The practice of reliable system design

Critical Computations

Introduction

- Real-time control represents the most challenging application for fault-tolerant computer designers since faulty computations can cause the loss of human life or expensive equipment
  - Flight control of commercial airlines, military fighters, and space shuttles.
  - Safety monitors in nuclear power plants, high-speed trains, and hospitals.
  - Continuous process control in chemical plants, oil well drilling/pumping, etc.
- ..... Automotive, Health,
- Since real-time controllers are often embedded into these larger systems, the fault-tolerant computers most often go unnoticed.

Requirements

no computation errors and strict respect of deadlines.

Deadlines ~ msec so, error detection and recovery time must be minimized.

Specially designed hardware operates with concurrent error detection so that incorrect data never leaves a faulty module.

Main solution

massive replication (hw and sw) and voting.

Hardware replication and voting can be applied at any level in the digital hierarchy, most often at the processor level.

Software approaches use replication in time or space. Repeated execution is usually impractical in situations with strict deadlines. Hence, software is typically replicated on different computers, with software voting on messages carrying the computed results between processors.

Key issue: synchronization.

Copies must be kept in lock-step, and the voting must be performed in a timely manner. If the copies become inconsistent, substantial effort may be required to return them to identical behavior.

SIFT

Software Implemented Fault Tolerance

J.H. Wensley, L. Lamport, J. Goldberg, M.W. Green, K.N. Levitt, P.M. Melliar-Smith, R.E. Shostak, C.B. Weinstock, "**SIFT: The Design and Analysis of a Fault-Tolerant Computer for Aircraft Control**", Proceedings of the IEEE, vol.66, no.10, 1978, pp.1240-1255.

Introduction

The loss of computer control for even a few milliseconds could lead to disaster. Thus, these experimental systems are designed for a failure probability of 10^{-9} during a 10-hour mission.

How does one verify that a system meets its design spec.?

10^{-10} failures per hour = 1.14 million operating years before failure (testing is impossible).

SIFT approach:

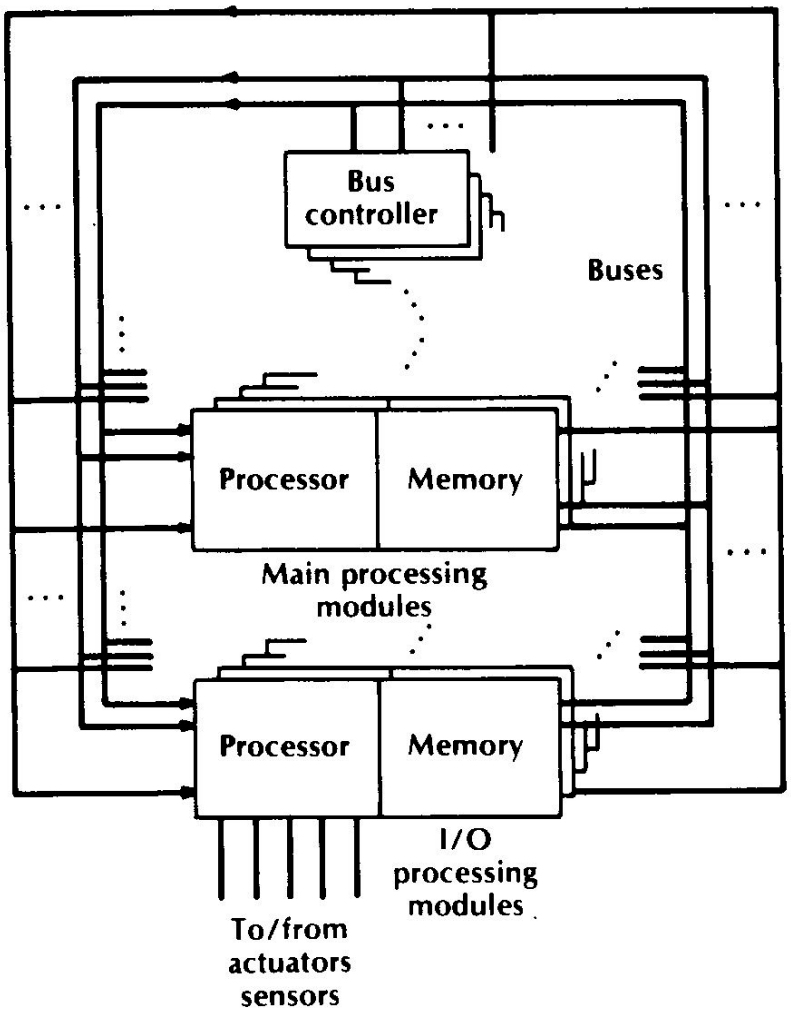
to mathematically prove the correctness of the system sw.

