Formal specification of N-modular redundancy

L. Mancini
M. Koutny

Abstract

This paper investigates N-Modular Redundancy (NMR) in the form of replicated computations in a concurrent programming model consisting of communicating processes. A formal specification of NMR is given to express the correct behaviour of the system in the presence of non-determinism. The COPY path expressions formalism is used as a formal model. Then some implementations are proposed which satisfy the given specification. This approach permits redundant systems to be robust with respect to failures in redundant processors, and also permits the use of software fault tolerance techniques such as N-version programming.


Series Editor: M. J. Elphick

© 1986 University of Newcastle upon Tyne.
Printed and published by the University of Newcastle upon Tyne, Computing Laboratory, Claremont Tower, Claremont Road, Newcastle upon Tyne, NE1 7RU, England.
Bibliographical details

MANCINI, Luigi

Formal specification of N-modular redundancy. By L. Mancini and M. Koutny.

Newcastle upon Tyne: University of Newcastle upon Tyne, Computing Laboratory, 1986.

(University of Newcastle upon Tyne, Computing Laboratory, Technical Report Series, no. 213.)

Added entries

KOUTNY, Maciej.
UNIVERSITY OF NEWCASTLE UPON TYNE.

Abstract

This paper investigates N-Modular Redundancy (NMR) in the form of replicated computations in a concurrent programming model consisting of communicating processes. A formal specification of NMR is given to express the correct behaviour of the system in the presence of non-determinism. The COPY path expressions formalism is used as a formal model. Then some implementations are proposed which satisfy the given specification. This approach permits redundant systems to be robust with respect to failures in redundant processors, and also permits the use of software fault tolerance techniques such as N-version programming.

About the author

Mr. L. Mancini has been at the Computing Laboratory since May, 1985 as a Research Associate.
Maciej Koutny is at the Computing Laboratory as a Research Associate from March 1985 to April 1986. Prior to this he was at Warsaw Technical University in Poland from 1982.

Suggested keywords

AGREEMENT
COSY
NONDETERMINISM
PATH EXPRESSIONS
RELIABILITY
REPLICATED PROCESSING
VOTING

Suggested classmarks (primary classmark underlined)

Dewey (18th): 001.6425 001.64404
U. D. C. 681.322.06
Formal Specification of N-Modular Redundancy

L. Mancini and M. Koutny (*)
Computing Laboratory
The University
Newcastle upon Tyne, NE1 7RU, U.K.

ABSTRACT

This paper investigates N-Modular Redundancy (NMR) in the form of replicated computations in a concurrent programming model consisting of communicating processes. A formal specification of NMR is given to express the correct behaviour of the system in the presence of nondeterminism. The COSY path expressions formalism is used as a formal model. Then some implementations are proposed which satisfy the given specification. This approach permits redundant systems to be robust with respect to failures in redundant processors, and also permits the use of software fault tolerance techniques such as N-version programming.

Index Terms – nondeterminism, voting, agreement, replicated processing, reliability, COSY, path expressions.


(*) On leave from the Institute of Mathematics, Warsaw Technical University, 00-661 Warszawa, Poland.
1. Introduction

The need to ensure correct input-output behaviour, and a high level of fault-masking in the case of real-time systems, has led designers to consider the application of N-Modular Redundancy (NMR) in the construction of software.

The SIPT [9] aircraft control computer system, for example, has demonstrated the possibility of obtaining reliable computations through the replication of programs on multiple computers and the use of majority voting.

This solution has many advantages, which include:
- the continuity of correct input/output behaviour in the case of real-time control systems is ensured;
- reliability is independent of the particular strategy of resource management;
- transient faults are masked by voting and do not cause reconfiguration.

This form of redundancy involves implementation problems if nondeterminism is allowed in the computational model, since it requires that in the absence of faults all the instances of a process have the same behaviour. However, adherence to this requirement is problematic in distributed computing systems.

The solution adopted by SIPT though simple, is very restrictive [8]. Consistency is maintained through adopting constraints on scheduling and communication. In particular, a planned schedule ensures that task results are available when required by other tasks (tasks are never required to wait for input values). In contrast, it has been shown in [7] that one can extend the concept of SIPT-like NMR:
- to provide fault tolerance support for a wide class of programs;
- to allow greater asynchrony between executions of program replications;
- to allow software fault tolerance techniques, such as N-version programming [1] to be incorporated in the NMR systems.

