Spécification et validation d'un protocole de communication adapté au temps réel

Philippe F.R. Belmans, Omar Drissi-Kaitouni

INRIA

UNITÉ DE RECHERCHE
INRIA-RENNES

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
BP105
78153 Le Chesnay Cedex
France
Tél (1) 39 63 55 11

Rapports de Recherche

N° 594

SPÉCIFICATION ET VALIDATION D'UN PROTOCOLE DE COMMUNICATION ADAPTÉ AU TEMPS RÉEL

Philippe F. R. BELMANS
Omar DRISSI KAITOUNI

Décembre 1986

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

Publication Interne n° 324
Décembre 1986
42 pages

SPECIFICATION ET VALIDATION D'UN PROTOCOLE DE COMMUNICATION ADAPTE AU TEMPS – REEL

SPECIFICATION AND VALIDATION OF REAL-TIME COMMUNICATION PROTOCOL

Philippe F. R. BELMANS
Omar DRISSI KAITOUNI

IRISA
Campus de Beaulieu
35042 Rennes Cedex

Résumé

Dans cet article, nous allons nous intéresser à la spécification et à la validation d'un protocole de communication destiné à assurer les échanges d'informations nécessaires dans les applications robotiques, en respectant les contraintes temps réel. Dans un premier temps, nous définirons ces contraintes et les impératifs s'y rattachant, ensuite, nous définirons nos besoins en tant qu'utilisateurs. A partir de ces hypothèses, nous présenterons une solution qui nous semble optimale, puis affinerons sa spécification et procèderons vérification de sa validité logique au moyen d'un logiciel de simulation (VEDA).

Abstract

This paper deals with a communication protocol which aims data exchanges involved in robotics applications, with hard strain real time background. First the protocole is described from the user's requirements; hence a logical model using communicating finite state machines decribes the behavior of the protocole. Simulations carried out on this model with a software tool called VEDA <point out errors and lack in specification and allow to verify the right behavior in different situations.

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE
(L.A. 227)
UNIVERSITÉ DE RENNES 1 I.N.S.A. DE RENNES

INSTITUT NATIONAL DE RECHERCHE
EN INFORMATIQUE ET EN AUTOMATIQUE
(LABORATOIRE DE RENNES)

Présentation du protocole de communication

1 contraintes temps – réel

Les applications robotiques constituent une famille particulière dans les systèmes de commande en temps réel rencontrés en automatique; rappelons qu'un système de commande est dit temps réel lorsque les dynamiques de l'environnement qu'il pilote sont indépendantes des siennes. Une caractéristique fondamentale, appelée temps de réponse du système à une sollicitation donnée, permet d'évaluer le comportement de celui – ci, et de vérifier l'adéquation commande/environnement. Le temps de réponse est le laps de temps écoulé entre le moment ou une sollicitation apparaît et le moment où celle – ci est prise en compte par le système pour le calcul de la commande. Le temps de réponse du système doit être adapté à celui de l'environnement pour assurer la stabilité de la commande; ceci se traduit par un ensemble de contraintes temporelles à respecter; on distingue deux familles de systèmes temps – réel:

- les systèmes **Temps Réel à Contraintes Relatives, ou TRCR** : dans de tels systèmes, on impose une loi de distribution des délais, la probabilité de dépasser un délai donné est bornée par un majorant connu; dans de tels systèmes, la probabilité d'un temps de réponse infini est non nulle; ceci est acceptable dans certaines applications telles que dans les réseaux bureautiques (ETHERNET), en revanche, dans le cadre de la commande de processus ceci est inadmissible, car un délai se traduit par un retard dans la boucle d'asservissement, et par un cortège de conséquences catastrophiques (instabilité, erreurs...);

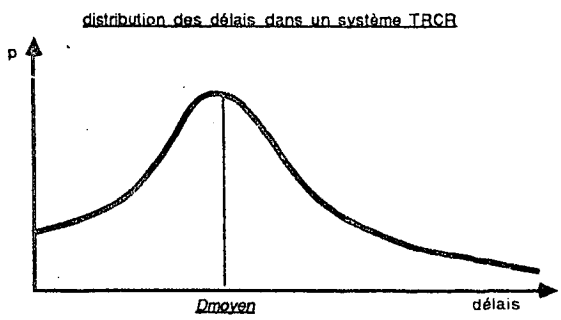

Figure 1. systèmes TRCR

- les systèmes **Temps Réel à Contraintes Strictes, ou TRCS** : dans de tels systèmes, les délais ne peuvent dépasser une limite donnée, soit la probabilité de dépasser cette limite doit être nulle; ces impératifs permettent d'assurer une commande adéquate; la stabilité du système, la précision de son pilotage, ainsi que sa sécurité de fonctionnement sont liées au choix de ce majorant; nos applications entrent dans ce cadre des systèmes TCRS.

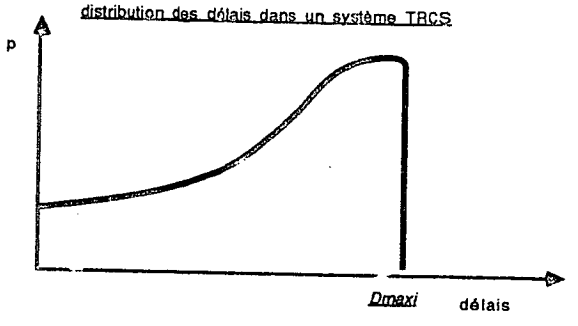

Figure 2. systèmes TRCS

2 besoins utilisateurs

Le système de commande robotique /BOR86/ repose sur l'utilisation de structures distribuées pour trois raisons principales :

- **modularité** : qui facilite la mise en oeuvre de systèmes robotiques, leurs extension ou leur réorganisation et leur maintenance;
- **flexibilité** : qui permet la reconfiguration dynamique des systèmes;
- **complexité** des algorithmes de commande nécessitant la répartition de la puissance de calcul.

Ces impératifs conduisent à préferer une structure de commande plus souple et plus équitable que la pyramide classique en commande distribuée qui n'est pas "flexible". Un tel système est constitué :

- d'un ensemble de modules : chaque module est une entité autonome comprenant des processeurs propres et les différents algorithmes requis pour le schéma de commande;

- d'un exécutif temps réel multitâche monoprocesseur /BEL85/ : cet exécutif est dupliqué dans chacun des modules et assure une gestion cohérente des différents objets (tâches, sémaphores, événements...) utilisés dans les algorithmes de commande;
- d'un système de communication, ou réseau de communication /SIM87/ : celui – ci assure le transit des informations entre modules, qui permettent la coopération dans tout le système robotique.

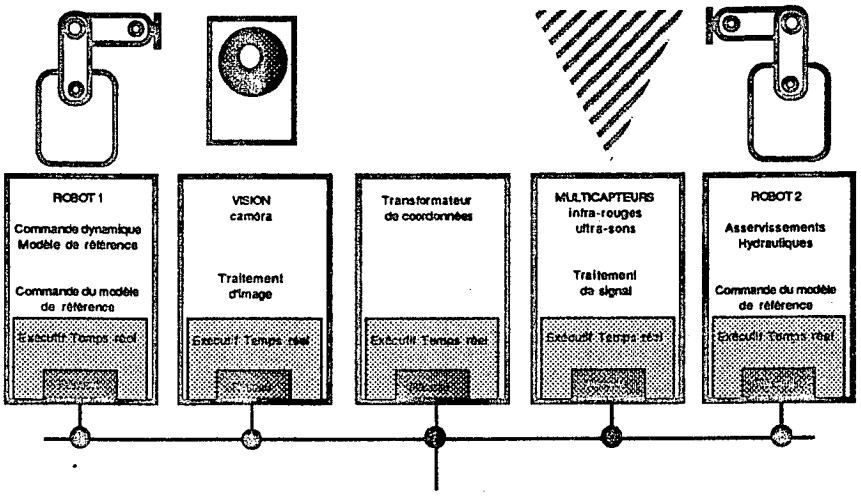

Figure 3. schéma d'un système de commande

Concrètement, pour respecter les besoins utilisateurs qui ont été exprimés, le réseau de communication devra respecter les quatre critères suivants :

- Débit garanti : le débit garanti concerne la capacité en informations qu'il est possible d'acheminer sur le réseau; typiquement, celui – ci est exprimé, en général, en Mégabits par seconde (Mb/s).
- Promptitude : les délais d'accès et les temps de réponse sont finis, majorables et connus à priori; cette condition est indispensable pour le pilotage de systèmes temps – réel où le temps joue un rôle déterminant dans la commande; tout retard en matière de transmission peut se traduire par une dégradation importante de la commande, voire une défaillance totale du système commandé; un retard existe dans la commande du système et, par conséquent, ce dernier peut devenir instable dans certains cas. Dans les applications temps – réel qui nous intéressent, ce temps d'accès doit être inférieur à la milliseconde; rappelons que les tâches d'asservissement ont une période de l'ordre de 1 ms.

- Robustesse : le transit des informations sur le réseau doit être un service fiable, s'effectuant avec une certaine sécurité; il est donc nécessaire de définir, pour la voie de communication, les comportements fautifs tolérables, le nombre maximum de fautes admissible pour un temps écoulé donné, le délai maximum de recouvrement du système en cas de défectuosité, la probabilité de mise en échec des différents constituants du système.

- Flexibilité : le système doit être évolutif; une reconfiguration de l'application, par adjonction d'un nouveau module par exemple, ne doit pas remettre en question la structure du système initial; de plus, le réseau de communication doit présenter une grande versatilité satisfaisant les besoins rencontrés dans les applications robotiques temps-réel, il ne doit pas être surdimensionné : plus un système informatique est ouvert, plus il nécessite du logiciel qui affecte les performances temporelles.

