalent of Parallel Redundancy schemes in reliability literature

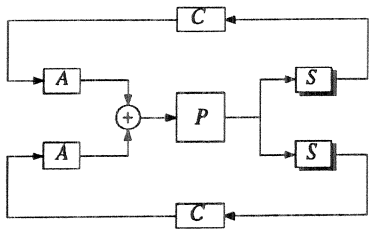
MCC — 2: System Configurations



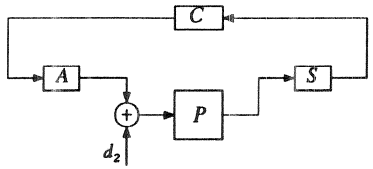
(Minimal Parallelism) (Maximal Parallelism)

- Nominally high performance, "limp-home" until reconfiguration/repair
- Complex ("Hard" redundancy)
- Eventual reconfiguration needed to maintain performance as well as to avoid system failure due to attrition
- Supports mixed implementation (Digital/Analog)

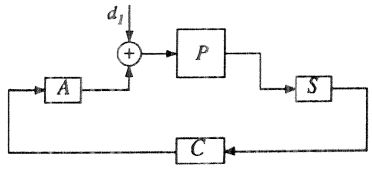
MCC — 3: Design Implications



(Nominal: Performance)



(Loop 2 Failure: Stability)



(Loop 1 Failure: Stability)

Problem Statement:

Given a compensator C that achieves a nominal performance spec., determine a *decomposition* $C = C_1 + C_2$ such that

(i) (P, C_1, C_2) is stable, with desired performance

(ii) (P, C_1) is stable

(iii) (P, C_2) is stable

Solution:

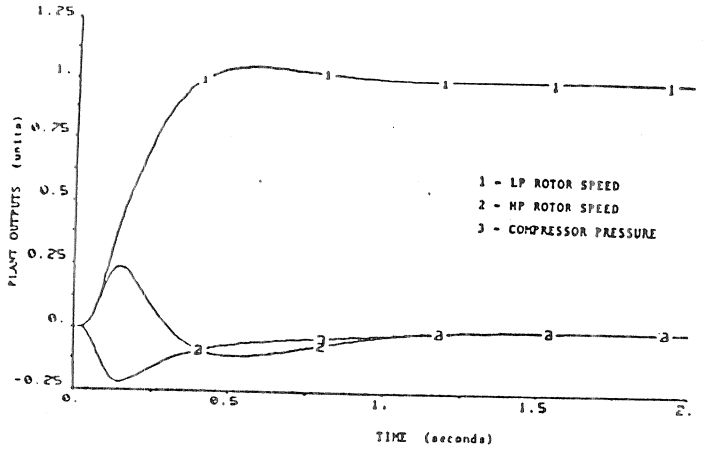
(i) Such a decomposition exists if and only if P can be stabilized by a stable compensator *(strong stabilization)*

(ii) Most plants have this property *(strong stabilization is generic for MIMO plants)*

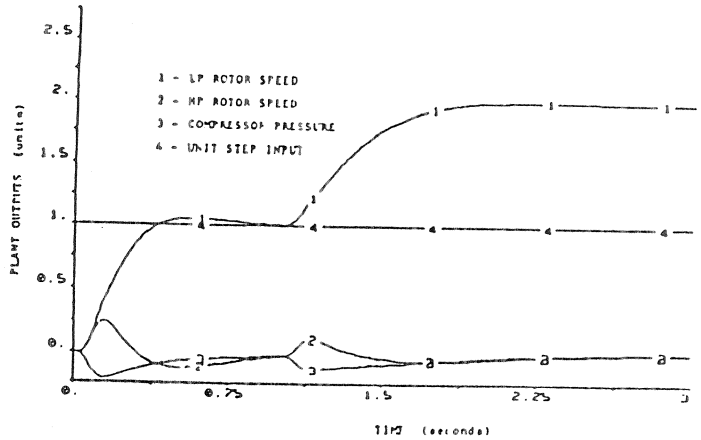
MCC–4: GE-21 Engine Simulation

- actuator, controller faults: sluggish transient behaviour (lower gain)
- sensor faults: disturbance injection (reference doubling)

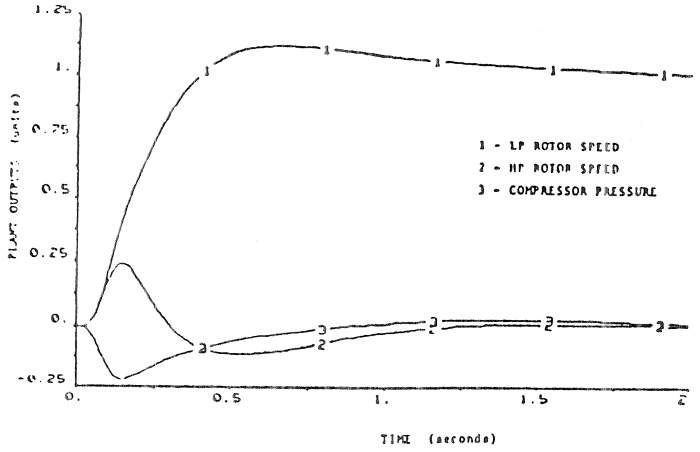
Nominal Step Response



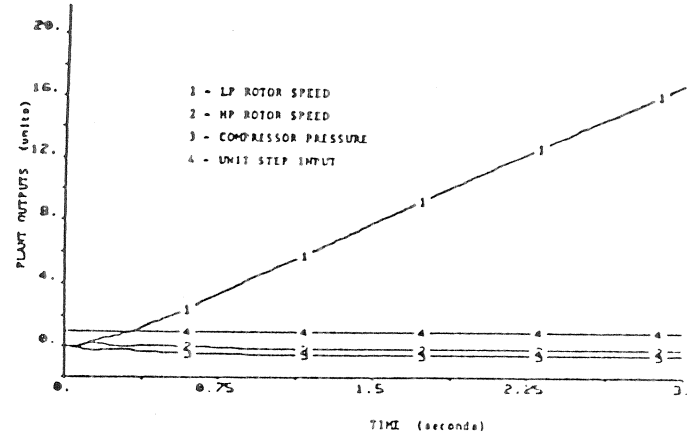
MCC: Sensor Failure



MCC: Actuator Failure



Standard Sensor Failure



FMS: Simultaneous Stabilization-1

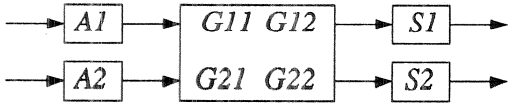
- Let k be the number of failure modes, i.e. k = # of actuators + # of sensors
- Let $P_0, ..., P_k$ be the models of the associated system

Then *Simultaneous Stabilization* involves finding a single compensator C that stabilizes each of the plant models $P_0, ..., P_k$

FMS: Simultaneous Stabilization-2

- Sensor, actuator failures modeled as structural perturbations

  Example:



$$\Rightarrow \qquad P = \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix}$$

- Sensor 2 fails: $y_2 = 0$
- Actuator 1 fails: $u_1 = 0$

$$P_{02} = \begin{bmatrix} P_{11} & P_{12} \\ 0 & 0 \end{bmatrix} \qquad P_{10} = \begin{bmatrix} P_{11} & 0 \\ P_{21} & 0 \end{bmatrix}$$

- In general, P_{ij} represents the equivalent system model with failures in the i^{th} actuator and j^{th} sensor.
- No redundant hardware required ("Soft" Redundancy)
- *Theoretically* possible to design for performance under failure
- Certain advantages afforded by non-square systems

SS-3: Existence Results

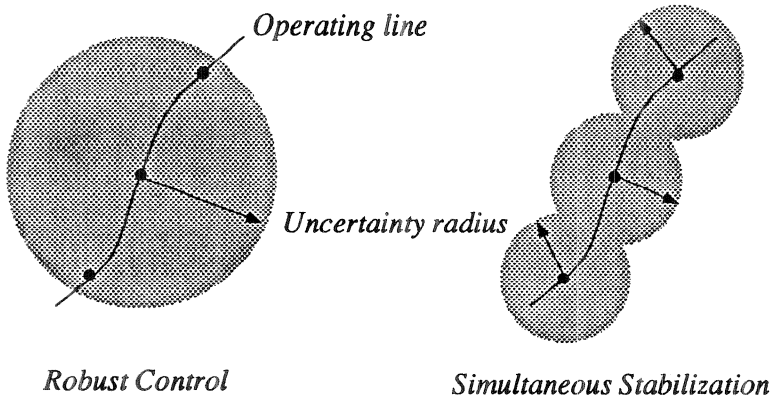
- System is tolerant to single failures if C stabilizes P, P_{01}, P_{02}, P_{10}, P_{20}
- Tolerant to multiple failures if C also stabilizes P_{ij} for all i, j

Fact: **(Linear time-invariant controllers)**

- SISO case
  - — Solution exists if and only if an "auxiliary plant" can be stabilized by a stable compensator (note similarity to MCC)
- MIMO case
  - — Given plants P_0, $P_1, ..., P_k$ of dimension $l \times m$, then *generically* there exists a simultaneously stabilizing controller if $k < \max(l, m)$

