Problèmes d'allocation dynamique d'adresses

Wassim Znaidi, Marco Fiore, Cédric Lauradoux, Marine Minier, Fabrice Valois

Problèmes d'allocation dynamique d'adresses

Wassim Znaidi[1] and Marco Fiore[1] and Cédric Lauradoux[1] and Marine Minier[1] and Fabrice Valois[1]

[1] *Université de Lyon, INRIA, INSA-Lyon, CITI laboratory*
mail: prénom.nom@insa-lyon.fr

L'allocation dynamique d'adresses est un problème important dans les réseaux ne disposant pas d'infrastructure centralisée. Cet article propose un modèle pour les algorithmes d'allocation d'adresses à état (*stateful*). A partir de ce modèle, nous étudions les caractéristiques des algorithmes PRIMEDHCP, QUADRATIC RESIDUE BASED DHCP et PROPHET. Nous montrons que PRIMEDHCP a un problème de dépassement de taille d'adresse, que QUADRATIC RESIDUE BASED DHCP est une version à deux sauts de DISTRIBUTED DHCP, et enfin que PROPHET n'est pas meilleur qu'une allocation aléatoire sans état (*stateless*) en terme de collision. La démarche de cet article est relativement originale puisque nous employons principalement des outils provenant de la cryptologie.

Keywords: Allocation dynamique d'adresses, algorithme d'allocation *avec état*, générateur pseudo-aléatoire.

1 Introduction

Les schémas d'allocation dynamique d'adresses se divisent en deux familles : les schémas dits *sans état* (Fig. 1) et ceux dits *avec état* (Fig. 2). Dans un algorithme sans état chaque nœud choisit lui-même sa propre adresse. Dans un algorithme avec état, un nœud est élu pour initier l'allocation d'adresses. Ce nœud racine reçoit une adresse choisie arbitrairement et fournit des adresses à tous les nœuds qui lui envoient une requête. En recevant une adresse, chaque nøeud devient alors capable d'attribuer des adresses à la demande. Dans la Fig. 2, le nœud racine reçoit l'adresse 1 et attribue ensuite les adresses 2, 3, 4, 5, 9. Le nœud d'adresse 9 attribue les adresses 10 et 13.

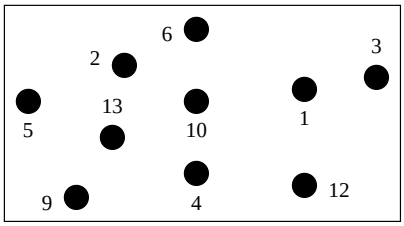

FIG. 1: Allocation sans état.

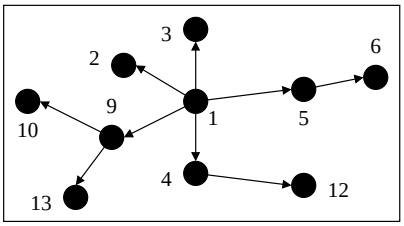

FIG. 2: Allocation avec état.

Dans l'esprit, les schémas *sans état* sont ceux qui correspondent le mieux à l'idée que l'on peut avoir de l'auto-organisation dans les réseaux. Chaque nœud choisit son adresse indépendamment des autres nœuds à l'aide d'un générateur pseudo-aléatoire. Néanmoins les algorithmes *sans état* ont une limite fondamentale qui est donné par le paradoxe des anniversaires. Considérons un réseau de n nœuds dans lequel l'espace des adresses est choisi entre 0 et $2^\ell - 1$. Alors la probabilité de collisions sur une adresse peut être approximé par : $\varepsilon \approx 1 - e^{\frac{-(n(n-1))}{2\times 2^\ell}}$. On a ainsi $\varepsilon > \frac{1}{2}$ quand $n = 2^{\ell/2}$. La relation précédente peut être transformée pour faire apparaitre le nombre maximal d'adresses choisit dans le réseau pour une probabilité ε donnée : $n_{max} \approx \sqrt{2 \times 2^\ell \times \ln \frac{1}{1-\varepsilon}}$. Si l'on considère des adresses de $\ell = 32$ bits comme dans IPv4 et que l'on fixe une probabilité de collision relativement faible[†] $\varepsilon = 2^{-20}$ (moins d'une chance sur un millions) alors un

[†] Cette notion est relativement subjective. En cryptographie $\varepsilon = 2^{-80}$ était en vogue pour la fonction SHA-1.

schéma sans état permet de gérer des réseaux de $n_{max} < 91$ (sic !). Pour avoir la même capacité que IPv4 ($n_{max} = 2^{32} - 2$), il faut avoir des adresses de 84 bits. Il est aussi possible d'affecter une allocation sans état sans garantir l'unicité des adresses ni même une borne supérieure sur la probabilité de collision. On utilise alors après l'allocation un algorithme de détection d'adresses dupliquées (DAD). Les algorithmes de DAD ont malheureusement un coût important en communication (voir [Vai02] par exemple).

