Towards a Theory of Replicated Processing

Luigi V. Mancini[1,2] and Giuseppe Pappalardo[1,3]

(1) Computing Laboratory, University of Newcastle upon Tyne, UK
(2) Dipartimento di Informatica, Università di Pisa, Italy
(3) Università di Reggio Calabria, Italy

Abstract.

In the N-Modular Redundancy (NMR) approach, a computation is made reliable by executing it on several computers, and determining its results by a decision algorithm. This paper investigates a formal approach to the use of NMR in replicated distributed systems, for which it introduces a notion of correctness based on consistency with their non-replicated counterpart, and a local correctness criterion. We discuss how a replicated system component may be implemented by N base copies, a majority of which is non-faulty. The formal approach sheds light on the necessity of coordinating the copies and on the requirements they should satisfy; in particular the difficulty of replicating synchronous communication is pointed out. A practical approach is also briefly examined and shown to be consistent with the formal model.

Inside every replicated system there is a non-replicated system trying to get out.

1. Introduction

In their pursuit of fault-masking, real-time systems designers have often employed N-Modular Redundancy (NMR) for the construction of reliable software. In this approach to fault-tolerance a computation is replicated in N copies which are executed by different processors, and reliable results are determined by performing a decision algorithm, e.g. majority voting, on the N outputs obtained.

Traditionally, the *base* object to which replication is applied has been a centralized computation [LV] [AK]. More recently, however, there has been considerable interest for the replication of distributed systems composed out of communicating processes. Distribution introduces some entirely new problems into NMR. Replicating processes is not enough - communication channels have to be replicated too, for a single channel would be a bottleneck for performance and a weak link for reliability. On the other hand, replicated communication, faults of arbitrary nature, and

the nondeterminism inherent in distribution may cause copies of a process to receive input messages that are different or in a different order; yet, for non-faulty copies to remain consistent, they must process the same input sequences.

There is now a rich literature on replicated distributed systems (RDS). Techniques for their construction are proposed in [C] [G] [M] [MS] [MP1]; deadlock has been studied in [MK] [KM]. The starting point of the present paper is the definition of correct behaviour of an RDS - an issue to which insufficient attention has been paid so far. Our investigation is carried out in the formal setting provided by the CSP trace model, which is briefly presented in Section 2. In Section 3 we define an RDS to be correct with respect to, or to be a *replication* of, a base distributed system, if the behaviour of the latter may be inferred from that of the RDS (whence the aphorism that opens this paper). In Section 3 we also derive a local correctness criterion: for an RDS to be the replication of the base system it is sufficient that each replicated process in the RDS is the replication of the corresponding process in the base system. In Section 4 we adopt majority voting as a decision algorithm, and address the problem of implementing a replication P^N of a base process P by assembling N copies of P, under the usual NMR fault assumption that a majority of copies is non-faulty. We show that non-faulty copies of P should be coordinated so that they all receive every input message of which P^N has received a majority, and so that they all process the same input message sequence. These results are similar to those given in [S2] [L1] for a distributed system containing a single replicated process, which does not therefore receive replicated input like ours. In Section 5 the design approach of [M] is shown to be consistent with our formal model. Finally, by way of conclusions, we discuss whether there are limitations on the class of base distributed systems that can be replicated. In particular we point out some difficulties in replicating synchronous communication.

2. The trace model of CSP

CSP [H] is a language for the description of concurrency. Informally, a CSP process P can be regarded as a black box characterized by a set αP called the *alphabet* of P. P may engage in interaction with its environment. Atomic instances of this interaction are called *actions* and must be elements of αP.

A *trace* of a process P is a sequence of actions that P can be observed to engage in. The set of the traces of P is denoted by τP. Traces provide information about the interactions a process may accept. They are suitable for reasoning about partial correctness: assuming a process does produce some traces, these may be shown to satisfy some required property. However, albeit partially correct, a process may refuse to do anything, i.e. may deadlock. Thus, for the purpose of verifying

total correctness, the description of a process should include, together with its traces, information about its *refusals*.

As deadlock in replicated distributed systems has already been studied successfully in [KM] and [MK], this paper will concentrate on partial correctness and can therefore adopt the so-called trace model of CSP [H]. In this model a process P is identified with a pair (A,T), satisfying the laws **P1**-**P4** below. In accordance with intuition, **P3** stipulates that the empty trace $<>$ is a trace of every P, and **P4** that any prefix of a trace of P is also a trace of P.

P1 A is a non-empty, countable set; it is defined to be αP.

P2 T is a subset of A^*; it is defined to be τP.

P3 $<> \in T$.

P4 If $t \in T$ and $t1$ is a prefix of t, then $t1 \in T$.

Thus CSP language operators are defined in terms of how they return a process, i.e. an alphabet-traces pair, by combining operands, which may be processes themselves. However, for our purposes, neither the syntax nor the semantics of CSP need be described in detail. In fact, essential to our treatment are only the identification of a process with its interaction traces, and the ability to compose processes. Accordingly, these are the only aspects of CSP that will be introduced in the next two subsections.

2.1 Actions and traces

Actions and channels

Actions may be denoted simply by identifiers. However it is convenient to introduce also structured actions of the form $c!v$, where v is a value and c a *channel*; $c!v$ is said to *occur* at c and to cause v to be *exchanged*. We associate with each process P a set of channels χP, and stipulate that if $c \in \chi P$ then the action $c!v$ is in the alphabet αP. Indeed, we shall deal with processes whose alphabet contains only actions of the form $c!v$, i.e. for such a process P:

$$\alpha P = \{c!v \mid c \in \chi P\}$$

As a consequence, we shall be able to identify P with the pair $(\chi P, \tau P)$ in lieu of $(\alpha P, \tau P)$.

Traces

The following notation is similar to that of [H] (t and u range over traces):

- $<>$ is the empty trace, $<a_1,\ldots,a_n>$ is the trace whose i-th element is action a_i;
- the trace $a^\frown t$ is obtained by prefixing action a to t; the trace $t^\frown u$ by appending u to t;

- $t \leq u$ holds true if t is a prefix of u; $t \leq^n u$ holds true if t is a prefix of u and their lengths differ by n or less.

