Technologie des applications client-serveur

UE RSX 102

Support de cours Tome 1

G. Florin, E. Gressier, S. Natkin

BIBLIOGRAPHIE

Les RFC UNIX/INTERNET

Les serveurs WEB ...

Coulouris , Dollimore "Distributed systems concepts and design" Addison Wesley 2000"

S. Natkin, "Les protocoles de sécurité de l'Internet" Dunod 2002.

Ed. Krol, “The whole Internet”, O’Reilly, 1992

R Orfali, D Harkey, J. Edwards, "The essential Client Server Survival Guide" Wiley 1995
En francais "Client-Serveur Guide de survie".

G Gardarin, O Gardarin, "Le client-serveur"
1996 Eyrolles

Introduction

Notions générales

Introduction
Situation actuelle

Mutation permanente des **concepts**, des **techniques** et des **organisations** associées aux applications informatiques.

Source principale un effort de **recherche** et de **production industrielle** sans précédent au niveau mondial qui permet:

- l'apparition de composants de rapidité et de complexité en croissance continue.

- la possibilité de développement de solutions logicielles et organisationnelles irréalistes quelques années auparavant.

Systèmes disponibles

On dispose à prix accessible pour les entreprises et le grand public de calculateurs puissants munis :

- de **systèmes d'exploitation** évolués

- d'interfaces **graphiques** évoluées

- de **capacités de stockage** énormes

- de moyens d'interconnexion "**réseaux locaux**" très performants

- dans un futur prévisible de moyens d'interconnexion à "**longue distance**" à bas prix et de bande passante importante.

Informatique "coopérative"

Applications coopératives ("Cooperative work")

Un ensemble d'entités logicielles **coopérant** au moyen d'un **réseau** à la réalisation d'une **tâche informatique**.

Algorithmique répartie ("Distributed Computing")

Les applications systèmes et réseaux indispensables au fonctionnement des machines en réseau.

IAD Intelligence artificielle distribuée, Multi-agent ("Distributed Artificial Intelligence")

Application coopérative mettant en relation des agents qui fonctionnent selon des approches dérivées de l'intelligence artificielle.

Informatique "coopérative" (suite)

Le calcul massivement parallèle ("Grid Computing")

Application de calcul scientifique réalisé par le travail en parallèle et en coopération d'un nombre élevé de processeurs.

Les systèmes répartis de contrôle commande de procédés industriels

Application de contrôle en temps réel de procédés tenant compte des contraintes de sûreté de fonctionnement.

Le client/serveur

Principal objet du cours

Le client serveur

Définir des **modèles de répartition**
des **données**
et des **traitements**
dans une **architecture réseau**.

Selon une approche **dissymétrique**:
- un client émet des **requêtes,**
- le serveur rend le **service** demandé.

Applicable à de nombreux problèmes de coopération,
en informatique de **gestion**
également aux autres domaines
algorithmique **répartie**,
informatique **industrielle**, ...

Dans son acception la plus complète:
possibilité de définir n'importe quelle **architecture de communication**.

- Chaque entité communicante est à la fois client et serveur
- Chaque entité émet et traite des requêtes

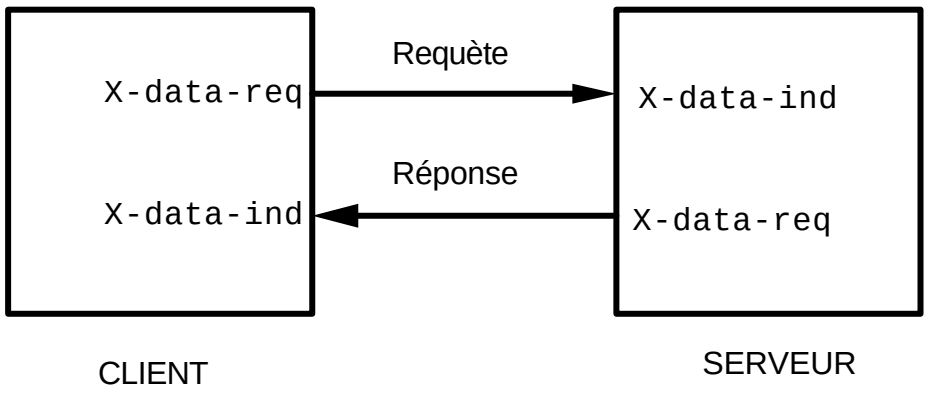
Communications et approche client-serveur

Client serveur en mode message

. Le client envoie dans un **premier message une requête** d'exécution d'un traitement à un serveur.

. Le serveur effectue le travail et fournit dans **un second message la réponse**.

On peut utiliser la communication par **messages** de données en mode **asynchrone** offerte par les protocoles de **transport**.

