Groupe de travail Réseau
Request for Comments : 1833
Catégorie : Sur la voie de la normalisation

R. Srinivasan, Sun Microsystems
août 1995
Traduction Claude Brière de L'Isle

Protocoles de liaison pour RPC ONC version 2

Résumé
Le présent document décrit les protocoles de liaison utilisés en conjonction avec les protocoles d'appel de procédure à distance (RPC, *Remote Procedure Call*) de calcul de réseau ouvert (ONC, *Open Network Computing*) (ONC RPC v2).

Table des matières

1. Introduction

Le présent document spécifie les protocoles de liens utilisés en conjonction avec ONC RPC version 2. Comme pré requis, on suppose que le lecteur est familiarisé avec les [RFC1831] et [RFC1832] qui décrivent les protocoles ONC RPC version 2 et de représentation de données externes (XDR, *eXternal Data Representation*).

Un service de RPC est identifié par son numéro de programme RPC, son numéro de version, et l'adresse de transport où il peut être joint. L'adresse de transport, à son tour, consiste en une adresse réseau et un sélecteur de transport. Dans le cas d'un service disponible sur TCP/IP ou UDP/IP, l'adresse réseau va être une adresse IP, et le sélecteur de transport va être un numéro d'accès TCP ou UDP. Un programme client a besoin de connaître le numéro de programme RPC, le numéro de version, et l'adresse de transport correspondant à un service afin d'utiliser le service. Parmi eux, le numéro de programme RPC et le numéro de version sont généralement incorporés dans le programme client, au titre de la définition de service. Le composant d'adresse réseau de l'adresse de transport est généralement disponible dans un nom de service, ou est donné dans un paramètre au programme client. Le sélecteur de transport (c'est-à-dire, l'accès TCP ou UDP) est généralement déterminé dynamiquement, et varie à chaque invocation du service. Les programmes de serveur allouent une adresse de transport, et l'enregistrent auprès d'un service de recherche bien connu (parce qu'il utilise un sélecteur de transport fixé, et réside à la même adresse réseau que le serveur). Les programmes clients consultent le service de recherche afin d'obtenir l'adresse de transport du serveur.

Un tel service de recherche est très désirable parce que la gamme des sélecteurs de transport bien connus est très petite pour certains transports et que le nombre de services est potentiellement très grand. En ne faisant fonctionner le service de recherche que sur un sélecteur de transport bien connu, les adresses de transport d'autres programmes distants peuvent être assurées en interrogeant le service de recherche.

Le présent document décrit trois versions d'un service de recherche, qui utilisent toutes le même numéro de programme RPC (100000). Elles utilisent toutes l'accès 111 sur les transports TCP et UDP. Les versions 3 et 4 sont décrites à la Section 2 ("Protocole de programme RPCBIND"). La version 2 est décrite à la Section 3 ("Protocole de programme de transposeur d'accès").

Les caractéristiques distinctives de RPCBIND (versions 3 et 4) sont que ce protocole utilise un format indépendant du transport pour l'adresse de transport, appelé format d'adresse universelle. Une adresse en format d'adresse universelle est une chaîne ASCII représentant l'adresse dépendant du transport. La représentation de chaîne des adresses correspondant au transport est définie par l'autorité d'adressage pour le transport. Le protocole RPCBIND peut être utilisé pour lier les clients et serveurs ONC RPC sur tout transport.

Par ailleurs, le transposeur d'accès (version 2) est un plus ancien protocole qui est spécifique de TCP et UDP. Il traite les accès TCP et UDP directement.

2. Protocole du programme RPCBIND

Le programme RPCBIND transpose les numéros de programme et version RPC en adresses universelles, rendant donc possible un lien dynamique des programmes distants.

Le programme RPCBIND est lié à une adresse bien connue de chaque transport pris en charge, et d'autres programmes enregistrent avec lui leur adresse de transport allouée dynamiquement. Le programme RPCBIND rend alors ces adresses publiquement disponibles.

