UNIVERSITÉ Pierre et Marie CURIE-Paris

École Doctorale : Informatique, Télécommunications, Electronique
Année Universitaire : 2005/2006

Thèse pour obtenir le grade de

Docteur de l'Université Paris 6

soutenue par **Youssef AZZANA**

le 5 Juillet 2006

Discipline: **Informatique**

Mesures de la topologie et du trafic Internet

Directeur de thèse: **Serge FDIDA**

Co-directeur de thèse: **Philippe ROBERT**

Membres du Jury

Président	Serge FDIDA	LIP6	Université Pierre et Marie Curie
Rapporteur	Gérard HEBUTERNE	INT	
Rapporteur	Jean-Marc VINCENT	I.M.A.G.	Université Joseph Fourier
Examinateur	Philippe ROBERT	RAP	INRIA Rocquencourt
Examinateur	Fabrice Guillemin	France Telecom R&D	
Examinateur	Patrick GALLINARI	LIP6	Université Pierre et Marie Curie

Remerciements

Je remercie Philippe ROBERT pour m'avoir accueilli dans son équipe et avoir encadré mon travail pendant ces années de thèse. Il m'a permis de goûter au domaine des algorithmes probabilistes dans l'Internet. J'ai apprécié en lui sa rigueur mathématique, ses nombreux conseils, son enthousiasme et bien d'autres qualités humaines. L'ambiance du projet RAP est une agréable incitation à s'intéresser à tous les aspects algorithmiques ou mathématiques ayant un lien avec les réseaux.

Je suis aussi reconnaissant à Serge FDIDA pour avoir accepté de diriger ma thèse et à Gérard HEBUTERNE et Jean-Marc VINCENT d'avoir accepté d'être rapporteurs de ma thèse.

Mes plus chaleureux remerciements s'adressent à tous mes compagnons de travail, Christine FRICKER avec qui j'ai partagé l'espace de travail dans une bonne convivialité, Virginie COLLETTE, Hanène MOHAMED, Yousra CHABCHOUB, Danièlle TIBI, Philippe ROBERT, Stelios KOUREMENOS, Nelson ANTUNES et Abdelghani BEN TAHAR.

Je voudrais également remercier Fabrice GUILLEMIN ingénieur R&D chez France Telecom et toute son équipe (Stéphanie POISSON, Walid SADDI, Jacqueline BOYER...) de leur collaboration étroite et de nous avoir accordé toutes les facilités pour mener à bien nos expériences au laboratoire de recherche de France Telecom.

Mes vives remerciements vont aussi à toute l'équipe MIRIAD (Philippe SULTAN, Eric GALLULA, André BALSA...) qui nous ont fourni les traces de routeurs sur lesquelles on a testé nos algorithmes.

Je remercie tous les membres de mon jury de thèse, Philippe ROBERT, Serge FDIDA, Fabrice GUILLEMIN, Gérard HEBUTERNE, Jean Marc VINCENT et Patrick GALLINARI, de me faire l'honneur d'assister à ma soutenance.

Merci enfin à toute ma famille, mes proches et mes amis.

Contents

Introduction **9**

0.1 Vue générale 9
0.2 Topologie de l'Internet 10
0.2.1 Topologie au niveau des systèmes autonomes 12
0.2.2 Topologie au niveau des routeurs 13
0.2.3 Quelques propriétés de l'Internet 13
0.2.4 Contributions de la thèse sur la topologie 15
0.3 Statistiques des flots 16
0.3.1 Statistiques des flots avec traces complètes 17
0.3.2 Statistiques des flots par échantillonnage 20
0.3.3 Contributions de la thèse sur l'algorithmique des grands flots 23

1 Topology discovery **25**

1.1 Introduction 25
1.2 Notation and definitions 27
1.2.1 Assumptions and notation 27
1.2.2 A dynamical picture for the placement of hosts 29
1.2.3 The shape of the network 29
1.3 Average number of discovered nodes 30
1.3.1 Distribution of the number of nodes discovered by two stations 30
1.3.2 Mean number of discovered nodes with p hosts 31
1.3.3 A Coupon Collector Analogy 33
1.4 Asymptotic results 36
1.4.1 Regular trees 36
1.4.2 Power law trees 39
1.4.3 The growth rate of the exploration process 40
1.4.4 Discussion: The asymptotic profile of tree structures . . . 42
1.5 Degree of discovered nodes 44
1.6 Experimental results 45
1.7 Variance 47
1.7.1 Expression of the variance 47
1.8 Variance asymptotics 51
1.8.1 Variance asymptotics for large n 51
1.8.2 Variance asymptotics for small λ 53

1.8.3 Examples . . . 54
1.9 Conclusion . . . 55

2 Galton-Watson trees **57**
2.1 Introduction . . . 57
2.2 Notation and definitions . . . 57
2.2.1 Assumptions and notation . . . 57
2.3 Asymptotic expansion in the uniform case . . . 59
2.3.1 The expansion of the first moment of $\mathbf{R}_N$ . . . 59
2.3.2 The variance of $\mathbf{R}_N$ . . . 61
2.4 Asymptotic expansion in the depth biased case . . . 64
2.5 Appendix . . . 66
2.6 Conclusion . . . 68

3 Détection des grands flots **69**
3.1 Introduction . . . 69
3.2 Filtre de Bloom . . . 70
3.3 Filtre de comptage . . . 73
3.4 Filtre de Varghese . . . 74
3.4.1 Description de l'algorithme . . . 74
3.4.2 Incrémentation conservative . . . 74
3.5 Filtres de Bloom parallèles . . . 75
3.5.1 Présentation de l'algorithme . . . 75
3.5.2 Calcul du nombre de faux positifs . . . 77
3.5.3 FBP avec effacement total . . . 82
3.5.4 FBP avec effacement progressif . . . 87
3.5.5 FBP avec effacement progressif et filtre virtuel . . . 88
3.5.6 Apport du filtre virtuel . . . 88
3.5.7 Résultats . . . 88
3.6 Analyse . . . 89
3.6.1 Effet du seuil de remplissage . . . 89
3.7 Les éléphants marquent leurs cases . . . 98
3.8 Conclusion . . . 100

4 Détection des grands flots par échantillonnage **103**
4.1 Échantillonnage . . . 103
4.1.1 Cas de la distribution géométrique . . . 106
4.1.2 Cas de la distribution Pareto . . . 106
4.1.3 Cas de la distribution Weibull . . . 107
4.1.4 Simulations . . . 112
4.2 Conclusion . . . 114

5 Conclusion générale **115**

6 ANNEXE : Introduction à la Méthode de Stein-Chen **117**

7 ANNEXE : Résumé du chapitre sur la topologie 123
7.1 Introduction . . . 123
7.2 Notations et définitions . . . 124
7.3 Nombre moyen de noeuds découverts . . . 125
7.4 Résultats asymptotiques . . . 126
7.5 Distribution du degré des noeuds découverts . . . 127
7.6 Variance . . . 127
7.7 Asymptotiques de la variance . . . 128
7.7.1 Asymptotique de la variance pour N grand . . . 128
7.7.2 Asymptotique de la variance pour λ petit . . . 128
7.8 Conclusion . . . 128

8 ANNEXE : Résumé du chapitre sur les arbres Galton Watson 129
8.1 Introduction . . . 129
8.2 Hypothèses et notations . . . 129
8.3 Développement asymptotique dans la cas uniforme . . . 130
8.3.1 Comportement asymptotique de la moyenne . . . 130
8.3.2 La variance de $\mathbf{R}_N$ . . . 131
8.4 Développement asymptotique dans la cas biaisé . . . 131
8.5 Conclusion . . . 132

Introduction

0.1 Vue générale

Cette thèse s'inscrit dans le cadre de la métrologie du trafic et de l'architecture de l'Internet. Le réseau Internet s'est agrandi de façon phénoménale et aléatoire. La connaissance de la topologie de l'Internet et de son modèle d'évolution permettent de maîtriser le comportement de certains protocoles comme ceux du routage pour choisir les chemins de coût minimal entre les noeuds ou bien dans le cas des services multicast afin de bien placer les serveurs de contenu et de réduire la bande passante utilisée. De plus, la prolifération de nouvelles applications telles que le Pair à Pair et l'augmentation continue des capacités des liens dans le réseau ont provoqué une augmentation significative des volumes de données. Ainsi, l'émergence et la montée en charge qu'a connu Internet au cours des trois dernières décennies appellent de nouveaux modèles de trafic adaptés à ces nouveaux services. En effet, jusqu'à présent, les modèles de prédilection utilisés dans l'évaluation des performance des réseaux s'appuyaient sur l'utilisation de la théorie des files d'attente et sur l'hypothèse que les processus d'arrivée des sessions sont poissonniens. Ces méthodes essayaient d'étendre des résultats connus pour les réseaux télécoms comme la formule d'Erlang au monde de l'Internet. Malheureusement, l'hypothèse des arrivées poissonniennes a été remise en question par des études qui montrent que certains trafics de données s'écartent très sensiblement du modèle poissonnien et manifestent plutôt des propriétés de dépendances longues et une sporadicité élevée. Ces études montrent que de tels trafics peuvent être modélisés en superposant plusieurs sources de type On/Off. Cette constatation induit une répercussion majeure sur le dimensionnement des capacités des mémoires des routeurs IP. En effet, avec l'hypothèse de sources On/Off, les niveaux d'occupation des tampons décroissent de façon polynomiale, tandis que dans le cas des modèles poissonniens ils décroissent exponentiellement.

Pour la majorité des utilisateurs, Internet est une boîte noire qui permet la connectivité des applications sans se soucier ni de combien de paquets sont générés ni comment ceux-ci sont acheminés à travers le réseau. Ce sont des protocoles comme IP (adressage des noeuds et routage dans Internet) et TCP (fiabilité de la délivrance des paquets) qui se chargent de ces détails. Un utilisateur final s'intéresse seulement au temps de chargement d'une page web ou de téléchargement d'un fichier, etc. Les mesures de trafic permettent de savoir ce qui ne fonctionne pas correctement dans le réseau et d'avoir une vision plus détaillée sur le fonctionnement de l'Internet, des différentes applications et aussi du

comportement des utilisateurs en stockant des informations telles que le temps d'émission des paquets, les adresses sources et destinations et même le contenu parfois. Les mesures de trafic sont essentielles pour au moins trois raisons:

- Résolution des problèmes du réseau: Les noeuds du réseau ne sont pas infaillibles. Une panne d'un lien ou d'un noeud ou une attaque de type déni de service peut engendrer le dysfonctionnement de tout le réseau ou une dégradation notable des performances.
- Débuggage de nouveaux protocoles: Avant de lancer de nouveaux protocoles dans les réseaux de production, ceux-ci sont souvent testés préalablement en effectuant une campagne de mesures pour en tester le bon fonctionnement, la conformité aux standards et la compatibilité avec les versions antérieures.
- Caractérisation de la charge: Il s'agit d'extraire des statistiques par application utilisée, par utilisateur ou par flot à des fins de facturation et d'ingénierie du réseau.

Il y a deux façons distinctes d'effectuer des mesures de trafic:

- Au niveau matériel: Il s'agit d'utiliser un équipement spécialement fabriqué pour la collecte et l'analyse des données. Ces sondes sont chères mais très rapides et permettent d'effectuer la collecte et l'analyse en temps réel sur des liens de coeur du réseau très chargés.
- Au niveau logiciel: Dans ce cas, des modifications logicielles sont apportées sur le noyau des sondes pour leur permettre d'effectuer la capture. Tcpdump rentre dans cette catégorie. Un autre exemple est l'utilisation des fichiers de log des serveurs et proxy web qui enregistrent les requêtes sur des pages web, les url demandées, les temps d'accès et les adresses des clients. Cette approche est nettement moins coûteuse que la première mais ne permet pas d'obtenir le même niveau de performance et de rapidité. C'est pour cette raison que l'étape d'analyse est souvent effectuée hors ligne.

On peut aussi diviser les mesures de trafic en deux catégories selon qu'on intervienne ou non dans le réseau:

- Mesures passives: C'est le cas le plus courant. Dans ce cas, l'instrument de mesure n'intervient pas dans le trafic du réseau en y injectant des paquets.
- Mesures actives: Dans ce cas, des paquets sont injectés dans le réseau par l'instrument de mesure pour en sonder les caractéristiques en calculant leurs délais, gigue, pertes... Des outils tels que ping, traceroute ou pathchar font partie de cette catégorie.

0.2 Topologie de l'Internet

Au premier abord, le réseau Internet peut être perçu comme chaotique et aléatoire, tant il croît de façon phénoménale. L'analyse de la topologie d'Internet

a suscité une vive attention durant les dernières années. Ainsi, les recherches récentes se sont intéressées au graphe d'Internet, s'il existe des propriétés intrinsèques invariables ou variant peu au cours du temps. Les chercheurs s'intéressent plus particulièrement au modèle d'évolution d'Internet au cours du temps et aussi à la possibilité de générer par simulation des graphes semblables à ceux d'Internet.

L'influence de la topologie de l'Internet sur une grande variété de protocoles réseau est fondamentale mais elle reste encore mal comprise. Les performances des applications et protocoles, surtout ceux du routage, robustesse du réseau sous attaque et l'ingénierie du trafic dépendent étroitement de la topologie du réseau. Par exemple, la connaissance et la compréhension de la topologie Internet peuvent servir pour la conception et le développement de nouveaux protocoles et algorithmes plus efficaces tirant profit de certaines propriétés topologiques. Elle aide aussi pour synthétiser des graphes de simulation plus réalistes et à l'estimation de paramètres utiles pour l'analyse de protocoles et la spéculation sur l'évolution future d'Internet. La modélisation de la topologie Internet est un sujet de recherche encore très ouvert malgré la grande attention qu'elle a suscité dernièrement. Plusieurs générateurs de graphes Internet ont été proposés mais la sélection de nombreux paramètres est laissée à l'intuition et l'expérience de l'expérimentateur. Aussi, ces générateurs échouent souvent à reproduire toutes les métriques importantes de l'Internet à la fois. De plus, à chaque fois qu'une nouvelle métrique s'avère importante, il faut rechercher d'autres générateurs. Afin de remédier à ce problème, on peut citer au moins deux solutions. La première consiste à découvrir les lois régissant l'évolution d'Internet ce qui permet de mettre en évidence certaines des propriétés topologiques. Ceci est un problème de recherche ouvert surtout que les forces qui régissent l'évolution d'Internet peuvent évoluer au cours du temps. Une méthode alternative consiste à se focaliser sur les dépendances entre toutes ces métriques et n'en retenir que celles qui engendrent les autres. Par exemple, se contenter de la corrélation des degrés des noeuds voisins est suffisant pour refléter la majorité des autres propriétés.

Le réseau Internet est divisé en plusieurs domaines inter-connectés entre eux. Ces domaines appelés systèmes autonomes (AS) correspondent à un groupe de réseaux gérés par un administrateur commun et peuvent contenir plusieurs routeurs. Chaque système autonome possède un numéro identifiant sur 16 bits délivré par l'IANA (Internet Assigned Numbers Authority) ou ses délégations. Ces numéros sont ainsi distribués aux opérateurs de télécommunications, aux fournisseurs d'accès Internet ou aux multinationales. Afin d'acheminer des routes entre les différents systèmes autonomes, les fournisseurs d'accès Internet utilisent des équipements utilisant le protocole BGP (Border Gateway Protocol) pour échanger leurs tables de routage et leurs communications lorsqu'une action nécessite l'établissement d'une connexion entre deux systèmes distants et autonomes. Donc, la topologie Internet peut être étudiée à deux niveaux de granularité distincts.

- Au niveau macroscopique des systèmes autonomes, dans ce cas chaque système autonome est représenté par un seul noeud et chaque lien représente

une interconnexion entre deux systèmes autonomes.

- Ou bien au niveau microscopique des routeurs, chaque routeur est représenté par un noeud.

0.2.1 Topologie au niveau des systèmes autonomes

Plusieurs études récentes reconstituent la topologie Internet au niveau des systèmes autonomes à partir des informations contenues dans les tables de routage BGP, puisque celles-ci contiennent des informations de routage détaillées au niveau AS pour chaque réseau ou préfixe de destination. Le défaut de cette méthode est qu'elle ne permet pas d'avoir des détails supplémentaires comme le nombre de liens entre deux systèmes autonomes qui donne des informations sur les liens d'interconnexion entre systèmes autonomes. Aussi, à cause de l'augmentation de la connectivité de l'Internet, certains routeurs agrègent plusieurs systèmes autonomes qui ont des adresses contiguës au sein d'un seul AS plus grand dans le but de réduire la taille des tables BGP. Ceci a pour effet que plusieurs systèmes autonomes ne sont pas découverts. En plus, à cause de certaines restrictions de routage, quelques AS restent cachés. Chang *et al* [7] proposent d'inférer la topologie au niveau des systèmes autonomes à partir de celle du niveau routeurs. Cette méthode offre l'avantage de distinguer les routeurs de bordure des AS qui constituent les points de liaison entre AS, en observant que ceux-ci disposent de plusieurs interfaces chacune ayant une adresse différente appartenant éventuellement à un AS différent. La résolution des interfaces d'un routeur donné se fait par le biais du logiciel Mercator [25]. Celui-ci envoie des requêtes UDP à une adresse d'un routeur donné qui déclenche le retour d'un message de contôle (ICMP) contenant comme adresse source une interface du routeur en question. Si cette interface est différente de l'adresse destination utilisée initialement, alors il s'agit d'un alias de ce routeur. Pour obtenir la liste complète des alias d'un routeur, il suffit de lui envoyer des requêtes pendant une longue durée en espérant qu'il utilise des interfaces différentes pour renvoyer les messages ICMP (lorsque le routage change). Si cette procédure est utilisée de façon abusive ou bien à cause des restrictions des pare-feu et des politiques de routage, certaines requêtes sont considérées comme des tentatives d'intrusion et sont tout simplement ignorées. L'idée de base proposée par Chang *et al* est d'établir une table de correspondance entre les interfaces des routeurs et leurs numéros d'AS, ce qui permet de déduire la topologie AS à partir de celle des routeurs. Une première table de correspondance est obtenue en se basant sur les informations des tables de routage BGP et IRR (Internet Routing Registry), en cas d'informations contradictoires entre ces deux sources, la priorité est donnée aux tables BGP qui sont mises à jour plus fréquemment. Néanmoins, il reste une proportion non négligeable d'adresses dont on ne connaît pas l'AS correspondant (presque 11%). Dans ce cas, les auteurs ont proposé plusieurs régles et heuristiques pour régler ce problème:

- Si un routeur de bordure a une interface correspondant à un AS X, alors ce routeur appartient soit à X soit à un autre AS qui a un lien de peering avec X

- Si un routeur de bordure a un voisin correspondant à un AS X, alors ce routeur appartient soit à X soit à un autre AS qui a un lien de peering avec X

En appliquant ces deux règles, ils ont ramené de 11% à 5.6% le pourcentage d'AS inconnus puis ils ont proposé une heuristique qui interpole l'AS des routeurs voisins.

Govindan et Reddy ont étudié la croissance de la topologie inter-domaines d'Internet entre 1994 et 1995 [24], ils ont observé que 75% des noeuds ont moins de deux interconnexions et que la connectivité croît au cours du temps. Panisot et Grad ont étudié la topologie Internet au niveau routeur au cours de la même période [36]. Les distances qu'ils ont relevées sont approximativement deux fois plus grandes par rapport à celles de la topologie inter-domaines, ce qui implique qu'un système autonome contient en moyenne deux routeurs au cours de cette période.

Tangmunarunkit *et al* [46] ont utilisé la topologie de l'Internet au niveau des systèmes autonomes et des routeurs pour quantifier la dégradation du routage en terme de longueur des chemins ou nombre de routeurs intermédiaires entre sources et destinations en présence de politiques de routages entre AS. En effet, à l'intérieur d'un même domaine, le routage se base sur le plus court chemin entre les noeuds, mais à cause de l'existence de quelque politiques de routages hiérarchiques et d'accords commerciaux inter-domaines, les AS peuvent choisir de façon indépendante les informations de routage de leurs voisins ce qui peut rallonger les chemins obtenus.

0.2.2 Topologie au niveau des routeurs

Plusieurs projets ont développé des outils pour obtenir la topologie de l'Internet au niveau des routeurs comme Mercator [25] et Rocketfuel [45]. L'idée de base est qu'une ou plusieurs machines sources envoient des messages traceroute vers un ensemble de machines destinataires en incrémentant graduellement le champ TTL du paquet (nombre maximum de routeurs que le message peut traverser), ce dernier est décrémenté dans chaque routeur le long du chemin entre la source et la destination. Lorsqu'un routeur s'aperçoit que le TTL est nul, il renvoie un message ICMP à la source contenant son adresse IP. Ceci, permet d'avoir toutes les adresses IP des routeurs intermédiaires entre les différentes sources et destinations. A la fin, les vues obtenues par les différentes sources sont fusionnées et les alias correspondants au même routeur sont identifiés.

0.2.3 Quelques propriétés de l'Internet

Le réseau Internet manifeste plusieurs propriétés et métriques fondamentales. Par exemple, il a été montré empiriquement que plusieurs paramètres importants caractérisant Internet suivent une distribution en puissance. Les lois en puissance sont largement utilisées dans les réseaux de télécommunication pour en décrire le trafic. En effet, les propriétés d'auto-similarité et de queue lourde se manifestant dans de telles réseaux sont étroitement liées aux distributions en puissance. Une variable X est dite à queue lourde si $P(X > x) =$

$k^a x^{-a} L(x)$ où $k \in R^+$ et $L(x)$ est une fonction à variation lente vérifiant: $\lim_{t\to\infty} L(tx)/L(x) = 1,\ \forall x > 0$. En particulier, une loi en puissance manifeste cette propriété de queue lourde. De plus, Leland *et al* [30] ont montré la nature auto-similaire du trafic LAN et Paxson et Floyd [37] ont décrit le trafic LAN comme une agrégation de contributions de sources ON-OFF à queues lourdes. Willinger et al [49] ont montré que l'auto-similarité au niveau macroscopique dans les réseaux LAN et WAN est le résultat des sources à queues lourdes au niveau microscopique. Faloutsos *et al* [16] se sont basés sur trois topologies Internet obtenus entre Novembre 1997 et Décembre 1998. Ils ont observé que des lois en puissance approchent à plus de 98% les distributions de certains paramètres. Certaines des puissances de ces lois varient très peu au cours du temps quand d'autres croissent de presque 10% au cours de la période entre Novembre 1997 et Décembre 1998 où la taille du réseau a augmenté substentiellement (45%). Ils ont introduit une nouvelle métrique caractérisant la densité et proposé une autre loi en puissance pour la décrire. Comme ces lois en puissance apparaissent naturellement dans d'autres contextes et qu'elles sont en accord avec les résultats expérimentaux, les auteurs sont convaincus que les observations faites sur la période susmentionnée resteront valables pendant un certain temps dans le futur et ne sont pas des simples coïncidences. Plus particulièrement, ils considèrent trois paramètres principaux: le degré des noeuds, le rang des noeuds et les valeurs propres du graphe du réseau Internet. Le degré d'un noeud désigne le nombre de voisins de ce noeud et sert donc à caractériser le degré de connectivité de celui-ci. La première observation faite par les auteurs concerne la relation entre le degré des noeuds et leurs rangs. Les rangs des noeuds sont obtenus en classant ceux-ci par ordre décroissant des degrés. Cette relation s'écrit $d_v \equiv r_v^R$ où d_v est le degé du noeud v, r_v son rang et R une constante. Plus précisément: $d_v = r_v^R / N^R$, car le degré minimum est égal à 1 et N est le nombre total de noeuds. Cette relation permet de déduire aussi le nombre total E de liens dans le réseau puisque celui-ci est la demi-somme de tous les degrés des noeuds. Ceci nous permet d'estimer à partir du nombre total de noeuds et de la puissance R le nombre de liens E. La deuxième loi concerne la fréquence des degrés. La fréquence d'un degré est le nombre de noeuds ayant le degré d. Celle-ci suit une loi en puissance: $f_d \equiv d^\epsilon$ avec ϵ une constante négative qui change lentement au cours du temps et prend une valeur typique autour de -2.48. Ainsi, on peut tester le réalisme d'un réseau en testant cette loi et en comparant la puissance obtenue par rapport aux valeurs normales. La troisième loi concerne les valeurs propres du graphe Internet. On sait que celles-ci sont étroitement liés à des propriétés topologiques tels que le diamètre, le nombre de liens, le nombre de marches d'une certaine longueur entre les sommets du graphe. Les auteurs ont observé qu'il existe une loi en puissance reliant ces valeurs propres à leurs ordres dans le sens décroissant: $\lambda_i \equiv i^\epsilon$. L'une des propriétés les plus utilisée dans la conception d'algorithmes de routage ou protocoles de multicast et aussi dans les systèmes P2P est celle du petit monde. Le graphe Internet est très connecté de telle sorte que presque tous les noeuds sont très proches en distance et que le diamètre d'Internet (distance maximale entre deux noeuds) est petit. Afin de quantifier cette propriété, on utilise en général le diamètre du graphe Internet ou plus précisément une fonction $P(h)$ qui renvoie le nombre total de paires

de noeuds à distance h. Lorsque h est petit (négligeable devant le diamètre), Faloutsos *et al* [16] montrent que cette fonction $P(h)$ suit elle aussi une loi en puissance en fonction de la distance h : $P(h) \equiv h^H$. D'autre part, il est évident que, lorsque h est plus grand que le diamètre, on a $P(h) = N^2$ puisque toutes les paires de noeuds sont à distance inférieure ou égale à ce diamètre. Cette observation est très utile dans plusieurs applications. Par exemple, elle permet de limiter la propagation de messages de diffusion ou de recherche de fichiers dans certains protocoles P2P comme Emule. De plus, le paramètre H caractérise bien la topologie du réseau, par exemple pour un réseau sous forme d'anneau $H = 1$ tandis que pour une grille $H = 2$. Ces différentes lois en puissance décrivent la topologie Internet en un seul paramètre, les travaux antérieurs s'appuyaient sur l'utilisation des valeurs moyennes, minimales et maximales, ceci fausse la topologie puisqu'il a été noté sur une topologie Internet que 85% des noeuds ont des degrés inférieurs à la moyenne. Dans [6] les auteurs cherchent des propriétés de la topologie de l'arbre de distribution du multicast. En effet, l'efficacité du multicast par rapport à l'unicast depend non seulement du nombre de récepteurs mais surtout de la topologie et de l'emplacement des serveurs de contenu. Cette efficacité est quantifiée en général par le rapport du nombre de liens utilisés sur celui du cas unicast où un message est envoyé individuellement à chaque destinataire. Les auteurs ont montré en particulier une autre loi en puissance reliant le nombre de noeuds dans l'arbre M et le nombre de destinataires N: $M \equiv N^\epsilon$ où ϵ proche de 0.66. Le graphe d'Internet présente aussi la propriété de petit monde, ceci signifie que pratiquement tous les noeuds du réseau sont des voisins les uns aux autres et que tout couple de noeuds peuvent être reliés en un nombre petit de sauts intermédiaires.

0.2.4 Contributions de la thèse sur la topologie

La majorité des projets de recherche qui se sont intéressés à la topologie de l'Internet se sont attelés au développement d'outils et d'algorithmes pour la découverte de la topologie sans en étudier l'efficacité. Ces études se basent souvent sur l'utilisation des informations contenues dans les tables de routage BGP ou bien sur l'utilisation de traceroute. Avec traceroute, une source identifie la liste des routeurs intermédiaires la séparant d'un ensemble d'adresses destinations choisies (des millions d'adresses). La vue obtenue par une seule source ne peut identifier qu'un petit arbre du graphe de l'Internet dont la racine représente cette adresse source. C'est pour cette raison que plusieurs sources (de l'ordre de la centaine) sont déployées et les différentes vues sont fusionnées pour reconstituer une image plus globale. Les points les plus abordés sont la réduction des redondances contenues dans les différentes vues et la résolution des alias désignant un même routeur. La vue obtenue pourrait être incomplète et ne refléter qu'une partie du graphe de l'Internet. C'est pour cela que nous avons développé un modèle stochastique pour l'étude de l'efficacité de traceroute dans la découverte de la topologie d'Internet. Nous avons choisi de considérer une topologie simple sous forme d'arbre qui reflète le grand degré d'hiérarchisation de l'Internet et nous nous sommes intéressés surtout au nombre total de noeuds qui collaborent dans ce processus de découverte ainsi qu'à l'influence de la topologie de

l'arbre considéré. Plus particulièrement, on a établi des formules analytiques qui donnent la moyenne et la variance de la proportion de noeuds découverts par rapport au nombre total de noeuds N lorsque le nombre de noeuds p intervenant dans la découverte est proportionnel à N ($p = \lambda N$). Puis on a effectué des approximations asymptotiques quand la taille du réseau devient très grande et aussi quand le nombre de noeuds participant dans la découverte est très petit, ce qui est le cas en pratique. Ces résultats montrent que l'efficacité de traceroute dans la découverte de la topologie dépend étroitement de la forme de l'arbre. Afin de caractériser cette dépendance, on a établi des quantités qui décrivent le profil de l'arbre. Plus précisément on montre que, plus l'arbre est dense aux niveaux inférieurs (proche des feuilles), plus rapide sera l'exploration. On s'est intéressé aussi à certaines propriétés du sous-arbre découvert. Plus particulièrement, on a établi la distribution des degrés de cet arbre. Dans un second temps, on a considéré des arbres aléatoires de type Galton-Watson où les degrés des différents noeuds sont tirés de façon indépendante suivant une distribution fixée. On a montré qu'au premier ordre, cet arbre aléatoire est équivalent à un arbre déterministe ou le degré est constant et égal à la moyenne des degrés des noeuds.

0.3 Statistiques des flots

Un flot désigne un transfert de données contenant plusieurs paquets ayant en commun un identifiant. Cet identifiant du flot peut être caractérisé de plusieurs façons en fonction du problème traité. Par exemple, si on se place dans le contexte de détection d'attaques dans le réseau, et qu'on veut repérer les cibles d'attaques, alors il vaut mieux définir un flot par l'adresse destination seulement. Le plus souvent, un flot est identifié par les champs suivants:

- Adresse et port source: Adresse IP et numéro de port de la machine émettrice.
- Adresse et port destination: Adresse IP et numéro de port de la machine destinataire.
- Numéro de protocole: Désigne le protocole utilisé.

Les mesures du trafic Internet sont très utiles à plusieurs égards par exemple pour la facturation des clients en fonction de la consommation, la détection d'anomalies et d'attaques mais aussi pour l'ingénierie et la gestion du trafic.

Les mesures effectuées sur plusieurs types de réseaux (y compris l'Internet) montrent que les statistiques des tailles des flots manifestent un comportement de queue lourde. Cette caractéristique a pour conséquence qu'une petite fraction des flots génère la majorité du volume du trafic, par exemple, il a été observé sur plusieurs traces Internet que moins de 0.02% des flots contribuent à plus de 60% du volume total du traffic. Ceci a donné naissance à la dichotomie entre flots souris et éléphants. Les flots souris sont des transferts de données courts et rapides. De ce fait, ils ne dépassent pas l'étape du slow start de TCP et ils sont donc indifférents au mécanisme de contrôle de TCP. Ils sont très nombreux

en nombre de flots mais leur contribution en volume est négligeable par rapport à celle des éléphants. Ces derniers sont des gros transfert de données, ils durent suffisamment longtemps et sont donc assujettis à la boucle de contrôle de TCP. D'un point de vue qualitatif, lorsque plusieurs éléphants empruntent le même goulot d'étranglement, ils finissent par se partager le débit de façon équitable (propriété min-max de TCP). Le seuil de 20 paquets fixé pour séparer les flots éléphants des souris est arbitraire. Comme les flots éléphants sont très volumineux, leur impact sur les performance du réseau est très significatif. Ceci montre l'importance de la détection des éléphants et la connaissance de leurs statistiques. En identifiant rapidement les éléphants, un opérateur réseau peut facilement entreprendre les actions nécessaires à l'envers du réseau ou source d'où émane ce trafic. Afin d'identifier les grands flots, les approches traditionnelles consistaient à collecter tous les paquets puis à extraire les statistiques des flots. Cette méthode souffre du problème du passage à l'échelle. En effet, pour un lien très haut débit, mesurer directement les statistiques des flots à la volée est hors de portée (puissance CPU limitée, contrainte de capacité de stockage, de mémoire et de temps d'accès). A titre d'exemple, un seul lien OC-48 à utilisation maximale peut générer approximativement 100Go d'en-têtes de paquets par heure. Cette méthode s'avère par ailleurs inutile dans plusieurs cas pratiques. Par exemple pour détecter les attaques dans un réseau, il n'est pas nécessaire de créer des enregistrements pour tous les flots puisque seulement une petite fraction des sources est responsable de ces attaques, et il est plus opportun de suivre seulement les adresses suspectes qui dépassent un certain critère (débit, sporadicité...).

0.3.1 Statistiques des flots avec traces complètes

Dans le but de remédier à ce problème, plusieurs techniques ont été utilisées récemment dans le contexte d'Internet pour estimer le nombre total de flots, leurs statistiques et détecter les grands flots (heavy hitters) à la volée et en une seule passe. Elle se basent en général sur l'utilisation de tables de hachages, de filtres de Bloom et sur l'utilisation de techniques d'estimation statistique comme l'algorithme de maximum de vraisemblance. Une table de hachage est une structure de données simple qui consiste en un tableau. On utilise une fonction de hachage pour y insérer les flots et en cas de collision entre flots différents, on crée une liste chaînée enregistrant tous les éléments correspondant à la case correspondante. Dans [44], une table de hachage est utilisée conjointement avec un filtre de comptage dans l'objectif d'accélérer la recherche de flots dans la table en réduisant la taille des listes en cas de collisions et en maintenant une seule copie de chaque flot inséré. Flajolet et Martin [21] proposent un algorithme de comptage probabiliste qui fournit un taux d'erreur de l'ordre de $0.78/\sqrt{m}$ où m est la mémoire utilisée. L'idée est de hacher tous les flots sur L bits et de se baser sur la position R du 0 le plus à gauche dans la représentation binaire de ces valeurs de hachage comme estimateur du nombre total de flots N. En effet, $R \sim \ln_2(\phi N)$. Cet estimateur n'est pas bon puisque augmenter ou diminuer la valeur de R de 1 revient à multiplier ou diviser l'estimation par deux. La précision de cet estimateur est nettement améliorée en répétant la même

procédure m fois et de prendre ensuite la moyenne. Wegman propose un autre algorithme de comptage probabiliste [19] appelé comptage adaptatif. Celui-ci utilise une idée similaire et plus simple. En effet, il utilise comme dans le cas du comptage probabiliste une fonction de hachage qui hache les flots sur L bits. Il fixe aussi des ensembles $P_0, P_1, P_2 \cdots$ où $P_0 \supset P_1 \supset P_2 \cdots$ de telle sorte que pour une valeur de hachage obtenue x on a: $\mathbb{P}(x \in P_j) = 1/2^j$. Dans la pratique, P_j est souvent l'ensemble des valeurs de hachage qui commencent par au moins j zéros consécutifs. L'algorithme proposé stocke une liste d'au maximum m valeurs de hachage et un entier δ qui désigne la profondeur. Initialement la liste est vide et la profondeur est nulle. La liste est remplie progressivement et dès qu'il y a m éléments différents, la profondeur δ est incrémentée. En voici une petite description algorithmique:

COMPTAGE ADAPTATIF:

— INITIALEMENT.
la liste vide et δ nul

— FLOT F ARRIVE.
Si F n'est pas dans la liste et si $F \in P_\delta$
$\rightarrow$ Insérer F dans la liste
Si la liste est pleine (m éléments)
$\rightarrow$ Garder dans la liste seulement les éléments appartenant à $P_{\delta+1}$
$\rightarrow \delta++$

A la fin, la liste contient l éléments et $2^\delta l$ est utilisé comme estimateur du nombre total de flots. L'erreur relative de cet estimateur est deux fois plus élevé mais il est non biaisé à l'opposé de celui proposé par Flajolet et Martin [21] qui ne l'est qu'asymptotiquement. Varghese *et al* [15] utilisent des tables (bitmaps) pour compter le nombre total de flots. Dans ce cas, l'erreur est proportionnelle à $1/m$ au lieu de $1/\sqrt{m}$. L'idée consiste à utiliser le remplissage du bitmap. Ils proposent ensuite d'utiliser un bitmap virtuel qui étend virtuellement la mémoire utilisée puis d'utiliser plusieurs bitmaps virtuels ayant des résolutions différentes: Basses résolutions lorsqu'il y a peu de flots et grandes résolutions quand il y a énormément de flots. Avec cette dernière solution, le choix du filtre virtuel adéquat se fait en fonction des différents taux de remplissage et l'insertion d'un flot se fait dans un seul filtre virtuel. Les filtres de Bloom [4] sont des structures de données très simples qui permettent de savoir si un élément appartient à un ensemble ou non en effectuant un nombre limité d'opérations et en utilisant un nombre réduit de bits par élément. Leur utilisation s'est vite répandue dans le contexte du Web, gestion de cache, routage... Varghese *et al* [14] généralisent le principe du filtre de Bloom pour estimer la multiplicité d'un élément dans un ensemble (par exemple les statistiques des flots vus par un routeur). Ils utilisent ainsi des compteurs qui sont incrémentés pour chaque occurrence d'un élément et afin de réduire l'effet des collisions, ils utilisent comme dans les filtres de Bloom k fonctions de hachage indépendantes. Ils ont proposé aussi une autre méthode basée sur l'échantillonnage (sample and hold) où chaque paquet est pris avec probabilité p et à chaque fois qu'un paquet est échantillonné, tous

les paquets ultérieurs de ce même flot le sont aussi. Ceci permet d'avoir une bonne estimation de la taille des grands flots. Dans [28], Kumar *et al* proposent une méthode se basant sur l'utilisation de l filtres de Bloom pour estimer la fréquence d'un élément. Pour ce faire, ils utilisent k fonctions de hachage dans chaque filtre de Bloom. A la réception d'un flot, celui-ci est inséré dans l'un de ces l filtres choisi au hasard en positionnant les cases correspondantes à 1. Afin d'estimer la fréquence de ce flot, on commence par compter le nombre θ de filtres de Bloom où les cases correspondantes à ce flot sont positionnées à 1, puis en utilisant un estimateur comme celui de maximum de vraisemblance, on fournit des estimations des fréquences des flots. L'ensemble des l filtres de Bloom sera désigné par la suite par l'abréviation SCBF. Dans le but d'accélérer l'algorithme, l'étape d'estimation statistique peut s'effectuer hors ligne en établissant des correspondances entre chaque valeur de θ entre 0 et l et la fréquence correspondante. Comme la propriété de queue lourde caractérisant la distribution des tailles des flots dans le trafic Internet implique que les fréquences de certains éléments peuvent être très grandes et qu'on sait par le problème de collecteur de coupons [3, 20] que presque tous les l filtres seront remplis par un flot au bout de $l \ln l$ insertions de celui-ci. En pratique pour une valeur de l égale à 32 ($l \ln l = 111$), il n'est pas possible de distinguer si un élément a été inséré 200 ou 400 fois. Ceci rend la distinction entre les fréquences dépassant $l \ln l$ impossible. Augmenter la valeur de l pourrait résoudre le problème mais dans ce cas la capacité de stockage est gaspillée inutilement, en plus la complexité de l'algorithme augmente pendant l'étape d'estimation où il faut vérifier si le flot en cours appartient à chacun des l filtres de Bloom. Pour remédier à ce problème, les auteurs introduisent aussi une technique de multi-résolution. Cette fois-ci, r ($r \geq 2$) SCBF sont utilisés en parallèle de telle façon à couvrir l'ensemble des valeurs possibles de multiplicités. Au i ème SCBF est associée une probabilité d'insertion p_i ($p_1 > p_2 > \cdots > p_r$): Un flot est inséré dans chacun des r filtres avec la probabilité associée à celui-ci. Les SCBF de petite résolution (ayant une probabilité petite) servent à estimer les statistiques des grands flots tandis que ceux ayant une grande résolution (grande probabilité) sont utilisés pour les petits flots. Lorsqu'on veut estimer la multiplicité d'un flot F, les différents SCBF fournissent un vecteur de r composantes $(\theta_1, \theta_2, \cdots, \theta_r)$ où θ_i désigne le nombre de filtres remplis au i ème SCBF. En général les valeurs θ_i sont différentes à cause des probabilités p_i différentes. Estimer la multiplicité du flot F à partir de l'observable $(\theta_1, \theta_2, \cdots, \theta_r)$ quand r est grand est une tâche ardue. Les auteurs proposent de choisir parmi les r SCBF, celui qui est le plus pertinent en se basant sur les valeurs des θ_i. Par exemple il est clair que si l'une des θ_i est très proche de l, alors le SCBF correspondant n'est pas pertinent et ne doit pas être pris en compte lors de l'estimation puisque tout ce qu'on peut en déduire est que la fréquence est très grande pour être estimée par ce SCBF. Le filtre le plus pertinent est celui qui minimise l'erreur relative incrémentale: celle-ci est définie comme le rapport entre le nombre moyen d'insertions pour passer de θ filtres remplis à $\theta + 1$ parmi les l filtres au total et le nombre moyen d'insertions pour atteindre θ filtres remplis: $\frac{l}{l-\theta} / \sum_{i=0}^{\theta-1} \frac{l}{l-i}$. Les auteurs utilisent les observations correspondant à ce filtre pertinent et ces voisins immédiats pour produire une estimation de la fréquence des flots. Ainsi,

ils établissent un tableau d'estimations qui pour chaque triplet de valeurs de filtres remplis correspond une estimation de la multiplicité.

Kumar *et al* [27] ont développé une méthode à base de table de hachage pour estimer les statistiques des flots. Cet algorithme recourt à l'estimation statistique. A l'arrivée de chaque paquet, son compteur correspondant dans la table de hachage est incrémenté. En l'absence de collisions entre flots, les statistiques des fréquences des compteurs correspondent exactement aux statistiques recherchées, mais comme la taille de la table doit être petite et que les fonctions de hachage ne sont pas parfaites, il y aura beaucoup de collisions. Ils utilisent donc la méthode ME pour essayer de déduire comment les collisions s'effectuent. En prenant initialement les statistiques des flots égales aux fréquences des compteurs, ils obtiennent une nouvelle estimation des statistiques en détaillant pour chaque compteur les collisions possibles qui ont produit la valeur correspondante: par exemple un compteur égal à 2 pourrait résulter de la collision de deux flots de taille 1 (1+1), ou bien à un seul flot de taille 2. Afin de réduire le nombre de cas possibles, ils limitent le nombre de collisions par case. En réitérant cette étape, l'algorithme converge vers l'estimation des statistiques.