Ces quatre critères permettent de définir l'adéquation d'un réseau à notre problème de communication. Dans une partie de la littérature disponible consacrée aux réseaux, le déterminisme du protocole utilisé apparaît comme étant un critère suffisant pour assurer la promptitude du réseau. Rappelons qu'un réseau de communication est déterministe lorsqu'on peut déduire, connaissant les entrées appliquées au système, l'état futur du système, si on excepte les pannes et erreurs; ceci permet de garantir que le nombre de transitions d'état séparant la soumission d'un message et sa transmission réussie est fini et borné supérieurement; cela ne permet absolument pas d'affirmer que les contraintes de promptitude seront satisfaites /LEL84/. En revanche, le détermisme apparaît comme une condition nécessaire à la promptitude.

Le contexte de l'étude étant présenté, nous alons maintenant aborder le protocole de communication de bas niveau qui regira l'accès au réseau.

3 spécification du protocole de base

Nous allons présenter le protocole qui va servir de base à notre étude; nous allons d'abord nous affranchir des erreurs qui se manifestent ordinairement dans les systèmes réels; celui-ci sera donc considéré comme parfait et le protocole comme idéal.

3.1 protocole idéal

A partir d'un travail de synthèse et de recherche /BEL86/, nous nous sommes orientés vers une solution reposant sur un protocole déterministe, à priorités dynamiques à échéances; le protocole de communication repose sur un algorithme décentralisé.

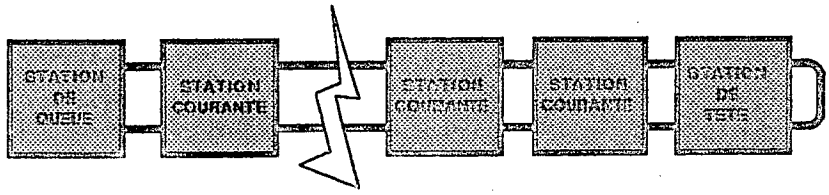

Figure 4. topologie du réseau

La voie de communication est constituée par un bus monodirectionnel en épingle /MIC82a/ /MIC82b/, constitué par une fibre optique assurant une liaison point à point entre les différentes stations du réseau; on distingue deux portions de voies:

- **la voie d'émission** sur laquelle s'effectue l'allocation de la ligne pour les différentes stations et sur laquelle sont émis les messages ;
- **la voie de réception** sur laquelle les différentes stations reçoivent leurs messages;

la connexion entre ces deux portions est assurée par la station de queue dont nous verons le rôle plus en détail.

Chacune des stations est dotée de quatre ports monodirectionnels :

- **une entrée en émission PEE** qui reçoit l'information incidente venant de l'amont sur la voie d'émission,
- **une sortie en émission PSE** qui assure la transmission vers l'aval du signal incident, ou permet le cas échéant à la station de transmettre son message si elle est prioritaire,
- **une entrée en réception PER** qui reçoit le signal venant de l'amont de la voie de réception, et donc qui permet à la station destinataire de lire le message qui lui est envoyé,
- **une sortie en réception PSR** qui transmet le signal reçu sur l'entrée reçu sur PER.

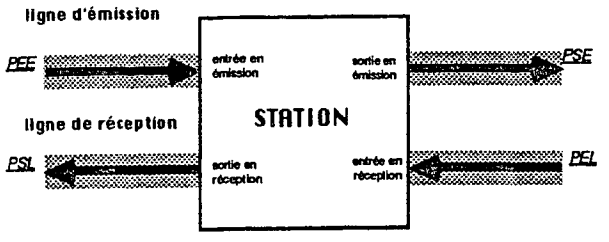

Figure 5. schéma d'un station

Dans le système, toutes les stations sont identiques, cependant, en cours de fonctionnement, on distingue suivant leur rôle, trois groupes de stations :

- **la station de tête,** dont le rôle est d'amorcer le cycle d'allocation – émission et de veiller a son bon déroulement; toute station peut devenir station de tête des lors qu'elle ne perçoit aucune activité sur sa ligne d'entrée en émission; cette particularité se manifeste au niveau protocole, en ce sens qu'il faut prévoir un algorithme spécifique pour celle – ci, ainsi qu'au niveau topologique;
- **les stations intermédiaires** assurent la continuité de la voie de communication, pour l'aspect topologique; quand à l'aspect protocolaire, l'algorithme régissant l'accès au réseau est le même pour toutes ces stations.
- **La station de queue** qui connecte l'extremité finale de la ligne d'émission au debut de la ligne de réception, pour l'aspect topologique, en reliant PSE avec PER; toute station peut devenir station de queue dès lors qu'elle ne détecte aucune activité sur sa ligne d'entrée en reception. Néanmoins, pour ce qui concerne l'aspect protocolaire, cette station utilise le même algorithme que les stations intermediaires.

Les informations qui circulent sur le réseau sont conditionnées sous forme de trame; une trame comprend :

- un drapeau START qui indique le début d'un message;
- un champ qui contient la PRIORITE du message;
- un champ ADRESSE du DESTinataire;
- un champ ADRESSE de l' EXPediteur du message;
- les DONnées utiles;

- un code d'erreur type CRC;
- un drapeau STOP pour indiquer la fin du message;

Figure 6. constitution d'une trame

A tout moment règne une activité sur la ligne : soit un message est en cours de transfert, soit un signal constitué de bits de remplissage; ceci permet d'assurer à tout moment la synchronisation binaire de l'electronique des stations, pour compenser, entre autres, les effets de d'erive thermique des circuits.

A la mise sous tension, toute station est station de tête, puisqu'aucune activité n' est observale sur les entrées en émission, donc chaque station va emettre des bits de remplissage; dès qu'une station détecte de l'activité sur son entrée en émission PEE (en fait des bits de remplissage), elle cesse son émission et devient station intermédiaire; ce phénomène se propageant à toutes les stations, une seule station est reconnue station de tête : c'est celle (sous tension) la plus en amont de la ligne d'émission. Le processus est tout à fait analogue pour la détermination de la station de queue; lorsqu'une station ne détecte aucune activité sur son entrée en réception PER, alors elle boucle sa sortie en émission sur son entrée en réception; la boucle est ouverte dès qu'une activité incidente sur la ligne a l'amont de l'entrée en réception PER est détectée; ce phénomène se propageant à toutes les stations, une seule station est élue station de queue : c'est celle (sous tension) la plus en aval de la ligne d'emission, ou la plus en amont de la ligne de réception.

A l'issue de cette phase d'initialisation, la topologie du réseau est configurée et le réseau peut entrer en activité normale; il est à noter que ce processus peut intervenir à tout moment au cours de l'activité du réseau, afin de remédier à des situations telles que rupture locale de la fibre, mise hors tension d'une station, fonctionnement anormal d'une station...

Cette initialisation étant faite, les échanges peuvent alors se produire. A un instant donné la station de tête émet une trame; la partie informations utiles de cette trame est significative si la station de tête a effectivement des informations à communiquer; sinon, elle ne contient aucune information et la trame est alors appelée **trame vide** et le message **message vide**. Lorsque le

message arrive sur le port d'entrée en émission station PEE, celle ci détecte le drapeau START et entre en phase de compétition;

- si la station n'a aucun message à émettre, alors elle laisse circuler le message, en assurant la continuité de la voie de communication; physiquement, il y a répetition du signal avec retard de 1 bit par station; cet aspect n'est néanmoins pas significatif pour la spécification logique du protocole.
- si la station a un message à émettre, alors, elle compare au vol, bit à bit, la priorité du message qu'elle souhaite émettre avec celle du message incident sur son port d'entrée en émission PEE; pendant la phase de comparaison au vol, dès qu'un bit de la priorité détermine le message de la station comme la plus prioritaire, le port d'entrée est désactivé, l'amont du réseau est isolé. La station émet alors sur l'aval du réseau le reste du champ de la priorité de son message et les autres constituants de la trame. Pendant la phase de comparaison au vol, si le message de la station est mons prioritaire, alors la station laisse circuler la suite du message amont.
- Par ce mécanisme, la station ayant le message le plus prioritaire obtient le droit exclusif d'accès au médium de communication. Les stations se trouvant à l'aval de celle-ci savent qu'elles ne sont pas prioritaires, puiqu'elles ont été imédiatement éconduites au cours de la phas de comparaison des priorités au vol; en ce qui concerne la station élue et toutes les postulantes à son amont, elle sont fixées sur leur sort lorsqu'elles lisent le champ expéditeur du message en transmission.
- Les stations ordinaires retournent vers leur état de repos, lorsqu'elles ont lu les adresses d'émission et de réception; la station destinataire stocke le message et vérifie son CRC; lorsqu'elle recôit le drapeau STOP qui marque la fin d'emission, elle passe dans son état de repos. Lorsque la station de tête a reçu le drapeau STOP, elle rétère ensuite le cycle d'allocation.

3.2 protocole réel

3.2.1 erreurs de transmission

Le protocole que nous venons de présenter était supposé idéal, en ce sens que le réseau était supposé exempt d'erreur. Cependant, dans la réalité des erreurs de transmission existent et se manifestent par des information erronées. Même si on peut en géeneral estimer correctement la probabilité de ces erreurs, et même si on peut les rendre aussi minimes que possible, il n'en demeure pas moins qu'elle doivent entrer en ligne de compte dans la conception du protocole; si le protocole n'est pas robuste, une erreur peut conduire à un blocage partiel, voire total du système.

L'utilisation d'une technologie à fibre optique impose l'utilisation d'un code de transmission des

informations équilibré, i.e. que le signal transmis comporte en moyenne autant de bits 0 que de bits 1; le code retenu est le 4B6B où on code tout groupe de 4 bits en un des 20 groupes de 6 bits équiibrés. Lorqu'au sein de l'information existe un seul bit erroné au plus par groupe de 6, alors le code est désiquilibré et il est alors possible de détecter immédiatement l'erreur avant que le message ne soit complètement émis, par le destinataire du message d'une part, par la station de tête d'autre part.

