Une approche à la validité des protocoles d'enchères par la méthode des tests de spécification

Christine Ecault

INRIA

CENTRE DE RENNES

IRISA

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél. (3) 954 90 20

Rapports de Recherche

N° 279

UNE APPROCHE À LA VALIDATION DES PROTOCOLES D'ENCHÈRE PAR LA MÉTHODE DES TESTS DE SPÉCIFICATION

Christine ECAULT

Mars 1984

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE
ET SYSTÈMES ALÉATOIRES

Campus Universitaire de Beaulieu
Avenue du Général Leclerc
35042 - RENNES CÉDEX
FRANCE
Tél. : (99) 36.20.00
Télex : UNIRISA 95 0473 F

Une Approche à la Validation des Protocoles d'ENCHERE

par la méthode des tests de Spécification. *

Publication Interne n° 223
36 pages

Christine ECAULT
IRISA - Université de Rennes I
Février 1984

Résumé :

Le propos de ce rapport est de présenter l'utilisation d'un outil de validation pour un algorithme d'ordonnancement équitable mis en oeuvre dans l'application ENCHERE. L'outil a été mis au point par une équipe du CNET Lannion. Nous présentons le langage de spécification utilisé pour décrire l'algorithme étudié, ainsi que le principe de traduction des spécifications en un programme exécutable. Après une brève description de l'application ENCHERE nous détaillons la spécification de l'algorithme et nous donnons quelques résultats d'exécution.

Abstract :

The purpose of this report is to present the use of a validation tool for a fair scheduling algorithm implemented in the system "ENCHERE". The tool was developed by a CNET-Lannion's team. First we present the specification language used to describe the algorithm and the translation principle of specifications to a simulation program. Then, after a brief description of the ENCHERE system, we give the specifications of the algorithm and detail some results obtained by executing the simulation program.

Rapport de Recherche

* Cette étude a été effectuée en liaison avec le département EVP du CNET Lannion et financée par la convention CNET No 82-1B 234 00 790 45 BCW/LAA.

INTRODUCTION

Plusieurs phases sont nécessaires à la conception d'un système informatique : parmi celles-ci on reconnait généralement comme primordiale celle de validation. Traditionnellement on distingue deux approches à la validation [JARD 83] :

1) une approche mathématique qui consiste en une analyse des spécifications formelles de l'application, en vue de démontrer l'absence d'incohérences.

2) une approche plus pragmatique qui procède par production de traces d'exécution possibles (à l'aide d'un logiciel adapté) puis analyse (le plus souvent manuelle) de ces traces.

L'approche que nous nous proposons d'expérimenter dans le travail ici présenté est plutôt du deuxième type. L'objectif visé consiste à valider un algorithme d'ordonnancement équitable mis en oeuvre dans l'application ENCHERE.

La démarche employée comporte essentiellement deux étapes :

1) La spécification de l'algorithme étudié. Elle est réalisée à l'aide d'un langage de spécification mis au point par une équipe du CNET Lannion [ISO 81].

2) La simulation et exécution avec production de traces. Bien entendu, pour effectuer cette simulation il est nécessaire de traduire (manuellement pour l'instant) le texte des spécifications en un programme exécutable sur ordinateur. En ce qui nous concerne le langage cible est SIMONE [BEZI 76].

Dans un premier chapître nous décrivons brièvement l'application support de notre étude. Le second chapître présente l'outil d'évaluation, i.e. le langage de spécification, le rôle et la spécification des modules du système, et le principe de la traduction en SIMONE. Le troisième chapître est relatif à la spécification de l'environnement d'ENCHERE et de l'algorithme d'ordonnancement équitable. Il présente, conformément au langage de snécification, les modules décrivant le protocole et l'environnement, et la configuration du système. Enfin nous donnons dans un dernier chapître les résultats concernant l'algorithme et quelques remarques vis à vis de l'outil d'évaluation.

1. PRESENTATION DE L'APPLICATION

Le système ENCHERE est un système réparti de ventes aux enchères entre des acheteurs et des vendeurs situés en des lieux géographiquement éloignés. Il doit permettre le traitement d'un grand nombre de ventes à une grande vitesse, tout en assurant fiabilité et équité entre acheteurs et vendeurs conformément aux règles du marché.

1.1 Description informelle du service

Une session de vente est constituée des ventes de lots des vendeurs aux acheteurs intéressés. Chaque session débute par l'insertion des vendeurs et des acheteurs dans le service.

La vente d'un lot se déroule de la manière suivante :

- Le vendeur envoie une proposition de vente aux acheteurs intéressés.
- Les acheteurs la reçoivent et chacun d'eux fait une offre (i.e. propose un prix).
- Quand tous ont fait une offre, le vendeur accepte ou refuse celle correspondant au meilleur prix. Sa décision est communiquée aux acheteurs (figure 1).

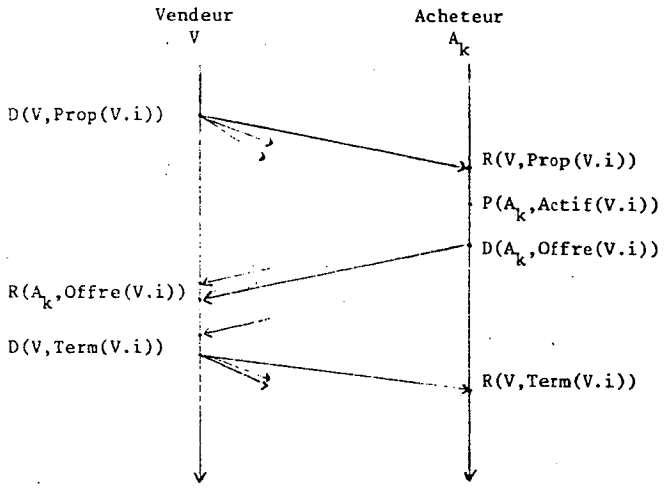

Figure 1 : Déroulement temporel d'une vente sur deux sites V et A_k. A_k participe à la vente V.i de V.

D : Diffusion de message.
R : Réception de message.
P : Production d'événements.

Contraintes de fonctionnement du service :

1) Les vendeurs sont indépendants les uns des autres, et les acheteurs de même. Les seuls liens entre acheteurs et vendeurs sont les messages échangés nécessaires à la transaction.

2) Le déroulement des transactions se fait de manière séquentielle : un acheteur ne traite pas plus d'une vente à la fois. Un vendeur doit terminer une transaction avant d'en commencer une autre.

Le système doit être égalitaire, ce qui se traduit par le respect des deux propriétés que nous exprimons informellement comme suit :

(P1) : Si deux acheteurs A et A' participent aux ventes de deux vendeurs V et V', ils doivent voir leurs propositions de

vente dans le même ordre.

(P2) : Aucune priorité n'est donnée aux propositions d'un quelconque vendeur pour un acheteur particulier : si à un instant d un acheteur a reçu deux propositions i d'un vendeur V et j d'un vendeur V', alors s'il traite la vente i de V en premier, il devra traiter la vente j de V' avant de traiter la vente (i+1) de V.