SS-4: SS vs. Robust Control

- "performance" $\propto \dfrac{1}{\text{uncertainty radius}}$



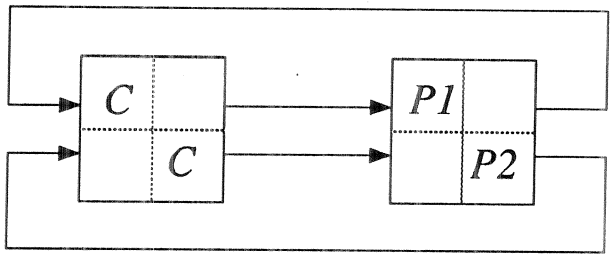
- Controller complexity is high, due to "internal models" for each plant

SS-5: Design Methods

- Several design approaches
  - — Direct state-space*
  - — Stable Factorization*
  - — Constrained Decentralized Control
- Applications to date motivated by:
  - — multiple operating points (power generation, propulsion, flight control)
  - — multiple-plants (T700–Apache, T700–Blackhawk)
- Outstanding Issues:
  - — Extension of stabilization to performance
    - → simultaneous loop-shaping, local pole-placement
  - — Fault coverage
    - → genericity result restrictive, but likely conservative ($k < \max(l, m)$)
    - → multirate compensation promises stabilization for any *finite* number of plants

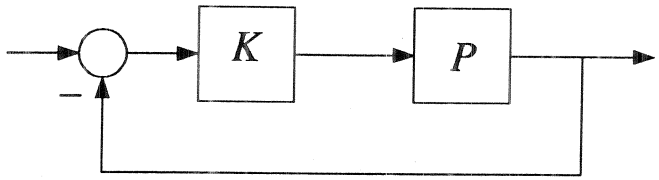
SS-6: Current Research

- Decentralized Control



  - Constrained decentralized control law design
    - — non-linear programming
    - — optimal constrained-structure output feedback

- Parametrization of Simultaneously Stabilizing Controllers
  - — Why parametrization? → systematic design

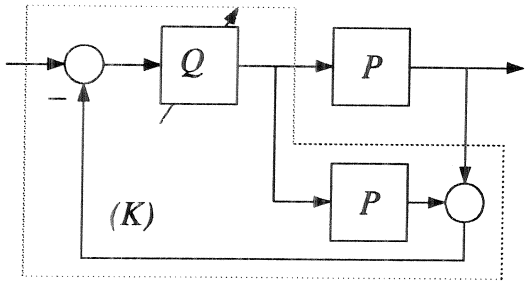


Nominal Unity-gain Feedback

SS-7: Parametrization

Case 1: P stable,

$K = Q(I - PQ)^{-1}$

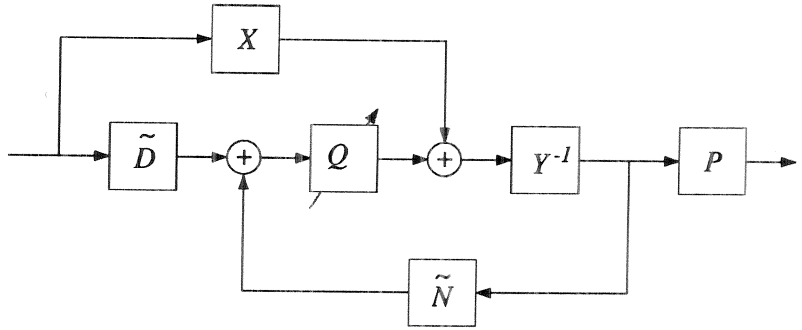


"Q" Parametrization

Case 2: P unstable,

$P = ND^{-1} = \tilde{D}^{-1}\tilde{N}$

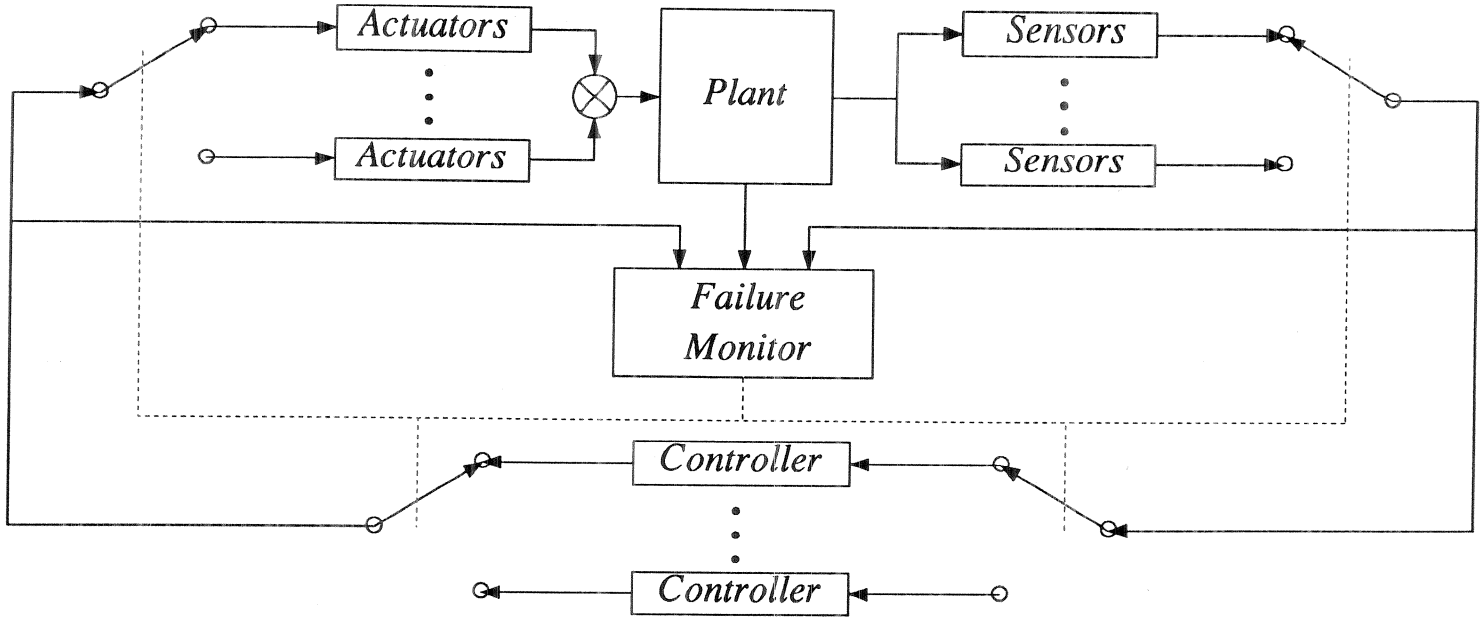
$XN + YD = I, \quad \tilde{N}\tilde{X} + \tilde{D}\tilde{Y} = I$



"Stable Factorization"

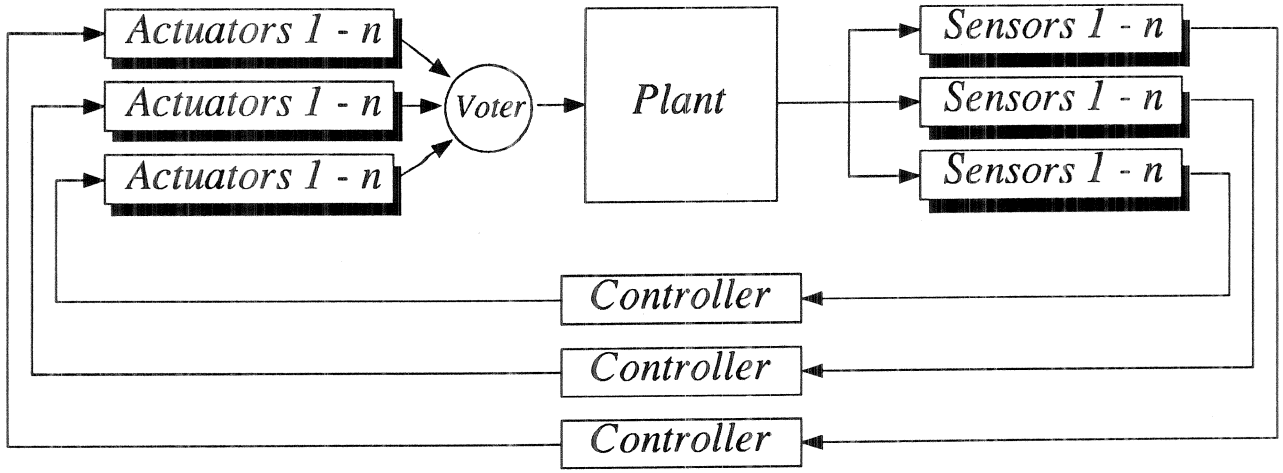
- Q parametrizes *all* stabilizing compensators
- Optimal selection of Q (H^∞, ISICLE, etc.)
- SS parametrization payoff? $\rightarrow$ reliability, *tuned* performance, *reduced complexity*