0.3.2 Statistiques des flots par échantillonnage

La solution de l'échantillonnage des paquets a suscité elle aussi un vif intérêt récemment parmi les industriels et la communauté des chercheurs. Ainsi, de plus en plus de routeurs embarquent des outils d'échantillonnage comme NetFlow et sFlow leur permettant de diminuer la charge de collecte des statistiques, la bande passante utilisée pour envoyer ces statistiques à un collecteur, la mémoire et l'espace de stockage de ce dernier. Il y a deux grandes catégories d'échantillonnage: échantillonnage par flots: l'objectif dans ce cas est d'échantillonner soit 0 paquets, soit tous les paquets d'un même flot. Cette méthode consiste à utiliser une fonction de hachage qui hache uniformément tous les flots vers l'intervalle $[0, 1]$, ainsi, un paquet est échantillonné si son image par cette fonction de hachage appartient à un sous ensemble prédéfini dans l'intervalle $[0, 1]$. La deuxième catégorie est celle de l'échantillonnage par paquet, elle est divisée elle aussi en deux sous-catégories: échantillonnage probabiliste et dans ce cas, le routeur décide de prendre chaque paquet avec probabilité p (p de l'ordre de 10^{-3} à 10^{-2}) ou bien échantillonnage périodique auquel cas le routeur retient un paquet tous les N paquets vus (N est de l'ordre de 100 à 1000). Cette dernière méthode introduit des corrélations puisqu'aucun des $N-1$ paquets suivant n'est échantillonné. Cet effet est négligeable sur des liens très chargés où les paquets d'un même flot sont espacés par les paquets des autres flots. Cette dernière forme est la plus simple et rapide à implémenter sur un routeur puisqu'elle nécessite le traitement d'un paquet tous les N vus et ne fait intervenir aucun calcul de fonction de hachage, mais tout simplement, l'incrémentation d'un petit compteur cyclique. C'est pour cette raison qu'elle est la plus adoptée en pratique. Ceci ramène le problème à celui de l'identification des éléphants à partir des données échantillonnées. Ce problème est très délicat, car si le taux d'échantillonnage est très faible, il se peut que certains éléphants s'apparentent à des souris ou même disparaissent après échan-

tillonnage si aucun paquet de ces flots n'est échantillonné. De plus, même si une grande partie des souris disparaîssent, comme ceux-ci sont très majoritaires en nombre de flots, beaucoup de flots souris vont persister et constituent un bruit d'estimation qu'il faut éliminer. Dans [12], Duffield *et al* proposent un autre type d'échantillonnage pour la facturation des clients en fonction de leur utilisation effective de la bande passante. Il se base sur le constat que la distribution du nombre de paquets ou d'octets par flot suit une loi à queue lourde, donc l'estimation de la taille totale d'un flot est très sensible à l'inclusion ou l'omission de certains grands paquets, donc, il est préférable d'échantillonner en priorité les grands paquets qui représentent un grand pourcentage de la taille totale d'un flot. Ainsi, Duffield *et al* optent pour une probabilité d'échantillonnage dépendante de la taille et plus particulièrement, croissante en fonction de la taille des paquets. Ils montrent qu'il existe un estimateur non biaisé et optimal pour l'estimation des tailles des flots et dont la variance est plus petite que celle du cas de l'échantillonnage uniforme (qui ne dépend pas de la taille des paquets). Cette méthode nécessite le traitement de chaque paquet pour calculer la probabilité de le retenir en fonction de sa taille.

L'échantillonnage induit des pertes d'informations qu'il est parfois facile de corriger. Par exemple si la taux d'échantillonnage est de $1/N$, alors une estimation satisfaisante du nombre total de paquets contenu dans la trace peut être obtenu en multipliant le nombre de paquets après échantillonage par le facteur N. Si l'échantillonnage est indépendant de la taille des paquets, alors une estimation du volume total de la trace en octets peut être obtenu de la même façon. Néanmoins, il n'est pas facile d'obtenir des caractéristiques plus détaillées comme la distribution des tailles des flots en paquets ou octets. Une approche naïve consisterait à faire correspondre aux flots de taille l la taille Nl avant échantillonnage. Ceci est totalement faux puisqu'un flot de taille l avant échantillonnage pourrait avoir initialement n'importe quelle taille supérieure à l et pas nécessairement Nl paquets. En plus, cette méthode présente plusieurs inconvénients, par exemple les tailles avant échantillonnage sont toutes multiples de N et certains flots risquent de disparaître après échantillonnage. Cette méthode échoue surtout pour déterminer les statistiques des petits flots. Dans [34], Mori *et al* supposent qu'ils connaissent à priori la distribution des tailles des flots et cherchent à repérer les flots éléphants après échantillonnage. Pour ce faire, l'utilisation du théorème de Bayes établit un compromis entre le taux de faux positifs (dûs aux souris qui sont déclarées à tort comme étant des éléphants) et celui des faux négatifs (éléphants non détectés). L'idée consiste à fixer un seuil $\hat{y}$ de telle façon que tout flot qui dépasse ce seuil en nombre de paquets après échantillonnage est déclaré éléphant. Ce seuil doit être choisi judicieusement afin de maintenir le taux de faux positifs inférieur à une borne ϵ tout en minimisant celui des faux négatifs. Si on note $TFP(y)$ (resp $TFN(y)$) le taux de faux positifs pour le seuil y (resp taux de faux négatifs), alors en remarquant que la fonction TFP est décroissante et que TFN est croissante, le seuil $\hat{y}$ est choisi de la façon suivante:

$$\hat{y} = \min_{y}\{y \mid TFP(y) \leq \epsilon\}$$

Afin de caractériser la distribution initiale des flots, les auteurs proposent trois méthodes:

- Par mesure directe sur le lien à partir des données non échantillonnées.
- A partir des données échantillonnées.
- Utilisation des lois en puissance caractéristiques du trafic Internet.

Le grand défaut de cette dernière méthode est qu'elle suppose une connaissance préalable de la distribution des tailles des flots. Ceci n'est pas toujours possible et en plus cette distribution peut changer au cours du temps. Dans [13], Duffield *et al* proposent de retrouver les statistiques initiales en se basant sur des informations supplémentaires contenus dans les entêtes des paquets TCP comme les drapeaux SYN initialisant une connexion TCP et FIN annonçant la fin d'une connexion TCP. Le trafic TCP est majoritaire par rapport aux autres trafics comme UDP. Par exemple sur une trace qu'ils ont considérée, le trafic TCP englobe 76% des flots, 84% des paquets et 95% du volume en octets. Comme certaines connexions TCP peuvent ne pas se terminer par un paquet FIN (comme les attaques de deni de service inondant des cibles par des paquets SYN isolés), il vaut mieux utiliser les drapeaux SYN plutôt que FIN. Duffield *et al* proposent dans un premier lieu des estimateurs pour le nombre total de flots originaux et le nombre total de paquets. Dans un second temps, ils attribuent des poids à des différentes tailles de flots en fonction des statistiques retrouvées après echantillonnage: Par exemple, pour chaque flot SYN de taille j après échantillonnage, comme il contenait au départ un seul paquet SYN, alors on lui attribue initialement une taille égale à $l_j = 1 + N(j-1)$ où N est le facteur d'échantillonnage. Afin de lisser la distribution des flots, on répartit un poids égal à 1 sur différentes tailles de flots centrées autour de l_j et on utilise ensuite la méthode de maximum de vraisemblance pour l'estimation.

Beaucoup de problèmes rencontrés lors de l'évaluation de performance et la gestion des réseaux sont dûs au fait que les changements de conditions dans le réseau sont observés avec un retard. Dans [8], Choi *et al* proposent un mécanisme pour détecter à temps des fluctuations brutales de charge de façon adaptative en ajustant le taux d'échantillonnage en fonction de la charge du lien au cours du temps. Ceci évite l'échantillonnage abusif ou insuffisant. L'idée clé est de diviser le temps en plusieurs unités de longueur Δ fixée en fonction du degré d'adaptabilité voulu. Durant chaque unité de temps, le taux d'échantillonnage est actualisé. Ainsi, plus Δ est petit, plus l'algorithme est réactif aux changements brutaux, mais nécessite davantage de calculs pour mettre à jour le facteur d'échantillonnage. Le grand intérêt de cette méthode est qu'elle établit, sur chaque intervalle Δ, une valeur minimale du taux d'échantillonnage qui garantit que le taux d'erreur de l'estimation du volume total en octets reste inférieur à une borne fixée au préalable. On montre que le taux d'échantillonnage est proportionnel au carré du coefficient S de variance de la distribution des tailles des paquets $S = (\sigma/\mu)^2$ où μ et σ sont, respectivement, la moyenne et l'écart type de la distribution des tailles des paquets pendant l'unité de temps actuelle. Choi *et al* proposent aussi un modèle auto-régressif pour mettre à jour ces coefficients au bout de chaque unité de temps.

0.3.3 Contributions de la thèse sur l'algorithmique des grands flots

Les algorithmes proposés dans la littérature ne sont pas bien adaptés pour l'estimation des statistiques des flots. En général, ces algorithmes donnent une bonne estimation du nombre total de flots mais ne fournissent pas plus de détails sur la distribution des tailles des flots. En effet, ces algorithmes n'étaient pas conçus initialement pour le trafic Internet mais plutôt pour des ensembles de données restreints. Dans cette thèse on a développé un algorithme temps réel qui peut être implémenté sur un routeur pour fournir les statistiques des flots et détecter les grands flots à la volée en utilisant une mémoire optimale (nombre de bits/élément optimal). Cet algorithme consiste en l'utilisation de plusieurs filtres de Bloom parallèles et d'un petit nombre de fonctions de hachage. La mise à jour de ces filtres consiste à calculer les valeurs de hachage d'un flot par ces fonctions, puis à effectuer des recherches et des insertions dans une table. De ce fait, le nombre d'opérations est limité ce qui rend cet algorithme simple à implémenter et à mettre à jour. Comme les différents filtres se saturent au bout d'un certain laps de temps, on a proposé par la suite plusieurs mécanismes adaptatifs pour effacer progressivement les filtres. L'utilisation de filtres parallèles rend cette opération facile à effectuer. Plusieurs politiques d'effacement ont été testées et comparées sur des traces réelles et on a établi analytiquement des comparaisons de performance entre certaines d'entre elles. Par exemple, on a comparé l'effacement brutal à l'effacement progressif et aussi l'effacement régulier (au bout de chaque durée fixée) au cas où celui-ci est contrôlé par le taux de remplissage. On a proposé un deuxième algorithme utilisant des tables de hachage et se basant sur le constat que les éléphants émettent plusieurs paquets au cours de laps de temps courts. L'idée de cet algorithme est de choisir un seuil de sporadicité (nombre de paquets consécutifs émis par un même flot). Si un flot dépasse ce seuil de sporadicité, il est considéré comme un éventuel flot éléphant et mis à part dans une liste de candidats pour être suivi dans le futur. Cette notion de sporadicité sera expliquée par la suite puisqu'on peut imaginer que, sur des liens très chargés, il est très probable que les paquets d'un même flot soient séparés par des paquets d'autres flots. Finalement, on s'est intéressé au problème de comptage et de détection des grands flots quand on dispose uniquement des données échantillonnées. On a montré que, lorsque la distribution initiale des tailles des flots suit une loi Pareto ou Weibull très lourde ($\beta < 1/2$), ce qui est généralement le cas, on peut retrouver la queue de distribution initiale des tailles des flots à partir de celle obtenue par échantillonnage en multipliant le paramètre d'échelle par le taux d'échantillonnage. Ces résultats peuvent servir aussi à caractériser les paramètres des distributions initiales.

Chapter 1

Topology discovery

A model for Internet topology discovery is proposed in this chapter. The efficiency of traceroute procedures to determine the complete set of routers of a collect network is investigated. Under some stochastic assumptions, explicit analytical expressions are obtained for the mean number of routers discovered when a subset of the stations is used in the traceroute procedure. Several tree architectures are then discussed when the total number of routers gets large, asymptotic expansions are derived. The results are compared with real data obtained from measurements.

- **Keywords: Internet, topology, traceroute, tree architectures**

1.1 Introduction

Over the past few years, a huge research effort has been devoted to the study of Internet topology. The absence of hierarchical structure, the lack of agreement between wide area network operators, local operators (metropolitan area networks), and service providers has led to a highly disordered expansion of the Internet. When considering the global topology of the Internet, different lines of investigations can be followed in order to describe the underlying structure of the network. One popular approach consists of using the theory of complex networks and random graph models (e.g. Erdös and Rényi, small worlds, etc.) Albert and Barabási [1] and Newman and Watts [35]. This line of investigation has all the more been supported by the fact that it has empirically been observed that the degree of routers obeys a power law distribution as reported in the celebrated paper by Faloutsos *et al* [16]. This approach consists to representing the Internet by means of a random graph satisfying some local properties observed by measurements. It is nevertheless worth noting that a model based on incomplete data may lead to erroneous interpretation of the reality, as argued in Radoslavov *et al* [38, 11].

Many projects have been initiated in order to discover the global topology of the Internet (CAIDA, AT&T, etc.). Most of them rely on the use of the traceroute capability offered by routers. By using traceroute, one can discover the routers along the path between a source and a destination. Even though traceroute may yield unreliable data, since paths may change within the network

and all routers do not respond to a traceroute request, it gives an indication on the global structure of the network (see for instance [22]). In particular, it allows the embedding of the Internet graph into a non Euclidean multi-dimensional space in order to evaluate the distance between two hosts Shavitt and Tankel [43] and Lakhina *et al* [29].

The inference of network topology is highly relevant for studying and possibly anticipating the propagation of attacks through the whole of the Internet (worms, DDoS, etc.). The magnitude of an attack greatly depends how the network is structured and it is of prime importance to be able to locate the weak points of the network and to implement tools to block the propagation of attacks.

In all the above cited studies, the network appears as black box whose structure is inferred by injecting probes and by reconstructing the global topology by different methods (tomography). The situation is however quite different for an network operator, who is able to know the topology of his own AS. By listening the routing messages exchanged between the different routers, it is possible to reconstruct the physical as well as the routing graph of the network (IGP information). Listening BGP routing messages can be used to know, to some extent, what happens outside a network operator's AS. IGP routing information also gives the state of the different links within the network. The instability of links is an important factor to account for when inferring the topology of an AS since the up or down state of interfaces or links impacts the routing graph and then the information obtained by a traceroute procedure.

In addition, for inferring the topology of the Internet, it is important to take into account the different components of the network. Indeed, the global Internet is composed of customer premises networks (LANs, home networks), access networks (GigaBit Ethernet or ATM networks possibly enhanced with level 3 functions), collect networks (including PoPs) and transit networks. The last ones, owned by tiers one operators, are composed of very high speed links (OC 12 links) connecting different collect networks, which are most of the time concentration networks in charge of collecting and distributing data among different users. Furthermore, at a macroscopic level, users connected to a collect network are geographically close one to each other (for instance in a same country). The topology of a collect network is, in first approximation, a tree, with possibly cross links at intermediate levels. Note that the existence of collect network naturally provides the global Internet with a small world structure. Collect networks are dense networks connected one to each other via high speed links between high capacity routers.

Scope of this chapter

In this chapter, we focus on the topology discovery of collect networks. While transit networks are composed of a relatively small number of routers, the structure of collect networks is much more intricate and involves a large number of routers in a rather small geographical area. The topology of such a network is not exactly a tree but this model can be used as a first approximation. Indeed, when inferring the topology of a network, we have to take into account different

factors, in particular the instability of links, the possibility that some routers do not respond to traceroute requests, etc. Thus, we are led to make some worst case assumptions on the behavior of the network. Assuming a tree structure amounts to assuming that the cross links at intermediate levels cannot be seen by a traceroute exploration and that any traceroute message has to go up to the root in order to discover the links along the path of a source and a destination not located in the area deserved by two different sons of the root node.

We consider a non-homogeneous tree network in which the degree of a node may depend upon the depth and place different traceroute capable hosts randomly on the leaves of the tree. We suppose that these hosts exchange traceroute messages and we compute the number of links discovered by the traceroute procedure. We specifically show that the number of links discovered rapidly increases for moderate values of the number of hosts but slowly increase when the number of hosts becomes large. This indicates that the discovery of the complete topology of a network requires a massive deployment of hosts exchanging traceroute messages.

The organization of this chapter is as follows: In Section 8.3.2, the main variables and the topologies of trees are introduced. Section 7.3 gives an explicit expression for the mean number of nodes discovered by a traceroute procedure used by a subset of the stations of the network. A connection is established with classical coupon collector problems. In order to have some insight on the impact of the topology of networks, Section 7.4 deals with asymptotic results, when the size of the network tends to infinity, of the formulas obtained in Section 7.3. In particular the rate of growth of the number of discovered routers is analyzed in great detail. In Section 7.5, we investigate the degree of a node, which is discovered by the traceroute procedure. The results obtained in the previous sections are compared in Section 1.6 against some experiments on real data from the Scan+Lucent map his map obtained from traceroutes collected by the Internet Mapping project at Lucent Bell Laboratories. We also obtain closed formulas for the variance of discovered nodes in Section 7.6 and derive expansions for it for large tree size in Section 7.7. Concluding remarks are presented in Section 8.5

1.2 Notation and definitions

1.2.1 Assumptions and notation

Throughout this chapter, we assume that the graph of the network is a partially homogeneous tree with height $n \geq 2$. For $1 \leq j < n$, a node of the jth level has d_j sons. The total number of nodes at level j is denoted by l_j and N is the total number of routers in the network. We clearly have

$$l_j = \prod_{k=1}^{j-1} d_k \quad \text{and} \quad N = \sum_{j=1}^{n} l_j,$$

with the convention $l_1 = 1$.

For $p \geq 1$, the set S of traceroute capable hosts is composed of p elements, taken at random on the last (nth) level of the tree; the nodes of this last level are referred to as the leaves of the tree network. Several traceroute capable hosts may be attached to a single leaf. In practice, when considering an ADSL collect network, the leaves of the tree network are the access routers to the IP backbone network. We assume that there is no restriction on number of hosts, which can be attached to an access router.

The traceroute capable hosts are terminals of customers, who performed traceroute procedures. In the following, that each of these users knows the IP addresses of the others in order to perform a traceroute. This implicitly assumes that there exists a server which knows the IP address of each user willing to perform a traceroute; this server is required since IP addresses for ADSL customers are dynamic.

Note it is also possible that any hosts may send traceroute messages to hosts it does not know explicitly but by guessing the IP address. This may however generate a large number of probes and very unreliable information. This is why we assume the existence of a server in charge of collecting the IP addresses of hosts performing traceroute.

With the above assumptions, any pair of hosts in S exchanges packets to determine the nodes between them. The routers discovered can embedded into a minimal subtree of the total graph, referred to as the spanning tree of the discovered routers. Its height is denoted by $H(N,p)$. Clearly, for $j \leq n$, one has $H(N,p) < n - j$ when none of the routers at level j has been discovered.

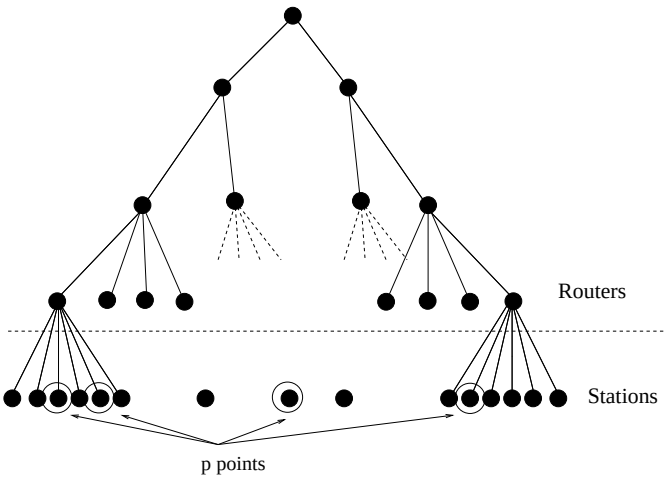


Figure 1.1: Partially Homogeneous Tree with d_1=d_2=2, d_3=4, d_4=6

One denotes by $D(N,p)$ the total number of nodes between level 1 and level n discovered by the traceroute procedure.

1.2.2 A dynamical picture for the placement of hosts

The random placement of the traceroute capable hosts at the leaves of the tree network can be seen as a dynamical stochastic process as follows: initially the p points are at the root of the tree, then they are thrown randomly on the d_1 sons; at the next step the subset of points at level 2 are thrown randomly among the d_2 sons, and so on ... until they reach the leaves of the tree.

With this description, when the points are at level j, let $(X_{1j}, \ldots, X_{l_j j})$ denotes the vector describing the number of points at each of the l_j nodes at level j. This vector has a multinomial distribution with parameter p and $(1/l_j, \ldots, 1/l_j)$. This means that for $(n_1, \ldots, n_{l_j}) \in \mathbb{N}^{l_j}$ such that $n_1 + \cdots + n_{l_j} = p$,

$$\mathbb{P}\left((X_{1j}, \ldots, X_{l_j j}) = (n_1, \ldots, n_{l_j})\right) = \frac{p!}{n_1! \ldots n_{l_j}!} \frac{1}{l_j^p}.$$

In the following sections, we study the properties of the random variable $D(N, p)$ describing the number of nodes discovered by the traceroute procedure, in particular its mean value as well as its asymptotic behavior when the height n of the tree tends to infinity while the ratio of the number of stations p to the total number of routers N in the network is of the order of $\lambda > 0$.

1.2.3 The shape of the network

The above tree model offers some flexibility with regard to the structure of the network, in particular via the choice of the number of sons of a node at level j, denoted by d_j. Ideally, this number should be a random variable and the tree would describe a the dynamics of a Galton-Watson branching process. In this chapter, we restrict the analysis to the case when d_j is deterministic.

In spite of the above restriction, one may consider different tree structures for the graph of the network. We shall specifically analyze the different cases defined as follows.

Definition 1. *For a tree with height n,*

1. *A regular tree (REG_d tree) with degree d is such that $d_j = d \geq 2$, $\forall j \in \{1, \ldots, n\}$.*

2. *Power law tree with index $\alpha > 0$, $1 \leq j \leq n$,*

   — *with increasing growth rate (PLI_α trees) if $d_j = \lfloor j^\alpha \rfloor$,*

   — *with decreasing growth rate (PLD_α trees) if $d_j = \lfloor (n - j)^\alpha \rfloor$,*

with $\lfloor y \rfloor$, the integer part of $y \in \mathbb{R}$.

As it will be seen in the following, even by restricting to the case of deterministic trees, the impact of the topology is crucial in the speed at which nodes of the network are discovered.

The tree structure with nodes having a degree growing as a power law is motivated by measurements of the Internet. Various studies have already shown that the underlying graphs of the Internet and of the web exhibit the power law property for the degree of the nodes. See Faloutsos *et al* [16].

1.3 Average number of discovered nodes

1.3.1 Distribution of the number of nodes discovered by two stations

In a first step, we examine the case of two stations placed at random at the leaves of the tree network and performing a traceroute procedure. We specifically compute the distribution of the number of nodes which are discovered by the two stations. The stations are randomly attached to the leaves of the tree, which are numbered from 1 to l_n. Station 1 sends traceroute packets to the other station. The number of routers discovered with this procedure is denoted by D. From the root of the tree, one denotes by H the maximal level for which the two stations under consideration are contained in the same subtree (see Figure 1.2).

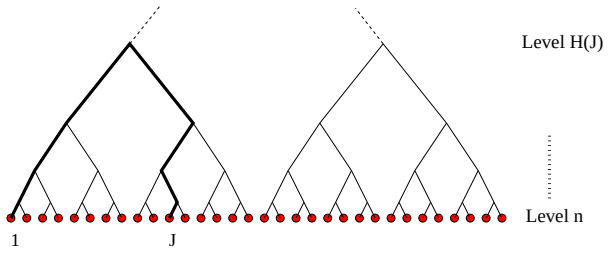


Figure 1.2: Routers discovered by two stations.

When the two stations are contained in a subtree, the root of this subtree is located at level H. It is obvious that $\mathbb{P}(H \geq 1) = 1$. We have $H \geq 2$ if and only if the two stations are contained in the same subtree issued from one of the d_1 sons of the root. The probability that the two stations are in such a subtree is equal to $1/d_1^2$. Since there are d_1 possibilities, we have $\mathbb{P}(H \geq 2) = 1/d_1$. More generally, we have $H \geq k$ if the two stations are contained in a subtree issued from one of the node at level k. Simple arguments show that we then have

$$\mathbb{P}(H \geq k) = \frac{1}{l_k} = \frac{1}{d_1 ... d_{k-1}}.$$

Since we deal with a tree, the number of routers discovered by the two stations can take the values $2k+1$ for $k = 0, ..., n-1$ and we clearly have by definition $D = 2(n - H) + 1$. By combining the different above arguments, we easily come up with the following result.

Proposition 1. *The number of routers discovered by two stations placed at random at the leaves at the tree network has the probability distribution*

$$k = 0, ..., n-1, \quad \mathbb{P}(D \leq 2k+1) = \frac{1}{d_1 \dots d_{n-1-k}}.$$

From the above result, we can easily deduce the following formula for the mean value of D.

Corollary 1. *The mean value of the random variable D is given by*

$$\mathbb{E}(D) = 2n + 1 - 2\sum_{k=1}^{n-1} \frac{1}{d_1 \dots d_k} \tag{1.1}$$

For a regular tree, we have $d_j = d > 1$ for all j and simple computations show that the mean number of routers discovered by two stations exchanging traceroute messages is given by

$$\mathbb{E}(D) = 2n + \frac{d+1}{d-1} - \frac{2}{(d-1)d^{n-1}}.$$

When n is large, we have $\mathbb{E}(D) \sim 2n$. In addition, under the same condition, the variance of the random variable D satisfies

$$\mathrm{Var}(D) \sim 4n\frac{d+1}{d-1}.$$

The above simple computations show that when n is large, the mean length of the path (in terms of discovered routers) is closed to the longest path through the network, equal to $2n-1$. Hence, even if the ratio of the number of discovered routers to the total number of routers in the network is small:

$$\frac{\mathbb{E}(D)}{N} \sim \frac{2n(d-1)}{d^n},$$

two stations placed at random can derive a fair estimate of the diameter of the network via a traceroute procedure.

The same property is satisfied for power law trees since

$$\sum_{k=1}^{\infty} \frac{1}{d_1 \dots d_k} < \infty.$$

1.3.2 Mean number of discovered nodes with p hosts

While in the previous section, we have investigated what we can learn from a traceroute procedure in a tree network by using only two hosts, we consider the case when there are p traceroute capable hosts placed at random at the leaves of the tree network. We assume that p stations are exchanging traceroute messages and we are interested in the total number of routers discovered by means of traceroute. We specifically have the following result.

Proposition 2. *The average number $D(N,p)$ of routers discovered with p stations is given by*

$$\mathbb{E}(D(N,p)) = \sum_{j=1}^{n} l_j \left(1 - \left(1 - \frac{1}{l_j}\right)^p\right) - \sum_{j=2}^{n} \frac{1}{l_j^{p-1}}. \tag{1.2}$$

Proof. For $1 \leq j \leq n$, a node A at the jth level has not been discovered if either

— with the dynamic picture, when the points are at level j, none of them is at node A. The probability of this event is given by

$$\left(1 - \frac{1}{l_j}\right)^p$$

— at level $j < n$, all the p points are at node A and all of them chose the same subtree below A. See Figure 7.1.

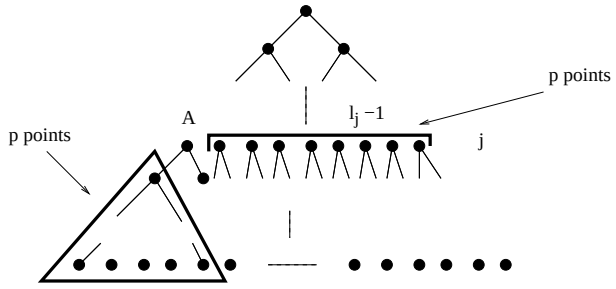


Figure 1.3: The Two Situations where Node A is not Discovered

Since there are d_j such trees, the probability of this event is given by

$$d_j \frac{1}{l_j^p} \frac{1}{d_j^p} = d_j \frac{1}{l_{j+1}^p}$$

The average number of nodes discovered is thus given by

$$\sum_{j=1}^{n} l_j \left[1 - \left(1 - \frac{1}{l_j}\right)^p - d_j \frac{1}{l_{j+1}^p} \mathbb{1}_{\{j<n\}}\right] = \sum_{j=1}^{n} l_j \left[1 - \left(1 - \frac{1}{l_j}\right)^p\right] - \sum_{j=2}^{n} \frac{1}{l_j^{p-1}},$$

and Equation (7.1) follows. □

When p is large, it is not difficult to see that the second term on the right hand side of Equation (7.1) is negligible (since $l_j \geq 2^{j-1}$). It is moreover convenient to write the first term as

$$\sum_{j=1}^{n} l_j \mathbb{P}\left(m_p \leq \frac{1}{l_j}\right) \tag{1.3}$$

where $m_p = \inf(U_i, 1 \leq i \leq p)$ and (U_i) are independent random variables uniformly distributed on $[0,1]$. This expression is in fact an equivalent of $\mathbb{E}(D(N,p))$ when p is large. Under this condition, the variable pm_p converges in distribution to the random variable X, which is an exponentially distributed random variable with parameter 1:

$$\mathbb{P}\left(pm_p \geq x\right) = \left(1 - \frac{x}{p}\right)^p \sim e^{-x}.$$

when p tends to infinity. Concerning the spanning tree of discovered routers, we have the following result.

Corollary 2. *For $0 \leq j < n$ and $p > 1$, the distribution of the height of the spanning tree of discovered routers is given by*

$$\mathbb{P}(H(N,p) \leq n - j) = \frac{1}{l_{j+1}^{p-1}},$$

and the proportion of nodes of level j discovered is given by

$$1 - \left(1 - \frac{1}{l_j}\right)^p - d_j \frac{1}{l_{j+1}^p} \mathbb{1}_{\{j<n\}}.$$

Proof. The second identity is clear from the above proof. To prove the first one, it is sufficient to remark that, in order to have no node of level j discovered, then all the points must be in some subtree whose root is at the $(j+1)$th level. □

1.3.3 A Coupon Collector Analogy

The case of symmetric networks

Let $D_j(N,p)$ denote the number of routers at level j which have been discovered when there are p traceroute capable hosts. The random variable $D_j(N,p)$ is equal to the number of different nodes seen when p nodes are independently drawn among l_j nodes. This random variable can be written as

$$D_j(N,p) \overset{\text{dist.}}{=} \sum_{i=1}^{l_j} \mathbb{1}_{\{i \in \{A_1, A_2, \ldots, A_p\}\}}, \tag{1.4}$$

where (A_i) are i.i.d. uniformly distributed random variables on $\{1, \ldots, l_{n-1}\}$. This is precisely the classical coupon collector variable. (See Comtet [10] for a general presentation of the coupon collector problem.)

Indeed, assume that we have l_j different types of coupons, which are drawn independently and uniformly. The random variable $D_j(N,p)$ represents the total number of different types of coupons after p trials. From the coupon collector's problem, it is known that

$$\mathbb{P}(D_j(N,p) = l_j) = \frac{l_j! \left\{ \begin{array}{c} p-1 \\ l_j - 1 \end{array} \right\}}{l_j^p} \tag{1.5}$$

where

$$\left\{ \begin{array}{c} n \\ k \end{array} \right\} = \sum_{\ell=1}^{k} \frac{(-1)^{k-\ell} \ell^{n-1}}{(k-\ell)!(\ell-1)!} \tag{1.6}$$

is a Stirling number of the second kind. These numbers satisfy the recursion

$$\left\{ \begin{array}{c} n \\ k \end{array} \right\} = \left\{ \begin{array}{c} n-1 \\ k-1 \end{array} \right\} + k \left\{ \begin{array}{c} n-1 \\ k \end{array} \right\}$$

and their generating function, for fixed k, is given by

$$\sum_{n=0}^{\infty} \left\{ \begin{array}{c} n \\ k \end{array} \right\} x^n = \frac{x^k}{(1-x)(1-2x)\dots(1-kx)}.$$

(See Wilf [48] for details).

The quantity $\mathbb{P}(D_j(N,p) = l_j)$ is the probability that with p traceroute capable hosts, all the nodes at level j are discovered. Conversely, one may consider the problem of determining the number of traceroute hosts one may need to discover all the nodes at level j. Let ν_j denote this random variable.

From the coupon collector problem, the probability $\mathbb{P}(\nu_j = p)$ is given by Formula (7.2). The mean value of the random variable ν_j is given by

$$\mathbb{E}(\nu_j) = 1 + \frac{l_j}{l_j - 1} + \frac{l_j}{l_j - 2} + \cdots + \frac{l_j}{2} + l_j$$

and its standard deviation by

$$\sigma_j^2 = l_j^2 \sum_{i=1}^{l_j} \frac{1}{i^2} - \left(1 + \frac{l_j}{l_j - 1} + \frac{l_j}{l_j - 2} + \cdots + \frac{l_j}{2} + l_j \right). \tag{1.7}$$

It is worth noting that, as $l_j$becomes large,

$$\mathbb{E}(\nu_j) = l_j \left(\log l_j + \gamma + O\left(\frac{1}{l_j} \right) \right), \tag{1.8}$$

where γ is the Euler constant. In addition, we have the Chernoff bound like formula

$$\lim_{l_j \to \infty} \mathbb{P}(\nu_j > l_j \log(l_j) + x l_j) = 1 - e^{-e^{-x}}.$$

The above limit gives the deviation of the random variable ν_j with respect to its mean value when $l_j \to \infty$.

From the above results on the coupon collector problem, we thus observe that for discovering the l_j routers at level j, we approximately need $l_j \log l_j$ traceroute capable hosts. In addition, for discovering the half of the nodes, we have

$$\mathbb{E}(\#\text{hosts needed for discovering } l_j/2 \text{ nodes}) = l_j \log 2 + O(1).$$

This indicates that in order to discover all the nodes at level j, we need a number of traceroute capable hosts, which is much larger than the number of nodes.

Take for instance a constant degree d, then the number of nodes in the network is $N = (d^n - 1)/(d - 1)$ and $l_n = d^{n-1}$. We then have $l_n = ((d-1)N + 1)/d$ and all the nodes in the network are discovered if all the nodes at level n are discovered. This requires a number ν_n of hosts such that

$$\mathbb{E}(\nu_n) \sim \frac{(d-1)N}{d} \log N$$

when N is large. The number of hosts required to discover all the nodes in the network is much greater than the number of nodes.

Hence, a complete topology discovery requires a massive deployment of hosts exchanging traceroute messages. This will be further illustrated in the next section when considering all the nodes in the network.

The case of asymmetric networks

So far, we have assumed that the network is completely symmetric. In reality, however, this is rarely the case. In fact, collect networks are highly asymmetrical. For instance, the number of users in a city (dense area) is much larger than the number of users on the country side (sparse area). Hence, when considering the discovery of concentration routers, located at intermediate levels, say, level $j < n$, the probability that a user belongs to a dense area is larger than the probability the user is in a sparse area.

Keeping in mind the analogy of the coupon collector's problem, we now assume that the probability of drawing a coupon of type r is denoted by p_r. Without loss of generality, we assume that $p_1 \leq p_2 \leq \ldots \leq p_{l_j}$.

If traceroute capable hosts are placed at random at the leaves of the network, the number of hosts needed to discover all the routers at level j is such that as $l_j \to \infty$,

$$\log \mathbb{P}(p_1 \nu_j - b_j \leq z) \sim -\sum_{r}^{l_j} \exp\left(-\frac{p_r}{p_1}(z + b_j)\right),$$

where b_j is chosen so that

$$\sum_{r=1}^{l_j} \exp\left(-p_r \frac{b_j}{p_1}\right) < \infty$$

as $l_j \to \infty$. (See Klaassen [26] for details.)

The above result shows that the growth rate of ν_j is determined by the less dense area since $\nu_j \sim b_j/p_1$ when $l_j \to \infty$. This indicates that the discovery of routers deserving sparse area is quite expensive if hosts are placed at random. It follows that the discovery of an asymmetrical network with sparse areas requires a coordination between traceroute capable hosts. Hosts cannot be drawn at random but the placement of hosts should take into account the density of the areas deserved by collect routers.

To quantitatively illustrate the above phenomenon, assume that the population deserved by routers is proportional to α^j for some $\alpha > 1$. In this case, we have $p_r = p_1 \alpha^{r-1}$ and $1/p_1 = (\alpha^{l_j} - 1)/(\alpha - 1)$. We can take $b_j = 1$ so that

$\nu_j \sim \alpha^{l_j}/(\alpha-1)$ when $l_j \to \infty$. We hence see that the number of hosts needed to discover routers deserving sparse areas is explosive.

The versatility of the coupon collector analogy

So far, we have considered tree networks. However, the coupon collector analogy still pertains for more complex topologies as long as the network presents a hierarchical structure. Indeed, if the network is organized in layers (or levels) and if a node at a given level deserves a certain population of users, we easily see that we can still use the coupon collector analogy for computing the number of hosts needed to discover the nodes at a given level. Moreover, in view of the preceding section, we see that the populations deserved by the different routers can be asymmetric. The application of this observation to more complex network topologies will be addressed in further studies.

1.4 Asymptotic results

To understand more closely the basic properties of topology discovery by means of traceroute, it is assumed in this section that the ratio of the number p of traceroute capable hosts to the total number N of routers in the network is fixed and equal to a constant λ and we suppose that N goes to infinity. We are specifically interested in the behavior of $\mathbb{E}(D(N,p))/N$ when N tends to infinity. This quantity gives information about the ratio of the number of routers discovered to the total number of nodes in the network.

1.4.1 Regular trees

In a first step, we investigate the case when the degree of nodes is constant, equal to $d \geq 2$.

Proposition 3. *For a regular tree with degree $d \geq 2$, when $N \to +\infty$ and $p/N \to \lambda$,*

$$\frac{\mathbb{E}(D(N,p))}{N} \sim T_{REG_d}(\lambda) \overset{def.}{=} \sum_{j=1}^{+\infty} \frac{d-1}{d^j}\left(1-\exp\left(-\frac{\lambda d^j}{d-1}\right)\right). \tag{1.9}$$

Proof. Using that $N \sim d^n/(d-1)$ and the equivalent (1.3) as N goes to infinity, one gets that

$$\frac{1}{N}\mathbb{E}(D(N,p)) \sim (d-1)\sum_{j=1}^{n} \frac{d^{j-1}}{d^n}\mathbb{P}\left(pm_p \leq \frac{\lfloor \lambda \frac{d^n}{d-1} \rfloor}{d^{j-1}}\right)$$

and then,

$$\frac{1}{N}\mathbb{E}(D(N,p)) \sim (d-1)\sum_{j=1}^{n} \frac{1}{d^j}\mathbb{P}\left(X \leq \frac{\lambda}{(d-1)}d^j\right)$$

where X is an exponentially distributed random variable with unit mean, since as mentioned in Section 7.3, the random variable pm_p converges in distribution to X. □

We now take benefit the closed form of the ratio of discovered nodes given by equation (7.3) in order to illustrate the speed of the exploration process. For this purpose, we study the dependence of the number of discovered nodes on the ratio $\lambda \sim p/N$ when N is large while λ is small. It describes in some sense the initial speed of the exploration process through the network. Indeed, the analysis gives the ratio of discovered nodes with a small number of traceroute capable hosts.

Proposition 4. *For a regular tree with degree $d \geq 2$, when λ tends to 0 the asymptotic proportion of discovered nodes satisfies the following equivalence*

$$T_{REG_d}(\lambda) \sim -\lambda \log_d(\lambda). \tag{1.10}$$

Proof. As it can easily be seen, the asymptotic behavior of Equation (7.5) when λ tends to 0 is quite delicate since the series is divergent. The technique used to get an expansion relies on a convenient rewriting of the equation together with the use of Fubini's Theorem for non-negative functions. (See Robert [42] for a general description of the method.)

We have

$$\begin{aligned}\sum_{j=1}^{+\infty} \frac{1}{d^j}\left(1-\exp\left(-\frac{\lambda d^j}{d-1}\right)\right) &= \sum_{j=1}^{+\infty} \frac{1}{d^j}\int_0^{+\infty} \mathbb{1}_{\left\{u\leq \frac{\lambda d^j}{d-1}\right\}} e^{-u}\,du \\ &= \int_0^{+\infty} \sum_{j=1}^{+\infty} \frac{1}{d^j}\mathbb{1}_{\left\{u\leq \frac{\lambda d^j}{d-1}\right\}} e^{-u}\,du,\end{aligned}$$

where we have used Fubini's theorem to exchange the sum and the integral signs. This entails that

$$\sum_{j=1}^{+\infty} \frac{1}{d^j}\left(1-\exp\left(-\frac{\lambda d^j}{d-1}\right)\right) = \frac{d}{d-1}\int_0^{\lambda a} e^{-u}du + \frac{1}{d-1}\int_{\lambda a}^{+\infty} \frac{1}{d^{\lfloor \log_d(u(d-1)/\lambda)\rfloor}} e^{-u}\,du, \tag{1.11}$$

with $a = d/(d-1)$.

The first term on the right hand side of Equation (1.11) vanishes when λ tends to 0. For the second term, we note that with regard to the asymptotic behavior for λ close to 0, the integration interval $[\lambda a, \infty]$ can be replaced by any interval $[\lambda a, b]$ with $b > \lambda a$. To simplify, we take $b = a$ so that

$$\int_a^{+\infty} \frac{1}{d^{\lfloor \log_d(u(d-1)/\lambda)\rfloor}} e^{-u}\,du \leq \lambda(1-e^{-a}),$$

since for $u \geq a$,

$$\lfloor \log_d(u(d-1)/\lambda)\rfloor \geq -\log_d \lambda.$$

It follows that the function $T_{\mathrm{REG}_d}(\lambda)$ defined by Equation (7.3) has the following equivalent in the neighborhood of 0

$$T_{\mathrm{REG}_d}(\lambda) \sim \int_{\lambda a}^{a} d^{\{\log_d(u(d-1)/\lambda)\}} \frac{\lambda e^{-u}}{u(d-1)}\, du = \frac{\lambda}{d-1}\left(\int_{\lambda}^{1} d^{\{\log_d u/\lambda\}} \frac{e^{-au}-1}{u}\, du + \int_{1}^{1/\lambda} \frac{d^{\{\log_d u\}}}{u}\, du\right) \tag{1.12}$$

with $\lfloor y \rfloor$ is the integer part of $y \in \mathbb{R}$ and $\{y\} = y - \lfloor y \rfloor$ its fractional part.

The asymptotic behavior of the function $T_{\mathrm{REG}_d}(\lambda)$ when λ tends to 0 is determined by the second term on the right hand side of Equation (1.12). Hence, as $\lambda \sim 0$, we have the equivalence

$$T_{\mathrm{REG}_d}(\lambda) \sim \frac{\lambda}{d-1} \int_{1}^{1/\lambda} \frac{d^{\{\log_d u\}}}{u}\, du.$$

If $\lambda = 1/(xd^m)$ with fixed $1 < x \leq d$ and $m \in \mathbb{N}$, then as $\lambda \to 0$

$$\begin{aligned}
\int_{1}^{1/\lambda} \frac{d^{\{\log_d u\}}}{u}\, du &\sim \int_{x}^{1/\lambda} \frac{d^{\{\log_d u\}}}{u}\, du \\
&= \sum_{k=0}^{m-1} \int_{xd^k}^{xd^{k+1}} \frac{d^{\{\log_d u\}}}{u}\, du \\
&= \sum_{k=0}^{m-1} \int_{1}^{d} \frac{d^{\{\log_d(xu)\}}}{u}\, du \\
&= -\log_d(\lambda x) \int_{1}^{d} \frac{d^{\{\log_d(xu)\}}}{u}\, du.
\end{aligned}$$

The last integral can be expressed as follows, recall that $\lambda = 1/(xd^m)$ with $x \in [1, d]$,

$$\begin{aligned}
&\int_{1}^{d} \frac{d^{\{\log_d(xu)\}}}{u}\, du \\
&\qquad = \int_{1}^{d/x} \frac{d^{\{\log_d(xu)\}}}{u}\, du + \int_{d/x}^{d} \frac{d^{\{\log_d(xu)\}}}{u}\, du \\
&\qquad = \int_{1}^{d/x} \frac{d^{\log_d(xu)}}{u}\, du + \int_{d/x}^{d} \frac{d^{\log_d(xu)-1}}{u}\, du \\
&\qquad = x\left(\frac{d}{x} - 1\right) + \frac{x}{d}\left(d - \frac{d}{x}\right) = d - 1.
\end{aligned}$$

Equivalence (7.5) is therefore established. □

From equivalence (7.5), it is worth noting that the ratio $T_{\mathrm{REG}_d}(\lambda)/\lambda$ is not constant but is equal to $-\log_d(\lambda)$. Thus, when the number p of traceroute capable hosts is small when compared to the number N of nodes in a tree network, the number of nodes discovered is more than linear in the ratio $\lambda = p/N$. This indicates that the speed of the exploration process is quite fast when the number of hosts is small. This encouraging observation is however

to be counterbalanced by the fact that, by Equation (7.3), the speed of the exploration process is nevertheless decreasing exponentially fast with respect to λ. In addition, the speed depends on the degree of nodes via $\log_d \lambda$. This indicates that the greater the degree of nodes, the smaller is the speed of the learning process.

