International Journal of Soft Computing 1(4): 247-254, 2006


# An Efficient and Reliable Parameterizable Recovery Approach for Distributed Applications

M. Aliouat, M. Nekkache and Z. Aliouat
Department of Computing Science Ferhat Abbes University,
Setif Route de Bejaia Sétif 19000 Algeria

**Abstract:** Designing fault tolerant distributed systems, particularly when taking into account both hardware and software faults, is a challenge requiring extra endeavor. In this effect, backward error recovery approach is enough general to be used to recover a failing distributed application. In this study, we have to deal with the fastidious problem of interprocess rollbacks propagation during recovery operation. The control of this undesirable situation, so called domino effect problem, with its qualitative and quantitative aspects, has been well addressed in the past and has led to many proposal schemes for its total prevention or minimization. Among these schemes, we focus attention to Randell's conversation and Kim's Programmer Transparent Coordination of recovering concurrent processes (PTC). In this study, we present a variant of PTC approach aiming to make recovery operation more trustworthy and efficient. This improvement is reached towards new concepts namely: S-propagation, accusation limit, definite invocation and 2 by 2 reliability. The definite invocation and 2 by 2 reliability are used to avoid the domino-effect qualitative aspect, while the Accusation limit should prevent the domino effect quantitative aspect and should ensure also that the S-propagation will be stopped at a satisfactory reliability level. Therefore, we deduce a parameter noted Accusation limit Level which controls the maximum number of accused recovery regions per any process involved in rollback. Our variant is as flexible and efficient as PTC but more reliable. Furthermore, the computation validity obtained through the acceptance tests, is ensured by a number of tests greater than in the conversation scheme.

**Key words:** Fault tolerant distributed systems, trustworthy and efficient recovery, domino effect

## INTRODUCTION

Due to the constant increase of complexity in software applications design and to the important growth in integration size of hardware components and the size of high performance computers from hundreds to tens of thousands of processors[1], the mean time to failure in these situations is becoming significantly shorter than the execution time of many current high performance computing application. Then the rate of fault occurrences (hardware transient faults and software faults) becomes more and more worrying. In order to face this problematic situation, many approaches coping with these faults (essentially software faults) have been put forward in the past. In distributed systems environment, two of these approaches are particularly investigated like conversation scheme[2,3] and Programmer Transparent Coordination (PTC) scheme[4,5]. The two schemes are based on the recovery blocks concept.

The main problem in taking into account the fault tolerance attribute in distributed computing systems is the control of domino-effect situation. This tricky problem is characterized by an avalanche of process rollbacks involved in recovery which should incur an unacceptable time and space overhead. Generally, to deal with issue, it has been proposed two types of approaches:

- Approach based on programmed coordination of Recovery Blocks (RB).
- Approach based on automatic coordination of Recovery Points (RP).

The conversation scheme[2] represents the first type while the PTC scheme[3] fits the second one.

The conversation structure requires that: The interactive application processes should have to be synchronized at the exit from the conversation and, the basic rules defining its semantic might be stated. The closure of the conversation ensures a high degree of reliability and may guarantee a confinement of the damages caused by an error. So, no error propagation out of the conversation members takes place. The domino

**Corresponding Author:** M. Aliouat, Department of Computing Science Ferhat Abbes University, Setif Route de Bejaia Sétif 19000 Algeria

effect is then controlled. This positive aspect is obtained to the detriment of the synchronization constraint[6] which may raise deadlock problem. The stringent programmer endeavor to coordinate the recovery blocks gives conversation a lack in flexibility and is programmer error prone. The efficient use of the conversation needs adequate languages, probably executable specification languages in order to relevantly verify its rules.