Cependant, dans d'autre cas, deux bits erronés peuvent se compenser en donnant une séquence équilibrée et la faute ne peut être détectée comme précédemment:

- si une telle erreur se produit dans la partie utile de la trame, soit dans les adresses, soit dans les données elles-même, il est en général possible de la detecter par la non concordance du CRC du message; si en plus de celà la comparaison du CRC n'a pas permis de déceler une erreur existante, les données peuvent être fausses, ce qui est a considérer dans la commande, ou les adresses fausses, donc des vérifications sont à prevoir dans le protocole de la couche supérieure; de telles erreurs ne sont détectables qu'après réception intégrale du message.
- si une telle erreur se produit en dégénérant le drapeau START ou STOP, on peut arriver a des blocages de toutes les stations sans précaution supplémentaire; pour y remedier, il faut utiliser des chiens de garde: la station de tête ayant déclenché une phase d'allocation-émission signalée par l'envoi d'un drapeau START sur PSE ,si au bout d'un temps de garde donné, elle n'a pas reçu le drapeau START sur PER, alors il y a erreur; un mécanisme analogue permet de traîter la dégénérescence du drapeau STOP.

Suivant les cas, les erreurs détecteés seront recouvrées à différents niveaux :

- pour celles nécessaitant la transmission intégrale du message, les procédures de recouvrement concernent les couches supérieures du protocole et conditionnent l'ordonnancement des messages de la station émétrice et réceptrice, ceci ne fait pas l'objet de la présente étude;
- pour les autres, il est nécessaire de prévoir des phases de réinitialisation et de resynchronisation des stations; dans ce cas la station de tête ayant détecté l'erreur envoie un message particulier ne comportant qu'un drapeau RESET; lorqu'une station le reçoit, elle passe alors dans l'état de repos et se resynchronise si nécessaire avec l'amont du réseau; lorsque la station de tête récupère le drapeau RESET sur son entrée PER, alors il est possible d'amorcer un nouveau cycle d'allocation-émission; si au bout d'un temps de garde donné le drapeau n'a pas été reçu par la station de tête, alors celle-ci réitère la procédure jusqu'à sa réussite.

3.2.2 prise en compte du temps

Par soucis de simplification, nous avons négligé la prise en compte du temps; celui-ci intervient à plusieurs niveau:

- transmission : la propagation du signal étant limité par la vitesse de la lumière, il y a retard entre l'apparition d'un evénement quelconque et son observation; à celà, il faut ajouter les retards introduits par les répetiteurs optiques pour chacune des stations; il faut également tenir compte du fait que l'émission d'un message dure un temps donné qui est fonction de la quantité d'information et de la bande passante utilisée;
- réaction : en toute rigueur, il faut tenir compte des temps de commutations des circuits électroniques; les changements d'états, contrairement au modèle idéal, ne se font pas à vitesse infinie; cependant, les grandeurs chronologiques étant d'un ordre très largement inférieur à celles intervenant dans la transmission, nous ne les prendrons pas en compte.

Numériquement, en tenant compte de besoins réels, la longueur des messages est typiquement:

start : 1 octet
priorité : 12 octets
adr.dest : 14 octets
adr.emet : 4 octets
données : 24 octets pour les messages courants
: 0 octet pour le message vide
crc : 2 octets
stop : 1 octet

Le débit brut utile après codage est de l'ordre de 20 mbps. Chaque station et chaque fibre introduit un retard estimé à deux bits en temps. Pour la simulation, l'unité de temps logique qui a été retenue correspond à la transmission de deux bits.

Nous venons de décrire brièvement le protocole de communication de bas niveau d'un réseau local temps réel; nous allons maintenant exprimer celui-ci sous forme d'automates communicants, le traduire en FDT-ESTELLE, puis procéder a sa simulation; celle-ci nous permettra d'une part de lever les indéterminations et donc de préciser son fonctionnement, et d'autre part de détecter les incohérences non détectées à la conception, pour aboutir à la formulation finale d'un protocole sain, exempt de toute anomalie.

Validation du protocole de communication

L'objectif principal de cette partie est de valider le protocole de communication spécifié dans la première partie; Valider un protocole consiste à vérifier la conformité de son comportement vis-à-vis des services attendus.

2 généralités

2.1 état de l'art

Toute vérification d'un système distribué a pour but d'établir la conformité entre le comportement du système réalisé et son comportement souhaité. On distingue trois méthodes de vérification /GRO85/:

- **La preuve théorique** : elle utilise la modélisation des systèmes et repose sur la déduction logique (règles d'inférences, systèmes de réecritures, systèmes de transitions, etc...)
- **La preuve exhaustive** : elle consiste à construire le graphe de tous les états possibles du système ;par un parcours éxsautif de tous les chemins du graphe, on vérifie les propriétés du service attendu. Cette méthode devient impraticable lorsque le graphe d'états est important (trés grand en nombre d'etats), ce qui est le cas pour la plupart des systèmes distribués. Cependant, elle reste efficace quant à la qualité du résulat obtenu. CESAR /QUE82/ (Certification, Evaluation et Spécification des Applications Réparties) est un des outils de validation existant utilisant cette méthode.
- **La preuve par simulation** : A partir d'une description formelle du protocole, la simulation permet de "faire vivre" ce protocole en permettant le jeu simultané de toutes ses variables. Au cours de la simulation , le comportement dans le temps du protocole est testé et la vérification consiste à s'assurer de la conformite de ce comportement vis à vis de la spécification du service. Mais il ne faut pas trop attendre de la simulation car elle ne donne jammais une réponse exacte à un probléme posé: elle ne fait que dégager les tendances générales du comportement du protocole. Donc elle n'apporte qu'un degré de confiance relatif dans le système réalisé.

Vu la complexité de notre protocole de communication, la dernière méthode nous paraît plus adaptée par rapport aux autres méthodes, pour le valider. Pour celà, nous utiliserons VEDA (Validation et Evaluation d'Algorithmes Distribués) qui est un outil de validation des systèmes distribués par simulation d'eveloppé au C.N.E.T. Lannion /GRO85/.

A partir d'une description du protocole en FDT – E (Version 1 d'ESTELLE) /EST../, VEDA génère un programme dont l'exécution simule le comportement du protocole.

Pour Vérifier les propriétés du protocole réalisé :

On peut construire la trace d'exécution du système; on s'assure ensuite de la conformité de celle – ci (verification manuelle);

on peut utiliser un mécanisme d'observation défini dans VEDA. Il permet de visualiser ce qui se passe dans le système et vérifier les propriétés du système.

2.2 présentation du langage FDT – E

Nous suggérons de souligner quelques caractéristiques essentielles du langage. Le langage FDT – E est conçu pour la spécification d'algorithmes distribués a été défini par l'ISO et le CCITT. Pour une description détaillée du langage voir /EST.../

Ce langage repose sur le concépte d'automate non déterministe communicant. Un automate d'etat fini communicant est défini par le sextuplet $<Q, E, A, T, S, q_0>$ où

- **Q** : est un ensemble fini d'etats
- **E** : est un ensemble fini d'entrées (prédicats)
- **A** : est un ensemble fini de sorties (actions)
- **T** : est une fonction de transition d'etat définie de **Q x E** vers **Q**
- **S** : est une fonction de sortie définie de **Q x A** vers **Q**
- q_0 : étant l'etat initial ($q_0 \in Q$)

T et S expriment le comportement de l'automate. La réception d'un message est considérée comme un prédicat d'entrée qui est vrai lorque ce message arrive sur le port d'entrée du processus. L'émission d'un message est considérée comme une action de sortie, elle provoque l'envoi du message sur le port de sortie correspendant.

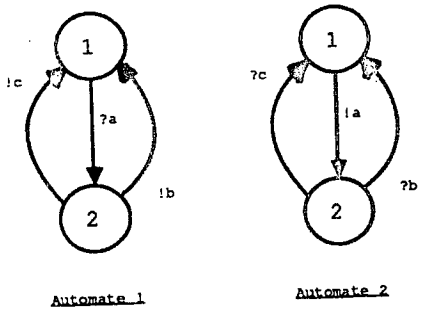

Figure 7. exemple de deux automates

En estelle, un automate est décrit par la notion de *processus*. L'ensemble des états est défini par l'instruction *etat*(e_0,e_1, ... ,e_n) et l'etat initial par : *initial* e_0. Les transitions sont présentées sous formes d'un ensemble de clauses représentant les conditions d'éxecution de cette transition, suivi d'un bloc d'instructions à exécuter dans le cas où cette transition est tirée. La communication entre automates s'éffectue par l'intermédiaire de deux ports connectés par un canal (*connect, canal*).

```
trans
        { clause }*
        debut
            < suite d'instructions >
        fin;
```

Une transition est dite tirable, si toutes les clauses d'entrées sont vraies.

Les clauses principales sont :

- *depuis* état$_1$ *vers* état$_2$, qui décrit le changement d'état de l'automate (la fonction de transition). etat$_1$ étant l'état initial. etat$_2$ étant l'etat final.
- *entrée* prt, qui est vrai si l'automate a une entrée sur le port de communication prt.
- *delai(x,y)*, introduit un retard à l'exécution d'une transition.
- *pourvu_que* $P(v_1,v_2, \ldots , v_n)$, qui spécidie un prédicat sue les variables internes de l'automate.
- *priorité(x)* qui définie un ordre de priorité entre des transitions tirables au même momemt.

Entre plusieurs automates parallèles, aucune variable ne peut être partagée. Le seul mode

d'interaction étant la communication par messages. Dans cet approche. l'élément conceptuel de base est l'*événement* auquel on associe une action. On distingue deux types d'événements :

Les événements d'entrée : **(?e)** ceux qui signalent l'arrivée d'un message sur un port.

Les événements de sortie : **(!e)** pour signaler la sortie d'un message sur un port.

Lorsqu'une transtion est tirée, l'exécution de la suite d'instructions correspondant est non interruptible (une exécution atomique) et est de durée nulle. Les transitions sont exclusives l'une par rapport à l'autre (par de transition en parallèle dans un même automate).

Le langage permet aussi de décrire un ensemble d'automates communicant entre eux. Chaque automate est une instensiation d'un module déclaré par l'intruction *bloc*. Un module peut contenir plusieurs sous – modules (*sous_blocs*). Le tout est décrit par l'instruction *raffinement*.

