Modélisation et Évaluation des Attaques Ciblées dans un Overlay Structuré

Emmanuelle Anceaume, Romaric Ludinard, Bruno Sericola, Frédéric Tronel

Modélisation et Évaluation des Attaques Ciblées dans un Overlay Structuré

E. Anceaume[*] — R. Ludinard[**] — B. Sericola[***] — F. Tronel[**]

[*] *Irisa UMR CNRS 6074*
Campus de Beaulieu, 263 avenue du Général Leclerc - Bâtiment 12 - 35042 Rennes cedex, France
emmanuelle.anceaume@irisa.fr

[**] *Supélec EA 4039*
Avenue de la Boulaie, 35510 Cesson-Sévigné, France
romaric.ludinard@supelec.fr
frederic.tronel@supelec.fr

[***] *Inria Rennes-Bretagne Atlantique*
Campus de Beaulieu, 263 avenue du Général Leclerc - Bâtiment 12 - 35042 Rennes cedex, France
bruno.sericola@inria.fr

RÉSUMÉ. *Dans cet article, nous nous intéressons aux attaques ciblées dans le cadre des systèmes pair-à-pair large échelle. Ces attaques ont pour but d'affaiblir les nœuds ciblés de manière à diminuer leur capacité à fournir ou à utiliser des services de l'overlay. Pour se prémunir de telles attaques, nous tirons parti du clustering de l'overlay sous-jacent. Cela permet de mettre en place un système de churn induit préservant la répartition aléatoire des identifiants des nœuds dans l'overlay et ainsi rendre impossible toute prédiction de l'adversaire quant à celle-ci. Nous montrons qu'en randomisant légèrement les opérations élémentaires de l'overlay, ainsi qu'en introduisant des temps de séjour adaptés, l'effet de ces attaques ciblées est sensiblement amoindri, et la propagation des effets de l'attaque à l'ensemble du système est évitée.*

ABSTRACT. *In this paper we consider the problem of targeted attacks in large scale peer-to-peer overlays. Targeted attacks aimed at exhausting key resources of targeted hosts to diminish the target node capacity to provide or receive services. To defend the system against such attacks, we rely on clustering and implement induced churn to preserve randomness of nodes identifiers so that adversarial predictions are impossible. We propose robust join, leave, merge and split operations to discourage brute force denial of services and pollution attacks. We show that combining a small amount of randomization in the operations, and adequately tuning the sojourn time of peers in the same region of the overlay allows to decrease the effect of targeted attacks at cluster level, but more importantly to prevent pollution propagation in the whole overlay.*

MOTS-CLÉS : *Chaînes de Markov, Overlay P2P clusterisé, Adversaire, Churn, Collusion*

KEY WORDS: *Markov chains, Clusterized P2P Overlays, Adversary, Churn, Collusion*

1. Introduction

Les systèmes pair-à-pairs structurés organisent classiquement leurs éléments (nœuds et données) autour d'une topologie structurée (hypercube, anneau, etc...) que l'on appelle *overlay*[1], et d'une fonction de hachage (MD5, SHA-1, ...). Les nœuds entrant dans le système sont ainsi placés en fonction d'un haché qui leur est attribué. La fonction de hachage permet de placer équitablement et pseudo-aléatoirement les nœuds au sein du système, engendrant ainsi des topologies équilibrées. Cependant, des attaques ciblées peuvent rapidement biaiser cette répartition uniforme et engendrer l'écroulement du système. Awerbuch et Scheideler ont ainsi montré [AWE 07] que les réseaux pair-à-pair ne peuvent pas survivre aux attaques byzantines si l'adversaire est capable de prédire le devenir de la topologie. Nous proposons dans cet article d'imposer aux nœuds du système de se déplacer régulièrement à l'intérieur de celui-ci de manière à conserver au mieux cette répartition uniforme et ainsi résister aux attaques ciblées.

Différentes approches ont été proposées afin de se prémunir des comportements malveillants dans les systèmes pair-à-pair. Une technique couramment utilisée consiste à sélectionner ses voisins en fonction de leur position dans l'overlay comme dans [RAT 01], [STO 01]. Une autre solution, consiste à remarquer que pendant ce type d'attaque, le voisinage des nœuds malveillants est plus grand que le voisinage moyen des pairs dans l'overlay. On peut donc borner la taille de ce voisinage afin de limiter le pouvoir de l'attaquant. Ainsi, Singht et al. [SIN 06] proposent de surveiller de manière anonyme le voisinage des autres nœuds. Cette solution n'est malheureusement efficace que dans des environnements peu dynamiques. Plus généralement, Castro [CAS 02] et Sit [SIT 02] ont montré qu'un routage redondant permet d'assurer la robustesse de celui-ci. Cette solution a été retenue dans l'implémentation de [FIA 05], [HIL 03]. Cependant, tous ces travaux reposent sur le fait qu'à tout moment, et à tout endroit dans l'overlay, la proportion de pairs malveillants est bornée et connue *a priori* permettant ainsi de s'appuyer sur des mécanismes tolérants aux byzantins. Awerbuch et Scheideler proposent dans [AWE 06] d'introduire du churn dans le système à chaque opération d'entrée/sortie, empêchant ainsi les nœuds malveillants de prédire le devenir de l'overlay après une séquence donnée d'entrée-sortie. Cependant, des expériences ont montré que cette solution est envisageable uniquement si l'overlay contient moins de 25% de nœuds malveillants [ANC 09] ou bien si l'overlay a une topologie statique [CON 06].

Dans la suite, nous présentons dans la section 2 les overlays pair-à-pairs clusterisés, puis dans la section 3 le comportement de l'adversaire est modélisé. Nous évaluons l'impact de cet adversaire au niveau local dans la section 4, puis au niveau du système dans la section 5. La section 6 conclut.