Aiming to alleviate the application programmer from the burden of coordinating the conversation recovery blocks (preventing additional errors source), the PTC scheme has been proposed in[4-6]. The aimed goal is to render efficient and easy the cooperation of recovery activities among concurrent processes, so the PTC scheme allows: 1. Independent design and uncoordinated recovery capabilities of distributed processes such that: Every process say Pi ($1 \leq i \leq N$) is liable for damage that it may cause (i.e., Pi is responsible for detection and error recovery). This principle leads to exclude the S-propagation (see definition D4) from consideration in recovery mechanisms. This type of recovery propagation means a propagation of states restoration to senders (processes) of application messages. The complementary and natural notion of the S-propagation is the R-propagation (states restoration of contamined receiver processes).

- Automatic recovery points setting before some message receptions.

The PTC scheme efficiency is essentially due to:

- Minimization of undoing computation amount during recovery actions, therefore lessening or preventing domino effect,
- Stating formal rules for managing recovery activities,
- Transparency in complete automation of recovery capabilities, so, no programmer involving is required

The PTC scheme has also some weaknesses such as:

- Weak trustworthiness in achieving reliable recovery
- Overhead in time and space,
- Exhausted Importer (EI) problem[7], where a sender process which provides erroneous messages, cannot correct them before all alternative's recovery blocks of receiving process (es) have been exhausted.
- Prohibiting S-propagation in the scheme, certainly makes easier cooperative backward error recovery, but may generate EI problem and can lead to domino effect situation. This may also raise the problem of trustworthiness in consumed messages and in delivered services. So, in this sense, the conversation scheme is more trustworthy.

The main drawback of the original PTC is the insufficient reliability which may be expected from a recovery operation to be achieved. Indeed, making a process responsible for detecting and correcting its own errors is very difficult to realize (indeed impossible). So, any communicating process Pi may be either messages producer or messages receiver one. Then, if process Pi fails, it is fair to consider two possible cases:

- The error is originated from Pi and Pi is in charge for error detection and recovery (PTC principle)
- The error is exported from any other process Pj via sent messages, then Pi is a victim and may accuse Pj to be an error source, this later case is not handled in PTC scheme. Therefore, the only way allowing Pi to recover correctly is:-to trigger Pj rollback via S-propagation,-to initiate its own rollback and-eventually to force, towards R-propagation, every Pk rollback (The process Pk has consumed Pi's messages).

Although interesting, both schemes, as conversation as PTC, have some limitations either in reliability or efficiency, either in implementation capabilities (complexity). So, it is to be expected that a "good" backward error recovery approach is one which can particularly provide following criteria:

- A high reliability in cooperative error detection and backward error recovery. This means allowing the two types of propagation (S and R-propagation),
- Minimization of undoing computation amount (notably, preventing domino effect situation),
- Efficiency in taking maximal advantage of processes concurrency,
- Transparency by complete automation of recovery capabilities,
- Easiness in implementation as a recovery mechanism free from domino effect.

To meet most of the previous criteria by a novel approach, that is, to improve existing recovery schemes (conversation and PTC), we propose a novel view of PTC (PTC-based Variant) providing an enhancement in recovery capabilities in following manner:

- Making PTC more reliable in placing more reliance on it recovery ability
- To give application programmer a choice to privilege either the reliability degree of a critical application or

the real-time application response time or making a trade-off between the two extremes.

## DEFINITIONS AND NOTATIONS

**Notations:** Let Pi, [1 ≤ I ≤ N] be a process, RBE[Pi.x] denotes the recovery block execution of process Pi, with a recovery point x. (Pi.x) denotes the recovery point x in Pi (P.x.a) denotes a Pseudo Recovery Point (PRP) in RBE [P.x].
AT: an Acceptance Test;
BRP: Base Recovery Point, set at the beginning of a Recovery Block (RB).

**Definitions:** Before PTC-V presentation, some preliminary definitions are needed to lighten some concepts. Let Pi, Pj and Pk be three communicating processes.

**D1:** If Pi is exporter (sender) of an application message M and Pj is importer (receiver) of that M (M flows from Pi to Pj), then when Pi fails to pass its AT, Pi has to revoke M so, Pi is said to be a Direct Revocator (DR) of Pj. Conversely, Pj is called a Direct Dependent (DD) of Pi. The same DR and DD notion may be also applied to RBE's of processes.

