Appel de procédure distance – RPC Remote Procedure Call

RSX 102
Anne WEI
CNAM Paris

1

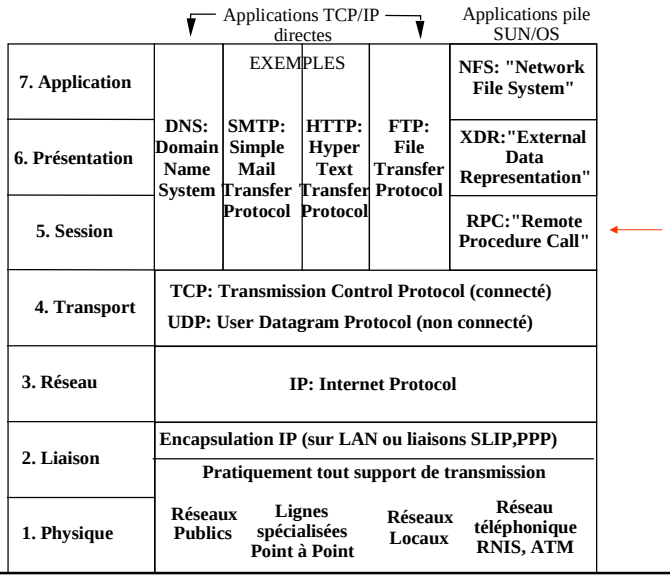
Plan

2

Bibliographie

- Gérard Florin , support de cours « Technologie pour les applications client-serveur, RSX102 »
- R. Thurlow «Remote Procedure Call Protocol Specification version 2», RFC 5531, mai 2009
- M. Eisler « eXternal data Representation (XDR) language » RFC 4506, mai 2006
- G. Coulouris, J. Dollimore et T. Kindberg, «Distributed Systems, Conceptions and Design » p. 183, 2002

Introduction (1)



Introduction (2)

- RPC (*Remote Procedure Call*), l'appel de procédure distante est une technique client-serveur. C'est-à-dire, c'est un outil du mode client-serveur via requêtes/réponses
- L' exécution de requête/réponse peut être locale ou distante
- le but de RPC a pour simplifier la programmation d'applications distribuées/réparties
- Ne pas se préoccuper de la localisation de la procédure
- Ne pas se préoccuper du traitement de défaillances
- RPC permet de développer des nombreuses applications client-serveur
- Tous types de services à accéder à distance (serveur de fichiers, serveur de ressources, serveur d'authentification, serveur d'accès à des données, serveur de traitements…)

5

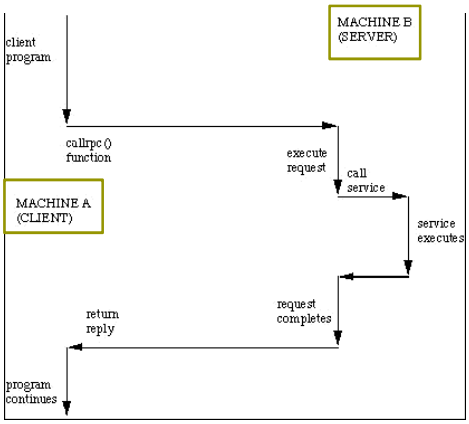
Introduction (2)

- RPC (*Remote Procedure Call*), l'appel de procédure distante est une technique client-serveur. C'est-à-dire, c'est un outil du mode client-serveur via requêtes/réponses
- L' exécution de requête/réponse pourra être locale ou distante
- le but de RPC a pour simplifier la programmation d'applications distribuées/réparties
- Ne pas se préoccuper de la localisation de la procédure
- Ne pas se préoccuper du traitement de défaillances
- RPC permet de développer des nombreuses applications client-serveur
- Tous types de services à accéder à distance (serveur de fichiers, serveur de ressources, serveur d'authentification, serveur d'accès à des données, serveur de traitements…)

6

Introduction (3)

- Le mécanisme de RPC



Une procédure RPC est identifiable par 3 paramètres :
- *numéro de la programme*
- *numéro de la version*
- *numéro de la procédure*

7

Introduction (4)

- les implantations de RPC traditionnelles
  - SUN ONC/RPC (ONC, Open Network Computing)
  - Systèmes de gestion de bases de données : procédures stockées
- les implantations de RPC dans les systèmes d'objets répartis
  - SUN Java RMI (Remote Methode Invocation)
  - OMG CORBA (Object Management Group)
- les implantation de RPC dans les systèmes de composants
  - Web Service et protocole SOAP *(Simple Object Access Protocol)*

8

Plan

1 Introduction de RPC
2 Fonctionnement de RPC
3 Gestions du contrôle et des données
4 Représentation des données, désignation et liaison
5 Tolérance aux pannes
6 Conclusion

