Reprise de processus dans un environnement distribué après pannes matérielles transitoires ou permanentes

Makhlouf Aliouat

▶ **To cite this version:**

Makhlouf Aliouat. Reprise de processus dans un environnement distribué après pannes matérielles transitoires ou permanentes. Modélisation et simulation. Institut National Polytechnique de Grenoble - INPG, 1986. Français. NNT : . tel-00320133

HAL Id: tel-00320133

https://theses.hal.science/tel-00320133

Submitted on 10 Sep 2008

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THESE

présentée à

L'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

pour obtenir
le **TITRE de DOCTEUR INGENIEUR "Informatique"**

par
ALIOUAT Makhlouf

==========

REPRISE DE PROCESSUS DANS UN ENVIRONNEMENT DISTRIBUE APRES PANNES MATERIELLES TRANSITOIRES OU PERMANENTES

==========

Soutenue le 21 avril 1986 devant la commission d'examen

M.	L. BOLLIET	Président
MM.	J.S. BANINO	
	B. COURTOIS	Examinateurs
	J. MOSSIERE	

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Président : Daniel BLOCH
Vice-Présidents : B. BAUDELET
R. CARRE
H. CHERADAME
J.M. PIERRARD

Année universitaire 1985-1986

Professeurs des Universités

E.N.S.E.E.G.

BEAUFILS	Jean-claude	LOUCHET	François
BESSON	Jean	PARIAUD	Jean-Charles
BONNETAIN	Lucien	RAMEAU	Jean-Jacques
BONNIER	Etienne	SOHM	Jean-Claude
DURAND	Françis	SOUQUET	Jean-Louis
GUYOT	Pierre		

E.N.S.E.R.G.

BARIBAUD	Michel	COUMES	André
BLIMAN	Samuel	GENTIL	Pierre
BUYLE BODIN	Maurice	GUERIN	Bernard
CHENEVIER	Pierre	POUPOT	Christian
COHEN	Joseph	SERMET	Pierre
COUMES	André		

E.N.S.I.E.G.

BARRAUD	Alain	JOUBERT	Jean-Claude
BAUDELET	Bernard	JOURDAIN	Geneviève
BLOCH	Daniel	LACOUME	Jean-Louis
BRISSONNEAU	Pierre	LONGEQUEUE	Jean-Pierre
BRUNET	Yves	MASSELOT	Christian
CAVAIGNAC	Jean-François	MORET	Roger
CHARTIER	Germain	PAUTHENET	René
CHERUY	Arlette	PERRET	René
DURAND	Jean-Louis	PERRET	Robert
FOULARD	Claude	POLOUJADOFF	Michel
GAUBERT	Claude	SABONNADIERE	Jean-Claude
IVANES	Marcel	SCHLENKER	Claire
JALINIER	Jean-Michel	SCHLENKER	Michel
JAUSSAUD	Pierre		

E.N.S.H.G.

BOIS	Philippe	LESPINARD	Georges
BOUVARD	Maurice	MOREAU	René
CAILLERIE	Denis	OBLED	Charles
LESIEUR	Marcel	PIAU	Jean-Michel
TROMPETTE	Philippe		

E.N.S.I.M.A.G.

FONLUPT	Jean	ROBERT	François
MAZARE	Guy	SAUCIER	Gabrielle
MOSSIERE	Jacques	VEILLON	Gérard

U.E.R.M.C.P.P.

CHERADAME	Hervé	RENAUD	Maurice
CHIAVERINA	Jean	ROBERT	André
GANDINI	Alessandro	SILVY	Jacques

Professeurs Associés

BLACKWELDER	Ronald	ENSHG
HAYASHI	Hirashi	ENSIEG
PURDY	Gary	ENSEEG

Professeurs à l'Université des Sciences Sociales (Grenoble II)

BOLLIET	Louis
CHATELIN	Françoise

Chercheurs du C.N.R.S.

Directeurs de recherche :

CAILLET	Marcel
CARRE	René
FRUCHART	Robert
JORRAND	Philippe
LANDAU	Ioan

Maître de recherche :

ALLIBERT	Colette	GIVORD	Dominique
ALLIBERT	Michel	EUSTATHOPOULOS	Nicolas
ANSARA	Ibrahim	JOUD	Jean-Charles
ARMAND	Michel	KAMARINOS	Georges
BINDER	Gilbert	KLEITZ	Michel
BONNET	Roland	LEJEUNE	Gérard
BORNARD	Guy	MERMET	Jean
CALMET	Jacques	MUNIER	Jacques
DAVID	René	SENATEUR	Jean-Pierre
DESPORTES	Jacques	SUERY	Michel
DRIOLE	Jean	WACK	Bernard

Personnalités agréées à titre permanent à diriger des travaux de recherche
(Décision du conseil scientifique)

E.N.S.E.E.G.

BERNARD	Claude	MALMEJAC	Yves (CENG)
CAILLET	Marcel	MARTIN GARIN	Régina
CHATILLON	Catherine	NGUYEN TRUONG	Bernadette
CHATILLON	Christian	RAVAINE	Denis
COULON	Michel	SAINFORT	Paul (CENG)
DIARD	Jean-Paul	SARRAZIN	Pierre
FOSTER	Panayotis	SIMON	Jean-Paul
GALERIE	Alain	TOUZAIN	Philippe
HAMMOU	Abdelkader	URBAIN	Georges(ODEILLO)

E.N.S.E.R.G.

BOREL	Joseph	DOLMAZON	Jean-Marc
CHOVET	Alain	HERAULT	Jeanny

E.N.S.I.E.G.

BORNARD	Guy	LEJEUNE	Gérard
DESCHIZEAUX	Pierre	MAZUER	Jean
GLANGEAUD	François	PERARD	Jacques
KOFMAN	Walter	REINISCH	Raymond

E.N.S.H.G.

ALEMANY	Antoine	MICHEL	Jean-Marie
BOIS	Daniel	ROWE	Alain
DARVE	Félix	VAUCLIN	Michel

E.N.S.I.M.A.G.

BERT	Didier	DELLA DORA	Jean
CALMET	Jacques	FONLUPT	Jean
COURTIN	Jacques	SIFAKIS	Joseph
COURTOIS	Bernard		

U.E.R.M.C.P.P.

CHARUEL	Robert

C.E.N.G.

CADET	Jean	NIFENECKER	Hervé
COEURE	Philippe (LETI)	PERROUD	Paul
DELHAYE	Jean-Marc (STT)	PEUZIN	Jean-Claude(LETI)
DUPUY	Michel (LETI)	TAIEB	Maurice
JOUVE	Hubert (LETI)	VINCENDON	Marc
NICOLAU	Yvan (LETI)		

Laboratoires extérieurs

C.N.E.T.

DEMOULIN	Eric	MERCKEL	Gérard
DEVINE	R.A.B.	PAULEAU	Yves
GERBER	Roland		

****

ECOLE NATIONALE SUPERIEURE DES MINES DE SAINT-ETIENNE

Directeur	: M. M. MERMET
Directeur des Etudes et de la formation	: M. J. LEVASSEUR
Directeur des Recherches	: M. J. LEVY
Secrétaire Général	: Mle M. CLERGUE

Professeurs de 1ère Catégorie

COINDE	Alexandre	Gestion
GOUX	Claude	Métallurgie
LEVY	Jacques	Métallurgie
LOWYS	Jean-Pierre	Physique
MATHON	Albert	Gestion
RIEU	Jean	Mécanique-Résistance des matériaux
SOUSTELLE	Michel	Chimie
FORMERY	Philippe	Mathématiques Appliquées

Professeurs de 2ème catégorie

HABIB	Michel	Informatique
PERRIN	Michel	Géologie
VERCHERY	Georges	Matériaux
TOUCHARD	Bernard	Physique industrielle

Directeur de recherche

LESBATS	Pierre	Métallurgie

Maîtres de recherche

BISCONDI	Michel	Métallurgie
DAVOINE	Philippe	Géologie
FOURDEUX	Angeline	Métallurgie
KOBYLANSKI	André	Métallurgie
LALAUZE	René	Chimie
LANCELOT	Francis	Chimie
LE COZE	Jean	Métallurgie
THEVENOT	François	Chimie
TRAN MINH	Canh	Chimie

Personnalités habilitées à diriger des travaux de recherche

DRIVER	Julian	Métallurgie
GUILHOT	Bernard	Chimie
THOMAS	Gérard	Chimie

Professeur à l'UER de Sciences de Saint-Etienne

VERGNAUD	Jean-Maurice	Chimie des Matériaux & chimie industrielle

****

Remerciements

Je tiens à remercier,

- Monsieur Louis BOLLIET, professeur, directeur du département informatique de l'institut universitaire de technologie de Grenoble, qui a bien voulu me faire l'honneur de présider le jury de cette thèse.

- Monsieur Jacques MOSSIERE, professeur à l'E.N.S.I.M.A.G, directeur du laboratoire génie logiciel, qui a bien voulu lire et critiquer cette thèse.

- Monsieur Jean-Serge BANINO, directeur du département recherche et développememt MATRA DATA SYSTEM, pour l'attention toute particulière qu'il a accordée à ce travail et qui a accepté d'en être le rapporteur exterieur.

- Monsieur Bernard COURTOIS, chargé de recherche au C.N.R.S, responsable de l'équipe architecture des ordinateurs du laboratoire T.I.M.3, qu'il trouve ici l'expression de ma gratitude de m'avoir accueilli dans son équipe et dirigé cette thèse.

Je tiens également à remercier **Hajet** ma femme pour l'aide morale et matérielle qu'elle m'a prodiguée tout au long de la réalisation de la frappe de ce mémoire.

à ma femme Hayet
à mes parents

TABLE DES MATIERES.

Chapitre I: GENERALITES

I.1. Introduction. 1

I.2. Définitions. 4

Chapitre II: CONCEPTS ET APPROCHES DE REPRISE

II.1. Introduction. 11

II.2. Exceptions et approches de reprise. 11
II.2.1. Approche explicite. 12
II.2.2. Approche Implicite. 13

Chapitre III: MECANISMES DE REPRISE, PANNES MATERIELLES ET FAUTES LOGICIELLES

III.1. Introduction. 17

III.2. Pannes et erreurs d'origine matérielle et logicielle. 18
III.2.1. Terminologie. 18
III.2.2. Sources d'erreurs. 18
III.2.3. Types de pannes et leur tolérance. 20

III.3. Mécanismes de reprise. 20
III.3.1. Caractéristiques. 20
III.3.1.1. Classes d'erreurs à reprendre. 21
III.3.1.1.1. Les erreurs logicielles. 21
III.3.1.1.2. Ambiguïté des erreurs matérielles et logicielles. 22
III.3.1.1.3. Pannes Matérielles. 24
III.3.1.2. Particularité des systèmes à reprendre. .. 26
III.3.1.2.1. Type de systèmes. 26
III.3.1.2.2. Type de pannes. 26
III.3.1.3. Intervention du programmeur d'application. 27

III.3.1.4. Couverture de panne et de recouvrement. .. 28
III.3.2. Stratégies de reprise. 28
III.3.2.1. Reprise d'un processus indépendant. 28
III.3.2.1.1. Rôle d'un mécanisme de reprise. 30
III.3.2.1.1.1. Panne transitoire. 30
III.3.2.1.1.2. Panne permanente. 31
III.3.2.2. Reprise de processus communicants. 32
III.3.2.2.1. Propagation de reprise. 32
III.3.2.2.2. Détermination d'une ligne de reprise. 33

III.4. Description de quelques mécanismes existants. 35
III.4.1. Mécanismes dépendant du programmeur. 35
III.4.1.1. Conversation. 35
III.4.1.2. La S-conversation. 37
III.4.2. Mécanismes transparents au programmeur. 39
III.4.2.1. Protocole de reprise de processus interactifs. 39
III.4.2.2. Mécanisme de communication à travers moniteur. 40

III.5. Conclusion. 44

Chapitre IV: STRATEGIES DE REPRISE APRES PANNES MATERIELLES TRANSITOIRES ET PERMANENTES.

IV.1. Introduction. 48

IV.2. Description sommaire des diverses stratégies. 48
IV.2.1. Stratégies de base. 48
IV.2.1.1. Stratégie A. 48
IV.2.1.2. Stratégie B. 49
IV.2.1.3. Stratégie C. 50
IV.2.1.4. Stratégie D. 50
IV.2.2. Stratégies mixtes. 52
IV.2.2.1. Stratégie AD. 52
IV.2.2.2. Stratégie AB. 52

PARTIE I.

Chapitre I: INTRODUCTION.

Chapitre II: STRATEGIE A.

II.1. Description générale. 57
II.1.1. Pannes transitoires. 57
II.1.2. Critères de non postpropagation. 58
II.1.3. Informations indispensables. 59
II.1.3.1. Informations pour satisfaire les critères(1,2). 59
II.1.3.2. Informations nécessaires à la postpropagation. 59
II.1.4. Pannes permanentes. 61
II.1.4.1. Informations indispensables. 62
II.1.4.2. Adoption des critères (1,2). 64

II.2. Stratégie A sans postpropagation. 65
II.2.1. Réalisation des critères (1,2). 65
II.2.2. Pannes transitoires. 66
II.2.3. Pannes permanentes. 67
II.2.3.1. Problème de multiples pannes. 68
II.2.3.1.1. Mise à jour des propagateurs et dépendants. 69
II.2.3.1.2. Multiples pannes. 70
II.2.3.1.2.1. pannes simultanées. 70
II.2.3.1.2.2. Pannes consécutives. 70
II.2.4. Perturbations et chaines de reprise. 72

II.3. Stratégie A avec postpropagation. 73
II.3.1. Pannes transitoires. 74
II.3.1.1. Emission et réception de msg. 75
II.3.2. Pannes permanentes. 75
II.3.3. Perturbations et chaines de reprise. 77

II.4. Conclusion. 78

Chapitre III: STRATEGIE B.

III.1. Description générale. 79
III.1.1. Informations indispensables. 80

III.2. Stratégie B avec postpropagation.
III.2.1. Pannes transitoires. 82
III.2.1.1. Renvoi des msg. 83
III.2.2. Pannes permanentes. 85
III.2.2.1. Reprise. 86

III.3. Stratégie B sans postpropagation. 88
III.3.1. Pannes transitoires. 88
III.3.1.1. Renvoi (ou réémission) des msg. 89
III.3.2. Pannes permanentes. 90

III.4. Capacités de reprise. 91

III.5. Perturbations et chaines de reprise. 92

III.6. Conclusion. 93

Chapitre IV: STRATEGIE C.

IV.1. Description générale. 95
IV.1.1. Informations indispensables. 95

IV.2. Stratégie C sans postpropagation. 97
IV.2.1. Pannes transitoires. 97
IV.2.2. Pannes permanentes. 100

IV.3. Stratégie C avec postpropagation. 102
IV.3.1. Pannes transitoires. 103
IV.3.2. Pannes permanentes. 104

IV.4. Perturbations et chaines de reprise. 105

IV.4.1. Cas de reprise sans postpropagation. 105
IV.4.2. Cas de reprise avec postpropagation. 106

IV.5. Conclusion. 107

Chapitre V: STRATEGIE D.

V.1. Description générale. 109
V.1.1. Informations indispensables. 110

V.2. Stratégie D sans postpropagation. 113
V.2.1. Pannes transitoires. 113
V.2.1.1. Reprise. 113
V.2.2. Panne permanente. 114
V.2.2.1. Reprise. 115

V.3. Stratégie D avec postpropagation. 115
V.3.1. Pannes transitoires. 115
V.3.1.1. Reprise. 116
V.3.2. Panne permanente. 116
V.3.2.1. Reprise. 117

V.4. Perturbations et chaines de reprise. 119

V.5. Conclusion. 121

Chapitre VI: STRATEGIE AB.

VI.1. Motivation. 123

VI.2. Informations nécessaires. 124

VI.3. Opération de reprise. 125
VI.3.1. Stratégie AB sans postpropagation. 126
VI.3.2. Stratégie AB avec postpropagation. 127

VI.4. Perturbations et chaines de reprise. 128

VI.5. Conclusion. 130

Chapitre VII: STRATEGIE AD.

VII.1. Motivation. 131

VII.2. Informations indispensables. 131

VII.3. Opération de reprise. 132
VII.3.1. Stratégie AD sans postpropagation. 132
VII.3.2. Stratégie AD avec postpropagation. 133

VII.4. Perturbations et chaines de reprise. 135

VII.5. Conclusion. 136

Chapitre VIII: COMPARAISON DES DIFFERENTES STRATEGIES.

VIII.1. Motivation. 137

VIII.2. Critères de comparaison. 138

VIII.3. Récapitulatif. 139
VIII.3.1. Stratégies et effet-domino. 139
VIII.3.1.1. Cas sans postpropagation. 139
VIII.3.1.2. Cas avec postpropagation systématique. .. 140
VIII.3.2. Comparaison. 141
VIII.3.2.1. Pannes transitoires et permanentes sans postpropagation. 141
VIII.3.2.2. Pannes transitoires et permanentes avec postpropagation. 142

VIII.4. Conclusion. 144

Chapitre IX: ROLL-BACK DIRECT.

IX.1. Pannes transitoires. 147

IX.2. Panne permanente. 150

IX.3. Comparaison des deux principes. 152
IX.3.1. Roll-backs progressifs. 152
IX.3.2. Roll-backs directs. 153

PARTIE II.

Chapitre I: INTRODUCTION.

I.1. Motivation. 155

I.2. Reprise en présence de multiples pannes permanentes. 157

I.2.1. Nécessité d'une politique de désignation de stations de reprise. 157

I.2.1.1. Désignation statique. 157

I.2.1.2. Désignation dynamique. 158

I.2.2. Stratégie A et multiple pannes permanentes. 159

Chapitre II: STRATEGIE A.

II.1. Latence de détection nulle. 163

II.1.1. Pannes transitoires. 163

II.1.2. Pannes permanentes. 164

II.2. Latence de détection non nulle. 165

II.2.1. Pannes transitoires. 166

II.2.2. Pannes permanentes. 166

II.3. Perturbations. 167

II.3.1. Cas de latence nulle. 167

II.3.2. Cas de latence non nulle. 168

II.4. Conclusion. 170

Chapitre III: STRATEGIE B.

III.1. Latence de détection nulle. 171

III.1.1. Pannes transitoires. 171

III.1.2. Pannes permanentes. 172

III.2. Latence de détection non nulle. 173

III.2.1. Pannes transitoires. 173
III.2.2. Pannes permanentes. 174

III.3. Stratégie B et perturbations. 174
III.3.1. Cas de latence nulle. 175
III.3.2. Cas de latence non nulle. 176

III.4. Conclusion. 178

Chapitre IV: STRATEGIE C.

IV.1. Latence de détection nulle. 179
IV.1.1. Pannes transitoires. 179
IV.1.2. Pannes permanentes. 181

IV.2. Latence non nulle. 181
IV.2.1. Reprise commune. 182

IV.3. Perturbations. 183
IV.3.1. Cas de latence nulle. 183
IV.3.2. Cas de latence non nulle. 183

IV.4. Conclusion. 184

Chapitre V: STRATEGIE D.

V.1. Latence de détection nulle. 185
V.1.1. Pannes transitoires. 186
V.1.2. Pannes permanentes. 186

V.2. Latence non nulle. 187
V.2.1. Pannes transitoires. 187
V.2.2. Pannes permanentes. 187

V.3. Perturbations. 188

V.4. Conclusion. 189

Chapitre VI: STRATEGIES MIXTES.

VI.1. Stratégie AD. 191

VI.2. Stratégie AB. 191

Chapitre VII: CARACTERISATION ET COMPARAISON DES STRATEGIES

VII.1. Caractéristiques. 193
VII.1.1. Stratégie A. 193
VII.1.2. Stratégie B. 193
VII.1.3. Stratégie AB. 194
VII.1.4. Stratégie C. 194
VII.1.5. Stratégie D. 194
VII.1.6. Stratégie AD. 195

VII.2. Comparaison. 196
VII.2.1. Critères. 196
VII.2.2. Cas de latence nulle. 197
VII.2.3. Cas de latence non nulle. 202

VII.3. Conclusion. 204

Chapitre VIII: CREATION DES Rp

VIII.1. Algorithme de création coordonnée des Rp. 205

VIII.2. Occurrence de panne lors de la création. 206

VIII.3. Interférence des décisions. 208

VIII.4. Evitement d'interblocage. 210

Chapitre IX: OPTIMISATION

IX.1. Elimination du protocole interprocessus. 213
IX.1.1. Stratégies sans contraintes. 213

IX.1.2. Stratégies avec contraintes. 214
IX.1.3. Avantages. 214

IX.2. Elimination des Rp inaccessibles. 214
IX.2.1. Détermination des Rp inaccessibles
en stratégie A. 215
IX.2.1.1. Critères (1,2) satisfaits. 215
IX.2.1.1.1. Détermination des Rp inaccessibles. .. 216
IX.2.1.2. Critères (1,2) non satisfaits. 217
IX.2.1.2.1. Détermination des Rp inaccessibles. .. 219
IX.2.2. Détermination des Rp inaccessibles
en stratégie B. 223
IX.2.2.1. Critères (1,2) satisfaits. 223
IX.2.2.2. Critères (1,2) non satisfaits. 225
IX.2.3. Détermination des Rp inaccessibles
en stratégie C. 226
IX.2.3.1. Critères (1,2) satisfaits. 226
IX.2.3.2. Critères (1,2) non satisfaits. 228
IX.2.4. Détermination des Rp inaccessibles
en stratégie D. 230
IX.2.4.1. Critères (1,2) satisfaits. 230
IX.2.4.2. Critères (1,2) non satisfaits. 230
IX.2.5. Détermination des Rp inaccessibles
en stratégies mixtes. 231
IX.2.5.1. Stratégie AB.
IX.2.5.1.1. Critères (1,2) satisfaits. 231
IX.2.5.1.2. Critères (1,2) non satisfaits. 231
IX.2.5.2. Stratégie AD. 232
IX.2.5.2.1. Critères (1,2) satisfaits. 232
IX.2.5.2.2. Critères (1,2) non satisfaits. 232

IX.3. Conclusion. 234

Chapitre X: CONCLUSION. **235**

ANNEXE. **237**

X.1. Algorithmes de la première partie.

X.1.1. Stratégie A.
X.1.1.1. Stratégie A sans postpropagation. 237
X.1.1.2. Stratégie A avec postpropagation. 239
X.1.1.3. Algorithme de réalisation des critères de non postpropagation. 243
X.1.2. Stratégie B.
X.1.2.1. Critères (1,2) satisfaits. 245
X.1.2.2. Critères (1,2) non satisfaits. 246
X.1.3. Stratégie C.
X.1.3.1. Algorithme de reprise en panne transitoire. 250
X.1.3.2. Algorithme de reprise commun aux deux types de pannes. 252
X.1.4. Stratégie D.
X.1.4.1. Critères (1,2) satisfaits. 255
X.1.4.2. Critères (1,2) non satisfaits. 256
X.1.5. Stratégie AB.
X.1.5.1. Critères (1,2) satisfaits. 259
X.1.5.2. Critères (1,2) non satisfaits. 260
X.1.6. Stratégie AD.
X.1.6.1. Critères (1,2) satisfaits. 263
X.1.6.2. Critères (1,2) non satisfaits. 263

X.2. Algorithmes de la deuxième partie.
X.2.1. Stratégie A. 266
X.2.1.1. Critères (1,2) non satisfaits (latence non nulle). 266
X.2.1.2. Critères (1,2) satisfaits (latence nulle). . 268
X.2.2. Stratégie B.
X.2.2.1. Cas de latence nulle. 269
X.2.2.2. Cas de latence non nulle. 270
X.2.3. Stratégie C.
X.2.3.1. Cas de latence nulle. 271
X.2.3.2. Cas de latence non nulle. 273
X.2.4. Stratégie D.
X.2.4.1. Cas de latence nulle. 276
X.2.4.2. Cas de latence non nulle. 278

REFERENCES. 280

Chapitre I

GENERALITES

I.1. INTRODUCTION

L'intérêt manisfesté à l'égard de l'amélioration de la fiabilité a conduit à l'élaboration de techniques de construction de logiciels tolérant les pannes. Les systèmes deviennent ainsi capables d'assurer les services spécifiés en dépit des défaillances matérielles ou logicielles auxquelles ils peuvent être confrontés.

Ces techniques dépendent de deux approches fondamentales:
- L'approche implicite (backward error recovery) [HOR74, RAN75-78, KIM79-80-82, RUS77-79-80, WOO80-81, ANT84, SHR79] et
- L'approche explicite (forward error recovery) [LIS79, GOO75, CRI79-80-82, JAL83, MEL77, LUC80, MAC77].

L'approche explicite, basée sur une identification et la reconnaissance de l'erreur, remédie à la défaillance par correction de l'état erroné du système, en agissant seulement sur la partie endommagée. Cette démarche nécessite au préalable une évaluation précise des dommages subits par le système. Les exceptions et leur traitement en sont des mécanismes de récupération d'erreur qui adoptent une telle approche.

Quant à l'approche implicite, elle se caractérise par une indépendance vis-à-vis de l'estimation et la prédiction des dommages causés, et vis-à-vis de l'application elle-même. La reprise de l'erreur, contrairement à l'approche explicite, consiste à considérer que l'état du système affecté ne peut être corrigé par suppression ou isolement de cette erreur. Il convient alors de restaurer un état du système (présumé correct) antérieur à l'instant de panne, à partir duquel une relance aurait lieu. Le mécanisme des blocs de reprise [HOR74, AND76, RAN75, SHR78] en est un exemple fondé sur une pareille approche.

Doter un système de processus concurrents [CRO78, BRI74, COR81] d'un mécanisme de reprise après panne implique une prise en compte du problème complexe de propagation d'erreur relatif aux échanges

d'information interprocessus. Ainsi, si ces échanges (ou communications) ne sont pas coordonnés suivant l'établissement des points de reprise, l'approche implicite devient sujette à des propagations de reprise incontrôlées pouvant dégénérées en effet-domino (D15). Cette dégénérescence entraine onéreusement la révocation d'une importante activité des processus concurrents.

La relance dans un tel système se fait à partir d'un état cohérent(D5) ou ligne de reprise (RL) de ce dernier, et la politique selon laquelle cette état est déterminé constitue une caractéristique du mécanisme de reprise adopté. Pour cela deux politiques sont disponibles:

- Détermination statique, où la RL est établie a priori lors de l'écriture des programmes;
- Détermination dynamique, où la RL est déterminée au cours de l'opération de reprise.

Une seconde caractéristique réside dans la relation existant entre le programmeur d'application et les mécanismes de reprise. Ces derniers peuvent être indépendants [KIM79-80] pour ceux qui adoptent l'approche implicite, et entièrement intégrés dans l'application (structures linguistiques) pour ceux établis à partir de l'approche explicite [LIS79,JAL84,BAN78,LUC80,MAC77].

Qu'il se base sur l'une ou l'autre des deux approches, un mécanisme de reprise se distingue fondamentalement par rapport à la catégorie de panne auquelle il est destiné. Autrement dit, traite t-il les pannes (ou fautes) matérielles, logicielles, ou les deux ? S'il s'agit de pannes matérielles, convient t-il aussi bien aux pannes transitoires qu'aux pannes permanentes ?

Le traitement des exceptions, ou plus généralement l'approche explicite sont essentiellement dédiés à la tolérance des fautes logicielles. L'approche explicite convient aussi bien en fautes logicielles (par exemple le schéma des blocs de reprise [AND76, RAN76, SHR78]) qu'aux pannes matérielles [BAR83]. Cependant, si cette dernière se prête mieux aux pannes matérielles, la reprise en pannes permanentes ne peut se réaliser sans une redondance des composants matériels. Aussi les systèmes distribués, de part leur grande disponibilité, offrent un environnement adéquat à l'amélioration de leur capacité de tolérance aux pannes pouvant

ainsi leur conférer un caractère de systèmes non-stop.

Le but de la présente étude consiste en la reprise de processus dans un environnement distribué après occurrence de pannes matérielles transitoires ou permanentes. Car jusqu'à maintenant, quasiment aucune étude n'a été consacrée à cette catégorie de pannes notamment où les deux types (transitoires et permanentes) sont traités conjointement. Cette indissociation revêt un caractère important au vu de l'intérêt et l'essor considérable que connaîssent actuellement les systèmes distribués.

Le système que nous considérerons sera constitué d'un ensemble de stations physiques (sites) autonomes, dont chaque station est dotée d'un système local, et dont les échanges entre processus des divers sites s'effectuent sur la base de messages (msg = abréviation de message).
Plusieurs stratégies de reprise fondées sur l'approche implicite sont présentées. Les perturbations de l'environnement qui peuvent résulter d'une opération de recouvrement sont évitées partiellement par une conservation de msg reçus, ou totalement en y respectant de surcroît des critères empêchant la postpropagation interprocessus (D7).

A noter que la liberté des processus intéractifs tant au niveau de leurs communications qu'au niveau de la création de leurs points de reprise est entièrement préservée, ce qui jusque là n'a pas été le cas dans bien des mécanismes proposés [RAN75, SIM84, JAL84, BAR83, GRE85].

I.2. DEFINITIONS

D1: Un point de reprise (Rp: Recovery point) est l'instant où l'activité d'un processus en cours d'exécution est interrompue momentanément pour sauvegarder l'état de ce dernier afin de pouvoir le restaurer ultérieurement en cas de reprise.

D2: Un processus P exprime un engagement envers un de ces Rp (commitment) lorsque ce dernier n'est plus la cible de P pour une éventuelle opération de reprise. Autrement dit, immédiatement après création d'un Rp_i, il y a engagement envers Rp_{i-1}. Notation: P # Rp_{i-1}.

D3: Une région de reprise (RR) est l'intervalle d'activité d'un processus entre l'établissement d'un Rp et l'engagement envers ce dernier (fig.3).

D4: Un Rp est dit actif s'il est le plus récemment créé i.e: il n'a pas encore reçu d'engagement (fig.3).

- On appelle Rp de relance (RpR) le Rp choisi par le processeur et à partir duquel sera relancée l'exécution du processus à reprendre.

- Un point de reprise Rp_i est dit dominant d'un autre Rp_j, si Rp_i précède Rp_j dans sa création. Rp_i et Rp_j appartiennent à un même processus P. Notation $Rp_i < Rp_j$.

D5: Pour tout couple Rp, Rp' de points de reprise appartenant respectivement aux processus P et P', Rp est dit propagateur direct de Rp' (noté Rp --> Rp'), si et seulement si un msg au moins circule de P vers P'.
Réciproquement, Rp' est appelé dépendant direct de Rp (cf fig3).

D6: Rp est dit propagateur indirect de Rp' (noté Rp --/-> Rp') si et seulement si:
- Soit Rp est propagateur direct de Rp';
- Soit (récursivement) il existe Rp" ∈ P" telque:
  Rp est propagateur direct de Rp" et, Rp" ou tout autre point de reprise Rp_i ∈ P" successeur de Rp" est un propagateur

indirect de Rp'.
Réciproquement, Rp' est appelé dépendant indirect de Rp (cf fig3).

D7: Un point de reprise Rp est dit Post-Initiateur Potentiel de Reprise (PIPR) de Rp' si et seulement si:
- Rp est actif, et
- Rp est propagateur indirect de Rp' (cf fig3).

D8: Un Rp est dit inaccessible s'il n'est plus accessible à toute opération de reprise lancée localement, ou résultant d'une propagation de cette dernière.

D9: On appelle état cohérent d'un système (ou ligne de reprise: Recovery Line (RL)), un état défini par un ensemble de points de reprise représentant chacun un processus communicant, et constituant ensemble une "barrière" infranchissable à toute propagation de reprise.

D10: On désigne par processus Défaillant, un processus qui s'exécutait sur une station potentiellement défaillante puis repris par une station de reprise.

D11: On appelle invocation précise, une invocation à la reprise dont le Rp de relance désigné dans le msg de reprise, indique précisément le RpR du processus invoqué.
Par opposition, une invocation aléatoire est l'opération d'invocation dont le RpR du processus invoqué, s'il existe, est déterminé à partir du RpR du processus invocateur fourni dans le msg.

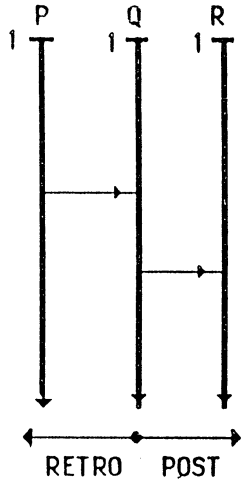
D12: - On appelle msg révoqué, un msg:
. Soit erroné i.e: les informations qu'il véhicule sont incorrectes;
. Soit correct, mais utilisé à tort, i.e: utilisé plusieurs fois alors qu'une seule est permise, ou bien utilisé à un instant inadéquat.
Autrement dit, son utilisation crée une perturbation.

- Un msg indispensable, est un msg capital à la réexécution d'un processus à reprendre, et dont l'acquisition peut provoquer la reprise du processus producteur pour le regénérer.

- On appelle rétropropagation (ou propagation par msg reçus), une propagation de reprise engendrée par un processus Pj pour atteindre un processus partenaire Pi dans le but de recréer pour Pj au moins un msg indispensable (cf figl).

- Une postpropagation (ou propagation par msg émis) d'un processus Pj vers un processus Pk, est une propagation de reprise provoquée par les msg émis par Pj et reçus par Pk lorsque ces derniers sont révocables.

Exemple:



La reprise du processus Q en Q1 provoque une rétropropagation du processus P en P1 et postpropagation de R en R1.

Types de propagations de reprise.

Figure 01

D13: On appelle graphe de reprise (GR) d'un processus P, le graphe qui décrit l'interaction de P avec ses partenaires Pj. Il contient les informations nécessaires au mécanisme de reprise (fig3).

D14: On dira qu'une propagation de reprise est de niveau n, si elle atteint n processus différents à partir du processus initiateur (défaillant).
Une propagation est d'ordre p relativement à un ou plusieurs processus, si la longueur de la chaine de reprise (roll-back)

est de p régions de reprise à partir de la région courante (fig3).

D15: L'effet-domino [RAN75] provoqué par une propagation de reprise interprocessus est caractérisé par une détermination d'un état cohérent du système au prix d'une importante révocation d'activité; notamment lorsque les processus impliqués se voient annuler leurs activités pour se relancer au début de leurs exécutions (voir exemple D14-D15).

D16: L'ensemble des Rp dépendants des Rp d'un processus P à un instant t, est appelé Liste de Dépendants (LD) de P.
Une sous liste de LD associée à un Rp donné de P est appelée Liste Immédiate de Dépendants de ce Rp (LID). Elle contient les dépendants directs de Rp.
De même, l'ensemble des Rp propagateurs directs des Rp de P à un instant donné constitue la Liste de Propagateurs de P (LP). L'ensemble des propagateurs directs d'un Rp de P est appelé Liste Immédiate de Propagateurs (LIP) de ce Rp (fig4).

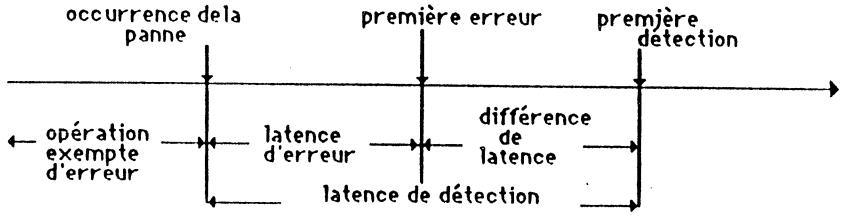
D17: Un point de reprise Rp' est dit rétro-initiateur potentiel de reprise (RIPR) d'un autre point de reprise Rp si et seulement si:
- Rp est actif et,
- Rp' est dépendant indirect de Rp.

D18: Un point de reprise Rp_i d'un processus Pj est appelé Initiateur Potentiel de Reprise (IPR) si et seulement si:
- Rp_i est RIPR ou bien
- Rp_i est PIPR.

D19: On appelle Identificateur de Région de Reprise, respectivement d'émission (IRRE) ou de réception (IRRR), l'identificateur du Rp associé à cette région de reprise, lorsqu'on considère un msg émis ou un msg reçu.

D20: On appelle latence de détection d'erreur [SHE78], l'intervalle de temps qui sépare l'occurrence d'une panne de la première détection de l'erreur générée.

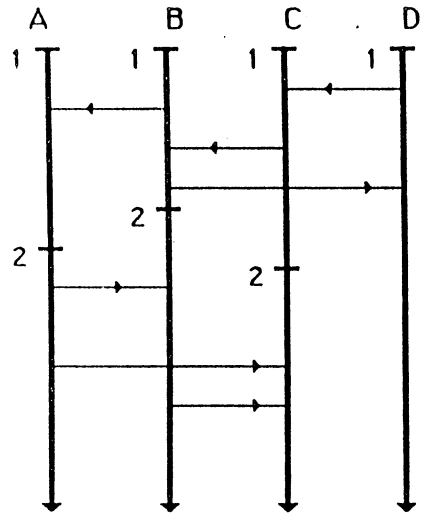
Exemple:



Latence de détection [She78]

Figure 02

Exemple récapitulatif:



un système de quatre processus intéractifs

Figure 03

(D1): A1, B1, C1... sont des points de reprise (Rp).

(D2): A1, B1, C1 ont reçu chacun un engagement de leur processus respectif.

(D3): L'activité du processus A dans l'intervalle [A1,A2] représente une région de reprise identifiée par le Rp A1.

(D4): Sont actifs les Rp suivants: A2, B2, C2, et D1. ∀Ai, ∀Aj Rp du processus A, Ai est dit dominant de Aj si i < j.

(D5): A2 est propagateur direct de B2, il est également propagateur indirect de C2.

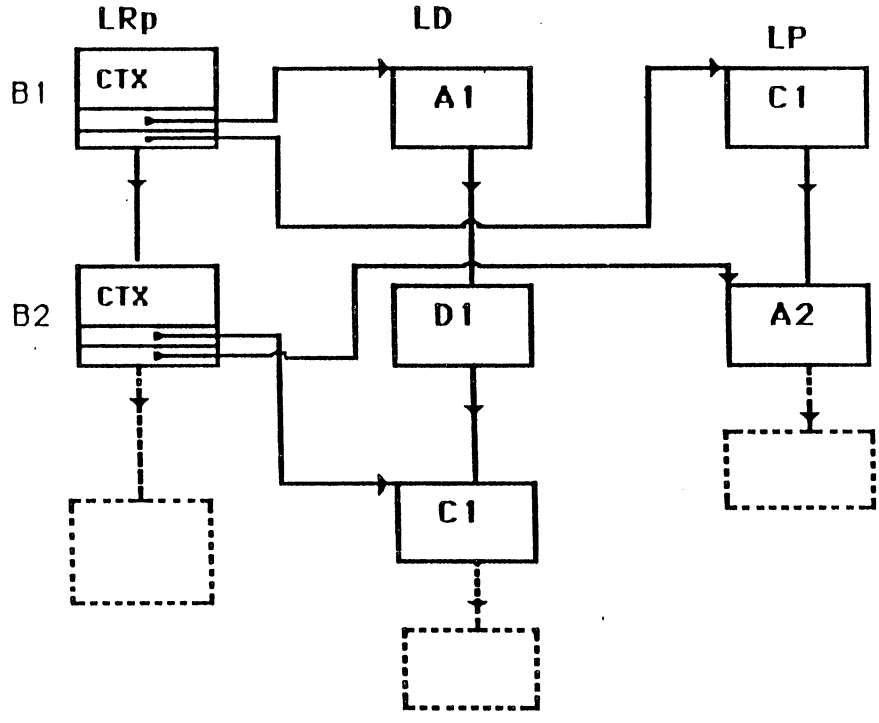
(D7): D1 est post-initiateur potentiel de reprise (PIPR) de A1, B1, C1; D1 est aussi RIPR de B1; C2 est RIPR de B2 et A2.

(D8): Selon la stratégie C (voir stratégies), aucun Rp de la figure 3 n'est inaccessible.

(D9): Lorsqu'une panne affecte un des trois processus A, B, C et que la postpropagation et rétropropagation s'imposent, l'ensemble des Rp <A2,B2,C2> forme une ligne de reprise. Une panne sur D donnerait comme RL: <A1,B1,C1,D1>.

(D14-15): L'effet domino peut être observé lorsqu'une panne affecte le processus D, et qu'une rétropropagation s'avère obligatoire. Ainsi, la propagation est de niveau n=3 et d'ordre $p_D=1$, $p_A= p_B =p_C= 2$.

(D16): Exemple de graphe de reprise (GR) du processus B de la fig.3.



Graphe de reprise

Figure 04

Chaque élément de la liste LRp comprend:

. Un pointeur de contexte (CTX) associé au Rp associé;

. Un pointeur de liste LID;

. Un pointeur de liste LIP.

La liste LD du processus B est constituée de <A1,D1,C1>, sa liste LP est <C1,A2>.

La liste LID du Rp B2 est formée de: <C1>, sa liste LIP est: <A2>.

Chapitre II

CONCEPTS ET APPROCHES DE REPRISE

II.1. Introduction:

La notion de reprise s'est imposée comme conséquence directe de l'intêret manifesté à l'égard de l'amélioration de la fiabilité des programmes, elle prend d'avantage d'ampleur à mesure que des applications requérant des logiciels hautement fiables [WEN 78] ne cessent de se développer.

C'est en fait, pour faire face aux erreurs engendrées par les pannes, qu'est née l'idée de doter les systèmes d'outils capables de leur assurer un fonctionnement le moins perturbé possible [AVI76, HEC79, AND81]. Ainsi des méthodes d'analyse sémantique et de preuves de programmes ont vu le jour, et permettent de s'assurer que certaines erreurs ne pourront jamais se produire mais ne garantissent aucunement que toutes les erreurs sont évitées, d'où la notion d'exceptions et de leur traitement [CRI79, MEL77, HOR74, NAC84]. Cette notion constitue l'élément de base des techniques de détection et de traitement d'erreur essentiellement d'origine logicielle, intégrées dans les langages de programmation [LUC80, LIS79, JAL84, GOO75, SHR79 MAC77,GRE85].

II.2. Exceptions et approches de reprise.

Une exception est définie [CRI79] comme ume tentative de violation à l'exécution d'une opération, d'une propriété invariante spécifiée sur un type de variables et propre à la sémantique du traitement qui y est effectué, et des informations manipulées.

La détection d'une exception, implique au préalable l'exécution d'un test représenté par l'évaluation d'une assertion [AND 79] associée à l'exception, et auquelle correspond un traitant défini de manière implicite ou explicite. Un traitant est un algorithme qui définit l'attitude à prendre à l'occurrence d'exception.

C'est cette caractéristique implicite ou explicite d'association exception-traitant qui définit respectivement

l'approche implicite (bacward error recovery) et l'approche explicite (forward error recovery), les termes "implicite" et "explicite" sont empruntés à [CRI79].

II.2.1. Approche explicite.

Dans cette approche [CRI79-80-82,LIS79,GOO75], on associe un traitant à toute opération susceptible de signaler une exception. Ce traitant explicite est établi selon la connaissance des causes de l'occurrence de l'exception (i.e: connaissance de sa précondition), et de la sémantique de l'opération qui accomplit le service attendu.

Le programmeur est ainsi capable de fournir un algorithme (traitant) pouvant réaliser un masquage de l'exception ou de pouvoir récupérer un état cohérent (D9) de l'opération invoquée (remise de l'état interne de la procédure appelée à celui ayant existé avant l'appel).

La fonction du traitant est la suivante:

- Rendre la précondition de l'exception fausse (masquage);
- Si masquage réussi, alors poursuite de l'exécution en séquence;
- Si toutefois la ou les tentative(s) de masquage ont échoué, alors restaurer l'état interne de l'appelé et propager l'exception vers l'appelant.

A la fin de l'exécution du traitant, le contrôle peut être rendu à la procédure ayant signalé l'exception, et plusieurs attitudes en cas d'échec peuvent être envisagées en particulier:

- Réexécuter l'instruction fautive de sorte que si l'erreur est transitoire (voir III.3.1.1), sa disparition éventuelle à la nouvelle exécution donnerait lieu à un traitement normal.
- Mais si au contraire on considère que d'après le concept même de l'exception, il n'est pas possible d'assurer, à l'occurrence de celle-ci, le service demandé alors sa récupération peut se résumer comme suit:

- Restauration éventuelle d'un état cohérent de la procédure appelée;

- Propagation de l'exception vers l'appelant qui se traduira par l'activation de traitant(s) associé(s) où une tentative éventuelle de remplacement du service demandé par un autre substituable (masquage) sera effectué. Si aucun masquage n'est prévu, ou que toute tentative se termine par un échec (occurrence de nouvelles exceptions), il y a de nouveau restauration et propagation vers l'appelant du niveau supérieur.

IL est montré dans [CRI79] que cette dernière stratégie nommée REPROME (Restauration et Propagatiopn suivi d'un Masquage Eventuel) est plus efficace que la première désignée par: MARRE (Masquage et Reprise sinon Restauration Eventuelle).

II.2.2. Approche Implicite.

Le respect de la règle de base de l'approche explicite, à savoir: " un traitant pour chaque opération susceptible de signaler une exception " s'avère difficile à maintenir dans bien des cas. En effet, sachant qu'à l'appel d'une procédure l'opération réalisée par cette dernière ne peut en aucun cas générer d'exception, il est inutile de lui associer un traitant. De même, l'activation d'un traitant peut générer des exceptions et les traitants de ces dernières peuvent en générer d'autres, d'où un nombre important de traitants explicites à définir, nombre qui compliquerait d'avantage l'écriture des programmes et qui, par conséquent diminuerait leur fiabilité.

Il est également des situations où les préconditions des exceptions (qui peuvent être dérivées par négation de la précondition de l'opération qui fournit le service attendu) ne sont pas toutes identifiées pour pouvoir leur attacher de manière explicite des traitants appropriés.

Aussi, toutes ces situations peuvent conduirent à des détections d'occurrences d'exceptions pour lesquelles il n'existe pas de traitants associés au niveau des procédures appelantes, c'est cette classe d'exceptions que l'on désigne par exception-

défaillance [CRI79,GOO,LIS74,AND76,MEL76]. Un traitant par défaut est attaché à ce type d'exceptions et activé par le mécanisme de tolérance aux exceptions lorsque ce dernier ne peut identifier un traitant prévu et désigné pour une exception donnée.

Peut-on éviter les exceptions-défaillances? Eviter ce type d'exceptions dans un système revient à prévoir le traitement de toute violation dynamique possible d'un invariant qui caractérise le type d'une variable, ce qui implique:

1. Connaissance et spécification des propriétés invariantes de tous les types de variables d'un système;
2. Spécifier pour chaque type toutes les exceptions possibles et prévoir les tests dynamiques qui détectent l'occurrence de ces exceptions;
3. Prévoir pour chaque activation de procédure autant de traitants explicites que d'exceptions possibles.

Pratiquement, le respect de ces trois points est difficilement réalisable (complexité des programmes, coût élevé) ce qui contraint à admettre l'importance du traitement par défaut des exceptions, d'où l'intérêt que revêt une utilisation combinée avec le traitement explicite afin de pouvoir accroître d'avantage la fiabilité des logiciels [JAL 84].

Qu'il soit explicite ou par défaut, le traitement des exceptions se caractérise par l'objectif de rendre atomiques les actions exécutées par le système en dépit des occurrences d'exceptions (prévues ou non), et les mécanismes développés se consacrent à résoudre les problèmes de la manière suivante:

- Tentative de masquage de l'occurrence de l'exception ou défaillance;
- Restauration d'un état cohérent éventuel de la procédure appelée;
- Propagation de l'exception (ou défaillance) vers l'appelant.

Le mécanisme des blocs de reprise [HOR74,RAN75-78,AND76-81] conçu pour le traitement par défaut des exceptions, constitue un exemple type de stratégie capable de résoudre les trois problèmes précités à savoir: le masquage, la restauration de l'état cohérent et propagation vers l'appelant.

Le problème du masquage qui nécessite la connaissance de la sémantique de l'opération qui effectue le service demandé peut être résolu dans ce mécanisme grâce à une déclaration d'un ou plusieurs algorithmes (appelés "alternants") en prévision d'une exception-défaillance de l'algorithme "primaire" désigné pour assurer le service standard. Les alternants sont prévus pour fournir le même résultat que le primaire avec toutefois une diminution acceptable de performance.

La détection de défaillance est accomplie par un test de validation du résultat obtenu à la fin de chaque exécution d'algorithme. Si toutefois le résultat est invalidé par le test, un mécanisme de restauration (recovery cache [LEE79]) de l'état avant l'entrée dans le bloc défaillant, crée l'environnement d'entrée adéquat dans le bloc alternant (problème de restauration). Lorsque toutes les tentatives de masquage se terminent par un échec, c'est-à-dire que l'exception persiste après l'exécution du dernier bloc alternant, un signal de défaillance assure la propagation vers le bloc englobant ou l'appelant (problème de propagation).

La tolérance aux exceptions-defaillances implique [MEL77] au moins la résolution des problèmes de restauration d'état cohérent, et de propagation vers l'appelant, ce qui attribue aux mécanismes qui s'y adaptent, une caractéristique importante représentée par la possibilité de pouvoir toujours déterminer un état cohérent d'une opération invoquée sans avoir à connaître la sémantique de cette dernière.

Dans les systèmes transactionnels [BAL84, LAM81], une attention particulière doit être consacrée au problème de latence de détection des exceptions-défaillances, qui est susceptible d'introduire une incohérence lors de la restauration d'état d'une transaction avortée.

En effet, si une transaction Ti s'est terminée anormalement avec une exception non détectée et laissant un état incohérent Ei, la détection d'une exception lors de l'exécution de la transaction Ti+1 implique une restauration de l'état correspondant à celui d'avant l'exécution de Ti+1, donc Ei (incohérent).
Dans ces conditions, le traitement par défaut n'est adéquat que pour des temps de latence de détection ne dépassant pas les

intervalles de temps séparant l'exécution de transactions successives. Les améliorations sont donc orientées vers le progrès à faire au niveau de la détection des exceptions [BES 81].

Une étude synthétique de ces deux approches est présentée dans [CRI79-80], elle s'adresse notamment aux programmes modulaires [PAR 72] en mettant l'accent sur les niveaux d'abstraction inférieurs au niveau processus. Ce dernier n'étant pasè d'ailleurs abordé; une utilisation combinée des deux approches y est proposée.

La présente section a pour but d'introduire les mécanismes de base de tolérance aux fautes logicielles, mécanismes généralement intégrés dans les langages de programmation, donc faisant partie intégrante de l'application elle-même. Elle sert de lien explicatif pour aborder les problèmes de tolérance aux pannes dans un contexte de systèmes de processus intéractifs relativement aux pannes d'origine matérielle, et auxquels s'ajoutent les problèmes non moins compliqués relatifs à la détermination d'un état cohérent global au système [FIS 83].

Dans les mécanismes linguistiques de traitement d'exceptions, on s'efforce de respecter l'atomicité des actions [BES80, LOM77, LAM81, LIS82] qui réalisent les services sollicités de sorte à confiner les dommages subits à l'occurrence d'exception. Ce confinement doit pouvoir atteindre l'ensemble des processus communicants dans un système distribué et par conséquent pouvoir étendre le concept de l'atomicité au niveau de la communication et l'exécution des processus.

Les deux axes fondamentaux suivis par la tolérance aux pannes viennent d'être rappelés tout en n'évoquant essentiellement que les pannes d'origine logicielle.

Or tout système informatique est intrinséquement formé d'un système abstrait (logiciel) et d'un système physique (matériel) en étroite corrélation, ce qui nous amène logiquement à considérer désormais les pannes matérielles.

On se propose alors dans la section suivante d'examiner les deux types de pannes, les possibilités et les capacités de reprise suivant les caractéristiques du système considéré.

Chapitre III

MECANISMES DE REPRISE, PANNES MATERIELLES ET FAUTES LOGICIELLES

III.1. Introduction.

La diversité des objectifs des systèmes et de leurs contraintes d'une part, les caractéristiques diverses des causes de leurs fonctionnements anormaux d'autre part, font qu'il est trés complèxe dans l'état actuel de la recherche de pouvoir mettre au point des mécanismes de reprise capables de recouvrir les différentes sources d'anomalies (complexité et coût exorbitant).

Aussi a-t-on défini des mécanismes qui s'adaptent à des situations et des classes de systèmes donnés, et se bornant à résoudre les problèmes de reprise relatifs à des causes d'erreurs données en appliquant des moyens appropriés (logiciels, matériels).

C'est cette complexité croissante et le coût encourru qui ont contribué à ce que l'étude de la reprise des systèmes après pannes ait manqué de généralité, et s'est souvent définie beaucoup plus par rapport à un domaine précis, matériel ou logiciel, que par rapport à l'union des deux.

On constate donc un manque d'homogénéisation de l'aspect matériel et logiciel dans les mécanismes de reprise après défaillance des systèmes, alors qu'il est admis [LAN77] qu'on ne peut dissocier les effets des pannes dûes au matériel des effets engendrés par les erreurs logicielles.

Avant d'examiner les mécanismes de reprise, il est important de distinguer les différentes sources d'anomalies pouvant altérer le fonctionnement correct d'un système.

III.2. Pannes et erreurs d'origine matérielle et logicielle.

III.2.1. Terminologie.

Pour éviter une ambiguïté dans la terminologie utilisée pour désigner ce qui peut affecter l'exécution normale d'un système pouvant provenir du matériel ou du logiciel, on définit les notions fondamentales suivantes [SIE82,LAP85,AND81,COU81].

a) Défaut (defect): Un changement physique du matériel.

b) Panne ou faute (fault): Etat erroné du matériel ou du logiciel résultant de défauts de composants, d'interférence physique de l'environnement ou d'une conception incorrecte (logiciel,matériel).

c) Erreur (error): Manifestation d'une panne dans un système. Lorsqu'une panne se produit, l'erreur qui en résulte est dite Latente. Lorsque le composant contenant la panne est activé, l'erreur devient effective.

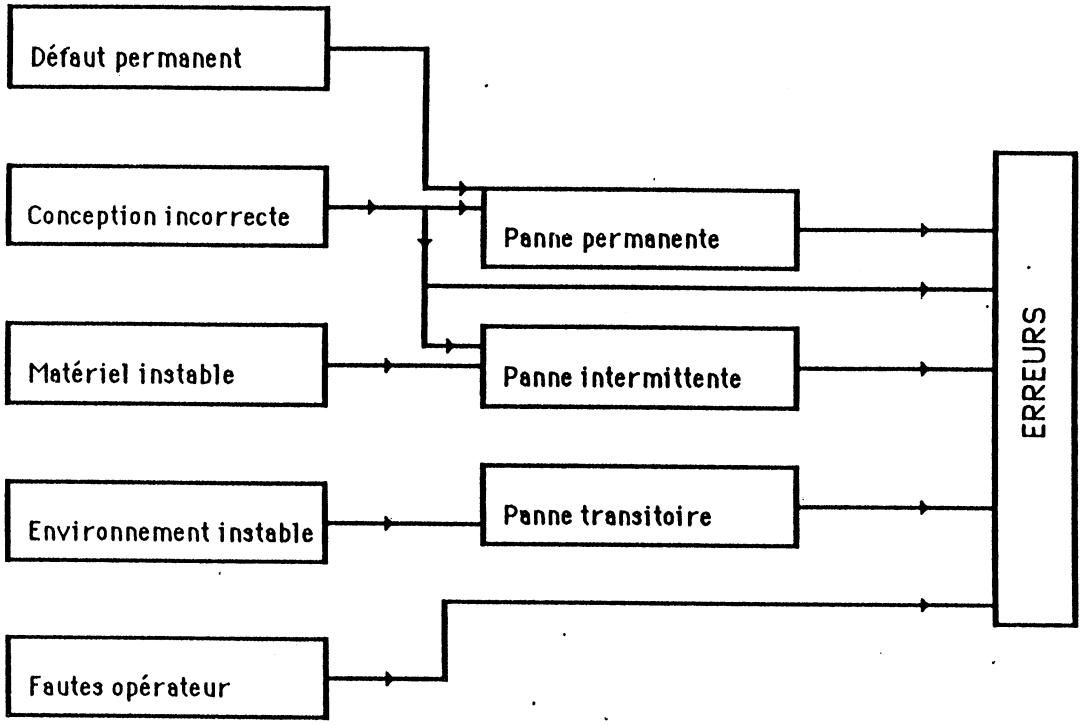
d) Défaillance (failure): Manifestation d'une erreur de sorte que le service fourni par le système ne soit plus conforme aux spécifications prévues.