The above remarks show that we rapidly learn about the topology of a tree network with a small initial number of hosts but the speed of learning decreases as the number of hosts increases.

1.4.2 Power law trees

Power Law trees with increasing degree

For power law trees with increasing degree, we have the following result.

Proposition 5. *For a power law tree with increasing degree, the proportion of discovered nodes is such that*

$$\lim_{\substack{N\to+\infty,\\ p/N\to\lambda}} \frac{D(N,p)}{N} = T_{PLI_\alpha}(\lambda) \stackrel{def.}{=} 1 - e^{-\lambda}. \tag{1.13}$$

Proof. For power trees with increasing degree, we have $l_{n+1} \sim (n!)^\alpha$. It is then quite easy to see that $N \sim l_n$ and if $p \sim \lfloor \lambda N \rfloor$, then

$$\frac{D(N,p)}{N} \sim \sum_{j=1}^{n} \frac{l_j}{l_n} \mathbb{P}\left(pm_p \leq \lambda \frac{l_n}{l_j} \right).$$

When $n \to +\infty$, all the terms of the series vanish except the last one with index n. One consequently obtains, by recalling that pm_p converges in distribution to an exponentially distributed random variable with parameter 1 when $p \to +\infty$,

$$\lim_{\substack{N\to+\infty,\\ p/N\to\lambda}} \frac{D(N,p)}{N} = 1 - e^{-\lambda},$$

and the result follows. □

Equation (1.13) is not really informative since it does not depend on the growth parameter α. This is not the case for regular trees, see Equation (7.3) which gives the growth rate of the number of discovered routers with respect to d (via the $\log_d$ term). An asymptotic analysis of this equation will give further insight in this case.

Power Law trees with decreasing degree

We can state the analogue of Proposition 5 for power law trees with decreasing degree. We specifically have:

Proposition 6. *For a PLD$_\alpha$ tree, when $N \to +\infty$ and $p/N \to \lambda$, then*

$$\frac{D(N,p)}{N} \to T_{PLD_\alpha}(\lambda)$$

with

$$T_{PLD_\alpha}(\lambda) = \sum_{j=1}^{+\infty} \frac{1}{H(\alpha)(j!)^\alpha} \left(1 - e^{-\lambda H(\alpha)(j!)^\alpha}\right) \tag{1.14}$$

and

$$H(\alpha) = \sum_{j=1}^{+\infty} \frac{1}{(j!)^\alpha}.$$

Proof. For a PLD$_\alpha$ tree, we have for large n, $l_j \sim ((n-1)!/(n-j)!)^\alpha$

$$\frac{N}{((n-1)!)^\alpha} \sim \sum_{j=1}^{+\infty} \frac{1}{(j!)^\alpha} = H(\alpha).$$

If Equivalence (1.3) is used again, one gets that, when N goes to infinity, the ratio $D(N,p)/N$ is equivalent to

$$\sum_{j=1}^{n} \frac{1}{H(\alpha)(n-j)!} \mathbb{P}\left(pm_p \leq \lambda H(\alpha)((n-j)!)^\alpha\right).$$

and Equation (1.14) follows. □

1.4.3 The growth rate of the exploration process

As before, the behavior of $T_{\mathrm{PLI}_\alpha}(\lambda)$ and $T_{\mathrm{PLD}_\alpha}(\lambda)$ when λ is small is investigated. From Equation (1.13), one directly concludes that $T_{\mathrm{PLI}_\alpha}(\lambda) \sim \lambda$ when λ gets small.

For a PLD$_\alpha$ tree, we have the following result.

Proposition 7. *For a PLD$_\alpha$ tree, when λ tends to 0 the asymptotic proportion of discovered nodes satisfies the following equivalence*

$$T_{PLD_\alpha}(\lambda) \sim \frac{\lambda \log(1/\lambda)}{\alpha \log\log(1/\lambda)}. \tag{1.15}$$

Proof. Equation (1.14) gives

$$\frac{1}{\lambda}T_{\mathrm{PLD}_\alpha}(\lambda) = \sum_{j=1}^{+\infty} \frac{1}{(j!)^\alpha \tilde{\lambda}} \left(1 - e^{-(j!)^\alpha \tilde{\lambda}}\right),$$

where $\tilde{\lambda} = \lambda H(\alpha)$. If h is the function defined by

$$h(x) = -\frac{1 - e^{-x}}{x},$$

Equation (1.14) shows that the quantity $T_{\mathrm{PLD}_\alpha}(\lambda)/\lambda$ is given by

$$\sum_{j=1}^{+\infty} \int_0^{+\infty} \mathbb{1}_{\{\tilde{\lambda}(j!)^\alpha \leq u\}} h'(u)\, du. \tag{1.16}$$

Let us introduce the classical Euler's Gamma function Γ defined by

$$\Gamma(x) = \int_0^{+\infty} u^{x-1} e^{-u}\, du$$

on $(0, +\infty)$ (recall that $\Gamma(n+1) = n!$ when $n \in \mathbb{N}$). The function Γ is a bijection from $[2, \infty) \to [1, \infty)$; its inverse is denoted by Γ^{-1}.

By Fubini's Theorem, Equation (1.16) can be rewritten as

$$\begin{aligned}
&\int_{\tilde{\lambda}}^{+\infty} \sum_{j=1}^{+\infty} \mathbb{1}\left\{ j + 1 \leq \Gamma^{-1}\left(\left(\frac{u}{\tilde{\lambda}} \right)^{1/\alpha} \right) \right\} h'(u)\, du \\
&= \int_{\tilde{\lambda}}^{+\infty} \left\lfloor \Gamma^{-1}\left(\left(\frac{u}{\tilde{\lambda}} \right)^{1/\alpha} \right) - 1 \right\rfloor h'(u)\, du.
\end{aligned} \tag{1.17}$$

Stirling's Formula for the Gamma function gives that $\Gamma(x) \sim x^{x-1/2} e^{-x} \sqrt{2\pi x}$ for a large x. Therefore, if $y = \Gamma(x)$, one gets the relation

$$\log y \sim x \log x,$$

if we write $x = \phi(y) \log y$, then

$$\frac{1}{\log x} \sim \phi(y)$$

hence $\phi(y) \to 0$ as $y \to +\infty$. Therefore, one gets

$$1 \sim \phi(y) \log \phi(y) + \phi(y) \log \log y,$$

as y gets large, so $\phi(y) \log \log y \sim 1$. Thus, the inverse function satisfies the equivalence

$$\Gamma^{-1}(y) \sim \frac{\log(y)}{\log(\log(y))}$$

when y tends to infinity.

Since

$$h'(x) = \frac{1-(1+x)e^{-x}}{x^2},$$

it is easy to see that the integral (1.17) is diverging when λ tends to 0. Moreover, it is equivalent to

$$\int_{\tilde{\lambda}a}^{+\infty} \left\lfloor \Gamma^{-1}\left(\left(\frac{u}{\tilde{\lambda}}\right)^{1/\alpha}\right) - 1 \right\rfloor h'(u)\,du,$$

for an arbitrary $a > 1$, since the remaining part of the integral converges as $\lambda \to 0$. By choosing a sufficiently large so that, for $\varepsilon > 0$,

$$1-\varepsilon \leq \frac{\Gamma^{-1}(x)}{\log(x)/\log(\log(x))} \leq 1+\varepsilon, \quad x \geq a,$$

it is sufficient to study the expansion of the quantity

$$I(\tilde{\lambda}) = \int_{\tilde{\lambda}a}^{+\infty} \frac{\log(u/\tilde{\lambda})/\alpha}{\log\left(\log(u/\tilde{\lambda})/\alpha\right)} h'(u)\,du. \tag{1.18}$$

Let us introduce

$$\phi(\tilde{\lambda}) = \log\left(\tilde{\lambda}^{-1}\right) \Big/ \log\log\left(\tilde{\lambda}^{-1}\right)$$

The integral (1.18) can be rewritten as

$$I(\tilde{\lambda}) = \frac{1}{\alpha}\int_{\tilde{\lambda}a}^{+\infty} \frac{\log(u)+\log(\tilde{\lambda}^{-1})}{D(u)} h'(u)\,du,$$

where

$$D(u) = -\log(\alpha) + \log(\log(\tilde{\lambda}^{-1})) + \log\left(1+\log(u)/\log(\tilde{\lambda}^{-1})\right).$$

Since $\log(u)h'(u)$ is integrable on $\mathbb{R}_+$, trite inequalities and Lebesgue's Theorem show that $I(\tilde{\lambda})/\phi(\tilde{\lambda})$ converges to

$$\int_0^{+\infty} h'(u)\,du = 1$$

as $\tilde{\lambda} \to 0$ and the proposition follows. □

1.4.4 Discussion: The asymptotic profile of tree structures

In order to give an intuitive explanation for the initial growth of the exploration process seen in the above section, we first introduce the concept of the profile of a tree.

Definition 2. *The asymptotic profile of a tree is given by the sequence* (p_k) *defined by*

$$p_k = \limsup_{n\to+\infty} \frac{\sum_{j=n-k}^{n-1} l_j}{N} = \limsup_{n\to+\infty} \frac{\sum_{j=n-k}^{n-1} l_j}{\sum_{j=1}^{n-1} l_j}.$$

For $k \geq 1$, the quantity p_k is the proportion of nodes of the tree in the last k levels. In the following, the notation $p_k(X)$ shall be used. where X is one of the classes of trees: REG_d, PLI_α and PLD_α. The following proposition is quite straightforward to prove.

Proposition 8. *For* $k \geq 1$,

$$\begin{aligned} p_k(REG_d) &= 1 - \frac{1}{d^k}, \\ p_k(PLD_\alpha) &= \sum_{j=1}^{k} \frac{1}{(j!)^\alpha} \Big/ \sum_{j=1}^{+\infty} \frac{1}{(j!)^\alpha}\ , \\ p_k(PLI_\alpha) &= \mathbb{1}_{\{k=1\}}, \end{aligned}$$

The initial growth rate of the exploration process of the three tree topologies proved in Section 1.4.3 is recalled, when the proportion λ of stations involved in the exploration process is small. We specifically have

$$\begin{aligned} T_{\mathrm{REG}_d}(\lambda) &\sim \lambda \log(1/\lambda), \\ T_{\mathrm{PLD}}(\lambda) &\sim \frac{\lambda \log(1/\lambda)}{\alpha \log\log(1/\lambda)}, \\ T_{\mathrm{PLI}}(\lambda) &\sim \lambda \end{aligned}$$

when λ tends to 0.

For the PLI_α tree, most of the N nodes are at the bottom of the network: $p_1 = 1$ and $p_k = 0$ if $k \neq 1$. So a traceroute procedure initiated by a host discovers essentially the router the hosts is attached to. This explains the low speed of the initial phase of the exploration. See the section on the coupon collector analogy.

On the contrary, see Proposition 8, for regular trees and PLD_α trees, a non-negligible proportion of nodes are in the upper levels of the network. Since a traceroute procedure discovers several routers in these layers, it speeds up the discovery rate of the process. Note that for the proportion of nodes above the k last levels is of the order of $1/d^k$ for REG_d trees and $1/((k+1)!)^\alpha$ for PLD_α trees, which explains that the exploration process is initially slightly faster for regular trees.

Growth rate for large λ

Formula (7.3) shows that the growth rate of the exploration process decreases exponentially fast with respect to λ for any tree architecture. Figure 1.4 displays

the proportion of nodes discovered as a function of the ratio λ. It clearly appears from this figure that the total discovery of the network requires a large number of hosts. The structure, which requires the smallest number of hosts, is the regular tree while the plower law increasing structure is the most demanding.

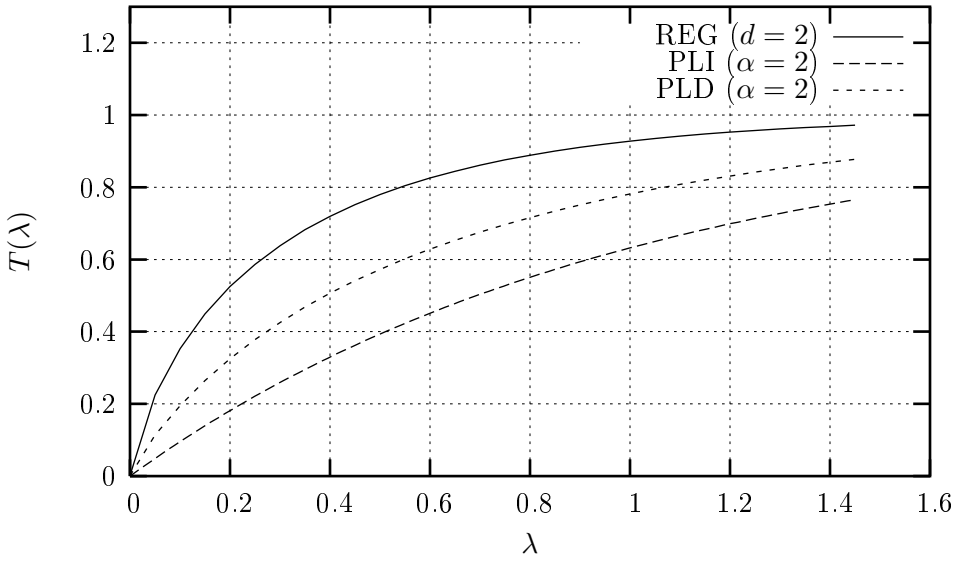


Figure 1.4: Proportion of discovered nodes by using $p = \lfloor \lambda N \rfloor$ stations for REG_d, PLD_α and PLI_α trees with $d = 2$ and $\alpha = 2$.

1.5 Degree of discovered nodes

In this section, we investigate the degree of a random node in the network seen by the exploration process with p traceroute capable hosts. That degree is denoted by the random variable L, with the convention that $L = 0$ if the node is not discovered. Before giving the distribution, let us introduce some additional notation: for $a \geq 1$ and $0 < k \leq a$, $(a)_k = a(a-1)\dots(a-k+1)$, $(a)_0 = 1$ and $(a)_k = 0$ for $k > a$. Moreover, let $b(p, n, l_j)$ be the Bernoulli probabilities defined by

$$b(p, n, l_j) = \binom{p}{n} \left(\frac{1}{l_j}\right)^n \left(1 - \frac{1}{l_j}\right)^{p-n}. \tag{1.19}$$

Proposition 9. *The degree L of a node discovered by the exploration by means of traceroute is given by: for $k \geq 1$,*

$$\mathbb{P}(L = k) = \sum_{j=1}^{n} \frac{l_j}{N} \frac{(d_j)_k}{1 - \left(1 - \frac{1}{l_j}\right)^p} \sum_{m=k}^{p} \frac{b(p, m, l_j)}{d_j^m} \left\{ \begin{array}{c} m \\ k \end{array} \right\} \tag{1.20}$$

where $\left\{ \begin{array}{c} m \\ k \end{array} \right\}$ *is the Stirling number of the second kind defined by Equation* (1.6).

Proof. Let us pick up a node at random. This node belongs to level j with probability l_j/N. For a discovered node at level j, given that it it is not 0, the number M of hosts in the subtree associated to this node has the distribution $\mathbb{P}(M = m) = b(p, m, l_j)/(1 - b(p, 0, l_j))$, where the quantities $b(p, m, l_j)$ are the Bernoulli probabilities defined by Equation (7.9).

Assume now that there m traceroute capable hosts in the subtree of a node at level j, which has degree d_j. The degree seen via traceroute is equal to the number of sons of the node considered associated to the traceroute hosts. This number L_j is exactly equal to the number of nonempty cells when m objects are distributed into d_j different cells. The distribution of this number is given by (see Riordan [40, p. 100])

$$\mathbb{P}(L_j = k) = \frac{(d_j)_k}{d_j^m} \left\{ \begin{array}{c} m \\ k \end{array} \right\}. \tag{1.21}$$

By deconditioning, Equation (7.10) follows. □

The mean value of the degree L of a discovered node is given by the following result.

Corollary 3. *The mean value of the random variable L is given by*

$$\mathbb{E}(L) = \sum_{j=1}^{n} \frac{l_j d_j}{N} \frac{1 - \left(1 - \frac{1}{d_j l_j}\right)^p}{1 - \left(1 - \frac{1}{l_j}\right)^p}. \tag{1.22}$$

Proof. The mean value of the random variable L_j with the distribution defined by Equation (1.21) is given by

$$\mathbb{E}(L_j) = d_j \left(1 - \left(1 - \frac{1}{d_j}\right)^m\right).$$

By deconditioning, Equation (7.11) follows. □

In the case of a regular graph with constant degree d, we have when $n \to \infty$ and $p = \lambda N$,

$$\mathbb{E}(L) \sim (d-1) \sum_{j=0}^{\infty} \frac{1}{d^j} \frac{1 - e^{\lambda d^j}}{1 - e^{\lambda d^{j+1}}}.$$

When λ tends to 0, we have $\mathbb{E}(L) \sim 1$. This means that at the beginning of the exploration process, the nodes seen in a regular tree network are seen only once.

1.6 Experimental results

Several experiments were conducted on a set of real graphs. We used the Scan+Lucent map which is a merge of a map from the Internet Mapping project at Lucent Bell Laboratories on November 1999 and a map obtained from the Mercator software. The Scan+Lucent map contains 284804 routers, the degree

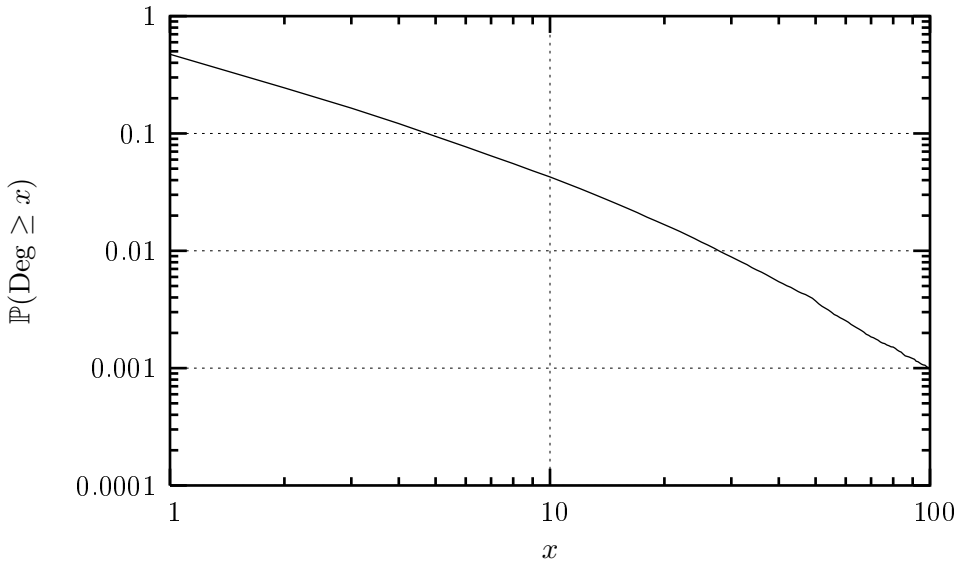


Figure 1.5: The function $x \to \mathbb{P}(\mathrm{Deg} \geq x)$ on a double logarithmic scale of Scan+Lucent graphs

of the nodes ranges from 1 (150397 stations) to 1978 (1 station) with an average of 3.02 and a variance of 8.624. Figure 1.6 gives an indication of how far the distribution of the degree is from a power law distribution.

By using the edges of this graph, several spanning trees have been constructed recursively. Root nodes are chosen either at random or are the nodes with the highest degree, the other levels are defined recursively in the following way:

- Bounded degree: if a node is in the tree, only d of its sons are chosen in the next level.

- Unbounded: all its sons are taken in the next level.

With this method, one does not, of course, recover the complete graph. But, as noted in the introduction, this gives an upper bounded on the real discovery process if these spanning trees are used instead of the "real" graph. It must be mentioned that, in our experiment, only a small fraction of edges connects nodes belonging to different subtrees of the top level. The edges which are discarded connect essentially nodes of different layers within the top subtrees.

Figure 1.6 shows, as expected, a steady growth at the origin when the degree is bounded to 2 or 4. When the degree is not bounded, the initial growth suggests a PLI tree architecture.

It must be noted that these experiments are incomplete since the size of the data sets available is not sufficiently large to investigate with a reasonable accuracy the parameters of the power law features of these architectures.

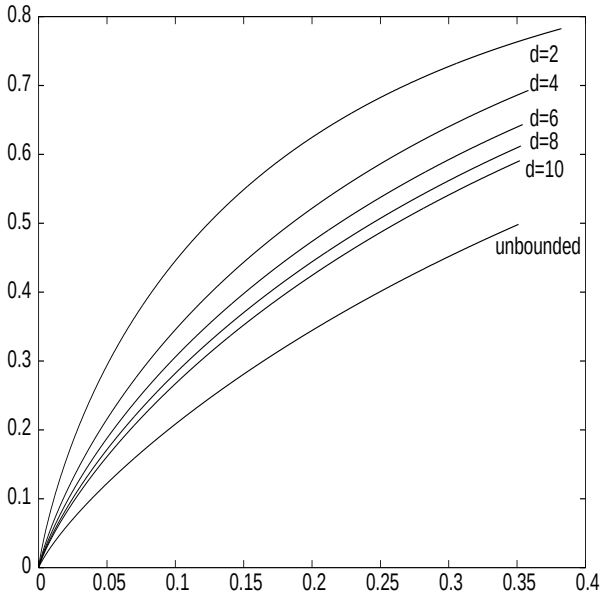


Figure 1.6: The ratio of discovered nodes for the spanning trees for Scan+Lucent graphs

1.7 Variance

1.7.1 Expression of the variance

We note (j,s) the node in level j, $0 \leq j < n$ and index s, $0 \leq s < l_j$ (in the same level, nodes are indexed from the left to the right). We note also $A_{j,s}$ the random variable which equals 1 if the node (j,s) is discovered and 0 otherwise. So we have:

$$D(N,p) = \sum_{(j,s)} A_{j,s}$$

with $0 \leq j < n$ and $0 < s < l_j$. Then,

$$\mathrm{Var}(D(N,p)) = \sum_{(j,s)} \mathrm{Var}(A_{j,s}) + \sum_{(j,s),(k,t)/(j,s)\neq(k,t)} \mathrm{Cov}(A_{j,s}, A_{k,t})$$

In order to explicit $\mathrm{Var}(D(N,p))$, we need the expressions of $\mathrm{Var}(A_{j,s})$ and $\mathrm{Cov}(A_{j,s}, A_{k,t})$. First, Let's calculate $\mathbb{E}(A_{j,s})\mathbb{E}(A_{k,t})$. we know that:

$$\mathbb{E}(A_{j,s}) = 1 - \left(1 - \frac{1}{l_j}\right)^p - \frac{1}{l_j}\left(\frac{1}{l_{j+1}}\right)^{p-1}$$

So,

$$\begin{aligned}\mathbb{E}(A_{j,s})\mathbb{E}(A_{k,t}) = 1 &- \left(1-\frac{1}{l_j}\right)^p - \left(1-\frac{1}{l_k}\right)^p + \left(1-\frac{1}{l_j}\right)^p\left(1-\frac{1}{l_k}\right)^p \\ &- \frac{1}{l_j}\left(\frac{1}{l_{j+1}}\right)^{p-1}\left(1-\left(1-\frac{1}{l_k}\right)^p\right) \\ &- \frac{1}{l_k}\left(\frac{1}{l_{k+1}}\right)^{p-1}\left(1-\left(1-\frac{1}{l_j}\right)^p\right) \\ &+ \frac{1}{l_j}\left(\frac{1}{l_{j+1}}\right)^{p-1}\frac{1}{l_k}\left(\frac{1}{l_{k+1}}\right)^{p-1}\end{aligned}$$

For $k \geq j$, each of the three last terms is inferior to $\frac{1}{l_j}\left(\frac{1}{l_{j+1}}\right)^{p-1}$. Consequently, we can write:

$$\mathbb{E}(A_{j,s})\mathbb{E}(A_{k,t}) = 1 - \left(1-\frac{1}{l_j}\right)^p - \left(1-\frac{1}{l_k}\right)^p + \left(1-\frac{1}{l_j}\right)^p\left(1-\frac{1}{l_k}\right)^p + r_{j,k} \tag{1.23}$$

Where, $|r_{j,k}| \leq \frac{3}{l_j}\left(\frac{1}{l_{j+1}}\right)^{p-1}$

To calculate $\mathbb{E}(A_{j,s}A_{k,t})$, we need to distinguish three separate cases:

- Nodes (j,s) and (k,t) are identical

In this case

$$\mathbb{E}(A_{j,s}A_{k,t}) = \mathbb{E}(A_{j,s}^2) = \mathbb{E}(A_{j,s}) = 1 - \left(1-\frac{1}{l_j}\right)^p - \frac{1}{l_j}\left(\frac{1}{l_{j+1}}\right)^{p-1}$$

Consequently, we get:

$$\mathrm{Var}(A_{j,s}) = \left(1-\frac{1}{l_j}\right)^p - \left(1-\frac{1}{l_j}\right)^{2p} + C_0(j) \tag{1.24}$$

With: $|C_0(j)| \leq \frac{4}{l_j}\left(\frac{1}{l_{j+1}}\right)^{p-1}$

- Node (j,s) is an ancestor of node (k,t): (k,t) is different from (j,s) and belongs to the subtree rooted at this node (cf figure 1.7)

In this case, the variable $A_{j,s}A_{k,t}$ equals 1 if and only if among the p leaves, at least one is in the set A and one other outside B. So,

$$\begin{aligned}\mathbb{E}(A_{j,s}A_{k,t}) &= \mathbb{P}(A \text{ and } \bar{B} \text{ not empty}) \\ &= 1 - \mathbb{P}(A \text{ empty}) - \mathbb{P}(\bar{B} \text{ empty}) + \mathbb{P}(A \text{ and } \bar{B} \text{ empty}) \\ &= 1 - \left(1-\frac{1}{l_k}\right)^p - \left(\frac{1}{l_{j+1}}\right)^p + \left(\frac{1}{l_{j+1}} - \frac{1}{l_k}\right)^p\end{aligned}$$

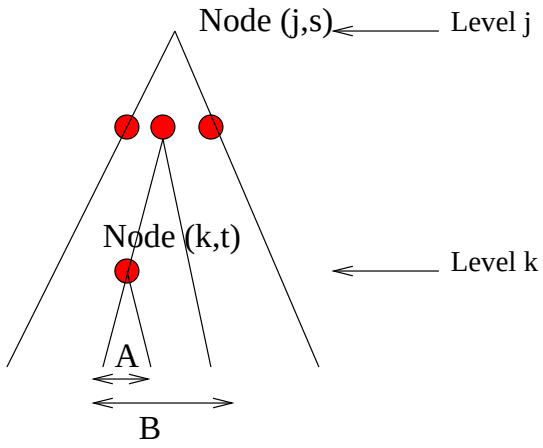


Figure 1.7: Node (j,s) is an ancestor of node (k,t)

As

$$\left|-\left(\frac{1}{l_{j+1}}\right)^p+\left(\frac{1}{l_{j+1}}-\frac{1}{l_k}\right)^p\right|=\left(\frac{1}{l_{j+1}}\right)^p\left(1-\left(1-\frac{l_{j+1}}{l_k}\right)^p\right)\leq\left(\frac{1}{l_{j+1}}\right)^p\leq\frac{1}{l_j}\left(\frac{1}{l_{j+1}}\right)^{p-1}$$

We can write,

$$\mathrm{Cov}(A_{j,s},A_{k,t})=\left(1-\frac{1}{l_j}\right)^p-\left(1-\frac{1}{l_j}\right)^p\left(1-\frac{1}{l_k}\right)^p+C_1(j,k)\qquad(1.25)$$

With: $|C_1(j,k)|\leq\frac{4}{l_j}\left(\frac{1}{l_{j+1}}\right)^{p-1}$

- Nodes (j,s) and (k,t) are separted: The path between nodes (j,s) and (k,t) passes throughout the tree top (cf figure 1.8)

In this case, the variable $A_{j,s}A_{k,t}$ equals 1 if and only if both sets A and B contain at least one leaf among the p selected ones. So,

$$\begin{aligned}\mathbb{E}(A_{j,s}A_{k,t})&=\mathbb{P}(A\text{ and }B\text{ not empty})\\&=1-\mathbb{P}(A\text{ empty})-\mathbb{P}(B\text{ empty})+\mathbb{P}(A\text{ and }B\text{ empty})\\&=1-\left(1-\frac{1}{l_j}\right)^p-\left(1-\frac{1}{l_k}\right)^p+\left(1-\frac{1}{l_j}-\frac{1}{l_k}\right)^p\end{aligned}$$

Consequently, we get:

$$\mathrm{Cov}(A_{j,s},A_{k,t})=\left(1-\frac{1}{l_j}-\frac{1}{l_k}\right)^p-\left(1-\frac{1}{l_j}\right)^p\left(1-\frac{1}{l_k}\right)^p+C_2(j,k)\qquad(1.26)$$

With: $|C_2(j,k)|\leq\frac{4}{l_j}\left(\frac{1}{l_{j+1}}\right)^{p-1}$

Finally, in order to lighten the notations, we introduce the indicatrice $T(j,k,s,t)$ which verifies:

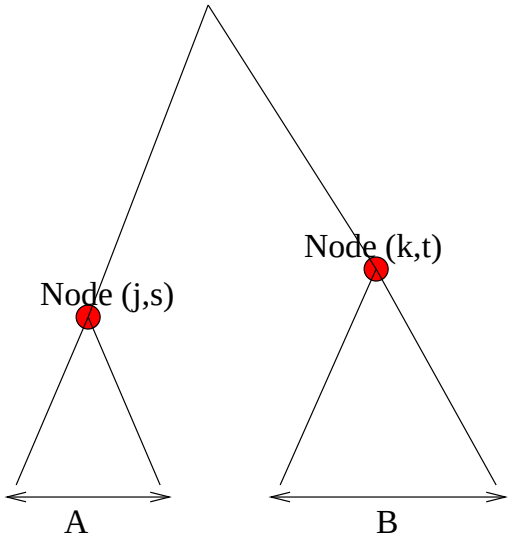


Figure 1.8: Nodes (j,s) and (k,t) are separated

For $0 \leq j < n, \quad 0 \leq k < n, \quad 0 \leq s < l_j, \quad 0 \leq t < l_k$

$$T(j,k,s,t) = \begin{cases} 0 & \text{if} \quad \text{Nodes } (j,s) \text{ and } (k,t) \text{ are identical} \\ 1 & \text{if} \quad \text{Node } (j,s) \text{ is an ancestor of node } (k,t) \\ 2 & \text{if} \quad \text{Nodes } (j,s) \text{ and } (k,t) \text{ are separated} \end{cases}$$

We note also,

$$\mathrm{Cov}(A_{j,s}, A_{k,t}) = \begin{cases} \mathrm{Var}(j) & \text{if} \quad \text{Nodes } (j,s) \text{ and } (k,t) \text{ are identical} \\ \mathrm{Cov}_1(j,k) & \text{if} \quad \text{Node } (j,s) \text{ is an ancestor of node } (k,t) \\ \mathrm{Cov}_2(j,k) & \text{if} \quad \text{Nodes } (j,s) \text{ and } (k,t) \text{ are separated} \end{cases}$$

$$\begin{aligned} \mathrm{Var}(D(N,p)) = & \sum_{T(j,k,s,t)=0} \mathrm{Var}(A_{j,s}, A_{k,t}) \\ & + \sum_{T(j,k,s,t)=1} \mathrm{Cov}(A_{j,s}, A_{k,t}) \\ & + \sum_{T(j,k,s,t)=2} \mathrm{Cov}(A_{j,s}, A_{k,t}) \end{aligned}$$

Therefore, we obtain,

$$\begin{aligned}\mathrm{Var}(D(N,p)) = &\sum_{j=0}^{n-1} l_j \mathrm{Var}(j) \\ &+ 2\sum_{j=0}^{n-2} l_j \sum_{k=j+1}^{n-1} \frac{l_k}{l_j}\mathrm{Cov}_1(j,k) \\ &+ 2\sum_{j=0}^{n-1} l_j \sum_{k=j+1}^{n-1} (l_k - \frac{l_k}{l_j})\mathrm{Cov}_2(j,k) \\ &+ \sum_{j=0}^{n-1} l_j(l_j-1)\mathrm{Cov}_2(j,j)\end{aligned}$$

If we consider $p = \lambda N$, for a fixed λ, we get:

$$\begin{aligned}\mathrm{Var}(D(N,p)) = &\\ &\sum_{j=0}^{n-1} l_j \left(\left(1-\frac{1}{l_j}\right)^p - \left(1-\frac{1}{l_j}\right)^{2p}\right) \\ &+ 2\sum_{j=0}^{n-2}\sum_{k=j+1}^{n-1} l_k \left(\left(1-\frac{1}{l_j}\right)^p - \left(1-\frac{1}{l_j}\right)^p\left(1-\frac{1}{l_k}\right)^p\right) \\ &+ 2\sum_{j=1}^{n-1}\sum_{k=j+1}^{n-1} (l_j l_k - l_k)\left(\left(1-\frac{1}{l_j}-\frac{1}{l_k}\right)^p - \left(1-\frac{1}{l_j}\right)^p\left(1-\frac{1}{l_k}\right)^p\right) \\ &+ \sum_{j=1}^{n-1}(l_j^2 - l_j)\left(\left(1-\frac{2}{l_j}\right)^p - \left(1-\frac{1}{l_j}\right)^{2p}\right) \\ &+ V(N,p)\end{aligned}$$

With,

$$|V(N,p)| \leq \sum_{j=0}^{n-1} l_j \sum_{k=0}^{n-1} l_k \frac{4}{l_j}\left(\frac{1}{l_{j+1}}\right)^{p-1} = 4\sum_{j=1}^{n}\left(\frac{1}{l_j}\right)^{p-1} N$$

Since, $\forall j \geq 1 : l_j \geq 2^j$, we obtain: $|V(N,p)| \leq \frac{4}{2^{p-1}-1}N$ and consequently: $V(N,p) = o(N)$

1.8 Variance asymptotics

1.8.1 Variance asymptotics for large n

We assume that:

$$\forall j \geq 1 : \lim_{n\to+\infty} \frac{N}{l_{n-j}} = a_j \tag{1.27}$$

Changing the variables j and k into $n-j$ and $n-k$, we get the following expression:

$$\begin{aligned}
&\frac{\mathrm{Var}(D(N,p))}{N} = \\
&\sum_{j=1}^{n} \frac{l_{n-j}}{N} \left(\left(1 - \frac{1}{l_{n-j}}\right)^{\lambda N} - \left(1 - \frac{1}{l_{n-j}}\right)^{2\lambda N} \right) \\
&+ 2 \sum_{j=2}^{n} \sum_{k=1}^{j-1} \frac{l_{n-k}}{N} \left(1 - \frac{1}{l_{n-j}}\right)^{\lambda N} \left(1 - \left(1 - \frac{1}{l_{n-k}}\right)^{\lambda N}\right) \\
&+ 2 \sum_{j=1}^{n-1} \sum_{k=1}^{j-1} \left(\frac{l_{n-j}}{N} \frac{l_{n-k}}{N} - \frac{l_{n-k}}{N^2} \right) N \left(\left(1 - \frac{1}{l_{n-j}} - \frac{1}{l_{n-k}}\right)^{\lambda N} - \left(1 - \frac{1}{l_{n-j}}\right)^{\lambda N} \left(1 - \frac{1}{l_{n-k}}\right)^{\lambda N} \right) \\
&+ \sum_{j=1}^{n-1} \left(\frac{l_{n-j}^2}{N^2} - \frac{l_{n-j}}{N^2} \right) N \left(\left(1 - \frac{2}{l_{n-j}}\right)^{\lambda N} - \left(1 - \frac{1}{l_{n-j}}\right)^{2\lambda N} \right) \\
&+ o(1)
\end{aligned}$$

Under the assumption 7.12, it's easy to see that: $\forall j,\ k \geq 1$

$$\begin{aligned}
&\lim_{n\to\infty} \left(1 - \frac{1}{l_{n-j}}\right)^{\lambda N} = e^{-\lambda a_j} \\
&\lim_{n\to\infty} N \left(\left(1 - \frac{1}{l_{n-j}} - \frac{1}{l_{n-k}}\right)^{\lambda N} - \left(1 - \frac{1}{l_{n-j}}\right)^{\lambda N} \left(1 - \frac{1}{l_{n-k}}\right)^{\lambda N} \right) = -a_j a_k \lambda e^{-\lambda a_j - \lambda a_k}
\end{aligned}$$

Then we have:

$$\begin{aligned}
\frac{\mathrm{Var}(D(N,p))}{N} \overset{n\to+\infty}{\longrightarrow} &\sum_{j=1}^{+\infty} e^{-\lambda a_j} \frac{1 - e^{-\lambda a_j}}{a_j} \\
&+ 2 \sum_{j=1}^{+\infty} \sum_{k=1}^{j-1} e^{-\lambda a_j} \frac{1 - e^{-\lambda a_k}}{a_k} \\
&- 2\lambda \sum_{j=1}^{+\infty} \sum_{k=1}^{j-1} e^{-\lambda a_j - \lambda a_k} \\
&- \lambda \sum_{j=1}^{+\infty} e^{-2\lambda a_j}
\end{aligned}$$

We note,

$$V(\lambda) = \lim_{n\to+\infty} \frac{\mathrm{Var}(D(N,p))}{N}$$

Proposition 10. *Finally we can rewrite $V(\lambda)$,*

$$\lim_{n\to+\infty} \frac{\mathrm{Var}(D(N,p))}{N} \overset{def}{=} V(\lambda)$$
$$= \lambda \left(2\sum_{j=1}^{+\infty} e^{-\lambda a_k} \left(\sum_{k=1}^{j} \left(\frac{1-e^{-\lambda a_k}}{\lambda a_k} - e^{-\lambda a_k} \right) \right) - \sum_{j=1}^{+\infty} e^{-\lambda a_j} \left(\frac{1-e^{-\lambda a_j}}{\lambda a_j} - e^{-\lambda a_j} \right) \right)$$

We note:

$$V_1(\lambda) = 2\sum_{j=1}^{+\infty} e^{-\lambda a_k} \left(\sum_{k=1}^{j} \left(\frac{1-e^{-\lambda a_k}}{\lambda a_k} - e^{-\lambda a_k} \right) \right)$$

and

$$V_2(\lambda) = \sum_{j=1}^{+\infty} e^{-\lambda a_j} \left(\frac{1-e^{-\lambda a_j}}{\lambda a_j} - e^{-\lambda a_j} \right)$$

1.8.2 Variance asymptotics for small λ

First, we define the function f as:

$$\forall x \geq 0 \ : f(x) = \sum_{j=1}^{+\infty} \mathbb{1}_{\{x \geq a_j\}}$$

Lemma 1. $\forall x, \lambda > 0 \ \left| f(\frac{x}{\lambda}) - f(\frac{1}{\lambda}) \right| \leq |\ln_2(x)|$

Proof. We assume that $x \geq 1$.
So, $f(\frac{x}{\lambda}) = f(x) + Card\{j \geq 1,\ \frac{1}{\lambda} < a_j \leq \frac{x}{\lambda}\}$. As $\forall j \geq 1,\ a_{j+1} \geq 2a_j,$ we conclude that $Card\{j \geq 1,\ \frac{1}{\lambda} < a_j \leq \frac{x}{\lambda}\} \leq \ln_2(x)$ and the lemma follows. □

Proposition 11.

$$V(\lambda) \sim \lambda^2 a_1 f\left(\frac{1}{\lambda}\right)$$

Proof. We know that: $\forall x \geq 0,\ 0 \leq \frac{1-e^{-x}}{x} - e^{-x} \leq \frac{x}{2}$, so:

$$\sum_{k=1}^{j} \left(\frac{1-e^{-\lambda a_k}}{\lambda a_k} - e^{-\lambda a_k} \right) \leq \frac{\lambda}{2} \sum_{k=1}^{j} a_k$$

As we have, $\forall k \geq 1,\ a_{k+1} \geq 2a_k$, we deduce that $\forall 1 \leq k \leq j,\ a_k \leq \left(\frac{1}{2}\right)^{j-k} a_j$.
Then we get:

$$0 \leq \sum_{k=1}^{j} \left(\frac{1-e^{-\lambda a_k}}{\lambda a_k} - e^{-\lambda a_k} \right) \leq \lambda a_j$$

Consequently:

$$2\sum_{j=1}^{+\infty} e^{-\lambda a_j} \left(\frac{1-e^{-\lambda a_j}}{\lambda a_j} - e^{-\lambda a_j} \right) \leq V_1(\lambda) \leq 2\sum_{j=1}^{+\infty} e^{-\lambda a_j} \lambda a_j$$

$$\begin{aligned}\sum_{j=1}^{+\infty} e^{-\lambda a_j}\lambda a_j &= -\sum_{j=1}^{+\infty}\int_{\lambda a_j}^{+\infty}(xe^{-x})'dx \\ &= \sum_{j=1}^{+\infty}\int_{\lambda a_1}^{+\infty}(x-1)e^{-x}\mathbb{1}_{\{x\geq\lambda a_j\}}dx \\ &= \int_{\lambda a_1}^{+\infty}(x-1)e^{-x}\sum_{j=1}^{+\infty}\mathbb{1}_{\{x\geq\lambda a_j\}}dx \\ &= \int_{\lambda a_1}^{+\infty}(x-1)e^{-x}f\left(\frac{x}{\lambda}\right)dx\end{aligned}$$

Using lemma 1 we obtain:

$$\sum_{j=1}^{+\infty} e^{-\lambda a_j}\lambda a_j \sim \lambda a_1 f\left(\frac{1}{\lambda}\right)$$

Similarly we obtain:

$$\sum_{j=1}^{+\infty} e^{-\lambda a_j}\left(\frac{1-e^{-\lambda a_j}}{\lambda a_j} - e^{-\lambda a_j}\right) \sim \lambda a_1 f\left(\frac{1}{\lambda}\right)$$

Then:

$$V_1(\lambda) \sim 2\lambda a_1 f\left(\frac{1}{\lambda}\right)$$

On the other hand,

$$V_2(\lambda) \sim \lambda a_1 f\left(\frac{1}{\lambda}\right)$$

And the result follows □

1.8.3 Examples

In this part, we explicit the asymptotic for $V(\lambda)$ for the three cases studied previously.

