

THÈSE DE DOCTORAT

de l'UNIVERSITÉ
DENIS DIDEROT – PARIS VII

Spécialité

Informatique Fondamentale

présentée par

STÉPHANE ERANIAN

Sujet de la thèse

Un service de synchronisation distribuée tolérant les pannes: implantation dans CHORUS

soutenue le 29 Septembre 1995 devant le jury composé de:

Paul Feautrier	Président
Guy Bernard Michel Raynal	Rapporteurs
François Armand Jean-Charles Fabre Hugues Fauconnier	Examineurs
Jean-Marie Rifflet	Directeur de Recherche

Un service de synchronisation distribuée tolérant les pannes : implantation dans CHORUS

Stéphane ERANIAN

Résumé - L'évolution des environnements informatique et notamment le développement des réseaux ouvre de nouvelles possibilités au niveau de la construction des systèmes d'exploitations : les systèmes répartis. Les nouvelles architectures sont basées sur des micro noyaux fournissant des abstractions et services de base à partir desquels on construit des systèmes complets via un ensemble de serveurs, disséminés sur plusieurs sites, communicants par échanges de messages. Cette répartition des tâches impose une certaine coordination afin de maintenir une cohérence d'ensemble. Ce maintien repose sur l'utilisation de synchronisations. Notre environnement cible est constitué d'un ensemble de sites fonctionnant sous le système micro noyau réparti CHORUS.

Le but de cette thèse est de fournir un service de synchronisation distribuée générique basée sur la notion de jeton pour garantir la propriété d'exclusion mutuelle répartie. Deux qualités de service sont fournies. La première est basée sur un algorithme centralisé, la seconde utilise un algorithme distribué, dérivé de celui de Naimi et Trehel, basé sur une structure arborescente dynamique. Nous présentons des résultats sur les performances obtenus à partir d'un prototype.

Dans un second temps, nous montrons comment ce service est rendu tolérant aux pannes franches de site sans perte de performances. Les services sont régénérés dynamiquement, les clients sont perdus et les sites peuvent être réintégrés après réparation. Nous n'utilisons aucune technique à base de points de contrôle ou transactions. Notre architecture repose sur un service générique externe, aM ou gestionnaire de pannes réparti. Il fournit des services de détection de pannes, de notifications sur pannes et d'aide au recouvrement. Le serveur de remplacement est choisi dynamiquement suivant un principe d'hospitalité. Le recouvrement exploite la nature répartie de l'environnement, il est basé sur la collecte de l'état auprès des clients survivants en vue de sa reconstruction dans le serveur. La tolérance est supportée dans les deux qualités de services.

Abstract - The evolution of computing environments and especially networks opens new opportunities in the way of building operating system : distributed systems. New architectures are based on micro kernels which provide basic abstractions and services on which one can build a complete system made of a set of servers, running on different sites, which communicate through a message passing facility. In this context, the distribution of work requires a certain degree of coordination in order to maintain a global consistency. This, in turn, requires the use synchronization primitives. Our target environment is the CHORUS micro kernel system.

The purpose of this PhD thesis is to provide a distributed synchronization service based on the token notion to ensure the distributed mutual exclusion property. Two qualities of service are provided. The first is based on a simple centralized algorithm, the second uses a distributed algorithm, derived from the one by Naimi and Trehel, based on a dynamic tree structure. We present a set of performance results gathered from our prototype.

In the second part, we explain how this service can support fail-stop site failure without loss in performances. Service is regenerated dynamically on another site, clients are lost and site may be restarted after repair. We do not use any checkpoints nor transactions based techniques. Our architecture is based on an external service, the aM or failure manager. It provides services for failure detection, failure notification and recovery help. The backup server is selected dynamically using a hospitality principle. The recovery takes fully benefit of the distributed nature of the environment, it is based on a distributed state collect from the remaining clients. Fault tolerance is supported for both qualities of service.

Table des matières

1	Introduction	5
1.1	Vers de nouvelles architectures des systèmes	6
1.2	Les problèmes de synchronisation	7
1.2.1	Un service de synchronisation distribuée pour CHORUS	9
1.2.2	Organisation du document	10
2	État de l'art	13
2.1	Introduction	13
2.2	Classification des algorithmes	13
2.2.1	Utilisation de la notion de permission	15
2.2.2	Utilisation de la notion de privilège	15
2.3	Les algorithmes basés sur des permissions	16
2.3.1	Les algorithmes à permissions individuelles	16
2.3.2	Les algorithmes à permissions d'arbitres	23
2.3.3	Un algorithme général	27
2.4	Les algorithmes basés sur des jetons	29
2.4.1	Les algorithmes à diffusion	31
2.4.2	Les algorithmes basés sur le multicast	34
2.4.3	Les algorithmes à structuration	35
2.4.4	Algorithme basé sur un arbre	41
2.5	Conclusions	49
3	Le système d'exploitation réparti CHORUS	51
3.1	Concepts fondamentaux des systèmes répartis	51
3.2	Le micro-noyau CHORUS	52
3.2.1	Introduction et historique	53
3.2.2	Architecture globale	53
3.2.3	La notion de site	54
3.2.4	La désignation des entités	54
3.2.5	Architecture interne	55
3.2.6	L'exécutif	56
3.2.7	Le superviseur	60
3.2.8	La gestion de la mémoire	61
3.3	Le système de communication	64
3.3.1	Principes	64
3.3.2	Les portes	64