$t \lceil C$, where C is a set of channels, is the trace obtained by *projecting* t on C, i.e. by deleting from t all the actions that do not occur at channels in C. A trace can also be filtered by a predicate p containing a free variable x that ranges on trace elements [B]; $t \lceil p$ is obtained from t by deleting all the elements x for which $p(x)$ is false. E.g.:

$$<c!1,d!2,c!3,e!6> \lceil \{c,e\} = <c!1,c!3,e!6>$$

$$<c!1,d!2,c!3,e!6> \lceil \{c,e\} \lceil (x \neq 3) = <c!1,e!6>$$

2.2 Composing processes

Parallel composition

Parallel composition models synchronous communication between processes. The process $P \| Q$ is obtained by connecting processes P and Q in such a way that: (i) each of them is free to engage independently in any action that is not in the other's alphabet, but (ii) they have to engage simultaneously in all the actions that are in both their alphabets. Therefore $P \| Q$ engages in $c!v$ iff: (i) $c \in \chi P - \chi Q$ and P engages in $c!v$ or $c \in \chi Q - \chi P$ and Q engages in $c!v$, or (ii) $c \in \chi P \cap \chi Q$ and both P and Q engage in $c!v$. It follows that if a trace of $P \| Q$ is projected on χP the result should be a trace of P, and likewise for Q. This justifies the formal definition:

$$\chi(P \| Q) = \chi P \cup \chi Q$$
$$\imath(P \| Q) = \{t \mid t \lceil \chi P \in \imath P \wedge t \lceil \chi Q \in \imath Q\}$$

Parallel composition is commutative and associative.

Concealment

Let P be a process and C a set of channels; then $P \backslash C$ is a process that behaves like P with the actions occurring at C made invisible. Formally:

$$\chi(P \backslash C) = \chi P - C$$
$$\imath P \backslash C = \{t \lceil (\chi P - C) \mid t \in \imath P\}$$

Concealment is associative in that $(P \backslash C) \backslash D = P \backslash (C \cup D)$. It is often used to conceal the interaction that takes place at the channels shared by processes combined by $\|$. It should be noted that in a more general scope than that of this paper the introduction of concealment would suggest the adoption of a failure-based model [H].

2.3 Specifications for CSP

It is possible to introduce a logic language having the set of traces as its intended model, and regard its formulae as specifications of CSP processes. Let ψ be a formula containing the variable t free; then we say that a process P satisfies the specification ψ, and write

$$P \textbf{ sat } \psi$$

if and only if ($\Rightarrow$ is used to denote logical implication):

$$\forall t.\ t \in traces(P) \Rightarrow \psi(t)$$

3. Modelling replicated distributed systems

3.1 The notion of N-replication

This section deals with the formalization of the notion of *N-replication* ($N>1$). Intuitively the N-replication of a *base* process P is a *replicated* process P^N which has N copies of each channel of P (Fig. 3.1), and behaves consistently with P (in a sense to be made precise later). The requirement on the channels may be formalized by:

R1 $\quad \chi P^N = \chi P \times \{1,...,N\}$

As a notational convenience $(c,n) \in \chi P^N$ is written $c[n]$; $c[n]$ is also termed a replicated channel or a *version* or *copy* of the *base* channel c. We shall also write c^N for $\{c[1],...,c[N]\}$ and C^N, where $C=\{c,d,...\}$, for $c^N \cup d^N \cup ...$

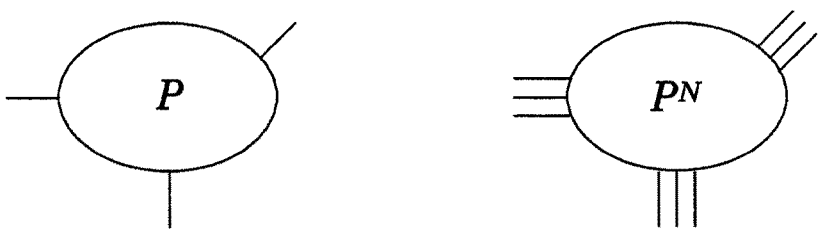

Fig. 3.1 - A base system P and its N-replication P^N

We now consider the notion of consistency between P^N and P. Ideally, P^N should behave as though it contained N copies of P, so that each trace of P^N should be one of P with the generic action $c!v$ replaced by a permutation of $\{c[1]!v,...,c[N]!v\}$. In general, however, this may be impossible to obtain in the presence of faults, the tolerance of which on the other hand is exactly the motivation

for replicating P. Thus we adopt a less demanding view: if P^N is an N-replication of P and t^N is a trace of P^N, it should be possible to extract from t^N a trace of P. Thus we assume the existence of a relation *extract* such that:

R2 $t^N \in \tau P^N \wedge extract(t^N,t) \Rightarrow t \in \tau P$

and

E1 $\forall t^N.\ \exists t.\ extract(t^N,t)$

A practical way of implementing an extraction function is to add identifiers to messages, as we shall call values exchanged by processes. In particular, messages exchanged by a base process via a given channel are assumed to carry distinct identifiers. So we assume that there exist two sets *Msgs* and *Idents*, and a function *id*: *Msgs*→*Idents*, such that:

R3 $aP = \{c!m \mid c \in \chi P \wedge m \in Msgs\}$
$aP^N = \{c[n]!m \mid c \in \chi P \wedge 1 \le n \le N \wedge m \in Msgs\}$
$t \in \tau P \wedge t = <...c!m1,...,c!m2,...> \Rightarrow id(m1) \neq id(m2)$

Consider now the trace $t^N \lceil c^N \lceil (id(x)=i)$ obtained from a trace t^N of P^N by including only those actions that occur at a copy of the base channel c and exchange a message of identifier i. We assume that if t is extracted from t^N, then each action $c!m$ of t is *elected* from the trace $t^N \lceil c^N \lceil (id(x)=i)$, where $i = id(m)$. For this purpose we assume the existence of a function *elect* such that:

E2 $elect(t^N \lceil c^N \lceil (id(x)=i)) = c!m \ \wedge \ id(m)=i$
$\vee\ elect(t^N \lceil c^N \lceil (id(x)=i)) = NIL$

E3 $extract(t^N,t) \wedge e = elect(t^N \lceil c^N \lceil (id(x)=i)) \Rightarrow$
if $e = NIL$ **then** $t \lceil c \lceil (id(x)=i) = <>$
else $t \lceil c \lceil (id(x)=i) = <e> \vee t \lceil c \lceil (id(x)=i) = <>$

The second disjunct of **E3**'s **else** branch implies that an elected message need not be an element of the extracted trace (although, by the first disjunct, the converse must hold). The reason for this will be made clear in Section 4.1.