This approach has been based on the observation that certain kinds of nondeterminism can produce inconsistency in majority voting even in the absence of faults.

The present paper is intended to provide a formal specification of a fully decentralized design for solving all consistency problems in NMR systems. It uses the COSY (Concurrent System) path expression formalism [6, 5] to define a specification and analyse problems related to the implementation of NMR systems. The concurrent programming model which has been chosen is based on a set of active entities, or processes, which run in a local protected environment, and interact using message passing only.

The paper is structured into five sections. In Section 2, the NMR requirements are presented. In order to model the synchronization requirements of an NMR system, in Section 3 the basic COSY notation is introduced, and in Section 4 a formal specification of NMR is given. In Section 5, two possible solutions to the problem of implementing modular redundancy are proposed. The first solution is based on the notion of atomic actions. The second solution utilizes a distributed algorithm for cooperation between the instances of the kernel of the system nodes.
2. NMR requirements

Every NMR implementation must satisfy the following requirement:

REQ All the processors of a NMR node service input requests in an identical order.

It is worth noting that according to requirement REQ all the non-faulty processors of
NMR node have to resolve nondeterminism (local or global) in an identical manner.

In a distributed system, meeting this requirement is a surprisingly difficult task.
For example, suppose that the code of the process Server allows input requests to be
received from two different processes ClientA and ClientB, and that to increase reliabil-
ity it is decided to triplicate the Server.

Assume that ClientA and ClientB send service requests srA and srB. Because of com-
munication delays, it is possible, for example, for Server1, Server2, and Server3 to
perform requests when:

- Server1 has received only the service request of ClientA;
- Server2 has received the service requests of both ClientA and ClientB;
- Server3 has received only the service request of ClientB;

Since the three copies are nondeterministic, Server1, Server2, and Server3 will
generally service different requests and so contravene requirement REQ. (It is worth
noting that a similar situation can also occur if nondeterminism in the communication
is expressed by input ports only).

In order to avoid a diversity of the computation which will generally result in
the sending of different messages, the implementation of communications must satisfy
requirement REQ, so as to prevent situations like the one described above.

3. Basic COSY Notation

COSY (Concurrent SYstems) is a formalism intended to simplify the study of syn-
chronic aspects of concurrent systems by abstracting away from all aspects of systems
except those which have to do with synchronisation.

A basic COSY program or generalized path is a collection of single paths enclosed
in program and endprogram parentheses. A single path is a regular expression
enclosed by path and end, as for instance:

Po = program
    path a;b,c end
    path (d:e)*;b end
endprogram

In every regular expression like the above, the semicolon denotes sequence (con-
catenation), and the comma denotes mutually exclusive choice. The comma binds more
strongly than semicolon, so that the expression "a;b,c" means 'first a, then b or c'. An
expression may be enclosed in conventional parentheses with Kleene star appended, as
for instance "(d:e)" which means that the enclosed expression may be executed zero or
more times. The expression appearing between path and end is implicitly "starred", so
that a single path describes a cyclic sequential subsystem. The formal description of
the COSY syntax may be found for instance in[5, 6].
The semantics of generalized paths can be described by means of vectors of strings.
For every regular expression \( E \), let \( |E| \) denote the regular language described by \( E \).
With every single path \( P = \text{path body end} \), we associate its set of events \( \text{Ev}(P) \) and its sets of cycles, \( \text{Cyc}(P) = \|\text{body}\| \).
For every alphabet \( A \) and every language \( L \subseteq A^* \), let

\[
\text{Pref}(L) = \{x \mid \exists y \in A^*, \ xy \in L\}
\]

From the set \( \text{Cyc}(P) \) we construct the set of firing sequences of \( P \), denoted by \( \text{FS}(P) \), as follows:

\[
\text{FS}(P) = \text{Pref}(\text{Cyc}(P)^*) = \text{Cyc}(P)^* \text{Pref}(\text{Cyc}(P))
\]