La définition formelle de ces propriétés dites d'équité peut être trouvée dans [BANA 83].

1.2 Mise en oeuvre des propriétés d'équité

Pour mettre en oeuvre les propriétés d'équité, il faut ordonnancer les ventes. Cela peut être réalisé par un mécanisme de numérotation tels ceux présentés dans [LAMP 78] ou [RICA 81]. La numérotation considérée dans cet article est basée sur le temps délivré par les horloges physiques locales des sites sur lesquels sont mis en oeuvre acheteurs et vendeurs.

1.2.1 Mise en oeuvre de P1

Pour assurer que les acheteurs voient les propositions dans le même ordre, les ventes sont estampillées de l'heure locale physique du vendeur au moment de l'émission de la proposition de vente. Le choix de la vente à activer sur un site acheteur se fait par comparaison des dates associées à chacune des propositions de vente. La vente choisie est celle dont la date est la plus ancienne.

1.2.2 Mise en oeuvre de P2

La seule utilisation des estampilles n'est pas suffisante pour

assurer P2, comme le montre l'exemple suivant :

Soit V1,V2,V3, des sites vendeurs proposant des ventes,
A1 un site acheteur lié à V1,V2 et V3.
V1 propose V1.i avec son heure locale h1=10
V2 propose V2.j avec son heure locale h2=15
V3 propose V3.k avec son heure locale h3=20
La technique de mise en oeuvre de P1 amène A1 à choisir V1.i. Supposons la durée de transaction égale à 3 unités de temps. Quand V1.i se termine, V1 lance une autre vente V1.i' qui peut être estampillée (H1'=15), tandis que A1 choisit V2.j. Quand V2.j se termine, A1 doit choisir V1.i', et P2 est alors violée.

Une solution à ce problème consiste à introduire la notion d'heure floue définie comme suit : deux sites S1 et S2 ont même heure floue si leurs horloges H(S1) et H(S2) vérifient la propriété : $|H(S1)-H(S2)|<dt$ (dt dérive maximale entre deux sites).

P2 est vérifiée ssi [BANA 82] :

(1) les sites qui participent à une même transaction ont même heure floue,

(2) la durée d'une transaction est plus grande que dt,

(3) le délai de transmission entre deux sites est borné par une valeur connue DT, et

(4) la dérive dt est supérieure à 2DT.

Les propriétés (P1) et (P2) étant vérifiées sous les conditions que nous venons d'énumérer nous présentons maintenant comment l'algorithme maintient ou rétablit celles-ci lorsque les hypothèses (1)-(4) sont mises en défaut.

1.3 Maintien des propriétés d'équité en cas de désynchronisation

Il se peut que les sites n'aient plus même heure floue, par exemple lorsque les horloges des sites n'ont plus même période.

Examinons les moyens offerts en [BANA 83] en vue de rétablir

la bonne propriété sur les heures floues.

Soient deux sites S1 et S2 tels que H(S1)-H(S2)>dt. Il est alors nécessaire de recaler l'horloge de S2, ce qui s'effectue de la façon suivante :

Chaque message envoyé est estampillé de l'heure locale sur le site. Lorsque S2 reçoit un message il compare l'heure d'émission du message à son heure locale pour vérifier s'il a même heure floue que l'émetteur.Si S2 retarde par rapport à S1, il recale son horloge avec l'heure reçue de S1, et il diffuse sa nouvelle heure aux autres sites avec lesquels il communique pour qu'ils se resynchronisent.

Pendant la désynchronisation, il est possible que la propriété P1 n'ait pas été respectée, par exemple dans la situation où l'horloge d'un acheteur Ai avance par rapport à celle d'un vendeur Vj. Une proposition de vente de Vj peut parvenir à Ai alors qu'une vente dont la date est postérieure est déjà activée. Il est donc nécessaire de rétablir P1. L'algorithme suivant est utilisé :
- L'acheteur Ai "défait" localement la vente en cours, en avise le vendeur concerné, et active la vente de Vj parvenue avec retard.
- Le vendeur Vj qui retarde recale son horloge.

On doit construire un modèle de système permettant de vérifier le maintien de (P1) et (P2) dans les situations suivantes :
1) en fonctionnement normal, avec les hypothèses énoncées en 1.1 et 1.2.2.
2) en cas de désynchronisation des horloges.
3) en cas d'allongement de la durée de transmission des messages.

Avant de détailler cette validation nous présentons l'outil utilisé.

2. PRESENTATION DE L'OUTIL DE VALIDATION

Avant d'arriver à la construction d'un modèle exécutable, le premier travail est l'établissement des spécifications de l'algorithme et de son environnement à l'aide d'un langage spécialement conçu pour spécifier les algorithmes distribués. A partir de cette spécification nous produisons manuellement le programme exécutable pour la simulation, actuellement réalisé en SIMONE (un traducteur automatique de la spécification est à l'étude au CNET Lannion).

2.1 Etablissement des spécifications

L'architecture d'un système est définie par les modules qui composent le système et par leurs connexions (figure 2) [JARD 81]. La description du modèle est donc constituée des spécifications des canaux permettent l'échange des messages, et spécifications des modules.

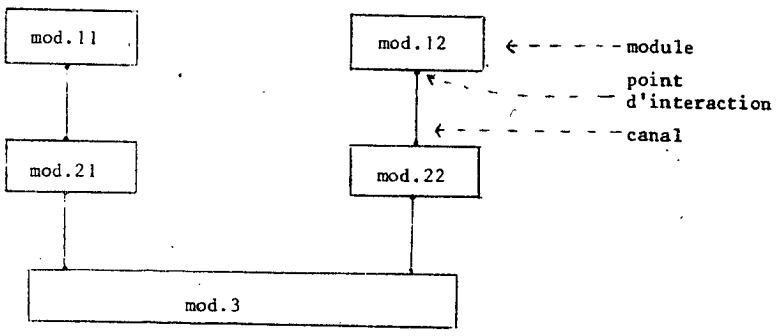

Figure 2 : Représentation de modules et de leurs liaisons.

2.1.1 Les canaux

Les canaux permettent la communication entre modules. La

première étape dans la description du système est la donnée des définitions de modèles de canaux (à partir desquels il est possible de construire le système en créant des exemplaires).

La spécification d'un modèle de canal comprend :

- les noms des utilisateurs du canal. Les modules qui communiquent via ce canal pourront ensuite s'identifier à l'un des utilisateurs.
- une liste des opérations fournies par ce canal associée à chaque nom.

La syntaxe d'un canal est la suivante :

```
Interactions
  nom_du_canal (partenaire_1,partenaire_2) is
    by  partenaire_1  : opération_1 (champ_1,champ_2,...);
                        opération_2 (...  );
                        ...
    by partenaire_2 : ... ;
                        ...
  end;
```

où nom_du_canal est l'identificateur du canal, partenaire_1 et partenaire_2 les noms des utilisateurs du canal, et operation_1 , opération_2 les noms des requêtes pouvant être émises par le partenaire_1 à destination du partenaire_2.