It seems reasonable to require that extraction should be context-independent, in the sense that if t is extracted from t^N, the contribution of $t^N \lceil C^N$ to t should be independent of the actions deleted from t^N by $\lceil C^N$.

E4 $extract(t^N,t) \Rightarrow extract(t^N \lceil C^N, t \lceil C)$

E5 $extract(s^N \lceil C^N, t) \Rightarrow \exists u.\ (u \lceil C = t \wedge extract(s^N,u))$

At this point it may be helpful to summarize the definition of N-replication. Given a relation *extract* and a function *elect* that satisfy **E1**-**E5**, P^N is said to be an N-replication of P with respect to *elect* and *extract* if and only if **R1**-**R3** hold true.

An important consequence of **R2** is that if the base system satisfies a specification, so does the N-replication once its traces have been filtered by *extract*. Hence the **sat** inference rule:

$$\begin{array}{l}\text{If } P \textbf{ sat } \psi \\ \text{then } P^N \textbf{ sat } \forall u.(extract(t^N,u)) \Rightarrow \psi(u)\end{array}$$

In view of this result, we shall say that P^N is *correct* with respect to P if P^N is an N-replication of P. Since this definition abstracts from the internal structure of P and P^N, it may be viewed as a global correctness criterion.

3.2 A local correctness criterion for replicated distributed systems

The property of being an N-replication of a base process distributes (in the algebraic sense) through parallel composition and concealment. Before proving these results as Theorems 3.1 and 3.2, it is interesting to discuss their implications.

A *distributed system* is defined as a set of *component* processes connected at their shared channels, which may or may not be concealed to the environment. Thus, in the formalism of CSP a distributed system DS takes the form:

$$DS = (P \parallel Q \parallel \ldots)\backslash C$$

A *replicated distributed system* DS^N is obtained from the distributed system DS by replacing each component process $P, Q,\ldots$ by an N-replication $P^N,Q^N,\ldots$ and concealing the channel set C^N:

$$DS^N = (P^N \parallel Q^N \parallel \ldots)\backslash C^N$$

The following theorems ensure that DS^N is effectively an N-replication of DS, i.e. is correct with respect to DS. This yields a local correctness criterion for distributed systems: DS^N is correct if its components P^N, Q^N,... are correct.

Theorem 3.1 If P^N and Q^N are N-replications of P and Q respectively, then $P^N \| Q^N$ is an N-replication of $P \| Q$.

Proof. Showing that **R1** and **R3** hold valid is routine. To verify **R2** we must show that

$$t^N \in \tau(P^N \| Q^N) \wedge extract(t^N,t) \Rightarrow t \in \tau(P \| Q).$$

The assumption $t^N \in \tau(P^N \| Q^N)$ implies, by the definition of parallel composition, that $t^N \lceil \chi P^N \in \tau P^N$ and $t^N \lceil \chi Q^N \in \tau Q^N$. By **E4**, from $extract(t^N,t)$ we may infer

$$extract(t^N \lceil \chi P^N, t \lceil \chi P) \quad \text{and} \quad extract(t^N \lceil \chi Q^N, t \lceil \chi Q)$$

which, owing to the hypothesis that P^N and Q^N are N-replications of P and Q respectively, imply that $t\lceil\chi P \in \iota P$ and $t\lceil\chi Q \in \iota Q$. Hence, by the definition of parallel composition, $t \in \iota(P \| Q)$. □

Theorem 3.2 If P^N is an N-replication of P, then $P^N \backslash C^N$ is an N-replication of $P \backslash C$.

Proof. The proof that **R1** and **R3** are valid under the proviso is routine. To verify **R2** we must show that

$$t^N \in \iota(P^N \backslash C^N) \wedge extract(t^N, t) \Rightarrow t \in \iota(P \backslash C)$$

The assumption $t^N \in \iota(P^N \backslash C^N)$ implies, by the definition of concealment:

(*) $\exists s^N.\ t^N = s^N \lceil (\chi P^N - C^N) \wedge s^N \in \iota P^N$

Hence, the assumption $extract(t^N, t)$ may be rewritten as $extract(s^N \lceil (\chi P^N - C^N), t)$. Thus by **E5**:

(**) $\exists u.\ u \lceil (\chi P - C) = t \wedge extract(s^N, u)$

From (*) and (**) we infer $s^N \in \iota P^N$ and $extract(s^N, u)$, which, owing to the hypothesis that P^N is an N-replication of P, imply $u \in \iota P$. So, since $u \lceil (\chi P - C) = t$ by (**), the definition of concealment implies $t \in \iota(P \backslash C)$. □

3.3 I/O distributed systems

The notion of *I/O process* generalizes that of pipe [H]. P is termed an I/O process if its channel set χP can be partitioned into a set of input channels ιP and one of output channels ωP, and for some f:

PS P **sat** $t \lceil \omega P \leq f(t \lceil \iota P)$

i.e. the output message sequence is a prefix of a function of the input message sequence. Note that P may be an I/O process by conforming to a stronger specification, e.g.

$$P \textbf{ sat } t \lceil \omega P \leq^1 f(t \lceil \iota P)$$

which implies **PS**; however the unavoidable delays between input and output entail the fact that

$$P \textbf{ sat } t \lceil \omega P = f(t \lceil \iota P)$$

is impossible unless $\iota P = <>$. An example of an I/O process is provided by a buffer, i.e. a process B that satisfies:

$$\iota B = \{in\} \qquad \omega B = \{out\} \qquad B \textbf{ sat } t \lceil out \leq (t \lceil in)[out/in]$$

where $[out/in]$ denotes renaming of channel *out* to *in*.

