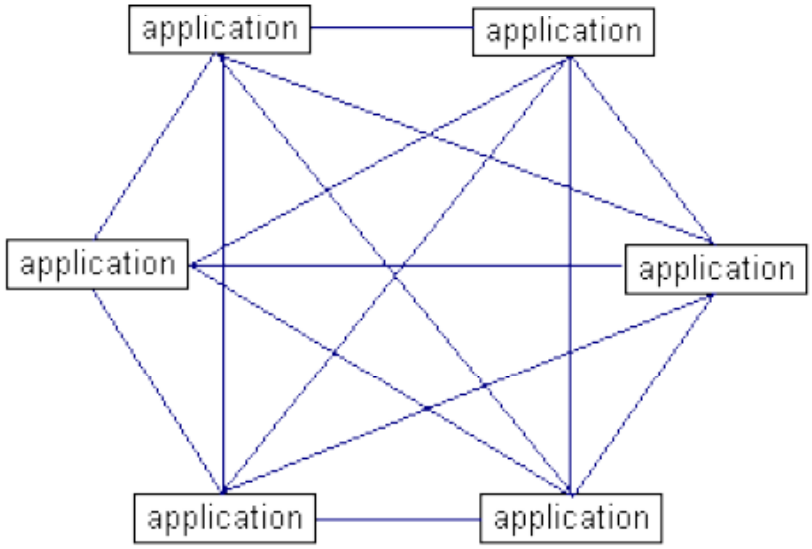
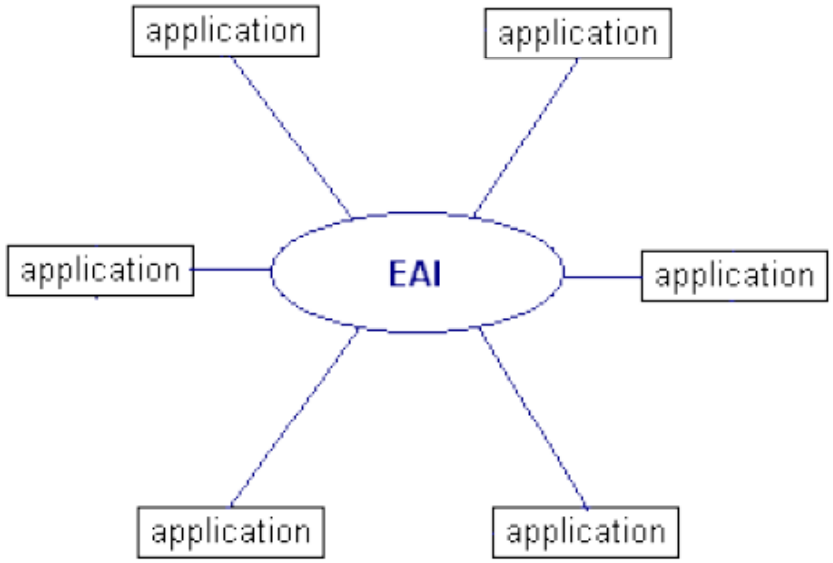
EAI et Intégration de systèmes

Définition

- L'EAI est un ensemble de méthodes, d'outils et de services qui concourent à faire communiquer des applications hétérogènes dans le cadre de l'entreprise traditionnelle,répartie ou étendue."
- L’objectif est de faire communiquer des applications développées à des moments différents, dans des technologies différentes et de manière indépendante.
- L'ERP est une première réponse à cette problématique. Elle ne couvre cependant pas tous les besoins de l'entreprise, il remplace une partie du système d'information seulement, mais pas la totalité des applications métiers

