p-52
54-60
319675

N91-17563

# MAFT:
# The Multicomputer Architecture for Fault-Tolerance

R. M. KIECKHAFER

Computer Science and Engineering
University of Nebraska – Lincoln
Lincoln, NE 68588–0115
(402) 472–2402

rogerk@fergvax.unl.edu

MAFT is a product of the Allied-Signal Aerospace Company, Columbia MD.

## Abstract

This presentation discusses several design decisions made and lessons learned in the design of the Multicomputer Architecture for Fault-Tolerance (MAFT). MAFT is a loosely coupled multiprocessor system designed to achieve an unreliability of less than $10^{-10}/hr$ in flight-critical real-time applications.

The presentation begins with an overview of the MAFT design objectives and architecture. It then addresses the fault-tolerant implemention of major system functions in MAFT, including Communication, Task Scheduling, Reconfiguration, Clock Synchronization, Data Handling and Voting, and Error Handling and Recovery.

Special attention is given to the need for Byzantine Agreement or Approximate Agreement in various functions. Different methods were selected to achieve agreement in various subsystems. These methods are illustrated by a more detailed description of the Task Scheduling and Error Handling subsystems.

# Presentation Overview

- INTRODUCTION
- SYSTEM FUNCTIONS
  - Communication
  - Task Scheduling
  - Task Reconfiguration
  - Clock Synchronization
  - Data Handling and Voting
  - Error Handling and Recovery
- SUMMARY

# Design Objectives

- RELIABILITY – $1.0 \times 10^{-9}$ over 10 hours.

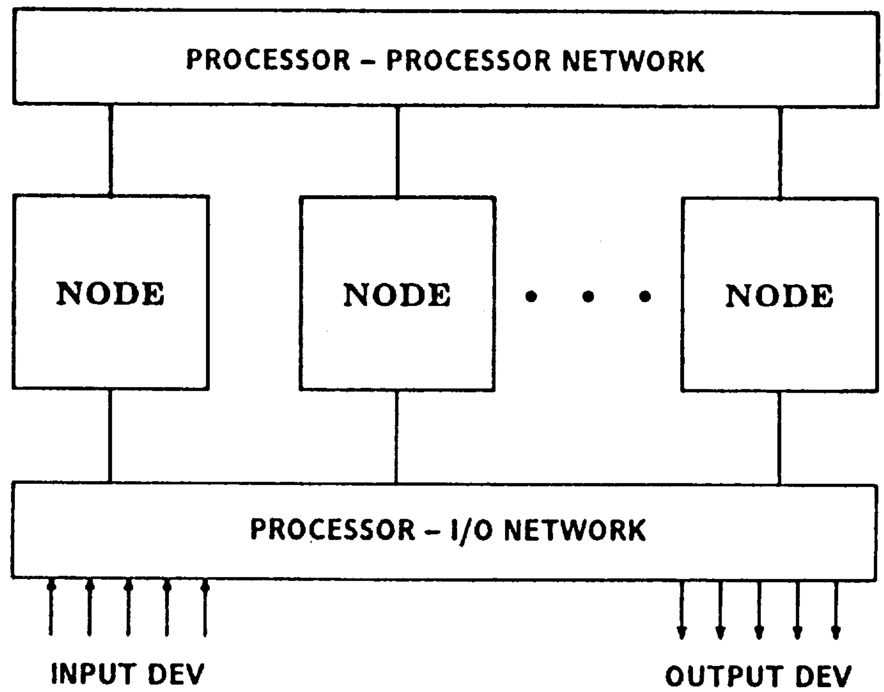
- PERFORMANCE

| | | |
|---|---|---|
| 200 | Hz. | – Max Task Iteration Rate |
| 5.5 | MIPS | – Max Computational Capacity |
| 1.0 | MBPS | – Max I/O Transfer Rate |
| 5.0 | ms. | – Min Transport Lag (Input $\rightarrow$ Output) |

- REUSABLE
  - Functional Partitioning
    - Application Specific Functions
    - Standard Executive Functions

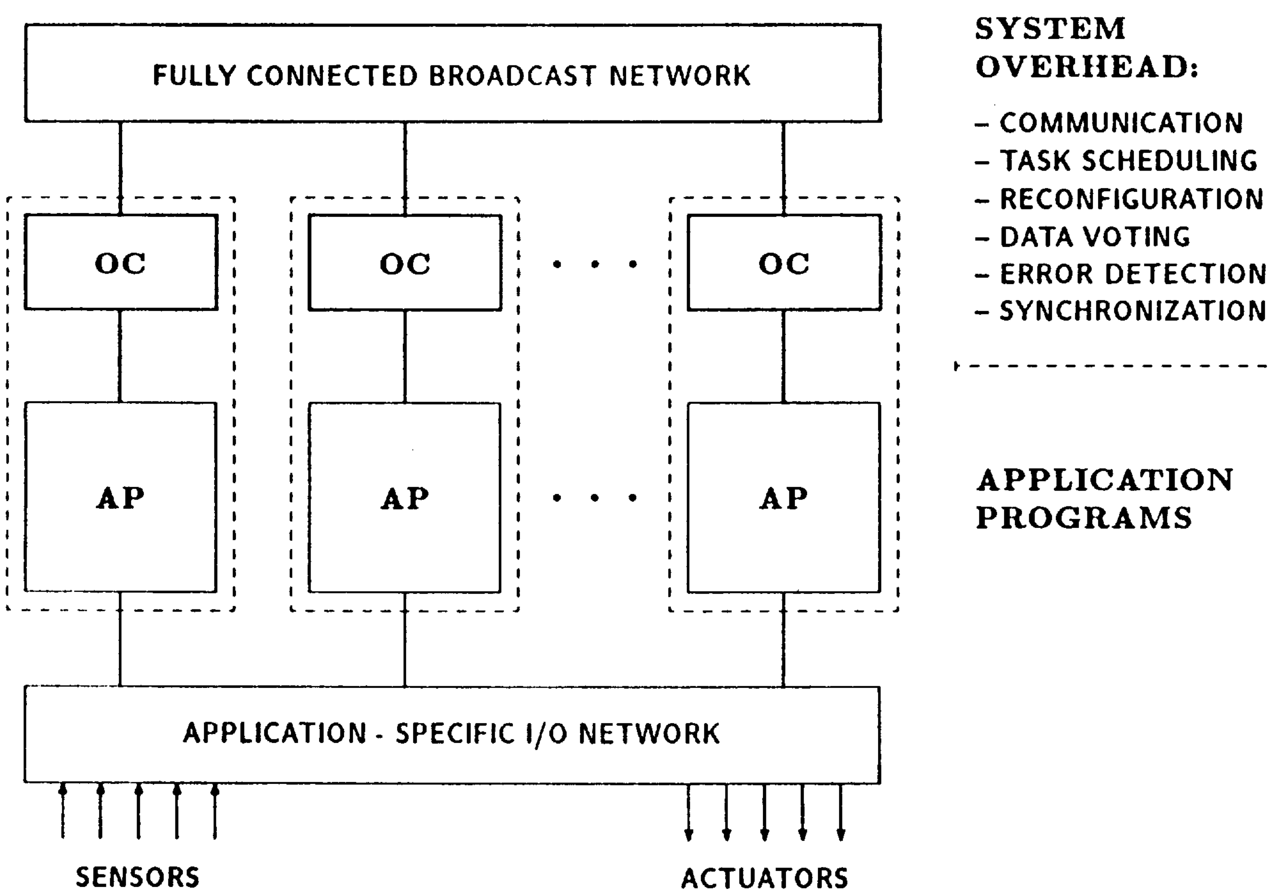
- LOW EXECUTIVE OVERHEAD
  - Physical Partitioning
    - Separate Executive Processor
    - Hardware Intensive