2. Overlays pair-à-pair

Un *overlay* est un réseau logique construit au dessus d'une infrastructure physique. Les nœuds de cet overlay, communément appelés *pairs*, communiquent entre eux en utilisant les liens logiques de l'overlay via les primitives de communication fournies pas le réseau sous-jacent. La structure de l'overlay est caractérisée par l'organisation des pairs à l'intérieur de celui-ci. On distingue clas-

1. Par soucis de lisibilité, le terme anglais *overlay* est préféré dans cet article à sa traduction française peu utilisée "réseau sous-jacent logique".

siquement deux familles d'overlays. Lorsque les pairs s'organisent de manière aléatoire, on parle d'overlay non structuré ou mesh. Ce type d'overlay s'appuie sur un graphe aléatoire. De façon similaire aux pairs, les données sont placées aléatoirement sur ce graphe. Aussi, pour trouver une donnée ou un pair, il faut utiliser des techniques d'inondation maîtrisée, ou des marches aléatoires. À l'inverse, les overlays dits structurés s'organisent selon une topologie basée sur des graphes réguliers de type anneau [STO 01], tore [RAT 01], ou hypercube [MAY 02]. Ainsi chaque pair souhaitant entrer dans l'overlay, se voit attribuer un identifiant unique qui détermine sa position dans celui-ci, permettant ainsi de le contacter de manière efficace (classiquement en un nombre poly-logarithmique de sauts en fonction de la taille de l'overlay). Cependant lorsque le *churn*, i.e. la fréquence d'entrée/sortie des pairs, est trop important la fréquence de maintenance des tables de routage est telle qu'elle peut nuire à la cohérence de celles-ci. Pour pallier ce problème, les pairs se regroupent au sein de *groupes* ou *clusters*. Dans ce cas, ce sont ces derniers qui s'organisent selon une topologie structurée. Cette approche permet également d'augmenter la robustesse du système vis-à-vis de stratégies malveillantes.

Plus spécifiquement, les overlays clusterisés s'appuient sur un graphe régulier (tore, hypercube,...) dont les sommets ne sont plus des pairs, mais des groupes de pairs. Ainsi quand un nœud entre dans le système, une identité lui est attribuée, et le nœud est placé en fonction de celle-ci et d'une métrique propre au système. Par exemple dans le cas de PeerCube [ANC 08], le nœud se place dans le cluster dont le label préfixe l'identité du nœud, tandis que dans eQuus [LOC 06], le nœud rejoint le cluster de nœuds dont il est le plus proche géographiquement. Cependant, pour garder un équilibre au sein des clusters, ceux-ci ont des tailles bornées. Ainsi, lorsque la taille d'un cluster tombe en deçà d'une constante *C*, celui ci fusionne avec un autre. La taille minimale *C*, est fixée en fonction de contraintes liées aux défaillances tolérées par le système. De la même manière, lorsque le cluster contient trop de pairs, i.e. sa taille dépasse la borne *Smax*, celui-ci se scinde en deux nouveaux clusters en répartissant les pairs entre eux. Cette borne est fixée en fonction de considérations liées au passage à l'échelle du système. Typiquement, $Smax = \mathcal{O}(\log U)$ où U est la taille du système.

Dans ce qui suit, nous considèrons que les clusters sont divisés en deux parties : le *core* et le *spare*. Le core est un ensemble de pairs de taille fixe *C* qui sont en charge de la gestion de la topologie de l'overlay et assurent le routage au sein de celui-ci. En utilisant des techniques basées sur les quorums, on peut ainsi réduire l'impact d'éventuels pairs malveillants. Le spare, quant à lui, ne participe pas aux opérations structurelles de l'overlay, et n'est visible que des membres du core, permettant ainsi d'absorber en partie le churn du système et ainsi de réduire les coûts de mise à jour liés au churn. Ainsi, lorsqu'un pair entre dans le cluster, il est placé dans le spare, de manière transparente pour le reste de l'overlay. En revanche, lorsqu'un pair quitte le cluster, deux cas peuvent se produire. Si le pair appartient au spare, alors son départ est transparent, il est simplement enlevé du spare. Par contre, lorsque le pair appartient au core, il faut renouveler la composition de ce dernier. Nous un introduisons un paramètre k propre au système. Nous nous proposons de choisir aléatoirement $k-1$ nœuds du core afin de renouveler en partie sa composition. Ces $k-1$ nœuds du core sont placés dans le spare, puis parmi tous les nœuds du spare, k sont aléatoirement choisis et placés dans le core. Ainsi, suivant la valeur de k on peut faire varier le taux de randomisation à chaque renouvellement. Dans le cas $k = 1$, on choisit un nœud du spare pour le placer dans le core. À l'inverse, lorsque $k = C$, on renouvelle en intégralité la composition du core. On notera $Protocole_k$, le renouvellement du core en utilisant k pairs fraîchement élus. Nous allons étudier dans la suite de cet article l'influence de ce paramètre k en présence d'un adversaire essayant de corrompre l'overlay.

Comme nous l'avons dit précédemment, il faut en permanence conserver une distribution aléatoire des identifiants au sein de l'overlay [AWE 07], et il faut s'assurer qu'aucun pair ne puisse rester au même endroit infiniment longtemps. Nous avons prouvé ces deux propriétés de manière analytique dans un précédent travail [ANC 09], où l'on considérait des régions de taille constante. Afin d'imposer des temps de séjours limités, et de conserver une distribution aléatoire des identifiants au sein de l'overlay, nous proposons d'utiliser des identités périssables. Ainsi lorsqu'un pair entre dans l'overlay pour la première fois, il acquiert un certificat auprès d'une autorité de confiance. Ensuite, en appliquant une fonction de hachage sur certains champs de son certificat il obtient une identité ID^0 valable pour une durée L. Lorsque cette identité atteint sa date de péremption, le nœud doit quitter l'overlay pour y rentrer à nouveau avec une nouvelle identité ID^n, n correspond alors au nombre de fois où le pair a été contraint de quitter l'overlay pour renouveller son identité : $n = \lceil (t - t_0)/L \rceil$, avec t_0 la date d'obtention du certificat, t l'instant courant, et L la durée de vie de l'identité, fixée par le système. L'identité ID^n est obtenue ainsi : $ID^n = \mathcal{H}(ID^0 \times n)$. La date de création t_0 est publique ainsi tout nœud peut vérifier la validité d'une autre identité. On dit que l'identité d'un pair est valide, si elle correspond à celle qu'un autre pair peut calculer au même moment. De cette manière, comme le placement des pairs dans l'overlay dépend de leur identité, tous les nœuds sont amenés à se déplacer au sein de celui-ci, assurant ainsi un brassage des identités permettant de conserver une distribution aléatoire des nœuds au sein de l'overlay. Nous modélisons cette durée de vie par un processus de décroissance exponentielle de demie-vie $t_{1/2} = \ln 2/(1 - d)$ où d représente la probabilité qu'une identité d'un pair n'ait pas expiré (par unité de temps). Ainsi, d est homogène à une fréquence. En pratique, prendre $L = 6.65 \times t_{1/2}$ assure que 99% des identités des pairs concernés ont expiré.