General Reconfiguration Scheme



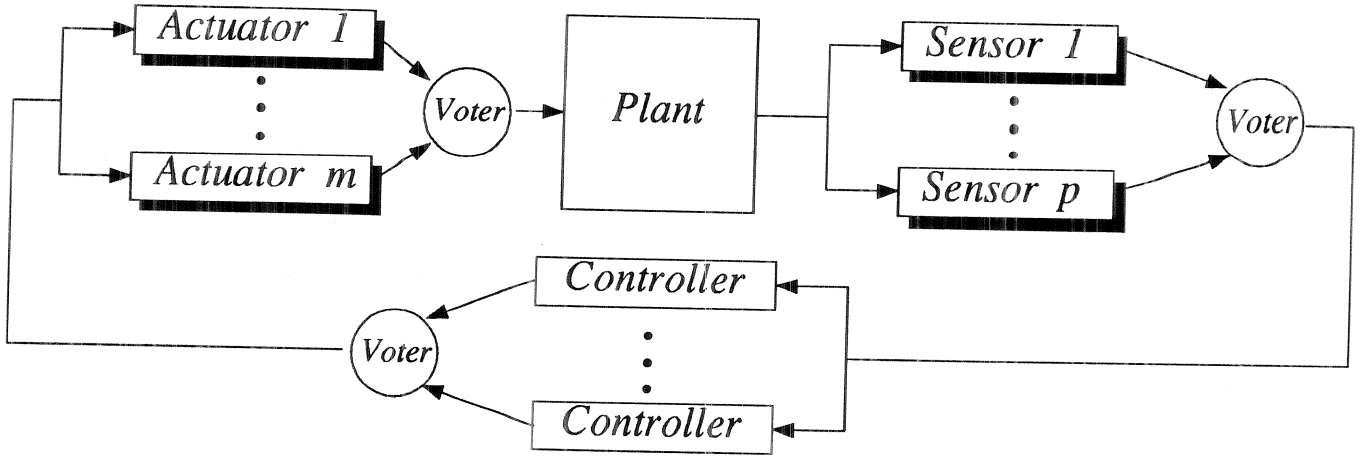
- Issues:
  - FDIR techniques, FDIR time
  - Switching probability and false alarms
  - Coverage

RCS: System-Level TMR



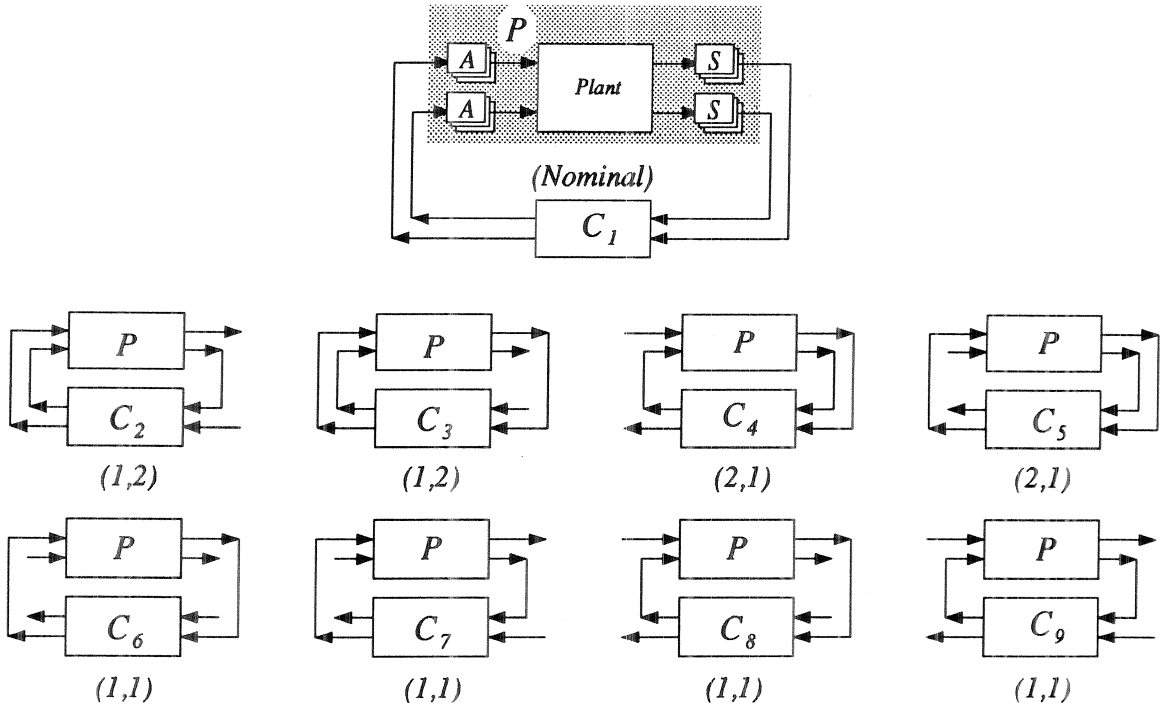
- Normally all three control channels are active
- One component failure deactivates entire channel, system reconfigured as duplex or simplex
- When all three channels fail, the control system fails
- Control system design is standard

RCS: Component-Level TMR



- Normally all units are active
- On component failure, TMR units reconfigure as duplex or simplex
- System fails when any one unit is exhausted
- Control system design is standard

RCS: Controller Switching Scheme



- Use TMR for high reliability in actuator, sensor subsystems
- Require Coverage equal to 1 if loop failures destabilizing
- "Bumpless Transfer" during controller switching
- Exponential complexity in I/O dimension

4. Fault Detection and Isolation

- Introduction
- FDI: Complexity
- Failure Propagation
- Control Charts
- Fault Trees
- Parity Space Techniques
- Detection Filters
- Observers

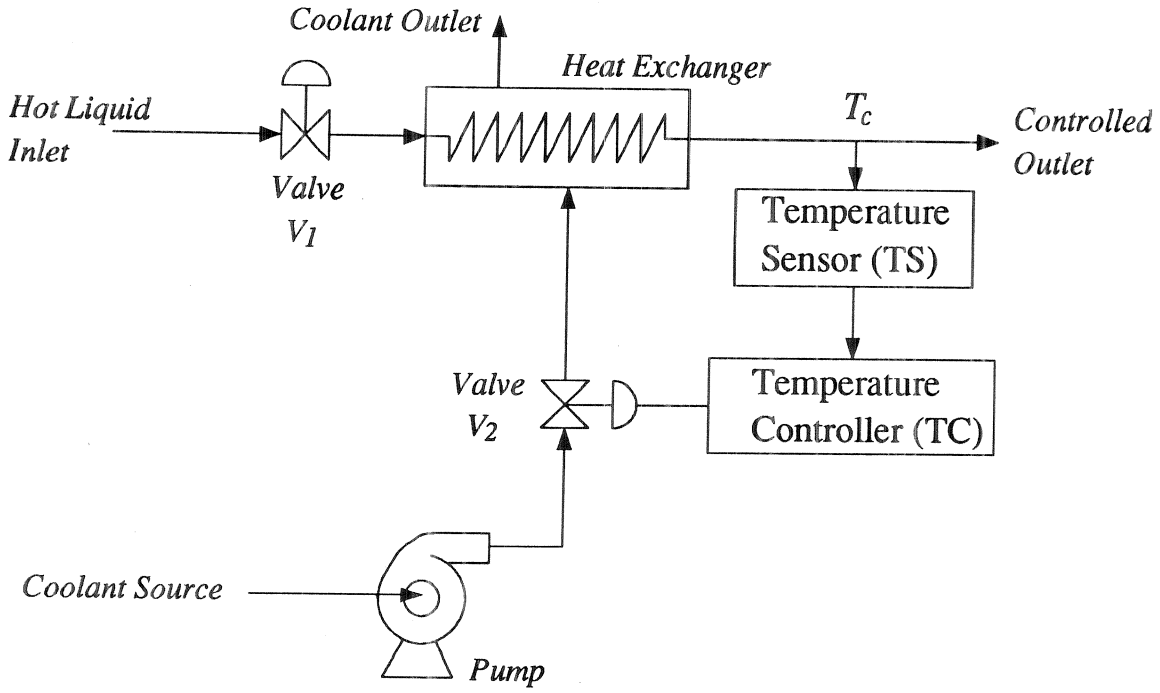
FDI: Introduction

- Very widely studied
- Nuclear, Power, Chemical, and Control Engg. Disciplines, as well as Medicine
- Control room design, Alarm processing, Medical diagnosis
- Techniques include:
  - Control Charts
  - Fault Trees
  - Digraphs
  - Parity Space Techniques
  - Observers
  - Detection Filter
  - Kalman Filter
  - Parameter Estimation
  - Expert Systems

FDI: Complexity

- Systems consist of large number of components interconnected in a complex way
- Each component can fail in a variety of ways
  - a valve may fail
    - → stuck open
    - → stuck closed
    - → stuck at a point
    - → and in many other ways ...
- Fault at one component propagates to others
  - Component connectivity provides these paths

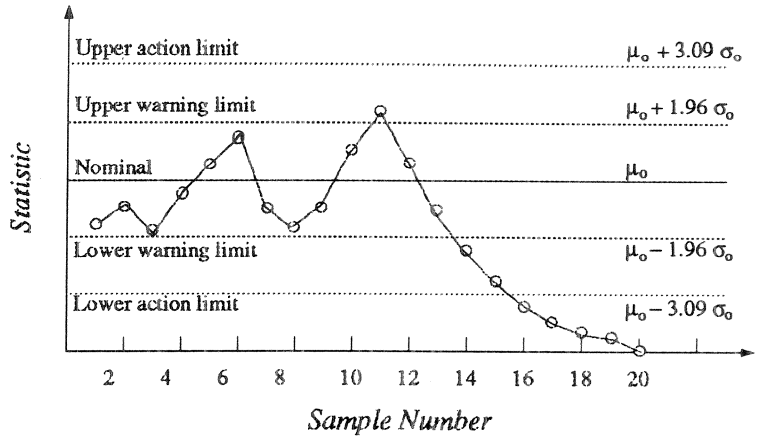
FDI: Failure Propagation



- A wild inrush of hot liquid will result in high T_c reading, high TS and TC settings while valve V_2 opens fully.
- All alarms attached to these components sound
- Every component appears to be faulty!