Le programme RPCBIND aide aussi au RPC de diffusion. Un certain programme RPC va généralement avoir des liens différents d'adresse de transport sur des machines différentes, de sorte qu'il n'y a pas de moyen de diffuser directement à tous ces programmes. Le programme RPCBIND a cependant une adresse bien connue. Ainsi, pour diffuser à un certain programme, le client envoie en fait son message au programme RPCBIND situé à l'adresse de diffusion. Chaque instance du programme RPCBIND qui prend la diffusion appelle alors le service local spécifié par le client. Quand le programme RPCBIND obtient la réponse du service local, il renvoie la réponse au client.

2.1 Spécification du protocole RPCBIND (en langage RPC)

```
/*
 * rpcb_prot.x
 *Protocole rpcbind, versions 3 et 4, en langage RPC.
 */

/*
 * Adresse rpcbind pour TCP/UDP
 */
const RPCB_PORT = 111;

/*
 * Transposition de (programme, version, identifiant de réseau) en adresse
 *
 * Identifiant de réseau  (r_netid) :
 * C'est une chaîne qui représente une identification locale pour un réseau. Elle est définie par un administrateur de système fondée sur des conventions locales, et ne peut pas dépendre d'avoir la même valeur sur chaque système.
 */
struct rpcb {
 unsigned long r_prog;                /* numéro de programme */
 unsigned long r_vers;                /* numéro de version */
 string r_netid<>;                    /* identifiant de réseau */
 string r_addr<>;                     /* adresse universelle */
 string r_owner<>;                    /* possesseur de ce service */
};

struct rp__list {
 rpcb rpcb_map;
 struct rp__list *rpcb_next;
};
```

```
typedef rp__list *rpcblist_ptr;          /* résultats de RPCBPROC_DUMP */

/*
 * Arguments des appels distants
 */
struct rpcb_rmtcallargs {
 unsigned long prog;                     /* numéro de programme */
 unsigned long vers;                     /* numéro de version */
 unsigned long proc;                     /* numéro de procédure */
 opaque args<>;                          /* argument */
};


/*
 * Résultats de l'appel distant
 */
struct rpcb_rmtcallres {
 string addr<>                           /* adresse universelle distante */
 opaque results<>;                       /* résultat */
};
```

/* rpcb_entry contient une adresse fusionnée d'un service sur un transport particulier, plus les informations de configuration de réseau associées. Une liste d'éléments rpcb_entry est retournée par RPCBPROC_GETADDRLIST. La signification et les valeurs utilisées pour les champs r_nc_* sont données ci-dessous.
 *
 * Identifiant de réseau (r_nc_netid) :