3. Spécification du comportement des nœuds malveillants

Les systèmes pair-à-pair sont des systèmes ouverts, c'est-à-dire que l'accès au système n'est pas réservé à certains nœuds. Inévitablement dans ce type de système, un certain nombre de nœuds essayent de manipuler le système. Ces comportements malveillants peuvent être isolés ou à l'inverse, peuvent impliquer la participation de plusieurs pairs conduisant à des stratégies complexes à tolérer. On modélise ces comportements par la présence d'un adversaire puissant capable de manipuler une partie des pairs de l'overlay. Ainsi, on considère que l'adversaire a, à sa disposition, une fraction μ ($0 < \mu < 1$) des nœuds pouvant entrer dans le système. Les nœuds contrôlés par l'adversaire sont dits *malveillants*, tandis que les autres sont dits *honnêtes*. Un nœud honnête ne peut faire la différence entre un autre nœud honnête et un nœud malveillant. Le but de l'adversaire est donc de faire rentrer le maximum des nœuds qu'il manipule dans l'overlay pour contrôler une importante portion de l'overlay de manière à faire dévier le système de son comportement normal. Au niveau du cluster, cela signifie qu'il suffit à l'adversaire de placer suffisamment de ses pairs dans le core du cluster, et ainsi de biaiser les prises de décisions faites par les membres du core. D'après [LAM 82], il suffit que l'adversaire place plus de $c = \lfloor (C - 1)/3 \rfloor$ pairs malveillants au sein du core pour imposer ses décisions. Un cluster dont le core est composé de plus de c pairs malveillants est dit *pollué*, sinon il est dit *sain*.

Comme les pairs sont placés dans l'overlay en fonction de leur identifiant, l'adversaire ne peut pas placer ses nœuds où il le souhaite dans l'overlay. Néanmoins, en faisant entrer suffisamment de

ses nœuds dans le système, il peut augmenter ses chances d'atteindre le cluster visé. De plus, comme les pairs entrant sont d'abord placés dans le spare, il lui faut attendre qu'un autre nœud quitte le core de manière à déclencher un renouvellement de sa composition pour éventuellement réussir à placer des nœuds malveillants au sein du nouveau core. Néanmoins, il peut arriver qu'il soit intéressant pour l'adversaire de provoquer un départ de ses nœuds. En effet, si celui-ci a déjà placé des pairs malveillants au sein du core, et qu'il y en a suffisamment dans le spare, il peut décider de provoquer un départ du core s'il estime que le processus de renouvellement peut tourner à son avantage. Plus formellement, l'adversaire applique la règle suivante :

Règle 1 (Stratégie du départ)**.** *Soit $\mathcal{C}$ un cluster tel qu'à l'instant t, son core est composé de x pairs valides et malveillants $0 < x \leq c$ et son spare contient y pairs valides et malveillants $y \geq 1$. Soit ν, un paramètre fixé par l'adversaire. Si la condition [1] est vérifiée alors l'adversaire retire un ses nœuds du core. Spécifiquement,*

$$\sum_{j=x+1}^{x-1+\min(k,y)} \Pr\left\{ \begin{array}{l} \textit{exactement } j \textit{ pairs malveillants } \in \mathcal{C} \\ \textit{après le retrait d'un pair malveillant du core} \end{array} \right\} > 1 - \nu. \qquad [1]$$

Le paramètre k correspond ici au taux de renouvellement des nœuds du core. Notons que dans le cas $k = 1$, cette règle ne s'applique jamais. En effet, dans le meilleur cas, l'adversaire remplace un nœud par un autre. D'autre part, l'adversaire n'applique pas non plus cette règle dans le cas où le cluster est faiblement peuplé. En effet, déclencher un départ dans un cluster de taille $C+1$ induit une fusion de deux clusters, et la création d'un nouveau core où l'adversaire n'a aucun pouvoir sur sa composition. De manière duale, si le cluster est pollué, l'adversaire bloque les arrivées de nouveaux nœuds de manière à éviter la scission de son cluster, ce qui pourrait éventuellement lui faire perdre son avantage. Ce comportement est décrit par la règle 2.

Règle 2 (Stratégie des arrivées)**.** *Soit $\mathcal{C}$ un cluster tel qu'à l'instant t, son core soit composé de $\ell > c$ pairs valides et malveillants, alors tout arrivée d'un pair p dans le cluster est ignorée si (p est honnête et $s > 1$) ou si ($s = \Delta - 1$) avec s la taille du spare et Δ la taille maximale du spare.*

4. Modélisation du comportement local

L'état d'un cluster dépend de la stratégie du renouvellement du core $Protocole_k$ employée, du comportement de l'adversaire, et évolue en fonction des arrivées et des départs des nœuds. Nous modélisons l'état d'un cluster et son évolution par une chaîne de Markov à temps discret X définie par $X = \{X_n, n \geq 0\}$. L'espace d'états de cette chaîne Ω est défini par $\Omega = \{(s, x, y) \mid 0 < s < \Delta, 0 \leq x \leq C, 0 \leq y \leq s\}$. Pour $n \geq 1$, $X_n = (s, x, y)$ représente l'état du cluster où après n évènements d'entrée/sortie, celui-ci a un spare de taille s contenant y pairs malveillants et x pairs malveillants dans le core. Par la suite, nous désignerons pas α la distribution initiale de X et M sa matrice de transition.