$DS = (P \| Q \| \ldots) \backslash C$ is termed an *I/O distributed system* if (i) the components $P, Q, \ldots$ are I/O processes, and (ii) whenever a channel c is shared among a subset D_c of components it is an output for only one of them; thus we can model both unicasting ($|D_c| = 2$: c is shared by two components only) and multicasting ($|D_c| > 2$: c is shared by more than two components). In the sequel we restrict our attention to replicated I/O distributed systems; this is not essential to ensure that replication is

possible, but makes clearer the intuition underlying the formal treatment, by ruling out the complex multi-party interactions permitted in CSP.

4. Implementing replication

Section 3 has shown that an I/O distributed system may be replicated by replicating its component I/O processes. This section addresses the problem of assembling N copies of a base I/O process P to implement an N-replication P^N of P despite given fault assumptions.

An important issue is determining the class of I/O processes for which an N-replication may be implemented from N base copies. Rather than indicating this class a priori, we prefer to let it emerge during the formal treatment.

4.1 Extraction by majority voting

Another important issue in the design of an N-replication P^N of a base process P is the interplay between the fault assumptions and the extraction strategy in terms of which P^N replicates P. However we shall restrict our treatment to the extraction which is nearly always used in practice: *majority voting*; the design goal then becomes the classical NMR one: to tolerate a minority of faulty copies. We use $\lfloor N \rfloor$ to denote the majority out of N votes, i.e. we let:

$$\lfloor 2p+1 \rfloor = \lfloor 2p \rfloor = p+1$$

Majority voting consists, in our approach, in defining the function *elect* as follows. To compute

$$elect(<c[n_1]!m_1,...c[n_K]!m_K>)$$

(where the messages m_k have all the same identifier) the trace argument is scanned from head to tail; the result is $c!m$ if m is the first value to occur $\lfloor N \rfloor$ times, *NIL* if there is no such m.

We further assume that the set *Idents* of message identifiers is totally ordered, and that the projection of an extracted trace on a given channel is ordered by message identifiers, without gaps:

E6 $extract(t^N,t) \Rightarrow no\_gaps(t)$ where $no\_gaps(t) \Leftrightarrow \forall c.\ no\_gap(t\lceil c)$

$no\_gap(u)$ is defined to be true if whenever $u = <...c!m1,c!m2,...>$, $id(m2)$ is the immediate successor of $id(m1)$ in the ordering defined over *Idents*. Finally, we require that all extracted traces should be maximal, in the sense that if $c!m$ can be elected it should be included in the extracted trace, provided that **E6** is not thereby violated. It follows that :

E7 $extract(t^N,t) \wedge extract(t^N,t1) \Rightarrow \forall c.\ t\lceil c = t1\lceil c$

We conclude this section by an example in which **E3**'s second disjunct applies, i.e. in which an elected message does not enter in the extracted trace:

$$extract(t^N,t) \wedge elect(t^N \lceil c^N \lceil (id(x)=i)) \neq NIL \wedge t \lceil c \lceil (id(x)=i) = <>$$

Assume that $N=3$ so that $\lfloor N/2 \rfloor = 2$, that m^α denotes a message of identifier $id(m^\alpha)=\alpha$, that $t^N = <c[1]!m^\alpha, c[2]!m^\alpha, c[1]!m^\beta, c[2]!m^\beta, c[1]!m^\gamma, c[1]!m^\delta, c[3]!m^\delta>$, and that $\alpha<\beta<\gamma<\delta$; then:

$$elect(t^N \lceil c^N \lceil (id(x)=\alpha)) = c!m^\alpha \qquad elect(t^N \lceil c^N \lceil (id(x)=\beta)) = c!m^\beta$$
$$elect(t^N \lceil c^N \lceil (id(x)=\gamma)) = NIL \qquad elect(t^N \lceil c^N \lceil (id(x)=\delta)) = c!m^\delta$$

Thus, if t and t^N are such that $extract(t^N,t)$ holds, then t cannot contain a message of identifier γ, for $elect(t^N \lceil c^N \lceil (id(x)=\gamma)) = NIL$; therefore t cannot contain the elected $c!m^\delta$ either, or otherwise *no_gaps* would be violated; formally:

$$t \lceil c \lceil (id(x)=\delta) = <>$$

4.2 The implementation

In this section we show how to build an N-replication P^N of an I/O process P by feeding N copies of P through a *coordinator COORD*.

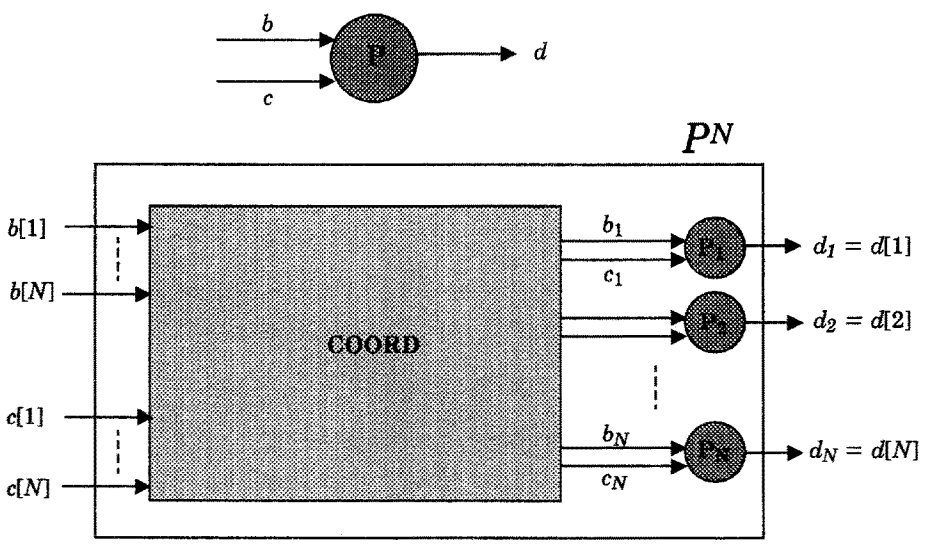

Fig. 4.1 - An architecture for a N-replication of the base process P

The architecture of P^N is shown in Fig. 4.1. The inputs to *COORD* coincide with the inputs to P^N:

$$\iota COORD = \iota P^N = \{c[1],...,c[N] \mid c \in \iota P\}$$

The N copies of P are denoted as $P_1,...,P_N$; accordingly, the channels of P_n $(1 \leq n \leq N)$ are named by subscripting the channels of P:

$$\iota P_n = \{c_n \mid c \in \iota P\} \qquad \omega P_n = \{d_n \mid d \in \omega P\} \qquad \text{for } 1 \leq n \leq N$$

Since the outputs of P^N are the union of those of $P_1,...,P_N$, we assume that if d is an output channel of P, then the outputs d_n of P_n and $d[n]$ of P^N coincide:

$$d_n = d[n] \quad \text{for all } d \in \omega P \text{ and } 1 \leq n \leq N$$

The copies $P_1,...,P_N$ are fed by the coordinator, so the outputs of the latter are divided into N groups of $|\iota P|$ channels, and each group coincides with the input channel set of some P_n. Finally, we introduce a useful convention: t and $t1$ range over the traces of P, t_n ranges over the traces of P_n and t^N over those of P^N. We shall often have to rename a trace t_n of P_n as a trace of P; this is accomplished by the substitution $t_n[\chi P/\chi P_n]$.

We now define the behaviour of the coordinator and of the copies of P. It is assumed that a predicate *non-faulty*(n) holds true for n if both P_n and its input from the coordinator are non-faulty. The coordinator specification is:

CS $\quad COORD$ **sat** $\quad \exists j.\ extract(u \lceil \iota COORD, j)$

$$\wedge \ (\forall n.\ non\text{-}faulty(n) \Rightarrow (u \lceil \iota P_n)[\chi P/\chi P_n] \leq j)$$

CS imposes two requirements on the generic trace u of $COORD$: (i) the non-faulty outputs of $COORD$ (viz. $u \lceil \iota P_n$) must be, up to channel renaming, prefixes of the same trace j, which (ii) must be extracted from the replicated input to $COORD$ (viz. $u \lceil \iota COORD$). Often, the former requirement has been assumed to be sufficient for replicated systems correctness; in fact it only guarantees that the states of the copies of P remain consistent, but the latter requirement is necessary as well for the input/output behaviour of P^N to be correct.

Non-faulty copies of the I/O process P behave like P up to channel renaming:

PB $\quad \forall n.\ non\text{-}faulty(n) \Rightarrow (t_n \in \iota P_n \Leftrightarrow t_n[\chi P/\chi P_n] \in \iota P)$

Thus the specification **PS** of Section 3.3 can be adapted to give, for all n:

P$_n$S $\quad P_n$ **sat** $\quad non\text{-}faulty(n) \Rightarrow (t_n \lceil \omega P_n)[\chi P/\chi P_n] \leq f((t_n \lceil \iota P_n)[\chi P/\chi P_n])$

We begin the proof that P^N is an N-replication of P by showing that under the additional assumption that f is monotonic, i.e.

A1 $\quad x \leq y \Rightarrow f(x) \leq f(y)$

it turns out that the outputs from the non-faulty copies of P are all prefixes of the same trace. As explained in Section 6, monotonicity can be interpreted as meaning that P is deterministic.

Lemma 4.1 If $t^N \in \tau P^N$, there exists a j such that if *non-faulty*(n) then $t^N \lceil \omega P_n[\chi P/\chi P_n] \leq f(j)$.

Proof. If $t^N \in \tau P^N$, the structure of P^N implies that there exist a trace $u \in \tau COORD$ and, for each n, a trace $t_n \in \tau P_n$ such that:

(*) $\quad u \lceil \iota P_n = t_n \lceil \iota P_n \qquad t_n \lceil \omega P_n = t^N \lceil \omega P_n$

We may therefore apply **CS** to u and infer that there exists a j such that, if *non-faulty*(n),

$$(u \lceil \iota P_n)[\chi P/\chi P_n] \leq j$$

whence, by the first equality (*):

$$(t_n \lceil \iota P_n)[\chi P/\chi P_n] \leq j$$

Since f is monotonic, we may apply it to both sides, and use $\mathbf{P_nS}$, transitivity and the second equality (*) to infer:

$$(t^N \lceil \omega P_n)[\chi P/\chi P_n] \leq f(j)$$ □

Assume that a majority of the traces $t^N \lceil \omega P_n$ $(1 \leq n \leq N)$ that make up $t^N \lceil \omega P^N$ are prefixes of a trace satisfying *no_gaps*; then it is possible to show that there exists a k such that, up to renaming, the projection on an output channel d of any trace extracted from $t^N \lceil \omega P^N$ is a prefix of the projection on $d[k]$ of t^N.