Permanent: Qualifie un défaut, une panne ou une erreur stable et continu, (un défaut permanent indique un changement physique irreversible du matériel).

Intermittent: Qualifie une panne ou une erreur dont l'occurrence occasionnelle est dûe à un matériel instable ou aux changements d'état du logiciel.

Transitoire: Qualifie une panne ou une erreur résultant des conditions temporaires de l'environnement.

III.2.2. Sources d'erreurs.



sources d'erreurs [SIE82]

Figure 01

La figure 1 donne les diverses sources d'erreurs et permet d'établir une certaine "hiérarchie" dans le traitement des défaillances.

Relativement à une origine matérielle, on peut observer trois niveaux.

- un niveau défaut (le plus bas),
- un niveau panne (intermédiaire),
- un niveau erreur (le plus haut).

Du point de vue externe au système on observe le niveau défaillance perçu par l'utilisateur.

Par contre, en considérant l'origine logicielle, on ne distingue que deux niveaux internes:

- niveau panne ou niveau faute (niveau bas),
- niveau erreur (niveau haut).

Au niveau erreur, la distinction entre origine matérielle et origine logicielle n'est pas toujours réalisable sauf cas particulier

où un masquage matériel est disponible. Par exemple dans un TMR(Triple Modular Redundancy) [SIE82], une violation d'une propriété invariante a une forte probabilité de ne provenir que du logiciel. Donc dans un cas général, si aucun disposif n'est prévu, une erreur ne peut être caractérisée de manière précise.
De même l'échec d'un bloc de reprise à plusieurs alternants a une forte probabilité de provenir d'une erreur matérielle. Dans la suite on associera le terme erreur à une origine logicielle, et le terme panne à ce qui provient du matériel (ceci pour faire une distinction entre les deux origines bienque cela contredise légérement la terminologie donnée en III.2.1.).

III.2.3. Types de pannes et leur tolérance.

Deux types de pannes sont considérées:
- les pannes permanentes,
- les pannes transitoires.

Les pannes intermittentes entrent dans la catégorie des pannes transitoires car leurs effets sont semblables, et leur distinction lors de la détection n'est pas possible.

La tolérance aux pannes varie en fonction du type de ces dernières, ainsi pour des pannes transitoires, elle s'accomplit en deux étapes:
- détection de la panne,
- reprise proprement dite.

Pour les pannes permanentes, leur traitement s'achève en quatre étapes:
- détection,
- localisation et reconfiguration matérielle,
- reconfiguration logicielle,
- reprise.

III.3. Mécanismes de reprise.

III.3.1. Caractéristiques.

Un mécanisme de reprise se caractérise par:
- la classe d'erreur à traiter;
- la classe de systèmes auquelle il est destiné;

- son indépendance vis-à-vis du programmeur d'application;
- son coût.

III.3.1.1. Classes d'erreurs à reprendre.

On peut distinguer deux classes d'erreurs suivant les sources qui les génèrent:

III.3.1.1.1. Les erreurs logicielles (ou exceptions).

Elles sont le produit d'une conception incorrecte d'un logiciel, deux approches ont été développées pour y remédier. il s'agit de l'approche implicite et l'approche explicite précédemment vues au chapitre II, dont une utilisation conjointe est appliquée dans le mécanisme de la S-CONVERSATION qu'on verra en III.4.1.2.

Ces erreurs qui dépendent essentiellement de la fiabilité du concepteur contraignent ce dernier à préparer leur traitement (détection et reprise) par des procédés entièrement logiciels, en accordant toutefois une grande confiance au matériel.

Un des problèmes importants de cette catégorie est sans doute celui de la détection, car elle peut s'avérer coûteuse ou latente. Coûteuse parcequ'il faut prévoir pour chaque erreur possible un test de détection approprié (test d'assertion), et un traitant de récupération. Latente, parcequ'elle peut être l'expression d'une spécification pouvant être ambiguë, incomplète ou contradictoire [GUT 80].

Exemple d'erreur logicielle (exceptions):

Soit un type abstrait PILE, deux opérations de manipulation de ce type. Opération EMPILER et DEPILER. L'algorithme réalisant ces deux opérations avec détection d'exceptions peut être:

```
Type PILE [1..N];
Entier S;
Procedure EMPILER (M:entier)
Debut
        Si S > N  alors        (*)
           SIGNAL Overflow;
        Sinon
            Debut
                PILE [S]:= M;
                S:= S + 1;
            fin
        finsi
Fin EMPILER.


ext fonction DEPILER : entier
    Debut
        Si S < 1 alors      (+)
           SIGNAL Underflow;
        Sinon
            Debut
                DEPILER := PILE [S];
                S := S - 1;
            fin
        finsi
Fin DEPILER.
```

(*) et (+) représentent des tests de préconditions d'occurrences d'exceptions. Si ces assertions sont vraies, alors une exception est détectée et signalée par l'activation du traitant associé; Overflow pour (*) et Underflow pour (+).

III.3.1.1.2. Ambiguïté des erreurs matérielles et logicielles.

Les erreurs quelles soient matérielles ou logicielles se situent au niveau le plus haut de la hiérarchie des anomalies conduisant à un mauvais fonctionnement d'un système. Aussi lorsqu'elles sont détectées, leur correction possible est associée au niveau plus bas (i,e: niveau panne ou faute).

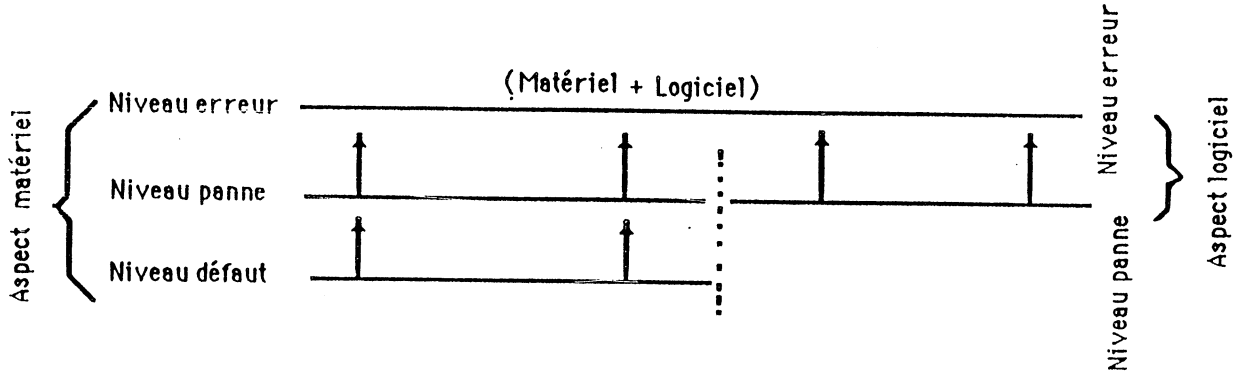
Par exemple, un utilisateur peut mettre à profit le mécanisme de déroutement d'un processeur (qui représente un mécanisme d'exception) pour simuler des instructions inexistantes. Ainsi lorsque cet utilisateur emploie dans son programme une instruction inexistante dans le répertoire d'instructions du processeur, il y a faute logicielle dont l'erreur latente produite devient effective lors de l'exécution de l'instruction.

Une détection de ce type d'exception peut engendrer l'alternative suivante:
L'exception est anticipée, alors un traitant explicite lui est associé; il a pour but de simuler l'instruction inexistante.
L'exception est imprévue (ou exception défaillance), un traitant par défaut est activé pour terminer le programme et fournir un diagnostic à l'utilisateur.

De même, lorsqu'un défaut physique se produit, un court-circuit par exemple, au niveau logique ceci se traduit par une panne matérielle assimilable à un collage à 0 ou 1 dont l'effet sur le niveau hiérarchique haut peut se traduire par une altération des variables ou code du programme. Cette altération du code peut provoquer une exception du type précédent (instruction inexistante) pour laquelle aucun traitement ne peut y remédier car elle a pour origine le matériel.

Donc une détection au niveau erreur est possible grâce aux tests d'assertion dont le traitement peut s'avérer inutile, car l'erreur est matérielle mais confondue avec une erreur logicielle.

Puisque la distinction à ce niveau, entre deux types d'erreurs, peut être ambiguë, la détection des pannes matérielles comprendra implicitement celle des erreurs qu'elles engendrent (cf fig. 02).



Hiérarchie des niveaux des causes d'erreurs.

Figure 02

Une panne matérielle peut conduire à la fois à un changement d'état du matériel et du logiciel pouvant dégénérer en un mauvais fonctionnement des systèmes, alors que l'étude de la tolérance a souvent tendance à considérer un des aspects (logiciel ou matériel) au détriment de l'autre pour lequel les problèmes sont supposés "résolus".

III.3.1.1.3. Pannes Matérielles.

Elles se situent à un niveau intermédiaire entre le niveau défaut et le niveau erreur, elles se répartissent en deux catégories: les pannes transitoires, et les pannes permanentes.

a) Les pannes transitoires.

Elles sont issues d'une combinaison de phénomène local (tels: décharges électrostatiques, tension de lignes, distribution thermique), et d'un phénomène universel (rayons cosmiques, radioactivité résiduelle..). L'irrégularité de leur occurrence rend difficile la construction d'un modèle propre. Il est également difficile de localiser la source de la panne.

Leur traitement est plutôt particulier puisque n'ayant pas subi de dommages irreversibles, le composant matériel passe successivement d'un état de fonctionnement normal à un état de fonctionnement anormal et réciproquement.

Ce basculement d'états peut provoquer un état erroné de composants logiciels qui restera inchangé tant que des actions prévues à cet effet ne soient pas entreprises. Ce traitement doit donc s'effectuer au niveau logiciel et peut être réalisé par des procédés entièrement logiciels (voir suite). Puisque leur apparition est liée à l'environnement et les interactions entre composants, les pannes transitoires occupent une place importante dans le domaine des circuits à haute intégration.

L'importance de la fréquence de leur occurrence (environ 90% des cas [Sie82]) milite en faveur de la construction de mécanismes de traitement adéquat pour limiter leurs effets. On verra dans III.4. des exemples de tels mécanismes.

b) Les pannes permanentes.

Issues d'un défaut physique permanent, elles perturbent le fonctionnement normal d'un composant jusqu'à remplacement ou réparation de ce dernier.

La régularité de leur occurrence permet de construire des modèles appropriés afin de mieux les caractériser [Lan77]. Leurs effets au niveau le plus élevé (erreur) se traduit par une perturbation du logiciel qui perd une partie ou la totalité de services assurés. Leur traitement dépend donc des dommages subits, et varie selon le système considéré.

Par exemple dans un système distribué, l'arrêt total d'un site peut influer sur le rendement global du système tout en ayant la possibilité d'assurer les services prévus sur le site défaillant. Cette opération peut être accomplie par redistribution des processus interrompus sur un site opérationnel du système.

Dans un réseau local où les sites sont constitués de micro-ordinateurs, on peut raisonnablement considérer qu'une panne permanente suffit pour déconnecter le site affecté, et effectuer une reconfiguration [BAR 82] matérielle et logicielle adéquate.
Cette opération a pout but de faire accomplir les services entamés sur le site défaillant, par un site opérationnel du réseau.
Cette migration de processus d'un site (ou station) vers un autre, pour assurer une continuité dans le déroulement d'une application

en dépit de pannes, correspond précisemment au but de notre étude (voir stratégies).

Remarque.

Des dispositifs appropriés sont conçus pour détecter les pannes matérielles, cependant il est difficile de déterminer s'il s'agit de panne permanente ou transitoire.
En général, lorsqu'une panne est détectée et diagnostiquée , elle est considérée transitoire et le mécanisme de reprise associé est alors activé. Si elle persiste après plusieurs tentatives, elle est alors prise pour une panne permanente et traitée comme telle.

III.3.1.2. Particularité des systèmes à reprendre.

La reprise dans un système dépend de deux facteurs essentiels:
- le type du système,
- le type de la panne.

III.3.1.2.1. Type de systèmes.

Le problème et les capacités de reprise diffèrent selon le type du système que l'on veut doter d'un mécanisme de tolérance aux pannes. On y distingue:
- les systèmes monoprocesseurs,
- les systèmes multiprocesseurs,
- les systèmes distribués ou réseaux [COR 81].

III.3.1.2.2. Type de pannes

Tout d'abord les pannes logicielles (ou fautes) ou plus précisément les erreurs logicielles permettent une implémentation de mécanisme de reprise quelque soit le type du système considéré, puisqu'elles n'ont aucune influence sur le matériel. Une conception incorrecte peut conduire à une incapacité d'assurer les services attendus malgrés un matériel en parfait état, la réciproque n'étant pas vraie (sauf sur système distribué).

Les pannes matérielles transitoires sont récupérables dans tout type de système au même titre que les erreurs logicielles mais selon des stratégies différentes.

Quant aux pannes permanentes, elles peuvent exclure les systèmes monoprocesseur puisqu'elles peuvent reclamer une redistribution de processus sur un matériel fonctionnel différent de l'original. Cette caractéristique de grande disponibilité est largement offerte par les systèmes distribués qui, de ce fait, se prêtent mieux aux possibilités d'implémentation de grandes capacités de tolérance aux pannes.

III.3.1.3. Intervention du programmeur d'application.

Une des caractéristiques fondamentales de la reprise en erreurs logicielles est la contribution du programmeur pour élaborer les mécanismes de reprise qui font partie intégrante de l'application elle-même (intégrés dans les langages de programmation).
On peut citer pour cela la technique des blocs de reprise appliquée au langage pascal [SHR79], le schéma de conversation dans [RAN75] et la S-conversation [JAL84], et plus généralement les mécanismes basés sur l'approche explicite (traitement des exceptions).

Les mécanismes développés pour reprendre après pannes matérielles se distinguent par leur indépendance vis-à-vis de l'application et libèrent donc le programmeur d'une tâche encombrante, d'autant plus que son intervention n'est pas toujours exempte de création de situations difficiles susceptibles de contredire les spécifications prévues dans le mécanisme lui-même.
Ces problèmes seront évoqués lors de l'étude de quelques mécanismes déjà existant.

Selon l'importance de la contribution du programmeur à l'élaboration des éléments définissant l'environnement de reprise, on peut classer les mécanismes en deux catégories:

- Les mécanismes transparents au programmeur tels ceux décrits dans [WOO81, KIM79-80-82, BAR83], et
- Les mécanismes dépendant du programmeur, en général, ceux qui sont intégrés dans des structures linguistiques tels le mécanisme des blocs de reprise et ses dérivés.

III.3.1.4. Couverture de panne et de recouvrement.

Cette notion se base sur l'estimation des informations altérées dans un système à la suite d'une panne. Elle traduit la confiance accordée au mécanisme de reprise traitant la panne d'avoir recouvré un état cohérent du système après l'opération de reprise. Elle utilise la notion de latence de détection de l'erreur (D20) et celle de la propagation de reprise qu'on verra dans la suite (étude des stratégies).

III.3.2. Stratégies de reprise.

Dans le chapitre II on a vu deux approches fondamentales de reprise, approche explicite et approche implicite. Si la reprise des erreurs logicielles s'adapte aux deux approches, celle des pannes matérielles par contre, ne convient qu'à l'approche implicite, car le masquage des erreurs produites dans ce cas ne peut se faire que par une réexécution du processus erroné. cette réexécution doit se faire à partir d'un point de reprise (D1) estimé exempt d'erreur, et antérieur à l'occurrence de la panne.
Pour cela, on adoptera dans toute la suite de l'étude l'approche implicite.

III.3.2.1. Reprise d'un processus indépendant.

a) Cas d'une erreur logicielle.

Lorsqu'une erreur est détectée (par un test d'assertion), le processus erroné ne pouvant progresser dans son évolution, se voit donc retourner en "arrière" à un point de reprise à partir duquel il tentera de masquer cette erreur.
Toutefois, puisque l'erreur est dans le code du processus, seule un code de substitution rendant un service identique est capable de réaliser un tel masquage (technique de blocs de reprise).
Schématiquement cette technique peut être représentée par:

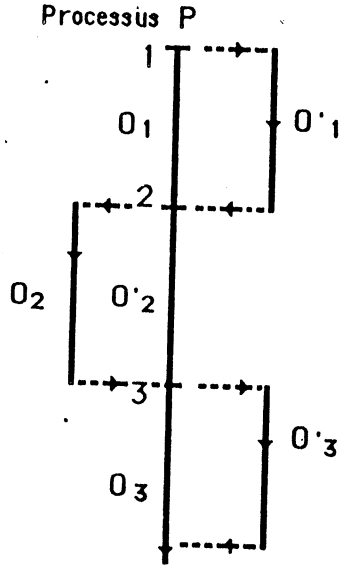


Schéma de blocs de reprise.

Figure 03

Lorsqu'une opération Oi qui implémente l'algorithme standard Ai est interrompue pour cause d'erreur, une opération Oi' implémentant l'algorithme alternant Ai' de Ai est alors exécutée.

La structure de blocs dans les langages de programmation adoptée à cette technique a donnée lieu au mécanisme des blocs de reprise [HOR74,AND76,SHR78] comme l'illustre la forme syntaxique suivante.

ENSURE < Assertion >

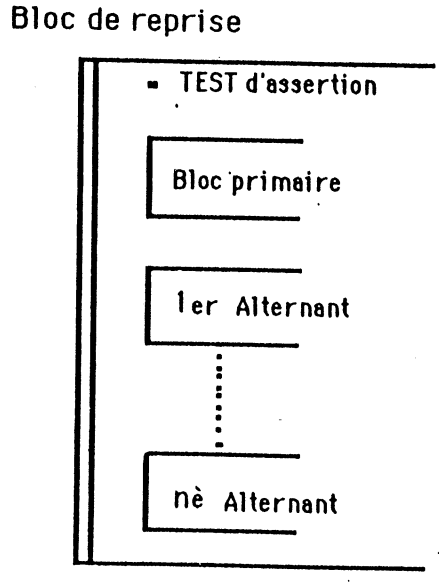
BY < Bloc primaire>

ELSE BY < 1ER Alternant>

⋮

ELSE BY < nè Alternant>

ELSE ERROR



Structure des blocs de reprise.

Figure 04

Cette structure de contrôle consiste en deux composants principaux:

1. Le test d'assertion qui doit être satisfait à la sortie du bloc de reprise (l'expression booléenne est évaluée à vrai)
2. un ensemble de blocs alternants, "réserve", du bloc primaire en cas d'erreurs.

L'exécution d'un bloc est validée par un test d'assertion; si l'assertion est vraie alors on continue en séquence, sinon un mécanisme automatique de restauration de l'état du processus à l'entrée du bloc de reprise est déclenché [LEE 80], puis le bloc alternant suivant est exécuté. Dans le cas où l'erreur persiste après exécution du dernier alternant, un signal d'erreur est émis vers l'exterieur: Soit vers un bloc englobant s'il en existe, ou termine anormalement le programme.

Une variante de ce schéma de blocs de reprise pour implémentation d'une tolérance aux fautes dans des applications temps-réel, basée sur le principe des échéances pour introduire un alternant, est proposée dans [CAM79,WEI80] le vocable de " deadline méchanism ".

b) Cas de pannes matérielles.

Qu'il s'agisse de pannes transitoires ou permanentes, l'état du processus affecté devient incohérent, il n'est pas avantageux de tenter de corriger cet état erroné mais plutôt de défaire ce qui a été fait depuis l'instant t d'occurrence de la panne. Donc une réexécution du processus erronné à un instant t' antérieur à t (appelé point de reprise: Rp) est nécessaire.
Les deux types de pannes ont une phase commune dans l'opération de reprise à savoir, la recherche d'un état cohérent du processus.

III.3.2.1.1. Rôle d'un mécanisme de reprise.

III.3.2.1.1.1. Panne transitoire.

Deux éléments sont essentiels pour accomplir un recouvrement d'erreur.

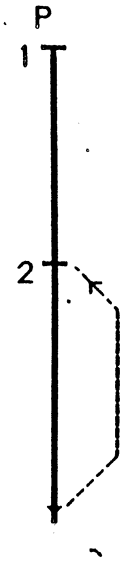
1- Un contexte représentant l'état du processus défaillant et associé à un Rp (D1). Ce contexte supposé inaltéré comprend les registres, les variables et les buffers de transmission.

2- Le code du processus.

L'action du mécanisme se traduit alors par:

- Restauration de l'état du processus à un point de reprise choisi.
- Réexécution du processus affecté.

La figure suivante illustre cette action.



Pi: point de reprise du processus P. Après détection de la panne, le mécanise fait reprendre P en P_2 (on suppose que la latence de détection est suffisamment faible pour que le contexte associé à P_2 soit non erroné.

Reprise d'un processus indépendant.

Figure 05

III.3.2.1.1.2. Panne permanente.

C'est le type de panne qui exige une reconfiguration matérielle sur un environnement approprié (réseau). Les éléments indispensables pour une opération de reprise sont:

- Une station (site) de reprise prévue pour relayer la station défaillante (voir section IV).
- Le code du processus à reprendre, et au moins un point de reprise (plus précisemment le contexte associé), l'ensemble étant communiqué avant la défaillance.

Le mécanisme de reprise travaille en réparti, une méthode distribuée de détection de ce genre de panne est impérative [PRE67, HOS83].
Ce mécanisme une fois lancé, effectue les mêmes actions qu'en panne transitoire ce qui signifie qu'après disponibilité des éléments adéquats , le mécanisme agit de la même façon dans les deux types de pannes. Cette analogie sera exploitée plus loin pour concevoir un même mécanisme de reprise traitant aussi bien les pannes transitoires que les pannes permanentes.

Remarque.

Le rôle d'un mécanisme tel qu'il vient d'être décrit, c'est-à-dire en considérant un processus unique est en fait un cas particulier s'adressant aux systèmes monoprocesseurs.
Le cas le plus général qui nous préoccupe est la reprise dans un système distribué où l'interaction interprocessus joue un rôle fondamentale, et soulève le problème complexe de la détermination d'un état cohérent (D9) regroupant un ensemble de processus communicants.

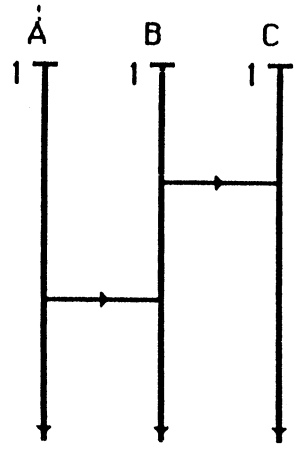
III.3.2.2. Reprise de processus communicants.

III.3.2.2.1. Propagation de reprise.

Soit P un ensemble de processus communicant par messages (msg), cette communication interprocessus est née d'un besoin de coopération pour la réalisation d'une tâche commune.
Ainsi un processus p de P élabore des informations qu'il transmet à un ou plusieurs de ses partenaires pj . Lorsque p est altéré par une panne, les informations produites par p et consommées par pj peuvent l'être aussi (révocables D12) alors la reprise de p pour éliminer les effets de la panne doit entrainer celle de tout partenaire pj ayant reçu au moins un msg de p. C'est ce qu'on appelle une propagation de reprise de p vers pj.

Cette propagation a pour but de défaire toutes les actions susceptibles d'être contaminées par les effets de la panne; autrement dit couvrir l'étendue des dommages causés à P.

Exemple: soit P = < A,B,C > .



Propagation de reprise.
Figure 06

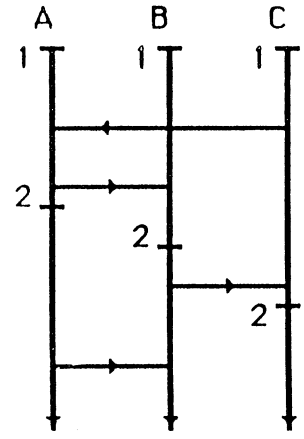
Lorsque le processus A est affecté par une panne, il "reprend" à son Rp A1. Il provoque la reprise de B en B1 qui à son tour propage cette action vers C qui reprendra en C1. Ainsi l'action globale effectuée par P se trouve annulée.

III.3.2.2.2. Détermination d'une ligne de reprise.

La propagation de reprise suit le flot d'information dans P jusqu'à atteindre un point de stabilité (arrêt "naturel") qui correspond à un état cohérent de P. Cet état cohérent ou ligne de reprise (D9) peut dans certain cas être obtenu au prix d'une révocation importante d'activités des processus de P, ce qui engendre un coût de l'opération de reprise inacceptable.
Il est donc indispensable de pouvoir éliminer ces situations déplaisantes en contrôlant la propagation.

L'origine de ces situations de révocation importante d'activité est dûe principalement à une liberté de communication entre processus et de l'établissement de leurs points de reprise. Donc contrôler la propagation de façon à atteindre une ligne de reprise (RL) avec un coût minimal, revient à contrôler ou coordonner la communication ou la création des Rp des processus.

Exemple de détermination de RL.



Reprise avec propagation dégénérant
en effet-domino.
Figure 07

La liberté accordée aux processus A,B,C d'évoluer librement, peut provoquer lors d'une reprise, une situation onéreuse où tout le travail effectué par A,B,C serait annulé (effet-domino: D15). C'est notamment le cas d'une panne affectant A ou B; la RL ainsi déterminée serait alors: <A1,B1,C1>.

On peut distinguer deux manières possibles de détermination de RL:

- Détermination dynamique (W0080), où la RL n'est pas connue a priori, elle est atteinte par un roll-back progressif de l'ensemble des processus impliqués dans la reprise. Par exemple figure 7, lorsque A tombe en panne , il reprend en A2 en propageant la reprise vers B qui reprend en B2 lequel la propage vers C, qui reprend en C1, puis A en A1 et B en B1. Le processus A a donc exécuté deux retours arrière ou roll-backs successifs A2,A1; C en a effectué un seul C1, et B en a fait deux: B2,B1.

L'inconvenient de la méthode revient au coût éventuellement important et les réexécutions inutiles possibles des processus (les relances en A2,B2,C2 sont inutiles). Il est cependant possible de limiter ces dernières par un roll-back direct (voir section roll-back direct), son avantage réside dans la liberté accordée aux

processus lors de leur évolution.

- Détermination statique. Dans ce cas la RL est prédéterminée. Pour ce faire, des contraintes sont imposées aux processus de manière à contrôler la propagation en cas de panne et subir un coût approximativement estimé à l'avance.
  Le schéma de la conversation [RAN75] et s-conversation [JAL 84] sont autant de mécanismes qui adoptent cette méthode, où les capacités de reprise sont étudiées de façon globale pour l'ensemble des processus intéractifs.

III.4. Description de quelques mécanismes existants.

III.4.1. Mécanismes dépendant du programmeur.

III.4.1.1. Conversation.

Ce mécanisme de reprise et de contrôle de propagation est basé sur la notion de blocs de reprise décrite précédemment. Il est dédié essentiellement pour la récupération d'erreurs logicielles.

Dans sa forme abstraite, une conversation est un bloc de reprise qui regroupe plusieurs processus et leur fournit des moyens pour coordonner leurs structures de blocs, et des outils de communication interprocessus.
Ainsi lorsqu'il est prévu que des processus vont communiquer durant leurs activités, ils s'organisent selon cette structure de contrôle pour former une conversation.

Après avoir fait partie d'une telle structure, un processus se voit isolé de tout non-participant et ne peut converser qu'avec les partenaires membres de la conversation. Chaque membre doit fournir un point de reprise, un bloc primaire, des blocs alternants, et un test d'acceptance pour valider le résultat obtenu en fin de chaque bloc. Ainsi chaque processus s'exécute indépendemment de ces partenaires mais une synchronisation en fin de bloc, après exécution du test de validation, est nécessaire.

Lorsqu'un processus membre de la conversation détecte une exception ou échoue au test, il proclame ce résultat et l'ensemble des participants abandonnent l'exécution courante pour reprendre à

l'aide d'un alternant. C'est l'ensemble des Rp établis à l'entrée de la conversation par les participants qui constitue la ligne de reprise (prédéterminée), et l'ensemble des tests à satisfaire forme la ligne de test. La ligne de reprise et la ligne de test représentent les frontières (en amont et en aval) de la conversation, et constituent les éléments de contrôle d'une éventuelle propagation.

Cette structure de coordination des communications, restrictive des perturbations et intégrable dans un langage de programmation, suscite certaines remarques:

1- La restriction imposée aux participants de se synchroniser pour quitter ensemble la conversation après satisfaction des tests d'acceptance limite les avantages du traitement parallèles [KIM79-82,RUS77].
Dans le cas contraire, où tout processus ayant passé avec succès son test d'acceptance est autorisé à poursuivre son exécution (sans éliminer son Rp de la ligne de test), il est possible de créer des situations semblables à celles de l'effet-domino. L'alternative est donc laissée au choix du concepteur.

2- L'intersection interdite de multiples conversations pour causes d'interblocage éventuel, et créations de situations irréversibles, implique une participation importante du programmeur pour mener à bien une conversation.

3- Le concept de base des blocs de reprise impose que tous les participants disposent du même nombre d'alternants. La stucture de communication entre blocs des participants étant identique pour tous les algorithmes d'un même bloc, une erreur dans cette structure conduirait à un échec de la conversation.

4- Compte tenu du fait qu'une déclaration de participation à une conversation et une entrée réelle dans celle-ci sont des opérations non nécessairement simultanées, des situations d'interblocage dûes à des désertions éventuelles de participants sont possibles. Un mécanisme de détection de déserteurs s'avère donc indispensable.

5- La déclaration statique de participation implique une connaissance de l'ensemble des membres de la conversation, une communication avec des processus créés dynamiquement pose un problème.

Bien que par essence la conversation ait été proposée comme un outil exprimant des actions atomiques pour réaliser une tolérance aux fautes logicielles, les pannes matérielles transitoires y sont prises en compte implicitement.

Basée sur l'approche implicite, la couverture de reprise fournie lui confère le caractère d'un mécanisme de reprise à grande fiabilité. Comme il a été argumenté dans [MEL77,CRI79], une utilisation conjointe avec l'approche explicite améliorerait d'avantage les capacités de reprise. C'est dans ce sens qu'est née l'idée de la S-conversation qui intègre dans une même structure les deux approches [JAL84]

III.4.1.2. La S-conversation (synchronized conversation) [JAL83].

Le cadre du raisonnement concernant le traitement implicite (backward error recovery) est identique à celui d'une conversation classique. Quant à celui du traitement explicite (forward error recovery), il se base sur la notion d'actions atomiques [LOM77,AND81,LAM81] qui existe déjà dans le concept de la conversation et adapté au traitement des exceptions.

Un processus participant à une action atomique ne peut échanger des informations qu'avec les membres de cette dernière. Si un quelconque partenaire lève une exception, alors chacun de ces participants invoque un traitant de l'exception. Si tous les processus sont capables de masquer cette exception, il y a alors poursuite en séquence de l'exécution de l'action, sinon il y a terminaison anormale et un signal d'exception est généré pour l'environnement externe de l'action défaillante (par exemple pour la s-conversation englobante). On observe plusieurs niveaux de synchronisation des processus:

- A l'entrée de la s-conversation où chaque participant déclare statiquement l'ensemble des membres avec lesquels il est susceptible de communiquer. Ainsi est établie la ligne de reprise.

- Au cours de l'évolution des processus de la s-conversation lorsqu' un des membres détecte une exception.

- A la sortie de la s-conversation, où chaque processus exécute son test de validation de ses résultats. L'ensemble de ces points de test constituent la ligne de test à partir de laquelle est prise la décision de réussite ou l'échec de la s-conversation.

Des points de test sont prévus à l'intérieur des processus, lorsqu'ils sont atteints, il y a attente jusqu'à ce qu'un mécanisme distribué de détection globale qui combine toutes les exceptions détectées localement, ait restitué à chacun des participants une exception commune issue de cette combinaison.
Si l'exception globale restituée est nulle (aucune détection locale n'a eu lieu), alors la conversation synchronisée se termine normalement; dans le cas contraire, des mesures appropriées de reprise sont alors invoquées, où chaque processus exécute un traitant pour tenter de masquer l'exception (forward error recovery).
Le résultat de cette tentative de masquage est examiné au niveau d'une seconde ligne de test et si l'exception rendue par le mécanisme de vote d'exception n'est pas nulle (échec) alors chaque processus effectue une reprise en exécutant un bloc alternant (backward error recovery).

Les exceptions peuvent donc être détectées soit au cours de l'exécution d'un bloc de reprise, auquel cas un mécanisme de vote d'exception est déclenché, soit en fin de bloc par l'exécution du test de validité.

La s-conversation qui n'est qu'une conversation dans laquelle est introduit l'aspect explicite du traitement des exceptions, hérite des mêmes remarques faites pour la conversation classique.

Outre l'avantage procuré par le traitement des exceptions anticipées, des difficultés résultant de cette synchronisation pour déterminer l'exception globale peuvent requérir des moyens adhoc

pour détecter un membre déserteur éventuel.

Une communication du type CSP [HOA 78], tel est d'ailleurs le cadre de présentation de la s-conversation, s'avère favorable à une implémentation de ce mécanisme.

Cependant, le mécanisme distribué de vote qui combine les differentes exceptions localement détectées, pour en extraire une commune, nécessite la connaissance de la sémantique de l'action globale, ce qui exige une fois de plus des efforts du programmeur. Cette synchronisation importante de processus augmente certes les capacités de reprise, mais induit en revanche un coût supplémentaire.

III.4.2. Mécanismes transparents au programmeur.

C'est la catégorie de mécanismes qui demandent le moins possible (ou pas du tout) la contribution du programmeur. La simplification de la tâche de ce dernier est obtenue au prix d'un accroissement d'overhead dans le système. Ces mécanismes font partie intégrante du système.

III.4.2.1. Protocole de reprise de processus intéractifs.

Vis-à-vis de la conversation classique, ce mécanisme de reprise décrit dans [WOO 81] s'applique aux pannes matérielles transitoires. Il se caractérise par une liberté totale accordée aux processus intéractifs tant au niveau de leur communication qu'au niveau de la création de leurs points de reprise. Deux critères constituent l'objectif de ce protocole à savoir:

- Détermination dynamique, par roll-backs progressifs, d'un état cohérent du système après occurrence de panne transitoire.
- Détermination de points de reprise inaccessibles (D8) en vue d'optimiser l'espace mémoire.

L'idée de base de mécanisme, consiste à enregistrer des informations relatives au flot d'échanges entre processus, ou plus précisemment enregistrer les relations de dépendance entre Rp des différents processus. Ces relations seront utilisées (le cas

échéant à l'occurrence de panne) pour propager l'opération de reprise afin de recouvrer un état cohérent du système.

Aucune contrainte particulière n'est imposée à l'évolution des processus, et cela représente une caractéristique importante mais dont l'efficacité est susceptible d'être contrebalancée par les effets néfastes d'une propagation incontrôlée.

A l'occurrence de panne, le processus défaillant, par l'intermédiaire de son historique de communication, propage son opération de reprise à tous ses partenaires qui agiront de la même façon. L'état cohérent de l'ensemble des processus est obtenu par roll-backs progressifs de tous les partenaires impliqués.

III.4.2.2. Mécanisme de communication à travers moniteur tolérant les pannes.

Basé sur la notion de blocs de reprise, donc destiné essentiellement à la reprise après des erreurs logicielles avec toutefois une prise en compte des pannes transitoires, ce mécanisme décrit dans [KIM80] aborde la reprise des processus concurrents en synchronisant la communication à travers les moniteurs [HOA74,KES77,AND83].
Le contrôle de la propagation de reprise y est obtenu en faisant précéder les références aux moniteurs (plus précisemment aux boîtes aux lettres contenant les msg des processus), par la création de Rp supplémentaires à ceux déjà établis à l'initiative du processus lors de son entrée dans un bloc de reprise (fig 9).

Ces Rp additionnels sont établis de sorte qu'ils appartiennent à une ligne de reprise, et garantissent l'arrêt de toute propagation. Cependant, puisque la gestion des Rp implique inévitablement un overhead en temps et espace, ce mécanisme n'est efficace que si le nombre de Rp créés reste dans une limite raisonnable.

Chaque processus est responsable de la détection et la correction de l'erreur dont il est l'origine. Ce qui signifie qu'il ne peut mettre en doute la validité des informations qu'il reçoit de ses partenaires. Il est certain qu'avec cette hypothèse, on a moins de difficultés à prévenir les situations dégérant en grandes perturbations, mais réutiliser les mêmes msg après une reprise

n'exclut en aucune façon la possibilité que ces derniers soient erronés, donc conduire à des actions de recouvrement vaines et reccurrentes.

La communication à travers moniteurs, exploités à des fins de récupération d'erreurs, nécessite leur adaptation pour inclure la notion de boites aux lettres, les structures de données adéquates, et les opérations qui les manipulent.

Du point de vue reprise, le moniteur se comporte comme un processus interactif. Il peut être dans un état révocable à la suite d'une mise à jour par un processus exécutant un bloc de reprise. Cette état de révocabilité dépend de celui du processus, qui à son tour dépend de celui du moniteur par référence à ce dernier (tantqu'un processus n'ait pas franchi avec succès son test d'acceptance en fin de bloc, il se trouve dans un état révocable).

Comme tout processus après défaillance, restaure son état d'entrée du bloc pour annuler l'effet de l'erreur, il est également de même pour le moniteur de retrouver un état exempt d'erreur existant avant la première mise à jour issue d'un bloc défaillant.
Pour faciliter cette restauration, avant la mise à jour du moniteur, le processus sauvegarde dans son propre espace l'état de ce dernier qu'il restaure en cas de reprise.
Un mécanisme d'élimination des Rp inaccessibles permettra de récupérer l'espace utilisé inutilement (non proposé dans [KIM80]).

Les msg d'information, les msg de contrôle de reprise (msg annonçant le résultat d'un test, msg proclamant l'inaccessibilité d'un Rp, msg d'invitation à la reprise...) sont transmis à travers le moniteur, ce mode de communication obéit à l'essence du moniteur qui impose certaines restrictions d'utilisation auxquelles les processus doivent se soumettre sans pour autant nuire à l'objectif d'une reprise efficace. Certains problèmes doivent être pris en compte:

- L'acquisition du moniteur se fait de façon exclusive, une stratégie adéquate de gestion de la file d'attente des processus, ou un mécanisme d'interruption pour forcer l'accès est indispensable pour éviter les situations où, par exemple un processus auquel un msg de reprise ait été déposé dans sa boite aux

lettres, ne peut obtenir l'accès au moniteur.

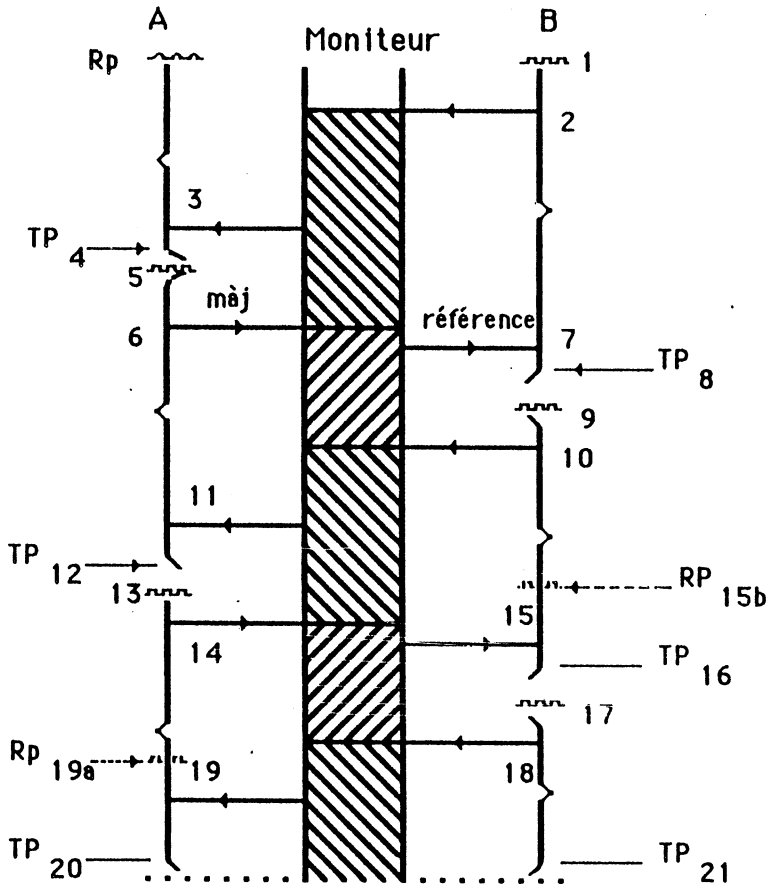
- Selon cette structure du moniteur, certains résultats relatifs à l'opération de reprise seront connus avec un certain retard susceptible de ralentir l'opération dans son ensemble.

Compte tenu d'un nombre acceptable de Rp à gérer, ce mécanisme transparent au programmeur se caractérise par les aspects suivants:

- Accroissement du degré de liberté au niveau de la communication interprocessus (vis-à-vis de la conversation), d'où la possibilité de dialogue avec des processus créés dynamiquement.

- Absence de coordination des structures de blocs, ce qui nécessite moins l'intervention du programmeur, donc restreindre le risque d'erreur qu'il peut provoquer (par exemple introduction d'interblocage).

- Le mécanisme est exempt de propagation incontrôlée (effet-domino) cependant l'étendue des dommages couverts demeure moindre par rapport à la conversation puisque l'on considère irrévocables les msg reçus.



Communication de processus à
travers moniteur [KIM80].

Figure 09

- Un changement de hachures indique un changement d'état du moniteur.

- Une référence au moniteur implique l'exécution d'une procédure moniteur sans changement d'état de celui-ci.

- Une accolade représente l'exécution d'un bloc de reprise, validée par l'exécution d'un test (Tp) associé.

Avant une référence au moniteur, il y a établissement d'un Rp additionnel. Lorsque A ou B échoue respectivement à leurs tests Tp_{20}, Tp_{21}, il y a inévitablement effet-domino.
Si un Rp additif Rp_{15} est établi immédiatement avant la référence B_{15}, un échec au Tp_{20} ferait reprendre A en A_{13}, B en Rp_{15B} et la propagation s'arrête.

Il en serait de même pour un échec de B en TP_{21} qui reprend en Rp_{17}, et par propagation A reprendra en Rp_{19A}; le roll-back est donc de longueur minimun.

III.5. CONCLUSION.

Bien que les mécanismes qu'on vient de voir convergent tous vers le même objectif: doter les systèmes d'outils leur permettant de survivre aux pannes. Cette survie est cependant quelque peu restrictive puisqu'elle concerne soit l'aspect logiciel, soit l'aspect matériel restreint aux pannes transitoires.

Ainsi, sous cette observation, la tolérance aux pannes (matérielles ou logicielles) a tendance à privilégier la notion de systèmes corrects et fiables au détriment d'une notion plus générale de systèmes corrects fiables et d'une grande disponibilité notamment lorsque les possibilités matérielles sont offertes.

Ceci nous amène à considérer le cas de pannes permanentes matérielles issues de défaut physique et réfractaires à toute tentative de recouvrement du genre précédemment décrit. Leurs effets peuvent conduirent à un arrêt du système jusqu'à intervention humaine (réparation). Le service attendu est donc interrompu de façon permanente à moins d'une redistribution sur un autre matériel opérationnel sur lequel ce service se poursuit comme prévu.

Cette catégorie de pannes ouvre donc une dimension nouvelle dans l'étude de leur tolérance, particulièrement dans les systèmes distribués, où le facteur de disponibilité matérielle est très important; On pourra alors parler de systèmes quasiment non-stop.

C'est selon cette optique que sera orientée notre étude pour élaborer des stratégies de reprise pouvant aussi bien traiter les pannes matérielles transitoires que permanentes. Pour cela on se basera sur la "philosophie" présentée dans [WOO81] pour les pannes transitoires auquelle on adoptera un contrôle de propagation et l'aspect disponibilité.

Chapitre IV

STRATEGIES DE REPRISE APRES PANNES MATERIELLES TRANSITOIRES ET PERMANENTES.

IV.1. Préliminaire.

Les stratégies que nous allons étudier dans la suite [ALI 85] s'appliquent particulièrement à un environnement distribué (elles s'adaptent également à un système centralisé relativement aux pannes transitoires).

L'élément de base pour déterminer les relations de dépendance directes ou indirectes entre processus, fondamentales pour connaître l'étendue des activités révocables, est représenté par l'intéraction interprocessus (transit des msg). Ce transit ou flot d'informations est enregistré au fur et à mesure de l'évolution des processus de l'application. C'est son historique qui permettra, par propagation de l'opération de reprise, de déterminer l'ensemble des processus contaminés par la panne et donc annuler les effets de cette dernière. Cette propagation s'avère indispensable pour récupérer un état cohérent du système, mais peut engendrer un coût important si elle n'est soumise à aucun contrôle.

En conséquence, si on souhaite privilégier l'exécution "naturelle" des processus sans pour autant leur imposer des contraintes, la conservation des msg peut constituer une solution partielle au problème du contrôle de la propagation de reprise. Elle peut fournir une solution totale si de plus des critères de non postpropagation (D12) sont satisfaits.

L'efficacité d'une stratégie dépend de la classe d'applications auquelle elle est destinée. Par exemple, une stratégie basée sur la conservation de msg serait moins performante sur une application dont les processus évoluent avec un haut degré d'interaction (cause: overhead créé par la gestion des msg).

La transparence des mécanismes de reprise vis-à-vis de l'application est une des caractéristiques importantes des stratégies étudiées dont l'objectif est double:

- Assurer aux mécanismes qui les implèmentent une autonomie complête sans assistance du programmeur.

- Pouvoir reprendre des processus d'un système distribué suite à des pannes matérielles transitoires ou permanentes.

Si les pannes transitoires peuvent être traitées par des mécanismes entièrement logiciels (WOO81,KIM80-82,HOR74,RAN75) n'exigeant de ce fait que le matériel origine, les pannes permanentes exigent outre un logiciel adéquat, un matériel de substitution approprié (redondance matérielle).

Les systèmes distribués [COR81, LIS82, LAM78] disposant implicitement de cette redondance matérielle, constituent un domaine favorable pour l'étude de moyens logiciels assurant une grande sureté de fonctionnement. Ce domaine peu exploré à l'heure actuelle (un article à ce sujet [BAR 83]) ouvre une voie de recherche dont les résultats obtenus sont loin de satisfaire les possibilités offertes par cette catégorie de systèmes. Autrement dit, l'évolution de ces systèmes doit prendre en compte celle relevant de la reprise pour mettre à profit les capacités de leur disponibilité.

Pour notre étude, on considèrera un ensemble de stations physiques (réseau) communicant par msg, et exécutant chacune (pour simplifier) un seul processus. Lorsque plusieurs processus y sont actifs, les procédures développées pour un seront étendues pour plusieurs.

Les algorithmes élaborés, traiteront les deux types de pannes, et seront identiques pour toute station/processus. Certaines hypothèses y sont nécessaires, notammment:

H1: La latence d'erreur est nulle (D20) sauf indication contraire.
H2: Pas de panne lors de l'opération de reprise.
H3: Le système de transmission est fiable (hypothèse classique).

L'hypothèse H1 est pratiquement difficile à respecter, aussi les conséquences d'une violation possible dans les diverses stratégies et dans un cas optimiste seront considérées. Pour

éviter d'avoir à considérer la reprise d'une reprise, cas complexe et coûteux, on suppose (H2) qu'il ne peut y avoir de panne pendant l'opération de recouvrement. Ce qui signifie, pour pouvoir mieux la respecter, que le temps requis par une opération de reprise devient un facteur important. Cette importance s'observe non seulement au niveau de l'overhead à faire subir à une application, mais aussi au niveau de la confiance accordée au mécanisme d'avoir agi correctement.
Autrement dit, plus le temps de reprise est grand, plus grande est la probabilité d'occurrence de panne durant la phase de recouvrement, d'où l'intérêt de minimiser ce temps [CHA75, GEL78].

Il est également des paramètres que l'on considère pour apprécier l'efficacité et la faisabilité d'un mécanisme de reprise, en particulier:

P1: Taille de l'espace mémoire nécessaire à l'implémentation du mécanisme (structures de données, espace de conservation des msg...)
P2: Longueur des chaines de reprise (résultant des propagations)
P3: Probabilité de reprise (correspondant à l'intervalle optimale entre Rp créés).
P4: Facteur de redistribution des processus.

Les paramètres P1 et P2 caractérisent l'overhead (temps, espace) encouru par un système suite à l'implémentation d'un mécanisme de reprise. Quant à P3, il permet de déterminer la période de création des Rp d'un processus [YOU74, GEL79]. Il permet aussi d'optimiser l'overhead induit par P1 et P2, et également estimer le temps de reprise afin d'éviter une occurrence de panne pendant l'opération de reprise elle-même.

Le facteur P4 n'a de signification que dans le cas de pannes permanentes. Il indique le nombre de fois qu'un même processus peut survivre à une panne, donc le nombre de redistributions (ou de migrations) qu'il peut subir. Il dépend de l'importance accordée à la terminaison normale d'une application. Ce facteur fait intervenir la politique de désignation de station(s) de reprise adoptée, il mesure en fait, le degré de disponibilité dans le réseau.

Comme la base de raisonnement de notre étude est l'historique du flot d'informations échangées entre processus, on considère

celui-ci total [W0081], c'est-à-dire que les informations élaborées et émises par un processus dépendent de celles antérieurement reçues par ce dernier(sauf si latence d'erreur nulle ou critères(1,2) satisfaits [voir stratégies]).

Trois types de stations (dans le réseau) jouent un rôle privilegié, et contribuent à l'élaboration de l'environnement utile au mécanisme qui adopte une des stratégies définies.

- La station défaillante, celle sur laquelle est survenue la panne;
- Une ou plusieurs stations dites de reprise, connues à l'avance ou déterminées aprés panne, et dont le rôle est d'assurer le relais de la station défaillante associée (le nombre de stations désignées caractérise le paramètre P4)
- Les autres stations du réseau ayant eu des échanges d'informations avec la défaillante.

Comme on ignore à priori quelles stations qui tomberont en panne, on considère alors potentiellement défaillante toute station du réseau.

Des stratégies de reprise ont été définies, une étude détaillée caractérisant chacune d'elle fait l'objet de sections dans la suite mais auparavant une description sommaire en guise d'introduction s'avère nécessaire.

IV.2. Description sommaire des diverses stratégies.

Cette section décrit sommairement les différentes stratégies à développer. Une étude plus détaillée consacrée à chacune d'elles fera l'objet de sections qui seront abordées dans la suite.

IV.2.1. Stratégies de base.

IV.2.1.1. Stratégie A.

Dans cette stratégie, la station défaillante conserve elle-même les msg qu'elle reçoit.

L'idée fondamentale de la stratégie réside dans la conservation

des msg reçus. L'intérêt de celle-ci est focalisé sur la minimisation des perturbations créées au sein des processus concurrents lorsque l'un deux est affecté par une panne. Cette limitation de perturbations ne serait appréciable que dans la mesure où le coût qui résulte du nombre de msg à conserver et du traitement inhérent à cette opération reste dans des limites acceptables.

Dans le cas de pannes transitoires, l'opération de reprise consiste à restaurer l'état du processus défaillant à son point de reprise le plus récemment établi, à partir duquel une réexécution est lancée. Cette action est éventuellement entreprise par tout processus ayant reçu au moins un msg du défaillant en deçà du point de relance (D4) de ce dernier; c'est ce qu'on appelle une postpropagation (D12). Cette postpropagation peut être éliminée lorsque des critères prévus à cet effet sont satisfaits (voir section qui étudie la stratégie).

Quant à la rétropropagation (D12), elle n'a pas de raison d'être car le processus repris "relit" localement, aux moments opportuns, les msg nécessaires à la nouvelle exécution.

En raison de la perte des informations disponibles antérieurement sur la station défaillante (msg, graphe de reprise,...), les pannes permanentes exigent une rétropropagation vers tout processus propagateur direct du processus defaillant. Cette opération a pour but de regénérer, pour le compte du processus défaillant, les msg indispensables à la reprise de ce dernier.

IV.2.1.2. Stratégie B.

La stratégie B se caractérise par le fait que la conservation des msg d'informations est assurée par les producteurs de ces msg. En fait, chaque station conserve les msg qu'elle émet pour pouvoir les réémettre ultérieurement à la demande des processeurs qui le désirent (ceux qui ont la charge de reprendre un processus). Cette demande est formulée lors d'une opération de reprise pour acquérir les msg indispensables. Deux cas peuvent se produire:

- Le producteur désigné est capable de satisfaire la demande, car les msg voulus sont disponibles, il y a alors simple réémission.

- Le producteur désigné ne peut satisfaire la demande, car il a été repris antérieurement (panne permanente), il y a alors rétropropagation obligatoire.

Concernant la postpropagation, le raisonnement est identique à celui de la stratégie A.

L'intérêt de la stratégie se manifeste particulièrement dans le traitement des pannes permanentes où la rétropropagation n'est pas systématique, contrairement à la stratégie A. De plus amples détails seront donnés dans la section qui étudie la stratégie.

IV.2.1.3. Stratégie C.

Dans cette stratégie, aucun msg n'est conservé. C'est la stratégie que l'on rencontre souvent dans la littérature spécialisée qui traite les erreurs logicielles, l'absence de conservation de msg lui attribue certaines caractéristiques:

- Elle exige moins d'espace mémoire par rapport aux autres stratégies.
- Elle introduit moins d'overhead durant l'exécution normale d'un système.
- Elle est relativement simple à mettre en oeuvre.
- Elle offre une plus grande couverture à l'élimination des informations altérées.

La systématisation de la rétropropagation à l'occurrence de panne et éventuellement de la postpropagation, constitue l'inconvenient majeure de la stratégie. Son utilisation de manière efficace (en atténuant les perturbations) serait obtenue en imposant une contrainte aux processus, au niveau de la création de leurs points de reprise (voir partie II).

IV.2.1.4. Stratégie D.

Dans cette stratégie, les msg à destination d'une station potentiellement défaillante sont captés et conservés par la station de reprise associée.

L'idée principale de la stratégie, est de créer, sur la station de reprise, un environnement favorable au traitement des pannes permanentes. Pour cela, tout msg à destination de la station défaillante est capté et conservé par la station de reprise correspondante. Ainsi, à l'occurrence de panne on adopte l'alternative suivante:

- La panne est transitoire, la station défaillante demande à sa station de reprise le renvoi des msg indispensables. Cette demande sera aisement satisfaite, et l'exécution reprend sur la station défaillante.

- La panne est permanente, les informations nécessaires existent sur la station de reprise qui reprendra l'exécution du processus défaillant.

La différence par rapport à la stratégie B, dans le cas des pannes transitoires, est que les msg réémis proviennent d'une seule station. Par rapport à la stratégie A, il n'y a plus de rétropropagation (panne permanente). Quant à la postpropagation, elle dépend de la satisfaction des critères définis à cet effet, (voir section correspondante pour plus de détails).

