Le Service RPC

Rappel

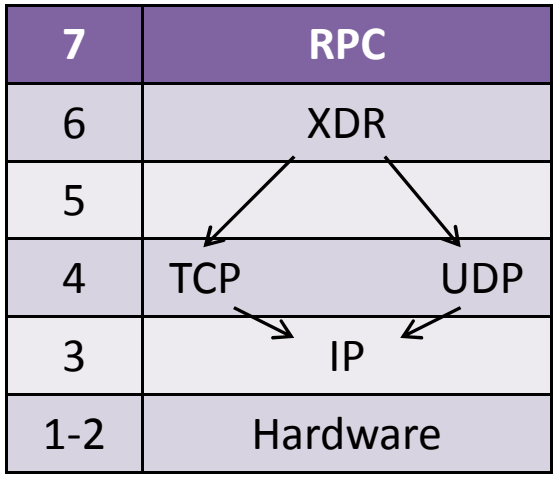
- Système réparti = ensemble de systèmes reliés entre eux pour réaliser une tâche commune
- Intergiciel :
  - Couche logicielle intermédiaire située entre l'application et le système d'exploitation de la machine
  - Permet de développer et déployer plus facilement une application répartie
  - Utilisation de fonctions d'aide au développement
    (création de souche ou squelette)
    - Masquer le complexité de l'infrastructure sous-jacente
    - Fournir des services communs

Le service RPC (1)

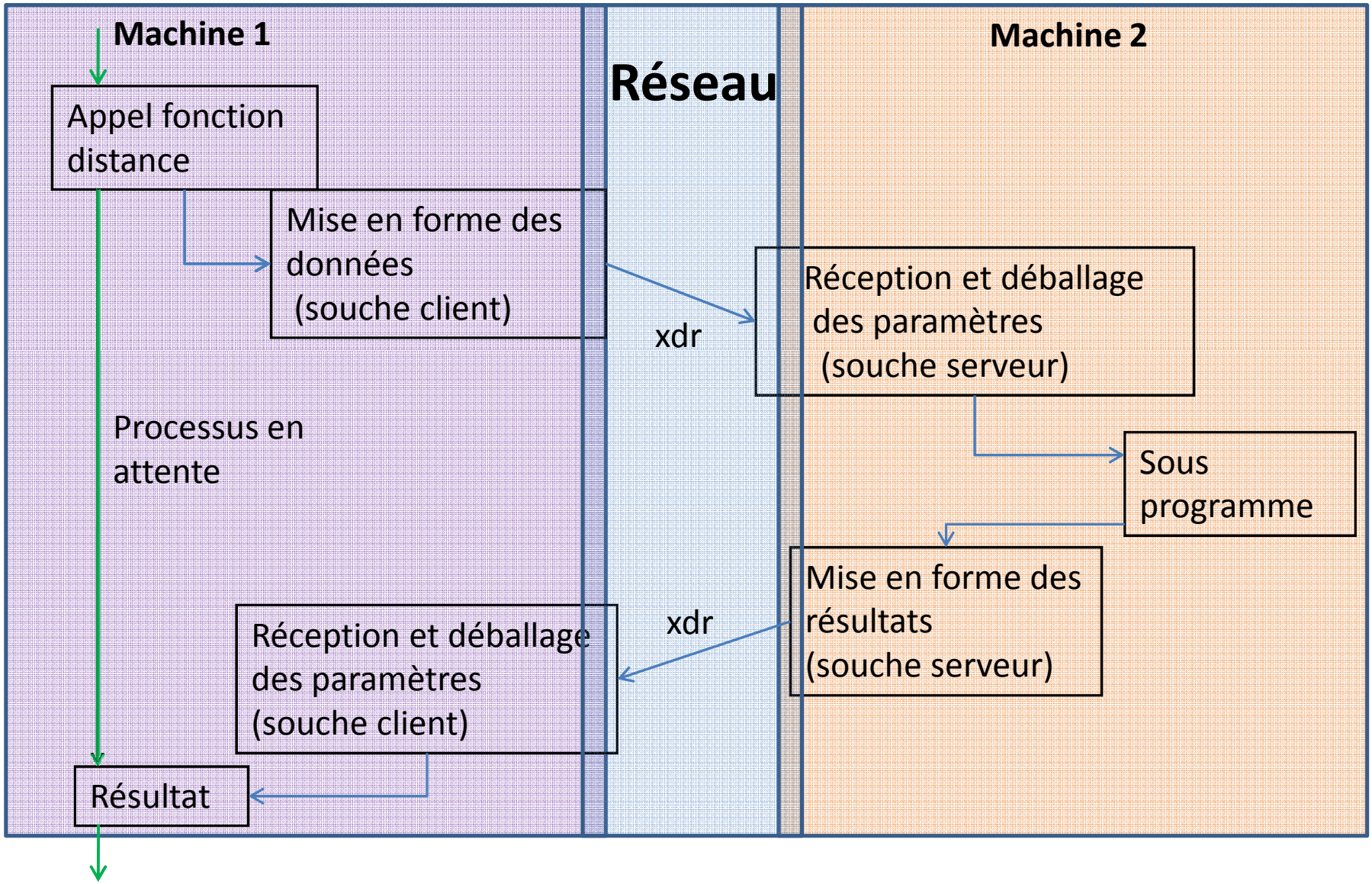
- RPC : Remote Procedure Call
  - But :
    - simplifier la programmation pour l'échange de messages
      - Le dialogue est géré de manière transparente pour le programmeur
      - Eviter la problématique de la programmation réseau
      - Possibilité de créer des souches ou squelettes
    - Permettre l'hétérogénéité entre les systèmes

Le service RPC (2)

- RPC défini par Sun en 1989
  - Remplacer les appels réseaux par des appels de sous-programme
    - Un processus appel une procédure avec paramètres et valeur de retour éventuel, qui s'exécute sur une machine distante (l'appel de procédure distante se programme de la même manière qu'en local…)
    - Utilisation d'un procédé synchrone
    - Echange des messages codés en XDR (eXternal Data Representation)
      → sérialisation des données
  - Exemples : nfs, rstats, …



Le service RPC (3)



Utilisation rpc

- Application RPC est identifié de **manière unique** :
  - Un numéro de programme
    - Ensemble d'applications qui ont des points communs (même sujet, même problématique, etc)
      - Serveur officiel : 0x00000000 – 0x1FFFFFFF
      - Adresse libre : 0x20000000 – 0x3FFFFFFF
  - Un numéro de procédure
    - Cela permet de faire le distinguo entre les applications pour un programme donné
  - Un numéro de version
    - Permet l'évolution sans couper le service.

Programmation rpc (1)

- Plusieurs niveaux de programmation
  - Pour l'interface simplifiée :
    - Fonction callrpc(..) : Côté client, permet de faire l'appel à la procédure distante
    - Fonction registerrpc(..) : Côté serveur, permet d'enregistrer la fonction au niveau du service rpc
    - Fonction svc_run() : Côté serveur, lance le service rpc

      ➡ Primitive bloquante

Programmation rpc (2)

- Callrpc() a 8 arguments:
  - 1) Nom de la machine distante (serveur rpc)
  - 2) N°programme, 3) N°version, 4) N°procédure
  - 5) Procédure pour "encoder" l'argument (XDR) et 6) l'argument
  - 7) Procédure pour "encoder" le résultat (XDR) et 8) le résultat

- Registerrpc() a 6 arguments:
  - 1) N°programme, 2) N°version, 3) N°procédure
  - 4) Chaîne de caractères indiquant la fonction associée au rpc
  - 5) Procédure pour "encoder" l'argument (XDR)
  - 6) Procédure pour "encoder" le résultat (XDR)

Remarques

- Les RPCs fonctionnent sur un port
  - →Nécessité d'avoir portmap qui tourne
    (le rpc, via rpcrregister vient s'enregistrer auprès du portmapper)
  - → rpcinfo –p hôte
- Chaque rpc qui fonctionne utilise un processus
  - Possibilité d'utiliser inetd

XDR (1)

- XDR : eXternal Data Representation
  - Permet de régler le problème de la non-unicité des données internes
  - Représentation standard indépendante du système (comme ASN.1)

- Utilisation de la sa **sérialisation**
  - **2 flux : XDR_ENCODE et XDR_DECODE**

Sérialisation XDR (1)

- Fonction prédéfinie pour sérialiser

  xdr_type(XDR * xdrptr, <type> *obj)

  avec **type = float, int, char, double, ..**

  - Flux XDR_ENCODE : transformation de obj en xdrptr
  - Flux XDR_DECODE : transformation de xdrptr en obj

  !! Xdr_string existe, mais entraîne des erreurs

  -> utilisation de xdr_vector

  xdr_vector (XDR *xdrptr, void *pt, in taille, int taille_case, xdrproc_t xdr_type)

  Ex : xdr_vector(xdrp, p->phrase, 1000,sizeof(char),(xdrproc_t)xdr_char)

Sérialisation XDR (2)

- Fonction prédéfinie pour sérialiser

Pour les structures, on sérialise chaque membre :

Exemple : struct struct_1 {
type1 champ1;
type2 champ2; }

```
Boot_t xdr_struct1 (XDR *xdrptr, struct struct1 *ptr)
    {
    if (!xdr_type1 (xdrptr, &ptr->champ1))  return false;
    if (!xdr_type2 (xdrptr, &ptr->champ2)) return false;
    return true; }
```

rpcl

- Description d'un programme dans le langage rpcl
  - Une procédure ne possède qu'un seul argument en entrée et en résultat

  ⟶ Utilisation de structures

- Syntaxe proche du C
  - Void , int, float, struct, string, tableaux,…

Fichier de définition du service → suffixe en .x

Exemple rpcl (1)

- Interface calcul.x

```
const max_nb = 100;
struct couple {
        float a;
        float b;}
typedef struct couple couple;

program  PROG_TEST {
    version ARITH_VERS {
        int  MULT_PROC (int) = 1;                    /*N° de procédure */
        couple RAC_PROC (struct couple) =2;
                } = 2;                               /* N° de version */
        }= 0x24545453;                               /* N° de programme */
```

Exemple rpcl (2)

- Génération des fichiers : rpcgen nom.x
- 4 fichiers sont créées :
  - Fonction de traduction XDR dans “nom_xdr.c” (généré)
  - Projection du contrat dans “nom.h” (généré)
- Côté client
  - nom_client.c : fonction main(), utilisation du service…
  - nom_clnt.c (généré): souche cliente
- Côté serveur
  - nom_serveur.c : implantation du service dans la fonction
  - calculation_svc.c (généré) : fonction main() + souche serveur