Lemma 4.2 If *extract*$(t^N \lceil \omega P^N, v_{ext})$, and there exists a v_{max} such that *no_gaps*(v_{max}) and the set

$$S = \{n \mid (t^N \lceil \omega P_n)[\chi P/\chi P_n] \leq v_{max}\}$$

has cardinality $|S| \geq \lfloor N \rfloor$, then for some $k \in S$:

$$\forall d.\ d \in \omega P \Rightarrow v_{ext} \lceil d \leq (t^N \lceil d[k])[\chi P/\chi P_n]$$

Proof. Let k be the index in $S = \{n \mid (t^N \lceil \omega P_n)[\chi P/\chi P_n] \leq v_{max}\}$ for which the trace $t^N \lceil \omega P_k[\chi P/\chi P_k]$ is the longest. If an action $d!m$ is in v_{ext}, by **E3** $d!m$ must have been elected, so the set

$$S_m = \{n \mid d[n]!m \text{ is in } t^N \lceil \omega P_n\}$$

must have cardinality $|S_m| \geq \lfloor N \rfloor$. Thus the hypothesis $|S| \geq \lfloor N \rfloor$ implies $S \cap S_m \neq \{\}$. It follows that $k \in S_m$ must hold, or if $h \in S \cap S_m$ the trace $(t^N \lceil \omega P_k)[\chi P/\chi P_k]$ could not be longer than $(t^N \lceil \omega P_h)[\chi P/\chi P_h]$. Thus the elements of v_{ext} must also be in $(t^N \lceil \omega P_k)[\chi P/\chi P_k]$, and, as a result, the elements of $v_{ext} \lceil d$ must also be in $(t^N \lceil d[k])[\chi P/\chi P_k]$ for $d \in \omega P$. That $v_{ext} \lceil d \leq (t^N \lceil d[k])[\chi P/\chi P_k]$ follows from the fact that both v_{ext} and $t^N \lceil \omega P_k$ satisfy *no_gaps*: v_{ext} by **E6** and $t^N \lceil \omega P_k$ by the hypothesis on v_{max}. □

Lemma 4.2 could be coupled in the obvious way to Lemma 4.1, if $f(j)$ returned a trace satisfying *no_gaps*. So we make the assumption:

A2 $\quad \forall j.\ no\_gaps(f(j))$

This allows us to prove:

Lemma 4.3 If $t^N \in \iota P^N$ and $|\{n \mid \textit{non-faulty}(n)\}| \geq \lfloor N \rfloor$, there exist a $t1 \in \iota P$, a j and a v such that:

$$\textit{extract}(t^N \lceil \iota P^N, j) \wedge \textit{extract}(t^N \lceil \omega P^N, v)$$
$$\wedge \quad t1 \lceil \iota P \leq j \wedge \forall d.\, d \in \omega P \Rightarrow v \lceil d \leq t1 \lceil d$$

Proof. Owing to the structure of P^N, if $t^N \in \iota P^N$ there exist $u \in \iota COORD$ and, for each n, $t_n \in \iota P_n$ such that:

(*) $\quad t^N \lceil \iota P^N = u \lceil \iota COORD \qquad u \lceil \iota P_n = t_n \lceil \iota P_n \qquad \forall d.\, d \in \omega P \Rightarrow t_n \lceil d_n = t^N \lceil d[n]$

Thus **CS** ensures that there exists a j such that $\textit{extract}(t^N \lceil \iota P^N, j)$ and, if n is non-faulty,

(**) $\quad (u \lceil \iota P_n)[\chi P / \chi P_n] = (t_n \lceil \iota P_n)[\chi P / \chi P_n] \leq j$

By **E1**, Section 3.1, there must exist a v such that $\textit{extract}(t^N \lceil \omega P^N, v)$. By Lemma 4.1, all non faulty copies, which are a majority by hypothesis, issue an output trace which is a prefix of $f(j)$, and it has been assumed that $\textit{no\_gaps}(f(j))$. Hence, letting $v_{max} = f(j)$ and $v_{ext} = v$, we may invoke Lemma 4.2 to infer that there exists a k such that $\textit{non-faulty}(k)$ and:

$$\forall d.\, d \in \omega P \Rightarrow v \lceil d \leq (t^N \lceil d[k])[\chi P / \chi P_k]$$

whence, by applying the last formula of (*):

(***) $\quad \forall d.\, d \in \omega P \Rightarrow v \lceil d \leq (t_k \lceil d_k)[\chi P / \chi P_k]$

Finally, as $t_k \in \iota P_k$ and k is non-faulty, **PB** implies that $t1 = t_k[\chi P / \chi P_k]$ is a trace of ιP. Thus (**) (taking $n = k$) and (***) may be rewritten as:

$$t1 \lceil \iota P \leq j \qquad \text{and} \qquad \forall d.\, d \in \omega P \Rightarrow v \lceil d \leq t1 \lceil d$$ □

We need one more assumption (whose intuitive meaning will be explained in Section 6) concerning the traces of the I/O process P:

A3 $\quad (t1 \in \iota P \wedge \forall c.\, (c \in \iota P \Rightarrow t1 \lceil c \leq t \lceil c) \wedge \forall d.\, (d \in \omega P \Rightarrow t \lceil d \leq t1 \lceil d)) \Rightarrow t \in \iota P$

We are now sufficiently well equipped to prove the main result of this section.

Theorem 4.4 If a majority of copies of P are non-faulty, then P^N is an N-replication of P.

Proof. We need to show that **R1**, **R2** and **R3** (Section 3.1) hold. **R1** and **R3** are trivial to prove, so we concentrate on **R2**, viz.:

$$t^N \in \iota P^N \wedge \textit{extract}(t^N, t) \Rightarrow t \in \iota P.$$

Assume $t^N \in \iota P^N$ and $\textit{extract}(t^N, t)$. By **E4**, Section 3.1:

(*) $\quad \textit{extract}(t^N \lceil \iota P^N, t \lceil \iota P) \quad \text{and} \quad \textit{extract}(t^N \lceil \omega P^N, t \lceil \omega P)$

By Lemma 4.3, there exist j and v such that:

$$\textit{extract}(t^N \lceil \iota P^N, j) \quad \text{and} \quad \textit{extract}(t^N \lceil \omega P^N, v)$$

whence, by (*) and **E7**, Section 4.1:

(**) $\quad \forall c.\, c \in \iota P \Rightarrow t \lceil c = j \lceil c \quad \text{and} \quad \forall d.\, d \in \omega P \Rightarrow t \lceil d = v \lceil d$

Lemma 4.3 also guarantees the existence of $t1 \in \iota P$ such that:

$$t1 \lceil \iota P \leq j \qquad \text{and} \qquad \forall d.\, d \in \omega P \Rightarrow v \lceil d \leq t1 \lceil d$$

