Abstract

The last decade has seen phenomenal growth in real-time systems. This trend is likely to continue and soon real-time systems will be pervading all streams of science and engineering. In spite of the growing popularity of real-time systems, their architectures still remain ad hoc and are very difficult to modify and maintain. The problem we believe is the lack of an architecture that binds the largely heterogeneous modules of real-time computation into an architectural framework. This forms the basis of this thesis. We make an attempt towards developing the concepts for building a general-pupose architecture for real-time systems. To this end, we propose a layered architecture for real-time systems, which we call LARTS (Layered Architecture for Real-Time Systems). LARTS structures various functional modules into a set of hierarchical layers, with the objective of making the architecture *flexible* by being applicable to a range of applications, *adaptable* to changes in technology, and *capable* of accommodating heterogeneity in a processing environment. We have explored some key issues such as, segmentation, scheduling, fault-tolerance and processing environment to study the functioning of LARTS.

With real-time systems making their appearance in large number of application areas, the nature of real-time tasks has changed. Tasks are no longer atomic in nature and often consist of several subtasks. However, most research efforts assume that each task is atomic and is an indivisible entity. In most situations a single value of an end-to-end global deadline restricts the parallel scheduling of these entities. To counter this, we have proposed three subtask deadline assignment policies, which are aimed at improving the schedulability of real-time tasks. The simulation results based on these policies suggest that substantial improvements in schedulability of real-time tasks can be achieved by assigning individual deadlines to subtasks.

For providing fault-tolerance, we used the primary-backup approach where two copies of a subtask are scheduled on different processors. Concepts such as *active* and *passive backups*, and three variants to this approach – BASP (Backup

As Soon as Possible), BALP (Backup As Late as Possible) and BOL (Backup Over Loading), are discussed in the thesis. To further investigate these concepts, fault-tolerant scheduling algorithms for three different processing environments namely: (i) shared-memory symmetric multiprocessor, (ii) distributed system and (iii) hypercube multicomputer, have been developed. The performance of the fault-tolerant scheduling algorithms for different subtask deadline assignment policies have been studied by constructing three simulation models based on the processing environments mentioned above and by carrying out extensive simulations.

Due to better resource reclamation/deallocation properties, the BALP scheme has better schedulabilty charactersitcs compared to BASP scheme for all three processing environments. The STF (Smaller Tasks First) subtask deadline assignment policy introduced by us manages the *laxity* more efficiently compared to the other schemes. Hence, the STF-BALP scheme performs better than other strategies for all three processing environments.