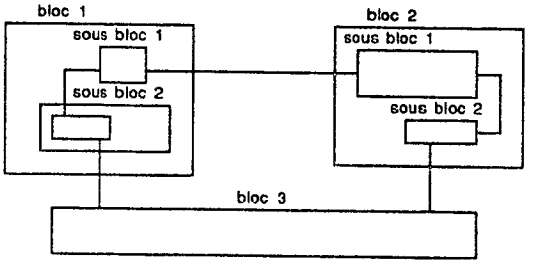

Figure 8. exemple de configuration de modules

3 modèle FDT – ESTELLE du protocole

3.1 configuration des modules

Pour la simulation du protocole nous distinguons les stations et les tronçons liant ces stations. Chaque station se décompose en trois blocs :

- un bloc **UTILISATEUR** qui gère l'expédition ou la réception des messages; nous avons choisi un modèle très simple sans ordonnancement de message: à tout instant, l'utilisateur a un ou zéro message à expédier; pour envoyer un message, l'utilisateur

utilise le port **PREQ**; pour obtenir l'état de ses requêtes, il utilise le port **PREP**, et enfin pour obtenir les messages qui lui sont destinés, il utilise le port **PMES**;

- un bloc **EMETTEUR** intégrant la procédure d'allocation à la voie de communication et chargé du conditionnement et de l'émission des messages; pour cela, il utilise les port **PEE** et **PSE** auxquels sont connectés les tronçons de fibre, les ports **CREQUETE** et **CREPONSE** pour le dialogue entre lui et l'utilisateur, et enfin le port **CER** pour la coopération avec le récepteur;
- un bloc **RECEPTEUR** chargé principalement de l'observation de la ligne de réception et de la transmission des messages destiné à l'utilisateur; pour cela, il utilise les ports **PSL** et **PEL** auxquels sont connectés les tronçons de fibre, le port **CER** pour coopérer avec l'émetteur, et enfin le port **PMES** pour transmettre à l'utilisateur les messages qui lui sont destinés.

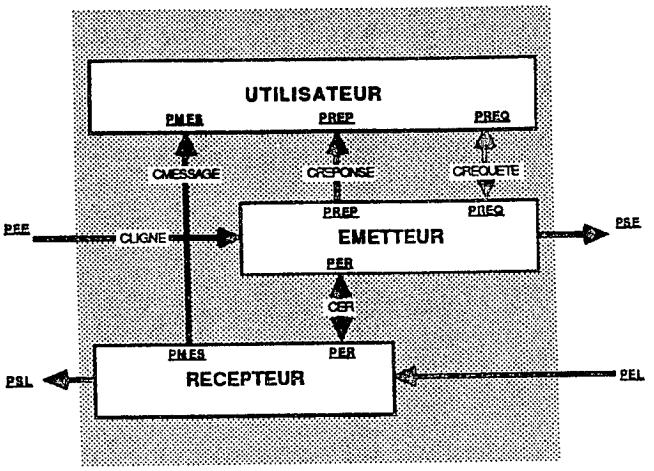

Figure 9. configuration d'une station

Il est également nécessaire de considérer un bloc **FIBRE** qui modélise le comportement de la fibre se traduisant par des retards; pour modéliser ces retard, on considère que les interactions arrivant sur son port **AMONT** sont répétées sur le port **AVAL** avec un retard constant qui englobe le retard de un bit introduit par chaque station, ainsi que la propagation à vitesse finie du signal lumineux dans les fibres optiques.

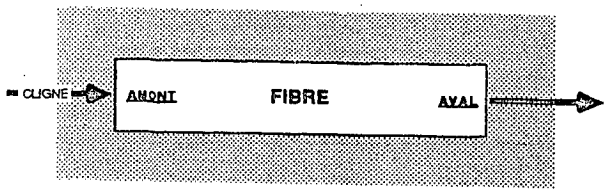

Figure 10. configuration du fibre

Les différents ports sont répertoriés suivant des types, ou encore **canaux** qui sont définis par le programmeur; le modèle de ce protocole possède cinq types de canaux permettant de décrire les interactions émises ou reçues sur les différents ports. Ajoutons que le mécanisme de communication est de type file au niveau des ports (analogue à la boîte a lettre non limitée en taille); dans ce cas, le flux d'interactions arrivant sur un port est organisé en FIFO, la gestion non méticuleuse de ce mécanisme par l'utilisateur conduit souvent à des interblocages : une interaction arrivant sur un port doit nécessairement être consommée à un moment donné et pour celà elle doit être avant tout attendue !

3.2 automates

Chaque bloc étant défini, on lui associe un automate pour décrire son comportement. Les interactions échangées entre les différents automates sont pour chaque type de canal :

- **CLIGNE** : sur ce type de canal circule le signal optique; ce signal véhicule trois types d'interactions :
  - *start, stop* : l'émission du drapeau start par la station de tête n'est pas instantaneé dans la réalité puisqu'il faut envoyer huit bits; en revanche, le signal start n'a aucune durée physique; il en est de même pour le stop; nous avons donc adopté la convention suivante pour modéliser fidèlement la réalité :lorqu'il y a émission d'un tel drapeau par un émetteur, l'événement apparaît à la fin de l'émission de ce drapeau; concrètement, il a été nécessaire d'affecter des délais aux gardes des transitions (ex ACTIVITE TETE vers REQUETE TETE, TXi vers suivant pour l'EMETTEUR)
  - *message* : l'événement message apparaît dès que le message commence a être transféré; il est à remarquer qu'il a été nécessaire d'introduire un délai pour exprimer l'envoi du message vide par la station de tête.
- **CMESSAGE** : sur ce canal seul ne circule que le signal message

- **CREPONSE** : sur ce canal circulent trois signaux de l'émetteur à l'utilisateur:
  - *rep0* : qui indique qu'un nouveau cycle d'allocation émission est commencé; ce signal est envoyé par l'émetteur à l'utilisateur pour lui demander s'il n'y a pas de message à transmettre;
  - *rep1* : qui indique à l'utilisateur que le message qu'il souhaitait transmettre est non prioritaire, donc n'est pas celui qui passe sur le réseau;
  - *rep2* : qui indique à l'utilisateur que sa requête a abouti, soit le message a été transmis;
- **CREQUETE** sur ce type de canal circule trois signaux de l'utilisateur à l'émetteur:
  - *dreq* : signale le debut d'une requête, soit la demande d'émission d'un message; ceci est une reponse possible au signal rep0;
  - *nreq* : signale que l'utilisateur n' a rien à èmettre; ceci est l'autre réponse possible au signal rep0;
  - *mreq* : signale le debut de transfert du message
  - *freq* : marque la fin du transfert du message; entre le signal mreq et freq s'ecoule le temps nécessaire à la transmission série du message.
- **PER** : sur ce type de canal trois signaux permettent la coopération émetteur récepteur:
  - émission : ce signal émis par l'émetteur indique au récepteur que le site est en train d'emettre un message; ayant émis ce signal, l'émetteur attend de recevoir du recepteur deux réponses possibles:
  - *succès* : est la réponse rendue si le recepteur a efeectivement reconnu l'adresse de la station comme étant celle qui émet;
  - *arrêt* : est la réponse rendue dans le cas contraire.

