Groupe de travail Réseau
Request for Comments : 3074
Catéorie : En cours de normalisation

Traduction Claude Brière de L'Isle

B. Volz, Ericsson
S. Gonczi, Network Engines, Inc.
T. Lemon, Internet Engines, Inc.
R. Stevens, Join Systems, Inc.
février 2001

Algorithme DHC d'équilibrage de charge

Résumé
Le présent document propose une méthode d'équilibrage de charge algorithmique. Elle permet à plusieurs serveurs coopérants de décider duquel devrait desservir un client, sans échanger d'informations au delà de la configuration initiale.

Le choix du serveur se fonde sur les adresses de contrôle d'accès au support (MAC, *Media Access Control*) de client de hachage du serveur lorsque plusieurs serveur du protocole de configuration dynamique d'hôte (DHCP, *Dynamic Host Configuration Protocol*) sont disponibles pour desservir les clients DHCP. La technique proposée permet un choix de serveur efficace lorsque plusieurs serveurs DHCP offrent leurs services sur un réseau, sans exiger aucun changement aux clients DHCP existants. La même méthode est proposée pour choisir le serveur cible d'un agent émetteur, tel qu'un relais du protocole d'amorçage (BOOTP, *Bootstrap Protocol*).

1. Introduction

Le présent protocole était à l'origine conçu pour prendre en charge l'optimisation d'un équilibrage de charge spécifique du protocole de reprise sur défaillance DHCP [FAILOVR]. Les auteurs ont ultérieurement réalisé qu'il pourrait être utilisé pour optimiser le comportement de serveurs DHCP coopérants et d'agents de relais BOOTP qui leur transmettent les paquets. La proposition rend possible de régler chaque serveur participant à accepter un pourcentage préconfiguré (approximatif) de la charge du client. Cela se fait par l'utilisation d'un algorithme de hachage déterministe qui pourrait être facilement appliqué aux autres protocoles qui ont des caractéristiques similaires.

2. Terminologie

La présente section discute aussi bien la terminologie des exigences génériques communes à de nombreuses spécifications de protocoles de l'IETF que de la terminologie introduite par le présent document.

2.1 Terminologie des exigences

Les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDE", "PEUT" et "FACULTATIF" dans ce document sont à interpréter comme décrit dans la [RFC2119].

2.2 Terminologie de l'équilibrage de charge

Le présent document introduit les termes suivants :

Délai de service (SD, *Service Delay*)
Paramètre d'équilibrage de charge, permettant un service différé d'un client par un serveur qui participe au schéma d'équilibrage de charge, au lieu d'ignorer le client.

Allocations de baquet de hachage (HBA, *Hash Bucket Assignments*)
Directive de configuration qui alloue un ensemble de valeurs de baquet de hachage à un serveur qui participe au schéma d'équilibrage de charge.

Identifiant de serveur (SID, *Server ID*)
Identifiant qui peut être utilisé pour désigner un des serveurs participants. Dans le contexte de DHCP, le SID est l'adresse IP ou le nom DNS du serveur.

Transaction de service (ST, *Service Transaction*)
Ensemble d'échanges client-serveur qui conduisent à ce qu'un serveur fournisse ou refuse un service à un client. Par exemple, l'échange de message DISCOVER/OFFER/REQUEST/ACK entre un serveur et un client DHCP est une transaction de service.

Identifiant de transaction de service (STID, *Service Transaction ID*)
Attribut des demandes des clients individuels utilisé pour l'équilibrage de charge.

3. Fondements et exigences externes

Comme les clients DHCP utilisent les diffusions UDP pour contacter les serveurs DHCP, un message DHCPDISCOVER de client peut être reçu par plus d'un serveur. Tous les serveurs qui reçoivent une telle diffusion peuvent répondre au client, le laissant choisir quel serveur il veut utiliser.

Lorsque un agent de relais BOOTP est utilisé, il transmet ou rediffuse normalement les diffusions du client à tous les serveurs configurés, de sorte qu'une inefficacité similaire est présente.

L'optimisation décrite permet à un serveur d'être choisi pour chacune de ces transactions en effectuant un calcul "servir" / "ne pas servir". Un agent de transmission peut effectuer le même calcul pour choisir une destination de transmission.

Dans l'un ou l'autre cas, le choix du serveur peut être calculé, sans que les participants aient à négocier qui doit répondre.