9

Notation de souches

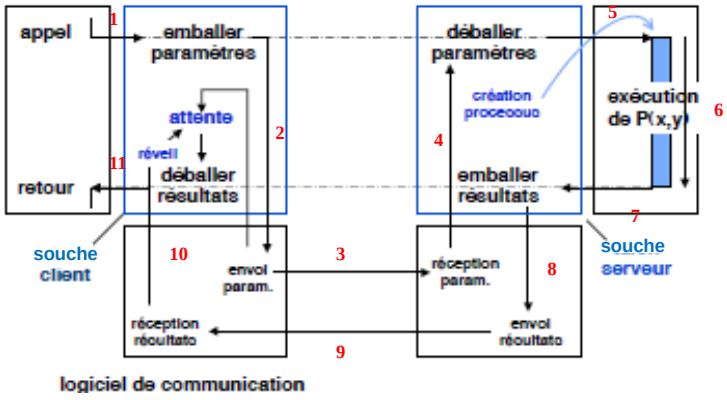
- RPC est un mode de réalisation par interception (*wrapping*)
- une procédure intercepteur (*wrapper*) intercepte l'appel d'un client vers un serveur et modifie le traitement serveur à sa guise
- l'intercepteur du côté client et l'intercepteur du côté serveur
- la décomposition en traitement avant et après le traitement serveur

- Souches : transformation d'appel local en appel distant
- Objectif de génération automatique des souches connaissant le profil d'appel de la procédure distante

- Nombreuse terminologie : Souches (Stubs), Talon, Squelettes (Skeletons)

10

Souches : diagramme global d'interaction



Source : Krakowiak

Les étapes d'un appel de procédure distante par message (1)

- **Étape 1 :** le client envoie un appel procédural vers la procédure souche client.
  Ensuite, la souche client emballe les paramètres et les aligne dans le message d'appel (*parameter marshalling*). Enfin, la souche détermine l'adresse du serveur et l'identifiant unique de l'appel. Elle met la procédure client en attente
- **Etape 2:** la souche client demande à une entité de transport local la transmission du message d'appel
- **Etape 3**: le message d'appel est transmis sur un réseau au site serveur
- **Etape 4**: le message d'appel est délivré à la souche serveur. Ensuite, la souche serveur crée (ou sélectionne) une nouvelle procédure. Enfin, elle déballe les paramètres.
- **Etape 5**: la souche serveur envoie l'appel effectif à la procédure serveur

Les étapes d'un appel de procédure distante par message (1)

- **Étape 1:** le client envoie un appel procédural vers la procédure souche client.

Ensuite, la souche client emballe les paramètres et les aligne dans le message d'appel (*parameter marshalling*). Enfin, la souche détermine l'adresse du serveur et l'identifiant unique de l'appel. Elle met la procédure client en attente

- **Etape 2 :** la souche client demande à une entité de transport local la transmission du message d'appel
- **Etape 3** : le message d'appel est transmis sur un réseau au site serveur
- **Etape 4**: le message d'appel est délivré à la souche serveur. Ensuite, la souche serveur crée (ou sélectionne) une nouvelle procédure. Enfin, elle déballe les paramètres.
- **Etape 5**: la souche serveur envoie l'appel effectif à la procédure serveur

13

Les étapes d'un appel de procédure distante par message (1)

- **Étape 1:** le client envoie un appel procédural vers la procédure souche client.

Ensuite, la souche client emballe les paramètres et les aligne dans le message d'appel (*parameter marshalling*). Enfin, la souche détermine l'adresse du serveur et l'identifiant unique de l'appel. Elle met la procédure client en attente

- **Etape 2:** la souche client demande à une entité de transport local la transmission du message d'appel
- **Etape 3**: le message d'appel est transmis sur un réseau au site serveur
- **Etape 4** : le message d'appel est délivré à la souche serveur. Ensuite, la souche serveur crée (ou sélectionne) une nouvelle procédure. Enfin, elle déballe les paramètres.
- **Etape 5** : la souche serveur envoie l'appel effectif à la procédure serveur

14

Les étapes d'un appel de procédure distante par message (2)

- **Etape 6 :** la procédure serveur exécute les paramètres
- **Etape 7** : la procédure serveur transmet à la souche serveur dans son retour de procédure les paramètres résultats
- **Étape 8 :** la souche serveur collecte les paramètres retour. Ensuite, elle les emballe dans un message. Elle demande à l'entité de transport serveur la transmission du message de réponse
- **Etape 9**: le message de réponse est transmis sur un réseau au site client
- **Etape 10**: le message de réponse est transmis à la souche client. La souche déballe les paramètres résultats
- **Etape 11**: la souche client envoie les résultats au client en mode local

15

Les étapes d'un appel de procédure distante par message (2)

- **Etape 6:** la procédure serveur exécute les paramètres
- **Etape 7**: la procédure serveur transmet à la souche serveur dans son retour de procédure les paramètres résultats
- **Étape 8:** la souche serveur collecte les paramètres retour. Ensuite, elle les emballe dans un message. Elle demande à l'entité de transport serveur la transmission du message de réponse
- **Etape 9** : le message de réponse est transmis sur un réseau au site client
- **Etape 10** : le message de réponse est transmis à la souche client. La souche déballe les paramètres résultats
- **Etape 11** : la souche client envoie les résultats au client en mode local