FDI: Control Charts

- Graphical means to monitor the system
- Plot of simple statistic such as sample mean
- Separate chart required for each variable



- System subjected to disturbances → response normally distributed with mean μ_o and variance σ_o. Structural changes → response exceeds action limits
- Statistic outside action limits indicates *"strong evidence of failure"*

FDI: Fault Trees

- Fault tree models graphically and logically portray various combinations of events that would eventually lead to System Failure
- The root of a fault tree is the System Failure event, denoted the **Top Event**
- Fundamental units are
  - — AND gate
  - — OR gate
- The subfailures that lead to the Top Event form the nodes at the next level and are connected to the Top Event by AND or OR gates.

Fault Isolation Using Fault Trees

- Gate Search Algorithm

Step 1. **Start** with A.

Step 2. Causative and TRUE events
B and C.
New top events: B and C.

Step 3. **Start** with B.

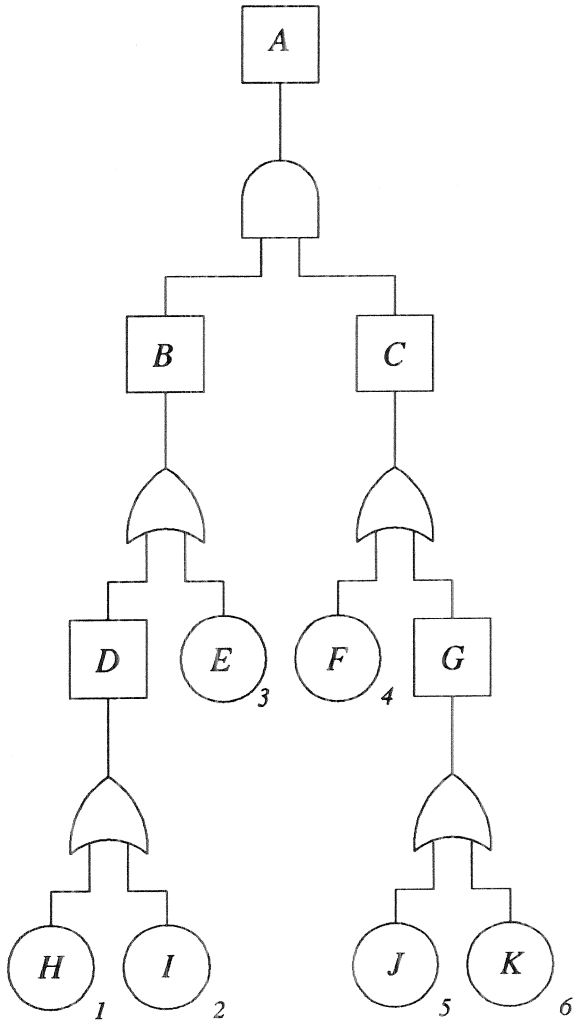
Step 4. Verify D — TRUE.
Current gate: D

Step 5. Verify I — FALSE.
Verify H — TRUE.

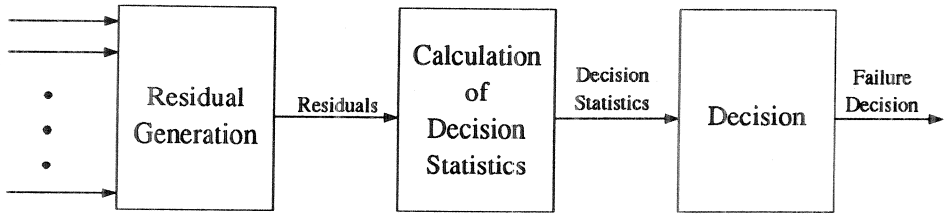
Step 6. **Start** with C.

Step 7. Verify G — TRUE.
Current gate: G

Step 8. Verify K — TRUE.
Stop.



FDI: Parity Space Approach



- The FDI process is conceptually
  - Residual Generation
    - → Residuals are difference between observed and expected sensor outputs
    - → Nominally "zero" with distinctive failure signatures
  - Decision Making
    - → Residuals are examined for failures
    - → One may use

      Simple threshold tests

      SPRT, etc.

Parity Vector: Example

- Suppose y_1, y_2 and y_3 are measurements of the same variable
- Define residuals: $\rho_1 = y_1 - y_2, \quad \rho_2 = y_2 - y_3$
- Decision rules:

  ρ_1 small, ρ_2 small $\rightarrow$ no failure

  ρ_1 large, ρ_2 small $\rightarrow$ y_1 bad

  ρ_1 small, ρ_2 large $\rightarrow$ y_3 bad

  ρ_1 large, ρ_2 large $\rightarrow$ y_2 bad

- Complete detection and isolation of n single-point failures possible using $n - 1$ parity relations

Parity Vector: Generation

- Direct Redundancy
  - — Suppose we have n sensors measuring p variables, $n > p$
  - — Outputs are algebraically related, i.e.
    $y = Cx; \quad y \in \Re^n, \quad x \in \Re^p, \quad C \in \Re^{n \times p}$
  - — Rows of C are linearly dependent i.e. we can find α such that $\alpha^T C = 0$
  - — We can find $(n - p)$ such α's which are linearly independent
- Define residual: $\rho = \alpha^T y = \alpha_1 y_1 + \alpha_2 y_2 + \cdots + \alpha_n y_n$
- $\rho < \epsilon \rightarrow$ no failure; $\rho > \epsilon \rightarrow$ failure present
- Partial or complete isolation possible using $(n - p)$ parity relations

Parity Vector: Temporal Redundancy

- One can use relationships among the time histories of inputs and outputs to generate parity vectors
- We can exploit the relationship between sensors for acceleration and velocity (for instance) for FDI

  Model: $v(k+1) = v(k) + Ta(k)$

  Parity relation: $\rho(k+1) = v(k+1) - v(k) - Ta(k)$

- Actuator FDI:

  Model: $y(k+1) = y(k) + Tu(k)$

  Parity relation: $\rho(k+1) = y(k+1) - y(k) - Tu(k)$

  If sensor is good and $\rho(k) > \epsilon$, then actuator is bad.

Generalized Parity Vectors

- Model:

$$x(k+1) = Ax(k) + b_1 u_1(k) + \cdots + b_m u_m(k)$$
$$y_j(k) = c_j x(k); \quad j = 1, 2, \cdots, p$$

- Observability matrix: $O_j(k) = [c_j' \quad (c_j A)' \quad \cdots \quad (c_j A^k)']'$
- Define n_j, n_o such that:

$$\begin{aligned} \text{Rank } \mathbb{O}_j &= k+1; \; k < n_j \\ &= n_j \; ; \quad k \geq n_j \\ n_o &= \max\left(n_j; \; j = 1, 2, \cdots, p\right) \end{aligned}$$

$$\begin{bmatrix} y_j(k) \\ y_j(k+1) \\ \vdots \\ y_j(k+n_j) \end{bmatrix} = \begin{bmatrix} c_j \\ c_j A \\ \vdots \\ c_j A^{n_j} \end{bmatrix} x(k) + \begin{bmatrix} 0 & 0 & \cdots & 0 \\ c_j B & & \cdots & 0 \\ & & \vdots & \\ c_j A^{n_j-1} B & & \cdots & 0 \end{bmatrix} \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+n_o) \end{bmatrix}$$

$$\mathbb{Y}_j(k; n_j) = \mathbb{O}_j(k; n_j)\, x(k) + \mathbb{B}_j(n_j)\, \mathbb{U}(k, n_o)$$

Single Sensor Fault Diagnosis

- Let ω_j be a vector such that

$$\omega_j \mathbb{O}_j (n_j) = 0$$

- Then

$$\rho_j (k) = \omega_j [\mathbb{Y}_j (k) - \mathbb{B}_j \mathbb{U} (k)]$$

- Suppose all actuators are good;

  then $\rho_j(k) > \epsilon \Rightarrow$ Sensor j is bad.