- Regular trees with degree $d \geq 2$

In this case: we have $l_j = d^j$ and $N = \frac{d^n-1}{d-1}$.
Hence, $a_j = \frac{d^j}{d-1}$
We have already seen that in this case $\forall x \geq 1\ f(x) = \lfloor \ln_d(x) \rfloor$ and consequently:

$$V(\lambda) \sim \lambda^2 \frac{d}{d-1}\ln_d\left(\frac{1}{\lambda}\right)$$

- PLD$_\alpha$ trees

Here, for n large, $l_j \sim (n!/(n-j)!)^\alpha$ and $N \sim (n!)^\alpha H(\alpha)$
where $H(\alpha) = \sum_{k=1}^{+\infty} 1/(j!)^\alpha$.
Hence, $a_j = H(\alpha)(j!)^\alpha$
We have also seen that in this case, $\forall x \geq 1\ f(x) = \lfloor \Gamma^{-1}(x) \rfloor$ and consequently:

$$V(\lambda) \sim \lambda^2 H(\alpha) \Gamma^{-1}\left(\frac{1}{\lambda}\right) \sim \lambda^2 H(\alpha) \frac{\ln\left(\frac{1}{\lambda}\right)}{\ln\left(\ln\left(\frac{1}{\lambda}\right)\right)}$$

- PLI_α trees

Here, for n large, $l_j \sim (j!)^\alpha$ and $N \sim ((n-1)!)^\alpha$
Hence, $a_1 = 1$ and $\forall j > 1,\ a_j = +\infty$
Therefore, $\forall x \geq 1\ f(x) = 1$ which gives the following result:

$$V(\lambda) \sim \lambda^2$$

1.9 Conclusion

We have investigated in this chapter the topology discovery of a collect network, represented by means of a tree network, when using ideal traceroute procedures. We have obtained, for different network structures, closed formulas for the mean number and variance of discovered nodes and for the speed of the exploration process when the height of the tree tends to infinity while the ratio of the number traceroute capable hosts to the number of nodes in the network remains constant. We have in particular introduced a connection with the coupon collector problem, which seems to appear as a generic tool in the problem of topology discovery.

From the analysis carried out in this chapter, we can make the following points:

- the rate of the exploration process is quite high with a small number of hosts but rapidly decreases when the number of hosts increases,
- the exhaustive exploration of a network requires a massive deployment of traceroute capable hosts,
- the existence of sparse areas significantly slows down the exploration process unless a coordination exists between the different hosts,
- the degree of nodes is not easy to determine when the number of hosts is small.

The validity of these results will be investigated in further studies for more complex network structures (e.g., random tree networks and complex networks à la Barábasi).

Chapter 2

Galton-Watson trees

- **Keywords: Galton Watson trees**

2.1 Introduction

In this chapter we will explore the efficiency of trace-route methods to discover the Internet topology. The network seen from a single host in the network is a tree. This tree is constructed recursively level by level. Indeed, the hosts at level $n+1$ are the neighbors of those at level n which do not already exist in the tree. Moreover, due to the intrinsic hierarchy of the Internet, the tree assumption is very realistic and only few nodes of the network do not belong to trees. We considered here the case of Galton-Watson trees. A Galon-Watson tree is a random tree where the node degree is a random variable. So, each node generates its own number of children obeying to this variable independently from the other nodes. Actually, the node degree is very skewed and exhibit some heavy tail properties. Indeed, the average node degree is around two but some nodes can have more than thousands of neighbors. This observation has all the more been supported by the fact that it has empirically been observed that the degree of nodes in the Internet obeys a power law distribution as reported in the celebrated paper by Faloutsos *et al* [16]. Galton-Watson trees includes the cases studied in the previous chapter where we studied different deterministic tree profiles and the node degree generation variable was taken constant.

The trace-route efficiency is quantified by the proportion of discovered nodes. We will also derive close formulas for the variance and some expansions when the tree is large.

2.2 Notation and definitions

2.2.1 Assumptions and notation

Throughout this chapter, we assume that the graph of the network is a Galton-Watson tree with height $n \geq 2$. Each node degree is generated independently obeying to a fixed random variable.

Notations

For $n \geq 0$, Z_n denotes the number of individuals at the nth level (nth generation). For $\ell \in \{1, \ldots, Z_n\}$, a node of the tree may be represented as a couple (n, ℓ), where n is its generation and ℓ its rank within the generation. We denote by R_N the total number of nodes between level 1 and level n discovered by the traceroute procedure. The quantity $\mathcal{T}_k^{n,\ell}$ denotes the sub-tree of $\mathcal{T}$ whose root is (n, ℓ) and containing all the nodes below (n, ℓ) whose depth is less or equal to k. The size of $\mathcal{T}_k^{n,\ell}$ is denoted as $T_k^{n,\ell}$. When the upper index (n, ℓ) is omitted, it indicates a random variable with the same distribution as the size of the tree unitl the kth generation, i.e. with the same distribution as $Z_1 + \cdots + Z_k$.

With these notations, it is easy to derive the relation

$$T_N = \sum_{i=0}^{n-1} Z_i + \sum_{\ell=1}^{Z_n} T_{N-n}^{n,\ell}. \tag{2.1}$$

The variable $\mathcal{N}$ is a counting measure on the tree representing the distribution of the points selected in the tree, for a subset A of the nodes of the tree, $\mathcal{N}(A)$ denotes the total number of points in A. A node ℓ of the tree $\mathcal{T}$ at level n belongs to the discovered tree whenever $\mathcal{N}(T_{N-n}^{\ell})$ is not 0, in particular the total size R_n of the discovered tree is given by

$$R_N = \sum_{n=0}^{N} \sum_{\ell=1}^{Z_{N-n}} \mathbf{1}\mathcal{N}(T_n^{N-n,\ell}) \neq 0 \tag{2.2}$$

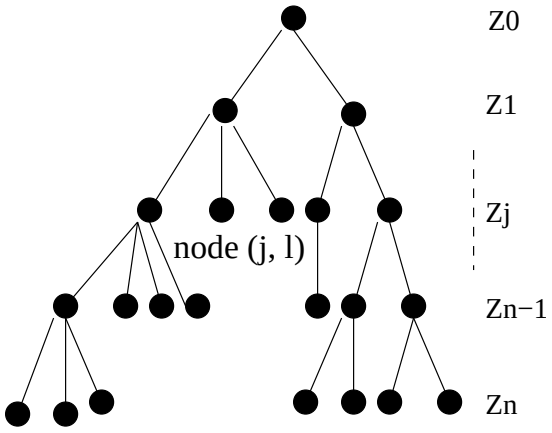


Figure 2.1: Galton-Watson tree

2.3 Asymptotic expansion in the uniform case

2.3.1 The expansion of the first moment of $\mathbf{R}_N$

Theorem 1. *The ratio of the average size R_N of the discovered tree with density $\lambda > 0$ and the total size of the tree T_N satisfies the relation*

$$M_1(\lambda) \stackrel{def.}{=} \lim_{N\to+\infty} \frac{\mathbb{E}(R_N)}{\mathbb{E}(T_N)} = \sum_{n=0}^{+\infty} \frac{m-1}{m^{n+1}} \left(1 - \mathbb{E}\left[\exp\left(-\lambda \sum_1^n Z_i\right)\right]\right).$$

If additionally the condition $\mathbb{E}(G\log G) < +\infty$ holds then, when $\lambda \searrow 0$,

$$M_1(\lambda) \sim \frac{\lambda m}{m-1}\left(\log_m \lambda + \log_m\left(\frac{m}{m-1}\right) + o(\lambda)\right). \tag{2.3}$$

Because of the term $\log_m \lambda$, Relation (8.3) states that the size of the multi-cast tree grows *very* quickly when few points are selected in the tree. In particular the derivative of M_1 at the origin is infinite.

Proof. The probability that the node ℓ at level n does not belong to the multi-cast tree is given by

$$\mathbb{P}\left(\mathcal{N}\left(\mathcal{T}_n^{N-n,\ell}\right) \neq 0\right) = 1 - \exp\left(-\lambda T_n^{N-n,\ell}\right).$$

By summing-up these relations, one gets that the expected value of R_N, the average number of nodes in the discovered tree is given by

$$\begin{aligned}\mathbb{E}(R_N) &= \sum_{n=0}^{N} \mathbb{E}(Z_{N-n})\left(1 - \mathbb{E}\left[\exp\left(-\lambda T_n\right)\right]\right) \\ &= \sum_{n=0}^{N} m^{N-n}\left(1 - \mathbb{E}\left[\exp\left(-\lambda \sum_1^n Z_i\right)\right]\right).\end{aligned}$$

Since $\mathbb{E}(G\log G) < +\infty$, Kesten-Stigum's result gives the existence of a non-negative random variable W such that $\mathbb{P}(W > 0) = 1$ (because of the assumption on the distribution of G) and $\mathbb{E}(W) = 1$ (See Lyons and Peres [31]) and that, almost surely,

$$\lim_{n\to+\infty} \frac{Z_n}{m^n} = W.$$

The asymptotic ratio of nodes in the discovered tree and in the initial tree is given by

$$M_1(\lambda) \stackrel{\text{def.}}{=} \lim_{N\to+\infty} \frac{\mathbb{E}(R_N)}{\mathbb{E}(T_N)} = \sum_{n=0}^{+\infty} \frac{m-1}{m^{n+1}} \left(1 - \mathbb{E}\left[\exp\left(-\lambda \sum_1^n Z_i\right)\right]\right)$$

and if

$$f(\lambda) \stackrel{\text{def.}}{=} \sum_{n=0}^{+\infty} \frac{m-1}{m^{n+1}} \left(1 - \mathbb{E}\left[\exp\left(-\lambda W \frac{m^{n+1}}{m-1}\right)\right]\right),$$

then

$$\frac{|M_1(\lambda) - f(\lambda)|}{\lambda} \leq \sum_{n=0}^{+\infty} \frac{m-1}{\lambda m^{n+1}} \left| \mathbb{E}\left[\exp\left(-\lambda \sum_1^n Z_i\right)\right] - \mathbb{E}\left[\exp\left(-\lambda W \frac{m^{n+1}}{m-1}\right)\right]\right|. \quad (2.4)$$

Since W is integrable, Lebesgue's Theorem gives that

$$\lim_{\lambda \to 0} \frac{1}{\lambda} \mathbb{E}\left(\exp\left(-\lambda \sum_1^n Z_i\right) - \exp\left(-\lambda W \frac{m^{n+1}}{m-1}\right)\right) = \mathbb{E}\left(\sum_1^n Z_i - \lambda W \frac{m^{n+1}}{m-1}\right) = 0 \quad (2.5)$$

One has

$$\frac{1}{m^{n+1}\lambda} \left| \mathbb{E}\left[\exp\left(-\lambda \sum_1^n Z_i\right)\right] - \mathbb{E}\left[\exp\left(-\lambda W \frac{m^{n+1}}{m-1}\right)\right]\right| \leq \frac{1}{m^{n+1}} \sum_{i=1}^n \mathbb{E}|Z_i - m^i W|.$$

According to Athreya and Ney [2] Theorem 1 page 54, for $n \geq 1$, there exists a sequence (W^i) of i.i.d. random variables with the same distribution as W such that,

$$Z_n - m^n W = \sum_{i=1}^{Z_n} (1 - W^i)$$

therefore, using Cauchy-Shwartz Inequality, one gets

$$\begin{aligned} |\mathbb{E}\left(Z_n - m^n W\right)| &\leq \sqrt{\mathbb{E}\left((Z_i - m^i W)^2\right)} \\ &= \mathrm{Var}(1-W) \sum_{i=1}^n \sqrt{\mathbb{E}(Z_i)} \\ &= \mathrm{Var}(1-W) \sum_{i=1}^n m^{i/2} \leq \frac{\mathrm{Var}(1-W)}{m-1} m^{(n+1)/2}. \end{aligned}$$

With this relation, one obtains

$$\frac{1}{m^{n+1}\lambda} \left| \mathbb{E}\left[\exp\left(-\lambda \sum_1^n Z_i\right)\right] - \mathbb{E}\left[\exp\left(-\lambda W \frac{m^{n+1}}{m-1}\right)\right]\right| \leq \frac{\mathrm{Var}(1-W)}{m-1} \frac{1}{m^{(n+1)/2}},$$

Relation (2.5) and Lebesgue's Theorem give that

$$\lim_{\lambda \to 0} \frac{M_1(\lambda) - f(\lambda)}{\lambda} = 0.$$

Hence, up to an expression which is of the order of $o(\lambda)$, the behavior at 0 of $M_1(\lambda)$ is equivalent to the behavior of $f(\lambda)$ as λ gets small. One gets that, if $h(u) = 1 - e^{-u}$ and $x = \lambda m/(m-1)$,

$$f(\lambda) = \sum_{n=0}^{+\infty} \frac{m-1}{m^{n+1}} \left(1 - \mathbb{E}\left[\exp\left(-xWm^n\right)\right]\right) = \Psi(h)(x) \sim x \log_m x,$$

by Proposition 15. The proposition is proved. □

2.3.2 The variance of $\mathbf{R}_N$

Notation.

If (n,ℓ) and (n',ℓ') are two nodes of the tree, the relation $(n,\ell) < (n',\ell')$ is said to hold whenever the nodes are distinct and if $\mathcal{T}_{N-n'}^{n',\ell'} \subset \mathcal{T}_{N-n}^{n,\ell}$, i.e. when (n',ℓ') is a node of the sub-tree $\mathcal{T}_{N-n}^{n,\ell}$ but not the root.

Proposition 12 (Asymptotic behavior of the variance)**.**

$$M_2(\lambda) \stackrel{def.}{=} \lim_{N\to+\infty} \frac{\mathrm{Var}(R_N)}{\mathbb{E}(T_N)} = \frac{m-1}{m} \sum_{n=1}^{+\infty} \frac{1}{m^n} \left[\mathbb{E}\left(e^{-\lambda T_n}\right)\left(1 - \mathbb{E}\left(e^{-\lambda T_n}\right)\right) \right.$$
$$\left. +2\sum_{k=0}^{n-1} \mathbb{E}\left[e^{-\lambda T_{n-k-1}} \mathbb{E}\left(e^{-\lambda T_k}\right)^{Z_{n-k}-1} \mathbb{E}\left(e^{-\lambda T_k}\left(1 - e^{-\lambda T_k}\right)\right)\right]\right] \quad (2.6)$$

Proof. Using Representation (8.2) for the size of the multi-cast tree, one gets the relation

$$R_N - \mathbb{E}(R_N) = \sum_{n=0}^{N} \sum_{\ell=1}^{Z_{N-n}} \Delta_n^\ell$$

with

$$\Delta_n^\ell = \mathbf{1}\mathcal{N}(T_n^{N-n,\ell}) \neq 0 - \mathbb{P}\left(\mathcal{N}(T_n^{N-n,\ell}) \neq 0\right).$$

By using the idependence properties of the Poisson distribution of the points in the nodes of the tree, one obtains the identity

$$\mathrm{Var}(R_N) - \sum_{n=0}^{N} \mathbb{E}(Z_{N-n})\mathrm{Var}(\Delta_n) = 2\mathbb{E}\left(\sum_{\substack{(n,\ell),(n',\ell')\in\mathcal{T} \\ (n',\ell')<(n,\ell)}} \Delta_n^\ell \Delta_{n'}^{\ell'} \right),$$

then, conditionally on the state of the tree, when $(n',\ell') < (n,\ell)$ one gets the identity

$$\mathbb{E}\left(\Delta_n^\ell \Delta_{n'}^{\ell'} \mid \mathcal{T}\right) = \exp\left(-\lambda T_n\right)\left(1 - \exp\left(-\lambda T_{n'}\right)\right).$$

By symmetry, this gives the following expression for the variance

$$U_n \stackrel{\text{def.}}{=} \mathrm{Var}(R_N) - \sum_{n=0}^{N} \mathbb{E}(Z_{N-n})\mathrm{Var}(\Delta_n)$$

$$= 2\mathbb{E}\left(\sum_{n=1}^{N} Z_{N-n} \exp\left(-\lambda T_n\right) \sum_{\substack{(N-n',l')\in\mathcal{T} \\ (N-n',l')<(N-n,1)}} \left(1 - \exp(\left(-\lambda T_{n'}^{N-n',\ell'}\right)\right) \right).$$

For two nodes of the tree such that $(N-n', l') < (N-n, 1)$, Equation (8.1) gives the relation

$$T_n \stackrel{\text{dist.}}{=} \sum_{k=0}^{n-n'-1} \widetilde{Z}_k + \sum_{\ell'=1}^{\widetilde{Z}_{n-n'}} T_{n'}^{N-n',\ell'},$$

where $(\widetilde{Z}_k, k \geq 0$ denotes another independent Galton-Watson process independent of (Z_n). By using this relation in the last expression for U_N, one gets

$$U_n = 2\sum_{n=1}^{N} \mathbb{E}(Z_{N-n}) \sum_{n'=0}^{n-1} \mathbb{E}\left[\exp\left(-\lambda \sum_{k=0}^{n-n'-1} Z_k \right) \right.$$

$$\left. \mathbb{E}\left[\exp\left(-\lambda T_{n'}\right) \right]^{Z_{n-n'}-1} \mathbb{E}\left[\exp\left(-\lambda T_{n'}\right)\left(1 - \exp\left(-\lambda T_{n'}\right)\right) \right] \right].$$

By dividing by $\mathbb{E}(T_N)$ on gets

$$\frac{U_n}{\mathbb{E}(T_N)} = \frac{2(m-1)}{m} \sum_{n=1}^{N} \frac{1}{m^n} \sum_{n'=0}^{n-1} \mathbb{E}\left[\exp\left(-\lambda \sum_{k=0}^{n-n'-1} Z_k \right) \right.$$

$$\left. \mathbb{E}\left[\exp\left(-\lambda T_{n'}\right) \right]^{Z_{n-n'}-1} \mathbb{E}\left[\exp\left(-\lambda T_{n'}\right)\left(1 - \exp\left(-\lambda T_{n'}\right)\right) \right] \right].$$

By letting N go to infinity, one obtains Relation (8.4). The proposition is proved. □

Proposition 13 (Asymptotic Behavior of $\lambda \to M_2(\lambda)$ at 0)**.** *Under the condition* $\mathbb{E}(G \log G) < +\infty$ *then,*

$$\lim_{\lambda \searrow 0} \frac{M_2(\lambda)}{\lambda \log_m \lambda} = \frac{m(m+1)}{(m-1)^2}. \tag{2.7}$$

Proof. Define

$$f_a(\lambda) \stackrel{\text{def.}}{=} \sum_{n=1}^{+\infty} \frac{m-1}{m^n} \left[\mathbb{E}\left(e^{-\lambda T_n}\right)\left(1 - \mathbb{E}\left(e^{-\lambda T_n}\right)\right) \right],$$

$$f_b(\lambda) \stackrel{\text{def.}}{=} \sum_{n=1}^{+\infty} \frac{m-1}{m^n} \sum_{k=0}^{n-1} \mathbb{E}\left[e^{-\lambda T_{n-k-1}} \mathbb{E}\left(e^{-\lambda T_k}\right)^{Z_{n-k}-1} \right] \mathbb{E}\left(e^{-\lambda T_k}\left(1 - e^{-\lambda T_k}\right)\right),$$

Equation (8.4) gives that $mM_2(\lambda) = 2f_b(\lambda) + f_a(\lambda)$.

Asymptotic behavior of f_a With similar arguments as in the proof of Theorem 2 the asymptotic behavior of $f_a(\lambda)$ when λ goes to 0 is equivalent to the asymptotic behavior of

$$\sum_{n=1}^{+\infty} \frac{m-1}{m^n} \left[\mathbb{E}\left(\exp\left(-\lambda \frac{W m^{n+1}}{m-1} \right) \right) \left(1 - \mathbb{E}\left(\exp\left(-\lambda \frac{W m^{n+1}}{m-1} \right) \right) \right) \right].$$

If W_1 and W_2 are two independent random variables with the same distribution as W, the above series can be rewritten as

$$\sum_{n=1}^{+\infty} \frac{m-1}{m^n} \mathbb{E}\left[\exp\left(-\lambda \frac{W_1 m^{n+1}}{m-1} \right) \left(1 - \mathbb{E}\left(\exp\left(-\lambda \frac{W_2 m^{n+1}}{m-1} \right) \right) \right) \right]$$
$$= (m-1) \sum_{n=1}^{+\infty} \left[\frac{1}{m^n} \mathbb{E}(h(x(W_1+W_2)m^n)) - \mathbb{E}(h(xW_1 m^n)) \right],$$

with $x = \lambda m/(m-1)$ and $h(u) = 1 - e^{-u}$. Consequenlty,

$$\lim_{\lambda \to 0} \frac{f_a(\lambda)}{\lambda \log_m \lambda} = \frac{m}{m-1}, \tag{2.8}$$

by Proposition 15.

Asymptotic behavior of f_b The following inequality is straighforward,

$$f_b(\lambda) \leq \sum_{n=1}^{+\infty} \frac{m-1}{m^n} \sum_{k=0}^{n-1} \mathbb{E}\left(e^{-\lambda T_k} \left(1 - e^{-\lambda T_k} \right) \right). \tag{2.9}$$

On the other hand for $0 \leq k < n$,

$$\mathbb{E}\left[e^{-\lambda T_{n-k-1}} \ \mathbb{E}\left(e^{-\lambda T_k} \right)^{Z_{n-k}-1} \right] \geq e^{-\lambda \mathbb{E}(T_{n-k-1})} \mathbb{E}\left(e^{-\lambda T_k} \right)^{\mathbb{E}(Z_{n-k})-1}$$
$$\geq \exp\left(-\lambda \frac{m^{n-k}}{m-1} \right) \left(e^{-\lambda \mathbb{E}(T_k)} \right)^{m^{n-k}-1}$$
$$\geq \exp\left(-\lambda \frac{m^{n-k}}{m-1} \right) \left(\exp\left(-\lambda \frac{m^{k+1}}{m-1} \right) \right)^{m^{n-k}-1} \geq \exp\left(-2\lambda \frac{m^{n+1}}{m-1} \right),$$

by a repeated use of Jensen's inequality. Finally one gets that

$$f_b(\lambda) \geq \sum_{n=1}^{+\infty} \frac{m-1}{m^n} \exp\left(-2\lambda \frac{m^{n+1}}{m-1} \right) \sum_{k=0}^{n-1} \mathbb{E}\left(e^{-\lambda T_k} \left(1 - e^{-\lambda T_k} \right) \right). \tag{2.10}$$

The difference between the upper bound (2.9) and the lower bound (2.10) is given by

$$\Delta(\lambda) = \sum_{n=1}^{+\infty} \frac{m-1}{m^n} \left(1 - \exp\left(-2\lambda \frac{m^{n+1}}{m-1} \right) \right) \sum_{k=0}^{n-1} \mathbb{E}\left(e^{-\lambda T_k} \left(1 - e^{-\lambda T_k} \right) \right)$$
$$\leq \sum_{n=1}^{+\infty} \frac{m-1}{m^n} \left(1 - \exp\left(-2\lambda \frac{m^{n+1}}{m-1} \right) \right) \sum_{k=0}^{n-1} \left(1 - e^{-\lambda \mathbb{E}(T_k)} \right),$$

by Jensen's Inequality again. Hence, if $x = \lambda m/(m-1)$, $h_1(u) = 1 - e^{-2x}$ and $h_2(x) = 1 - e^{-x}$ then,

$$\Delta(\lambda) \leq (m-1)\sum_{n=1}^{+\infty} \frac{h_1(xm^n)}{m^n} \sum_{k=0}^{n-1} h_2(xm^k) = (m-1)\sum_{n=1}^{+\infty} h_2(xm^k) \sum_{n>k} \frac{h_1(xm^n)}{m^n}$$
$$= (m-1)\sum_{n=1}^{+\infty} \frac{h_2(xm^k)}{m^k} \sum_{n\geq 1} (m-1)\frac{h_1(xm^{n+k})}{m^n} = \Psi\left(h_1\Psi(h_2)\right)(x)$$

By Proposition 15, as x goes to 0,

$$\frac{h_1(x)\Psi(h_2)(x)}{x} \sim \frac{2x^2 \log_m(x)}{x} \to 0,$$

Hence, $\lim_{\lambda\to 0} \Delta(\lambda)/(\lambda \log_m \lambda) = 0$, so

$$\frac{f_b(\lambda)}{\lambda \log_m \lambda} \sim \frac{1}{\lambda \log_m \lambda} \sum_{n=1}^{+\infty} \frac{m-1}{m^n} \sum_{k=0}^{n-1} \mathbb{E}\left(e^{-\lambda T_k}\left(1 - e^{-\lambda T_k}\right)\right).$$

Again with similar arguments as in the proof of Theorem 2, on the scale $\lambda \log_m \lambda$, it is enough to study the behavior of

$$\sum_{n=1}^{+\infty} \frac{m-1}{m^n} \sum_{k=0}^{n-1} \mathbb{E}\left[\exp\left(-\lambda \frac{Wm^{k+1}}{m-1}\right)\left(1 - \exp\left(-\lambda \frac{Wm^{k+1}}{m-1}\right)\right)\right]$$
$$= \sum_{n=1}^{+\infty} \frac{m-1}{m^n} \sum_{k=0}^{n-1} \mathbb{E}(h_1(xWm^k)),$$

as λ goes to 0, with $h(u) = e^{-u}(1 - e^{-u})$ and $x = \lambda m/(m-1)$. Since

$$\sum_{n=1}^{+\infty} \frac{m-1}{m^n} \sum_{k=0}^{n-1} \mathbb{E}(h_1(xWm^k))$$
$$= (m-1)\sum_{k=1}^{+\infty} \mathbb{E}(h_1(xWm^k)) \sum_{n>k} \frac{1}{m^n} = \sum_{k=1}^{+\infty} \frac{\mathbb{E}(h_1(xWm^k))}{m^k},$$

therefore, by Proposition 15, one gets the equivalence

$$\lim_{\lambda\to 0} \frac{f_b(\lambda)}{\lambda \log_m \lambda} = \frac{m}{(m-1)^2}. \tag{2.11}$$

Relations (2.8) and (2.11) complete the proof of the proposition. □

2.4 Asymptotic expansion in the depth biased case

It is assumed in this section that $\alpha \in [0, 1)$ and, for $n \geq 0$, the number of points at level n of the tree has a Poisson distribution with parameter α^n and that these

points are uniformly distributed among the nodes of this level. Consequently, the average total number of points is $1/(1-\alpha)$.

The previous notations concerning the tree $\mathcal{T}$ are kept. For $n \geq 0$ and $1 \leq \ell \leq Z_n$, the quantity $\mathcal{T}^{n,\ell}$ denotes the sub-tree of $\mathcal{T}$ whose root is (n,ℓ). the average total number of points in $\mathcal{T}^{n,\ell}$ is $(\alpha/m)^n/(1-\alpha)$ Note that the depth of this sub-tree is not limited at it was the case in the uniform model.

As in the uniform case, a node ℓ of the tree $\mathcal{T}$ at level n belongs to the multi-cast tree whenever $\mathcal{N}(T^{\ell,n})$ is not 0, in particular the total size S_n of the multi-cast tree is given by

$$S_\alpha = \sum_{n=0}^{+\infty} \sum_{\ell=1}^{Z_n} \mathbf{1}\mathcal{N}(\mathcal{T}^{\ell,n}) \neq 0 \tag{2.12}$$

$$\mathbb{E}\left(S_\alpha\right) = \sum_{n=0}^{+\infty} \mathbb{E}(Z_n)\mathbb{P}\left(\mathcal{N}(\mathcal{T}^{\ell,n}) \neq 0\right) = \sum_{n=0}^{+\infty} m^n \left(1 - \exp\left(-\left(\frac{\alpha}{m}\right)^n \frac{1}{1-\alpha}\right)\right)$$

The growth rate of the exploration process is then given by

$$\frac{\mathbb{E}\left(S_\alpha\right)}{\mathbb{E}(\mathcal{N}(\mathcal{T}))} = \sum_{n=0}^{+\infty} (1-\alpha) m^n \left(1 - \exp\left(-\left(\frac{\alpha}{m}\right)^n \frac{1}{1-\alpha}\right)\right),$$

following a similar method as in the proof of Proposition 15 in the appendix, by writing $1-\exp(-u)$ as the integral of $\exp(-t)$ on $[0,u]$ and by using Fubini's Theorem, one gets the representation

$$\begin{aligned}(m-1)\frac{\mathbb{E}\left(S_\alpha\right)}{\mathbb{E}(\mathcal{N}(\mathcal{T}))} &= (1-\alpha)\int_0^{1/(1-\alpha)} \left(m^{\lceil \log[1/(u(1-\alpha))]/\log[m/\alpha]\rceil} - 1\right) e^{-u}\, du \\ &= (1-\alpha)\int_0^{1/(1-\alpha)} \frac{m^{\{\log_{m/\alpha}[1/(u(1-\alpha))]\}}}{m^{\log_{m/\alpha}(u(1-\alpha)}} e^{-u}\, du - (1-\alpha)\left(1 - e^{-1/(1-\alpha)}\right),\end{aligned}$$

where $\{z\} = \lceil z \rceil - z$.

$$\begin{aligned}&(1-\alpha)\int_0^{1/(1-\alpha)} \frac{m^{\{\log_{m/\alpha}[1/(u(1-\alpha))]\}}}{m^{\log_{m/\alpha} u(1-\alpha)}} e^{-u}\, du \\ &\qquad = (1-\alpha)^{-\log\alpha/(m-\log\alpha)} \int_0^{1/(1-\alpha)} \frac{m^{\{\log_{m/\alpha}[1/(u(1-\alpha))]\}}}{u^{\log m/\log(m/\alpha)}} e^{-u}\, du\end{aligned}$$

Note that

$$H(\alpha) \leq \int_0^{1/(1-\alpha)} \frac{m^{\{\log_{m/\alpha}[1/(u(1-\alpha))]\}}}{u^{\log m/\log(m/\alpha)}} e^{-u}\, du \leq mH(\alpha)$$

with

$$H(\alpha) = \int_0^{1/(1-\alpha)} \frac{1}{u^{\log m/\log(m/\alpha)}} e^{-u}\, du,$$

by integrating by parts, one has the equivalence $H(\alpha) \sim 1/\log\alpha$ as $\alpha \nearrow 1$. On deduces the following proposition.

Proposition 14. *As $\alpha \nearrow 1$, the equivalence*

$$\mathbb{E}\left(S_\alpha\right) \sim \frac{1}{(1-\alpha)} \int_0^{+\infty} \frac{m^{\{\log_{m/\alpha}[1/(u(1-\alpha))]\}}}{u^{\log m/\log(m/\alpha)}} e^{-u}\, du \tag{2.13}$$

holds, so that the growth rate of the multi-cast tree satisfies the relation

$$\frac{1}{m-1} \leq \liminf_{\alpha \nearrow 1} (1-\alpha)^2 \mathbb{E}\left(S_\alpha\right) \leq \limsup_{\alpha \nearrow 1} (1-\alpha)^2 \mathbb{E}\left(S_\alpha\right) \leq \frac{m}{m-1},$$

recall that the average number of points is $\mathbb{E}(\mathcal{N}(\mathcal{T})) = 1/(1-\alpha)$.

2.5 Appendix

The proposition is helpful to derive the asymptotics of the size of the discovered tree. Its proof relies on a simple rewriting of the expression to expand, together with Fubini's Theorem. See Robert [42].

Proposition 15. *If V is a positive integrable random variable, $m > 1$ and h a non-negative bounded differentiable function on $[0, +\infty)$ such that $h(0) = 0$ and the function $|h'|$ is in $L_1(\mathbb{R}_+, du)$, if*

$$\Psi(h)(x) = \sum_{n=0}^{+\infty} \frac{m-1}{m^n} \mathbb{E}\left(h\left(xVm^n\right)\right), \quad x \geq 0, \tag{2.14}$$

then

$$\lim_{x \to 0} \frac{\Psi(h)(x)}{-x \log_m(x)} = \mathbb{E}(V) h'(0).$$

Proof. The function h being bounded and $m > 1$ the function $\Psi(h)$ is well defined. If $h'(0) > 0$, Fatou's Lemma applied successively gives the relation

$$\begin{aligned} \liminf_{x \to 0} \frac{\Psi(h)(x)}{x} &\geq \sum_{n=0}^{+\infty} \liminf_{x \to 0} \frac{m-1}{m^n} \mathbb{E}\left(\frac{h\left(xVm^n\right)}{x}\right) \\ &\geq \sum_{n=0}^{+\infty} \frac{m-1}{m^n} \mathbb{E}\left(\liminf_{x \to 0} \frac{h\left(xVm^n\right)}{x}\right) = \sum_{n=0}^{+\infty} (m-1) \mathbb{E}\left(V\right) h'(0) = +\infty, \end{aligned}$$

therefore the ratio $\Psi(h)(x)/x)$ diverges as $x \to 0$.

Since h is non-negative and $|h'|$ integrable with respect to Lebesgue measure on $\mathbb{R}_+$, Fubini's Theorem applied two times shows that $\Psi(h)$ can be expressed

as

$$\Psi(h)(x) = \sum_{n=0}^{+\infty} \frac{m-1}{m^n} \mathbb{E}\left(h\left(xVm^n\right)\right) = \mathbb{E}\left(\sum_{n=0}^{+\infty} \frac{m-1}{m^n} h\left(xVm^n\right)\right)$$
$$= \mathbb{E}\left(\sum_{n=0}^{+\infty} \frac{m-1}{m^n} \int_0^{+\infty} h'(u)\mathbf{1}u \le xVm^n \, du\right)$$
$$= \mathbb{E}\left(\int_0^{+\infty} h'(u) \sum_{n=0}^{+\infty} \frac{m-1}{m^n} \mathbf{1}u \le xVm^n \, du\right)$$
$$= \mathbb{E}\left(\int_0^{xV} h'(u)\, du\right) + \mathbb{E}\left(\int_{xV}^{+\infty} \frac{1}{m^{\lfloor \log_m(u/xV) \rfloor}} h'(u)\, du\right),$$

where $\lfloor y \rfloor$ is the integer part of $y \in \mathbb{R}$. Since the ratio $\Psi(h)(x)/x$ diverges as $x \to 0$, clearly

$$\frac{\Psi(h)(x)}{x} \sim \mathbb{E}\left(\int_{xV}^{+\infty} \frac{1}{xm^{\lfloor \log_m(u/xV) \rfloor}} h'(u)\, du\right),$$

Using that, for $u \ge V$, one has

$$xm^{\lfloor \log_m(u/xV) \rfloor} \ge xm^{\lfloor \log_m(1/x) \rfloor} \ge xm^{\log_m(1/x)-1} = m^{-1},$$

hence

$$\mathbb{E}\left(\int_V^{+\infty} \frac{1}{xm^{\lfloor \log_m(u/xV) \rfloor}} |h'(u)|\, du\right) \le m^{-1} \int_0^{+\infty} |h'(u)|\, du.$$

One gets therefore the equivalence, as $x \to 0$

$$\frac{\Psi(h)(x)}{x} \sim \mathbb{E}\left(\int_{xV}^{V} \frac{1}{xm^{\lfloor \log_m(u/xV) \rfloor}} h'(u)\, du\right) = \mathbb{E}\left(V \int_{xV}^{V} m^{\{\log_m(u/xV)\}} \frac{h'(u)}{u}\, du\right)$$
$$= \mathbb{E}\left(V \int_{xV}^{V} m^{\{\log_m(u/xV)\}} \frac{h'(u) - h'(0)}{u}\, du\right) + h'(0)\mathbb{E}\left(V \int_{xV}^{V} \frac{m^{\{\log_m(u/xV)\}}}{u}\, du\right),$$

where $\{y\} = y - \lfloor y \rfloor$ is the fractional value of $y \in \mathbb{R}$. Since the first term of the right hand side of the last equation is bounded as x goes to 0, only the second term has to be considered, for $x < 1$,

$$\int_{xV}^{V} \frac{m^{\{\log_m(u/xV)\}}}{u}\, du = \int_1^{1/x} \frac{m^{\{\log_m(u)\}}}{u}\, du$$
$$= \sum_{k \ge 0 : m^k \le 1/x} \int_{m^k}^{m^{k+1}} \frac{m^{\{\log_m(u)\}}}{u}\, du + O(1) = \lfloor -\log_m(x) \rfloor + O(1).$$

It is then quite easy to conclude the proof of the proposition. □

2.6 Conclusion

We have extended the results obtained for deterministic trees to the case of Galton-Watson trees. We derived closed formulas for the mean and the variance of the number of discovered nodes. We also derived expansions when the network size is large to have more insight on the exploration speed. In the first order, a Galton-Watson tree of average node degree equal to d can be approximated by a regular tree of constant degree d.

Chapitre 3

Détection des grands flots

On s'intéresse ici aux algorithmes de comptage des différents flots traversant un routeur. Plus particulièrement on veut estimer le nombre des éléphants, c'est à dire les flots qui dépassent 20 paquets. Un flot peut être caractérisé de différentes façons en prenant par exemple comme identifiant les adresses IP et numéros de port des machines source et destination. Les techniques utilisées récemment dans ce domaine ne sont pas adaptées au trafic Internet et souffre de problèmes de passage à l'échelle ou d'adaptation. On va donc présenter deux contributions algorithmiques pour pallier à ces problèmes dans les sections 3.5 et 3.7.

– **Mots clés : Eléphants, filtres de bloom, hachage**

3.1 Introduction

Les mesures de trafic sont très importantes pour la facturation des clients en fonction de l'usage [12], pour l'ingénierie du trafic et pour la détection d'anomalies. En effet, la distribution des tailles des flots peut servir pour détecter les attaques DDoS [32]. Si l'attaquant utilise des adresses IP usurpées, on notera alors une augmentation significative du nombre de flots de taille 1. Dans le cas aussi des vers qui se propagent sur le réseau, si ces vers ne changent pas de taille au cours de leur propagation, on observera une augmentation brutale des flots d'une certaine taille. En plus les fournisseurs d'accès Internet peuvent induire les applications utilisées par les clients (Peer-to-Peer, voix sur IP, web, ftp...) sans même regarder les contenus des paquets. Afin de détecter les intrusions et les ports scan, les algorithmes répandus enregistrent une entrée pour chaque connexion active. Mais ceci souffre du problème de passage à l'échelle, en plus n'est clairement pas nécessaire puisque seulement une petite partie de ces connexions est responsable des intrusions. Ainsi il faut aussi que l'algorithme soit capable d'écarter rapidement les flots normaux et de maintenir seulement des entrées pour les flots suspects qui ont ouvert plus qu'un certain nombre de connexions pendant un intervalle de temps court. Les approches récentes focalisent sur les grands flots (ceux qui dépassent un certain nombre de paquets ou consomment plus qu'un certain pourcentage de la bande passante). En effet, on sait que seulement une petite partie des flots constitue la majorité du trafic, par

exemple [18] montre que 9% des flots circulant entre les AS constituent 90% du volume total d'octets échangés entre toutes les paires AS. Pour beaucoup d'applications, la connaissance de ces grands flots est suffisante pour caractériser le trafic. Une grande partie des algorithmes proposés essayent d'inférer les statistiques des flots à partir des flots échantillonnés récupérés par NetFlow. Ceci conduit à des erreurs très grandes à cause du taux d'échantillonage maintenu faible pour ne pas surcharger le routeur. Après avoir rappelé quelques techniques utilisées récemment dans ce domaine, surtout les filtres de Bloom et de Varghese, et partant du constat que ces techniques ne sont pas adaptées au trafic Internet, on va présenter dans les sections 3.5 et 3.7 deux algorithmes. Le premier utilise les filtres de Bloom comme brique de base et introduit un mécanisme adaptatif pour vider régulièrement les filtres avant qu'ils ne s'engorgent, le deuxième s'appuie sur le fait que les paquets d'un éléphant arrivent en rafale et donc crée des entrées seulement pour ces flots arrivant en rafale.

Remerciements :

Nous tenons à remercier Philippe SULTAN de l'équipe MIRIAD de l'INRIA de nous avoir fourni gracieusement les traces sur lesquelles nous avons effectué les mesures.

3.2 Filtre de Bloom

Un filtre de Bloom est un algorithme qui permet d'estimer rapidement en quelques opérations seulement si un élément appartient à un ensemble $A = \{F_1, F_2, \ldots, F_n\}$ avec grande probabilité. Il suffit de stocker une représentation abstraite de cet ensemble. Cet algorithme a été inventé par Burton BLOOM en 1970 [4]. L'idée derrière cet algorithme est de représenter l'ensemble A sous forme d'un vecteur de $M = km$ bits. On utilise k fonctions de hachage h_i aléatoires et indépendantes. Initialement tous les bits sont initialisés à 0. Il y a deux façons de construire un filtre de Bloom, en effet la première consiste à scinder la mémoire totale en k étages indépendants, chacun étant une mémoire de taille m (cf figure 3.1), chaque fonction de hachage hache les éléments de A vers l'un de ces étages et les bits aux positions $h_1(F_i), h_2(F_i), \ldots, h_k(F_i)$ sont mis à 1 (quelque soit la valeur précédente de ce bit, ainsi un bit peut être mis à 1 plusieurs fois).

La deuxième façon consiste à utiliser un seul étage avec des fonctions de hachage dans l'intervalle $[1..M]$ (cf figure 3.2). Ces deux façons sont asymptotiquement équivalentes pour des tailles de mémoires M élevées.

Si on veut déterminer si un élément B appartient à l' ensemble A, on regarde les différents bits aux positions $h_1(B), h_2(B), \ldots, h_k(B)$ si au moins l'un d'eux est resté égal à 0 alors B n'appartient certainement pas à A (donc pas de faux négatifs). Sinon, on suppose que $B \in A$ même si il y a une certaine probabilité de se tromper : c'est la probabilité de faux positif. Les paramètres m et k du filtre doivent être choisis de telle façon à minimiser cette probabilité de faux positif. D'une part, on peut remarquer que les erreurs sont dues aux collisions

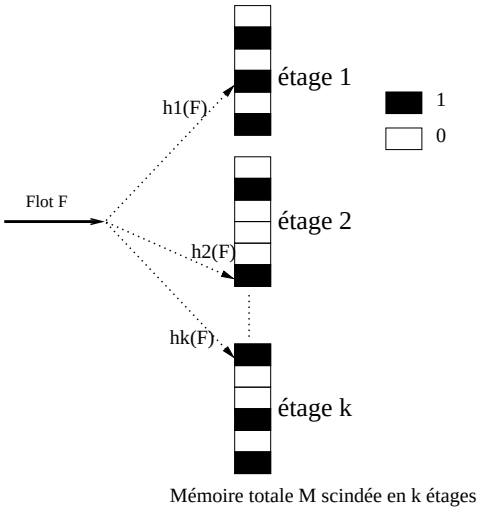


Fig. 3.1: k étages, une fonction de hachage par étage

puisque pour un élément qui n'appartient pas à A, on peut avoir tous ses bits positionnés à 1 seulement à cause des contributions des autres flots. Ainsi, si on augmente la valeur de k, on augmente la probabilité que sur au moins l'un des bits, le flot considéré ne collisionne avec aucun autre flot. D'autre part, plus on augmente la valeur de k, plus grande sera la mémoire car celle-ci est proportionnelle à k et en plus le temps d'insertion d'un élément dans le filtre augmente. Donc, il faut chercher un compromis entre la taille de la mémoire, le taux d'erreur et la rapidité de cet algorithme.

Remarquons qu'après insertion des n éléments de A dans le vecteur, comme pour chaque élément k bits sont mis à 1 de façon indépendante et aléatoire, alors la probabilité P qu'un certain bit reste encore à 0 est égale à $P = \left(1 - \frac{1}{m}\right)^n$. Donc la probabilité de faux positif F est égale à :

$$F = \mathbb{P}(\text{faux positif}) = (1-P)^k = \left(1 - \left(1 - \frac{1}{m}\right)^n\right)^k \sim \left(1 - e^{-\frac{n}{m}}\right)^k$$
$$\sim \left(1 - e^{-\frac{kn}{M}}\right)^k$$

Pour une mémoire donnée M, minimiser F est équivalent à minimiser $F^{\frac{n}{M}}$. Comme la fonction $(1-e^{-x})^x$ admet un minimum absolu en $x = \ln 2$, alors le nombre de fonctions de hachage k optimal (qui minimise la probabilité de faux positif sous mémoire constante) vérifie $k = \ln(2)\frac{M}{n}$ et donc :

$$P = \tfrac{1}{2} \text{ et } F = \left(\tfrac{1}{2}\right)^k$$

La structure simple des filtres de Bloom rend l'implémentation de certaines opérations facile à réaliser. Par exemple, si on suppose qu'on a deux filtres de Bloom qui représentent deux ensembles S_1 et S_2 qui en plus contiennent le même nombre de bits et utilisent les mêmes fonctions de hachage, un filtre de

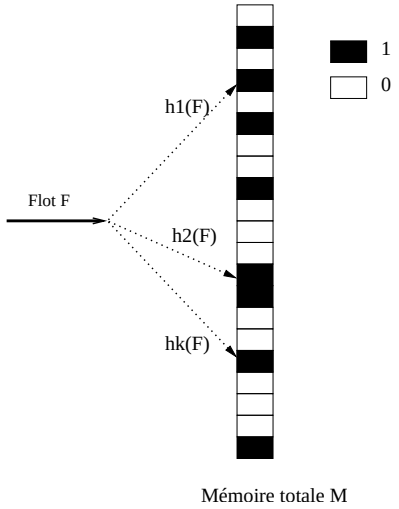


FIG. 3.2: 1 étage, k fonctions de hachage par étage

Bloom qui représente l'union de S_1 et S_2 peut être obtenu tout simplement en effectuant une opération OR des deux filtres initiaux.

