Network Working Group
Request for Comments : 951
Septembre 1995

Bill Croft (Stanford University)
John Gilmore (Sun Microsystems)

Protocole d’amorçage (BOOTP)

Table des matières

1 Statut de ce document

Ce RFC suggère un protocole pour la communauté ARPA-Internet, et invite toute discussion et suggestion visant à son amélioration. La distribution de ce document est libre.
Cette traduction française a été effectuée par Frédéric Delanoy.
Vous pouvez le contacter ici.

2 Résumé

Ce RFC décrit un protocole d'amorçage IP/UDP nommé BOOTP, qui permet à une machine cliente sans disque dur de découvrir sa propre adresse IP, l'adresse d'un hôte serveur, et le nom d'un fichier à charger en mémoire pour exécution. On peut se représenter l'amorçage comme une opération se produisant en *deux phases*. Ce RFC décrit la première phase, qui pourrait être appelée « détermination d'adresse et sélection de fichier de démarrage ». Après l'obtention de cette adresse et de l'information sur le nom de fichier, on passe à la seconde phase de l'amorçage où se produit un transfert de fichier. Celui-ci utilisera typiquement le protocole TFTP [9], puisqu'il est escompté que les deux phases résident dans la PROM du client. Néanmoins, BOOTP pourrait également fonctionner avec d'autres protocoles comme SFTP [3] ou FTP [6].

Nous suggérons que le logiciel de la PROM du client fournisse un moyen d'effectuer un amorçage complet sans intervention de l'utilisateur. C'est le type d'amorçage qui se produirait lors d'une mise sous tension inattendue. Un mécanisme devrait être fourni pour que l'utilisateur puisse fournir manuellement l'adresse et l'information sur le nom de fichier nécessaires pour passer outre le protocole BOOTP et entrer directement dans la phase de transfert de fichier. Si un moyen de stockage non volatile est disponible, nous suggérons d'y conserver les réglages par défaut et de passer outre le protocole BOOTP à moins que ces réglages ne provoquent l'échec de la phase de transfert de fichier. S'il n'y a pas d'information cachée [1] disponible, l'amorçage devrait revenir à la phase 1 et utiliser BOOTP.

Voici un bref aperçu du protocole :

1. Un seul échange de paquets est effectué. Des temporisateurs sont utilisés pour la retransmission jusqu'à la réception d'une réponse. La même composition de paquet est utilisée dans les deux directions. Des champs de longueur fixée à un maximum raisonnable sont utilisés pour simplifier la définition de structure et l'analyse syntaxique.
2. Un champ 'opcode' (code opératoire) existe et comprend deux valeurs. Le client diffuse un paquet 'bootrequest' (requête d'amorçage). Le serveur répond ensuite avec un paquer 'bootreply' (réponse d'amorçage). La requête d'amorçage contient l'adresse matérielle du client et son adresse IP, s'il la connaît.
3. La requête peut optionnellement contenir le nom du serveur à partir duquel le client veut obtenir une réponse. C'est prévu afin que le client puisse exiger l'amorçage depuis un hôte spécifique (p.ex. s'il existe de multiples versions du même fichier de démarrage ou si le serveur est situé dans un réseau/domaine éloigné. Le client ne doit pas s'occuper de services d'annuaire/de noms de domaine ; au lieu de cela, la fonction est placée sur le serveur BOOTP.
4. La requête peut optionnellement contenir le nom de fichier « générique » à partir duquel démarrer. Par exemple, 'unix' ou 'ethertip'. Quand le serveur envoie la réponse d'amorçage, il

[1] placée en mémoire cache

remplace la valeur de ce champ par le nom de chemin complètement qualifié du fichier de démarrage approprié. Pour déterminer ce nom, le serveur peut consulter sa propre base de données mettant en corrélation adresse et requête de nom de fichier du client avec un fichier de démarrage particulier personnalisé pour ce client particulier. Si le nom de fichier de la requête d'amorçage est une chaîne de caractères vide, alors le serveur renvoie un champ nom de fichier indiquant que le fichier « par défaut » doit être chargé pour ce client.