 * C'est une chaîne qui représente une identification locale pour un réseau. Elle est définie par un administrateur de système fondée sur des conventions locales, et ne peut pas dépendre d'avoir la même valeur sur chaque système.
*
 * Sémantique du transport (r_nc_semantics):
 * Cela représente le type de transport, et a les valeurs suivantes :
 * NC_TPI_CLTS (1) Sans connexion
 * NC_TPI_COTS (2) En mode connexion
 * NC_TPI_COTS_ORD (3) En mode connexion avec fermeture en douceur
 * NC_TPI_RAW (4) Transport brut
 *
 * Famille de protocole (r_nc_protofmly) :
 * Cela identifie la famille à laquelle appartient le protocole. Les valeurs suivantes sont définies :
 * NC_NOPROTOFMLY "-"
 * NC_LOOPBACK "loopback"
 * NC_INET "inet"
 * NC_IMPLINK "implink"
 * NC_PUP "pup"
 * NC_CHAOS "chaos"
 * NC_NS "ns"
 * NC_NBS "nbs"
 * NC_ECMA "ecma"
 * NC_DATAKIT "datakit"
 * NC_CCITT "ccitt"
 * NC_SNA "sna"
 * NC_DECNET "decnet"
 * NC_DLI "dli"
 * NC_LAT "lat"
 * NC_HYLINK "hylink"
 * NC_APPLETALK "appletalk"
 * NC_NIT "nit"
 * NC_IEEE802 "ieee802"
 * NC_OSI "osi"
 * NC_X25 "x25"

```
 *  NC_OSINET           "osinet"
 *  NC_GOSIP            "gosip"
 *
 * Noms de protocole (r_nc_proto) :
 * Cela identifie un protocole dans une famille. Ceux actuellement définis sont :
 *  NC_NOPROTO          "-"
 *  NC_TCP              "tcp"
 *  NC_UDP              "udp"
 *  NC_ICMP             "icmp"
 */
struct rpcb_entry {
 string       r_maddr<>;              /* adresse fusionnée de service */
 string       r_nc_netid<>;           /* champ identifiant de réseau */
 unsigned long  r_nc_semantics;       /* sémantique du transport */
 string       r_nc_protofmly<>;       /* famille de protocole */
 string       r_nc_proto<>;           /* nom du protocole */
};

/*
 * Liste des adresses prises en charge par un service.
 */
struct rpcb_entry_list {
 rpcb_entry rpcb_entry_map;
 struct rpcb_entry_list *rpcb_entry_next;
};

typedef rpcb_entry_list *rpcb_entry_list_ptr;

/*
 *Statistiques de rpcbind
 */

const rpcb_highproc_2 = RPCBPROC_CALLIT;
const rpcb_highproc_3 = RPCBPROC_TADDR2UADDR;
const rpcb_highproc_4 = RPCBPROC_GETSTAT;

const RPCBSTAT_HIGHPROC = 13;     /* nombre de processus dans rpcbind V4 plus un */
const RPCBVERS_STAT        = 3;     /* seulement pour rpcbind V2, V3 et V4 */
const RPCBVERS_4_STAT      = 2;
const RPCBVERS_3_STAT      = 1;
const RPCBVERS_2_STAT      = 0;

/* Liste de lien de toutes les statistiques sur getport et getaddr */
struct rpcbs_addrlist {
 unsigned long prog;
 unsigned long vers;
 int success;
 int failure;
 string netid<>;
 struct rpcbs_addrlist *next;
};

/* Liste de lien de toutes les statistiques sur rmtcall */
struct rpcbs_rmtcalllist {
 unsigned long prog;
 unsigned long vers;
 unsigned long proc;
 int success;
 int failure;
 int indirect;                       /* que ce soit callit ou indirect */
 string netid<>;
```

```
 struct rpcbs_rmtcalllist *next;
};

typedef int rpcbs_proc[RPCBSTAT_HIGHPROC];
typedef rpcbs_addrlist *rpcbs_addrlist_ptr;
typedef rpcbs_rmtcalllist *rpcbs_rmtcalllist_ptr;

struct rpcb_stat {
 rpcbs_proc              info;
 int                     setinfo;
 int                     unsetinfo;
 rpcbs_addrlist_ptr      addrinfo;
 rpcbs_rmtcalllist_ptr   rmtinfo;
};


/*
 * Une structure rpcb_stat est retournée pour chaque version de rpcbind surveillée.
 */

typedef rpcb_stat rpcb_stat_byvers[RPCBVERS_STAT];

/*
 * Structure netbuf, utilisée pour mémoriser la forme spécifique du transport d'une adresse de transport universelle.
 */
struct netbuf {
 unsigned int maxlen;
 opaque buf<>;
};


/*
 * Procédures rpcbind
 */
program RPCBPROG {
 version RPCBVERS {
    bool
    RPCBPROC_SET(rpcb) = 1;

    bool
    RPCBPROC_UNSET(rpcb) = 2;

    string
    RPCBPROC_GETADDR(rpcb) = 3;

    rpcblist_ptr
    RPCBPROC_DUMP(void) = 4;

    rpcb_rmtcallres
    RPCBPROC_CALLIT(rpcb_rmtcallargs) = 5;

    unsigned int
    RPCBPROC_GETTIME(void) = 6;

    netbuf
    RPCBPROC_UADDR2TADDR(string) = 7;

    string
    RPCBPROC_TADDR2UADDR(netbuf) = 8;
 } = 3;
```

```
 version RPCBVERS4 {
    bool
    RPCBPROC_SET(rpcb) = 1;

    bool
    RPCBPROC_UNSET(rpcb) = 2;

    string
    RPCBPROC_GETADDR(rpcb) = 3;

    rpcblist_ptr
    RPCBPROC_DUMP(void) = 4;

/*
* Note : RPCBPROC_BCAST a la même fonctionnalité que CALLIT ; le nouveau nom est destiné à indiquer que cette
        procédure devrait être utilisée pour RPC en diffusion, et RPCBPROC_INDIRECT devrait être utilisé pour les
        appels indirects.
*/
    rpcb_rmtcallres
    RPCBPROC_BCAST(rpcb_rmtcallargs) = RPCBPROC_CALLIT;

    unsigned int
    RPCBPROC_GETTIME(void) = 6;

    netbuf
    RPCBPROC_UADDR2TADDR(string) = 7;

    string
    RPCBPROC_TADDR2UADDR(netbuf) = 8;

    string
    RPCBPROC_GETVERSADDR(rpcb) = 9;

    rpcb_rmtcallres
    RPCBPROC_INDIRECT(rpcb_rmtcallargs) = 10;

    rpcb_entry_list_ptr
    RPCBPROC_GETADDRLIST(rpcb) = 11;

    rpcb_stat_byvers
    RPCBPROC_GETSTAT(void) = 12;
 } = 4;
} = 100000;
```

