Communications are Everything: A Design Methodology for Fault-Tolerant Concurrent Systems

A.M. Tyrrell
Department of Electronics
University of York, Heslington, York, Y01 5DD, Email: amt@ohm.york.ac.uk

Abstract
Limiting the extent of error propagation when faults occur and localising the subsequent error recovery are crucial elements in the design of fault tolerant parallel processing systems. Both activities are made easier if the designer associates fault tolerance mechanisms with the underlying communications of the system. With this in mind, this paper has investigated the design of such systems, which enforces a design concentrating on the modelling and analysis of interprocess communications providing a better match between the needs of the fault-tolerant mechanisms and the communication structures themselves.

Keywords
Fault-tolerance, concurrent systems, communications, software design.

1 INTRODUCTION
A distributed processing system, comprising a set of discrete processing units, offers the user not only the prospect of increased efficiency and throughput through parallelism, but its inherent redundancy might also be exploited to enhance reliability. To do so requires a properly designed fault tolerance infrastructure which maintains the integrity of the system under fault conditions, in particular communications. This paper describes a design methods which concentrates on the communications within the system, which facilitates the design, placement and implementation of fault-tolerant software mechanisms across a parallel system to ensure safe operations in the presence of faults.

Fault tolerance is often incorporated into a design as a ruggedisation process to protect a process or set of processes regarded as critical to safe system operation (Lee and Anderson 1991). The fault tolerance mechanisms are required to recognise faults by the errors they cause and to prevent error migration from the faulty process to elsewhere in the system, so that error recovery is localised. The extent of the error recovery operation can be limited if the communications structure in the system can be analysed accurately, and a boundary can be identified within the state-space of the distributed system across which error propagation by interprocess communication is impossible; it must include all processes which interact with the function being protected and exclude all processes that do not interact with it. In other words, the state-space of the system has to be partitioned into a hierarchy of atomic actions (Jalote and Campbell 1986). It is then possible to design a distributed error detection and recovery mechanism around the atomic action which ensures that all the processes affected by the fault
co-operate in recovery. This localisation of fault tolerance simplifies the design and can help to meet timing constraints in real-time systems (Anderson and Knight 1983).

The design described in this paper concentrates on the communications mechanisms within an application, and within the fault tolerance mechanisms themselves. The design shows how different communication structures help not only in the design of the particular application itself, but more importantly in the design of the fault-tolerant mechanisms protecting the system against faults latent in the system.

2 ATOMIC ACTIONS AND FAULT-TOLERANCE

Firstly, let us consider the crucial role communications play in the operation of fault-tolerant mechanisms in a parallel processing environment. To an external observer the activity of a process is defined by its sequence of external interactions; any internal actions (of which there may be many) can not affect the external observer, at least until the next external interaction. This allows the concept of an atomic action to be derived: the activity of a set of processes is defined as an atomic action if there are no interactions between that set of processes and the rest of the system for the duration of that activity. The extension to hierarchically nested atomic actions is straightforward. These concepts are well-known in distributed transaction processing (Mancini and Shrivastava 1988) from which field many other attributes of atomic actions, such as serialisability, failure atomicity and permanence of effect can be defined.

The process of identifying the atomic actions within a parallel system design brings into clear focus the structure of interprocess interactions and thus the route by which errors might propagate under fault conditions — an obviously crucial aspect in the detection and implementation of the fault tolerant mechanism. All common mechanisms for providing fault tolerance in parallel systems, such as forward error recovery (Randell 1975), N-version programming (Avizienis 1985), conversations (Randell 1975), consensus recovery blocks (Scott et al. 1987) and distributed recovery blocks (Kim and Welch 1989), have to cope with error confinement and achieve this by imposing logic structures 'around' atomic actions.

A generalised fault tolerant mechanism could be considered as a co-ordinated set of recoverable blocks, with one recoverable block in each interacting process, allowing distributed error detection and recovery. The mechanism is bounded by a set of start states (entry line), a set of finish states (exit line) and two side walls which completely enclose the set of interacting processes which are party to the mechanism, and across which interprocess interactions are prohibited. The structure is indicated diagrammatically in Figure 1. Note that it is the communication pattern that defines the side walls, processes which are interacting are within the side walls (processes R, S and T), processes which do not interact are outside the side walls (processes P and Q).

Two types of communications are illustrated in Figure 1; the lines between the 'recoverable processes' represent the application interactions, and are of a consequence of data requirements between the parallel processes. It is these interactions that will define where atomic action exist within the system structure, and thus where fault-tolerant mechanisms should be placed. The second type of communications are those forced upon the application by the fault-tolerant mechanisms. These will typically consist of exchanging data values for voting and/or for comparison, of passing reconfiguration information and signals around the system, and for the recovery of the parallel processes within the fault-tolerant mechanism. This second class of communication would not be present in non fault-tolerant systems, and in many respects should be more secure than the 'normal' application communications.

The entry line defines the start of the atomic action and consists of a co-ordinated set of recovery points for the participating processes. These processes may enter the atomic action asynchronously. The exit line comprises a co-ordinated set of acceptability tests, or voting procedures. Only if all participating processes pass their respective acceptability tests (or the voting procedures are successful) is the mechanism deemed successful and all processes exit, in synchronism, from the action. If any acceptability test is failed, recovery is initiated and