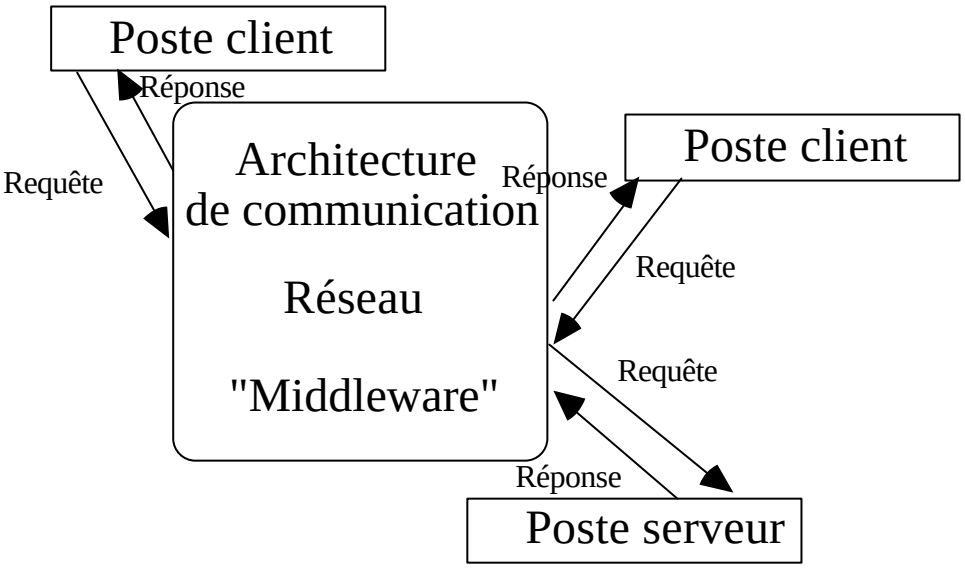


Client serveur en appel de procédure distante

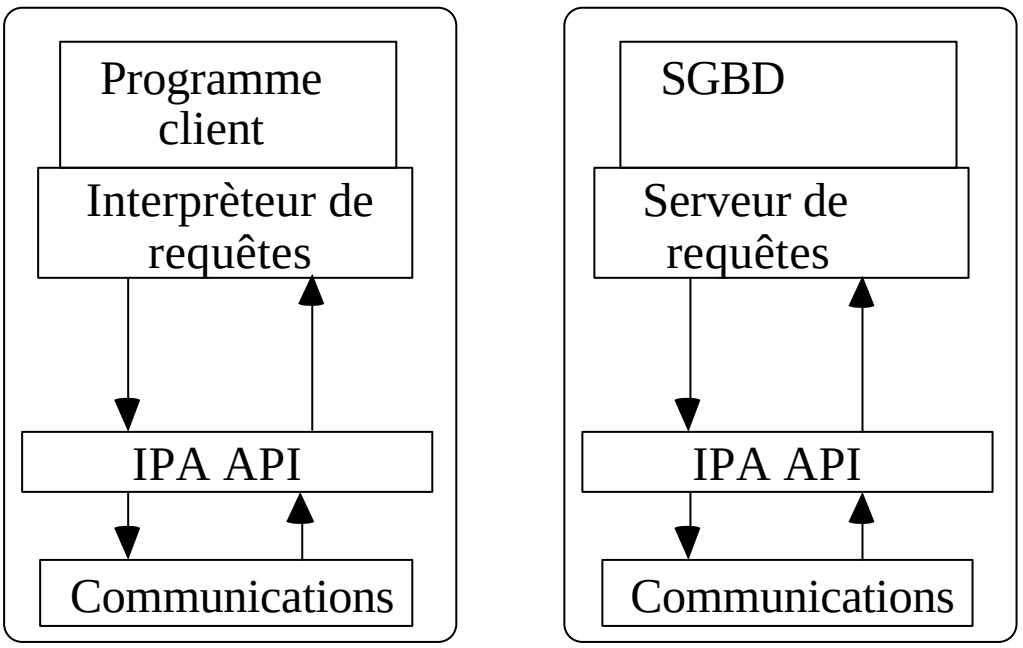
Présentation **syntaxique et sémantique** du mécanisme précédent **en terme d'appel de procédure distante.**

Architecture générale

Fonctionnement le plus fréquent en client-serveur: mode requête réponse pour une population de clients et de serveurs.



IPA Interface de programmation d'application API ("Application Programming Interface")



Applications types.

Transactionnel

OLTP "On Line Transaction Processing"

Les applications comportent:
- des opérations d'accès en lecture écriture à des bases de données,
- des traitements
- des affichages en mode graphique sur des postes de travail.

Aide à la décision

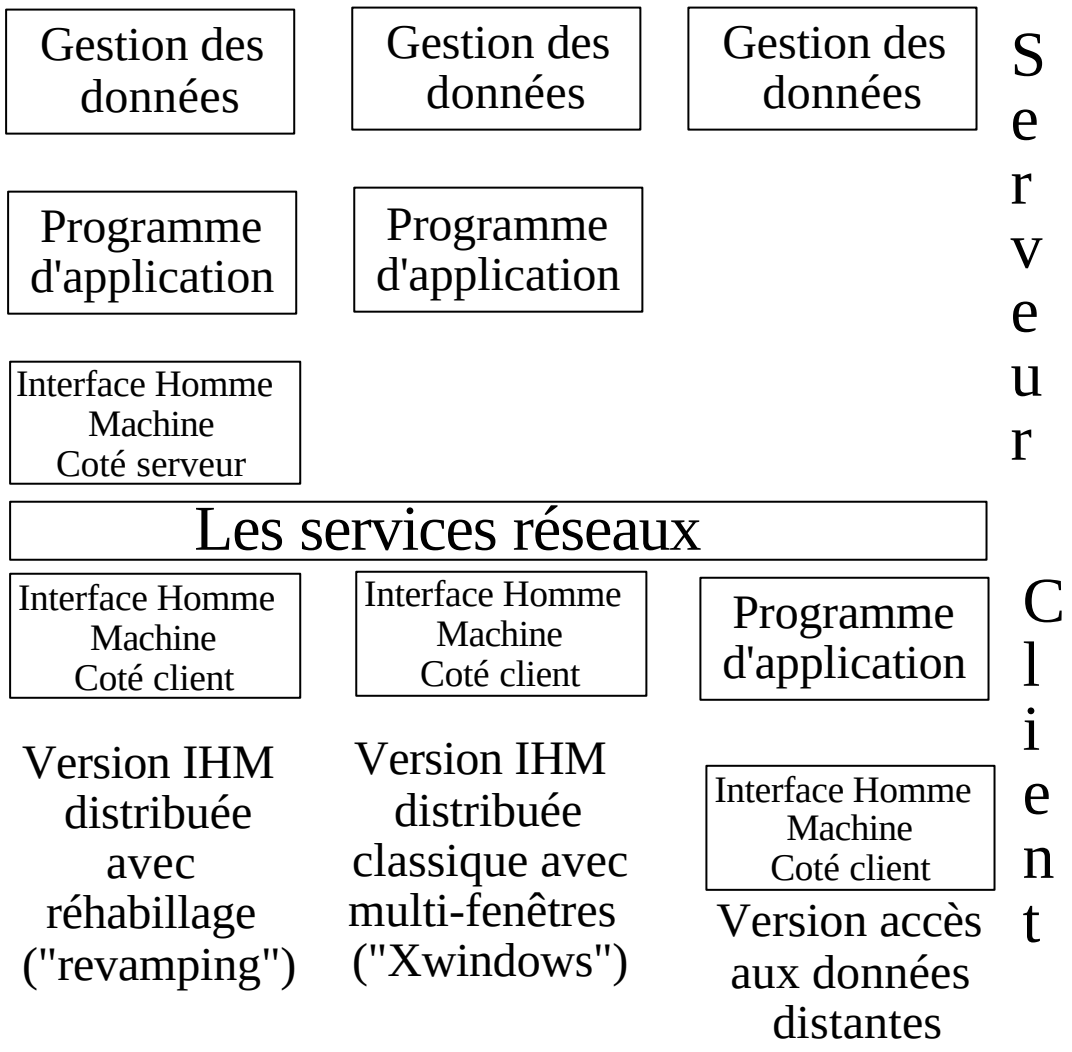
EIS/DS "Executive Information System/Decision Support")

Les applications comportent:
des accès complexes bases de données,
des traitements de synthèse/valorisation
des affichages (tableaux, histogrammes etc)