On désigne par S l'ensemble des états où le cluster est sain, défini par $S = \{(s, x, y) \mid 0 < s < \Delta, 0 \leq x < c, 0 \leq y \leq s\}$ et P l'ensemble des états où le cluster est pollué, défini par $P = \{(s, x, y) \mid 0 < s < \Delta, c \leq x \leq C, 0 \leq y \leq s\}$. Ainsi, le cluster évolue entre des états sains et pollués avant d'atteindre un état où il doit soit fusionner avec un autre cluster, soit se scinder en

deux nouveaux clusters. Ces états constituent des classes absorbantes pour la chaîne de Markov. On note alors A_S^m l'ensemble des états où le cluster est sain et qu'il doit fusionner avec un autre cluster : $A_S^m = \{(s,x,y) \mid (s=0) \wedge (0 \leq x < c)\}$, A_P^m l'ensemble d'états où le cluster est pollué et doit fusionner avec un autre cluster, $A_P^m = \{(s,x,y) \mid (s=0) \wedge (c \leq x \leq C)\}$, et A_S^ℓ l'ensemble d'états où le cluster doit se scinder en deux, $A_S^\ell = \{(s,x,y) \mid (s=\Delta)\}$. Ainsi, on peut partitionner la matrice de transition M suivant ces ensembles d'états :

$$M = \begin{pmatrix} M_S & M_{SP} & M_{SA_S^m} & M_{SA_S^\ell} & M_{SA_P^m} \\ M_{PS} & M_P & M_{PA_S^m} & M_{PA_S^\ell} & M_{PA_P^m} \\ 0 & 0 & M_{A_S^m} & 0 & 0 \\ 0 & 0 & 0 & M_{A_S^\ell} & 0 \\ 0 & 0 & 0 & 0 & M_{A_P^m} \end{pmatrix} \quad [2]$$

où M_{UV} est la sous-matrice de M de dimensions $|U| \times |V|$, comprenant les probabilités de transitions des états de U vers les états de V, avec $U, V \in \{S, P, A_S^m, A_P^m, A_S^\ell\}$. De la même manière on partitionne le vecteur initial $\alpha = (\alpha_S \;\; \alpha_P \;\; \alpha_{A_S^m} \;\; \alpha_{A_S^\ell} \;\; \alpha_{A_P^m})$. La figure 1 montre une vue agrégée de la chaîne de Markov X.

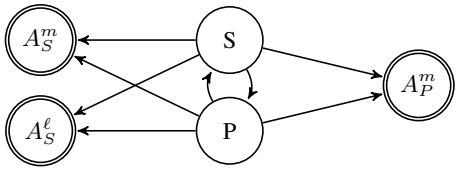


Figure 1. *Vue agrégée de la chaîne de Markov X. Les états transitoires sains et pollués sont représentés par S and P. Les états absorbants sont représentés par A_S^m, A_S^ℓ and A_P^m.*

Le calcul des probabilités de transitions (i.e. les coefficients de la matrice de transition M) sont décrits par la figure 2. Dans cet arbre, chaque branche est étiquetée par la probabilité associée, et les feuilles de l'arbre représentent les états du cluster, atteignables depuis la racine de l'arbre. Les valeurs de ces transitions dépendent *i)* du type d'opération (arrivée, départ du spare, départ du core), *ii)* du type de pair impliqué dans l'opération (honnête ou malveillant), *iii)* de la composition du cluster (sain, pollué). La probabilité associée à la transition est obtenue en sommant tous les produits des différentes probabilités sur les arcs allant de l'état initial à l'état considéré, suivant les différents chemins. Par exemple, si on considère la transition $(s,x,y) \rightarrow (s+1,x,y+1)$ décrite par la branche gauche de l'arbre, elle correspond au scénario où un pair malveillant souhaite rentrer dans un cluster pollué. Grâce à la règle 2, le pair entrant réussit à intégrer le cluster, amenant celui-ci dans l'état $(s+,x,y+1)$. De la même manière, si l'on considère la branche droite, nous sommes alors dans le cas où le cluster est pollué et l'un des pairs malveillants p n'a plus une identité valide (péremption). Le pair p doit alors quitter le cluster, et un processus de renouvellement est déclenché. Le cluster étant encore pollué, l'adversaire peut biaiser le processus de renouvellement et choisir un de ses nœuds afin de le placer dans le nouveau core, plaçant le cluster dans l'état $(s-1,x,y-1)$. Pour des raisons évidentes de place, nous n'allons pas décrire ici l'intégralité des transitions, le lecteur intéressé pourra les trouver dans [ANC 10]. Nous nous intéressons au comportement du cluster au regard de la puissance de l'adversaire μ, la fréquence du churn induit d, et du taux k de randomisation

suivant les différents $Protocole_k$. On considérera par la suite la distribution initiale δ correspondant à l'état $(\Delta/2, 0, 0)$.

La chaîne de Markov X considérée est absorbante. On s'intéresse alors au temps passé dans les états sains (resp. pollués) avant l'absorption. On définit $T_S^{(k)}$ (resp. $T_P^{(k)}$) la variable aléatoire du temps passé dans les états sains (resp. pollués) avant l'absorption. Ainsi, $T_S^{(k)} = \sum_{n=0}^{\infty} 1_{\{X_n \in S\}}$ et $T_p^{(k)} = \sum_{n=0}^{\infty} 1_{\{X_n \in P\}}$. D'après [SER 90], l'espérance de $T_S^{(k)}$ (resp. $T_P^{(k)}$) est donnée par :

$$E(T_S^{(k)}) = v(I-R)^{-1}\mathbb{1} \text{ et } E(T_P^{(k)}) = w(I-Q)^{-1}\mathbb{1} \quad [3]$$

avec $v = \alpha_S + \alpha_P(I - M_P)^{-1}M_{PS}$, $R = M_S + M_{SP}(I - M_P)^{-1}M_{PS}$, $w = \alpha_P + \alpha_S(I - M_S)^{-1}M_{SP}$ et $Q = M_P + M_{PS}(I - M_S)^{-1}M_{SP}$. La notation $\mathbb{1}$ représente le vecteur colonne de dimension adéquate ne contenant que des 1.

