

## Cours n°10

# Appel de procédures à distance

## Sommaire

### 1. Le protocole RPC (Remote Procedure Call)

- Mécanismes et services
- Flots de description XDR
- Développement d'applications

### 2. Les packages RMI (Remote Method Invocation)

- Interface et classe serveur
- Registre des noms et lancement du serveur

## INTRODUCTION

### Bibliographie

Jean-Marie Rifflet & Jean-Baptiste Yunes, UNIX : Programmation et communication, Dundod, 2003

R. Srinivasan, RPC: Remote Procedure Call Protocol Specification Version 2, RFC 1831, 1995

W. Harold, Using Extensible Markup Language-Remote Procedure Calling (XML-RPC) in Blocks Extensible Exchange Protocol (BEEP) RFC 3529, 2003

William Grosso, Java RMI, O'Reilly, 2006

[java.sun.com/javase/technologies/core/basic/rmi/](http://java.sun.com/javase/technologies/core/basic/rmi/)

## 1

## PROTOCOLES RPC

### Introduction

#### Mécanisme permettant d'appeler des fonctions sur un système distant

Syntaxe similaire à celle des appels locaux

Utilisation quasi transparente

#### Histoire et Implémentation

1976 : Première description de services distants dans la RFC 707

1981 : Première implémentation par la société Apollo sous le système Unix

1984 : Normalisation par l'OSF de Distributed Computing Environment / RPC

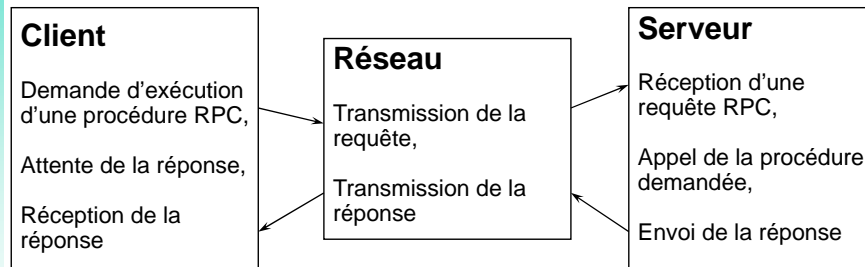
1985 : Réécriture par Microsoft de DCE/RPC (version MSRPC)

1988 : Description dans la RFC 1050 par la société SUN (version 1)

1995 : RFC 1831 (Sun RPC version 2)

1998 : XML-RPC

## Schéma de communication



## Déclaration du service

## Entête d'un message

Identificateurs de requête et de service (96 bits)

## Localisation d'un service (client)

Appel au démon portmap (port 111) sur le serveur avec en paramètre l'identification du service,  
retourne le port correspondant au service

## Enregistrement du service (serveur)

Appel au démon portmap avec en paramètre l'identification et le port correspondant au service,  
Mise à jour d'une table de correspondance

## Serveur inetd

Serveur de serveur RPC  
Création du serveur en cas de demande,  
Gain en place mémoire (+),  
Plus faible réactivité (-)

## Interface de communication

## Stub (Talon)

Interface de communication,  
client stub + serveur stub

## Client stub (adaptateur client)

Simulation d'une procédure locale,  
Encapsulation des arguments d'appel,  
Envoi au serveur stub sous forme de messages,  
Réception du message résultat,  
retourne le résultat

## Serveur stub (adaptateur serveur ou squelette)

Décodage du message envoyé par le client stub,  
exécution de la procédure demandée avec les arguments d'appel,  
récupération et encapsulation du résultat,  
renvoi au client stub sous forme d'un message

## Filtres et Flots XDR [RFC 1832]

## Normalisation des échanges de données binaires

Représentation binaire différente (format, taille, ordre)  
Alignements des champs d'une structure

## XDR (eXternal Data Representation)

Norme pour la description et le codage des données,  
Indépendant des architectures matérielles et logicielles  
Codage sur des mots de 32 bits (Complétion par des 0, format big-endian, IEEE)  
Pas de codage du type

## Filtres XDR (Procédure codant ou décodant un type de données)

Types de base du langage C/C++, tableaux et pointeurs, union (pas d'objet)

## Flots XDR (Suite d'octets représentant des données au format XDR)

Gestion à l'aide d'un pointeur sur une structure XDR,

Trois opérations sur un flot :

Encodage des données sur un flot (XDR\_ENCODE)

Décodage des données d'un flot (XDR\_DECODE)

Libération de la mémoire alloué pour un décodage (XDR\_FREE)

Flots d'entrées-sorties standard, en mémoire, d'enregistrements

## 1.3 DEVELOPPEMENT D'APPLICATIONS

### Trois méthodes de développement

#### 1) Générateur d'applications RPCGEN

Génération d'un client stub, d'un serveur stub, des filtres XDR  
Programmation de l'interface en RPCL

#### 2) Trois procédures en Stub standard (mode UDP)

int **registrpc(...)**

Initialisation du service RPC,  
Enregistrement séparé de chaque fonction

void **svc\_run()**

Attente d'appel de fonctions

int **callrpc(...)**

Appel de la RPC correspondante,  
codage des paramètres, décodage des résultats, bloquant