2.2 Fonctionnement de RPCBIND

RPCBIND est contacté au moyen d'une adresse allouée spécifique du transport utilisé. Pour TCP/IP et UDP/IP, par exemple, c'est le numéro d'accès 111. Chaque transport a une telle adresse bien connue qui lui est allouée. Voici une description de chacune des procédures prises en charge par RPCBIND.

2.2.1 RPCBIND Version 3

RPCBPROC_SET :
Quand un programme devient disponible sur une machine, il s'enregistre auprès du RPCBIND qui fonctionne sur la même machine. Le programme passe son numéro de programme "r_prog", son numéro de version "r_vers", l'identifiant de réseau "r_netid", l'adresse universelle "r_addr", et le possesseur du service "r_owner". La procédure retourne une réponse booléenne dont la valeur est VRAI si la procédure a réussi à établir la transposition et FAUX autrement. La procédure refuse d'établir une transposition si il en existe déjà une pour l'ensemble ordonné de ("r_prog", "r_vers", "r_netid"). Noter que ni "r_netid" ni "r_addr" ne peut être NUL, et que "r_netid" devrait être un identifiant de réseau valide sur la machine qui fait l'appel.

RPCBPROC_UNSET :
Quand un programme devient indisponible, il devrait se désenregistrer du programme RPCBIND sur la même machine. Les paramètres et résultats ont la même signification que dans RPCBPROC_SET. La transposition du triplet ("r_prog", "r_vers", "r_netid") en "r_addr" est supprimée. Si "r_netid" est NUL, toutes les transpositions spécifiées par l'ensemble ordonné de ("r_prog", "r_vers", *) et les adresses universelles correspondantes sont supprimées. Seul le possesseur du service ou le super utilisateur est autorisé à supprimer un service.

RPCBPROC_GETADDR :
Étant donnés un numéro de programme "r_prog", un numéro de version "r_vers", et un identifiant de réseau "r_netid", cette procédure retourne l'adresse universelle sur laquelle le programme attend les demandes d'appel. Le champ "r_netid" de l'argument est ignoré et le "r_netid" est déduit de l'identifiant de réseau du transport sur lequel la demande est venue.

RPCBPROC_DUMP :
Cette procédure fait la liste de toutes les entrées dans la base de données de RPCBIND. La procédure ne prend aucun paramètre et retourne une liste de programme, version, identifiant de réseau, et adresses universelles.

RPCBPROC_CALLIT :
Cette procédure permet à un appelant d'invoquer une autre procédure distante sur la même machine sans connaître l'adresse universelle de la procédure distante. Elle est destinée à la prise en charge des diffusions à des programmes distants arbitraires via l'adresse universelle de RPCBIND. Les paramètres "prog", "vers", "proc", et args sont le numéro de programme, numéro de version, numéro de procédure, et les paramètres de la procédure distante.