**D2:** When [Pi.y] becomes DR of an RBE[Pj.x] at reception (in Pj) of a message M and if [Pi.y] had a DR [Pk.z] before sending M, then [Pk.z] is said an Indirect Revocator (IR) of [Pj.x] if [Pk.z] was not previously DR of Pi. Every IR of [Pi.y] is an IR of [Pj.x] if it was not DR of Pi.

Recursively, if Pj imports information message M from Pi, then every DR of Pi, before sending M, becomes DR of Pj. Every RBE is DR of its own process and every DR of an RBE is also a DR of the including RBE (in case of nested RB's).

For example: RBE [P2.a], in Fig. 1, is a DR of RBE [P4.a] and RBE [P0.a] is an IR of RBE [P4.a].

- If [Pi.x] is a DD of Pj, after message reception M in Pj, then every DD of [Pi.x] acquired before reception of M is called *Indirect Dependent* (ID) of Pj. Every ID of Pj is also ID of its every DR.

Example: In Fig. 1, [P5.a] is an ID of [P2.a]

Note that: The DR's of a PRP are those acquired at its establishment. The DR's of an RB are: Itself plus the DR's union of every PRP included in this RB. The DR's of a process Pi are the DR's union of every RB in Pi. A Direct Revocator Set (DRS) is the DR's set of PRP's and RB's, (or process's DR's).
The Revocator Set (RS) of a RBE[Pi.x] is: RS=DR∪IR

**D3:** If RBE [Pj.y] receives application message M from RBE[Pi.x] then [Pj.y] is said Direct Accusator (DA) of

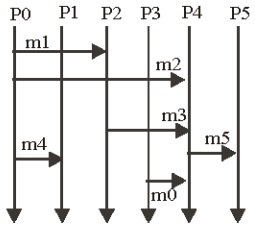

Fig. 1: Definite invocation

[Pi.x]. Conversely, [Pi.x] is called *Accused* of [Pj.y]. Every DA of [Pj.y] is DA of all its accused's. For example, in Fig. 1, [P4.a] is a DA of [P2.a] and [P2.a] is an accused.

- When [Pj.y] is DA of [Pi.x], then every DA of [Pj.y] acquired before reception of message M becomes *Indirect Accuser* (IA) of [Pi.x]. Every IA of [Pj.y] is also IA its accused's. In Fig. 1, [P4.a] is IA of [P0.a].

The Accuser's Set (AS) of a RBE[Pj.y] is: AS=DA∪IA

- IF RBE[Pk.z] is a DA of [Pi.y] and [Pj.y] is a DR of [Pi.x] then, if [Pi.x] and [Pk.z] have no relation between them, then [Pi.x] and [Pk.z] are mutually IR.

Note that: The relation "is accuser of" is a sub-relation of 'is a revocator of"; the former one is used for invocation with accusation, while the latter is used for invocation with revocation. The relation "is revocator of" is associated to the RBE's and their accuseds, but the relation "is a revocator of" corresponds to processes and their DR's. The DA Set (DAS) of every RBE in process Pj is a subset of DD Set of Pj.

**D4:** Let Pi, Pj, Pk ($i \neq j \neq k$) be three communicating processes such that: Pk imports Pj's messages and Pj imports Pi's messages, we said *S-propagation* of rollbacks issued from Pj to reach every partner process Pi such that Pj is accuser of Pi. Conversely, an *R-propagation* is a rollback propagation issued from any process Pj to reach every process Pk such that Pj is a DR of Pk.

## PTC-V BASIC CONCEPTS

The system model for which our recovery approach has been developed may have the following characteristics:

- The system is distributed over nodes and for the sake of simplicity, we assume that each node performs a single application process.
- The computation trustworthiness of processes is ensured by recovery blocks which may be overlapped.