Generalized Parity Space

- We have

$$\omega_j \, \mathbb{O}_j\left(n_j\right) = 0;\ j = 1, 2, \cdots, p$$

$$\begin{bmatrix} \omega_1 & \omega_2 & \cdots & \omega_p \end{bmatrix} \begin{bmatrix} \mathbb{O}_1\left(n_1\right) \\ \vdots \\ \mathbb{O}_p\left(n_p\right) \end{bmatrix} = 0$$

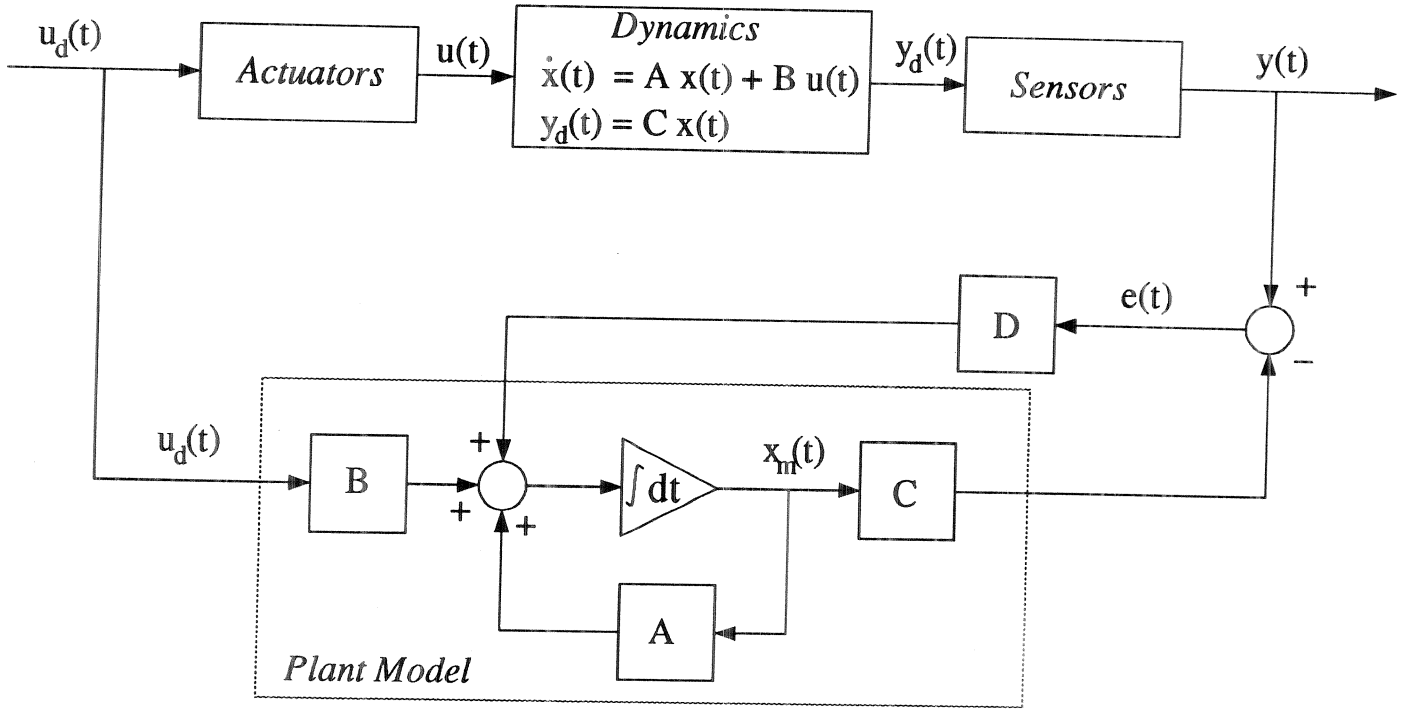
- Defines a reduced-order model for the system
- Let Q denote the basis of all such ω vectors
- Then define:

$$\rho\left(k\right) = Q \left[\begin{bmatrix} \mathbb{Y}_1\left(k; n_1\right) \\ \mathbb{Y}_2\left(k; n_2\right) \\ \vdots \\ \mathbb{Y}_p\left(k; n_p\right) \end{bmatrix} - \begin{bmatrix} \mathbb{B}_1\left(n_1\right) \\ \mathbb{B}_2\left(n_2\right) \\ \vdots \\ \mathbb{B}_p\left(n_p\right) \end{bmatrix} \mathbb{U}\left(k; n_o\right) \right]$$

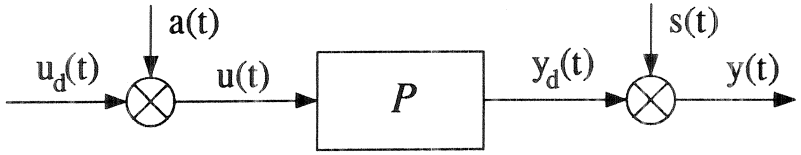
- $\rho(k)$ characterizes all redundancies in the system useful for FDI

The Detection Filter (DF)

- A special-purpose observer



DF-1: Sensor & Actuator Failure Models



- i^{th} Actuator failure:

  $u = u_d + [0 \quad \cdots \quad 0 \quad 1 \quad \cdots \quad 0]' a_i(t)$

- j^{th} Sensor failure:

  $y = y_d + [0 \quad \cdots \quad 1 \quad 0 \quad \cdots \quad 0]' s_j(t)$

- Failure modes
  - Complete sensor (actuator) failure: $s_i = -y_{d_i} \quad (a_i = -u_{d_i})$
  - Bias: $s_i \ (a_i) =$ constant
  - Noise: $s_i = s_i(t) \quad (a_i = a_i(t))$
- Operating Conditions
  - Nominal: $y - Pu_d = 0$
  - Sensor failure: $y - Pu_d = s_i$
  - Actuator failure: $y - Pu_d = Pa_i$

DF-2: Dynamics of the Detection Filter

- Nominal Plant

$$\dot{x} = Ax + Bu$$
$$y = Cx$$

- Detection Filter

$$\dot{x}_m = (A - DC)\, x_m + Bu + Dy$$
$$y_m = Cx_m$$

- Failed Plant

$$\dot{x} = Ax + Bu + f_1 a_1 + \cdots + f_m a_m$$
$$y = Cx + \hat{e}_1 s_1 + \hat{e}_1 s_1 + \cdots + \hat{e}_p s_p$$

- Error Signals

$$e = x - x_m$$
$$\bar{e} = J\,(y - y_m)$$

  - "No Failure" Dynamics

$$\dot{e} = (A - DC)\, e$$
$$\bar{e} = JCe$$

  - Choose $(A - DC)$ stable, then

$$e(t) \;\; \& \;\; \bar{e}(t) \;\; \rightarrow \;\; 0 \;\; \text{as} \;\; t \;\; \rightarrow \;\; \infty$$

DF-3: Fully Measurable Case

- Suppose $C = I_n$ i.e. there are as many measurements as states

  — choose D such that $(A - DC) = -\sigma I$.

- Under no failure conditions

  $\dot{e} = -\sigma e; \quad \bar{e} = e$

- In case of actuator failure, then

  $\dot{e}(t) = -\sigma e(t) + f_i a_i(t)$

  or

  $e(t) = f_i \left[\int_o^t e^{-\sigma(t-\tau)} a_i(\tau) d\tau \right]$

- $e(t)$ propagates along f_i thus signifying failure in the i^{th} actuator

- Similarly, one can show that the persistent error due to sensor failure lies in a plane formed by two distinct vectors

DF-4: General Case

- We have
$$\bar{e}(s) = \left[JC(sI - A + DC)^{-1}F\right] a(s) \;=\; G(s)\,a(s)$$
- Choose J and D such that "$G(s)$" is diagonal
- $\bar{e}_i(s) = g_i a_i [0 \quad 0 \quad 1 \quad 0 \quad \cdots \quad 0]'$ (i^{th} row)
- Failure in the i^{th} actuator propagates along $\hat{e}_i$
- Monitor $\bar{e}_i$ and its direction. Any activity along $\hat{e}_i$ denotes i^{th} actuator failure.

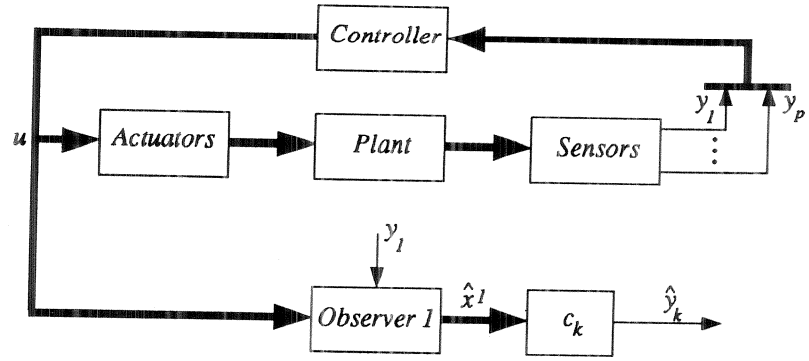
FDI Using Observers

- Primarily useful for sensor fault-detection and isolation
- Plant model

$$\dot{x} = Ax + Bu$$

$$y_i = c_i x; \quad i = 1, \ldots, p$$

- Let (c_1, A) be observable



- Construct an observer using u and y. Let $\hat{x}$ be the state estimate.
- Define: $\hat{y}_k = c_k \hat{x}$, $e_k = y_k - \hat{y}_k$ (residual)
  - Rule 1: $e_k = 0$ for all $k \rightarrow$ no fault
  - Rule 2: $e_k = 0, k \neq j; \quad e_j \neq 0, j \neq 1 \rightarrow j^{th}$ sensor bad
  - Rule 3: $e_k \neq 0$ for all $k \rightarrow$ sensor 1 is bad

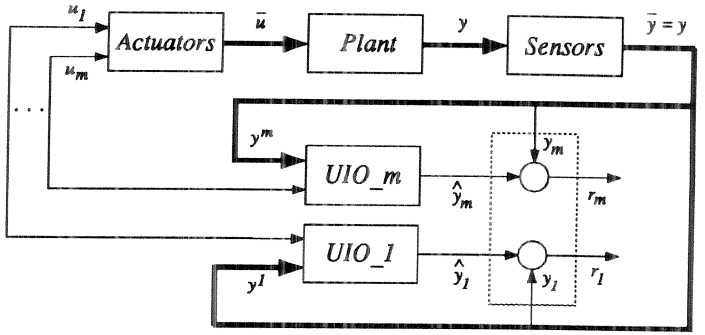
FDI: Unknown Input Observers (UIO)

- Extend Observer Approach to Actuator FDI
- Plant model

$$\dot{x} = Ax + Bu + Ed; \quad d \to m \times 1$$

$$y = Cx; \quad y \to p \times 1$$

d is unknown input



- UIO exists if and only if: (i) $p \geq m$, (ii) Rank $CE = m$, (iii) Invariant zeros of (C, A, E) are stable

  — Rule 1: $r_i \neq 0$ and $r_j = 0, j \neq i \to i^{th}$ actuator is bad

  — Rule 2: $r_i \neq 0$, $i \in \{i_1, \ldots, i_k\}$ and $r_j = 0$ otherwise $\to$ actuators $i_1, \ldots, i_k$ are all bad

  — Rule 3: If $r_i \neq 0$ for all i, then a sensor has failed

5. Fault-Tolerant Control Computers (FTCCs)

- Requirements
- Currently available systems
- Fault-tolerant multiprocessor (FTMP)
- Fault-tolerant Parallel Processor (FTPP)

FTCC: Requirements

- Computer Controller has three functions (Three-stage pipe)
  - — Data Acquisition.
  - — Control Law Computation.
  - — Output and Display.