5. Dans le cas de clients qui ne connaissent pas leur adresse IP, le serveur doit également disposer d'une base de données associant adresses matérielles et adresses IP. L'adresse IP du client est ensuite placée dans un champ de la requête d'amorçage.
6. Certaines topologies de réseau (comme celle de Stanford) peuvent être telles qu'il n'existe pas de serveur TFTP directement raccordé sur un câble physique donné (p.ex. quand tous les hôtes et passerelles présents sur un câble ne possèdent pas de disque dur). En coopérant avec des passerelles voisines, BOOTP peut permettre aux clients de démarrer depuis des serveurs situés à plusieurs sauts, au travers de ces passerelles. Voyez la section Amorçage au travers de Passerelles plus bas. Cette partie du protocole ne requiert aucune action spéciale de la part du client. L'implémentation est optionnelle et requiert une petite quantité de code supplémentaire dans les passerelles et serveurs.

3 Format des Paquets

Tous les nombres présentés sont décimaux, à moins d'une indication contraire. Le paquet BOOTP est encapsulé dans un datagramme UDP [7] IP [8] standard. Pour simplifier, on suppose que le paquet BOOTP n'est jamais fragmenté. Tous les champs numériques présentés dans l'« ordre des octets standard », c.-à-d. que les bits d'ordre supérieur (de poids fort) sont envoyés en premier.

Dans l'en-tête IP d'une requête d'amorçage, le client fournit sa propre adresse IP source s'il la connaît, ou zéro sinon. Quand l'adresse du serveur est inconnue, l'adresse IP destination sera l'adresse de diffusion 255.255.255.255 (broadcast). Cette adresse signifie « diffuser sur le câble local (je ne connais pas mon adresse réseau) » [4].

L'en-tête UDP contient les numéros de port source et destination. Le protocole BOOTP utilise deux numéros de ports réservés, « client BOOTP » (n° de port 68) et « serveur BOOTP » (n° de port 67). Le client envoie ses requêtes en utilisant « serveur BOOTP » comme port de destination, habituellement à l'adresse de diffusion. Le serveur envoie ses réponses en utilisant « client BOOTP » comme port de destination ; en fonction des facilités offertes par le noyau ou le pilote du serveur, cela peut être envoyé ou non à l'adresse de diffusion (voir les explication plus loin dans la section intitulée « Problème de l'Oeuf et de la Poule » plus bas). La raison pour laquelle *deux* ports réservés sont utilisés, est que l'on veut éviter de « réveiller » et de programmer les démons serveurs BOOTP, quand une réponse d'amorçage doit être diffusée vers un client. Puisque le serveur et les autres hôtes n'écouteront pas le port « client BOOTP », de telles diffusions entrantes seront filtrées au niveau du noyau. Nous ne pourrions pas simplement permettre au client de choisir un numéro de port « aléatoire » pour le champ de port source UDP, puisque étant donné que la réponse du serveur peut être diffusée, un numéro de port choisi aléatoirement pourrait troubler les autres hôtes qui sont justement en train d'écouter sur ce port.

Le champ de longueur UDP est fixé à la somme des longueurs des parties UDP et BOOTP du paquet. Le champ de total de contrôle UDP peut être fixé à zéro par le client (ou le serveur) s'il le

souhaite, pour éviter cette surcharge supplémentaire dans une implémentation PROM. Dans la section « Traitement des Paquets » plus bas, l'expression « [Total de contrôle UDP] » est utilisée à chaque fois que le total de contrôle peut être vérifié ou calculé.

CHAMP	OCTETS	DESCRIPTION
op	1	Type de message/code opératoire du paquet 1 = BOOTREQUEST (requête d'amorçage) 2 = BOOTREPLY (réponse d'amorçage)
htype	1	Type d'adresse matérielle Voyez la section ARP dans le RFC "Assigned Numbers" « 1 » = 10mb ethernet
hlen	1	Longueur de l'adresse matérielle (p.ex. 6 pour ethernet 10 Mbps).
hops	1	Fixé par le client à zéro ; peut être utilisé par des passerelles lors de d'amorçages au travers de passerelles.
xid	4	ID de transaction : nombre aléatoire utilisé pour associer cette requête de démarrage avec la réponse qu'elle génère.
secs	2	Rempli par le client, secondes écoulées depuis le début de sa tentative d'amorçage. Inutilisé
ciaddr	4	Adresse IP du client ; remplie par le client dans la requête d'amorçage si elle est connue
yiaddr	4	« Votre » adresse IP (client) ; remplie par le serveur si le client ne connaît pas sa propre adresse (si ciaddr valait 0).
siaddr	4	Adresse IP du serveur ; renvoyée dans réponse d'amorçage par le serveur
giaddr	4	Adresse IP de la passerelle ; utilisée dans l'amorçage au travers de passerelles optionnel.
chaddr	16	Adresse matérielle du client ; remplie par le client.
sname	64	Nom de l'hôte serveur, chaîne de caractères terminée par un caractère NUL.
file	128	Nom du fichier de démarrage, chaîne de caractères terminée par un caractère NUL ; nom « générique » ou null dans la requête d'amorçage, nom de chemin de répertoire complètement qualifié dans la réponse d'amorçage.
vend	64	zone optionnelle spécifique au vendeur pourrait p.ex. être le type/numéro de série du matériel dans la requête, ou une « capacité »/handle [2] du système de fichiers distant lors de la réponse. Cette information peut être mise de côté pour utilisation lors d'une troisième phase d'amorçage ou par le noyau.