- The interprocess communication system is assumed to be reliable, i.e., the messages cannot be altered neither lost.

**Basic concepts:** The approach we propose to improve some insufficiency of previously mentioned schemes is based on some concepts like:

**Definite invocation**

**Definition:** A Definite Invocation (DI) initiated by a process Pi[8,9] is an invocation for a recovery where a called process Pj must roll back (one for all) to a recovery point supplied in the invocation message.

The aim of definite invocation is to avoid the qualitative aspect of the domino effect (preventing cyclic state restorations may be generated from progressive roll-back). This DI concept may be used in order to get an optimization in time and space overhead, since every process involved in cooperative recovery operation has to do one and only one rollback. In other words, a failing process is the only one which has to propagate state restoration to its partners. The failure of process Pi in recovery region (interval between two recovery points) implies revocation of all previously sent messages and therefore, each process received at least one message of those (potentially) corrupted messages has to be invoked, by revocation , to roll back. Every message received in a recovery region may be error prone then, any process originating of those messages may be invoked, by accusation, to roll back. So, to give traditional PTC scheme more reliability and making it more realistic, PTC-V take into account the two types of recovery propagation previously defined (see D4). The systematic use of S-propagation in concurrent systems, where recovery blocks are not well coordinated, may raise the quantitative aspect of domino effect (intolerable undoing computation amount). As to prevent this fastidious and costly situation, we introduce two additional new concepts like: 2 by 2 reliability and accusation limit:

**2 by 2 reliability**

**Definition:** A 2 by 2 reliability is the trustworthiness granted to an information message which has been validated by two correlated acceptance tests.

This concept relies on the trust conferred to an information message which has been validated by an acceptance test in sending recovery block (in message producer) and in the corresponding receiving one (in message consumer). This principle ensures that any message M, issued from a RB[Pi] such that: RB[Pi] is invoked (for rolling back by accusation) from RB[Pj] and M is not suspected to be erroneous by any other RB[Pk], M is not resent to RB[Pk] during roll back. This may contribute to detect possible message alteration during transmission.

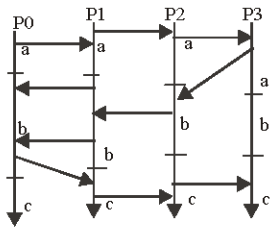


Fig. 2: Accusation limit

From 2 by 2 reliability concept, an invoked process by accusation, must reproduce only messages for which it has been accused. So, during recovery operation, a subset of received messages must be (locally) compensated instead of recreated. Like this, the message compensation operation is achieved according to saving and deleting formal rules. The situation depicted in Fig. 1 illustrates this principle. For example, when P2 fails at its acceptance test in RBE[P2.a], according to 2 by 2 reliability concept, we may have:

- Due to message m1, which may be erroneous, P0 is invoked by accusation,
- While message m3 and m5 may be corrupted, process P4 and P5 are invoked by revocation.
- Process P1 is not concerned with recovery, since neither RBE[P1.a] nor RBE[P0.a] has to revoke message m4. P0 agrees with P1 about correctness of message m4, since their acceptance tests have been successfully passed.

With definite invocation, Processes P0, P4 and P5 are invoked by P2 to roll back respectively from, [P0.a], [P4.a], [P5.a]. With message compensation mechanism, process P4 has only to read message m6 without triggering P3 to roll back for reproducing it. This may be used for the benefit of stopping systematic S-propagation.

**Accusation Limit**

**Definition:** Let Pi be a process involved in a recovery operation, if L is a number of accused Recovery Blocks belonging to Pi, then L is said: level of *accusation limit*.

By means of accusation limit, the S-propagation may stops at a level which may be considered as reaching an acceptable degree of reliability. It may be used efficiently to take into account the error detection latency. The example depicted in Fig. 2 shows:

If P2 fails at acceptance test associated to [P2.c], then by accusation, the states restoration will be propagated until the initial system state is reached i.e. the state defined by the recovery line {[P0.a], [P1.a], [P2.a], [P3.a]. When accusation limit principle is used and if for instance L is set to 2 (L = 2), then the recovery line will be {[P2.b], [P1.b], [P0.b], [P3.a]}.