# Loosely-Coupled Multiprocessor




- Node ⇒ Processor and Private Memory
- No Shared Memory
- Message-Based Inter-Node Communication
- Common Operating System

# MAFT System Architecture



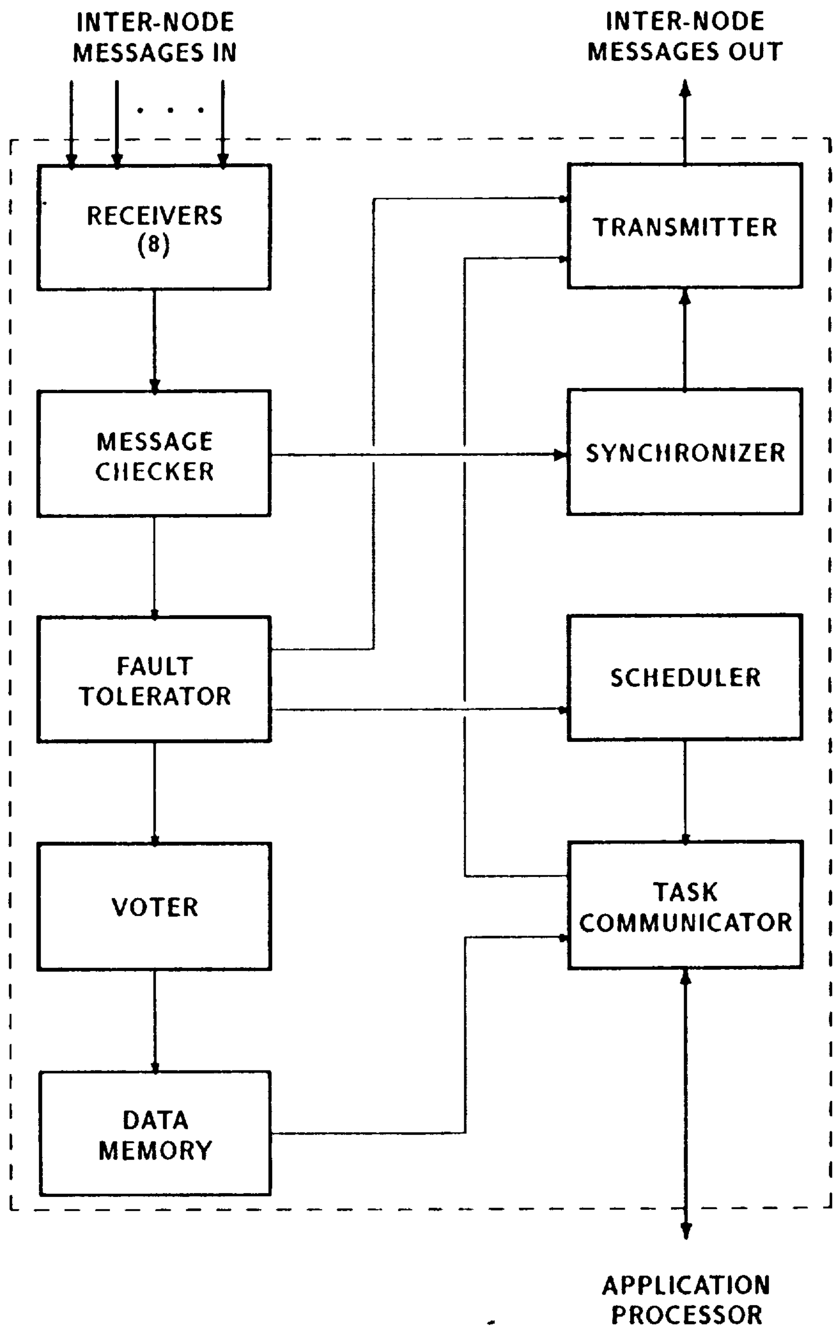

- **OC $\Rightarrow$ Operations Controller:**

  Special Purpose Device Common to All MAFT Systems.

- **AP $\Rightarrow$ Application Processor:**

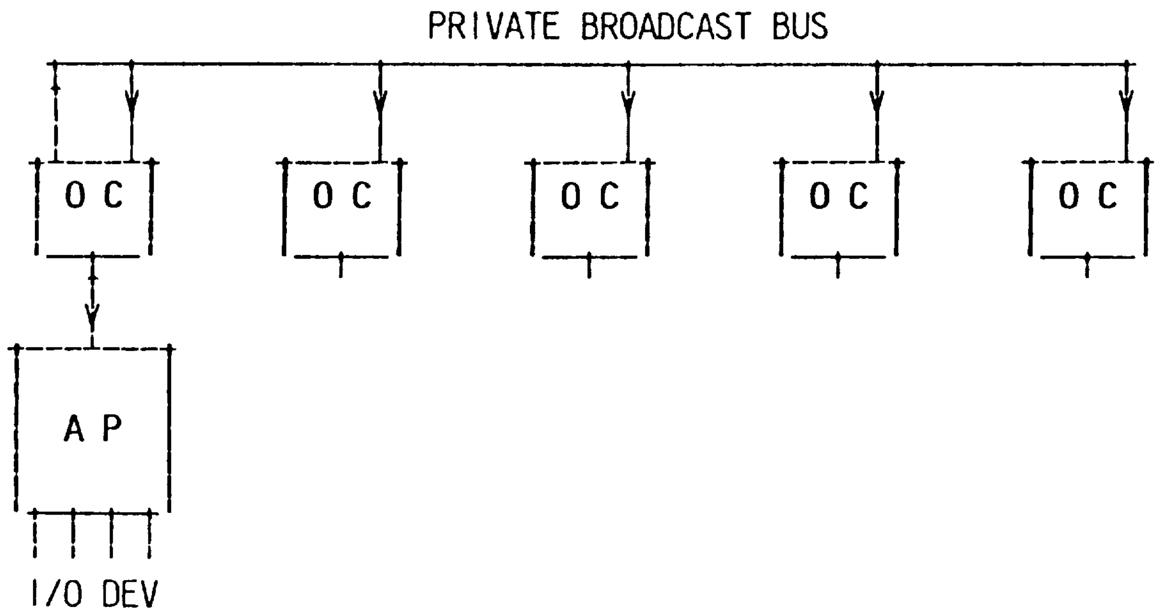
  General Purpose Application-Specific Processor.