With each generalized path

\[
P_0 = \text{program } P_1 \ldots P_n \ldots \text{end program}
\]

we associate the set of events \( \text{Ev}(P_0) = \text{Ev}(P_1) \cup \ldots \cup \text{Ev}(P_n) \), and the set \( \text{VFS}(P_0) \), its set of permitted histories. To define \( \text{VFS}(P_0) \) some additional notions are necessary.
Let us put \( A = \text{Ev}(P_0), A_i = \text{Ev}(P_i) \) for \( i = 1,\ldots,n \).
For every \( i = 1,\ldots,n \), let \( h_i : A^* \rightarrow A^* \) be an erasing homomorphism given by:

\[
a \in \text{Ev}(P_i) \Rightarrow h_i(a) = a \quad \text{and} \quad a \notin \text{Ev}(P_i) \Rightarrow h_i(a) = \epsilon
\]

for every \( a \in A \), where \( \epsilon \) denotes the empty string.
Let us define a concatenation on \( A_1^* \times \ldots \times A_n^* \) as follows:

\[
(x_1,\ldots,x_n)(y_1,\ldots,y_n) = (x_1y_1,\ldots,x_ny_n)
\]

for all \( (x_1,\ldots,x_n),(y_1,\ldots,y_n) \in A_1^* \times \ldots \times A_n^* \).
For every \( x \in A^* \), let \( \overline{x} = (h_1(x),\ldots,h_n(x)) \), and let \( \text{Vev}(P_0) = \{\overline{a} : a \in A\} \).
The set of all possible histories (behaviours) of \( P_0 \), denoted by \( \text{VFS}(P_0) \), is defined by:

\[
\text{VFS}(P_0) = (\text{FS}(P_1) \times \ldots \times \text{FS}(P_n)) \cap \text{Vev}(P_0)^*
\]

A generalized path \( P_0 \) is said to be adequate (resp. deadlock-free) iff

\[
\forall x \in \text{VFS}(P_0) \forall a \in \text{Ev}(P_0) \exists z \in \text{Vev}(P_0)^*, \ xza \in \text{VFS}(P_0)
\]
\[
\text{(resp. } \forall x \in \text{VFS}(P_0) \forall a \in \text{Ev}(P_0) \exists z \in \text{Vev}(P_0)^*, \ xa \in \text{VFS}(P_0) )
\]

Adequacy is a property akin to absence of partial system deadlock.
4. Formal Specification of NMR

The systems we are considering consist of a set of NMR nodes interacting via
direct communication. Logical connections are established by means of input and output
ports. Each NMR node possesses exactly one output port and at least one input port.

[Diagram of system organization (NMR-graph)]

At a higher level of abstraction a system may be represented as in Fig. 1. Here,
circles represent NMR nodes, small white circles represent output ports, and small
black circles represent input ports. Each pair of corresponding input and output
ports is connected by a directed arc which represents a communication channel from
output port to input port. One may observe that the input port a is not connected to
any other ports of the nodes N1, N2 and N3. This means that a is devoted to external
communication, i.e. it can receive requests from the outside environment. Each node
with such a port (itself called an external port) will be called a gateway node. The
whole picture shown in Fig. 1 will be called an NMR-graph.

An NMR-graph models the whole system of communicating NMR nodes. The internal
structure of an NMR node is illustrated in Fig. 2, where the NMR node N3 is taken as an
example. It is composed of n (n is fixed) independent modules Mi which are
represented as pairs of subcomponents, Vi and Ti, corresponding to the voting and the
task computation performed by Mi. One can see that the ports of N3 are replicated and
assigned to each module Mi.

It is worth noting that nodes of an NMR-graph do not share output ports. On the
other hand, input ports can be shared, e.g. the nodes N1 and N2 share the input port b.

The form of communication which is discussed is synchronous and can be considered
as "one-to-many", i.e. a message sent from an output port, say b, is simultaneously and
instantaneously received at all input ports b.

Let us consider an NMR-graph G with the set of NMR-nodes \{Nj: j \in J\}, |J| = r \geq 1.
The output port of each node Nj will be denoted by \overline{p}_j. Moreover, we assume (pi: i \in I),
I \cap J = \emptyset, to be the set of external input ports of G.
The basic kind of event we will use is:

$m[j,k]$ - message $(j \in (I \cup J), 1 \leq k \leq n)$:
message sent by the $k$-th module in node $N_j$ (if $j \in I$, then $m[j,k]$ denotes the $k$-th request received at external input port $p_j$).