The level L is tightly dependent on the application constraints, particularly:

- Reliability Degree (RD) to be wished for the running application,
- Desirable Response Time (RT) for this application.

Since L represents the maximum number of accused RB's in each involved process, L is then proportional to RD and RT. An estimation of the function giving L may be:

$F_L$ = C1*RD + (1/C2)*RT, where C1 is the detection coverage factor (C1=1 => perfect acceptance test) and C2 is the deadline factor (when C2 ➔ ∞ => RT has no effect). The justification of $F_L$ is application dependent, since RD and RT may not be strictly satisfied in the same time. According to the application requirements, we may privilege RD over RT (and vice versa) and as may be the case, the value of L must be consequent. It is noteworthy that $F_{-L}$ may be significantly used to estimate the strategy effect on the application behavior (to appreciate incurred time/space overhead).

## PTC-V OPERATIONAL PRINCIPLES

To take decision concerning recovery activities, PTC-V uses the dependency relationships between processes. This concerns automatic setting of new RP's, discarding unusable ones and identifying the processes which are involved in a recovery operation.

**Basic recovery points rules**

**R0: Insertion rule of Pseudo Recovery Point (PRP):** A PRP is set in process Pi whenever P imports application messages from any exporter process Pj and there exists at least one Direct Revocator of Pj which is not included in DR set of Pi.

**R1: Pseudo Recovery Point discarding rule:** Any PRP (P.x.a) may be discarded if all members of (P.x.a) Revocator Set and (P.x.a) Accusator Set are partially validated.

**R2: BRP destruction rule:** Any BRP [P.x] may be discarded if:

- The acceptance test associated to [P.x] is successfully passed and
- All members of RS([P.x]) are partially validated and
- All members of AS([P.x]) are partially validated.

**R3: Accusation limit rule:** Let L be the accusation limit level, when getting a new DR[Pj.y], any RBE[Pi.x] (such that: i ≠ j, Pi and Pj are linked with accusation relation) must keep only the L accused RB belonging to the same process Pj. These RB's are the most recent ones.

**Coordination of RBE states:** In order to correctly determine its state, any RBE must keep information concerning the states of RS and AS components. For this, the following actions have to be performed, like: Transmitting the states of RBE's and handling the related information.

**Communication of RBE states:** For states communication between RBE's, state messages have been defined:

**Invocation message (IM):** When a RBE fails to pass its acceptance test, a definite invocation message is sent. The IM is piggybacked by the set of direct accuser's and direct dependents of the failing RBE. This information forms the valid Recovery Line.

**Application message (AM):** This type of message is considered as a state message because it carries the DR set of emitting process with information message issued from application process. This enables to keep trace of new DR's and to maintain the revocation relation in every node of the system.

**Dependency message (DM):** This type of message is sent from an RBE[P.x] when an AM is received. The DM contains the DR set (previously received in AM) and the DA of RBE[P.x]. The DM is essential to maintain the accusation relation.

**Validation message (VM):** It is sent when an RBE passes successfully its acceptance test. Its aim is to validate the RBE state. The VM contains all DRS and DAS elements (having already passed their tests) with their recent states. The VM is useful for maintaining the IR and IA relationships.

**Messages compensation principle:** The main aim of message compensation is to avoid re-creating some indispensable messages for rolling back processes, since these messages are not revoked and have been locally saved during first reception. So, although a process Pi is linked to another failing one Pj via AD relation, the Pj's rollback may leave the process Pi not concerned. For this purpose, a lot of computation may be saved, resulting particularly in:

- The number of RB alternates to be performed during recovery may be less than in ordinary computation; the probability of new error occurrences (notably transient hardware faults) is lowered, so greater is the likelihood of recovery success.