# Operations Controller Block Diagram



# COMMUNICATION

# INTER-PROCESSOR COMMUNICATIONS




- INTRA-NETWORK COMMUNICATION
  - MESSAGES TRANSMITTED ON PRIVATE SERIAL BROADCAST BUSSES
  - ALL NODES RECEIVE, CHECK AND PROCESS ALL MESSAGES
  - MESSAGE TYPES
    - DATA (8/16/32B INT OR BOOL, IEEE STD 32B FLOAT)
    - TASK COMPLETED / STARTED / BRANCH
    - SYNCHRONIZATION / BRANCH INTERACTIVE CONSISTENCY
    - ERROR REPORT

- OC / AP COMMUNICATION
  - 16 BIT ASYNCHRONOUS P.I.O. INTERFACE
  - LOOKS LIKE "JUST ANOTHER I/O PORT" TO AP
  - COMPATIBLE W/ EXISTING UNIPROCESSOR OPER SYST

# Message Handling

- TRANSMITTER
  - Format Msg – NID, Msg Type, Framing, ECC
  - Broadcast Msg

- RECEIVERS – 1 per incoming link
  - Accept Properly Framed Bytes
  - Buffer Byte for Message Checker

- MESSAGE CHECKER
  - Poll Receivers – 6.4 $\mu s$ cycle
  - Physical and Logical Checks
  - Steer Good Messages to Other Subsystems
  - Dump Bad Messages into "Bit-Bucket"

# LOCAL AP/OC INTERFACE OPERATIONS

1. **TASK SWITCHING PROCESS**

   - AP: DONE WITH LAST TASK, WHAT IS THE TASK IDENTIFICATION (TID) NUMBER OF THE NEXT TASK.

   - OC: HERE IT IS

2. **TRANSFER DATA FROM OC TO AP**

   - AP: GIVE ME THE NEXT INPUT DATA VALUE

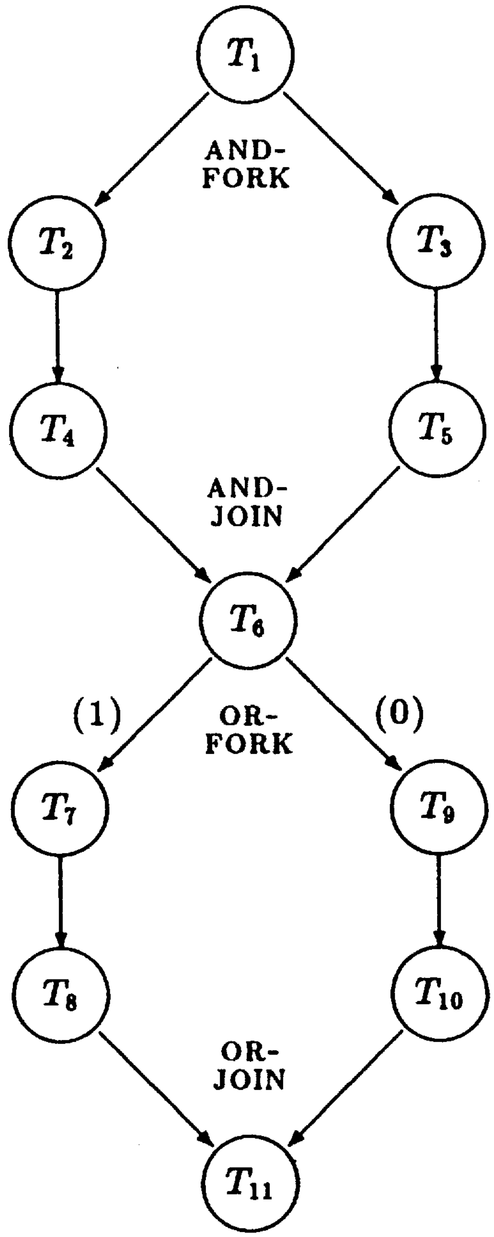
   - OC: HERE IT IS

3. **TRANSFER DATA FROM AP TO OC**

   - AP: HERE'S THE NEXT OUTPUT DATA VALUE

   - OC: I GOT IT

# Typical Task System



# PERFORMANCE ISSUES

- STRICTLY PERIODIC SCHEDULER
  - Fast – Freq Well Above Spec – 500 Hz. vs. 200 Hz.
  - Simple – Binary Freq Dist ($f_i = 2^{-i} f_0$)
  - Flexible – Conditional Branching
  - Efficient – Don't Keep AP Waiting
- NON-PREEMPTIVE
  - Scheduler Complexity
  - Context Switching Time – Unknown Funct of AP
  - High Frequencies – Short Tasks
- NO OC INTERRUPTS – I/O
  - Scheduler Complexity
  - Predictability
  - High Frequencies – Polling
  - DMA or IOP access to AP Memory

# O.C. View of a Task

- INTERNAL FUNCTION IS BLACK BOX

- VISIBLE PROPERTIES OF A TASK
  - Priority (static, unique)
  - Iteration Period
  - Precedence Constraints
  - Min and Max duration Limits
  - Fixed Input and Output Shared Data Sets
  - Branch Condition (asserted at completion)

# FAULT-TOLERANCE ISSUES – I

- VARIABLE MODULAR REDUNDANCY
  - Specify Redundancy of Each Individual Task
  - Redundancy Matches Criticality
  - No More Copies Than Necessary