16

Problèmes à traiter

- Comment le client connait l'adresse du serveur?
- Comment emballer et déballer les paramètres d'appel et de réponse vis-à-vis des entités hétérogènes? (structure complexe, représentation de données, message avec grande taille…)
- Comment gérer simultanément les nombreux appels ?
- Comment traiter des pannes? (gestion de procédures, mode de détection…)
- Comment générer automatiquement une souche?
- …

17

Avantages et améliorations de RPC

- Avantages
- RPC est développé en invocation distante par message
- Une technique applicable en univers hétérogènes moyennant des conversion
- Très simple à réaliser

- Améliorations possibles
- Par appel léger (message avec petite taille ; 32 ou 64 bits)
- Par Sun XDR (*eXternal Data Representation)* déjà utilisé dans NFS

18

Plan

1 Introduction de RPC
2 Fonctionnement de RPC
3 Gestions du contrôle et des données
4 Représentation des données, désignation et liaison
5 Tolérance aux pannes
6 Conclusion

19

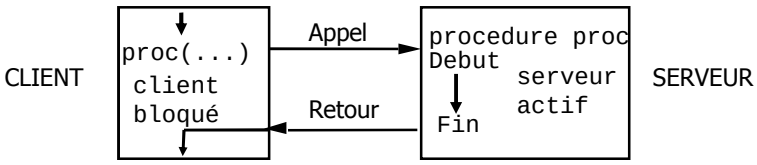
Gestion du contrôle

- Gestion du contrôle à côté client – parallélisme
- Gestion du contrôle à coté serveur – parallélisme
- Structures de contrôle réparti par composition d'appels distants

20

RPC en mode synchrone chez le client

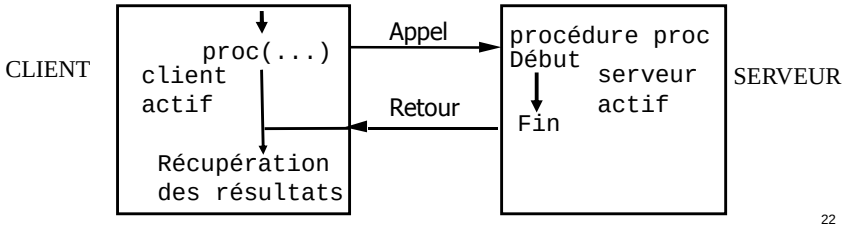
- En mode synchrone, l'exécution du client est suspendu tant que la réponse du serveur n'est pas revenue ou qu'une condition d'exception n'a pas entraîné un traitement spécifique.
- Avantage : le flot de contrôle est le même que dans l'appel en mode centralisé
- Inconvénient : le client reste inactif



21

RPC en mode asynchrone chez le client

- En mode asynchrone, le client poursuit son exécution immédiatement après l'émission du message de l'appel
- la procédure distante s'exécute en parallèle avec la poursuite du client
- le client récupère les résultats de l'appel quand il en a besoin (primitive spéciale)



22

Cas particulier du mode asynchrone : invocation asynchrone à sens unique

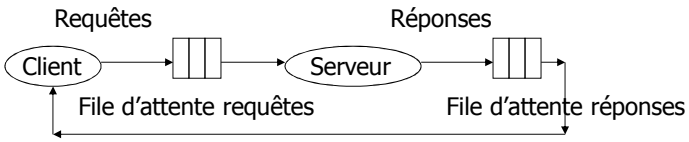
- Terminologie : «oneway», invocation asynchrone sans réponse
- Invocation asynchrone utilisée pour déclencher une procédure qui ne retourne pas de résultats. Pour obtenir un dialogue il faut prévoir d'autres procédures en sens inverse.
- Applications: Service-orienté architecture, par exemple

- Avantage : Utilisation d'un mode appel de procédure distante est de mode message
- Inconvénients : uniquement possible en l'absence de retour de résultat, pas d'informations sur la terminaison du travail demandé
- exemple : CORBA oneway

23

Exécution séquentielle des appels chez le serveur

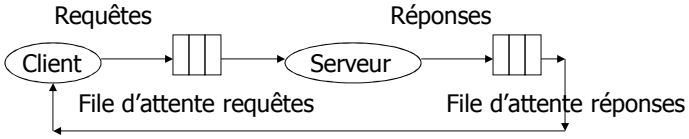
- Les requêtes d'exécution sont traitées l'une après l'autre par le serveur
- Si la couche transport assure la livraison en séquence et que l'on gère une file d'attente FIFO, on a un traitement ordonné des suites d'appels
- exemple RPC SUN: traitement séquentiel des requêtes mais utilisation de UDP (mode non-connecté) . Les requêtes non ordonnées. Chez le client, c'est le mode synchrone.