SW-implemented FT relies primarily on SW mechanisms to achieve reliability.

The hardware consists of independent computers communicating with other computers over unidirectional serial links (N computers: N(N-1) links).

SIFT

- Software-Implemented Fault Tolerance designed by SRI International, and intended for real-time control of aircraft.
- Fault tolerance is accomplished as much as possible by programs rather than by hardware.
- Fault tolerance includes error detection and correction, diagnosis, reconfiguration, and the prevention of a faulty unit from having an adverse effect on the whole system.



Structure of the SIFT system

SIFT-model

- The SIFT system executes a set of tasks (sequence of iterations).
- The input data to each iteration of a task are the output data produced by the previous iteration of some collection of tasks.
- Reliability is achieved by having each iteration of a task independently executed by a number of modules.
- After executing the iteration, a processor places the iteration's output in the memory associated with the processor.

SIFT model 2

- A processor that uses the output of this iteration determines its value by examining the output generated by each processor that executed the iteration (value is chosen by a two-out-of-three vote).
  - If all copies of the output are not identical, then an error has occurred. Such errors are recorded in the processor's memory, and these records are used by the executive system to determine which units are faulty.
  - Voting is performed only on the input data to tasks rather than on every partial result. Thus, the tasks need to be only loosely synchronized (for example, to within 50 microseconds).

Loose synch

- Benefit of loose synchronization: a task's iteration can be scheduled at slightly different times by different processors.
- The probability of correlated failures caused by simultaneous transient failures of several processors is highly reduced.

Variable redundancy

- The number of processors executing a task can vary with the task and can be different for the same task at different times (according to its criticality)
- The allocation of tasks to modules is, in general, different for each module. It is determined dynamically by a task (*global executive*), which diagnoses errors to determine which modules and buses are faulty.
- When the global executive decides that a module has become faulty, it reconfigures the system by appropriately changing the allocation of tasks to modules.

Fault Isolation

- Preventing a faulty unit from causing incorrect behavior in a non-faulty unit.
- Fault isolation is a more general concept than damage isolation (preventing physical damage from spreading beyond carefully prescribed boundaries).
- Fault isolation in SIFT requires not only isolating damage, but also preventing a faulty unit from causing incorrect behavior either by corrupting the data of the non-faulty unit or by providing invalid control signals (those that request service, grant service, effect timing synchronization between units, etc.)

Corruption - 1

- Protection against the corruption of data is provided by the way in which units can communicate.
- A processing module can read data from any processing module's memory, but it can write only into its own memory.
- Thus a faulty processor can corrupt the data only in its own memory and not in that of any other processing modules.
- All faults within a module are treated as if they have the same effect: namely, that they produce bad data in that module's memory.
- The system does not attempt to distinguish the nature of a module fault. (e.g. a faulty memory from a processor that puts bad data into an otherwise non-faulty memory).

Corruption - 2

- A faulty processor can obtain bad data if it reads them from a faulty processing module or over a faulty bus. To avoid the generation of incorrect results, fault masking techniques are used.
- Invalid control signals could produce incorrect behavior in a non-faulty unit:
  - the unit carries out the wrong action (possibly by doing nothing),
  - the unit does not provide service to other units.
- In SIFT this is avoided by making each unit autonomous, with its own control.
- Improper control signals are ignored, and time-outs are used to prevent the unit from hanging up, waiting for a signal that never arrives.

Fault Masking

- To avoid malicious effects of corrupted data the processor receives multiple copies of them.
- Each copy is obtained from a different memory, over a different bus, and the processor uses majority voting to obtain a correct version of the data.
- The most common case: a processor obtains three copies of the data, providing protection from a single faulty unit.

Fault Masking - Reconfiguration

- After identifying the faulty unit, the system is reconfigured to prevent that unit from having any further effect.
  - If the faulty unit is a processing module, then the tasks that were assigned to it are reassigned to other modules.
  - If it is a bus, then processors request their data over other buses.
  - After reconfiguration, the system is able to withstand a new failure, assuming that there are enough non-faulty units remaining.
- SIFT flexibility is very high: the number of processors executing a task can vary with the task and can be changed dynamically (this is very useful in the aircraft control).