- GLOBAL VERIFICATION
  - Consensus Defines Correctness
  - All Functions Observable and Predictable
  - Replicated Global Scheduler
  - Completed/Started (CS) Message:
    - Node I.D.
    - Started Task I.D.
    - Branch Condition

# Message Passing Robustness

- Delivery NOT GUARANTEED

- Single Msg Error Detect. NOT GUARANTEED
  - ECC coverage $\geq (1 - 1 \times 10^{-6})$ per msg

- Repeated Undet. Errors PROBABILISTICALLY PRECLUDED

# TASK SCHEDULING

# FAULT-TOLERANCE ISSUES – II

- DISSIMILARITY BETWEEN COPIES
  - Dissimilar Software and Hardware
    - Guards Against Generic Faults
    - No Guarantee – Knight, Levenson, St. Jean
    - Best Chance of Detecting Error
    - Only Chance of Masking Error
  - Implications
    - Different Numerical Results
    - Different Execution Times
  - Impact on Scheduler
    - Min and Max Execution Time Limits
    - Vote on Branch Conditions in CS Messages

# FAULT-TOLERANCE ISSUES – III

- BYZANTINE AGREEMENT
  - Definition
    - Agreement on All Messages
    - Validity of Agreement
  - Necessity in MAFT
    - Consensus Defines Correctness
    - Must Have Single Consensus
  - Preconditions for Disagreement
    - Initial Disagreement – Enhanced by Dissimilarity
    - Assymetric Communication – Minimized by Busses
  - Solution – Interactive Consistency (Pease et al.)
    - Global Receipt of All Messages
    - Periodic Synchronized Re-Broadcast Rounds
    - Vote on Received Re-Broadcasts
    - Use Voted Values For All Scheduling Decisions

# IMPACT OF FAULT-TOLERANCE

- ALL COPIES DONE BEFORE SUCCESSORS RELEASED
- MAX EXECUTION TIMERS – ASSURE PROGRESS
- CONFIRMATION DELAY – MEAN 2.5 SUB.
  - Only Affects Successors
  - Efficiency Requires Parallel Paths
- FAULT-TOLERANCE LEVELS
  - Single Asymmetric (Byzantine) Fault
  - Double Symmetric Fault
  - Reliability Modelling – $10^{-10}/hr$ with 5 Nodes

# MAFT Timing Hierarchy

| PERIOD | SPEC | DEFINITION | BOUNDARY |
|---|---|---|---|
| SUB-ATOMIC | Min $400\mu s$ | I.C. Rebroadcast Period<br><br>Min Guaranteed Task Duration | Task Inter. Cons. (TIC) Message |
| ATOMIC | Min 2–2.8 $ms$ | Highest Freq. Task<br><br>Clock Sync. Period | System State (SS) Message |
| GENERAL ITERATION | $2^i$ Atom. Per. | Intermed. Freq.Tasks | System State (SS) Message |
| MASTER | Max $1K$ Atom. Per. | Lowest Freq. Task | System State (SS) Message |

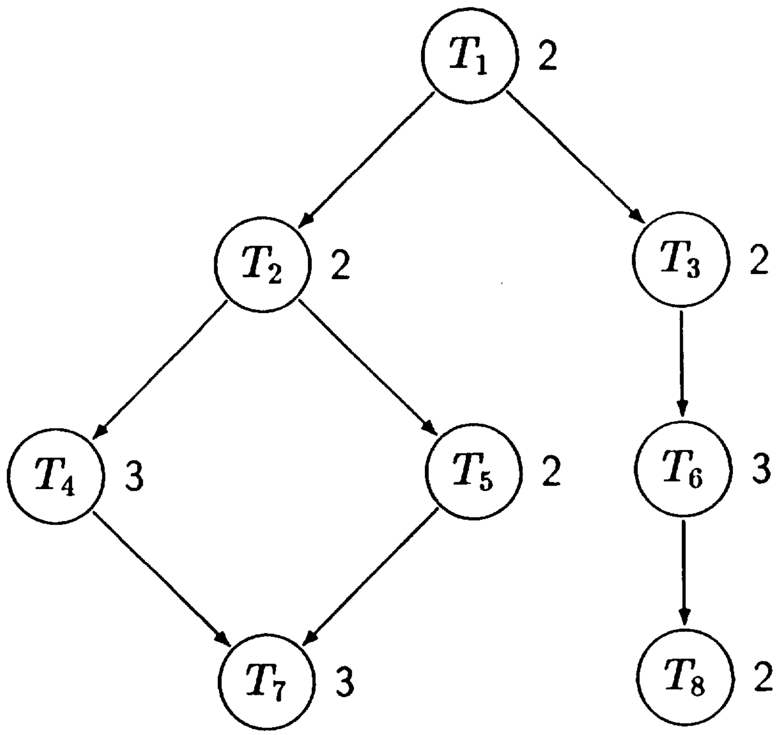
# Scheduling Stability Problem

- SCHEDULING INSTABILITY – Anomalous or unpredictable variations in total execution time (Makespan) due to variations in system parameters.

- MULTIPROCESSOR ANOMALIES – Observation that Makespan can be *increased* by:
  - Increasing Number of Processors,
  - Relaxing Precedence Constraints,
  - Decreasing Individual Task Durations.

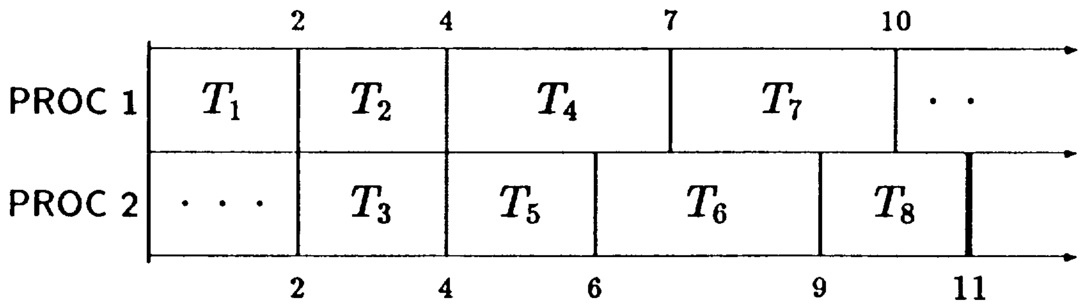
- DYNAMIC FAILURE – Condition where all tasks execute properly *except* that deadlines are missed.
  - Can occur in a fault-free system,
  - Can be induced by instability.

