Intergiciels et coordination distribuée robuste dans les systèmes temps réel

Gérard Le Lann

INRIA Rocquencourt

Gerard.Le_Lann@inria.fr

1. Prolégomènes

2. Solutions asynchrones pour le temps réel ?

3. Consensus Uniforme

4. Coordination Uniforme

5. Conclusions

1. Prolégomènes

Consensus exact ou approché, diffusion atomique, élection de leader, reconfiguration dynamique, appartenance à un groupe, redéfinition en-ligne de mission/stratégie, …

sont des exemples de services transversaux à de nombreux domaines applicatifs (avionique, spatial, télécoms, bourses et marchés financiers, nucléaire, …)

► algorithmes CDR

(Coordination/Consensus Distribué(e) Robuste)

L'obtention de ces services pose des problèmes informatiques plus ou moins difficiles, selon les modèles (1) de calcul, (2) de défaillances, considérés.

Les premiers produits commercialisés (ISIS, par exemple) reposaient sur des ‘solutions’ CDR intergicielles en contradiction flagrante avec des résultats d'impossibilité établis vers le milieu des années 80, ce qui explique en partie les échecs subis par ces produits – et les déboires des «clients» (donneurs d’ordre et industriels, contrôle de trafic aérien, bourses, etc.) qui ont investi dans ces produits.

Au cours des dix dernières années, un grand nombre de solutions, accompagnées de leurs preuves, ont été établies par la communauté scientifique. Cependant …

Un problème ‘caché’ difficile est celui de la **détection des vraies défaillances en temps borné calculable, prédictible** et, si possible, optimal. Autrement dit, que l'application considérée soit ou ne soit pas « temps réel », peu importe :

pour qu'un produit CDR soit utilisable, il est obligatoire qu’il soit accompagné d'un ‘guide’ permettant de calculer des bornes supérieures exactes sur des temps de réponse (valeurs des réveils qui servent à détecter des défaillances, latence pour obtenir un consensus, …)

► **problèmes d’ordonnancement « temps réel »**

Nos travaux récents et actuels sont fondés sur des coopérations scientifiques internationales (USA, Canada, Autriche) et sur des enseignements obtenus à l'occasion de deux projets d'intergiciels distribués robustes et temps réel, menés entre 1996 et 1999 :

* le **projet ORECA**, financé par la DGA; partenaire industriel Dassault Aviation; solution "synchrone",
* le **projet ATR**, financé par la DGA, le MENRT, le CNRS; partenaires industriels Thomson-Airsys, Dassault Aviation, Axlog Ingénierie; partenaires scientifiques : Univ. Paris 7, LIX; solution "asynchrone".

Propriétés désirées pour les services CDR :

SafeP (sûreté), LiveP (vivacité), DepP (confiance), TimeP (ponctualité).

Minimum requis, dans tous les cas réels : preuves de SafeP !

Solution asynchrone : ne repose sur aucune hypothèse de « timing ».

Solution synchrone ou partiellement synchrone : repose sur des hypothèses de « timing » (réveils, comptages de temps, délais de transmission des messages, …).

Nancy Lynch, 1996:
"It is impossible or inefficient to implement the synchronous model in many types of distributed systems."

⇒ "distribué synchrone" est une oxymore pour la plupart des applications et systèmes réels !

2. Solutions asynchrones pour le temps réel?

Universellement admis :

Solutions asynchrones ► taux de couverture les plus proches de 1 ☺

Solutions synchrones : si une des hypothèses de « timing » est violée en fonctionnement opérationnel, on perd TimeP et toutes les autres propriétés (aucune des propositions connues ne permet d'éviter cela).

Solutions asynchrones, pour les problèmes CDR : on ne perd jamais SafeP (même si on perd TimeP).

Affirmations gratuites :

Solutions asynchrones inenvisageables pour le temps réel ☹

Solutions asynchrones nécessairement moins efficaces que les solutions synchrones ☹

Solutions asynchrones inenvisageables pour le temps réel ?

Faux.

(1) Une solution conçue (et prouvée correcte) dans un modèle asynchrone peut être implémentée dans un système conforme à un modèle synchrone.

(2) Les hypothèses « synchrones » ne sont nécessaires que pour exprimer les bornes sup. des temps de réponse (établir les conditions de faisabilité des propriétés TimeP), c'est-à-dire seulement en fin de phase de conception, avant dimensionnement/déploiement.

⇒ concept de "late binding"
(d'une solution à un modèle de calcul)

Solutions asynchrones nécessairement moins efficaces que les solutions synchrones ?

Faux.

Il existe de nombreux contre-exemples (circuits intégrés, sciences du vivant, …). Pour les problèmes CDR, raisons et démonstration données dans [HLL02].