Let $N_j$ be a node (shown in Fig. 3) with output port $p_j$ and input ports $p_1, ..., p_m$. To specify the behaviour of $N_j$ the following other types of events will be used:

$v[j,k,i]$ - voting:
- voting of $n$ messages received at input port $p_i$ which is performed by the module $M_k$ in $N_j$;

$vm[j,k,i]$ - voted message:
- voted message (related to $n$ messages received at input port $p_i$ in $M_k$) which will be performed by the task $T_k$ in $N_j$;

$sf[j,k,i]$ - set flag:
- the setting of a flag which allows $M_k$ in $N_j$ to receive the $n$-th message on input port $p_i$;

$rf[j,k,i]$ - release flag:
- the releasing of the flag set by $sf[j,k,i]$;

$t[j,k,i]$ - task:
- computation of task $T_k$ in $N_j$ related to messages received by $M_k$ at input port $p_i$.

The behaviour of the module $M_k$ in $N_j$ is specified by the sequence of single paths Module$(j,k)$ defined as follows:
Fig. 3 Internal structure of Nj.

\[
\begin{array}{c}
M_1 \\
p_1 : V_1 \rightarrow T_1 \quad p_j \\
\vdots \\
M_n \\
p_1 : V_n \rightarrow T_n \quad p_j
\end{array}
\]

Module\((j,k) = \)
\[
\begin{align*}
\text{IP}(j,k,1) & \ldots \text{IP}(j,k,m) \\
\text{VT}(j,k,1) & \ldots \text{VT}(j,k,m) \\
\text{FM}(j,k) & \\
\text{TC}(j,k)
\end{align*}
\]

(I) (II) (III) (IV)

Single paths appearing in Module\((j,k)\) are defined in the following way \(1 \leq i \leq m\):

\[
\begin{align*}
\text{IP}(j,k,i) &= \text{path } \text{Exp}(m[i,1],\ldots,m[i,n]) \text{ end} \\
\text{VT}(j,k,i) &= \text{path } W_1;\ldots;W(n-1);s[j,k,i];(r[j,k,i];s[j,k,i])^*; \\
&\quad W_n;v[j,k,i];v[m[j,k,i];r[j,k,i] \text{ end} \\
\text{FM}(j,k) &= \text{path } s[j,k,1],\ldots,s[j,k,m];r[j,k,1],\ldots,r[j,k,m] \text{ end} \\
\text{TC}(j,k) &= \text{path } (v[m[j,k,1],\ldots,v[m[j,k,m]];t[j,k];m[j,k] \text{ end}
\end{align*}
\]

where

\[
\begin{align*}
W_1 &= \ldots = W(n-1) = W_n = (m[i,1],\ldots,m[i,n]) \\
\text{Exp}(m[i,1],\ldots,m[i,n]) \text{ denotes any regular expression written in the COSY syntax which generates the language consisting of all strings } x_1x_2\ldots x_n \text{ satisfying:} \\
x_i &\in \{m[i,1],\ldots,m[i,n]\} \text{ and } x_i \neq x_j \text{ for } i \neq j. \\
\text{For example, } \text{Exp}(m[i,1],m[i,2]) \text{ may be defined as } (m[i,1];m[i,2]);(m[i,2];m[i,1]).
\end{align*}
\]

Referring to Fig. 3, the sequence of single paths (I),(II),(III) expresses the behaviour of \(V_k\), and the single path (IV) expresses the behaviour of \(T_k\). In particular:

\[
\text{IP}(j,k,i) = \text{(Input Port)}; \\
\text{specifies the behaviour of the input port } p_i \text{ which is associated with the module } M_k \text{ in } N_j;
\]
\( Vt(N_j,M_k,N_i) \) - (Voting):

corresponds to the voting of messages received at the input port \( pi \) which is associated with the module \( M_k \) in \( N_j \);

\( FM(j,k) \) - (Flags Manager):

associated with the module \( M_k \) in \( N_j \) allows at most one of \( m \) flags (each one assigned to a different input port) to be set at any time. This, however, (see the definition of (II)) prevents the module \( M_k \) from receiving \( n \) messages at two different input ports at any time;

\( TC(j,k) \) - (Task Computation):

specifies the behaviour of the task \( T_k \).