Cette propriété est utilisée pour le routage dans les réseaux overlay. Supposons par exemple qu'on a un réseau sous forme d'arbre, chaque noeud envoie la liste de ses fichiers et celle de ses descendants sous forme de filtre de Bloom à son parent. Ainsi en effectuant une simple opération OR de l'ensemble de ces filtres, chaque noeud peut maintenir un filtre de Bloom qui représente les fichiers de ses descendants. En plus, chaque noeud garde la liste de ses fichiers dans un autre filtre de Bloom. Lorsqu'un noeud quelconque cherche un fichier donné, il regarde d'abord dans le filtre de ses propre fichiers puis sur le filtre de ses descendants. Si le fichier existe dans le filtre des descendants, il effectue alors la recherche seulement dans l'arbre des descendants. Dans le cas contraire, il envoie la requête à son parent. Rhea et Kubiatowicz [39] l'utilisent dans un algorithme de routage probabiliste dans les réseaux Peer-to-Peer, dans ce cas, chaque noeud du réseau Peer-to-Peer maintient un ensemble de filtres de Bloom. Il y a un filtre pour chaque distance d jusqu'à une distance maximale de telle sorte que le d-ième filtre représente les fichiers disponibles à une distance de d sauts.

Afin de détecter rapidement les boucles dans les réseaux unicast ou multicast, Whitaker et Wetherall [47] proposent de remplacer le mécanisme basé sur l'utilisation du TTL en ajoutant dans l'entête des paquets un petit filtre de Bloom qui représente l'ensemble des noeuds traversés, chaque noeud dispose d'un masque. Si, en faisant une opération OR entre ce masque et le filtre de Bloom, celui-ci reste inchangé, alors il y a très probablement une boucle.

Les filtres de Bloom trouvent aussi leur application dans le contexte du web pour une gestion distribuée du cache [17], par exemple pour savoir si un URL existe dans le cache d'un serveur ou non. Dans ce cas, au lieu que chaque proxy de cache diffuse périodiquement la liste de son contenu aux autres proxy, il

n'envoie qu'un filtre de Bloom représentant le contenu de son cache. Comme le contenu des caches peut changer entre les mises à jour périodiques, on pourrait avoir des faux positifs et des faux négatifs aussi, ce qui provoque le rallongement du temps de recherche d'un URL. Cet effet est négligeable si les contenus des caches ne changent pas fréquemment.

L'algorithme Stochastic Fair Blue utilise les filtres de Bloom pour la détection des flots agressifs ou non élastiques dans les files d'attente.

Il faut noter aussi que l'une des applications historiques des filtres de Bloom est la correction de l'orthographe des éditeurs de texte [33] en hachant tous les mots du dictionnaire dans un vecteur, ce qui permet donc de filtrer les mots qui n'appartiennent pas au dictionnaire. Un recueil assez détaillé des applications des filtres de Bloom est présenté dans [5].

3.3 Filtre de comptage

Pour un ensemble A qui varie au cours du temps avec des insertions et des effacements d'éléments, avec des filtres de Bloom habituels, comme les bits ne peuvent prendre que des valeurs égales à 1 ou 0, il n'est pas possible d'effacer un élément du filtre en réinitialisant toutes les cases de cet élément à 0 car on affecte tous les autres éléments qui tombent dans au moins une de ces cases et donc le filtre ne reflète plus l'ensemble A. Ainsi le seul moyen d'effacer un élément d'un filtre de Bloom est de reconstruire tout le filtre de Bloom de nouveau à chaque fois qu'on veut effacer un élément ce qui peut s'avérer irréaliste si ça arrive fréquemment. Les filtres de comptage [5] remédient à ce problème en remplaçant les bits par des petits compteurs et pour chaque élément $F_i \in A$, les compteurs aux positions $h_1(F_i),\ h_2(F_i), \ldots$ et $h_k(F_i)$ sont incrémentés de 1. Pour effacer l'élément F_i, il suffit de décrementer les compteurs des positions lui correspondant.
Comme on a besoin d'allouer de la mémoire pour les compteurs, il est important d'estimer la valeur maximale de ceux-ci. La moyenne du compteur maximum après insertion de n entrées avec k fonctions de hachage dans une mémoire de taille M est donnée par (cf [23]) :

$$\Gamma^{-1}(M)\left(1+\frac{kn/M}{\ln\Gamma^{-1}(M)}\right)+O\left(\frac{1}{\ln^2\Gamma^{-1}(M)}\right).$$

Comme,

$$\mathbb{P}(max(c)\geq i)\leq M\binom{nk}{M}\frac{1}{M^i}\leq m\left(\frac{enk}{iM}\right)^i$$

alors, lorsque $\frac{nk}{M}=\ln 2$ on a :

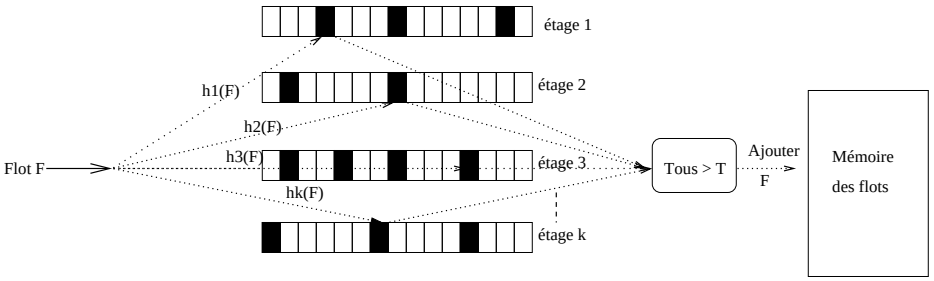
$$\mathbb{P}(max(c)\geq i)\leq M\left(\frac{e\ln 2}{i}\right)^i.$$

Ainsi, pour $i\geq 16$, cette probabilité devient très faible. Donc 4 bits par compteur sont suffisants.

3.4 Filtre de Varghese

3.4.1 Description de l'algorithme

Cet algorithme décrit dans [14] s'inspire des filtres de Bloom et les généralise au cas des ensembles multiset (ensembles contenant des éléments répétés). En effet, cette fois on ne s'intéresse pas seulement à la question de savoir si un élément appartient à un ensemble donné ou non mais aussi de savoir combien de fois il y est présent. Il peut être utilisé par exemple dans un routeur pour déterminer les grands flots (flots dépassant un certains seuil T de paquets) : Ce filtre est constitué de k étages, chaque étage est une mémoire de m cases. Chaque case contient un compteur qui est incrémenté par 1 à chaque fois qu'un flot est haché dedans. On dispose aussi de k fonctions de hachage à valeurs dans l'intervalle $[1..m]$, chacune est associée à un étage et sera utilisée pour hacher les flots dans l'étage correspondant. Au début, tous les compteurs sont initialisés à 0. Lorsque le routeur reçoit un flot F, il incrémente le compteur de la case $h_1(F)$ dans le premier étage, celui de la case $h_2(F)$ dans le second et ainsi de suite. A la fin de cette opération, si tous les compteurs mis à jour sont supérieurs à T (un seuil fixé à l'avance) alors le flot F est déclaré grand flot et sera stocké dans une mémoire dédiée.



Pour un flot F qui se présente au routeur, le minimum des compteurs des cases $h_1(F),\ h_2(F), \ldots$ et $h_k(F)$ est pris comme estimateur du nombre de fois que le routeur a vu le flot F jusqu'à présent. Cet estimateur surévalue le nombre d'occurrences du flot F puisque les compteurs des cases $h_1(F),\ h_2(F), \ldots$ et $h_k(F)$ sont incrémentés aussi par d'autres flots que F à cause des collisions. Il est évident que tous les flots de taille plus grande que T seront identifiés. Par contre, à cause des collisions, il se peut qu'un petit flot soit pris pour un grand, ce sont les faux positifs.

Si au lieu de s'intéresser au nombre de paquets par flot, on veut détecter les flots dont le volume transmis en octets dépasse T, alors l'algorithme peut être adapté facilement en incrémentant les valeurs des compteurs cette fois-ci par la taille des paquets et non pas par 1.

3.4.2 Incrémentation conservative

Dans le cas où on veut estimer le nombre d'octets transmis par un flot, dans [14], Estan et Varghese proposent un mécanisme qui améliore sensiblement l'estimation et par conséquent diminue le nombre de faux positifs en modifiant

la façon avec laquelle les compteurs sont incrémentés. Considérons un paquet du flot F qui se présente au routeur. Au lieu d'incrémenter tous les compteurs des cases $h_1(F),\ h_2(F), \ldots$ et $h_k(F)$ par la taille du paquet on procède comme suit : On prend tout d'abord le minimum min de ces compteurs, celui ci représente une estimation du nombre d'octets transmis par le flot F avant l'arrivée du dernier paquet. Ainsi, min incrémenté de la taille du dernier paquet $taille(paquet)$ est une estimation du nombre d'octets transmis par le flot F. Donc, on n'incrémente que les valeurs des compteurs qui sont inférieurs à $min + taille(paquet)$ par contre ceux qui sont supérieurs à $min + taille(paquet)$ restent inchangés. On pourrait être tenté de mettre les valeurs de ces derniers compteurs aussi à la valeur $min+taille(paquet)$, ceci n'affectera pas le flot F mais les autres flots qui tombent dans ces cases ce qui risque de dégrader notablement les estimations.

Dans le cas où on s'intéresse au nombre de paquets par flot et non pas au nombre total d'octets transmis, alors il ne faut incrémenter que les compteurs dont la valeur est minimale.

Le tableau 3.1, obtenu par analyse d'une trace INRIA d'une durée de 75 min, contenant approximativement 22 millions de paquets, 770 000 flots dont 49351 éléphants, montre que les filtres de Varghese ne sont pas adaptés aux trafics volumineux comme celui de l'Internet. En effet, les filtres se saturent rapidement et par conséquent, quasiment tous les flots arrivant après cette saturation sont considérés comme éléphants.

Durée (min)	Nb éléphants	Nb éléphants estimés	
		200 kbits	400 kbits
5	3860	3870	3861
10	7527	8080	7635
20	14777	22590	16094
30	21627	60127	26167
40	28647	119718	39698
50	34572	221523	98095
60	40757	370632	242466
70	46562	469569	342700
75	49351	496351	370187

Tab. 3.1: Filtres de Varghese non adaptés au trafic Internet

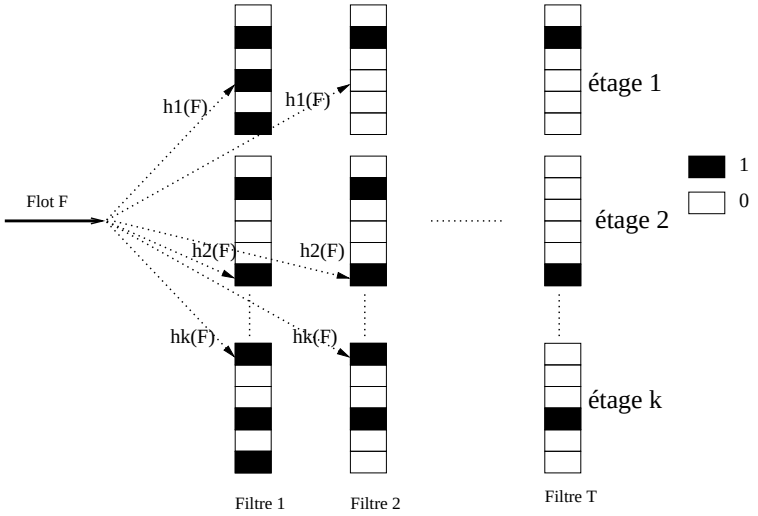
3.5 Filtres de Bloom parallèles

3.5.1 Présentation de l'algorithme

Ici, on va présenter un algorithme qui pallie à ces problèmes. On cherche à estimer le nombre de flots ayant plus que T paquets (si $T = 20$, il s'agit alors des flots éléphants) et aussi le nombre de flots ayant i paquets pour $i \leq T$). L'algorithme présenté ci-dessous en plus d'implémenter l'incrémentation conservative présente d'autres avantages qu'on verra par la suite.

On utilise T filtres de Bloom en parallèle, chaque filtre i est composé de

$k \geq 2$ étages dont chacun est une mémoire de m bits et on dispose de k fonctions de hachage h_j avec $1 \leq j \leq k$, chacune relative à un étage. Ces fonctions de hachage sont les mêmes dans les différents filtres. On suppose que toutes les fonctions de hachage sont indépendantes. Chaque filtre j est doté d'un compteur C_j qui compte le nombre de flots qui atteignent ce filtre. On verra par la suite que le compteur C_j est utilisé comme estimateur du nombre de flots de taille plus grande que j.



Par la suite FBP désignera un filtre de Bloom Parallèle. Voici un pseudo-code de l'algorithme :

FBP :

— INITIALEMENT.
Les bits des T filtres initialisés à 0, compteurs C_j des filtres nuls
— FLOT F ARRIVE.
Soit i le premier filtre où : $\prod_{j=1}^{k} h_j(F) = 0$
$\longrightarrow$ Remplir toutes les cases $h_j(F)$ du i-ième filtre par des 1
$\longrightarrow$ Incrémenter C_i

Initialement, tous les filtres et les différents compteurs sont initialisés à 0. A l'arrivée d'un flot F, on regarde le premier filtre où F n'existe pas, c'est le premier filtre où au moins un des bits correspondant aux positions : $h_1(F), h_2(F), \ldots, h_k(F)$ est resté égal à 0. A ce moment, on incrémente le compteur de ce filtre de 1 et on met toutes les cases $h_1(F), h_2(F), \ldots, h_k(F)$ de ce filtre à 1. Si le flot F existe dans les T filtres, alors on suppose que ce flot contient plus que T paquets et on décide de le stocker dans un filtre des éléphants ou explicitement dans une mémoire de flots.

L'idée de cet algorithme est que si les mémoires sont bien dimensionnées pour minimiser les collisions, alors le filtre i contiendra tous les flots sans exception

de taille plus grande que i en plus d'un nombre négligeable de flots de taille plus petite que i. Ainsi, on utilise le compteur C_i du filtre i comme estimateur du nombre de flots de taille plus grande que i.

On peut mentionner qu'à partir du FBP on peut retrouver facilement le filtre de comptage correspondant et vice versa. En effet, pour chaque case donnée, le nombre de filtres parallèles ayant un 1 à cette case représente la valeur du filtre de comptage de cette case. De plus, la façon avec laquelle on remplit le FBP correspond à celle de la mise à jour du filtre de comptage avec incrémentation conservative.

On peut remarquer que le remplissage des filtres est décroissant en fonction du rang des filtres, plus particulièrement, si une case dans le i-ème filtre est remplie, alors elle l'est aussi dans le filtre $i-1$ pour $2 \leq i \leq T$. Ainsi, le premier filtre sera le plus chargé, donc il faut dimensionner m de telle façon que son taux de remplissage (la proportion des cases remplies par rapport à la mémoire m) reste inférieure à un certain seuil. On minimise ainsi les collisions entre les flots et par conséquent le nombre de faux positifs.

Si on note n le nombre total de flots qui traversent le routeur, il faudra alors prendre une mémoire m de chaque étage qui vérifie :

$$m \sim \frac{n}{\ln 2}.$$

Les FBP offrent aussi l'avantage d'optimiser la taille mémoire, puisqu'on peut prendre des tailles mémoire décroissante en fonction du rang du filtre. En effet, dans le filtre i par exemple, seulement les flots de taille plus grande que i et une partie négligeable des flots de taille plus petite que i atteignent ce filtre.

3.5.2 Calcul du nombre de faux positifs

Comme on l'a mentionné précédemment, le i-ième filtre contient tous les flots de taille supérieure ou égale à i et une proportion de flots de taille plus petite que i. Ainsi, si on considère le T-ième filtre, alors celui-ci contient tous les éléphants plus une proportion de souris. Remarquons aussi que, parmi ces souris, celles qui collisionnent avec un éléphant dans les mêmes cases des différents étages n'ont aucune influence sur le comptage des éléphants. Par conséquent, seulement les souris qui atteignent le dernier filtre et qui ne collisionnent avec aucun éléphant dans les k étages sont des faux positifs. Afin de bien paramétrer la mémoire de cet algorithme, on va par la suite calculer le nombre de ces faux positifs.

On note N_f ce nombre de faux positifs. On suppose qu'on a E éléphants et S souris, toutes les souris sont de taille 1. Ainsi le nombre de souris par case suit une loi binomiale $\mathcal{B}(S, \frac{1}{m})$. Les différents filtres contiennent k étages. Considérons un k uplet $\underline{i} = (i_1, \cdots, i_k)$ correspondant à des numéros de cases prises au hasard dans chacun des étages de 1 à k.

On va calculer la probabilité que cet uplet de cases soit rempli dans le T-ième filtre alors qu'il n'y a aucun éléphant qui soit haché dans ces cases et qu'il y a au moins une souris qui tombe dans ces cases. Pour alléger les notations, on considère les événements :

- $S_{\underline{i}} = \{\text{Au moins une souris commune dans les cases } (i_1, \cdots, i_k)\}$

- $\overline{S_{\underline{i}}} = \{$Aucune souris commune dans les cases $(i_1, \cdots, i_k)\}$
- $E_{\underline{i}} = \{$Au moins un éléphant commun dans les cases $(i_1, \cdots, i_k)\}$
- $\overline{E_{\underline{i}}} = \{$Aucun éléphant commun dans les cases $(i_1, \cdots, i_k)\}$
- $D_{\underline{i}} = \{$Les cases $(i_1, \cdots, i_k)$ sont remplies dans le dernier filtre$\}$.

On note alors :

- $A_{\underline{i}} = S_{\underline{i}} \cap D_{\underline{i}} \cap \overline{E}_{\underline{i}}$
- $B_{\underline{i}} = S_{\underline{i}} \cap D_{\underline{i}}$
- $C_{\underline{i}} = S_{\underline{i}} \cap E_{\underline{i}}$

Au vu de la remarque précédente et qu'une même souris ne peut pas être dans deux k-uplets différents, alors on a l'expression suivante du nombre de faux positifs :

$$N_f = \sum_{\underline{i}} \mathbb{1}_{A_{\underline{i}}}$$

D'où :

$$\mathbb{E}(N_f) = m^k \mathbb{P}(A_{\underline{i}})$$

On sait aussi que, s'il y a au moins un éléphant dans les cases $(i_1, \cdots, i_k)$, alors ces cases seront remplies dans le dernier filtre donc

$$E_{\underline{i}} \subset D_{\underline{i}}.$$

D'où $C_{\underline{i}} = S_{\underline{i}} \cap D_{\underline{i}} \cap E_{\underline{i}}$. Par conséquent $B_{\underline{i}} = A_{\underline{i}} \cup C_{\underline{i}}$. Or les événements $A_{\underline{i}}$ et $C_{\underline{i}}$ sont disjoints, donc on obtient le résultat suivant :

$$\mathbb{P}(A_{\underline{i}}) = \mathbb{P}(B_{\underline{i}}) - \mathbb{P}(C_{\underline{i}}).$$

Remarque 1. $\mathbb{P}(C_{\underline{i}}) = \left(1 - \left(1 - \frac{1}{m^k}\right)^S\right)\left(1 - \left(1 - \frac{1}{m^k}\right)^E\right)$

Pour calculer $\mathbb{P}(B_{\underline{i}})$, on utilise les notations suivantes :

- ⋆ $\underline{n} = (n_1, \cdots, n_k)$ est le nombre de souris dans les cases de l'uplet $\underline{i}$
- ⋆ $\mathbb{P}(n)$ est la probabilité d'avoir n souris dans une case fixée du filtre.
- ⋆ $X_{\underline{n}}$ est la variable aléatoire qui désigne le nombre de souris hachées vers $(i_1, \cdots, i_k)$ sachant qu'il y a n_j souris dans la case i_j pour $1 \leq j \leq k$
- ⋆ p=$1 - \left(1 - \frac{1}{m}\right)^E$ est la probabilité qu'au moins un éléphant soit haché dans une case fixée du filtre.

Proposition 16.

$$\mathbb{P}(B_{\underline{i}}) = \sum_{j=1}^{k} \binom{k}{j} \sum_{\substack{n_1 < T, \cdots, n_j < T \\ n_{j+1} \geq T, \cdots, n_k \geq T}} \prod_{i=1}^{k} \mathbb{P}(n_i) \mathbb{P}(X_{\underline{n}} \geq 1) p^j. \tag{3.1}$$

Démonstration. On sépare l'événement $B_{\underline{i}}$ en k événements disjoints :

$$B_{\underline{i}} = \bigcup_{j=1}^{k} B_{\underline{i}}^{(j)}$$

où $B_{\underline{i}}^{(j)} = B_{\underline{i}} \cap \{$exactement j cases de l'uplet $\underline{i}$ contiennent plus que T souris$\}$.

On peut donc écrire :

$$\mathbb{P}(B_{\underline{i}}) = \sum_{j=1}^{k} \mathbb{P}\left(B_{\underline{i}}^{(j)}\right).$$

Il est clair que

$$\mathbb{P}\left(B_{\underline{i}}^{(j)}\right) = \binom{k}{j} \sum_{\substack{n_1<T,\cdots,n_j<T \\ n_{j+1}\geq T,\cdots,n_k\geq T}} \prod_{i=1}^{k} \mathbb{P}(n_i)\mathbb{P}(X_{\underline{n}} \geq 1)p^j$$

d'où le résultat. □

On veut maintenant trouver une limite de cette probabilité lorsque la taille du filtre m tend vers l'infini. Pour ce faire, on commence par montrer un résultat de convergence des termes $\mathbb{P}(X_{\underline{n}} \geq 1)$.

Lemme 1.

$$\mathbb{P}(X_{\underline{n}} \geq 1) \leq \frac{\prod_{i=1}^{k} n_i}{S^{k-1}} \tag{3.2}$$

Démonstration. La probabilité qu'une souris fixée soit commune entre les cases i_1, $i_2, \cdots$ et i_k sachant qu'il y a n_j souris dans la case i_j pour $1 \leq j \leq k$ est égale à

$$\prod_{i=1}^{k} \left(1 - \frac{\binom{S-1}{n_i}}{\binom{S}{n_i}}\right) = \frac{\prod_{i=1}^{k} n_i}{S^k}$$

Donc,

$$\mathbb{E}(X_{\underline{n}}) = S\frac{\prod_{i=1}^{k} n_i}{S^k} = \frac{\prod_{i=1}^{k} n_i}{S^{k-1}}.$$

D'autre part, on sait que

$$\mathbb{P}(X_{\underline{n}} \geq 1) \leq \mathbb{E}(X_{\underline{n}}).$$

le résultat s'en suit. □

Lemme 2. *Pour $k = 2$ on a l'inégalité suivante :*

$$1 - e^{\frac{-n_1 n_2}{S}} \leq \mathbb{P}(X_{\underline{n}} \geq 1) \tag{3.3}$$

Démonstration.

$$\begin{aligned}\mathbb{P}(X_{\underline{n}} \geq 1) &= 1 - \frac{\binom{S-n_2}{n_1}}{\binom{S}{n_1}} = 1 - \frac{(S-n_1)!(S-n_2)!}{(S-n_1-n_2)!S!} \\ &= 1 - \frac{(S-n_1)\cdots(S-n_1-n_2+1)}{S\cdots(S-n_2+1)} = 1 - \prod_{j=S-n_2+1}^{S} \left(1 - \frac{n_1}{j}\right)\end{aligned}$$

Ainsi

$$\mathbb{P}(X_{\underline{n}} \geq 1) \geq 1 - \left(1 - \frac{n_1}{S}\right)^{n_2} \geq 1 - e^{\frac{-n_1 n_2}{S}}.$$

□

Lemme 3. *Soit* $\underline{n}' = (n_1, \cdots, n_{k-1})$, $\forall k \geq 2$

$$\left(1 - e^{\frac{-n_k}{S}}\right) \mathbb{P}(X_{\underline{n}'} \geq 1) \leq \mathbb{P}(X_{\underline{n}} \geq 1) \tag{3.4}$$

Démonstration.

$$\begin{aligned}
\mathbb{P}(X_{\underline{n}}) &= \sum_{j \geq 1} \mathbb{P}(X_{\underline{n}'} = j) \mathbb{P}(X_{j,n_k} \geq 1) \\
&\geq \sum_{j \geq 1} \mathbb{P}(X_{\underline{n}'} = j) \mathbb{P}(X_{1,n_k} \geq 1) \\
&= \mathbb{P}(X_{\underline{n}'} \geq 1) \mathbb{P}(X_{1,n_k} \geq 1) \\
&\geq \mathbb{P}(X_{\underline{n}'} \geq 1) \left(1 - e^{\frac{-n_k}{S}}\right)
\end{aligned}$$

D'où le résultat. □

Proposition 17.

$$\forall k \geq 2, \ \forall n_1, \cdots, n_k \geq 0 \lim_{S \to +\infty} S^{k-1} \mathbb{P}(X_{n_1, \cdots, n_k} \geq 1) = \prod_{i=1}^{k} n_i \tag{3.5}$$

Démonstration. Par récurrence :
- Pour $k = 2$: Avec les lemmes 1 et 2 on a :

$$S\left(1 - e^{\frac{-n_1 n_2}{S}}\right) \leq S\mathbb{P}(X_{n_1,n_2} \geq 1) \leq n_1 n_2$$

Comme : $\lim_{S \to +\infty} S\left(1 - e^{\frac{-n_1 n_2}{S}}\right) = n_1 n_2$ Alors la propriété est vraie pour $k = 2$.
- Supposons que, $\forall n_1, \cdots, n_{k-1} \geq 0$ $\lim_{S \to +\infty} S^{k-2} \mathbb{P}(X_{\underline{n}'} \geq 1) = \prod_{i=1}^{k-1} n_i$. Avec les lemmes 1 et 3 on a :

$$S^{k-1}\left(1 - e^{\frac{-n_k}{S}}\right) \mathbb{P}(X_{\underline{n}'} \geq 1) \leq S^{k-1} \mathbb{P}(X_{\underline{n}} \geq 1) \leq \prod_{i=1}^{k} n_i$$

Comme par hypothèse on sait que $\lim_{S \to +\infty} S^{k-2} \mathbb{P}(X_{\underline{n}'} \geq 1) = \prod_{i=1}^{k-1} n_i$ et que $\lim_{S \to +\infty} S\left(1 - e^{\frac{-n_k}{S}}\right) = n_k$ alors
$\lim_{S \to +\infty} S^{k-1} \mathbb{P}(X_{\underline{n}} \geq 1) = \prod_{i=1}^{k} n_i$

Par conséquent la propriété est vérifiée aussi pour k. □

Proposition 18. *Pour $S = \alpha m$ et $E = \beta m$ où α et β sont des constantes strictement positives, on a*

$$\lim_{S\to+\infty} S^{k-1}\mathbb{P}(B_{i_1,\cdots,i_k}) = \alpha^k \left(1 - e^{-\beta}\mathbb{P}(\mathcal{P}(\alpha) < T-1)\right)^k \tag{3.6}$$

où $\mathcal{P}(\alpha)$ est une loi de poisson de paramètre α.

Démonstration.

$$S^{k-1}\mathbb{P}(B_{i_1,\cdots,i_k}) = \sum_{j=1}^{k} \binom{k}{j} \sum_{\substack{n_1<T,\cdots,n_j<T \\ n_{j+1}\geq T,\cdots,n_k\geq T}} \prod_{i=1}^{k} \mathbb{P}(n_i) S^{k-1}\mathbb{P}(X_{n_1,\cdots,n_k} \geq 1) p^j$$

Comme $\forall 1 \leq i \leq k$:

$$\begin{aligned}\mathbb{P}(n_i) &= \frac{S!}{(S-n_i)!n_i!}\left(\frac{1}{m}\right)^{n_i}\left(1-\frac{1}{m}\right)^{S-n_i} \leq \frac{S^{n_i}}{n_i!}\left(\frac{1}{m}\right)^{n_i} \\ &\leq \frac{\alpha^{n_i}}{n_i!}\end{aligned}$$

Donc,

$$\prod_{i=1}^{k} \mathbb{P}(n_i) S^{k-1}\mathbb{P}(X_{\underline{n}} \geq 1) \leq \prod_{i=1}^{k}\left(n_i \frac{\alpha^{n_i}}{n_i!}\right)$$

Puisque la série $\sum_{\substack{n_1<T,\cdots,n_j<T \\ n_{j+1}\geq T,\cdots,n_k\geq T}} \prod_{i=1}^{k}\left(n_i \frac{\alpha^{n_i}}{n_i!}e^{-\alpha}\right)$ converge, alors le théorème de convergence dominée nous donne :

$$\begin{aligned}&\lim_{S\to+\infty} S^{k-1}\mathbb{P}(B_{i_1,\cdots,i_k}) \\ &= \sum_{j=1}^{k}\binom{k}{j} \sum_{\substack{n_1<T,\cdots,n_j<T \\ n_{j+1}\geq T,\cdots,n_k\geq T}} \lim_{S\to+\infty}\left(\prod_{i=1}^{k}\mathbb{P}(n_i)S^{k-1}\mathbb{P}(X_{n_1,\cdots,n_k}\geq 1)p^j\right) \\ &= \sum_{j=1}^{k}\binom{k}{j} \sum_{\substack{n_1<T,\cdots,n_j<T \\ n_{j+1}\geq T,\cdots,n_k\geq T}} \prod_{i=1}^{k}\left(n_i\frac{\alpha^{n_i}}{n_i!}e^{-\alpha}\right)\left(1-e^{-\beta}\right)^j \\ &= \sum_{j=1}^{k}\binom{k}{j} \sum_{\substack{n_1<T-1,\cdots,n_j<T-1 \\ n_{j+1}\geq T-1,\cdots,n_k\geq T-1}} \prod_{i=1}^{k}\left(\frac{\alpha^{n_i}}{n_i!}\alpha e^{-\alpha}\right)\left(1-e^{-\beta}\right)^j \\ &= \sum_{j=1}^{k}\binom{k}{j}\left(\sum_{n<T-1}\frac{\alpha^n}{n!}\alpha e^{-\alpha}\right)^j\left(\sum_{n\geq T-1}\frac{\alpha^n}{n!}\alpha e^{-\alpha}\right)^{k-j}\left(1-e^{-\beta}\right)^j \\ &= \alpha^k\left(1-e^{-\beta}\mathbb{P}(\mathcal{P}(\alpha)<T-1)\right)^k.\end{aligned}$$

□

Ceci nous donne la proposition suivante

Proposition 19. *L'erreur relative du nombre d'éléphants estimés vérifie :*

$$\lim_{m\to\infty} \frac{\mathbb{E}(N_f)}{E} = \frac{\alpha}{\beta}\left(1 - e^{-\beta}\mathbb{P}(\mathcal{P}(\alpha) < T-1)\right)^k \tag{3.7}$$

où $\mathcal{P}(\alpha)$ est une loi de poisson de paramètre α.

Le terme $1 - e^{-\beta}\mathbb{P}(\mathcal{P}(\alpha) < T-1)$ peut être interprété comme la probabilité qu'une case contenant déjà une souris de taille 1 soit remplie dans les T filtres parallèles. On voit d'une part que l'erreur relative décroît au fur et à mesure que T augmente, ceci est tout à fait prédictible puisque les souris franchissent les filtres de plus en plus difficilement. D'autre part, on voit aussi que, plus β croît, plus le nombre de faux positifs augmente.

Cette interprétation nous permet de généraliser la proposition précédente au cas général où les souris ont des tailles quelconques entre 1 et $T-1$. En effet, si on suppose que pour tout i entre 1 et $T-1$, $S_i = \alpha_i m$, où S_i est le nombre de souris de taille égale à i, alors, le nombre de souris de taille égale à i dans une case donnée suit asymptotiquement (quand m est grand) une la loi de Poisson $\mathcal{P}(\alpha_i)$ de paramètre α_i. Par conséquent, une souris de taille i est considérée comme faux éléphant si, dans chacune de ses cases, il y a au moins un éléphant (terme $1 - e^{-\beta}$) ou bien la contribution des paquets des autres souris dépasse $T - i$ (le terme $\mathbb{P}(Z \geq T - i)$).

Proposition 20. *L'erreur relative du nombre d'éléphants estimés dans le cas général vérifie :*

$$\lim_{m\to\infty} \frac{\mathbb{E}(N_f)}{E} = \sum_{i=1}^{T-1} \frac{\alpha_i}{\beta}\left(1 - e^{-\beta}\mathbb{P}(Z < T-i)\right)^k. \tag{3.8}$$

où $Z = \sum_{i=1}^{T-1} i\mathcal{P}(\alpha_i)$ et les $\mathcal{P}(\alpha_i)$ sont des variables de loi de poisson de paramètre α_i indépendantes.

Le dimensionnement des filtres requiert une connaissance préalable du nombre de flots total n. Si, par exemple, on sous-estime n en choisissant une mémoire petite, alors tous les filtres seront remplis rapidement et les estimations seront très erronées. En plus, on voudrait que cet algorithme tourne sans interruption en temps réel dans un routeur. Dans les trois paragraphes suivants, on présentera quelques idées pour améliorer cet algorithme.

3.5.3 FBP avec effacement total

– **Durée fixée :** Une première adaptation de cet algorithme consiste à fixer un intervalle de temps Δ au bout duquel les premiers $T-1$ filtres sont effacés. On peut considérer, soit le temps réel, soit le temps paquet et, dans ce dernier cas, après un certain nombre de paquets, les premiers filtres sont effacés.
L'idée est que si le FBP n'est pas saturé pendant chaque intervalle Δ grand, alors on peut détecter la majorité des éléphants sur au moins l'un de ces intervalles.

FBP : Durée Δ fixée :

— Initialement.
Les bits des T filtres initialisés à 0, compteurs C_j des filtres nuls
— Flot F Arrive.
Si durée intervalle actuel plus grande que Δ
$\longrightarrow$ Effacer les $T-1$ premiers filtres
$\longrightarrow$ Commencer un nouvel intervalle
Soit i le premier filtre où : $\prod_{j=1}^{k} h_j(F) = 0$
$\longrightarrow$ Remplir toutes les cases $h_j(F)$ du i-ième filtre par des 1
$\longrightarrow$ Incrémenter C_i

Initialement tous les filtres et les compteurs sont initialisés à 0. On procède comme auparavant pendant un intervalle de longueur Δ. A la fin de chaque période de longueur Δ, on efface tous les filtres à l'exception du dernier (qui stocke les éléphants) et on recommence. Si, au lieu de s'intéresser seulement au nombre de grands flots (ceux ayant plus que T paquets), on veut aussi les stocker et avoir une estimation sur leur taille, alors on peut utiliser à la place du dernier filtre une mémoire des flots qui note explicitement ces grands flots et leurs compteurs qui estiment leurs tailles.

D'une part, il faut choisir un Δ suffisamment grand pour pouvoir supposer que tous les flots de taille supérieure à T émettront au moins T paquets au cours de l'un de ces intervalles. D'autre part il ne faut pas que Δ soit très grand car la taille des filtres est proportionnelle au nombre de flots arrivant pendant l'intervalle Δ.

Afin de montrer l'apport de l'effacement des filtres, on considère un cas simple : Toutes les souris sont de taille 1, aucun éléphant dans le flot et une seule fonction de hachage. L'effacement est effectué au bout de chaque intervalle de longueur Δ. On note t le premier instant d'un faux positif et $\tilde{t}$ celui du cas où il n'y a pas d'effacement. On a :

$$\begin{aligned}
\mathbb{P}(t > n\Delta) &= \mathbb{P}^{mn}(\mathcal{P}(\alpha) < T-1) \\
&\quad \text{et} \\
\mathbb{P}(\tilde{t} > n\Delta) &= \mathbb{P}^{m}(\mathcal{P}(n\alpha) < T-1) \\
&= \left(\sum_{i=0}^{T-2} \frac{(n\alpha)^i}{i!}\right)^m e^{-mn\alpha} \\
&= \frac{(n\alpha)^{m(T-2)}}{(T-2)!^m} e^{-mn\alpha} \left(1 + \frac{m(T-2)}{\alpha}\frac{1}{n} + O\left(\frac{1}{n^2}\right)\right)
\end{aligned}$$

où $\alpha = \frac{\Delta}{m}$. Ainsi,

$$\ln\left(\frac{\mathbb{P}(\tilde{t} > n\Delta)}{\mathbb{P}(t > n\Delta)}\right) \sim -m \ln\left(\sum_{i=0}^{T-2} \frac{\alpha^i}{i!}\right) n + m(T-2)\ln(n).$$

– **Remplissage fixé :** Sur certains routeurs très chargés, les filtres risquent de se remplir très rapidement avant même Δ. Donc, au lieu de fixer la longueur des intervalles au bout de laquelle les filtres seront réinitialisés, on efface les filtres dès que le premier filtre atteint un certain seuil de remplissage fixé à l'avance.

FBP : Remplissage fixé :

— Initialement.
Les bits des T filtres initialisés à 0, compteurs C_j des filtres nuls
— Flot F Arrive.
Soit i le premier filtre où : $\prod_{j=1}^{k} h_j(F) = 0$
$\longrightarrow$ Remplir les cases $h_j(F)$ du i-ième filtre par des 1
$\longrightarrow$ Incrémenter C_i
Si i=1
$\longrightarrow$ Mettre à jours le taux de remplissage
Si taux de remplissage plus grand que *seuil*
$\longrightarrow$ Effacer les $T-1$ premiers filtres

On prend comme critère le remplissage du premier filtre parce qu'il est le plus chargé. Donc, on doit maintenir un compteur supplémentaire qui compte le nombre de cases remplies dans le premier filtre et qui déclenche l'effacement des filtres dès que le taux de remplissage dépasse le seuil.

Résultats

On a retenu la version du FBP où l'effacement des filtres est contrôlé par le remplissage. On va par la suite montrer les résultats de simulations pour étudier l'effet de la taille des mémoire des filtres ainsi que le seuil de remplissage.

On dispose d'une trace de durée totale de 75 min qui contient approximativement 22 millions de paquets et 770 000 flots. Le flot le plus gros représente 177960 paquets. La figure 3.3 montre la distribution des tailles des flots de cette trace :

Les éléphants représentent approximativement 7% du total des flots. Le tableau 3.2 récapitule le nombre d'éléphants et de flots pour différentes durées entre 5 min et 75 min :

– **Effet de la taille des mémoires :** On fixe ici le taux de remplissage à 80% et on fait varier les tailles des filtres (cf figure 3.4).

On peut remarquer que, plus on augmente la mémoire, mieux on détecte les éléphants. En effet, pour des tailles de filtres grandes, on réduit le nombre de collisions entre les flots (flots qui partagent les mêmes cases) et par conséquent peu de souris arrivent à atteindre le dernier filtre des éléphants grâce aux contributions des autres flots. On voit aussi qu'à partir d'une mémoire de 200 kbits, on peut estimer et détecter les éléphants avec une erreur inférieure à 3%. De plus, comme cet algorithme repère seulement les éléphants qui émettent plus que 20 paquets sur au moins l'une des périodes (la fin d'une période correspond au dépassement du seuil de remplissage, à ce moment-là, on efface les filtres des souris et on garde seulement les éléphants détectés), alors il vaut mieux que ces

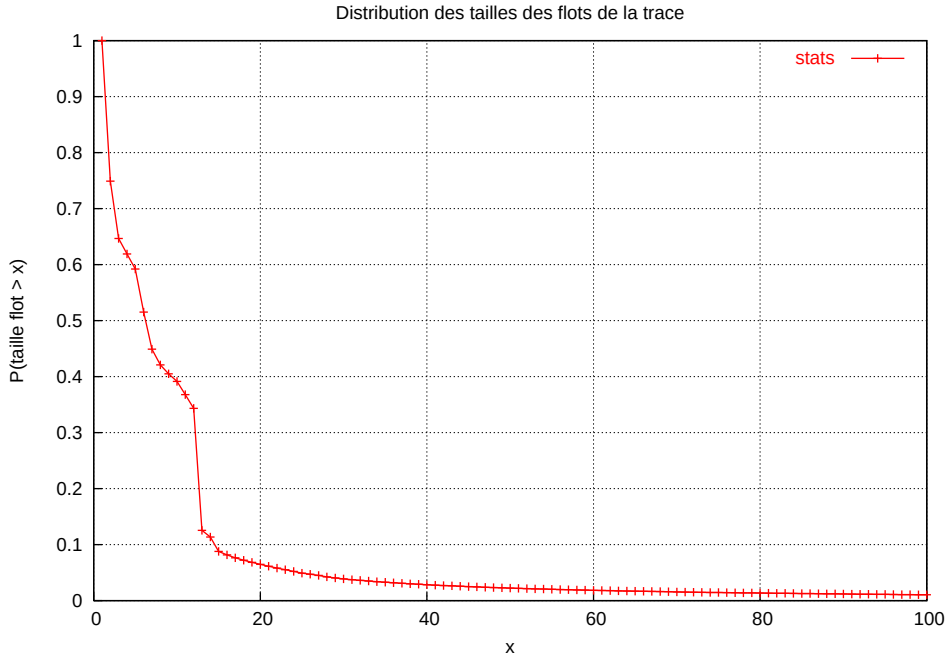


FIG. 3.3: Distribution des tailles des flots

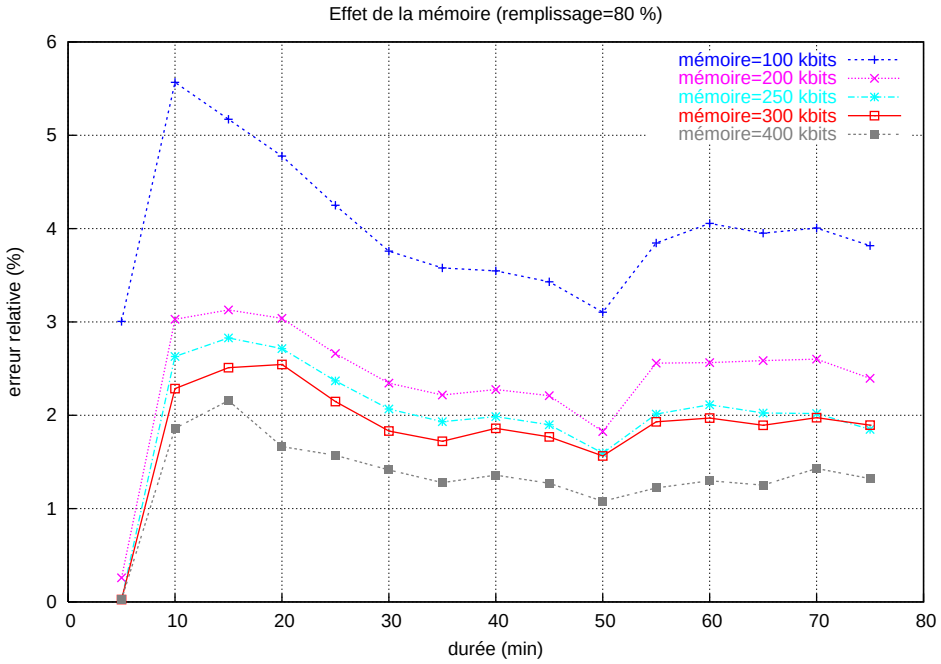


FIG. 3.4: Effet de la mémoire (taux de remplissage= 80%)

Durée (min)	Nb flots	Nb éléphants
5	29122	3860
10	167322	7527
20	261340	14777
30	332651	21627
40	398074	28647
50	500664	34572
60	650400	40757
70	751019	46562
75	771120	49351

TAB. 3.2: Le nombre d'éléphants et de flots

intervalles soient suffisamment grands pour récupérer le maximum d'éléphants. Ainsi, plus la mémoire est grande, plus longues seront ces périodes.