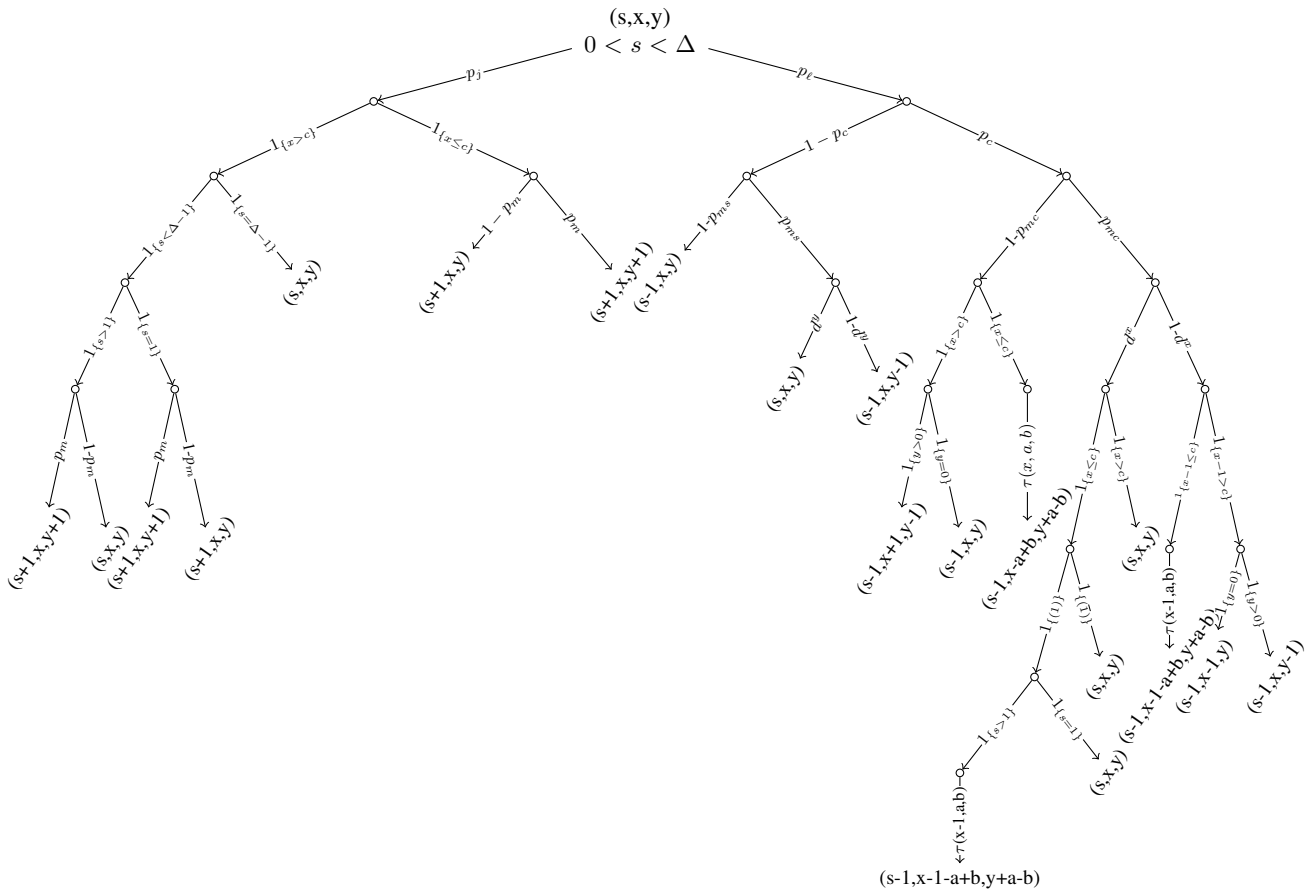
La figure 3 compare le temps moyen passé par le cluster dans les états sains et pollués avant l'absorption, pour les protocoles $Protocole_1$ et $Protocole_C$ pour différentes valeurs de μ et d. Dans la suite de cet article, nous ne nous intéresserons qu'à ces deux protocoles. En effet, le $Protocole_1$ représente le minimum de randomisation possible dans la génération d'un nouveau core tandis que $Protocole_C$ représente la randomisation maximale. Étonnamment, augmenter le taux de randomisation ne rend pas le protocole plus résilient vis-à-vis de l'adversaire. Deux choses ressortent de cette comparaison. On constate que $Protocole_1$ se comporte mieux que $Protocole_C$, c'est-à-dire que pour toutes valeurs de μ et d fixées, $Protocole_1$ passe plus de temps dans les états sains que $Protocole_C$, et moins de temps dans les états pollués. De plus, on constate qu'à μ constant, le temps passé dans les états pollués augmente avec d tandis que celui passé dans les états sains diminue. En effet, plus d s'approche de 1, plus la durée des identités des pairs est longue, plus on tend vers un système où les nœuds ne sont pas obligés de se déplacer dans l'overlay, permettant en particulier aux nœuds malveillants de rester en place de manière à être élus dans le core et ainsi corrompre le cluster. Le tableau 1 montre les valeurs des temps passés dans les états sains et pollués pour des valeurs de d proches de 1.