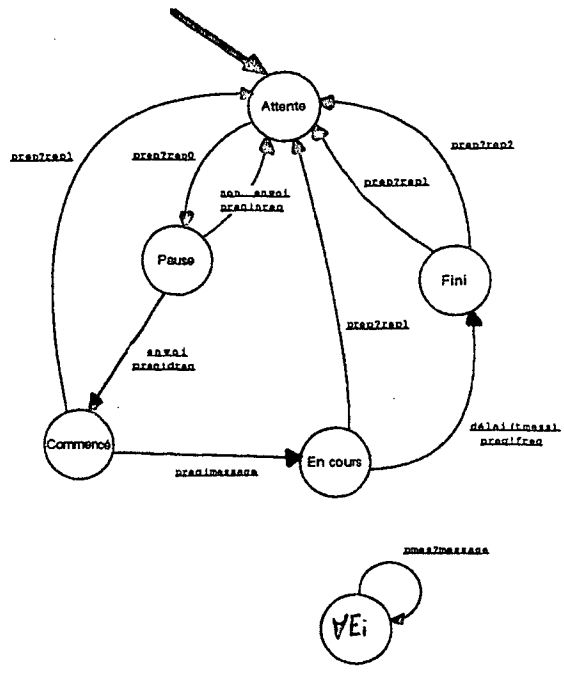

Figure 11. automate de l'utilisateur

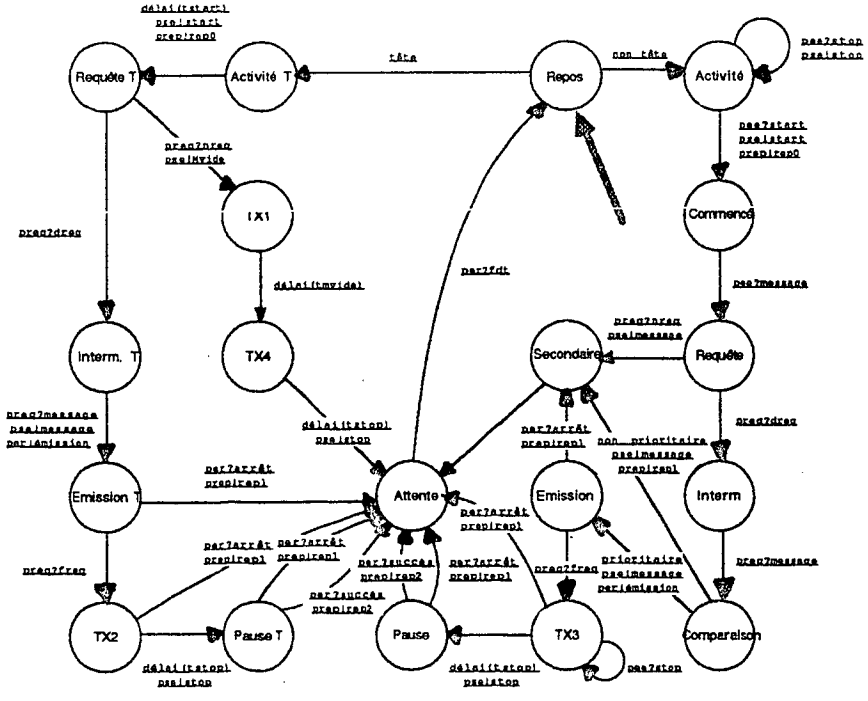

Figure 12. automate de l'émetteur

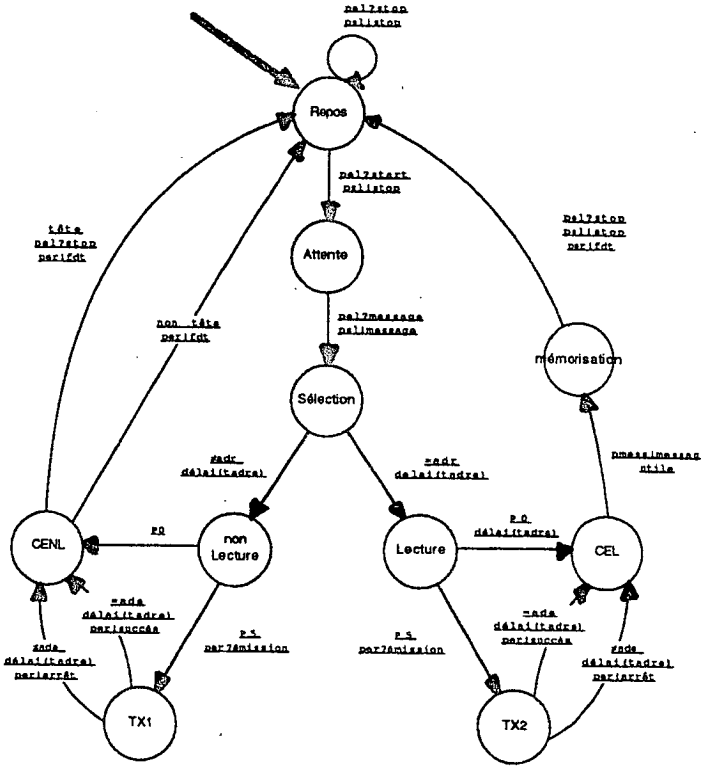

Figure 13. automate du récepteur

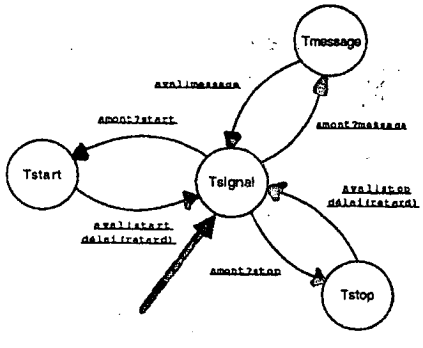

Figure 14. automate de la fibre

3.3 description en FDT – E du protocole

Dans ce module,la génération des messages est aléatoire; lorsqu'il reçoit le signal rep0 de l'émetteur, l'utilisateur génère un message avec une probabilité p fixée dans le programme; si ce message est créé, ses caractéristiques (sauf sa longueur) sont tirées aléatoirement. Les sources des différents modules sont regroupés dans les annexes.

3.4 simulation

3.4.1 conditions de simulation

Nous avons simulé le protocole pour un réseau comportant trois stations; l'unité logique correspond au temps de transmission de deux bits, les temps de transmission des messages sont donc évalués en temps logique; le retard introduit par chaque fibre correspond à une unité logique de temps.

Une première version où les messages étaient générés interactivement a permis la mise au point progressive du protocole; enfin la deuxième version qui intègre la génération automatique des messages nous a permis d'observer le comportement du système sur une durée de 20000 unités logiques de temps.

3.4.2 traces de simulation

A l'issue des simulations infructueuses, l'analyse de la trace d'exécution a permis de diagnostiquer les erreurs de conception ainsi que les omissions dans la spécification du protocole. Le comportement souhaité a pu être observé pendant la simulation; le second stade de cette étude va consister en l'utilisation d'observateurs chargés d'analyser les séquences d'interactions échangées entre blocs; ces observateurs seront déduits automatiquement des propriétés du protocole exprimées en logique temporelle. A l'heure actuelle, les simulations du protocole réel exempt d'erreurs nous ont permis de tester sa relative validité et de vérifier sa conformité par rapport aux souhaits exprimés.

```
VEDA
chargement de Veda en cours. patience...

11/28/86  1124.1 fwt Fri


-- Systeme de simulation VEDA --
-- CNET Lannion A SLC/EVP. Version 1_003 --

veda :sim automatique.

Executeur Veda : version 1_000
Lecture des parametres (entiers ou reels)
Temps max de simu et germe (entier) sont les deux premiers
:200 4.

-- debut de la simulation -- noyau VEDA 04/85

T =    4.00..... EMETTEUR      1 ---(....start....)--> EMETTEUR      2.....
T =    4.00..... EMETTEUR      1 ---(....repO.....)--> UTILISATEUR   1.....
T =    4.00..... UTILISATEUR   1 <--(....repO.....)--- EMETTEUR      1.....
*---MESSAGE---*
T =    4.00..... UTILISATEUR   1 ---(....dreq.....)--> EMETTEUR      1.....
T =    4.00..... UTILISATEUR   1 ---(message_local)--> EMETTEUR      1.....
T =    4.00..... EMETTEUR      1 <--(....dreq.....)--- UTILISATEUR   1.....
T =    4.00..... EMETTEUR      1 <--(message_local)--- UTILISATEUR   1.....
T =    4.00..... EMETTEUR      1 ---(message_local)--> EMETTEUR      2.....


T =    5.00..... EMETTEUR      2 <--(....start....)--- EMETTEUR      1.....
T =    5.00..... EMETTEUR      2 ---(....start....)--> EMETTEUR      3.....
T =    5.00..... EMETTEUR      2 ---(....repO.....)--> UTILISATEUR   2.....
T =    5.00..... UTILISATEUR   2 <--(....repO.....)--- EMETTEUR      2.....
T =    5.00..... UTILISATEUR   2 ---(....nreq.....)--> EMETTEUR      2.....
T =    6.00..... EMETTEUR      3 <--(....start....)--- EMETTEUR      2.....
T =    6.00..... EMETTEUR      3 ---(....start....)--> EMETTEUR       .....
T =    6.00..... EMETTEUR      3 ---(....repO.....)--> UTILISATEUR   3.....
T =    6.00..... RECEPTEUR     3 <--(....start....)--- RECEPTEUR     2.....
T =    6.00..... RECEPTEUR     3 ---(....start....)--> RECEPTEUR      .....
T =    6.00..... UTILISATEUR   3 <--(....repO.....)--- EMETTEUR      3.....
*---MESSAGE---*
T =    6.00..... UTILISATEUR   3 ---(....dreq.....)--> EMETTEUR      3.....
T =    6.00..... UTILISATEUR   3 ---(message_local)--> EMETTEUR      3.....
T =    6.00..... EMETTEUR      2 <--(message_amant)--- EMETTEUR      1.....
```

Figure 15. exemple de trace d'exécution

3.4.3 conclusions

Après avoir resitué nos besoins utilisateur dans un contexte temps réel et exprimé nos objectifs, nous avons dans un premier temps décrit un protocole de communication bas niveau répondant a notre problème. Ensuite nous avons dégagé un modèle logique de ce protocole qui tient compte de la grandeur physique temps, sans toutefois tenir compte des erreurs de transmission de message qui se manifestent toujours plus ou moins dans la réalité. La simulation de ce modèle logique nous à permis de déceler les incohérences et les carences dans sa spécification. A ce stade de l'étude, nous pensons avoir conçu un protocole d'une grande fiabilité (relative). La suite des travaux va consister d'une part à intégrer les erreurs, et donc a augmenter la robustesse du protocole, et d'autre part à modéliser l'utilisation de façon plus prècise en augmentant le degré d'abstraction pour obtenir un modèle complet intégrant les services existant au niveau APPLICATION

BIBLIOGRAPHIE

[BEL85] P.Belmans : Synthèse sur les RE'seaux Locaux Temps Réel – P.Interne I.R.I.S.A. no 288, mars 1986

[BEL85] P.Belmans, J.J.Borrelly, M.L.Silly, D.Simon : NESTOR : Noyau d'Exécutif pour le Suivi en Temps réel des applications Orientées Robotique – P.Interne I.R.I.S.A. no 267, septembre 1985

[BOR86] J.J.Borrelly, D.Simon : Specification of a Sublocal Network for Robotics – papier présenté à l' Advanced Seminar on Real-Time Local Area Networks –

[FDT85] Présentation du langage FDT-E (Estelle) – un langage de spécification des protocoles – document CNET/LAA/SLC – version du 22 mars 1985

[GRO85] R.Groz, C.Jard, C.Lassudrie : Attacking a Complex Distributed Algorithm from Different Sides : an Experience with Complementary Validation Tools – Computer Networks, vol. 10, No 5, december 1985

[GRO85] R.Groz, C.Jard, J.F.Momin : VEDA : a Software Simulator for the Validation of Protocol Specifications – Comnet 85, Budapest, octobre 1985

[GRO86] R.Groz : Unrestricted verification of protocol Properties on a simulation using an observer approch – 6th IFIP WG 6.1 Workshop, Montréal, juin 86

[LEL84] G.Le Lann : Protocoles de gestion des Accès Multiples et Réseaux Locaux Temps-Réel – Séminaire INRIA – mars 1984

[MIC82a] G.Michel, J.Rouillart, G.Charles, D.Tranvaux : CICS – A VLSI-Based Network for Distributed Process-Control – Proceedings 3rd International Conference on Distributed Computing Systems, Miami, Fort Lauderdale, Fa USA, 18-22 oct 1982

[QUE82] J.P.Queille : Le système CESAR : description, spécification et analyse des applications réparties – Thèse de Docteur Ingénieur – IPIAG France, juin 82

[SIM87] D.Simon : Réseau Temps Réel pour la Robotique et la Commande distribuée – à paraître