De prime abord, les algorithmes avec état ont l'air peu attractif par rapport à la solution sans état. Il faut en effet élire un nœud racine, effectuer des requêtes et y répondre. Néanmoins, on peut attendre d'un algorithme avec état qu'il produise des adresses sans collision. On peut ainsi se passer des algorithmes de DAD. Parmi les algorithmes avec état qui ont été proposés, on peut citer DISTRIBUTED DHCP [MP02], PROPHET [ZNM03], PRIMEDHCP [HT05] et QUADRATIC RESIDUE BASED DHCP [CSX+08]. Cet article dresse un tableau relativement noir des algorithmes avec état : au final aucun n'apporter une réelle amélioration par rapport à la solution sans état (mis à part DISTRIBUTED DHCP). Pour le démontrer, nous employons principalement des outils de théorie des nombres et statistique qui proviennent de la cryptologie. Afin d'être concis, nous omettrons les références liées à cette discipline mais le lecteur pourra les trouver en consultant les ouvrages classiques [MOVR01, GG03].

Nous proposons dans la section suivante un modèle de machine à états finis qui représente le comportement d'un nœud dans la majorité des algorithmes avec état. Nous décrivons aussi les principaux algorithmes étudiés. Puis nous analysons en Section 3 PROPHET, PRIMEDHCP et QUADRATIC RESIDUE BASED DHCP. Enfin, nous concluons en louant les avantages de DISTRIBUTED DHCP.

2 Modèle pour l'allocation avec état

Le comportement d'un nœud lors d'une allocation avec état peut être décrit à l'aide d'une machine à états finis (Fig.3). Cette machine devient opérationnelle au moment où le nœud reçoit une adresse du réseau, *i.e.* l'état interne m_1 est initialisé. Cet état interne est modifié par une fonction de mise à jour g_s. La fonction f_p permet d'obtenir une adresse à partir de l'état interne ainsi que toutes les valeurs permettant d'initialiser la machine à état finis associée à la nouvelle adresse. Les fonctions g_s et f_p appartiennent respectivement aux familles de fonctions $\mathcal{G}$ et $\mathcal{H}$. Le choix particulier d'une fonction se fait à l'aide des paramètres s et p. La variable x définit l'initialisation de m_1. Ce modele peut facilement supporter des paramètres supplémentaires.

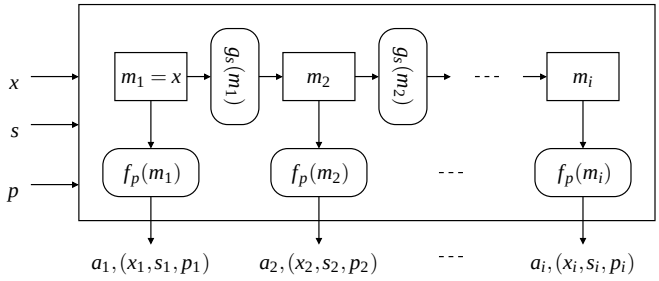

FIG. 3: Machine à état finis pour l'allocation avec état.

La Table 1 résume les fonctions de mise à jour et d'extraction d'adresses employées dans les algorithmes considérés dans cet article. Les détails relatifs à cette table seront données dans la section suivante.

3 Analyse

3.1 PrimeDHCP

PrimeDHCP [HT05] est fondée sur la décomposition unique d'un nombre en facteurs premiers. L'allocation se déroule ainsi : un nœud choisit une adresse arbitraire $a$et calcule la fonction lpf(a) le plus grand premier de la décomposition de a (lpf *largest prime factor*). L'état interne de la machine à états finis est

Algorithme	Fonction de mise à jour	Fonction d'extraction
DISTRIBUTED DHCP [MP02]	$g_s(m_t) = \begin{cases} \lfloor \frac{m_t}{2} \rfloor & \text{si } \lfloor \frac{m_t}{2} \rfloor \geq s, \\ m_t & \text{sinon.} \end{cases}$	$f_p(m_t) = \begin{cases} \lfloor \frac{m_t}{2} \rfloor + 1 & \text{si } \lfloor \frac{m_t}{2} \rfloor \geq p, \\ \text{rien} & \text{sinon.} \end{cases}$
PROPHET [ZNM03]	$g_s(m_t) = c_s + 1$	$f_p(m_t) = (p + h(m_t) \bmod q) + 1$
PRIMEDHCP [HT05]	$g_s(m_t) = \text{nextprime}(m_t)$	$f_p(m_t) = \text{lpf}(p) \times m_t$
QRB DHCP [CSX+08]	$g_s(m_t) = m_t^2 \bmod N$	$f_p(m_t) = m_t^2 \bmod N$