Exemple :

```
    Interactions
      Service_de_vente (utilisateur,systeme) is
(1)   by utilisateur : creer_prop (lot : lots);
(2)                    creer_term (lot : lots);
(3)                    finvente;
(4)   by systeme : fin_offre (vente : ventes);
(5)                nouv_vente;
      end;
```

L'"utilisateur" et le "systeme" utilisent le canal "service_de_vente" pour communiquer. L'"utilisateur" peut effectuer les requêtes creer_prop, creer_term et finvente à destination du "systeme" (lignes 1-2-3), lequel peut adresser les requêtes fin_offre et nouv_vente à l'"utilisateur" (lignes 4-5).

2.1.2 Les modules

a) Les interactions du module

Les modules interagissent entre eux via des canaux. L'en-tête d'un module est constituée de la liste des canaux qu'il utilise.

Par exemple :

```
module Util-vendeur (EV: service_de_vente(utilisateur);
                     TM: service_de_temporisation(demandeur));
```

L'en-tête de Util_vendeur décrit l'interface de l'utilisateur vendeur. Le module utilise le canal EV, de type service_de_vente. Les échanges autorisés via ce canal ont été décrits dans la déclaration des canaux (cf 2.1.2). Le module Util_vendeur l'utilise en tant qu'"utilisateur".
Le second canal, TM, est employé pour l'utilisation d'un service interne de gestion de temporisation.

b) description des modules

Le modèle utilisé pour la description des modules est un automate à transitions d'états finis. Il est donc nécessaire de caractériser la notion d'état d'un module. Cet état plus des fonctions de transition permettent de déterminer les transitions possibles du module.

L'espace des états du module est défini par un ensemble de variables. Chaque combinaison de valeurs prises par ces variables définit un état.

Une transition est caractérisée par :

- une action : elle peut modifier les valeurs des variables et donc l'état du module, et peut produire une ou plusieurs interactions avec d'autres modules.
- une condition de déclenchement de l'action : celle-ci est une

expression booléenne sur les variables des modules et éventuellement une condition sur l'arrivée de messages.

La syntaxe d'une transition est :

when <condition> do <action>

- Une condition portant sur l'arrivée d'une requête s'écrit :

<nom_de_canal>.<requête>(paramètres formels)

Lorsque ce booléen est égal à vrai, c'est qu'il y a arrivée d'une requête par le canal nommé. A partir de cet instant l'action peut être effectuée.

- L'action est une suite d'instructions PASCAL.

L'instruction particulière d'émission d'une requête sur un canal s'écrit :

<nom_de_canal>.<requête>(valeur_des_paramètres)

Exemple de transition :

```
(1) when UTA.creer_offre (la_vente) do
(2)  begin if etat=attoff and la_vente=dernactiv then
(3)        begin  IAV[lavente.V].offre(moi,ma_date,la_vente);
(4)              etat := attfin      end
     end;
```

Cette transition est effectuée par le module Entité-Achat. La condition de déclenchement de l'action est l'arrivée d'un message "creer_offre" par le canal "UTA" (ligne 1). Sous réserve que cette réponse soit bien attendue (ligne 2), le module envoie une requête "offre" par le canal IAV[vendeur] au vendeur concerné par cette vente (ligne 3).

2.1.3 Configuration du système

Nous avons montré comment décrire l'échange des messages et les différents modules. La dernière étape de la construction du modèle global est la description de la configuration: il s'agit de construire cette configuration à l'aide d'exemplaires de modules

du système et de leurs connexions.

Il est possible de nommer chaque exemplaire de module ainsi créé et de s'y référer. Les interactions s'effectuent de port à port, et dans une partie dite "connection" il est possible de définir les liaisons port à port ainsi que leur sens (uni ou bidirectionnel).

Exemple :

```
modules
  UTIA : array[1..maxacht] of Util-Acheteur;
  UTIV : array[1..maxvend] of Util-Vendeur;
  ENTA : array[1..maxacht] of Entite-Achat;
  ENTV : array[1..maxvend] of Entite-Vente;
  RESE : Reseau;

connection
  for all V in [1..maxvend] do
   begin UTIV[V].EV  ---> ENTV[V].UTV;
         ENTV[V].UTV ---> UTIV[V].EV;
         for all A in [1..maxacht] do
          ENTV[V].IAV[A] ---> RESE.IAV[2,A];
         RESE.IAV[1,V] ---> ENTV[V].IAV[V]
   end;
  for all A in [1..maxacht] do
   begin UTIA[A].EA  ---> ENTA[A].UTA;
         ENTA[A].UTA ---> UTIA[A].EA;
         for all V in [1..maxvend] do
           ENTA[A].IAV[V] ---> RESE.IAV[1,V];
         RESE.IAV[2,A] ---> ENTA[A].IAV[A]
   end;
  end;
```

Sur cet exemple on définit maxacht exemplaires de modules Util-Acheteur et Entite-Achat, maxvend exemplaires de modules Util-Vendeur et Entite-Vente, et un exemplaire de module Réseau. Les liaisons entre modules sont décrites port à port. Ainsi le module UTIV[V] est connecté par le port EV au module ENTV[V].

Schéma de configuration associé à l'exemple ci-dessus

(figure 3) :

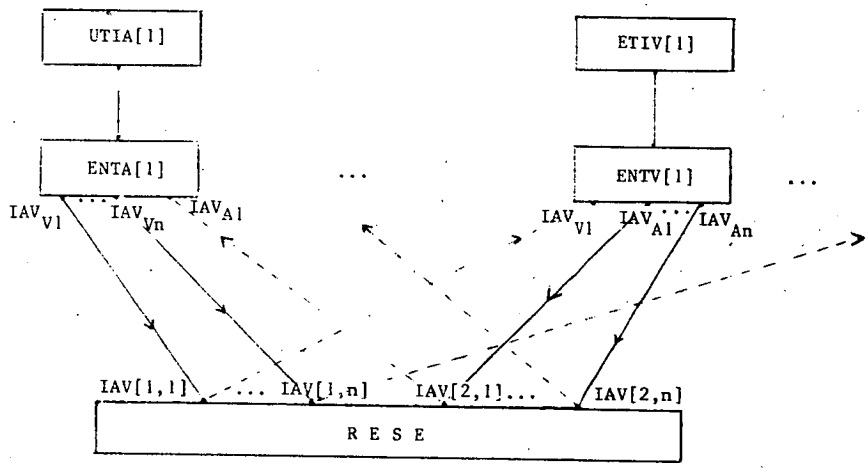

Figure 3 : Exemple de configuration de système.

UTIA[1] (resp. UTIV[1]) est un exemplaire de module Util-Acheteur (resp. Util-Vendeur)

ENTA[1] (resp. ENTV[1]) est un exemplaire de module Entité-Achat (resp. Entité-Vente)

RESE est un exemplaire de module Réseau

2.2 Construction du modèle executable

A partir des spécifications du système dans le langage que nous venons de décrire nous dérivons manuellement un programme exécutable. Le langage cible est SIMONE; il permet l'expression du parallélisme ; les modules sont traduits par des processus.