L'approche est probabiliste par nature, parce qu'il est presque impossible de prévoir quel client va être le prochain à demander le service. Pour de courtes périodes, le pourcentage réel de clients servis par un serveur donné va probablement dévier du pourcentage désiré. Lorsque le nombre de demandes augmente, le pourcentage réel de la charge traitée par chaque serveur va s'approcher du pourcentage configuré.

4. Vue d'ensemble

Les serveurs DHCP DOIVENT utiliser l'option Identifiant de client comme STID si elle est présente. Si aucune option Identifiant de client n'est présente, le champ hlen du paquet DHCP DOIT être utilisé comme longueur des données à hacher, et le contenu du champ chaddr DOIT être les données à hacher. Au plus, les seize premiers octets de l'identifiant de client ou du champ chaddr sont utilisés.

La proposition transpose le STID en une valeur de hachage en utilisant la fonction de la section 6. La valeur de hachage résultante peut alors être utilisée pour décider qui devrait répondre à la demande, ou qui devrait être la cible de la transmission.

La fonction de hachage fournie génère des valeurs de hachage de 0 à 255, et donne une distribution de baquet de hachage très paire pour des STID aléatoires, et aussi pour les séquences de STID qui ont un schéma. L'allocation de ressources est réalisée par l'allocation d'un ensemble de valeurs de hachage spécifiques pour chaque serveur participant.

Un serveur va seulement servir une demande si le hachage du STID de la demande correspond à une des valeurs de hachage qui lui est allouée.

Tout baquet de hachage non alloué aux serveurs va résulter en des ST de client entièrement ignorées. (Dans certains scénarios, ce peut être un résultat souhaitable.) Il n'est pas nécessaire que les STID soient uniques, mais ils devraient avoir une variété suffisante pour répartir la charge entre chaque serveur.

Les HBA PEUVENT être transmises comme des messages, encapsulées dans des messages d'autres protocoles, par exemple, de messagerie électronique, ou d'option DHCP Protocole de reprise sur défaillance.

Les mises en œuvre de serveur DHCP peuvent facultativement être configurables pour traiter un cas où l'équilibrage de charge est effectué mais où le serveur qui est supposé répondre n'est pas disponible, ou est hors des adresses convenables.

Les mises en œuvre de serveur DHCP qui fournissent cette capacité DEVRAIENT régler le paramètre de configuration Service différé (DS, *Delayed Service*) au nombre de secondes à attendre après que la première demande du client a été envoyée avant de répondre à un client, lorsque le hachage ne permettrait normalement pas que le client soit servi.

Un serveur DHCP qui fournit cette capacité DEVRAIT utiliser la valeur qui est dans le champ secs de la demande du client si cette valeur est différente de zéro. Comme certains clients peuvent ne pas mettre correctement en œuvre le champ secs, un serveur DHCP PEUT garder trace de la première instance d'une transaction de client à laquelle il ne répondrait normalement pas. Si le serveur reçoit une demande d'un client qui a le même identifiant de transaction qu'une demande précédemment enregistrée, et si le champ secs dans le second paquet est zéro, le serveur DHCP PEUT utiliser le temps écoulé (en secondes) entre la première demande du client et la suivante, au lieu du champ secs.

5. Fonctionnement

5.1 Configuration

L'étape de configuration consiste à allouer des valeurs de hachage aux serveurs disponibles. Cela se fait en fournissant une ou plusieurs allocations de baquet de hachage (HBA, *Hash Bucket Assignment*). Elles peuvent venir d'un fichier de configuration, du registre Windows NT, de EEPROM, etc.. Autrement, des valeurs de baquet de jetons pourraient être allouées en utilisant un algorithme sur lequel on s'est mis d'accord. Par exemple, "toute valeur impaire est servie par le serveur A et toute valeur paire par le serveur B".

5.2 HBA destiné à un serveur

Lors de la configuration d'un serveur spécifique, une HBA de la forme d'une simple transposition de bits de valeurs de 32 octets DEVRAIT être utilisée.

Le premier octet de la transposition binaire de la HBA représente les valeurs 0 à 7 de la HBA, l'octet suivant, les valeurs de 8 à 15, et ainsi de suite, avec l'octet de trente secondes qui représente les valeurs de 248 à 255. Dans chaque octet, les bits de moindre poids de cet octet représentent la plus petite valeur de HBA dans cet octet.

Chaque bit de la HBA est associé à une valeur de hachage possible. Si un bit est établi dans la transposition, cela signifie que le serveur receveur DOIT servir chaque demande du client, où le STID donne la valeur de hachage correspondante.