- Decrease in messages communication may advantageously alleviate the transmission system load. Another resulting advantage is the possibility of stopping S-propagation roll back, since a process may not be constrained to rollback for merely reproducing (to a demander one) an indispensable message.

**R4: Compensation rule:** During a rollback of a process Pi, any message M will be compensated if:

- M is not produced by a Process Pj invoked (to roll back) by revocation i.e. Pj is neither a failing process nor a DD of a failing one.
- M is sent by a process Pj invoked by accusation, but M does not belong to the set of messages for which Pj is accused.

**R5: Rule of message savings to be compensated:** If a process Pi acquires a new DR subsequently to reception of message M, then Pi has to set a PRP (Pi.x.a) and M must be saved in compensation list of (Pi.x.a). If Pi receives M without new DR, then M will be saved in compensation list of PRP established when the sender of M is acquired as a DR.

**R6: Rule of deleting compensation list:** A compensation list associated to a PRP(pi.x.a) belonging to an RBE [Pi.x] of Pi may be deleted if:

- Pi has successfully passed its acceptance test associated to [Pi.x] and
- All AS elements related to [Pi.x] are partially validated and
- All PRP set before (Pi.x.a) are deleted,

The condition 3 above ensures that if Pi is invoked by revocation, then the rollback will takes place from (Pi.x.a) or from a PRP subsequent to (Pi.x.a). In this, after having deleted, a compensation list is no longer needed. Note that: The compensation list is updated when one of the two state messages IM or AM is received (According to rule R6, the compensation lists associated to PRP's of an RB, are discarded one by one; but if the time factor is critical, deletion of those lists may be done only during BRP discarding).

**Recovery algorithm:** When an RBE[Pi.x] $\in$ Pi fails, Pi has to broadcast a definite Invocation Message to roll back. Any process, say Pj, receiving such a message may be in one of the following cases:

- Pj is an accused of [Pi.x],
- Pj is a Direct Dependent of [Pi.x].
- Pj has no relation with [Pi.x].

Thereafter, Pj has to perform the following actions:

- Determine its situation: If concerned, Pj updates the states of RBE's contained in RS and AS lists.
- If Pj is an accused, then Pj has to roll back if IM contains a RBE[Pi.x] such that: RBE[Pi.x] is direct accuser of RB[Pj.y]
- If Pj is a direct dependent then, Pj has to determine, according to RBE supplied in IM, the PRP from which the roll back will be performed.

Any process Pj having identified its roll back point from IM message has to perform a single state restoration. So, Pj initiates a new recovery block alternate if it is accused but re-executes only the current RB alternate if it is a direct dependent.

- If Pj has no relation with [Pi.x], then two cases may be considered:
- There exists a least a process Pk which is in dependency relation with Pj such that Pk is invoked to roll back so, Pj must update the RBE's states contained in RS and AS sets.
- Pj has no relation with any invoked process then, the IM message is merely ignored.

Three types of failures may be occurred:

- Single failure corresponding to a single failing process,
- Concurrent failures corresponding to several failing processes
- Cumulative failures related to one or many processes failed during recovery operation.

PTC-V ensures a recovery operation exempted from domino effect for the precedent failure cases; this is done according to: - A single rollback per process involved in this operation according to definite invocation,

- A minimization of undoing computations via PRP insertion for revoked processes and accusation limit for accused ones.

## PTC-V PROPERTIES VALIDATION

In this section, we present some lemma related to some properties which are used to prove:-The high reliability and efficiency improvement of PTC-V,-the efficient maintaining of dependency relationships and

complete coordination of interprocess recovery activities. The preliminary comparison study of conversation and PTC schemes [8] has revealed that conversation is more reliable than PTC.