24

Exécution séquentielle des appels chez le serveur

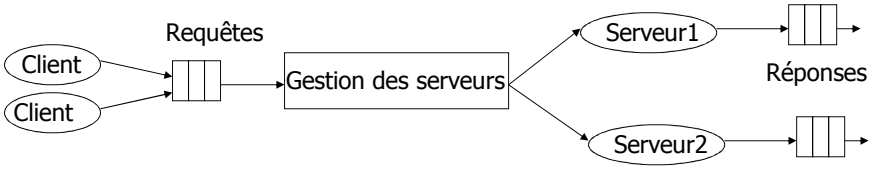
- Les requêtes d'exécution sont traitées l'une après l'autre par le serveur
- Si la couche transport assure la livraison en séquence et que l'on gère une file d'attente FIFO, on a un traitement ordonné des suites d'appels
- exemple RPC SUN: traitement séquentiel des requêtes mais utilisation de UDP (mode non-connecté). Les requêtes non ordonnées. Chez le client, c'est le mode synchrone.



25

Exécution parallèle des appels chez le serveur

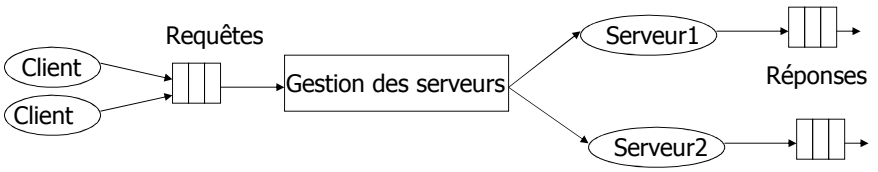
- Le serveur crée un processus ou une activité (un processus léger ou « thread ») par appel
- les appels sont exécutés concurremment
- si les procédures manipulent des données globales rémanentes sur le site serveur, le contrôle de concurrence doit être géré.
- Exemple: CORBA Notion d'adaptateur d'objets



26

Exécution parallèle des appels chez le serveur

- Le serveur crée un processus ou une activité (un processus léger ou « thread ») par appel
- les appels sont exécutés concurremment
- si les procédures manipulent des données globales rémanentes sur le site serveur, le contrôle de concurrence doit être géré.
- Exemple : CORBA Notion d'adaptateur d'objets



27

Gestion des données applicatives sans données partagées persistantes

- Si les données sont locales à la procédure, pas de problème
- Si les données sont applicatives partagées : on a le problème de persistance causé par les variables d'instance, le partage de fichiers et le partage de bases de donnée

- la **situation idéale** du cas où l'appel de procédure s'exécute en fonction uniquement des paramètres d'entrée en produisant uniquement des paramètres résultats
- exemple, un calcul d'une fonction scientifique simple
- il n'y a pas de modification de données rémanentes sur le site serveur
- pas de problème pour la tolérance aux pannes et pour le contrôle de concurrence

28

Gestion des données applicatives sans données partagées persistantes

- Si les données sont locales à la procédure, pas de problème
- Si les données sont applicatives partagées: on a le problème de persistance causé par les variables d'instance, le partage de fichiers et le partage de bases de donnée

- la **situation idéale** du cas où l'appel de procédure s'exécute en fonction uniquement des paramètres d'entrée en produisant uniquement des paramètres résultats
- exemple, un calcul d'une fonction scientifique simple
- il n'y a pas de modification de données rémanentes sur le site serveur
- pas de problème pour la tolérance aux pannes et pour le contrôle de concurrence

29

Gestion des données applicatives avec données partagées persistantes

- les exécutions successives manipulent des données persistantes sur le site serveur
- le contexte sur le site distant est modifié par une exécution (un serveur de fichier réparti, un serveur de bases de données, une opération d'écriture, la structure de donnée modifiée par les méthodes d'un objet)

- Problème de contrôle de concurrence

- Problème des pannes en cours d'exécution

- Solution: le couplage d'une gestion transactionnelle avec une approche RPC ou Système d'objets répartie

- exemple: EJB Session (Entreprise JavaBeans)

30

Gestion des données applicatives avec données partagées persistantes

- les exécutions successives manipulent des données persistantes sur le site serveur
- le contexte sur le site distant est modifié par une exécution (un serveur de fichier réparti, un serveur de bases de données, une opération d'écriture, la structure de donnée modifiée par les méthodes d'un objet)
- Problème de contrôle de concurrence
- Problème des pannes en cours d'exécution
- Solution : le couplage d'une gestion transactionnelle avec une approche RPC ou Système d'objets répartie (solution de « conteneur/composant »)
- Exemple : EJB Session (Entreprise JavaBeans)

31

Gestion des données protocolaires

- La terminologie avec ou sans état porte sur l'existence ou non d'un descriptif pour chaque relation client-serveur du côté serveur
- Notation d'état : un ensemble de données rémanentes au niveau du protocole pour chaque relation client-serveur
- l'état (descriptif) permet de traiter les requêtes dans l'ordre d'émission
- il permet également de traiter une requête en fonction des caractéristiques de la relation client-serveur (QoS, par exemple)