# Sample Task System

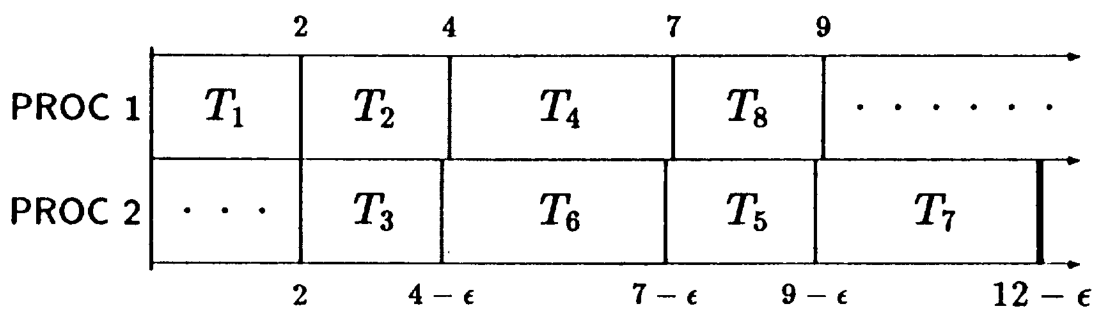


# Instability of Sample Task System

- <u>STANDARD GANTT CHART</u> (max task durations)




- <u>NON-STANDARD GANTT CHART</u> (shorten $T_3$ by $\epsilon$)




- <u>WHAT HAPPENED?</u>
  - $T_3$ finished before $T_2$,
  - $T_6$ "ready" before $T_5$,
  - $T_5$ displaced by $T_6$ $\Rightarrow$ Priority Inversion,
  - Critical path ($T_2 \rightarrow T_7$) impeded.

## Previous Work

- GRAHAM (1969) – Bound Magnitude of Instability

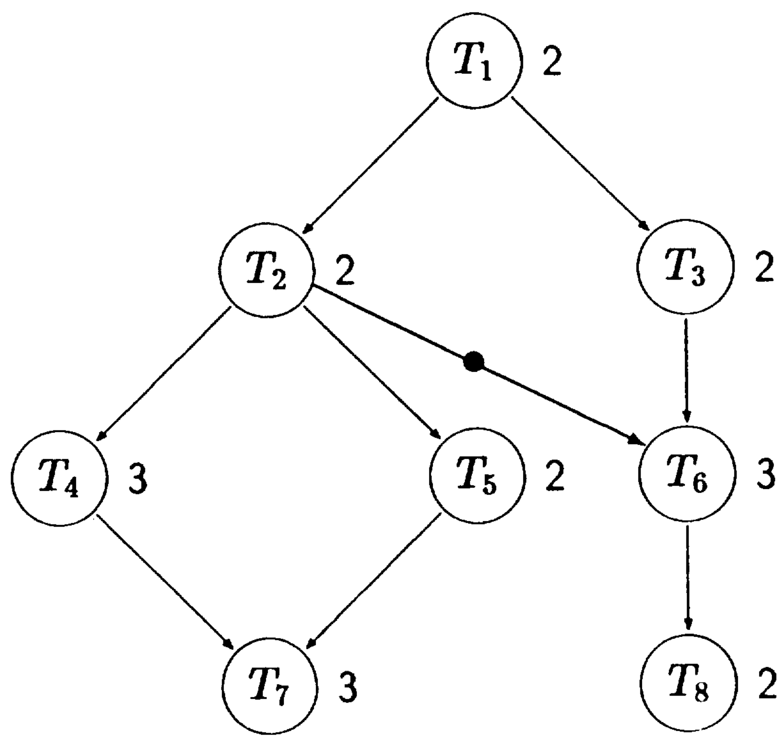
$$\frac{\omega'}{\omega} = 2 - \frac{1}{N}$$

  - $\omega$ = Makespan of Standard Gantt Chart,
  - $\omega'$ = Makespan of worst-case schedule,
  - $N$ = Number of Processors.

- MANACHER (1967) – Stabilization Algorithm
  - Necessary Pre-conditions
    i. $\exists$ "fork" in Precedence Graph,
    ii. Successors of forking task run in parallel on Standard Gantt Chart,
    iii. Possible priority inversion around fork.
  - Solution – Impose Artificial Dependency around fork.

# Stabilized Task System

- MANACHER ARTIFICIAL DEPENDENCY ($T_2 \rightarrow T_6$)




- EFFECT
  - $T_2$ is common parent for both $T_5$ and $T_6$,
  - $T_6$ will be "ready" no earlier than $T_5$,
  - $T_5$ precedes $T_6$ in priority list,
  - $T_6$ can not be selected before $T_5$.

# Limitations of Manacher's Solution

- Sufficient, but not always necessary
- Adds Scheduling Overhead (resolve edge)
- Unrealistic System Model
  - Assumes no scheduler overhead,
  - Assumes dynamic allocation,
  - Allows for no Confirmation Delay,
  - Ignores minimum duration bounds,
  - Does not predict magnitude of instability.

## Current Research

- Find Necessary *and Sufficient* Stability Conditions.

- Develop Stabilization Strategies
  - Task System Stabilization
    - Edge Stabilization (Manacher)
    - Vertex Stabilization
    - Hybrid Stabilization
  - Run-Time Scheduler Stabilization
    - Limited Scan Depth
  - Scheduling Algorithm Stabilization
    - Sched. Algorithm Assigns Priorities
    - Constrain to Preclude Necessary Conditions

- Extend System Environment
  - Scheduler Overhead
  - Static Allocation
  - Confirmation Delay
  - Minimum Duration Bounds

# SYNCHRONIZATION

# MAFT Synchronization

- Periodically Exchange System State (SS) Msgs
  - SS Msg $\Rightarrow$ "Atomic Period" Boundary
  - Synchronization Period = 2 Atomic Periods

- Loosely Synchronized Individual Clocks
  - Msg Exchange $\Rightarrow$ No Separate Clock Lines
  - Physical Separation $\Rightarrow$ Damage Tolerance
  - Robustness to "Common Upset" events

- Synchronization Modes

- Steady State – Maintain Existing Synchronization
- Warm Start – Converge to Existing Operating Set
- Cold Start – Form Initial Operating Set
  - · Interactive Convergence to synchronize
  - · Interactive Consistency $\Rightarrow$ Steady State
  - · Origin of Two-phase algorithm