- The Workload is a fixed group of tasks executed repetitively at a frequency sufficient to maintain stability and performance (sampling time).

- Real-time control tasks are critical (hard dead-line)

  - — If response time exceeds nominal value, dynamic catastrophic failure results.

- FTCCs should have high Reliability (10^{-9} failures/hour) and high Throughput (50+ MIPS).

FTCC: Requirements-2

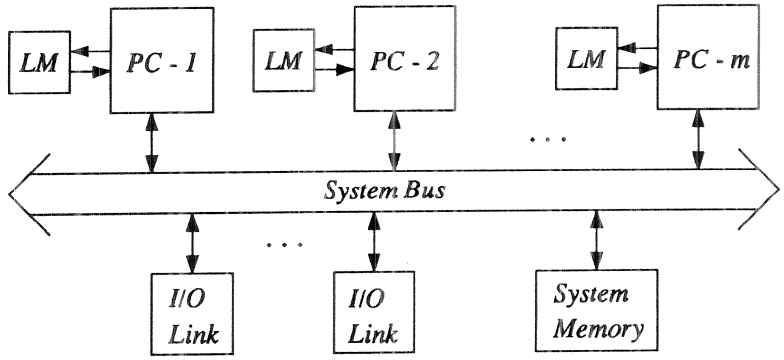
- Dynamic scheduling and System reconfiguration of the redundant elements should provide enough fault-tolerance so that mission goals could be accomplished in the presence of failures.
- Degraded levels of workload should be defined for each degraded FTCC configuration
- Damage Tolerance is another parameter which can be accomplished with loose coupling and physical separation.

FTCC: Some Architectures

- FTMP : Fault-Tolerant Multiprocessor
- SIFT : Software Implemented Fault-Tolerance
- FTPP : Fault-Tolerant Parallel Processor
- MAFT : Multicomputer Architecture for Fault-Tolerance
- ..

FTMP : System Architecture & Operation

- Four Major Components
  - — Processing Clusters.
  - — Input / output Links.
  - — Time Shared System Bus.
  - — System Memory.



PC: Processing Cluster
LM: Local memory

- Processing Clusters.
  - — Each cluster works on a single task at a time
  - — Each cluster could be a TMR with local memories
  - — All clusters are identical

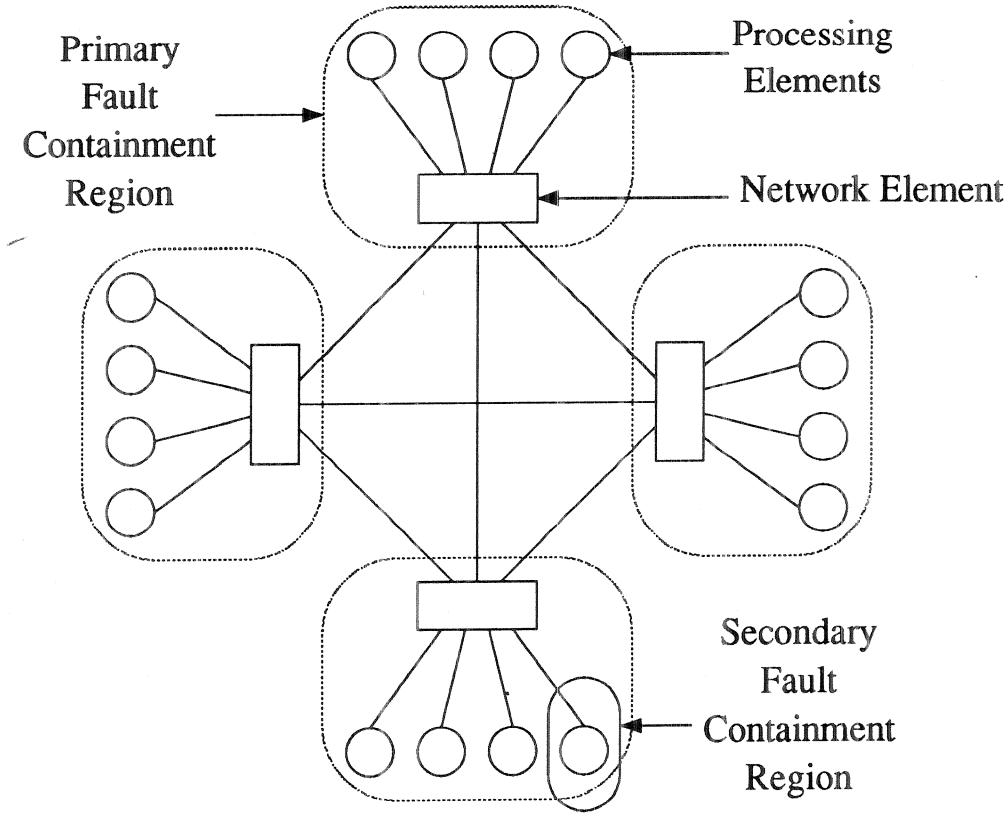
FTMP : System Architecture & Operation-2

- Input / output Links.
  - — Allows to read data from sensors.
  - — Transmits data to actuators.

- System Bus.
  - — Interconnects all clusters, I /O links and memories.
  - — Bus is redundant but acts as a Unibus. Only one cluster transmits and receives data over all copies of the bus at a time.
  - — Bus access is determined by polling.

- System Memory.
  - — Redundancy for reliability reasons.
  - — Only one memory unit is addressable over the Bus.

Fault-Tolerant Parallel Processor (FTPP)



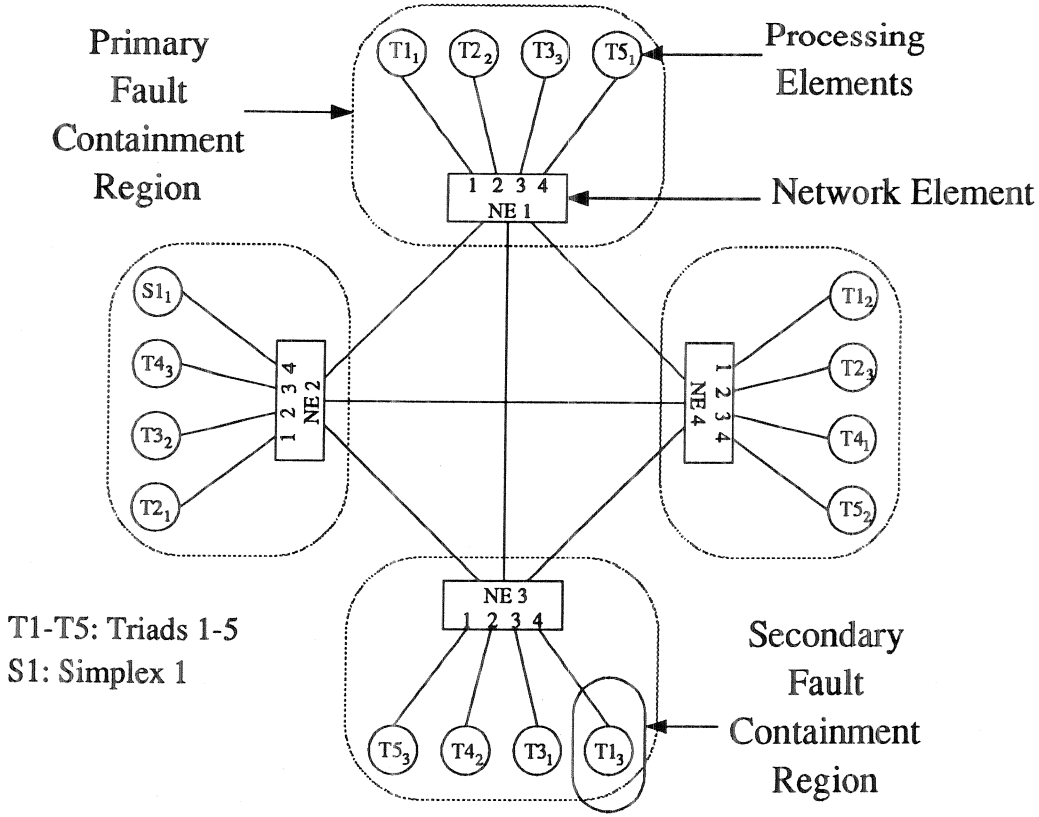
Fault-Tolerant Parallel Processor (FTPP)

- FTPP has several Clusters.
- Each Cluster has Network Elements (NEs) and Processing Elements (PEs).
- Network Elements are fully connected.
- The Network elements host one or more Processing Elements.
- The Network element and the Processing elements connected to it form a Fault Containment Region (FCR).
- The failure probabilities of different FCRs are independent.

