Niveau 7/OSI :

APPEL DE PROCEDURES DISTANTES (RPC)

Un système d'appel de procédures distantes (RPC: Remote Procedure Call) apporte une technique simple pour concevoir des applications distribuées, caractérisées par des temps de réponse courts. Il est à la base des systèmes **client-serveur**. De tels systèmes existent en milieu homogène depuis le début des années 1980 (notamment sur des systèmes Unix fournis par Sun , Xerox puis DEC).

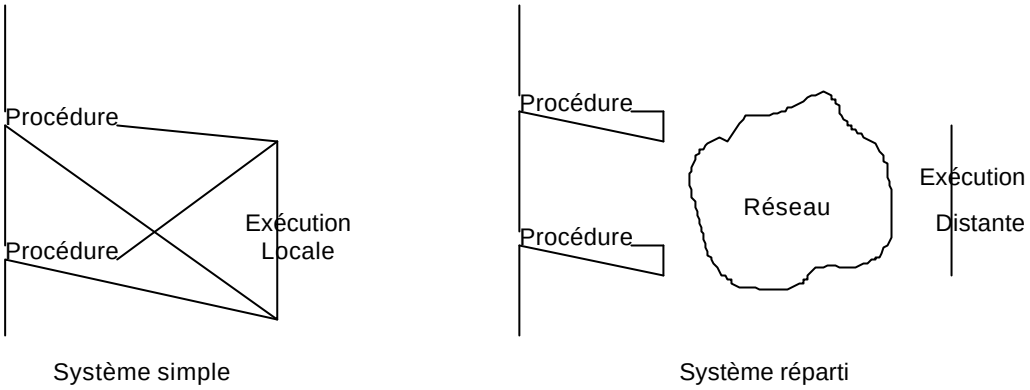
Pour les systèmes hétérogènes, un standard européen ECMA (ECMA127) a été spécifié. Un standard OSI est en cours d'élaboration depuis 1990; il reprend les concepts développés par l'ECMA (avec quelques adaptations). Il se situe au niveau 7/OSI (application) et utilise le service d'opérations distantes ROSE.

Dans le monde Unix, il est appelé à un essor important. Il doit constituer la base des applications réparties. Les projets de normalisation d'Unix prévoient de l'utiliser directement au dessus de la couche Transport : Transport OSI classe 4, TCP ou XTP à travers l'interface d'accès au service Transport XTI. Dans ce cas, on travaille en milieu homogène. Les RPC existants travaillent normalement en milieu homogène. Une certaine hétérogénéité est permise en utilisant le protocole XDR (eXternal Data Representation) pour coder les données transférées . (Ce produit joue le rôle de syntaxe de transfert).

1. Le modèle du standard RPC

1.1. Le concept de RPC ouvert

Le concept d'appel de procédures distantes consiste à étendre le mécanisme classique d'appel de procédures, existant dans la plupart des langages et s'exécutant avec des procédures dans des bibliothèques locales, à des systèmes ouverts où ces procédures sont exécutées sur un autre système accessible à travers un réseau.



Système simple Système réparti

Le programme appelant n'est normalement pas modifié. Sur le système local, des procédures "agents", représentant les procédures distantes qui effectuent réellement les traitements, sont invoquées par le programme appelant . Elles réalisent l'appel réel aux procédures distantes à travers le réseau. Ces procédures réelles sont installées sur un "serveur de procédures".

Les objectifs principaux sont :

- offrir au programmeur d'applications un mécanisme d'appel de procédures classiques prenant en charge les complications liées aux communications
- permettre l'utilisation de plusieurs langages de programmation, utilisés sous des systèmes d'exploitation différents, pour écrire et exécuter les différents composants d'une application distribuée
- permettre le développement d'une application distribuée dans un environnement local et de la rendre "distribuée" avec pas ou peu de changements.

Un système de RPC définit le moyen d'exécuter une procédure sur un (ou des) serveur(s) de procédures à travers une **interface standard D1**. Cette interface **définit les règles d'utilisation des procédures distantes** et permet une certaine **indépendance entre leur conception et leur appel pour exécution.**

1.2. Principe d'une application distribuée utilisant le RPC

Le composant distant d'une application est généralement construit comme un **service complet et nommé.** Les primitives qui composent ce service sont implantées sous la forme d'une **famille de procédures distantes**. Cette famille constitue le composant distant de l'application.