Exemple de résultat : (C dénote un taux de couverture ;

D_m = borne sup temporelle (TimeP) démontrée dans le modèle m)

- **ou bien $D_{sync} = D_{Async}$ et $C(D_{Sync}) < C(D_{Async})$**
- **ou bien $C(D_{Sync}) = C(D_{Async})$ et $D_{sync} > D_{Async}$**

► Si on choisit une solution synchrone, on perd soit en taux de couverture, soit en performances.

Solutions asynchrones pour :

Consensus Uniforme Rapide (FastUCS)

Coordination Uniforme Rapide (FastUCN)

◆ Implantation expérimentale par une société d'ingénierie :

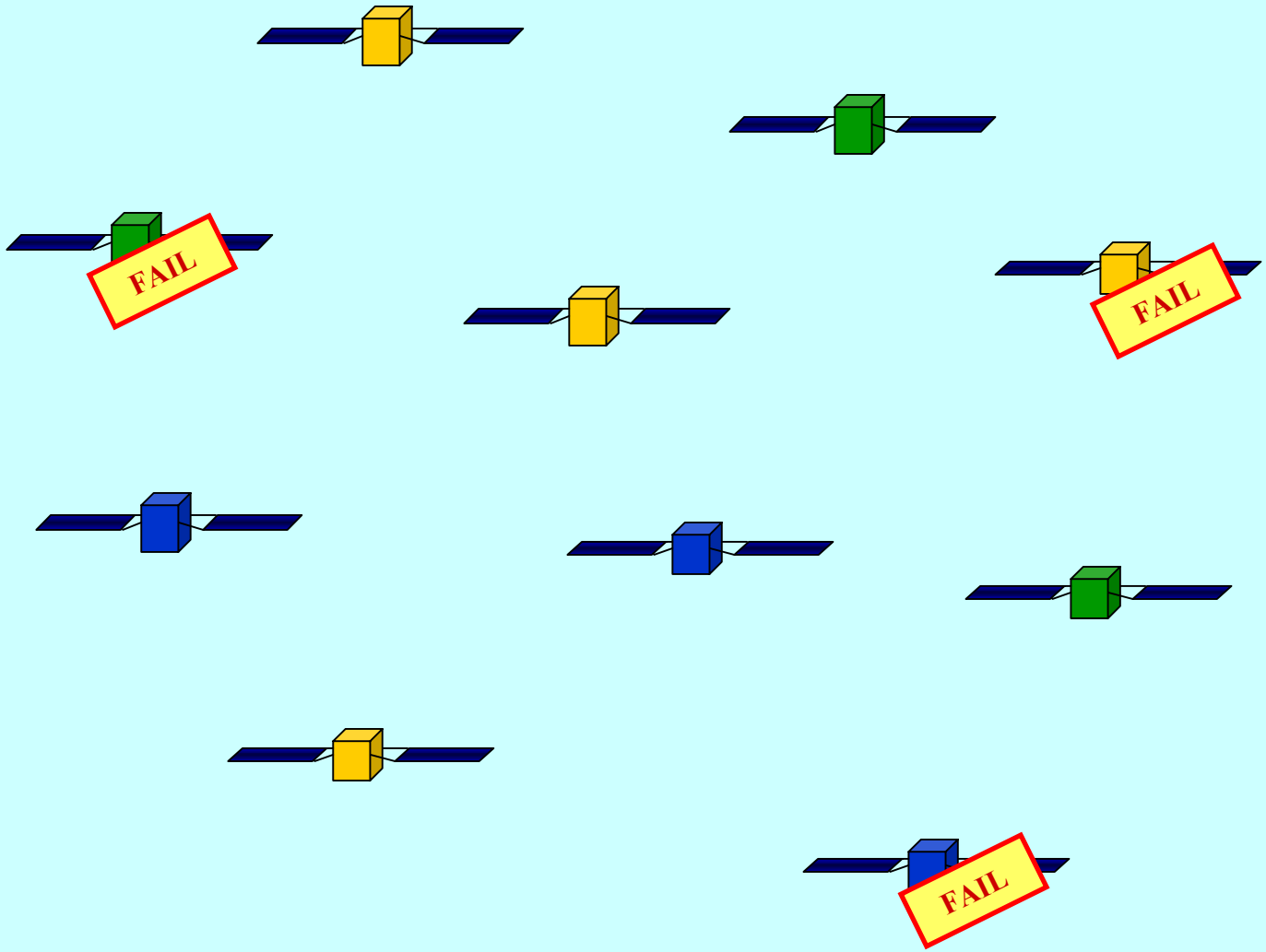
- Niveau intergiciel pour les algorithmes,
- Niveau intergiciel ou inférieur (p.e., sur UDP) pour les FD (Failure Detectors).