4 Problème de l'Œuf et de la Poule

Comment le serveur peut-il envoyer un datagramme IP au client, si celui-ci ne connaît pas (encore) sa propre adresse IP ? À chaque fois qu'une réponse d'amorçage est envoyée, la machine émettrice effectue les opérations suivantes :

1. Si le client connaît sa propre adresse IP (le champ « ciaddr » est différent de zéro), alors le paquet ne peut être envoyé comme un paquet normal, puisque le client répondra aux requêtes ARP [5] ;
2. Si le client ne connaît pas encore sa propre adresse IP (ciaddr nul), alors le client ne peut répondre aux requêtes ARP transmises lors de la réponse d'amorçage. Il y a deux options :

[2] NdT : difficilement traduisible : manipulateur, gestionnaire

(a) Si le transmetteur dispose des points de raccordement (hooks) à un pilote ou au noyau nécessaires pour construire « manuellement » une entrée dans le cache d'adresses ARP, alors il peut remplir une entrée en utilisant les champs ' chaddr ' et ' yiaddr '. Bien sûr, cette entrée devrait faire l'objet d'une temporisation, comme toute autre entrée créée par le code ARP normal lui-même. Le transmetteur de la réponse d'amorçage peut ensuite envoyer simplement la réponse d'amorçage à l'adresse IP du client. UNIX (BSD 4.2) dispose de cette capacité.

(b) Si le transmetteur ne dispose pas de ces points de raccordement au noyau, il peut simplement envoyer la réponse d'amorçage à l'adresse IP de diffusion sur l'interface appropriée. Cela ne représente qu'une seule diffusion supplémentaire par rapport au cas précédent.

5 Utilisation d'ARP par le Client

La PROM du client doit contenir une implémentation simple d'ARP, le cache d'adresses pouvant p.ex. ne disposer que d'une seule entrée. Cela permettra un amorçage utilisant uniquement la seconde phase (TFTP) quand le client connaît les adresse IP et fichier de démarrage.

À chaque fois que le client s'attend à recevoir une réponse TFTP ou BOOTP, il devrait être préparé à répondre à une requête ARP pour sa propre association adresse IP/adresse matérielle (si elle est connue).

Puisque la réponse d'amorçage contiendra (dans l'encapsulation matérielle) l'adresse source matérielle du serveur/de la passerelle, le client PEUT éviter d'envoyer une requête ARP pour obtenir l'adresse IP du serveur/passerelle à utiliser dans la phase TFTP suivante. Néanmoins, cela ne devrait être traité que comme un cas spécial, car il est désirable de toujours permettre un amorçage utilisant uniquement la seconde phase comme décrit plus haut.

6 Comparaison avec RARP

Un protocole plus ancien, le *Reverse Address Resolution Protocol* (protocole de résolution d'adresse inverse, RARP) [1], a été proposé pour permettre à un client de déterminer son adresse IP, connaissant son adresse matérielle. Néanmoins, RARP avait l'inconvénient d'être un protocole de la couche liaison de données (non basé sur IP/UDP). Cela signifie que RARP ne pouvait être implémenté que sur les hôtes ayant subi des modifications spéciales au niveau noyau ou pilote pour accéder aux paquets « bruts ». Puisqu'il existe beaucoup de noyaux réseaux actuellement, pour lesquels chaque source est maintenue par des organisations différentes, un protocole d'amorçage qui ne requiert pas de modifications au noyau jouit d'un avantage décisif.