Par exemple, si un serveur est configuré avec une HBA des 32 octets suivants :

FF FF FF FF FF FF 00 00 (0 - 63)
FF FF FF FF FF FF FF FF (64 - 127)
00 00 00 00 00 00 00 00 (128 - 191)
00 00 00 00 00 00 00 00 (192 - 255)

il DOIT alors servir toute demande de client dont le STID se hache dans les valeurs du baquet de 0 à 47 et 64 à 127.

5.3 Paramètre de service différé

Le paramètre Service différé est facultatif.

Si le paramètre n'est pas configuré, la HBA établit une politique Servir/Ne pas servir stricte.

Si le paramètre est configuré, le serveur qui n'est pas supposé servir une demande spécifique (sur la base de la HBA et du hachage du STID) est admis à répondre, après que S secondes se sont écoulées depuis la première tentative du client pour obtenir le service. Un serveur PEUT utiliser le champ secs dans l'en-tête BOOTP pour déterminer le temps écoulé depuis que le client a essayé d'obtenir le service, ou il PEUT garder trace des demandes répétées de toute autre façon.

5.4 HBA destiné à un émetteur

Lorsque on reconfigure un agent de transmission (par exemple, un relais BOOTP) les HBA qui consistent en paires d'identifiants de serveur / valeurs de baquet de hachage PEUVENT être utilisés.

Ici, l'identifiant de serveur (SID) désigne le serveur chargé du baquet de hachage spécifié. L'agent de transmission transmet chaque demande de client, où le STID donne la valeur de hachage spécifiée, au serveur désigné par le SID.

L'identifiant de serveur peut être tout attribut unique de serveur (par exemple, adresse IP, nom DNS, etc.) qui a une signification dans le contexte de l'opération d'agent de relais.

Un transmetteur peut être configuré pour transmettre un certain paquet à plus d'un serveur. Par exemple, un relais BOOTP pourrait être établi pour partager la charge entre deux paires de serveurs principal/de secours, chaque paire utilisant le protocole DHCP de reprise sur défaillance [FAILOVR]. Dans ce cas, un paquet qui est destiné à une paire de serveurs aura à transmettre aux deux serveurs, le principal et le secondaire, de la paire.

Un fichier de configuration possible pour un agent de transmission (par exemple, un relais BOOTP) peut ressembler à cela :

```
192.33.43.11 192.33.43.12: 0..24;
192.33.43.13:  25..55;
192.33.43.15:  56..128;
192.33.43.16: 129 130 131 200..202;
```

La configuration ci-dessus consiste en 4 HBA. La première HBA donnée en exemple dit : "Toute demande de client, où le STID donne une valeur de hachage de 0 à 24, sera transmise aux deux serveurs 192.33.43.11 et 192.33.43.12".

La quatrième HBA donnée en exemple déclare : "Toute demande de client, où le STID donne une valeur de hachage de 129, 139, 131, 200, 201 ou 202, sera transmise au serveur 192.33.43.16.

6. Fonction de hachage pour équilibrage de charge

La fonction de hachage suivante est une mise en œuvre en langage C de l'algorithme connu sous le nom de "hachage de Pearson". L'algorithme de hachage de Pearson a été à l'origine publié dans [PEARSON].