Ce service peut être soit spécifique d'une application soit un partage de ressources (à travers des procédures d'appel à ces ressources).

L'**appel, pour exécution**, des procédures constituant ce service distant depuis le service client doit être **transparent, c'est-à-dire constitué d'un appel normal de procédure**, écrit dans le langage de l'hôte (client).

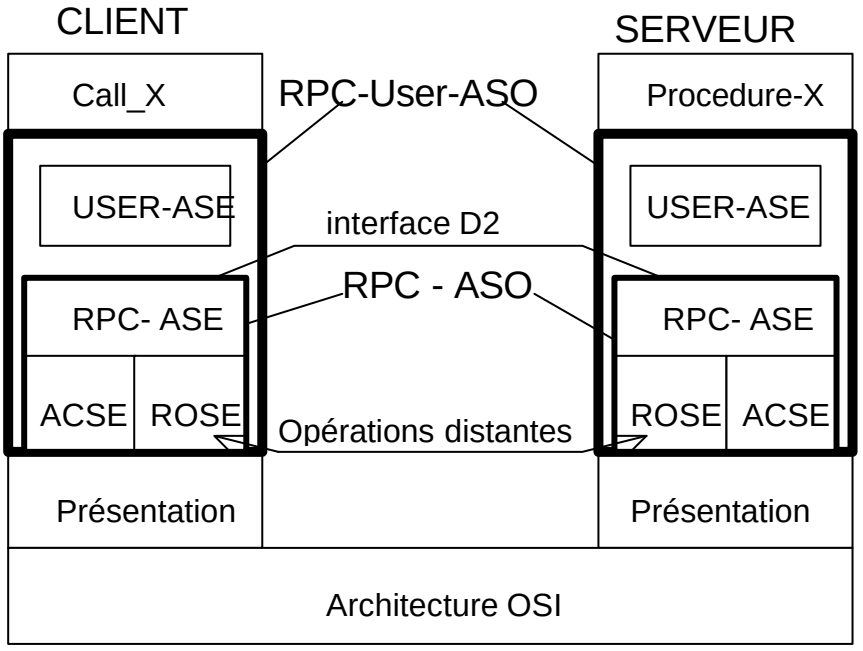
Cet appel a deux destinations : une immédiate dans l'hôte, l'autre, finale, dans le serveur de procédures. La destination immédiate de l'appel est une procédure locale, "agent" de la procédure distante, qui prend en charge les opérations e communication avec le serveur distant, dans lequel se trouve les procédures à exécuter, destination finale de l'appel.

La procédure local ("agent", "stub") attend, pour retourner au programme d'application, le résultat d'exécution, généré par la procédure appelée, ou une condition d'exception en cas d'anomalie d'exécution ou de communication. Comme dans un système localisé l'appel de procédure est "synchrone" (bloquant).

1.3. Architecture du modèle

Le schéma ci-dessous donne l'architecture d'un système RPC dans le cadre du Modèle de Référence OSI. Dans un système homogène, il peut être simplifié.

Pour s'exécuter, dans le cadre du modèle OSI, le RPC s'appuie sur un service



d'opérations distantes ROSE et le service commun d'association ACSE. Il est mis en oeuvre dans des entités du service application (ASE) spécialisées. L'ensemble de ces entités est regroupé dans des "objets de service application " (ASO) selon l'architecture illustrée.