– **Effet du taux de remplissage :** On fixe ici la mémoire des filtres à 400 kbits et on fait varier le taux de remplissage (cf figure 3.5).

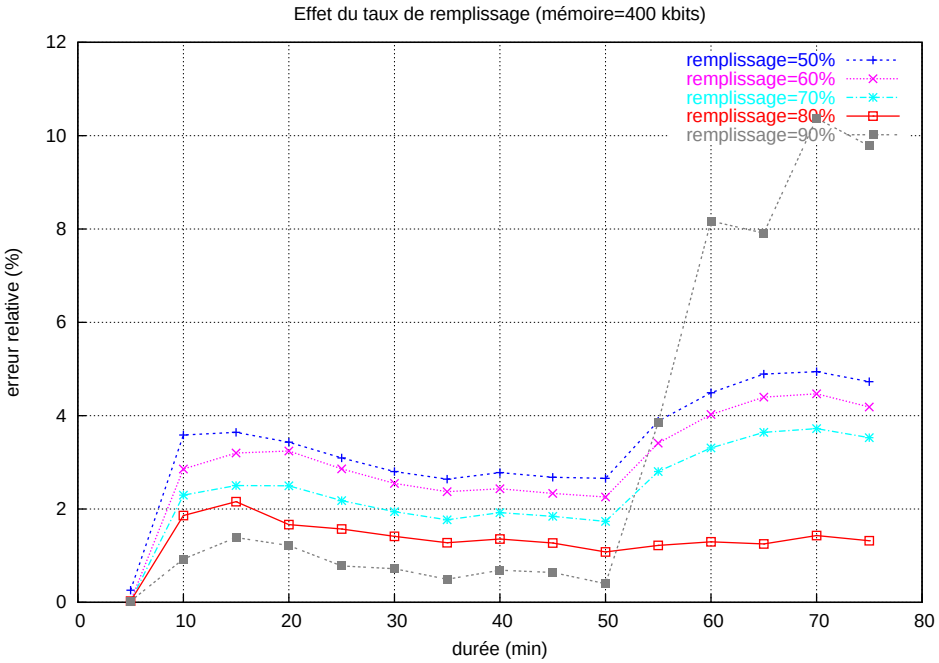


FIG. 3.5: Effet du taux de remplissage (Mémoire= 400 kbits)

Pour une mémoire fixée, plus le seuil de remplissage croît, plus la durée des périodes de mesure augmente. Par conséquent les éléphants ont suffisamment de temps pour émettre au moins 20 paquets sur l'un des intervalles. Néanmoins, lorsque le taux de remplissage est choisi très élevé, les durées sont plus longues mais les filtres restent plus longtemps dans un état surchargé. Ceci provoque beaucoup de collisions pendant ces périodes de surcharge et donc beaucoup de souris arrivent jusqu'au dernier filtre, ce qui provoque une dégradation notable

des performances (courbe avec taux de remplissage égal à 90%). Ainsi pour réduire l'erreur de cet algorithme, il vaut mieux opter pour un taux de remplissage pas très élevé (comme ça, on réduit l'effet des périodes de surcharge des filtres) et dimensionner les filtres avec une mémoire assez grande pour que la durée des périodes de mesures soit suffisante pour détecter tous les éléphants.

Signalons aussi que l'ajout de ce mécanisme adaptatif du taux de remplissage améliore nettement les performances de l'algorithme. Le tableau 3.3 compare le nombre d'éléphants trouvés en choisissant le taux de remplissage égal à 80% puis 100% (les filtres ne sont jamais effacés).

Durée (min)	Nb éléphants	200 kbits		400 kbits	
		taux remplissage		taux remplissage	
		80%	100%	80%	100%
5	3860	3870	3870	3861	3861
10	7527	7299	8080	7387	7635
20	14777	14328	22590	14531	16094
30	21627	21122	60127	21321	26167
40	28647	27997	119718	28258	39698
50	34572	33946	221523	34199	98095
60	40757	39717	370632	40228	242466
70	46562	45355	469569	45896	342700
75	49351	48169	496351	48699	370187

Tab. 3.3: Apport du mécanisme adaptatif

3.5.4 FBP avec effacement progressif

Lorsque le taux de remplissage dépasse le seuil de saturation, on voudrait idéalement effacer la contribution des souris et garder celle des éléphants. Or, l'effacement brutal des filtres revient à retrancher $T-1$ paquets de tous les flots. Ceci peut être amélioré en effaçant seulement quelques paquets des flots, on note PK ce nombre de paquets. Ainsi on supprime rapidement les contributions des souris et plus lentement celle des éléphants. Voici la description de l'algorithme :

FBP : Effacement progressif :

— Initialement.
Les bits des T filtres initialisés à 0, compteurs C_j des filtres nuls
— Flot F Arrive.
$\longrightarrow$ On met à jour le FBP et les compteurs C_j comme précédemment.
$\longrightarrow$ On met à jour le taux de remplissage
Si taux de remplissage plus grand que *seuil*
$\longrightarrow$ Effacer les PK premiers filtres
$\longrightarrow$ Les déplacer vers la fin du filtre parallèle
$\longrightarrow$ On met à jour le taux de remplissage

Pour mettre à jour le taux de remplissage, il suffit de maintenir le taux de remplissage des filtres au positions $1+iPK$ puisque le filtre à la position $PK+1$ devient premier après chaque effacement.

3.5.5 FBP avec effacement progressif et filtre virtuel

Effacer quelques paquets des flots peut fausser les statistiques, imaginons par exemple un éléphant qui a émis 19 paquets. Si on lui supprime quelques paquets et que ce même flot émet un 20ième paquet juste après l'effacement, on risque de ne pas le détecter. Ainsi, une autre amélioration de cet algorithme consiste à utiliser en plus du filtre réel un autre filtre virtuel. Le filtre virtuel sert à décider quand un flot doit être supprimé du filtre réel. Le filtre réel quand à lui stocke les statistiques réels des flots jusqu'à leur suppression. On dispose ainsi de deux FBP, l'un réel et l'autre virtuel.

FBP : Effacement progressif avec filtre virtuel :

— Initialement.
Les bits des T filtres initialisés à 0 dans les deux FBP, compteurs C_j du FBP réel mis à 0.
— Flot F Arrive.
⟶ On met à jour les deux FBP et les compteurs C_j comme précédemment.
⟶ On met à jour le taux de remplissage du filtre réel
Si taux de remplissage plus grand que *seuil*
⟶ Effacer les PK premiers filtres du FBP virtuel et les déplacer vers la fin
Pour chaque case : Si elle est égale à 0 dans le premier filtre du FBP virtuel
⟶ On réinitialise à 0 le contenu de cette case dans tous les filtres du FBP réel
⟶ On décrémente le taux de remplissage

3.5.6 Apport du filtre virtuel

L'ajout du filtre virtuel améliore sensiblement les performances. La figure 3.6 compare les résultats des deux algorithmes avec ou sans filtre virtuel. On a fixé la mémoire totale à 1Mbits (500 kbits pour chacun des deux filtres parallèles dans le cas où on utilise le filtre virtuel) et on a pris $PK = 1$.

3.5.7 Résultats

On va par la suite montrer les résultats de simulations de l'algorithme d'effacement progressif avec filtre virtuel pour étudier l'effet de la taille des mémoires des filtres ainsi que PK. On utilise la même trace que précédemment et on fixe le taux de remplissage à 50%.

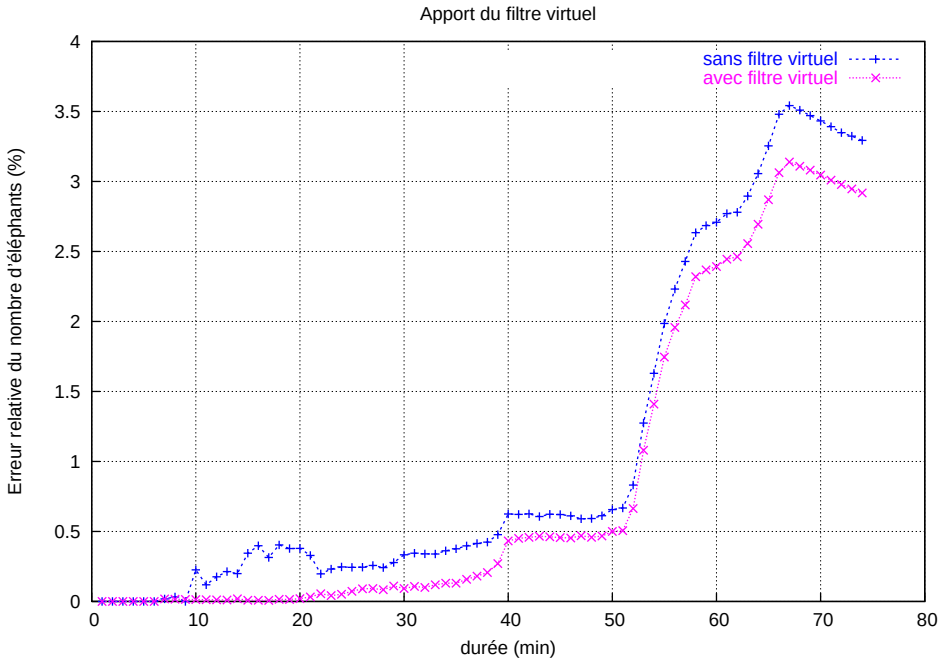


Fig. 3.6: Comparaison de l'erreur relative

– **Effet de la taille des mémoires :** On prend $PK = 1$ et on fait varier la taille des filtres (cf figure 3.7, 3.8, 3.9).

Comme prévu, on voit que plus la taille des filtres augmente, meilleur est l'estimation des statistiques des éléphants.

– **Effet de PK :** On prend $M = 1Mbits$ et on fait varier PK (cf figure 3.10, 3.11, 3.12).

Ces résultats confirment le fait que, plus petit est PK, meilleure est l'estimation. La courbe avec $PK = 20$ représente l'effacement total des filtres.

– **Simulation sur une trace ADSL :** On a simulé notre algorithme avec effacement progressif et filtre virtuel (une mémoire totale de 2 Mbits et $PK = 1$) sur une trace ADSL de France Telecom, d'une durée totale d'une heure et contenant 175048 éléphants. On l'a comparé à l'algorithme avec effacement total (cf figure 3.13). On voit bien que l'effacement progressif s'adapte plus. Dans les deux cas, l'erreur relative se stabilise au bout d'un certain temps.

3.6 Analyse

3.6.1 Effet du seuil de remplissage

Dans cette section, on veut étudier l'apport de l'effacement total des filtres, on s'intéresse plus particulièrement à la distribution du nombre de faux positifs. Pour simplifier, on considère qu'on a seulement une seule fonction de hachage (k=1). Le trafic est constitué de flots souris toutes de taille égale à 1. Ainsi, un faux positif résulte de l'accumulation de plus que T (T=20) souris dans une même case. Le filtre est effacé entièrement lorsqu'on atteint $r\%$ du remplissage

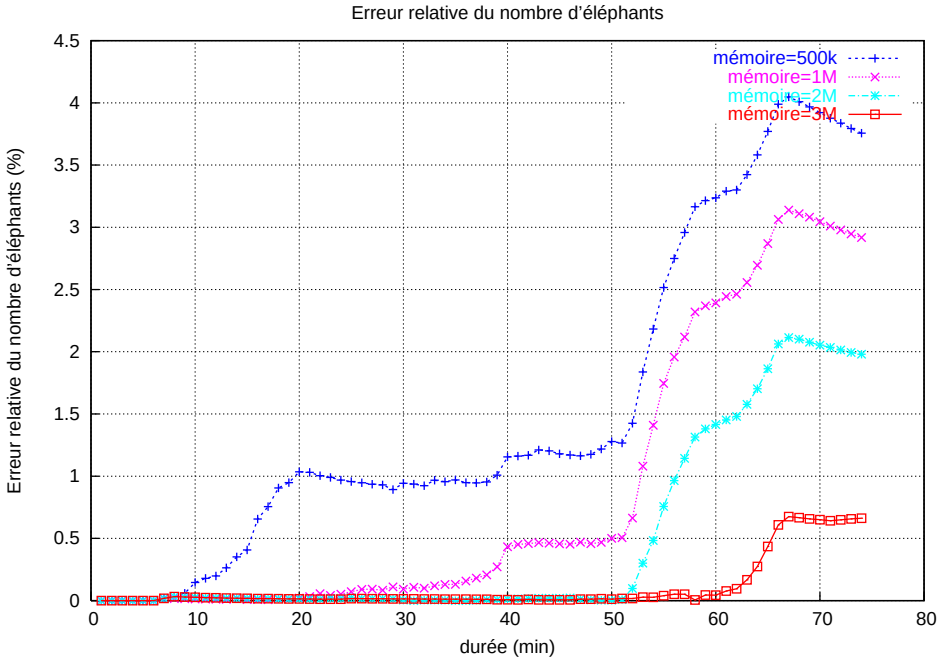


FIG. 3.7: Effet de la mémoire ($PK = 1$)

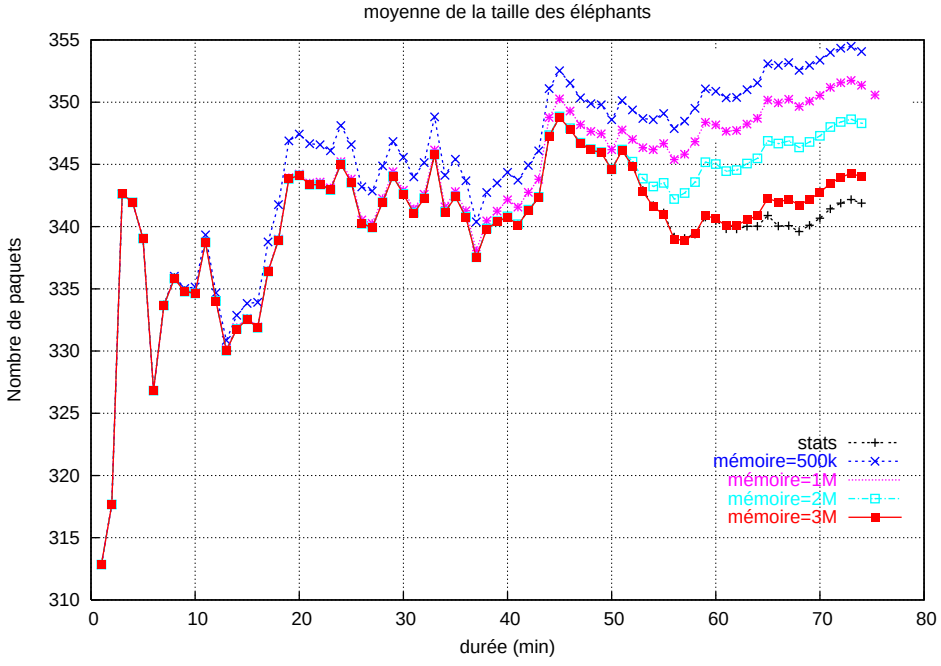


FIG. 3.8: Effet de la mémoire ($PK = 1$)

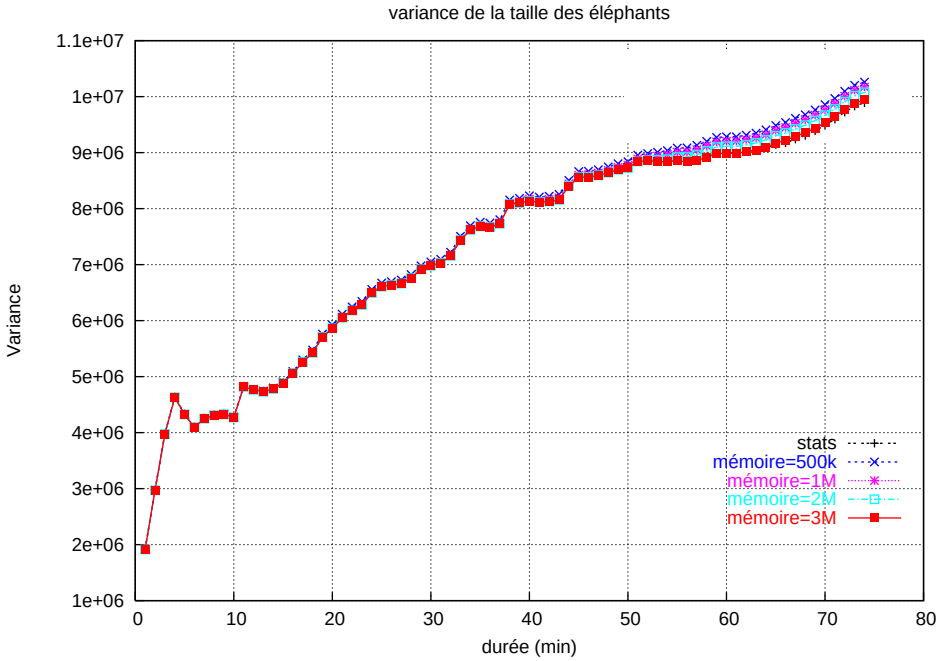


FIG. 3.9: Effet de la mémoire ($PK = 1$)

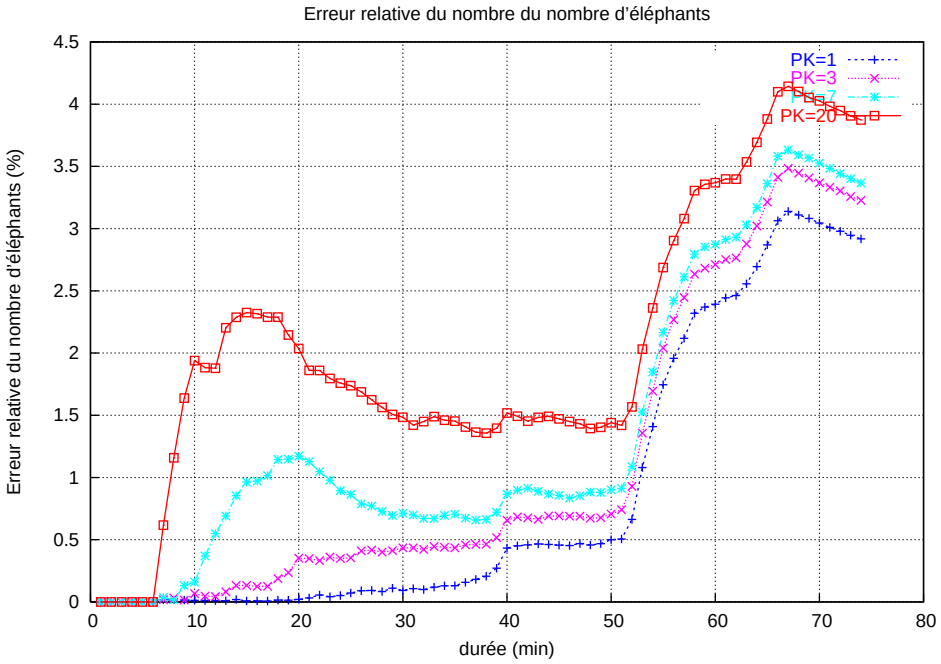


FIG. 3.10: Effet de la politique d'effacement ($M = 1Mbits$)

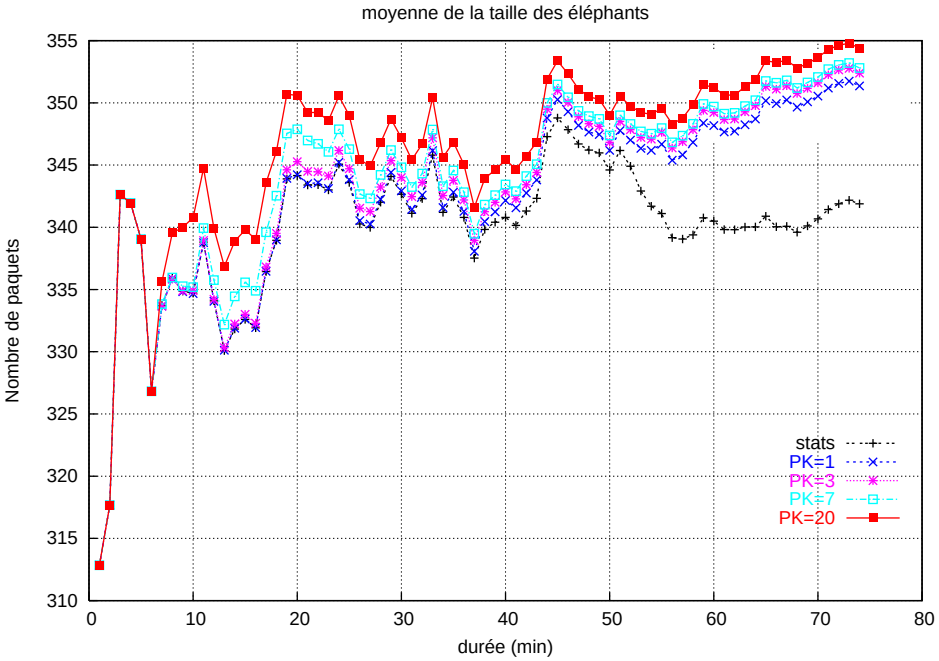


FIG. 3.11: Effet de la politique d'effacement ($M = 1Mbits$)

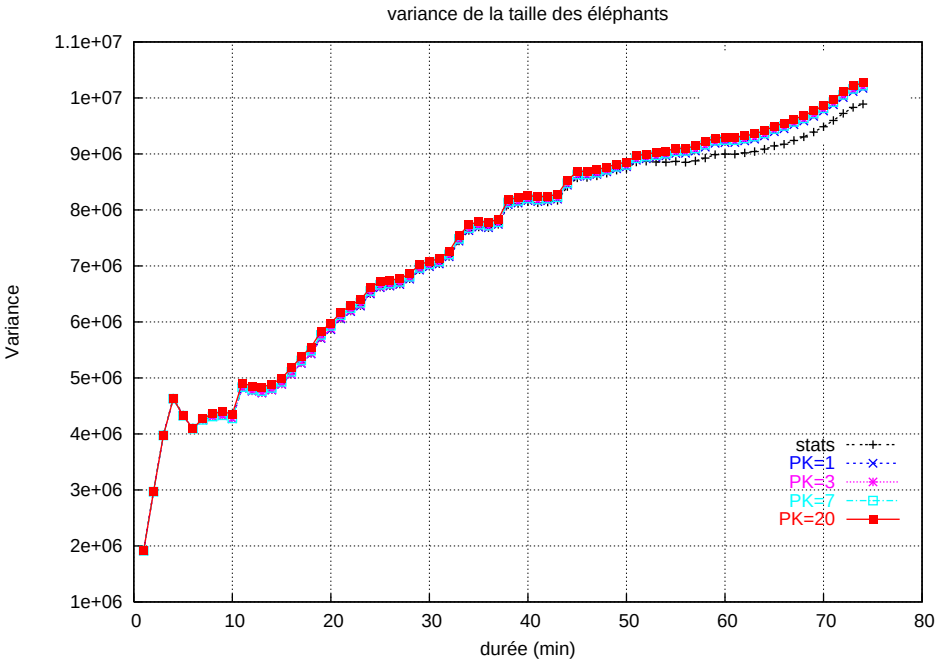


FIG. 3.12: Effet de la politique d'effacement ($M = 1Mbits$)

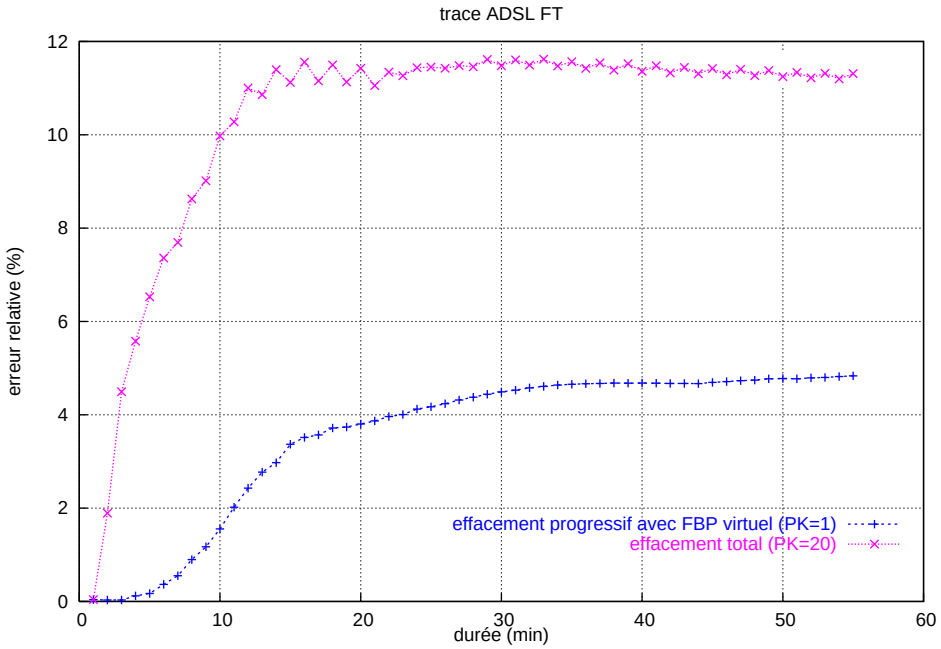


FIG. 3.13: Effet de la mémoire ($M = 1Mbits$)

du premier filtre parallèle. On note τ_r la variable aléatoire qui désigne l'instant d'atteinte du seuil de remplissage $r\%$. Notre problème est donc équivalent à jeter des boules dans m urnes et, dès que rm urnes deviennent remplies, on compte le nombre d'urnes qui contiennent plus que T boules. On sait qu'on peut écrire

$$\tau_r = rm + \sum_{i=1}^{rm} Z_i$$

où les variables Z_i sont des variables géométriques indépendantes de paramètre $p_i = 1 - i/m$. Une variable Z_i désigne le nombre de boules qu'il faut lancer pour remplir une nouvelle urne sachant que i urnes le sont déjà.
On note W_r le nombre d'urnes qui dépassent T à l'instant τ_r, c'est aussi le nombre de faux positifs pendant l'intervalle $[0,\ \tau_r]$.

La variable W_r peut s'écrire sous la forme :

$$W_r = I_1 + I_2 + \cdots + I_{rm},$$

où I_i est la variable aléatoire qui vaut 1 lorsque la i-ième case remplie du filtre dépasse T à l'instant τ_r et vaut 0 sinon. Dans notre cas, les variables (I_i) sont dépendantes, donc on ne peut pas appliquer la loi des grands nombres. La méthode de Stein-Chen donne une borne supérieure de la distance entre la distribution de W_r et celle d'une variable poissonnienne de paramètre $\mathbb{E}(W_r)$. Notons aussi que $I_i = 1$ est équivalent à mettre au moins $T - 1$ boules dans l'urne i sachant qu'on en a lancé $\tau_r - rm$ dans rm urnes (puisque chaque urne remplie contient au moins une boule).

Lemme 4 (Moyenne du nombre de faux positifs). *Si W_r désigne le nombre de faux positifs, m la mémoire total du filtre utilisé et $\phi = \frac{\tau_r - rm}{rm}$, alors $\mathbb{E}(W_r)$ vérifie la relation suivante*

$$\mathbb{E}(W_r) = rm\mathbb{E}(I) = rm\mathbb{E}(R_{T-1}(\phi)) - \mathbb{E}(\frac{\phi}{2}R''_{T-1}(\phi)) + O(1/m) \tag{3.9}$$

et

$$\lim_{m\to+\infty} \frac{1}{rm}\mathbb{E}(W_r) = \mathbb{E}(R_{T-1}(\phi))$$

avec

$$R_{T-1}(\phi) \overset{def.}{=} \sum_{j\geq T-1} \frac{\phi^j}{j!}e^{-\phi}.$$

Démonstration. On note W_r^a le nombre d'urnes qui contiennent exactement $a+1$ boules. Alors,

$$\mathbb{E}(W_r^a) = rm\mathbb{E}\left(\binom{\tau_r - rm}{a}\frac{1}{(rm)^a}\left(1-\frac{1}{rm}\right)^{\tau_r - rm - a}\right)$$

On obtient alors l'équivalence suivante :

$$\begin{aligned}\frac{1}{rm}\mathbb{E}(W_r^a) &\sim \mathbb{E}\left(\frac{1}{a!}e^{-\phi}\left(\phi^a - \frac{1}{2rm}a(a-1)\phi^{a-1} + \frac{1}{2\phi}(2a-\phi)\phi^a\right)\right) + O(1/m^2)\\ &= \mathbb{E}\left(\frac{\phi^a}{a!}e^{-\phi}\right) - \frac{1}{2rm}\mathbb{E}\left(\phi\frac{d^2}{d\phi^2}\left(\frac{\phi^a}{a!}e^{-\phi}\right)\right) + O(1/m^2).\end{aligned}$$

En additionnant ces termes pour $a \geq T-1$ et en vérifiant que le terme $O(1/m)$ reste valide, on obtient,

$$\mathbb{E}(W_r) = rm\mathbb{E}(R_{T-1}(\phi)) - \mathbb{E}(\frac{\phi}{2}R''_{T-1}(\phi)) + O(1/m).$$

□

Lemme 5 (Moment d'ordre 2 du nombre de faux positifs). *Si W_r désigne le nombre de faux positifs, m la mémoire totale du filtre utilisé et $\phi = \frac{\tau_r - rm}{rm}$, alors $\mathbb{E}(W_r^2)$ vérifie la relation suivante*

$$\mathbb{E}\left[W_r^2\right] = \mathbb{E}\left[W_r\right] + rm(rm-1)\mathbb{E}\left(R_{T-1}(\phi)^2\right) - (rm-1)\mathbb{E}\left(\frac{\phi}{2}\left(R_{T-1}(\phi)^2\right)''\right) + O(1/m^2) \tag{3.10}$$

avec

$$R_{T-1}(\phi) \overset{def.}{=} \sum_{j\geq T-1} \frac{\phi^j}{j!}e^{-\phi}$$

Démonstration. On note W_r^a le nombre d'urnes qui contiennent exactement $a+1$ boules. Soit $I_{i,a}$ l'indicatrice qui vaut 1 lorsque la i-ième urne remplie (parmi les rm urnes au total) contient exactement $a+1$ boules, alors pour $a \neq b$,

$$\mathbb{E}(W_a^r W_b^r) = \sum_{1\leq i\neq j\leq rm} \mathbb{E}(I_{i,a}I_{j,b}) = rm(rm-1)\mathbb{E}(I_{1,a}I_{2,b})$$

et $\mathbb{E}((W_r^a)^2) = \mathbb{E}(W_r^a) + rm(rm-1)\mathbb{E}(I_{1,a}I_{2,a})$.
D'autre part, on a pour a, $b \in \mathbb{N}$

$$\begin{aligned}\mathbb{E}(I_{1,a}I_{2,b}|\tau_r) &= \frac{(\tau_r - rm)!}{a!\,b!(\tau_r - rm - a - b)!}\frac{1}{(rm)^{a+b}}\left(1-\frac{2}{rm}\right)^{\tau_r - rm - a - b} \\ &= \frac{1}{a!\,b!}e^{-2\phi}\left(\phi^{a+b} - \frac{1}{2rm}(a+b)(a+b-1)\phi^{a+b-1} + \frac{2}{rm}\phi^{a+b}\,(a+b-\phi)\right) + O(1/m^2) \\ &= \frac{\phi^{a+b}}{a!\,b!}e^{-2\phi} - \frac{\phi}{2rm}\frac{d^2}{d\phi^2}\left(\frac{\phi^{a+b}}{a!\,b!}e^{-2\phi}\right) + O(1/m^2).\end{aligned}$$

En additionnant ces termes on obtient

$$\mathbb{E}\left[W_r^2\right] = \mathbb{E}\left[W_r\right] + rm(rm-1)\mathbb{E}\left(R_{T-1}(\phi)^2\right) - (rm-1)\mathbb{E}\left(\frac{\phi}{2}\left(R_{T-1}(\phi)^2\right)''\right) + O(1/m^2).$$

□

Lemme 6 (Convergence en loi de la variable ϕ)**.** *Lorsque m tend vers l'infini, la variable $\phi = \frac{\tau_r - rm}{rm}$ converge en loi vers une variable constante :*

$$\phi \overset{loi}{\to} \phi^* \overset{def.}{=} -\frac{\ln(1-r)}{r} - 1 \tag{3.11}$$

Démonstration. On sait que

$$\tau_r = rm + \sum_{i=1}^{rm} Z_i$$

où les variables Z_i sont des variables géométriques indépendantes de paramètre $p_i = 1 - i/m$. Ainsi :

$$\phi = \frac{\sum_{i=1}^{rm} Z_i}{rm}$$

Les variables Z_i sont indépendantes mais pas de même loi, donc, on ne peut pas appliquer directement la loi des grands nombres pour montrer la convergence presque sure.

$$\mathbb{E}(e^{-t\phi}) = \prod_{i=1}^{rm}\mathbb{E}(e^{-\frac{t}{rm}Z_i}) = \prod_{i=1}^{rm}\frac{1-i/m}{1-i/me^{-\frac{t}{rm}}} = \prod_{i=1}^{rm}\frac{m-i}{me^{\frac{t}{rm}}-i}e^t$$

D'autre part, on a :

$$\prod_{i=1}^{rm}(m-i) = (-1)^{rm+1}\frac{\Gamma(rm+1-m)}{\Gamma(1-m)}$$

et,

$$\prod_{i=1}^{rm}(me^{\frac{t}{rm}}-i) = (-1)^{rm+1}\frac{\Gamma(rm+1-me^{\frac{t}{rm}})}{\Gamma(1-me^{\frac{t}{rm}})}$$

Donc,

$$\mathbb{E}(e^{-t\phi}) = \frac{\Gamma(1-me^{\frac{t}{rm}})}{\Gamma(1-m)}\frac{\Gamma(rm+1-m)}{\Gamma(rm+1-me^{\frac{t}{rm}})}e^t$$
$$\sim (1-m)^{t/r}(rm+1-m)^{t/r}e^t \sim (1-r)^{t/r}e^t = e^{t\left(\frac{\ln(1-r)}{r}+1\right)}$$

D'où le résultat. □

Avant d'énoncer les résultats concernant la distribution du nombre de faux positifs, on va rappeler la définition de la distance en *variation totale*. Plus précisément, si Q_δ est une variable de Poisson de paramètre δ (c.à.d $\mathbb{P}(Q_\delta = n) = \delta^n \exp(-\delta)/n!$) et W est une variable aléatoire, alors la distance en variation totale entre les lois de Q_δ et W est donnée par

$$\|\mathbb{P}(W \in \cdot) - \mathbb{P}(Q_\delta \in \cdot)\|_{tv} = \sup_{A\subset\mathbb{N}} |\mathbb{P}(W \in A) - \mathbb{P}(Q_\delta \in A)|$$
$$= \frac{1}{2}\sum_{n\geq 0}\left|\mathbb{P}(W=n) - \frac{\delta^n}{n!}e^{-\delta}\right|$$

Notons qu'on a la propriété suivante : pour tout sous-ensemble A de $\mathbb{N}$,

$$|\mathbb{P}(W \in A) - \mathbb{P}(Q_\delta \in A)| \leq \|\mathbb{P}(W \in \cdot) - \mathbb{P}(Q_\delta \in \cdot)\|_{tv}$$

de telle sorte que cette distance donne une estimation de la déviation de la distribution de W par rapport à une distribution poissonnienne.

En vue de démontrer le théorème centrale limite, quand δ est pris égal à $\mathbb{E}(W)$, alors la distance en variation totale contrôle l'écart avec une distribution gaussienne (cf équation (6.2)).

Théorème 1 (Théorème centrale limite)**.** *Le nombre de faux positifs vérifie*

$$\lim_{m\to+\infty}\sup_{y\in\mathbb{R}}\left|\mathbb{P}\left(\frac{W_r - \mathbb{E}(W_r)}{\sqrt{\mathbb{E}(W_r)}} \leq y\right) - \int_{-\infty}^{y}\frac{e^{-u^2/2}}{\sqrt{2\pi}}\,du\right| \leq R_{T-1}(\phi^*) + \frac{1}{R_{T-1}(\phi^*)}\frac{(\phi^*)^{2T-3}e^{-2\phi^*}}{(T-2)!^2} \tag{3.12}$$

avec

$$\phi^* = -\frac{\ln(1-r)}{r} - 1$$

Si le taux de remplissage r est faible, on a l'équivalence suivante :

$$W_r \sim rm\frac{(\phi^*)^{T-1}}{(T-1)!} + \sqrt{\frac{rm}{(T-1)!}}(\phi^*)^{(T-1)/2}\,G$$

avec G variable aléatoire gaussienne centrée de variance 1.

Démonstration. L'équation (6.2) nous donne :

$$\sup_{y\in\mathbb{R}}\left|\mathbb{P}\left(\frac{W_r - \mathbb{E}(W_r)}{\sqrt{\mathbb{E}(W_r)}} \leq y\right) - \int_{-\infty}^{y}\frac{e^{-u^2/2}}{\sqrt{2\pi}}\,du\right| \leq \|\mathbb{P}(W_r \in \cdot) - \mathbb{P}(Q_{\mathbb{E}(W_r)} \in \cdot)\|_{tv}$$

D'autre part, on a (cf chapitre 6) :

$$\|\mathbb{P}(W_r \in \cdot) - \mathbb{P}(Q_{\mathbb{E}(W_r)} \in \cdot)\|_{tv} \leq 1 - \frac{\mathrm{Var}(W_r)}{\mathbb{E}(W_r)}$$

Or,

$$\begin{aligned}
1-&\frac{\mathrm{Var}(W_r)}{\mathbb{E}\,(W_r)} = \\
&- (rm-1)\frac{\mathbb{E}\left(R_{T-1}^2(\phi)\right)}{\mathbb{E}\,(R_{T-1}(\phi))} + rm\mathbb{E}\,(R_{T-1}(\phi)) + \frac{\mathbb{E}\left(\phi\left(R_{T-1}^2(\phi)\right)''\right)}{2\mathbb{E}\,(R_{T-1}(\phi))} \\
&- \mathbb{E}\left(\frac{\phi}{2}\left(R_{T-1}(\phi)\right)''\right) + o(1) \\
&\leq \mathbb{E}\,(R_{T-1}(\phi)) + \frac{\mathbb{E}\left(\phi\left(R_{T-1}^2(\phi)\right)''\right)}{2\mathbb{E}\,(R_{T-1}(\phi))} - \mathbb{E}\left(\phi\left(R_{T-1}(\phi)\right)''\right) + o(1).
\end{aligned}$$

Comme les différentes fonctions $R_{T-1}(\phi)$, $\phi\left(R_{T-1}^2(\phi)\right)''$ et $\phi\left(R_{T-1}(\phi)\right)''$ sont continues et que la variable ϕ converge en loi vers ϕ^* (lemme (6)) alors on a

$$\begin{aligned}
&\lim_{m\to+\infty} \mathbb{E}\,(R_{T-1}(\phi)) = R_{T-1}(\phi^*) \\
&\lim_{m\to+\infty} \mathbb{E}(\phi\left(R_{T-1}^2(\phi)\right)'') = \phi^*\left(R_{T-1}^2(\phi^*)\right)'' \\
&\lim_{m\to+\infty} \mathbb{E}(\phi\left(R_{T-1}(\phi)\right)'') = \phi^*\left(R_{T-1}(\phi^*)\right)''
\end{aligned}$$

$$\begin{aligned}
\lim_{m\to+\infty} 1 - \frac{\mathrm{Var}(W_r)}{\mathbb{E}\,(W_r)} &\leq R_{T-1}(\phi^*) + \phi^*\frac{\left(R'_{T-1}(\phi^*)\right)^2}{R_{T-1}(\phi^*)} \\
&= R_{T-1}(\phi^*) + \frac{1}{R_{T-1}(\phi^*)}\frac{(\phi^*)^{2T-3}e^{-2\phi^*}}{(T-2)!^2}.
\end{aligned}$$

Quand le remplissage est faible, cette borne tend vers 0 et donc on a

$$\frac{W_r - \mathbb{E}(W_r)}{\sqrt{\mathbb{E}(W_r)}} \overset{loi}{\to} G.$$

Comme $\mathbb{E}(W_r) \sim rm\frac{(\phi^*)^{T-1}}{(T-1)!}$, alors on a le résultat. □

Maintenant, on voudrait savoir s'il vaut mieux effacer les filtres lorsqu'on atteint le seuil de remplissage $r\%$ (durée élastique) ou bien les changer au bout d'intervalles de durées fixes (cette durée est fixée égale à la moyenne des durées du premier cas). On a bien l'intuition que dans le premier cas, on s'adapte mieux puisque les filtres ne dépassent jamais le seuil de remplissage alors que dans le deuxième cas ceux-ci peuvent rester longtemps saturés et induire des erreurs conséquentes.

Corollaire 1. *On note N_1 la moyenne du nombre de faux positifs du premier cas et N_2 celui du second cas. Pour un taux de remplissage faible,*

$$\frac{N_1}{N_2} \sim \frac{T}{2^{T-1}}.$$

Pour un taux de remplissage élevé (proche de 1),

$$\frac{N_1}{N_2} \sim \frac{1}{r}.$$

Démonstration. On pose $\phi^* = -\frac{\ln(1-r)}{r} - 1$. D'une part, le lemme 4 nous donne

$$N_1 \sim rm \sum_{j=T-1}^{+\infty} \frac{(\phi^*)^j}{j!} e^{-\phi^*}$$

d'autre part, la proposition 19 nous permet d'écrire que

$$N_2 \sim m \sum_{j=T}^{+\infty} \frac{(r(\phi^*+1))^j}{j!} e^{r(-\phi^*+1)}.$$

Quand r est faible, $\phi^* \sim r/2$. Donc il est clair que, $N_1 \sim rm\frac{(r/2)^{T-1}}{(T-1)!}$ et $N_2 \sim m\frac{r^T}{T!}$
Pour r élevé : $N_1 \sim rm$ et $N_2 \sim m$ d'où le résultat. □

Ce corollaire corrobore l'intuition que l'effacement déclenché par l'atteinte du taux de saturation est meilleur que lorsque celui-ci est effectué au bout d'intervalles fixes. De plus, plus T croît, plus cette différence devient flagrante.

3.7 Les éléphants marquent leurs cases

Dans ce paragraphe, on va présenter un autre algorithme qui détecte et compte les éléphants. L'idée de base de cet algorithme est que les éléphants émettent plusieurs paquets au cours de labs de temps très courts donc, dès que le routeur reçoit un certain nombre de paquets consécutifs d'un même flot, il le marque comme flot candidat (qui pourrait devenir éléphant) et le suit au cours du temps. Ainsi, on élimine une grande partie des souris qui ne réussissent pas à émettre un minimum de paquets consécutifs.