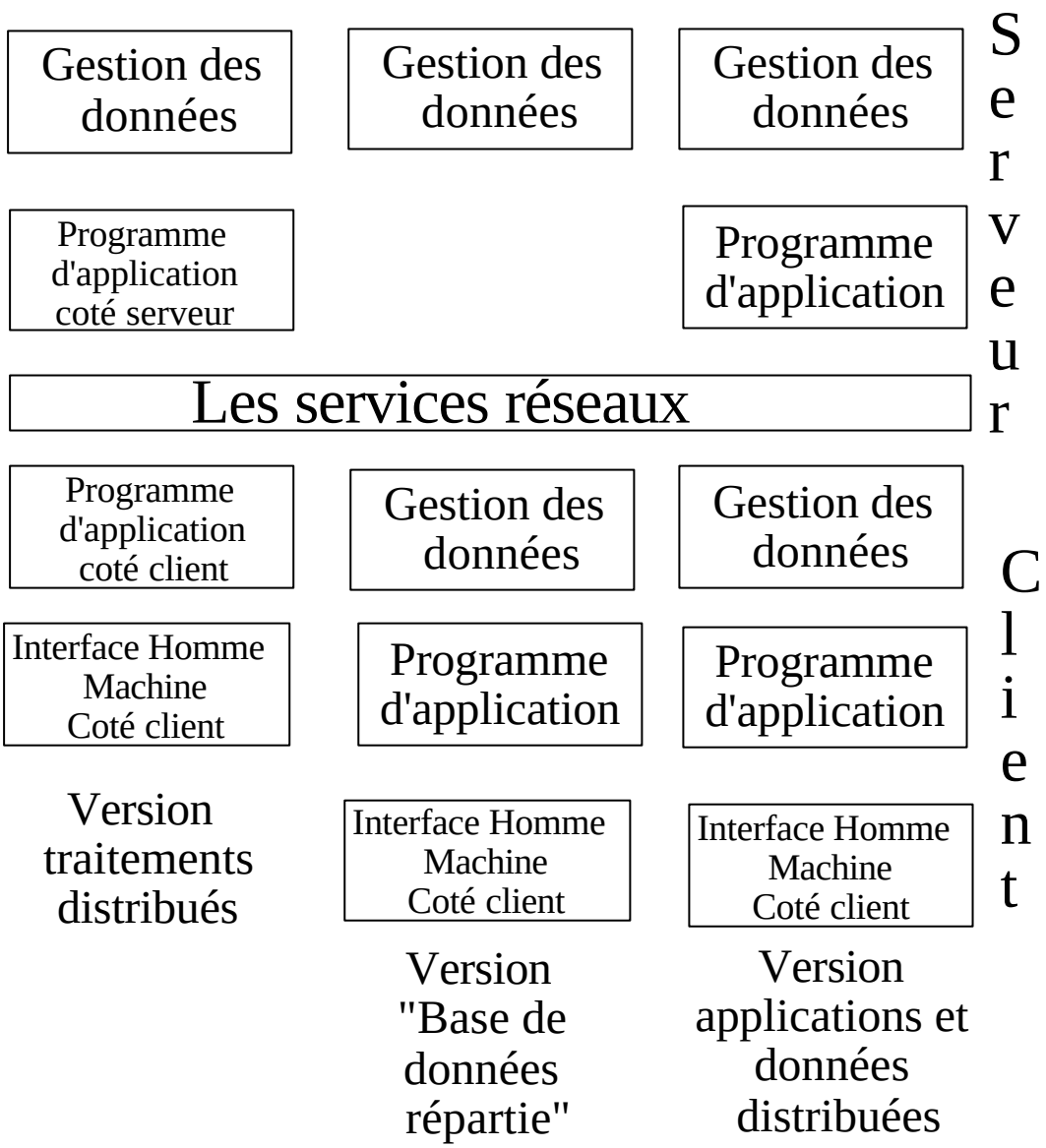
Le client-serveur en informatique de gestion

Les six versions du client serveur selon l'implantation des fonctions (Classification du Gartner Group)

Les trois versions de base



Les trois versions avancées



Les services réseaux orientés données ("Middleware de données")

Assurer les échanges de données entre un client et un serveur **en masquant les différents problèmes** potentiels liés à:

- la **répartition** des données et traitements (accès distant, baisse de performance)
- **l'hétérogénéité** des matériels et des logiciels en opération.

Exemple de travail le plus fréquent en client/serveur: envoyer des requêtes d'accès à des données (type SQL) d'un client vers un serveur et recevoir les résultats.

- **Ouvrir une connexion** entre entités
- Envoi de **requêtes SQL** vers le serveur
- **Conversion des formats** de requêtes
- **Exécution** des requêtes
- **Envoi** des résultats
- **Conversion des résultats**
- (Gestion des erreurs)
- **Fermeture de connexion**.

Problèmes à résoudre

Qui réalise les fonctions classiques des systèmes centralisés que l'on rencontre maintenant en univers réparti.

Quelques exemples

- Accès et partage des ressources

- Intégrité et cohérence des données

- Sécurité du système

- Gestion des performances

- Administration de l'ensemble

Fonctions qui regroupent des aspects multiples

- Contrôle d'exécution
- Contrôle de l'accès aux données
- Contrôle des communications

(à considérer simultanément).

Traitées par des outils différents

- Système d'exploitation local.

- Architecture de communication.

- Base de donnée (locale ou distribuée)

- Moniteur transactionnel

- Protocoles additionnels spécifiques

Un domaine considérable en cours d'établissement avec une terminologie considérable et très opaque (concepts, standards, produits)

Quelques exemples d'outils

- Les SGBD relationnels centralisés utilisés en accès distant au moyen du réseau par des stations (EDA-SQL + liaison APILINK).

- Les SGBD répartis (SGBD distribués accès à distance) (ORACLE).

- Les architectures de réseaux
(Internet avec l'application WEB + java).

- Les micro noyaux de systèmes (windows NT) répartis (Chorus).

- Les systèmes d'objets distribués
(OMG-CORBA, OLE COM).

Besoin d'une classification

La classification des réseaux

Différentes propositions de structuration des logiciels réseaux:

Solution **hiérarchique** (en couche).
Solution **"objet"**.

Normalisation / Système **propriétaire**

Expérimentation / Solution **confirmée**

Les piles de protocole

Une solution encore nécessaire pour comprendre les fonctions à réaliser:

- Choix du découpage des fonctions à réaliser entre **différentes couches de logiciels**.
- Choix de définition des **fonctions à réaliser par les différents niveaux**.
- **Implantation** des niveaux.

Exemple d'architecture importante:

INTERNET

INTERNET

<table>
<tr><td></td><td colspan="2">Applications TCP/IP directes</td><td>Applications pile SUN/OS</td></tr>
<tr><td>7. Application</td><td colspan="2" rowspan="3">EXEMPLES
SMTP "Simple Mail Transfer Protocol"
FTP: "File Transfer Protocol"</td><td>NFS: "Network File System"</td></tr>
<tr><td>6. Présentation</td><td>XDR:"External Data Representation"</td></tr>
<tr><td>5. Session</td><td>RPC:"Remote Procedure Call"</td></tr>
<tr><td>4. Transport</td><td colspan="3">TCP: Transmission Control Protocol (connecté)
UDP: User Datagram Protocol (non connecté)</td></tr>
<tr><td>3. Réseau</td><td colspan="3">IP: Internet Protocol</td></tr>
<tr><td rowspan="2">2. Liaison</td><td colspan="3">Encapsulation IP (sur LAN ou liaisons SLIP,PPP)</td></tr>
<tr><td rowspan="2">1. Physique</td><td colspan="3">Pratiquement tout support de transmission
Réseaux Publics Lignes spécialisées Point à Point Réseaux Locaux Réseau téléphonique RNIS, ATM</td></tr>
</table>

NIVEAU TRANSPORT (rappel)

. Le premier des niveaux utilisable par l'utilisateur pour développer des applications.

. **Il réalise un service de transmission fiable entre processus (transmission de bout en bout, "end to end").**

. Selon les options de conception il assure:

- **Gestion des connexions.**
  . protocoles en mode connecté
  . protocoles en mode non connecté.
- **Négociation de qualité de service**.
- **Multiplexage/éclatement**
- **Contrôle d'erreur**.
- **Contrôle de flux**.
- **Contrôle de séquence.**
- **Segmentation.**

Exemples de transports: Internet

TCP "Transmission Control Protocol".

UDP "User Datagram Protocol".

Novell

SPX "Sequenced Packet eXchange"

En fait XEROX XNS: SPProtocol

NIVEAU SESSION

Dans sa définition de base (OSI) le niveau session structure et synchronise les dialogues point à point en mode message.

L'approche OSI.

. Le transport offre **une voie logique** de communication ("**un tube**").