Scheduling

- The aircraft control function places 2 types of timing requirements on the SIFT system:
  - Output to the actuators must be generated with specified frequency.
  - Transport delay (the delay between the reading of sensors and the generation of output to the actuators based upon those readings) must be kept below specified limits.

Scheduling

- To fulfill these requirements, an iteration rate is specified for each task:
  - The scheduling strategy must guarantee that the processing of each iteration of the task will be completed within the time frame of that iteration.
  - It does not matter when the processing is performed, provided that it is completed by the end of the frame.
  - The time needed to execute an iteration of a task is highly predictable.

  (The iteration rates required by different tasks differ, but they can be adjusted somewhat to simplify the scheduling).

Scheduling strategy

Four scheduling strategies were considered:

- Fixed preplanned (non-preemptive),
- priority scheduling,
- deadline scheduling,
- simply periodic scheduling.

➢ The scheduling strategy chosen for the SIFT system is a slight variant of the simply periodic method.

- Each task is assigned to one of several priority levels.
- Each priority level corresponds to an iteration rate, and each iteration rate is an integral multiple of the next lower one.

Process Synchronization

- The SIFT inter-task and inter-processor communication mechanism allows a degree of asynchronism between processors and avoids the lockstep traditional in ultrareliable systems.
- Up to 50 μs of skew between processors can readily be accommodated, but even this margin cannot be assured over a 10-hour period with free-running clocks unless unreasonable requirements are imposed on the clocks.
- Thus, the processors must periodically resynchronize their clocks to ensure that no clock drifts too far from any other.

Clock Synchronization

- The traditional clock resynchronization algorithm for reliable systems is the median clock algorithm, requiring at least three clocks.
- each clock observes every other clock and sets itself to the median of the values that it sees.
- The justification for this algorithm is that, in the presence of only a single fault, either the median value must be the value of one of the valid clocks or else it must lie between a pair of valid clock values.
- This algorithm does not prevent a system failure in the worst case.

SIFT Hardware

- Standard elements when possible!
- Special design is needed only in the bus system and in the interfaces between the buses and the processing modules.

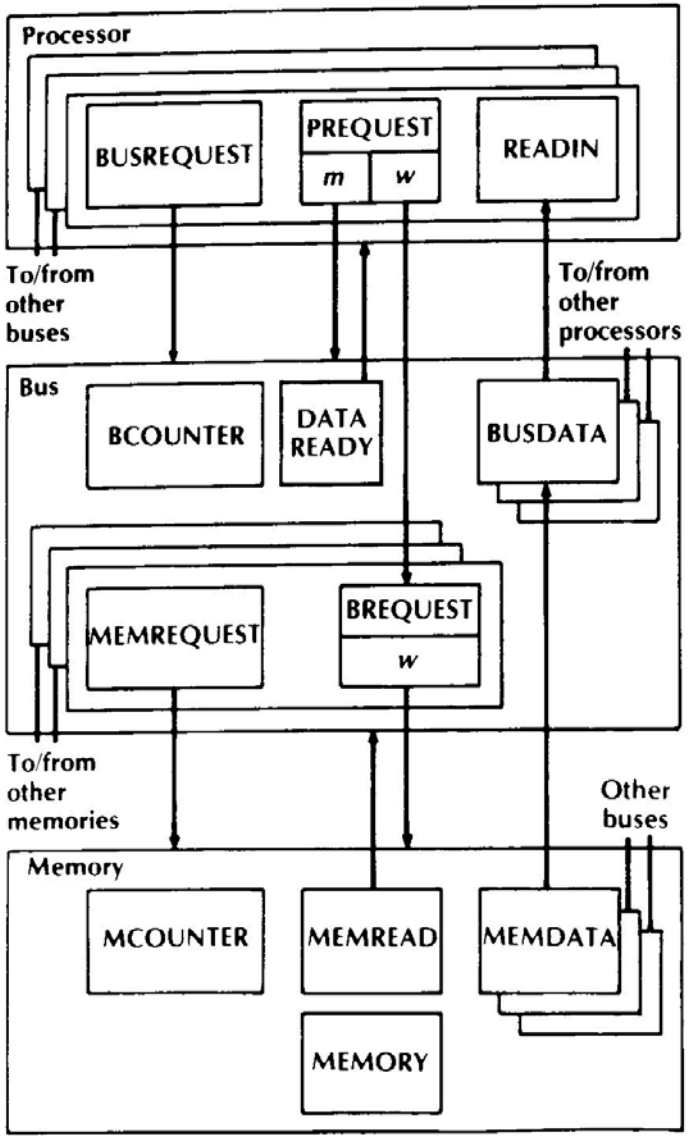
TABLE 10–7 *Major parameters of the SIFT system, engineering model*

