A DISTRIBUTED ALGORITHM FOR DETECTING COMMUNICATION DEADLOCKS

N. Natarajan

National Centre for Software Development
and Computing Techniques
Tata Institute of Fundamental Research
Homi Bhabha Road, Colaba
Bombay 400 005, India

ABSTRACT

A distributed system is an interconnected network of computing elements or nodes, each of which has its own storage. A distributed program is a collection of processes. Processes execute asynchronously, possibly in different nodes of a distributed system, and they communicate with each other in order to realize a common goal. In such an environment, a group of processes may sometimes get involved in a communication deadlock. This is a situation in which each member process of the group is waiting for some member process to communicate with it, but no member is attempting communication with it. In this paper, we present an algorithm for detecting such communication deadlocks. The algorithm is distributed, i.e., processes detect deadlocks during the course of their communication transactions, without the aid of a central controller. The detection scheme does not assume any a priori structure among processes, and detection is made "on the fly" without freezing normal activities. The proposed scheme is appropriate to be implemented within runtime support or kernel of distributed programming languages.

I. INTRODUCTION

A distributed system is an interconnected network of computing elements, or nodes, each of which has its own storage which is not shared with any other node. A program that executes in the environment of a distributed system is called a distributed program. A distributed program is composed of several processes which cooperate to reach a common goal. The various processes of a distributed program may execute in different
nodes of a distributed system. Since there is no shared storage between the nodes, processes executing in different nodes can cooperate only by communicating messages. If inter-process communication mode is synchronous [5], i.e., if a process can send or receive a message only if another process is ready to perform a matching operation of receiving or sending the message, then sometimes, a group of processes may get involved in a communication deadlock. This is a situation in which each member process of the group is waiting for any one process of a subgroup to communicate with it, but no member is attempting communication with it. A communication deadlock may arise even when the communication mode is asynchronous with respect to the sender, i.e., when the sender does not wait for the message to be received. In such a case, a group of processes is involved in a communication deadlock if every member is attempting only a receive operation and there is no message in transit.

In this paper, we present a distributed scheme for detecting such communication deadlocks. The scheme is appropriate to be implemented within a kernel of a distributed programming language. Such a kernel must detect a communication deadlock when it occurs since the deadlock is a kind of communication failure. When it detects a deadlock, the kernel can report this to some member or all members of the group involved in the deadlock. Upon receiving this notification, a process can initiate appropriate action. If the deadlock is unexpected, its occurrence can be handled by processes as a fault in the design of the program. However, distributed programs can be designed such that the occurrence of a communication deadlock is not a design fault but a normal event. For example, the design of a distributed program can use the occurrence of a communication deadlock to signify the termination of a distributed computation. It has been reported that in some applications, such as distributed simulation, using the occurrence of communication deadlocks in this manner makes the implementation efficient [1]. Distributed programs for implementing relaxation algorithms can also use communication