	$\mu = 0\%$			$\mu = 20\%$			$\mu = 30\%$		
d	0.95	0.99	0.999	0.95	0.99	0.999	0.95	0.99	0.999
$E(T_S^{(1)})$	12.0	12.0	12.0	11.88	11.84	11.83	11.54	11.48	11.47
$E(T_P^{(1)})$	0.0	0.0	0.0	1.14	699.7	511810822.	5.96	12597.	9299884149

Tableau 1. *$E(T_S^{(k)})$ et $E(T_S^{(k)})$ en fonction de μ et d. Dans ces expérimentations, $k = 1$, $C = 7$, $\Delta = 7$, and $\alpha = \delta$.*

On s'intéresse à présent aux états absorbants de la chaîne de Markov. En effet, on peut penser qu'en temps normal il se produit des fusions et des scissions de clusters dans les même proportions. Nous nous intéressons à présent aux conséquences sur le système de l'utilisation de ces identités périssables. On réécrit la matrice de transition M comme suit :

$$M = \begin{pmatrix} T & R_S^m & R_S^\ell & R_P^m \\ 0 & * & * & * \end{pmatrix} \text{ avec } T = \begin{pmatrix} M_S & M_{SP} \\ M_{PS} & M_P \end{pmatrix}$$



Probabilités	Valeur	Signification
p_j (resp. p_ℓ)	$1/2$	Pr{une arrivée (resp. départ) se produise}
p_c	$C/(C+s)$	Pr{que le pair appartienne au core}
p_m	μ	Pr{le pair entrant soit malveillant}
p_{mc}	x/C	Pr{le membre du core soit malveillant}
p_{ms}	y/s	Pr{le membre du spare soit malveillant}
	$1-d^x$ (resp. $1-d^y$)	Pr{un pair malveillant du core (resp. spare) ne soit plus valide}
$1_{\{A\}}$	1 si la condition A est vérifiée, 0 sinon	fonction indicatrice
$\tau(x,a,b)$	$q(k-1,C-1,a,x)q(k,s+k-1,b,y+a)$	Pr{générer un core (resp. spare) contenant $x-a+b$}
	avec $q(k,\ell,u,v) = \binom{v}{u}\binom{\ell-v}{k-u} \,/\, \binom{\ell}{k}$	(resp. $y-b+a$) pairs malveillants avec
		$\max(0, k-1-(C-1-x)) \le a \le \min(x, k-1)$, et
		$\max(0, k-(s+k-1-(y+a)) \le b \le \min(y+a, k)$

Figure 2. *Diagramme de transition des états de X*

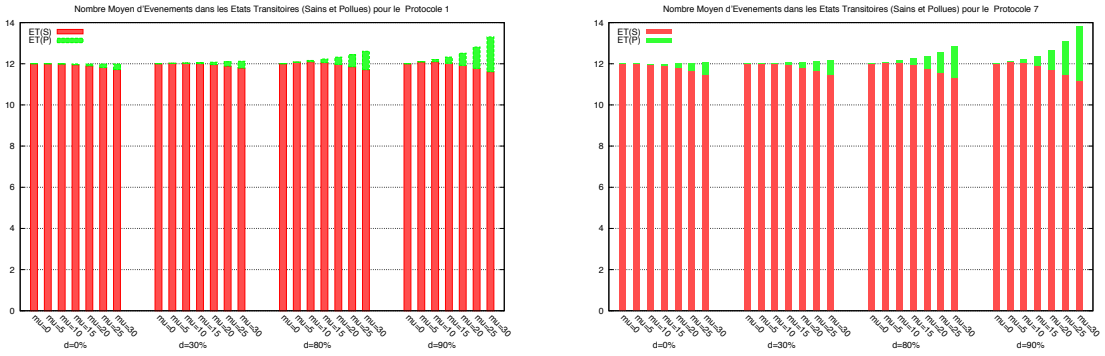


Figure 3. *$E(T_S^{(k)})$ (Relation [3]) représenté en haché, et $E(T_P^{(k)})$ (Relation [3]) représenté en plein en fonction de k, μ et d. Dans ces expérimentations, $C = 7$, $\Delta = 7$. La figure de gauche correspond au $Protocole_1$, et celle de droite au $Protocole_C$*

$$\text{et } R_U^v = \begin{pmatrix} M_{SA_U^v} \\ M_{PA_U^v} \end{pmatrix} \text{ avec } U \in \{S, P\} \text{ et } v \in \{m, \ell\}.$$

On fait de même avec la distribution initiale $\alpha = (\alpha_T \ 0 \ 0 \ 0)$, avec $\alpha_T = (\alpha_S \ \alpha_P)$. La probabilité $p(A_S^\ell)$ pour la chaîne de Markov X d'être absorbée dans A_S^ℓ est donnée par :

$$p(A_S^\ell) = \alpha_T (I - T)^{-1} R_S^\ell \mathbb{1}. \qquad [4]$$

Les probabilités que la chaîne de Markov X soit absorbée dans les autres classes s'obtiennent de la même manière.

La figure 4 montre ces différentes probabilités dans le cas de $Protocole_1$. En l'absence de comportement malveillant, (cas $\mu = 0$), le cluster demeure sain, jusqu'à être absorbé. Conformément à l'intuition première, les absorptions dues à une fusion ou à une scission sont équiprobables. À l'inverse, en présence d'un comportement malveillant, le cluster a tendance a fusionner avec un autre. En effet, dès que l'adversaire a réussi à polluer un cluster, il peut bloquer les arrivées de nœuds dans le cluster. En revanche, il ne peut pas empêcher les départs. Les arrivées et les départs ne sont donc plus équiprobables, et par conséquent, la probabilité de fusion est supérieure à celle de scission.

5. Modélisation de la dynamique globale

On s'intéresse maintenant à l'impact du comportement malveillant sur l'ensemble du système, et en particulier à la proportion de clusters pollués dans le système au cours du temps. On considère un overlay constitué de n clusters $\mathcal{C}_1$, ..., $\mathcal{C}_n$. Chaque cluster $\mathcal{C}_i$ implémente le protocole $Protocole_k$.