On utilise une mémoire de m cases, on dispose aussi de deux fonctions de hachage h_1 et h_2, la première sert à hacher les flots vers l'ensemble $\{1..m\}$, tandis que la deuxième est utilisée pour identifier les flots. On fixe aussi un seuil R ($R \leq 20$). Chaque case enregistre une liste des identifiants (générés avec la fonction h_2) des flots candidats et du flot en cours qui pourrait devenir candidat. On maintient un compteur pour chaque flot (candidats et flots en cours) dans la case. Le compteur des candidats sert à déterminer si ceux-ci deviennent des éléphants (quand le compteur dépasse 20). Le flot en cours lui aussi dispose d'un compteur qui compte le nombre de paquets consécutifs reçus de ce flot et sert à

déterminer si ce flot devient candidat (quand ce compteur dépasse R). Quand un nouveau flot arrive, on le hache avec la fonction h_1 pour savoir la case à laquelle il appartient. S'il est déjà candidat on incrémente son compteur de 1 puis, si ce compteur devient supérieur à 20, alors ce flot est déclaré éléphant. Si ce flot est le flot en cours de la case en question, alors on incrémente son compteur de 1, si le compteur devient supérieur à R, alors ce flot devient candidat. Sinon, si ce flot n'est ni candidat ni flot en cours de la case considérée, alors il remplace le flot en cours et son compteur est mis à 1. La mise à jour du contenu des cases s'effectue de la façon suivante :

ÉLÉPHANTS MARQUENT LEURS CASES :

— INITIALEMENT.
Toutes les cases et compteurs à 0
— FLOT F ARRIVE.
Si F est candidat dans la case $h_1(F)$
$\longrightarrow$ Incrémenter $compteur(h_2(F))$
Si $compteur(h_2(F)) = 20$:
$\longrightarrow$ F est éléphant.

Si F est le flot en cours de la case $h_1(F)$:
$\longrightarrow$ Incrémenter $compteur(h_2(F))$
Si $compteur(h_2(F)) = R$:
$\longrightarrow$ Ajouter F à la liste des candidats de la case $h_1(F)$.
Sinon :
$\longrightarrow$ F remplace le flot en cours de la case $h_1(F)$
$\longrightarrow$ $compteur(h_2(F)) = 1$

– **Effet de la mémoire.** Ici, on fixe le seuil à 10 et on fait varier la taille de la mémoire (cf figure 3.14). On voit que l'erreur baisse au fur et à mesure qu'on augmente la taille des mémoires car, dans ce cas, on sépare les flots en plus de groupes et donc seulement les flots d'un même groupe (qui tombent dans la même case) se perturbent mutuellement. Ainsi, on récupère plus de flots candidats (cf tableau 3.4). On observe aussi que, même pour une taille de mémoire faible comme 5 kbits, l'erreur ne dépasse pas 4% puisqu'une case peut contenir un nombre illimité de candidats.

– **Effet du seuil.** Ici, on fixe la tailles des filtres à 20 kbits et on fait varier le seuil R (cf figure 3.15). Plus on augmente la valeur de R, moins on consomme de mémoire et meilleure sera l'estimation et la détection des éléphants. Par exemple, il est évident que pour $R = 1$, tout flot est candidat, donc il faudra beaucoup de mémoire pour stocker tous les flots avec leurs compteurs mais l'estimation sera parfaite, et plus on augmente la valeur de R, moins il y aura de flots candidats puisqu'une grande partie des souris et des petits éléphants peut être filtrée et par conséquent, on aura besoin de moins de mémoire et l'estimation devient de plus en plus imprécise.

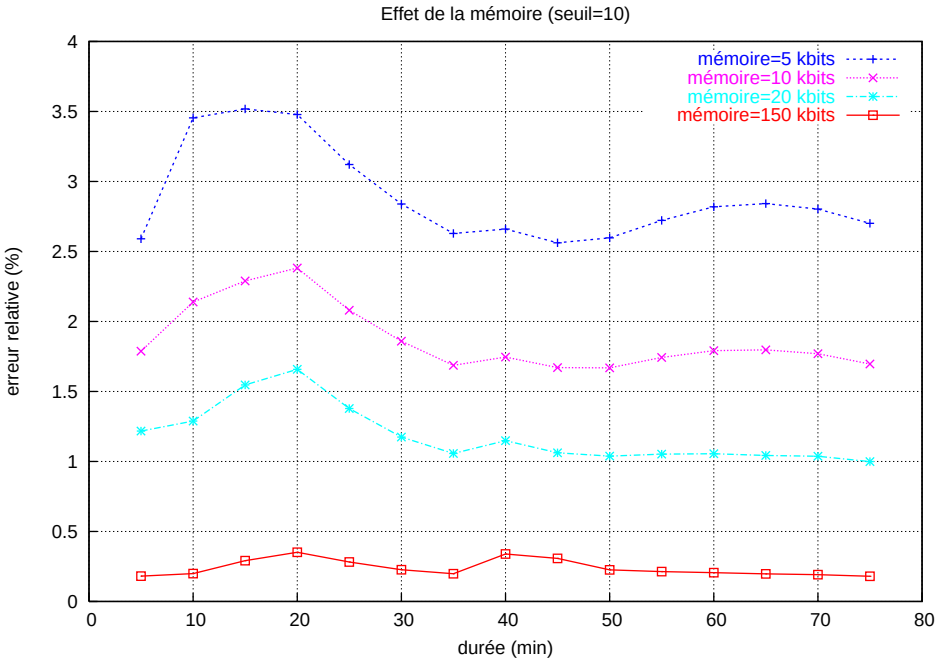


Fig. 3.14: Effet de la mémoire (seuil = 10)

3.8 Conclusion

Dans ce chapitre, on a proposé deux algorithmes pour le comptage et la détection des flots éléphants. Le premier s'appuie sur les les filtres de Bloom parallèles, on a proposé un mécanisme adaptatif pour vider régulièrement les filtres afin de maintenir le taux de remplissage inférieur à un certain seuil fixé en avance. Cet algorithme est facile et rapide d'implémentation. Le second algorithme utilise une table de hachage et crée des entrées seulement pour les flots dont les paquets arrivent consécutivement un certain nombre de fois.

Dans les deux cas, l'erreur relative du nombre d'éléphants est maintenue très petite et stabilisée sur de longues durées.

Durée (min)	Nb éléphants	Mémoire=5kbits		Mémoire=150kbits	
		candidats	éléphants	candidats	éléphants
5	3860	7701	3760	8050	3853
10	7527	14783	7267	15896	7512
15	11313	21872	10915	23515	11280
20	14777	28720	14263	30774	14725
25	18073	35323	17509	37811	18022
30	21627	42793	21013	45595	21578
35	25263	50042	24599	53111	25213
40	28647	56304	27885	59592	28550
45	31541	62068	30733	65641	31444
50	34572	104403	33674	111408	34494
55	37919	160963	36887	172473	37838
60	40757	210516	39608	225856	40673
65	43629	256343	42389	275152	43543
70	46562	275010	45257	295057	46473
75	49351	280602	48018	300887	49262

Tab. 3.4: Le nombre de candidats en fonction de la mémoire

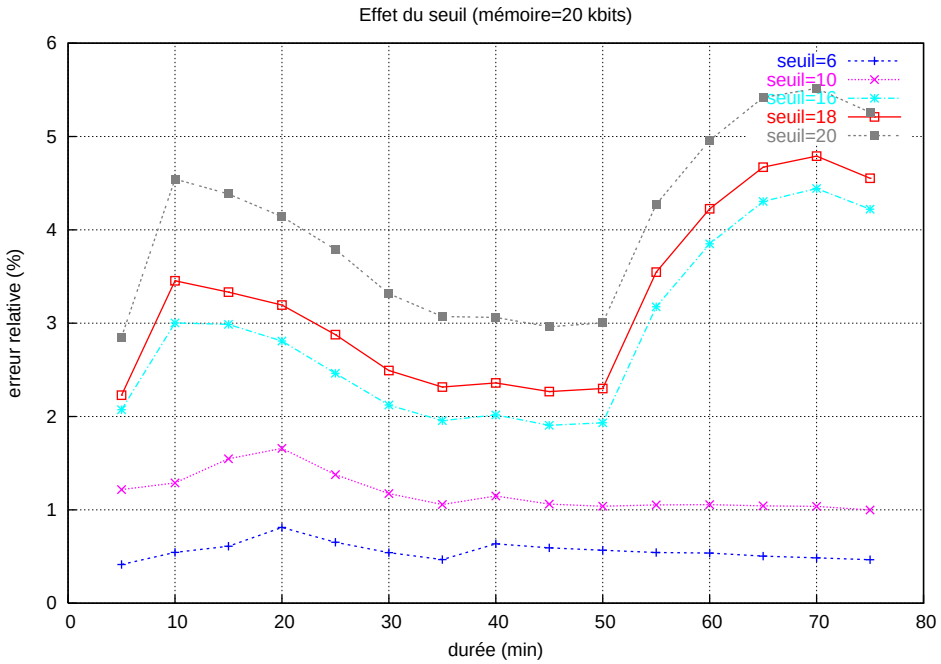


Fig. 3.15: Effet du seuil (mémoire = 20 kbits)

Chapitre 4

Détection des grands flots par échantillonnage

4.1 Échantillonnage

Dans la pratique, on ne dispose pas de la totalité de la trace des paquets traversant un routeur mais seulement d'une partie. En effet, les paquets sont échantillonnés par Netflow dans les routeurs. L'échantillonnage peut s'effectuer de deux façons :

- Échantillonnage périodique : Dans ce cas, on ne prend qu'un paquet tous les N paquets reçus, où N est typiquement de l'ordre de 100 à 1000. L'objectif est de réduire le volume des données et le temps de traitement.
- Échantillonnage probabiliste : Ici, chaque paquet est pris avec probabilité p où p est de l'ordre de 10^{-3} à 10^{-2}.

Ces deux façons d'échantillonner donnent presque les mêmes résultats lorsque le trafic traversant le routeur est très élevé et que le facteur d'échantillonnage reste modéré (cf figure 4.1 et 4.2 où on a représenté l'erreur relative de l'échantillonnage probabiliste par rapport à l'échantillonnage périodique).

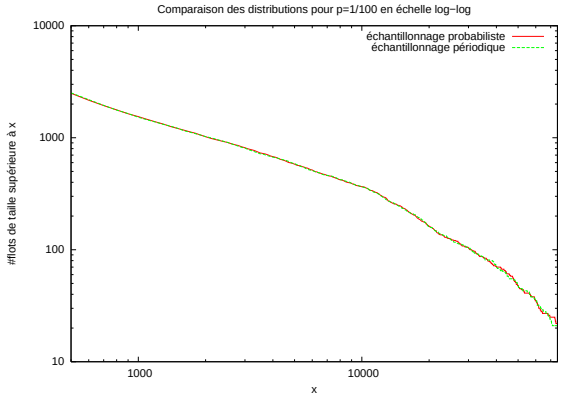


Fig. 4.1: Échantillonnage périodique vs probabiliste, p=1/100

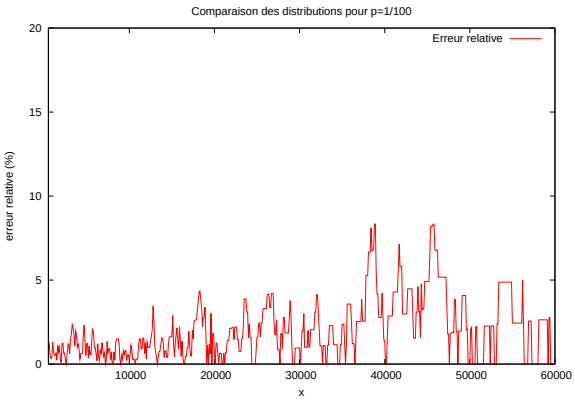


FIG. 4.2: Erreur relative, p=1/100

Dans la suite on considère seulement l'échantillonnage probabiliste. Ainsi pour un flot initial de taille i, on va en échantillonner un certain nombre de paquets qui suit une loi binomiale de paramètres $(i,\ p)$. L'objectif dans la suite est d'estimer les statistiques des flots initiaux à partir de ceux des flots après échantillonnage. Il est clair que, plus le taux d'échantillonnage est élevé, plus on peut inférer les flots originaux. On considère qu'on a un nombre N de flots très élevé, chacun d'entre eux a un nombre de paquets qui suit une loi H. On note N_l le nombre de flots (parmi les N au total) de taille supérieure ou égale à l. De la même façon, on définit $\hat{N}_l$ comme étant le nombre de flots de taille supérieure ou égale à l après échantillonnage. Tout d'abord il est évident que :

$$\mathbb{E}(N_l) = N\mathbb{P}(H \geq l)$$

et

$$\mathbb{E}(\hat{N}_l) = N\mathbb{P}(\hat{H} \geq l)$$

où $\hat{H} = \sum_{i=1}^{H} B_i$ et les variables $(B_i)_{i\in\mathbb{N}}$ sont des variables de Bernoulli indépendantes de même paramètre p et indépendantes de H. La variable aléatoire $\hat{H}$ décrit la distribution des tailles des flots après échantillonnage. On pourrait avoir l'intuition que les flots de taille supérieure à l, seront ramenés à des tailles supérieures à lp après échantillonnage et donc d'estimer N_l à partir de $\hat{N}_{lp}$. Ceci, n'est pas toujours vrai. En effet, il suffit de considérer que la loi H suit une distribution géométrique (cas irréaliste en pratique) pour s'en convaincre (voir section 4.1.1). Heureusement, on sait qu'il a été montré empiriquement que H suit plutôt une distribution de type Pareto ou Weibull auquel cas on va montrer que $N_l \sim_{l\to+\infty} \hat{N}_{lp}$ (voir sections 4.1.2 et 4.1.3).
Si on note S_n la marche aléatoire suivante : $S_n = \sum_{i=1}^{n} B_i$ et T_l le temps d'atteinte de l par cette marche aléatoire :

$$T_l = \min_n \{S_n \geq l\}$$

Alors, on peut remarquer qu'on peut écrire :

$$P\left(\hat{H} \geq l\right) = P\left(S_H \geq T_l\right) = P\left(H \geq T_l\right)$$

On s'intéressera dans la suite aux distributions H vérifiant la propriété :

$$P(\hat{H} \geq l) \sim_{l \to +\infty} P(H \geq l/p) \tag{4.1}$$

Pour ces distributions on a : $N_l \sim_{l \to +\infty} \hat{N}_{lp}$

Lemme 7. *Si la fonction de répartition de la distribution H est concave sur l'intervalle $[l, +\infty[$, alors*

$$P(\hat{H} \geq l) \geq P\left(H \geq l/p\right).$$

Démonstration. Comme $T_l \geq l$, en utilisant l'inégalité de Jensen, on a $P\left(H \geq T_l\right) \geq P\left(H \geq \mathbb{E}(T_l)\right)$, or $\mathbb{E}(T_l) = l/p$ d'après la proposition 21, d'où le résultat. □

Quasiment toutes les distributions usuelles (Pareto, Weibull, géométrique...) ont des fonctions de répartition concaves pour l grand. Ce lemme équivaut à

$$\mathbb{E}(\hat{N}_{lp}) \geq \mathbb{E}(N_l).$$

On établit tout d'abord quelques propriétés de la variable T_l.

Proposition 21 (Transformée de Laplace de T_l)**.**

$$\forall t \geq 0,\ \mathbb{E}\left(e^{-tT_l}\right) = e^{-l \ln\left(1+\frac{e^t-1}{p}\right)}$$

Démonstration. On note $a \wedge b = \min(a, b)$. On a alors pour tout $u \geq 0$, $(M_n)_n = \left(\frac{e^{uS_n}}{\mathbb{E}(e^{uB})^n}\right)_n$ est une martingale.
Comme, $\forall n \in \mathbb{N},\ \mathbb{E}(M_n) = 1$ et que T_l est un temps d'arrêt pour cette martingale alors on a : $\mathbb{E}(M_{T_l \wedge n}) = 1$. D'autre part, $S_{T_l \wedge n} \leq l$ et $0 \leq T_l \wedge n$ ce qui nous permet de majorer les variables $M_{T_l \wedge n}$

$$\forall n \in \mathbb{N},\ M_{T_l \wedge n} \leq e^{ul}.$$

Comme $\lim_{n \to +\infty} M_{T_l \wedge n} = M_{T_l}$, alors le théorème de convergence dominée nous donne

$$\mathbb{E}(M_{T_l}) = \lim_{n \to +\infty} \mathbb{E}(M_{T_l \wedge n}) = 1$$

soit, pour tout $u \geq 0$

$$\mathbb{E}\left(\mathbb{E}(e^{uB})^{-T_l}\right) = e^{-lu}.$$

En effectuant le changement de variable : $e^t = \mathbb{E}(e^{uB})$, on obtient le résultat. □

En particulier, on obtient les deux premiers moments de la variable T_l

$$\mathbb{E}(T_l) = \frac{l}{p} \text{ et } \mathrm{Var}(T_l) = \frac{l(1-p)}{p^2}.$$

Proposition 22 (Théorème centrale limite pour T_l)**.**

$$\frac{T_l - l/p}{\sqrt{l(1-p)/p^2}} \xrightarrow[l \to +\infty]{loi} \mathcal{N}(0, 1) \tag{4.2}$$

Démonstration. On a :

$$\mathbb{E}\left(e^{-t\frac{T_l-l/p}{\sqrt{l}}}\right) = e^{l\left(\frac{t}{p\sqrt{l}}-\ln\left(1+\frac{e^{\frac{t}{\sqrt{l}}}-1}{p}\right)\right)} = e^{\frac{t^2}{2}\frac{1-p}{p^2}} + o(1)$$

D'où le résultat. □

Dans la suite on montrera tout d'abord un exemple où la propriété 4.1 n'est pas vérifiée, puis on l'établira pour les deux distributions les plus utilisées pour décrire la distribution des tailles des flots : Pareto et Weibull.

4.1.1 Cas de la distribution géométrique

Supposons que la taille des flots H vérifie

$$\mathbb{P}(H \geq n) = e^{an},\ a < 0$$

Ceci équivaut à ce que H suit une loi géométrique de paramètre p où $p = e^a,\ a < 0$.

Ainsi

$$P\left(\hat{H} \geq l\right) = P\left(H \geq T_l\right) = \mathbb{E}(e^{aT_l})$$

Or, d'après la proposition 21 on a

$$\mathbb{E}(e^{aT_l}) = e^{-l\ln\left(1+\frac{e^{-a}-1}{p}\right)}.$$

Or, la fonction $(e^x-1)/x$ est strictement croissante sur $\mathbb{R}$, d'où $(e^{-a}-1)/(-a) < (e^{-a/p}-1)/(-a/p)$ et après quelques manipulations, on obtient

$$\forall a < 0,\ 0 < p < 1,\ -\ln\left(1+\frac{e^{-a}-1}{p}\right) > \frac{a}{p}.$$

Donc,

$$\mathbb{E}(e^{aT_l}) = e^{-l\ln\left(1+\frac{e^{-a}-1}{p}\right)} \not\sim e^{al/p}.$$

Donc, la propriété 4.1 n'est pas vérifiée pour une distribution géométrique et on retrouve bien le résultat du lemme 7 dans ce cas.

4.1.2 Cas de la distribution Pareto

Si H a une distribution Pareto, on a

$$\mathbb{P}(H \geq n) \sim_{n\to+\infty} \alpha n^s \text{ avec } \alpha > 0 \text{ et } s < 0 \text{ des constantes.}$$

Donc, la densité de probabilité décroît de façon polynomiale. Ainsi

$$P\left(\hat{H} \geq l\right) = P\left(H \geq T_l\right) \sim_{l\to+\infty} \alpha\mathbb{E}(T_l^s).$$

D'après la proposition 22, on a

$$T_l = \frac{l}{p} + \frac{\sqrt{l(1-p)}}{p}X \text{ où } X \xrightarrow[l\to+\infty]{loi} \mathcal{N}(0,1).$$

Ainsi

$$\mathbb{E}(T_l^s) = \left(\frac{l}{p}\right)^s \mathbb{E}\left(1 + \sqrt{\frac{1-p}{l}}X\right)^s.$$

Or la variable $\sqrt{\frac{1-p}{l}}X \xrightarrow[l\to+\infty]{loi} 0$ et que la fonction $f(x) = (1+x)^s$ est continue, alors

$$\lim_{l\to+\infty} \mathbb{E}\left(1 + \sqrt{\frac{1-p}{l}}X\right)^s = 1,$$

ce qui donne, quand l tend vers $+\infty$,

$$P\left(\hat{H} \geq l\right) \sim \alpha \left(\frac{l}{p}\right)^s \sim P\left(H \geq l/p\right).$$

Par conséquent, la distribution Pareto vérifie bien la propriété 4.1.

4.1.3 Cas de la distribution Weibull

Si H suit une distribution Weibull, on a

$$\mathbb{P}(H \geq n) \sim e^{-n^\beta} \text{avec } 0 < \beta < 1.$$

Remarque 2. *Normalement on a $\mathbb{P}(H \geq n) \sim e^{-\alpha n^\beta}$ avec $\alpha > 0$ et $0 < \beta < 1$. En réécrivant cette équivalence sous la forme $\mathbb{P}(H \geq n/\alpha^{1/\beta}) \sim e^{-n^\beta}$ et quitte à considérer la quantité $\mathbb{P}(H \geq n/\alpha^{1/\beta})$, on peut prendre $\alpha = 1$.*

Ainsi

$$P\left(\hat{H} \geq l\right) = P\left(H \geq T_l\right) \sim_{l\to+\infty} \mathbb{E}(e^{-T_l^\beta}).$$

Or,

$$\begin{aligned}\mathbb{E}(e^{-T_l^\beta}) &= \sum_{n=l}^{\infty} e^{-n^\beta} \binom{n-1}{l-1} p^l (1-p)^{n-l} \\ &= \left(\frac{p}{1-p}\right)^l \frac{1}{(l-1)!} F_l\end{aligned}$$

où

$$F_l = \sum_{n=l}^{\infty} e^{-n^\beta} (1-p)^n \frac{(n-1)!}{(n-l)!} = \sum_{n=l}^{\infty} e^{-n^\beta + \ln(1-p)n + \sum_{k=1}^{l-1} \ln(n-k)}.$$

Dans la suite, on définit la fonction f de la façon suivante :

$$\forall x > l,\ f(x) = -x^\beta + \ln(1-p)x + \sum_{k=1}^{l-1} \ln(x-k).$$

On réécrit alors F_l sous la forme

$$F_l = \sum_{n=l}^{\infty} e^{f(n)}.$$

La méthode de Laplace consiste à trouver le maximum x_l de la fonction f ($f'(x_l) = 0,\ f''(x_l) \leq 0$) et de remarquer que, lorsque l tend vers l'infini, la somme ci-dessus se concentre autour des valeurs de n proches de x_l

$$F_l = \sum_{n=l}^{\infty} e^{f(n)} \sim \sum_{n=l}^{\infty} e^{f(x_l)+\frac{f''(x_l)}{2}(n-x_l)^2} \sim e^{f(x_l)} \sum_{n=l}^{\infty} e^{\frac{f''(x_l)}{2}(n-x_l)^2}. \tag{4.3}$$

Notons donc x_l une solution de l'équation

$$f'(x_l) = 0. \tag{4.4}$$

Proposition 23 (Comportement asymptotique de x_l)**.** *Si x_l est solution de l'équation* (4.4)*, alors*

$$x_l = \frac{l}{p} + Al^{\beta} + B + O(l^{\beta-1})$$

et

$$f''(x_l) \sim -\frac{p^2}{(1-p)l}$$

où $A = -\beta\frac{1-p}{p^{\beta+1}}$ et $B = \frac{1}{2}$.

Démonstration. On a

$$f'(x_l) = -\beta x_l^{\beta-1} + \ln(1-p) + \sum_{k=1}^{l-1} \frac{1}{x_l - k} = 0. \tag{4.5}$$

Or,

$$\sum_{k=1}^{l-1} \frac{1}{x_l - k} = \Psi(x_l) - \Psi(x_l - l + 1) \sim \ln\left(\frac{x_l}{x_l - l + 1}\right)$$

où la fonction Ψ est définie de la façon suivante : $\Psi(x) = (\ln(\Gamma(x)))' = \Gamma'(x)/\Gamma(x)$. Comme $\beta < 1$ implique que $\lim_{l\to+\infty} x_l^{\beta-1} = 0$, alors on a $\lim_{l\to+\infty} \ln\left(\frac{x_l}{x_l-l+1}\right) = -\ln(1-p)$. Ceci nous donne que $\lim_{l\to+\infty} \frac{x_l}{x_l-l+1} = \frac{1}{1-p}$ et par conséquent

$$x_l \sim \frac{l}{p}.$$

Notons maintenant $x_l = \frac{l}{p} + \epsilon_1$ avec $\epsilon_1 = o(l)$. On sait que $\Psi(x) = -\gamma + \ln(x) - \frac{1}{2x} + o(x^{-1})$ où γ est la constante d'Euler et que $x_l \sim l/p$. Donc, on peut écrire

$$\Psi(x_l) = -\gamma + \ln(x_l) - \frac{p}{2l} + o(l^{-1})$$
$$\Psi(x_l - l + 1) = -\gamma + \ln(x_l - l + 1) - \frac{p}{2(1-p)l} + o(l^{-1}).$$

D'où

$$\Psi(x_l) - \Psi(x_l - l + 1) = \ln\left(\frac{x_l}{x_l - l + 1}\right) + \frac{p^2}{2(1-p)l} + o(l^{-1}).$$

Si on pose $x_l = \frac{l}{p} + \epsilon_1$ avec $\epsilon_1 = o(l)$, alors on a

$$\ln\left(\frac{l/p+\epsilon_1}{l(1/p-1)+\epsilon_1+1}\right) + \ln(1-p) - \beta x_l^{\beta-1} = -\frac{p^2}{2(1-p)l} + o(l^{-1}),$$

ce qui donne,

$$\ln\left(1+\frac{\epsilon_1}{l/p}\right) - \ln\left(1+\frac{\epsilon_1+1}{l(1/p-1)}\right) - \beta x_l^{\beta-1} = -\frac{p^2}{2(1-p)l} + o(l^{-1}).$$

Par conséquent, on a au premier ordre l'équivalence suivante

$$\frac{\epsilon_1}{l/p} - \frac{\epsilon_1}{l(1/p-1)} \sim \beta x_l^{\beta-1} \sim \beta\left(\frac{l}{p}\right)^{\beta-1}.$$

ou encore

$$\epsilon_1 \sim -\beta\frac{1-p}{p^{\beta+1}} l^\beta.$$

En réinjectant cet équivalent dans l'équation (4.5), on obtient de la même façon le développement de x_l au troisième ordre. Ceci conclut la démonstration de la première partie de la proposition. D'autre part, on a

$$f''(x_l) = -\beta(\beta-1)x_l^{\beta-2} - \sum_{k=1}^{l-1}\frac{1}{(x_l-k)^2}.$$

Or,

$$\sum_{k=1}^{l-1}\frac{1}{(x_l-k)^2} = \frac{1}{x_l}\left(\frac{1}{x_l}\sum_{k=1}^{l-1}\frac{1}{(1-\frac{k}{x_l})^2}\right) \sim \frac{1}{x_l}\int_0^p \frac{dx}{(1-x)^2} \sim -\frac{p^2}{(1-p)l}.$$

Comme $\beta - 2 < -1$, alors $f''(x_l) \sim -\frac{p^2}{(1-p)l}$. Ceci implique que pour l assez grand $f''(x_l) < 0$, et par conséquent x_l est un maximum pour la fonction f. □

Théorème 2 (Comportement asymptotique des statistiques après échantillonnage)**.**

$$\mathbb{P}(\hat{H} \geq l) \sim e^{-\left(\frac{l}{p}\right)^\beta (1+Apl^{\beta-1})^\beta} \frac{\left(1+Apl^{\beta-1}\right)^{l/p+Al^\beta}}{\left(1+\frac{Ap}{1-p}l^{\beta-1}\right)^{l(1/p-1)+Al^\beta}}$$

avec $A = -\beta\frac{1-p}{p^{\beta+1}}$.

Démonstration. L'équation (4.3) nous permet d'écrire :

$$F_l \sim e^{f(x_l)}\sum_{n=l}^{\infty} e^{\frac{f''(x_l)}{2}(n-x_l)^2}.$$

En scindant la somme ci-dessus en deux parties correspondant aux intervalles $[l, x_l]$ et $[x_l, +\infty[$, on peut écrire

$$\sum_{n=l}^{\infty} e^{\frac{f''(x_l)}{2}(n-x_l)^2} = \sum_{n=l}^{x_l} e^{\frac{f''(x_l)}{2}(n-x_l)^2} + \sum_{n=x_l}^{\infty} e^{\frac{f''(x_l)}{2}(n-x_l)^2}$$

Afin de trouver des équivalents simples à ces deux dernières quantités, il suffit de remarquer que la fonction $e^{\frac{f''(x_l)}{2}(x-x_l)^2}$ est monotone sur chacun des deux intervalles $[l, x_l]$ et $[x_l, +\infty[$. Ceci nous permet d'écrire

$$\int_{l-1}^{x_l} e^{\frac{f''(x_l)}{2}(x-x_l)^2}dx \leq \sum_{n=l}^{x_l} e^{\frac{f''(x_l)}{2}(n-x_l)^2} \leq \int_{l}^{x_l+1} e^{\frac{f''(x_l)}{2}(x-x_l)^2}dx$$

et

$$\int_{x_l}^{\infty} e^{\frac{f''(x_l)}{2}(x-x_l)^2}dx \leq \sum_{n=x_l}^{\infty} e^{\frac{f''(x_l)}{2}(n-x_l)^2} \leq \int_{x_l-1}^{\infty} e^{\frac{f''(x_l)}{2}(x-x_l)^2}dx.$$

En effectuant le changement de variable $t = \sqrt{-f''(x_l)}(x - x_l)$, on obtient

$$\int_{\sqrt{-f''(x_l)}(l-1-x_l)}^{0} \frac{e^{-t^2/2}}{\sqrt{-f''(x_l)}}dt \leq \sum_{n=l}^{x_l} e^{\frac{f''(x_l)}{2}(n-x_l)^2} \leq \int_{\sqrt{-f''(x_l)}(l-x_l)}^{\sqrt{-f''(x_l)}} \frac{e^{-t^2/2}}{\sqrt{-f''(x_l)}}dt$$

et

$$\int_{0}^{\infty} \frac{e^{-t^2/2}}{\sqrt{-f''(x_l)}}dt \leq \sum_{n=x_l}^{\infty} e^{\frac{f''(x_l)}{2}(n-x_l)^2} \leq \int_{-\sqrt{-f''(x_l)}}^{\infty} \frac{e^{-t^2/2}}{\sqrt{-f''(x_l)}}dt.$$

En utilisant le second résultat de la proposition 23, on obtient alors l'équivalence suivante

$$\sum_{n=l}^{\infty} e^{\frac{f''(x_l)}{2}(n-x_l)^2} \sim \int_{-\infty}^{+\infty} \frac{e^{-t^2/2}}{\sqrt{-f''(x_l)}}dt = \sqrt{\frac{-2\pi}{f''(x_l)}}. \tag{4.6}$$

Ceci nous permet d'écrire

$$F_l = \sum_{n=l}^{\infty} e^{f(n)} \sim e^{f(x_l)}\sqrt{\frac{-2\pi}{f''(x_l)}} \sim \frac{e^{f(x_l)}\sqrt{2\pi(1-p)l}}{p}.$$

Il reste à déterminer un équivalent pour $e^{f(x_l)}$. Le développement asymptotique de x_l dans la proposition 23 nous permet d'écrire

$$\frac{e^{f(x_l)}}{(l-1)!} \sim e^{(-l/p+Al^\beta)^\beta + \ln(1-p)(l/p+Al^\beta+B)} \frac{\Gamma(l/p + Al^\beta + B)}{\Gamma(l(1/p-1) + Al^\beta + B + 1)\Gamma(l)}.$$

En utilisant la formule de Stirling, on obtient

$$\begin{aligned}
&\frac{\Gamma(l/p + Al^\beta + B)}{\Gamma(l(1/p-1) + Al^\beta + B + 1)\Gamma(l)} \\
&\sim \frac{p}{\sqrt{2\pi(1-p)l}} \frac{(l/p + Al^\beta + B)^{l/p+Al^\beta+B}}{(l(1/p-1) + Al^\beta + B)^{l(1/p-1)+Al^\beta+Bl^l}} \\
&\sim \frac{p}{\sqrt{2\pi l}(1-p)^{B+1/2}} \frac{(l/p + Al^\beta)^{l/p+Al^\beta}}{(l(1/p-1) + Al^\beta)^{l(1/p-1)+Al^\beta l^l}} \\
&\sim \frac{p}{\sqrt{2\pi l}(1-p)^{B+1/2}} \frac{p^{-l}}{(1-p)^{l(1/p-1)+Al^\beta}} \frac{\left(1 + Apl^{\beta-1}\right)^{l/p+Al^\beta}}{\left(1 + \frac{Ap}{1-p}l^{\beta-1}\right)^{l(1/p-1)+Al^\beta}}.
\end{aligned}$$

Sachant que $\mathbb{P}(\hat{H} \geq l) \sim \mathbb{E}(e^{-T_l^\beta}) = \left(\frac{p}{1-p}\right)^l \frac{1}{(l-1)!} F_l$ et après quelques simplifications, on trouve le résultat énoncé. □

Corollaire 2. *Une loi Weibull obéit à la propriété 4.1 ssi* $0 < \beta < 1/2$.

Démonstration. Si $0 < \beta < 1/2$, alors

$$e^{-\left(\frac{l}{p}\right)^\beta (1+Apl^{\beta-1})^\beta} \sim e^{-\left(\frac{l}{p}\right)^\beta},\ (1+Apl^{\beta-1})^{l/p} \sim e^{Al^\beta}$$

$$\left(1+\frac{Ap}{1-p}l^{\beta-1}\right)^{l(1/p-1)} \sim e^{Al^\beta} \text{ et } \left(1+Apl^{\beta-1}\right)^{Al^\beta} \sim \left(1+\frac{Ap}{1-p}l^{\beta-1}\right)^{Al^\beta}.$$

Donc,

$$\mathbb{P}(\hat{H} \geq l) \sim e^{-\left(\frac{l}{p}\right)^\beta}.$$

Si $\beta \geq 1/2$, alors en écrivant l'équivalent de $\mathbb{P}(\hat{H} \geq l)$ énoncé au théorème 2 sous la forme $e^{Q(l)}$ où Q est un polynôme, on voit facilement que

$$\mathbb{P}(\hat{H} \geq l) \not\sim e^{-\left(\frac{l}{p}\right)^\beta}$$

□

On en conclut que pour une distribution Weibull de paramètre $\beta < 1/2$, la variable $\hat{H}$ est asymptotiquement de type Weibull dont le paramètre d'échelle est divisé par p.

Théorème 3 (Théorème centrale limite pour les variables N_l et $\hat{N}_l$)**.** *Si N ou l est très grand, on a les deux convergences suivantes,*

$$\frac{N_l - \mathbb{E}(N_l)}{\sqrt{\mathbb{E}(N_l)}} \xrightarrow[l\to+\infty]{loi} \mathcal{N}(0,1), \tag{4.7}$$

$$\frac{\hat{N}_l - \mathbb{E}(\hat{N}_l)}{\sqrt{\mathbb{E}(\hat{N}_l)}} \xrightarrow[l\to+\infty]{loi} \mathcal{N}(0,1). \tag{4.8}$$

Démonstration. Si N tend vers l'infini, alors, pour tout $l \in \mathbb{N}^*$, comme

$$N_l = \sum_{i=1}^{N} \mathbb{1}_{\{H_i \geq l\}}$$

et

$$\hat{N}_l = \sum_{i=1}^{N} \mathbb{1}_{\{\hat{H}_i \geq l\}}$$

et que ces indicatrices sont indépendantes, alors le théorème centrale limite nous donne directement le résultat.

Si maintenant N est fini et l très grand, l'équation (6.2) de l'annexe 6 nous permet d'écrire que

$$\sup_{y\in\mathbb{R}} \left| \mathbb{P}\left(\frac{N_l - \mathbb{E}(N_l)}{\sqrt{\mathbb{E}(N_l)}} \leq y\right) - \int_{-\infty}^{y} \frac{e^{-u^2/2}}{\sqrt{2\pi}}\, du \right| \leq d_{\mathcal{VT}}(\mathcal{L}(N_l),\ Q_{\mathbb{E}(N_l)}).$$

D'autre part, le théorème 5 de Chen-Stein dans l'annexe 6 nous donne une majoration de cette borne ne dépendant que de la moyenne et variance de N_l :

$$d_{\mathcal{VT}}(\mathcal{L}(N_l),\ Q_{\mathbb{E}(N_l)}) \leq 1 - \frac{\mathrm{Var}(N_l)}{\mathbb{E}(N_l)}.$$

Or

$$\mathbb{E}(N_l) = N\mathbb{P}(H \geq l)$$

et

$$\mathrm{Var}(N_l) = N\mathbb{P}(H \geq l)(1 - \mathbb{P}(H \geq l))$$

alors

$$d_{\mathcal{VT}}(\mathcal{L}(N_l), Po(\mathbb{E}(N_l))) \leq \mathbb{P}(H \geq l).$$

Comme pour l très grand, cette borne tend vers 0, alors on a le résultat. Le raisonnement est le même pour $\hat{N}_l$. □

Ce théorème prouve que pour des valeurs de l grandes, N_l est un bon estimateur de $\mathbb{E}(N_l)$ avec une erreur relative de l'ordre de $1/\sqrt{\mathbb{E}(N_l)}$.

4.1.4 Simulations

On dispose d'une trace de durée totale de 75 min qui contient approximativement 22 millions de paquets et 770 000 flots. Le flot le plus gros contient 177960 paquets. La figure 4.3 montre la distribution des tailles des flots de cette trace :

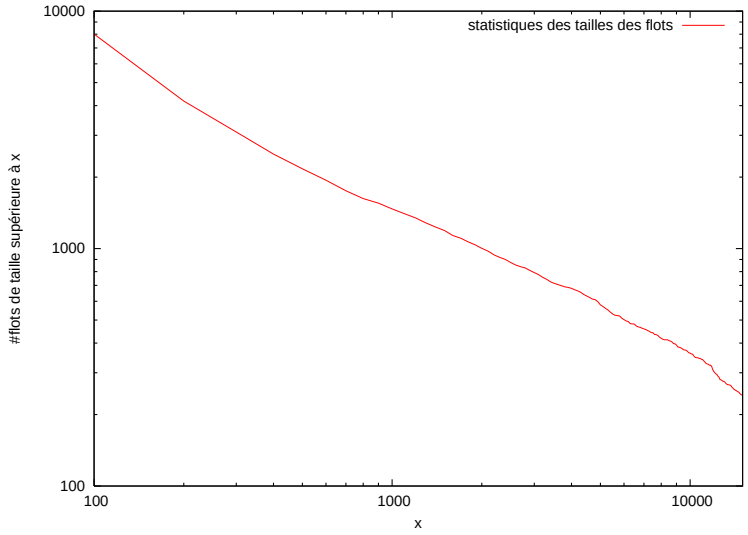


FIG. 4.3: Distribution des tailles des flots en échelle log-log

On voit bien que cette trace exhibe un comportement d'une loi Pareto où le logarithme de la CDF est proportionnel à celui des tailles des flots. On a considéré deux valeurs pour la probabilité d'échantillonnage, $p = 1/100$ et $p = 1/500$.

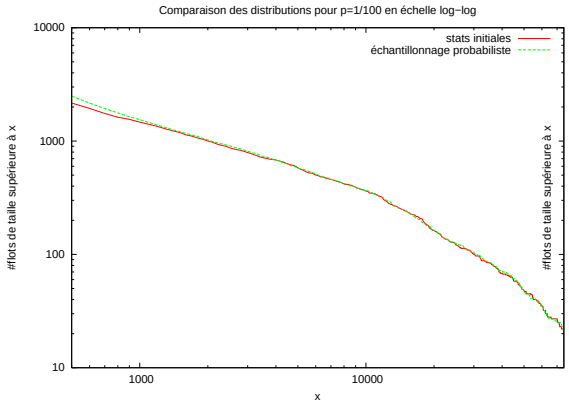


FIG. 4.4: Échantillonnage vs stats initiales, p=1/100

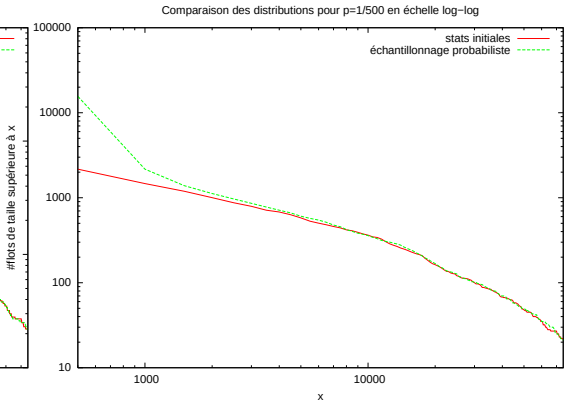


FIG. 4.5: Échantillonnage vs stats initiales, p=1/500

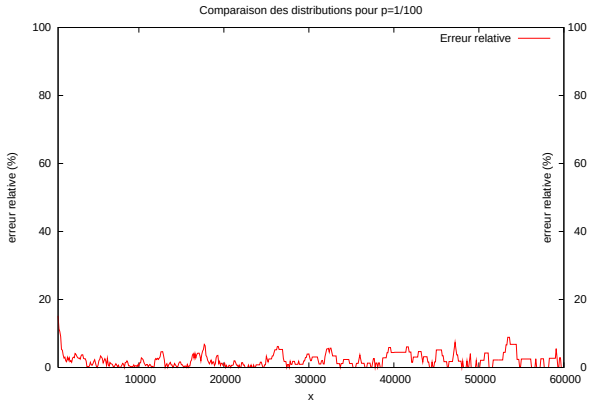


FIG. 4.6: Erreur relative, p=1/100

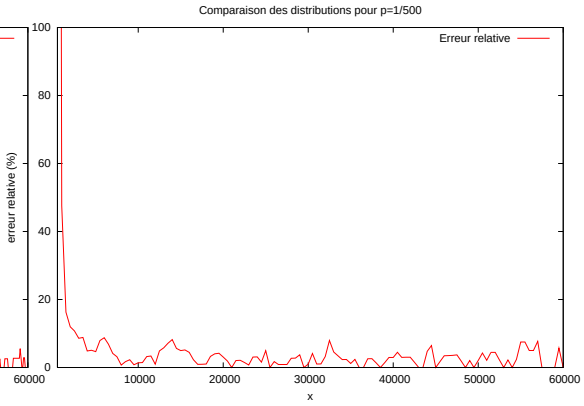


FIG. 4.7: Erreur relative, p=1/500

On s'aperçoit que, plus on augmente la probabilité d'échantillonnage p, meilleure est l'estimation, même lorsque la taille l des flots à laquelle on s'intéresse est petite, tandis que, lorsque p devient très petit, l'estimation ne reste valable que pour des tailles l très élevées.

4.2 Conclusion

On a réussi à montrer qu'on peut obtenir les statistiques des grands flots à partir des données échantillonnées pourvu que la distribution initiale des tailles des flots ait une propriété semblable à celle de queue lourde. En pratique ceci est vérifié, puisqu'il a été montré empiriquement que cette dernière suit en général une loi en puissance ou de type Weibull. Ceci a aussi comme application la caractérisation de la distribution des tailles des flots.