◆ En cours d'examen, pour prototypage « ciblé », par une agence spatiale et un industriel du domaine.

◆ Solutions potentielles pour les systèmes autonomes, pour les allocations dynamiques de ressources (bandes de fréquences, slots temporels, processus/processeurs, …), …







FAIL
FAIL
FAIL

3. Consensus Uniforme (1994)

Communications fiables, défaillances des processus = crash.
Initialement, tout processus k propose une valeur v_k. Une seule est retenue comme Décision finale.

Sûreté (SafeP)

- Validité uniforme: si un processus Décide valeur v, alors v a été proposée initialement par un processus (éventuellement défaillant).
- Intégrité uniforme : tout processus (éventuellement défaillant) Décide au plus 1 fois.
- Accord uniforme : deux processus quelconques (éventuellement défaillants) ne peuvent Décider différemment.

Vivacité (LiveP)

- Tout processus non défaillant finit par Décider.
  [en au plus L unités de temps – **TimeP** – pour le temps réel]

4. Solution pour Consensus Uniforme Rapide

Phase = un multicast de message + des calculs sur les messages reçus.
D = durée pire cas d'une phase.
f = nombre de défaillances pendant Consensus.

Solutions optimales synchrones: borne pire cas = (f+1) D_{sync}

Dans [HLL02] :
FastUCS, solution algorithmique asynchrone, qui repose sur des FastFD (détecteurs de défaillances rapides dérivés des détecteurs de défaillances de Chandra-Toueg).

Sémantique des FastFD : « time free ». Implémentation possible dans un modèle partiellement synchrone (« late binding »).

Dans [HLL02], on donne les expressions analytiques des valeurs des réveils qui permettent d'implémenter les sémantiques « Strong FD » ou « Perfect FD ».

FastUCS ➡ borne pire cas optimale. Pour la plupart des systèmes réels :

borne pire cas = D_{async} [rappel : $D_{async} < D_{sync}$]

Exemple numérique :

Système sur Ethernet, 500 processeurs, 5% capacité prise par les FastFD (processeurs et communication), f = 3 :

- **latence de détection de vraie défaillance = 54 ms,**
- **borne pire cas pour Consensus Uniforme avec FastUCS = 407 ms.**
  (borne > 1,6 s avec une solution synchrone optimale)

5. Coordination Uniforme (2000)

Communications fiables, défaillances des processus = crash.

Initialement, tout processus k soumet une contribution c_k (calcul partiel, état local (p.e., file d'attente), observation locale sur l'environnement,…). Les contributions sont « agrégées », selon des règles de calcul qui dépendent uniquement de la sémantique applicative. Une seule des « agrégations » est retenue comme Décision finale.

Sert pour (exemples) :

ordonnancement distribué, redéfinition autonome de mission/stratégie et re-vérification/validation (apprentissage en-ligne), satisfaction de contraintes distribuées (backtracking, etc.), reconfiguration en-ligne, …

Sûreté (SafeP)

- Validité uniforme : si un processus Décide A, alors A est une « agrégation » de contributions proposées initialement par des processus (éventuellement défaillants).
- Intégrité uniforme : tout processus (éventuellement défaillant) Décide au plus 1 fois.
- Accord uniforme : deux processus quelconques (éventuellement défaillants) ne peuvent Décider différemment.

Vivacité (LiveP)

- Tout processus non défaillant finit par Décider.

[en au plus L unités de temps – **TimeP** – pour le temps réel]

Solution FastUCN construite à partir de FastUCS.

5. CONCLUSIONS

➥ Par « mariage » de différentes disciplines (p.e., algorithmes distribués, tolérance aux fautes, ordonnancement temps réel, IA, optimisation combinatoire), on peut résoudre des problèmes réputés « difficiles », et fournir les preuves.

➥ En tenant compte des réalités (p.e., inévitabilité des phénomènes de conflits sur ressources partagées, de files d'attente, asynchronismes), on peut découvrir des solutions efficaces (performances, implantation).

➥ En cours, extensions des résultats obtenus à d'autres modèles de défaillance (p.e., logiciels non fiables, défaillances Byzantines des processeurs, contrat CNES 2001-2002).

➽ **Détails dans [HLL02]:**

« Fast Asynchronous Uniform Consensus in Real-Time Distributed Systems »

J.-F. Hermant, G. Le Lann, IEEE Transactions on Computers, vol. 51, n° 8, August 2002,
Special Issue on Asynchronous Real-Time Distributed Systems.