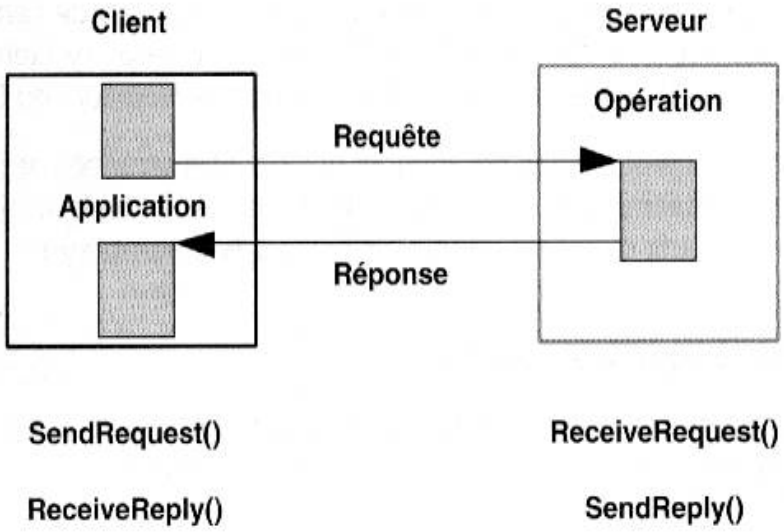
Modèle spagetthi et solution EAI



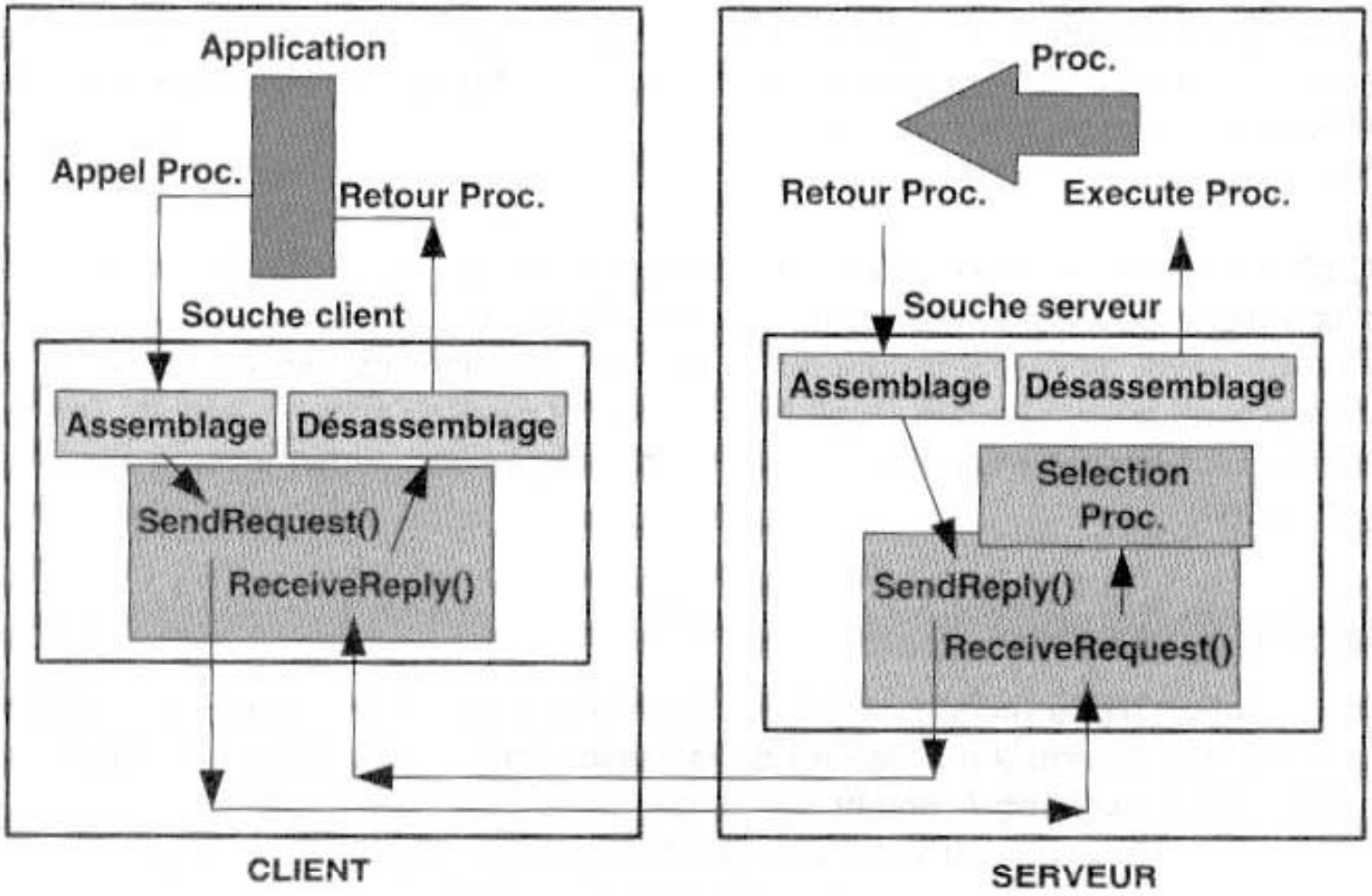


Modèles de communication entre applications

- Protocoles de type question-réponse

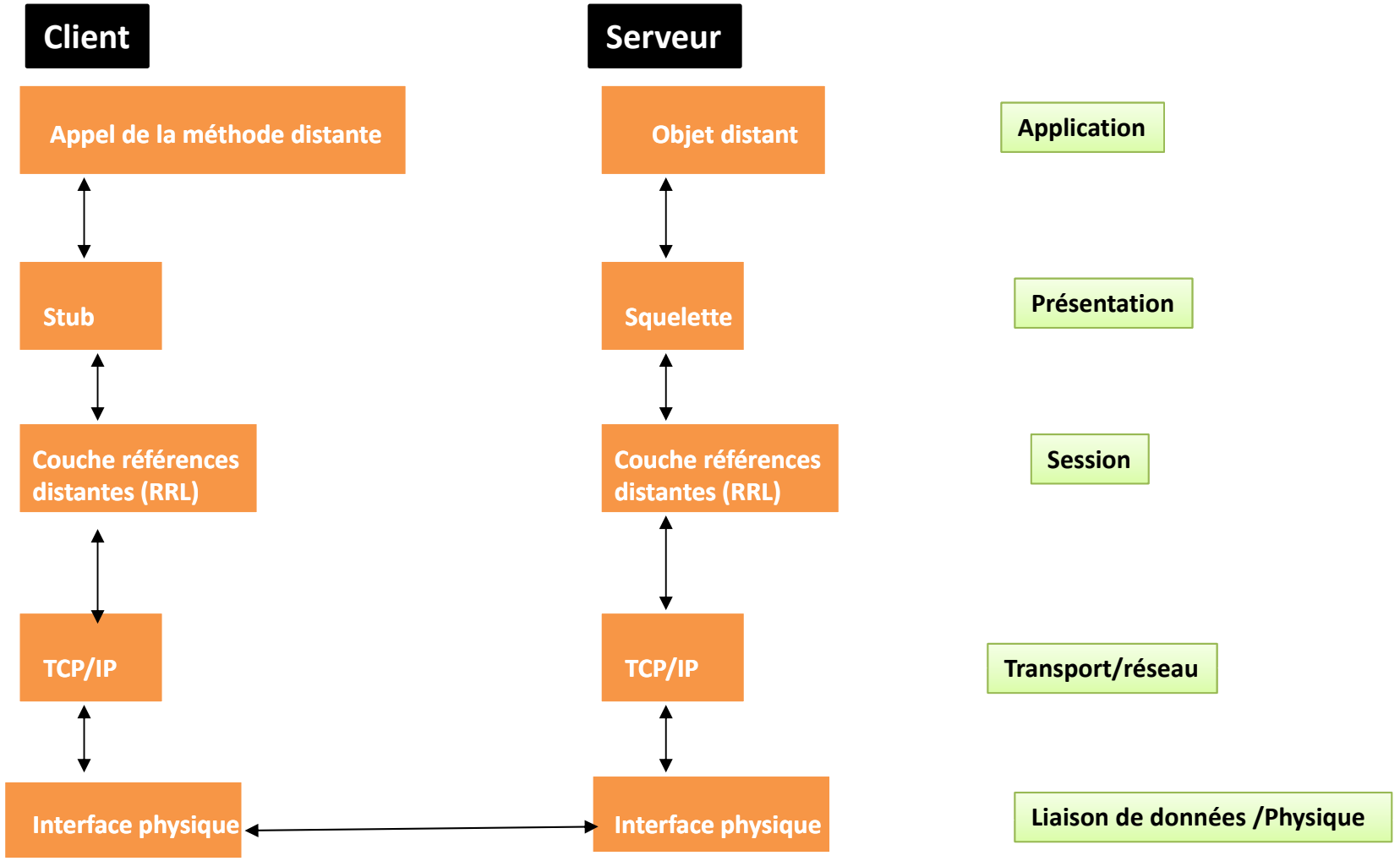


Le modèle RPC



Le modèle RMI

Architecture RMI



Etapes:

1) Côté serveur
   a) Définir l'interface distante (le stub ou la souche).
   - L'interface distante doit dériver de l'interface Remote.
   - Toute méthode de l'interface doit lever l'exception RemoteException dans sa clause throws.

```
package entreprise.rh;
import java.rmi.*;
public interface ISalarie extends Remote {
public String getNom(int id) throws RemoteException;
}
```

- b) Implémenter le service (l'interface)

```
package entreprise.rh;

import java.util.ArrayList;
import java.rmi.*;
import java.rmi.server.UnicastRemoteObject;

public class Salarie extends UnicastRemoteObject implements ISalarie {
	ArrayList<String> salaries  = new ArrayList<>();// Java 7

	public Salarie() throws RemoteException {
		salaries.add(1, "Talal");
		salaries.add(2, "Marssame");
		salaries.add(3, "Hilal");
	}