Annexe 1 : Description du protocole en ESTELLE

```
systeme res;

const     n =3;
          m =2;
          vrai =true;
          faux =false;
          tete_stations =1;

          tstart = 4;
          tmess = 144;
          tstop = 4;
          tadre = 16;
          tmvide = 32;
          valeur_retard = 1;
          taux = 50;

bloc reseaux;
          raffinement r_reseau pour reseaux;

          type      station =1..n;
                    troncon =1..m;
                    adresse =0..n;
                    portion =1..2;
                    type_message = article
                                        pri : integer;
                                        adr : adresse;
                                        ade : adresse;
                                        don : integer;
                                        crc : boolean;
                                        fin;

          canal cligne;
                    start;
                    contenu(message : type_message);
                    stop;

          canal cmessage;
                    contenu(message : type_message);

          canal creponse;
                    rep0;
                    rep1;
                    rep2;

          canal crequete;
                    dreq;
                    contenu(message : type_message);
                    freq;
                    nreq;

          canal cer;
                    arret;
                    emission;
                    succes;
                    fdt;

bloc utilisateurs(pmes : cmessage; prep : creponse; preq :crequete);
processus putilisateur(numero : station) pour utilisateurs;
```

```
            file pmes: preq: prep:
            etat(attente,pause,commence,en_cours,fini);

            var         reponse : char;
                        envoi_message : boolean;
                        message_r : type_message;
                        message_e : type_message;
                        hasard : integer;

            initialisation
                        debut
                        prochain attente;
                        envoi_message :=faux;
                        fin;
            procedure tracer(c:char;t:integer);
             debut
              write('T = ',time:6:2,'.....');
               write(' UTILISATEUR ',numero:2,' ');
               si c='<' alors write('<--(') sinon write('---(');
               cas t de
                0: write('....repO.....');
                1: write('....nreq.....');
                2: write('....dreq.....');
                3 : write('....rep1.....');
                4 : write('message_local');
                5 : write('....freq.....');
                6 : write('....rep2.....');
                fin;
              si c='<' alors write(')---') sinon write(')-->');
              writeln(' EMETTEUR     ',numero:2,'.....');
            fin;

            trans
            entree pmes.contenu(message_r)
            depuis attente,pause,commence,en_cours,fini
            vers meme
            priorite(10)
                        debut
writeln('*********************************');
write('la station numero.');
writeln(numero);
write('recoit le message.');
writeln(message_r.adr,message_r.ade,message_r.crc);
writeln('*********************************');
writeln;
                        fin;

           trans
           entree prep.repO
           depuis attente
           vers pause
                        debut
tracer('<',0);

                        hasard := randomint(0,100);
                        si hasard <= taux alors
                                    debut
                                    writeln('*---MESSAGE---*');
                                    envoi_message :=vrai;
                                    message_e.pri :=randomint(0,32000);
```

```
                                message_e.adr :=randomint(0,n);
                                message_e.ade :=numero;
                                message_e.don :=0;
                                message_e.crc :=vrai;
                                fin
                        sinon envoi_message :=faux;
                        fin;

            trans
            depuis pause
            pourvu_que non(envoi_message)
            vers attente
            delai(0,0)
                        debut
                        sortie preq.nreq;
tracer('>',1);
                        fin;


            trans
            depuis pause
            pourvu_que envoi_message
            vers commence
            delai(0,0)
                        debut
                        sortie preq.dreq;
                        envoi_message:=faux;
tracer('>',2);
                        fin;

            trans
            entree prep.repi
            depuis commence
            vers attente
                        debut
tracer('<',3);
                        fin;

            trans
            depuis commence
            vers en_cours
            delai(0,0)
                        debut
                        sortie preq.contenu(message_e);
tracer('>',4);
                        fin;

            trans
            entree prep.repi
            depuis en_cours
            vers attente
                        debut
tracer('<',3);
                        fin;

            trans
            depuis en_cours
            vers fini
            delai(tmess)
                        debut
```

```
                    sortie preq.freq;
tracer('>',5);
                    fin;

          trans
          entree prep.rep2
          depuis fini
          vers attente
                    debut
tracer('<',6);
                    fin;

          trans
          entree prep.rep1
          depuis fini
          vers attente
                    debut
tracer('<',3);
                    fin;


bloc emetteurs(prep : creponse; preq : crequete; per : cer; pse, pee : cligne);
processus pemetteur(numero : station) pour emetteurs;
          file pee; pse; preq; prep;
          etat(repos, activite_tete, activite, requete, requete_tete, comparaison,
          attente, emission_tete, emission, secondaire, pause, pause_tete,
          commence,intermediaire ,intermediaire_tete, tx1, tx2, tx3,tx4);

          var       tete : boolean;
                    reponse : char;
                    message_vide : type_message;
                    message_amont : type_message;
                    message_local : type_message;

          initialisation
                    debut
                    prochain repos;
                    si numero = tete_stations alors debut
                              tete:=vrai;
                              message_vide.pri :=-1;
                              message_vide.ade :=0;
                              message_vide.adr :=0;
                              message_vide.don :=0;
                              message_vide.crc :=faux;
                              fin
                    sinon tete:=faux;
                    fin;
            procedure tracer(c:char;t:integer;n1:integer;n2:integer);
              debut
                    si n2<=n alors
             debut
              write('T = ',time:6:2,'.....');
              write(' EMETTEUR     ',numero:2,' ');
              si c='<' alors write('<---(') sinon write('---(');
              cas t de
               0:write('....repO.....');
               1:write('....nreq.....');
               2:write('....dreq.....');
               3:write('....rep1.....');
               4:write('message_local');
```

```
                5:write('....freq.....');
                6:write('....rep2.....');
                7:write('....start....');
                8:write('message_amant');
                9:write('....arret....');
               10:write('....stop.....');
               11:write('....succes...');
               12:write('message_vide.');
               13 :write('....fdt.....');
               14:write('...emission..');
            fin;
            si c='<' alors write(')---') sinon write(')-->');
             cas n1 de
               1:write(' UTILISATEUR ');
               2:write(' RECEPTEUR   ');
               3:write(' EMETTEUR    ');
             fin;
             writeln(n2:2,'.....');
           fin;
         fin;

      trans
      depuis repos
      pourvu_que non(tete)
      vers activite
      delai(0,0)
                debut
                fin;

      trans
entree pee.stop
      depuis activite
      vers meme
                debut
      sortie pse.stop;
tracer('<',10,3,numero-1);
tracer('>',10,3,numero+1);
                fin;
      trans
      entree pee.start
      depuis activite
      vers commence
                debut
                sortie pse.start;
                sortie prep.rep0;
tracer('<',7,3,numero-1);
tracer('>',7,3,numero+1);
tracer('>',0,1,numero);
                fin;

      trans
      entree pee.contenu(x)
      depuis commence
      vers requete
                debut
                message_amont :=x;
tracer('<',8,3,numero-1);
                fin;

      trans
```

```
            entree preq.dreq
            depuis requete
            vers intermediaire
                        debut
  tracer('<',2,1,numero);
                        fin;

            trans
            entree preq..ireq
            depuis requete
            vers secondaire
                        debut
                        sortie pse.contenu(message_amont);
  tracer('<',1,1,numero);
  tracer('>',8,3,numero+1);
                        fin;

            trans
            entree preq.contenu(x)
            depuis intermediaire
            vers comparaison
                        debut
                        message_local :=x;
 tracer('<',1,1,numero);
                        fin;

            trans
            depuis comparaison
            pourvu_que message_local.pri>message_amont.pri
            vers emission
            delai(0,0)
                        debut
                        sortie pse.contenu(message_local);
                        sortie per.emission;
 tracer('>',4,3,numero+1);
 tracer('>',14,2,numero);
                        fin;

           trans
           depuis comparaison
           pourvu_que message_local.pri <=message_amont.pri
           vers secondaire
           delai(0,0)
                       debut
                       sortie pse.contenu(message_amont);
                       sortie prep.rep1;
tracer('>',8,3,numero+1);
tracer('>',3,1,numero);
                       fin;

          trans
          entree per.arret
          depuis emission
          vers secondaire
                      debut
                      sortie prep.rep1;
tracer('<',9,2,numero);
tracer('>',3,1,numero);
                      fin;
```

```
          trans
          entree preq.freq
          depuis emission
          vers tx3
                    debut
tracer('<',5,1,numero);
                    fin;

          trans
          depuis tx3
          vers pause
          delai(tstop)
                    debut
                    sortie pse.stop;
tracer('>',10,3,numero+1);
                    fin;

          trans
          entree pse.stop
          depuis tx3
          vers meme
          priorite(5)
                    debut
tracer('<',10,3,numero-1);
                    fin;

          trans
          depuis secondaire
          vers attente
delai(0,0)