The specification of the behaviour of \( N_j \) is the sequence of single paths defined as follows:

\[ \text{Node}(N_j) = \text{Module}(j,1) \ldots \text{Module}(j,n) \]

Finally, assuming that \( J = \{1, \ldots, r\} \), the formal specification of the NMR system which is described by the NMR-graph \( G \) is expressed by the following generalized path:

\[ \text{NMRSYST}(G) = \text{program Node}(N_1) \ldots \text{Node}(N_r) \text{ endprogram.} \]

The specification presented satisfies the fundamental requirement of NMR systems, namely REQ which is introduced in Section 2. To prove this we need some auxiliary notions. Let

\[ P = \text{NMRSYST}(G) = \text{program P}_1 \ldots \text{P}_l \text{ endprogram} \]

and let \( \text{VM} \) denotes the set of all events of \( \text{Ev}(P) \) which represent voted messages, i.e. those which have the form \( \text{vm}[j,k,i] \).

For every \( x \in \text{VFS}(P) \), and for every \( i \in \{1, \ldots, l\} \), let \( x|P_i = x_i \), where \( x = (x_1, \ldots, x_i, \ldots, x_l) \), i.e. \( x|P_1, \ldots, x|P_l \) denote local histories of sequential subsystems which are represented by \( P_1, \ldots, P_l \).

Finally, let \( H : \text{Ev}(P)^* \rightarrow (I \cup J)^* \) denote a homomorphism defined as follows:

\[ a \in (\text{Ev}(P) - \text{VM}) \Rightarrow H(a) = 6 \]
\[ a = \text{vm}[j,k,i] \in \text{VM} \Rightarrow H(a) = i \]

for each event \( a \in \text{Ev}(P) \).
DEFINITION

We say that a history \( x \in \text{VFS}(P) \) satisfies REQ iff for every \( j \in J \) and for all \( k, k' \in \{1, \ldots, n\} \) one of the following two conditions is satisfied:

\[
H(x; \text{TC}(j,k)) = H(x; \text{TC}(j,k'))
\]

\( \frac{1}{2} b \in (I \cup J) \), \( H(x; \text{TC}(j,k)) = H(x; \text{TC}(j,k')) b \) or

\[
H(x; \text{TC}(j,k')) = H(x; \text{TC}(j,k)) b
\]

That is, if a history satisfies a requirement REQ, then all \( n \) modules of each particular NMR node performed the same sequences of requests or these sequences differ in that there is one request (represented by \( b \)) which has not yet been performed by a subset of modules, say \( \{M_1\} \). In such a case \( b \) is the next request to be performed by \( \{M_1\} \).

THEOREM

Each history \( x \in \text{VFS}(P) \) satisfies REQ.

Proof (outline)

Without loss of generality we assume that \( k = 1, k' = 2 \), and that \( N_j \) is defined as before (see Fig.3). Let \( x \in \text{VFS}(P) \).

For every \( i \in \{1, \ldots, m\} \) we define \( S<x(Vt(j,1,i)) \) as follows:

Let \( y = x(Vt(j,1,i)) \). If \( y \in \text{Cyc}(Vt(j,1,i)) \), then \( S<y> = 0 \); otherwise \( y \) may be unambiguously decomposed as \( y = wz \), where \( w \in \text{Cyc}(Vt(j,1,i)) \) and \( z \in \text{Pref}(\text{Cyc}(P)) \). \( z \in \text{Pref}(\text{Cyc}(P)) \), and we define \( S<y> \) to be the number of occurrences of events representing messages (i.e. \( m[1,1], \ldots, m[i,n] \)) within \( z \).

(1) \( S<x(Vt(j,1,i)) \neq n \)

For every \( s \in (I \cup J) \), and for every \( b \in (I \cup J) \) we denote by \( \#(b,s) \) the number of occurrences of \( b \) within \( s \).

One can prove that:

\[
\#(i,H(x; \text{TC}(j,1))) - \#(i,H(x; \text{TC}(j,2))) \leq 1
\]

for every \( i \in \{1, \ldots, m\} \). Moreover, if (for instance)

\[
\#(i,H(x; \text{TC}(j,1))) - \#(i,H(x; \text{TC}(j,2))) = 1
\]

then

\[
S<x(Vt(j,2,1)) = n
\]

We prove the theorem by induction on \( l(x) \), the length of \( x \), which is standardly defined as follows:

\[
l(x) = 0
\]

\( \forall x \in \text{VFS}(P) \forall a \in \text{Ev}(P) \), \( l(xa) = 1 + l(x) \)
Assume that \( x \in \text{VFS}(P) \) satisfies \( \text{REQ} \), and that \( xa \in \text{VFS}(P) \), where \( a \in \text{Ev}(P) \). Clearly, if \( a \notin A = \{\text{vm}[j,1,1], \ldots, \text{vm}[j,1,m], \text{vm}[j,2,1], \ldots, \text{vm}[j,2,m]\} \), then \( xa \) satisfies \( \text{REQ} \), so we assume \( a \in A \).