**Lemma 1:** PTC-V is more reliable than conversation and PTC protocols.

**PTC-V vs traditional PTC:** When naturally and systematically allowed, the S-propagation and R-propagation provide more reliability in PTC-V, since in PTC only R-propagation is triggered. Indeed, let Pi and Pj be two communicating processes, when Pj fails after having received (consumed) messages fro Pi, it is logical to consider that the error source may be either Pi or Pj itself. The only way to recover efficiently with maximum of trustworthiness is to restore Pj state (at error free one) and propagate this decision to Pi for doing the same operation. Since PTC-V, contrary to classical PTC scheme, uses both types of propagation, it is obvious that a recovery action is more trustworthy (reliable) in PTC-V protocol.

**PTC-V vs conversation:** Let Pi, i $\in$[1, N] be a set of communicating processes involved in a recovery operation and L is the level accusation limit. Let C be a conversation enclosing the N processes. In C, each process Pi is represented by a recovery block so, when a process Pj in C fails, N acceptance tests should be performed to cooperate for validating the computation inside C, while in PTC-V this computation should be validated by N*L acceptance tests at least ($L \geq 1$).

Furthermore, the recomputation of the current recovery block alternate of a revoked process may be sufficient to discard error effects which have been propagated to it via received messages. This may be used in knowledge when practically the current alternate is designed to be more efficient than the next one.

Note that: As it is known, the efficiency of RB alternatives is inversely proportional to their chronological order, so the re-execution of the current alternate may grant to PTC-V more efficiency (contrary to systematically executing the next alternate as in conversation scheme).

**Lemma 2:** According to rule R3, the number of accusers and accuseds managed for each process is minimized. If L is the accusation limit level, N is the number of interacting processes and A is the accused number of an RBE, then A is such that: A=N*L.

**Proof:** Every RBE [Pi.x] may have a number X of DR belonging to a process Pj, X will increase with interactions between Pj and [P.x]. If L is the accusation limit level, then [Pi.x] will keep at most, L accuseds belonging to Pj. Then, at any time, the RBE [Pi.x] will maintain no more than L*N accuseds. This number represents the accuseds number transmitted by process Pi to its Direct Dependent.

Any RBE[Pi.x] having acquired X (X = L) accuseds, after receiving a message from a process Pj, keeps no more than L accuseds of Pj ([Pi.x] will be a DA of Pj). For the remainder of accused RBE's (i.e. X - L), [Pi.x] is only a DD for them. They can revoke it but [Pi.x] cannot accuse them (in consequence of accusation limit). So, if A represents the total number of accuseds maintained in each node, then A is such that: A=N*L.

**Lemma 3:** Comparatively to PTC and according to BRP deleting rule and accusation limit, PTC-V manages a minimal number of BRP per process.

**Proof:** According to BRP deleting Rule (R2), only the needed BRPs for an invocation by revocation or accusation are maintained. With the accusation limit, every Direct Accuser [Pi.x] of a process Pj, may accuse no more than L recent recovery blocks of Pj; the rest of RB's of Pj cannot be accused by [Pi.x] and its DA. So, the destruction of their BRP is not delayed until partial validation of [Pi.x] and its DA will take place. The BRP destruction is then anticipated, since it is based on the RS and AS sets which are reduced by the rule R3.

## SPECIFICATION OF PTC-V SCHEME

In order to validate the behavioral properties of PTC-V, a formal specification is needed. For this purpose, the temporal logic and algebraic data types formalisms have been used to both specifying the functional aspect and temporal constraints.

The Abstract modules[10,11], combining together temporal logic and algebraic data types in same modules, was the formal specification tool we used.

The target fault tolerant system has then been specified via an abstract system formed by a set of modules namely: Application, PTC-V, network, operating system, which cooperate for achieving recovery capabilities.

The basic recovery module of this system is PTC-V which uses a set of cooperative sub-modules such: Invocation, Validation, Interaction and Dependency[9].

The intra and extra interactions modules have been specified with temporal logic connectors expressing: priorities, authorization conditions, etc.