Remarques:

Parmi les quatres stratégies précédemment décrites, on peut déceler dès à présent certaines faiblesses notamment:

- Faiblesse de la stratégie A vis-à-vis des pannes permanentes.

- Pénalisation relative des producteurs, sensibilité à l'augmentation du volume du traffic msg, et absence éventuelle de producteurs, tels sont les éléments militant en faveur d'une amélioration de la stratégie B.

- L'absence éventuelle d'une station de reprise (en stratégie D) peut conduire à agir comme en stratégie A (rétropropagation en panne permanente).

Ces différentes remarques nous amènent à envisager des stratégies mixtes , qui regrouperont les avantages présentés par les stratégies précédentes.

IV.2.2. Stratégies mixtes.

IV.2.2.1. Stratégie AD.

La stratégie AD s'énonce comme suit: "les msg reçus et conservés par la station potentiellement défaillante sont captés et conservés par la station de reprise associée".

Les aspects positifs des stratégies A et D se trouvent donc réunis. Aussi, la rétopropagation est éliminée quelque soit le type de panne survenue. La satisfaction des critères de non post-propagation lui permet de réaliser une performance dans la minimisation des perturbations lors de la reprise.

IV.2.2.2. Stratégie AB.

Dans cette stratégie, les msg conservés par les producteurs sont aussi conservés par la station potentiellement défaillante. En raison du taux élevé d'apparition de pannes transitoires vis-à-vis des pannes permanentes, on privilègie le traitement de la première catégorie par une diminution des activations des producteurs pour satisfaire les demandes de renvoi. Une contribution moindre à une augmentation du volume du traffic msg est également à constater.

PREMIERE PARTIE.

Chapitre I

INTRODUCTION.

L'étude des diverses stratégies que nous allons développer dans cette première partie, sera considérée en accordant à tout processus de la "communauté" interactive, une évolution autonome. Cette autonomie est à observer tant au niveau des échanges d'informations entre processus partenaires, qu'au niveau de l'établissememt des points de reprise. En ce sens que chaque processus est libre de créer un Rp quand il le désire et de procéder à un échange d'informations également lorsqu'il le désire. Aucune contrainte n'est donc imposée aux processus interactifs.

C'est cette liberté d'action vis-à-vis de ces deux éléments (échanges de msg, création de Rp) fondamentaux à toute opération de reprise, que se justifie cette notion de "stratégies sans contraintes", par laquelle on désigne ce premier volet de l'étude.

Nous avons déjà fait référence précédemment à ce sujet, à savoir que deux cas peuvent se produire lors de l'étude de recouvrement de pannes:

1. Laisser agir librement les processus sans leur imposer de contraintes, tel est l'objectif de cette première partie, qui favorise l'évolution "naturelle", et dont découlent les avantages suivants:

- Souplesse et absence de restriction au cours des communications interprocessus. Tout processus échange des msg avec qui il veut et quant il le veut.

- ce qui évite la participation du programmeur, et élimine toute source supplémentaire possible d'erreurs.

- L'overhead généré par la création de Rp pénalise le processus pour lequel est accomplie cette action et lui seulement. Il n'y a aucune incidence sur le déroulement des processus partenaires.

Cependant, cette situation de non-contraintes ou "d'anarchie" des processus à l'égard de leurs interactions, et de création de

Rp, bien qu'attrayante au vu des avantages précités, souffre néanmoins du risque éventuel d'une propagation de reprise incontrôlée pouvant dégénérer en effet-domino. Pour éviter que cet aspect positif ne soit contrebalancé par d'éventuelles importantes révocations d'activité, il est nécessaire d'instaurer un moyen de contrôle des propagations qui soit le moins restrictif possible. La conservation des msg reçus, l'adoption des critères de non postpropagation, constituent conjointement, pour notre étude, cet outil de contrôle cherché.

2. Le second cas consiste, pour éviter toute situation de perturbation importante du système, à contrôler les propagations issues d'une opération de reprise, en imposant aux processus concurrents certaines contraintes relatives soit à leurs communications [BRI 84], soit à la céation de leurs Rp. Cette situation d'application de contraintes, fait l'objet de la seconde partie de notre étude, et désignée sous le vocable de " stratégies avec contrainte ".

L'objectif visé par l'étude des stratégies avec contraintes est double:

- Montrer clairement comment peut se dégrader la situation au niveau du système lorsque des mesures palliatives de l'effet domino ne sont pas envisagées; et comment peut évoluer un tel système lorsqu'un contôle de propagation y est instauré. Ce contrôle est bien entendu en accord avec le respect de liberté concédée aux processus.

- Quelles seraient les limites du système, particulièrement ses capacités de reprise sans distinction du type de pannes, et les diverses caractéristiques relatives au coût de l'implèmentation d'une stratégie donnée (pénalisation à faire subir au système en le dotant d'un mécanisme de reprise).

Signalons que dans le souci d'introduire le moins d'overhead possible dans le système, ce dernier ne peut tolérer qu'une seule panne permanente. Quant à la tolérance des pannes transitoires, leur nombre n'est pas limité. L'unicité de cette panne permanente peut révéler une insuffisance, mais celle-ci n'est point critique, car on estime que reprendre une station physique défaillante en

distribuant ses processus sur une station de reprise, est acceptable au vu d'un ralentissement global de l'application si plusieurs stations venaient a défailler.

Les sections suivantes proposent une étude détaillée de chaque stratégie dans le cadre de cette première partie. Une comparaison est établie pour permettre de faire un choix en fonction des caractéristiques du système pour lequel un mécanisme de reprise y est nécessaire.

Chapitre II

STRATEGIE A

" Les msg reçus sont conservés par la station potentiellement défaillante SPD. "

II.1. Description générale.

Afin de mieux percevoir les problèmes qui particularisent chacune des catégories de pannes, et ceux qui leur sont communs pour pouvoir les traiter par les mêmes algorithmes, l'étude de chaque stratégie sera conduite en considérant d'abord les pannes transitoires à taux d'apparition le plus élevé, puis les pannes permanentes, et aboutir enfin à une intégration dans un algorithme de reprise commun (voire annexe).

II.1.1. Pannes transitoires.

L'occurrence d'une panne de ce type peut se traduire par un "arrêt" temporaire du processus en cours d'exécution faisant suite à une altération de l'environnement interne de ce dernier. Cette altération peut corrompre le vecteur d'état du processus, ces variables, son code, et les informations produites et transmises à des processus partenaires. Cette corruption est supposée avoir lieu après la sauvegarde du dernier contexte du processus, contexte que l'on identifie par un point de reprise (Rp: D1).

Grâce à ce contexte présumé exempt d'erreur (hypothèse H1), le processus affecté est capable de restaurer l'état antérieur à la panne et tenter une nouvelle exécution à partir de ce point n'ayant pas encore reçu d'engagement (D2). Cette réexécution sera soumise aux mêmes évènements ayant eu lieu lors de l'exécution précédente, en particulier la nécessité absolue de réutiliser les mêmes msg d'information antérieurement consommés. Ceci est possible grâce à la disponibilité locale de ces derniers (caractéristique de la stratégie). Ainsi cette opération de reprise s'effectuera sans perturbation des processus émetteurs pour regénérer au processus défaillant les msg indispensables (D12). Ayant considéré que la latence d'erreur (D20) est nulle (H1), et qu'un processeur ne peut

tomber en panne exactement pendant l'émission d'un msg, tous les msg émis avant la défaillance sont considérés corrects, ce qui justifie les critères de non postpropagation (D12) ci-aprés:

II.1.2. Critères de non postpropagation.

La postpropagation peut être éliminée si l'un au moins des critères suivant est satisfait.

- Critère 1: A la réception d'un msg, le processeur est capable de reconnaître l'utilité de l'emission de ce dernier. Autrement dit, le processeur récepteur d'un msg peut décider de l'élimination de celui-ci parcequ'il provient d'une réexécution imposée par une opération de reprise, ou de son utilisation parcequ'elle s'avère opportune.

- Critère 2: Le processeur de reprise (celui sur lequel est repris un processus) reconnaît les msg déjà émis lors de l'exécution antérieure, évite leur réémission.

- Critère 3: Le processeur de reprise ne peut satisfaire le critère 2 , et tous les msg sont réémis mais sans provoquer de perturbations à la récèption, car les actions engendrées par les msg sont idempotentes.

Si aucun de ces trois critères n'est satisfait, la pospropagation devient impérative jusqu'à détermination d'un état cohérent du système (D9).

Remarque:

La ligne de reprise qui définit un état cohérent du système est atteinte soit par roll-backs progressifs des processus impliqués dans l'opération de reprise, soit par roll-backs directs (voir section "roll-back direct").

II.1.3. Informations indispensables.

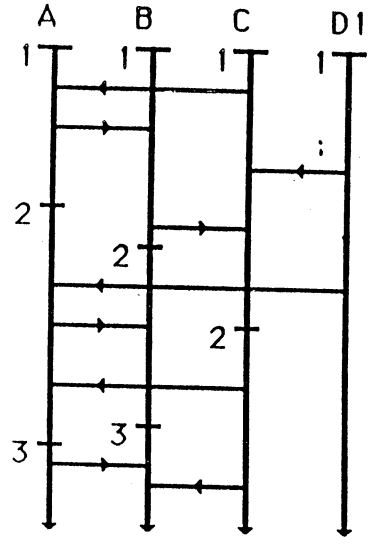
II.1.3.1. Informations pour satisfaires les critères(1,2).

Pour pouvoir réaliser le critère 1, chaque processus doit conserver les informations relatives aux msg reçus et leurs émetteurs notamment: Le numéro de séquence du msg reçu, et l'identificateur de son émetteur. Le critère 2 , symétrique du critère 1, nécessite en particulier que chaque processus conserve le numéro de chaque msg émis, et l'identificateur de son destinataire. Quant au critère 3, aucune information n'est nécessaire.

II.1.3.2. Informations nécessaires à la postpropagation.

Dans le cas où les critères précédents font défaut, il y a lieu de pouvoir propager l'opération de reprise pour atteindre l'ensemble des processus partenaires du processus défaillant, ayant reçu au moins un msg révocable. Cette opération n'est possible qu'à travers les informations identifiant tous les échanges établis dans le sens défaillant-partenaires, échanges enregistrés au niveau de chaque processus. Or l'unité de roll-back est la région de reprise (D3), et la cible d'une propagation étant un point de reprise, il est donc intéressant de pouvoir préciser dans un msg d'invocation destinée à un partenaire, le Rp à partir duquel ce dernier doit reprendre. Cette invocation précise (D11), nécessite au préalable un protocole interprocessus. Ce protocole signifie qu'après chaque réception de msg, il y ait transmission de l'Identificateur de Région de Reprise de Réception (IRRR:D19) du processus récepteur au processus émetteur. On obtient ainsi, un graphe d'échanges du type processus ---> partenaires ci-aprés.

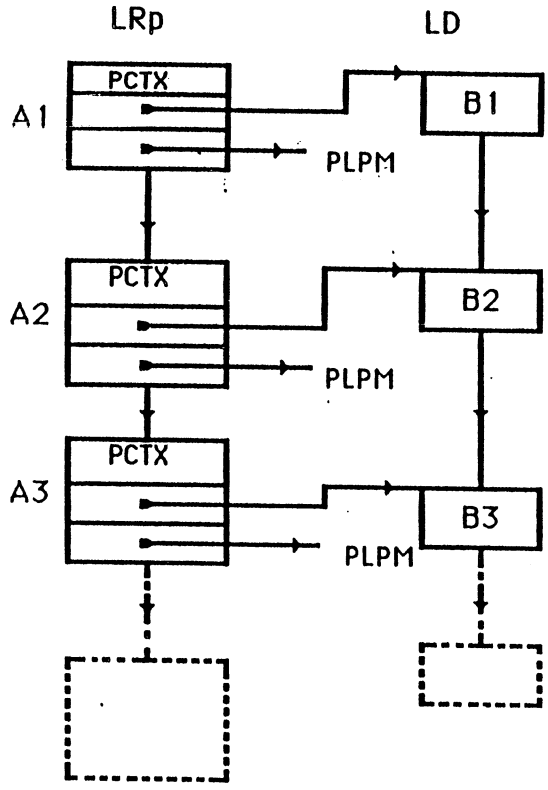
Exemple: Soit un ensemble P de processus intéractifs: P=<A,B,C,D>.



processus intéractifs.

Figure 01

Chaque processus enregistre des points de reprise qu'il associe à d'éventuelles IRRR. Il établit ainsi une relation de dépendance entre des IRRE et des IRRR, comme l'indique le graphe suivant (fig.02) construit pour le processus A de la figure 01.



LRp: Liste de Rp établie pour le processus A.

LD: Liste de dépendants directs de A (D16).

PLPM: Pointeur vers la liste de pointeurs des msg reçus dans la RRR associée à Ai

PCTX: pointeur du contexte associé à Ai.

Graphe d'association des IRRE aux IRRR correspondants.

Figure 02

Un tel graphe a pour unique utilité de pouvoir réaliser une invocation précise lorsque le processus A est repris.
CTX: pointeur du contexte associé à Ai.
PLPM: pointeur vers la liste des pointeurs des msg reçus dans IRRR associée à Ai.

II.1.4. Pannes permanentes.

La station sur laquelle est survenue la panne devient inutilisable, et toutes les informations qui s'y trouvent conservées sont considérées irrémédiablement perdues. Il peut y avoir:

- Perte des msg conservés, ce qui implique une rétropropagation (D12) vers tous les processus émetteurs, au profit du processeur de reprise. Cette opération a pour but de recréer les msg indispensables au processus défaillant.

- Perte également du graphe d'échanges: processus--->partenaires, ce qui empêche d'effectuer la propagation à la manière des pannes transitoires. Il faut donc déclencher une rétropropagation pour l'acquisition des msg, et une postpropagation si les critères (1,2) ne sont pas satisfaits.

II.1.4.1. Informations indispensables.

Pour que le processus de reprise puisse accomplir sa tâche, les informations suivantes doivent être disponibles:

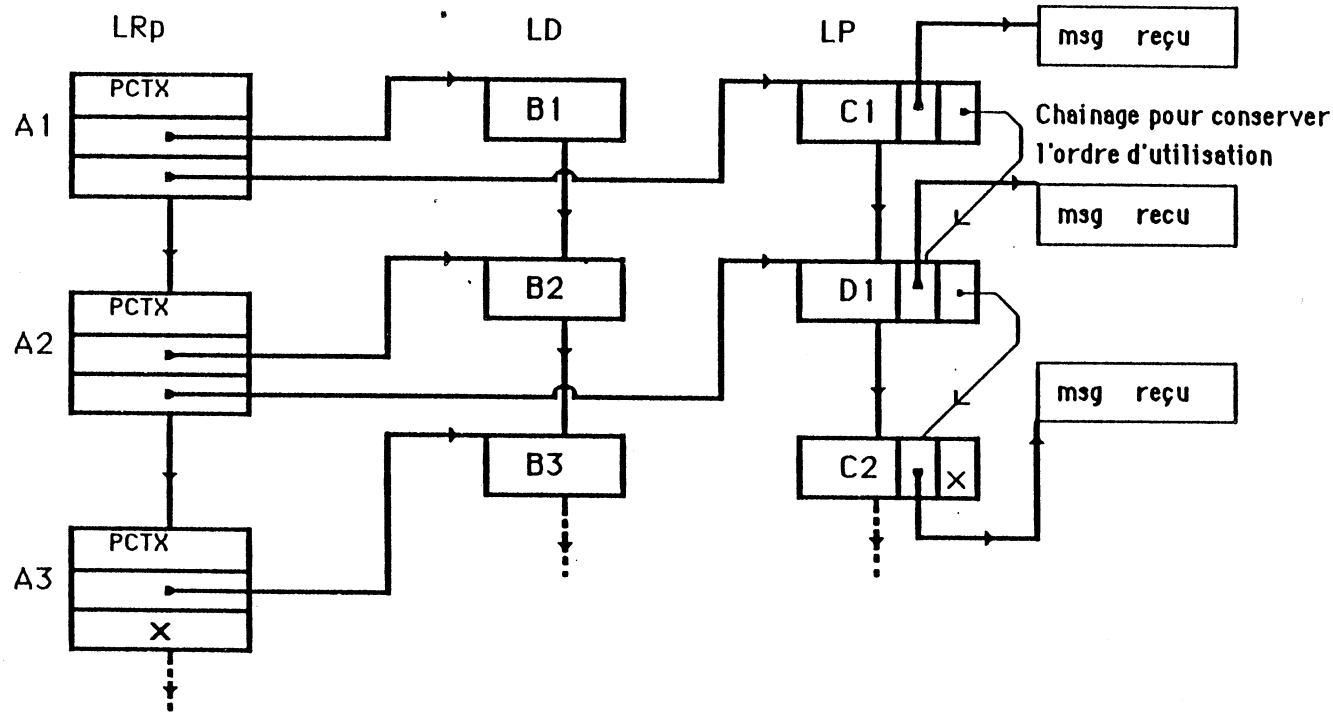
- Le code du processus défaillant.

- Un ou plusieurs Rp transmis par la SPD à la station de reprise associée lors de l'exécution normale du processus.

Ce sont les seules informations indispensables a priori. Le processus de reprise ne peut invoquer directement les processus concernés, pour créer les msg désirés (invocation précise), puisqu'il ignore leurs IRRE. Il procède alors à une invocation aléatoire (D12) en indiquant dans le msg de reprise, comme point de relance RpR (D4), le dernier Rp du processus à reprendre (dernier Rp car utilisation de roll-backs progressifs). Les processeurs concernés ne peuvent répondre à l'invocation du processeur de reprise que s'ils possèdent des IRRR du processus défaillant, d'où l'indispensabilité de ces derniers.

Cependant, à la réexécution du processus défaillant, suite à la perte d'informations relatives aux critères (1,2), le processeur de reprise se trouve dans l'incapacité d'éviter la postpropagation. Aussi même dans le cas où cette postpropagation s'avère obligatoire, il ne peut la provoquer lui-même. Seuls les processus en possession des IRRE du défaillant, précisement le RpR figurant dans le msg, peuvent dans ce cas répondre à cette invocation aléatoire. Donc tout processus doit établir et maintenir la liste des IRRE de

tous les msg reçus. On aboutit ainsi à la construction d'un graphe de reprise par processus, plus complêt que celui défini en 1.1.2.2 fig 2, répondant à la fois aux pannes transitoires et aux pannes permanentes; d'où le graphe de la fig.2 devient:



PCTX: pointeur de contexte associé à Ai

LRp : Liste de Rp établie pour A

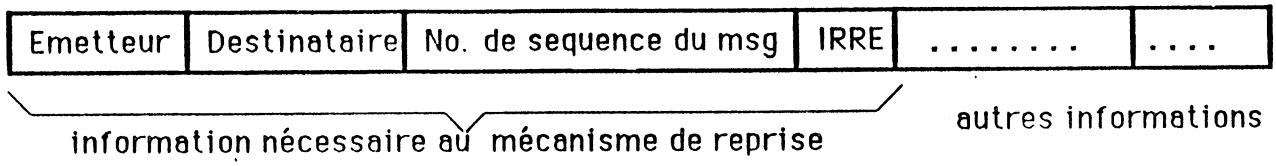
LD: Liste de dépendants directs de A

LP: Liste de propagateurs directs de A

Graphe de reprise (D13) pour le processus A de la fig 1.

Figure 03

Les informations permettant aux processeurs de satisfaire les critères (1,2) et d'établir les listes LP, imposent l'établissement d'un certain protocole d'accord entre processus intéractifs au niveau de la structure des msg échangés (fig 4).



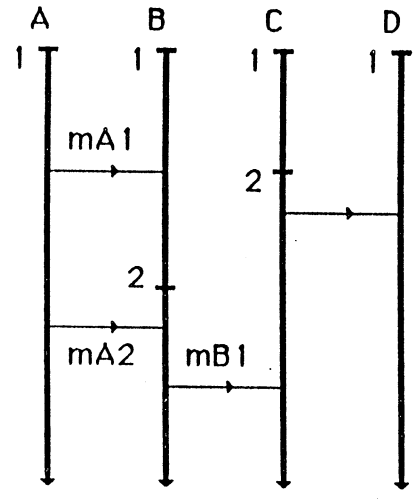
Structure d'un msg.

Figure 04

II.1.4.2. Adoption des critères (1,2).

Si en pannes transitoires aucun problème ne s'oppose à ce que un des critères (1,2) soit satisfait, il est toutefois impossible au processeur de reprise d'assurer un quelconque de ces critères et ce, par manque des informations adéquates. Cette impossibilité est bien entendu relative non pas à la faisabilité, mais plutôt au coût encouru si on souhaiterait disposer des informations requises localement aux processeurs de reprise.

Il est cependant possible d'éviter la postpropagation en assurant les critères (1,2) non pas au sein du processeur de reprise, mais au niveau des autres processeurs partenaires. Ainsi le premier agit comme si le critère 3 est satisfait. Il incombe alors à ces derniers de réaliser le critère 1. Cette solution n'est pas entièrement satisfaisante comme le montre l'exemple suivant.



Perturbations des msg redondants.

Figure 05

En supposant que B serait le processus défaillant, celui-ci serait repris en B2 par le processseur de D (par exemple). C n'étant nullement affecté par cette reprise, le msg mB1 serait réémis à la nouvelle exécution de B, mais il serait détecté par C qui éviterait sa prise en compte, car le critère 1 est satisfait. Par contre, A reprendrait en A1, et comme seul le critère 1 est assuré, A réémettrait les msg mA1 et mA2 dont seul mA2 est indispensable. Le msg mA1 provoquerait inévitablement une perturbation au sein de B (repris par le processeur C). Il est donc indispensable que les deux critères (1,2) puissent être simultanément satisfaits par chaque processus.

Cette action de reconnaissance et détection de msg "redondants" n'est déclenchée qu'au niveau des processeurs de reprise après occurrence de pannes transitoires. Elle impliquerait des processeurs exécutant les partenaires du processus défaillant, seulement aprés panne permanente.

Donc tout processeur doit être capable, en fonction de la situation qui se présente, de satisfaire l'un ou l'autre des critères (1,2). Comme le critère 2 évite les réémissions inutiles il s'avère donc être plus performant, et son utilisation doit être aussi fréquente que possible.

II.2. Stratégie A sans postpropagation.

II.2.1. Réalisation des critères (1,2).

Pratiquement, une manière possible pour assurer les critères (1,2) de non postpropagation, est que tout processeur maintiendrait pour chaque processus qu'il exécute, une table regroupant les informations nécessaires à cet effet. Cette table est mise à jour régulièrement à chaque émission ou réception de msg d'informations. Elle contiendra l'identificateur, le numéro du dernier msg reçu et le numéro du dernier msg émis, de ou vers un processus partenaire.

Exemple: Pour le processus B de la figure 1, on aura:

Identitité du Processus	No du dernier msg émis	IRRE	No du dernier msg reçu	IRRR
A	—	A3	3	B3
C	1	C2	3	B3

Table d'information relative
aux critères (1,2).
Figure 06

Remarque:

Cette table associée à chaque processus de toute station potentiellement défaillante, disparaît à l'occurrence d'une panne permanente. Les critères (1,2) peuvent être assurés malgré son absence (par les processus partenaires du défaillant).

II.2.2. Pannes transitoires.

Hypothèse:

La panne est détectée est diagnostiquée.

Le processeur de la station défaillante effectue l'opération de reprise, qui consiste en la démarche suivante.

1- Il choisit comme Rp de relance (RpR) le point de reprise actif (D4). ce choix est justifié par l'absence de tout retour de propagation (msg disponibles et critères 1 ou 2 satisfait). Le processus défaillant se trouve ainsi sans aucune influence vis-à-vis de l'environnement externe.

2- Il restaure l'état du processus correspondant au RpR choisi.

3- Il réexécute le processus.

Pour éviter la postpropagation, le processeur doit donc respecter le critère 2; pour cela il utilise, les informations de la table fig 6 et le numéro de séquence du dernier msg émis avant l'enregistrement du RpR choisi. Ce numéro étant sauvegardé en même temps que le contexte associé au RpR. Ainsi à la réexécution, on reconstitue le séquencement des msg dans la région de reprise réexécutée, et on détecte les msg dont la réémission doit être évitée.
Par exemple figure 1, la sauvegarde du contexte associé à A3 doit inclure celle du numéro du msg émis immédiatement avant, soit le numéro 2. A la reprise, A3 est restauré avec le numéro de sequence 2 et le séquencement de la région de reprise identifiée par A3 sera rétabli. D'ou le prochain msg à émettre serait 2+1=3 qui sera détecté en consultant la table.

II.2.3. Pannes permanentes.

Hypothèse:

La panne est détectée, diagnostiquée et la reconfiguration appropriée est réalisée.

La sauvegarde du numéro du dernier msg émis dans une région de reprise, en même temps que le Rp qui définit la région de reprise suivante, et leur transmission ensemble au processeur de reprise, est plus justifiée pour les pannes permanentes du fait de la perte d'informations.
Les deux critères (1,2) sont indispensables, et la non postpropagation est assurée non pas par le processeur de reprise mais par ses partenaires. D'où l'incapacité d'assurer cette non postpropagation en présence de deux ou plusieurs pannes successives ou simultanées.
Le processeur de reprise, disposant localement du code du processus défaillant et un ou plusieurs de ses Rp, effectue les actions suivantes.

1- Emission vers chaque station/processus (y compris lui-même) d'un msg de reprise dans lequel est précisé un RpR du processus défaillant Pi. Ce msg fait référence à une invocation aléatoire

pour les partenaires du défaillant, et implicitement à une invocation précise pour ce dernier. Une fois le msg reçu, le processeur de reprise restaure l'état du processus associé au RpR, et entame la réexécution. Le RpR choisi est le plus récemment établi puisqu'il s'agit de la première panne et aucun roll-back au-delà de RpR n'est possible (critères (1,2) satisfaits). Autrement dit, le RpR appartient à la ligne de reprise.

2- A la réception d'un msg de reprise, le processeur récepteur Pr détermine s'il existe un de ses processus Pj tel que: Il existe Rpj appartenant à Pj et Rpj --> RpR, i.e: Rpj est propagateur direct de RpR (D5). Dans le cas où il en existe plusieurs, le plus dominant est choisi.

Si Rpj existe alors:

3- Destruction de tous les successeurs de Rpj sans affecter leurs listes LID et LIP (D16). Cette destruction est impérative lorsqu'on considère différents les intervalles de création des Rp entre différentes réexécutions successives d'un même processus.

4- Restauration de l'état de Pj associé à Rpj, puis réexécution effective.

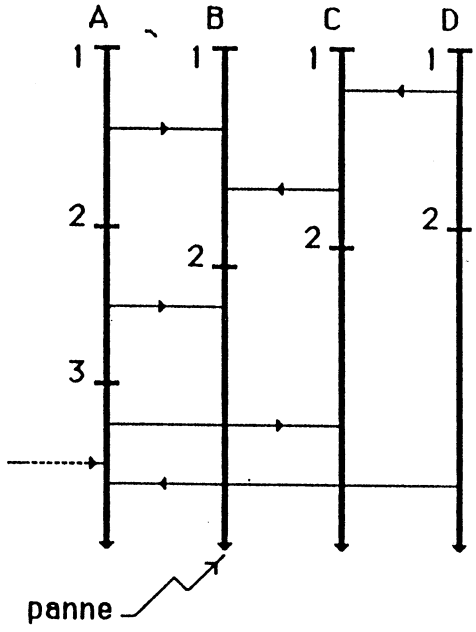
Remarque:

Dés lors le processus défaillant est repris, le processeur lui reconstitue son graphe de reprise (à partir de RpR) et la table des informations relative aux critéres (1,2). Les processeurs atteints par la rétropropagation veilleront à ce que seuls les msg à destination du processus défaillant seront réémis.

II.2.3.1. Problème de multiples pannes.

II.2.3.1.1. Mise à jour des propagateurs et dépendants.

A l'étape 3 de II.2.3., il peut exister une incohérence dans les listes LID et LIP des propagateurs et dépendants directs du RpR du processus défaillant. Pour pouvoir les éviter, il faut procéder à une mise à jour de ces listes juste avant la destruction des successeurs de RpR. L'exemple suivant illustre cette situation:



Exemple d'incohérence possible
des listes LID et LIP.
Figure 07

Supposons que B tombe en panne, il est repris par un processeur Prx qui provoque la rétropropagation de A en A2 (car A2-->B2). A3 successeur du RpR A2 sera détruit car le successeur ultérieur Ai de A2 sera soit antérieur ou postérieur à l'ancien A3. Or A3 inexistant est enregistré dans la LIP de C2 et dans la LID de D2, il faut donc mettre à jour ces listes en y faisant remplacer A3 par A2. Cette mise à jour ait lieu suite à un msg émis à cet effet par A. Lorsque Ai successeur de A2 sera créé (par exemple à l'emplacement indiqué par --->), C et D qui évoluent sans être affectés par la reprise doivent être mis à jour pour prendre en compte Ai. C'est-à-dire D2 aura non plus A2 comme dépendant direct mais Ai nouvellement créé. On pourra donc avoir une mise à jour à la suppression des successeurs et à la création de nouveaux Rp.

II.2.3.1.2. Multiples pannes.

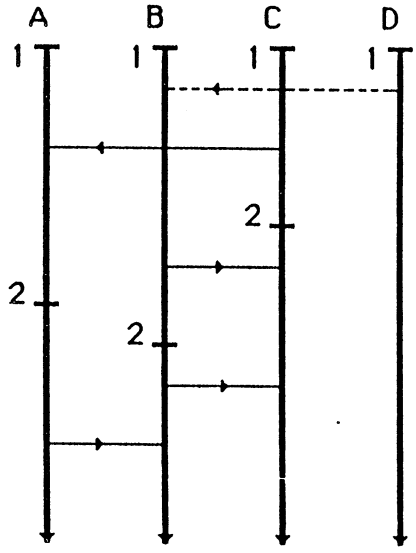
II.2.3.1.2.1. pannes simultanées.

L'occurrence de deux ou plusieurs pannes simultanées fait apparaître une indétermination sur les échanges ayant pu avoir lieu entre plusieurs processus défaillants. Ce qui ne permet guère d'accorder une quelconque confiance à toute opération de reprise (sauf reprendre les processus défaillants à partir de leurs débuts!). C'est le cas où les processus A,B (fig 7) tombent en panne en même temps; quels seront alors leurs RpR ? et si l'une des deux stations est station de reprise de l'autre, que faire ? Ces questions seront abordées dans la partie II de notre étude.

II.2.3.1.2.2. Pannes consécutives.

Ce cas concerne des pannes séparées par des intervalles de temps supérieurs à la durée de l'opération de reprise (hypothèse H2). Ainsi lorsqu'un processus Pi est repris à partir d'un RpR (soit Rpi), son graphe de reprise est reconstitué à partir de Rpi, tout son historique antérieur devient inconnu. Une panne suivante affectant un processus Pj telque Pj soit repris à partir de Rpj, et il existait Rpi-1 appartenant à Pi telque Rpi-1 ---> Rpj, ne serait jamais détecté lors de la reprise.

Exemple:



Exemple d'incohérence dans
une opération de reprise.
Figure 08

Soit une première panne affectant B, B serait repris par le processeur de A (par exemple) en B2; par rétropropagation A serait contraint de reprendre en A2, C et D demeurent non impliqués. L'opération de reprise étant terminée, une panne suivante qui affecterait C le ferait reprendre (par le processeur de D) en C2. Comme le msg de reprise fait référence à une invocation aléatoire , celui-ci aura pour effet de faire reprendre B en B1 (rétropropagation) mais par manque d'informations seule la relation B2--->C2 sera détectée par le processeur de reprise de B, alors B sera repris finalement en B2 ce qui donnerait une incohérence à l'opération de reprise.

Puisque cette situation ne peut provenir que des dépendants directs du processus défaillant, une solution possible mais relativement coûteuse consiste à procéder par roll-backs progressifs et préventifs (au niveau du processeur de reprise). Ainsi les dépendants directs répondront éventuellement à l'invocation aléatoire en indiquant seulement le RpR qui convient de choisir pour le processus défaillant.

Par exemple, lorsque B (fig 8) tombe en panne le premier, C dirigera le processeur de reprise à choisir B1 comme RpR et non B2 ce qui permettra de prévenir la situation qui vient d'être décrite mais favoriserait de longues chaines de reprise. C'est ainsi que D qui, jusque là n'est pas impliqué, se verrait atteint par la rétropropagation (en supposant existante la relation D1 --> B1 en pointillés).

En définitive, pouvoir reprendre n pannes transitoires et une panne permanente unique, paraît être raisonnable dans bien des applications. L'overhead ainsi encouru demeure dans des limites acceptables.

La partie II " stratégies avec contraintes" traite ce problème de façon satisfaisante.

II.2.4. Perturbations et chaines de reprise.

Compte tenu, du fait qu'aucune propagation n'est possible en pannes transitoires, la longueur d'un roll-back ne peut dépasser celle de la région de reprise courante.
Autrement dit, la propagation est de niveau n = 0, et d'ordre p =1 (D14), ce qui constitue un cas optimal.
Globalement sans distinction de type de panne, la situation à redouter est celle engendrée par l'effet domino (D15). Or dans le cas présent, ce dernier ne peut être provoqué que par des processus qui invoquent et propagent la rétropropagation (processus repris à la suite de panne permanente).

Tout processus possédant son graphe de reprise (i.e: non encore repris ou repris aprés pannes transitoires) constitue pour toute rétropropagation un point d'arrêt. comme la postpropagation est éliminée, il ne peut y avoir de retour de propagation, et l'absence de cycle implique l'absence d'effet-domino.

Proposition:

La stratégie A avec critères (1,2) satisfaits est exempte de longues chaines de reprise (effet domino). Dans l'hypothèse de tolérance d'une panne permanente unique, la propagation est au plus de niveau n = m où m est le nombre de propagateurs directs du Rp de relance du processus défaillant.

Preuve:

Supposant qu'il existe un cycle de propagation atteignant l'ensemble P de processus intéractifs Pi (i=1,N), alors il existe au moins un processus ayant fait au moins deux roll-backs consécutifs. Or toute rétropropagation vers un processus ne peut provoquer qu'un seul roll-back, et qu'un roll-back unique (vers le RpR) est possible au niveau du processus défaillant.

Donc le processus à double roll-back est le processus défaillant, et comme il ne peut y avoir deux défaillants simultanés (hypothèse) pour provoquer mutuellement une rétropropagation, un roll-back (parmi les deux) est issu d'une postpropagation: contradiction de l'hypothèse (critères (1,2) satisfaits).
D'où l'hypothèse de l'existence d'un cycle de propagation est fausse. Puisqu'un cycle ne peut se produire, le cas le plus défavorable est celui où, parmi les N processus interactifs, N-1 sont propagateurs directs du défaillant, ce qui provoque la reprise de l'ensemble des processus. Dans ce cas le niveau de la propagation est n=N-1.

II.3. Stratégie A avec postpropagation.

La propagation est inévitable pour deux raisons:

- Soit que les critères prédéfinis ne peuvent être satisfaits.
- Soit que c'est une conséquence d'une prise en compte d'une latence d'erreur non nulle.

On a supposé précédemment (Hypothèse H1) que la latence était nulle, donc les msg émis juste avant la panne étaient corrects, ce qui n'est pas vrai en général. Même dans les cas les plus

optimistes, les msg émis dans la région de reprise courante, avant la détection de la panne, sont susceptibles d'être erronés. La suppression de la rétropropagation par réutilisation des msg conservés, suppose implicitement que toute panne détectée a pour origine le lieu de sa détection. Elle ne peut donc être le fait d'une contamination par msg reçus qui relève du domaine des erreurs logicielles (détection par tests d'assertions).

II.3.1. Pannes transitoires.

Après détection et diagnostic de la panne, le processeur de reprise effectue les opérations suivantes:

1- Emission d'un msg de reprise (invocation précise) à chaque dépendant direct du Rp actif choisi comme RpR. Puisque chaque processus possède la totalité de son graphe de reprise, une invocation aléatoire est aussi possible (le principe des roll-backs directs peut être appliqué).

2- Renommage du Rp de relance pour pouvoir distinguer un retour de propagation de la même opération de reprise vers le processus initiateur, d'une nouvelle propagation lancée par un autre processus distant atteignant le RpR. Il est possible d'éviter ce renommage et les mises à jour qui en résultent, à condition de transmettre dans le msg d'invocation un identificateur de session de reprise.

3- Emission d'un msg de mise à jour à tous les propagateurs directs du Rp de relance, pour remplacer l'ancien identificateur de ce dernier par le nouveau. Emission d'un msg de mise à jour à la station de reprise associée.

4- Destruction des successeurs de RpR et de sa liste LID. Restauration de l'état du processus en ce Rp, et réexécution.

Remarque:

Pour qu'il n'y ait pas de distinction entre l'initiateur de reprise et les autres partenaires, les quatres étapes précédentes sont exécutées par tout processus impliqué dans l'opération de reprise. Donc pour exécuter un algorithme unique, le msg d'invocation de reprise sera émis en premier à l'initiateur lui-même.

II.3.1.1. Emission et réception de msg.

Vus que les critères (1,2) ne sont pas satisfaits, la structure des msg est légèrement différente de celle de la figure 4. Le champ " numéro de séquence relatif à l'émetteur" s'avère inutile. Puisque tous les msg d'un processus relancé sont réémis, il est plus fiable d'annuler l'effet de conservation au niveau des processus destinataires. Autrement dit, dès qu'un msg est reçu, il est réutilisé et conservé de nouveau (l'ancienne version étant éliminée). Cette attitude correspond à une prise en compte d'une latence non nulle, situation relativement plus réaliste et moins onéreuse que le cas de postpropagation imposée par l'incapacité de satisfaire les critères (1,2).

La rétropropagation étant évitée, il est important de procéder à la mise à jour des propagateurs des Rp détruits (successeurs de RpR), et de rétablir la dépendance avec ces propagateurs lorsque de nouveaux Rp seront créés (problème identique qu'en II.2.3.1.1).

II.3.2. Pannes permanentes.

Hypothèse:

Panne détectée, diagnostic effectué et reconfiguration appropriée réalisée.

La restriction des informations disponibles sur la station de reprise à savoir: le code et les Rp du processus défaillant, impose au processeur de reprise de procéder à une invocation aléatoire. Les actions suivantes sont alors exécutées.

1- Emission d'un msg de reprise à toute station/processus y compris au processeur de reprise lui-même. Ce msg contient en particulier un RpR choisi par le processeur, soit RpRj. Ce dernier peut faire référence à deux significations possibles:

a- Le msg vehicule une invocation aléatoire, et une détermination d'un état cohérent global par roll-backs progressifs est adoptée (utilisé dans la suite).

b- Le msg n'étant en fait qu'une enquête pour récupérer des RL secondaires (ou intermédiaires) qui seront analysées par le processeur de reprise pour déterminer une RL globale, à partir de laquelle des invocations précises seront émises (voir "roll-back direct").

2- Une fois le msg émis, le processeur renomme RpRj: soit RpRj'. Il restaure l'état du processus en ce point et relance l'exécution.

Lorsqu'un processus Pi reçoit un msg d'invocation (une panne unique est tolérée), il agit de la façon suivante:

1- Détermination d'un RpRi appartenant à Pi:

. Recherche d'un Rpi appartenant à Pi telque Rpi--->RpRj (Rpi propagateur direct de RpRj).

. Recherche d'un Rpk appartenant à Pi telque RpRj--->Rpk (RpRj propagateur direct de Rpk). Si Rpi et Rpk existent tous deux alors, RpRi est le dominant des deux, Sinon RpRi prend la valeur du Rp trouvé soit Rpi, soit Rpk.

2- Renomme RpRi trouvé, et émet un msg de mise à jour à chacun de ses propagateurs.

3- Emet un msg de reprise à chaque dépendant de RpRi (invocation aléatoire est possible), détruit les Rp successeurs de RpRi et leurs listes LID.

4- Restauration du contexte du processus associé à RpRi, relance effective de l'exécution.

Remarque:

L'invocation précise qu'on peut utiliser systématiquement en pannes transitoires, et qui élimine les phases de recherche inévitables de l'invocation aléatoire, s'avère inopérante lors d'une panne permanente. C'est pourquoi une utilisation conjointe est impérative. Elles sont loin d'être compétitives mais plutôt complémentaires, puisque la première (précise) atteint exactement le nombre de stations/processus concernés avec autant de msg à émettre, alors que la seconde n'émet qu'un seul msg (diffusion)

qui sera traité par l'ensemble des stations.

II.3.3. Perturbations et chaines de reprise.

La conservation des msg permet certes, d'éviter la rétropropagation, mais dans le cas actuel où on prend en compte ensemble les pannes transitoires et permanentes et où la postpropagation systématique est capable d'annuler l'effet souhaité de la conservation, il est difficile de pouvoir quantifier ou borner de façon précise toute opération de reprise.

la propagation interprocessus dans son ensemble , constitue le facteur fondamental des perturbations. Comme elle peut être générée par les deux éléments (rétro et/ou postpropagation) en éliminer un, réduirait de façon notable les longues chaines de reprise, mais ne permet en aucun cas d'affirmer l'évitement de l'effet-domino. Notons que tout processus intéractif non dépendant indirect de l'initiateur de reprise servirait à stopper la propagation.

Conserver les msg c'est aussi les gérer, et plus leur nombre est important, plus est important le temps consacré à leur gestion. L'overhead ainsi subi peut représenter un facteur non négligeable qu'il faudrait éventuellement prendre en compte en même temps que l'espace de conservation exigé pour opter en faveur d'une telle stratégie (A). La section "comparaison des stratégies" prend en compte ces problèmes.

II.4. Conclusion

On vient d'étudier la stratégie A selon deux optiques:

- La première avec latence nulle , où la conservation des msg combinée avec les critères de non postpropagation, fournit des résultats intéressants vis-à-vis des perturbations.

- La seconde avec une latence relativement faible de sorte que le Rp actif soit exempt d'erreur, et avec la postpropagation systématique qui en découle. Les chaines de reprise pouvant être générées dans ce cas, peuvent être importantes sans toutefois que l'on puisse les borner avec certitude, mais pour lesquelles la conservation de msg constitue un facteur limitatif de ces perturbations.

Comme c'est la postpropagation qui est susceptible de créer ces situations de dégénérescence en effet-domino, les systèmes assymétriques de processus communicants où les processus consommateurs de msg sont plus importants que les producteurs, se prêtent mieux à la stratégie.

La conservation indispensable pour l'élimination de la rétropropagation peut être soumise à des contraintes d'overhead issu de la gestion des msg et d'espace nécessaire. Aussi est-il important de considérer l'efficacité de cette stratégie dépendant du degré d'interaction entre processus. Les inconvenients majeurs de la stratégie relativement à la seconde optique proviennent d'une incapacité de:

- Borner les propagations interprocessus, et

- Traiter plusieurs pannes permanentes.

Ces situations sont le produit d'une liberté accordée aux processus au cours de leurs évolutions. Des contraintes sont introduites pour remédier aux insuffisances précitées, elles font l'objet de la partie II de notre étude (stratégies avec contrainte).

Chapitre III

STRATEGIE B

" Les msg reçus par une station potentiellement défaillante sont conservés par leurs producteurs."

III.1. Description générale.

dans cette stratégie, tous les msg d'informations produits par une station STi, et émis vers une ou plusieurs stations STj, sont conservés au sein de STi. Cette conservation permet aux processus repris, de pouvoir se faire réémettre les msg indispensables à leurs réexécutions, sans affecter le déroulement de leurs processus émetteurs. Cette réémission peut se faire , soit à la demande des consommateurs (notamment lors des pannes transitoires), ou à l'initiative des producteurs lorsqu'ils détectent des stations en panne auxquelles ils sont liés.

La stratégie B apparaît comme "symétrique" de la stratégie A, comme le sont d'ailleurs les critères (1,2) de non postpropagation. Puisque l'unité de reprise d'un processus est la région de reprise, il serait adéquat qu'un processus puisse demander le renvoi de msg reçus à partir de son Rp de relance. Les producteurs doivent ainsi adopter une structure de buffers de conservation permettant d'associer des msg émis à une Région de Reprise de Réception (RRR) donnée.

Par le biais de cette conservation, la rétropropagation peut être évitée quelque soit le type de pannes considéré. Quant à la postpropagation, elle peut être évitée ou non selon que les critères (1,2) sont ou non satisfaits. Cependant la rétropropagation peut s'avérer indispensable lors de multiples pannes permanentes, cas qu'on considèrera à part dans la suite. Notons que l'on observe, une amélioration vis-à-vis des pannes permanentes par rapport à la stratégie A, et une augmentation sensible du traffic de msg. On peut également constater une répartition des msg conservés, ce qui peut jouer un rôle important pour la

ressource espace-mémoire.

III.1.1. Informations indispensables:

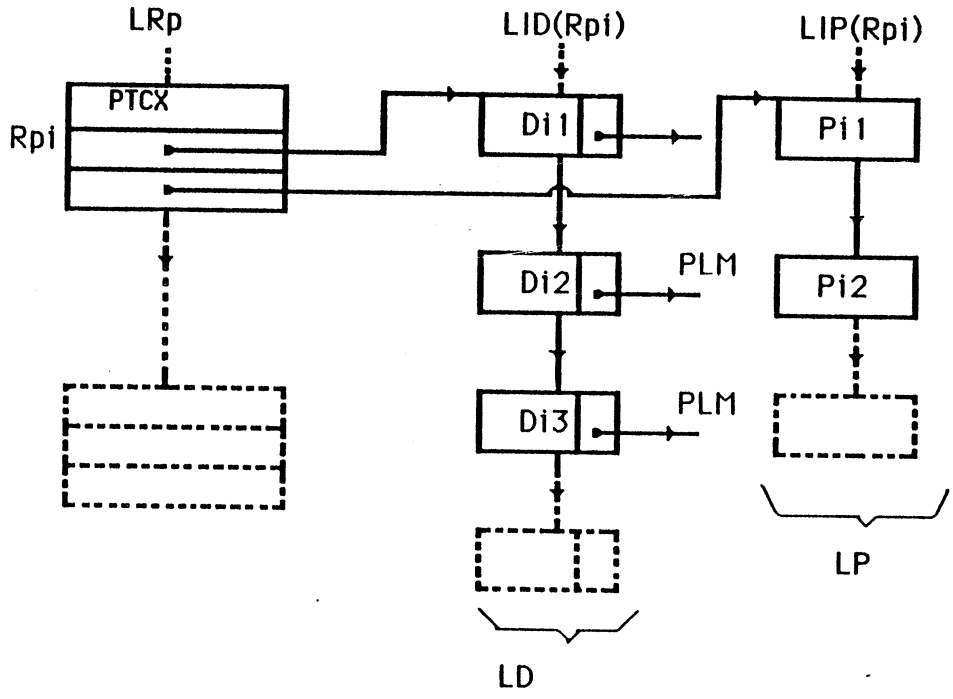
a- Graphe de reprise.

Pour pouvoir invoquer les deux types de propagation, le graphe de reprise est indispensable. Il comprend en l'occurrence:

- La liste des Rp du processus, dont chaque élément Rpi est constitué des trois pointeurs suivants.
  . Un pointeur indiquant le contexte associé à Rpi,
  . un pointeur indiquant la liste LID de Rpi,
  . un pointeur désignant la liste LIP de Rpi.
- La liste LD du processus (LD est l'union des LID de tous les Rpi).
- La liste LP (D16) du processus (LP est l'union des LIP de tous les Rpi).

Le renvoi de msg indispensables à un récepteur en reprise (base de la stratégie), impose une structure de données qui permet d'associer les msg émis aux RRR. Ainsi, le processeur de reprise peut émettre un msg de renvoi aléatoire dans lequel est précisé le RpR du processus repris Pi. Il appartient alors aux processus propagateurs Pj de déterminer l'ensemble des msg qu'il est nécessaire de renvoyer. Donc, chaque élément de la liste LD repérera une liste de pointeurs de msg à destination des processus dépendants, comme le montre la figure 01 suivante.

Exemple:



PLM: Pointeur vers la liste des msg conservés, émis dans la RRE associée à Rpi, et reçus dans la RRR associée à Dij.

Graphe de reprise
en stratégie B.
Figure 01

PLM: pointeur vers la liste des msg conservés, émis dans la RRE associée à Rpi, et reçus dans la RRR correspondante.

b- Critères (1,2) d'évitement de la postpropagation.

Pour pouvoir satisfaire ces critères, des informations d'identification des msg relatifs aux émetteurs et aux récepteurs sont indispensables. On peut ainsi utiliser le même traitement qu'en stratégie A (fig 6).

c- Structure des msg.

Pour pouvoir assurer a et b, il est nécessaire d'appliquer la même structure de msg que celle définie en stratégie A. Ainsi tout

msg comporte deux parties:

- Une partie véhiculant les informations destinées au processus récepteur et au mécanisme de transmission.

- La seconde partie est destinée exclusivement au mécanisme de reprise.

III.2. Stratégie B avec postpropagation.

III.2.1. Pannes transitoires.

La caractéristique de ce type de pannes est que les informations relatives à l'opération de reprise sont toujours disponibles, tout en présumant qu'elles ne soient pas altérées par la panne (dans le cas contraire, une redondance de ces informations ou de structures de données s'avèrerait utile [TAY80]), mais cela est en dehors de notre étude. La demande de renvoi des msg indispensables formulée par le processus défaillant Pi aux producteurs Pj, peut être établie de deux manières possibles:

- Renvoi aléatoire , où les producteurs sollicités, déterminent eux même, en fonction du RpR communiqué par le demandeur, les msg qu'ils doivent éventuellement réémettre. Le terme aléatoire (identique en invocation aléatoire) signifie que le msg de renvoi est émis à toute station/processus, qui y donne suite au cas où elle serait concernée.

- Renvoi précis : signifie qu'on désigne dans le msg de renvoi, le producteur concerné et qu'on lui fournit des repères pour qu'il puisse déterminer correctement les msg à renvoyer. Ces repères sont formés du RpR du processus demandeur, et de son propagateur direct qui appartient à Pj. Une autre façon de solliciter un renvoi précis, utilisée particulièrement en pannes transitoires, consiste à fournir au producteur désigné, un numéro de msg à partir duquel le renvoi serait effectué.

L'opération de reprise diffère peu selon que la panne est transitoire ou permanente. Lorsqu'une panne est detectée et diagnostquée, le processeur effectue les opérations suivantes:

1- Emission d'un msg de renvoi des msg indispensables à chaque propagateur direct de RpR (à chaque élément de la liste LIP de RpR). Lorsque plusieurs propagateurs appartiennent à un même processus, la relation de dominance (D4) doit être appliquée pour optimiser le nombre de msg de renvoi (MDR).

Comme la postpropagation est systématique, quand un processeur reçoit un MDR, il doit s'assurer que ce renvoi ne sera pas redondant, puisque s'il est en même temps propagateur et dépendant, sa reprise forcée provoquerait une double réémission. Cette situation est susceptible de compromettre le bon déroulement de l'opération de reprise. On peut donc dire, que lorsque les relations de propagateurs et de dépendants coexistent, la seconde peut annuler la première.

2- Un retour de propagation vers un RpR est possible, aussi pour le distinguer d'une nouvelle opération de reprise, il est important de renommer RpR. Comme l'ancien identificateur peut être enregistré dans d'autres processus, il faut le mettre à jour. Cette mise à jour peut être incluse dans l'étape 1 auquel cas, seront précisés dans MDR, le RpR et sa nouvelle identification, ce qui épargnerait l'étape de mise à jour.

3- Emission d'un msg de reprise à chaque dépendant de RpR (invocation précise). Une invocation aléatoire est également possible.

4- Destruction de tous les successeurs de RpR, s'il en existe.

5- Destruction de la liste LID de RpR.

6- Restauration de l'état du processus en RpR, et réexécution.

III.2.1.1. Renvoi des msg:

Le succès de toute opération de reprise est conditionné par une bonne gestion des msg au sein des producteurs. Ainsi leur réutilisation doit se faire dans des conditions identiques à leur première utilisation; notamment en ce qui concerne l'ordre de réémission qui doit être conforme à celui de la conservation.

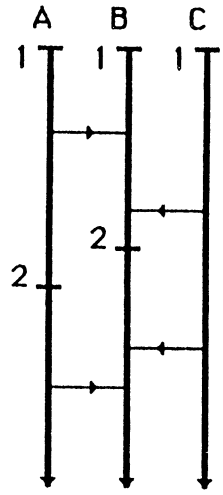
Lorsqu'un processeur reçoit un msg de renvoi, il agit selon le protocole défini avec son émetteur; plusieurs cas sont possibles:

a- Le MDR précise un numéro de msg qui doit être réémis puis les suivants, ce qui signifie, sont prises en compte les actions subséquentes:
Vis-à-vis du producteur, tout msg émis est conservé et chainé selon l'ordre d'émission, dans une liste relative au destinataire, ou dans une liste générale à tous les récepteurs.
Vis-à-vis du consommateur, le numéro du premier msg reçu de chaque processus Pi, dans chaque RRR, est enregistré en même temps que le Rp propagateur (dans LIP) figurant dans le msg.

Ainsi, lorsqu'une reprise ait lieu à partir d'un RpR (noté Rpi), un MDR est adressé à tout Rpj appartenant à LIP (Rpi) et ce, avec application de la relation de dominance: i.e. si Rpj et Rpk appartiennent tous deux à LIP(Rpi), et Rpj, Rpk appartiennent au même processus P tel que Rpj < Rpk alors, seul Rpj est pris en compte. Ce MDR précise à la fois Rpj et le numéro du premier msg émis dans la Région de Reprise d'Emission (RRE) associée à Rpj. Une fois l'opération de renvoi est déclenchée, celle-ci est soumise au schéma classique du producteur-consommateur [CRO78,COR81].

b- Le MDR indique le RpR du processus P repris. Cela implique que le producteur associe à chaque msg émis la RRR au niveau de P (fig 1); c'est le renvoi aléatoire. Une recherche dans la liste LD du producteur permet de déterminer l'occurrence de RpR et les pointeurs des msg à renvoyer.

c- Le renvoi précis, où le processeur de reprise émet un MDR à chaque élément de la liste LIP de RpR, en précisant donc dans MDR, le propagateur direct Rp de RpR. Ainsi, si les msg émis dans la RRE associée à Rp sont envoyés sans précaution, une perturbation à la réception est possible (voir exemple suivant).



On a : C1 ⟶ B1
et C1 ⟶ B2
Si B est défaillant, il sera repris en B2 en émettant un MDR à C prècisant B2 comme RpR. Si C agit sans prècaution en renvoyant les msg associés à la RRE de C1, B recevra deux msg au lieu de un ce qui fauserait la reprise.

Exemple de perturbation à la réception de msg.

Figure 02

Donc le renvoi précis est insuffisant, il sera néanmoins très utile lorsqu'une contrainte sera introduite dans la stratégie au niveau de la deuxième partie.
Il est cependant possible de l'adopter à condition de faire une vérification sur les msg réémis à la manière du critère 1.

III.2.2. Pannes permanentes.

Hypothèses:

Pour des raisons de coût généré par la gestion des informations à dupliquer sur les stations de reprise, on considère l'occurrence d'une panne unique (le nombre de pannes transitoires n'est pas limité).

La détection des pannes permanentes peut se faire de deux manières:

- L'occurrence de panne rend inopérante la station défaillante, la détection se fait par les stations opérationnelles avoisinantes. Après quoi, la commutation vers le processeur de reprise s'opère à l'initiative de ce dernier. Cette commutation est accomplie après reconfiguration matérielle appropriée, et déconnexion de la

station défaillante.