2.2.1 Expression de la communication

Un prélude standard (développé à Lannion) gère la communication entre les processus, à l'aide d'un moniteur SIMONE. Le moniteur est constitué de procédures qui traitent l'envoi et l'attente des messages.

Dans le programme de simulation l'information transmise d'un processus à l'autre est une structure de type evenement définie

comme :

```
Evenement = record t : type_du_message;
                   b : type_du_port
            end;
```

où :

-> type_du_message est une structure définissant tous les messages existants. Elle est produite à partir de la spécification des modèles de canaux.

Par exemple :

Spécification des canaux :

```
Interactions
  Service_de_vente (utilisateur,système) is

   by utilisateur : creer_prop(lot:lots);
                    creer_term(lot:lots);
   by systeme : fin_offre(lot:lots);
  end;

  Service_achat (utilisateur,systeme) is

   by utilisateur : creer_offre(lavente:ventes);
   by systeme : actif(lavente:ventes);
  end;
```

Définition de la structure type_du_message dans le programme SIMULE :

```
type_du_message = record
 case canal : integer of
  1 : (Service_de_vente : record
        case sorte : integer of
         1 : creer_prop : record lot:lots end;
         2 : creer_term : record lot:lots end;
         3 : fin_offre : record lot:lots end;
        end);

  2 : (Service_achat : record
        case sorte : integer of
         1 : creer_offre : record lavente:ventes end
         2 : actif : record lavente:ventes end;
        end);
      end;
```

-> type_du_port est une structure permettant d'associer les canaux aux modules. Cette structure est déduite des déclarations de canaux dans les en-têtes des modules.

Par exemple :

En-têtes de modules dans la spécification :

```
module Util-Vendeur (EV:service_de_vente(utilisateur);
                     TM:service_de_temporisation
                                    (demandeur));

module Entite-Vente (UTV:service_de_vente(systeme));
```

Structure type_du_port :

```
type_du_port = record
  case typeentite : integer of
    1 : (UTIV : record EV,TM : boolean end);

    2 : (ENTV : record UTV : boolean end);
```

De plus une procédure du moniteur appelée "traduction" est

déduite de la description des connexions dans la partie configuration de la spécification (cf 2.1.3). Cette procédure est appelée lors de l'envoi d'un message pour déterminer le ou les processus destinataires, ainsi que le canal utilisé par le message [JARD 82].

2.2.2 Traduction des modules

a) Forme du module

Chaque module est traduit par un processus SIMONE. Tout processus engendré par la traduction a la forme suivante :

```
Process Modl (mon_nom : integer);
 (* déclaration de variables, procédures et fonctions *)
 begin
  initialisation;
  while true do
    begin  moniteur.waitfor (mon_nom,mr);
           action
             :
             :
    end;
```

où mon_nom est l'identificateur du processus et mr l'évènement arrivant au module.

b) traduction des transitions

Une transition, spécifiée par : when condition do action ,

est traduite par : if condition then action .

- L'action est transcrite intégralement puisqu'elle est déjà en PASCAL, excepté l'envoi d'une requête.

Par exemple, l'instruction UTV.finoffre(ma_vente) se traduit :

```
ms.b.typeentite := 3;
ms.b.ENTV.UTV := true;
ms.t.interface := 1;
ms.t.Service_de_vente.sorte := 4;
ms.t.Service_de_vente.Finoffre.vente:=ma_vente;
moniteur.send(mon_nom,ms);
```

L'évènement envoyé est ms. Le contenu de la requête est dans le champ t de ms, l'expéditeur et le canal employé sont indiqués dans le champ b, et sont utilisés par la procédure "traduction" du

moniteur pour transmettre l'évènement au module désiré.

- La condition portant sur l'arrivée d'une requête qui s'écrit : <nom_de_canal>.<requête>(param.formels), est traduite par un if sur la structure "évènement" pour reconnaître le message.

Exemple : when EV.creer_prop (vente) do ... se traduit par

```
if (mr.b.UTIV.EV) and (mr.t.Service_de_vente=1) then..
```

Le premier test permet de savoir si le message reçu est arrivé par le canal EV au module UTIV , le second test permet de connaître le numéro du message reçu par ce canal (et parconséquent son nom).

2.2.3 Traduction de la configuration

Si dans la spécification la configuration est construite à l'aide d'exemplaires de modules, dans le programme SIMONE les exemplaires de processus sont créés dans le programme principal. Par exemple :

Spécification :

```
Modules
  UTIL : array [1..N] of utilisateur;
  SYST : array [1..N] of system;
  RESE : reseau;
```

Traduction en SIMONE :

```
begin (* programme *)
  for i := 1 to N do
    begin start UTIL(i);
          start SYST(i+N)
    end;
  start RESE(2*N+1);
```

Les processus UTIL, ENTIT et RESE ont été décrits dans la partie déclaration de processus du programme. Ainsi le processus décrivant "utilisateur" est déclaré par :

```
process UTIL(mon_nom : integer).
```

A l'exécution les valeurs 1 à N seront affectées aux paramètres mon_nom des processus UTIL, etc.

Le programme SIMONE ainsi obtenu est exécutable.

2.2.4 Production de traces

Une procédure 'ecriremessage' est utilisée pour produire les traces des messages échangés. A l'exécution les traces sont

rangées dans un fichier que analysons ensuite pour savoir si le déroulement des opérations est conforme au service attendu.

Suite à cette présentation du langage de spécification nous décrivons dans le prochain chapître la spécification de l'algorithme de ventes utilisé dans le système ENCHERE.

3. SPECIFICATION DE L'ALGORITHME D'ORDONANCEMENT D'ENCHERE

Ce chapitre présente la spécification du système ENCHERE. Chaque module utilise une horloge physique pour estampiller les transactions. Nous décrivons les modules nécessaires à la spécification du système.

3.1 Présentation des modules du système

Dans le système ENCHERE nous distinguons deux types de sites : les acheteurs et les vendeurs. Chaque site peut être structuré en deux niveaux :

- le niveau utilisateur, décrivant l'interface service/usager.
- le niveau système, qui décrit l'algorithme d'ordonnancement utilisé.

De plus il est nécessaire de prendre en compte le réseau. Il est représenté par un module qui précise les conditions dans lesquelles communiquent les entités de protocole.

Nous nous proposons de détailler de préférence les modules associés à un acheteur dans les paragraphes qui suivent, puisqu'on y trouve les points importants de l'algorithme.

3.2 Les modules acheteur

Un site acheteur donné est décrit par deux modules :

- Util-Acheteur, qui décrit le comportement externe de l'utilisateur
- Entite-Achat, qui exprime le déroulement de l'algorithme d'ordonnancement en ce qui concerne l'acheteur.

3.2.1 Le module Util-Acheteur

Le module Util-Acheteur n'a d'interactions qu'avec l'Entité-Achat, avec laquelle il communique via le canal 'Service_achat'. Il peut envoyer des commandes d'insertion ou de retrait pour la session, des messages de liaison ou déliaison avec un ou des vendeurs. Pour chaque proposition de vente reçue d'un vendeur, il transmet une offre (le prix qu'il propose) lorsqu'il est avisé du traitement de la vente. Après acceptation ou refus de la meilleure offre par le vendeur, l'acheteur reçoit le résultat de la vente. En cas d'annulation, il reçoit aussi une indication de l'Entité-Achat.