                    debut
                    fin;

          trans
          entree per.arret
          depuis pause
          vers attente
                    debut
                    sortie prep.rep1;
tracer('<',9,2,numero);
tracer('>',3,1,numero);
                    fin;

          trans
          entree per.succes
          depuis pause
          vers attente
                    debut
                    sortie prep.rep2;
tracer('<',11,2,numero);
tracer('>',6,1,numero);
                    fin;


          trans
          depuis repos
          pourvu_que tete
          vers activite_tete
          delai(0,0)
```

```
                    debut
                    fin;

          trans
          depuis activite_tete
          vers requete_tete
          delai(tstart)
                    debut
                    sortie pse.start;
                    sortie prep.rep0;
tracer('>',7,3,numero+1);
tracer('>',0,1,numero);
                    fin;

          trans
          entree preq.dreq
          depuis requete_tete
          vers intermediaire_tete
                    debut
tracer('<',2,1,numero);
                    fin;

          trans
          entree preq.nreq
          depuis requete_tete
          vers tx1
                    debut
                    sortie pse.contenu(message_vide);
tracer('<',1,1,numero);
tracer('>',12,3,numero+1);
                    fin;

          trans
          depuis tx1
          vers tx4
          delai(tmvide)
                    debut
                    fin;

          trans
          depuis tx4
          vers attente
          delai(tstop)
                    debut
                    sortie pse.stop;
tracer('>',10,3,numero+1);
                    fin;

          trans
          entree preq.contenu(x)
          depuis intermediaire_tete
          vers emission_tete
                    debut
                    message_local :=x;
                    sortie pse.contenu(message_local);
                    sortie per.emission;
tracer('<',4,1,numero);
tracer('>',4,3,numero+1);
                    fin;
```

```
          trans
          entree per.arret
          depuis emission_tete
          vers attente
                    debut
                    sortie prep.rep1;
tracer('<',9,2,numero);
tracer('>',3,1,numero);
                    fin;

          trans
          entree preq.freq
          depuis emission_tete
          vers tx2
                    debut
tracer('<',5,1,numero);
                    fin;

          trans
          depuis tx2
          vers pause_tete
          delai(tstop)
                    debut
                    sortie pse.stop;
tracer('>',10,3,numero+1);
                    fin;

          trans
          entree per.arret
          depuis pause_tete
          vers attente
                   debut
                   sortie prep.rep1;
tracer('<',9,2,numero);
tracer('>',3,1,numero);
                   fin;
                    trans
                    depuis tx2 vers attente
                    entree per.arret
                    debut
                     sortie prep.rep1;
                     tracer('<',9,2,numero);
                     tracer('>',3,1,numero);
                     fin;
                    trans
                     depuis tx3 entree per.arret vers attente
                       debut
                          sortie prep.rep1;
                         tracer('<',9,2,numero);
                         tracer('>',3,1,numero);
                          fin;

          trans
          entree per.succes
          depuis pause_tete
          vers attente
                   debut
                   sortie prep.rep2;
tracer('<',11,2,numero);
tracer('>',6,1,numero);
```

```
                    fin;

          trans
          entree pee.stop
          depuis attente
          vers meme
          priorite(10)
                    debut
                    sortie pse.stop;
tracer('<',10,3,numero-1);
tracer('>',10,3,numero+1);
                    fin;

          trans
          entree per.fdt
          depuis attente
          vers repos
                    debut
tracer('<',13,2,numero);
                    fin;


bloc recepteurs(pmes : cmessage; per : cer; pel, psl : cligne);
processus precepteur(numero : station) pour recepteurs;
          file psl; pel; per; pmes;
          etat(initial, repos, attente, non_lecture, controle_e_n1, lecture,
          controle_e_1,selection, tx1, tx2, tx3, tx4,memorisation);
          var       message : type_message;
                    tete : boolean;

          initialisation
                    debut
                    si numero = tete_stations alors tete :=vrai sinon tete :=faux;
                    prochain repos
                    fin;
         procedure trace(c:char;t:integer;n1:integer;n2:integer);
          debut
           write('T = ',time:6:2,'.....');
            write(' RECEPTEUR   ',numero:2,' ');
           si c='<' alors write('<--(') sinon write('---(');
            cas t de
             7:write('....start....');
             8:write('...message...');
             9:write('....arret....');
            10:write('....stop.....');
            11:write('...succes....');
            13:write('.... fdt.....');
            14:write('..emission..');
            fin;
          si c='<' alors write(')---') sinon write(')-->');
          cas n1 de
           1:write(' UTILISATEUR ');
           2:write(' RECEPTEUR   ');
           3:write(' EMETTEUR    ');
           fin;
          writeln(n2:2,'.....');
        fin;
```

```
          trans
          entree pel.start
          depuis repos
          vers attente
                    debut
                    sortie psl.start;
trace('<',7,2,numero-1);
trace('>',7,2,numero+1);
                    fin;

          trans
          entree pel.contenu(x)
          depuis attente
          vers selection
                    debut
                    message :=x;
                    sortie psl.contenu(message);
trace('<',8,2,numero-1);
trace('>',8,2,numero+1);
                    fin;

          trans
          depuis selection
          pourvu_que message.adr<>numero
          vers non_lecture
          delai(tadre)
                    debut
                    fin;

          trans
          depuis selection
          pourvu_que message.adr=numero
          vers lecture
          delai(tadre)
                    debut
                    fin;


          trans
          depuis non_lecture
          vers controle_e_nl
          delai(0)
                    debut
                    fin;

          trans
          entree per.emission
          depuis non_lecture
          vers tx1
          priorite(5)
                    debut
trace('<',14,3,numero);
                    fin;

          trans
          depuis tx1
          pourvu_que (message.ade<>numero)
          vers controle_e_nl
          delai(tadre)
                    debut
```

```
                    sortie per.arret;
trace('>',9,3,numero);
                    fin;

          trans
          depuis tx1
          pourvu_que (message.ade=numero)
          vers controle_e_n1
          delai(tadre)
                    debut
                    sortie per.succes;
trace('>',11,3,numero);
                    fin;

          trans
          entree pe1.stop
          depuis controle_e_n1
          pourvu_que tete
          vers repos
                    debut
                    sortie ps1.stop;
                    sortie per.fdt;
trace('<',10,2,numero-1);
trace('>',10,2,numero+1);
trace('>',13,3,numero);
                    fin;

          trans
          depuis controle_e_n1
          pourvu_que non(tete)
          vers repos
          delai(0)
                    debut
                    sortie per.fdt;
trace('>',13,3,numero);
                    fin;

          trans
          depuis lecture
          vers controle_e_1
          delai(0)
                    debut
                    fin;

          trans
          entree per.emission
          depuis lecture
          vers tx2
          priorite(5)
                    debut
trace('<',14,3,numero);
                    fin;

          trans
          depuis tx2
          pourvu_que (message.ade<>numero)
          vers controle_e_1
          delai(tadre)
                    debut
                    sortie per.arret;
```

```
trace('>',9,3,numero);
                    fin;

          trans
          depuis tx2
          pourvu_que (message.ade=numero)
          vers controle_e_1
          delai(tadre)
                    debut
                    sortie per.succes;
trace('>',11,3,numero);
                    fin;

          trans
          entree pe1.stop
          depuis memorisation
          vers repos
                    debut
                    sortie ps1.stop;
                    sortie per.fdt;
trace('<',10,2,numero-1);
trace('>',10,2,numero+1);
trace('>',13,3,numero);
                    fin;
                  trans
                  depuis controle_e_1
                  vers memorisation
                  delai(0)
                   debut

                      sortie pmes.contenu(message);
                       trace('>',8,1,numero);
                    fin;

          trans
          entree pe1.stop
          depuis repos
          vers meme
                    debut
                    sortie ps1.stop;
trace('<',10,2,numero-1);
trace('>',10,2,numero+1);
                    fin;


bloc fibres(aval, amont : cligne);
processus pfibre(numero : troncon; chaine : portion ; retard : integer) pour fibres;
          file aval;amont;
          etat(tsignal, tstart, tstop, tmessage);

          var       message_transmis : type_message;

          initialisation
                    debut
                    prochain tsignal;
                    message_transmis.pri :=-1;
                    message_transmis.ade :=0;
                    message_transmis.adr :=0;
                    message_transmis.don :=0;
                    message_transmis.crc :=faux;
```

```
                fin;

        trans
        entree amont.start
        depuis tsignal
        vers tstart
                debut
                fin;

        trans
        depuis tstart
        vers tsignal
        delai(retard)
                debut
                sortie aval.start
                fin;

        trans
        entree amont.stop
        depuis tsignal
        vers tstop
                debut
                fin;

        trans
        depuis tstop
        vers tsignal
        delai(retard)
                debut
                sortie aval.stop
                fin;

        trans
        entree amont.contenu(x)
        depuis tsignal
        vers tmessage
                debut
                message_transmis :=x;
                fin;

        trans
        depuis tmessage
        vers tsignal
        delai(retard)
                debut
                sortie aval.contenu(message_transmis);
                fin;


sous_blocs utilisateur : tableau[station] de utilisateurs;
           emetteur : tableau[station] de emetteurs;
           recepteur : tableau[station] de recepteurs;
           fibre : tableau[1..m, portion] de fibres;
configure
          debut
          tout i:1..m faire
                  debut
                  connecte emetteur[i].pse =fibre[i,1].amont;
```