Thus, by (**) and transitivity:

$\forall c.\ c \in \iota P \Rightarrow t1 \lceil c \leq t \lceil c$ and $\forall d.\ d \in \omega P \Rightarrow t \lceil d \leq t1 \lceil d$

whence , by **A3,** we may conclude that $t \in \tau P$. □

5. An example: the Join algorithm

In this section we consider an example implementation of replication. Fig. 5.1, which should be

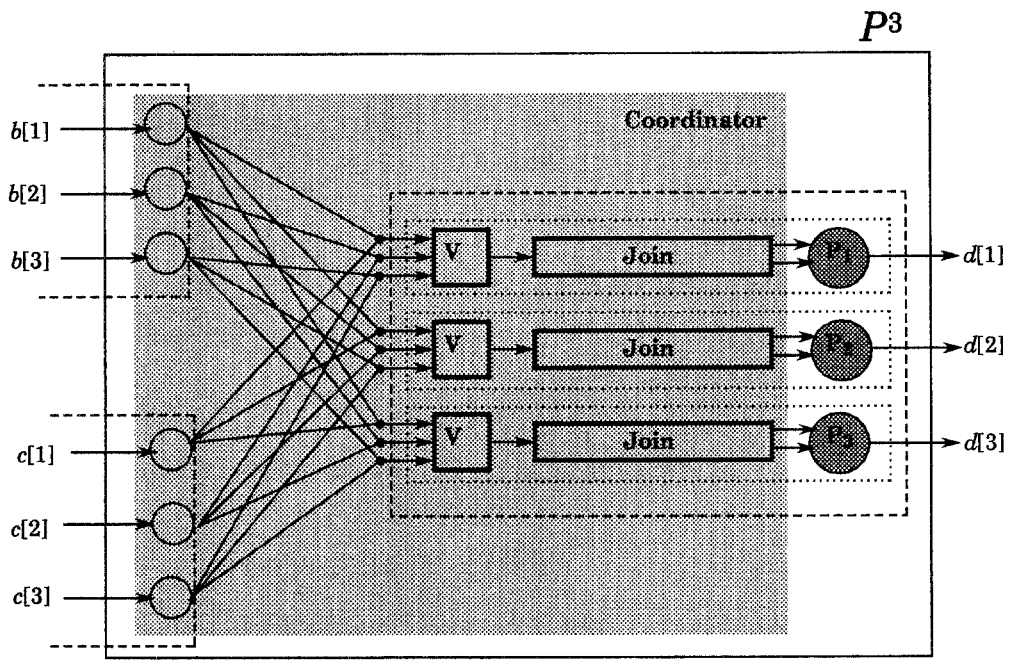

Fig. 5.1. The 3-replication of P according to *Join*

compared to Fig 4.1, shows the coordinator proposed in [M]. This coordinator is composed out of $N^*|\iota P|$ distributors and N modules. Each distributor is fed through a replicated input channel and is connected via a link to every module. The modules contain a majority voter **V** and a *Join* unit. It is assumed that distributors and links may fail in any fashion that does not violate the property **DELIVER**: if a message enters the coordinator in a majority of copies, it will be delivered to every voter in a majority of copies. Periodically, the *Join* units perform a byzantine agreement [LSP] to agree on the voted messages that will be passed to the copies of the base process P. In [MP2,MP3] we give a formal proof that the coordinator satisfies the specification **CS** of Section 4.2 regardless of how many modules are faulty.

In a real implementation, the components inside the dashed borders in Fig. 5.1 are located at the same site, and those inside the dotted borders probably run on the same processor. If links are conventionally attributed to the distributors' site, the assumption that a minority of processors per site may fail ensures that (i) **DELIVER** is respected and (ii) non-faulty outputs of the coordinator feed non-faulty copies of P and must be a majority. As stated above, (i) guarantees that the coordinator satisfies its specification; therefore (ii), as shown in Section 4.2, guarantees that P^N implements an N-replication of P.

6. Concluding remarks

By way of conclusions we discuss the properties **A1-A3** introduced in Section 4.2. There we have presented a technique by which an I/O process P that satisfies:

$$P \textbf{ sat } t\restriction\omega P \leq f(t\restriction\iota P)$$

can be replicated provided that:

A1 f is monotonic, i.e. $x \leq y \Rightarrow f(x) \leq f(y)$;

A2 (messages in a trace carry identifiers and) for all traces j and channels c the identifiers of $f(j)\restriction c$ are ordered without gaps;

A3 $(t1 \in \tau P \wedge \forall c.\,(c \in \iota P \Rightarrow t1\restriction c \leq t\restriction c) \wedge \forall d.\,(d \in \omega P \Rightarrow t\restriction d \leq t1\restriction d)) \Rightarrow t \in \tau P$.

The intended meaning of **A1** is that the base system P should be deterministic - a term which is unfortunately quite overloaded (see e.g. [H]). In this context it is employed to mean that in any state P may offer to engage in at most one output action (this property may be shown to be equivalent to f being monotonic). Indeed, if different non-faulty copies of P can transform the same input into different outputs, it is hard to see how to ensure the correctness of P^N. We deem therefore that **A1** is a requirement of a fundamental nature, though it may take different forms in different treatments. E.g., Cooper reached similar conclusions in his design and implementation of a distributed replicated system [C], (although he also allows for relaxing determinism at the cost of transparency).

A2 is a requirement which most real systems satisfy by using sequence numbers for messages. Sequence numbers are unnecessary if physical channels do not reorder input messages; in this case, however, messages should be considered to be implicitly numbered in accordance with the arrival order. (As an aside, note that CSP channels are synchronous, so that physical channels that may reorder messages or be otherwise faulty have to be modelled inside the coordinator.) Other systems make use of increasing timestamps [L1] [L2]. There will in general be gaps between the timestamps of consecutive messages; however, since the fulfillment of the input

stability condition [S1] ensures that messages are processed in the order they were sent, we may still consider messages to be implicitly numbered. Thus timestamp-based systems too may be regarded - at a convenient abstraction level - as satisfying **A2**.