TAB. 1: Fonctions de mise a jour et d'extraction d'adresse.

initialisé avec $m_1 = \text{lpf}(a)$. La fonction de mise à jour consiste à remplacer m_t, $t \geq 1$ par le plus petit nombre premier supérieur à m_t (nextprime). On notera que cette fonction est commune à tous les nœuds. La génération d'une adresse consiste à multiplier l'état interne m_t avec l'adresse du nœud. Le défaut de cette méthode est que la taille de l'état interne comme celui des adresses augmentent très vite. Pour illustrer cette augmentation, nous décrivons le nombre d'adresses que peut allouer un nœud sans débordement, ainsi que la taille des adresses pour dans un reseau à i saut.

Taille des adresses à un saut – La taille des adresses croit très vite ce qui limite la capacité d'adressage d'un nœud. Faisons l'hypothèse que la racine a pour adresse 1. Le nombre d'adresses que peut allouer la racine est donnée par la fonction $\pi(n)$ de comptage des premiers strictement inferieurs à n. Le théorème de Chebyshev nous dit que : $\pi(n) = \Theta\left(\frac{n}{\ln n}\right)$. On peut directement en déduire combien de nœuds à un saut de la racine ne pourront pas allouer d'adresses :

$$\pi(2^\ell) - \pi(2^{\ell-1}) > \frac{2^\ell}{3 \cdot (\ell+1) \cdot \ln(2)} \text{ (Postulat de Bertrand).}$$

Taille des adresses à i sauts – On considère un nœud racine ayant pour adresse a. On sait par définition la propriété suivante pour toutes adresses a_i allouées à i sauts de la racine :

$$a_i \geq \text{lpf}(a)^i \times a.$$

La distance maximale d en nombre de sauts que l'on peut atteindre avec des adresses de ℓ bits est donc : $p^d \times a \leq 2^\ell$. On a alors : $d \leq \frac{\ell - \ln a}{\ln \text{lpf}(a)}$.

3.2 Quadratic Residue Based DHCP

Cet algorithme [CSX+08] utilise une seule fonction pour la mise a jour de l'état interne et pour l'extraction des adresses : $m_t^2 \bmod N$ avec $N = pq$, p et q étant deux grands nombres premiers. La théorie des résidus quadratiques permet en effet de construire t séquences de nombres disjoints sur $\mathbb{Z}/N\mathbb{Z}$. Cet algorithme n'est autre que celui proposé par Blum Blum Shub pour la génération de nombre aléatoire basée sur des fonctions à sens unique. QRB DHCP est en faites adapté au réseau où chaque nœud est au maximum à deux sauts du nœud racine. Le nœud racine va allouer les t séquences disjointes à son voisinage à un saut. Puis chacun de ces nœuds utilise la relation quadratique pour allouer des adresses. *L'utilisation de cet algorithme est donc relativement limitée à un type particulier de topologies.*

Notons au passage que les auteurs laissent en suspens la question du choix de p et q. Ce choix est important car il influence le nombre t de séquences et leur taille. Ils considèrent l'utilisation de nombre premier 1-sûr (*1-safe primes* : $p = 2p_1 + 1$, p_1 premier) ou 2-sûr (*2-safe primes* : $p = 2p_1 + 1$ avec $p_1 = 2p_2 + 1$ et p_1, p_2 premiers). Pour des nombres premiers p et q 1-sûr et avec 2 n'étant pas un générateur de $\mathbb{Z}/q_1\mathbb{Z}^*$, on sait que le nombre et la longueur des séquences dépends de la factorisation de $p_1 - 1$ et $q_1 - 1$: $t \approx 2 \times r \approx \sqrt{p_1 - 1} \approx \sqrt{\sqrt{N}/2}$, avec r le plus petit facteur de $\frac{p_1-1}{2}$. La longueur maximale de ces séquences est : $(q_1 - 1) \times t \approx \sqrt{\sqrt{N}/2} \times \sqrt{N/2}$.

3.3 Prophet

L'état interne de PROPHET consiste en k compteurs $m_t = c_1, \cdots c_k$, initialisée aléatoirement. La fonction g_s de mise à jour consiste à incrémenter le compteur numéro s. La fonction d'extraction d'adresses est une

composition : $f_p = v_p \circ h$, avec $v_p(x) = (p + x \bmod q) + 1$ et $h(m_t) = \prod_{i=1}^{k} p_i^{c_i}$ et p_i les k premiers nombres premiers.