Ainsi, l'opération de reprise peut être déclenchée par le processeur de reprise selon un msg de renvoi aléatoire.
A noter que les autres processeurs concernés ne peuvent agir selon leurs propres initiatives, en exécutant un renvoi sans attendre le MDR; et cela pour éviter de provoquer une perturbation à la réception (voir cas fig 2).

- L'occurrence de panne est détectée par le processeur de la station défaillante après de vaines reprises d'une panne présumée transitoire. La commutation vers le processeur de reprise est établie selon un protocole entre les deux stations. Il est possible dans pareil cas, d'envisager un transfert d'informations: Tel le graphe de reprise et autres (ces informations étant présumées exemptes d'erreurs).

III.2.2.1. Reprise:

La panne étant supposée détectée diagnostiquée, et la reconfiguration appropriée est effectuée. Le processeur de reprise procède de la façon suivante pour initier l'opération de reprise.

1- Emission d'un msg de renvoi (MDR) aléatoire, avec Rpi comme point de relance. Le msg comprendra également la nouvelle identification Rpi' de Rpi pour éviter les msg de mise à jour.

2- Restauration de l'état du processus défaillant Pi en Rpi' et réexécution.

Tout processeur Prj ayant reçu un MDR, exécute les actions suivantes:

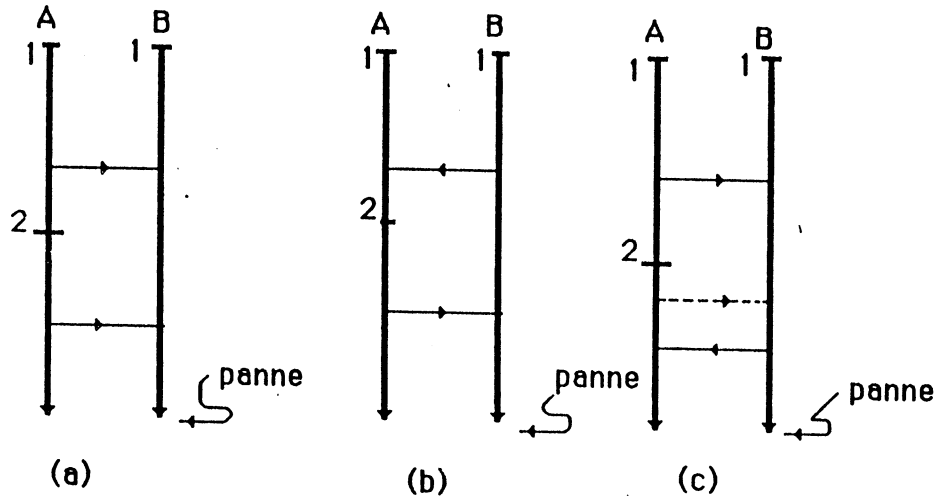
1- Détermination de Rpj appartenant à Pj telque Rpj --> Rpi' (Rpj est propagateur direct de Rpi').

2- Détermination d'un Rpk appartenant à Pj telque Rpi' --> Rpk (Rpk est dépendant direct de Rpi').

3- Si Rpk et Rpj existent, alors deux cas peuvent se présenter:

a- Rpk est dominant, alors la postpropagation vers Rpk l'emporte sur le renvoi des msg (le renvoi est implicite).

b- Rpj est dominant, alors le renvoi doit s'effectuer dans la ou les region(s) de reprise délimitée(s) par Rpj et Rpk. A partir de Rpk, la reprise se substitue au renvoi (voir figures ci-dessous).



Différents cas possibles de priorité
entre renvoi et postpropagation

Figure 03

Fig 3a : Le renvoi de msg s'effectue à partir de A1.
Fig 3b : La postpropagation prime le renvoi (renvoi implicite).
Fig 3c : Il y a renvoi dans la région de reprise associée à A1, puis reprise à partir de A2.

Bien entendu, lorsqu'un seul élément existe, Rpj ou Rpk, on procède soit au renvoi, soit à la reprise par postpropagation.

4- Si Rpk existe alors, il y a postpropagation.

. Renommer Rpk (soit Rpk');

. Emettre un msg de renvoi (qui tient lieu aussi de msg de mise à jour de Rpk), à tout propagateur direct de Rpk. Ce msg de renvoi reste soumis au cas 3 précédent;

. Emission d'un msg de postpropagation à tout dépendant direct de Rpk;

. Destruction de la liste LID (Rpk);

. Destruction des successeurs de Rpk;

. Restauration de l'état de Pj associé à Rpk;

. Réexécution effective. ;

III.3. Stratégie B sans postpropagation.

Le procédé pour satisfaire les critères (1,2), et les informations qui leurs sont nécessaires, restent identiques à ceux de la stratégie A.

III.3.1. Pannes transitoires.

L'élimination de la postpropagation par satisfaction des critères (1,2) d'une part, l'évitement de la rétropropagation par conservation, au niveau de chaque producteur, des msg émis d'autre part, fait que chaque processus se trouve isolé, vis-à-vis de ses partenaires, des effets de toute opération de reprise. Cette isolation attribue aux facteurs de propagation des valeurs optimales: Niveau n = 0, ordre p = 1.

Lorsqu'une panne est détectée, le processeur Pr du processus défaillant Pi exécute les actions suivantes:

1- Puisqu'aucune propagation n'est possible, Pr choisit comme RpR, le point de reprise actif Rpi, et émet un msg de renvoi aléatoire en précisant Rpi.

2- Pr restaure l'état du processus Pi en Rpi, puis lance la réexécution.

Tout processeur Prj recevant un msg de renvoi du type précédent, agit comme suit:

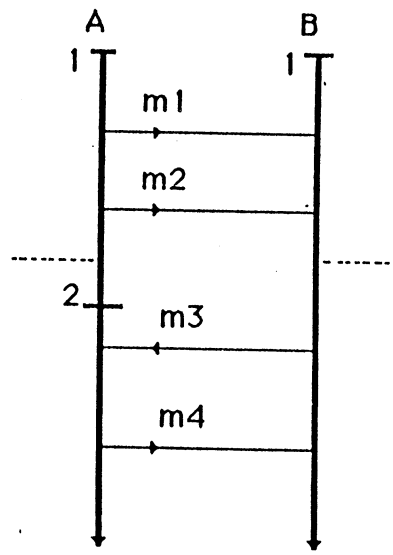
- Suspenssion éventuelle du traitement en cours;

- Détermination de l'ensemble des msg à renvoyer. Cette opération est accomplie grâce aux références fournies dans le msg (émetteur, RpR ou numéro du msg à partir duquel le renvoi sera effectué), et à la liste LD locale;

- Réémission effective des msg demandés.

III.3.1.1. Renvoi (ou réémission) des msg.

Une fois qu'un processeur ait déterminé la liste des msg à transmettre, et ait entamé la réémission, celui-ci n'est point à l'abri de pannes. Aussi une défaillance éventuelle le mettrait dans une situation de reprise telle qu'il doit continuer de fournir les msg réclamés par le processus antérieurement défaillant. Donc un historique des msg réémis doit être maintenu sauvegardé pour éviter des situations de perturbation du type ci-dessous.
L'exemple ci-après illustre de telles perturbations.



Cas de renvoi sans sauvegarde de
l'historique des msg réémis.
Figure 04

Soit le processus B en panne, il est repris en B1, un msg de renvoi (MDR) est émis à A qui doit renvoyer à B les msg m1,m2,m4. Supposons que A, après avoir réémis m1, tombe en panne lui aussi, ce qui le ferait reprendre en A2, sans toutefois manquer à la

réémission de m2, autrement une situation d'interblocage est possible (B1 attend m2 et A2 attend m3).

La satisfaction du critère 2 joue un rôle primordial en particulier, lorsque A ayant réémis m4, détecte une panne et décide de reprendre en A2. B n'étant nullement affecté par cette reprise, se verrait réémettre m3, sans avoir à se préoccuper ni de la réémission de m4 (puisqu'elle n'aura pas lieu), ni de sa conservation comme le veut la stratégie (puiqu'une version est déjà conservée).

III.3.2. Pannes permanentes.

Comme dans tous les cas de pannes permanentes, et dans l'hypothèse d'une reprise satisfaisante pour panne unique, les critères (1,2) sont satisfaits non pas par le processus défaillant Pi, mais par les processus propagateurs directs ou dépendants.
L'opération de reprise peut être accomplie par l'exécution des étapes suivantes.

- Un MDR aléatoire étant émis aux producteurs concernés, ce msg comporte comme RpR, le point de reprise Rpi le plus récemment enregistré;

- Restauration de l'état de Pi en Rpi;

- Relance effective de Pi.

Tout processeur Pr qui reçoit un MDR y donne suite en éxécutant les opérations suivantes:

. Détermination d'un Rpj ϵ Pj, Pj ϵ Pr tel que Rpj --> Rpi.

. La liste des pointeurs de msg à réémettre associée à Rpj étant déterminée, le processeur Pr entame le renvoi de msg, tout en assurant les dispositions nécessaires pour éviter les perturbations causées par une éventuelle reprise de Pj sur Pi.

On constate que les perturbations provoquées par les propagations demeurent optimales, puisque les valeurs des facteurs de ces dernières: Niveau n = 0, ordre p = 1, sont les plus faibles possibles.

Notons que pour des raisons d'optimisation de l'espace mémoire, chaque fois qu'un processus P établisse un nouveau point de reprise Rpi, il avise les producteurs concernés pour éliminer les msg conservés relatifs au point de reprise prédécesseur Rpi-1.

III.4. Capacités de reprise après récupération de la panne permanente.

Pour les mêmes raisons évoquées en stratégie A (II.2.3.1.2.), on ne peut reprendre plusieurs pannes permanentes sans avoir à admettre l'introduction de contrainte au niveau du déroulement des processus.

Cette situation a conduit à adopter dans cette première partie, le recouvrement d'un nombre important de pannes transitoires et d'une panne permanente unique. Cette attitude est dictée par le soucis d'éviter d'encourrir un overhead inacceptable.

Cependant, certaines stratégies sont moins efficaces que d'autres à assurer leurs capacités de reprise en pannes transitoires, après récupération de la panne permanente; c'est notamment le cas des stratégies B et D.

En effet, l'occurrence d'une panne permanente sur une station SPDi, volatilise l'ensemble des informations qui s'y trouvent, en particulier les msg conservés par les producteurs de SPDi. Il existe alors une station SPDj ayant antérieurement reçu des msg de SPDi, sur laquelle tout recouvrement de panne transitoire exigerait une action à la manière de la stratégie C (car absence de msg indispensables sur SPDi).

On n'évoque pas la possibilité d'une redondance de ces msg conservés sur plusieurs stations, puisqu'une telle solution ne ferait qu'accroître d'avantage les inconvenients relatifs à l'overhead généré, et à la taille de l'espace de conservation réclamée.

Donc, les stratégies B et D sont susceptibles d'"inefficacité" de reprise en pannes transitoires survenant postérieurement au recouvrement d'une panne permanente.

III.5. Perturbations et chaines de reprise.

1- Critères (1,2) satisfaits.

a- Panne transitoires.

Toute opération de reprise concerne l'unique processus défaillant. La propagation qui en résulte prend des valeurs optimales: Niveau n = 0, ordre p = 1.

b- Pannes permanentes.

La situation est identique qu'en (a), d'où les valeurs de propagation: n = 0, p = 1.

c-Sans distinction de catégorie de pannes, la propagation reste inchangée, d'où niveau n = 0, ordre p = 1.

2- Critères (1,2) non satisfaits.

Bien que la conservation des msg constitue un facteur important de limitation de la propagation interprocessus, la systématisation de la postpropagation peut générer des situations de longues chaines de reprise. Il n'est donc pas possible de contrôler cette propagation sans avoir à introduire des contraintes, tel serait le cas dans la partie II "Stratégies avec contrainte".

Notons toutefois que la stratégie B est relativement sensible à un accroissement du volume du traffic-msg, ce qui ne constitue guère de problème dans les réseaux locaux [DSA80,MAC79, MET76].

III.6. CONCLUSION.

L'introduction de la stratégie B a pour objectif une amélioration du traitement des pannes permanentes, visant à réduire les effets de perturbations interprocessus, donc d'entreprendre des opérations de reprise à moindre coût.

La répartition de la conservation des msg indispensables au processus défaillant; à travers les différents producteurs, constitue un avantage pour un système de stations physiques où la ressource mémoire y fait défaut. Cette répartition permet également de distribuer l'overhead issu de la gestion de ces msg, et d'avoir ainsi une pénalisation uniforme sur diverses stations.

Du côté inconvénients, on remarque la faiblesse de la stratégie à maintenir ses performances de reprise en pannes transitoires après traitement d'une panne permanente. Elle a également tendance à accroître le traffic-msg qui peut éventuellement "dégrader" les performances du système de transmission. Notons toutefois que, relativement aux pannes transitoires, la stratégie B s'avère moins performante que la stratégie A.
Les procédures de reprise de la stratégie B sont données en annexe.

Chapitre IV

STRATEGIE C

" Aucun msg n'est conservé ".

IV.1. Description générale.

La stratégie C par sa généralité et sa large "couverture" de panne est la plus fréquemment décrite dans la litérature qui traite de la tolérance aux pannes logicielles dans les systèmes infomatiques. Sa particularité vis-à-vis des autres stratégies qu'on définit, s'observe dans son indépendance à l'égard de la conservation des msg, ce qui éviterait donc l'overhead résultant de la gestion de ces derniers. Ceci constitue un avantage certain en absence de panne, mais en revanche créerait une situation de pénalisation importante du système à l'occurrence de panne.

En effet, l'indisponibilité locale des msg indispensables après l'occurrence de la panne (quelque soit son type) et au moment du lancement de l'opération de reprise, nécessite impérativement une rétropropagation. Quant à la postpropagation, il y a lieu de considérer l'alternative suivante. Elle serait obligatoire, car aucun des critères (1,2) n'est satisfait (ou bien, latence non nulle), elle serait évitée dans le cas contraire.

IV.1.1. Informations indispensables.

Comme dans les stratégies précédemment étudiées, pour pouvoir invoquer une rétropropagation ou une postpropagation, il est indispensable de disposer pour chaque processus P, d'un graphe de reprise décrivant son historique de communication. Ce graphe se réduit à sa forme la plus simple, il comprend les trois listes fondamentales:

- La liste LRp formée des Rp du processus P, dont chaque élément Rpi comprend trois pointeurs:
  - . Un pointeur désignant le contexte associé à Rpi;
  - . Un pointeur indiquant la liste LID de dépendants directs de Rpi;

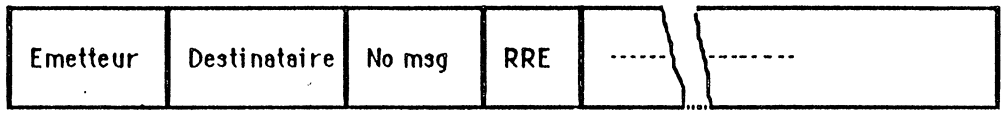
. Un pointeur désignant la liste LIP de propagateurs directs de Rpi.

- La liste LD de P, formée par l'union des listes LID de tous les Rpi du processus P.

- La liste LP de P, constituée par l'union des listes LIP de tous les Rpi de P.

Les informations entrant dans la formation de ces listes sont: Soit disponibles localement tels les Rp, soit extraites des msg d'information reçus tels les éléments des listes LIP, ou d'un protocole d'accord interprocessus relativement aux listes LID, de sorte que tout processus récepteur d'un msg, doit transmettre à son émetteur l'identificateur de la RRR.

La structure d'un msg peut être la suivante:



Structure d'un msg

Figure 01

L'illustration suivante montre un exemple de graphe de reprise pour cette stratégie C.

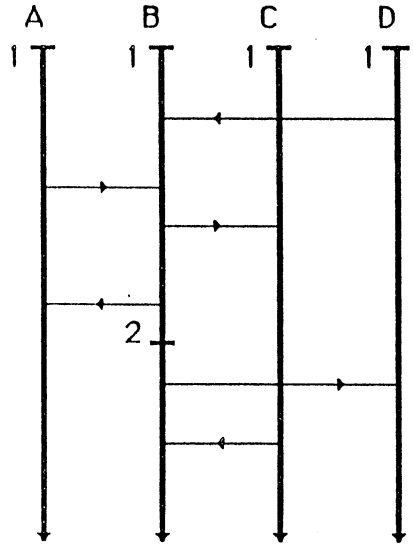
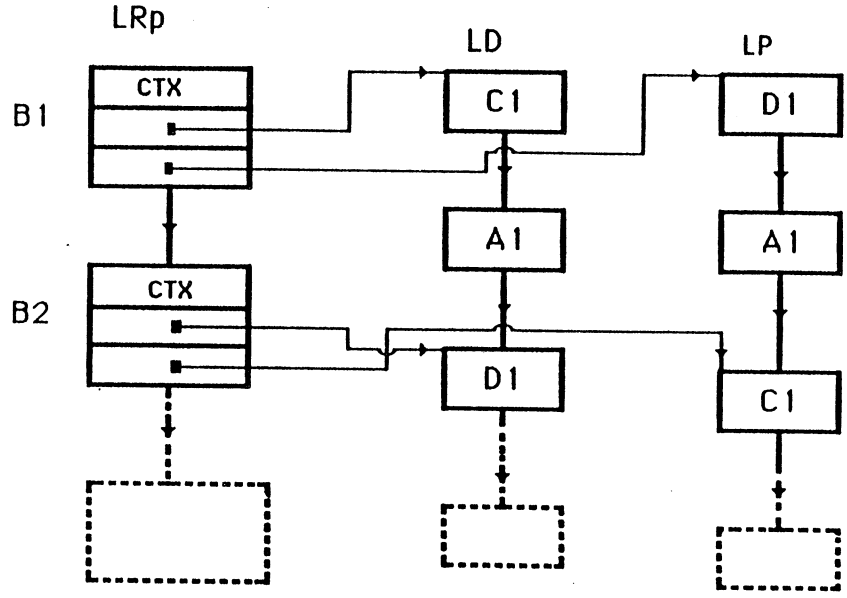


Figure 2a



Graphe de reprise pour le processus
B de la fig 2a

Figure 2b

La rétropropagation étant systématique, pour limiter les perturbations il serait fort intéressant d'éviter la postpropagation. Pour cela, les informations nécessaires pour faire valider les critères (1,2) sont identiques que pour les stratégies A et B. Elles consistent en: Les numéros des msg reçus et leurs RRE, les numéros des msg émis et leurs RRR (voir fig 6 stratégie A).

IV.2. Stratégie C sans postpropagation.

IV.2.1. Pannes transitoires.

La postpropagation étant éliminée, la rétropropagation s'impose de façon systématique. Les listes LD et LP étant présumées exemptes d'erreurs, il y a lieu d'invoquer la reprise de tout propagateur direct du RpR choisi (invocation précise ou aléatoire).

Soit Pi le processus défaillant et Pri le processeur qui exécute Pi, RpRi appartenant à Pi: Le point de relance choisi provisoirement par Pri. On adopte la méthode de détermination de la

ligne de reprise globale par synthèse des RL intermédiaires parvenues à Pri par ses dépendants.

Lorsqu'une panne est détectée, celle-ci étant a priori considerée comme transitoire. C'est le nombre de réessais de récupération sans succès qui la transforme en panne permanente, et qui déclenche le traitement approprié, en particulier la commutation notifiée par Pri au processeur de la station de reprise correspondante.

Soit donc la panne détectée et diagnostiquée, Pri étant informé, procède à l'exécution des actions suivantes:

1- Emission d'un msg d'enquête de RL intermédiaires. Ce msg consiste à informer les propagateurs directs de RpRi de la décision de Pi de reprendre en ce point. Pri, une fois le msg émis en y spécifiant RpRi, se met en attente de réponses des propagateurs, pour entamer réellement l'opération de reprise (entre temps un autre processus peut être exécuté pour éviter une oisiveté de Pri).

Tout propagateur direct qui reçoit ce msg agit à son tour comme Pri. C'est à dire, il émet un msg d'enquête à ses propres propagateurs. Lorsqu'un propagateur s'avère être une feuille dans cet arbre de la hiérarchie (voir section Roll-back direct), il communique alors au noeud père, une RL intermédiaire dans laquelle il est impliqué, puis se met en attente de réponse.

De proche en proche, le noeud racine représenté par le processus Pi, recevra un ensemble de RL intermédiaires dont Pri fera la synthèse pour en tirer la RL globale. Aprés acquisition de cette RL globale, un msg d'invocation précise est émis à chaque processus concerné.

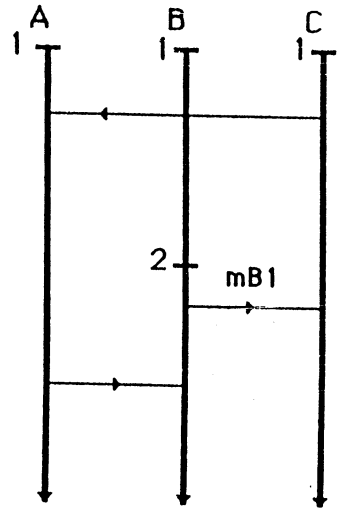
2- Renommage du RpR, et émission d'un msg de mise à jour à chaque dépendant de RpR appartenant à un processus non repris.

3- Destruction des successeurs de RpR (s'il en existe) et de sa liste LIP.

4- Restauration de l'état du processus défaillant en RpR, et réexécution de Pi.

Les critères (1,2) étant satisfaits, chaque émission de msg après opération de reprise, est soumise au préalable à un contrôle de validation de ces critères.

Nous adoptant le principe de recherche de RL par msg d'enquête plutôt que par roll-backs progressifs qui s'avère insuffisant comme on peut le constater dans l'exemple suivant.



Insuffisance du roll-back progressif.

Figure 03

Soit B le processus défaillant, le premier roll-back aura lieu à partir de B2, par rétropropagation, A se relance en A1 et C en C1. La postpropagation étant évitée (inutile dans ce cas), le msg émis après B2 lors de la première exécution, ne sera pas émis à la seconde. Ainsi, il est probable que B reçoive un msg de reprise émis par C (atteint par la rétropropagation) alors que B se retrouve après le msg mB1. Deux cas sont alors envisageables:

1- Le RpR: B2 n'a pas été renommé, alors B se relance une seconde fois et provoquera une seconde rétropropagation, ce qui pourrait engendrer un cycle de la même opération de reprise si des moyens adhoc, pour éviter cette situation, n'ont pas été appliqués.

2- Le RpR, B2 a été renommé, le msg de reprise émis par C tardivement (conséquence des délais de transmission) n'est pas pris en compte, alors C restera bloqué en attente de msg ou recevra ultérieurement un msg de B qui provoquerait une perturbation en C. Cette situation peut être évitée par transmission d'un msg de mise à jour à C juste après renommage de B2, on retombe alors dans le premier cas.

Donc, la stratégie C avec critères (1,2) satisfaits, nécessite parfois de faire une "entorse" au maintien de ces critères. En fig 3, pour éviter deux roll-backs en B2, le msg mB1 doit être réémis à la première exécution en sachant bien sûr que B2 est son propre propagateur indirect par le principe de roll-back direct (RL globale). On évite ainsi le cas 2 précédent.

Remarque:

Pour pouvoir détecter un cycle de propagation et éventuellement le renommage des RpR et les msg de mise à jour qui en résultent, un mécanisme de reconnaissance d'une opération de reprise est nécessaire. Par exemple à chaque msg de reprise émis, on y transmet également les propagateurs ou les dépendants déjà reçus et le RpR courant (exemple fig 3, B transmettra <B2> à A qui transmettra à son tour à C <B2,A1> lequel transmettra à B <B2,A1,C1>.

IV.2.2. Pannes permanentes.

Comme dans tous les cas de ce type de pannes, et compte tenu des informations en possession du processeur de reprise, seule une invocation aléatoire est possible. On suppose évidemment que le code et les Rp de chaque processus d'une station potentiellement défaillante sont enregistrés sur la station de reprise associée.

Hypothèses:

- On suppose que le processeur de reprise Pri "connaît" l'ensemble des processus dépendants et propagateurs directs du processus défaillant (un msg diffusé répond parfaitement à l'hypothèse).

- Une panne unique est récupérable (pour plusieurs, voir partie II).

- On suppose également que la panne est détectée, diagnostiquée et la reconfiguration appropriée réalisée (voire algorithmes en annexe).

Soient Pi le processus défaillant, Pri le processeur de reprise; soit RpRi le RpR choisi provisoirement par Pri pour le processus Pi (RpRi est le plus récemment enregistré par Pi).

Pour les raisons invoquées juste avant, on adopte le principe de détermination de RL globale par msg d'enquête (roll-back direct).

1- Le processeur de reprise Pri, émet un msg d'enquête diffusé à l'ensemble des processus interactifs, puis se met en "attente" de réponses.

2- Tout processus Pj appartenant au processeur Prj recevant un tel msg, détermine s'il existe Rpj appartenant à Pj tel que Rpj --> RpRi. Si oui alors, existe t-il Rpk appartenant à Pk processus de Prk tel que Rpk --> Rpj ? Si oui, émettre msg d'enquête à Pk (Pj est un noeud) et attendre. Sinon (Pj est une feuille), Pj transmet sa RL intermédiaire < RpRi,Rpj> à Pri et attendre.

Donc lorsqu'un processus reçoit un msg d'enquête, il détermine sa position dans l'arbre de "rétropropagation":

- Soit il est feuille, alors il communique la RL intermédiaire, formée de son RpR et de celui spécifié dans le msg, au noeud père.

- Soit il n'est pas feuille, alors il joue le rôle d'un noeud père.

Cette transmission de RL intermediaires des fils vers les pères permettra de rassembler l'ensemble des RL au niveau du processus initiateur de reprise (racine), où sera reconstituée la RL globale.

3- Une fois la RL globale est déterminée, le processeur initiateur de la reprise (ou racine) procède à une invocation précise, puis détruit les successeurs de son RpR (s'il en existe) et supprime la liste LIP de ce dernier. Concernant la liste LID de RpR, les éléments appartenant à des processus non affectés par la reprise, demeurent inchangés. Par contre, ceux des processus impliqués dans la rétropropagation seront, soit soumis à des mises à jour, soit détruits pour être reconstitués avec de nouveaux identificateurs lors des interactions.

4- Renommage du RpR (ou bien adopter un moyen d'identfication de l'opération de reprise). Restauration de l'état du processus correspendant au RpR, puis réexécution effective.

Remarque:

On constate qu'il peut s'avérer fastidieux de mettre à jour les éléments des listes LID (3) surtout lorsque de nouveaux successeurs sont créés. Il serait alors plus facile de détruire aussi la liste LID (3) et réémettre tous les msg, tout en réalisant le critère 1. Ce qui permet la reconstruction de cette liste comme au cours de la première exécution. Ainsi à la réception, un msg peut être éliminé sans réutilisation (grâce à son numéro) mais donnerait lieu à un renvoi de RRR, ce qui permettrait la construction au niveau de l'émetteur de la liste LID.

IV.3. Stratégie C avec postpropagation.

Comme mentionné précédemment en stratégies A et B, et c'est valable pour l'ensemble des autres stratégies, provoquer systématiquement la postpropagation, donc ignorer les critères (1,2), relève plutôt d'une prise en compte d'une latence non nulle, que des possibilités de faisabilité. On va donc considérer la propagation sous toutes ses formes, ce qui peut engendrer, par l'incapacité de pouvoir la contrôler, de longues chaines de reprise.

IV.3.1. Pannes transitoires.

Hypothèse:

La panne est détectée et diagnostiquée, le graphe de reprise du processus défaillant Pi est inaltéré.

L'étape fondamentale de toute opération de reprise est la recherche de la RL commune à l'ensemble des processus interactifs. Deux approches y ont été proposées:
- par roll-backs progressifs;
- par roll-back direct.

a- Par roll-backs progressifs. Le processeur de reprise Pri, après détection, exécute les actions suivantes.

1- Soit Rpi le Rp actif de Pi; Pri choisi Rpi comme point de relance (RpR). Emission d'un msg d'invocation aléatoire à l'ensemble des processus Pj partenaires de Pi, en y précisant Rpi (une invocation précise est également possible).

2- Destruction des listes LID et LIP de RpR, et renommage de RpR. Restauration de l'état du processus Pi au point de relance Rpi, et réexécution de Pi.

Tout processus Pj ayant reçu un msg du type précédent, détermine s'il est impliqué ou non dans cette opération de reprise. Autrement dit, il vérifie les assertions suivantes:
- Existe-t-il Rpj appartenant à Pj tel que Rpj --> Rpi ? et
- Existe-t-il Rpk appartenant à Pj tel que Rpi --> Rpk ?.

Dans l'affirmative, et Rpk différent de Rpj alors, Pj choisit comme RpR, le plus dominant des deux. Dans le cas où un seul de ces points existe, il est pris comme RpR de Pj. Le RpR étant trouvé, les actions suivantes sont alors exécutées:
- Provoquer la reprise de tous les dépendants et propagateurs directs de RpR sauf Rpi;
- Destruction des successeurs de RpR (s'ils existent);
- Destruction des listes LID et LIP de RpR;
- Renommage de RpR et restauration de l'état de Pj en RpR;
- Relance de Pj.

La différence d'action entre l'initiateur de reprise et les autres processeurs concernés n'est pas trés significative. Après l'émission du msg d'invocation (destiné aussi à l'initiateur lui-même), l'ensemble des processus interactifs agissent de la même façon, ce qui permettra d'avoir un algorithme de reprise unique. On remarque également la facilité avec laquelle on peut réaliser cette opération de reprise.

b- Le principe de recherche de la RL par roll-back direct (i.e. par msg d'enquête) est décrit dans la section "roll-back direct". Une fois la RL globale déterminée, l'initiateur de reprise procède à une invocation précise, renomme son RpR, restaure l'état du processus défaillant, et relance l'exécution de ce dernier.

IV.3.2. Pannes permanentes.

Il n'y a vraiment pas une grande différence avec les pannes transitoires au niveau de l'opération de reprise. L'invocation aléatoire étant impérative pour le processeur de reprise, les autres partenaires peuvent user également de l'invocation précise particulierement vis-à- vis du processus défaillant.

L'hypothèse de reprise d'une panne unique étant adoptée, aucune limitation sur le nombre de pannes transitoires à reprendre n'est considerée. Le processeur de reprise après détection, diagnostic et reconfiguration, lance l'opération de recouvrement en agissant comme suit:
Soient Pi le processus défaillant, Pri le processeur de reprise; Rpi appartenant à Pi le dernier Rp communiqué à Pri lors de la première exécution de Pi. On adopte le pricipe de recherche de RL globale par roll-back direct, puisque la méthode de roll-backs progressifs donnera les mêmes actions qu'en pannes transitoires.

- Pri lance donc un msg d'enquête de RL intermédiaires à l'ensemble des processeurs Prj exécutant les processus interactifs. Ce msg comportera en particulier Rpi, après quoi, Pri se met en attente de réponses (ou exécutera pendant ce temps d'autres processus). Il est à noter que Pi joue désormais le rôle d'un noeud racine dans la hiérarchie de propagation.

- Lorsqu'un processus Pj (plutôt Prj, le processeur exécutant Pj) reçoit un msg du type ci-dessus, il doit déterminer sa position dans la hiérarchie précitée, à savoir est-il un noeud intermédiaire ou une feuille? Si Pj est feuille alors, il renvoit à son père immédiat la RL intermédiaire qu'il estime convenir, puis se met en "attente" de msg de reprise. Si Pj est noeud alors, il émet aux processus dépendants directs et propagateurs directs un msg d'enquête, et se met ensuite en "attente" de RL intermédiaires provenant de ces derniers (voir roll-back direct).

On observe donc une émission de msg d'enquête de la racine vers les feuilles et un renvoi de RL provenant de plusieurs fils pour en déduire, au niveau du père, une RL unique. Le processeur de reprise Pri après avoir récupéré l'ensemble des RL intermédiaires, en détermine une RL globale, puis exécute les actions suivantes:

1- Emission d'un msg d'invocation précise à tous les processus impliqués, connus par l'intermédiaire de la RL globale;

2- Destruction de tous les Rp successeurs du RpR déterminé grâce à la RL. Destruction des listes LID et LIP de ce nouvel RpR, soit Rpi' celui-ci (Rpi peut être Rpi');

3- Restauration de l'état de Pi en Rpi' et relance de Pi.

IV.4. Perturbations et chaines de reprise.

IV.4.1. Cas de reprise sans postpropagation.

Pour pouvoir contrôler les perturbations issues d'une opération de recouvrement de panne, et éviter de ce fait de longues et coûteuses chaines de reprise, il est indispensable de s'assurer du contrôle des deux types de propagation. Ainsi, si la satisfaction des critères (1,2) permet d'empêcher la postpropagation, la rétropropagation s'avère obligatoire, et capable d'annuler les effets de la première.

Dans pareilles conditions, il n'est pas possible d'évaluer ou de borner, en termes de paramètres de propagation (n,p), les conséquences d'une opération de reprise. Notons toutefois que la

probabilité de dégénérescence en effet-domino est beaucoup moindre que lorsque les critères ne sont pas satisfaits. Ceci est notamment vrai dans les systèmes de processus communicants où les échanges assymétriques prédominent (i.e: l'émission d'un msg ne donne pas forcement suite à une prochaine réception ou réciproquement).

IV.4.2. Cas de reprise avec postpropagation.

Quelque soit le type de panne considéré, l'insatisfaction des critères (1,2) implique inévitablement l'invocation (selon nécessité) des deux types de propagation. Aucun contrôle n'est donc possible, et les perturbations qui en découlent se stabilisent après avoir atteint, de façon "naturelle", un état cohérent du système. Cet état peut être déterminé après révocation éventuelle d'une importante activité. Dans pareil cas, on ne peut non plus évaluer, ni la longueur des chaines de reprise, ni le niveau des propagations. Cependant la probabilité qu'un système puisse dégénérer en effet-domino est susceptible d'atteindre des valeurs maximales relativement aux autres stratégies. De façon générale, et vis-à-vis des perturbations, la stratégie C est la plus sensible à l'effet-domino.

IV.5. CONCLUSION.

La stratégie C qu'on vient d'étudier permet certes, de reprendre des processus affectés par des pannes matérielles, tant transitoires que permanentes, mais son application demeure spécifique aux systèmes peu exigeant en délais de réponse. En effet, l'incapacité de pouvoir contrôler de façon certaine la propagation interprocessus, fait que le coût encouru lors d'une opération de reprise est susceptible d'atteindre des proportions incompatibles à l'égard des objectifs requis par certaines applications (temps réel).

Cependant, compte tenu de l'overhead minimal créé lors de l'exécution normale du système, et de la couverture de reprise qu'elle assure, la stratégie C est applicable dans bien des systèmes à vocation non-stop, où la contrainte temps de réponse peut parfois subir certaines fluctuations.

L'élimination de la postpropagation peut jouer un rôle important dans la limitation des longues chaines de reprise, sans pour autant pouvoir les délimiter. Cette situation, dûe principalement à la liberté totale accordée à l'évolution des processus, se trouve largement basculée en faveur d'une nécessité d'un contrôle de la propagation et ce, en imposant une contrainte de coordination au niveau de la création des Rp (voir partie II).

L'aspect positif de la stratégie, telle qu'elle vient d'être présentée, réside notamment dans les points suivants:

- Une plus grande confiance sur l'élimination totale des effets de la panne (prise en compte implicite d'une latence non nulle);
- Une liberté totale accordée aux processus au cours de leurs exécutions;
- Une pénalisation minimale de l'application au cours de l'exécution normale;
- Une mise en oeuvre de la stratégie relativement simple.

Quant à l'inconvenient majeur affaiblissant la stratégie, celui-ci se situe au niveau d'une création possible de perturbations importantes à l'occurrence de panne. Ces perturbations sont

provoquées par l'absence de contrôle de propagations qui peuvent résulter d'une opération de reprise.

Chapitre V

STRATEGIE D

"Les msg reçus par la station potentiellement défaillante (SPD), sont captés et conservés par la station de reprise associée (SR)".

V.1. Description générale.

Si la conservation des msg en stratégie A joue un rôle fondamental au niveau des propagations de reprise à l'occurrence des pannes transitoires, les pannes permanentes ne bénéficient pas de ce privilège. C'est pourquoi la stratégie D se propose de répondre à un traitement mieux adapté et moins coûteux des pannes permanentes sans pour autant sacrifier celui des pannes transitoires.

En effet, tous les msg reçus par une station potentiellement défaillante sont captés, et conservés pour une utilisation ultérieure possible, dans la station de reprise correspondante. Ainsi, à l'occurrence de panne et au cours de la reprise on peut adopter l'alternative suivante:

- La panne est transitoire, alors le processeur de reprise, pour pouvoir relancer le processus défaillant, sollicite de la station de reprise l'émission des msg indispensables à la relance;

- La panne est permanente, la station défaillante devient inopérante et la communication s'effectue vers la station de reprise où l'environnement adéquat à la réexécution se trouve disponible. Ainsi le processus défaillant est repris sans difficulté sur la station de reprise.

Il est à remarquer que, l'interruption des processus (suite aux pannes), leurs relances par les mécanismes de reprise, et d'une façon générale toute cette activité extra-application n'est pas perçue de l'extérieur du système; sinon qu'une dégradation sensible des perfermances.

Outre l'espace nécessaire à cette conservation et les traitements qui en découlent, l'influence sur le traffic msg,

particulièrement la charge du système de transmission, constitue un facteur important qui contribue à améliorer de façon notable l'efficacité de la stratégie.

C'est ainsi que la stratégie D, s'adapterait beaucoup mieux sur des médiums de communication favorisant la diffusion (par exemple les réseaux locaux du type Ethernet (MET 76]).

V.1.1. Informations indispensables.

Les informations fondamentales exigées par une implèmentation de la stratégie se répartissent en deux catégories:

- Celles nécessitées par la construction du graphe de reprise et pour la gestion de la conservation des msg. Autrement dit, les informations permettant de lancer l'opération de reprise souhaitée.

- Celles permettant de satisfaire les critères (1,2) de non post-propagation.

Concernant le graphe de reprise, il sera identique à celui de la stratégie C, auquel on adjoint un moyen permettant au processeur de reprise (cas de pannes transitoires) de pouvoir demander les msg indispensables de façon adéquate. L'exemple suivant montre une représentation possible de ce graphe:

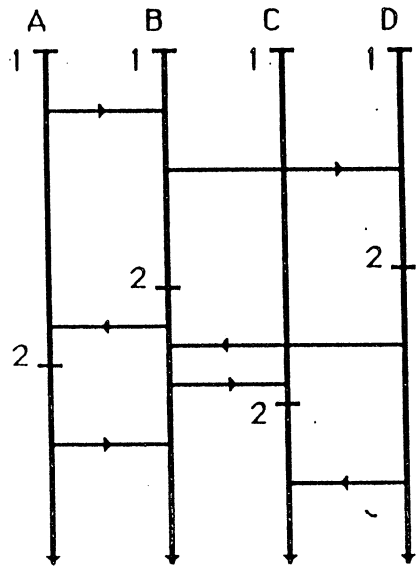
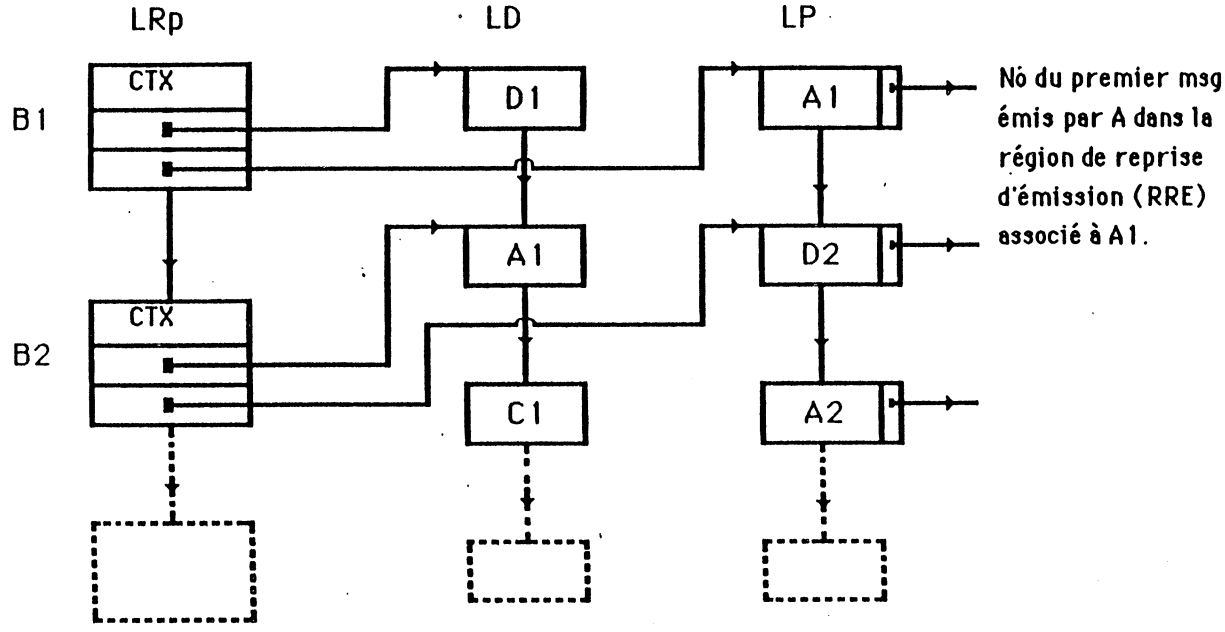


Figure 1a



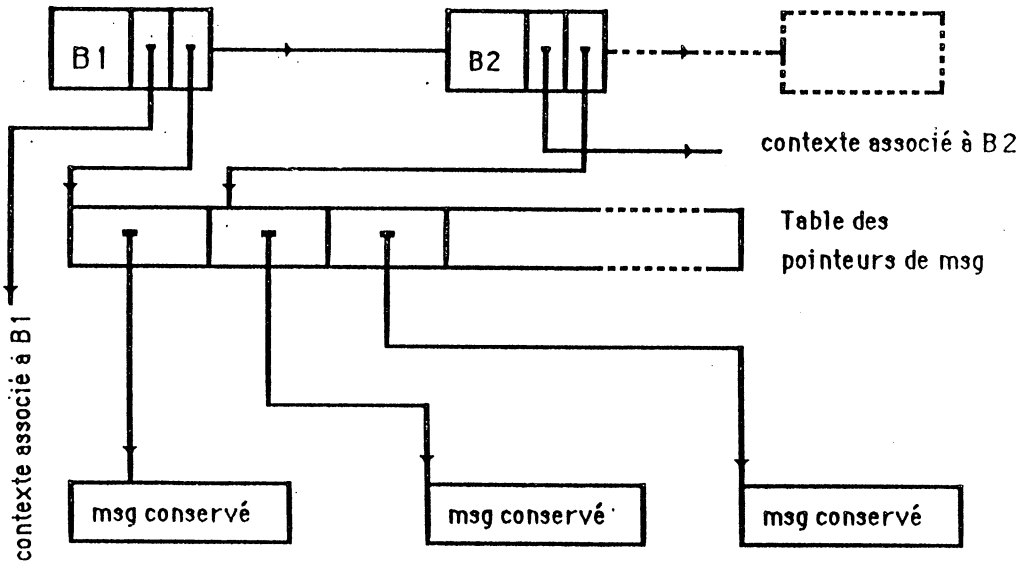
Graphe de reprise pour B.

Figure 1b

Quant aux informations indispensables à la réalisation des critères de non postpropagation, elles demeurent identiques à celles décrites dans les stratégies précédentes (A,B,C).

Du côté stations de reprise, chacune d'elle est tenue de conserver les msg captés, selon une structure de données capable de fournir (en réponse à une demande) les msg indispensables désirés. Cette "fourniture" doit se faire sans création de perturbation au niveau de la réception.

Une manière possible de réaliser cette tâche consiste, à enregistrer l'ensemble des msg captés, dans une liste gérée selon la stratégie First-in, First-out. Ces msg doivent avoir une correspondance avec les Rp de la station défaillante. Par exemple (fig 1a) si B est le processus défaillant et "D" la station de reprise associée, alors:
Sur la station B, on y construira le graphe de la fig 1b et sur "D" on pourra établir la structure suivante.



Structure de données possible adaptée à une Station de reprise ("D") pour le processus Défaillant B.

Figure 02

La station de reprise qui se charge continuellement de capter et de conserver tous les msg d'informations destinés à la SPD associée, doit se charger également de préparer l'environnement de travail nécessaire au mécanisme de reprise. Les éléments de cet environnement qui permettent d'établir la structure de données appropriée, sont en partie extraits des msg saisis tels:

- L'identificateur de l'emetteur,
- La RRE et
- Le numéro du msg émis associé à cette RRE.

Cependant, pour pouvoir recouvrer une panne permanente, il est fondamental, après avoir choisi ou déterminé un RpR , de reconnaître exactement les msg à réutiliser. C'est-à-dire, les msg reçus dans la RRR identifiée par le RpR. Pour cela, la SR peut capter les msg de renvoi des RRR émis par la SPD vers ses émetteurs, et établir ainsi la relation entre Rp conservés et les msg captés.

On constate alors une activité "parasite" contribuant à accroître l'overhead dans la SR. C'est pourquoi, il serait plus

"économique" que chaque fois qu' un Rp (plutôt le contexte associé) est communiqué par la SPD à sa SR, celui-ci peut être accompagné:
- Du numéro du dernier msg reçu dans la RRR précédant immédiatement la création de cet Rp, et
- De la RRE qui lui est associée.

De cette façon sera établie la structure de données indispensable à la SR pour pouvoir traiter une éventuelle panne permanente (fig 2). Notons que la structure des msg d'informations reste identique à celle définie dans les stratégies précédentes.

V.2. Stratégie D sans postpropagation.

V.2.1. Pannes transitoires.

La validation des critères (1,2) implique l'élimination de la postpropagation, de même, le renvoi des msg indispensables par la SR à la SPD associée résulte un évitement de la rétropropagation. Ainsi, chaque occurrence de panne n'influe que sur la région de reprise courante du processus affecté; ce qui donne un résultat optimal au niveau des perturbations. On peut alors quantifier la propagation de reprise suivant les valeurs prises par ses facteurs: niveau n = 0, ordre p = 1.

On rappelle que dans pareil cas (pannes transitoires), l'adoption du critère 2 s'avère la plus efficace puisqu'elle évite les émissions de msg inutiles. La réalisation en est plus aisée et consiste, après la relance du processus défaillant, à soumettre toute émission de msg à un contrôle l'infirmant ou le confirmant. L'élément indispensable à cette opération de contrôle est le compteur de msg émis qui serait sauvegardé et restauré en même temps que les Rp des processus.

V.2.1.1. Reprise.

Hypothèses:

- Tous les msg reçus par la SPD sont captés et conservés par la SR et ce, dans le même ordre.
- Les contextes associés aux Rp sont exempts d'erreurs (idem pour

le graphe de reprise).

Soient Pi le processus défaillant, Rpi son Rp actif. Le processeur de reprise Pri, après détection et diagnostic de la panne, exécute les actions suivantes (cf. algorithmes en annexe):

1- Emission (par Pri) d'un msg de renvoi des msg indispensables à la SR associée. Ce msg doit précisé en particulier, le numéro du premier msg reçu dans la RRR identifiée par Rpi, ainsi que l'identificateur du Rp propagateur de Rpi (fig 1b). Une structure possible du msg de renvoi peut être la suivante:

Emeteur	Récepteur	Type(renvoi)	No du PMR	RRE	— · · · · — · · · ·

PMR: Premier msg reçu

Structure d'un msg de renvoi.
Figure 03

2- Une fois le msg émis, le processeur Pri restaure l'état de Pi en Rpi et relance Pi.

Après quoi, s'instaurent le schéma classique du producteur-consommateur entre les stations SR et SPD, et le mécanisme chargé d'assurer le critère 2 de non postpropagation.

Du côté de la station SR, rien n'exclut une éventuelle panne. Et si l'on veut éviter une propagation ou une perturbation, la station est tenue d'identifier tous les msg renvoyés pour assurer à son tour ultérieurement le critère 2. Autrement dit, garder une trace du renvoi effectué.

V.2.2. Panne permanente.

Hypothèses:

Outre les hypothèses envisagées en pannes transitoires, on rajoute celle de récupération d'une panne unique. On élimine ainsi la

possibilité d'absence d'une SR. Le cas où plusieurs pannes surviennent mais ne concernent jamais un couple (SR,SPD) est traitable à condition de définir une stratégie de désignation de stations de reprise.

V.2.2.1. Reprise.

Soient Pi le processus défaillant, Rpi appartenant à Pi le Rp le plus récemment enregistré par Pri (Pri processeur de reprise de Pi). Après détection, diagnostic et reconfiguration, Pri agit comme suit:

1- Restauration de l'état de Pi en Rpi;

2- Relance de Pi.

On constate que l'opération de reprise en soi est relativement simple. Cependant Pri en réexécutant Pi doit se charger aussi de la reconstitution de son graphe de reprise à partir de Rpi.
Les informations nécessaires à cet effet sont fournies d'une part, par les msg conservés localement (RRE, numéro du msg...) et d'autre part, par les processus dépendants de Pi auxquels il appartient d'assurer le critère 1. Ces derniers doivent également respecter le protocole qui consiste à transmettre à Pi les identificateurs des RRR des msg reçus de nouveau, sans qu'il soit accordé à ces msg une quelconque action.

V.3. Stratégie D avec postpropagation.

V.3.1. Pannes transitoires.

Les critères (1,2) n'étant pas satisfaits, la postpropagation s'avère donc systématique. Seule l'évitement de la rétropropagation permet d'atténuer d'éventuelles situations de longues chaines de reprise. Si dans le cas précédent chaque RpR appartient à la RL, dans le cas présent, la détermination de la RL de façon explicite, selon une des deux méthodes précédemment décrites (roll-backs directs ou progressifs), est impérative.

les hypothèses restent identiques à celles du cas V.2. Soient Pi le processus défaillant, Pri le processeur qui l'exécute; Rpi appartenant à Pi le point de reprise actif.

V.3.1.1. Reprise.

Une fois la panne détectée et diagnostiquée, Pri exécute les actions suivantes:

1- A l'aide d'un msg de reprise, Pri invoque chaque dépendant de Rpi à reprendre (invocation précise ou aléatoire).

2- Détruit tous les successeurs de Rpi (s'il en existe) et la liste LID de Rpi.

3- Renomme Rpi pour éviter les retours de propagation fastidieux et inutiles vers Rpi, et envoie un msg de mise à jour à chacun des propagateurs directs de Rpi.

4- Demande, par un msg de renvoi, à la station de reprise associée les msg reçus dans la RRR identifiée par Rpi (identique qu'en cas V.2.).

5- Restauration de l'état de Pi en Rpi et réexécution.

Tout processus Pj dépendant de Pi et ayant reçu un msg de reprise exécute des actions similaires aux (1,2,3,4,5).

V.3.2. Panne permanente.

Hypothèses:

Toute station défaillante SDi, possède une station de reprise SRj opérationnelle. Les hypothèses adoptées au cas V.2. sont bien entendu maintenues.

Dans pareil cas, il est clair que le processeur de reprise Pri soit contraint de procéder à une invocation aléatoire pour provoquer la reprise au niveau des processus Pj, dépendants du processus défaillant Pi. Un retour éventuel de propagation vers Pi ne peut se faire que par l'intermédiaire d'une invocation précise.

On rappelle que pour pouvoir continuer à traiter les pannes transitoires après avoir déjà traité une panne permanente, il est indispensable qu'une stratégie de désignation de stations de reprise soit adoptée. En effet, lorsqu'une station SPDi est station de reprise d'une autre station SPDj, une défaillance de SPDi

priverait SPDj de l'ensemble des msg conservés. Cette situation ne permet guère une reprise éventuelle de SPDj, à moins d'agir comme en stratégie C (imposer une rétropropagation de tous les processus propagateurs directs du défaillant).

V.3.2.1. Reprise:

Que la détermination de la RL fasse partie intégrante de l'opération de reprise (roll-backs progressifs) et devient ainsi implicite, ou qu'elle constitue une procédure annexe (roll-back direct), les étapes de la reprise demeurent identiques dans les deux cas.

Soient Pi le processus défaillant, Pri son processeur de reprise et RpR le point de relance de Pi. Après avoir détecté et diagnostiqué la panne, et après reconfiguration matérielle et éventuellement logicielle, Pri réalise les opérations suivantes:

1- Emission d'un msg d'invocation aléatoire à tous les dépendants directs de Pi, en précisant RpR (roll-backs progressifs).

2- Détermination des msg indispensables correspondant à RpR, et transfert dans la file d'attente des msg de Pi.

3- Renommage de RpR et émission d'un msg de mise à jour à tous les propagateurs directs de RpR (msg diffusé).

4- Restauration de l'état de Pi en RpR, et réexécution de Pi.

Remarque:

L'étape 3 peut être incluse dans l'étape 1 et on évite ainsi l'emission de msg de mise à jour.

Les étapes ci-dessus caractérisent l'attitude du processus de reprise, attitude qui n'est pas totalement identique à celles des processus dépendants ou propagateurs de Pi. Lorsqu'un processeur Prj reçoit un msg d'invocation aléatoire (diffusé), il détermine le rôle qu'il doit assumé dans cette opération de reprise globale. Prj exécute donc les actions suivantes:

a- Il détermine s'il existe un processus Pj, tel qu'il existe Rpj appartenant à Pj et Rpj --> RpR (RpR dépendant direct de Rpj). Si tel est le cas, alors Prj procède à la mise à jour de RpR (le remplacer par son nouvel identificateur spécifié dans le msg d'invocation (l'étape 1 comprend l'étape 3'). Sinon, existe-t-il Rpj ∈ Pj tel que RpR --> Rpj (RPR propagateur direct de Rpj)? Si un tel Rpj n'existe pas alors Prj n'est point concerné par la reprise. Dans le cas contraire Prj y est impliqué, il agit alors en exécutant les actions suivantes.

b- Prj ayant déterminé Rpj comme point de relance, doit provoquer la postpropagation de tous les dépendants de Rpj.

1- Emission d'un msg d'invocation précise à tout dépendant de Rpj (éléments de la liste LID de Rpj). Une invocation aléatoire est possible sauf pour Pi auquel une invocation précise est obligatoire. Il serait avantageux de déteminer si Prj est propagateur d'un Rp appartenant à Pi et dominant de RpR, auquel cas émettre une invocation précise, puis émettre une invocation aléatoire à tous les autres en y incluant le nouvel identificateur du point de relance.

2- Destruction de tous les successeurs de Rpj (s'il en existe) et émission de msg de mise à jour à tous les propagateurs de Rpj. Destruction de la liste LID de Rpj.

3- Demande de msg indispensables à la station de reprise de Prj, par l'émission d'un msg de renvoi.

4- restauration de l'état de Pj en Rpj, et réexécution.

Dans le cas où le principe de roll-back direct est adopté, l'opération de reprise, de façon générale, sera abordée en deux étapes:

1- Détermination de la RL globale et émission de msg de reprise à tous les processus consernés y compris le défaillant lui-même.

2- Réaction des processus récepteurs vis-à-vis du msg de reprise. Les actions ainsi effectuées sont identiques à celles des pannes transitoires (stratégie C, IV.3.1.).

V.4. Perturbations et chaines de reprise.

Commme il a été déjà exprimé au cours de l'étude, l'efficacité d'une stratégie se mesure en fonction de certains paramètres, dont le plus important s'avère être l'overhead encourru par le système. Ce dernier étant introduit par l'élaboration de l'environnement de travail nécessaire au mécanisme de reprise d'une part, et d'une révocation des activités lors de l'opération de reprise d'autre part.