32

Gestion des données protocolaires sans état

- les appels successifs d'une même procédure s'exécutent **sans liens** entre eux
- Il peut y avoir modification de données globales rémanentes sur le site serveur mais chaque opération du point de vue du protocole s'effectue sans référence au passé.
- NFS (*Network File System*) de SUN système de fichier réparti basé sur RPC sans état. La lecture/écriture du nième article d'un fichier dont toutes les caractéristiques utiles (nom, droit d'accès) sont passées au moment de l'appel.
- HTTP (HyperText Transfer Protocol) s'exécute de méthodes distante sans état

33

Gestion des données protocolaires sans état

- les appels successifs d'une même procédure s'exécutent **sans liens** entre eux
- Il peut y avoir modification de données globales rémanentes sur le site serveur mais chaque opération du point de vue du protocole s'effectue sans référence au **passé**.
- NFS (*Network File System*) de SUN basé sur RPC sans état. La lecture/écriture du nième article d'un fichier dont toutes les caractéristiques utiles (nom, droit d'accès) sont passées au moment de l'appel.
- HTTP (*HyperText Transfer Protocol*) s'exécute de méthodes distante sans état

34

Gestion des données protocolaires avec état

- les appels successifs s'exécutent en fonction d'un état de la relation client-serveur laissé par les appels antérieurs
- Il est nécessaire de maintenir la gestion de l'ordre de traitement des requêtes
- exemple : accès au serveur de fichiers réparties doit être séquentiel

35

Plan

1 Introduction de RPC
2 Fonctionnement de RPC
3 Gestions du contrôle et des données
4 Représentation des données, désignation et liaison
5 Tolérance aux pannes
6 Conclusion

36

Représentation des données

- Rappelons que la représentation des données est un problème classique dans la transmission
- Il est nécessaire de convertir la représentation des données si le client et le serveur
  - n'utilisent pas le même codage (Big Endian, Little Endian)
  - n'utilisent pas le même format (type de caractère, entier, flottant…)
  - solution placée classiquement dans la couche 6 – représentation du modèle OSI => ASN1 (*Abstract Syntax Notation One*)/BER (*Basic Encoding Rules)*
- Dans un réseau, le passage de paramètre par valeur est également un problème à résoudre. Une solution est d'utiliser le langague XDR (*eXternal data Representation (XDR) language*) si le réseau est du type Internet

37

Passage par valeur dans l'appel de procédure distante : IDL

- Insuffisance des normes de syntaxe abstraite et de transfert en mode message asynchrone : ASN1/BER
- Définition de nouveaux langages de syntaxe abstraite adaptés aux appels de procédure distante: IDL *(Interface Definition Language*)
- En appel de procédure distante : génération automatique du code des souches (client et serveur) à partir de la syntaxe abstraite
- Les souches fabriquent la syntaxe de transfert en réalisant l'alignement des paramètres dans les messages à transmettre (« marshalling »)
- pour chaque IDL, en général, il faut une redéfinition d'une syntaxe de transfert

38

IDL
(Interface Definition Language) (1)

- Le but de l'interface IDL a pour
- être indépendant des langages évolués utilisant le RPC. Le client en C++ et le serveur en java, par exemple
- être indépendant des systèmes différents (Windows et Linux, par exemple)

- IDL permet aux utilisateurs de définir un identifiant unique pour chaque interface

- Elle Supporte les caractéristiques particulières des langages objets (définir des relations d'héritage entre définitions et d'interfaces)

- Elle peut également structurer les interfaces en modules

39

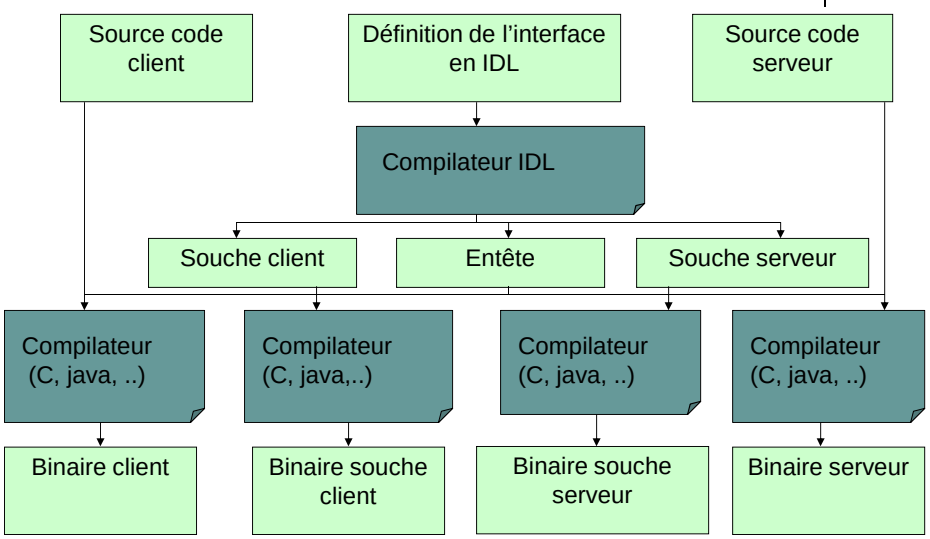
IDL
(Interface Definition Language) (2)