System Parameters	Engineering Model	Maximum	System Parameters	Engineering Model	Maximum
Main processors	5*	8	Main Memories		
Main memories	5	8	Word length	16 bits	Same
I/O processors	5	8	Capacity	32K words	64K
I/O memories	5	8	Type	Semiconductor RAM**	Same
Buses	5	8			
External interfaces	5	8	I/O Processors		
Main processors			Word length	8 bits	Same
Word length	16 bits	Same	Type	Intel 8080	Same
Addressing capability	32K words	64K	I/O Memories		
			Word length	8 bits	Same
Speed	500K IPS	Same	Capacity	4K bytes	Same
Arithmetic modes	Fixed point Double length Floating point	Same	Buses		
			Speed	< 10 msec per word Bit serial	Same
Type	Bendix BDμ	Same	I/O Interfaces		
			Type	1553A MIL-STD	Same

*In addition, a spare unit of each type is to be built.
**Program memory would be ROM for actual flight use.

Data Transfers

- The figure shows the connections among processors, buses, and memories.
- The varying replications of these connections are shown for each type of unit.
- Within each unit are shown a number of abstract registers that contain data or control information.
- Arrows that terminate at a register indicate the flow of data to the register.
- Arrows that terminate at the boundary of a unit indicate control signals for that unit.



- The interconnection system units designed especially for the SIFT system are

- the processor-to-bus interfaces,
- the buses,
- the bus-to-memory interfaces.

- These units all operate autonomously and contain their own control, which is implemented as a simple micro-programmed controller.
- The design of the interfaces permits simultaneous operation of all units. For example, a processor can simultaneously read data from its memory and from another memory, while at the same time another processor is reading from the first processor's memory.

The software system

- The software consists of
- - the application software: it performs the actual flight-control computations and it is structured as a set of iterative tasks)
- - the executive software: it is responsible for the reliable execution of the application tasks and implements the error-detection and reconfiguration mechanisms)
-
- Additional support software to be run on a large support computer is also provided.

The Executive software - 1

The SIFT executive software performs the following functions:

1. Run each task at the required iteration rate.
2. Provide correct input values for each iteration of a critical task (masking any errors).
3. Detect errors and diagnose their cause.
4. Reconfigure the system to avoid the use of failed components.

To perform the last three functions, the executive software implements the previously described techniques of redundant execution and majority voting.

The executive software is structured into three parts: the global executive task, the local executive, and the local-global communicating tasks.

The Executive software - 2

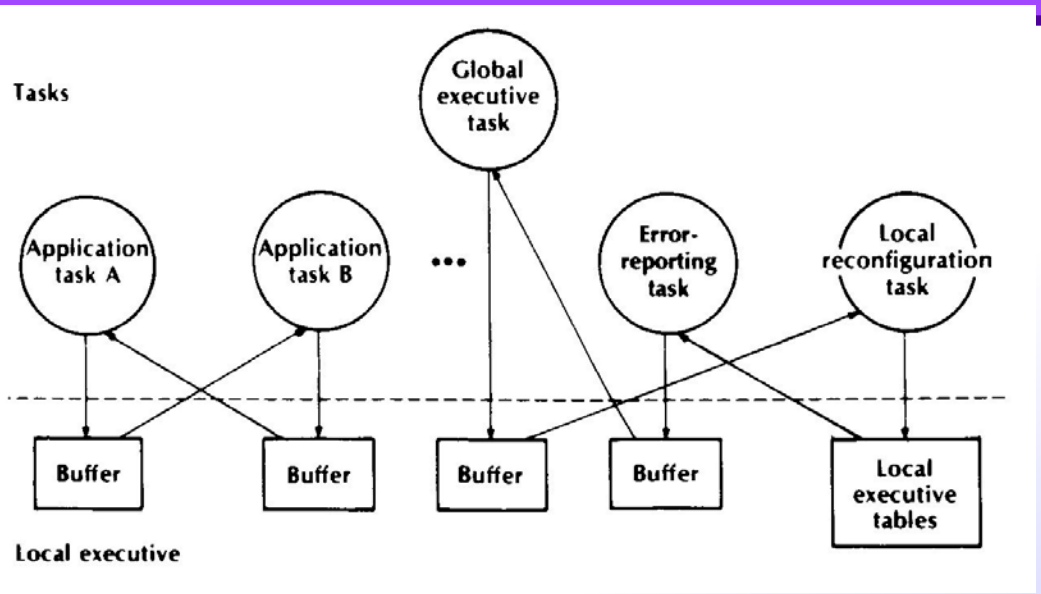
- One global executive task is provided for the whole system.
- It is run just like a highly critical application task, being executed by several processors and using majority voting to obtain the output of each iteration.
- It diagnoses errors to decide which units have failed and determines the appropriate allocation of tasks to processors.