PRECEDING PAGE BLANK NOT FILMED

# DATA HANDLING AND VOTING

# Typical Sync. Values

- $\epsilon = 7\ \mu sec$ – 600 ft. separation
- $\rho = 5 \cdot 10^{-5}$
- $R = 20\ msec \Rightarrow 10\ msec$ Atomic Pd. $\Rightarrow$ 100 Hz.
- $\rho R = 1\ \mu sec$
- No Faults: Max $\delta = 8.5\mu\ sec$
- With Faults: Max $\delta = 16.5\mu\ sec$

# Data Management

- DATA GENERATED BY AP
- BROADCAST IN DATA MESSAGE
- RECEIVED AND PROCESSED BY ALL NDOES
  - Static Limit Check
  - On-The-Fly Vote
  - Dynamic Deviance Check



PRECEDING PAGE BLANK NOT FILMED

# On-The-Fly Voting I

- TRIGGERED BY DATA MESSAGE ARRIVAL

- DATA ID ACTS AS UNIQUE VARIABLE NAME

- USE ALL PREVIOUS COPIES OF SAME DATA ID
  - MS or MME (programmer selectable)
    - Sort Serially – High-Order-Bit First
    - Select 2 "Medial" Values
    - Average (Add and Shift)
  - No I.C. Vote for Boolean Types
    - Difficult to implelement round 2
    - Usually Control Data for Mode Switch
    - $\exists$ Better Way for Mode Switch

# On-The-Fly Voting II

- DEVIANCE CHECK
  - Compare Each Copy to Voted Value
  - Excessive Difference $\Rightarrow$ error
  - Programmer Sets Limits
  - Generate Error Vector $\Rightarrow$ Source Nodes

- TERMINATE
  - Scheduler Says All Copies Done
  - Send Error Vector to Fault-Tolerator
  - Send Voted Value to Data Memory
  - Swap On-line/Off-line Buffers in Data Memory
  - Clear Previously Received Copies from Voter

# ERROR HANDLING AND RECOCVERY

# Fault Classifications

- BYZANTINE (MALICIOUS)

Pease et al. (1982)

  - $N \geq 3t + 1$
  - $r \geq t$

- MALICIOUS $\cup$ BENIGN (self-evident)

Meyer and Pradhan (1987)

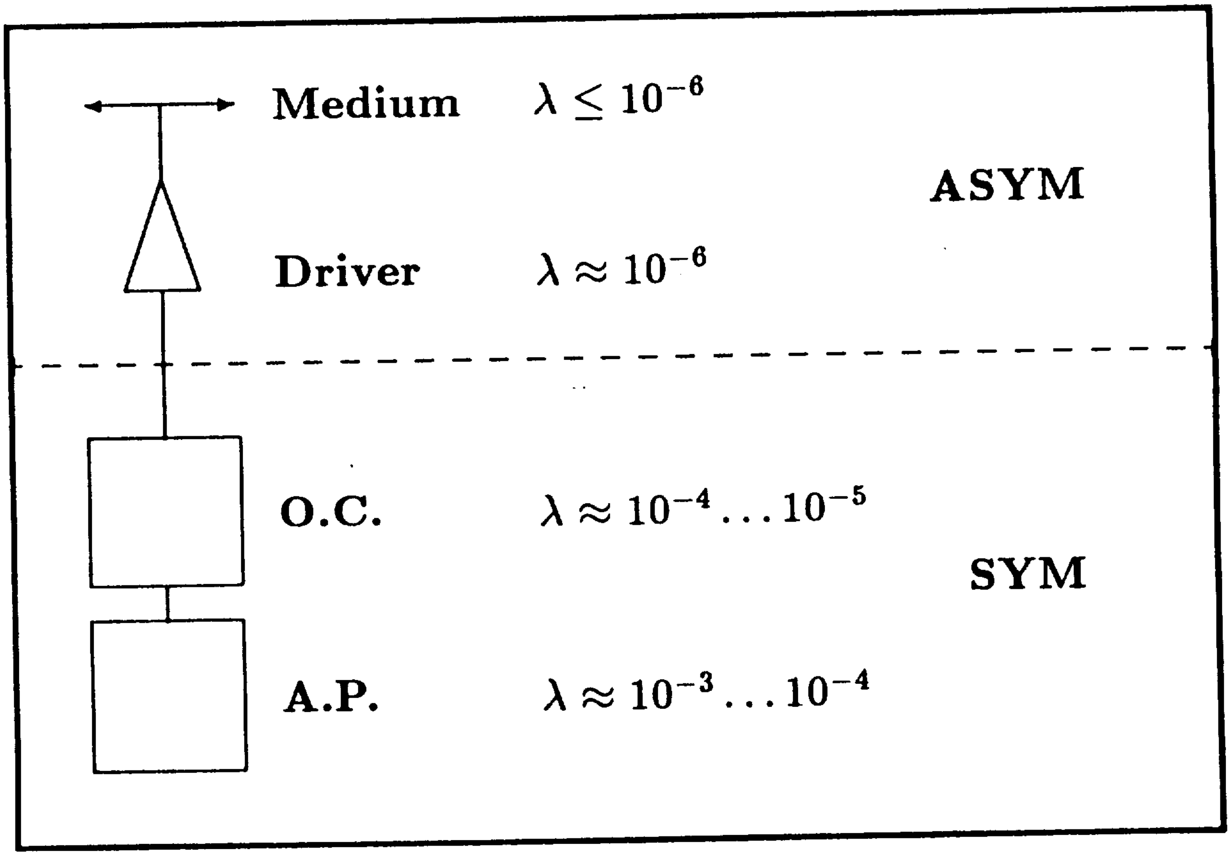
  - $t = m + b$
  - $N \geq 3m + b + 1$
  - $r \geq m$

- (ASYMMETRIC $\cup$ SYMMETRIC) $\cup$ BENIGN

Thambidurai and Park (1989)

  - $t = a + s + b$
  - $N \geq 3a + 2s + b + r + 1$
  - $r \geq a$

# Fault Classes by Source




- Can Estimate Separate $\lambda$'s
  - $\lambda_{asym} \approx 10^{-6}$
  - $\lambda_{sym} \approx 10^{-3} \ldots 10^{-4}$
- Generic Fault = Multiple Symmetric
  - $\lambda_{gen} \approx 10^{-5}$ ?

