A good review of techniques for distributed simulation can be found in [3,4]. Generally speaking, a need to evaluate failure-recovery strategies in any distributed simulator can be divided into three components: logical processes (LP), an image, and distributed simulation. In our project, it will be simulator software. The underlying communication mechanism is PVM (Parallel Virtual Machine). The current version of the simulator runs on a network of workstations connected by an Ethernet. The purpose of the simulator is to study recovery strategies for embedded systems. This paper describes a distributed simulation package for real-time systems. The package is intended to be used to evaluate failure-recovery strategies.

Abstract

Key Words—real-time, PVM, distributed, central clock

Amherst, MA 01003
University of Massachusetts
Department of Electrical and Computer Engineering

Kai Yu, Kiron Toumidi, Israel Koren, and C. M. Krishna

SIMULATOR

INTRODUCTION TO A FAULT-TOLERANT

DISTRIBUTED REAL-TIME SYSTEM

Introduction to Modeling and Simulation
The simulator can be viewed as a layered structure as shown in Figure 2. Physical Layer 1

- Task and Fault Generators
- Protocol
- Virtual Networks (VN's), that permit the simulation of various network topologies and
  that is being simulated.
- Virtual Processors (VP's), which are meant to simulate individual processors in the system

The categories into which the VP's can be classified are as follows:

- In the shape of a system console.

A functional view of the simulator is shown in Figure 1. There are three classes of VP's: a clock
passing messages across a network of machines.

PVM packet (x): this packet offers an efficient and transparent communication mechanism for
as mentioned earlier, message passing between the logical processes is supported by the
slowest than optimistic schemes; however, they are simpler to implement.

Processes are almost allowed to free-run, and any violation of consistency is corrected when the event occurs. On the other hand, a conservative scheme does not allow violations
in optimistic schemes, processes can be divided into two types: conservative (1, 6) and optimistic (5).

In practice, in general, they can be distinguished into two types: conservative (1, 6) and optimistic (5). In
a proper synchronization scheme, there is a large number of possible synchronization techniques that
Perhaps the most difficult (and important) part of building a distributed simulator is choosing
results.

The purpose of this synchronization is to maintain sequential consistency in the simulation interprocess message passing scheme for communication, and a mechanism for synchronizing the
Virtual Processor: Master: The master function consists of task allocation, system recovery.

Fault Injection Module: The fault injection module can inject both permanent and transient faults.

Each task, the duration of the transient faults, well as the duration of the transient faults. The user specifies (through the user interface) the fault rate at each processor as tasks.

Fault Injection Module: The fault injection module can inject both permanent and transient faults.

Task Generation Module: Periodic and aperiodic (sporadic) tasks can be defined by these.

Layer consists of the following:

Above Virtual Layer 1 is Virtual Layer 2, which consists of the heart of the simulator. This provides synchronization for the LPS.

Layer Physical Layer 2 is Virtual Layer 1, which is the central clock protocol. This clock

Figure 1: The Functional View of the Simulator
3.2 Clock Mechanism

Results are displayed to the user as they are generated by the simulator.

The console is a 'system initialization' routine, a simulator controller (allowing the user to suspend and restart the simulation) and windows which display the simulation results. These windows provide a user interface to the network model, allowing easy modifications and debugging.

3.1 System Console

3 Implementation

The network protocol. It also includes the algorithm to check the master virtual processor.

Algorithm, the network routing algorithm, the check pointing scheme (to facilitate rollback) and
and scheduling. It provides control about data parameters, specifies the task allocation and scheduling.

Virtual System Layer 3 provides parameters and endtoend for Layer 2. As we have already
seen, it provides information about data parameters, specifies the task allocation and scheduling.

Our simulator is specifically designed to allow easy extensions. For example, a user may decide

IEEE 802.3 protocols on a thin network, as well as point-to-point networks.

Virtual Network: The virtual network module allows the user to simulate the desired
system. They have functions such as task scheduling and execution, and local recovery.

Virtual Processor: Share: Those processes simulate individual processors of the simulated
The appropriate recovery action will be specified by Virtual Layer 3.

The application processors distribute the tasks among the functional processors.
disconnect the failed processor and distribute its tasks among the functional processors.

A spare of one is maintainable (i.e., move all the tasks on the failed processor to the spare), and
3.3 Virtual Processor

The set of virtual processors can be broken down into four subsets:

- **System Initialization:** During system initialization, this unit sets up the local VP.
- **System Maintenance:** During system maintenance, this unit sets up the local VP.
- **System Test:** During system test, this unit sets up the local VP.
- **System Execution:** During system execution, this unit sets up the local VP.

The structure of the virtual processors (VPs) is shown in Figure 4.

### Figure 3: Central Clock

![Central Clock Diagram]

- **LP-2**: Message arrived at VT-1
- **LP-1**: Message sent at VT-1
- **OWC**: One-way message count
- **LCT**: Current virtual time
- **NET**: Next virtual time
- **LET**: Last seen time
- **OMC**: One-way message count
- **(net, LET, OMC)**: The current state of the system

The above approach can give rise to a race condition due to delays in message passing. For the sake of simplicity, assume that a message is transmitted immediately after it is sent by the source processor.
shown in Figure 6. When the system is started, the virtual master is set to unselected. This fact is detected by the virtual structure, schedules tasks, updates status tables appropriately, and passes the structure up to the simulated network structure.

- Slave: The slave simulates the individual processors in the simulated system. It creates the task structure, schedules tasks, updates status tables appropriately, and passes the structure up to the routing table to account for changes in the network structure.

- Virtual Control: The virtual control is responsible for running the master election algorithm, and carries out system reconfiguration when necessary.

- Master: The job of the master is to allocate and reallocate tasks, run fault-detection and recovery, and other housekeeping tasks. It is also responsible for executing user commands and other processes.

Figure 6: Virtual Processor
We would like to acknowledge the technical assistance of Pooja Laxmi, Steve Miron, and...
References