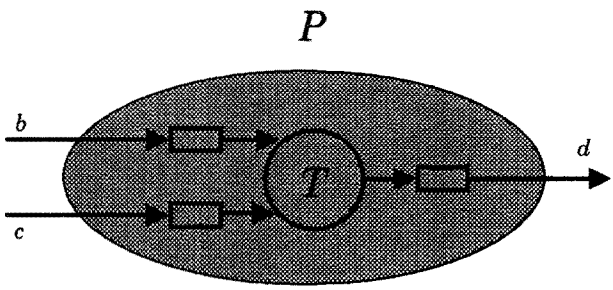

Fig. 6.1. A process satisfying **A3**

A3 implies that P behaves as though it contained a processing I/O element T communicating with the environment via unbounded buffers (Fig. 6.1). Indeed if $t1$ is known to be a trace of P and t is obtained from $t1$ as specified in **A3**, then t must also be a trace of P, for the added input messages and the missing output messages may be lingering inside the queues. Intuitively, the need for **A3** may be understood as follows: as the coordinator cannot help introducing some buffering, its effect can be ignored only if communication between components of the distributed system takes place through unbounded buffers anyway. This may be seen as a limitation on the class of distributed systems that can be implemented by replication. This rather strong view may be perhaps mitigated by weakening the notion of implementation, so as to allow a target distributed system with bounded buffers to be implemented by one with the same components and larger buffers. Either view should be reflected in extant NMR designs by a corresponding assumption, which is instead generally omitted.

At this point the reader may wonder how fundamentally **A3** limits the class of distributed systems that can be implemented - in a strict sense - by replication. Actually, two solutions do come to mind to replicate the synchronous, unbuffered communication advocated in [H]: (1) timing message exchanges at regular intervals, large enough to guarantee that all the N-replications involved are ready; (2) introducing some form of replicated, reliable acknowledgements. None of these however seems particularly appealing. Rather than embedding synchronous message passing in replication mechanisms, it is probably best to implement it on top of the abstraction of reliable buffered processes offered by the kind of replication discussed here.

Acknowledgements

The authors are grateful to Prof. B. Randell and Prof. S.K. Shrivastava for their comments and suggestions. This work was supported by the Royal Signals and Radar Establishment of the U.K. Ministry of Defence.

References

[AK] Avizienis, A., Kelly, J.K.J., "Fault tolerance by design diversity: concepts and experiments", *IEEE Computer*, vol. 17, no. 8, pp. 67-80, Aug. 1984.

[B] Bird, R. S., "The promotion and accumulation strategies in transformational programming", *ACM Transactions on Programming Languages and Systems*, vol. 6, no. 4, Oct. 1984.

[C] Cooper, E, "Replicated distributed programs", *Proc. of the 10th ACM Sym. on Operating Systems Principles*, pp. 63-78, Washington, Dic. 1985.

[G] Goldberg, J., "SIFT: A provable fault-tolerant computer for aircraft flight control", *Inform. Processing 80 Proc. IFIP Congr.*, pp. 151-156, Tokyo, Japan, Oct. 1980.

[H] Hoare, C.A.R., "Communicating sequential processes", *Prentice Hall International*, 1985.

[KM] Koutny, M., and Mancini, L., "Synchronizing events in replicated computations", Technical Report TR/237, Computing Laboratory, University of Newcastle upon Tyne, June 1987 (to appear in *The Journal of Systems and Software*).

[L1] Lamport, L., "The implementation of reliable distributed multiprocess sustems", *Computer Networks*, pp. 95-114, vol. 2, no. 2, May 1978.

[L2] Lamport, L., "Time, clocks and the ordering of events in a distributed system", *Comm. ACM*, vol. 21, no. 7, pp. 558-565, July 1978.

[LSP] Lamport, L., Shostak, R., Pease, M., "The Byzantine Generals problem", *ACM Transactions on Programming Languages and Systems*, pp. 382-401, vol. 4, no. 3, July 1982.

[LV] Lyons, R.E., Vanderkulk, W., "The use of triple-modular redundancy to improve computer reliability", *IBM Journal of Research and Development*, pp. 200-209, vol. 6, no. 2, Apr. 1962.

[M] Mancini, L., "Modular redundancy in a message passing system", *IEEE Trans. Software Eng.*, pp. 79-86, vol. SE-12, no. 1, Jan. 1986.

[MK] Mancini, L., Koutny, M., "Formal specification of N-modular redundancy", *1986 ACM Computer Science Conference*, pp. 199-204, Cincinnati, Ohio, Feb. 1986.

[MP1] Mancini, L., Pappalardo, G., "The *Join* algorithm: ordering messages in replicated systems", *Safecomp '86*, pp. 51-55, Sarlat, France, Oct. 1986.

[MP2] Mancini, L., Pappalardo G., "On resolving nondeterminism in replicated distributed systems", *IFIP Conf. on Distributed Processing*, Amsterdam, The Netherlands, Oct. 1987.

[MP3] Mancini, L., Pappalardo G., "Proving correctness properties of a replicated synchronous program", to appear in *The Computer Journal*.

[MS] Mancini, L., Shrivastava, S.K., "Exception handling in replicated systems with voting", *16th Int. Conf. on Fault Tolerant Computing*, pp. 384-389, Vienna, Austria, July 1986.

[MSS] Melliar-Smith, P.M., Schwartz, R., "Formal specification and mechanical verification of SIFT: a fault-tolerant flight control system", *IEEE Trans. on Computers*, vol. C-31, no. 7, pp. 616-630, July 1982.

[S1] Schneider, F.B., "Synchronization in distributed programs", *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 2, pp. 125-148, Apr. 1982.

[S2] Schneider, F.B., "The state machine approach", in Paul, M., and Siegert, H.J. (eds.), *Distributed systems - methods and tools for specification, an advanced course*, LNCS vol. 190, pp. 444-454, Springer-Verlag, 1985.