La fonction de hachage est peu coûteuse en calcul, elle exige une recherche dans une matrice et une opération OUx pour chaque octet clé. Pour faire fonctionner cette proposition, toutes les mises en œuvre interopérables DOIVENT utiliser cette fonction de hachage, avec l'ensemble de valeurs de tableau de mixage données ci-dessous :

```
/* Un "tableau de mixage" de 256 valeurs distinctes, en ordre pseudo-aléatoire. */

unsigned char  loadb_mx_tbl[256] ={
251, 175, 119, 215, 81, 14, 79, 191, 103, 49, 181, 143, 186, 157,  0,
232, 31, 32, 55, 60, 152, 58, 17, 237, 174, 70, 160, 144, 220, 90, 57,
223, 59,  3, 18, 140, 111, 166, 203, 196, 134, 243, 124, 95, 222, 179,
197, 65, 180, 48, 36, 15, 107, 46, 233, 130, 165, 30, 123, 161, 209, 23,
97, 16, 40, 91, 219, 61, 100, 10, 210, 109, 250, 127, 22, 138, 29, 108,
244, 67, 207,  9, 178, 204, 74, 98, 126, 249, 167, 116, 34, 77, 193,
200, 121,  5, 20, 113, 71, 35, 128, 13, 182, 94, 25, 226, 227, 199, 75,

27, 41, 245, 230, 224, 43, 225, 177, 26, 155, 150, 212, 142, 218, 115,
241, 73, 88, 105, 39, 114, 62, 255, 192, 201, 145, 214, 168, 158, 221,
148, 154, 122, 12, 84, 82, 163, 44, 139, 228, 236, 205, 242, 217, 11,
187, 146, 159, 64, 86, 239, 195, 42, 106, 198, 118, 112, 184, 172, 87,
2, 173, 117, 176, 229, 247, 253, 137, 185, 99, 164, 102, 147, 45, 66,
231, 52, 141, 211, 194, 206, 246, 238, 56, 110, 78, 248, 63, 240, 189,
93, 92, 51, 53, 183, 19, 171, 72, 50, 33, 104, 101, 69, 8, 252, 83, 120,
76, 135, 85, 54, 202, 125, 188, 213, 96, 235, 136, 208, 162, 129, 190,
132, 156, 38, 47, 1, 7, 254, 24, 4, 216, 131, 89, 21, 28, 133, 37, 153,
149, 80, 170, 68, 6, 169, 234, 151
};
```

```
unsigned char loadb_p_hash(
const unsigned char *key,                /* La clé à hacher */
const int len )                          /* Longueur de clé en octets  */
{
unsigned char hash  = len;
int i;

      for (i=len ; i > 0 ;  )
         hash = loadb_mx_tbl  [ hash ^ key[ --i ] ];

      return( hash );
}

int accept_service_request(
      const unsigned char HBA[32],       /* Transposition binaire du baquet de hachage */
      const unsigned char *key,          /* Identifiant de la transaction de service */
      const int len  )                   /* longueur de l'identifiant ci-dessus */
{
unsigned char hash = loadb_p_hash(key,len);
int index        = (hash >> 3) & 31;
int bitmask       = 1 << (hash & 7);

                                         /* retourne 1 si on doit servir cette transaction */
return((HBA[index] & bitmask) != 0);
}
```

7. Considérations pour la sécurité

Par elle-même, la présente proposition n'apporte aucune sécurité, ni n'a d'impact sur la sécurité existante. Les serveurs qui utilisent cet algorithme sont responsables de s'assurer que si le contenu de la HBA est transmis sur le réseau au titre du processus de configuration d'un serveur, ce message sera sécurisé contre l'altération, car l'altération de la HBA pourrait résulter en déni de service pour certains clients ou tous.

8. Références

[FAILOVR] Kinnear, K,, Droms, R., Rabil, G., Dooley, M., Kapur, A., Gonczi, S. et B. Volz, "DHCP Failover Protocol", *(Non publiée)*

[PEARSON] The Communications of the ACM Vol.33, n° 6 (juin 1990), pp. 677-680.

[RFC2131] R. Droms, "Protocole de configuration dynamique d'hôte", mars 1997. *(Mise à jour par les* RFC *3396 et* 4361*)*

[RFC2132] S. Alexander et R. Droms, "Options DHCP et Extensions de fabricant BOOTP", mars 1997.

9. Remerciements

Des remerciements particuliers à Peter K. Pearson, l'auteur du hachage de Pearson qui nous a gentiment accordé la permission d'utiliser son algorithme, en toute liberté.

La présente proposition découle de l'idée originale de hachage des adresses MAC en un seul bit par Ted Lemon, durant une discussion du a protocole de reprise sur défaillance tenue à CISCO Systems en février 1999. Rob Stevens a suggéré l'utilisation potentielle de cet algorithme pour des besoins qui vont au delà de ceux du protocole de reprise sur défaillance.

Un grand merci à Ralph Droms, Kim Kinnear, Mark Stapp, Glenn Waters, Greg Rabil et Jack Wong pour leure commentaires durant les discussions.

10. Adresse des auteurs

Bernie Volz
Ericsson
959 Concord Street
Framingham, MA 01701
téléphone : +1-617-513-9060
mél : bernie.volz@ericsson.com

Steve Gonczi
Network Engines, Inc.
25 Dan Road Canton,
MA 02021-2817
téléphone : 781-332-1165
mél : steve.gonczi@networkengines.com

Rob Stevens
Join Systems, Inc.
1032 Elwell Ct Ste 243
Palo Alto CA 94203
téléphone : (650)-968-4470
mél : robs@join.com

Ted Lemon
950 Charter Street
Redwood City, CA 94043
mél : ted.lemon@nominum.com

11. Déclaration complète de droits de reproduction

Remerciement

Le financement de la fonction d'édition des RFC est actuellement fourni par l'Internet Society.