FTPP : Configuration

- The Cluster is configured in to Virtual Groups.
- Each Virtual Group is a Fault Masking Group (FMG) or a simplex.
- An FMG has (2n+1) PEs executing (2n+1) identical copies of computation.
- Output comparisons provide masking of any n processor faults
- A minimal FMG is a TMR
- No two members of the same FMG subscribe to the same network element.

Fault-Tolerant Parallel Processor (FTPP)



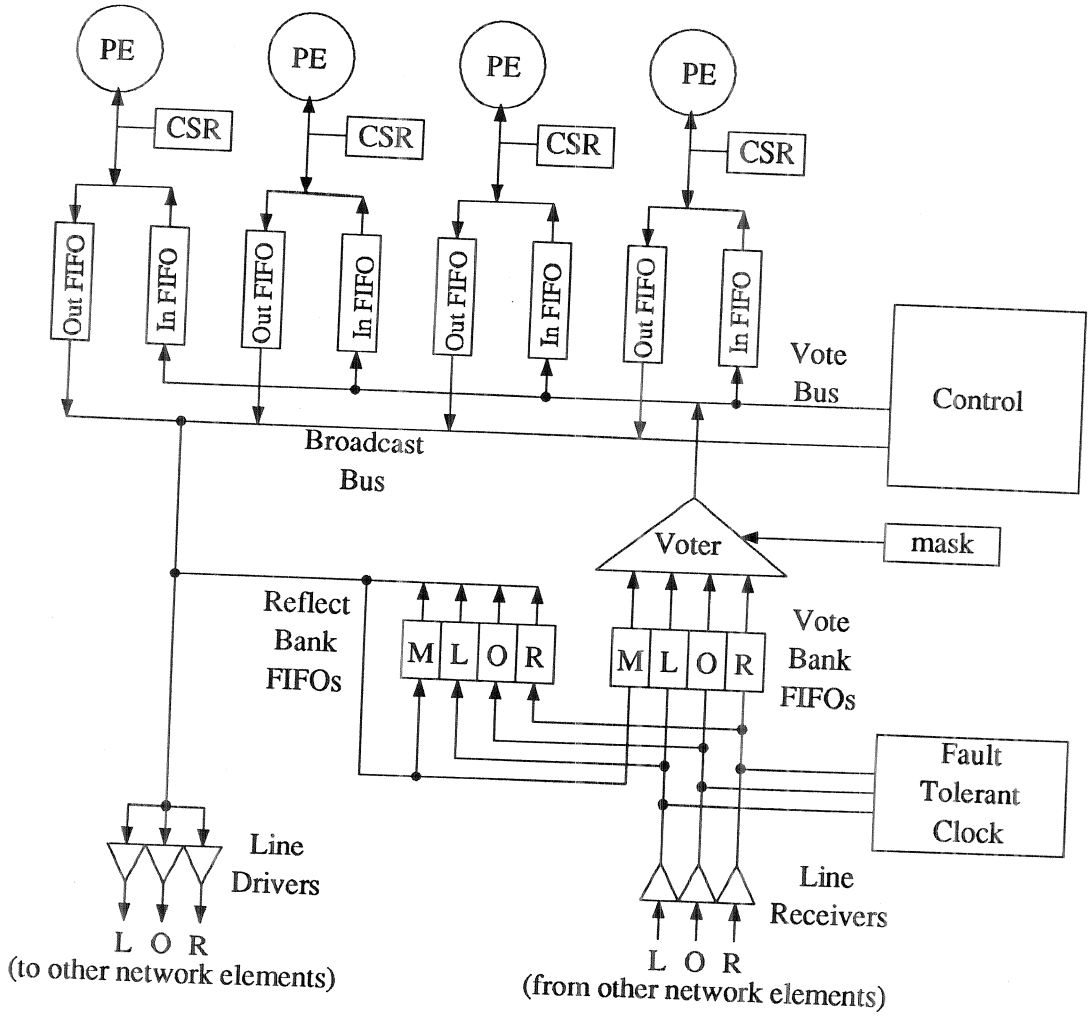
FTPP: Operation

- Sending a message
  - PE will check out FIFO for buffer space
  - If out FIFO is full then PE is blocked.
  - If out FIFO has space, PE will write the message in it.
  - PE will also write in control FIFO the exchange class and the source member #.

- Receiving a message (Through In FIFO)
  - NE checks destination free condition before writing the message in processor IN FIFO.
  - NE transmits if IN FIFO is free, otherwise holds the message for the next cycle.

- Sensitivity Analysis
  - Reliability as a function of # of processors, sensors, actuators

Fault Containment Region (FCR)



FTPP : Network Element Operation

- Local Exchange Request Pattern (LERP) is the status of exchange requests from PEs connected to a NE.
- System Exchange Request Pattern (SERP) is the aggregate of all LERPs.
- All NEs have the same copy of SERP.
- Valid message condition (VMC) is satisfied when all FMGs request for trasmission
- Destination free condition (DFC) is satisfied when INFIFOs of destination FMG are free.
- Control Logic determines the set of enabled messages i.e. messages for which VMC & DFC are satisfied.

6. Software Fault-Tolerance

- Definition of Software Reliability
- How to achieve it?
- Recovery Blocks
- N-Version Programming

Software Reliability & Availability

- Defined as in the case of hardware systems

Definition: (Reliability)

"Software *reliability* is the probability that the program performs successfully, according to specifications, for the given mission period."

Definition: (Availability)

"Software *availability* is the probability that the program is performing successfully, according to specifications, at a given point in time."

Software Fault-Tolerance: How to Get It?

- Software tolerance of design faults is a frequent specification
- Fundamental vehicle for covering design faults is extensive use of *design redundancy*
- Fault-tolerant programs contain multiple routines that are produced from "independent" algorithms, written "independently", based on a common specification for the same computation
- Fault-tolerant programs require
  - Design redundancy
  - Hardware redundancy
  - Execution time redundancy

Software Fault-Tolerance: Recovery Blocks

- A *recovery block*, a language construct, is any segment of a large program containing
  - Primary block
  - Acceptance test
  - Sequence of alternate blocks
- All blocks designed to yield the same computational result
- The *acceptance test* is a logical expression to determine the acceptability of the execution results from the blocks.
- Example tests:
  - Matrix positive definite?
  - An equation holds?
- Recovery block:

Ensure A_1

by B_1

else by B_2

else by B_n

else error

Software Fault-Tolerance: N-Version Programming

- N-Version programming involves the execution of several *independently* developed versions of a software algorithm and a voting system that outputs a majority result
- *Independent* means programming efforts are carried out by individuals or groups who do not interact and possibly use different algorithms, programming languages or translators
- The conjecture is that independent efforts would greatly reduce correlated errors

7. Dependability Analysis of FTCS

- Performance Measures
- Coverage
- Reliability Analysis of TMR
- Numerical Solutions
- Software Packages
- An Illustration

Performance Measures-1

- Probabilities
  - System failure due to *Attrition*
  - System failure due to *Imperfect Coverage.*
  - System failure during *Mission time.*
  - System operation with *specified performance degradation.*
  - System operation above a *critical level of performance.*
  - *Reliability*
  - *Unreliability*

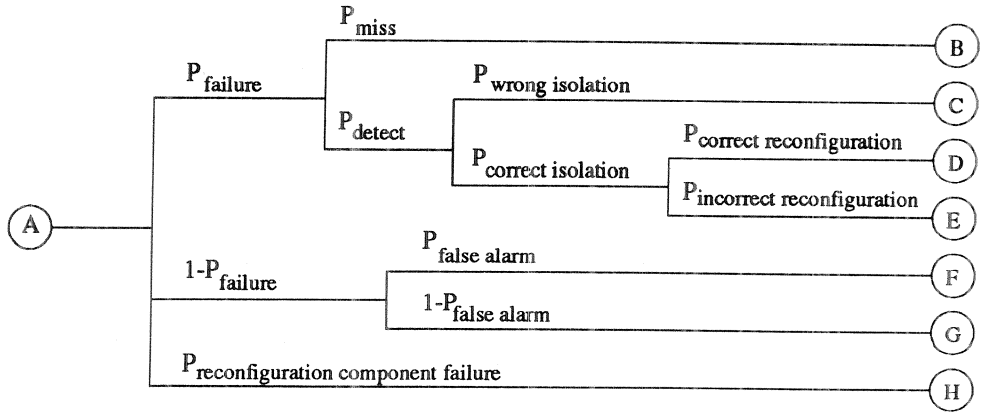
- Dynamic probability of failure
  - Failure to complete execution within *dead-line*

Performance Measures-2

- Statistics of Random Variables
  - — MTTSF: Mean time to System Failure.
    - → Primarily used for comparing the resilience of different architectures to attrition i.e. the exhaustion of system resources due to failures
  - — ENNFE (j) : Expected Number of Nonfaulty Elements of Type j.
- Sensitivity Analysis
  - — Reliability and MTTF as function of # of *Processors, sensors, Actuators,* and *Coverage.*