La stratégie D, par la disponibilité locale des msg indispensables à l'occurrence de panne permanente, et par l'acquisition de ces derniers au niveau de la SPD via la SR associée, répond favorablement à l'objectif d'une minimisation des perturbations inter-processus lors des actions de reprise. Ces perturbations, et l'overhead qui en découle, sont d'autant plus optimisés que les critères (1,2) sont satisfaits.
Si dans ce cas, il est possible d'évaluer la propagation génératrice des perturbations, avec niveau n = 0, et ordre p = 1; il n'est pas possible dans le cas plus générale, avec critères (1,2) non satisfaits, de pouvoir lui attribuer une limite, si ce n'est une estimation probabiliste.

En effet, toute propagation est régie par deux composantes: la rétropropagation, annulée par la conservation des msg, et la post-propagation, qui s'impose de façon systématique, font que les effets de l'une sont susceptibles d'être annulés par l'autre. Ainsi, on ne peut qu'affirmer qu'il est moins probable d'avoir de longues chaines de reprise avec une rétropropagation évitée que dans le cas contraire. Ce qui est d'ailleurs d'autant plus valable que le système est à interactions assymétriques, où le nombre de processus consommateurs de msg est important devant celui des producteurs (par exemple un producteur et plusieurs consommateurs).

Concernant les transmissions, plus la charge du système est importante, plus est important le taux d'erreurs, et plus cette charge augmente d'avantage. Ainsi, au niveau d'une SR, en plus des interruptions d'activité provoquées par la capture et la conservation des msg destinés à la SPD, le renvoi de ces msg solicité par cette dernière à l'occurrence de panne transitoire, ne ferait qu'augmenter la pénalisation au sein de la SR.

Il est possible d'alléger cette pénalisation, et de contribuer moins à la surcharge du système de transmission, en adoptant une stratégie mixte. Cette dernière serait issue d'une combinaison des aspects positifs des stratégies A (en pannes transitoires) et D (en panne permanente). Cette amélioration est obtenue au prix d'une augmentation d'espace requis par les msg à conserver, car dans la stratégie mixte AD, chaque SPD conserverait à la fois les msg qu'elle reçoit et ceux reçus par la SR correspondante. La stratégie mixte AD fait l'objet de la section qui suit.

V.5. CONCLUSION.

La stratégie D est destinée à privilégier beaucoup plus la récupération de panne permanente au détriment des pannes transitoires. Elle adopte ainsi une attitude contraire à la stratégie A, mais traite les pannes transitoires d'une façon plus satisfaisante que ne le fait A vis-à-vis des pannes permanentes.

Elle permet de doter les systèmes de capacités de tolérance aux pannes et de ce fait, leur assure une grande disponibilité concourant à leur attribuer un fonctionnement non-stop. L'overhead encouru par son implémentation en mécanisme de reprise au niveau du système s'avère satisfaisant, particulièrement lorsque les critères de non postpropagation sont satisfaits.

On vient d'étudier la stratégie avec l'hypothèse de l'occurrence d'une panne permanente unique. Le problème de plusieurs pannes est indissociable de celui de la désignation de stations de reprise, et de la coordination des processus lors des interactions ou de la création des Rp. Ce problème constitue la deuxième partie de notre étude: "stratégies avec contrainte". Cependant on estime que pouvoir reprendre une station défaillante unique, constitue un résultat encourageant pour la conception des systèmes non-stop, en particulier dans les réseaux locaux où le nombre de stations n'est pas considérable.

La stratégie D étant particulièrement sensible à un accroîssement du traffic msg, aussi pour éviter des problèmes relatifs à une congestion du réseau, il serait important d'utiliser des mediums de transmission favorisant une communication offrant une large diffusion.
Les algorithmes de reprise sont décrits en annexe.

Chapitre VI

STRATEGIE AB

" Les msg d'information sont conservés par les producteurs et par la station potentiellement défaillante SPD."

VI.1. Motivation.

La stratégie AB est issue d'une combinaison des deux stratégies A et B. L'objectif fondamental de cette union, est de tirer meilleur parti des aspects positifs de chacune d'elles pour améliorer les perturbations éventuellement générées à l'occurrence de panne. En effet, la stratégie A, relativement aux pannes transitoires (taux d'apparition le plus important), apparait comme étant le plus avantageuse. Cependant, elle souffre d'une tendance à générer de longues chaines de reprise en panne permanente. Cette faiblesse est compensée par la stratégie B qui agit à l'encontre de A, où les pannes transitoires trouvent un traitement "moins" efficace, en l'occurrence:

- L'activation des producteurs concernés pour fournir les msg indispensables induit une certaine activité qui contribue à augmenter l'overhead au niveau de ces stations émettrices.

- Le renvoi de ces msg augmente le volume du traffic msg qui influe sur la charge du système de transmission, et peut provoquer des situations de congestion ou de dégradation de performance.

La stratégie AB qui permet de remédier à certaines faiblesses rencontrées en A et B, n'ait obtenue cette amélioration au niveau des perturbations qu'au détriment d'autres facteurs tels, l'espace mémoire nécessité par la conservation, et l'overhead résultant de la gestion de celle-ci.

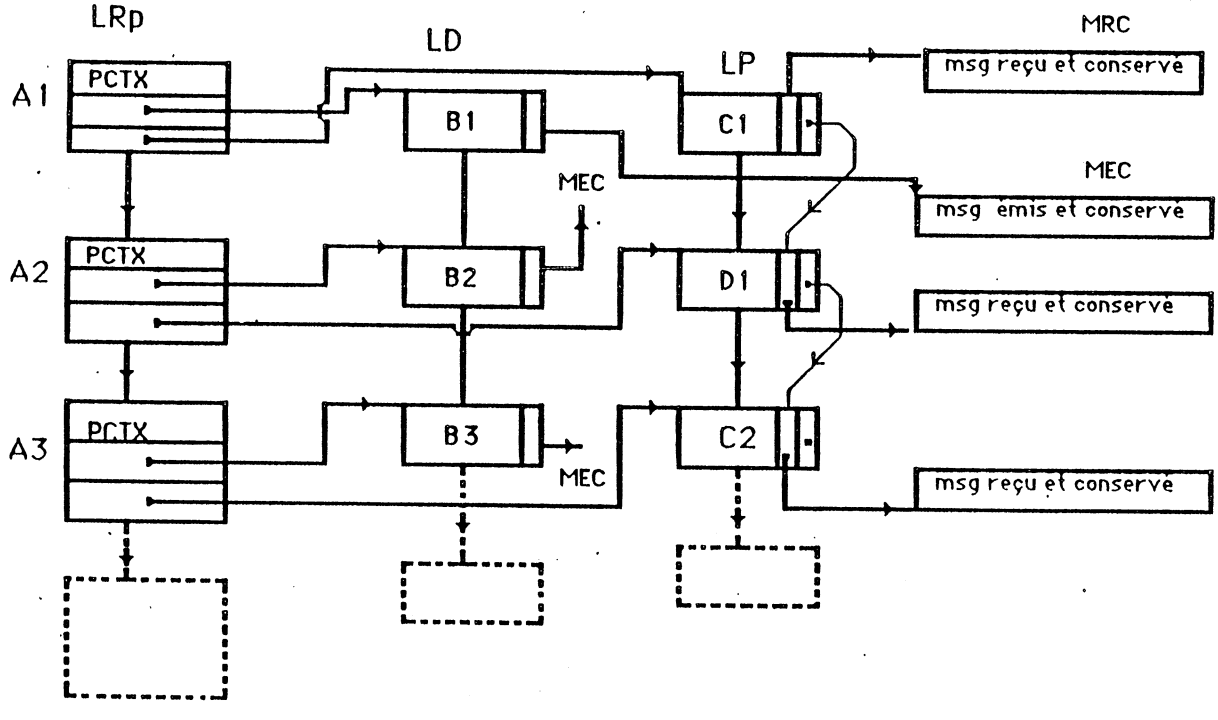
Ainsi, toute station est tenue de conserver les msg émis et les msg reçus, ce qui peut éventuellement représenter une pénalisation non négligeable lorsque le degré d'intéractivité des stations s'avère important.

VI.2. Informations nécessaires.

En fonction du type de panne, transitoire ou permanente, la stratégie AB se comporte respectivement en stratégie A ou en stratégie B. Ainsi, les informations nécessaires seront regroupées dans une structure de données qui répondra aux impératifs de chaque cas à savoir:

- Réutilisation locale des msg conservés après reprise d'une panne transitoire; on retrouve la structure adoptée en A, fig 3.
- Possibilité de pouvoir solliciter le renvoi des msg indispensables lors d'une reprise après panne permanente de la même façon qu'en stratégie B (fig 1.).

En fusionnant ces deux structures, on obtient une structure possible adaptable pour la stratégie AB. Pour le processus A de la fig 1 (stratégie A) cette structure peut être la suivante.



Structure de données possible adoptée
en stratégie AB.

Figure 01

Quant aux informations et à la structure de données nécessaires à la réalisation des critères (1,2) de non postpropagation, elles restent identiques à celles définies et décrites dans les stratégies précédentes. La structure des msg est également identique à celles des autres stratégies.

En définitive, tout ce qui a été dit en stratégie A, concernant les pannes transitoires, reste valable et applicable à cette stratégie. De même en ce qui concerne les pannes permanentes, l'attitude à prendre est celle de la stratégie B. Les hypothèses restent également identiques.

VI.3. Opération de reprise.

Pour éviter de détailler les différentes étapes de reprise qui caractérisent chaque type de pannes, déjà décrites en A et B, on procède donc à une étude qui s'inscrit dans une considération commune.

Soient Pr, le processeur de reprise, P le processus défaillant et RpR appartenant à P, un point de relance de P. Supposons que la panne est détectée, diagnostiquée, et la reconfiguration éventuelle réalisée. Il s'agit de traiter la panne selon sa catégorie, donc de distinguer d'abord son type puis d'activer les différentes actions caractérisant son traitement. Cette distinction n'est pas évidente puisqu'elle peut être imposée à Pr, c'est notamment le cas où une station devient subitement inopérante (panne du processeur...), ou bien laissée à l'appréciation du processeur qui décide que la panne (par récidive), doit basculer du type transitoire vers le type permanent.

L'élimination de la postpropagation joue un rôle important au niveau de la perturbation de l'environnement externe au processus. Il est donc d'un intérêt certain de considérer l'opération de reprise en fonction de la réalisation des critères (1,2) précédemment définis.

VI.3.1. Stratégie AB sans postpropagation.

1- S'il s'agit d'une panne transitoire, et comme aucune propagation n'est possible, le Rp actif définit alors un état cohérent du système et Pr accomplit sans difficulté l'opération de reprise en exécutant les actions suivantes (voire algorithmes en annexe):

- Restauration de l'état de P en RpR;

- Relance du processus P.

Après quoi, Pr assure le respect du critère 2 et le principe de réutilisation des msg conservés et ce, respectivement par l'intermédiaire des informations sauvegardées à cet effet (fig 6, A) et de la liste LIP de RpR.

2- S'il s'agit de panne permanente, Pr agit comme suit:

- Ne disposant pas d'informations sur les producteurs concernés, Pr émet un msg de renvoi (MDR) aléatoire dans lequel sont précisées les informations nécessaires à la détermination des msg à renvoyer;

- Restaure l'état de P en RpR;

- Relance l'exécution de P.

Relativement à Pr, l'opération de reprise est terminée. Tout processeur Pri différent de Pr, qui reçoit le msg MDR agit par:

- Détermination d'un Rpi appartenant à Pi exécuté par Pri, tel que Rpi --> RpR; Dans l'affirmative, Pri entame le renvoi des msg sollicités, tout en tenant compte d'une panne transitoire pouvant affecter Pri. Autrement dit, Pri doit garder une trace des msg réémis pour éviter des situations déplaisantes causées au niveau du processus defaillant par de multiples renvois de mêmes msg.

VI.3.2. Stratégie AB avec postpropagation.

La postpropagation systématique exigée par l'insatisfaction des critères (1,2) rend l'opération de reprise plus onéreuse que dans le cas précédent. Ainsi, lorsque la panne est considérée transitoire, Pr accomplit les étapes suivantes (algorithmes en annexe):

1- Emission d'un msg de reprise (invocation précise) à chaque dépendant direct du Rp actif choisi comme RpR. On adopte ici le principe de détermination de la RL par roll-backs progressifs, quoiqu'un roll-back direct est possible. Il est également possible d'appliquer une invocation aléatoire sauf à l'égard d'un processus antérieurement défaillant affecté par une panne permanente.

2- Renommage du RpR et émission d'un msg de mise à jour à tout propagateur direct de RpR qui ne lui est pas dépendant.

3- Destruction des successeurs de RpR et de sa liste LID. Restauration de l'état du processus et réexécution.

Bien entendu, ces actions sont exécutées par tout processus atteint par la postpropagation; et tout ce qui concerne l'émission et la réception des msg demeurent identique qu'en stratégie A.

L'émission de msg de mise à jour aux propagateurs directs de RpR, insensibles à l'opération de reprise, peut être éliminée par l'utilisation de l'invocation aléatoire dans laquelle le msg porte le RpR d'un processus repris et la nouvelle identification de ce RpR. Ainsi les processus dépendants prendront acte de la postpropagation, et les propagateurs procèderont uniquement à une mise à jour.

Lorsque la panne est considérée permanente, Pr agit comme en stratégie B par l'exécution des opérations suivantes:

1- Emission d'un msg de renvoi (MDR) aléatoire, avec RpR comme point de relance et RpR' la nouvelle identification de RpR.

2- Restauration de l'état du processus défaillant en RpR et réexécution de ce dernier. Ainsi se termine l'opération de reprise pour le processus défaillant.

Tout processeur Pri recevant un MDR doit définir sa position vis-à-vis de ce msg. Plusieurs attitudes sont possibles:

- Pri n'est nullement concerné.

- Un processus Pi appartenant à Pri est dépendant du processus repris, auquel cas Pri procède à la reprise de Pi comme décrit en stratégie B (III.2.1.)

- Pri exécute un processus Pi propagateur direct du processus repris (émeteur du MDR); il effectue alors la mise à jour adéquate, et exécute le renvoi des msg sollicités (voir III.2.1.1.).

- Il existe un couple (Rpi,Rpj) appartenant à Pi exécuté par Pri telque Rpi --> RpR et RpR --> Rpj, et Rpi dominant Rpj. Alors, il y a renvoi et mise à jour au niveau de Rpi et reprise à partir de Rpj (III.2.1.1.).

Remarque:

Outre le respect des hypothèses faites en stratégies A et B, on suppose bien entendu (H3) que le mécanisme de transmission est fiables, i.e. que le msg d'invocation aléatoire en diffusion est reçu par l'ensemble des processus du système.

VI.4. Perturbations et chaines de reprise.

Le désordre ou perturbations causé au système par l'apparition d'une panne peut être évalué en termes de facteurs de propagation à savoir: niveau n et ordre p de propagation (D14).

- Lorsque les critères (1,2) de non postpropagation sont satisfaits, tout processus est isolé de ses partenaires vis-à-vis des effets d'une éventuelle opération de reprise. Cet isolement nous donne des valeurs des facteurs de propagations satisfaisantes, on obtient alors: niveau n = 0 ordre p = 1 et ce, quelque soit le type de panne. A noter également qu'un Rp unique peut être sauvegardé par processus, resultat assez probant vis-à-vis de l'optimisation de l'espace mémoire requis par le mécanisme de reprise.

- Dans le cas où les critères (1,2) ne sont pas satisfaits, l'effet de la conservation des msg au niveau des perturbations peut être contrebalancé par la postpropagation systèmatique. Ainsi, il n'est pas possible de pouvoir ni évaluer les facteurs de propagations, ni leur donner des limites. On peut cependant estimer que l'ampleur des perturbations et la longueur des roll-backs demeurent raisonnables par rapport à un mécanisme où la rétropropagation est aussi systématique. Ces considérations aléatoires sont plutôt influencées par la nature du système considéré.

VI.5. CONCLUSION.

Compte tenu du fait que la stratégie AB est issue d'une combinaison sélective des stratégies A et B, elle hérite ainsi de leurs avantages respectifs, notamment à l'égard des effets de perturbations à l'occurrence de panne. Cependant, la gestion des msg et les structures de données qui les manipulent, nécessaires au mécanisme de reprise, accroîssent le facteur overhead en absence de panne. Il est difficile (voire pas possible) d'évaluer cet overhead sachant qu'il dépend étroitement de l'intéraction interprocessus.

L'espace mémoire nécessaire aux structures de données et qui dépend également des échanges entre processus est sujet à une augmentation conséquente, en particulier lors d'une postpropagation systématique. Autrement dit, l'application d'une telle stratégie (comme d'ailleurs toutes les autres) requière une attention particulière sur les paramètres qu'il convient de privilègier au détriment des autres.

Il n'existe aucune stratégie "universelle" capable de s'adapter à n'importe quel système. C'est selon les caractéristiques d'un système, et en fonction de la priorité accordée à certains paramètres (overhead en absence de panne, à l'occurrence de panne, espace mémoire, augmentation du traffic msg...) que le choix est déterminé.

Notons toutefois que la stratégie accorde un privilége aux perturbations, notamment lorsque les critères (1,2) sont satisfaits. Une comparaison des diverses stratégies (voir section du même nom) permet d'établir une sélection en fonction de certains paramètres privilégiés.
Pour les algorithmes de reprise se referer en annexe.

Chapitre VII

STRATEGIE AD

" Les msg reçus et conservés par la station potentiellement défaillante (SPD), sont captés et conservés par la station de reprise associée (SR)."

VII.1. Motivation.

Par duplication de la conservation des msg au niveau de la SPD et de la SR associée, on regroupe les avantages, de la stratégie A où les pannes transitoires trouvent un traitement satisfaisant, et de la stratégie D qui répond mieux à la récupération de pannes permanente. Ces deux caractéristiques font que dans la stratégie AD, il n'y a aucun transfert de msg indispensables lors d'une opération de reprise. Donc aucune pénalisation des processus émetteurs lors d'une éventuelle opération de rerpise, notamment lorsque les critères (1,2) sont satisfaits. Cette isolation de processus défaillant qui attribue aux propagations des valeurs optimales est obtenue au détriment d'autres facteurs intervenant dans l'efficacité de la stratégie.

En effet chaque SPD se voit contrainte de conserver beaucoup plus de msg; ce qui signifie, disposer d'un espace de conservation conséquent, et supporter l'overhead issu de la gestion de ces msg et de la mise en place des structures de données adéquates.

VII.2. Informations indispensables.

Puisque la stratégie mixte AD est un panache des deux stratégies A et D, on y retrouve donc:

- Le graphe de reprise tel qu'il est défini pour A (fig 3) et ce, afin d'assurer les impératifs de la propagation et de l'utilisation de msg conservés (notamment en pannes transitoires).

- Une structure de données inhérente au caractère de station de reprise, grâce à laquelle la réutilisation des msg à l'occurrence de panne permanente est possible (fig 2, D).

Les informations nécessaires pour satisfaires les critères (1,2) de non postpropagation restent identiques à celles définies dans les stratégies précédentes.

VII.3. Opération de reprise.

Pannes transitoires.

Que les critères (1,2) soient ou non satisfaits, l'opération de reprise est identique qu'en stratégie A; on applique donc le même algorithme.

Panne permanente.

On adopte également une attitude semblable à celle de la stratégie D; on applique ainsi un algorithme identique.

VII.3.1. Stratégie AD sans postpropagation.

Tout type de panne confondu, l'opération de reprise avec critères (1,2) satisfaits peut être conduite de la façon suivante. Une fois la panne détectée, diagnostiquée et reconfiguration éventuelle réalisée, le processeur de reprise Pri devant reprendre le processus défaillant Pi agit différemment selon que la panne est transitoire ou permanente.

a) S'il s'agit d'une panne transitoire, seul Pi est impliqué dans cette reprise, auquel cas Pri exécute les actions suivantes:

1- Restauration de l'état de Pi en Rpi;

2- Relance de Pi;
Etant entendu que Pri assure le critère 2 comme dans les autres stratégies.

b) S'il s'agit d'une panne permanente, l'opération de reprise ne concerne que Pi. Comme Pri ne peut assurer aucun des critères (manque d'informations), il doit informer les dépendants directs de Pi de prendre en charge la validation du critère 1, par l'émission d'un msg diffusé. Aprés quoi, les actions à exécuter sont identiques aux 1 et 2 ci-dessus, avec toutefois une utilisation de structure de données différente et propre à ce type de panne et à la stratégie (fig 2,D).

L'intégration dans un algorithme unique des deux types de panne est aisée, il importe de distinguer quel type doit-on traiter (voir algorithme en annexe).

VII.3.2. Stratégie AD avec postpropagation.

Lorsque les critères (1,2) ne sont pas satisfaits, trois types de processus sont concernés par la reprise.

- Le processus défaillant P,
- Les processus dépendants directs de P qui doivent reprendre et propager la postpropagation,
- Les processus propagateurs directs de P, auxquels il faudrait communiquer la nouvelle identification du RpR renommé. Cette identification peut être transmise dans le msg d'invocation aléatoire afin d'éviter l'émission de msg de mise à jour.

En intégrant les deux types de pannes dans un même algorithme, l'opération de reprise peut être décrite de la façon suivante:
Soient Pr le processeur de reprise, P le processus défaillant et Rp le point de relance.
Supposons que la panne est détectée et diagnostiquée et la reconfiguration éventuelle effectuée.
S'il s'agit d'une panne transitoire, Pr a le choix entre une invocation précise et une invocation aléatoire, sauf s'il y a eu déjà une panne permanente, auquel cas, le processus affecté nécessite une invocation précise. Cette invocation précise est imposée par le fait que le processus, ne disposant pas de la totalité de son graphe de reprise, se voit incapable de répondre à une invocation aléatoire.

S'il s'agit de panne permanente, Pr ne peut adopter que l'invocation aléatoire; d'où une utilisation conjointe des deux types d'invocation s'impose.

Donc à l'occurrence de panne transitoire, Pr ayant choisi un RpR pour P (soit Rp), lance l'opération de reprise par:

1- Emission d'un msg d'invocation précise à tout dépendant direct de Rp,

2- Renommage de Rp, et émission d'un msg de mise à jour pour tout propagateur direct de Rp qui n'est pas en même temps dépendant direct de Rp.

3- Destruction de la liste LID de Rp et de ses successeurs éventuels.

4- Restauration de l'état de P en Rp et relance de P.

Dans le cas d'une panne permanente, Pr émet un msg d'invocation aléatoire (diffusé), déclenche le mécanisme de réutilisation des msg conservés, restaure l'état de P en Rp et lance l'exécution (on adopte ici le principe du roll-backs progressifs).

Lorsqu'un processeur Prj reçoit un msg d'invocation, il détermine l'attitude à prendre vis-à-vis de l'opération de reprise, y est-il impliqué ? Si oui comment va-t-il réagir?
- Si le msg d'invocation spécifie le type d'invocation, Prj saurait exactement l'action à réaliser. Autrement dit, si le msg fait référence explicitement à une invocation précise, cela épargnerait à Prj une recherche inutile à la manière d'une invocation aléatoire.

Donc si l'invocation est précise, ce qui signifie, qu'il existe un processus Pj appartenant à Prj tel que il existe Rpj appartenant à Pj et Rp --> Rpj, alors Prj exécute les mêmes actions que 1,2,3,4 précédentes avec les éléments appropriés Pj et Rpj.
- Si l'invocation est aléatoire, à priori Prj ignore s'il est ou non impliqué, il effectue alors les recherches qui s'imposent à savoir: Si un tel Rpj existe, Pj doit être repris en ce point, et Prj exécute les actions 1,2,3,4 précédentes. Sinon, Prj détermine s'il existe Rpj appartenant à Pj tel que Rpj --> Rp. Dans

l'affirmative, Prj procède à une mise à jour de Rp par sa nouvelle identification fournie dans le msg. Cette mise à jour s'effectue dans la liste LID de Rpj. En définitive, la stratégie AD agit tantôt en stratégie A (pannes transitoires), tantôt en stratégie D (panne permanente).

VII.4. Perturbations et chaines de reprise.

a- Critères (1,2) satisfaits.

Quelque soit le type de panne, l'opération de reprise ne concerne que le processus défaillant. Cette isolation du processus fait que toute propagation est de niveau n = 0 et d'ordre p = 1 , atteignant ainsi des valeurs op.....ant relatif à l'optimisation de l'espace mémoire est: un seul Rp par processus est indispensable à sauvegarder pour le mécanisme de reprise. Si relativement à des perturbations, à la longueur des chaines de reprise et vis-à-vis du nombre de Rp à maintenir par processus, la stratégie AD s'avère être la plus performante, en revanche l'overhead généré par la gestion des msg à conserver est le plus important. cet overhead est naturellement fonction du degré d'intéraction entre processus.

b- Critères (1,2) non satisfaits.

Puisque la postpropagation s'impose de façon systématique et qu'elle est capable d'annuler l'effet de la conservation, il n'est pas possible de prévoir avec certitude, ni la longueur des chaines de reprise, ni l'ampleur des perturbations. Le rôle de la conservation des msg étant la limitation de la propagation, il est alors moins probable d'avoir, à l'occurrence de panne, des situations de dégénérescence provoquant une révocation d'une importante quantité d'activité.

VII.5. CONCLUSION.

L'étude faite sur les stratégies A et D a révélé qu'il est possible d'améliorer certains facteurs jouant un rôle important dans l'efficacité d'une opération de reprise, notamment:

- Améliorer le traitement des pannes permanentes de la stratégie A, et celui des pannes transitoires de la stratégie D.
- Accroître les capacités de reprise de la stratégie D (capacités éventuellement réduites aprés recouvrement de la panne permanente).

La stratégie AD, fusion des stratégies A et D, a pour but essentiel de réduire le plus possible, les perturbations issues des propagations interprocessus. Cette réduction est réalisée en limitant l'opération de reprise au processus défaillant et lui seul, en particulier lorsque les critères (1,2) sont satisfaits.

Si cette stratégie AD s'avère être la moins perturbatrice de l'environnement, cet avantage est obtenu au détriment d'autres facteurs par exemple:

- L'espace de conservation de msg qui peut réclamer des proportions importantes, mais ne constitue guére un handicap pour des stations ayant un espace secondaire conséquent.
- Le facteur overhead relatif à la gestion des msg se trouve également en augmentation sans pour autant poser de problème, sauf dans un cas exceptionnel de trés forte intéraction entre processus communicants.

Comme déjà mentionné précédemment, le choix d'une stratégie à implémenter dépend essentiellement de deux facteurs:

- Les caractéristiques et les exigences du système candidat à être doté de mécanisme de reprise.
- Les caractéristiques et les capacités de reprise de la stratégie donnée.

Puisqu'il n'y a pas de stratégie "universelle", un compromis entre plusieurs facteurs privilégiés est nécéssaire pour déterminer le choix de la stratégie à adopter.

Chapitre VIII

COMPARAISON DES DIFFERENTES STRATEGIES.

VIII.1. MOTIVATION.

Après l'étude des différentes stratégies, il est intéressant de pouvoir les comparer pour déterminer la stratégie qui répond le mieux aux caractéristiques d'une application donnée. Il est difficile (voire impossible) de prétendre à l'élaboration d'une stratégie "universelle" (applicable à tout type de systèmes), car soit que les exigences du système ne peuvent être respectées, ou que le coût encouru par l'implémentation d'une telle stratégie s'avèrerait inacceptable.

L'exhaustivité des stratégies précédemment présentées est dû au fait que chacune d'elle avantage un paramètre particulier (optimisation) pouvant guider le choix d'utilisation. Pour pouvoir faire cette comparaison, il est indispensable de définir des critères, en rapport avec les exigences matérielles et logicielles des systèmes pour lesquels une stratégie peut être adoptée.

Ces contraintes d'applicabilité peuvent être relatives aux:

- Stations physiques: Limitation de l'espace mémoire, de l'espace secondaire, capacités de transmission, facteur de disponibilité (lié à la probabilité de reprise).

- Applications: L'overhead toléré par l'application, suite à l'adjonction d'un mécanisme de reprise implémentant une stratégie, doit être conforme aux limites acceptables, tant en absence de pannes qu'en leurs occurrences.

Sous cette considération, bien que le nombre de critères de comparaison à définir puisse être important, on n'en retient basiquement que quatre, qui paraissent mieux satisfaire les objectifs à atteindre. Il est cependant difficile, en raison du comportement aléatoire des processus (au niveau des échanges et de la création des Rp), d'effectuer une comparaison formelle (stricte); c'est pourquoi, certaines affirmations n'ont qu'un caractère

"probabiliste".

VIII.2. Critères de comparaison.

Quatre critères de base sont donc retenus, il s'agit :

C1 (TP): Probabilité de provoquer une perturbation inter-processus à l'occurrence de panne, lors du lancement de l'opération de reprise. Si possible la propagation qui en est l'origine, sera quantifiée en terme de: niveau n et ordre p (D14) [ou Tendance à créer des Perturbations].

C2 (TAO): Overhead introduit sur une application, dû à la préparation de l'environnemt nécessaire à une hypothétique opération de reprise (overhead en absence de panne) [ou Tendance à une Augmentation de l'Overhead].

C3 (TAT): Sensibilité à un acroissement du volume du traffic msg dans le réseau (ou Tendance à l'Accroissement du Traffic-msg).

C4 (TEM): Taille de l'espace mémoire nécessaire aux structures de données, et taille de l'espace de conservation (ou Tendance à une demande d'Espace Mémoire).

Puisqu'il n'est pas possible de déterminer exactement l'augmentation de chaque facteur (espace, overhead, traffic msg) encouru en appliquant une stratégie donnée, on utilisera alors comme référenciel, la stratégie donnant une valeur minimale pour un facteur donné, pour établir une comparaison du même facteur vis-à-vis des autres stratégies.

Par exemple, le facteur overhead en absence de panne (TAO) se voit attribuer la valeur minimale en stratégie C, puisque aucune activité pénalisante n'est consacrée à la gestion des msg à conserver. Les autres stratégies seront ainsi comparées à C, relativement au facteur TAO (Tendance à l'Acroissement Overhead en absence de pannes).

VIII.3. Récapitulatif.

Avant de dresser un tableau récapitulatif établissant la comparaison des stratégies en fonction des critères définis, il est important de considérer la situation de l'effet-domino au regard des différentes stratégies.

VIII.3.1. Stratégies et effet-domino

VIII.3.1.1. Cas sans postpropagation

Stratégies	Rétropropagation	Postpropagation	Effet-domino
A	Systématique en panne permanente, concerne les processus propagateurs directs du défaillant.	éliminée	Peu probable avec des perturbations importantes possibles
B	Possible dans le cas exceptionnel de panne transitoire succédant une panne permanente (1)	éliminée	Pas possible
C	Systématique	éliminée	Probable et perturbations importantes possibles
D	Possible dans un cas exceptionnel (2)	éliminée	Peu probable semblable à A
AD	éliminée	éliminée	pas possible
AB	éliminée	éliminée	pas possible

- Critéres (1,2) satisfaits -

(1): La rétropropagation peut être au plus imposée vers une une station/processus (un producteur)

(2): La rétropropagation est possible lorsqu'une station de reprise est sollicitée pour un renvoi de msg alors qu'elle se trouve défaillante.

VIII.3.1.2. Cas avec postpropagation systématique

Stratégies	Rétro propagation	Post propagation	Effet-domino
A	Systématique en panne permanente	Systématique	Stratégie sensible $P_A \approx P_D < P_C$
B	cas exceptionnel (1)	Systématique	mons sensible $P_{AD} < P_B < P_A < P_C$
C	Systématique	Systématique	Stratégie la plus sensible P_C sert de référence.
D	cas exceptionnel (2)	Systématique	Sensible $P_D \approx P_A < P_C$
AB	éliminée	Systématique	La moins sensible $P_{AB} \approx P_{AD}$
AD	éliminée	Systématique	La moins sensible que AB P_{AD} le plus faible.

-Critères (1,2) non satisfaits-

L'effet-domino est susceptible d'être provoqué par la rétropropagation et/ ou la postpropagation. Si P_c est la probabilité qu'une opération de reprise dégénère en effet-domino en stratégie C, la situation des autres stratégies sera comparée à P_c (borne supérieure) car elle est la plus forte, et à P_{AD} (borne inférieure) car elle est la plus faible.

VIII.3.2. Comparaison

VIII.3.2.1. Pannes transitoires et permanentes sans postpropagation.

↓ Stratégies — Critères de comparaison Ci →

	C1	C2	C3	C4
A	p = x 0 ≤ n ≤ m	$TAO_C \leq TAO_B \leq TAO_A \leq TAO_D$	TAT_A faible	TEM_A : peut être important mais fonction des intéractions
B	p = 1 n = 0	$TAO_B \leq TAO_A$	TAT_B important à cause des renvois de msg	$TEM_B \leq TEM_A$
C	p = x n = x	TAO_C : minimale il sert de référence	TAT_C relativement important si roll-back progressif	TEM_C : minimal
D	p = 1 n = 0	$TAO_D < TAO_A$	$TAT_D \approx TAT_B$	$TEM_D \approx TEM_A$
AD	p = 1 n = 0	TAO_{AD} maximal	TAT_{AD} minimal (référentiel)	TEM_{AD} : maximal
AB	p = 1 n = 0	$TAO_{AB} \leq TAO_{AD}$	$TAT_{AB} \approx TAT_A$	$TEM_{AB} \approx TEM_{AD}$

- Pannes transitoires ou permanentes sans postpropagations (critéres (1,2) non satisfaits _

x : inquantifiable

On peut grossièrement écrire:

$$TAO_C < TAO_B \leq TAO_A \leq TAO_D < TAO_B \quad TAO_{AD}$$

$$TAT_{AD} < TAT_A \leq TAT_{AB} < TAT_C \quad TAT_D \quad TAT_D \qquad (I)$$

$$TEM_C < TEM_B < TEM_A \quad TEM_D < TEM_{AB} \leq TEM_{AD}$$

Pour le paramètre TP (C1), notons que les valeurs de propagation pour C sont inquantifiables, avec une probabilité importante de pouvoir créer des perturbations et de longues chaines de reprise.

Le niveau n de propagation en stratégie A est borné, l'ordre p est variable en fonction des interactions des propagateurs avec le processus défaillant. Cependant aucun cycle de propagation n'est possible, ce qui donnerait une faible probabilité d'avoir un ordre important par processus repris.

VIII.3.2.2. Pannes transitoires et permanentes avec postpropagation.

On trouve le même ordre que (I) concernant les paramètres TAO, TAT, TEM. Toutefois les facteurs de propagation (n,p) changent considérablement, il est alors impossible de quantifier les perturbations et ce, à cause du fait que la postpropagation peut éliminer l'effet de la conservation. De façon probabiliste on peut écrire:

$$P_{AB} \gtrsim P_{AD} < P_B \quad P_D \leq P_A < P_C. \qquad (II)$$

Critères / Stratégies	C1: Tendance à créer des Perturbations à l'occurrence de panne	C2: Tendence à l'Augmentation de l'Overhead généré en absence de panne	C3: Tendance à un Accroissement du Traffic msg	C4: (TEM) Tendance à une demande d' Espace Mémoire important
A	TP_A : peut être important mais $TP_A < TP_C$	TAO_A : acceptable mais fonction des intéractions $TAO_C < TAO_B \leq TAO_A \leq TAO_D$	TAT_A : faible	TEM_A : fonction des intéractions
B	$TP_B < TP_A$	$TAO_B \leq TAO_A$	TAT_B : le plus important des stratégies	$TEM_B \leq TEM_A$
C	TP_C : Le plus important car rétro et postpropagation	TAO_C : le plus faible possible	$TAT_C \leq TAT_B$	TEM_C : le moins important
D	$TP_D \approx TP_B$	$TAO_D > TAO_A$	$TAT_D \approx TAT_B$	$TEM_D \approx TEM_A$
AD	$TP_{AD} \approx TP_{AB}$	TAO_{AD} Le plus élevé possible	TAT_{AD} Le moins important	TEM_{AD} : Le plus élevé
AB	TP_{AB} est le moins important	$TAO_{AB} \approx TAO_{AD}$	$TAT_{AB} \approx TAT_{AD}$	$TEM_{AB} \approx TEM_{AD}$

- Pannes transitoires ou permanentes -
avec postpropagation systématique
(critères (1,2) non satisfaits)

VIII.4. CONCLUSION.

Comme il a été déjà souligné à plusieurs reprises, en raison de la liberté dont disposaient les processus communicants d'implanter indépendemment leurs régions de reprise, il n'est pas possible d'évaluer, dans le cas le plus général (critères (1,2) non satisfaits), les effets d'une perturbation globale à l'occurrence de panne. Néanmoins, l'élimination de la postpropagation associée à l'application de la conservation des msg, permet d'isoler totalement ou partiellement (stratégie A) tout processus vis-à-vis de ses partenaires, et borner ainsi les valeurs des propagations. Ce résultat est important dans la mesure où les processus évoluent sans crainte de contamination par les effets d'une panne.

Dans le cas où la postpropagation s'impose de façon systèmatique, la conservation qui assure l'évitement de la rétropropagation contribue grandement à atténuer les situations de pénalisations importantes, mais ne permet en aucune façon d'éliminer avec certitude des situations de grandes perturbations.

Le comportement des processus face aux pannes, dépend fondamentalement de leurs échanges mutuels. Comme ces interactions vis-à-vis du mécanisme de reprise relèvent d'évènements aléatoires, on ne peut prédire a priori avec exactitude l'évolution de la situation au niveau: de l'espace mémoire, de l'influence sur les transmissions, ou de l'overhead induit sur le système en service normal ou exceptionnel (à l'occurrence de panne).

Les pannes transitoires dont la fréquence d'apparition est considérable, constituent un élément important qui guide le choix sur l'adoption d'une stratégie. Cette attitude ne doit pas pour autant sacrifier le cas de pannes permanentes. C'est ainsi que la stratégie A qui s'adapte le mieux aux pannes transitoires, s'avère moins satisfaisante pour une panne permanente.

En définitive, le choix d'une stratégie est conditionné par les caractéristiques des systèmes à doter de capacités de tolérance aux pannes d'une part, et des aptitudes de la stratégie d'autre part. Un compromis entre les divers paramètres d'efficacité peut servir à déterminer la stratégie à adopter.

Par exemple, si la ressource mémoire parait critique et les interactions interprocessus sont trés importantes, les stratégies mixtes sont moins efficaces. De même, si on cherche à privilégier le traitement normal (moins d'overhead en absence de pannes) au détriment d'une perturbation éventuellement importante en traitement exceptionnel (à l'occurrence de panne), La stratégie C répond le mieux à cet objectif.

Les relations I, II précédentes peuvent être utilisées pour la recherche d'un compromis. Ce dernier doit tenir compte également des capacités de reprise des stratégies. Ainsi, après recouvrement de la panne permanente, les stratégies B et D en sont particulièrement concernées (voir III.4).

La limitation majeure, observée au cours de l'étude des stratégies dans cette première partie, concerne la reprise d'une panne permanente unique; le cas de plusieurs est traité dans la seconde partie avec introduction d'une contrainte sur la création des Rp. Notons toutefois que cette limitation ne constitue point un handicap important pour bien des systèmes.

Chapitre IX

ROLL-BACK DIRECT

Pour éviter le cas de détermination de ligne de reprise (RL) par roll-backs progressifs, il est possible de prendre une décision distribuée de sorte que l'opération de reprise puisse donner lieu à un seul roll-back au niveau de chaque processus impliqué. Cette décision nécessite impérativement la disponibilité de l'historique des interactions du processus avec ses partenaires. C'est pourquoi différentes attitudes seront adoptées en fonction du type de panne considéré. A noter que le principe du roll-back qu'on décrit est applicable à une détermination dynamique de RL globale.

IX.1. Pannes transitoires.

On supose, comme d'ailleurs tout au long de l'étude, que cette catégorie de pannes n'altère en aucune façon les informations contenues dans le graphe de reprise (dans le cas contraire, une redondance de ces informations serait certainement nécessaire). Une décision "concertée" nécessite une synchronisation de tous les processus impliqués dans l'opération de reprise.
En effet, lorsqu'une panne est diagnostiquée, le processeur de reprise par l'intermédiaire du graphe de reprise, émet un msg préliminaire à tous ses dépendants directs, puis se met en attente de réponse. Ce msg constitue une sorte d'enquête , qui dès réception peut donner lieu à deux attitudes possibles:

- Soit le processus récepteur constitue une "feuille" ou noeud terminal dans le graphe de communication.

- Soit il constitue un noeud intermédiaire , et il propage le msg à ses dépendants directs de la même manière que son propagateur direct.

Soient P l'ensemble des processus communicants, Pi $\in$ P un processus défaillant (affecté par une panne), et "-->" le symbole indiquant la relation d'interaction entre deux régions de reprise représentées par les Rp associés; (Rpi--> Rpj <==> Rpj est dépendant direct de Rpi ou réciproquement Rpi est propagateur

direct de Rpj).

On considère Pi ϵ P (défaillant) initiateur de reprise comme superviseur, déterminant la Ligne de Reprise Globale (RLG) à partir des RL intermédiaires (ou provisoires) fournies par chaque dépendant de Pi qui représente un noeud dans le graphe des communications. Après diagnostic, Pi émet un msg d'enquête "aléatoire" (avec Rpi comme RpR) à tout Pj ϵ P tel que il existe Rpj ϵ Pj et Rpi--> Rpj, puis se met en attente de réponse. Si Pj est noeud terminal (n'ayant pas de dépendants directs) c'est-à-dire: il n'existe pas Pk processus de P, telque Rpj --> Rpk , alors Pj communique à Pi une "ligne de reprise" temporaire les concernant, constituée par Rpi et Rpj. Si au contraire il existe Pk ϵ P tel que, il existe Rpk ϵ Pk et Rpj --> Rpk, i.e. Pj est un noeud intermédiaire, alors Pj agit de la même façon que Pi en se considérant temporairement initiateur de reprise pour récupérer les RL provisoires de ses dépendants. Ces RL sont alors transmises au propagateur direct de Pj.

De proche en proche ce cheminement donnerait lieu à un ensemble de RL intermédiaires qui convergeraient vers l'initiateur principal (noeud racine). Ce dernier, par synthèse, déterminerait une RL globale (RLG) représentant l'ensemble des processus impliqués.

La première étape étant terminée, l'initiateur principal de reprise effectue la seconde étape, par l'émission de msg d'invocation précise aux processus concernés.

Remarque:

La RLG est l'union des RL intermédiaires avec la propriété de dominance (D4) des Rp. C'est à dire, si Rpi et Rpj sont deux points de reprise de P , et si Rpi et Rpj appartiennent tous deux à RL, alors le plus dominant des deux représentera seul le processus P dans RLG.

Exemple:

Soit le schéma suivant:

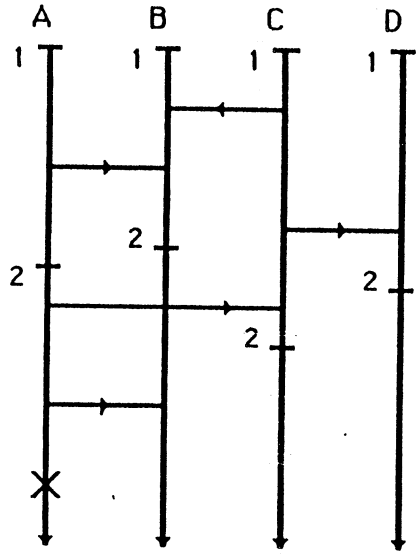
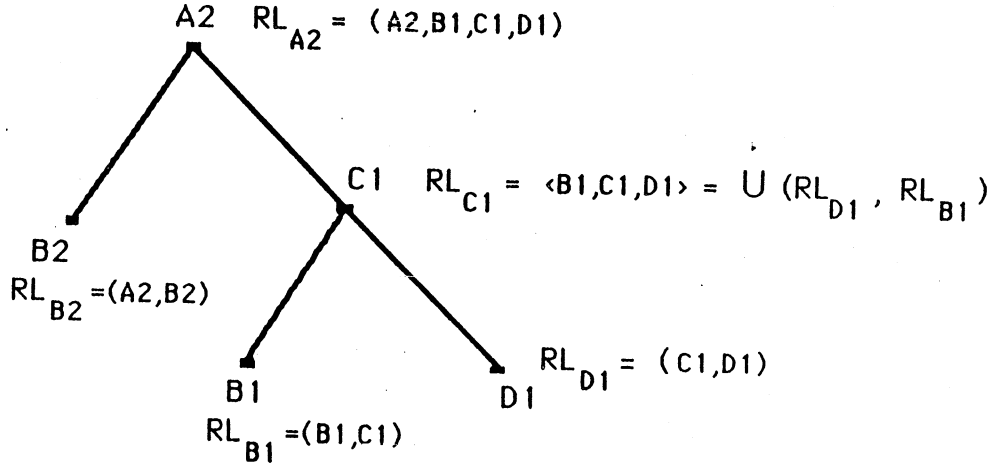


Figure 1a



Arbre de détermination de RL
(appliqué à la stratégie A).

Figure 1b

Soit A le processus défaillant, le processeur de A émet donc un msg d'enquête aléatoire, avec A2 comme RpR.
Selon les relations:

- A2 --> B2 au niveau de B, <A2,B2> constitue une ligne de reprise intermédiaire qui sera communiquée à A.

- A2 --> C1: grâce aux relations d'interaction C1 --> B1 et C1 --> D1, C1 constitue un noeud intermédiaire. Il joue le rôle d'initiateur secondaire de reprise et agit comme tel. C1 se met donc en attente des RL fournies par ses dépendants. Il obtient ainsi: <C1,D1> et <C1,B1> qu'il combine pour fournir à son propagateur direct la RL intermédiaire <C1,B1,D1>. A2 aura donc deux RL (intermédiaires): <A2,B2> et <C1,B1,D1> dont l'union donnerait la RLG <A2,B1,C1,D1> (fig 1b).

La seconde étape consiste , par l'intermédiaire de la RL globale déterminée, en une invocation précise de tous les processus concernés. L'exemple précédent (fig 1b), s'applique à la stratégie A car les RL intermédiaires ont été déterminées selon le principe de la stratégie (pas de rétropropagation).

Donc de façon générale, le principe du roll-back direct qui applique de façon ineffective celui du roll-back progressif, dépend du principe de la stratégie auquelle il s'applique. Il peut donc être adopté en présence de n'importe quel type de propagation (postpropagation et/ou rétropropagation).

Remarque:

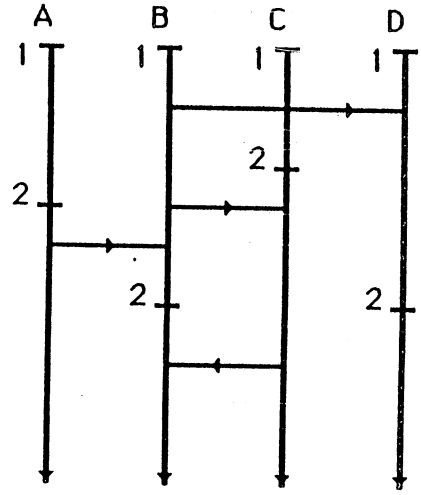
La définition classique de RL (D9) n'est en faite valable que pour la stratégie C, on l'adopte en fonction du type de propagation imposée. Pour éviter le problème de cycle lors du processus de détermination des RL intermédiaires, et distinguer plusieurs initiateurs éventuels de reprise, il est nécessaire d'adopter un moyen d'identification de ces initiateurs (par exemple session de détermination).

IX.2. Panne permanente.

Compte tenu des limitations des stratégies dans cette première partie, à savoir traitement d'une occurrence unique de panne permanente, le principe de détermination de la RL globale demeure applicable de la même manière qu'en pannes transitoires.

Le cas de multiple pannes constitue un problème complexe en rapport direct, avec le manque de contraintes au niveau des processus, et au degré de disponibilité souhaité dans un réseau. Toutefois, l'introduction d'une contrainte d'établir conjointement les Rp (voir partie II), évite systématiquement aux processus des roll-backs progressifs.

Continuer à adopter le principe de roll-bak direct tel qu'il vient d'être décrit (après recouvrement de la panne permanente), peut susciter un problème d'aptitude similaire à celui des capacités de reprise en stratégies B et D (voir B.III.4.). Ce problème tant évoqué, est dû au manque d'informations au niveau du processus défaillant pour pouvoir donner suite à un msg d'enquête. La situation peut être illustrée par l'exemple suivant.



Problème d'inefficacité après recouvrement
d'une panne permanente.

Figure 02

Soit P = < A,B,C,D>, un ensemble de processus communicants. Soit une panne transitoire affectant le processus A. Pour la stratégie A avec postpropagation, par exemple, A émet un msg d'enquête à son dépendant direct unique B (avec B2 comme RpR provisoire) puis se met en attente. B étant un noeud intermédiaire agit comme son propagateur direct A, par émission d'un msg d'enquête à ses dépendants directs C et D, puis attend les RL

intermédiaires. D est noeud terminal, il répond par le renvoi de la RL intermédiaire (RLi) <B1,D1>, puis attend. C est un noeud intermédiaire, il agit comme tel et récupère la RLi <C2,B2> (car B est "terminal") qu'il communique à B. B reçoit ainsi deux RLi : <B1,D1> et <B1,C2,B2> qu'il combine pour déduire une RLi: <A2,B1,C2,D1>. Cette dernière est finalement communiquée à l'initiateur A2 auquel est dévolue la tâche de provoquer une invocation précise en B1, C2, et D1.

Supposons qu'au lieu que ce soit une panne transitoire qui affecte le processus A, c'est une panne permanente qui survient sur B. Compte tenu du principe de la stratégie A, B est repris en B2, et C en C2 (rétropropagation). La RL pour cette opération de reprise est <B2,C2>. L'évolution des processus reprend après un certain délai au terme duquel une situation semblable à celle de la fig 2 est rétablie. Une panne transitoire qui survient sur A, poserait le problème d'application du principe de roll-back direct défini.

En effet bien que A puisse émettre un msg d'enquête à B, celui-ci occupant la position d'un noeud intermédiaire ne peut agir à la manière de A , par émission d'un msg d'enquête à ses dépendants directs C et D (par manque de la liste LD, il les ignore). On retrouve alors la solution adoptée en panne permanente pour provoquer la reprise des processus concernés (invocation aléatoire). On utilise ainsi un msg d'enquête aléatoire qui répond bien aux exigences du principe, sous la condition indispensable de la reprise d'une panne permanente unique. Donc deux types de msg d'enquête sont nécessaires selon les situations: enquête précise enquête aléatoire.

IX.3. Comparaison des deux principes.

IX.3.1. Roll-backs progressifs:

Avantages:

- Respect de l'hypothèse H2.

- Lorsqu'un processus Pi est relancé, il considère que l'opération de reprise est terminée pour lui, bien qu'un retour éventuel de propagation peut lui imposer un autre roll-back. Ceci prend en compte beaucoup plus facilement un recouvrement de plusieurs opérations de reprise (plusieurs pannes transitoires).
- Simplicité de réalisation des opérations de reprise.

Inconvenient:

Beaucoup plus de perturbations lors de l'opération de reprise.

IX.3.2. Roll-backs directs.

Avantages:

Evitement de relances inutiles de processus, d'où un coût de l'opération de reprise moindre, et les perturbations entre processus sont restreintes.

Inconvenients:

- Le nombre de msg indispensables pour l'opération de reprise devient plus important.
- Des situations d'interblocage causées par des attentes mutuelles de RLi sont possibles, donc prévoir un ménanisme de rupture d'attente circulaire.
- L'hypothèse H2 est moins respectée, d'où la probabilité de panne pendant la reprise est plus élevée.

DEUXIEME PARTIE.

Chapitre I

INTRODUCTION.

I.1. Motivations.

Nous avons vu dans la première partie "stratégies sans contrainte" que l'absence de contrainte à imposer aux processus, procure des avantages certains, mais limite en contrepartie les capacités de reprise particulièrement en pannes permanentes (tolérance d'une panne unique). Bien que cette restriction ne constitue aucunement un handicap pour certains systèmes, il est cependant des cas où des applications exigent une plus grande disponibilité matérielle qui leur assure une exécution continue jusqu'à terminaison normale.

Cet objectif de traitement non-stop implique une aptitude au recouvrement d'un nombre important de pannes permanentes. Ce nombre ne doit être limité que par le seuil de dégradation du système relatif à l'overhead issu de la charge éventuelle des stations de reprises suite à une redistribution des processus des stations défaillantes. Pour répondre à cet objectif de façon satisfaisante, nous avons été amené à imposer une contrainte aux processus interactifs, de sorte que la création des Rp doit se faire de manière coordonnée.

Ainsi, lorsqu'un processus Pi décide d'établir un point de reprise Rpi, cette décision est communiquée à l'ensemble des processus Pj ayant eu des échanges directs ou indirects avec Pi depuis la création de Rpi-1, qui donneront suite à ce msg, par la création de leurs Rpj respectifs. Cette coordination lors de l'établissement des Rp, confère au processus décideur un rôle de superviseur de l'action et à ses partenaires un rôle de subordonnés.

Bien que la contrainte imposée prive les processus communicants de choisir délibérement l'instant de création de leurs Rp (cette privation n'est pas absolue), elle n'a aucune restriction sur les capacités de communication.

L'étude des diverses stratégies assujetties à cette contrainte et constituant cette seconde partie, revêt un interêt particulier en l'occurrence:

- Optimisation du nombre de Rp à sauvegarder par processus.
- Capacités de reprise accrue.
- Simplicité des algorithmes de reprise.

Une comparaison de ces stratégies est également établie avec plus de précision, aidant ainsi à déterminer celle qui convient le mieux au système donné.

I.2. REPRISE EN PRESENCE DE MULTIPLES PANNES PERMANENTES.

I.2.1. Nécessité d'une politique de désignation de stations de reprise.

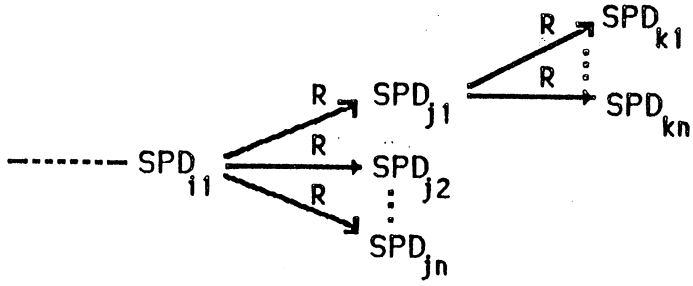
Bien que la contrainte de reprise d'une seule occurrence de panne permanente imposée par les stratégies précédemment étudiées, ait principalement pour origine, le manque de l'historique de communication des processus défaillants sur les stations de reprise, une politique adéquate de désignation des stations de reprise est nécessaire et d'autant plus justifiée dès lors on considère l'apparition de plusieurs pannes.

Cette politique doit tenir compte du taux de disponibilité souhaité dans le réseau considéré. Autrement dit, quel serait le nombre maximal de pannes tolérables par le système. De façon générale, deux approches sont possibles:
- Désignation statique,
- Désignation dynamique.

I.2.1.1. Désignation statique.

Le traitement des pannes permanentes implique nécessairement une redondance des informations à utiliser lors de la reprise. La désignation statique consiste à définir pour chaque station potentiellement défaillante SPDi plusieurs stations de reprise SRj et ce, avant le lancement de l'application. Au cours de l'évolution des processus de l'application, chaque SPDi transmet à ses partenaires SRj les informations nécessaires au mécanisme de reprise (Rp, code...). Ainsi, on observe une redondance de code, Rp..., au niveau de SPDi et les SRj associées. Le nombre de SRj désignées pour chaque SPDi est fonction du nombre de pannes que l'on se fixe de reprendre, et du coût généré par la redondance créée au sein des SRj.