	@Override
	public String getNom(int id) {
		return salaries.get(id);
	}
}
```

- Le constructeur doit lever l'exception RemoteException qui peut être déclenchée dans le constructeur de la classe parente UnicastRemoteObject.
- Pour gérer les appels distants la classe Salarie doit dériver de la classe UnicastRemoteObject .
- Dans le cas où la classe dérive déjà d'une autre classe, il faut appeler la méthode statique ExportObject dans le constructeur.
  - UnicastRemoteObject.exportObject(this, 0); // cette méthode l'objet distant Salarie pour recevoir les appels distants, sur un numéro de port anonyme.

c) Implémenter le serveur

```
package serveur1;

import entreprise.rh.*;
import java.rmi.AlreadyBoundException;
import java.rmi.RemoteException;
import java.rmi.registry.*;

public class Serveur {
	public static void main(String[] args) {
		try {
			/* LocateRegistry.getRegistry(String URL, int port)
			* Retourne une référence vers l'annuaire.
			* url: par défaut localhost,  port : par défaut 1099 */
			Salarie salarie = new Salarie();
			Registry annuaire = LocateRegistry.getRegistry();
			annuaire.bind("Salarie", salarie);
			System.out.println("Serveur prêt");
		} catch (RemoteException ex) {
			System.err.println("Erreur Serveur:" + ex.getMessage());
		} catch (AlreadyBoundException ex) {
			System.err.println("Objet déjà lié:" + ex.getMessage());
		}
	}
}
```

d) Démarrer le Registry

- Outil : rmiregsitry (jdk\bin)

e) Démarrer le serveur

- java –cp ***classDir*** -Djava.rmi.server.codebase=file:***Dossier***/ Serveur
- La propriété java.rmi.server.codebase définit le dossier où l'interface est stockée.

2) Côté Client

```
package client1;

import java.rmi.AccessException;
import java.rmi.NotBoundException;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.*;
import entreprise.rh.ISalarie;

public class Client1 {
	public static void main(String[] args) {
		try {
			Registry annuaire = LocateRegistry.getRegistry();
			ISalarie s = (ISalarie) annuaire.lookup("Salarie");
			System.out.println(s.getNom(1));
		} catch (RemoteException ex) {
				System.err.println("Erreur Client:" + ex.getMessage());
		} catch (NotBoundException ex) {
				System.err.println("Erreur Client:" + ex.getMessage());
		}
	}
}
```