3.3.3	Les messages	65
3.3.4	Les groupes	65
3.3.5	L'adressage	66
3.3.6	La qualité de service	68
3.4	Les sous-systèmes	69
3.4.1	Principes	69
3.4.2	Le sous-système CHORUS/MiX V.4	69
3.4.3	Le sous-système CHORUS/ClassiX	76
3.4.4	Les autres sous-systèmes	80
3.5	Conclusion	83
4	Le service de synchronisation distribuée	85
4.1	Introduction	85
4.2	But du service	85
4.3	Jeton ou permissions ?	87
4.4	Description générale	89
4.4.1	Architecture globale	89
4.4.2	Rôle du serveur	90
4.4.3	Rôle du client	91
4.4.4	Interface utilisateur	91
4.4.5	Identification du jeton	93
4.4.6	Identification des clients	94
4.4.7	Création d'un jeton	95
4.4.8	Partage d'un jeton	96
4.4.9	Destruction d'un jeton	97
4.4.10	Les qualités de service	98
4.4.11	Caractéristiques principales des protocoles	99
4.4.12	Remarques sur la présentation des algorithmes	100
4.4.13	Atomicité des opérations	101
4.5	Le service centralisé	101
4.5.1	Principes	101
4.5.2	Caractérisation du jeton	102
4.5.3	Algorithmique	104
4.5.4	Déroulement des requêtes	115
4.5.5	Les mécanismes de callback	120
4.5.6	Complexité	124
4.6	Le service distribué	125
4.6.1	Principes	125
4.6.2	Adaptation de l'algorithme de Naimi et Trehel	126
4.6.3	Caractérisation du jeton	127
4.6.4	Algorithmique	127
4.6.5	Le problème de la suppression	138
4.7	Évaluations	147
4.7.1	Construction du prototype	147
4.7.2	Conditions de tests	149
4.7.3	Les types de test	149
4.7.4	Les tests de complexité	149

4.7.5	Les tests de comportement	152
4.8	Conclusion	160
5	La tolérance aux pannes	161
5.1	Introduction	161
5.2	État de l'art	162
5.2.1	Un peu de terminologie	162
5.2.2	Les différents types de pannes	163
5.2.3	Éléments de stratégies	164
5.2.4	Les algorithmes basés sur les jetons	164
5.3	Environnement de travail	167
5.4	Choix de construction	169
5.4.1	Terminologie	169
5.4.2	But	170
5.4.3	Techniques pour la tolérance aux pannes	171
5.4.4	Choix	175
5.5	Architecture globale	179
5.6	Le gestionnaire de pannes: αM	179
5.6.1	Principes	180
5.6.2	Architecture interne	180
5.6.3	Désignations des entités	183
5.6.4	Le mandataire du côté client: Proxy αM	185
5.6.5	Communication entre proxy et agent	187
5.6.6	Gestion des sites	187
5.6.7	Gestion des services	192
5.6.8	Le mécanisme de recouvrement	200
5.6.9	Détection de panne	208
5.6.10	Conclusions	209
5.7	Description générale	209
5.7.1	Introduction	209
5.7.2	Nouvelle architecture	210
5.7.3	Qualité de service	211
5.7.4	Identification du jeton	211
5.7.5	Indicateur de passage	212
5.7.6	Identification des clients	212
5.7.7	Déclaration du service	213
5.7.8	Création du jeton	213
5.7.9	Partage du jeton	215
5.7.10	Suppression d'un client	215
5.7.11	Caractéristiques des protocoles	215
5.8	Principes généraux du recouvrement	218
5.8.1	La perte de clients	218
5.8.2	Le recouvrement du service	218
5.9	Le mode centralisé	220
5.9.1	Pannes de clients	220
5.9.2	Pannes du serveur	228
5.9.3	Pannes du client et des serveurs	240

5.10	Le mode distribué	240
5.10.1	Avantages inhérents	241
5.10.2	Défauts inhérents	242
5.10.3	Principes du diagnostic	242
5.10.4	Pannes de clients	247
5.10.5	Panne du serveur	258
5.10.6	Pannes du client et des serveurs	259
5.11	Le redémarrage de sites	259
5.11.1	Hypothèses et restrictions	259
5.11.2	Principes	260
5.11.3	Placement dans l'anneau	260
5.11.4	Le numéro d'incarnation	261
5.11.5	Les informations sur les sites	264
5.11.6	Les informations sur les services	264
5.11.7	Le réactivation effective	265
5.12	Conclusions	265
6	Conclusions	269
6.1	Bilan	269
6.1.1	Le service de synchronisation	269
6.1.2	La tolérance aux pannes	271
6.2	Perspectives	274
6.2.1	Le service de base	274
6.2.2	Vers un nouveau mode	276
6.2.3	Restructuration des protocoles	280
6.2.4	La tolérance aux pannes	280
6.3	Épilogue	282
A	Le mode hybride	283
A.1	La complexité	283
A.2	L'efficacité	283
A.3	Conclusions	285