## Error Detection

- Errors Are Manifested In Messages
  - Physical: ECC, framing, length
  - Contents: values
  - Timing or sequencing
  - Existence or non-existence

- Log Errors Over One Atomic Period
  - Errors reported by all subsystems
  - Fault-Tolerator records errors
  - $\exists$ 31 separate error "flags"
  - $\exists$ Unique "Penalty Weight" $PW$ for each flag
  - $\exists$ "Incremental Penalty Count" $IPC$ for each node
  - FOR each flag $f$ reported against node $i$:
    - $IPC(i) := IPC(i) + PW(f)$

## Error Reporting

- Broadcast ERR($i$) Message
  - At beginning of next Atomic Period
  - Contents:
    - $IPC(i)$
    - $BPC(i)$ – Base (current) penalty count
    - All Error Flags for node $i$

- No ERR Message $\Rightarrow$ No Detections

# BPC Manipulation

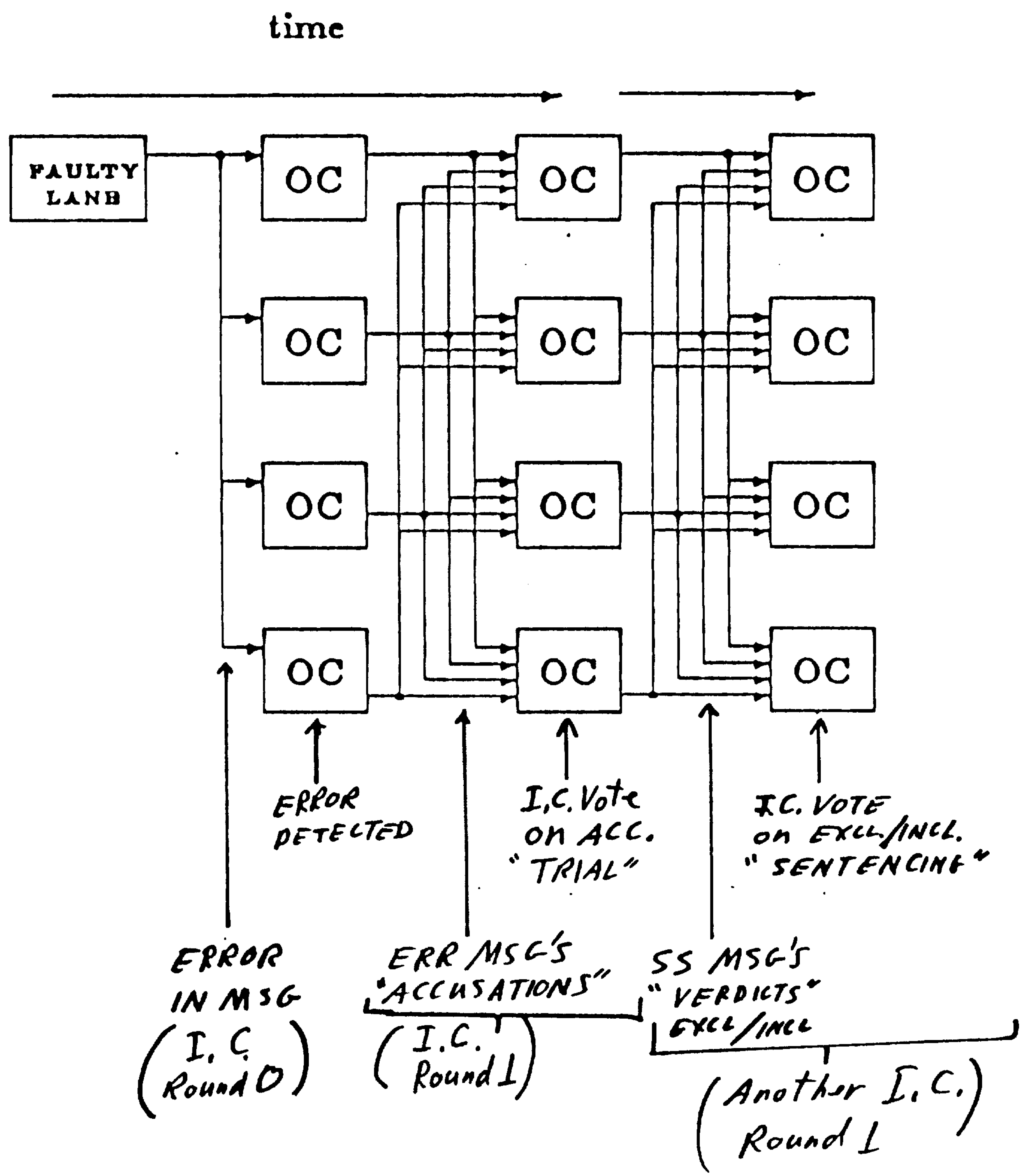
- BPC $\Rightarrow$ Health Of Node

- Increasing BPC – ERR Message Vote
  - Vote on $BPC(i)$
  - Vote on $IPC(i)$
  - $BPC(i) := BPC(i) + IPC(i)$

- Decreasing BPC – Fixed decrement
  - $\exists$ Penalty Decrement value $PD$
  - At New Master Period
  - $BPC(*) := BPC(*) - PD$
  - Allows For Eventual Readmission

# Exclusion/Readmission

- Recommend Exclusion/Readmission
  - $\exists$ Exclusion Threshold $T_{excl}$
  - $\exists$ Admission Threshold $T_{adm}$
  - Recommend in next SS message:
    - $BPC(i) \geq T_{excl} \Rightarrow$ Exclude $i$
    - $BPC(i) \leq T_{adm} \Rightarrow$ Readmit $i$
    - $T_{adm} < BPC(i) < T_{excl} \Rightarrow$ No Change
- I.C. Vote on Recommendations
  - Consistent System State is Critical
  - Free (needed for cold-start)
  - Highly Degraded Systems
  - Common Mode Upset Recovery




ERROR HANDLING (SIMPLEX I.C.)

## Sed Quis Custodiet ...III

- AP – Diagnostics in Workload

- OC – System Level Self-Test
  - Errors Very Rare
  - Inject Faults to Excercise Error Detection
    - Special self-test Task ID
    - Suspend normal Transmitter Ops
    - Tranmsit string from self-test ROM
    - Can transmit ANY test scenario
  - Test Results Based On
    - False/Missed Accusations
    - Cyclic Link Check
  - Independent of Actual Bit-Stream
  - Rotate "Originator" Duty
  - Complete Coverage If ANY One Node Correct

# Version Management

- SSV = System State Vec – eg (2,1,1)
- VMV = Version Management Vec – eg (1,1,1)
- WMV = Workload Management Vec – (SSV) or (VMV)

- Vectors Used By Different Subsystems

| | | |
|---|---|---|
| Data Voter | VMV | Inactive Copy Ignored For Vote |
| Dev Checker | SSV | Inactive Copy Still Monitored |
| Scheduler | WMV | Inactive Copy May Not Run |