L'exemple suivant décrit les messages échangeables via le canal 'Service_achat' :

```
Service_achat (utilisateur,systeme) is
   by utilisateur : creeroffre (lavente:ventes);
                    creerinsert;
                    creerdesins;
                    lier (v_ens:set of sites);
                    delier(v_ens:set of sites);

   by systeme : actif (lavente:ventes);
                annulation (lavente:ventes);
                ventterm (lavente:vente);
end;
```

3.2.2 L'Entité-Achat

Le module Entite-Achat permet de décrire l'algorithme utilisé pour ordonner le choix des propositions.

Nous avons dû lui adjoindre un module Horloge pour modéliser l'horloge locale d'un site. L'Horloge interagit uniquement avec l'Entité-Achat en lui envoyant périodiquement des impulsions. L'Entité-Achat utilise cette heure pour le choix des propositions

et pour dater l'envoi de ses messages.

Le module Entite-Achat interagit d'une part avec le module Util-Acheteur, et d'autre part avec les modules Entite-Vente (cf 3.3.2) par l'intermédiaire du réseau. Les canaux utilisés sont nommés dans l'en-tête du module (cf 2.1.2) :

```
module Entite-Achat (UTA: service_achat (systeme);
                     IAV:  array [1..maxvend] of marché(acheteur);
                     HE :  service_horlogerie );
```

Nous explicitons la spécification des points intéressants de l'algorithme présentés dans la première partie.

3.2.2.1 Choix de la vente à traiter

Pour respecter les propriétés (P1) et (P2), l'acheteur doit choisir la vente à activer en fonction de la date associée à chaque vente. A la réception d'une proposition de vente l'Entité-Achat examine cette proposition . Lorsque la date de la vente est inférieure à (heure_locale - dt) la vente est mise dans un ensemble evt0. C'est à partir de cet ensemble qu'est déterminée la vente à traiter. Voici la procédure d'activation :

```
procedure activervente;
 begin ventemin := 'une vente quelconque de evt0';
       for all vente in evt0 do
 (1)    begin if (vente.date < ventemin.date) or
                    ((vente.date=ventemin.date) and
                    (vente.v<ventemin.v))
 (2)           then ventemin := vente
        end;
       evt0 := evt0-[ventemin];
 (3)   dernactiv := ventemin;
 (4)   UTA.actif(ventemin)
 end;
```

La vente sélectionnée est "dernactiv"(ligne 3), sa date est la plus petite parmi les ventes de evt0 (lignes 1-2), et

l'Util-Acheteur est avisé qu'il peut alors offrir un prix pour cette vente (ligne 4).

3.2.2.2 Traitement en cas de désynchronisation

Lorsqu'un module Entité-Achat reçoit une proposition de vente "en retard", i.e. telle que : vente.date<(heure_locale - dt), la procédure suivante est exécutée :

```
(1) if dern_activ.date>vente.date
      then begin {annuler 'dernactiv' et traiter 'vente'}
(2)         UTA.annulation(dern_activ);
            IAV[dern_activ.v].abort(moi,ma_date,dern_activ);
            dern_activ := vente;
(3)         UTA.actif(vente)
          end
      else {on peut mettre cette vente dans evt0 étant donné
             sa date}
           evt0:=evt0+[vente]
```

Si la date de la vente en cours de traitement (dern_activ) est postérieure à celle de la vente reçue (ligne 1), alors la vente dern_activ doit être annulée (ligne 2) avant d'activer la vente reçue (ligne 3).

Cette procédure met en oeuvre l'algorithme de maintien des propriétés (P1) et (P2) en cas de désynchronisation (cf 1.2.3).

3.2.2.3 Recalage des horloges

Chaque message transitant d'Entité-Vente à Entité-Achat (et réciproquement) est estampillé de l'heure locale du site expéditeur (cf 1.2.1), aussi à chaque réception de message une procédure est exécutée pour voir si la dérive entre les deux horloges n'est pas supérieure à dt (localement au site récepteur). La procédure de recalage est la suivante :

```
procedure recaler (S:sites; heure_maj:dates);
  var site:sites;
  begin if (heure_maj - ma_date > dt)  {si je retarde}
```

```
            then begin ma_date:=heure_maj; {je me mets à jour}
                 for all site in ens_lies-[S] do
                  RH[site].maj(moi,heure_maj)
                  {j'avise les autres qui sont liés avec moi}
            end
      end;
```

3.3 Les modules vendeur

Un site vendeur est structuré de la même manière qu'un site acheteur. Nous distinguons un module Util-Vendeur, décrivant le comportement externe du vendeur, et un module Entité-Vente permettant de décrire l'algorithme proprement dit.

3.3.1 Le module Util-Vendeur

L'utilisateur communique avec l'Entité-Vente en utilisant le canal 'Service_de_vente'. Les commandes envoyés par le module Util-Vendeur sont les suivantes : insertion , proposition d'un lot et fin de la vente. Il peut recevoir des messages lui signalant la fin des offres des acheteurs pour un lot, le recommencement d'une vente, le passage à la vente d'un autre lot.

3.3.2 L'Entité-Vente

Le module Entité-Vente spécifie l'algorithme vu du vendeur. Un module Horloge lui est associé, qui lui permet de dater ses ventes.

Décrivons le déroulement d'une vente : L'algorithme exécuté à la production d'une proposition de vente est le suivant :

```
(1) when UTV.creer_prop (lot) do
(2)  begin TM.mise_en _garde (1,delai_de_vente);
           for all acht in achtlies do
(3)          begin IAV[acht].prop(moi,ma_date,ma_vente);
(4)                repatt[acht]:=OFF
             end;
           nbrep := 0
     end;
```

Lorsqu'une proposition est produite par l'Util-Vendeur (ligne 1), le module Entité-Vente la diffuse aux acheteurs (ligne 3). Il arme un réveil pour borner la durée de la vente (afin de détecter et réagir à un blocage du système) (ligne 2) et attend les offres de chaque acheteur lié avec lui (ligne 4).

Quand toutes les offres lui sont parvenues, le module avise l'Util-Vendeur, qui produit alors la terminaison. Le module Entité-Vente la diffuse aux acheteurs. Dès lors vendeur et acheteurs peuvent passer à une autre vente.

Lorsqu'une vente est annulée, soit par un acheteur, qui envoie au vendeur un message "annulation", soit par le vendeur quand la durée maximum de la vente est dépassée, l'Entité-Vente envoie aux acheteurs un message "annuler" .

3.4 Le réseau

Le module réseau respecte une propriété essentielle : les messages ne se perdent pas et ne se doublent pas.

Une file d'attente pour chaque couple de modules permet le transfert des messages entre Entités-Ventes et Entités-Achats. La file est examinée à chaque top d'un réveil : si un message se trouve dans la file il est envoyé à son destinataire. Le temps imparti entre chaque top du réveil doit permettre dans la majeure partie des cas de borner le délai de transmission par un temps DT (cf 1.2.2). Les retards sont simulés en armant le réveil avec un temps plus long.