A chaque occurrence de panne, cette redondance diminue, c'est-à-dire, si on a:



Relation de désignation de stations de reprise.

Figure 0

Autrement dit, chaque station SPDii possède n stations de reprise SRji. Lorsque SPDji, qui est à la fois station potentiellement défaillante et station de reprise, tombe en panne, elle est reprise par SPDki. Il existe alors une station SPDii qui n'a que n-1 stations de reprise. Par exemple, si pour chaque station SPDi d'un réseau R, il ne lui est désigné au début de l'application qu'une et une seule station de reprise (n=1), l'occurrence de la première panne laisserait une station de R sans station de reprise. Donc il est impossible de pouvoir traiter une seconde panne.

De façon générale, en considérant que les n stations d'un réseau soient dotées de la même probabilité de panne, reprendre m pannes permanentes nécessite que toute station SPDi ait m stations de reprise. Ce qui implique que les informations de reprise (code, Rp....) soient répliquées dans m stations. La valeur de ce facteur m joue un rôle primordial dans le coût induit par le mécanisme de reprise élaboré (coût relatif à la gestion des msg). Ce coût pénalise l'application de façon inévitable, même en absence de panne. C'est pourquoi l'approche dynamique de désignation, privilégie le déroulement du système en absence de panne, en évitant un traffic intense de msg et les traitements qui en résultent.

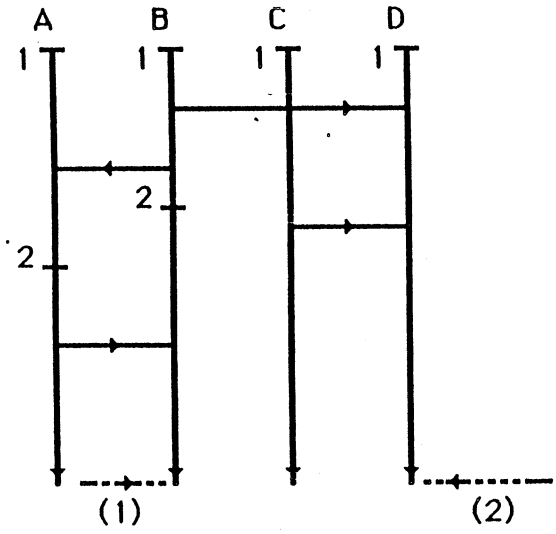
I.2.1.2. Désignation dynamique.

Elle consiste à attribuer à m une valeur faible (m=1, ou m=2 si on tient compte de deux pannes simultanées), et de reconstituer cette valeur à chaque occurrence de panne, lors de la

reconfiguration. Ainsi, on observera un transfert d'informations (code, Rp...) de l'ancienne station de reprise vers la station nouvellement désignée. Ce transfert aura lieu lors de la phase de reconfiguration (avant relance).

I.2.2. Stratégie A et multiple pannes permanentes.

La structure de données adoptée lors de la première partie de l'étude, à savoir maintenir sur la station de reprise seulement le code et les (ou le) Rp de processus, ne permettait pas de reprendre plusieurs pannes. Cette insuffisance est dûe à l'absence de l'historique des échanges des processus, absence qui empêche toute réponse à un msg de reprise. La figure ci-dessous illustre bien les incapacités de la structure de données précédente.



Insuffisance de la reprise
en partie I de l'étude.
Figure 01

Selon le mécanisme des roll-backs progressifs, une panne affectant B le ferait reprendre par le processeur de reprise A en B2, qui déclencherait la rétropropagation vers A2. Ainsi est atteint un état cohérent. Une seconde panne survenue sur D le ferait reprendre par le processeur de C en D1; par rétropropagation C reprend en C1. Par manque de l'historique d'intéraction de B sur A et de celui de D sur C, il n'est pas possible de connaître l'interaction

B1-->C1, ce qui invalide le traitement de la seconde panne.

Pour remédier à ce problème, il faut pouvoir disposer à chaque occurrence de panne, de l'historique des communications (graphe de reprise) des processus défaillants. L'idée de conserver une copie de cet historique, sur la station de reprise, pose un problème coûteux, celui du maintien de l'intégrité et la cohérence de ces copies.

Cependant pour éviter d'avoir des incertitudes relatives à l'interaction entre Rp antérieurs au Rp de relance (par exemple, après la relance de B en B2, B1 demeure indispensable lors de la défaillance de D sans qu'on le sache), il est possible d'intervenir sur les décisions des processus pour l'établissement des Rp, en imposant qu'à toute RRE courante (D19), la RRR associée est aussi courante.

Cette coordination lors de la création des Rp [Bar 83], consiste à imposer aux processus communicants d'établir leurs Rp au même instant. C'est-à-dire, dès qu'une décision est prise par le processus Pi d'établir un nouveau Rpi, celle-ci est propagée à tous les partenaires Pj ayant échangé des informations avec Pi depuis Rpi-1. Une propriété fondamentale résulte de cette approche:
- Chaque Rp actif appartient à une ligne de reprise.

Conséquences:

1- Les roll-backs sont de longueur minimale.

2- Toute opération de reprise est exempte d'effet-domino.

3- A tout instant de l'évolution d'un processus, seul le Rp actif est indispensable; donc un contexte unique est sauvegardé par processus.

Même en violant l'hypothèse de latence nulle (H1) émise au début de l'étude, autrement dit invalider les critères (1,2) de non postpropagation, mais considérer une latence relativement faible de sorte que les contextes associés aux Rp actifs soient exemptes d'erreurs, la longueur des roll-backs demeure constante. Cette longueur est fixée à celle de la région de reprise courante, soit une propagation toujours d'ordre 1.

La suite est consacrée à l'étude des diverses stratégies soumises à la contrainte imposée aux processus lors de la création de leurs Rp.

Chapitre II

STRATEGIE A.

"La station potentiellement défaillante (SPD) conserve les msg reçus."

Une structure de données identique à celle utilisée dans la partie I est aussi adoptée dans cette deuxième partie. Cependant, compte tenu du fait qu'un seul contexte par processus est maintenu sauvegardé, et que l'historique des interactions de chaque processus ne concerne que la région de reprise courante, le volume des informations à sauvegarder se trouve considérablement réduit. Cette réduction concerne particulièrement l'espace de conservation, et la taille du graphe de reprise.

II.1. Latence de détection nulle.

Conséquences:

- Les critères (1,2) de non postpropagation peuvent être satisfaits.
- La conséquence 3 de la propriété fondamentale est largement justifiée.

Hypothèse:

Les contextes sauvegardés, associés aux Rp sont présumés exempts d'erreurs.

II.1.1. Pannes transitoires.

Soient Pi le processus affecté par la panne, Rpi appartenant à Pi est actif et point de relance de Pi. Après détection et diagnostic de la panne, le processeur Pri exécutant Pi effectue les opérations suivantes:

1- Restauration de l'état de Pi associé à Rpi;

2- Réexécution de Pi.

Remarque:

Puisqu'aucun type de propagation n'est possible, les opérations de réception de msg se traduisent d'abord par une réutilisation des msg conservés, tandisque les opérations d'émission sont soumises au préalable à un contrôle nécessité par la satisfaction du critère 2 prédéfini.

Perturbations.

Puisque l'opération de reprise ne concerne que le processus défaillant et lui seul, il n' y a pas de perturbation à proprement dit. La propagation n'existe donc pas, et les valeurs de ses facteurs sont: niveau n = 0 et ordre p = 1 .

II.1.2. Pannes permanentes.

La phase de détection et diagnostic de la panne étant supposée effectuée, la reconfiguration appropriée est également présumée réalisée. En d'autres termes, la station défaillante est logiquement déconnectée, et toute station fonctionnelle du réseau identifie la nouvelle reconfiguration. Le processeur de reprise Pri, reprenant le processus défaillant Pi, exécute les actions suivantes:

1- Emission d'un msg de reprise proclamant une invocation aléatoire (Dll) à toutes les stations (y compris Pri elle-même). Ce msg doit comporter le Rp de relance (Rpi) du processus à reprendre (Pi).

2- Restauration de l'état du processus défaillant Pi correspondant au Rp de relance Rpi (Rpi actif).

3- Lancement de la réexécution.

Lorsqu'un processeur Prj reçoit un msg de reprise du type précédent, il agit comme suit:

1- Prj détermine sa situation vis-à-vis de l'opération de reprise qui vient d'être lancée. Autrement dit, Prj est-il impliqué dans une rétropropagation? C'est à dire: existe-t-il Rpj appartenant

à Pj processus de Prj tel que Rpj --> Rpi. Si tel est le cas alors 2;

2- Prj restaure l'état de Pj en Rpj et relance Pj.
Cette réexécution est impérative puisqu'elle doit fournir les msg indispensables à Pi et à lui seul (critère 2 satisfait).

Remarque 1:

Le cas de plusieurs pannes simultanées n'introduit aucun problème, car bien qu'on ignore au niveau des stations de reprise respectives, les interactions ayant eu lieu au cours des exécutions normales des processus défaillants, ces derniers vont se créer mutuellement les msg indispensables.

Remarque 2:

On constate jusqu'à présent que l'opération de reprise est à l'initiative du processeur de reprise. Cependant, il est possible, après diagnostic et reconfiguration, que la décision d'effectuer un roll-back soit prise de façon asynchrone au niveau de chaque processeur exécutant un processus propagateur du défaillant. Ainsi lorsqu'un processeur Pri eût connaissance de la défaillance d'un processeur Prj, Pri détermine immédiatement si un de ses processus est concerné par une éventuelle rétropropagation. Dans l'affirmative, il effectue la reprise nécessaire sans attendre l'invocation.

II.2. Latence de détection non nulle.

Conséquence:

Les critères (1,2) de non postpropagation ne peuvent être satisfaits, pour supprimer toute "contamination" la postpropagation s'impose systématiquement.

On suppose dans ce cas que la latence est non nulle mais suffisament faible pour que les contextes sauvegardés soient sans erreurs.

II.2.1. Pannes transitoires.

Soient Pi le processus défaillant, Rpi ∈ Pi le point de reprise actif, et Pri le processeur exécutant Pi. Une fois la panne est détectée et diagnostiquée, Pri accomplit les opérations suivantes (algorithmes cf. annexe):

1- Emission d'un msg de reprise du type invocation précise ou aléatoire;

2- Restauration de l'état du processus Pi au point Rpi;

3- Relance de Pi.

Tout processus Prj ayant reçu un msg du type précédent agit de la manière suivante:
S'il s'agit d'une invocation aléatoire alors, Prj détermine si un des processus qu'il exécute Pj est dépendant direct de Pi. Dans l'affirmative, ou bien si l'invocation est précise alors, Prj agit exactement comme Pri en exécutant les opérations 1,2,3 ci-dessus avec les éléments appropriés à savoir Pj et Rpj.

Des retours de postpropagation vers l'initiateur de l'opération de reprise, sont possibles, et pour éviter un "bouclage" de cette opération, il importe d'identifier les sessions de reprise, laquelle identification sera transmise dans les msg d'invocation.

II.2.2. Pannes permanentes.

Détection et diagnostic de la panne étant supposés accomplis, la reconfiguration appropriée étant réalisée, le processeur de reprise Pri procède à l'opération de reprise de la manière suivante (voire algoritmes en annexe);

1- Emission d'un msg d'invocation aléatoire;

2- restauration de l'état du processus défaillant Pi à son unique Rp de relance Rpi, sauvegardé dans la station Pri;

3- relance de Pi.

Tout processeur Prj qui reçoit un msg de reprise du type aléatoire y donne suite de la façon suivante:

1- Détermination d'une éventuelle interaction d'un processus Pj qu'il exécute avec le défaillant Pi; c'est à dire existe-t-il Rpj appartenant à Pj et Pj appartient à Prj telque Rpj --> Rpi ou Rpi --> Rpj. Dans l'affirmative alors 2;

2- Pour tout Rpk appartenant à LID (Rpj) émettre msg de reprise;

3- Restauration de l'état de Pj correspondant à Rpj;

4- Relance de Pj.

Remarque:

- L'emploi de l'invocation aléatoire est impératif pour le processeur de reprise, mais elle est moins performante que l'invocation précise; c'est pourquoi une utilisation des deux est plus efficace.

- Tout msg d'invocation doit identifier la session de reprise à laquelle il appartient et ce, pour éviter que des processus soient repris plusieurs fois inutilement, pour le compte d'une même opération de reprise (éviter des cycles de propagations inutiles).

II.3. Perturbations.

II.3.1. Cas de latence nulle.

Cette hypothèse de latence nulle (Hl) nous permet d'adopter les critères de non postpropagation et de renforcer l'hypothèse des contextes exempts d'erreurs. Soient Pi le processus défaillant, et m ($m \geq 0$) le nombre de processus Pj, propagateurs directs de Pi depuis la création de la dernière ligne de reprise (RL).

Proposition 1.

Quelque soit la panne affectant Pi, la propagation générée au cours de l'opération de reprise est d'ordre 1 (p=1) et de niveau au plus égal à m ($n \leq m$).

Preuve:

Sachant que chaque Rp de tout processus communicant appartienne à la RL la plus récente, et qu'un seul Rp est maintenu sauvegardé par processus alors, tout roll-back ne concerne que la région de reprise courante, donc de longueur égale à 1 (p=1). La latence d'erreur étant nulle, les critères (1,2) sont satisfaits, tout propagateur direct de Pi stoppe la rétropropagation. Comme la postpropagation est éliminée, le nombre de processus impliqués dans la reprise est exactement égal à 1 en panne transitoire (processus défaillant), et au plus égal à m+1 en panne permanente; d'où le niveau de propagation est $n \leq m$.

II.3.2. Cas de latence non nulle.

Vue la complexité du problème de la latence, on considère cette dernière non nulle mais suffisamment faible de sorte que les contextes sauvegardés soient corrects.

Proposition 2:

Si m ($m \geq 0$) est le nombre de processus Pj impliqués directement ou indirectement dans des échanges avec le défaillant Pi et ce, après création de la RL, la propagation générée lors de la reprise de Pi est d'ordre 1 (p=1), et de niveau n au plus égal à m ($n \leq m$).

Preuve:

Compte tenu du fait que toutes les interactions entre processus se font uniquement entre régions de reprise courantes, tout roll-back éventuel se fait obligatoirement vers le Rp actif, d'où l'ordre de propagation p=1. Ce résultat est une conséquence de la contrainte imposée que l'on peut formalisée par: Quelque soit Pj appartenant à P (ensemble de processus interactifs), Il existe un et un seul Rpj de Pj tel que Rpj appartient à RL et Rpj est actif.

La rétropropagation étant impérative lors des pannes permanentes, la postpropagation est également systématique. Le cas le plus défavorable peut s'observer lorsque tous les Pj sont impliqués dans la propagation, c'est à dire un niveau égal à m (n=m). On peut

donc conclure que $n \leq m$ quelque soit la panne.

Remarque:

Bien que n=m dans le cas défavorable, ce résultat demeure important sachant qu'il y ait seulement révocation des activités des m régions de reprise courantes, ce qui n'est pas le cas en stratégies sans contrainte (partie I).

II.4. CONCLUSION.

L'introduction de la contrainte imposée aux processus communicant depuis la dernière ligne de reprise, d'établir simultanément leurs Rp, génére certes un certain overhead, mais procure un avantage important en exemptant toute opération de reprise de dégénérer en effet-domino. Puisqu'on ne conserve qu'un seul contexte par processus en cas de latence faible, l'espace mémoire nécessité par les Rp et les msg conservés se trouve largement réduit.

Notons qu'aucune limitation sur le nombre de pannes à reprendre n'est imposée par la méthode, sauf si ce n'est par la charge d'activité à faire subir aux stations de reprise.
Les algorithmes de reprise sont décrits en annexe (partie II).

Chapitre III

STRATEGIE B

" Les msg produits et émis sont conservés par leurs producteurs."

Ainsi tous les msg d'informations émis par une station sont conservés par cette dernière. Ce qui signifie qu'une forte interaction d'un processus avec son environnement nécessiterait un espace de conservation plus important et un temps de gestion des msg conséquent. L'overhead à faire subir à l'application dans pareils cas, se trouve considérablement réduit lorsqu'on applique la contrainte de création simultannée des Rp de processus, puisque seuls les msg émis dans la RRE sont conservés. On adoptera la même structure de données que dans l'étude sans contrainte précédente.

la sémantique de la stratégie consiste à fournir aux processeurs de reprise (quelque soit la panne) les msg indispensables à la relance des processus à reprendre. Cette action de réémission de msg conservés, fait suite, soit à une demande des processeurs concernés, soit à l'initiative des producteurs après détection des processus défaillants dont ils sont propagateurs.

Bien entendu à chaque reprise, les processus concernés vont reconstruire leurs graphes de reprise.

III.1. Latence de détection nulle.

III.1.1. Pannes transitoires.

Il n' y a pas de postpropagation car les critères (1,2) sont satisfaits.

Soient Pi le processus défaillant, Rpi son point de reprise actif. Quelque soit Pj tel que il existe Rpj θ Pj et Rpj --> Rpi, le processeur Pri exécutant Pi demande à tout Prj exécutant un Pj de lui fournir les msg émis par Pj à Pi et reçus dans la région de reprise associée à Rpi. Bien entendu, une synchronisation entre les émetteurs Pj et le récepteur Pi est nécessaire pour éviter le problème

de perte de msg (synchronisation entre n producteurs et un consommateur).

De l'aspect perturbation de l'environnement de Pi, il en ressort:

- Tout Prj, tel que Pj --> Pi, suspend momentanément le processus en cours d'exécution pour réémettre les msg indispensables.

- Il n' y a ni postpropagation, ni rétropropagation.

Puisque les critères (1,2) sont assurés, aucun problème ne se pose quant à une éventuelle panne durant les réémissions de msg. Les actions entreprises par Pri après détection et diagnostic de la panne affectant Pi sont:

1- Emission d'un msg de renvoi à tout Prj tel que: Pj --> Pi (i.e. à tout élément de LIP (Rpi)).

2- Restauration de Pi en Rpi.

3- Relance de Pi à partir de Rpi.

III.1.2. Pannes permanentes.

Soient Pi le processus défaillant, Pri son processeur de reprise. Le diagnostic et la reconfiguration étant supposés réalisés, Pri procède à l'émission de msg de renvoi des msg indispensables de la même manière qu'une invocation aléatoire, sauf qu'il n'y a pas réellement de rétropropagation. C'est à dire, quelque soit Rpj ϵ Pj tel que Rpj --> Rpi, Pri émet un msg de renvoi diffusé à Prj (processeur de Pj) en précisant Rpi. Le renvoi peut être systématique lorsque chaque Prj, étant informé de la panne, détecte l'interaction Pj --> Pi, sans attendre donc le msg de renvoi émis par Pri.

Les critères (1,2) assurent la non postpropagation, tandisque le renvoi élimine la rétropropagation. De la même façon que l'invocation, on distingue ici un renvoi précis en pannes transitoires et un renvoi aléatoire en pannes permanentes. Les actions effectuées par Pri après reconfiguration sont:

1- Emission d'un msg de renvoi aléatoire contenant le Rp de relance (soit Rpi);

2- Restauration de l'état de Pi en Rpi;

3- Relance de Pi.

Après réception d'un msg du type précédent, tout processeur Prj exécute les étapes suivantes:

1- Détermine s'il est concerné par le renvoi demandé, i.e. existe-t-il Rpj ∈ Pj tel que Rpj --> Rpi ?

2- Si un tel Rpj est trouvé (Rpj --> Rpi) alors Prj exécute la procédure de renvoi des msg indispensables associés à la région de reprise de réception correspondant à Rpi.

III.2. Latence de détection non nulle.

Hypothèse:

Les contextes associés aux Rp sont corrects.

III.2.1. Pannes transitoires.

La postpropagation étant systématique, chaque processus qui exécute un renvoi, doit prendre en compte qu'il est susceptible d'être soumis à une postpropagation, donc éviter de réémettre plusieurs fois les mêmes msg. Tout processus peut être assujetti à un renvoi, à une postpropagation, ou bien aux deux à la fois. Si Pi est le processus défaillant, Pri son processeur de reprise, et Rpi appartenant à Pi le point de reprise actif, alors après diagnostic Pri agit comme suit:

1- Quelque soit Rpj ∈ Pj tel que Rpj --> Rpi et non (Rpi --> Rpj), Pri émet un msg de renvoi à Prj (renvoi précis ou aléatoire);

2- Quelque soit Rpj ∈ Pj tel que Rpi --> Rpj, Pri émet un msg de reprise (invocation précise de préférence);

3- Pri restaure l'état de Pi en Rpi, puis relance Pi.

Tout Prj qui reçoit un msg de renvoi, exécute la réémission nécessaire. Tout Prj recevant un msg de reprise, exécute les étapes 1,2,3 précédentes. A noter qu'un msg de reprise prime un msg de renvoi (le premier annule le second).

III.2.2. Pannes permanentes.

Il est possible que les msg indispensables soient réémis au processeur de reprise Pri de façon asynchrone vis-à-vis de la relance de Pi. Autrement dit, après diagnostic et reconfiguration, tout processeur Prj dont un de ses processus Pj ait eu des interactions avec Pi, se verrait soit exécuter la procédure de renvoi de msg nécessaires à Pi (car Pj --> Pi), soit reprendre Pj
(car Pi--> Pj).
Le second cas fait de Pri l'initiateur de l'opération de reprise, dont un msg de renvoi aléatoire serait adressé à l'ensemble des partenaires Pj . D'où les actions suivantes effectuées par Pri.

1- Emission d'un msg de renvoi diffusé précisant Rpi;

2- Restauration de Pi en Rpi;

3- Relance de Pi à partir de Rpi.

Tout processeur Prj exécutant Pj et recevant un msg de renvoi réagit comme suit:

1- S'il existe Rpj ∈ Pj tel que Rpj --> Rpi et non (Rpi --> Rpj) alors Prj est soumis à la réémission des msg indispensables à Pi;

2- S'il existe Rpj ∈ Pj tel que Rpi --> Rpj, ou Rpi --> Rpj et Rpj-->Rpi alors:

3- Quelque soit Rpk appartenant à LID (Rpj), Prj provoque la post-propagation de Pk;

4- Restauration de Pj en Rpj, puis Relance de Pj.

III.3. Stratégie B et perturbations.

III.3.1. Cas de latence nulle.

Hypothèse:

On suppose qu'à un instant t, une panne et une seule est possible.

Soient Pi le processus défaillant à un instant t, et Pri son processeur de reprise. Soient m ($m \geq 0$) le nombre de processus Pj propagateurs directs de Pi à l'instant t, et P l'ensemble des processus interactifs depuis la RL.

Proposition 1:

Quelque soit la panne affectant Pi, la propagation dans P est de niveau n=0 (D14) et d'ordre p = 1; ces valeurs sont les plus optimales possibles.

Preuve:

1- Puisqu'à un instant donné t, il n'existe qu'une seule RL, celle contenant les RP actifs des processus de P (conséquense de la contrainte), et d'après l'hypothèse, tout processus défaillant Pi est relancé à partir du Rp actif, d'où l'ordre de propagation égal à 1.

2- Supposons que le niveau n de propagation est: n>0, alors il existe q processus à reprendre et q>1. Deux cas possibles sont à envisager:

1- Il existe au moins deux occurrences de pannes au même instant t, d'où contradiction de l'hypothèse;

2- Il y a propagation, contradiction avec le fait que la postpropagation est annulée par les critères (1,2) et la rétropropagation par le renvoi de msg conservés.
Ainsi le niveau $n \leq 0$, et comme par définition (D14) $n \geq 0$ alors: n=0.

L'hypothèse précédente n'a de signification que pour prendre en

compte la politique de désignation de stations de reprise. Elle permet dans le cas présent, d'éviter que deux pannes permanentes puissent survenir simultanément au même instant t, l'une affectant la station potentiellement défaillante, l'autre sa station de reprise. Dans pareil cas, il est manifestement impossible de pouvoir procéder à une reprise (faiblesse de la désignation dynamique).

Remarque 1:

En considérant l'éventualité de N occurrences simultanées de pannes (permanentes et/ou transitoires), toute opération de reprise lancée à un instant t, ne peut affecter qu'au plus N processus. Cette affirmation est justifiée par l'élimination de tout type de propagation.

Remarque 2:

Bien que les perturbations aient pour origine fondamentale la propagation de reprise interprocessus, il existe d'autres paramètres ayant une influence sur ces perturbations que l'on doit considérer (tels la fréquence d'exécution des procédures de renvoi de msg indispensables, le volume du traffic msg ...).

III.3.2. Cas de latence non nulle.

Proposition 2:

Si m est le nombre de processus Pj de l'ensemble P, interactifs depuis l'établissement de la RL commune jusqu'à l'instant t (t instant d'occurrence d'une panne affectant Pi de P), La propagation résultant de la reprise de Pi est d'ordre p=1 pour tout Pj et de niveau au plus égal à m-1.

Preuve:

D'après la construction de la ligne de reprise (RL), la propagation est d'ordre p=1. Puisqu'il ne peut y avoir de cycle de propagation pour contraindre la relance au-delà du Rp actif, le cas le plus défavorable s'observe lorsqu'il existe un cycle d'interactions

entre les régions de reprise actives de l'ensemble des processus communicants. La postpropagation engendrée ainsi par Pi atteint les m-1 processus Pj, d'où le niveau n de propagation est tel que:

$0 \leq n \leq m-1$.

III.4. CONCLUSION.

Nul doute qu'une amélioration importante au niveau performance est apportée à la stratégie par la contrainte introduite (relativement à l'étude sans contrainte). Elle s'observe dans:

1- Latence nulle:

- Propagation optimale;
- Espace de conservation considérablement réduit;
- volume du traffic msg réduit;
- nombre de pannes à reprendre limité seulement par la charge imposée aux stations de reprise.

2- Latence non nulle:

- Propagation de reprise contrôlée et bornée;
- Espace de conservation (msg , historique des processus, Rp) identique qu'en 1;
- Volumme du traffic msg réduit;
- Capacités de reprise identique qu'en 1.

Les procédures de reprise sont décrites en annexe (partie II).

Chapitre IV

STRATEGIE C

" Aucun msg n'est conservé."

Comme déjà mentionné antérieurement, la stratégie C offre à toute opération de reprise la plus grande couverture d'élimination d'erreurs, puisque dans le cas général (latence non nulle) les deux types de propagation s'imposent. Ainsi à l'occurrence de panne, l'ensemble des processus communicant depuis l'établissement de leur ligne de reprise commune, vont devoir être repris.

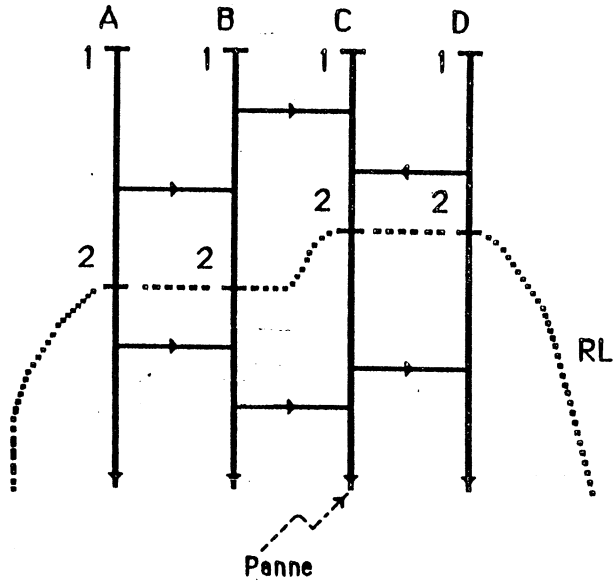
Du point de vue perturbation, la contrainte imposée permet de la réduire considérablement, puisque la propagation est bornée et l'effet domino est éliminé.

IV.1. Latence de détection nulle (H1).

Conséquence: Critères (1,2) de non postpropagation satisfaits.

IV.1.1. Pannes transitoires.

Soient Pi appartenant à P (ensemble de processus interactifs depuis la RL) le processus en "panne", Pri le processeur de reprise, et Rpi ϵ Pi le Rp actif. Quelque soit Pj ϵ P tel que il existe Rpj ϵ Pj et Rpj --/-> Rpi, Pj doit être repris par Prj en Rpj. Autrement dit, tous les propagateurs directs de Pi vont reprendre. La situation peut être la suivante.



Exemple de reprise
sans postpropagation.
Figure 01

Supposons C en panne, il sera relancé en C2 or:

A2 --/-> C2
B1 --/-> C2
} ==> propagation de reprise de C2 vers A2 et B2,

D2 n'est nullement affecté, car pas de postpropagation.

Pratiquement, lorsque la panne est diagnostiquée, Pri exécute les actions suivantes:

a- Pour tout propagateur direct Rpj de Rpi, émettre un msg d'invocation précise à Prj avec le Rp de relance précisé: Rpj.

b- Restaurer l'état de Pi en Rpi;

c- Réexécuter Pi.

Tout processeur Prk recevant un msg d'invocation précise, effectue les actions a,b,c précédentes (avec k devient j et j devient i).
Pour éviter de faire la distinction entre Pri (processeur de reprise du processus défaillant Pi) et les autres Prj, et pour

avoir un algorithme unique, on peut considérer que tout Rpi est son propre propagateur. C'est à dire Rpi appartiendrait à LID(Rpi), ainsi Pri et Prj exécuteront les mêmes actions, et Pri s'enverrait en premier un msg d'invocation précise de la même façon que pour les Prj.

IV.1.2. Pannes permanentes.

Après diagnostic et reconfiguration, quelque soit Pj ∈ P telque il existe Rpj ∈ Pj et Rpj --/-> Rpi, Pj doit être repris. La différence avec les pannes transitoires est que Pri ne peut émettre un msg d'invocation précise, il utilise obligatoirement une invocation aléatoire. Ainsi lorsque Prj reçoit un msg d'invocation, il n'exécute la relance que s'il existe Rpj ∈ Pj tel que Rpj --> Rpi. Donc Pri émet un msg d'invocation aléatoire à tous les Prj, après quoi, il relance Pi en Rpi (après restauration de l'état associé à Rpi).

Tout Prj qui reçoit un tel msg y donne suite par:

a- Détermine s'il existe Rpj tel que Rpj --> Rpi;

b- Si oui, soit il émet un msg d'invocation aléatoire proclamant la reprise en Rpj, donc agir comme Pri, soit procéder à une invocation précise, par émission d'un msg à tout Rpk tel que Rpk --> Rpj;

c- Restaure l'état de Pj en Rpj;

d- Relance Pj.

IV.2. Latence non nulle.

les contextes associés aux Rp sont supposés exempts d'erreurs. Les critères(1,2) n'étant pas satisfaits, d'où la conséquence suivante.

Conséquence:

La rétropropagation, et la postpropagation sont obligatoires. C'est à dire, si Pi ∈ P est exécuté par Pri avec Rpi le Rp actif, une panne affectant Pi conduirait à réaliser la proposition suivante:

(1) ∀ Pj ∈ P, tel que il existe Rpj ∈ Pj et Rpj--/->Rpi
ou (exclusif) Rpi --/-> Rpj, Pj doit être repris.

Cette proposition est valable tant en panne transitoire qu'en panne permanente. Selon la fig 1, toute panne affectant un quelconque des processus (A,B,C,D) provoquerait impérativement leurs relances à partir de la RL. La différence entre les deux types de pannes réside essentiellement au niveau du type d'invocation employée. C'est pourquoi on donne l'algorithme de reprise commun (cf. annexe partie II).

IV.2.1. Reprise commune.

Après diagnostic et reconfiguration, deux cas sont possibles.

1- L'initiative de l'opération de reprise est prise par le processeur Pri dont les Prj ne s'y engagent qu'après réception des msg de reprise du premier;

2- L'opération de reprise est exécutée simultanément au niveau de Pri et des Prj. Ces derniers ayant pris connaissance de la panne, entament les actions nécessaires, sans attendre les msg d'invocation de Pri. Autrement dit, vérifier si les Prj satisfont la proposition (1).

Remarque:

Selon le cas 1, tous les stations/processus exécutent un algorithme identique, mais l'opération de reprise est légèrement ralentie par les délais de transmission des msg d'invocation. Ce ralentissement concerne notamment les propagateurs et dépendants directs de Pi, puisque les autres processus concernés n'agiront qu'à la suite de la propagation engendrée par les premiers.
Le cas 2 offre une "simultanéité" dans l'opération de reprise, relativement aux propagateurs et dépendants directs du défaillant. Toutefois l'algorithme exécuté par Pi et celui des stations/processsus ne sont pas rigoureusement les mêmes.

IV.3. Perturbations.

IV.3.1. Cas de latence nulle.

L'élimination de la postpropagation permet certes, de réduire notablement la probabilité pour que l'ensemble P de processus communicant à l'instant t d'occurrence de la panne ne soit entièrement impliqué dans la reprise, mais ne permet d'évaluer les facteurs de la propagation. On peut dire cependant que la propagation est telle que: ordre p = 1 et niveau n: $0 \leq n \leq m$ avec une probabilité p', où m= card(P).

IV.3.2. Cas de latence non nulle.

Si p' est la probabilité pour que le niveau n de propagation au cas 1 soit égal à m, alors dans le cas 2, Prob(n=m) = 2p' et ce, en considérant: Prob(n=m) = p' pour la postpropagation et

Prob(n=m) = p' pour la rétropropagation.

Donc l'ordre de propagation est p=1, et le niveau n est tel que:

$0 \leq n \leq m$ avec une probabilité 2p'.

IV.4. CONCLUSION.

Dans la partie I, la stratégie C se révèle comme étant la plus "vulnérable" à l'effet domino tant en latence nulle qu'en latence non nulle (respectivement critères (1,2) satisfaits, non satisfaits).
L'introduction de la contrainte dans cette partie II, élimine cet effet domino d'avoir un niveau de propagation relativement important, autrement dit étendre l'opération de reprise à un grand nombre de processus partenaires du défaillant, demeure encore la plus grande des stratégies bien que cette probabilité est en diminution notable dans cette seconde partie.

En plus de l'effet domino d'autres améliorations ont été apportées notamment:

- Augmentation des capacités de reprise, où le nombre de pannes tolérées n'est limité que par la charge des stations de reprise.

- Pour tout processus, l'ordre de propagation p est toujours égal à 1.

- Diminution sensible du traffic msg.

- Optimisation de l'espace mémoire relatif au graphe de reprise et aux contextes associés aux Rp.

- Mise en oeuvre moins compliquée de la stratégie.

L'inconvenient majeur réside dans l'augmentation d'overhead qui résulte de la création coordonnée des Rp.

Chapitre V

STRATEGIE D.

" Les msg à destination de la station potentiellement défaillante (SPD) sont captés et conservés par la station de reprise correspondante (SR)."

Hypothèse:

Tout msg d'information reçu par une SPDi, est également reçu par la SRj associée.

Cette hypothèse est particulièrement réalisable dans des réseaux à médium de transmission favorisant la diffusion (exemple réseau ethernet...). La stratégie D, par une demande de renvoi de msg adressée à une seule station, lors de panne transitoire, et ne nécessitant aucun renvoi lors de panne permanente, apporte une amélioration certaine vis-à-vis des stratégies A,B,C. Cette amélioration réside dans:

- La perturbation causée par le renvoi de msg (relativement à B). Une seule station est impliquée dans ce renvoi.
- La perturbation causée par la rétropropagation inévitable en A (panne permanente) et en C (quelque soit la panne). Une comparaison entre les différentes stratégies sera faite dans la suite.

L'introduction de la contrainte "création simultanée des Rp" apportera, comme pour les autres stratégies, une réduction notable du coût généré lors d'une opération de reprise.

V.1. Latence de détection nulle (H1).

Conséquence:

Critères (1,2) satisfaits (pas de postpropagation).

V.1.1. Pannes transitoires.

Soient Pi ∈ P le processus de panne, Rpi ∈ Pi son Rp actif, Pri le processeur exécutant Pi, et Prj le processeur de station de reprise. Après diagnostic, Pri effectue les actions suivantes:

1- Emission d'un msg de renvoi à Prj;

2- Restauration de l'état de Pi en Rpi;

3- Réexécution de Pi.

On reconnait bien le rôle que va jouer Prj en tant que producteur unique et Pri consommateur unique, alors que dans la stratégie B, on a un schéma de n producteurs et un seul consommateur.

V.1.2. Pannes permanentes.

Prj possède une copie de Rpi, une copie du code de Pi et l'ensemble des msg reçus par Pi depuis Rpi jusqu'à l'instant de panne. Moyennant ces informations, Prj réexécute Pi d'où: Après diagnostic et reconfiguration appropriée, Prj exécute les étapes suivantes:

1- Restauration de l'état de Pi en Rpi;

2- Réexécution de Pi.

Remarque:

L'exécution de Pi est soumise aux règles du système local, c'est-à-dire à l'ordonnancement des processus qui y sont exécutés.

Une caractéristique importante de la stratégie, est que l'opération de reprise elle même, ne concerne que deux stations: La station défaillante et la station de reprise associée. Ceci représente un facteur important dans l'optimisation des perturbations de l'environnement de processus.

V.2. Latence non nulle.

La latence n'étant pas nulle mais suffisamment faible pour que les Rp soient sans erreurs.

Conséquence:

Les critères (1,2) ne sont pas satisfaits d'où postpropagation systématique.

V.2.1. Pannes transitoires.

Le processeur Pri agit comme suit:

1- ∀ Pj ∈ P tel que, il existe Rpj ∈ Pj et Rpi --> Rpj, Pri provoque la reprise de Pj.

2- Emission d'un msg de renvoi à la station de reprise.

3- Restauration de l'état de Pi en Rpi.

4- Relance de Pi.

Tout processeur Prj qui reçoit un msg d'invocation de reprise (invocation précise ou aléatoire) doit agir de la même façon que Pri.

V.2.2. Pannes permanentes.

Le diagnostic et la reconfiguration étant supposés connus, le processeur de reprise Prj (relais de Pri) exécute les actions suivantes (voire algorithmes en annexe, partie II):

1- Emission d'un msg d'invocation aléatoire pour satisfaire la proposition suivante:
∀ Pk ∈ P tel que il existe Rpk ∈ Pk et Rpi --> Rpk.

2- Restauration de l'état de Pi en Rpi.

3- Relance de Pi.

V.3. Perturbations.

a- Cas de latence nulle.

Deux stations sont impliquées lors de la reprise, la station défaillante et la station de reprise associée. Que ce soit en panne transitoire ou en panne permanente, seul le processus affecté est repris, ainsi les facteurs de propagation sont évalués à: niveau n = 0 , ordre p =1. Ces valeurs sont identiques à celles de la stratégie B, sauf que les perturbations sont moindres, puisque seule la station de reprise est concernée par le renvoi de msg (pannes transitoires seulement).

b- Cas de latence non nulle.

Soit P l'ensemble des processus interactifs depuis la dernière RL, et soit card(P)=m.
La rétropropagation étant éliminée, soit par renvoi des msg indispensables (panne transitoire), soit par disponibilité de ces derniers dans la station de reprise (panne permanente), la postpropagation systématique ne permet guère de quantifier exactement la propagation. Toutefois les valeurs extrémales des facteurs sont:
$0 \leq$ niveau n $\leq m$, et ordre p=1.

Remarque:

Sachant que les pannes transitoires sont les plus fréquentes(~ 90% des cas [Sie82]), il est évident de pouvoir privilégier leur traitement. Ainsi, une amélioration est donnée dans les stratégies mixtes suivantes.

V.4. CONCLUSION.

Relativement à une latence nulle, l'étude de la stratégie D dans les parties I et II, montre que ses caractéristiques (perturbations, espace mémoire, overhead, capacités de reprise...) demeurent inchangées. La contrainte introduite en partie II ne fait que pénaliser d'avantage le système, de ce fait, il serait plus avantageux d'opter pour la stratégie selon la partie I.

Au contraire, dans le cas de latence non nulle, l'avantage procuré par la contrainte (partie II) est important et s'observe notamment dans:

- L'optimisation de l'espace de conservation (sauvegarde d'un contexte unique par processus, nombre de msg conservés réduit à ceux de la région de reprise courante), et réduction de la taille du graphe de reprise.

- Perturbations bornées, propagations sans effet-domino.

- Réduction du traffic msg (plus de msg de mise à jour des listes LD et LP, ni de renvois systématiques des identificateurs de régions de reprise de réception).

- Roll-backs de longueur minimale (p=1) pour tout processus repris.

Toutefois, si la contrainte permet d'apporter certaines améliorations importantes, elle apporte également un inconvenient, celui de l'overhead généré par la création coordonnée des Rp. Cet overhead reste tout de même dépendant de la fréquence de création des Rp.

Chapitre VI

STRATEGIES MIXTES.

VI.1. Stratégie AD.

"Les msg reçus et conservés par la station potentiellement défaillante(SPD) sont captés et conservés par la station de reprise associée (SR)."

Cette stratégie combine les aspects positifs du traitement des pannes transitoires de la stratégie A, et du traitement des pannes permanentes en stratégie D. On en déduit les valeurs des facteurs de propagation suivantes:

a- En latence nulle.

niveau de propagation: n=0, et ordre p=1. Ce sont les mêmes valeurs qu'en stratégie D, sauf qu'on élimine les renvois de msg indispensables, donc une contribution moindre à l'augmentation de:
- L'overhead en station de reprise.
- La charge du système de transmission.

b- En latence non nulle.

Les valeurs des facteurs de propagation deviennent les suivantes:
$p = 1$ et $0 \leq n \leq m$ avec $m = card(P)$; (P: ensemble de processus communicants).

Cependant, la probabilité pour que n tende vers m demeure inférieure à celle des autres stratégies, puisque la rétropropagation est éliminée dans les deux types de pannes.

VI.2. Stratégie AB.

" Les msg reçus et conservés par la station potentiellement défaillante(SPD) sont conservés par les producteurs."

Cette stratégie est issue de la combinaison, de la stratégie A donnant un meilleur traitement des pannes transitoires, et de la

stratégie B s'adaptant relativement bien aux pannes permanentes.

Conséquences:

a- Latence nulle (postpropagation évitée).

Les facteurs de propagation sont: p = 1, n = 0.

La perturbation est plus importante qu'en AD, car tous les producteurs du processus défaillant sont activés d'où:

- Overhead au niveau des stations émettrices de msg indispensables.
- Augmentation de la charge du système de transmission.

b- Latence non nulle (postpropagation systématique).

Bien que la rétropropagation n'existe pas, la postpropagation systématique suffit à elle seule de créer des incertitudes sur la quantification précise de la propagation. On ne peut ainsi que borner cette dernière:

Ordre p = 1, niveau n, tel que $0 \leq n \leq m$; où m = card(P): nombre de processus interactifs depuis la création de la dernière RL.

Chapitre VII

CARACTERISATION ET COMPARAISON DES STRATEGIES

VII.1. Caractéristiques.

Parmi les diverses stratégies étudiées, certaines d'entre elles conviennent mieux à un type d'application donné que d'autres. C'est pourquoi, les caractériser en fonction de la nature du système destiné à être doté d'un mécanisme de reprise, ne ferait qu'améliorer leur efficacité.

VII.1.1. Stratégie A.

La conservation des msg est certes très importante pour freiner la perturbation, mais faut-il encore disposer de l'espace nécessaire pour le faire. Plus le nombre de msg reçus est important, plus est important le temps de gestion de ces derniers (overhead). Aussi, il est évident de considérer l'efficacité de la stratégie dépendant du degré d'interaction des stations du réseau. Par exemple, un système de n producteurs et un consommateur s'accomoderait moins bien à une telle stratégie, vis-à-vis du consommateur.

VII.1.2. Stratégie B.

A l'inverse de la stratégie A qui concentre les msg reçus au sein d'une seule station, la stratégie B a tendance à les répartir au niveau de leurs émetteurs. Ceci a pour conséquence d'exiger moins d'espace de conservation par station potentiellement défaillante, et les activitées de gestion des msg se trouvent également distribuées.
Ainsi, dans l'exemple précedent, chaque producteur conserverait les msg qu'il émet de la région de reprise courante, ce qui allégerait la contrainte espace de conservation au niveau du consommateur.
En revanche , la charge du système de transmission subirait une augmentation relative, ce qui requière un médium rapide de transmission.

VII.1.3. Stratégie AB.

La stratégie est issue de la combinaison de la stratégie A relativement aux pannes transitoires, et de la stratégie B vis-à-vis des pannes permanentes. Ainsi, chaque station potentiellement défaillante (SPD) se charge de la conservation des msg qu'elle reçoit d'une part, et des msg qu'elle émet d'autre part. Cette attitude exige des ressources de conservation plus importantes et une gestion conséquente des msg.

Les capacités de reprise sont importantes puisqu'aucune contrainte relative au nombre de pannes à traiter n'est imposée, et de plus la stratégie s'adapte aussi bien pour une politique statique de désignation , qu'une politique dynamique.
Bien que le nombre de msg échangés au cours de la vie d'un processus peut être important, celui des msg conservés ne concerne que l'interaction dans la région de reprise courante. Ce dernier peut être modulé par l'intervalle de temps séparant la création des Rp successifs.

VII.1.4. Stratégie C.

C'est la stratégie qui convient le mieux aux systèmes dont la ressource mémoire est critique, mais dont l'overhead à accepter peut éventuellemnt atteindre des limites importantes. Le traitement de multiple pannes permanentes y est largement toléré et les politiques des désignation de stations de reprise (politique statique ou dynamique) peuvent y convenir.

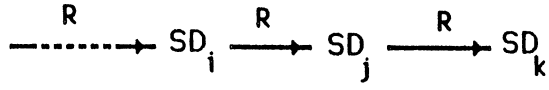
VII.1.5. Stratégie D.

Cette stratégie qui se veut une amélioration du traitement des pannes permanentes (relativement à la stratégie A), adopte un producteur unique pour le renvoi des msg en pannes transitoires. Autrement dit, elle constitue la solution "centralisée" de la stratégie B.

Ainsi, l'espace de conservation des msg et la gestion qui en résulte paraissent identiques qu'en stratégie A. Toutefois, dans

une perspective d'un traitement commun relatif aux deux catégories de pannes, les perturbations globales résultant d'une éventuelle opération de reprise, s'avèrent plus satisfaisantes qu'en A ou B.

L'importante fréquence d'occurrence des pannes transitoires et l'influence de leur traitement sur la charge du système de transmission permettent d'envisager des améliorations possibles (stratégie AD). Les capacités de reprise de la stratégie dépendent étroitement d'une politique statique de désignation comme le montre l'exemple suivant:
Soient trois stations SPDi, SPDj, et SPDk telles que:



(R: relation de désignation de stations de reprise.)

Soit une panne permanente affectant SPDj, alors puisque les msg reçus par SPDj sont conservés en SPDk, celle-ci prend le relais sans problème. SPDi n'ayant plus de station de reprise, ne peut faire face, non seulement à une éventuelle panne permanente, mais à toute autre panne transitoire puisque les msg conservés sont perdus. Seule une utilisation de la stratégie C après la première panne permanente, ou l'application d'une redondance de station de reprise, peuvent résoudre ce problème.

Le premier cas revient à une utilisation alternée, selon le besoin, des stratégies C et D, et supporter les perturbations importantes qui en résultent.

VII.1.6. Stratégie AD.

Puisque la stratégie AD est la combinaison des aspects positifs de la stratégie A (relativement aux pannes transitoires), et de la stratégie D (vis-à-vis des pannes permanentes), il en résulte que les perturbations qui peuvent être engendrées sont réduites au minimum. Toutefois, compte tenu du fait que chaque SPDi peut être aussi station de reprise d'une SPDj ($i \neq j$), le nombre de msg à conserver et l'espace nécessaire, sont plus important que dans les autres stratégies.

L'overhead relatif à la gestion de ces msg peut également être important. Puisque ce dernier est nettement très inférieur devant celui provoqué par les propagations de reprise, on estime que la stratégie AD est la plus performante vis-à-vis des pénalisations encourues par les applications. Comme la stratégie D, la stratégie AD est particulièrement dépendante de la politique statique de désignation. Dans le cas contraire, il est possible qu'on ne puisse traiter (selon le respect de la stratégie) les pannes permanentes survenant après la première occurrence de ces dernières.

VII.2. Comparaison.

Il est difficile d'établir une comparaison précise entre les stratégies, sachant que leurs utilisations et leurs efficacités sont conditionnées par la classe du système auquelle elles sont dédiées. Aussi, il serait par exemple défavorable d'employer les stratégies AB et AD pour rendre un système tolérant les pannes, dont les capacités de conservation de msg y font défaut, avec une interactivité élevée des processus.

Pour pouvoir effectuer une comparaison, on se basera sur les critères suivants qui définissent globalement l'efficacité d'une stratégie.

VII.2.1. Critères:

C1: Probabilité P' d'engendrer une perturbation lors d'une opération de reprise. Si possible, la propagation sera quantifiée en fonction de ses facteurs: ordre p, niveau n.

C2: Overhead subi par l'application, dû à la préparation de l'environnement nécessaire au mécanisme de reprise pour récupérer une panne (overhead en absence de panne).

C3: Sensibilité à une augmentation du volume du traffic des msg (augmentation de la charge du système de transmission).

C4: Capacités de conservation des msg (espace de conservation, et structure nécessaire à la gestion des msg).

VII.2.2. Cas de latence nulle.

Vue que la stratégie C ne s'intéresse pas à la conservation des msg, seul le graphe de reprise d'un processus est construit au cours de l'exécution. L'overhead induit en absence de panne étant le plus faible possible vis-à-vis des autres stratégies, il servira alors de référence pour estimer celui des stratégies restantes. Ainsi, OV_A: Overhead généré par l'application de la stratégie A est estimé à:

$$OV_A = OV_C + OGM_A ;$$

où OGM_A est le temps nécessaire à la gestion des msg conservés en stratégie A.

$$OV_B = OV_C + OGM_B ;$$

$$OV_{AD} = OV_A + OV_D$$

$$= OV_C + OGM_A + OV_C + OGM_D$$

$$= 2OV_C + OGM_A + OGM_D ;$$

De même:

$$OV_{AB} = OV_A + OV_B$$

$$= OV_C + OGM_A + OV_C + OGM_B$$

$$OV_{AB} = 2OV_C + OMG_A + OMG_B ;$$

$$OV_{AB} - OV_{AD} = OGM_B - OGM_D$$

La comparaison entre OV_{AB} et OV_{AD} revient à comparer OGM_D et OGM_B. Aussi, il est difficile d'estimer ces deux derniers paramètres. Cependant, pour un même nombre de msg échangés entre un même ensemble de processus interactifs m, m = card(P), la concentration éventuelle de msg au niveau de la station de reprise (ses propres msg et ceux destinés à la station défaillante) en stratégie D, peut engendrer d'avantage de pénalités qu'en stratégie B. On peut écrire:

$OGM_B \leq OGM_D$

d'où $OV_{AB} \leq OV_{AD}$.

Puisque la gestion des msg captés, au niveau d'une station de reprise (D), est sensiblement identique à celle qui aurait lieu en station défaillante (A), alors $OGM_D = OGM_A$.

Le facteur OGM_x (overhead dû à la gestion des msg conservés en stratégie x) est proportionnel au nombre de msg à conserver; il s'en suit:

$OGM_B \leq OGM_A = OGM_D$; car le nombre de msg conservés en A ou en D est réparti en B sur un certain nombre de stations.

D'où finallement:

$$OV_C < OV_B \leq OV_A = OV_D < OV_{AB} \leq OV_{AD}.$$

Concernant C3, on peut dire ce qui suit.

La stratégie AD constitue une référence relativement à l'influence exercée sur le traffic msg, puisque hormis les msg d'invocation à la reprise, le processus défaillant dispose de tout l'environnement exigé par la réexécution. Ainsi, STM_{AD} (Sensibilité au Traffic Msg) prend la valeur la plus faible.

Pour une même probabilité de panne, STM_B représente la valeur la plus élevée puisqu'il y a renvoi des msg des producteurs vers le processeur de reprise et ce, quelque soit le type de panne.

Vue l'importance de la fréquence d'occurrence des pannes transitoires vis-à-vis des pannes permanentes, le renvoi des msg de la station de reprise à la station défaillante (en pannes transitoires) donne à la stratégie D, une valeur de STM_D immédiatement inférieure à STM_B.

D'où $STM_D < STM_B$.

Pour raison de fréquence faible d'apparition de pannes permanentes, STM_A est le plus faible après STM_{AD}.

Ainsi, on peut écrire:

$$STM_A < STM_D < STM_B.$$

La stratégie C fait reprendre l'ensemble des processus interactifs depuis la dernière RL; le traffic msg sera approximativement

semblable à celui de l'exécution normale antérieure à l'opération de reprise, sans augmentation notable. On peut écrire:

$$STM_C = STM_A.$$

La stratégie AB ne fait intervenir le renvoi de msg qu'en panne permanente qui peut éventuellement contribuer à augmenter le traffic d'où:

$$STM_A \leq STM_{AB}.$$

Finalement on peut écrire en général:

$$STM_{AD} < STM_C = STM_A < STM_{AB} < STM_D < STM_B.$$

Relativement à C4, on peut dire:

Soit EM_x: l'espace mémoire nécessité par la stratégie x. Parmi l'ensemble des stratégies, la stratégie C s'avère la moins consommatrice d'espace, car aucun msg n'est conservé. Ainsi EM_C sera pris comme référence pour la comparaison ci-après.

$EM_A = EM_C + EGM_A$; EGM_x: espace nécessité par la gestion des msg en stratégie x.

EGM_A: correspond à l'espace mémoire nécessité par la mise en place de la structure de données pour la conservation et la réutilisation ultérieure des msg.

$$EM_B = EM_C + EGM_B.$$

Vu que le nombre de msg gérés par la station est en général plus important en stratégie A qu'en stratégie B, alors:

$EGM_B \leq EGM_A$ (1) ; d'où $EM_B \leq EM_A$.

$$EM_D = EM_C + EGM_D;$$

Du point de vue conservation des msg, la stratégie D est symétrique de A, aussi EGM_D est sensiblement égal à EGM_A d'où

$$EM_D = EM_A \quad (2)$$

$$EM_{AD} = EM_A + EM_D$$

$$= EM_C + EGM_A + EM_C + EGM_D$$

$$= 2EM_C + EGM_A + EGM_D \qquad (3)$$

$$= 2EM_C + 2EGM_A = 2(EM_C + EGM_A)$$

$$= 2EM_A.$$

$$EM_{AD} = 2EM_A.$$

$$EM_{AB} = EM_A + EM_B = EM_C + EGM_A + EM_C + EGM_B$$

$$= 2EM_C + EGM_A + EGM_B \qquad (4)$$

$$EM_{AB} - EM_{AD} = (4) - (3) = EGM_B - EGM_D.$$

$$(2) \Longrightarrow EM_{AB} - EM_{AD} = EGM_B - EGM_A$$

$$(1) \Longrightarrow EGM_B - EGM_A \leq 0$$

Donc $EM_{AB} \leq EM_{AD}$.

D'où finalement:

$$EM_C < EM_B \leq EM_A = EM_D < EM_{AB} \leq EM_{AD}.$$

Récapitulatif.

soit Pr_x: la perturbation engendrée par la stratégie x.