- WMV = SSV
  - Inactive Copy Still Executing
  - Actual Tasks Being Monitored
  - Best for Generic Fault Detection

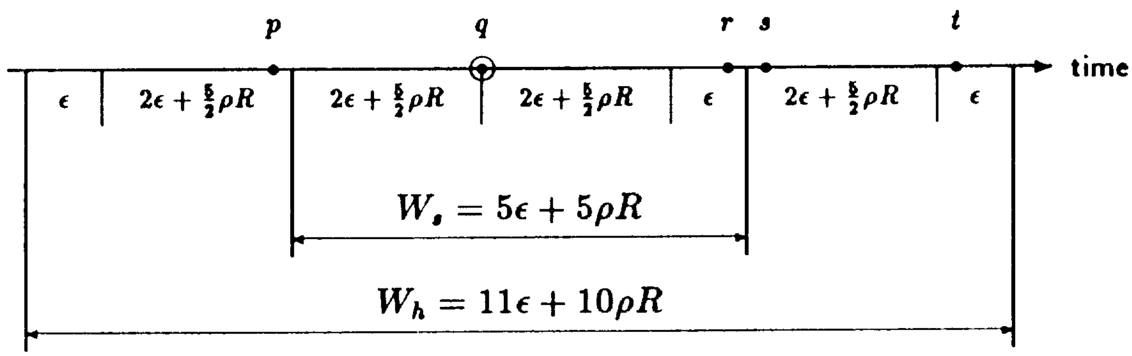
- WMV = VMV
  - Inactive Copy Doing Something Else
  - Will Not Be Affected By Generic
  - Can Activate To Replace Sibling
  - Best For Generic Recovery

# Synchronizer Error Detection

- MAFT error detection is by consensus
  - Each node reports errors on all nodes.
  - Majority vote confirms or denies accusations.
  - Disagreement with majority may itself be an error.

- Faulty node must be detected by majority of nodes
  - Must be "far enough" out of sync
  - There exists a region of ambiguity
  - Defines size of "Sync Window"

# Synchronizer Error Windows




- $W_s$ = SOFT ERROR WINDOW
  - Spans Range of Receipts from Non-Faulty Nodes
  - Error May Not Be Confirmed
  - Inherent Ambiguity
  - Must Suspend Error Disagreement Penalties

- $W_h$ = HARD ERROR WINDOW
  - IF Any non-faulty node detects a Hard-Error
    THEN All non-faulty nodes detect an Error
  - Can demand Corroboration

# Typical Sync. Window Values

- $\epsilon = 7\ \mu sec$ – 600 ft. separation
- $\rho = 5 \cdot 10^{-5}$
- $R = 20\ msec \Rightarrow 10\ msec$ Atomic Pd. $\Rightarrow$ 100 Hz.
- $\rho R = 1\ \mu sec$
- No Faults: Max $\delta = 8.5\mu\ sec$
- With Faults: Max $\delta = 16.5\mu\ sec$
- $W_s = 40\mu\ sec$
- $W_h = 87\mu\ sec$

# SUMMARY

SUMMARY COMMENTS ON THE APPLICATION OF MAFT TECHNOLOGY

1. CAPABILITIES
   - BASIS OF A GENERIC REAL-TIME MULTICOMPUTER SYSTEM
   - REMOVES F.T. OVERHEAD FROM APPLICATION PROCESSOR
   - HANDLES ALL REDUNDANCY MANAGEMENT WITHIN COMPUTER
   - ASSISTS IN REDUNDANCY MANAGEMENT OF I/O SYSTEM

2. FLEXIBILITY
   - INDEPENDENT OF I/O ARCHITECTURE
   - HIGHLY RECONFIGURABLE AND GRACEFULLY DEGRADABLE
   - PROVIDES MECHANISMS, NOT POLICIES

3. USABILITY

# ADVANTAGES OF APPROACH

- PARTITIONED APPROACH SIGNIFICANTLY REDUCES PROCESSOR OVERHEAD
- DATA DRIVEN ARCHITECTURE MUCH FASTER THAN SOFTWARE IMPLEMENTATION
- NOT DEPENDENT UPON ARCHITECTURE OF APPLICATION PROCESSOR
- REDUNDANCY IS "TASK-BASED" AND FLEXIBLE
- SUITABLE FOR HIGH RELIABILITY AND HIGH PERFORMANCE APPLICATIONS

# MAFT Bibliography

## References


[Dar88] Darwiche, A.A., and F.M. Doerenberg, "Application of the Bendix/King Multicomputer Architecture for Fault-Tolerance in a Digital Fly-By-Wire Control System", *Midcon*, Aug 1988.

[Glu86] Gluch, D.P., and M.J. Paul, "Fault-Tolerance in Distributed Digital Fly-by-Wire Flight Control Systems", *AIAA/IEEE Seventh Digital Avionics Systems Conference*, Oct 1986.

[Kie87] Kieckhafer, R.M., "Task Reconfiguration in a Distributed Real-Time System," *Eighth IEEE Real-Time Systems Symposium*, Dec 1987.

[Kie88] Kieckhafer, R.M., et al, "The MAFT Architecture for Distributed Fault-Tolerance", *IEEE Trans. Computers*, V. C-37, No. 4, pp. 398-405, Apr 1988.

[Kie89] Kieckhafer, R.M., "Fault-Tolerant Real-Time Task Scheduling in the MAFT Distributed System," *Proc, Twenty-Second Hawaii International Conference on System Sciences*, Jan 1989.

[McE88] McElvany, M.C., "Guaranteeing Deadlines in MAFT," *Proc. IEEE Real-Time Systems Symp.*, pp. 130-139, Dec 1988.

[Tha88] Thambidurai, P.M., and Y.K. Park, "Interactive Consistency with Multiple Failure Modes", *Proc. Seventh Reliable Dist Systems Symp.*, Oct 1988.

[Tha88a] Thambidurai, P.M. *Critical Issues in the Design of Distributed, Fault-Tolerant, Hard Real-Time Systems*, Ph.D. Dissertation, Dept. of Electrical Engineering, Duke University, 1988.

[Tha89] Thambidurai, P.M., et al., "Clock Synchronization in MAFT", *Nineteenth Fault-Tolerant Computing Symposium*, pp. 142-151, Jun 1989.

[Wal88] C.J. Walter, "MAFT: An Architecture for Reliable Fly-by-Wire Flight Control," *Proc. AIAA/IEEE Eighth Digital Avionics Systems Conference*, pp. 415-421, Oct 1988.