Annexe 2 : Trace d'une simulation

```
VEDA
chargement de Veda en cours. patience...

11/28/86  1124.1 fwt Fri


-- Systeme de simulation VEDA --
-- CNET Lannion A SLC/EVP. Version 1_003 --

   toute commande se termine par .
   tapez aide pour connaitre la liste des commandes
   tapez listinfo pour avoir la liste des informations
   disponibles du systeme
   tapez info <nom_info>  pour obtenir des renseignements sur  <nom_info>


veda :
:cas res.

veda :sim automatique.

Executeur Veda : version 1_000
Lecture des parametres (entiers ou reels)
Temps max de simu et germe (entier) sont les deux premiers
:200 4.

-- debut de la simulation -- noyau VEDA 04/85

T =    4.00.....  EMETTEUR      1 ---(....start....)--> EMETTEUR      2.....
T =    4.00.....  EMETTEUR      1 ---(....repO.....)--> UTILISATEUR   1.....
T =    4.00.....  UTILISATEUR   1 <--(....repO.....)--- EMETTEUR      1.....
*---MESSAGE---*
T =    4.00.....  UTILISATEUR   1 ---(....dreq.....)--> EMETTEUR      1.....
T =    4.00.....  UTILISATEUR   1 ---(message_local)--> EMETTEUR      1.....
T =    4.00.....  EMETTEUR      1 <--(....dreq.....)--- UTILISATEUR   1.....
T =    4.00.....  EMETTEUR      1 <--(message_local)--- UTILISATEUR   1.....
T =    4.00.....  EMETTEUR      1 ---(message_local)--> EMETTEUR      2.....


T =    5.00.....  EMETTEUR      2 <--(....start....)--- EMETTEUR      1.....
T =    5.00.....  EMETTEUR      2 ---(....start....)--> EMETTEUR      3.....
T =    5.00.....  EMETTEUR      2 ---(....repO.....)--> UTILISATEUR   2.....
T =    5.00.....  UTILISATEUR   2 <--(....repO.....)--- EMETTEUR      2.....
T =    5.00.....  UTILISATEUR   2 ---(....nreq.....)--> EMETTEUR      2.....
T =    6.00.....  EMETTEUR      3 <--(....start....)--- EMETTEUR      2.....
T =    6.00.....  EMETTEUR      3 ---(....start....)--> EMETTEUR      4.....
T =    6.00.....  EMETTEUR      3 ---(....repO.....)--> UTILISATEUR   3.....
T =    6.00.....  RECEPTEUR     3 <--(....start....)--- RECEPTEUR     2.....
T =    6.00.....  RECEPTEUR     3 ---(....start....)--> RECEPTEUR     4.....
T =    6.00.....  UTILISATEUR   3 <--(....repO.....)--- EMETTEUR      3.....
*---MESSAGE---*
T =    6.00.....  UTILISATEUR   3 ---(....dreq.....)--> EMETTEUR      3.....
T =    6.00.....  UTILISATEUR   3 ---(message_local)--> EMETTEUR      3.....
T =    6.00.....  EMETTEUR      2 <--(message_amant)--- EMETTEUR      1.....
T =    6.00.....  EMETTEUR      2 <--(....nreq.....)--- UTILISATEUR   2.....
T =    6.00.....  EMETTEUR      2 ---(message_amant)--> EMETTEUR      3.....
T =    7.00.....  RECEPTEUR     2 <--(....start....)--- RECEPTEUR     1.....
T =    7.00.....  RECEPTEUR     2 ---(....start....)--> RECEPTEUR     3.....
T =    7.00.....  EMETTEUR      3 <--(message_amant)--- EMETTEUR      2.....
T =    7.00.....  EMETTEUR      3 <--(....dreq.....)--- UTILISATEUR   3.....
T =    7.00.....  EMETTEUR      3 <--(....nreq.....)--- UTILISATEUR   3.....
T =    7.00.....  EMETTEUR      3 ---(message_amant)--> EMETTEUR      4.....
T =    7.00.....  EMETTEUR      3 ---(....repi.....)--> UTILISATEUR   3.....
T =    7.00.....  RECEPTEUR     3 <--(...message...)--- RECEPTEUR     2.....
T =    7.00.....  RECEPTEUR     3 ---(...message...)--> RECEPTEUR     4.....
T =    7.00.....  UTILISATEUR   3 <--(....repi.....)--- EMETTEUR      3.....


T =    8.00.....  RECEPTEUR     1 <--(....start....)--- RECEPTEUR     0.....
T =    8.00.....  RECEPTEUR     1 ---(....start....)--> RECEPTEUR     2.....
T =    8.00.....  RECEPTEUR     2 <--(...message...)--- RECEPTEUR     1.....
T =    8.00.....  RECEPTEUR     2 ---(...message...)--> RECEPTEUR     3.....


T =    9.00.....  RECEPTEUR     1 <--(...message...)--- RECEPTEUR     0.....
T =    9.00.....  RECEPTEUR     1 ---(...message...)--> RECEPTEUR     2.....
T =   23.00.....  RECEPTEUR     3 ---(.... fdt.....)--> EMETTEUR      3.....
T =   23.00.....  EMETTEUR      3 <--(.... fdt.....)--- RECEPTEUR     3.....
T =   24.00.....  RECEPTEUR     2 ---(.... fdt.....)--> EMETTEUR      2.....
```

```
T =  25.00..... RECEPTEUR     1 <--(..emission...)--- EMETTEUR      1.....
T =  41.00..... RECEPTEUR     1 ---(...succes....)--> EMETTEUR      1.....
T = 148.00..... UTILISATEUR   1 ---(....freq.....)--> EMETTEUR      1.....
T = 148.00..... EMETTEUR      1 <--(....freq.....)--- UTILISATEUR   1.....
T = 152.00..... EMETTEUR      1 ---(....stop.....)--> EMETTEUR      2.....
T = 152.00..... EMETTEUR      1 <--(....succes...)--- RECEPTEUR     1.....
T = 152.00..... EMETTEUR      1 ---(....rep2.....)--> UTILISATEUR   1.....
T = 152.00..... UTILISATEUR   1 <--(....rep2.....)--- EMETTEUR      1.....
T = 153.00..... EMETTEUR      2 <--(....stop.....)--- EMETTEUR      1.....
T = 153.00..... EMETTEUR      2 ---(....stop.....)--> EMETTEUR      3.....
T = 154.00..... EMETTEUR      3 <--(....stop.....)--- EMETTEUR      2.....
T = 154.00..... EMETTEUR      3 ---(....stop.....)--> EMETTEUR      4.....
T = 154.00..... RECEPTEUR     3 <--(....stop.....)--- RECEPTEUR     2.....
T = 154.00..... RECEPTEUR     3 ---(....stop.....)--> RECEPTEUR     4.....
T = 155.00..... RECEPTEUR     2 <--(....stop.....)--- RECEPTEUR     1.....
T = 155.00..... RECEPTEUR     2 ---(....stop.....)--> RECEPTEUR     3.....

T = 156.00..... RECEPTEUR     1 <--(....stop.....)--- RECEPTEUR     0.....
T = 156.00..... RECEPTEUR     1 ---(....stop.....)--> RECEPTEUR     2.....
T = 156.00..... RECEPTEUR     1 ---(.... fdt.....)--> EMETTEUR      1.....
T = 156.00..... EMETTEUR      1 <--(.... fdt.....)--- RECEPTEUR     1.....
T = 160.00..... EMETTEUR      1 ---(....start....)--> EMETTEUR      2.....
T = 160.00..... EMETTEUR      1 ---(....rep0.....)--> UTILISATEUR   1.....
T = 160.00..... UTILISATEUR   1 <--(....rep0.....)--- EMETTEUR      1.....
T = 160.00..... UTILISATEUR   1 ---(....nreq.....)--> EMETTEUR      1.....
T = 160.00..... EMETTEUR      1 <--(....nreq.....)--- UTILISATEUR   1.....
T = 160.00..... EMETTEUR      1 ---(message_vide)--> EMETTEUR       2.....

T = 161.00..... EMETTEUR      2 <--( ...start....)--- EMETTEUR      1.....
T = 161.00..... EMETTEUR      2 ---(....start....)--> EMETTEUR      3.....
T = 161.00..... EMETTEUR      2 ---(....rep0.....)--> UTILISATEUR   2.....
T = 161.00..... UTILISATEUR   2 <--(....rep0.....)--- EMETTEUR      2.....
T = 161.00..... UTILISATEUR   2 ---(....nreq.....)--> EMETTEUR      2.....
T = 162.00..... EMETTEUR      3 <--(....start....)--- EMETTEUR      2.....
T = 162.00..... EMETTEUR      3 ---(....start....)--> EMETTEUR      4.....
T = 162.00..... EMETTEUR      3 ---(....rep0.....)--> UTILISATEUR   3.....
T = 162.00..... RECEPTEUR     3 <--(....start....)--- RECEPTEUR     2.....
T = 162.00..... RECEPTEUR     3 ---(....start....)--> RECEPTEUR     4.....
T = 162.00..... UTILISATEUR   3 <--(....rep0.....)--- EMETTEUR      3.....
+---MESSAGE---*
T = 162.00..... UTILISATEUR   3 ---(....dreq.....)--> EMETTEUR      3.....
T = 162.00..... UTILISATEUR   3 ---(message_local)--> EMETTEUR      3.....
T = 162.00..... EMETTEUR      2 <--(message_amant)--- EMETTEUR      1.....
T = 162.00..... EMETTEUR      2 <--(....nreq.....)--- UTILISATEUR   2.....
T = 162.00..... EMETTEUR      2 ---(message_amant)--> EMETTEUR      3.....
T = 163.00..... RECEPTEUR     2 <--(....start....)--- RECEPTEUR     1.....
T = 163.00..... RECEPTEUR     2 ---(....start....)--> RECEPTEUR     3.....
T = 163.00..... EMETTEUR      3 <--(message_amant)--- EMETTEUR      2.....
T = 163.00..... EMETTEUR      3 <--(....dreq.....)--- UTILISATEUR   3.....
T = 163.00..... EMETTEUR      3 <--(....nreq.....)--- UTILISATEUR   3.....
T = 163.00..... EMETTEUR      3 ---(message_local)--> EMETTEUR      4.....
T = 163.00..... EMETTEUR      3 ---(...emission..)--> RECEPTEUR     3.....
T = 163.00..... RECEPTEUR     3 <--(...message...)--- RECEPTEUR     2.....
T = 163.00..... RECEPTEUR     3 ---(...message...)--> RECEPTEUR     4.....

T = 164.00..... RECEPTEUR     1 <--(....start....)--- RECEPTEUR     0.....
T = 164.00..... RECEPTEUR     1 ---(....start....)--> RECEPTEUR     2.....
T = 164.00..... RECEPTEUR     2 <--(...message...)--- RECEPTEUR     1.....
T = 164.00..... RECEPTEUR     2 ---(...message...)--> RECEPTEUR     3.....

T = 165.00..... RECEPTEUR     1 <--(...message...)--- RECEPTEUR     0.....
T = 165.00..... RECEPTEUR     1 ---(...message...)--> RECEPTEUR     2.....

T = 179.00..... RECEPTEUR     3 <--(..emission...)--- EMETTEUR      3.....
T = 180.00..... RECEPTEUR     2 ---(.... fdt.....)--> EMETTEUR      2.....
T = 180.00..... EMETTEUR      2 <--(.... fdt.....)--- RECEPTEUR     2.....
T = 195.00..... RECEPTEUR     3 ---(...succes....)--> EMETTEUR      3.....
T = 196.00..... EMETTEUR      1 ---(....stop.....)--> EMETTEUR      2.....
T = 197.00..... EMETTEUR      2 <--(....stop.....)--- EMETTEUR      1.....
T = 197.00..... EMETTEUR      2 ---(....stop.....)--> EMETTEUR      3.....
Fin normale de la simulation
Temps courant : 200
```

Imprimé en France

par

l'Institut National de Recherche en Informatique et en Automatique