BOOTP fournit cette fonction de recherche adresse IP vers adresse matérielle, en plus des autres fonctionnalités utiles décrites dans les sections présentées plus haut.

7 Traitement du Paquet

7.1 Transmission par le Client

Avant de créer le paquet pour la première fois, c'est une bonne idée de nettoyer entièrement le tampon de paquets en le remplissant avec des zéros ; cela placera tous les champs dans leur état par défaut. Le client crée ensuite un paquet avec les champs suivants.

L'adresse IP destination est fixée à 255.255.255.255 (l'adresse de diffusion) ou à l'adresse IP du serveur (si elle est connue). L'adresse IP source et 'ciaddr' sont fixés à l'adresse IP du client si elle est connue, ou à 0 sinon. L'en-tête UDP est défini avec la longueur adéquate ; port source = 'client BOOTP', port destination = 'serveur BOOTP'.

'op' est fixé à '1', c.-à-d. BOOTREQUEST. 'htype' est fixé au type d'adresse matérielle comme affecté dans la section ARP du RFC "Assigned Numbers". 'hlen' est fixé à la longueur de l'adresse matérielle, p.ex. '6' pour ethernet 10 Mbps.

'xid' est fixé à un id de transaction « aléatoire ». 'secs' est fixé au nombre de secondes qui se sont écoulées depuis que le client à commencé à démarrer. Cela permettra aux serveurs de savoir depuis combien de temps un client essaie de s'amorcer. Quand le nombre devient grand, certains serveurs peuvent devenir plus « compatissants » envers un client qu'ils ne servent normalement pas. Si un client n'a pas d'horloge adéquate, il pourrait construire une estimation grossière en utilisant un temporisateur sous forme de boucle. Ou bien il pourrait choisir de simplement envoyer ce champ avec toujours la même valeur fixée, disons 100 secondes par exemple.

Si le client connaît son adresse IP, 'ciaddr' (et l'adresse IP source) sont fixés à cette valeur. 'chaddr' est rempli par l'adresse matérielle du client.

Si le client veut restreindre l'amorçage à un nom de serveur particulier, il peut placer une chaîne de caractère terminée par un caractère NUL dans 'sname'. Le nom utilisé devrait être n'importe lequel des noms ou surnoms permis pour l'hôte désiré.

Le client a plusieurs possibilités pour remplir le nom de champ 'file' (fichier). S'il est laissé vide, la signification est « Je veux démarrer à partir du fichier par défaut pour ma machine ». Un nom de fichier vide peut également signifier « Je ne m'intéresse qu'à la découverte des adresses IP des client/serveur/passerelle, je ne me soucie pas des noms de fichiers ».

Le champ peut également être un nom « générique » comme 'unix' ou 'gateway' (passerelle) ; cela signifie « amorçe le programme nommé configuré pour ma machine ». Finalement, le champ peut être un nom de chemin de répertoire complètement qualifié.

Le champ 'vend' peut être rempli par le client avec des chaînes ou structures spécifiques au vendeur. Le type/numéro de série du matériel peut par exemple y être placé. Néanmoins, le fonctionnement du serveur BOOTP *ne devrait pas dépendre* de l'existence de cette information.

Si le champ 'vend' est utilisé, il est recommandé d'y placer un « nombre magique » de 4 octets comme premier élément. Cela permet à un serveur de déterminer quel type d'information il voit dans ce champ. Les nombres peuvent être affectés en utilisant le processus de « nombre magique » habituel – vous en prenez un et il est magique. Des nombres magiques différents pourraient être utilisés lors des réponses d'amorçage et lors des requêtes d'amorçage pour permettre au client d'entreprendre des actions spéciales avec l'information de réponse.

[Total de contrôle UDP]

7.2 Stratégie de Retransmission du Client

Si aucune réponse n'est reçue durant une certaine période de temps, le client devrait retransmettre la requête. L'intervalle de temps doit être choisi prudemment afin de ne pas inonder le réseau. Réfléchissez au cas d'un câble supportant 100 machines qui viennent de redémarrer après une coupure de courant. Le simple fait de retransmettre la requête toutes les quatre secondes inondera le réseau.