The Executive software - 3

- Each processing module has its own local executive and local-global communicating tasks.
- The local-global communicating tasks are the error-reporting task and the local reconfiguration task.
- Each of these tasks is regarded as a separate task executed on a single processor rather than as a replication of some more global task, so there are as many separate error-reporting tasks and local reconfiguration tasks as there are processors.

Logical structure

- The replication of tasks and their allocation to processors is not visible.



- Tasks communicate with one another through buffers maintained by the local executives.
- Note: the single global executive task is aware of (and communicates with) each of the local executives but the local executives communicate only with the single (replicated) global executive task and not with each other.
- In this logical picture, application tasks communicate with each other and with the global executive but not with the local executives.

FIGURE 10–14
Arrangement of application tasks within SIFT configuration

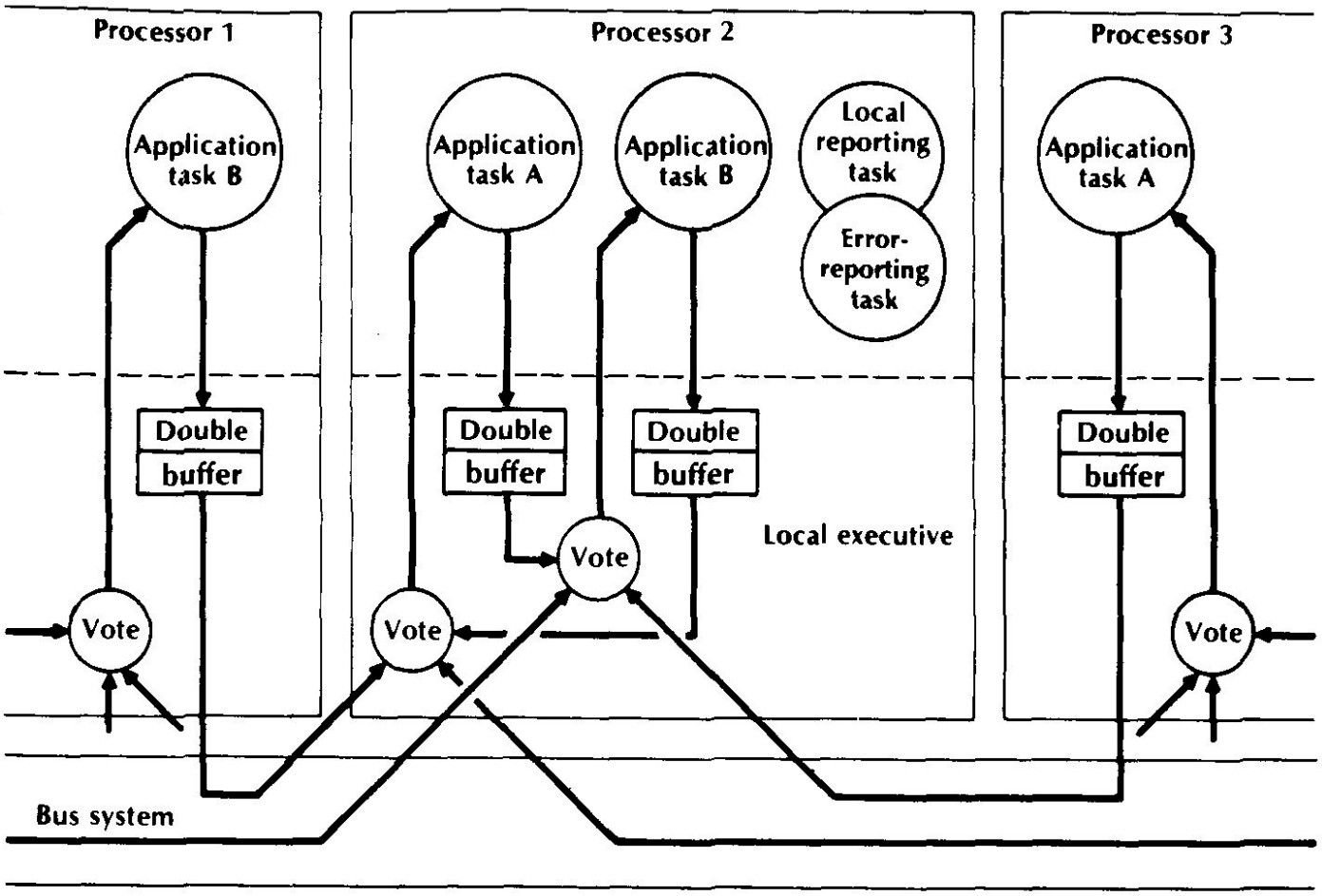
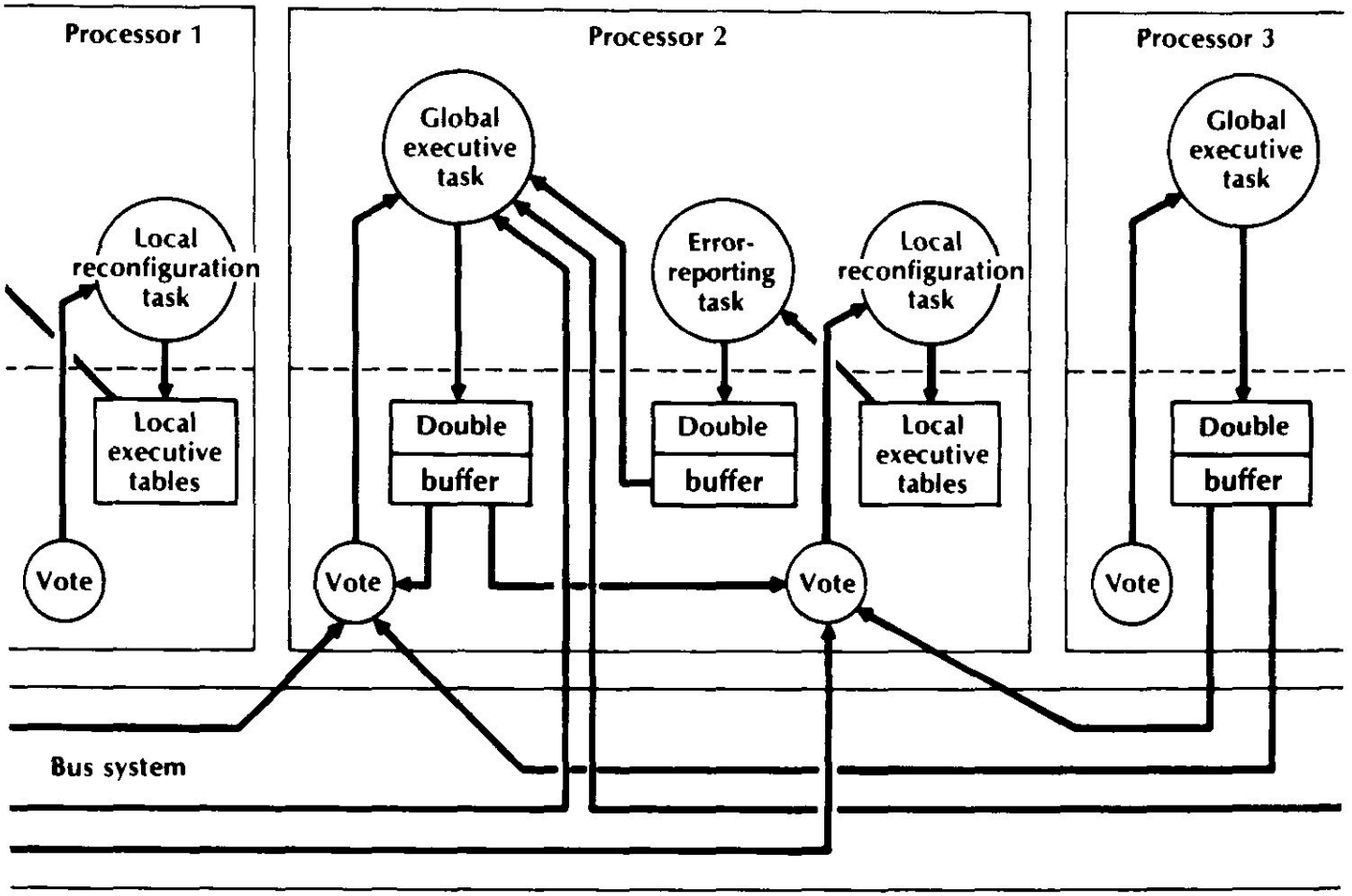