. Le mode de communication utilisé est **le mode message asynchrone.**

. **La session** structure les échanges pour y ajouter de la **tolérance aux pannes et de la synchronisation** :

- **Activités** (travail important)
- **Dialogue** (partie d'un travail)
- Notion de **point de synchronisation** pour délimiter des parties d'un échange en vue de la reprise.

Difficulté majeure de l'approche OSI Manque de **fonctions de synchronisation**. **Normes** CCITT X215 ISO 8026 Service X225, 8027 Protocole de session

APD Appel de Procédure Distante "RPC Remote Procedure Call"

. Le mode de communication du transport étant **le mode message asynchrone,** la session est définie pour offrir à l'usager un mode de dialogue de type synchrone.

=> Permettre à un usager d'exécuter une procédure ou une fonction sur un autre site:
. **Problème délicat :** Assurer en environnement réparti une sémantique pour l'appel de procédure distante voisine de celle connue en univers centralisé.

- Exemples : Rpc traditionnels
SUN-OS : Internet Protocole RPC
OSF-DCE : "Distributed Computing Environment"

- Exemples : Systèmes d'objets répartis
JAVA RMI :"Remote Method Invocation"
CORBA :"Common Object Request Broker Architecture"
WS-SOAP :"Web Services - Simple Object Access Protocol"

NIVEAU PRÉSENTATION

Traite du codage des données échangées: différents sites ayant des représentations différentes peuvent utiliser les données.

Les conversions

. Nécessaire pour tous les types de données
Types caractères, numériques, construits.

Syntaxe abstraite : Permet la définition d'une grande variété de structures de données (analogue de la syntaxe de définition de types dans un langage évolué).

Syntaxe de transfert : Représentation unique dans le réseau des valeurs échangées pour transférer les données.

Exemples de standards de conversion

- Réseaux publics

Syntaxe abstraite ASN1: **X208**

Syntaxe de transfert :**X209**

- SUN-OS/ Internet

XDR : "eXternal Data Representation".

- Systèmes d'objets répartis CORBA

IDL : "Interface Definition Language".

CDR : Common Data representation.

- Web services

WSDL : "Web Services Definition Language".

RPC/Encoded : "Syntaxe de transfert XML"

Les protocoles de sécurité

Objectif

Echanger des informations de façon sécuritaire en résolvant des problèmes de:

- **confidentialité**
- **intégrité**
- **authentification**

Problème central du développement des applications réseau par exemple de commerce électronique.

Exemples

Utilisation de techniques de cryptographie
DES, RSA, ...

Définition de protocoles de sécurité

Au niveau session SSL (Secure Socket Layer)

Aux autres niveaux (L2TP, IPSEC, SHTTP, Kerberos ….)

NIVEAU APPLICATION

Le niveau application est défini pour fournir à l'utilisateur des fonctions dont il a besoin couramment.

- **En termes d'un cadre** de développement d'une application informatique répartie (Exemple: structuration objet).
- **En termes de protocoles**

fonctions réseaux pré définies qui déchargent l'utilisateur de travaux répétitifs de programmation d'applications souvent utilisées.

On distingue aussi les applications qui concernent des échanges automatisés entre calculateurs de celles qui concernent le travail des personnels.

- **Travail Collectif,"Groupware"**

Échange de documents multimédia
Planification des activités ("Workflow")
Courrier électronique, Télé conférence
Gestion des agendas

Protocoles d'application

Le transfert de fichiers

Objectif

Déplacer des fichiers d'un site à un autre (plats en général).
Très nombreux protocoles proposés

Exemples

Internet FTP "File Transfer Protocol"
OSI/IEC FTAM "File Transfer Access and Management"

L'accès aux fichiers distants

Objectif

Accès unifié en univers réparti à différents fichiers (réalisation des requêtes d'accès).

Exemples

SUN-OS NFS "Network File System"
OSI/IEC FTAM "File Transfer Access and Management"

La gestion transactionnelle répartie

Objectif

La cohérence,
La persistance des données,
L'optimisation des performances.

Assurer le maintien cohérent d'un ensemble de données réparties en présence de nombreuses transactions concurrentes et de pannes.

Exemples

En univers Ouvert

Normes OSI/IEC TP et XOPEN/XA
"Transaction Processing"

Produits propriétaire:

Moniteurs IBM CICS, TUXEDO

L'accès aux bases de données distantes

Objectif

Permettre à un client d'accèder à une base de données distante (le plus souvent au moyen de requêtes SQL)

Exemples

Normalisation de facto Microsoft

ODBC

"Open Data Base Connectivity"

Proposition du groupe SAG "SQL Access Group" d'un ensemble de requêtes CLI "Call level Interface".

Autre proposition

Consortium Borland, IBM, Novell.
IDAPI : "Integrated Database Application Programming Interface"

Produits propriétaires

EDA-SQL de Information Builders Inc
Interface de communication API/SQL connectant de très nombreuses bases de données sur des plates formes clientes.

La désignation

Objectif

Gérer des **annuaires** permettant à un utilisateur de retrouver des attributs d'un nom symbolique (principalement l'adresse réseau).

Exemples

Internet **DNS** "Domain Name System"
OSI/IEC **DS** "Directory Services"
Annuaire **LDAP** "Lightweight Directory Access Protocol'

La messagerie

Objectif

Permettre d'échanger du courrier électronique entre usagers

Exemples

Internet **SMTP** "Simple Mail Transfer Protocol"
OSI **MHS** "Message Handling System"

L'échange de données informatisées

Objectif

Permettre d'échanger des documents administratifs standards sur des réseaux de transmission de données (commandes, factures, ...) entre agents économiques.

Exemple

Normes EDI "Electronic Data Interchange"

L'échange de documents électroniques

Objectif

Définir et échanger des documents généraux structurés de types particuliers (page, lettre, journal, livre, catalogue).

Exemples

HTML "HyperText Markup Language"
XML "eXtended Markup Language"
Très nombreuses propositions de formats.

L'accès aux informations distantes WWW "World Wide Web"

Objectif

Permettre l'accès à des données distantes pour des personnes.

- Systèmes de désignation **URL** ("Uniform Resource Locator"),
- Protocole de communication **HTT**P (Hyper Text Transfer Protocol"),
- Initialement format **HTML** + format **MIME** Extension format **XML**

Extension : permettre les communications entre programmes : notion de services web (**SOAP** ‘ Simple Object Access Protocol’, **WSDL** ‘Web Services Definition language’ **UDDI** 'Universal Description Discovery and Integration').

L'administration de réseaux

Objectif

Permettre l'accès à des variables gérées par des agents d'administration:

- Lecture de l'état d'un appareil, de statistiques de fonctionnement.
- Positionnement de variables.

Internet SNMP : 'Simple Network Management Protocol'.

WBEM: 'Web Based Enterprise Management'

Conclusion

Les réseaux et systèmes répartis

Un problème extrêmement complexe au centre de l'informatique actuelle et à venir.

Une évolution permanente des concepts des outils et des différentes propositions commerciales.

Un marché très concurrentiel qui induit une compétition souvent inutile dans l'offre, une opacité importante dans les fonctions offertes par les produits.

Premier Chapitre

L'INGÉNIERIE DES PROTOCOLES DE COMMUNICATION

Plan du cours

Première partie

I.1 Le mode message asynchrone.

I.2 L'interface socket

Seconde partie

II.1 Le mode appel de procédure distante.,

II.2 Exemple: CORBA

INTRODUCTION

- **Le contrôle réparti** c'est l'ensemble des moyens offerts à un programmeur pour **développer des applications dans un univers réparti** (réseau, système réparti,..)

Application utilisateur

Interface de Programmation d'Applications **(IPA - API)**

Mécanisme d'exécution
Mécanisme de communication
Mécanisme de mémorisation

Démarche descendante du développement logiciel

- La conception de l'application.

- Les étapes de raffinement.

- La programmation : l'adaptation finale aux outils disponibles (syntaxe et sémantique de l'IPA)

I Les moyens de développement disponibles

Introduction Aux IPA - Distribuées

"DAPI Distributed Application Programming Interface"

Syntaxe et sémantique des moyens de programmation **utilisables pour l'expression d'un comportement réparti**

Comportant donc nécessairement des fonctionnalités de **communication et de synchronisation.**

=> La description des interfaces de programmation d'application distribuées devrait comprendre:

. **Les comportements en mode normal** de fonctionnement.

. **Les comportements en présence de pannes.**

. **Le comportement temporel.**

Styles de description des interfaces

Approches services systèmes/réseaux

Les plus répandues:

Sémantiques plus simples

Accessible de tous les langages

Mais contrôles limités

Problèmes de désignation.

Exemples. Interface socket (**TCP** Internet).

Approches langages

Très nombreuses propositions

Vérifications de cohérence

Désignation plus facilement résolue

(pour une application complètement définie)

Syntaxes/Sémantiques complexes

LA COMMUNICATION DANS L'INTERFACE DE PROGRAMMATION

- La communication par messages : l'outil de base (Système RC4000 [Brinch Hansen 1970])

- Développement d'interfaces de programmation d'applications réparties de complexité croissante:

En termes de richesse du schéma de communication/synchronisation offert.

Impliquant un nombre de messages échangés de plus en plus élevé.

. Le mode message asynchrone.

. Le mode rendez-vous.

. Le mode appel de procédure distante.

. Le mode mémoire partagée répartie

=> Modes de communication encore le plus souvent définis en point à point.

=> Evolution en cours vers des interactions de groupes.

Critères de comparaison des outils

Facilité d'utilisation

Syntaxe des primitives de service

Compréhension de la sémantique

Simplicité de la désignation.

Facilité d'extension à l'univers réparti d'applications existantes

Efficacité (performances)

Richesse des mécanismes

de synchronisation (des propriétés d'ordre)

Interopérabilité (normalisation réelle)

Comparaison des interfaces

Chaque mode présente des **avantages et des inconvénients.**

Possède des **défenseurs et des détracteurs**

L'étude comparative des différents modes **est très délicate.**

La situation est très **évolutive.**

=> Pas d'inutiles débats.

=> Faire des essais.

=> Disposer de plusieurs modes.

Utilisation de modes différents.

Problèmes à résoudre :

Pour une utilisation simultanée de **plusieurs modes** de communication.

=> Développement de **méthodologies** de spécification, conception, programmation **supportant plusieurs sémantiques.**

I.1

LE MODE MESSAGE ASYNCHRONE

INTRODUCTION

- Basé directement sur le mode de communication par message (un message).

- Un service comprenant essentiellement deux primitives pour communiquer et se synchroniser.

TYPE COM_MESSAGE_ASYNCHRONE;
    METHOD envoyer (id_émetteur, id_récepteur, message, compléments);
    METHOD recevoir (id_émetteur, id_récepteur, message);
    METHOD ...;
END COM_MESSAGE_ASYNCHRONE.

identificateurs : ports de communication
messages : zones de données (typées).
compléments : selon les sémantiques multiples définissant les qualités de services

Nombreuses variantes sémantiques du mode message asynchrone

- Propriétés **d'ordre**.

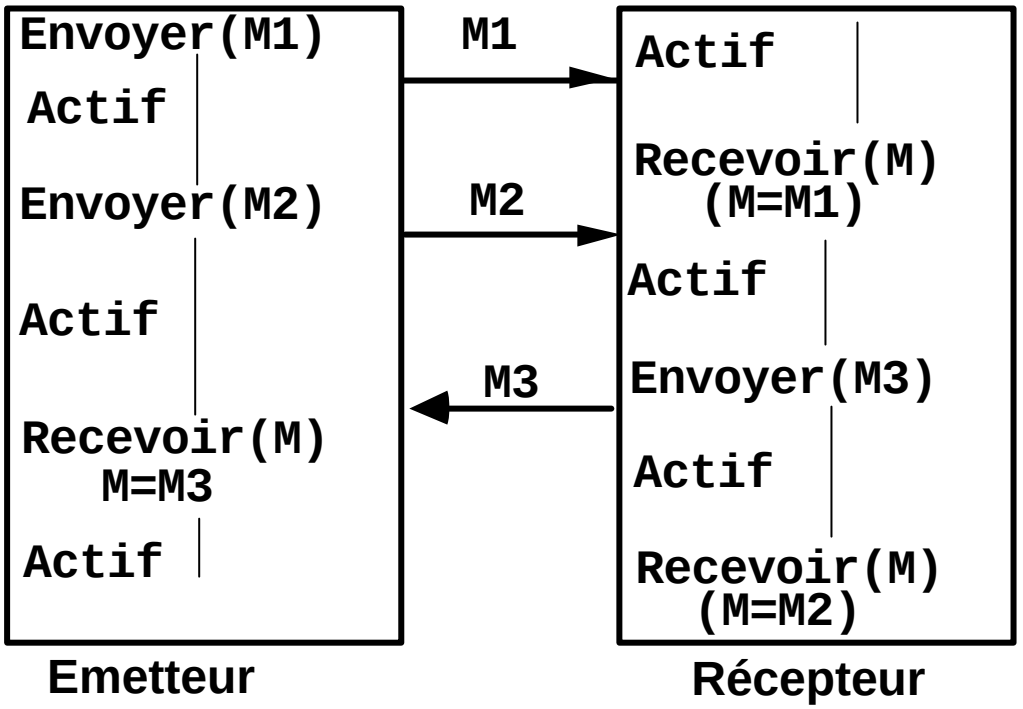
- Propriétés **de tolérance aux pannes**.

- Propriétés **temporelles**

- Autres **qualités de service.**

- Problème abordé ici: **la synchronisation et l'utilisation des primitives**.

Aspect principal pour la communication et la synchronisation: l'asynchronisme entre l'émetteur et le récepteur



- Le mode message asynchrone réalise un **"producteur-consommateur" réparti** entre un émetteur et un récepteur.

- Complet parallélisme autorisé entre l'émetteur et le récepteur.

Rien n'empêche aussi l'une des entités (l'émetteur) de suspendre son exécution.

Détails du comportement

L'émetteur

. Ayant demandé une émission, **reprend la main et continue son exécution** ***"immédiatement"*** **après** la prise en compte de sa demande.

. Le message est transmis (au rythme du transport d'informations par le réseau de communication) donc de façon **asynchrone** avec le comportement émetteur**.**

Le récepteur

Ayant décidé de prendre en compte un nouveau message il acquiert un message (le premier) en instance.

-celui qui se présente à après le recevoir

-un message qui se trouve dans une file d'attente de réception.

Utilisation du mode message asynchrone Aspect d'activation à distance

- Les messages sont le plus souvent typés:

Ils sont **une demande de traitement distant** associé au typage

Utilisant les données du message pour paramétrer l'exécution distante.

- La **nature du traitement déclenché** dépend du contexte dans lequel le message est pris en compte.

Sémantique de l'activation

a) De type **activation en parallèle ("fork")** puisque en mode normal l'émetteur et le récepteur continuent.

b) Interprétable également comme un **branchement inconditionnel ("goto")** à distance si l'émetteur se suspend définitivement après l'émission.

Aspect d'échange de données

- Le mode message permet une opération d'affectation de variable à distance:

envoyer (M=écrire(v) , id_dest)

recevoir (M'=lire(v) , id-emet)

. Avec possibilité de **gestion de cohérence des types** des variables v.

. Avec **une sémantique de consistance des données entre l'émetteur et le récepteur très faible.**

Repose sur les propriétés d'ordre, temporelles, de tolérance aux pannes spécifiées par la qualité de service.

Propriété minimale de cohérence des communications point à point: la causalité

envoyer (M=écrire) -> recevoir (M'=lire)

$ t1 : émetteur.écrire (v,t1)

=> $ t2>t1 : recepteur.lire (v, t2)

Propriétés supplémentaires: rendez-vous, mémoire répartie.

Spécification des applications utilisant le mode message asynchrone

Principes Généraux

Solution Des Automates Synchronisés

Comportement séquentiel des processus:

=> Ensemble de processus (entités réseaux) **séquentiels** communicants (très souvent en mode point à point deux entités seulement).

=> Les méthodes de spécification de chaque processus utilisent des **automates d'état fini**

Interaction entre processus par échange de message

=> Les processus communicants se synchronisent par messages asynchrones.

Représentation graphique des applications: automates

États

Il est défini par **un ensemble significatif de variables locales** de chaque processus.

Chaque état doit être **interprétable aisément** en terme d'évolution locale de l'application répartie.

Choix des états:

Étape préliminaire importante de la spécification

. Pas un niveau de détail trop fin (trop grand nombre d'états) mais suffisamment pour représenter l'évolution au niveau de détail souhaité.

Transitions.

Elles comportent deux mentions:
les conditions, et les actions.
Elles sont représentées par les arcs du graphe.

Condition

Les transitions sont franchissables lorsqu'une condition booléenne ou garde est satisfaite:

. condition booléenne portant sur les variables locales.

. condition booléenne portant sur les échanges (requête de service, message entrant).

. condition booléenne portant sur les signaux internes (horloge).

Action

- C'est l'opération réalisée lors du franchissement de la transition: les traitements à réaliser lors du passage à l'état suivant (manipulation de variable, envoi de message).
- Le détail des actions ne doit pas être important sinon le modèle est illisible (renvoyer à des textes).

Les commandes d'échanges

- Représentation des opérations sur les diagrammes (conditions et actions)

condition recevoir(M) ?M

action envoyer(M) !M

- Désignation

- La commande d'émission (d'une entité P1) doit désigner un destinataire (P2).

Ex :Dans le processus P1 :P2!M1

- La commande de réception (exécutée par un processus P2) doit évoquer une source P1.

Ex :Dans le processus P2 :P1?M1

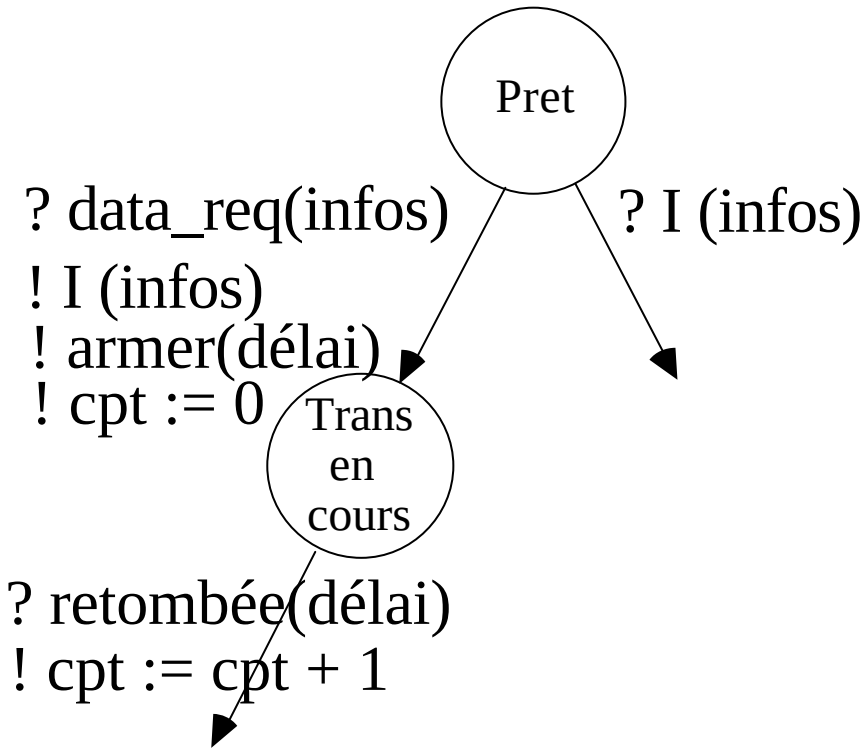
Typage

Il doit y avoir correspondance de type entre la variable de réception et la valeur émise (possibilité de variable article).

Type[message_émis] =

Type [message_reçu]

Exemple



Sémantique de l'alternative constituée par l'existence de plusieurs transitions en un état

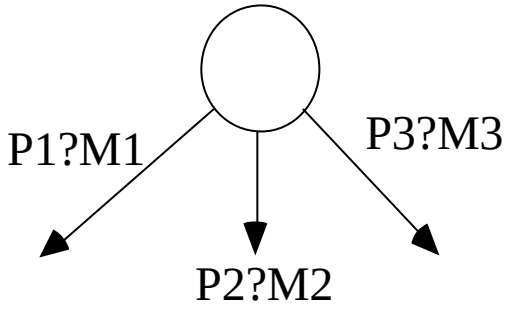
Évaluation des gardes (conditions booléénnes)

. Une garde est **franchissable** si elle est constituée par **une expression booléenne** portant sur des variables locales à **valeur vrai**.

. Une garde est **franchissable** s'il s'agit d'une **commande d'échange** satisfaite : en fait une réception dont le processus source à fait parvenir un message du type attendu qui se trouve en tête de la file d'attente.

. Une garde **combinaison des deux cas précédents.**

Alternative en un état



Indéterminisme de l'évaluation

- On choisit **l'une quelconque des transitions ayant sa condition booléenne satisfaite** (sa garde ouverte). Une garde franchissable quelconque est sélectionnée

=> **La liste des commandes** associée à la garde est **exécutée**.

- Aucune garde n'est franchissable mais il existe des gardes avec condition de réception pouvant être satisfaite par une émission

=> **Attente que l'une d'elles soit satisfaite.**

- Aucune garde n'est franchissable:

. elles sont toutes booléennes

. aucun message ne sera envoyé qui corresponde à une réception attendue

=> **Erreur d'exécution.**

Différents automates

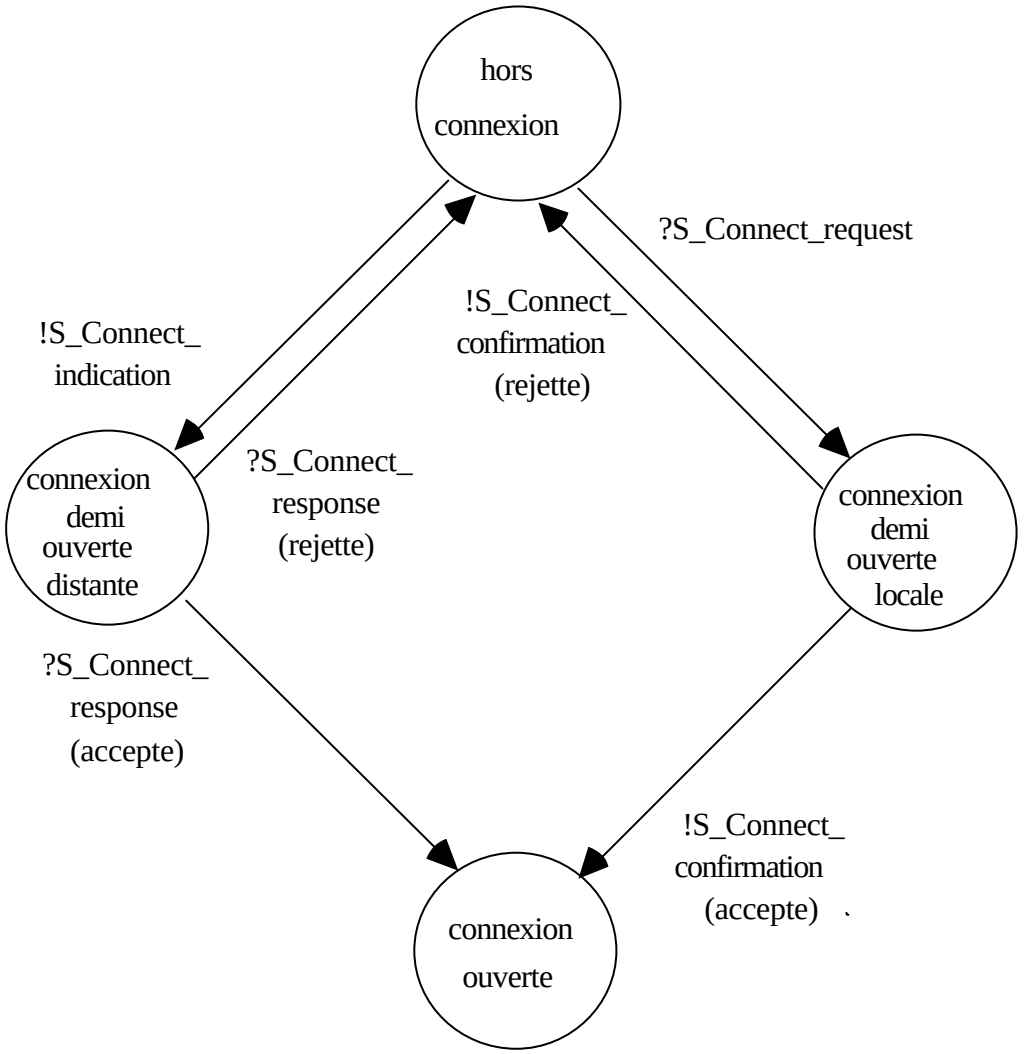
Automate du service

. Établissement de **la liste des unités de service** (primitives, SDU) de service échangeables entre le niveau n et le niveau n+1.

. **Définition des enchaînements autorisés** de primitives sous la forme d'un automate d'état.

=> Toutes les successions légales qu'un observateur de l'interface n n+1 peut observer.

Exemple de comportements autorisés du service de connexion de session OSI.



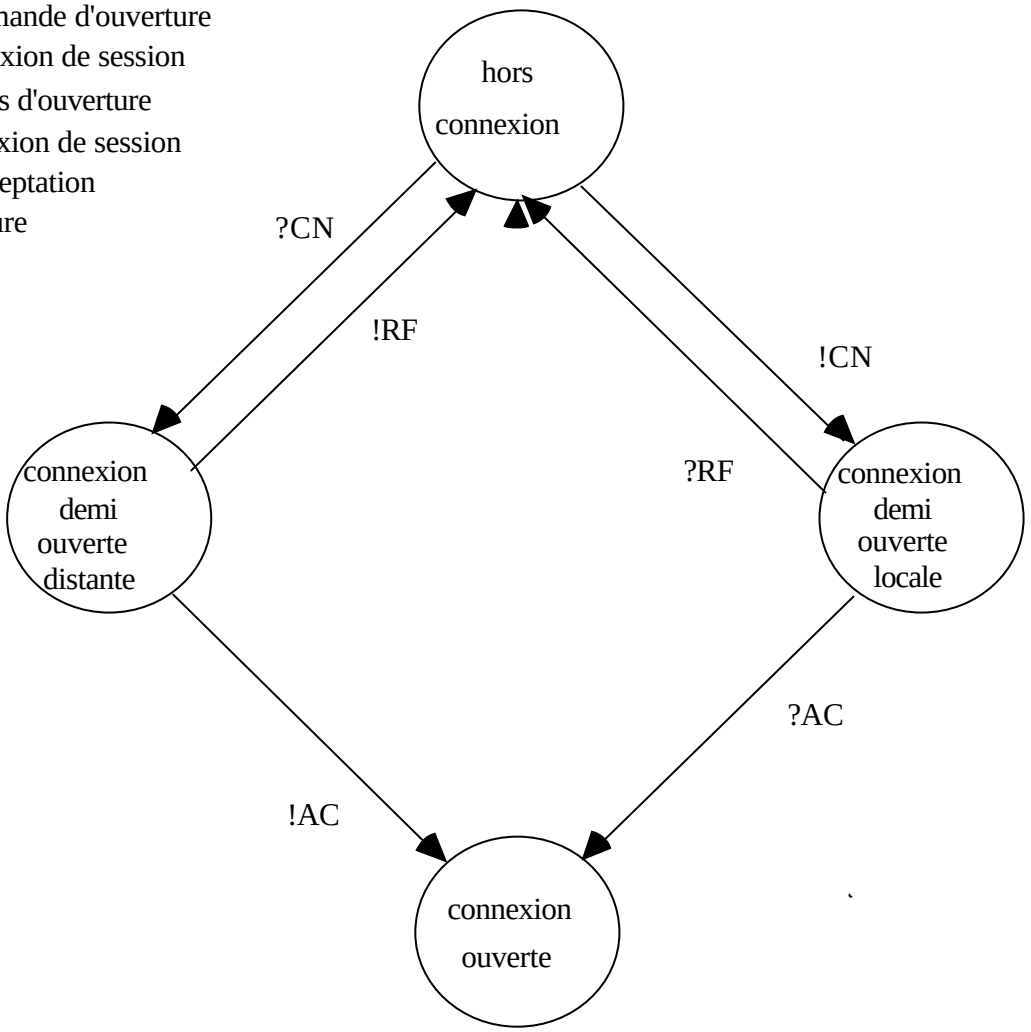
Automate du protocole

. Établissement de la liste des unités de protocole (messages, PDU) échangés entre deux niveaux n.

. Définition de tous les enchaînements de PDU qu'un observateur de la voie de communication peut observer).

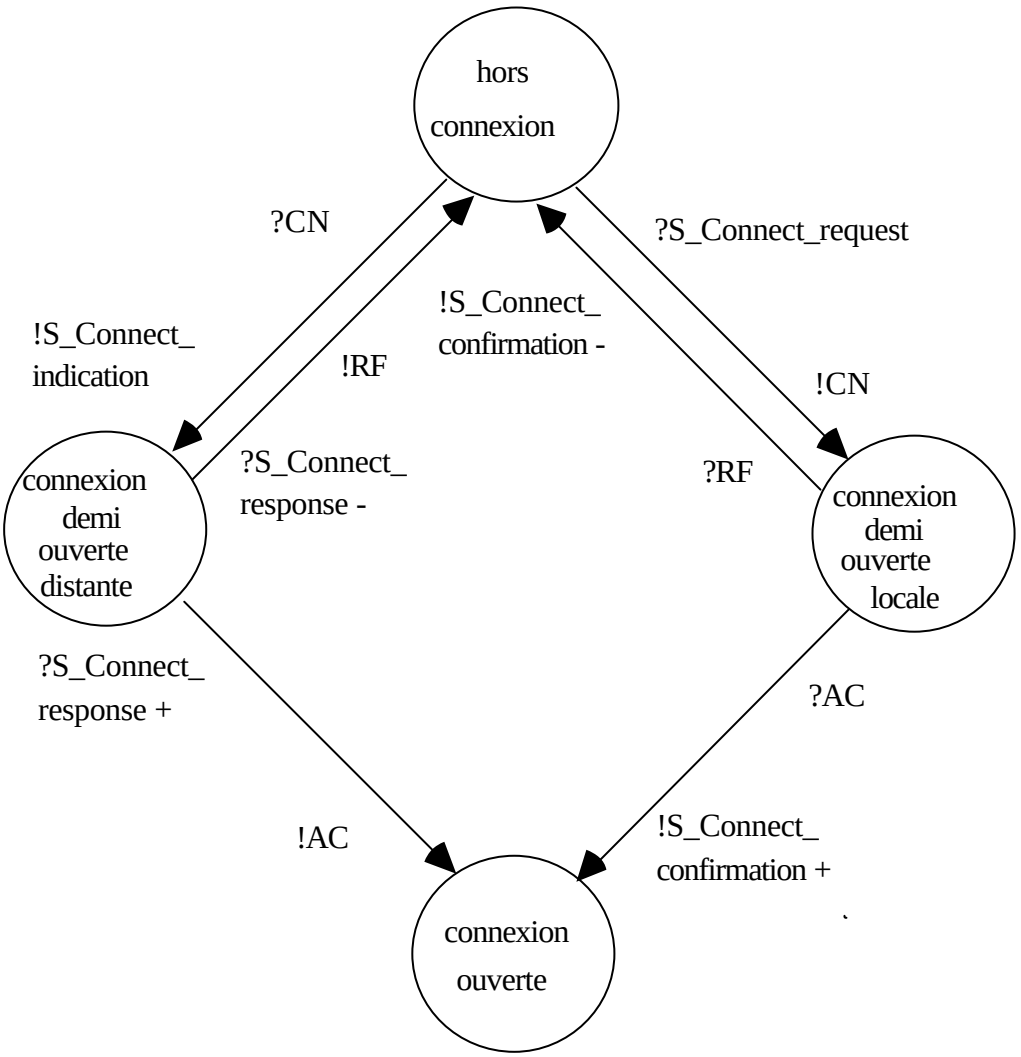
Ex: Une partie du protocole d'ouverture de connexion de session

CN: Demande d'ouverture de connexion de session
RF: Refus d'ouverture de connexion de session
AC: Acceptation d'ouverture



Automate complet

. Réunion des deux automates de service et de protocole



Validation des spécifications

Problèmes de constructions

Réceptions non spécifiées

Dans un état un message peut se présenter dont le cas n'a pas été prévu

Interblocages

Dans un état un message (ou une configuration) est attendu qui ne peut jamais se présenter.

Plus généralement

Définition d'assertions de bon fonctionnement sur le modèle à automates communicants.

- assertions portant sur des variables d'état (booléennes)

- assertions portant sur des trajectoires (logique temporelle)

Méthodes de validation systématiques employées

Simulation comportementale

Exécution en **simulation** du comportement décrit par les automates: une partie seulement des comportements sont couverts.

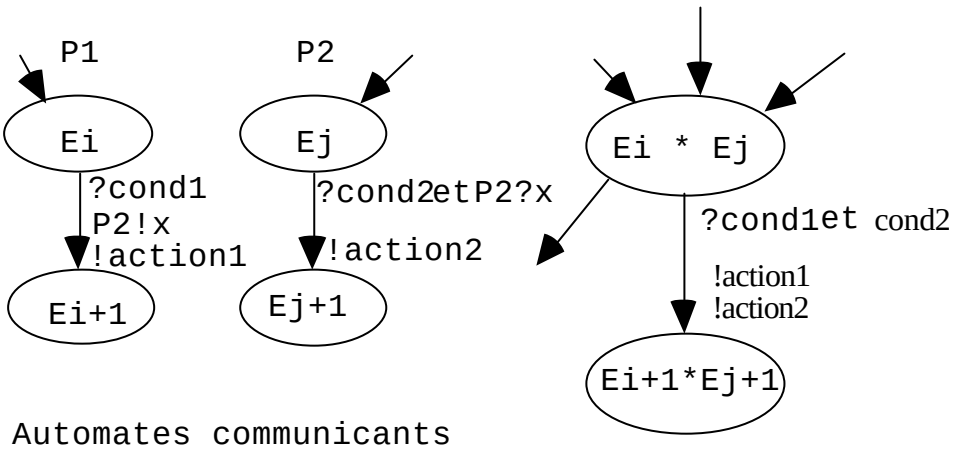
Détection d'erreurs et correction (technique s'apparentant au test du modèle).

Difficulté majeure : On ne connaît pas la couverture du test (le taux d'erreurs résiduelles après le test).

Analyse exhaustive.

- L'application se présente comme un ensemble d'automates communicants

- Réalisation du produit des automates (produit "synchronisé" tenant compte des communications).



Automates communicants

- Obtention du graphe d'accessibilité du système complet

- Vérification d'assertions sur le graphe complet.

Difficulté majeure

Explosion combinatoire de la taille des espaces d'état.

Exemples d'IPA distribuées en mode message asynchrone

Presque toutes les IPA des architectures "ouvertes" de réseaux

- **Internet, OSI, SNA**

Presque toutes les IPA des systèmes répartis classiques,

- **Chorus , Mach, Amoeba**

De nombreux langages de programmations

- **Langages de spécifications de protocoles** (Estelle, ...)

- **Langages "acteurs" de l'intelligence artificielle distribuée** (Act1, ...)

Z; Font exception : les propositions récentes qui abandonnent le mode message:

Les interfaces **orientées RPC** (**orientées objets**), orientées **transactionnel** (**partage de mémoire)**.

Conclusion : mode message asynchrone

- C'est le mode le plus basique

Comparable à l'assembleur (affectation, branchement).

- Le moins contraignant il permet aux utilisateurs par des échanges successifs, la construction de tout type de protocole.

- L'utilisateur n'a pas en général envie d'être obligé de construire ses propres outils d'où:

L'enrichissement du mode message en termes de qualité de service par les couches successives des protocoles réseau.

Le besoin d'autres schémas prédéfinis plus complexes.

- Le mode message asynchrone est encore **le mode privilégié des interfaces**

On peut prévoir à terme son "**enfouissement**" dans les couches internes.