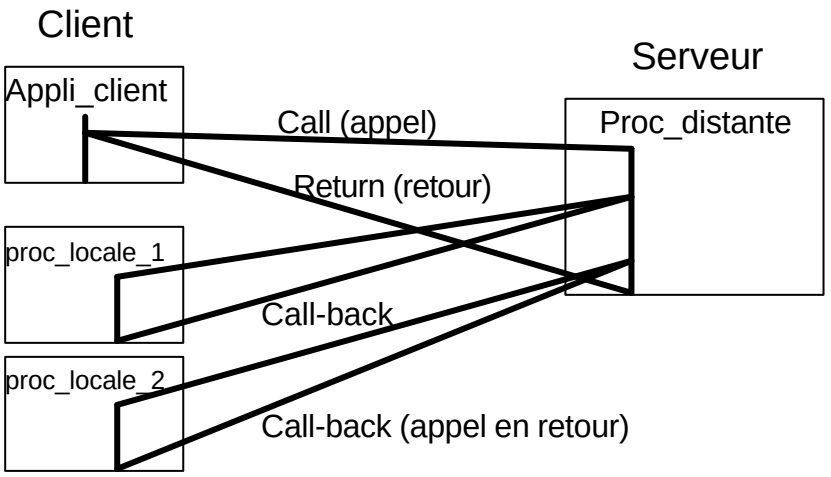
L'Objet de Service Application RPC (RPC-ASO) est composé de trois éléments de service d'Application : RPC-ASE, ROSE et ACSE. L'entité RPC-ASE traite le protocole RPC essentiellement en créant et en codant des opérations dont le transfert est demandé au service d'opérations distantes ROSE, après qu'une association ait été établie par ACSE. Chaque appel de procédure et chaque retour de procédure est traité par une demande d'opération distante. Cette opération comporte de nom de la procédure et ses paramètres.

Les appels du système client sont pris en compte par un élément de service d'Application spécifique (User-ASE) qui utilise le service RPC proprement dit à travers une interface standard D2.

Cette entité constitue avec l'objet de service Application RPC-ASO un objet de Service Application composite ; RPC-User-ASO. C'est l'interface avec ce service qui accessible au programme utilisateur.

Une architecture symétrique est réalisée coté serveur. Cet ensemble modulaire permet de réaliser l'indépendance vis à vis du système d'exploitation et du langage aussi bien coté serveur que coté client, ainsi que du système de communication.

Le RPC doit aussi supporter de manière transparente à l'utilisateur le mécanisme de call-back qui permet à une procédure exécutée sur le serveur distant de faire, pour son propre compte, des appels à des procédures de second niveau exécutées sur le système client. Un call-back peut être récursif jusqu'à un niveau de profondeur arbitraire (



1.4. Le service RPC OSI

Un RPC ouvert présente un degré important d'indépendance vis à vis de l'environnement hôte et rend à l'utilisateur un service spécifié à l'interface D2. Ce service est caractérisé par :

- son indépendance vis à vis du langage.

  Le service RPC (et le serveur d'opérations distantes) doivent être indépendants du ou des langages de programmation choisis pour l'application. Les procédures sur le serveur de procédures distantes sont aussi écrites dans un langage quelconque.

  Cependant on devra observer une restriction dans les paramètres d'appels ou de retour (quelque soit le langage ...) : ceux-ci doivent être effectivement transmis; il n'est bien sûr pas possible de ne passer qu'un pointeur sur ces données, celui-ci ne pouvant avoir qu'une signification locale. Dans le cas de tables ou de structures de données importantes, ceci peut éventuellement diminuer les performances du système, le temps de transfert pouvant parfois être assez important.

- son indépendance vis à vis de la machine

  La représentation des données manipulées par le RPC doit être indépendante de la machine hôte (et du serveur). Ceci est réalisé en codant ces données d'appel ou de retour en syntaxe ASN.1 (OSI 8825).

  Remarque : dans les RPC non-OSI on utilise souvent le langage XDR (extension des structures de données du langage C à cet effet).

- son indépendance vis à vis du réseau de télécommunication

  Elle est assurée par l'utilisation de protocoles standards. Cependant les performances du systèmes sont très dépendantes du réseau de télécommunications traversé et de l'efficacité des ensembles logiciels de communication. On trouvera Le RPC sur des réseaux offrant un débit suffisant : réseaux locaux (Ethernet, Token-Ring, FDDI) ou réseaux métropolitain ou étendus à haut débit (DQDB ou ATM).

Le service RPC comprend **sept primitives de service** disponibles à l'interface D2. Nous utiliserons les notations OSI en notant entre parenthèses les notations ECMA.

- **RPCBind (BeginRPCBinding)** :
  Etablit un lien entre les entités RPC en proposant des valeurs d'attributs. Cette primitive n'implique aucune interaction directe entre des procédures. Ce **service, toujours invoqué par le client** (architecture client-serveur) retourne à l'utilisateur une référence "Refbind" qui identifie le lien (Bind) unique géré par les RPC-ASE communicants.