FIGURE 10–15
Arrangement of executive tasks within SIFT configuration



The local-global communicating task

- Each processor runs its local reconfiguration task and error-reporting task at a specified frequency.
- These two tasks communicate with the global executive via buffers.
- The local executive detects an error when it obtains different output values for the same task iteration from different processors.
- It reports all such errors to the error-reporting task which performs a preliminary analysis and communicates its results to the global executive task.
- These results are also used by the local executive to detect possibly faulty units before the global executive has diagnosed the errors.
- The local reconfiguration task maintains the tables used by the local executive to schedule the execution of tasks (it uses information provided to it by the global executive).

The global executive

- It uses the results of every processor's error task to determine which processing modules and buses are faulty.
- When the global executive decides that a component has failed, it initiates a reconfiguration by sending the appropriate information to the local reconfiguration task of each processor.
- The global executive may also reconfigure the system as a result of directives from the application tasks (e.g. in a change of flight phase)

The global executive - 2

- To permit rapid reconfiguration, we require that the program for executing a task must reside in a processor's memory before the task can be allocated to that processor.
- In the initial version of SIFT, there is a static assignment of programs to memories.
- The program for a critical task usually resides in all main processor memories, so the task can be executed by any main processor.

The local executive

- It is a collection of routines to perform the following functions:
- (1) run each task allocated to it at the task's specified iteration rate;
- (2) provide input values to and receive output values from each task iteration;
- (3) report errors to the local executive task.

- A processor's local executive routine can be invoked from within that processor by a call from a running task, by a clock interrupt, or by a call from another local executive routine.
- There are four types of routines.
- - The error-handier routine
- - The scheduler routine
- - The buffer-interface routines
- - The voter routine

Fault detection

- Fault detection is the analysis of errors to determine which components are faulty.
- It is based on the processor/bus error table, an (MxN) matrix, where m is the number of processors and n the number of buses in the system.

- One processor/bus error table for each processor (maintained by its local executive's error handier).
- An entry Xp[i,j] in processor p's table represents the number of errors detected by processor p's local executive that involve processor i and bus j.

Fault detection -2

Suppose that processor p is reading from processor q using bus r. There are five distinct kinds of errors that cause a matrix value to change:

- 1. The connection from bus r to processor q is faulty.
- 2. The connection from processor p to bus r is faulty.
- 3. Bus r is faulty.
- 4. Processor q is faulty.
- 5. Processor p is faulty.

- The Error Handler performs the analysis,
  The possible actions:
  - In case 1, processor p will stop using bus r to talk to processor q.
  - In cases 2 and 3, processor p will stop using bus r, and will report to the global executive that bus r is faulty.
  - In case 4, processor p will report to the global executive that processor q is faulty.

Fault detection -3

- The global executive task makes the final decision about which unit is faulty. To do this, it reads the faulty processor reports provided by the error-reporting task.
  - If two or more processors report that another processor is faulty, then the global executive decides that this other processor has indeed failed.
  - If two or more processors report that a bus is faulty, then the global executive decides that the bus has failed.
- If the global executive is unable to determine the faulty unit, it must await further information (e.g. allocating diagnostic tasks).
- It can be shown that in the presence of a single fault, the above procedure cannot cause the global executive to declare a non-faulty unit to be faulty.

Conclusions

- SIFT has been a 'revolution'
  - The SIFT team provided breakthroughs in the fundamental theory and algorithms for achieving reliable distributed system operation in the presence of Byzantine failure modes, specifically focusing on the key problems of clock synchronization and consensus.
  - SIFT developed and demonstrated the first software based implementation of a fault tolerant computer
  - among the first to create extensive analytical proofs of correctness of their algorithms.
- The impact of this work goes far beyond this implementation in that its groundbreaking conceptual framework spawned an entire new area of distributed systems theory and underlies many existing fault tolerant computer designs.