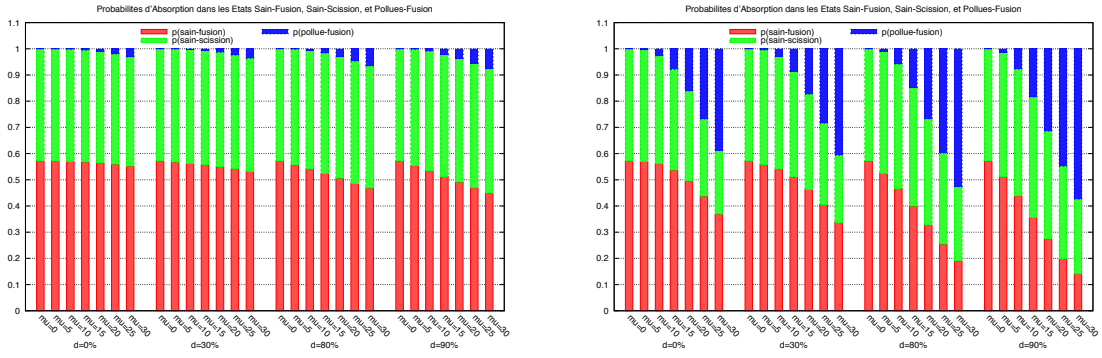


Figure 4. *$p(A_S^m)$, $p(A_S^\ell)$ et $p(A_P^m)$ (cf. Relation [4]) représentées respectivement en rouge haché, plein et vert haché en fonction de μ et d. Dans ces expérimentations, $k = 1$, $C = 7$, et $\Delta = 7$.*

Les arrivées et départs de nœuds se produisent de manière uniforme au sein de l'overlay. Cela signifie qu'à chaque opération, le cluster $\mathcal{C}_i$ a une probabilité $p_i = 1/n$ d'être concerné par l'opération. Ainsi, on considère, pour $n \geq 1$, n chaînes de Markov $X^{(1)}, ..., X^{(n)}$ identiques à X, i.e. chacune des X_i a le même espace d'état Ω, la même matrice de transition M ainsi que la même distribution initiale α. Cependant, ces n chaînes de Markov ne sont pas indépendantes. En effet, dès qu'une arrivée ou un départ se produit dans le système, un et un seul cluster est concerné. Cela signifie donc qu'à chaque transition, une et une seule chaîne est activée pour effectuer cette transition. On s'intéresse à présent, aux deux variables aléatoires $N_S^n(m)$ et $N_P^n(m)$ représentant respectivement le nombre de clusters dans un état sain et pollué après m opérations dans l'overlay. On définit $N_S^n(m)$ et $N_P^n(m)$ par

$$N_S^{(n)}(m) = \sum_{h=1}^{n} 1_{\{X_m^{(h)} \in S\}} \text{ et } N_P^{(n)}(m) = \sum_{h=1}^{n} 1_{\{X_m^{(h)} \in P\}}.$$

Nous avons précédemment prouvé [ANC 11] le théorème suivant :

Théorème 1. *$\forall\, h = 1, \ldots, n$, $m \geq 0$ et $j \in \Omega$, on a :*

$$\Pr\{X_m^{(h)} = j\} = \sum_{\ell=0}^{m} \binom{m}{\ell} \left(\frac{1}{n}\right)^{\ell} \left(1 - \frac{1}{n}\right)^{m-\ell} \Pr\{X_\ell = j\}. \qquad [5]$$

Même si les chaînes de Markov sont dépendantes, elles sont identiques et chacune est choisie avec la même probabilité $p_i = 1/n$, ce qui explique que cette expression soit indépendante de h.

L'espérance de ces deux variables aléatoires est donnée par le théorème suivant :

Théorème 2. *$\forall n \geq 1$ et $m \geq 0$,*

$$\frac{E(N_S^{(n)}(m))}{n} = \alpha \left(\frac{1}{n}T + \left(1 - \frac{1}{n}\right) I\right)^m \mathbb{1}_S \text{ et } \frac{E(N_P^{(n)}(m))}{n} = \alpha \left(\frac{1}{n}T + \left(1 - \frac{1}{n}\right) I\right)^m \mathbb{1}_P \qquad [6]$$

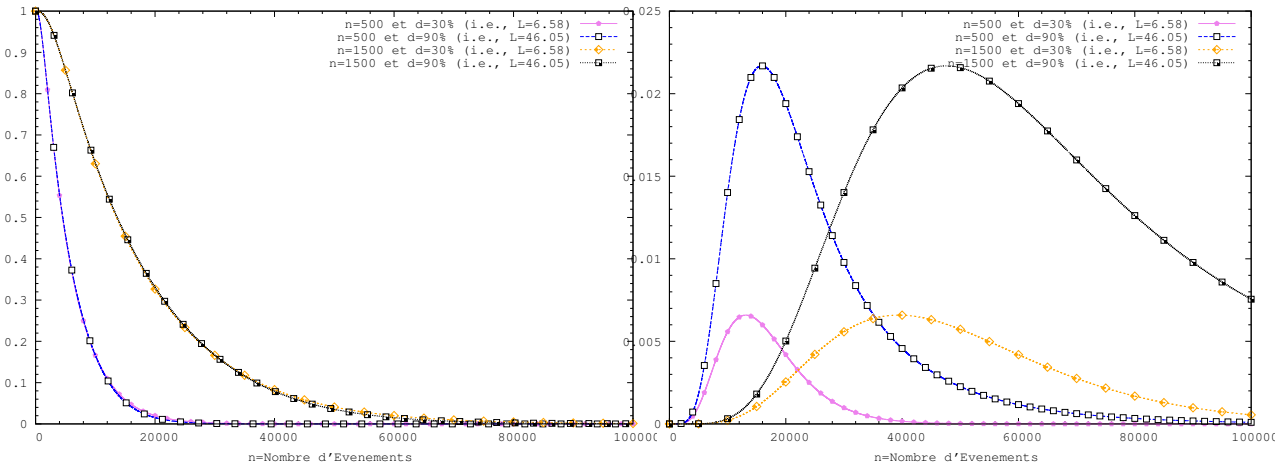


Figure 5. $\frac{E(N_S^{(n)}(m))}{n}$ *et* $\frac{E(N_P^{(n)}(m))}{n}$ *en fonction de* m *pour différentes valeurs de* n *et* d.

avec T la sous-matrice des états transitoires d'ordre $|S \cup P|$, I la matrice identité d'ordre $|S \cup P|$ et $\mathbb{1}_S$ (resp. $\mathbb{1}_P$) de dimension $|S \cup P|$, le vecteur de projection sur S (resp. P).