#### 3) Une vingtaine de procédures en Stub expert

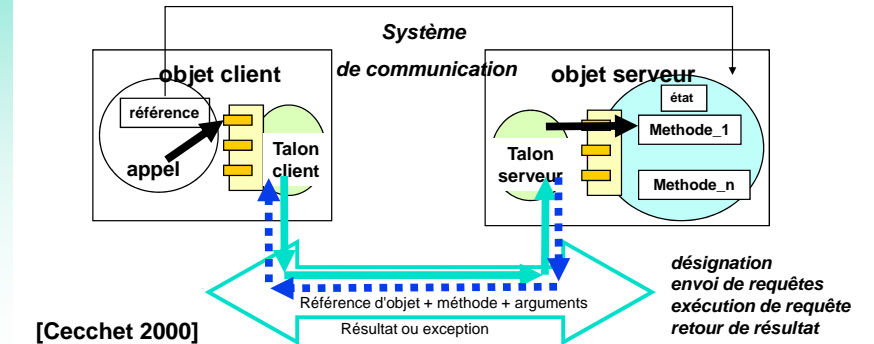
Connaissances réseau indispensables (socket...)

## 2. LES PACKAGES RMI (Remote Method Invocation)

### Principes

#### RPC objet intégré au langage Java

Interaction d'objets situés dans des espaces d'adressage différents sur des machines distinctes  
Simple à mettre en œuvre : un objet distribué se manipule comme tout autre objet Java



## 2. LES PACKAGES RMI (Remote Method Invocation)

### Contenu

#### Package Java.rmi

Interfaces [Remote](#)

Classes [MarshaledObject](#) [Naming](#) [RMISecurityManager](#)

Exceptions [AccessException](#) [AlreadyBoundException](#) [ConnectException](#)  
[ConnectIOException](#) [MarshalException](#) [NoSuchObjectException](#)  
[NotBoundException](#) [RemoteException](#) [ServerError](#)  
[ServerException](#) [StubNotFoundException](#)  
[UnknownHostException](#) [UnmarshalException](#)

#### Package Java.rmi.registry

Interfaces [Registry](#)

Classes [LocateRegistry](#)

#### Package java.rmi.server

Interfaces [RemoteRef](#) [RMIFailureHandler](#) [ServerRef](#) [Unreferenced](#)

Classes [ObjID](#) [RemoteObject](#) [RemoteObjectInvocationHandler](#)  
[RemoteServer](#) [RemoteStub](#) [RMIClassLoader](#) [RMIClassLoaderSpi](#) [UID](#)  
[UnicastRemoteObject](#)

Exceptions [ExportException](#) [ServerNotActiveException](#)  
[SocketSecurityException](#) [ServerCloneException](#)

## 2. LES PACKAGES RMI (Remote Method Invocation)

### Principe de mise en oeuvre

#### Définition du talon (stub) de la classe serveur

Identique cotés client et serveur,  
Méthodes publiques de la classe serveur  
Interface dérivée de l'interface [Remote](#)

#### Définition de la classe serveur

Coté serveur,  
Classe dérivée de la classe [UnicastRemoteObject](#) implémentant le talon

#### Lancement du serveur

Création d'un registre des objets serveur ([LocateRegistry](#))  
Création d'une instance de classe de la classe serveur  
Inscription de cet objet sur le registre des objets serveurs ([Registry](#))

#### Utilisation

Localisation de l'objet distant ([Registry](#)) et création du talon  
Appel des méthodes à partir du talon

## 2.1 INTERFACE ET CLASSE SERVEUR

### Définition du talon

#### Interface Remote

Conteneur des méthodes publiques de la classe répartie  
Traitement obligatoire de l'exception `RemoteException` ou utilisation de `throws`

#### Appel de Méthodes mais d'un constructeur

Pas de création de l'objet à distance  
Remplacement du constructeur par  
    Constructeur local (appelé sur le serveur)  
    Méthode d'initialisation (appelée du client)

```
public interface Hello extends java.rmi.Remote {  
    String sayHello() throws java.rmi.RemoteException;  
}
```

## 2.1 INTERFACE ET CLASSE SERVEUR

### Définition du talon – Exemple

```
public interface LireToutUnTexte extends java.rmi.Remote {  
  
    /** lecture d'un fichier texte dans la chaîne sTexte  
     * @param ft nom du fichier */  
    public void Initialisation(String ft) throws java.rmi.RemoteException;  
  
    /** Recherche du nombre d'occurrences de la chaîne s dans sTexte  
     * @param s chaîne cherchée  
     * @return nombre d'occurrences */  
    public int Chercher(String s) throws java.rmi.RemoteException;  
  
    /** Affichage du contexte des occurrences de s dans sTexte  
     * @param s chaîne cherchée  
     * @param taille taille du contexte */  
    public void ChercherVoir(String s, int taille) throws  
        java.rmi.RemoteException;  
}
```