Comme stratégie possible, vous pourriez envisager la méthode appelée « exponentiel binaire » [3], semblable à celle utilisée par ethernet lors des collisions [4]. Ainsi, par exemple, si le premier paquet arrivait à l'heure 0:00, le deuxième serait là à :04, ensuite :08, :16, :32 et enfin :64. Vous devriez également rendre chaque temps aléatoire ; cela se ferait de façon similaire à la spécification ethernet en démarrant avec un masque de bits et en l'associant via un ET binaire avec un nombre aléatoire pour obtenir la première transmission. À chaque transmission réussie, le masque est incrémenté d'une longueur de 1 bit. Cela double le délai moyen entre chaque transmission.

Après que le délai de transmission « moyen » ait atteint environ 60 secondes, il ne devrait plus être augmenté, mais toujours être aléatoire.

Avant chaque retransmission, le client devrait mettre à jour le champ 'secs'. [Total de contrôle UDP]

7.3 Le Serveur Reçoit BOOTREQUEST

[Total de contrôle UDP]

- Si le port UDP destination be correspond pas au port 'serveur BOOTP', jeter le paquet.
- Si le champ de nom du serveur (sname) est vide (aucun serveur particulier spécifié), ou si 'sname' est spécifié et correspond à notre nom ou surnom, alors continuer le traitement du paquet.
- Si le champ sname est spécifié, mais ne vaut pas « us », alors il y a plusieurs possibilités :
  - Vous pouvez choisir de simplement jeter ce paquet.
  - Si un recherche de nom sur sname montre qu'il est sur ce même câble, jeter le paquet.
  - Si sname est situé dans un réseau différent, vous pouvez choisir de faire suivre le paquet à cette adresse. Si c'est le cas, vérifiez le champ 'giaddr' (adresse de passerelle). Si 'giaddr' est nul, remplissez-le avec mon adresse ou avec l'adresse d'une passerelle qui peut être utilisée pour aller dans ce réseau. Ensuite, faites suivre le paquet.
- Si l'adresse IP du client (ciaddr) est zéro, alors le client ne connaît pas sa propre adresse IP. Essayez de rechercher l'adresse matérielle du client (chaddr, hlen, htype) dans notre base de données. Si aucune correspondance n'est trouvée, jetez le paquet.
- Sinon, nous avons maintenant une adresse IP pour ce client ; utilisez-la pour remplir le champ 'yiaddr' (votre adresse IP).

Nous vérifions maintenant le champ de nom du fichier de démarrage (file). Le champ sera vide si le client ne s'intéresse pas aux noms de fichiers, ou désire utiliser le fichier de démarrage par défaut. Si le champ n'est pas vide, il est utilisé en tant que clé de recherche dans une base de données, comme l'est l'adresse IP du client. S'il y a un fichier par défaut ou un fichier générique (éventuellement indexé par l'adresse du client) ou un nom de chemin complètement spécifié qui correspond, alors remplacez le champ 'file' par le nom de chemin complètement spécifié du fichier de démarrage sélectionné. Si

[3] exponential backoff

[4] attente entre 0 et i -1 unités de temps pour émettre après la $i^{ème}$ collision

le champ n'est pas vide et qu'aucune correspondance n'est trouvée, alors le client demande un fichier que nous n'avons pas ; jetez le paquet, quelque autre serveur BOOTP pourrait en disposer.

Le champ de données spécifique au vendeur 'vend' devrait à présent être vérifié et, si un type de données reconnu est spécifié, des actions spécifiques au client devraient être entreprises, et une réponse devrait être placée dans le champ de données 'vend' du paquet réponse. Par exemple, une station de travail cliente pourrait fournir une clé d'authentification et recevoir du serveur un moyen d'accès à des fichiers distants, ou un jeu d'options de configuration, qui peuvent être passées au système d'exploitation qui va être lancé sous peu.

