3. **PVM: Parallel Programming Tools**

**PVM started out as a research project initiated in 1989 by Oak Ridge National Laboratory (ORNL) in the United States. Since then, the project has taken another turn, and other organizations and research laboratories have become involved (ORNL, University of Tennessee, Carnegie Mellon University and Pittsburgh Supercomputing Centre at Emory University, Atlanta). The research carried out on this project is funded mainly by the research programme in applied mathematics by the American Energy Research Laboratory, the American Department of Energy, the American National Foundation for Science and Tennessee State. This series of software tools is available free to the world scientific community, in the interests of scientific research.**

3.1 **PVM Presented**

PVM (parallel virtual machine) is a series of software tools aimed at facilitating multi-agent distributed programming on heterogeneous Unix machines (sequential, parallel or supercomputers), interconnected via a network. These machines can be PCs under Linux or SCO, Unix stations, or even parallel computers or supercomputers connected up to the network. PVM enables the user to see all of the machines as if they form the CPU (processor) network of a huge virtual multiprocessor computer lacking a shared global memory (Fig. 3.1), similar to transputer machines. Each Unix machine in the network is called the *host* of the virtual parallel machine. It will therefore be possible to share the processes of an application requiring a large computation (large simulation programs, multi-agent distributed applications in A.I, image processing applications, complex multimedia applications, etc.) between the different machines available on the network. In this chapter, the term virtual machine will designate this series of interconnected machines.

PVM proposes primitives for:

- initialization and automatic triggering of tasks on the virtual machine;
- establishing communication between tasks;
- synchronization between tasks, etc.
With PVM, a task constitutes the unit of refinement of a distributed application. Recall that a distributed application is composed of a series of tasks or agents communicating via the exchange of messages. Each task fulfils a specific function conforming to the general semantics of the distributed application. PVM offers facilities which allow the implementation of the majority of interaction and execution models seen in the preceding chapters (Fig. 3.2).

PVM supports heterogeneity at the application, machine and network levels. In doing this, it enables the tasks of an application to profit from the available architecture, to find the best possible organization and structure for the application. This particularly occurs with the allocation of tasks with high CPU time consumption to machines within the network which possess the required capabilities. PVM supports all types of necessary data conversion to allow the exchange of data between machines using different internal representations (words, integers, floating points). It also enables the interconnection of *hosts* to the virtual machine using different network types (Ethernet, token, packet transmission: Transpac, or simply line series). A PVM system is composed of two parts:

1. The distributed execution kernel (see DES in Chapter 1). This is supported by a DAEMON Unix process: pvm3 or simply pvmd. This process is present or duplicated on all hosts of the virtual machine. The pvm3 kernel was designed in this way so that any Unix system user could install it without difficulty on this machine.

2. User interface or PVM primitives library (libpvm.3). This contains the routines or functions that can be used in user programs to create tasks and establish communication between them, to co-ordinate and synchronize tasks and for