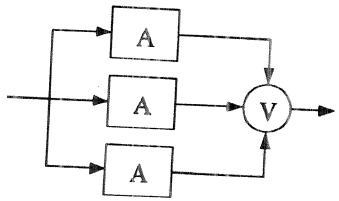
Coverage

- Coverage = Prob. (system recovers | fault occurs)
- Coverage reflects the ability of the system to recover from failure that occurs at a particular operating condition.
- Coverage is < 1
  - — Near Coincident Faults
  - — Single Point Failures
  - — Imperfect FDI
  - — Reconfiguration Time
- Coverage is very difficult to quantify



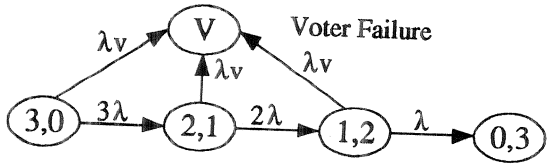
Reliability Analysis of TMR

- TMR Structure:

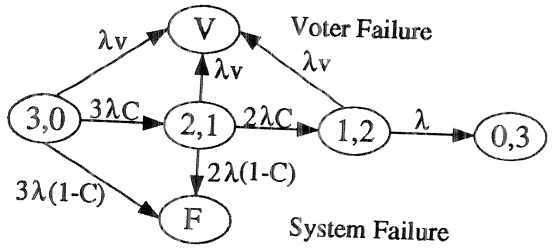


- Markov chain is a state transition diagram where each state depicts a particular operational configuration of the system

TMR Markov model: Perfect Coverage



TMR Markov model: Imperfect Coverage



- P(F) = Prob. of system failure due to imperfect coverage
- P(V) = Prob. of single point failure
- P(0,3) = Prob. of failure due to attrition

Numerical Solutions

- The reliability model leads to a Continuous-Time Markov Chain

$$\frac{dP(t)}{dt} = QP(t); \quad P(0) = \pi_o$$

$$P(t) = [P_1(t), P_2(t), \ldots, P_n(t)]$$

$P_i(t)$ = Prob. system is in state i at time t

Q = Infinitesimal Generator of the CTMC

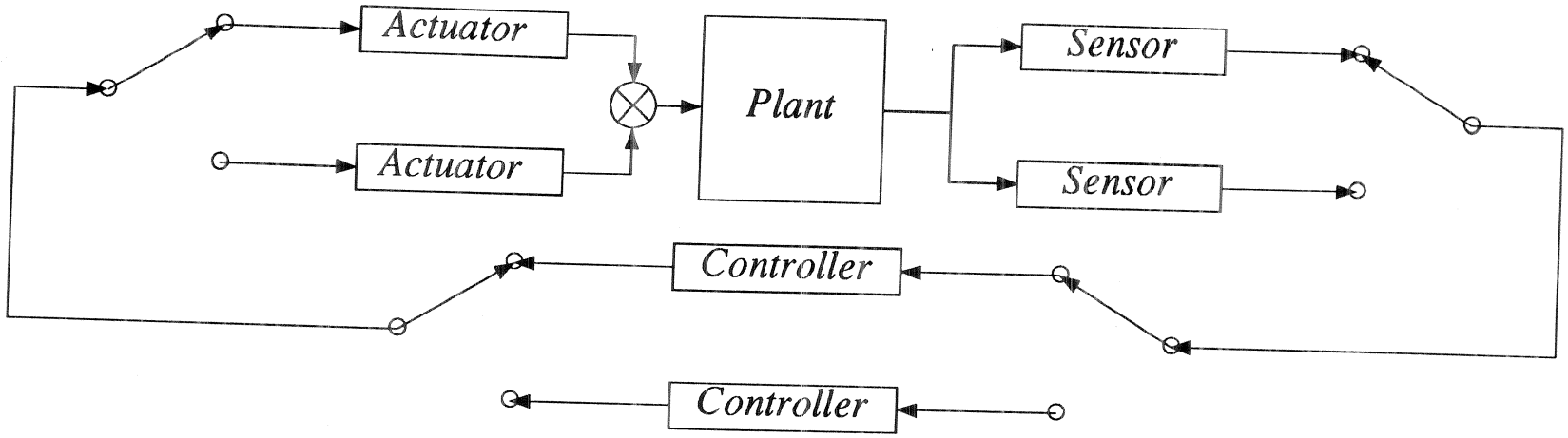
$$R(t) = \sum_{i \in \text{UP states}} P_i(t)$$

- Similarly other probabilitites can be calculated

Software Tools

- HARP : The Hybrid Automated Reliability Predictor
- SHARPE : Symbolic Hierarchical Automated Reliability and Performance Evaluator.
- SAVE : System Availability Estimator.
- SURE : Semi-Markov Unreliability Range Evaluator.
- ...

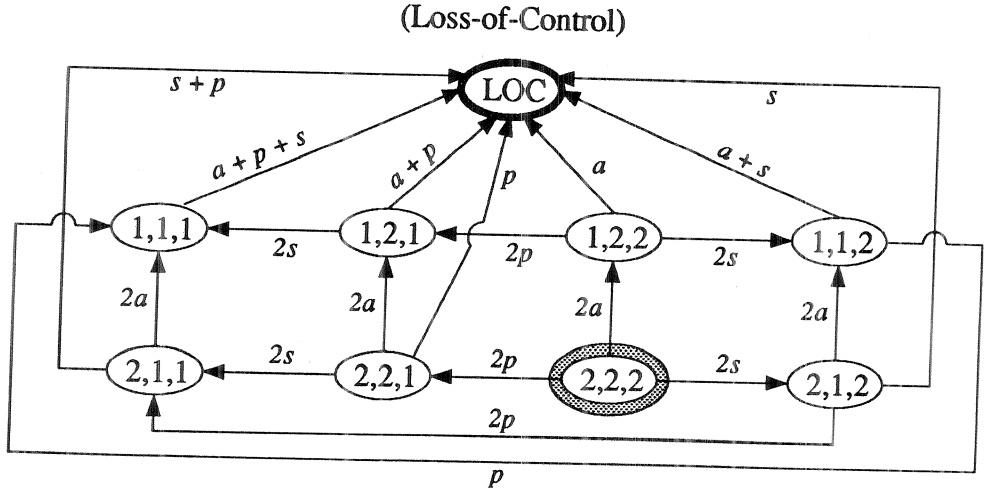
Analysis of FTC^{2}S: An Illustration



- System has two actuators, two sensors, and two controllers
- Normal operation possible with one actuator, one sensor, and one controller
- On failure of either component, reconfiguration takes place

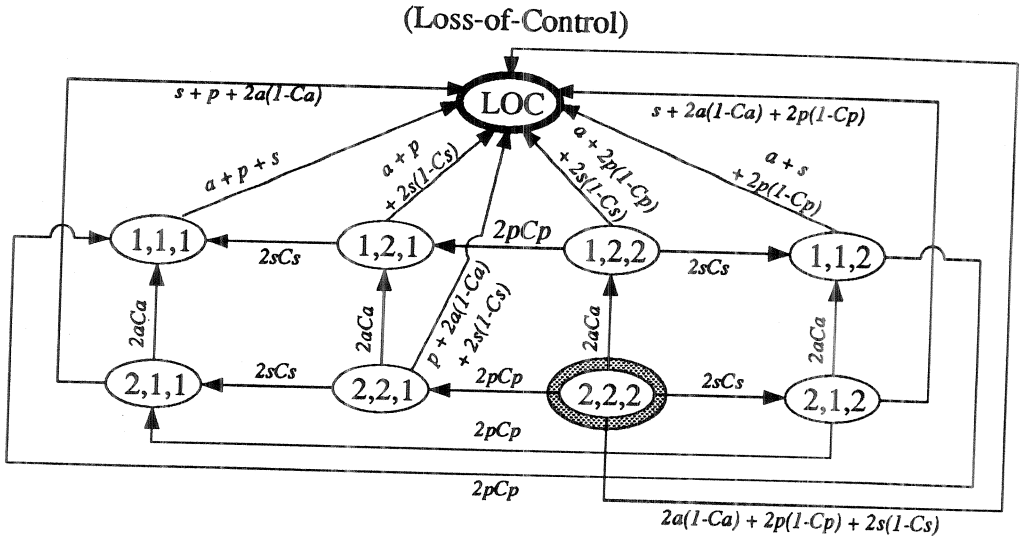
Analysis Illustration: Perfect Coverage

- Perfect Coverage model of (2,2,2) FTCS
- State (α,β,γ) = (α actuators working, β sensors working, γ processors working)
- a, s, p are failure rates of actuators, sensors and processors respectively



Analysis Illustration: Imperfect Coverage

- Imperfect Coverage model of (2,2,2) FTCS
- α, β, γ as before, *Ca*, *Cp*, and *Cs* are coverage probabilities for actuator, processor and sensor systems respectively



8. Conclusions

- We presented several *Redundancy Management Schemes* for achieving high degree of *fault-tolerance*.
- Design of *Stabilizing Controllers* for given failure modes is a standard problem.
- Available FDIR schemes for *Sensors, Actuators,*and the *Computer System* have to be analysed for *Coverage*.
- Integrated FTC^{2}SD Studies for *Bench Mark Problems* would show deficiences and indicate strategies for improvement
  - — Sophistication of FDIR.
  - — Speed of Reconfiguration
  - — Masking vs Reconfiguration.
  - — Sensitivity analysis w.r.t. redundancy, coverage, and failure rates