CHAPTER 7
CONCLUSIONS AND FUTURE EXTENSIONS

7.1. INTRODUCTION

An attempt to give all the aspects of shared data/resource, a separate identity of their own led to the introduction of the two concepts, Monitor and Server Task. One major drawback of both these concepts is that of excessive serialisation. However, both these concepts are excellent tools for structuring Distributed Operating Systems functioning as Servers.

In a multiprocessor environment, if the monitor or a server task is scheduled on a processor/node, there would be a great deal of communication to be handled by this processor/node thus leading to a network congestion. A more serious problem occurs if this processor/node breaks down. In this case the entire set of services offered by the monitor or the server task are no more available. It is to be noted that a part of a service being crippled is more desirable than the entire service being lost suddenly.

The objectives of this thesis as mentioned earlier are:

- to propose a structuring concept to overcome the drawbacks discussed above while ensuring mutual exclusion
- to implement the model in a multi-processor environment
- to examine issues pertaining to load-balancing aspects.

These have been achieved and the contributions are given in the next section.
7.2. KEY CONTRIBUTIONS

Distributed Operating System design is a challenging task with several intricacies. Concurrent Monitor is a structuring concept that can be adapted to distributed systems with either centralised or distributed control. The capabilities of Concurrent Monitor to address the Mutual Exclusion and Synchronisation problems is studied in this thesis. The fault-tolerance aspects of the message based implementation is also examined. The use of colored tokens permits a host of load balancing techniques to be used in conjunction with the Concurrent Monitor.

The key contributions of this work are

- A structuring concept for server operating systems which is also an effective solution for the mutual exclusion problem.

- Implementation of Concurrent Monitor on the processor pool model and the work station model of distributed systems.

- A novel memory model to reduce the performance bottleneck in distributed shared memory systems.

- Adaptation of the Concurrent Monitor to provide distributed control by use of colored tokens.

- Simulation studies of the memory model and fault tolerant aspects of the implementation of Concurrent Monitor.

- An adaptation of a load balancing algorithm to greatly reduce the execution time by using additional storage space.
7.3. FUTURE EXTENSIONS

As is evident from the design of Concurrent Monitor it can be supported by a programming language. The language design and the compiler construction form a natural extension to the work presented in the thesis. Preliminary investigations reveal that Intel i860 processor is best suited for implementing the compiler. Automatic identification of the groups of accept statements that can be run on separate processors is another challenging problem to reckon with. The other aspects of operating systems mentioned in Chapter 1 are also important and further study can be carried out on these issues. The key issues are ordering of events, clocks and the termination problem. A large number of approaches to load balancing can be studied with respect to Concurrent Monitor. Fault-tolerance is another vast area that can serve as future extension of the work presented. The arbitrariness involved in the definition of the Ada server task is carried over into the semantics of the Concurrent Monitor. As a consequence, a solution for a synchronisation problem using concurrent monitors may yield incorrect results for certain execution sequences. A comparative study of successful execution traces of Ada model and concurrent monitor has wide ranging applications in the area of distributed data bases.

With the focus now firmly on the Servers and their performance, Concurrent Monitor may serve as an efficient mechanism with better performance.