Un autre module simule la machine de communication qui traite les insertions et retraits des acheteurs et vendeurs. Ces

opérations doivent être traitées en exclusion mutuelle. Ce module le réalise de manière centralisée.

3.5 Configuration

On crée un nombre maxacht d'exemplaires des modules Entité-Achat et Util-Acheteur, et un nombre maxvend d'exemplaires des modules Util-Vendeur et Entité-Vente. On crée aussi un nombre (maxacht+maxvend) d'exemplaires de modules Horloge, un pour chaque Entité-Vente et Entité-Achat. On définit ensuite les connexions entre les modules, ainsi que nous l'avons montré dans le second chapître. Le dessin suivant montre la configuration du système

(figure 4).

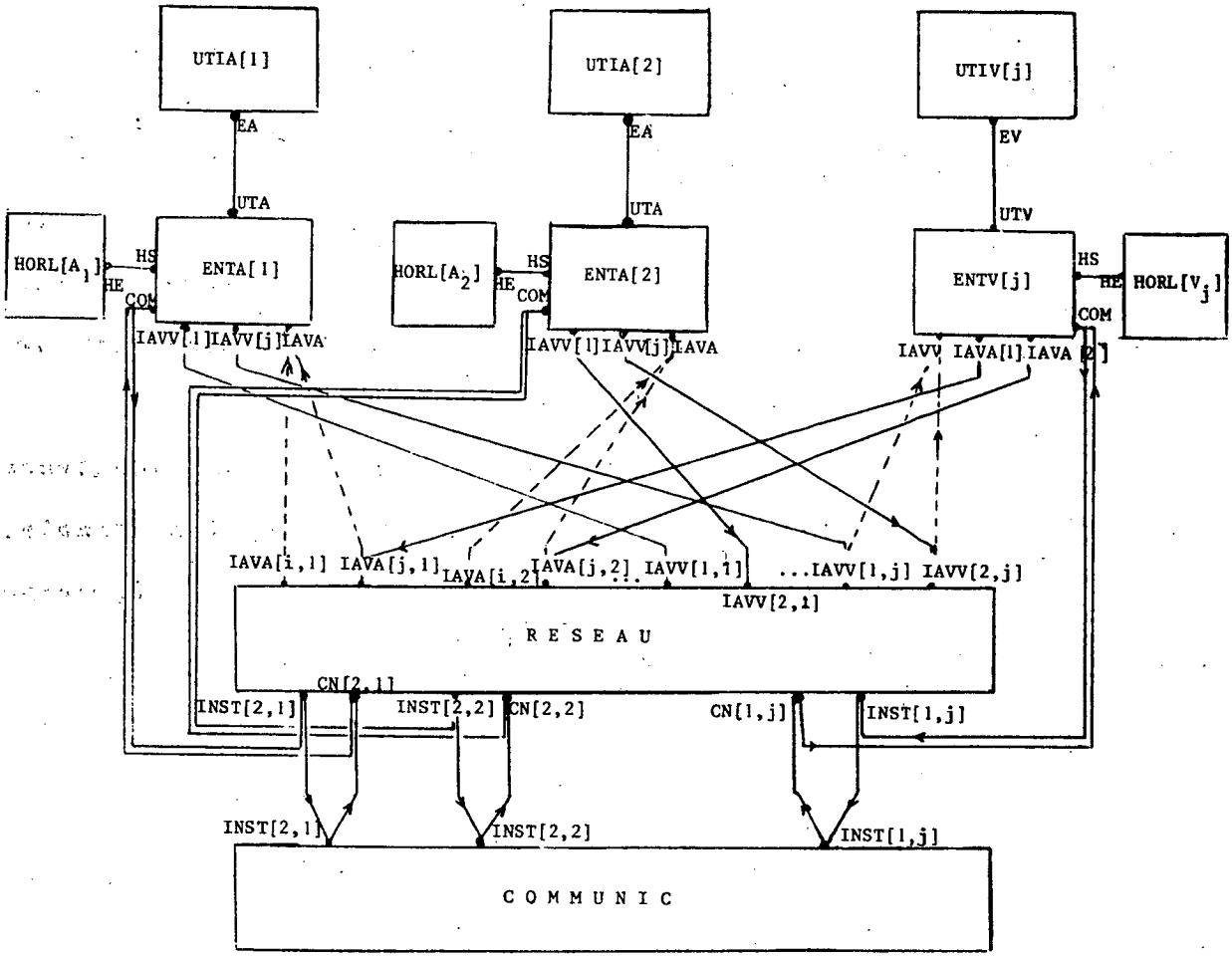

Figure 4 : Configuration du système.

La description complète des spécifications peut être trouvée dans [ECAU 83].

4. QUELQUES RESULTATS CONCERNANT L'ALGORITHME

Nous donnons ici des exemples de trace résultant de l'exécution du programme SIMONE, obtenu à partir de la spécification du système. Différentes situations ont été envisagées, qui nous permettent de montrer le déroulement de la vente d'un lot, le respect de la propriété P1 de l'équité, et le recalage des horloges en cas de désynchronisation.

Pour notre propos nous montrons quelques exemples de trace obtenus à partir d'exécutions dans une configuration simple, composée de deux acheteurs A1, A2, et deux vendeurs V1, V2, comme le représente la figure 5.

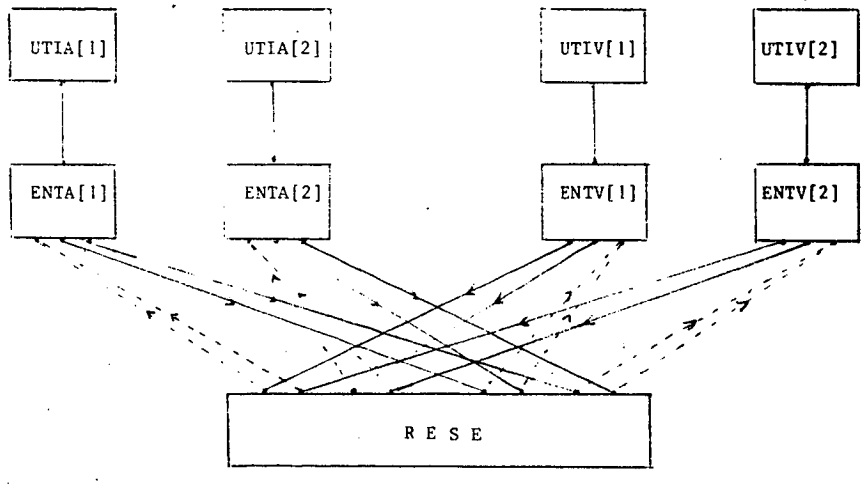

Figure 5 : Configuration utilisée dans les exemples d'exécution.