- La structure d'un fichier IDL consiste en 3 éléments principaux
- Module : un espace de définitions
- Interface : un regroupement de services
- Méthode : un service

- un exemple en IDL CORBA

```
module StockObjects          {
 struct Quote { string symbol; long at_time; double price; long volume;};
 exception Unknown{};
 interface Stock { Quote get_quote(); raises(Unknown); // lit la cotation.
  void set_quote(in Quote stock_quote); // écrit la cotation
                    };
 interface StockFactory { Stock create_stock( in string symbol,  in string description ); };
                                    };
```

IDL
Génération des souches



Quelques exemples d'IDL

- SUN ONC/RPC (*Open Network Computing Remote Procedure Call*)
  - XDR (eXternal Data Representation)
- OMG CORBA (*Object Management Group Common Object Request Broker Architecture*)
  - IDL Corba – format CDR (*Common Data Representation*)
- SUN Java RMI (*Java Remote Method Invocation*)
  - Langage Java – Protocole JRMP (*Java Remote Method Protocol*)
- Microsoft – DCOM (*Distributed Component Objet Model*)
  - MIDL (*Microsoft IDL*) – DCOM Protocol ORPC (*Objet RPC format NDR*)

42

Plan

1 Introduction de RPC
2 Fonctionnement de RPC
3 Gestions du contrôle et des données
4 Représentation des données, **désignation** et **liaison**
5 Tolérance aux pannes
6 Conclusion

43

Désignation

- Objets à désigner
- Désignation du protocole permettant l'accès distant (TCP, par exemple)
- Désignation de l'hôte où se trouve le serveur (@IP)
- Désignation du point d'accès de serveur transport (n° de port)
- Désignation de la procédure
- Désignation globale indépendante de la localisation dans le but de re-configurer des services (pannes…)

- Méthodes de désignation
- Statique: localisation du serveur est connue à la compilation
- Dynamique: localisation du serveur n'est pas connue à la compilation dans le but de séparer le nom du serveur de la sélection de la procédure qui va l'exécuter

44

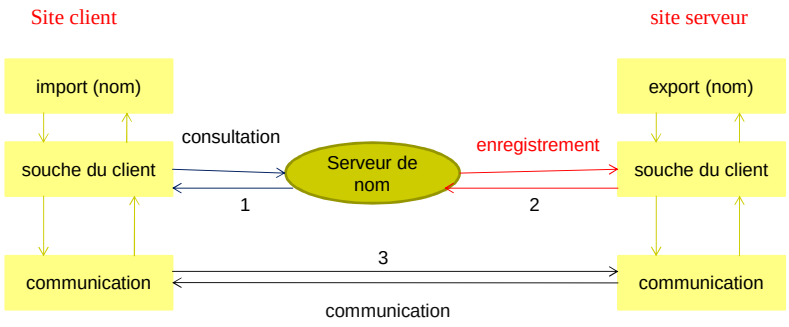
Désignation

- Objets à désigner
  - Désignation du protocole permettant l'accès distant (TCP, par exemple)
  - Désignation de l'hôte où se trouve le serveur (@IP)
  - Désignation du point d'accès de serveur transport (n° de port)
  - Désignation de la procédure
  - Désignation globale indépendante de la localisation dans le but de re-configurer des services (pannes…)
- Méthodes de désignation
  - Statique : localisation du serveur est connue à la compilation
  - Dynamique : localisation du serveur n'est pas connue à la compilation dans le but de séparer le nom du serveur de la sélection de la procédure qui va l'exécuter