If \( H(x|TC(j,1)) = H(x|TC(j,2)) \), then \( xa \) satisfies \( \text{REQ} \). Otherwise, by the induction hypothesis, we may assume (without loss of generality) that \( H(x|TC(j,1)) = H(x|TC(j,2))k \), where \( k \in \{1, \ldots, m\} \). Thus, by (iii), \( S<xa|Vt(j,2,k)> = n \).

We now consider two cases:

**Case 1:** \( a = \text{vm}[j,1,1] \). By (ii), \( i \neq k \). Thus, by (iii), \( S<xa|Vt(j,2,i)> = n \). On the other hand, \( S<xa|Vt(j,2,k)> = S<x|Vt(j,2,k)> = n \) which produces a contradiction with (i).

**Case 2:** \( a = \text{vm}[j,2,1] \). By (i), \( S<x|Vt(j,2,1')> \neq n \) for \( i' \neq k \). Thus, \( i = k \), so \( H(xa|TC(j,1)) = H(x|TC(j,2)) \).

Therefore, \( xa \) satisfies \( \text{REQ} \).

Another important property of the specification presented is that if \( G \) is acyclic (i.e., only indirect cycles are possible), then \( P \) is an adequate generalized path. Such a property may be treated as a first step on a way to demonstrate that the specification of replication of modules we have presented does not lead to additional deadlock situations, i.e., if \( \text{NMRSYST}(G) \) is adequate (or deadlock-free) for \( n = 1 \), then \( \text{NMRSYST}(G) \) is adequate (deadlock-free) for any \( n > 1 \).

**THEOREM**

If \( G \) is acyclic, then \( P \) is an adequate generalized path.

**Proof (sketch)**

One can prove the theorem in two steps.

First, for every \( x \in \text{VFS}(P) \) there is \( z \in \text{Vev}(P)* \) such that \( xz \in \text{VFS}(P) \) and

\[(1) \quad xz|Pi \in \text{Cyc}(Pi)*\]

for every \( i \in \{1, \ldots, l\} \). One can prove this by induction on the number of nodes satisfying (i). The induction starts from nodes which have no successor nodes, and in each step one takes a node which does not satisfy (i) and all its successors (even indirect) satisfy (i). The crucial point here is that in each single path \( Vt[j,k,1] \) there is an expression \( \text{rf}[j,k,1], \text{sf}[j,k,1]* \) between \( \text{sf}[j,k,1] \) and \( \text{wn} \) which enables the flag \( \text{sf}[j,k,1] \) to be released (if set) after each history \( x \in \text{VFS}(P) \) such that \( S<x|Vt[j,k,1]> = n-1 \).

In the second step one can prove that for any \( x \in \text{VFS}(P) \) such that (i) is satisfied for all \( Pi \) and for every \( a \in \text{Ev}(P) \) there is \( z \in \text{Vev}(P)* \) such that \( xza \in \text{VFS}(P) \). This follows essentially from the acyclicity of \( G \) and the assumption that each node possesses at least one input port. Thus for each node \( Nj \) there is a gateway node \( N1 \) such that there is a directed path from \( N1 \) to \( Nj \). []
5. Implementation of NMR

The given specification of NMR is not founded on, and does not suggest, any specific implementation. However, an efficient implementation should try to be reasonably fair and should ensure that communications are not delayed unreasonably often. In particular, it appears to be crucial to ensure:

- a fair behaviour of the replicated processes;
- the required response time in case of real time control system.

In this section two possible fair implementations will be briefly presented and discussed.

The system architecture considered here consists of a set of nodes, each one consisting of a processor, private memory, and input-output devices. These nodes are connected by a communication subsystem, which may include shared memory.

The kernel code of the operating system is replicated in the private memory of each processor and each copy contains the local data structures for the dispatching of processes.