Pour comprendre les performances de PROPHET, il faut observer deux choses : (1) toutes les machines à états finis génèrent des valeurs dans le même domaine, (2) la famille $\mathcal{H}$ des fonctions d'extraction est du fait du modulo une famille de fonctions de hachage. On peut donc s'attendre à retrouver le paradoxe des anniversaires, si ce n'est que dans PROPHET les valeurs choisies en entrée des fonctions ne sont pas choisies uniformément. Puisque un résultat purement théorique semble hors de portée, on peut s'en remettre aux statistiques. Nous avons employé le *Birthday Spacing test* défini par Marsaglia [Mar96]. Soit $a_0, a_1, \cdots, a_{m-1}$ la liste triée de m adresses choisis dans l'intervalle $[0, 2^\ell]$. On définit la liste des espacements suivants : $a_0, a_1 - a_0, a_2 - a_1, \cdots, a_{m-2} - a_{m-3}, a_{m-1} - a_{m-2}$. On note Y la variable aléatoire correspondant à m moins le nombre distinct d'espacement. Komlos a prouvé que pour une génération aléatoire des adresses Y suit asymptotiquement une loi de Poisson de paramètre $\lambda = \frac{m^3}{2^{n+2}}$. A partir d'une séquence d'adresses observée, on peut faire un test de conformité χ^2 pour vérifier si un algorithme se comporte ou non comme un générateur aléatoire. Ce test est pratiquement toujours vérifié par PROPHET. En terme de collision sur les adresses, PROPHET se comporte comme une generateur aleatoire, *i.e.* l'allocation sans état.

PROPHET *accumule donc les défauts des algorithmes d'allocation avec état avec ceux des algorithmes sans état.*

4 Conclusion

Nous avons que PRIMEDHCP génère des adresses trop grandes, QUADRATIC RESIDUE BASED DHCP ne fonctionne que pour une topologie particulière de réseaux et que PROPHET a le même taux de collision sur les adresses qu'une allocation aléatoire. Il ne reste pour ainsi dire plus que DISTRIBUTED DHCP [MP02]. DISTRIBUTED DHCP implémente un système d'ensemble d'adresses qui sont sub-divisés au cours de l'allocation. Comme le montre la Table 1, on peut avoir avec DISTRIBUTED DHCP des *problèmes de famine* : un nœud n'est pas capable de trouver une adresse dans son voisinage car tous les ensembles d'adresses ont été consommés. Pour lutter contre cette famine, DISTRIBUTED DHCP implémente un système de récupération qui permet d'obtenir une adresse d'un voisin lointain (à plus d'un saut). Cette méthode semble finalement la plus raisonnable. *Finalement, les premiers (proposés) seront les derniers (à être utilisés).*

Références

[CSX+08] Xiaowen Chu, Yi Sun, Ke Xu, Zeeshan Sakander, and Jiangchuan Liu. Quadratic Residue Based Address Allocation for Mobile Ad Hoc Networks. In *IEEE International Conference on Communications - ICC '08*, pages 2343–2347, Beijing, China, 2008.

[GG03] Joachim Von Zur Gathen and Jurgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, second edition, 2003.

[HT05] Yuan-Ying Hsu and Chien-Chao Tseng. Prime DHCP : a prime numbering address allocation mechanism for MANETs. *IEEE Communications Letters*, 9(8) :712–714, 2005.

[Mar96] Georges Marsaglia. DIEHARD : A battery of tests of randomness, 1996. `http://stat.fsu.edu/~geo/diehard.html`.

[MOVR01] Alfred J. Menezes, Paul C. Van Oorschot, Scott A. Vanstone, and R. L. Rivest. *Handbook of Applied Cryptography*. CRC press, fifth edition, 2001.

[MP02] Mansoor Mohsin and Ravi Prakash. IP address assignment in a mobile ad hoc network. In *Military Communications Conference - MILCOM 2002*, volume 2, pages 856–861, 2002.

[Vai02] Nitin H. Vaidya. Weak duplicate address detection in mobile ad hoc networks. In *ACM international symposium on Mobile ad hoc networking & computing - MobiHoc '02*, pages 206–216, Lausanne, Switzerland, 2002. ACM.

[ZNM03] Hongbo Zhou, Lionel M. Ni, and Matt W. Mutka. Prophet Address Allocation for Large Scale MANETs. In *IEEE INFOCOM 2003*, pages 1304–1311, San Francisco, USA, 2003.