UTIA[i] et ENTA[i] sont les exemplaires de modules Util-Acheteur et Entité-Achat

UTIV[j] et ENTV[j] sont les exemplaires de modules Util-Vendeur et Entité-Vente

Notation :

Dans les exemples suivants un évènement apparaissant dans la trace est noté comme suit :

tps_simu - SITE { <- | -> } <nom_d'evenement><parametres> [-site X]

où :

SITE est le nom du site sur lequel se produit l'évènement.

-> ou <- indique s'il s'agit d'une émission (->) ou d'une réception (<-) d'évènement.

<nom_d'evenement> est le nom de l'évènement produit sur le site.

<parametres> est la liste des paramètres de l'évènement. Par exemple : nom de l'expéditeur, date d'émission, vente concernée...

[-site X] est le nom du site destinataire (s'il existe) s'il s'agit d'une émission, ou le nom du site expéditeur s'il s'agit d'une réception.

4.1 Déroulement de la vente d'un lot

L'exemple suivant présente le déroulement de la vente du lot numero 4 du vendeur V2 aux acheteurs A1 et A2 avec des hypothèses de fonctionnement normal.

```
(1) -  214 - ENTV 2 -> : PROPOSITION: V 2 DATE  122 VENTE(DATE:  122 V: 2 LOT: 4)
    -  214 - ENTV 2 -> : PROPOSITION: V 2 DATE  122 VENTE(DATE:  122 V: 2 LOT: 4)
    -  215 - ENTA 2 <- : PROPOSITION: V 2 DATE  122 VENTE(DATE:  122 V: 2 LOT: 4)
    -  217 - ENTA 1 <- : PROPOSITION: V 2 DATE  122 VENTE(DATE:  122 V: 2 LOT: 4)
(2) -  230 - ENTA 1 -> : ACTIVATION:VENTE(DATE:  122 V: 2 LOT: 4)
(4) -  232 - ENTA 1 -> : OFFRE: A 1 DATE  129 VENTE(DATE:  122 V: 2 LOT: 4)
    -  235 - ENTV 2 <- : OFFRE: A 1 DATE  129 VENTE(DATE:  122 V: 2 LOT: 4)
(3) -  248 - ENTA 2 -> : ACTIVATION:VENTE(DATE:  122 V: 2 LOT: 4)
(5) -  251 - ENTA 2 -> : OFFRE: A 2 DATE  130 VENTE(DATE:  122 V: 2 LOT: 4)
    -  254 - ENTV 2 <- : OFFRE: A 2 DATE  130 VENTE(DATE:  122 V: 2 LOT: 4)
(6) -  255 - ENTV 2 -> : TERMINAISON: V 2 DATE  142 VENTE(DATE:  122 V: 2 LOT: 4)
    -  255 - ENTV 2 -> : TERMINAISON: V 2 DATE  142 VENTE(DATE:  122 V: 2 LOT: 4)
    -  255 - ENTA 1 <- : TERMINAISON: V 2 DATE  142 VENTE(DATE:  122 V: 2 LOT: 4)
    -  259 - ENTA 2 <- : TERMINAISON: V 2 DATE  142 VENTE(DATE:  122 V: 2 LOT: 4)
```

commentaire

(1) : V2 propose son lot 4 aux acheteurs A1 etA2 à la date 122 (locale à V2).

(2) (3) : Les acheteurs A1 et A2 activent la vente de V2.

(4) : A1 fait une offre pour la vente V2.4 à la date 129 (locale à A1).

(5) : A2 fait une offre pour la vente V2.4 à la date 130 (locale à A2).

(6) : V2, ayant reçu toutes les offres, termine la vente et communique sa décision aux acheteurs, à la date 142 (locale à V2).

4.2 Illustration de la propriété P1

Nous présentons dans ce qui suit une illustration de la propriété P1 (cf 1.1). Les acheteurs A1 et A2 participent aux ventes V2.6 et V1.7 des vendeurs V2 et V1.

```
(1)- 242 - ENTV 2 -> : PROPOSITION: V 2 DATE  135 VENTE(DATE:  135 V: 2 LOT: 6)
   - 242 - ENTV 2 -> : PROPOSITION: V 2 DATE  135 VENTE(DATE:  135 V: 2 LOT: 6)
(2)- 244 - ENTA 1 <- : PROPOSITION: V 2 DATE  135 VENTE(DATE:  135 V: 2 LOT: 6)
(3)- 245 - ENTA 2 <- : PROPOSITION: V 2 DATE  135 VENTE(DATE:  135 V: 2 LOT: 6)
(4)- 259 - ENTA 2 -> : ACTIVATION:VENTE(DATE:  135 V: 2 LOT: 6)
(5)- 261 - ENTV 1 -> : PROPOSITION: V 1 DATE  145 VENTE(DATE:  145 V: 1 LOT: 7)
   - 261 - ENTV 1 -> : PROPOSITION: V 1 DATE  145 VENTE(DATE:  145 V: 1 LOT: 7)
   - 263 - ENTA 2 -> : OFFRE: A 2 DATE  142 VENTE(DATE:  135 V: 2 LOT: 6)
(6)- 264 - ENTA 1 <- : PROPOSITION: V 1 DATE  145 VENTE(DATE:  145 V: 1 LOT: 7)
   - 265 - ENTA 2 <- : PROPOSITION: V 1 DATE  145 VENTE(DATE:  145 V: 1 LOT: 7)
   - 265 - ENTV 2 <- : OFFRE: A 2 DATE  142 VENTE(DATE:  135 V: 2 LOT: 6)
(8)- 266 - ENTA 1 -> : ACTIVATION:VENTE(DATE:  135 V: 2 LOT: 6)
   - 267 - ENTA 1 -> : OFFRE: A 1 DATE  142 VENTE(DATE:  135 V: 2 LOT: 6)
   - 268 - ENTV 2 <- : OFFRE: A 1 DATE  142 VENTE(DATE:  135 V: 2 LOT: 6)
(9)- 269 - ENTV 2 -> : TERMINAISON: V 2 DATE  148 VENTE(DATE:  135 V: 2 LOT: 6)
   - 269 - ENTV 2 -> : TERMINAISON: V 2 DATE  148 VENTE(DATE:  135 V: 2 LOT: 6)
(10)- 273 - ENTA 2 <- : TERMINAISON: V 2 DATE  148 VENTE(DATE:  135 V: 2 LOT: 6)
(11)- 273 - ENTA 1 <- : TERMINAISON: V 2 DATE  148 VENTE(DATE:  135 V: 2 LOT: 6)
(12)- 283 - ENTA 2 -> : ACTIVATION:VENTE(DATE:  145 V: 1 LOT: 7)
   - 284 - ENTA 2 -> : OFFRE: A 2 DATE  152 VENTE(DATE:  145 V: 1 LOT: 7)
   - 284 - ENTV 1 <- : OFFRE: A 2 DATE  152 VENTE(DATE:  145 V: 1 LOT: 7)
(13)- 286 - ENTA 1 -> : ACTIVATION:VENTE(DATE:  145 V: 1 LOT: 7)
   - 288 - ENTA 1 -> : OFFRE: A 1 DATE  152 VENTE(DATE:  145 V: 1 LOT: 7)
   - 293 - ENTV 1 <- : OFFRE: A 1 DATE  152 VENTE(DATE:  145 V: 1 LOT: 7)
   - 295 - ENTV 2 -> : PROPOSITION: V 2 DATE  161 VENTE(DATE:  161 V: 2 LOT: 7)
   - 296 - ENTV 1 -> : TERMINAISON: V 1 DATE  155 VENTE(DATE:  145 V: 1 LOT: 7)
   - 296 - ENTV 1 -> : TERMINAISON: V 1 DATE  155 VENTE(DATE:  145 V: 1 LOT: 7)
   - 297 - ENTA 2 <- : TERMINAISON: V 1 DATE  155 VENTE(DATE:  145 V: 1 LOT: 7)
   - 302 - ENTA 1 <- : TERMINAISON: V 1 DATE  155 VENTE(DATE:  145 V: 1 LOT: 7)
   - 302 - ENTA 2 <- : PROPOSITION: V 2 DATE  161 VENTE(DATE:  161 V: 2 LOT: 7)
```