## 2.1 INTERFACE ET CLASSE SERVEUR

### Définition de la classe serveur

#### Classe UnicastRemoteObject

Prototype de la classe serveur  
Implémentation obligatoire des méthodes de l'interface  
Création du talon serveur à l'instanciation d'un objet

```
public class HelloServeur extends UnicastRemoteObject  
implements Hello {  
  
    private String msg;  
  
    // Constructeur  
    public HelloServeur() throws java.rmi.RemoteException {  
        this.msg = InetAddress.getLocalHost().getHostName();  
    }  
    // Implémentation de la méthode distante.  
    public String sayHello() throws java.rmi.RemoteException {  
        return "Bonjour de la part de " + msg;  
    }  
}
```

## 2.1 INTERFACE ET CLASSE SERVEUR

### Définition de la classe serveur – Initialisation

```
public class LireToutUnTexteServeur extends UnicastRemoteObject  
implements LireToutUnTexte {  
    private String Stexte = "";  
    private BreakIterator bi;  
  
    public LireToutUnTexteServeur() throws java.rmi.RemoteException {}  
  
    /** Initialisation lecture d'un fichier texte dans la chaîne sTexte  
     * @param ft nom du fichier */  
    public void Initialisation(String ft) {  
        String ligne;  
        try {  
            BufferedReader br = new BufferedReader(new FileReader(fr));  
            while ((ligne=br.readLine()) != null) { Stexte = Stexte.concat(ligne);  
            }  
            br.close();  
            // segmentation en mots  
            bi = BreakIterator.getWordInstance(); bi.setText(Stexte);  
        }  
        catch (IOException e) { e.printStackTrace(); }  
    }  
}
```

## 2.2 REGISTRE ET LANCEMENT DU SERVEUR

### Registre des objets serveurs

#### Classe LocateRegistry

Méthodes statiques pour gérer un registre des objets serveurs

##### Création

```
Registry LocateRegistry.createRegistry(int p)
```

Création d'un registre local acceptant des requêtes sur le port p (1099 par défaut)

##### Lecture locale

```
Registry LocateRegistry.getRegistry()
```

Référence sur le registre local sur le port par défaut (1099)

```
Registry LocateRegistry.getRegistry(int p)
```

Référence sur le registre local sur le port p

##### Lecture distante

```
Registry LocateRegistry.getRegistry(String h)
```

Référence sur le registre de la machine de nom h sur le port par défaut (1099)

```
Registry LocateRegistry.getRegistry(String h, int p)
```

Référence sur le registre de la machine de nom h sur le port p

## 2.2 REGISTRE ET LANCEMENT DU SERVEUR

### Inscription des objets sur le registre

#### interface Registry

```
void bind(String URL, Remote obj)
```

Modification d'un lien entre une URL et un objet distant

```
void rebind(String URL, Remote obj)
```

Création d'un lien entre une URL et un nouvel objet distant

```
void unbind(String URL)
```

Destruction d'un lien entre une URL et un objet distant

```
String[] list(String URL)
```

Renvoie la liste des noms enregistrés sur une URL

```
Remote lookup(String URL)
```

Localisation de l'objet distant à partir de son URL, renvoie d'un talon

#### Classe Naming

Méthodes statiques de même nom utilisées sur le client

Utilisation du même port pour la création du registre et l'URL

## 2.2 REGISTRE ET LANCEMENT DU SERVEUR

### Lancement du serveur

#### LancementServeur.java

```
int port = 12536;
String host = InetAddress.getLocalHost().getHostName();

// Création du serveur de nom - rmiregistry
Registry r = LocateRegistry.createRegistry(port);

// Création d'une instance des objets serveur
HelloServeur o1 = new HelloServeur();
LireToutUnTexteServeur o2 = new LireToutUnTexteServeur();

// Inscription des serveurs Hello et Lire
r.rebind("//"+host+": "+port+"/Hello", o1);
r.rebind("//"+host+": "+port+"/Lire", o2);

// Liste des serveurs inscrits
String s[] = r.list();
for (int i = 0; i < s.length; i++) System.out.println(s[i]);

//irpl:12536/Lire
//irpl:12536/Hello
```

## 2.2 REGISTRE ET LANCEMENT DU SERVEUR

### Appel de méthodes distantes

#### testAdolphe.java

```
int port = 12536; String host = "irpl";
String URL = "//"+host+": "+port+"/";

Registry r = LocateRegistry.getRegistry(host, port);
// référence sur le talon de l'objet distant
Hello h = (Hello) r.lookup(URL+"Hello");
// appel à une méthode distante
System.out.println(h.sayHello());

LireToutUnTexte lt = (LireToutUnTexte) r.lookup(URL+"Lire");
lt.Initialisation("util/adolphe.txt");
String s = Keyboard.getString("mot à chercher? ");
System.out.println("Il y a " + lt.Chercher(s) + " occurrences de ce mot dans le texte");

Bonjour de la part de irpl
mot à chercher?
Adolphe
Il y a 39 occurrences de ce mot dans le texte
```