On a:

C1: $Pr_B = Pr_D = Pr_{AD} = Pr_{AB} < Pr_A \leq Pr_C$

C2: $OV_C < OV_B \leq OV_A = OV_D < OV_{AB} = OV_{AD}$

(I)

C3: $STM_{AD} < STM_C = STM_A < STM_{AB} < STM_D < STM_B$

C4: $EM_C < EM_B \leq EM_A = EM_D < EM_{AB} \leq EM_{AD}$

d'où le tableau suivant :

↓ STRATEGIES CRITERES →

	C1	C2	C3	C4
A	P = 1 $0 \leq n \leq m$	$OV_A \approx OV_C + OGM_A$ (2)	$STM_A \approx STM_{AD} + M_A$ (3)	$EM_A \approx EM_c + EMG_A$ (4)
B	n = 0 p = 1	$OV_B \approx OV_C + OGM_B$	$STM_b \approx STM_{AD} + M_B$	$EMB_B \approx EMC_C + EGM_B$
C	p = 1 $0 \leq n \leq m$	Overhead induit (OV_C) induit par C constitue une référence puisque il est le plus faible.	$STM_C \approx STM_{AD} + M_C$	Espace mémoire (EM_C) nécessité par C est considéré comme référence vis-à-vis des autres stratégies car il est le plus faible.
D	P = 1 n = 0	$OV_D \approx OV_C + OGM_D$	$STM_D \approx STM_{AD} + M_D$	$EM_D \approx EM_C + EGM_D$
AD	P = 1 n = 0	$OV_{AD} \approx OV_A + OV_D$ $OV_{AD} \approx 2OV_C + OGM_A + OGM_D$	La sensibilité à l'accroissement du traffic msg (STM) étant la plus faible STM servira de référence	$EM_{AD} \approx EM_A + EM_D$ $\approx 2EM_C + EGM_A + EGM_D$
AB	p = 1 n = 0	$OV_{AB} \approx OV_A + OV_B$ $\approx 2OV_C + OGM_A + OGM_B$	$STM_{AB} \approx STM_{AD} + M_{AB}$	$EM_{AB} \approx EM_A + EM_B$ $\approx 2EM_C + EGM_A + EGM_B$

- Latence d'erreur nulle, critères(1,2) satisfaits-

(1): m = nombre de processus ayant échangé des informations avec le processus défaillant.

(2): OGM_x : Overhead dû à la gestion des msg conservés en stratégie X

(3): M_x : nombre de msg nécessités lors de la reprise par la stratégie X

(4): EGMx = Espace requis par la gestion des msg en stratégie x

VII.2.3. Cas de latence non nulle.

C1: Toute panne confondue (transitoire ou permanente), la probabilité P' pour qu'une propagation soit de niveau n=m (m=card(P)), est conditionnée par deux facteurs: la postpropagation et la rétropropagation. Ainsi la stratégie C peut être dotée de la plus grande valeur; vient juste avant la stratégie A puisqu'elle requière les deux facteurs en panne permanentes. Les stratégies D et AD sont soumises à l'unique postpropagation jusqu'à la première panne permanente, après quoi elles requièrent les deux.
D'où:
$P'_D = P'_{AD} < P'_A < P'_C$. Les stratégies B et AB n'exigent que la postpropagation quelque soit le type de panne, On a alors:
$P'_B = P'_{AB}$. Finalement:

$$P'_B = P'_{AB} < P'_D = P'_{AD} < P'_A < P'_C.$$

C2: Le paramètre overhead en absence de panne est identique qu'en cas de latence nulle, d'où :

$$OV_C < OV_B < OV_A = OV_D < OV_{AB} = OV_{AD}.$$

C3: La sensibilité à l'augmentation du volume du traffic msg, se trouve légèrement accrue suite à l'émission et la propagation des msg de reprise. Elle reste cependant dans l'ordre du cas de latence nulle.
On a approximativement:

$$STM_{AD} < STM_C = STM_A < STM_{AB} < STM_D < STM_B.$$

C4: Semblable au cas de latence nulle.

$$EM_C < EM_B \leq EM_A = EM_D < EM_{AB} \leq EM_{AD}.$$

On retrouve les différents critères dans le tableau comparatif suivant.

Critéres → / Stratégies ↓	C1: Paramètres de propagation ou probabilité de générer une perturbation	C2: Overhead généré en absence de panne	C3: Sensibilité à l'accroissement du traffic msg	C4: Sensibilité à une forte demande d'espace mémoire
A	p = 1 0 ≤ n ≤ m	Identique au cas de latence nulle	Relativement semblable au cas de latence nulle, avec toutefois une légère augmentation dûe aux msg de reprise	Identique au cas de latence nulle
B	p = 1 0 ≤ n ≤ m	//	//	//
C	p = 1 0 ≤ n ≤ m	//	//	//
D	p = 1 0 ≤ n ≤ m	//	//	//
AD	p = 1 0 ≤ n ≤ m	//	//	//
AB	p = 1 0 ≤ n ≤ m	//	//	//

-Latence d' erreur non nulle-
critères (1,2) non satisfaits.

VII.3. CONCLUSION.

Le choix d'une stratégie est déterminé en fonction des contraintes de l'application et des caractéristiques de la stratégie. Ainsi, par exemple, si on ne s'intéresse qu'au problème de perturbations (à l'occurrence de panne) et qu' on veuille privilègie l'optimisation de l'overhead en absence de panne, au détriment d'un accroissement du volume du traffic-msg, la stratégie B répond au mieux à cet objectif (voir récapitulatif (I)).

Si les capacités de conservation des msg ne font pas défaut, et si l'environnement matériel est soumis à une fréquence importante d'apparition de pannes transitoires, tout en acceptant un certain overhead en exécution normale et une limite de ce dernier en exécution exceptionnelle, le choix de la stratégie à adopter serait porté sur AB.

La stratégie C qui parait la plus perturbatrice à l'occurrence de panne, est la plus attrayante pour ses exigences minimales: de l'espace mémoire, et de son overhead en absence de panne.

En définitive, une stratégie bonne pour une application donnée peut ne pas l'être pour une autre. C'est donc le privilège accordé à un critère Ci (i= 1,4) ou à un compromis établi entre plusieurs, et en fonction des exigences de l'application, que sera déterminée la stratégie à adopter.

Chapitre VIII

CREATION DES Rp

VIII.1. Algorithme de création coordonnée des Rp.

La création simultanée de Rp au sein de processus interactifs doit être synchronisée de sorte à pouvoir récupérer un état cohérent du système même à l'occurrence de panne survenant pendant la phase de création. Cette phase doit également prendre en compte la cohérence des Rp des stations SPDi avec leurs copies sur les stations de reprise SRi associées [WIL79, MIL80]. Lorsqu'une décision de création d'un Rp au profit d'un processus Pi est prise par le système d'une SPDi, celle-ci est transmise à tous les systèmes des des stations SPDj telle que, il existe un pocessus Pj de SPDj et Pj --> Pi ou bien Pi --> Pj, et ce, depuis la région de reprise courante de Pi (Pj est propagateur ou dépendant direct de Pi).

* SPDi joue désormais, le rôle de l'initiateur de l'opération ou noeud-racine. Après l'émission de cette décision sous forme d'un msg du type CREER-Rp(initiateur, émétteur), l'exécution courante de Pi est suspendue, et deux attitudes sont alors possibles:

1- Attente d'une réponse à la décision.

2- Créer un Rp (avec transmission d'une copie sur station de reprise).

* Toute station SPDj recevant un tel msg, détermine son rang dans cette opération hiérarchisée; deux cas sont alors possibles:

1- SPDj est un noeud intermédiaire ou père, c'est à dire: Il existe SPDk tel que il existe Pk de SPDk et Pk --> Pj ou Pj --> Pk (relativement à la Région de Reprise(RR) courante), alors SPDj agit comme père, en diffusant la décision à tous ses fils exepté son père; puis crée localement un Rp pour Pj, et transmet une copie à la station de reprise associée.

2- SPDj est un noeud terminal ou feuille, alors il crée un Rp pour Pj, puis transmet une copie à la station de reprise de SPDj où est sauvegardé un ancien Rp de Pj. Une fois cette opération terminée, le noeud feuille émet un msg à son père proclamant cette terminaison, puis se met en attente de validation.

* Lorsqu' un noeud intermédiaire (ou père) reçoit de tous ses fils le msg de terminaison de création des Rp (local et station de reprise), il informe à son tour son père, puis attend la validation.

* Lorsque le noeud racine (initiateur) reçoit de ses fils les msg de terminaison (l'ensemble des processus ayant donc créé un Rp sur le site origine et sur le site de reprise), SPDi émet un msg diffusé de validation. Ce msg consiste, dès sa réception à ce que tout noeud SPDj (père ou feuille) émette un msg de validation à son site de reprise pour mettre à jour l'ancien Rp par le nouveau, puis procéder à un engagement envers l'ancien Rp sur SPDj. Ainsi, SPDj vient de créer un nouveau Rp à la fois sur le site local de Pj et sur le site de reprise. Pj peut donc continuer son exécution interrompue par la création d'un Rp.

Pour l'initiateur SPDi, il est plus avantageux d'éviter une attente de réponse à un msg de création de Rp. Aussi, SPDi va combler cette attente par la création d'un nouveau Rp pour Pi, puis transmet une copie à la station de reprise de Pi, et éventuellement se met en attente de réponse.
Ce n'est qu'après avoir validé lui-même localement le Rp créé pour Pi, et ordonné à la station de reprise de considérer comme seul valide le nouveau Rp, que SPDi transmet son msg diffusé de validation.

VIII.2. Occurrence de panne lors de la création.

Une panne transitoire qui survient au cours de l'opération de création de Rp, peut être traitée par le processeur de la station sur laquelle elle s'est manifestée, par un retour au début de l'étape de création de ce Rp. Il n'y a pas eu d'échanges avec les autres stations donc, pas de contamination.

Le problème provient d'une panne permanente, susceptible de

générer:
- Soit une incohérence entre copies des Rp sur une station SPDi et la station de reprise correspondante;
- Soit entre Rp constituant la nouvelle ligne de reprise; c'est à dire, RL contient de nouveaux et d'anciens Rp.

Une règle à observer est : Toute opération de création de Rp doit être atomique. Autrement dit, lorsqu'une panne (permanente) se manifeste, on doit être dans l'un des cas suivants:

- La RL disponible est celle formée des anciens Rp, on défait ainsi, tout ce qui concerne les nouveaux Rp.

- La RL disponible est celle formée par les nouveaux Rp, l'ancienne est soit détruite, ou le sera au moment opportun.

Donc lorsqu'une panne est détectée entre l'émission du msg proclamant la décision de créer un nouveau Rp, et la réception de la réponse correspondante (Rp-créé) au niveau du noeud racine, l'opération est avortée et l'état cohérent est donné par l'ancienne RL.

De même, lorsque la panne est détectée après l'émission du msg de validation, la nouvelle RL étant disponible, il est possible que la station défaillante n'ait pas réussi à ordonner à sa station de reprise de valider le dernier Rp. Il est alors indispensable qu'elle puisse le faire en dépit de la panne. Pour cela, soit que le msg de validation émis par la racine est diffusé, donc destiné et capté par toute station en attente de validation; ou que la station de reprise en attente de validation, sachant que sa SPD est en panne, émet un msg d'enquête pour déterminer s'il faut ou non valider. Le premier cas étant plus efficace, puisqu'il optimise le nombre de msg émis et évite, la transmission hiérarchique de père en fils et le cas où la station défaillante est père de la station de reprise associée.

Au cas où le noeud racine SPDi tombe lui-même en panne après avoir validé le Rp de sa station de reprise, celle-ci prend le relais, en émettant un msg de validation comme l'aurait fait SPDi.

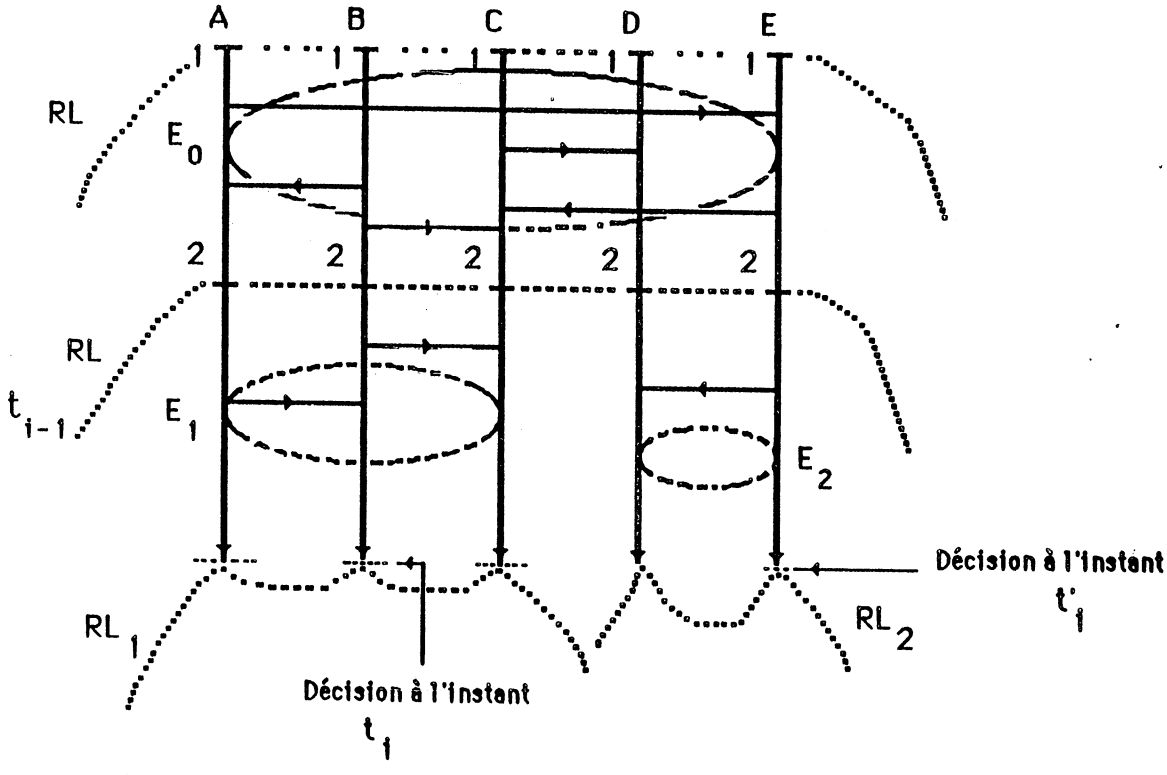
Du point de vue espace, un seul Rp est maintenu sauvegardé par processus lors du déroulement de l'application (il en existe deux

pendant l'opération de création, l'ancien Rp est supprimé après validation).

VIII.3. Interférence des décisions.

Il est possible que la décision de création de Rp soit prise simultanément par deux (ou plusieurs) stations, ou à des intervalles de temps qui se recouvrent. Il importe alors qu'un seul noeud racine ait ce pouvoir de décision. Cette interférence concerne, bien entendu, les décisions prises par des stations appartenant à un même ensemble de processus communicants, dont les échanges ont eu lieu dans les régions de reprise courantes.

Exemple:



Cas d'interférence de décisions.

Figure 01

Les processus <A,B,C,D,E> qui constituaient à l'instant t_{i-1} un même ensemble P de processus communicants, et concernés par une décision unique, se sont scindés en deux sous-ensembles P1= <A,B,C>

et P2= <D,E> indépendants au sens de la décision. Chacun d'eux n'est concerné que par sa propre décision, bien qu'à l'instant $t_i = t'_i$ deux décisions simultanées peuvent être prises.
Soit par exemple dans P1, deux décisions interférentes émanant de A et C, il faut que le rôle du noeud racine soit assuré exclusivement par A ou C, et qu'au niveau de B une seule décision soit prise en compte.

Pour résoudre ce problème, on peut adopter un mécanisme de priorité, basée par exemple sur le principe d'ordonnancement par estampille [LAM 78]. Ainsi un noeud recevant plusieurs msg de création de Rp, issus de racines différentes, et reconnus comme tels grâce à l'identification qui figure dans le msg, agit selon le principe de l'algorithme, en exécutant l'action prévue (création de Rp) et en transmettant éventuellement le msg aux noeuds fils. Chacun des initiateurs recevra le msg émis par les autres, et en fonction de l'estampille véhiculée dans le msg reçu, il déterminera le rôle à jouer: Agir en tant que noeud intermédiaire ou terminal, ou bien maintenir sa position d'initiateur car il est plus prioritaire.

Dans P1, A et C recevront les msg de création de Rp, et chacun déterminera l'attitude à prendre (feuille ou racine). Même dans le cas où la décision est prise par une station unique SPDi, il arrive qu'un noeud intermédiaire ou feuille ait plusieurs pères, par exemple en E0 (fig 1), si l'initiateur est SPD_A, le noeud intermédiaire SPD_C recevra deux msg de création appartenant à une même session émanant de SPD_B et de SPD_E. SPD_C doit exécuter une seule fois l'action auquelle fait référence un msg et répondre aux deux pères (SPD_B et SPD_E).

Finalement, les msg impliqués sont de trois types.

- Msg de création de Rp diffusé par l'initiateur (ou racine), il peut être du type:
  CREER-Rp (initiateur, estampille-initiateur, émetteur, récepteur);

- Msg de fin de création:
  Rp-CREE (initiateur,émetteur,récepteur);

- Msg de validation: VALIDER (initiateur, émetteur,récepteur).

Initiateur: fait référence à une identification de session de création de Rp, associée à l'initiateur lui-même. Dans VALIDER, récepteur peut indiquer une identification particulière pour permettre une diffusion (un seul msg destiné à l'ensemble des processus concernés).

VIII.4. Evitement d'interblocage.

Il est des situations où un risque d'interblocage, entre processus en attente de réponse aux msg CREER-Rp, est possible. Ces situations sont dûes à l'existence d'un cycle de communication interprocessus. Par exemple dans EO (fig 1), si on suppose A comme initiateur, un msg CREER-Rp sera reçu par B et E, lesquels agiront en tant que noeuds intermédiaires (pères), et C recevra deux msg CREER-Rp provenant de B et E.
Ces msg peuvent ne pas être reçus en même temps par C (fort probable) alors, après réception de l'un, C agit comme noeud intermédiaire en émettant un msg CREER-Rp à l'autre dont le msg n'est pas encore reçu. C attendra donc une réponse de E ou de D, lequel à son tour attend une réponse de C, après lui avoir transmis un msg CREER-Rp: C'est l'interblocage.

Pour éviter cet interblocage, il faut rompre le cycle d'attente. Pour cela, une fois le msg CREER-Rp reçu, le noeud SPD_i le diffuse vers les SPD_j adéquats (sauf vers le (ou les) père(s)), puis exécute l'action demandée. Si entre temps SPD_j a aussi émis le même msg vers SPD_i, il faut que l'un d'eux puisse jouer le rôle de feuille, et céder celui du Père à l'autre.

La solution revient à l'utilisation d'<u>estampille</u> qui sera véhiculée dans les msg pour départager les noeuds pères. C'est ainsi que dans l'exemple précédent, lorsque C ayant reçu de B le msg CREER-Rp, le transmet à E lequel dès réception (entre temps E a émis le même msg à C) saura, après comparaison de l'estampille de C et la sienne, l'attitude à tenir envers C. Soit il se considère comme noeud terminal et transmet à C une réponse Rp-CREE au moment

opportun, ou comme noeud intermédiaire et attendre de C la réponse au msg CREER-Rp. La même situation est observée au niveau de C vis-à-vis de E.

Donc le msg CREER-Rp portera, non seulement l'estampille du noeud racine, mais aussi celle de tout noeud émetteur.

Chapitre IX

OPTIMISATION

IX.1. Elimination du protocole interprocessus.

Une contrainte implicite présente dans toute stratégie, est l'obligation de tout processus récepteur de renvoyer à son émetteur, la région de reprise de réception de tout msg reçu. Il est possible, dans certain cas, de pouvoir éliminer cette contrainte qui est la base de la constitution des listes LD des processus. Donc d'éliminer de ce fait l'invocation précise lors de la postpropagation.

Dans la première partie de l'étude "stratégies sans contraites", il a été montré que la reprise d'une panne permanente nécessite impérativement et conjointivement l'application des deux types d'invocation. D'où l'optimisation ne peut concerner que les pannes trasitoires, mais son importance est plus grande au niveau de la partie II "stratégies avec contraintes", comme nous allons le voir.

IX.1.1. Stratégies sans contraintes.

- Lorsqu'un processus émet un msg d'information, l'identificateur du processus récepteur est enregistré dans la liste LD de l'émetteur. Cette liste qui contenait avant des indentificateurs de Rp, contiendra désormais ceux des processus dépendants directs.

- Dès la réception d'un msg, le récepteur porte dans la liste LIP du Rp courant, l'identificateur du Rp propagateur qui figure dans le msg; la liste LP reste donc inchangée.

Lors d'une opération de reprise, suite à une panne transitoire, la rétropropagation peut être invoquée de façon précise. Quant à la postpropagation, le processeur de reprise, par l'intermédiaire de la liste LD du processus défaillant, désigne dans le msg de reprise, le processus concerné qui doit déterminer lui-même son RpR en fonction du RpR du défaillant fourni dans le msg. Ce msg peut

être de la forme: REPRISE (Emetteur, RpR de l'émetteur, Récepteur). Cependant, l'application de ce principe ne convient guère aux pannes permanentes, donc "utilisable" en partie I.

IX.1.2. Stratégies avec contraintes.

Ce cas se prête beaucoup à l'optimisation, car quelque soit le type d'invocation, un seul Rp est impliqué. Ainsi lorsqu'un processus est atteint par une propagation, le RpR est déjà connu même en absence de précision.

Le même principe qu'en IX.1.1. est adopté, la liste LD d'un processus Pi sera constituée d'identificateurs de tous les processus dépendants directs de Pi, la liste LP, à l'inverse de IX.1.1., peut être formée uniquement d'identificateurs de processus propagateurs directs de Pi. Il n'y a en fait aucune ambiguïté que ces listes LP et LD soient constituées d'identificateurs de Rp ou de processus, puisque tout partenaire de Pi est représenté par un élément unique dans ces listes.

IX.1.3. Avantages.

L'absence de contrainte imposée à tout processus Pi de transmettre à ses émetteurs les RRR dès réception de msg, résulte une amélioration au niveau de:

- Contribution moindre à surcharger le système de transmission.
- Réduction sensible de l'overhead créé par la gestion de msg (interruptions du processus, pour la prise en compte des renvois, moins fréquentes).
- Table d'informations nécessaires à la satisfaction des critères (1,2) de non postpropagation est fortement réduite (à 40%) puisque les colonnes réservées aux IRRE et IRRR (fig 6 stratégie A) sont inutiles.

IX.2. Elimination des Rp inaccessibles.

Lorsqu'une coordination, lors de l'établissement des communications interprocessus ou lors des créations de Rp, est prévue de sorte que la RL soit prédéterminée, l'optimisation du nombre de Rp

à maintenir sauvegardés se trouve implicitement considéré. C'est le cas dans l'étude des stratégies avec contrainte (partie II) ou du schéma de conservation (Ran75).

Par contre, lorsque la RL est déterminée au cours de l'opération de reprise, donc de façon dynamique, et sachant qu'une RL ne peut contenir q'un Rp au plus par processus, il existe alors des Rp maintenus enregistrés inutilement. Il importe alors de pouvoir les reconnaître afin de les éliminer et récupérer l'espace occupé, notamment lorsque la ressource espace-mémoire fait défaut.

Apparemment, lorsque les critères (1,2) sont satisfaits, les stratégies B,D,AD,AB nécessitent la sauvegarde d'un Rp unique. Autrement dit, tout Rp ayant reçu un engagement de son processus devient inaccessible (éliminable). Cette propriété, surtout valable en pannes transitoires, diminue l'efficacité de la reprise relative à ces dernières, notamment après recouvrement de la panne permanente. Pour pouvoir maintenir cette efficacité, il importe de traiter ce problème avec la plus grande considération.

Soit P un ensemble de processus communicants, et soit "#" le symbole exprimant l'engagement (D2) d'un processus Pi ∈ P, envers un de ses points de reprise Rpi (notation: Pi # Rpi).

Axiome:

∀ Pi, ∀Rpi ∈ Pi, Rpi est soit indispensable (non éliminable) soit inaccessible (éliminable).

IX.2.1. Détermination des Rp inaccessibles en stratégie A.

IX.2.1.1. Critères (1,2) satisfaits.

La postpropagation est donc évitée.

proposition 1:

> Tous types de pannes confondus, avec critères (1,2) satisfaits (une panne permanente unique tolérée), on a:
> ∀ Rpi ∈ Pi, Pi ∈ P, Rpi est indispensable ssi une

au moins des conditions suivantes est réalisée.

1- ¬ (Pi # Rpi); (¬ = non)

2-] Rpj ϵ Pj telque Rpi --> Rpj et ¬(Pj # Rpj).

Preuve:

1- De part le principe de la stratégie A (et avec critères (1,2) satisfaits), tout processus affecté par une panne, est repris à partir de son Rp actif. Ce dernier n'ayant pas encore reçu d'engagement, il est donc indispensable (D12) (on rapelle que tout processus qui crée un nouveau Rpi, exprime son engagement envers Rpi-1). Donc tantqu'un Rp est actif, il est indispensable.

2- Tout Rp ayant reçu un engagement se trouve dans l'une des situations suivantes:

a- Il n'existe aucun Rpj ϵ Pj tel que Rpj --> Rpi ou Rpi --> Rpj, alors Rpi est inaccessible. Autrement dit, aucun échange n'ait lieu dans la région de reprise associée à Rpi.

b- Il existe Rpj ϵ Pj telque Rpi --> Rpj, alors deux cas sont possibles:

- ¬ (Pj # Rpj), alors par rétropropagation nécessitée par une panne permanente affectant éventuellement Pj, Rpi est alors indispensable.

- Pj # Rpj, alors Rpi est inaccessible car toute rétropropagation serait stoppée avant d'atteindre Pi.

Finalement, ∀ Rpi ϵ Pi, Pi ϵ P, Rpi est éliminable s'il ne satisfait aucune des conditions précédentes.

IX.2.1.1.1. Détermination des Rp inaccessibles.

Sachant que tout Rp ne vérifiant pas la proposition 1 acquière la propriété d'inaccessibilité, un algorithme possible pour déterminer ces Rp peut être le suivant.

1- Après chaque engagement d'un processus Pi envers Rpi (Rpi+1 étant crée), la liste LIP de Rpi est examinée; si celle-ci est non vide alors:
Proclamation par Pi de l'engagement envers Rpi en émettant un msg à cet effet à tout propagateur direct de Rpi. Le msg peut être de la forme: Engagement(Pi, Rpi). Si la liste LID de Rpi est vide alors Rpi devient inaccessible et les actions appropriées sont exécutées (élimination de Rpi de la liste LRp de Pi, suppression du contexte qui lui est associé). Sinon Pi "marque" Rpi comme ayant reçu un engagement.

2- Lorsqu'un processus Pj reçoit un msg d'engagement de la part d'un partenaire Pi, Pj agit comme suit:

. Examine sa liste LD pour "marquer" Rpi;

. Si Rpj (Rpj --> Rpi) et tout élément de sa liste LID sont marqués alors Rpj est déclaré inaccessible et les actions appropriées sont exécutées.

IX.2.1.2. Critères (1,2) non satisfaits.

Dans le cas le plus général, où la latence est relativement prise en compte, imposant ainsi une postpropagation systématique, la détermination des Rp inaccessibles devient plus complexe et onéreuse. Comme la ressource espace peut s'avérer critique, il importe de récupérer l'espace occupé par des contextes de Rp maintenus sauvegardés inutilement.

Proposition 2:

∀ Rpi ∈ Pi, Pi ∈ P, Rpi est indispensable ssi:
Une au moins des conditions suivantes est réalisée.
1- ¬ (Pi # Rpi).
2- Il existe Rpj ∈ Pj, Pj ∈ P, telque Rpj est soit PIPR (D7) de Rpi, ou bien Rpi --> Rpj et ¬ (Pj # Rpj).
3- Il existe Rpj ∈ Pj telque Rpj --/-> Rpi, et Rpj est indispensable.

Condition nécessaire:

a- Rpi indispensable ==> condition 1 ou condition 2 ou condition 3.

Montrons plutôt que la contraposée de a) est vraie. C'est à dire: (Pi#Rpi) et il n'existe pas Rpj ∈ Pi tel que Rpj est PIPR de Rpi ou bien [Rpi --> Rpj et ¬ (Pj#Rpj) et il n'existe pas Rpj ∈ Pj telque Rpj --/-> Rpi et Rpj indispensable] ==> Rpi est inaccessible. Evident, puisque cela signifie soit qu'aucun échange n'ait lieu dans la région de reprise associée à Rpi, soit que tout dépendant et/ou propagateur indirect de Rpi a reçu un engagement. Autrement dit, Rpi est en "amont" d'une ligne de reprise, et d'après l'axiome Rpi est inaccessible.

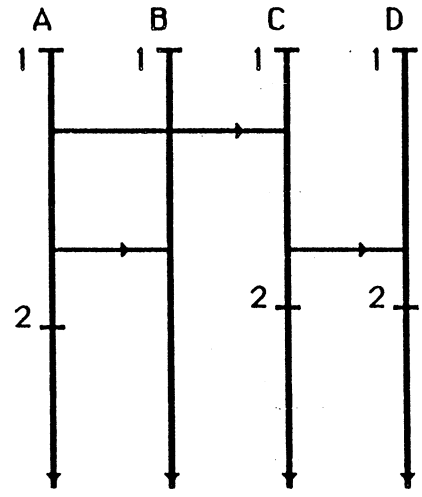
Condition suffisante:

b- Condition 1 ou condition 2 ou condition 3 ==> Rpi indispensable.

Utilisons le raisonnement par contraposition.
Soit Rpi inaccessible, alors ∀ Pj en panne, la propagation ne peut atteindre Rpi d'où: Rpi a reçu un engagement de Pi et, soit Rpi est isolé (pas d'interactions dans la région de reprise associée à Rpi), ou bien tout dépendant indirect et propagateur indirect Rpj de Rpi, Rpj a reçu un engagement de Pj, donc Rpj ne peut être un PIPR ou un RIPR. Contraposée de b vraie ==> b vraie.

La troisième condition d'indispensabilité résulte directement de la notion de propagation suivant la relation de dépendance entre Rp. Tantqu'un Rp est indispensable, tous ses dépendants indirects le sont aussi.

Exemple:



Exemple de Rp indispensables.

Figure 01

- Le Rp B1 est indispensable car ¬ (B#B1); condition 1.
- A1 est indispensable car B1 est RIPR de A1; condition 2.
- C1 et D1 sont ausssi indispensables car ils sont dépendants indirects de A1 qui est indispensable; condition 3.

IX.2.1.2.1. Détermination des Rp inaccessibles.

Soient Pi un processus quelconque de P, et Rpi ∈ Pi un point de relance. ∀ Rpi ∈ Pi, Rpi est indispensable si une au moins des conditions suivantes est vérifiée.

1- ¬(Pi#Rpi);
2- Il existe Rpj appartenant à Pj telque Rpj est PIPR de Rpi;
3- Il existe Rpj appartenant à Pj telque Rpi --> Rpj et ¬(Pj#Rpj);
4- Il existe Rpj appartenant à Pj telque Rpj --/-> Rpi et Rpj indispensable.

Autrement dit, un Rpi pour lequel aucune des conditions ci-dessus n'est satisfaite peut être déclaré inaccessible, donc éliminable.

Application:

- Le cas trivial de détermination d'un Rp inaccessible, est celui où aucun échange n'ait lieu dans la région de reprise associée. Donc, lorsqu'un point de reprise Rpi est nouvellement créé, si les listes LID et LIP de son prédécesseur Rpi-1 sont vides, Rpi-1 peut être immédiatement éliminé (y compris le contexte associé).

- Lorsque Rpi est tel que sa liste LIP est vide, et que tous ses dépendants directs Rpj ont reçu chacun un engagement ($\forall$Rpj, Pj#Rpj), il devient inaccessible dès qu'il reçoit lui même un engagement.

- Le cas le plus complexe s'observe lorsque Rpi possède au moins un PIPR. Autrement dit, Rpi est inaccessible s'il ne possède aucun PIPR et tous ses dépendants directs ont reçu chacun un engagement. Tout le problème revient donc à savoir quand un Rp n'a plus de PIPR, d'où la connaisance de l'ensemble de ces derniers. Deux solutions sont possibles:

1- Une liste de PIPR de l'ensemble de Rp d'un processus est gérée au fur et à mesure de la création de Rp et de la "naissance" des PIPR. C'est à dire, chaque fois qu'un point de reprise Rpi acquière un nouveau propagateur, celui-ci étant PIPR, est enregistré dans la liste des PIPR de Rpi, puis cette liste est transmise à tous les dépendants de Rpi qui l'enregistreront et la transmettront de la même façon que Rpi.

   - Lorsqu'un Rpj reçoit un engagement de Pj (après création de Rpj+1), il est éliminé de sa propre liste PIPR (car tant qu'il est actif Rpj est considéré comme son propre PIPR). Un msg proclamant cet engagement est émis à tous ses dépendants, qui dès réception l'élimineront de leurs propres listes PIPR, et le msg est ainsi diffusé à leurs dépendants qui agiront de la même façon.

   - Lorsque la liste PIPR d'un Rpi est vide après création de son successeur Rpi+1, et que tous ses dépendants directs ont cessé d'être actifs, Rpi peut être éliminé de la liste LRp du processus. Le contexte et les différentes listes

(LID,LIP,PIPR) du Rpi sont également éliminés.

Remarque:

Un Rp devient inaccessible lorsque en particulier les deux conditions suivantes sont conjointivement vérifiées.
1- Le Rp ne possède aucun PIPR;
2- Tous ses dépendants directs éventuels ont reçu chacun un engagement.

La structure de données adoptée à cet effet, peut être une liste PIPR semblable aux listes LD et LP déjà existantes. On distinguera de la même façon pour chaque Rpi une liste immédiate de PIPR (qu'on notera LIPIPR) contenant uniquemqnt les PIPR de Rpi. L'union de ces LIPIPR formera la liste LPIPR pour l'ensemble des Rp d'un processus. Ceci étant relatif à la condition 1.
Quant à la condition 2, on procèdera au marquage des dépendants directs de Rpi dès réception de msg de ces derniers proclamant leurs inactivités. Ainsi la condition 2 devient pratiquement: "Tous les dépendants directs sont marqués". IL est possible toutefois de les inclure dans la liste LPIPR et de procéder par élimination de la même façon que les PIPR. Ainsi un Rpi est inaccessible si sa liste LIPIPR est vide (réunion de toutes les conditions en une seule, y compris le cas trivial).

2- Une seconde solution consiste à créer la liste de PIPR de Rpi de la façon suivante:

1- A la création de Rpi, sa liste PIPR est créée, elle contient l'élément Rpi;

2- Chaque fois que Rpi acquière un nouveau propagateur ou dépendant direct, celui-ci est inclu dans PIPR de Rpi;

3- Lorsqu'un Rp, soit Rpj ϵ Pj est tel que pj#Rpj alors Pj proclame cet engagement en émettant un msg à cet effet à:
. Tous les propagateurs directs de Pj.
. Tous les dépendants directs de Pj en leur transmettant tous les PIPR de Rpj.

4- Tout processus ayant reçu un msg proclamant un engagement envers Rpi, enregistre dans PIPR de Rpj (Rpi --> Rpj) l'ensemble des Rp transmis avec une marque identifiant l'inactivité de Rpi. Les propagateurs directs se contentent seulement de marquer Rpi inactif.

5- Après émission ou réception d'un msg d'engagement, la liste PIPR du Rp impliqué est examinée, si tous les éléments sont marqué, alors le Rp correspondant est éliminable.

Remarque:

Les deux solutions sont très proches l'une de l'autre quoique la seconde génère moins de msg.

Exemple

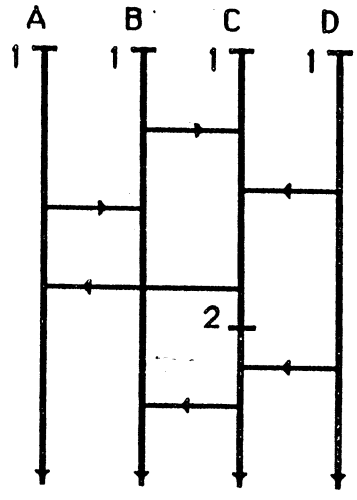
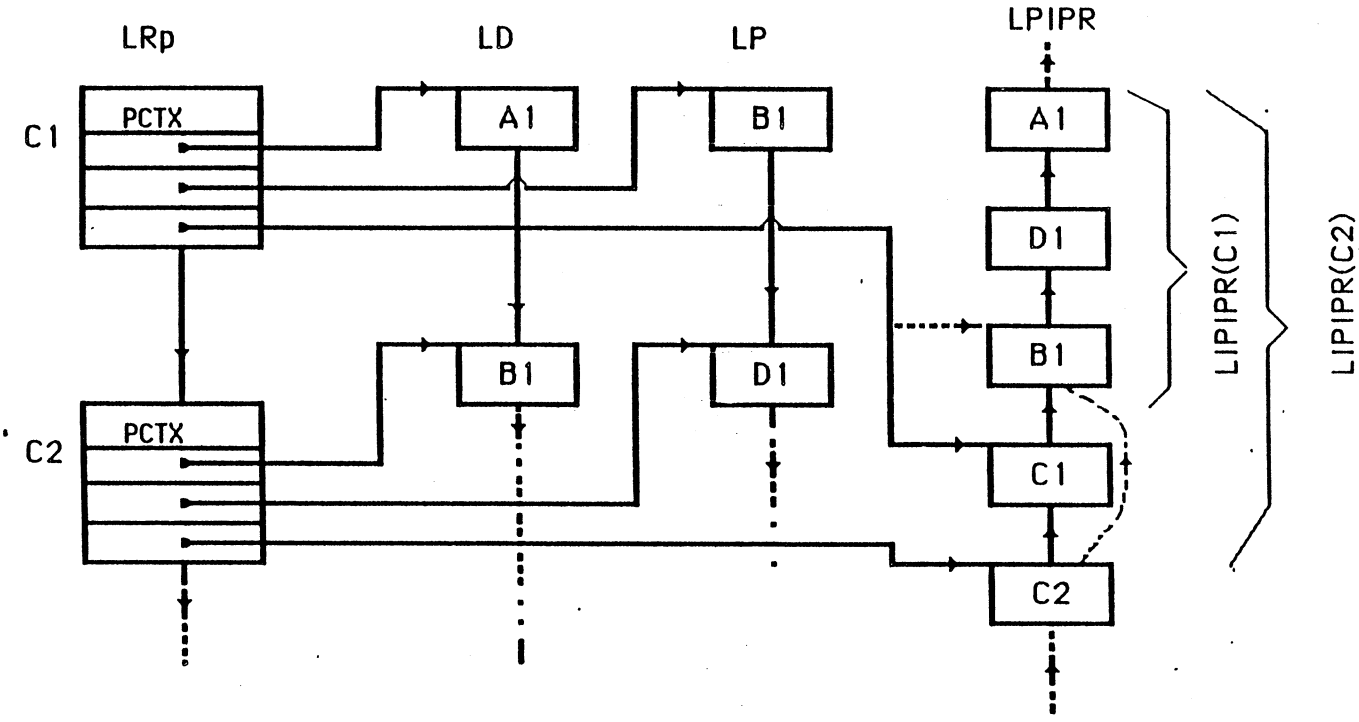


Figure 2a



Structure de données possible regroupant la reprise et la détermination des Rp inaccessibles (processus C).

Figure 2b

A noter que selon la relation de dominance entre Rp d'un même processus, les PIPR d'un Rpi sont aussi PIPR de son successeur Rpi+1. Après création de C2, C1 reçoit l'engagement de C, en éliminant C1 de LIPIPR (C1), puis proclamation à tous les dépendants directs et propagateurs de C1.

IX.2.2. Détermination des Rp inaccessibles en stratégie B.

IX.2.2.1. Critères (1,2) satisfaits.

Soit Rpi un point de reprise d'un processus Pi. ∀ Rpi ∈ Pi, Rpi est indispensable ssi une au moins des conditions suivantes est vérifiée:

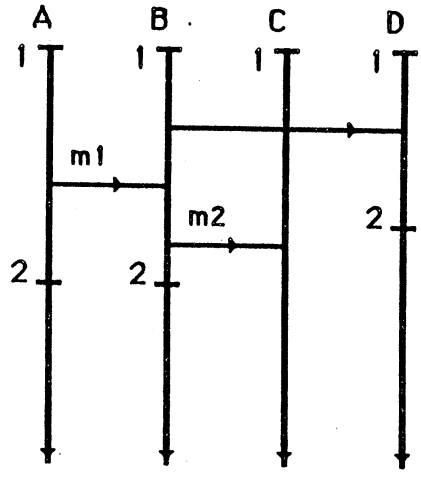
1- ¬ (Pj#Rpj);

2- Il existe Rpj ∈ Pj tel que Rpi --> Rpj et ¬ (Pj#Rpj), (Rpj dépendant direct de Rpi).

Tout Rpi qui ne satisfait aucune de ces conditions peut être

déclaré inaccessible, dont le contexte associé peut être éliminé. Ces conditions tiennent compte d'une plus grande capacité de reprise à savoir, continuer à reprendre les pannes transitoires après recouvrement de l'unique panne permanente autorisée.

Cette attitude signifie plus clairement que dans le cas de l'absence d'une station productrice pour refournir les msg indispensables (panne permanente de celle-ci), la station défaillante impliquée serait contrainte d'invoquer une rétropropagation du processus concerné, comme le montre la figure suivante.



Cas de rétropropagation forcée.

Figure 03

En respectant scrupuleusement le principe de la stratégie B, il y a lieu de considérer alors que tout Rp ayant reçu un engagement devient inaccessible, tels A1, B1, D1.

Cependant comme évoqué en II.4., si on veut améliorer les capacités de reprise en pannes transitoires, il est impératif d'agir, le cas échéant, en invoquant une rétropropagation. C'est notamment la situation représentée par l'exemple suivant (fig 03).

Soit B en panne permanente, le mécanisme de reprise le fait reprendre en B2, et l'exécution continue sur une des stations A,C,D (par exemple). Une éventuelle panne transitoire sur C, le fait

reprendre en C1 avec nécessité de disposer du msg m2. Ce qui ne peut être possible que par provocation de la rétropropagation vers B1, puisque la station de reprise de B ne peut fournir m2. D'où l'indispensabilité de B1 bien qu'il ait reçu un engagement de B (condition 2).

Les conditions précédentes étant identiques à celles de la stratégie A (critères (1,2) satisfaits), les algorithmes de détermination de Rp inaccessibles demeurent applicables dans le présent cas.

IX.2.2.2. Critères (1,2) non satisfaits.

Dans ce cas la postpropagation devient systématique.

∀ Rpi ∈ Pi, Rpi est indispensable ssi une au moins des conditions suivantes est vérifiée.

1- ¬ (Pi#Rpi);
2- Il existe Rpj ∈ Pj tel que: Rpj est PIPR de Rpi;
3- Il existe Rpj ∈ Pj tel que: Rpi --> Rpj et ¬ (Pj#Rpj);
4- Il existe Rpj ∈ Pj tel que: Rpj --/-> Rpi et Rpj est indispensable;

Un Rp est inaccessible s'il ne satisfait aucune de ces conditions.

Application

D'un point de vue pratique, un point de reprise, Rpi devient inaccessible lorsque:

- Aucun échange n'ait lieu dans la région de reprise qu'il identifie. Ce qui revient à dire que ses listes LID et LIP sont vides.

- Tous ses propagateurs indirects ont reçu chacun un engagement, et aucun de ses dépendants directs n'est actif. Il faut donc pouvoir identifier tous les propagateurs indirects de Rpi, tous ses dépendants directs et de reconnaitre quand tous ces Rp cessent d'être actifs. Pour cela on peut adopter l'algorithme suivant:

1- Pour tout Rpi créé, une liste LRPI de Rp indispensables est créée en même temps que les listes LID et LIP de Rpi. LRPI contiendra comme premier élément Rpi.

2- Chaque fois que Rpi acquière un nouveau propagateur ou dépendant Rpj, celui-ci est inséré dans LRPI(Rpi). Si Rpj est propagateur alors, Pi le transmet à tous les dépendants de Rpi qui l'insèreront dans leurs LRPI et agiront de la même façon que Pi. Si Rpj est dépendant, Pi le transmet à tous les autres dépendants de Rpi qui incluront Rpj dans leurs LRPI et agiront de la même façon que Pi.

Les étapes 1 et 2 concernent la construction des listes LRPI au fur et à mesure de l'évolution des processus.

3- Chaque fois que Pi exprime un engagement envers Rpi, Rpi est éliminé de sa liste LRPI.

- Un msg proclamant cet engagement est émis à tout propagateur direct Rpj de Rpi dont l'action à entreprendre est l'élimination de Rpi et de la liste LRPI de Rpj.

- Un msg du même type est également émis à tout dépendant direct Rpk de Rpi qui, dès réception élimine Rpi de LRPI de Rpk, puis diffuse le msg aux éventuels dépendants qui agiront de manière identique.

4- A chaque réception de msg proclamant un engagement, ou création de Rpi, la liste LRPI du Rp concerné (soit le précédent de Rpi s'il s'agit de création, ou bien le Rp auquel s'adresse le msg) est examinée, dans le cas où celle-ci est vide, ce Rp devient ainsi inaccessible.

IX.2.3. Détermination des Rp inaccessibles en stratégie C.

IX.2.3.1. Critères (1,2) satisfaits.

A la différences des stratégies précédentes et relativement à ce cas de critères satisfaits, la détermination des Rp inaccessibles s'avère plus compliquée en raison de la rétropropagation systématique imposée par la stratégie. $\forall$ Rpi Θ Pi, Rpi est

indispensable ssi une au moins des conditions suivantes est réalisée.

1- ¬ (Pi# RPi);

2- Il existe Rpj ϵ Pj tel que Rpj est RIPR (D17) de Rpi;

Autrement dit, un point de reprise Rpi est inaccessible, si aucune des conditions ci-dessus n'est satisfaite. Du point de vue pratique, il est nécessaire de connaitre tous les RIPR d'un point de reprise afin de pouvoir proclamer, le cas échéant, son inaccessibilité. On retrouve les deux méthodes décrites en stratégie A, à savoir:

1- Construction de la liste LRIPR au fur et à mesure de l'évolution des échanges.

- Lorsqu'un Rp est créé (soit Rpi), une liste LIRIPR lui est créée en même temps que les listes LID et LIP. Cette liste contient Rpi (étant actif, Rpi est son propre RIPR).

- Chaque fois que Rpi acquière un nouveau dépendant Rpj, celui-ci est inclu dans la LIRIPR de Rpi, puis un msg est émis à tous les propagateurs directs de Rpi, leur transmettant ce nouveau RIPR, qu'est Rpj.

- Lorsqu'un msg de ce type: Ajouter(Rpi, Rpj) est reçu par Pk, tel que Rpk --> Rpj, l'élément Rpj est ajouté à la liste LIRIPR de Rpk.

- Chaque fois que Rpi cesse d'être actif (Pi # Rpi), Rpi cesse également d'être RIPR. Pi doit proclamer cette cessation en émettant un msg à cet effet à tous les propagateurs directs de Rpi, puis supprime Rpi de sa liste LIRIPR.

- Dès réception d'un msg de ce type: CESSATION-RIPR(Rpi) par Pk, tel que Rpi était RIPR de Rpk, Pk élimine Rpi de la liste LIRIPR de Rpk, et ce msg est diffusé à tous les propagateurs directs éventuels de Rpk.

- Lorsqu'un Rpi n'a plus de RIPR, sa liste LIRIPR étant vide, il devient inaccessible, donc éliminable.

Remarque:

L'émission de msg proclamant la cessation de RIPR d'un Rp, peut être faite par un seul msg diffusé à l'intention de tous les processus interactifs. On limitera ainsi le nombre de msg à émettre, nécessaires à cet effet.

2- Les transmissions des listes LIRIPR d'un Rpi à ses propagateurs directs se fait après que celui-ci ait reçu un engagement.

- A la création d'un Rpi, une liste LIRIPR lui est créée en même temps que ses listes LID et LIP. Cette liste contiendra comme premier élément Rpi.

- Au fur et à mesure que Rpi acquière de nouveaux Rp dépendants directs, ces derniers sont inclus dans la LIRIPR de Rpi.

- Lorsque Rpj cesse d'être RIPR (n'est plus actif), Rpj est "marqué" comme tel, un msg proclamant cette cessation de RIPR et véhiculant la liste LIRIPR de Rpj, est émis à tout propagateur direct Rpk de Rpj.

- Dès réception d'un msg de ce type: CESSATION-RIPR(Rpj, LIRIPR) par Pk, (Rpk --> Rpj), celui-ci inclut la liste LIRIPR dans celle de Rpk et le msg est diffusé éventuellement aux propagateurs directs de Rpk qui agiront de la même façon.

- L'examen de la liste LIRIPR d'un point de reprise Rpi après avoir eu un engagement, permet de le déclarer inaccessible (éliminable) si tous les éléments de la liste LIRIPR (y compris lui même) sont "marqués" non-RIPR.

IX.2.3.2. Critères (1,2) non satisfaits.

Sachant que quelque soit le type de panne, les deux types de propagations sont systématiques, un Rpi est indispensable s'il existe au moins un point de reprise Rpj qui soit IPR (D18). C'est à dire:

∀ Rpi ∈ Pi, Rpi est indispensable si une au moins des conditions suivantes est satisfaite.

1- ¬ (Pi # Rpi).

2- Il existe Rpj ∈ Pj tel que Rpj est PIPR de Rpi.

3- Il existe Rpj ∈ Pj tel que Rpj est RIPR de Rpi.

Donc, un Rpi est inaccessible ssi:
- Il a reçu un engagement de Pi, et
- Il ne possède aucun IPR.

Application:

pour pouvoir déterminer quand un Rp est inaccessible, il faut d'abord connaître tous ses IPR, et quand ces derniers cessent d'être actifs. Pour ce faire, on peut adopter l'algorithme suivant:

- A tout Rpi créé, une liste LIPR destinée à contenir les IPR de Rpi est également créée en même temps que les listes LID et LIP. Cette liste va contenir en premier lieu l'élément Rpi.

- Chaque fois que Rpi acquière un nouveau dépendant ou propagateur direct Rpj, celui-ci est inséré dans la LIPR de Rpi, puis un msg de rajout d'IPR est émis à tout propagateur et dépendant autre que Rpj.

- Dès réception d'un msg de ce type: RAJOUT(Rpj) par Pk tel que Rpi --> Rpk et/ou Rpk --> Rpi, Pk insère Rpj dans la LIPR de Rpk, et transmet le msg de la même façon que Pi.

- Lorsqu'un point de reprise Rpi reçoit un engagement de Pi (Rpi cesse d'être IPR), ce dernier élimine Rpi de sa liste LIPR, puis proclame cet évènement à tous les dépendants et propagateurs directs de Rpi. Ces derniers agiront de la même façon que Pi.

- Aprés engagement de Pi envers Rpi, sa liste LIPR est examinée, si elle est vide, alors Rpi est inaccessible, d'où le contexte associé peut être éliminé.

IX.2.4. Détermination des Rp inaccessibles en stratégie D.

IX.2.4.1. Critères (1,2) satisfaits.

Apparemment, grâce à la conservation des msg évitant une rétropropagation, le maintient d'un Rp unique par processus se révèle comme étant une solution optimale au problème d'optimisation de l'espace mémoire. Cependant, l'absence d'une station de reprise, à l'occurrence d'une panne transitoire sur la station potentiellement défaillante associée, conduit à imposer une rétropropagation pour recréer les msg conservés en station de reprise, et perdus après panne permanente de celle-ci.

Ceci nous amène donc à reconsidérer le problème si l'on veut éviter cette situation, et augmenter d'avantage les capacités de la tolérance aux pannes.

La situation au niveau de la détermination des Rp inaccessibles est, dans pareil cas, similaire à celle de la stratégie A. Il suffit donc de se référer à cette stratégie pour d'avantage d'informations. A noter cependant, que lorsqu'un contexte est éliminé au niveau d'une SPD, un msg doit être émis à la SR correspondante pour agir de manière similaire en supprimant la copie du contexte éliminé et les msg conservés associés à la RRR correspondant au Rp supprimé.

IX.2.4.2. Critères (1,2) non satisfaits.

On retrouve de nouveau le cas de la stratégie A; un point de reprise Rpi (RPi ϵ Pi) est indispensable s'il satisfait au moins une des conditions suivantes:

1- Rpi est actif, i.e. ¬ (Pi#Rpi).
2- Il existe Rpj ϵ Pj telque Rpj est PIPR de Rpi.
3- Il existe Rpj ϵ Pj telque Rpi --> Rpj et ¬ (Pj#Rpj).

Les mêmes algorithmes que ceux décrits en stratégie A peuvent donc s'appliquer intégralement à la stratégie D.

IX.2.5. Détermination des Rp inaccessibles en stratégies mixtes.

IX.2.5.1. Stratégie AB.

IX.2.5.1.1. Critères (1,2) satisfaits.

Si les pannes transitoires ne posent aucun problème et permettent de maintenir un Rp unique par processus, les pannes permanentes nécessitent des algorithmes de détermination des Rp inaccessibles, propres à une stratégie donnée.

Tous types de pannes confondus, un point de reprise Rpi ∈ Pi est indispensable si une au moins des conditions suivantes est réalisée.

1- ¬ (Pi#Rpi);

2- Il existe Rpj ∈ Pj Rpi --> Rpj et ¬ (Pj#Rpj).

Tout Rpi ne vérifiant aucune de ces conditions peut être déclaré inaccessible.

Remarque:

Lorsqu'un Rp devient inaccessible, le contexte associé devient inutile, il y a lieu de l'éliminer. Cependant ce Rp (identificateur) et les msg reçus dans la région de reprise associée (et conservés) ne sont pas éliminés systématiquement, car leur présence peut être importante.

La situation en stratégie AB, relativement à l'inaccessibilité des Rp, notamment avec non postpropagation, est identique à celle des stratégies A et B. Aussi les algorithmes qui y sont développés restent valables pour la stratégie AB.

IX.2.5.1.2. Critères (1,2) non satisfaits.

Les conditions d'indispensabilité des Rp des processus sont identiques à celles définies pour la stratégie B. La détermination des Rp inaccessibles dans le cas présent, peut donc adopter le même algorithme.

IX.2.5.2. Stratégie AD.

Comme toutes les stratégies précédentes, l'optimisation du nombre de Rp à maintenir sauvegardés (par processus) peut varier selon qu'il y ait ou non postpropagation systématique.

IX.2.5.2.1. Critères (1,2) satisfaits.

Un point de reprise Rpi ∈ Pi est indispensable si la condition suivante est satisfaite.

¬ (Pi#Rpi), i.e. Rpi n'a pas encore reçu d'engagement de Pi.

Ainsi tout Rpi ayant reçu un engagement peut être déclaré inaccessible (en vertu de l'axiome précédent), ce qui signifie qu'un seul Rp peut être maintenu par processus (sauf bien entendu, le cas où juste avant un engagement, il peut en exister temporairement deux Rp).

Donc, dans ce cas de non postpropagation, l'algorithme de détermination de Rp inaccessible est considérablement simplifié, puisqu'il y a systématiquement suppression de Rpi, après établissement de son successeur Rpi+1 au sein du processus Pi.

IX.2.5.2.2. Critères (1,2) non satisfaits.

Ce cas de postpropagation systématique qui est peut annuler l'effet de la conservation des msg, nécessite le maintien d'un nombre de Rp conséquent. Rpi ∈ Pi est indispensable si une au moins des conditions suivantes est réalisée.