45

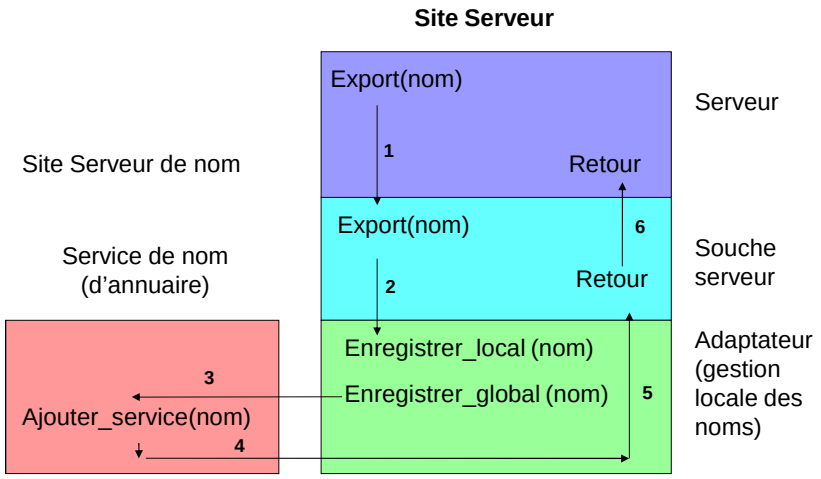
Liaison

- Déterminer la localisation du serveur (@ du serveur)
  - Liaison statique à la compilation (pas d'appel à un serveur de nom)
  - Liaison en exécution au premier appel (appel du serveur de nom lors du premier appel)
  - Liaison en exécution à chaque appel (appel du serveur de nom à chaque appel)



46

Liaison - enregistrement (1)



47

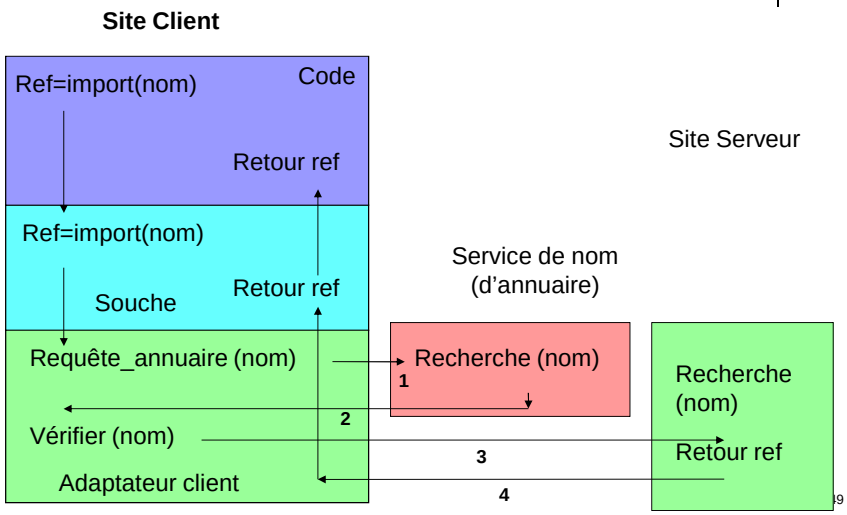
Liaison - enregistrement (2)

- Enregistrement dans une base de données le nom de serveur
- transmission du nom de serveur
- transmission de l'adresse de serveur
- la durée
- tous attributs complémentaires si plusieurs serveurs rendent le même service
- le serveur de noms confirme l'enregistrement

Exemple, DNS (*Domain Name System*) d'Internet

48

Liaison - consultation (3)



Plan

Tolérance aux pannes (1)

- Deux méthodes :
- Les composants tolérants aux pannes. Si chaque composant, à son tour, peut continuer à fonctionner lorsque l'un de ses sous-composants est en panne, alors le système entier pourra continuer à fonctionner.
- La redondance signifie avoir une sauvegarde des composants qui peut prendre la relève dès qu'un composant tombe en panne
- La redondance passive: Un seul des composants réalise effectivement les traitements et est affecté aux sorties (le primaire). En cas de panne du primaire l'un des composants (calculateur par exemple) inactifs (secondaire) est

sélectionné et activé pour prendre en charge le service.

- La redondance massive:
- La redondance sélective:

51

Tolérance aux pannes (1)

- Deux méthodes:
- Les composants tolérants aux pannes. Si chaque composant, à son tour, peut continuer à fonctionner lorsque l'un de ses sous-composants est en panne, alors le système entier pourra continuer à fonctionner.
- La redondance signifie avoir une sauvegarde des composants qui peut prendre la relève dès qu'un composant tombe en panne
- *La redondance passive*: Un seul des composants réalise effectivement les traitements et est affecté aux sorties (le primaire). En cas de panne du primaire, l'un des composants inactifs (secondaire) est sélectionné et activé pour prendre en charge le service.
- *La redondance massive*:
- *La redondance sélective*:

52

Tolérance aux pannes (2)

- La redondance signifie avoir une sauvegarde des composants qui peut prendre la relève dès qu'un composant tombe en panne
- *La redondance massive* : Tout le monde doit recevoir les entrées en diffusion dans le même ordre pour traiter les données dans le même ordre et fournir les résultats dans le même ordre aux « voteurs « . La synchronisation entre les productions de résultats doit permettre la réalisation du vote majoritaire.
- *La redondance sélective active* : Dans la redondance active tous les composants réalisent les traitements. Le gestionnaire de la redondance traite les sorties pour tolérer différentes classes de panne des serveurs.