## FORMAL VALIDATION OF SYSTEM PROPERTIES

The proof of properties allows the system to be confined in its original specification i.e. avoiding divergence from the last one. The concurrent properties are of two types:

Safety property and liveness property[12,13]. For proving these properties, two methods are used:

- Invariants method for safety property, - Attached assertions method for liveness property. To validate properties, a system of formal proofs is required. For this, the abstract module method has been extended to include an axiomatic proof system (set of theorems, axioms and inference rules of a classical and temporal logic). Such a system has been used to demonstrate safety and liveness properties of the target distributed system adopting PTC-V scheme. So, it was proved that:
- The progression of the target system is effective (liveness property) i.e. "Every request made to PTC-V modules is eventually served",
- The system is domino effect free and exempt from deadlocks (safety property).

## CONCLUSION

From scrutiny of previously proposed recovery schemes like conversation and PTC, it results, in each one, some weaknesses either in reliability or in efficiency. PTC-V is put forward in order to cope with these lacks and attempts to make the recovery capabilities more efficient and reliable. This aim is feasible with new concepts such as: Definite invocation (avoiding progressive rolls back) and accusation limit (allowing a flexible based-application trade off time-space overhead and reliability degree provided). These concepts combined with 2 by 2 trustworthiness notion may achieve a tolerable undoing amount of computation. Even though no restriction is imposed to the S-propagation, PTC-V is domino effect free.

A formal specification of a liable distributed system using this approach has been realized and a formal validation of its essential properties has been proved. Due to the space restriction of the paper, the specification description and formal validation of the latter one have been deliberately omitted.

## ACKNOWLEDGMENT

The research work reported here was supported in part by the National Agency for Research and Development under grant B1901/58/05.

## REFERENCES

1. Adiga, N.R. *et al.*, 2002. An overview of the BlueGene/L supercomputer In Proceedings of the super computing Conference (SC'2002), Baltimore MD, USA, pp: 1-22.
2. Randell, B., 1975. System Struture for Software faut Tolerance IEEE Trans. on Soft. Eng., N°2.
3. Tyrrell, A., 1986. Design of Reliable Software in Distributed Systems using Conversation Scheme, IEEE, Trans. on Soft. Eng. N°9.
4. Kim, K.H. and J.H. You, 1990. A Highly Decentralised Implementation Model for the Programmer Transparent Coordination (PTC) Scheme for Cooperative Recovery FTC 20.
5. Kim, K.H., 1988. Programmer Transparent Coordination of Recovering Concurrent processes: Philosophy and rules for efficient implementation " IEEE, Trans. on Soft. Eng. N°6.
6. Kim, K.H., 1989. Performance of Look ahead execution in the conversation scheme, IEEE, Trans. on Comp. N°8.
7. Kim, K.H., J.H. You and A. Abou-el-naga, 1986. A Scheme for Coordinated Execution of Independently designed Recoverable Distributed Processes FTCS 16.
8. Aliouat, M., 1986. Reprise de processus en environnement distribué après occurrences de pannes matérielles transitoires ou permanentes Docteur Ingénieur Thesis IMAG/TIM3 lab., I.N.P.G, Grenoble, France.
9. Aliouat, M., 1992. Recovery in Distributed Systems from Solid Faults SAFECOMP'92, Zurich.
10. Kroger, F., 1987. Abstract Modules combining algebraic and temporal logic specification means TSI, N°6.
11. Kroger, F. and F. Simon, 1987. Abstract Modules: An Approach to Specification of Concurrent systems by means of temporal logic and ADT's prog. Specif. Proc. Worhshop, RP N°8711.
12. Owicki, S. and L. Lamport, 1982. Proving Liveness properties of Concurrent Programs ACM Trans. on Prog. Lang. and Sist. N°3.
13. Kurose, J.F. and Y. Yemini, 1982. The Specification and Verification of a Connection Establishment Protocole using Temporal Logic in Protocol Specification, Testing and Verification. Ed. C. Sunshine. North- Holland Publishing Company.