- **RPCUnBind (End RPCBinding)** :
  Permet aux utilisateurs de détruire le lien de référence "RefBind", établi par la primitive ci-dessus, entre les RPC-ASE

- **(GetRPCAttributes)**
  Cette primitive n'a qu'une portée locale. Elle permet aux utilisateurs du service de prendre connaissance de certains paramètres associés au lien "RefBind" en cours.

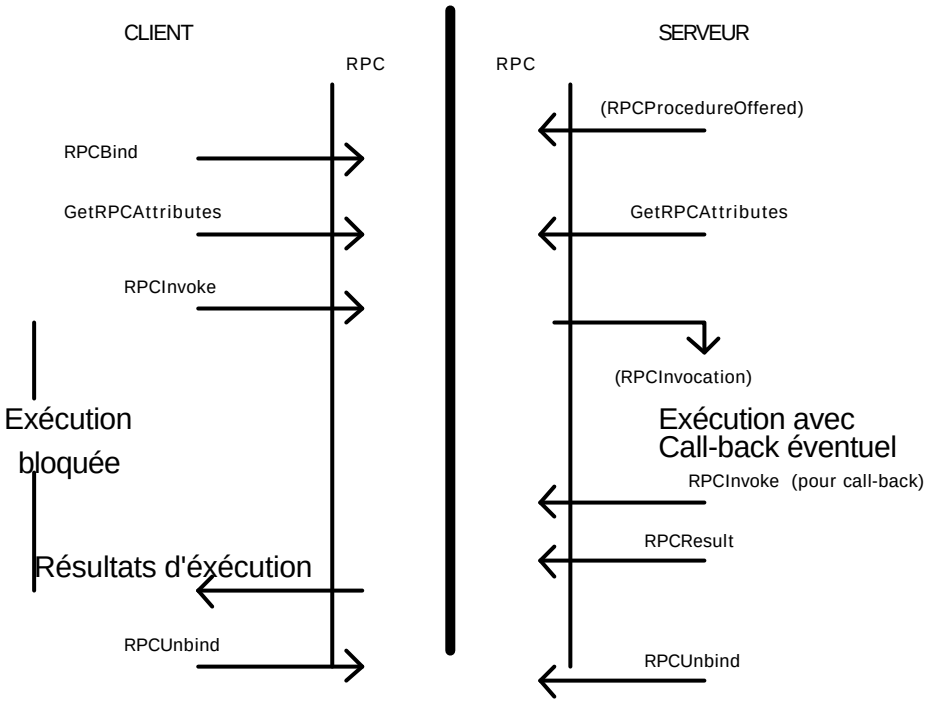
- **RPCInvoke (RPCCall)** :
  Permet de demander l'exécution d'un procédure distante particulière. C'est la primitive de base pour le client. Le serveur réservera l'utilisation de cette primitive à l'appel en call-back.

- **(RPCProcedureOffered)** :
  Permet au serveur de donner la liste des procédures offertes à l'exécution distante (offre au clients). Les procédures offertes en appel call-back sont aussi précisées dans cette liste.

- **(RPCInvocation)** :
  Cette primitive constitue l'indication d'appel, pour exécution, d'une procédure. Elle permet au RPC-ASE du serveur de demander l'exécution d'une procédure qui figure sur la liste des procédures offertes à exécution distante.

- **RPCResult (RPCReturn)** :
  Permet au serveur de confier les résultats d'exécution d'une procédure à son RPC-ASE. Les paramètres de retour sont fournis à l'utilisateur dans l'échange de paramètres de la procédure appelante qui se termine.

  Le schéma ci-dessous montre l'enchaînement de ces primitives.



1.5. Service d'opération distante (ROSE) nécessaire

Pour s'exécuter, le RPC défini par l'OSI ou l'ECMA s'appuie sur le serveur d'opérations distantes ROSE. RPC est caractérisé par

- son asymétrie : système client-serveur dans lequel l'initiative revient toujours au client
- le mécanisme synchrone d'exécution. Chaque appel de procédure est bloquant jusqu'à réception des résultats d'exécution. Les appels en retour (call-back) sont eux aussi bloquant pour l'exécution de la procédure distante et se déroulent sur le système client pendant que l'initiateur de l'appel original attend la fin de la procédure demandée.

Dans ces conditions, le service ROSE utilise une **classe 1 d'association** (seul l'initiateur de l'association peut invoquer des opérations) et une **classe 1 d'opérations (opération synchrone avec réponse en cas de succès ou d'échec)**. Les appels en retour (call-back) sont supportés par le mécanisme d'**opérations liées, de type parent-enfant**. L'opération résultant de l'appel originel est parent, les autres sont des opérations enfants.

2. Architectures réelles pour un RPC et exemples d'utilisation

2.1. Structure modulaire

Un élément de service Application RPC (RPC-ASE) se décompose en plusieurs modules différents sur le système client et le système serveur.

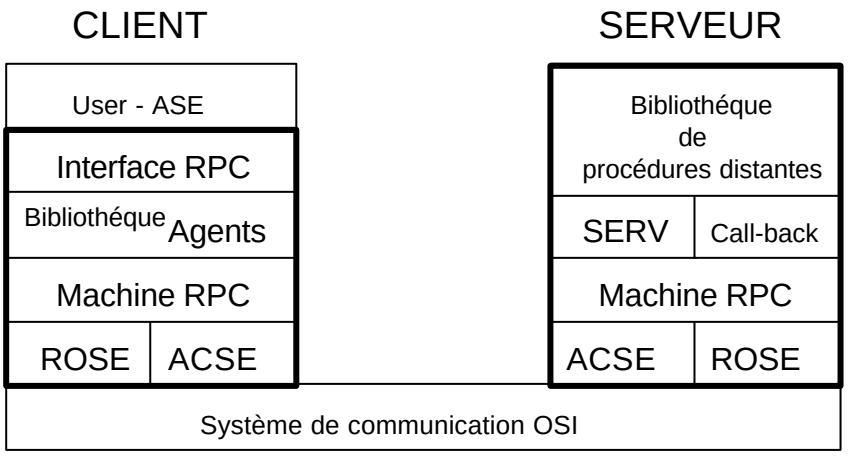
Coté client on doit trouver un module d'interface avec l'application client (User-Ase) de manière à faciliter l'accès au système depuis des applications très diverses. Ce module permet d'accéder à la bibliothèque des procédures "agents" ("Stub"). En effet à chaque procédure exécutable sur le serveur correspond une procédure locale spécifique et très brève chargée de récupérer les paramètres d'appel, de lancer le module RPC supportant le protocole proprement dit et de transmettre à l'appelant les résultats.

Sur le client et le serveur se trouvent ces modules RPC traitant le protocole proprement dit. Ils doivent en particulier coder les messages en langage ASN.1 et les placer dans des opérations distantes soumise à ROSE.

Sur le serveur, le destinataire final des appels est le système d'exécution des procédures. Celles-ci sont stockées dans une bibliothèque de procédures offertes. Elles peuvent être chargées et exécutées en fonction des demandes.

Un module logiciel reçoit les demandes d'exécution du RPC-ASE serveur et réalise l'appel réel de procédure. L'interface D1 peut n'être qu'implicite si le serveur est un système spécialisé. Un module spécial doit être implanté pour gérer les appels en retour; appelé par la procédure distante il sert d'interface avec le module supportant le protocole RPC proprement dit.

Le schéma ci-dessous, donné à titre indicatif, illustre ce type d'architecture.



2.2. Exemple d'utilisation

Le serveur offre un ensemble de procédures, notamment une procédure (EigenV) permettant le calcul des valeurs et vecteurs propres d'une matrice et un procédure (Print) permettant l'impression d'un fichier d'un système client sur une imprimante supportée par le serveur.

Des primitives, non standards, ont été introduites pour synchroniser les modules lors de l'exécution. Ce sont RPCReponse, RPCWaitBind, RPCWaitInvoke.

Sur le client

Programme_client

"
"
EigenV (matrice_entrée, valeurs_propres, vecteurs_propres, status)
"
Print (Fichier_matrice, status)
"

User-ASE

"
Interface_RPC (EigenV, matrice_entrée)
"
Interface_RPC (Print, Fichier_matrice)
"

Interface RPC

"
Appel Agent_EigenV (matrice_entrée)
"
Appel Agent_Print (fichier_matrice)

Agent_EigenV

```
begin
    RPCBind()
    RPCInvoke (Operation1,...)
        RPCReponse ( )
    RPCUnbind

end
```

Agent_Print

```
begin
    RPCBind
    RPCInvoke (Operation8,...)
        RPCReponse ()
    While call_back
        Appel_opération_enfant_1        ; ouvrir fichier
        Appel_opération_enfant_2        ; lire fichier
        Appel_opération_enfant_3        ; fermer fichier
         RPCResult(...)
            RPCReponse ()
    Endwhile
    RPCUnbind ()
end
```

Table des opérations locales

Opérations enfants	Entrée
1	Open_in
2	Read_Buffer
3	CloseFile

Sur le serveur

EigenV

```
Calcul (matrice_entrée, valeurs_propres, vecteurs_propres)
    begin

    end
```

Print

```
Printfile (fichier)
    begin
        appel_call_back (Num_oper_enfant_1)
        appel_call_back (Num_oper_enfant_2)
        appel_call_back (Num_oper_enfant_3)
    end
```

Serv

```
main ()
RPCOffer ()
While (en activité)
        RPCWaitBind
        RPCWaitInvocation
                exécuter procédure associée à numéro d'opération transmis
                par ROSE
        RPCResult
        RPCEnd
EndWhile
RPCEndOffer
```

Call-back

```
begin
     RPCInvoke (Num_oper_enfant)
     RPCReponse (...)
end
```

Opérations demandées et procédures

Procédure	Opération	Entrée
EigenV	1	Calcul
S_fichier	2	Open_in
S_fichier	3	Open_out
Print	8	Printfile

Les modules sont en fait réalisés par des procédures dans lesquelles sont appelées les procédures de niveau inférieur.

2.3. Implantation sous Unix sur un service de niveau 4/OSI

RPC, en milieu (quasi)-homogène, est implanté normalement au dessus du service Transport, que ce soit un transport OSI (classe 4) ou un service TCP. Cependant on utilise le plus souvent le **protocole sans connexion UDP** de l'ensemble Inet.

Dans les nouvelles versions d'Unix, il utilisera l'interface standard XTI.

L'ensemble RPC est constitué de 3 sous-couches. La sous-couche supérieure est complètement transparente au programmeur. Elle sert à administrer le RPC et par exemple à déterminer le nombre d'utilisateurs à un instant donné.

La couche intermédiaire est accessible directement depuis le programme utilisateur. Dans les bibliothèques Unix sont fournies des fonctions permettant de mettre en oeuvre le RPC. La plus simple est callrpc ( ) qui exécute un appel de procédure distante. La fonction registerrpc() fournit un numéro unique sur tout le système pour identifier la connexion client-serveur établie.

La couche inférieure est utilisée pour des applications plus sophistiquées qui demande la modification de certains paramètres du RPC fixés par défaut.

Si les données doivent être converties en langage de transfert XDR, on utilise les processus de sérialisation appelés par des procédures du type xdr_int (), xdr_long (), xdr_short (), xdr_string (), etc. Ces procédures sont utilisées dans une procédure xdr_simple () qui sera placée dans les paramètres d'appel de rpccall ()

Par exemple :
callrpc (hostname, PROGNUM, VERSNUM, PROCNUM, xdr_simple, &simple,...)

xdr_simple appelle dans son code xdr_int, xdr_short ou xdr_long par exemple.

Coté serveur s'effectue la désérialisation. Le logiciel lance le module de transport (par exemple UDP) et se met en attente d'attachements (Bind) d'un utilisateur. Suite à un rpccall, il enregistre la demande, fait un appel service pour réaliser la procédure demandée. Si la procédure a abouti normalement, il revoit la réponse à l'appelant.

Le serveur de ressources NFS (Network Files System) utilise RPC.

Dans un proche avenir, ce type de protocole devrait être utilisé de manière systématique pour traiter la majeure partie des applications réparties sous Unix.