Pannes causées par RPC

- Appelant et appelé peuvent tomber en panne indépendamment
- les messages d'échange (appel ou retour) peuvent être perdus
- le réseau est déjà saturé
- Le protocole n'est pas fiable (UDP, par exemple)
- le temps de réponse peut-être très long en raison de surcharges diverses (le réseau par exemple)

54

Panne du serveur

- attente indéfinie par le client d'une réponse qui ne viendra peut être jamais

Solutions :

- le client décide de la stratégie de reprise
  - abandon
  - Reprise sur le même site
  - Reprise sur un autre site
- Plusieurs exécutions successives de la même procédure pour une seule demande

55

Panne du client

- le serveur est dit orphelin
- Réalisation de travaux inutiles (utilisation de ressources qui ne sont plus accessibles pour des tâches utiles)
- risque de confusion après relance du client entre les nouvelles réponses attendues et les réponses de l'ancienne requête

Solution :

- le serveur doit détruire les tâches inutiles et distinguer les requêtes utiles des vielles requêtes (une technique de numéros de séquence qui correspondant aux périodes d'activité successives du client)

56

Perte de messages

- Cas facile : pertes traitées par une couche transport fiable
- Cas difficile : il existe du mécanisme de traitement des pertes de messages à prévoir dans la conception du RPC
  - La réponse acquitte la demande
  - La prochaine requête acquitte la réponse
- Lancement de plusieurs exécutions de la même requête

57

Techniques de reprise arrière (1)

- **Reprise arrière complète** est nécessaire si
  - l'état du serveur indéterminé (panne en cours d'exécution)
  - le client est dans un état incertain (panne au moment de la modification en retour des variables persistantes du client)

-> l'état du client et celui du serveur doivent être restaurés aux valeurs avant le premier appel

- **Reprise arrière du serveur seul** est nécessaire si
  - il n'y a pas de perte de cohérence de l'état du serveur
  - il n'y a pas de sauvegarde de l'état client. C'est-à-dire, on peut s'en passer le client réalise seulement l'écriture des paramètres résultats.
  - Il n'y a pas de sauvegarde de l'état du serveur : au moment d'une tentative *n* l'état du serveur est donc celui qui résulte de la dernière tentative *n-1*.

58

Techniques de reprise arrière (2)

- **Reprise arrière du client seul** est nécessaire si
- Les états après sont conservés si des exécutions répétées du code serveur sur les données résultant de l'exécution précédente produisent des résultats identiques

59

Sémantique en cas de panne (1)

- En cas de panne, la sémantique est définie par le mécanisme de reprise **au moins une fois**
- C'est-à-dire, plusieurs appels possibles si perte de réponse du client
- C'est-à-dire, l'opération est acceptable si elle est idempotente du serveur

Définitions:

- une opération a le même effet qu'on l'applique une ou plusieurs fois, ou encore qu'en la réappliquant on ne modifiera pas le résultat
- *F* est *idempotente* si l'exécution de la procédure ne modifie pas l'état du serveur (lecture du kième article d'un fichier)
- *F* est *idempotente* si l'état du serveur est modifié seulement à la première exécution de *F* (l'écriture du kième article)
- L'écriture en fin de fichier est une opération *non idempotente*
- L'incrémentation d'une variable est *non idempotente*.

60

Sémantique en cas de panne (1)

- En cas de panne, la sémantique est définie par le mécanisme de reprise **au moins une fois**
- C'est-à-dire, plusieurs appels possibles si perte de réponse du client
- C'est-à-dire, l'opération est acceptable si elle est idempotente du serveur

Définitions :

- une opération a le même effet qu'on l'applique une ou plusieurs fois, ou encore qu'en la réappliquant on ne modifie pas le résultat
- *F* est *idempotente* si l'exécution de la procédure ne modifie pas l'état du serveur (lecture du kème article d'un fichier)
- *F* est *idempotente* si l'état du serveur est modifié seulement à la première exécution de *F* (l'écriture du kème article)
- L'écriture en fin de fichier est une opération *non idempotente*
- L'incrémentation d'une variable est *non idempotente*.

61

Sémantique en cas de panne (2)

- En cas de panne, la sémantique est définie par le mécanisme de reprise **au plus une fois**
- On n'exécute pas deux fois une fonction non idempotente
- On n'exécute pas deux fois une même procédure (identification des requêtes)
- Si tout va bien, le résultat est retourné à l'utilisateur. Si un problème est détecté, il est signalé à l'utilisateur
- Aucune tentative de reprise n'est effectuée. Elle est laissée entièrement à la charge du client

62

Conclusion

- **l'appel de procédure** (mode de communication support naturel de l'approche client-serveur) est une structure de contrôle bien connue.
- RPC s'intègre à l'univers réparti des concepts modernes de génie logiciel : approche objets, approches composants, modularité, encapsulation, réutilisation par délégation.
- RPC doit traiter les problèmes client-serveur suivants :
  - de conception
  - de désignation et de liaison
  - de présentation des données échangées
  - de synchronisation
  - de contrôle de concurrence
  - de tolérance aux pannes et de sécurité
  - de performances
  - de disponibilité d'outils conviviaux.

63