1- ¬ (Pi#Rpi);

2- Il existe Rpj ∈ Pj telque Rpj est PIPR de Rpi.

Donc, Rpi est inaccessible s'il ne satisfait aucune de ces conditions. Autrement dit, Rpi est inaccessible s'il n'est pas actif et s'il ne possède aucun PIPR. Déterminer l'inaccessibilité d'un Rp revient à exécuter un algorithme permettant d'affirmer que Rp est non actif et exempt de PIPR.

structure de données nécessaires.

Pour pouvoir appliquer un tel algorithme, il est nécessaire que tout point de reprise Rpi ∈ Pi ait une liste immédiate de tous les

Rp qui lui sont PIPR, (liste LIPIPR) localisée dans le graphe de reprise de Pi et repérée de la même façon que les listes LID et LIP.

Construction de la liste LPIPR.

* Chaque fois qu'un point de reprise Rpi est nouvellement créé, une liste LIPIPR(liste immédiate de PIPR) est aussi créée pour Rpi (voir fig 2b), elle contient comme premier élément Rpi lui-même.

* Chaque fois que Rpi acquière un nouveau propagateur Rpk, celui-ci est inséré dans la liste LIPIPR de Rpi, et Pi émet un msg de rajout de PIPR à tous les dépendants directs Rpj de Rpi en leur transmettant Rpk.

* La réception d'un tel msg au niveau de Pj donne suite à une action identique que celle entreprise par Pi.

Algorithme de détermination de Rp inaccessibles.

* Lorsqu'un processus Pi vient de terminer la création de Rpi+1, il accomplit un engagement envers Rpi (prédécesseur de Rpi+1) en éliminant Rpi de sa liste LIPIPR.

* Pi proclame cet engagement à tout dépendant direct Rpj de Rpi à l'aide d'un msg de la forme: ENGAGEMENT(Rpi).

* Tout Pj recevant un tel msg agit comme Pi, par élimination de Rpi de la liste LPIPR de Pj, puis diffuse le msg aux dépendants directs de Rpj.

* Chaque fois qu'une opération d'élimination de Rp ait lieu dans une liste LIPIPR, celle-ci est examinée, et si elle est vide alors le Rp associé est inaccessible.

Remarque:

Un cas trivial d'inaccessibilité de Rpi s'observe lorsque la région de reprise qui lui est associée n'est impliquée dans aucun échange d'information. Si telle est la situation après création de Rpi+1, Rpi est alors systématiquement éliminé de la liste LRp de Pi y compris son contexte.

IX.3. CONCLUSION.

la détemination des Rp inaccessibles au niveau de chaque stratégie permet, en plus de l'optimisation du nombre de contextes à maintenir, d'apprécier le paramètre "C4: Espace mémoire nécessaire" pour une comparaison de stratégies plus précise relativement à ce paramètre. Si en pannes transitoires, l'ensemble des stratégies (sauf C) avec postpropagation évitée, nécessite le maintien d'un Rp unique par processus (résultat optimal), l'adjonction des pannes permanentes modifie entièrement ce résultat excepté pour la stratégie AD où il demeure inchangé.

On admet certes, que la reconnaissance des Rp inaccessibles ne peut se faire sans contribuer à accroître l'overhead généré par l'implémentation d'une stratégie donnée, cependant cette contribution est moins importante que ne l'est l'espace mémoire à récupérer, notamment lorsque cette ressource fait défaut au système.

Quant à l'obligation à laquelle doivent se soumettre les processus récepteurs de msg d'information, de transmettre à leurs émetteurs les identificateurs des régions de reprise de réception (RRR), celle-ci peut être entièrement abolie dans le cas de "stratégies avec contrainte". Ce qui procure un avantage certain tant au niveau overhead (contribution moindre) qu'au niveau transmission (faible sensibilité à une surcharge).

Cependant, cette mesure ne peut s'appliquée qu'aux pannes transitoires dans le cas de "stratégies sans contraintes". Ce qui la réfute lorsque l'on considère une étude globale des deux types de pannes.

Chapitre X

CONCLUSION

L'étude que nous avons entreprise dans cette thèse, et qui concerne la tolérance aux pannes matérielles dans les systèmes informatiques répartis, a débouché sur une proposition d'un ensemble de stratégies de repise. Cet ensemble numérique est justifié par l'impossibilité de pouvoir prétendre à une stratégie "universelle", applicable à toute application donnée. C'est pourquoi, chaque stratégie proposée a ses propres caractéristiques et s'adapte mieux à un système donné en fonction des spécifications et exigences de ce dernier.

L'étude s'est bornée en première partie (partie I) à considérer les diverses stratégies dans un système distribués sans imposer aucune contrainte à la "communauté" de processus interactifs. Il a été montré dans ce contexte que les perturbations de l'environnement à l'occurrence de pannes peuvent être évitées partiellement par une réutilisation des msg conservés. L'élimination totale de ces dernières est obtenue on y respectant, de surcroît, certains critères définis à cet effet.
A noter que l'évitement de ces perturbations (ou effet-domino) est généralement atteint en limitant la liberté des processus au niveau de leurs communication ou de la création de leurs points de reprise. Cette liberté est largement maintenue dans l'etude relative à cette première partie.

Cependant, le contexte précédent relève d'une certaine particularité, puisqu'il suppose une latence d'erreur nulle, situation difficile à réaliser en pratique.

La seconde partie de notre étude (partie II) constitue une généralisation de la première, en introduisant une contrainte aux processus communicants, relativement à une coordination de la création des points de reprise. On évite ainsi toute situation de dégénérescence en effet-domino, et on optimise également l'espace mémoire réclamé par la sauvegarde des contextes des processus correspondant aux points de reprise. Cette situation nous soustrait de la détermination et l'élimination des points de reprise

sauvegardés inutilement comme cela est le cas en partie I, où des algorithmes traitant ce problème ont été proposés.

L'originalité de cette étude réside dans la reprise après pannes matérielles transitoires et surtout permanentes, car très peu de travaux ont été consacrés à ce sujet, notamment où la reprise est accomplie par des moyens logiciels.
Finalement, on a montré la faisabilité de mécanismes de reprise pouvant adopter les stratégies proposés. On aurait toutefois réalisé l'implémentation d'une de ces dernières pour valider les algorithmes proposés si les moyens matériels (disponibilité d'un réseau local) l'avaient permis.
Il serait cependant plus avantageux d'étudier la possibilité d'intégrer dans ces stratégies le traitement des fautes logiciels, pour répondre entièrement au besoin d'une reprise quelque soit le type de panne survenue. Ce travail peut être un prolongement de celui qui vient d'être réalisé.

ANNEXE.

Cette annexe regroupe la partie algorithmique de la thèse notamment les différentes procédures de reprise relatives à chaque stratégie présentée.

1. Algorithmes de la première partie.

1.1. Stratégie A.

1.1.1. Stratégie A sans postpropagation.

Algorithme unique pour les deux types de pannes.

```
PROCEDURE REPRISE ( RpR, Emet)
     % Emet: emetteur du msg de reprise %
     % RpR: point de relance extrait du msg %
Identificateur: RpR, Emet;
Debut
  Si msg-reçu = mon-msg  alors  % initiateur de reprise %
     Debut
              Restaurer état (RpR);
              Restaurer compteur-msg;
              Réexécution;
     fin
  Sinon  % invocation aléatoire %
      Debut
        Recherche dans list LD l'occurrence de RpR;
        Si recherche négative alors
             Sortie;
        Sinon % soit RpR' le propagateur direct de RpR trouvé %
           Debut
               Si RpR' non actif alors
                  Debut
                    Pour tout successeur Rp de RpR' faire
                      Emettre msg de màj à propagateurs de Rp;
                    finfaire
                    Détruire successeurs de RpR';
                  fin
               finsi
               Restaurer état (RpR');
               Restaurer  compteur-msg;
               Réexecution;
           fin
        finsi
      fin
  finsi
fin REPRISE.
```

1.1.2. Stratégie A avec postpropagation.

Algorithme unique pour les deux types de pannes. Les deux invocations (aléatoire et précise) sont employées.

```
PROCEDURE REPRISE (RpR,alea, emet)
    % ces paramètres sont extraits du msg de reprise %
Identificateur: RpR, emet;
Booleen alea, panne-transitoire;
Debut
   Si msg-reçu = mon-msg alors % initiateur de reprise %
         Si panne-transitoire alors
            REP(RpR);
         Sinon
            Debut
               Renommer RpR en RpR';
               Emettre msg de màj à station de reprise;
               Restaurer état (RpR);
               Réexécution;
            fin
         finsi
   Sinon  % processeur autre que l'initiateur %
       Si  non alea alors
           REP(RpR);
       Sinon
           Debut
               Détermination d'un eventuel dependant direct de
                     RpR (soit RpR1);
               Détermination d'un éventuel propagateur direct de
                     RpR (RpR2);
               Si RpR1 et RpR2 existent alors
                  Debut
                    RpR' = dominant (RpR1,RpR2);
                    REP(RpR');
                  fin
               Sinon
                  Si RpR1 ou RpR2 existe alors
                     Debut
                       RpR'= RpR1 ou RpR2;
                       REP(RpR');
                     fin
                  finsi
               finsi
         fin
```

```
        finsi
    finsi
fin REPRISE.
```

```
PROCEDURE REP(Rp)
    % cette procédure évite d'avoir à répéter plusieurs actions
    % identiques dans l'algorithme précédent.%
Identificateur: Rp;
Debut
  Pour tout élément rp ϵ LID(Rp) faire
       Emettre msg de reprise en rp (invocation précise);
  fin faire
  Détruire liste LID (Rp);
  Détruire tout successeur de Rp (s'il existe);
  Renommer Rp, soit Rp';
  Pour tout élément rp ϵ LIP(Rp) faire
       Emettre msg de màj à rp;
         % remplacer ancien dépendant par Rp'%
  fin faire
  Restaurer état(Rp');
  Relancer processus;
fin REP.
```

Remarque:

- Du point de vue implémentation, la procédure REPRISE étant ininterruptible, il serait adéquat qu'elle contienne la partie de la procédure REP.

- L'invocation aléatoire seule n'est pas suffisante à cause des retours de propagations vers l'initiateur de reprise de la panne permanente (absence du graphe de reprise).

1.1.3.

Algorithme de réalisation des critères de non postpropagation.

a) Critère 1: Le processeur récepteur reconnaît les msg émis inutilement.

Soient Ei l'émetteur du msg reçu, Ri le récepteur de ce msg.

```
PROCEDURE CRITERE1 (Table,Ei,n-msg-reçu)
   % n-msg-reçu: numéro du msg reçu %
Debut
    Si Ei n'est pas élément de Table alors
         Traiter msg-reçu; (1)
    Sinon
       Si n-msg-reçu ≤ n-dernier-msg-reçu (Ei) alors
             Sortie;   % msg inutile %
       Sinon
            Traiter msg-reçu; (2)
       finsi
    finsi
fin CRITERE1.
```

(1): L'action traiter msg-reçu consiste à mettre à jour la table en insérant le nouvel émetteur avec le n-msg-reçu, et mettre le msg lui-même en file d'attente.

(2): Consiste en la mise à jour de la table par le n-dernier-msg-reçu de Ei, et l'insérer dans la file d'attente des msg de Ri.

Remarque:

Cette procédure est appelée par la procédure de réception de msg, soit pour éliminer le msg reçu, ou le mettre en file d'attente.

b) Critère 2: Le processeur de reprise reconnaît les msg antérieurement émis (pannes transitoires), évite leur réémission aux destinataires sauf si ces derniers les réclament (rétropropagation).

```
Ext-Fonction CRITERE2 (Table,Ri,n-msg-a-émettre) %fonction booleenne%
Booléen repris-Ri %indique si Ri nécessite une réémission de msg%
                  % n-msg-à-émettre: numéro du msg à émettre%
Booleen critère2;
Debut
    Si Ri n'est pas élément de Table alors
        Debut
            Mise à jour de Table;
            CRITERE2 :=vrai;
        fin
    Sinon
        Si repris-Ri alors
            Si n-msg-à-émettre > n-dernier-msg-émis alors
                Mise à jour de Table;
            finsi
            CRITERE2 := vrai;
        Sinon
            Si n-msg-à-émettre ≤ n-dernier-msg-émis alors
                CRITERE2 := faux;
            Sinon
               Debut
                   Mise à jour de Table;
                   CRITERE2 := vrai;
               fin
            finsi
        finsi
    finsi
fin CRITERE2.
```

Remarque:

Cette fonction est appelée par la procédure d'émission de msg pour confirmer ou infirmer une émission. Le booléen repris-Ri est positionné par la procédure REPRISE qui reconnaît les processus réclamant les msg indispensables.

1.2. Stratégie B.

1.2.1. Critères (1,2) satisfaits.

Algorithme de reprise commun aux deux types de pannes.

```
PROCEDURE REPRISE (RpR, Emetteur)
  % RpR et identificateur de l'émetteur sont les paramètres de la
    procédure %
  % P: processus défaillant ayant RpR; Pr: processeur exécutant
    la procédure %
Identificateur: RpR, Emetteur;
Debut
     Si msg-reçu = mon-msg alors  % msg de l'initiateur de reprise%
          Debut
            Restaurer état(RpR);
            Exécuter P;
          fin
     Sinon    % C'est un MDR aléatoire%
        Debut
          Recherche d'un processus Pi ϵ Pr tel que, il existe
          Rpi ϵ Pi et Rpi--> RpR;
          Si Rpi existe alors
              Exécuter le renvoi;
          finsi
        fin
     finsi
Fin REPRISE.
```

Remarque:

- L'exécution du renvoi consiste, une fois RpR est détecté (RpR appartient à la liste LD d'un processus de Pr), à accéder à la liste des msg conservés, et à entamer la réémission en tenant à jour un compteur de msg réémis pour éviter les perturbations causées par une éventuelle panne transitoire.

- On suppose dans cet algorithme, comme d'ailleurs dans tous les autres, que les stations se connaissent mutuellement, ou qu'une diffusion de msg est possible.

- Il serait fort avantageux qu'un Rp puisse aussi identifier son processus (simplicité).

1.2.2. Critères (1,2) non satisfaits.

Algorithme commun aux deux catégories de pannes.

```
PROCEDURE REPRISE (RpR', RpR, Emetteur)
    % RpR' nouvelle identification de RpR %
    % RpR étant le Rp de relance choisi par%
    % le processeur de reprise%
Identificateur: RpR', RpR, Emetteur;
Variable logique: panne-transitoire;

Debut
    Si msg-reçu = mon-msg alors  % initiateur de reprise )
       Si panne-transitoire alors
             Debut
                 REP(RpR');
                 Restaurer état(RpR');
                 Réexécution;
             fin
       Sinon
           Debut
               Restaurer état(RpR');
               Réexécution;
           fin
       finsi
    Sinon
        Debut
            Recherche d'un dépendant direct de RpR (soit Rpj);
            Recherche d'un propagateur direct de RpR (soit Rpk);
            Si Rpj et Rpk existent alors
               Si Rpj est dominant alors
                     Debut
                         REP(Rpj);
                         Restaurer état(Rpj);
                         Réexécution;
                     fin
                Sinon
                    Debut
                        Renvoil (RpR,Rpk);
                        REP(Rpj);
                        Restaurer état(Rpj);
                        Réexécution;
                    fin
```

```
            finsi
        Sinon
            Si Rpk existe alors
                    Renvoi2 (RpR);
            Sinon
                Debut
                    REP(Rpj);
                    Restaurer état(Rpj);
                    Réexécution;
                fin
            finsi
        finsi
    finsi
Fin REPRISE.
```

```
PROCEDURE REP(Rp)

Debut
    Pour-tout dépendant direct Rpi de Rp faire
        Emettre msg de reprise (avec Rpi);
    fin-faire
    Pour-tout propagateur direct Rpj de Rp faire
        Emettre msg de renvoi (Rp);
    fin-faire
Fin REP.
```

Remarque:

- La variable logique panne-transitoire est positionnée (vrai) lors du premier essai pour tenter de recouvrer une hypothétique panne transitoire. Elle sera repositionnée (faux) lorsque les essais tentés restent vains, et sa valeur sera communiquée au processeur de reprise (panne permanente).

- Le renvoi de msg indispensables concerne deux cas possibles. Renvoi1 (RpR,Rpk) signifie: renvoi de tous les msg émis dans la RRE correspondant à Rpk et reçus dans la RRR associée à RpR , et eux seulement (exemple Renvoi1 (B1,A1), fig 3c). Renvoi2 (RpR) signifie: renvoi des msg reçus dans la RRR identifiée par RpR; aucune contrainte sur la (ou les) RRE n'est exigée (exemple Renvoi2 (B1),fig 3a).

1.3. Stratégie C.

1.3.1. Algorithme de reprise en panne transitoire.

Le processeur de reprise, après détection de la panne, émet un msg d'invocation aléatoire à tous ses partenaires, y compris lui-même.

```
PROCEDURE REPRISE (RpR)
  % RpR: est le Rp de relance du processus à reprendre %
Identificateur RpR;
Variable: Rp;
Rp:=nil;

Debut
    Si msg-reçu ≠ mon-msg  alors % partenaire du processus %
                                  %défaillant %
         Debut
           Recherche dans la liste LD de l'occurrence de RpR;
              Si recherche positive alors
                 soit Rpi le Rp associé;
              finsi
           Recherche dans la liste LP de l'occurrence de RpR;
              Si recherche positive alors
                 soit Rpj le Rp associé;
              finsi
           Si Rpi et Rpj existent alors
                 Rp:= Dominant (RPi;Rpj);
           sinon
              Si Rpi ou Rpj existe alors
                  Rp:= Rpi ou Rpj;
              finsi
           finsi
         fin
    sinon            % initiateur de reprise %
       Rp:= RpR;
    finsi
   Debut
       Renommer contenu de Rp;
       Détruire successeur du contenu de Rp;
       Détruire listes LIP et LID du contenu de Rp;
       Restaurer état (contenu de Rp);
       Réexécution;
   fin
Fin REPRISE.
```

1.3.2. Algorithme de reprise commun aux deux types de pannes.

Principe de recherche de la RL: roll-backs progressifs.

```
PROCEDURE REPRISE (RpR)
  % RpR: point de relance précisé dans le msg d'invocation %
Variable: Rp;
Identificateur RpR;

Debut
     Si msg-reçu = mon-msg alors  % msg reçu = msg émis par%
                                  % l'initiateur de reprise%
        Debut
            Renommer RpR;
            Détruire successeurs(RpR);
            Détruire listes LID et LIP de RpR;
            Restaurer état(RpR);
            Lancer exécution;
        fin
     Sinon      % le msg spécifie une propagation de reprise%
         Debut
            Recherche dans LD de l'occurrence de RpR;
              Si recherche positive alors
                 soit Rpi le Rp associé à RpR; % Rpi--> RpR%
                 Eliminer occurrence de RpR;
             finsi
            Recherche dans LP de l'occurrence de RpR;
              Si recherche positive alors
                 soit Rpj le Rp associé;  %RpR--> Rpj%
                 Eliminer l'occurrence de RpR;
              finsi
            Si Rpi et Rpj existent alors
                   Rp:= Dominant (Rpi,Rpj);
            Sinon
                Si Rpi n'existe pas alors
                       Rp:= Rpj;
                Sinon
                       Rp:= Rpi;
                finsi
            finsi
            Si Rp ≠ nil  alors
               Debut
```

```
                    Renommer contenu de Rp;
                    Détruire successeurs du contenu de Rp;
                    Emettre msg d'invocation précise à tout
                         dépendant du contenu de Rp;
                    Détruire liste LID du contenu de Rp;
                    Emettre msg d'invocation précise à tout
                         propagateur du contenu de Rp;
                    Détruire liste LIP du contenu de Rp;
                    Restaurer état du contenu de Rp;
                    Relancer l'exécution;
                fin
            finsi
        fin
    finsi
fin REPRISE.
```

Remarque:

Dans l'algorithme, on a distingué les actions exécutées par les initiateurs de reprise, qu'ils soient en panne transitoire ou panne permanente, des autres processus gagnés par la propagation. Pour pouvoir exécuter la même séquence d'opération par les deux catégories d'initiateurs, l'invocation aléatoire s'impose.

Il est cependant possible de se soustraire à cette attitude en distingant l'initiateur de reprise de la panne permanente, de ceux des pannes transitoires et ce, en identifiant le type de panne traitée. Cette identification peut être véhiculée dans le msg de reprise et précisera particulièrement s'il s'agit d'une invocation aléatoire ou précise.

Il serait avantageux (voire indispensable) de réaliser un ordre totale parmi les Rp d'un même processus afin de pouvoir appliquer la relation de dominance (D4). Il serait également préférable que l'identification d'un Rp puisse permettre de reconnaître le processus auquel il appartient.

1.4. Stratégie D.

1.4.1. Critères (1,2) satisfaits.

Algorithme commun aux deux types de pannes.

```
PROCEDURE REPRISE (Rp, PTRANS)
   % Rp: point de relance, PTRANS: indique le type de panne%
Identificateur: Rp; Booleen: PTRANS;

Debut
    Si PTRANS alors     % c'est une panne transitoire %
          Debut
               INIT (Renvoi,SD,SR,RRE,NMSG);  (1)
                  % SD: station défaillante, SR: station de reprise%
                  % RRR et NMSG déterminent à partir de quel msg
                  % sera effectué le renvoi%
               Restauration état (Rp);
               Relance de l'exécution;
          fin
    Sinon          % Panne permanente %
        Debut
            Transférer les msg identifiés par Rp dans
                        la file d'attente ; (2)
            Restauration état (Rp);
            Relance de l'exécution;
        fin
    finsi
Fin REPRISE.
```

(1): La primitive INIT a pour objet d'initialiser la procédure de Renvoi localisée sur la SR, en lui communiquant les paramètres SD,et une identification (RRE,NMSG) à partir de laquelle les msg seront renvoyés.

(2): Dans le cas d'une panne permanente, il n'y a pas de renvoi de msg, Le processeur de reprise transférera dans la file d'attente de msg du processus défaillant,les msg associés à Rp (fig 2). le booléen PTRANS permet d'indiquer, si la panne est

transitoire (vrai) ou permanente afin d'invoquer ou non la procédure de renvoi de msg conservés.

1.4.2. Critères (1,2) non satisfaits.

Algorithme commun aux deux types de pannes.

Dans cet algorithme, inévitablement les deux types d'invocation seront utilisés. Invocation précise ou aléatoire pour tout processus non affecté par une panne permanente, invocation aléatoire impérative initiée par le processeur de reprise traitant le processus défaillant. Il est possible pour tout processeur d'inclure dans le msg de reprise qu'il émet, le type d'invocation auquel il fait référence. Ainsi tout processus récepteur d'un tel msg saura, s'il doit reprendre au RpR spécifié dans le msg, ou bien déterminer si un de ses Rp est lié à ce dernier.

Les mécanismes de recupération de msg indispensables au processus défaillant n'étant pas identiques dans les deux types de pannes, il y a lieu de pouvoir identifier chacun de ces types afin de leur appliquer le mécanisme approprié.

```
PROCEDURE REPRISE (RpR,Aleatoire,Rp)
Identificateur: RpR,Rp, Prp;
Booleen: Aleatoire;        % indique le type d'invocation %
Booleen: Panperm;          % indique le type de panne     %

Debut
    Si  msg-reçu = mon-msg alors    % initiateur de reprise %
        Si Panperm  alors
           Debut        % traiter panne permanente %
               Transferer msg conservés identifiés par RpR
                 dans la file d'attente contenant RpR;
               Renommer RpR en Rp;
               Restaurer état (Rp);
               Réexécution;
           fin
        Sinon           % traiter panne transitoire %
           Debut
               Invocation aléatoire (RpR,Aleatoire,Rp);
               Pour tout successeurs Rpj de RpR faire
                  Detruire Rpj;
               fin faire
               Renommer RpR en Rp;
               INIT (Renvoi,SD,SR,RpR,Rp,NMSG);
               Restaurer état (Rp);
               Relancer exécution;
           fin
        finsi
    Sinon          % cas de processus atteint par la propagation%
        Si Aleatoire alors
           Debut
               Recherche d'un Rpi dépendant direct de RpR;
               Recherche d'un Rpj propagateur direct de RpR;
               Si  Rpi et Rpj existent alors
                   Si Rpj est dominant de Rpi alors
                      Mise à jour de RpR (remplacer par Rp);
                   finsi
                   REP (Rpi,Rp);
               Sinon
                   Si  Rpi trouvé alors
```

```
                        REP (Rpi,Rp);
                    Sinon
                        Mise à jour de RpR;
                    finsi
                finsi
            fin          % invocation précise%
              REP (RpR,Rp);
        finsi
   finsi
Fin REPRISE.

PROCEDURE REP (Rp1,Rp2)
  % Cette procédure a pour effet de regrouper toutes actions %
  % identiques dans la procédure REPRISE précédente %
Identificateur: Rp1,Rp2;

Debut
    Renommer Rp1 par Rp2;
    Pour tout élément Rp de LID (Rp2) faire
         Invocation précise (Rp);
    fin faire
    Pour tout élément Rp de LIP (Rp2) ¬ ∈ LID(Rp2) faire
          Emettre msg de mise à jour (Rp1,Rp2);
    fin faire
    Pour tout successeur Rpi de Rp2 faire
          Détruire Rpi;
    fin faire
    Detruire liste LID (Rp2);
    INIT (Renvoi,SD,SR,Rp1,Rp2,NMSG);
    Restaurer état(Rp2);
    Réexécution;
Fin REP.
```

1.5. Stratégie AB.

1.5.1. Critères (1,2) satisfaits.

```
PROCEDURE REPRISE (RpR,TRANS)
Identificateur: RpR; % Rp de relance fourni par le processeur %
Booleen: TRANS;      % booleen positionné par le processeur pour%
                     % indiquer le type de panne %
Debut
     Si TRANS alors       % panne transitoire %
        Debut
            Preparer réutilisation des msg; (*)
            Restaurer état(RpR);
            Relance;
        fin
     Sinon                % reprise d'une panne permanente %
        Debut
            Renvoi (Emetteur, RpR, Diffusion);
                    % émission d'un msg de renvoi aléatoire %
            Restaurer état(RpR);
            Relance;
        fin
     finsi
Fin REPRISE.
```

(*): En général lorsqu'un processus exécute une opération de réception de msg (RECEIVE), le processeur scrute la file d'attente de msg reçus (gérée en FIFO) et si un msg est disponible, il est alors livré au processus. Dans notre cas, les msg à réutiliser sont localisés dans une liste accessible à travers RpR. Cette liste peut être considérée comme file d'attente provisoire et particulière, ou que ces msg seront transférés dans la file d'attente du processus concerné (défaillant).

1.5.2. Critères (1,2) non satisfaits.

```
PROCEDURE REPRISE (RpR, Alea, RpR')
   % RpR': nouvel identificateur de RpR éventuellement vide %
Identificateur: RpR, RpR';
Booleen: Alea, panperm;
   % Alea: positionné par le processeur en fonction de %
   % l'information%
   % récupérée du msg de reprise %

Debut
    Si msg-reçu = mon-msg  alors
        Si panperm  alors % cas de reprise d'une panne permanente%
           debut
               Renommer RpR par RpR';
               Restaurer état(RpR');
               Réexécution;
           fin
        Sinon    %cas de reprise d'une panne transitoire %
            REP(RpR, RpR');
        finsi
    Sinon         % processus autres que l'initiateur%
        Si Alea  alors   % invocation aléatoire%
           debut
             Détermination d'un éventuel dépendant direct de RpR
                      (soit Rp1 le résultat);
             Détermination d'un éventuel propagateur direct de RpR
                      (soit Rp2 le résultat)
             Si Rp1 et Rp2 existent alors
                   Si Rp2 est dominant alors
                       Mise à jour de LID(Rp2) par RpR';
                   finsi
                   REP(Rp1, Rp12) ;
                     % Rp12 est le nouvel identificateur de Rp1%
             Sinon
                   Si Rp1 ou Rp2 existe alors
                        Si Rp2 existe alors
                            Mise à jour de LID(Rp2) par RpR2;
                        Sinon
                            REP(Rp1, Rp12);
                        finsi
```

```
                    finsi
                finsi
              fin
          Sinon            % invovation précise%
              REP(RpR, RpR1) % RpR1: nouvel identificateur de RpR%
          finsi
      finsi
Fin REPRISE.


PROCEDURE REP(Rp1, Rp2)
Identificateur: Rp1, Rp2;

Debut
     Pour tout élément Rpi ϵ LID(Rp1) faire
          Emettre msg invovation précise;
     fin faire
     Pour tout élément Rpj ϵ LIP(Rp1) et non à LID(Rp1) faire
          Emettre msg de mise à jour (Rp1,Rp2);
     fin faire
     Renommer Rp1 par Rp2;
     Detruire successeurs(Rp2) et LID (Rp2);
     Restaurer état(Rp2);
     Relance;
Fin REP.
```

Commentaire:

La procédure REP doit son existence à l'évitement d'une redondance d'actions identiques dans la procédure REPRISE. Quant à la procédure REPRISE, elle est appelée par le processeur qui reçoit un msg d'invocation (y compris l'initiateur lui-même). Les paramètres qui lui sont communiqués sont extraits du msg de reprise. Certaines informations sont locales au processeur telles: la variable logique panperm positinnée par le processeur avant l'appel de REPRISE. La procédure REP accède au graphe de reprise commun.

1.6. Stratégie AD.

1.6.1. Critères (1,2) satisfaits.

```
PROCEDURE REPRISE (RpR,TRANS)
Identificateur: RpR;
Booleen: TRANS;     % indique le type de panne %

Debut
    Si TRANS alors    % la panne est transitoire %
            Debut
                Transfert des msg à réutiliser en file d'attente
                   du processus à reprendre;  (*)
                Restauration de l'état du processus en RpR;
                Réexecution;
            fin
    Sinon             %panne permanente %
            Debut
                Emission d'un msg diffusé pour déclencher la
                     validation du critère 1 au niveau des
                          dépendants du défaillant; (*)
                Préparer msg à réutiliser;
                Restauration de l'état du processus en RpR;
                Relance;
            fin
    finsi
Fin REPRISE.
```

(*): Des précautions doivent être prises lors de l'utilisation des msg, de sorte à éviter une interférence avec ceux qui arrivent nouvellement dans la file d'attente.

1.6.2. Critères (1,2) non satisfaits.

Soit P le processus impliqué dans l'opération de reprise.

```
PROCEDURE REPRISE (RpR,Rp, Alea)
Identificateur: RpR, Rp;
Booleen: Panperm, Alea;

Debut
    Si msg-reçu = mon-msg alors  %initiateur de reprise%
        Si panperm alors  %initiateur traite une panne permanente%
            Debut
                Préparer réutilisation des msg conservés;
                Renommer RpR par Rp;
                Restaurer état(Rp);
                Réexécution;
            fin
        Sinon       % initiateur traite une panne transitoire%
            REP(Rp);
        finsi
    Sinon           % processeur autre que l'initiateur%
        Si Alea alors  % invocation aléatoire%
            Debut
                Recherche d'un dépendant Rpi de RpR;
                Recherche d'un propagateur Rpj de RpR;
                Si Rpi et Rpj existent alors
                        Si Rpj est dominant alors
                           Mise à jour de RpR par Rp dans LID(Rpj);
                        finsi
                        REP(Rpi);
                Sinon
                        Si Rpj existe alors
                           Mise à jour de RpR dans LID(Rpj) par Rp;
                        Sinon
                           REP(Rpi);
                        finsi
                finsi
            fin
        finsi
    finsi
Fin REPRISE.
```

```
PROCEDURE REP(Rp)
Identificateur: Rp;

Debut
    Pour tout élément Rpi ∈ LID(Rp) faire
        Emettre msg invocation précise à Rpi;
    fin faire
    Renommer Rp (soit Rp');
    Pour tout élément Rpj ∈ LIP(Rp') non à LID(Rp') faire
        Emettre msg mise à jour (Rp,Rp');
    fin faire
    Détruire successeurs(Rp') et LID(Rp');
    Restaurer état(Rp');
    Réexécution;
fin
Fin REP.
```

Remarque:

Lorqu'un msg d'invocation est reçu, le processeur récepteur y extrait les paramètres nécessaires (RpR, Rp, Alea) qu'il communique à la procédure REPRISE.
Panperm: variable logique globale qui n'est utilisée que par les initiateurs de reprise.
Les procédures REPRISE et REP utilisent une structure commune: le graphe de reprise.

2. Algorithmes de la deuxième partie.

2.1. Stratégie A.

Algorithmes de reprise commun aux deux types de pannes.

2.1.1. Critères (1,2) non satisfaits (latence non nulle).

Après détection, diagnostic et reconfiguration appropriée éventuelle, le processeur de reprise (celui de la station défaillante s'il s'agit de panne transitoire, ou celui de la station de reprise en cas de panne permanente) émet un msg d'invocation aléatoire ou précise, selon qu'il s'agisse d'une panne permanente ou transitoire, aux stations fonctionnelles concernées (à toutes les stations en cas d'invocation aléatoire). Après réception d'un tel msg, tout processeur exécute la procédure suivante.

```
PROCEDURE REPRISE (RpR)
     % RpR: identificateur d'un Rp de relance %
Identificateur: RpR;
Debut
    Si invocation-aléatoire alors
          debut
              Si RpR = mon-RpR alors     % initiateur lui-même%
                 RELANCE1 (RpR);
              Sinon
                 Existe-t-il Rpj ∈ Pj tel que
                   Rpj --> RpR ou RpR --> Rpj;
                 Si Rpj trouvé  alors
                    RELANCE1 (Rpj);
                 finsi
              finsi
          fin
    Sinon
         RELANCE1 (RpR);
    finsi
Fin REPRISE.
```

```
PROCEDURE RELANCE1 (Rp)
   % Rp: point de relance %
Identificateur: Rp;
Debut
    Pour tout élément Rpi ∈ LID(Rp) faire
        Emettre msg de reprise à Pi (Rpi ∈ Pi);
    fin faire
    Restaurer état(Rp);
    Exécution de P;
fin
Fin RELANCE1.
```

2.1.2. Critères (1,2) satisfaits (latence nulle).

```
PROCEDURE REPRISE (RpR)
Identificateur: RpR;   % RpR point de relance %
Debut
    Si  invocation- aléatoire alors
        Debut
            Si RpR = mon-RpR alors
               RELANCE2 (RpR);
            Sinon
                Existe-t-il Rpj ∈ Pj tel que Rpj --> RpR;
                Si Rpj trouvé alors
                   RELANCE2 (Rpj);
                finsi
            finsi
        fin
    Sinon
        RELANCE2 (RpR);
    finsi
Fin REPRISE.


PROCEDURE RELANCE2 (Rp)
Identificateur: Rp;
Debut
    Restaurer état(Rp);
    Réexécution;
fin
Fin RELANCE2.
```

Remarque:

Etant donné que les deux types d'invocation peuvent être utilisés simultanément au cours d'une même opération de reprise, il importe de pouvoir faire la distinction, car les traitements respectifs sont différents (par exemple adjoindre à l'identificateur du RpR figurant dans le msg, l'identificateur du processus auquel il appartient).

2.2. Stratégie B.

2.2.1. Cas de latence nulle.

Algorithme commun aux deux types de pannes.

Aprés diagnostic et reconfiguration éventuelle (panne permanente), le processeur de reprise Pri devant reprendre Pi en Rpi (Rpi actif), émet un msg de renvoi aléatoire à toutes les stations y compris celle de Pri. A la réception d'un tel msg, chaque processeur récepteur Prj exécute l'algorithme suivant.

```
PROCEDURE RENVOI (Rpi)
Identificateur: Rpi;

Debut
    Si msg-reçu = mon-msg alors
            Debut
                Restaurer état(Rpi);
                Relancer exécution;
            fin
    Sinon
        Debut
            Si il existe Rpj ϵ Pj tel que Rpj --> Rpi alors
                    REEMET (Rpi);
            finsi
        fin
    finsi
fin
Fin RENVOI.
```

La procédure RENVOI est identique pour tout station/processus. La procédure REEMET se charge de réémettre les msg indispensables au demandeurs tout en étant asservie aux capacités de réception de ce dernier.

2.2.2. Cas de latence non nulle.

Algorithme commun aux deux types de pannes et identique pour tout station/processus.

```
PROCEDURE REPRISE (Rpi)
Identificateur: Rpi;

Debut
    Si msg-reçu = mon-msg alors
       Debut
          Restaurer état(Rpi);
          Relancer exécution;
       fin
    Sinon
        Si il existe Rpj ∈ Pj tel que Rpj --> Rpi et
              non (Rpi --> Rpj)  alors
           REEMET (Rpi);
        Sinon
            Si il existe Rpj ∈ Pj tel que Rpi --> Rpj
                  ou bien Rpi --> Rpj et Rpj --> Rpi alors
                Debut
                   Pour tout Rpk ∈ LID(Rpj) faire
                      Provoquer la reprise de Pk;
                   fin faire
                   Restaurer état(Rpj);
                   Relancer exécution;
                fin
            finsi
        finsi
    finsi
fin
Fin REPRISE.
```

2.3. Stratégie C.

2.3.1. Cas de latence nulle.

Algorithme commun aux deux types de pannes. L'algorithme suivant est exécuté par tout processeur qui reçoit un msg d'invocation de reprise (y compris le processeur initiateur de reprise).

```
PROCEDURE REPRISE (Rp)
     % Rp: identificateur d'un point de reprise%
Identificateur: Rp;

Debut
    Si invocation-aléatoire  alors
         Si msg-reçu = mon-msg  alors
            Debut
                Restaurer état(Rp);
                Relancer exécution;
            fin
         Sinon
             Si il existe Rpj ∈ Pj tel que Rpj --> Rp  alors
                Debut
                  Pour tout élément Rpk ∈ LIP(Rpj) et ≠ Rp faire
                       Emettre msg de reprise;
                  fin faire
                  Restaurer état(Rpj);
                  Relancer exécution;
                fin
             finsi
         finsi
    sinon
         Debut
             Pour tout élément Rpk ∈ LIP(Rp) et non
                       à l'émtteur de l'invocation  faire
                  Emettre msg de reprise (Rpk);
             fin faire
             Restaurer état(Rp);
             Relancer exécution;
         fin
    finsi
fin
Fin REPRISE.
```

2.3.2. Cas de latence non nulle.

Algorithme traitant conjointivement les deux types de pannes et commun à toute station/processus.

```
PROCEDURE REPRISE(Rp)
     %Rp: identificateur d'un point de reprise%
Identificateur: Rp;

Debut
     Si invocation-aléatoire alors
          Si msg-reçu = mon-msg alors   %initiateur de reprise%
               Debut
                    Restaurer état(Rp);
                    relancer exécution;
               fin
          Sinon
               Si il existe Pj tel que Rpj ∈ Pj et Rpj --> Rp
                     ou (inclusif) Rp --> Rpj  alors
                  Debut   % propagation de reprise%
                       Pour tout Rpk ∈ LIP(Rpj) et ≠ Rp faire
                            Emettre msg de reprise (Rpk);
                       fin faire
                       Pour tout Rpl ∈ LID(Rpj) non à LIP(Rpj) et
                                     ≠ Rp faire
                            Emettre msg de reprise (Rpl);
                       fin faire
                       Restaurer état(Rpj);
                       Relancer exécution;
                  fin
               finsi
          finsi
     sinon
          Debut
               Pour tout Rpk ∈ LIP(Rp) et non à émetteur faire
                    Emettre msg de reprise (Rpk);
               fin faire
               Pour tout Rpl ∈ LID(Rp) et non à LIP(Rp) faire
                    Emettre msg de reprise (Rpl);
               fin faire
               Restaurer état(Rp);
               Relancer exécution;
          fin
     finsi
```

```
fin
Fin REPRISE.
```

2.4. Stratégie D.

2.4.1. Cas de latence nulle.

Algorithme unique pour tout station/processus et pour les deux types de pannes matérielles.

```
PROCEDURE REPRISE (Rp,Trans)
   % Rp est le point de relance désigné dans le msg de reprise %
Identificateur: Rp;
Booleen: Trans    % Trans indique le type de panne à récupérer%

Debut
    Si Trans   alors   % panne transitoire %
       Debut
           Emettre msg de renvoi(Rp);
           Restaurer état(Rp);
           Relancer exécution;
       fin
    Sinon    % Rp est une copie sur station de reprise%
             % panne permanente  %
        Debut
            Restaurer état(Rp);
            Relancer exécution;
        fin
    finsi
fin
Fin REPRISE.
```

Remarque:

La distinction entre les deux types de pannes étant difficile à faire, alors dès sa détection, une panne est considérée transitoire et traitée comme telle. Si après plusieurs tentatives de reprise, le résultat reste vain, la panne devient permanente et signalée à la station de reprise associée. Dans le cas d'une panne du processeur telle que la détection soit effectuée par les stations partenaires, l'initiative de reprise revient à la station de reprise correspondante.

La distinction du type de panne au niveau de l'Algorithme peut se faire par l'intermédiaire des paramètres fournis au moment de l'appel de la procédure REPRISE. Ainsi, le paramètre Trans est positionné par la station défaillante lorsque la panne est considérée transitoire, et par la station de reprise dans le cas contraire.

2.4.2. Cas de latence non nulle.

Algorithme de reprise commun aux deux types de pannes.

```
PROCEDURE REPRISE (Rp,Trans)
     % procédure exécutée à chaque réception de msg de reprise%
Identificateur: Rp;  % Rp: point de relance figurant dans le msg%
Booleen : Trans      % Trans: indique le type de panne %

Debut
    Si msg-reçu = mon-msg alors    % cas de station défaillante ou
                                   % de reprise (initiateur) %
       Debut
           Si Trans alors          % panne transitoire %
              ACTION (Rp);  % exécution des actions de reprise %
           Sinon              %panne permanente  %
              Debut
                  Restaurer état(Rp);
                  Relancer exécution;
              fin
           finsi
       fin
    Sinon     % le msg de reprise est reçu par une station autre
              % que la station défaillante ou la station de reprise%
        Si il existe Pi telque Rpi ∈ Pi et Rp --> Rpi  alors
              ACTION (Rpi);
        finsi
    finsi
fin
Fin REPRISE.
```

```
PROCEDURE ACTION (Rp)

Debut
    Emettre msg de renvoi (Rp);
    pour tout élément ∈ LID(Rp) faire
        Emettre msg de reprise;
    fin faire
    Restaurer état(Rp);
    Relancer exécution;
fin
Fin ACTION.
```

REFERENCES

[ALI 85] ALIOUAT M., COURTOIS B. "Recovery processes from Transient or permanent error in Distributed Environment" soumis à publication.

[AND 76] ANDERSON T., KERR R. "Recovery blocks in action" Proc, 2nd Int'l conf. on software Engineering, San Francisco, october 1976 pp 447-457.

[AND 81] ANDERSON T., LEE P.A. "Fault tolerance principles and practice" Prentice Hall 1981.

[AND 79] ANDREWS D.M. "Using executable assertions for testing and fault tolerance," FTCS-9 Digest of papers, pp. 102-105 (june 1979).

[AND 83] ANDRE F., HERMAN D., VERJUS J.-P. "Synchronisation de programmes parallèles." Dunod ed. 1983.

[ANT 84] ANTOLA A., SCARABOTOLLO N. "Backward error recovery in concurrent environment," Advances in microprocessing and microprogramming, EUROMICRO 1984, pp. 45-53.

[AVI 76] AVIZIENIS A. "Fault-tolerant Systems," IEEE Transactions on Computers, Vol.C-25, No.12, December 1976.

[BAL 84] BALTER R., DECITRE P. "Validation et reprise dans le système transactionnel coopérant SCOT," TSI. Technique et Science Informatique, vol.3, No.2, 1984.

[BAR 83] BARIGAZZI G. "Application transparent setting of recovery points" Fourth Symposium on Reliability in Distributed Software and Databases, pp. - , october 1984.

[BAR 82] BARIGAZZI G., CIUFFOLETTI A., STRIGINI L. "Reconfiguration procedure in a distributed multiprocesseur system," Digest of papers, FTCS-12, pp. 73-76, june 1982.

[BES 81] BEST E., CRISTIAN F. "Systematic Detection of Exception Occurrences," Science of Computer Programming, Vol.1, No.1, North Holland Pub. Co., pp. 115-144.

[BES 80] BEST E. "Atomicity of Activities," pp.225-250 in Lecture Notes in Computier Science 84, ed. W. Brauer, Springer-Verlag, Berlin 1980.

[BRI 84] BRIATICO D., CIUFFOLETTI A., SIMONCINI L. "A Distributed Domino-Effect Free Recovery Algorithm," 4th Symposium on Reliability in Distributed Software and Databases, Maryland, October 1984.

[BRI 73] BRINCH H. "Operating System Principles," Prentice-Hall, Englewood Cliff, New-Jersey (1973).

[CAM 79] CAMPBELL R.H, HORTON K.H., BELFORD G.G. "Simulations of a Fault-Tolerant Deadline Mechanism," Digest of Papers FTCS-9 9th Annual International Symposium on Fault-Tolerant Computing Madison (WI), pp.95-101 (June 1979).

[COR 81] CORNAFION (nom collectif). "Systèmes informatiques répartis: Concepts et Techniques." Dunod ed. 1981.

[COU 81] COURTOIS B. "Test et LSI," Thèse es-sciences INPG. 1981.

[CRI 79] CRISTIAN F. "Traitement des exceptions dans les programmes modulaires" Doct. Thesis University of Grenoble 1979.

[CRI 80] CRISTIAN F. "Exception handling and software fault tolerance," Digest of papers FTCS-10 10th Intl. symposium on fault-tolerant computing systems, Kyoto, pp. 97-103 (october 1980).

[CRI 82] CRISTIAN F. "Exception handling and software fault tolerance" IEEE trans. on computers, C. 31 (1982) pp 531-539.

[CRO 77] CROCUS (nom collectif). "Systèmes d'Exploitation des Ordinateurs," Dunod Informatique, 2e éd. (Décembre 1977).

[DSA 80] DSA: "Architecture des Systèmes Distribués: Description Générale," Cii Honeywell Bull, Référence 15-F8-8246, 1980.

[FIS 83] FISHER M.J., GRIFFETH N.D., LYNCH N.A. "Global States of a Distributed System," IEEE Transactions on Software Engineering, vol.SE-8, May 1983, pp.198-202.

[GEL 78] GELENBE E., DEROCHETTE D. "Performance of rollback recovery systems under intermittent failures," CACM 21(6), pp. 493-499 (june 1978).

[GEL 79] GELENBE E. "On the optimum checkpoint interval," JACM 26(2), pp. 259-270 (april 1979).

[GOO 75] GOODENOUGH J. "Exception handling-issues and proposed notation" CACM, 18(12), pp. 683-696 (december 1975).

[GRE 85] GREGORY S.T., KNIGHT J.C. "A new approach to backward error recovery" FTCS-15, Digest of papers pp.404-409, (june 1985).

[GUT 80] GUTTAG J.V., HORNING J.J "Formal Specification as a Design Tool," Conference Record of 7th Annual ACM Symposium on Principles of Programming Languages, Las Vegas (NV, pp.251-261 (June 1980).

[HEC 79] HECHT H. "Fault-tolerant Software," IEEE Transactions on reliability, vol. R-28, No.3, pp. 227-232, august 1979.

[HOA 74] HOARE C.A.R. "Monitors: An Operating System Structuring Concept," Communications of the ACM 17(10), pp.549-557, october 1974.

[HOA 78] HOARE C.A.R. "Communicating Sequential Processes," Communications of ACM, Vol.21, 8 (August 1978).

[HOR 74] HORNING J.J. and al. "A program structure for error detection and recovery" lecture note in computer sciences, 16 springer verlag 1974.

[HOS 83] HOSSEINI S.H. KUHL J.G. REDDY S.M. "An Integrated Approach to Error Recovery in Distributed Computing Systems," Digest of Papers FTCS-13, pp.56-63 (June 1983).

[JAL 84] JALOTE P., CAMPBELL R.H. "Fault tolerance using communicating sequential processes" Digest of papers FTCS-14, pp. 347-352, FLORIDA (june 1984).

[KES 77] KESSEL J.L.W. "An alternative to Event Queues for Synchronization in Monitors," Communications of ACM, Vol.20, 7, pp.500-503 (July 1977).

[KIM 79] KIM K.H. "Error detection, reconfiguration and recovery in distributed processing systems" in Proc. 1st int. conf. on distributed computing systems, oct. 1979, pp. 284-295.

[KIM 80] KIM K.H. "An implementation of a programmer-transparent scheme for coordinating concurrent processes in recovery" in Proc. COMPSAC 1980 pp. 615-621.

[KIM 82] KIM K.H. "Approches to mechanization of the conversation scheme based on monitors" IEEE trans. on software eng. vol SE 8,3 may 1982.

[LAM 78] LAMPORT L. "The Implementation of Reliable Distributed Multiprocess Systems," Computer Networks 2(1978) pp. 95-114.

[LAM 78] LAMPORT L. "Time, Clocks and the Ordering of Events in a Distributed System," CACM, Vol. 2, 1 (Feb. 1978).

[LAM 81] LAMPSON B.W. "Atomic Transactions in Distributed Systems-architecture and Implementation, Lecture Notes in Computer Science, 105, Berlin, Springer-Verlag, 1981, chap.11.

[LAN 77] LANDRAULT C. "Prevision de la Sureté de Fonctionnement des Systèmes Numériques Réparables," Thèse, Automatique Toulouse INP, 1977.

[LAP 85] LAPRIE . "Dependable computing and fault tolerance: Concepts and Terminology. Digest of papers FTCS-15, pp. 2-11, (june 1985).

[LEE 80] LEE P.A., GHANI N., HERON K. "A recovery cahe for the PDP-11" IEEE transactions on computers C-29 (6), pp. 546-549 (june 1980).

[LIS 79] LISKOV B., SNYDER A. "Exception handling in CLU" IEEE Trans. on Softw. Eng. SE-5, 6 1979.

[LIS 82] LISKOV B. "On Linguistic Support for Distributed Programs," IEEE Transactions on Software Engineering, vol.SE-8, No.3, may 1982.

[LOM 77] LOMET D.B. "Process Structuring, Synchronization, and Recovery Using Atomic actions," SIGPLAN Notices 12(3), pp.128-137 (March 1977).

[LUC 80] LUCKHAM D.C., POLAK W. "Ada Exception Handling: An Axiomatic Approach," ACM Transactions on Programming Languages and Systems 2(2), pp. 225-233 (april 1980).

[MAC 79] MACCHI C., GUILBERT J.F. "Téléinformatique," ed. Dunod Informatique 1979.

[MAC 77] MACLAREN M.D. "Exception Handling in PL/I," SIGPLAN Notices 12(3), pp. 101-104 (March 1977).

[MEL 77] MELLIAR SMITH P.M, RANDELL B. "The role of programmed exception handling" SIGPLAN Notices 12, 3, 1977.

[MET 76] METCALFE R.M., BOGGS D.R. "ETHERNET: Distributed Packet Switching for Local Computer Networks," CACM, vol.19, No.7, July 1976.

[MIL 80] MILENKOVIC M. "Synchronisation of Concurrent Updates in Redundant Distributed Databases," Proceedings of the International Symposium on Distributed Databases Paris FRANCE.

[NAC 84] NACKMAN L.R., RUSSELL H.T. "A hierarchical exception handler binding mechanism," Software-Practice and Experience, vol. 14(10), pp. 999-1007 (october 1984).

[PAR 72] PARNAS D.L. "On the Criteria to be Used in Decomposing Systems into Modules," Communications of the ACM, Vol.15, No.12, pp.1053-1058, December 1972.

[PRE 67] PREPARATA F.P., METZE G., CHIEN R.T. "On the connection assignment problem of diagnosable systems," IEEE Transactions on computers, vol. EC 16, december 1967.

[RAN 75] RANDELL B. "System structure for software fault tolerance" IEEE trans. on soft. eng. SE 1;2 (june 1975) pp. 220-232.

[RAN 78] RANDELL B., LEE P.A., TRELEAVEN P.C. "Reliability issues in computing system design," Computing surveys, vol.10 No.2, june 1978.

[RUS 77] RUSSELL D.L. "Process back up in producer-consumer systems" Proc. 6th symp. on operating system principle, Purdue, nov. 1977 pp. 151-157.

[RUS 79] RUSSELL D.L., TIEDEMAN M.J. "Multiprocess recovery using conversations," Digest of papers FTCS-9: Ninth Annual International Symposium on Fault-tolerant Computing, Madison (WI), pp. 106-109 (june 1979).

[RUS 80] RUSSELL D.L. "State restoration in systems of communicating processes" IEEE trans. on soft. eng. SE 6(2) pp. 183-194 march 1980.

[SHE 78] SHEDLETSKY J.J "A rollback interval for networks with an imperfect sel-checking property," IEEE transactionss on computers, vol C-27, No.6 june 1978.

[SHR 79] SHRIVASTAVA S.K "Concurrent pascal vith backward error recovery: implementation", Software- practice and experience 9 (12) pp. 1021- 1033 (dec. 1979).

[SHR 79] SHRIVASTAVA S.K. "Concurrent Pascal with Backward Error Recovery: Language Features and Exemples," Software - Practice and Experience, vol. 9, pp. 1001-1020, 1979.

[SHR 78] SHRIVASTAVA S.K., AKINPELU A.A. "Fault-tolerant sequential programming using recovery blocks," Digest of papers, FTCS-8, Toulouse, june 1978 pp 207.

[SIE 82] SIEWIOREK D.P., SWARZ R.S. "The Theory and Practice of Reliable System design (1982).

[TAY 80] TAYLOR D.J., MORGAN D.E., and BLACK J.P. "Redundancy in data structures: Improving software fault tolerance," IEEE Transactions on software engineering SE-6(6), pp.585-594 nov. 1980.

[WEG 83] WEGNER P., SMOLKA S.A. "Processes, tasks and monitors: A comparative study of concurrent programming primitives," IEEE transactions on software engineering vol. SE-9, No.4, july 1983.

[WEI 80] WEI A.Y. et al. "Application of the fault-tolerant deadline mechanism to a satellite on-board computer system," Digest of papers FTCS-10, Kyoto, pp.55-77 (october 1980).

[WEN 78] WENSLEY J.H., et al. "SIFT: Design and analysis of a fault-tolerant computer for aircraft control," Proceedings of the IEEE, vol.66, No.10, october 1978.

[WIL 79] WILMS P. "Etude et Comparison d'Algorithmes de Maintien de la Cohérence dans les Bases de Données Réparties," Thèse de Docteur-Ingenieur INPG, Novembre 1979,.

[Woo 80] WOOD W.G. "Recovery Control of Communicating Processes in a Distributed System," Technical Report 158, Computing Laboratory University of Newcastle Upon Tyne England, November 1980.

[WOO 81] WOOD W.G "A decentralised recovery control protocol". Digest of paper FTCS 11 june 1981 pp. 159-164.

[YOU 74] YOUNG J.W. "A first order approximation to the optimum checkpoint interval," CACM 17(9), pp. 530-531 (september 1974).

AUTORISATION de SOUTENANCE

VU les dispositions de l'article 3 de l'arrêté du 16 avril 1974

VU les rapports de présentation de Messieurs

. J.S BANINO, Directeur du département recherche et développement MATRA DATA Système
. B. COURTOIS, Chargé de recherche

Monsieur ALIOUAT Makhlouf

est autorisé à présenter une thèse en soutenance en vue de l'obtention du diplôme de DOCTEUR-INGENIEUR, spécialité "Informatique".

Fait à Grenoble, le 21 mars 1986

D. BLOCH
Président
de l'Institut National Polytechnique
de Grenoble

P.O. le Vice-Président,