Placez mon adresse IP (serveur) dans le champ 'siaddr'. Fixez le champ 'op' à BOOTREPLY. Le port UDP destination est fixé à 'client BOOTP'. Si l'adresse du client ('ciaddr') est non nulle, envoyez-y le paquet ; sinon, si l'adresse de passerelle 'giaddr' est non nulle, fixez le port UDP destination à 'serveur BOOTP' et envoyez le paquet à 'giaddr' ; sinon, le client est présent sur l'un de nos câbles mais ne connaît pas encore sa propre adresse IP – utilisez l'une des méthodes décrites dans la section « Problème de l'Oeuf et de la Poule » plus haut pour l'envoyer au client. Si vous le faites et qu'il y a de multiples interfaces sur cet hôte, utilisez le champ 'yiaddr' (votre adresse IP) pour trouver sur quel réseau (câble/interface) envoyer le paquet. [Total de contrôle UDP]

7.4 Le Serveur/Passerelle Reçoit BOOTREPLY

[Total de contrôle UDP] Si 'yiaddr' (votre adresse IP [celle du client]) se réfère à l'un de nos câbles, utilisez l'une des méthodes décrite dans la section 4 plus haut pour le propager vers le client. Assurez-vous de l'envoyer vers le port UDP destination 'client BOOTP'

7.5 Réception par le Client

N'oubliez pas de traiter les requêtes ARP pour ma propre adresse IP (si je la connais). [Total de contrôle UDP] Le client devrait éliminer les paquets qui

- ne sont pas adressés (IP/UDP) au port d'amorçage
- ne sont pas des réponses d'amorçage
- ne correspondent pas à mon adresse IP (si je la connais) ou mon adresse matérielle
- ne correspondent pas à mon id de transaction.

Autrement, nous avons reçu une réponse couronnée de succès. 'yiaddr' contiendra mon adresse IP, si je ne la connaissais pas avant. 'file' est le nom de fichier de la 'read request' (requête de lecture) TFTP. L'adresse du serveur est dans 'siaddr'. Si 'giaddr' (adresse de passerelle) est non nulle, alors les paquets devraient y être acheminés en premier lieu, afin d'arriver au serveur.

8 Amorçage au travers de Passerelles

Cette partie du protocole est optionnelle et requiert du code supplémentaire pour la coopération entre passerelles et serveurs, mais il permet un amorçage par l'intermédiaire de passerelles. C'est principalement utile quand les passerelles sont des machines sans disque dur. Les passerelles contenant des disques dur (p.ex. une machine UNIX faisant office de passerelle), pourraient également faire tourner leurs propres serveurs BOOTP/TFTP.

Les passerelles écoutant les requêtes d'amorçage diffusées peuvent décider de rediriger ou de rediffuser ces requêtes « lorsque cela est approprié ». Par exemple, la passerelle pourrait disposer, dans ses tables de configuration, d'une liste d'autres réseaux ou hôtes devant recevoir une copie de toute requête d'amorçage diffusée. Même si un champ 'hops' existe, c'est une mauvaise idée de rediffuser globalement les requêtes, car il y a de grandes chances que des boucles dues à la diffusion se produisent.

La propagation pourrait commencer immédiatement, ou attendre que le champ 'secs' (secondes écoulées depuis que le client essaye de démarrer) dépasse un certain seuil.

Si une passerelle décide de propager la requête, elle devrait examiner la valeur du champ 'giaddr' (adresse IP de la passerelle). Si elle est nulle, elle devrait placer sa propre adresse IP (sur le câble de réception) dans ce champ. Elle peut également utiliser le champ 'hops' pour contrôler optionnellement à quelle distance le paquet est propagé. Hops devrait être incrémenté lors de chaque redirection. Par exemple, si hops dépasse '3', le paquet devrait probablement être éliminé. [Total de contrôle UDP]

Nous recommandons de placer cette fonction de redirection spéciale dans les passerelles. Mais cela ne doit pas forcément être le cas. Pour autant qu'existe un quelconque « agent de redirection BOOTP » sur le réseau comprenant le client en phase d'amorçage, l'agent peut effectuer la redirection lorsque cela est approprié. Ainsi, ce service peut être ou pas situé sur la passerelle.

Au cas où un agent de redirection n'est pas situé dans la passerelle, l'agent pourrait s'épargner un peu de peine en plaçant l'adresse de diffusion de l'interface recevant la requête d'amorçage dans le champ 'giaddr'. Ainsi, la requête serait redirigée en utilisant des passerelles normales, n'impliquant pas d'agent de redirection. Bien sûr, le désavantage de cette situation est que vous perdez la possibilité d'utiliser la méthode non-diffusée[5] de renvoi de la réponse, causant une surcharge supplémentaire pour chaque hôte sur le câble du client.