Note : cette procédure envoie une réponse seulement si la procédure a été exécutée avec succès et est silencieuse (pas de réponse) autrement.

La procédure retourne l'adresse universelle du programme distant, et le résultat de la procédure distante.

RPCBPROC_GETTIME :
Cette procédure retourne l'heure locale sur sa propre machine en secondes depuis minuit le premier janvier 1970.

RPCBPROC_UADDR2TADDR :
Cette procédure convertit les adresses universelles en adresses spécifiques du transport.

RPCBPROC_TADDR2UADDR :
Cette procédure convertit les adresses spécifiques du transport en adresses universelles.

2.2.2 RPCBIND, Version 4

La version 4 du protocole RPCBIND inclut toutes les procédures ci-dessus, et en ajoute plusieurs autres.

RPCBPROC_BCAST :
Cette procédure est identique à la procédure de la version 3 de RPCBPROC_CALLIT. Le nouveau nom indique que la procédure devrait seulement être utilisée pour les RPC en diffusion. RPCBPROC_INDIRECT, défini ci-dessous, devrait être utilisé pour les appels RPC indirects.

RPCBPROC_GETVERSADDR :
Cette procédure est similaire à RPCBPROC_GETADDR. La différence est que le champ "r_vers" de la structure rpcb peut être utilisé pour spécifier la version intéressante. Si cette version n'est pas enregistrée, aucune adresse n'est retournée.

RPCBPROC_INDIRECT :
Similaire à RPCBPROC_CALLIT. Au lieu d'être silencieuse sur les erreurs (comme que le programme n'est pas enregistré sur le système) cette procédure retourne une indication de l'erreur. Cette procédure ne devrait pas être utilisée pour RPC en diffusion. Elle est destinée à être seulement utilisée avec les appels RPC indirects.

RPCBPROC_GETADDRLIST :
Cette procédure retourne une liste des adresses pour l'entrée rpcb donnée. Le client peut être capable d'utiliser le résultat pour déterminer des transports de remplacement qu'il peut utiliser pour communiquer avec le serveur.

RPCBPROC_GETSTAT :
Cette procédure retourne les statistiques sur l'activité du serveur RPCBIND. Les informations font la liste du nombre et des

sortes de demandes que le serveur a reçues.

Note : toutes les procédures sauf RPCBPROC_SET et RPCBPROC_UNSET peuvent être invoquées par les clients qui fonctionnent sur une machine autre que celle où RPCBIND fonctionne. RPCBIND n'accepte que des demandes RPCBPROC_SET et RPCBPROC_UNSET par les clients qui fonctionnent sur la même machine que le programme RPCBIND.

3. Protocole du programme de transposeur d'accès

Le programme de transposeur d'accès transpose le programme RPC et les numéros de version en numéros d'accès spécifiques du transport. Ce programme rend possible un lien dynamique des programmes distants. Le protocole de transposeur d'accès diffère des plus récents protocoles RPCBIND en ce qu'il est spécifique du transport dans son traitement d'adresses.

3.1 Spécification du protocole de transposeur d'accès (en langage RPC)

```
const PMAP_PORT = 111;                  /* numéro d'accès portmapper */
```

Transposition de (programme, version, protocole) en numéro d'accès:

```
struct mapping {
  unsigned int prog;
  unsigned int vers;
  unsigned int prot;
  unsigned int port;
};
```

Valeurs acceptées pour le champ "prot" :

```
const IPPROTO_TCP = 6;              /* numéro de protocole pour TCP/IP */
const IPPROTO_UDP = 17;             /* numéro de protocole pour UDP/IP */
```

Liste des transpositions :

```
struct *pmaplist {
  mapping map;
  pmaplist next;
};
```

Arguments de callit :

```
struct call_args {
  unsigned int prog;
  unsigned int vers;
  unsigned int proc;
  opaque args<>;
};
```

Résultat de callit :

```
struct call_result {
  unsigned int port;
  opaque res<>;
};
```

Procédures de transposeur d'accès :

```
program PMAP_PROG {
```

```
    version PMAP_VERS {
      void
      PMAPPROC_NULL(void)        = 0;

      bool
      PMAPPROC_SET(mapping)      = 1;

      bool
      PMAPPROC_UNSET(mapping)    = 2;

      unsigned int
      PMAPPROC_GETPORT(mapping)  = 3;

      pmaplist
      PMAPPROC_DUMP(void)        = 4;

      call_result

      PMAPPROC_CALLIT(call_args) = 5;
    } = 2;
  } = 100000;
```

3.2 Fonctionnement du protocole de transposeur d'accès

Le programme de transposition d'accès prend actuellement en charge deux protocoles (UDP et TCP). Le transposeur d'accès est contacté en lui parlant sur le numéro d'accès alloué 111 (SUNRPC) sur l'un ou l'autre de ces protocoles.

Voici une description de chacune des procédures de transposition d'accès :

PMAPPROC_NULL :
Cette procédure ne fonctionne pas. Par convention, la procédure zéro de tout protocole ne prend pas de paramètre et ne retourne pas de résultat.

PMAPPROC_SET :
Quand un programme devient d'abord disponible sur une machine, il s'enregistre auprès du programme de transposeur d'accès sur la même machine. Le programme passe son numéro de programme "prog", son numéro de version "vers", son numéro de protocole de transport "prot", et l'accès "port" sur lequel il attend une demande de service. La procédure retourne une réponse booléenne dont la valeur est "VRAI" si la procédure a réussi à établir la transposition et "FAUX" autrement. La procédure refuse d'établir une transposition si il en existe déjà une pour le triplet "(prog, vers, prot)".

PMAPPROC_UNSET :
Quand un programme devient indisponible, il devrait se désenregistrer du programme de transposeur d'accès sur la même machine. Les paramètres et résultats ont une signification identique à celles de "PMAPPROC_SET". Les champs protocole et numéro d'accès de l'argument sont ignorés.

PMAPPROC_GETPORT :
Étant donnés un numéro de programme "prog", un numéro de version "vers", et un numéro de protocole de transport "prot", cette procédure retourne le numéro d'accès sur lequel le programme attend les demandes d'appel. Une valeur d'accès toute de zéros signifie que le programme n'a pas été enregistré. Le champ "port" de l'argument est ignoré.

PMAPPROC_DUMP :
Cette procédure énumère toutes les entrées dans la base de données du transposeur d'accès. La procédure ne prend pas de paramètre et retourne une liste des valeurs de programme, version, protocole, et accès.

PMAPPROC_CALLIT :
Cette procédure permet à un client d'invoquer une autre procédure distante sur la même machine sans connaître le numéro d'accès de la procédure distante. Elle est destinée à prendre en charge les diffusions à des programmes distants arbitraires via l'accès bien connu du transposeur d'accès. Les paramètres "prog", "vers", "proc", et les octets de "args" sont le numéro de programme, le numéro de version, le numéro de procédure, et les paramètres de la procédure distante.

Notes :
(1) Cette procédure n'envoie une réponse que si la procédure a été exécutée avec succès et est silencieuse (pas de réponse) autrement.
(2) Le transposeur d'accès communique avec le programme distant en utilisant seulement UDP.

La procédure retourne le numéro d'accès du programme distant, et la réponse est celle de la procédure distante.

Références

[RFC1831] R. Srinivasan, "RPC : Spécification de la version 2 du protocole d'appel de procédure à distance", août 1995. *(Expérimentale, Obsolète, voir* RFC5531)

[RFC1832] R. Srinivasan, "XDR : norme de représentation des données externes", août 1995. *(Obsolète, voir* RFC4506) *(D.S.)*

Considérations sur la sécurité

Les questions de sécurité ne sont pas discutées dans le présent mémoire.

Adresse de l'auteur

Raj Srinivasan
Sun Microsystems, Inc.
ONC Technologies
2550 Garcia Avenue
M/S MTV-5-40
Mountain View, CA 94043
USA

téléphone : 415-336-2478
Fax : 415-336-6015
mél : raj@eng.sun.com