Les états de $S \cup P$ étant transitoires, la matrice T est sous-stochastique, par conséquent, $T/n + (1 - 1/n)I$ l'est aussi, et donc pour $n \geq 1$, on a :

$$\lim_{m\to\infty} \frac{E(N_S^{(n)}(m))}{n} = \lim_{m\to\infty} \frac{E(N_P^{(n)}(m))}{n} = 0.$$

La figure 5 illustre l'évolution du pourcentage moyen de clusters sains (resp. pollués) en fonction du nombre d'opérations, pour différentes tailles de système ($n = 500, 1500$), et différents temps de séjours ($d = 0.3, 0.9$). Une valeur de d proche de 0 signifie pour le nœud une durée de vie quasi nulle et donc qu'il doit quasiment se déplacer dans l'overlay en permanence. À l'inverse, une valeur de d égale à 1 symbolise une durée de vie infinie. Ajuster la valeur de d permet donc de trouver un compromis acceptable entre durée de vie des nœuds et proportion de clusters pollués.

6. Conclusion

En l'absence de churn induit, des nœuds malveillants peuvent très rapidement corrompre l'intégralité d'un système pair-à-pair. Étonnamment, introduire une faible randomisation dans les algorithmes de renouvellement du core est plus intéressant vis-à-vis de la résistance face à l'adversaire en termes d'implémentation. De plus, en introduisant une mobilité forcée des nœuds à l'intérieur du système et en paramétrant habilement les temps de séjours, on peut réduire efficacement la propagation de la pollution des clusters tout en limitant le surcoût engendré par l'utilisation de churn induit via des identités périssables.

7. Bibliographie

[ANC 08] ANCEAUME E., BRASILEIRO F., LUDINARD R., RAVOAJA A., « PeerCube : an Hypercube-based P2P overlay robust against collusion and churn », *Proceedings of the IEEE Int'l Conference on Self-Adaptive and Self-Organizing Systems*, 2008.

[ANC 09] ANCEAUME E., BRASILEIRO F., LUDINARD R., SERICOLA B., TRONEL F., « Analytical Study of Adversarial Strategies in Cluster-based Overlays », *Proceedings of the International Workshop on on Reliability, Availability, and Security (WRAS)*, 2009.

[ANC 10] ANCEAUME E., LUDINARD R., SERICOLA B., TRONEL F., « Performance Analysis of Large Scale Peer-to-Peer Overlays using Markov Chains », rapport n° 1963, 2010, IRISA.

[ANC 11] ANCEAUME E., CASTELLA F., LUDINARD R., SERICOLA B., « Markov Chains Competing for Transitions : Application to Large Scale Distributed Systems », rapport n° 1953, 2011, INRIA.

[AWE 06] AWERBUCH B., SCHEIDELER C., « Towards a Scalable and Robust DHT », *Proceedings of the 18th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2006.

[AWE 07] AWERBUCH B., SCHEIDELER C., « Towards Scalable and Robust Overlay Networks », *Proceedings of the International Workshop on Peer-to-Peer Systems (IPTPS)*, 2007.

[CAS 02] CASTRO M., DRUSCHEL P., GANESH A., ROWSTRON A., WALLACH D. S., « Secure Routing for Structured Peer-to-Peer Overlay Networks », *Proceedings of the Symposium on Operating Systems Design and Implementation*, 2002.

[CON 06] CONDIE T., KACHOLIA V., SANKARARAMAN S., HELLERSTEIN J. M., MANIATIS P., « Induced Churn as Shelter from Routing-Table Poisoning », *Procs of the 13th thirteenth Annual Symposium on Network and Distributed System Security (NDSS'06)*, 2006.

[FIA 05] FIAT A., SAIA J., YOUNG M., « Making chord robust to byzantine attacks », *Proceedings of the Annual European Symposium on Algorithms*, 2005.

[HIL 03] HILDRUM K., KUBIATOWICZ J., « Asymptotically efficient approaches to fault-tolerance in peer-to-peer networks », *Proceedings of the International Symposium on Distributed Computing (DISC)*, 2003.

[LAM 82] LAMPORT L., SHOSTAK R., PEASE M., « The Byzantine Generals Problem », *ACM Transactions on Programming Languages and Systems*, vol. 4, 1982.

[LOC 06] LOCHER T., SCHMID S., WATTENHOFER R., « eQuus : A Provably Robust and Locality-Aware Peer-to-Peer System », *Proceedings of the Int'l Conference on Peer-to-Peer Computing*, 2006.

[MAY 02] MAYMOUNKOV P., MAZIERES D., « Kademlia : A peer-to-peer information system based on the XOR metric », *Proceedings of the International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[RAT 01] RATNASAMY S., FRANCIS P., HANDLEY M., KARP R., SHENKER S., « A Scalable Content-Addressable Network », *Proc. of the ACM SIGCOMM*, 2001.

[SER 90] SERICOLA B., « Closed form solution for the distribution of the total time spent in a subset of states of a Markov process during a finite observation period », *Journal of Applied Probability*, vol. 27, 1990.

[SIN 06] SINGH A., NGAN T., DRUSHEL P., WALLACH D., « Eclipse Attacks on Overlay Networks : Threats and Defenses », *Proceedings of the International Conference on Computer Communications*, 2006.

[SIT 02] SIT E., MORRIS R., « Security Considerations for Peer-to-Peer Distributed Hash Tables », *Proceedings of the Int'l Workshop on Peer-to-Peer Systems*, 2002.

[STO 01] STOICA I., LIBEN-NOWELL D., MORRIS R., KARGER D., DABEK F., KAASHOEK M. F., BALAKRISHNAN H., « Chord : A Scalable Peer-to-peer Lookup Service for Internet Applications », *Proc. of the ACM SIGCOMM*, 2001.