9 Exemple de Base de Données d'un Serveur BOOTP

À des fins de suggestion, nous présentons ici un exemple de base de données constituée d'un fichier texte que le programme serveur BOOTP pourrait utiliser. La base de données possède deux sections, délimitées par une ligne contenant un signe % en colonne 1. La première section contient un « répertoire par défaut » et des associations entre noms génériques et répertoires/noms de chemin. Le premier nom générique dans cette section est le « fichier par défaut » que vous obtenez quand la requête d'amorçage contient un champ 'file' vide.

La seconde section « convertit » une adresse/un type d'adresse matérielle en adresse IP. Vous pouvez optionnellement surcharger le nom générique par défaut en fournissant un nom générique spécifique à une adresse IP. Un élément 'suffixe' est également optionnel ; s'il est fourni, tout nom générique spécifié par le client sera accédé en suffixant au préalable 'suffixe' au 'nomDeChemin' approprié à ce nom générique. Si ce fichier n'est pas trouvé, alors on cherchera le simple 'nomDeChemin'. Cette option 'suffixe' permet d'effectuer un tas de réglages génériques personnels en un minimum d'efforts. Ci-dessous, le format général est présenté ; les champs sont délimités par un ou plusieurs espaces ou tabulations ; les champs vides de queue peuvent être omis ; les lignes blanches et celles commençant par '#' sont ignorées.

[5] cfr section 4

```
# ligne de commentaire

répertoirePersonnel
nomGénérique1    nomDeChemin1
nomGénérique2    nomDeChemin2
...

% fin des noms génériques, début des conversions d'adresses

nomHôte1 typeMatériel adresseMatérielle1 adresseIP1 nomGénérique suffixe
nomHôte2 typeMatériel adresseMatérielle2 adresseIP2 nomGénérique suffixe
...
```

Voici un exemple spécifique. Notez que le nombre 'hardwaretype' est le même que celui fourni dans la section ARP du RFC 'Assigned Numbers'. Les nombres 'hardwaretype' et 'ipaddr' sont spécifiés en décimal ; 'hardwareaddr' est en hexadécimal.

```
# dernière mise à jour par smith

/usr/boot
vmunix          vmunix
tip             ethertip
watch           /usr/diag/etherwatch
gate            gate.

% fin des noms génériques, début des conversions d'adresses

hamilton        1 02.60.8c.06.34.98     36.19.0.5
burr            1 02.60.8c.34.11.78     36.44.0.12
101-gateway     1 02.60.8c.23.ab.35     36.44.0.32      gate 101
mjh-gateway     1 02.60.8c.12.32.bc     36.42.0.64      gate mjh
welch-tipa      1 02.60.8c.22.65.32     36.47.0.14      tip
welch-tipb      1 02.60.8c.12.15.c8     36.46.0.12      tip
```

Dans l'exemple ci-dessus, si 'mjh-gateway' effectue un amorçage par défaut, il ira chercher le fichier '/usr/boot/gate.mjh'.

Remerciements

Ross Finlayson (*et. al.*) a produit deux RFC plus anciens discutant de l'amorçage via TFTP [2] en utilisant RARP [1].

Nous voudrions également remercier Noel Chiappa, Bob Lyon, Jeff Mogul, Mark Lewis et David Plummer pour les travaux précédents et commentaires qu'ils nous ont fournis.

Bibliographie

[1] Ross Finlayson, Timothy Mann, Jeffrey Mogul, Marvin Theimer, *A Reverse Address Resolution Protocol*, RFC 903, NIC, Juin 1984.

[2] Ross Finlayson, *Bootstrap Loading using TFTP*, RFC 906, NIC, juin 1984.

[3] Mark Lottor, *Simple File Transfer Protocol*, RFC 913, NIC, septembre 1984.

[4] Jeffrey Mogul, *Broadcasting Internet Packets*, RFC 919, NIC, octobre 1984.

[5] David Plummer, *An Ethernet Address Resolution Protocol*, RFC 826, NIC, septembre 1982.

[6] Jon Postel, *File Transfer Protocol*, RFC 765, NIC, juin 1980.

[7] Jon Postel, *User Datagram Protocol*, RFC 768, NIC, août 1980.

[8] Jon Postel, *Internet Protocol*, RFC 791, NIC, septembre 1981.

[9] K. R. Sollins, Noel Chiappa, *The TFTP Protocol*, RFC 783, NIC, juin 1981.