commentaire

(6) : A1 a reçu deux propositions de vente de V1 et V2.

(8) : il choisit d'activer la vente V2.6, parce que sa date est la plus petite.

(4) : La vente activée par V2 est aussi V2.6

(12) : Après avoir traité V2.6 les deux acheteurs traitent V1.7 .

4.3 Illustration du recalage des horloges en cas de désynchronisation

Lorsqu'un site S1 "retarde" par rapport à un site S2, S1 recale son horloge avec l'heure reçue de S2, et diffuse sa nouvelle heure aux sites avec lesquels il est lié (cf 1.3).

Voici une trace d'exécution où cet algorithme est appliqué : Le site V2 est lié aux sites A1 et A2 et le site V1 est lié au site A2. La dérive maximale entre deux sites est dt=6.

```
   -  167 - ENTA 2 -> : OFFRE: A 2 DATE   97 VENTE(DATE:   81 V: 2 LOT: 4)
   -  169 - ENTV 1 -> : PROPOSITION: V 1 DATE   83 VENTE(DATE:   83 V: 1 LOT: 5)
(1)-  171 - ENTV 2 <- : OFFRE: A 2 DATE   97 VENTE(DATE:   81 V: 2 LOT: 4)
(2)-  171 - ENTV 2 -> : MISE A JOUR HORAIRE: EXP 2 DATE   97-SITE A 1
(3)-  173 - ENTA 1 <- : MISE A JOUR HORAIRE: EXP 2 DATE   97
(4)-  173 - ENTA 1 -> : MISE A JOUR HORAIRE: EXP 1 DATE   97-SITE V 1
   -  174 - ENTA 1 -> : ACTIVATION:VENTE(DATE:   81 V: 2 LOT: 4)
   -  175 - ENTA 1 -> : OFFRE: A 1 DATE   97 VENTE(DATE:   81 V: 2 LOT: 4)
   -  175 - ENTV 2 <- : OFFRE: A 1 DATE   98 VENTE(DATE:   81 V: 2 LOT: 4)
   -  175 - ENTA 1 <- : PROPOSITION: V 1 DATE   83 VENTE(DATE:   83 V: 1 LOT: 5)
   -  177 - ENTV 1 <- : MISE A JOUR HORAIRE: EXP 1 DATE   97
```

Commentaire

(1) : V2 reçoit une offre de A2, son horloge retarde.

(2) Il la recale et avise l'acheteur A1 avec lequel il est lié.

(3) : A1 reçoit l'avis de mise à jour. Il retarde également.

(4) : Il recale son horloge puis diffuse sa nouvelle heure aux sites voisins (i.e. V1 dans notre cas).

Nous constatons, sur des configurations simplifiées, que les traces d'exécution du système produites par simulation sont

conformes au service attendu.

D'autres simulations nous ont permis de valider les différentes propriétés d'équité en cas de désynchronisation. Cependant une validation complète de l'algorithme suppose la prise en compte de configuration beaucoup plus étendues, ce qui a pour effet de produire un fichier de traces très important. Comme l'analyse des traces est pour l'instant manuelle, l'exploitation de ce fichier s'avère irréaliste pour de telles configurations. Un outil de vérification automatique des traces devient à notre avis indispensable.

5. CONCLUSION

Nous avons présenté dans cet article l'application d'un outil de validation à un algorithme distribué d'ordonnancement équitable mis en oeuvre dans ENCHERE.

La première étape de la démarche employée est la spécification de l'algorithme. Cette phase du travail nous a permis de donner une définition précise de l'algorithme. Nous avons pu notamment expliciter les "cas limites", rarement prévus dans une première présentation d'un algorithme.

La seconde étape est la traduction des spécifications en un programme de simulation dont le résultat de l'exécution est un fichier de traces, chaque élément de la trace étant un évènement observable du système. Pour les raisons évoquées précédemment cette validation n'a été faite que pour des configurations restreintes. Pour qu'elle devienne vraiment significative (i.e. tenant compte de configurations importantes), l'étude et la réalisation d'un outil pour la vérification automatique des propriétés du système à partir des traces obtenues est l'objet de nos préoccupations actuelles.

REMERCIEMENTS

Je remercie J-P. Banâtre et M. Banâtre de l'IRISA, C. Jard et J-M. Pitié du CNET Lannion pour l'aide et les conseils qu'ils m'ont apportés au cours de ce travail.

BIBLIOGRAPHIE

[BANA 82] BANATRE M., LAPALME G.
ENCHERE : A Distributed Auction Bidding System.
3rd Int. Conf. Distributed Computing System, Miami (Octobre 1982), pp.833-837.

[BANA 83] BANATRE M.
Le système ENCHERE: une Expérience dans la Conception et la Réalisation d'un système réparti.
Rapport Interne IRISA (Janvier 1984).

[BEZI 76] BEZIVIN J., KAUBISCH W.H., LEROY A., NEBUT J.L., RANNOU R.
Manuel de référence SIMONE.
Publication IRIA-SFER (novembre 1976).

[ECAU 83] ECAULT C.
Spécification de l'algorithme d'ordonnancement équitable des transactions dans le système ENCHERE.
Rapport technique IRISA (Septembre 1983).

[JARD 81] JARD C.
Définition d'un modèle de simulation pour la validation de protocoles.
CNET Lannion,département EVP,note technique LAA/SLC/49 (juin 1981).

[JARD 83] JARD C., BOCHMANN G.V.
An approach to testing specifications.
Proc. of ACM SIGPLAN Symp. on High-Level Debugging, Pacific Grove (mars 1983).

[JARD 82] JARD C.
Spécification et validation d'un algorithme distribué d'exclusion mutuelle. Mise en oeuvre de la simulation : Méthode et résultats.
CNET Lannion,dep.EVP, note technique LAA/SLC/93 (juillet 1982).

[LAMP 78] LAMPORT L.
Time, Clocks and the Ordering of Events in a Distributed System.
CACM 21,7 (juillet 1978).

[RICA 81] RICART G., AGRAWALA A.
An Optimal Algorithm for Mutual Exclusion in Computer Networks.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique