



Intergiciels (*Middleware*)

Introduction



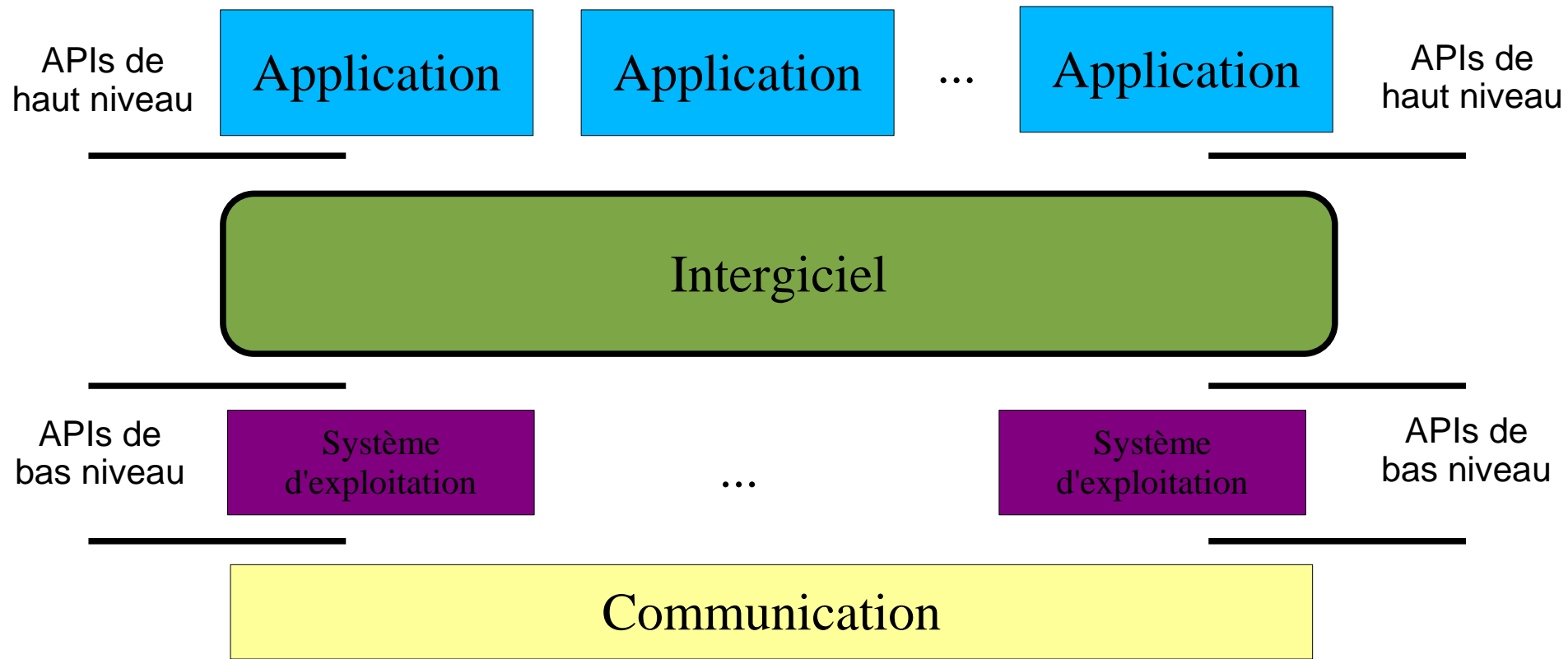
Intergiciels : séances

- Lundi 15/11 (13h30-17h30)
 - ♦ Cours « Intergiciels, RMI »
 - ♦ TP RMI
- Mercredi 22/11 (13h30-17h30)
 - ♦ TP RMI
- Lundi 29/11 (13h30-17h30)
 - ♦ Cours « Corba »
 - ♦ TP Corba
- Lundi 6/12 (13h30-17h30)
 - ♦ Exposés
- Mardi 7/12 (13h30-17h30)
 - ♦ Exposés
- Lundi 13/12 (13h30-17h30)
 - ♦ TP Corba

Pré-requis : Programmation Java

Introduction aux intergiciels

L'intergiciel (*Middleware*) est la « couche du milieu »





Fonctions d'un intergiciel

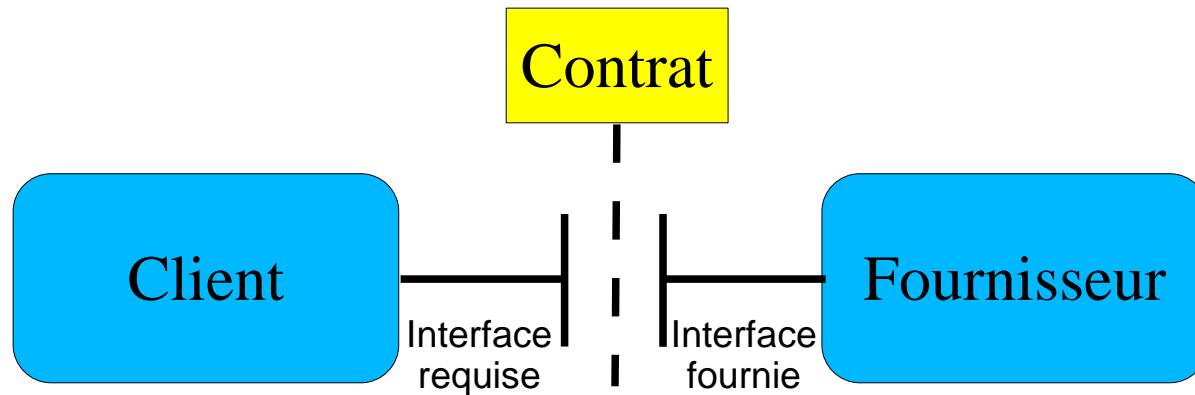
- Quatre fonctions
 - ◆ Fournir une interface ou API (Applications Programming Interface) de haut niveau aux applications
 - ◆ Masquer l'hétérogénéité des systèmes sous-jacents
 - ◆ Rendre la répartition invisible (« transparente »)
 - ◆ Fournir des services répartis d'usage courant
- Un intergiciel vise à faciliter la programmation répartie
 - ◆ Développement, évolution, réutilisation des applications
 - ◆ Portabilité des applications entre plate-formes
 - ◆ Interopérabilité d'applications hétérogènes



Services, contrats et interfaces

- Définition
 - ◆ Une application répartie est constituée d'un ensemble de « composants » mis en interaction
 - ◆ Un service est « un comportement défini par contrat, qui peut-être implémenté et fourni par un composant pour être utilisé par un autre composant, sur la base exclusive du contrat »
- Mise en oeuvre
 - ◆ Un service est accessible via une ou plusieurs interfaces
 - ◆ Une interface décrit l'interaction entre client et fournisseur du service
 - Point de vue opérationnel : définition des opérations et structures de données qui concourent à la réalisation du service
 - Point de vue contractuel : définition du contrat entre client et fournisseur

Définitions d'interfaces (1)



- La fourniture d'un service met en jeu 2 interfaces
 - ◆ Interface requise (côté client)
 - ◆ Interface fournie (côté fournisseur)
- Le contrat spécifie la compatibilité entre ces interfaces
 - ◆ Chaque partie est une boîte noire pour l'autre
 - ◆ Client et fournisseur peuvent être remplacés par des composants respectant le contrat

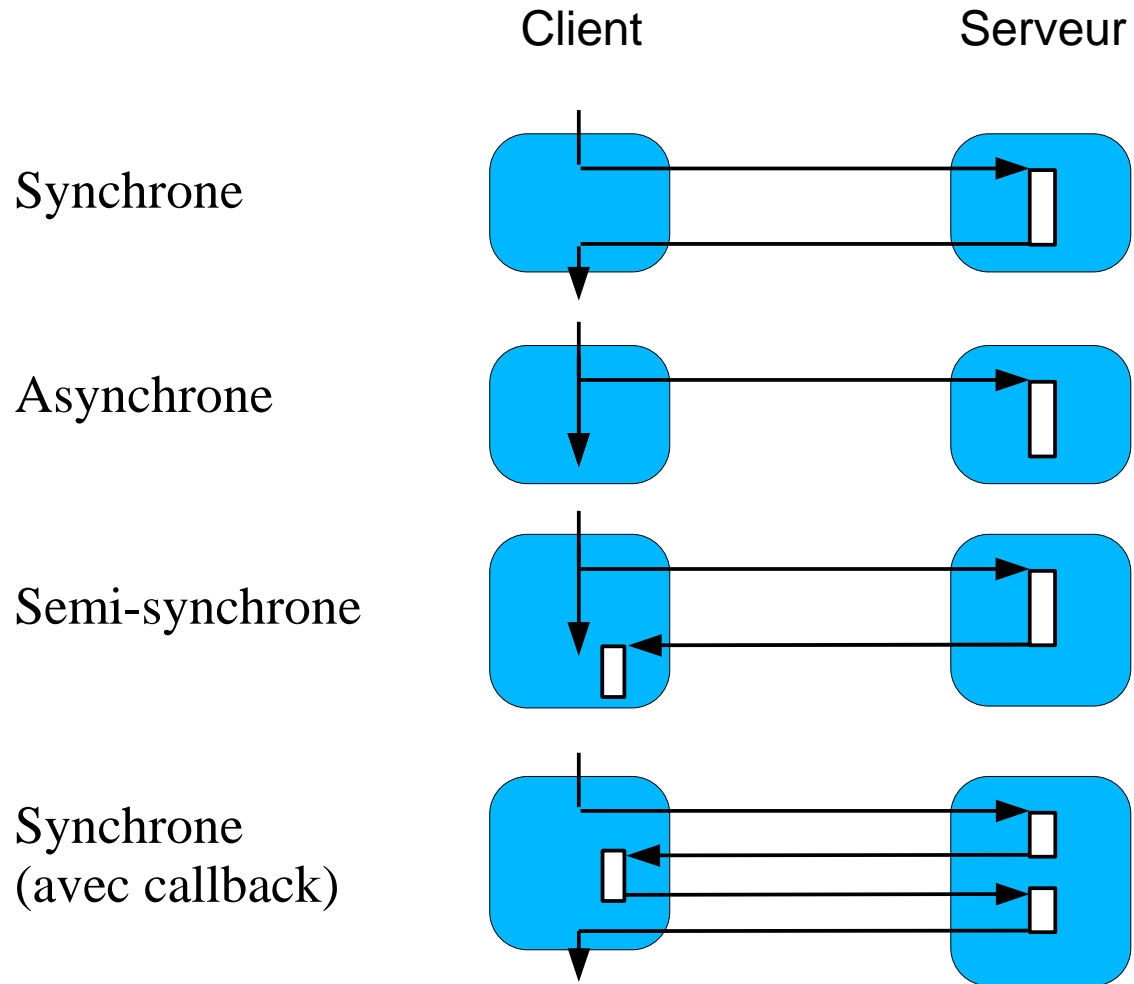


Définitions d'interfaces (2)

- Partie opérationnelle
 - ◆ Description dans un langage de prog., (e.g. interface Java)
 - ◆ Description dans un IDL (*Interface Definition Language*)
- Partie contractuelle
 - ◆ Plusieurs niveaux de contrats
 - Sur la forme : spécification de types -> conformité syntaxique
 - Sur le comportement : assertions -> conformité sémantique
 - Sur les interactions entre méthodes : synchronisation
 - Sur les aspects non fonctionnels : QoS, ...



Schémas d'interaction





Liaison répartie

La mise en relation d'un client et d'un fournisseur passe par plusieurs étapes

- ◆ La **résolution** qui transforme un *nom symbolique* ou une description en une *référence*

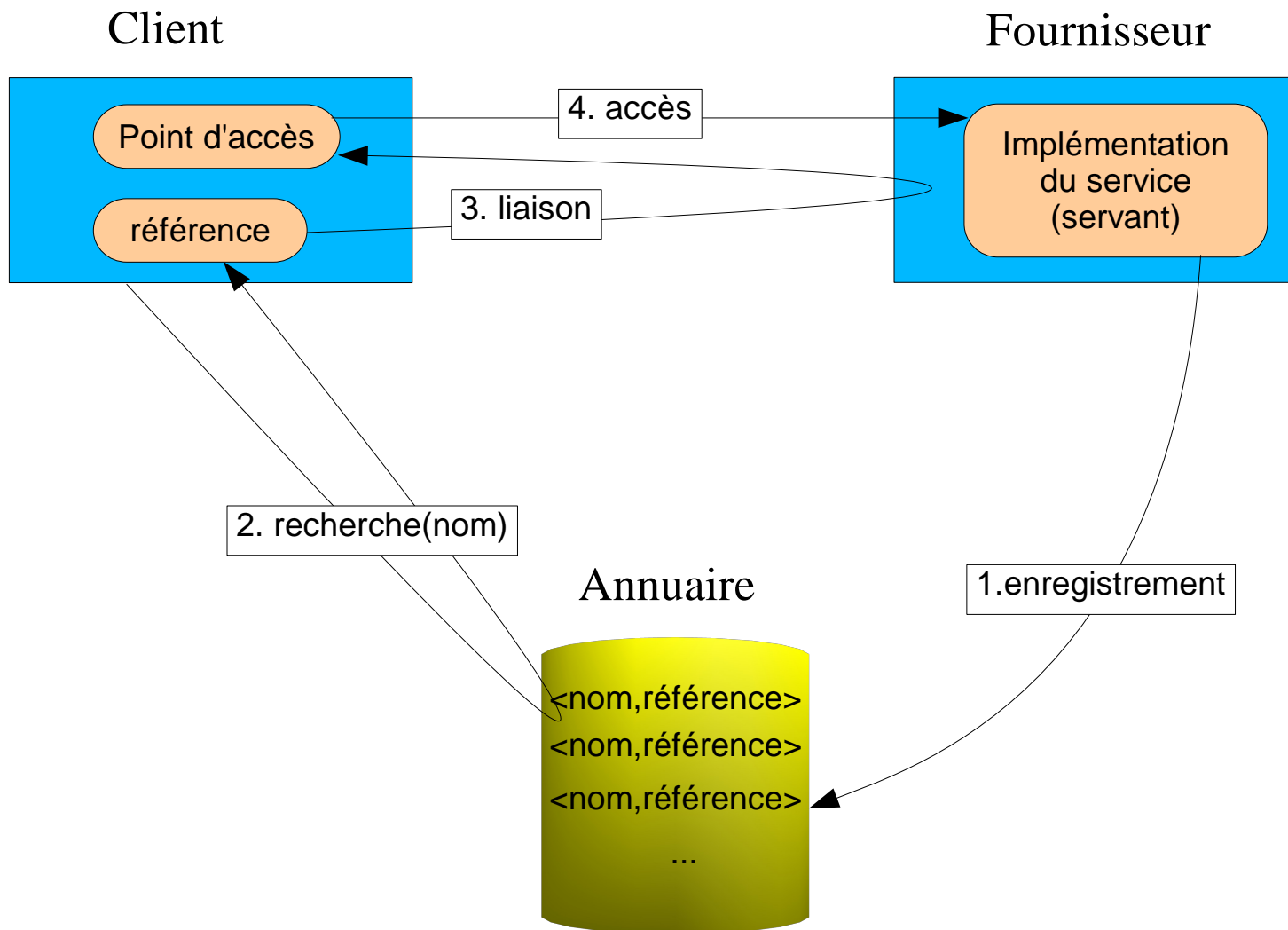
ex: rmi://myprinters/hp4550 -> 193.54.183.167

- ◆ La **liaison** qui crée chez le client un *point d'accès* au fournisseur à partir d'une *référence*

ex: 193.54.183.167 -> _PrinterStub.class



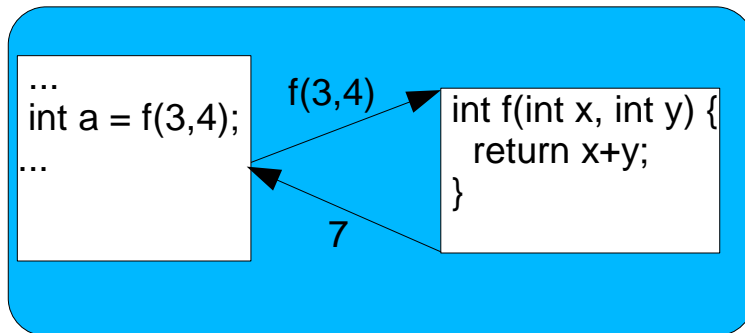
Annuaire



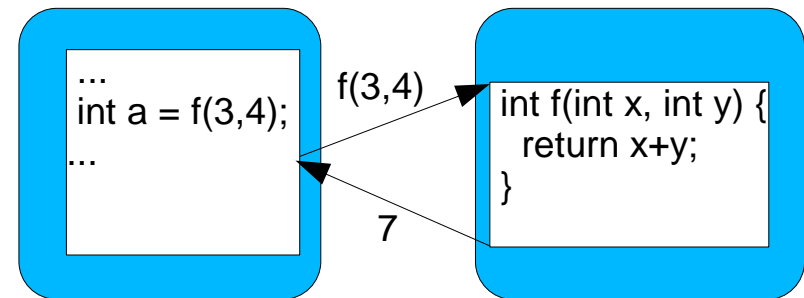
Remote Procedure Call (RPC)

L'appel de procédure à distance (RPC) permet d'invoquer une fonction distante dans des applications client-serveur écrite dans un langage procédural.

Appel local



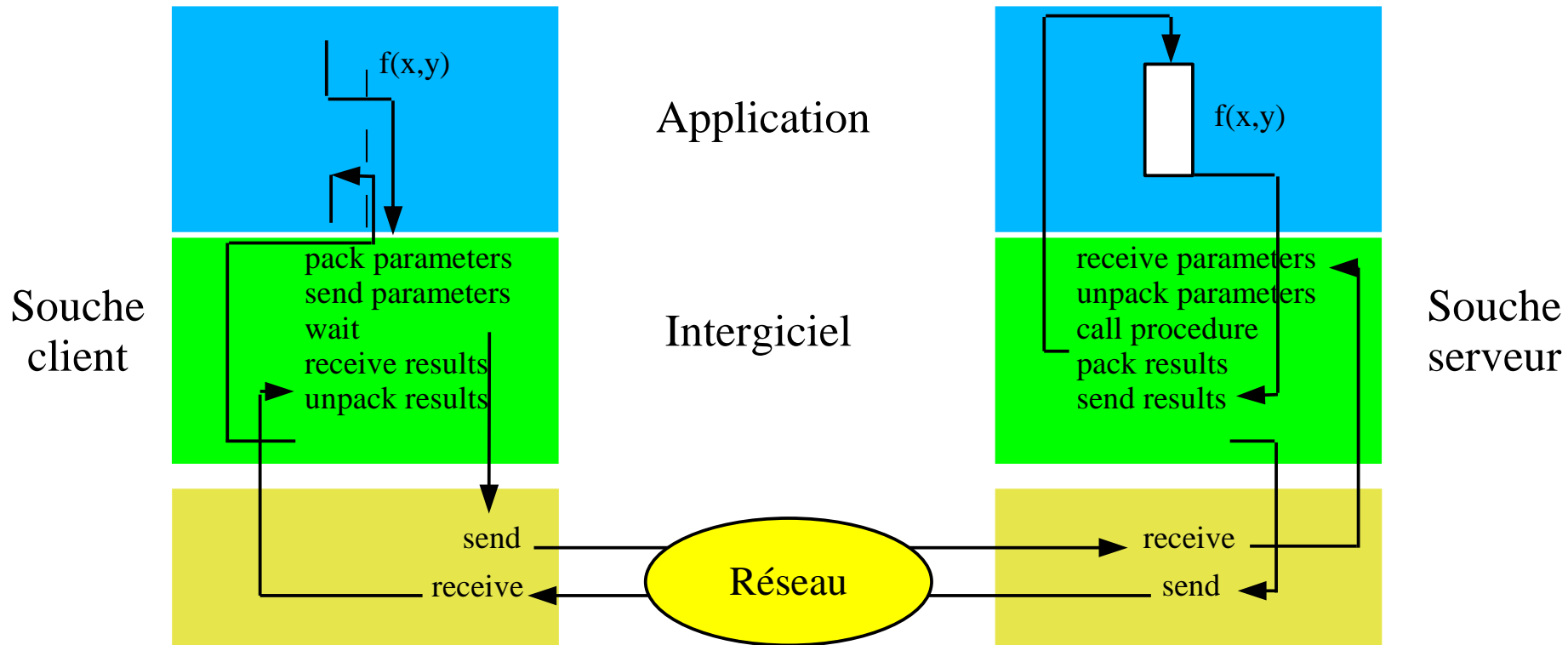
Appel distant



Le même traitement est requis (sauf défaillance réseau)



RPC (2)





Diverses formes de Middleware

- Messages
 - ◆ JMS, JORAM, LPD
- RPC
 - ◆ XML-rpc, SOAP
- Object Request Broker
 - ◆ RMI, CORBA, Eclipse Communication Framework
- Accès transparent aux données
 - ◆ ODBC, Peer-to-Peer (JXTA, Chord), space-based architecture, GSN
- Accès transparent aux composants hétérogènes
 - ◆ OSGI, Sodium, OLE, COM, Kpart, JQuery
- Accès transparent aux composants distribués
 - ◆ SETI@home (BOINC), Folding@home, MMO(RP)G (RTF)