The first of the two proposed solutions is based on the notion of atomic actions [2]. The second solution utilizes a distributed algorithm for achieving cooperation between the instances of the kernel of the system nodes.

None of the proposed solution involves the synchronisation of the task replications and/or any planned scheduling.

To illustrate clearly the effectiveness of the solutions, it will be assumed that faults can only corrupt messages, but that they do not cause messages to be lost, i.e. all the copies of a message reach the receivers within a finite time.

In implementing communication, it is convenient to specify a time interval indicating the maximum delay $\triangle$ between the sending and receiving of all the copies of a message from a multiple module. In general, $\triangle$ is equal to the scheduling delay plus the communications delay.

5.1. Solution based on atomic actions

An action is a unit of work. It appears to be primitive to its surrounding environment. That is, an action appears to be atomic to other actions. Once begun, an action either completes by committing or fails by aborting. If an action aborts, it has no permanent effect on its environment. This must be supported by synchronization and recovery mechanisms that ensure that changes made by a failed action are undone and that partial results of failed actions are not incorporated in the results of other actions that eventually commit.

This solution requires the following assumption to be satisfied:

A.1 The message send operation is an atomic broadcast [3]. This means that the communications involving the replications of the same port either completes by success or fails.

No assumption is made on communications related to distinct ports.

The sending of a message to an NMR node $P$ from the modules $Q_1, \ldots, Q_n$ of $Q$ proceeds as follows.
1. The kernel of the sending module Qi records the message in the insertion buffers allocated on the nodes where the modules P1...Pn of P run. These communications are atomic.

2. After n messages have been received from the same port, the kernel related to Pi perform the voting. Before the voted message is inserted in the appropriate buffer to be processed by Pi, a mark is added. This mark (for instance a sequence number) specifies the incoming ordering.

3. If there are more than one input port containing n messages each kernel performs the voting according to a priority list referring to the senders. The priority of the senders is kept locally at each Pi. After the message from the sender Qi has been voted, the lowest value is assigned to the priority of Qi. The aim of the priority list is to guarantee the fairness of the implementation.

The non-deterministic strategy followed by the receiver processes consists of selecting the message with the smallest mark value.

The above solution satisfies the given specification. The modules of an NMR node service the input requests in an identical order.

One of the interesting features of this solution is its simplicity, which allows the introduction of modular redundancy without a significant modification of the kernel.

Additional important advantages of the proposed solution are:
- it can provide fault tolerance support for a wide class of programs;
- it allows greater asynchrony between executions of program replications;
- it allows software fault tolerance techniques, such as N-version programming [1] to be incorporated in the NMR systems.

There are no reliability problems in marking messages because marking is performed by the kernel of the node where the receiver process runs.

5.2. Using decentralized agreement

The preceding solution uses marks which imply a total order on the voted messages received from the various modules of an NMR node. In order to assure the same order for all the modules, we adopted atomic communications.

An alternative approach is possible: the kernel instances of each NMR node can agree (by a decentralized protocol) on the value of the mark which must be assigned to each received message. The kernel cooperation algorithm must be reliable, guaranteeing the agreement even in presence of faulty nodes.

A solution using an algorithm with the necessary reliability requirement, namely the signed message algorithm of interactive consistency presented in [4], is discussed in [7]. It can be shown that this solution satisfies the given specification. In this case the order among the messages of the modules P1...Pn is guaranteed by an agreement among the kernels K1...Kn on the set of messages which are to be ordered and the fact that every kernel of K1...Kn uses the same criterion for message ordering.
6. Concluding Remarks

A formal specification of NMR systems has been presented for a concurrent programming model consisting of communicating processes.

This work allows the replicated modular processes to execute with nondeterminism, and also permits software fault tolerance techniques, provided certain specified conditions are met. Instead of merely running identical copies of a process, this approach gives the opportunity to run processes having different implementations but satisfying the same specification. As shown in [1] this guards against software faults.

The concurrent model we investigated can be extended to enable the modelling of larger classes of applications. For example, it would be useful to investigate models which allow:
- nodes with several output ports;
- asynchronous communication;
- queues of voted messages.

The development of specification of more general systems is a topic for future research.

7. Acknowledgement

The authors are grateful to Prof. B. Randell for his comments and suggestions. This work was supported by the Royal Signals and Radar Establishment of the Ministry of Defence, and the Science and Engineering Research Council.
References