Chapitre 5

Conclusion générale

La thématique de cette thèse porte sur l'analyse de grands réseaux comme celui d'Internet. L'objectif est d'analyser et de comprendre les propriétés quantitatives du réseau. Cette thèse se décompose en deux parties complémentaires qui concernent la découverte de la topologie et la détection des grands flots selon certains critères. Le premier chapitre porte sur la caractérisation de la topologie Internet, plus particulièrement, nous avons étudié l'efficacité de traceroute pour explorer le réseau Internet. Pour ce faire, nous nous sommes basés sur l'hypothèse que le réseau Internet manifeste une structure hiérarchique et peut être approché au premier ordre par un arbre. Des sondes placées aléatoirement dans le réseau s'échangent des requêtes traceroute et permettent de découvrir les routeurs intermédiaires. On a considéré plusieurs architectures d'arbres (Réguliers, PLI et PLD) afin d'étudier l'efficacité de traceroute. On a retenu la proportion de routeurs découverts par rapport au nombre total de routeurs comme indicateur de l'efficacité de traceroute. On a aussi calculé la variance de cette quantité pour déduire l'effet de l'emplacement aléatoire des sondes. Dans un second temps, on a développé des asymptotiques quand le réseau devient très grand en maintenant le nombre de sondes proportionnel au nombre total de noeuds du réseau. Ces développements permettent de mieux comprendre l'évolution des propriétés quantitatives du réseau lors du passage à l'échelle. Comme dans la pratique le nombre de sondes est petit, on a aussi développé asymptotiquement ces quantités dans ce cas. Ceci permet de comprendre la vitesse initiale de découverte du réseau. Par exemple, dans un arbre régulier cette vitesse initiale est infinie et s'annule au fur et à mesure que le nombre de sondes croît. Nous avons aussi introduit des quantités qui traduisent la façon dont les noeuds sont répartis dans les différents niveaux de l'arbre et reflètent par conséquent le profil de l'arbre. Ces quantités permettent de comprendre et d'avoir une intuition sur la vitesse d'exploration. On a établi aussi la distribution du degré des routeurs découverts. Dans le chapitre 2, ces résultats ont été généralisés à des architectures d'arbres aléatoires de type Galton-Watson. Ces arbres présentent un aspect attrayant et généralisent les arbres déterministes étudiés auparavant. En effet, dans de tels arbres, chaque routeur possède un nombre aléatoire de voisins qui suit une distribution fixée. L'étude de ces arbres est motivée par le constat qu'empiriquement, il a été montré que le le degré des noeuds dans le

réseau internet suit une loi en puissance. On a montré plus particulièrement dans ce cas, qu'un arbre de Galton-Warson peut être approché au premier ordre par un arbre régulier de même degré moyen. Dans le chapitre 3 commence le second volet de la thèse. On s'est intéressé dans ce chapitre à la détection de grands flots. Ce problème est d'une importance primordiale puisqu'il permet de détecter les attaques et de facturer les gros transferts. On a développé deux algorithmes temps réel et à mémoire optimale, le premier se base sur les filtres de Bloom et introduit un mécanisme adaptatif qui permet de libérer la mémoire régulièrement. On a proposé plusieurs politiques d'effacement telles que, l'effacement total au bout de périodes fixes ou lorsque le taux de remplissage des filtres atteint un seuil critique, l'effacement progressif et l'ajout du filtre virtuel pour avoir plus de précision concernant les statistiques des éléphants surtout les plus petits. Une expérimentation de ces différentes politiques est conduite et une comparaison analytique est établie. Le chapitre 4 porte sur les méthodes de collectes de données par des techniques d'échantillonnage et permet d'établir un lien entre échantillonnage et queue de distribution lourde. Il permet entre autres de déduire les paramètre des distributions originales des tailles des flots en paquets.

Ces différents résultats se basent sur l'utilisation des techniques tels que la méthode de Stein-Chen, de Laplace, des calculs d'asymptotiques, de convergence en loi, de modèles d'urnes et du problème de collectionneur de coupons. Ces résultats peuvent s'appliquer aussi bien au monde des réseaux qu'à des domaines connexes tels que le traitement des bases de données ou bien pour étudier statistiquement des échantillons de populations.

Chapitre 6

ANNEXE : Introduction à la Méthode de Stein-Chen

Dans cette annexe, on va présenter le cadre général d'utilisation de la méthode de Stein-Chen et les principaux résultats dont ceux utilisés dans les chapitres 3 et 4. Pour plus de détails, se référer au livre [3].

La méthode de Stein-Chen sert à approcher la distribution de variables aléatoires sous forme de somme de plusieurs indicatrices par une variable poissonnienne. En effet, si on considère une variable aléatoire $W = I_1 + I_2 + \cdots + I_n$ où I_i est une variable aléatoire qui prend seulement les valeurs 0 ou 1, alors cette méthode nous permet d'approcher la variable W par une loi de poisson de paramètre $\mathbb{E}(W)$ sous certaines conditions.

Afin de montrer cette approximation, il faut montrer que la quantité $\sup_{A\subset\mathbb{N}} |\mathbb{P}(W \in A) - \mathbb{P}(Po(\mathbb{E}(W)) \in A)|$ est très petite.

Cette dernière quantité est appelée communément la distance en variation totale entre la distribution de W et celle d'une loi de Poisson de paramètre $\mathbb{E}(W)$ [41]. En général la distance en variation totale entre deux distributions U et V de mesures μ et σ est définie par :

$$d_{\mathcal{VT}}(\mathcal{L}(U), \mathcal{L}(V)) = \sup_{A\subset\mathbb{N}} |\mathbb{P}(U \in A) - \mathbb{P}(V \in A)|$$

On a ainsi la propriété suivante : pour tout sous ensemble A de $\mathbb{N}$,

$$|\mathbb{P}(U \in A) - \mathbb{P}(V \in A)| \leq d_{\mathcal{VT}}(\mathcal{L}(U), \mathcal{L}(V))$$

de telle sorte que cette distance donne une estimation de la déviation de la distribution de U par rapport à celle de V

Lemme 8 (Expression explicite de la distance en variation totale)**.** *Si U et V sont deux distributions de mesures respectives μ et σ alors*

$$d_{\mathcal{VT}}(\mathcal{L}(U), \mathcal{L}(V)) = \frac{1}{2}\sum_{n\geq 0} |\mathbb{P}(U = n) - \mathbb{P}(V = n)| \tag{6.1}$$

Démonstration. On note

$$\mathcal{A}^+ = \{j \in \mathbb{N},\ \mu_j \geq \sigma_j\} \text{ et } \mathcal{A}^- = \{j \in \mathbb{N},\ \mu_j < \sigma_j\}$$

$\mathcal{A}^+$ et $\mathcal{A}^+$ sont complémentaires dans $\mathbb{N}$. Pour tout sous ensemble B non vide de $\mathbb{N}$ différent de $\mathcal{A}^+$ et $\mathcal{A}^+$ tel que $B^+ = B \cap \mathcal{A}^+ \neq \emptyset$ et $B^- = B \cap \mathcal{A}^- \neq \emptyset$ on a

$$\begin{aligned}
&|\mathbb{P}(U \in B) - \mathbb{P}(V \in B)| = \\
&|(\mathbb{P}(U \in B^+) - \mathbb{P}(V \in B^+)) + (\mathbb{P}(U \in B^-) - \mathbb{P}(V \in B^-))| \\
&< \sup(\mathbb{P}(U \in B^+) - \mathbb{P}(V \in B^+), \mathbb{P}(V \in B^-) - \mathbb{P}(U \in B^-))) \\
&< \mathbb{P}(U \in A^+) - \mathbb{P}(V \in A^+)
\end{aligned}$$

Or,

$$\mathbb{P}(U \in A^+) - \mathbb{P}(V \in A^+) = \frac{1}{2} \sum_{n \geq 0} |\mathbb{P}(U = n) - \mathbb{P}(V = n)|$$

D' où le résultat □

Rappelons que la distance en variation totale d'une variable W par rapport à une variable poissonnienne de même moyenne ($\lambda = \mathbb{E}(W)$) contrôle l'écart avec une distribution gaussienne. Ce résultat peut servir pour démontrer le théorème centrale limite dans certains cas.

$$\sup_{y \in \mathbb{R}} \left| \mathbb{P}\left(\frac{W - \mathbb{E}(W)}{\sqrt{\mathbb{E}(W)}} \leq y \right) - \int_{-\infty}^{y} \frac{e^{-u^2/2}}{\sqrt{2\pi}} \, du \right| \leq d_{\mathcal{VT}}(\mathcal{L}(W), Po(\mathbb{E}(W))). \quad (6.2)$$

Dans la suite on note $W = I_1 + I_2 + \cdots + I_n$ où les variables aléatoires I_i sont des indicatrices qui ne peuvent prendre que des valeurs égales à 0 ou 1. On note $\lambda = \mathbb{E}(W)$ et $p_i = \mathbb{E}(I_i)$. Considérons donc un sous ensemble quelconque $A \subset \mathbb{N}$, On construit alors la fonction suivante :
$g = g_{\lambda,A} : \mathbb{N} \to \mathbb{R}$ de la façon suivante

$$\lambda g(j+1) - j g(j) = \mathbf{1}[j \in A] - Po(\lambda)\{A\}, \ j \geq 0 \quad (6.3)$$

La valeur de $g(0)$ importe peu, celle ci est prise par défaut égale a 0. La solution de l'équation (6.3) est obtenu récursivement à partir de $j = 0$. Il s'en suit immédiatement que pour toute variable aléatoire a valeurs dans $\mathbb{N}$ on peut écrire :

$$\mathbb{P}[W \in A] - Po(\lambda)\{A\} = \mathbb{E}\{\lambda g(W+1) - W g(W)\} \quad (6.4)$$

Ainsi, il suffit de pouvoir majorer le terme à droite de l'équation (6.4) uniformément par rapport à tous les ensembles A. Pour ce faire, on va établir quelques propriétés de la fonction g. Tout d'abord remarquons qu'on peut réécrire l'équation 6.3 de la façon suivante

$$\begin{aligned}
&\lambda^{j+1} \frac{g(j+1)}{j!} e^{-\lambda} - \lambda^j \frac{g(j)}{(j-1)!} e^{-\lambda} = \\
&Po(\lambda)\{A \cap \{j\}\} - Po(\lambda)\{A\} Po(\lambda)\{\{j\}\}, \quad j \geq 0
\end{aligned}$$

Donc, si on note $U_j = \{0, 1, \cdots, j\}$ et en additionnant ces dernières équations de 0 jusqu' a j on obtient

$$g(j+1) = \lambda^{-j-1} j! e^{\lambda} (Po(\lambda)\{A \cap U_j\} - Po(\lambda)\{A\} Po(\lambda)\{U_j\}), \quad j \geq 0 \quad (6.5)$$

Définissons aussi :

$$||g|| = ||g_{\lambda,A}|| = \sup_j |g_{\lambda,A}(j)|; \ \Delta g = \Delta g_{\lambda,A} = \sup_j |g_{\lambda,A}(j+1) - |g_{\lambda,A}(j)|$$

Le lemme suivant utilise l'expression de la fonction g dans (6.5) afin d'établir des majorations pour ces deux quantités.

Lemme 9 (Propriétés de la fonction g). *Si la fonction g résout l'équation* (6.3), *alors on a :*

$$||g|| \leq \min(1, \lambda^{-1/2}), \ \Delta g \leq \lambda^{-1}(1 - e^{-\lambda}). \quad (6.6)$$

Pour deux sous ensembles $A, B \subset \mathbb{N}$ On a :

$$g_{\lambda,A\cup B} = g_{\lambda,A} + g_{\lambda,B} - g_{\lambda,A\cap B} \quad (6.7)$$

Ce lemme donne une majoration de la fonction g indépendemment de l'ensemble A. On s'aperçoit que la fonction g est bornée. Afin d'illustrer une application de l'équation (6.4), considérons le cas où les variables aléatoires $I_1, I_2, \cdots, I_n$ sont indépendantes. On peut écrire dans ce cas

$$\mathbb{E}(I_i g(W)) = \mathbb{E}(I_i g(W_i + 1)) = p_i \mathbb{E} g(W_i + 1)$$

où $W_i = W - I_i$ à cause de l'indépendance de I_i et W_i. Donc,

$$\mathbb{E}\{\lambda g(W+1) - W g(W)\} = \sum_{i=1}^{n} p_i \mathbb{E}\{g(W+1) - g(W_i + 1)\}$$

Comme les variables W et W_i sont égales sauf si $I_i = 1$ auquel cas $W - W_i = 1$ et que cet évènement se produit avec probabilité p_i, il est donc clair que

$$|\mathbb{P}[W \in A] - Po(\lambda)\{A\}| \leq \Delta g \sum_{i=1}^{n} p_i^2 \leq 2||g|| \sum_{i=1}^{n} p_i^2 \quad (6.8)$$

Finalement on obtient

$$d_{\mathcal{VT}}(\mathcal{L}(W), Po(\lambda)) \leq \lambda^{-1}(1 - e^{-\lambda}) \sum_{i=1}^{n} p_i^2 \leq \min(1, e^{-\lambda}) \sum_{i=1}^{n} p_i^2 \quad (6.9)$$

On a utilisé l'indépendance des variables $I_1, I_2, \cdots, I_n$ seulement pour établir que $\mathbb{E}(I_i g(W)) = p_i \mathbb{E} g(W_i + 1)$. Dans le cas où ces variables ne sont pas indépendantes, on peut écrire tout simplement : $\mathbb{E}(I_i g(W)) = p_i \mathbb{E} g(W | I_i = 1)$ qui mène à l'équation

$$\mathbb{P}[W \in A] - Po(\lambda)\{A\} = \sum_{i=1}^{n} p_i [\mathbb{E} g(W+1) - \mathbb{E}\{g(W | I_i = 1)\}] \quad (6.10)$$

Si on suppose qu'on peut construire un couplage (U, V), c.à.d pour chaque valeur de i des variables aléatoires U_i et V_i de telle façon que $U_i \overset{\mathcal{D}}{=} W$ et $V_i + 1$ a la même distribution que W conditionnellement à $I_i = 1$ alors le résultat suivant est immédiat.

Théorème 4. *Avec les notations précédentes et quelque soit le choix du couplage* (U, V) *on a*

$$d_{\mathcal{VT}}(\mathcal{L}(W), Po(\lambda)) = |\sum_{i=1}^{n} p_i[\mathbb{E}g(U_i+1) - \mathbb{E}\{g(V_i+1)\}]| \leq \Delta g \sum_{i=1}^{n} p_i \mathbb{E}|U_i - V_i| \tag{6.11}$$

Comme $\Delta g \leq \lambda^{-1}$, la distance en variation totale entre la distribution de W et celle dune variable poissonnienne de paramètre λ est inférieure à la moyenne de quantités $\mathbb{E}|U_i - V_i|$ pondérées par les poids p_i. Par conséquent si le couplage (U, V) peut être construit de telle façon que les quantités $\mathbb{E}|U_i - V_i|$ sont petites, alors le théorème 4 produit une excellente approximation poissonnienne. Si on revient au cas ou les variables aléatoires $I_1, I_2, \cdots, I_n$ sont indépendantes, il suffit de prendre comme couplage $U_i = W$ et $V_i = W_i$.

Le théorème précédent peut être simplifié davantage si on peut construire un couplage monotone c.à.d $U_i \geq V_i$ presque sûrement, auquel cas

$$\begin{aligned}\sum_{i=1}^{n} p_i \mathbb{E}|U_i - V_i| &= \sum_{i=1}^{n} p_i \mathbb{E}\{(U_i+1) - (V_i+1)\} = \lambda \mathbb{E}(W+1) - \sum_{i=1}^{n} \mathbb{E}(I_i W) \\ &= \lambda - \mathrm{Var}W\end{aligned}$$

Théorème 5. *Avec les notations précédentes et sous condition de l'existence d'un couplage monotone* (U, V) *on a*

$$d_{\mathcal{VT}}(\mathcal{L}(W), Po(\lambda)) \leq (1 - e^{-\lambda}) \left(1 - \frac{\mathrm{Var}W}{\lambda}\right) \tag{6.12}$$

Ce dernier théorème établit une majoration de la distance en variation totale qui ne dépend pas du couplage monotone construit mais nécessite seulement le calcul de la moyenne et la variance de W. Pour une variable très proche d'une variable poissonnienne sa moyenne est très proche de sa variance et donc on retrouve bien que sa distance en variation totale est très petite.

On note dorénavant $\Gamma = \{1, 2, \cdots, n\}$. On suppose que pour toute valeur $i \in \Gamma$ il existe des variables aléatoires $(J_{ji}; j \in \Gamma)$ définies dans le même espace de probabilité que $(I_j; j \in \Gamma)$ vérifiant

$$\mathcal{L}(J_{ji}; j \in \Gamma) = \mathcal{L}(I_j; j \in \Gamma | I_i = 1) \tag{6.13}$$

et notons $V_i = \sum_{j \neq i} Jji$. Il découle du théorème 4 que

$$\begin{aligned}d_{\mathcal{VT}}(\mathcal{L}(W), Po(\lambda)) &\leq \frac{1 - e^{-\lambda}}{\lambda} \sum_{i \in \Gamma} p_i \mathbb{E}|W - V_i| \\ &= \frac{1 - e^{-\lambda}}{\lambda} \sum_{i \in \Gamma} p_i \mathbb{E}|I_i + \sum_{j \in \Gamma} (I_j - J_{ji})|\end{aligned}$$

Cette dernière expression peut être sensiblement simplifiée sous des hypothèses additionnelles sur les variables J_{ji}. En effet, supposons que pour chaque

i, l'ensemble $\Gamma_i = \Gamma - \{i\}$ peut être partitionné en sous ensembles Γ_i^-, Γ_i^+ et Γ_i^0 tels que

$$J_{ji} \begin{cases} \leq I_j, & si \quad j \in \Gamma_i^- \\ \geq I_j, & si \quad j \in \Gamma_i^+ \end{cases} \tag{6.14}$$

Si aucune des deux conditions n'est vérifiée alors $j \in \Gamma_i^0$. Dans les cas particuliers ou $\Gamma_i = \Gamma_i^-$ ou $\Gamma_i = \Gamma_i^+$ on introduit la définition suivante.

Définition 1. *Les variables aléatoires indicatrices $(I_i; i \in \Gamma)$ sont dites positivement liées si pour chaque i des variables aléatoires $(J_{ji}; j \in \Gamma)$ peuvent être construites de telle façon que $\Gamma_i = \Gamma_i^+$ et négativement liées si $\Gamma_i = \Gamma_i^-$*

Afin d'exploiter la partition de Γ définie par (6.14), on peut observer que

$$p_i \mathbb{E} J_{ji} = \mathbb{P}(I_i = 1)\mathbb{E}(I_j | I_i = 1) = \mathbb{E}(I_i I_j)$$

Par conséquent, la covariance de I_i et de I_j peut être exprimée comme suivant

$$\mathrm{Cov}(I_i, I_j) = \mathbb{E}(I_i I_j) - p_i p_j = p_i \mathbb{E}(J_{ji} - I_j) \begin{cases} \leq 0, & si \quad j \in \Gamma_i^- \\ \geq 0, & si \quad j \in \Gamma_i^+ \end{cases}$$

Ainsi

$$\begin{aligned} p_i \mathbb{E}|I_i + \sum_{j \in \Gamma_i} (I_j - J_{ji})| &\leq p_i \mathbb{E} I_i + p_i \mathbb{E} \sum_{j \in \Gamma_i^-} (I_j - J_{ji}) + p_i \mathbb{E} \sum_{j \in \Gamma_i^+} (J_{ji} - I_j) \\ &\quad + p_i \mathbb{E} \sum_{j \in \Gamma_i^0} (I_j + J_{ji}) \\ &= p_i^2 - \sum_{j \in \Gamma_i^-} \mathrm{Cov}(I_i, I_j) + \sum_{j \in \Gamma_i^+} \mathrm{Cov}(I_i, I_j) \\ &\quad + \sum_{j \in \Gamma_i^0} (\mathbb{E}(I_i I_j) + p_i p_j) \end{aligned}$$

Nous avons donc démontré le théorème suivant

Théorème 6.

$$\begin{aligned} d_{\mathcal{VT}}(\mathcal{L}(W), Po(\lambda)) \leq \frac{1 - e^{-\lambda}}{\lambda} \Bigg(&\sum_{i \in \Gamma} p_i^2 + \sum_{i \in \Gamma} \sum_{j \in \Gamma_i^-} |\mathrm{Cov}(I_i, I_j)| \\ &+ \sum_{i \in \Gamma} \sum_{j \in \Gamma_i^+} \mathrm{Cov}(I_i, I_j) + \sum_{i \in \Gamma} \sum_{j \in \Gamma_i^0} (\mathbb{E}(I_i I_j) + p_i p_j) \Bigg) \end{aligned}$$

Les corollaires suivants sont des résultats immédiats de ce théorème

Corollaire 3. *Si les variables aléatoires $(I_i; i \in \Gamma)$ sont positivement liées ($\Gamma_i = \Gamma_i^+$) alors*

$$d_{\mathcal{VT}}(\mathcal{L}(W), Po(\lambda)) \leq \frac{1 - e^{-\lambda}}{\lambda} \left(\mathrm{Var} W - \lambda + 2 \sum_{i \in \Gamma} p_i^2 \right)$$

Corollaire 4. *Si les variables aléatoires* $(I_i; i \in \Gamma)$ *sont négativement liées (*$\Gamma_i = \Gamma_i^-$*) alors*

$$d_{\mathcal{VT}}(\mathcal{L}(W), Po(\lambda)) \leq (1 - e^{-\lambda}) \left(1 - \frac{\mathrm{Var}W}{\lambda}\right)$$

Notons finalement que la méthode de Stein-Chen permet de majorer aussi la différence entre $\mathbb{E}f(W)$ et $Po(\lambda)\{f\}$ pour des fonctions tests f où la dernière quantité désigne la moyenne de f respectivement à la distribution $Po(\lambda)$. En effet, pour une fonction donnée f, il suffit de résoudre de la même façon que précédemment l'équation

$$f(j) - Po(\lambda)\{f\} = \lambda g(j+1) - jg(j) \tag{6.15}$$

pour la fonction inconnue g. Ceci nous permet d'écrire

$$\mathbb{E}f(W) - Po(\lambda)\{f\} = \mathbb{E}\{\lambda g(j+1) - jg(j)\} \tag{6.16}$$

Donc, en établissant des majorations adéquates pour $||g||$ et Δg, on obtient une majoration de $\mathbb{E}f(W) - Po(\lambda)\{f\}$. Ceci montre que cette méthode peut être appliquée dans le contexte de toute distance d qui vérifie

$$d(\mathcal{L}(W), Po(\lambda)) = \sup_{f \in \mathcal{F}} |\mathbb{E}f(W) - Po(\lambda)\{f\}| \tag{6.17}$$

Pour un ensemble de fonctions tests $\mathcal{F}$. La distance de Wasserstein fait partie de cet ensemble. Pour la distance en variation totale, l'ensemble $\mathcal{F}_{\mathcal{VT}}$ est celui des fonctions $(f_A, A \subset \mathbb{N})$ de la forme

$$f_A = \mathbf{1}[. \in A]$$

Chapitre 7

ANNEXE : Résumé du chapitre sur la topologie

Un modèle pour la découverte de la topologie Internet est proposé dans ce chapitre. L'efficacité des méthodes basées sur traceroute pour déterminer l'ensemble des routeurs d'un réseau de collecte est étudiée. Sous certaines hypothèses, des expressions explicites sont obtenues pour le nombre moyen de routeurs découverts quand un nombre de routeurs s'échangent des messages traceroute. Plusieurs types d'arbres sont abordés et des résultats asymptotiques sont détaillés lorsque la taille du réseau est grande. Tous ces résultats sont comparés aux données réelles obtenues par mesure.

– **Mots clés : Internet, topologie, traceroute, architectures d'arbre**

7.1 Introduction

L'étude de la topologie Internet est importante à plusieurs égards. Elle permet de bien maîtriser la topologie afin d'équilibrer le routage et la charge sur les liens, de simuler de nouveaux algorithmes, d'optimiser les performances de certains protocoles comme le multicast et aussi de repérer les points névralgiques afin de limiter la propagation des attaques (vers, DDoS, etc.).

Durant les dernières années, plusieurs projets et travaux se sont attelés à l'étude de la topologie Internet. A premier abord, celle-ci parait désordonnée à cause du manque de structure hiérarchique. Pour déterminer la structure d'un réseau, plusieurs méthodes peuvent être citées. Une approche populaire consiste à utiliser la théorie de graphes complexes (e.g. Erdös et Rényi, petit monde, etc.) en se basant sur des propriétés locales observées par mesure comme la loi des degrés des noeuds qui suit une loi Pareto [16].

Plusieurs projets s'appuient sur l'utilisation de traceroute pour la découverte de la topologie (CAIDA, AT&T, etc.). Traceroute permet de découvrir les noeuds intermédiaires entre une sources et une destination. Même si traceroute exige que le routage entre la source et la destination ne change pas au cours du déroulement de la procédure et que certains routeurs intermédiaires rejettent les messages traceroute, cette méthode fournit une bonne sur la structure globale du réseau.

La topologie peut aussi être obtenue en se basant sur celle du niveau des AS. En effet, un fournisseur d'accès connaît la topologie au sein de son système autonome en écoutant les messages IGP échangés entre les routeurs. Les informations des tables BGP fournit des données sur les interconnexions des AS.

De plus, afin de mieux comprendre la structure Internet, il est primordial de prendre en considération les différents composants du réseau (réseaux locaux, d'accès, de collecte ou de transit). Les réseaux de collecte sont des réseaux denses, présentant la propriété de petit monde. Ils ressemblent à des arbres et sont reliés entre eux par des réseaux de transit très rapides.

Objectif de ce chapitre

Dans ce chapitre on focalise sur la topologie des réseaux de collecte. Ceux du transit sont composés de peu de routeurs. La topologie de telles réseaux n'est pas exactement un arbre mais ce modèle peut servir comme première approche. De plus, il faut aussi prendre en compte d'autres facteurs comme l'instabilité des liens, les routeurs qui ne répondent pas aux requêtes traceroute, etc.

On considère des réseaux en arbre non homogènes où le degré d'un noeud dépend de son niveau dans l'arbre. Les sondes traceroute sont réparties aléatoirement parmi les feuilles de l'arbre et elles s'échangent des messages traceroute. On établit le nombre de liens découverts et montre en particulier que celui-ci croît significativement pour des valeurs modérés de nombre de sondes tandis qu'il se stabilise pour un nombre élevé de sondes. Ceci indique que la découverte de la topologie complète requière un déploiement massif des sondes traceroute.

Voici l'organisation de ce chapitre : Dans la section 8.3.2, les variables importantes et les topologies étudiées sont introduites. La section 7.3 donne une expression explicite du nombre moyen de noeuds découverts. L'analogie avec le problème de collectionneur de coupon est faite. Afin de mieux comprendre ces résultats et la vitesse d'exploration de réseau, la section 7.4 donne des résultats asymptotiques quand ce réseau est très grand. Dans la section 7.5, on étudie le degré des noeuds vus par traceroute. Des développement concernant la variance sont obtenus dans la section 7.6 et des développement asymptotiques dans la section 7.7. Ces différents résultats ont été comparés avec des mesures réelles obtenues par le projet Scan+Lucent des laboratoires Lucent Bell.

7.2 Notations et définitions

On suppose que le graphe étudié est un arbre de hauteur $n \geq 2$. Pour $1 \leq j < n$, un noeud au niveau j a d_j fils. Le nombre de noeuds au niveau j est noté l_j et N le nombre total de noeuds dans l'arbre.

$$l_j = \prod_{k=1}^{j-1} d_k \quad \text{and} \quad N = \sum_{j=1}^{n} l_j,$$

avec $l_1 = 1$.

L'ensemble S des sondes est composé de p éléments. Celles-ci sont placées aléatoirement sur les feuilles au niveau n et s'échangent des requêtes trace-route. Les routeurs découverts sont recouverts par un arbre minimal de hauteur $H(N,p)$. Il est clair que, pour $j \leq n$, on a $H(N,p) < n-j$ si aucun routeur du niveau j n'est découvert.

On note aussi $D_j(N,p)$ (resp $D(N,p)$) le nombre total de noeuds du niveau j découverts (resp dans tout l'arbre).

On considère en particulier trois types d'arbres :

Définition 2. *1. Arbre régulier (arbre REG_d) de degré d. Ici, $d_j = d \geq 2$, $\forall j \in \{1,\ldots,n\}$.*

2. Arbres en puissance de paramètre $\alpha > 0$, $1 \leq j \leq n$,
— Arbre croissant (arbre PLI_α) si
$d_j = \lfloor j^\alpha \rfloor$,
— Arbre décroissant (arbre PLD_α) si
$d_j = \lfloor (n-j)^\alpha \rfloor$,
où $\lfloor y \rfloor$ désigne la partie entière de $y \in \mathbb{R}$.

7.3 Nombre moyen de noeuds découverts

Proposition 24. *La moyenne du nombre de routeurs découverts $D(N,p)$ est*

$$\mathbb{E}(D(N,p)) = \sum_{j=1}^{n} l_j \left(1 - \left(1 - \frac{1}{l_j}\right)^p\right) - \sum_{j=2}^{n} \frac{1}{l_j^{p-1}}. \tag{7.1}$$

Ce résultat est obtenu en constant qu'un noeud A n'est pas découvert dans deux cas (cf figure 7.1). Le premier est quand les p sondes appartiennent toutes au sous-arbre de l'un de ses fils. Le deuxième cas est quand aucune sonde n'appartient au sous arbre de A.

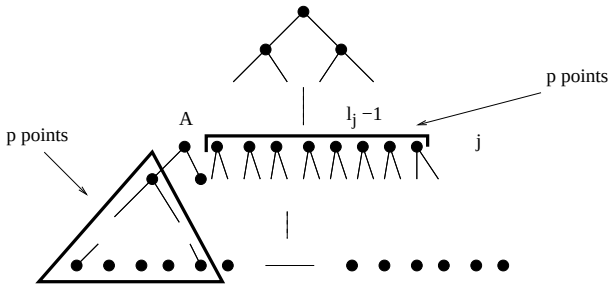


Fig. 7.1: Les deux situations où le noeud A n'est pas découvert

D'autre part, le problème de collectionneur de coupons permet de calculer la distribution de la variable $D_j(N,p)$. En effet, on a

$$\mathbb{P}(D_j(N,p) = l_j) = \frac{l_j! \left\{ \begin{array}{c} p-1 \\ l_j - 1 \end{array} \right\}}{l_j^p} \tag{7.2}$$

où $\left\{ \begin{array}{c} p-1 \\ l_j-1 \end{array} \right\}$ est un nombre de Stirling du second type. Ceci, permet d'avoir la distribution du nombre de sondes nécessaires pour découvrir la totalité du réseau. Ceci est aussi possible dans le cas d'arbres asymétriques, où les noeuds d'un même niveau ne sont pas équivalents. Aussi l'application du collectionneur de coupons reste valable pour d'autres topologies que des arbres.

7.4 Résultats asymptotiques

Afin de mieux comprendre la découverte de la topologie, on suppose que le nombre de sondes p est proportionnel au nombre total de noeuds N : $p = \lambda N$ et que N tend vers l'infini. On s'intéresse plus particulièrement à la proportion de noeuds découverts $\mathbb{E}(D(N,p))/N$ quand N tend vers l'infini.

On note $T(\lambda) = \lim_{N\to\infty} \mathbb{E}(D(N,p))/N$. Celle-ci désigne le taux d'exploration.

Proposition 25. *Quand $N \to +\infty$ et $p/N \to \lambda$, on obtient les expressions suivantes de $T(\lambda)$ pour les trois architectures d'arbres abordées*

$$T_{REG_d}(\lambda) \stackrel{def.}{=} \sum_{j=1}^{+\infty} \frac{d-1}{d^j} \left(1 - \exp\left(-\frac{\lambda d^j}{d-1}\right)\right) \tag{7.3}$$

$$T_{PLI_\alpha}(\lambda) \stackrel{def.}{=} 1 - e^{-\lambda} \tag{7.4}$$

$$T_{PLD_\alpha}(\lambda) \stackrel{def.}{=} \sum_{j=1}^{+\infty} \frac{1}{H(\alpha)(j!)^\alpha} \left(1 - e^{-\lambda H(\alpha)(j!)^\alpha}\right) \tag{7.5}$$

Où $H(\alpha) = \sum_{j=1}^{+\infty} 1/(j!)^\alpha$.

En pratique, le nombre de sondes p est très petit par rapport au nombre total de noeuds. Ainsi λ est très petit et donc, il est très utile d'étudier le comportement de ces quantités pour λ petit. Ceci, nous donne aussi une idée sur la vitesse d'exploration quand peu de sondes sont déployées.

Proposition 26. *Quand $\lambda \to 0$, on a les développements suivants pour la vitesse d'exploration des différents arbres*

$$T_{REG_d}(\lambda) \sim -\lambda \log_d(\lambda) \tag{7.6}$$

$$T_{PLI_\alpha}(\lambda) \sim \lambda \tag{7.7}$$

$$T_{PLD_\alpha}(\lambda) \sim \frac{\lambda \log(1/\lambda)}{\alpha \log\log(1/\lambda)}. \tag{7.8}$$

Afin de fournir une explication intuitive du comportement initiale (avec peu de sondes) du processus d'exploration, on a définit des quantités qui reflètent la structure de l'arbre. Ces quantités expriment le fait que plus l'arbre est concentré dans les niveaux inférieurs (proche des feuilles), plus lente est la vitesse de l'exploration. Ceci s'explique par le fait que les requêtes ne remontent pas suffisamment dans l'arbre et restent limitées aux niveaux inférieurs.

7.5 Distribution du degré des noeuds découverts

Nous avons aussi abordé d'autres questions comme la moyenne et plus généralement la loi du degré des noeuds découverts. Tout d'abord, on définit la probabilité de bernoulli suivante

$$b(p,m,l_j) = \binom{p}{m}\left(\frac{1}{l_j}\right)^m\left(1-\frac{1}{l_j}\right)^{p-m}. \tag{7.9}$$

C'est la probabilité qu'un noeud du niveau j ait m sondes parmi les p dans son sous-arbre. En utilisant des résultats du collectionneur de coupon, on obtient facilement les résultats suivants

Proposition 27. *Le degré L d'un noeud découvert est donné par : pour $k \geq 1$,*

$$\mathbb{P}(L=k) = \sum_{j=1}^{n}\frac{l_j}{N}\frac{(d_j)_k}{1-\left(1-\frac{1}{l_j}\right)^p}\sum_{m=k}^{p}\frac{b(p,m,l_j)}{d_j^m}\left\{\begin{array}{c} m \\ k \end{array}\right\} \tag{7.10}$$

et

$$\mathbb{E}(L) = \sum_{j=1}^{n}\frac{l_j d_j}{N}\frac{1-\left(1-\frac{1}{d_j l_j}\right)^p}{1-\left(1-\frac{1}{l_j}\right)^p}. \tag{7.11}$$

où $\left\{\begin{array}{c} m \\ k \end{array}\right\}$ *est un nombre de stirling du second type.*

Ces formules montrent plus particulièrement que lorsque le nombre de sondes est petit, les noeuds découverts ne sont vus en moyenne qu'une seule fois.

7.6 Variance

On a par la suite étudié la variance du nombre de noeuds découvert et on a développé des asymptotiques quand la taille du réseau est grande pour les différentes architectures d'arbres étudiées.

$$\begin{aligned}
\mathrm{Var}(D(N,p)) = &\\
&\sum_{j=0}^{n-1} l_j\left(\left(1-\frac{1}{l_j}\right)^p-\left(1-\frac{1}{l_j}\right)^{2p}\right)\\
&+2\sum_{j=0}^{n-2}\sum_{k=j+1}^{n-1} l_k\left(\left(1-\frac{1}{l_j}\right)^p-\left(1-\frac{1}{l_j}\right)^p\left(1-\frac{1}{l_k}\right)^p\right)\\
&+2\sum_{j=1}^{n-1}\sum_{k=j+1}^{n-1}(l_j l_k-l_k)\left(\left(1-\frac{1}{l_j}-\frac{1}{l_k}\right)^p-\left(1-\frac{1}{l_j}\right)^p\left(1-\frac{1}{l_k}\right)^p\right)\\
&+\sum_{j=1}^{n-1}(l_j^2-l_j)\left(\left(1-\frac{2}{l_j}\right)^p-\left(1-\frac{1}{l_j}\right)^{2p}\right)\\
&+o(N)
\end{aligned}$$

7.7 Asymptotiques de la variance

7.7.1 Asymptotique de la variance pour N grand

On définit les quantités $(a_j)_{j\in\mathbb{N}}$ de la façon suivante

$$\forall j \geq 1 : \lim_{n\to+\infty} \frac{N}{l_{n-j}} = a_j \tag{7.12}$$

On obtient le résultat asymptotique suivant quant $N \to \infty$ et $p/n \to \lambda$

$$V(\lambda) \stackrel{def}{=} \lim_{n\to+\infty} \frac{\mathrm{Var}(D(N,p))}{N} =$$
$$= \lambda \left(2\sum_{j=1}^{+\infty} e^{-\lambda a_k} \left(\sum_{k=1}^{j} \left(\frac{1-e^{-\lambda a_k}}{\lambda a_k} - e^{-\lambda a_k} \right) \right) - \sum_{j=1}^{+\infty} e^{-\lambda a_j} \left(\frac{1-e^{-\lambda a_j}}{\lambda a_j} - e^{-\lambda a_j} \right) \right)$$

7.7.2 Asymptotique de la variance pour λ petit

On définit tout d'abord la fonction f comme suivant

$$\forall x \geq 0 \ : f(x) = \sum_{j=1}^{+\infty} \mathbb{1}_{\{x \geq a_j\}} \tag{7.13}$$

On obtient alors,

Proposition 28.

$$V(\lambda) \sim \lambda^2 a_1 f\left(\frac{1}{\lambda}\right) \tag{7.14}$$

7.8 Conclusion

Nous avons étudié la découverte de la topologie par traceroute dans un réseau de collecte sous forme d'arbre. Nous avons obtenu des expressions de la moyenne du nombre de noeuds découverts, la loi des degrés des routeurs découverts et la variance. Nous avons développé des asymptotiques de ces expressions quand le réseau est très grand et aussi quand le nombre de sondes est petit.

Chapitre 8

ANNEXE : Résumé du chapitre sur les arbres Galton Watson

– **Mots clés : arbres de Galton Watson**

8.1 Introduction

Dans ce chapitre, on va étudier l'efficacité de traceroute pour la découverte de la topologie d'Internet. Le réseau perçu à partir d'un noeud est sous forme d'arbre. De plus, à cause de l'hiérachie de l'Internet, l'hypothèse d'arbre est très réaliste et seulement une petite partie des noeuds n'appartiennent pas à des arbres. On considère plus particulièrement le cas des arbres Galon Watson. Ceux-ci sont des arbres aléatoires où le degré des noeuds suit une variable aléatoire. Dans ce cas, chaque noeud génère un nombre aléatoire de fils selon cette distribution et indépendemment des autres noeuds. En effet, il a été montré empiriquement que la distribution des noeuds manifeste la propriété de queue lourde et par conséquent, quelques noeuds peuvent avoir des degrés beaucoup plus élevés par rapport à la moyenne. Ces arbres généralisent les cas étudiés dans le chapitre précèdent et englobent aussi bien les arbres déterministes qu'aléatoires. L'efficacité de traceroute est quantifiée par la proportion de noeuds découverts. On a établit aussi des formules analytiques pour la variance et des développements asymptotiques quand la taille du réseau devient grande.

8.2 Hypothèses et notations

Dans ce chapitre, on suppose que le graphe est un arbre de Galton Watson de hauteur $n \geq 2$. Chaque noeud génère de façon indépendante son degré selon une distribution fixée.

Notations

Pour $n \geq 0$, Z_n désigne le nombre de noeuds au niveau n (n ème génération). Pour $\ell \in \{1, \ldots, Z_n\}$, un noeud est représenté par le couple (n, ℓ), où n est sa génération et ℓ son rang à l'intérieur de sa génération. On note, R_N le nombre

total de noeuds découverts dans l'arbre. $\mathcal{T}$ désigne l'arbre complet et la quantité $\mathcal{T}_k^{n,\ell}$ désigne le sous-arbre de $\mathcal{T}$ de profondeur égale à k et dont la racine est (n,ℓ). La taille de $\mathcal{T}_k^{n,\ell}$ est désignée par $T_k^{n,\ell}$. Quand l'indice (n,ℓ) est omis, cette quantité représente une variable aléatoire de même distribution que la taille de l'arbre jusqu'au niveau k, c'est à dire, de même distribution que $Z_1 + \cdots + Z_k$.

Avec ces notations, il est facile de déduire que,

$$T_N = \sum_{i=0}^{n-1} Z_i + \sum_{\ell=1}^{Z_n} T_{N-n}^{n,\ell}. \tag{8.1}$$

La variable $\mathcal{N}$ est une mesure de comptage. Par exemple, pour un sous-ensemble A de $\mathcal{T}$, $\mathcal{N}(A)$ désigne le nombre total de sondes placées dans A. Un noeud (n,ℓ) est découvert ssi $\mathcal{N}(T_{N-n}^{n,\ell}) \neq 0$, en particulier la taille total R_N des noeuds découverts est donnée par

$$R_N = \sum_{n=0}^{N} \sum_{\ell=1}^{Z_{N-n}} \mathbf{1}\{\mathcal{N}(T_n^{N-n,\ell}) \neq 0\} \tag{8.2}$$

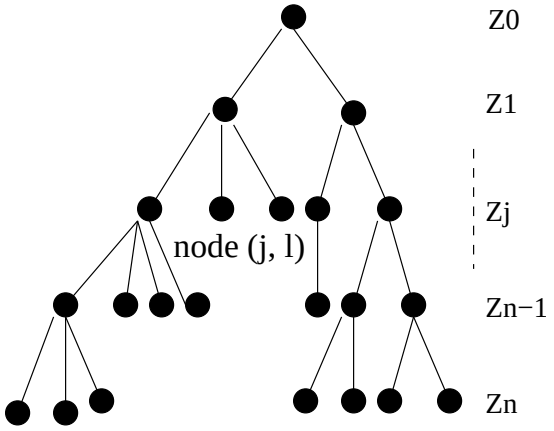


Fig. 8.1: arbre de Galton Watson

8.3 Développement asymptotique dans la cas uniforme

8.3.1 Comportement asymptotique de la moyenne

Theorem 2. *La proportion de noeuds découverts vérifie la relation suivante*

$$M_1(\lambda) \stackrel{def.}{=} \lim_{N\to+\infty} \frac{\mathbb{E}(R_N)}{\mathbb{E}(T_N)} = \sum_{n=0}^{+\infty} \frac{m-1}{m^{n+1}} \left(1 - \mathbb{E}\left[\exp\left(-\lambda \sum_{1}^{n} Z_i\right)\right]\right).$$

Si de plus la condition $\mathbb{E}(G \log G) < +\infty$ *est vérifiée, alors, quand* $\lambda \searrow 0$,

$$M_1(\lambda) \sim \frac{\lambda m}{m-1}\left(\log_m \lambda + \log_m\left(\frac{m}{m-1}\right) + o(\lambda)\right). \tag{8.3}$$

Grâce au terme $\log_m \lambda$, la relation (8.3) indique que l'exploration croît très rapidement quand peu de sondes sont déployées. En particulier, la dérivée de M_1 à l'origine est infinie.

8.3.2 La variance de $\mathbf{R}_N$

Notation.

Si (n,ℓ) et (n',ℓ') sont deux noeuds distincts de l'arbre, la relation $(n,\ell) < (n',\ell')$ est vérifiée si $\mathcal{T}_{N-n'}^{n',\ell'} \subset \mathcal{T}_{N-n}^{n,\ell}$, c'est à dire quand (n',ℓ') appartient au sous-arbre $\mathcal{T}_{N-n}^{n,\ell}$.

Proposition 29 (Comportement asymptotique de la variance)**.**

$$M_2(\lambda) \stackrel{def.}{=} \lim_{N\to+\infty} \frac{\mathrm{Var}(R_N)}{\mathbb{E}(T_N)} = \frac{m-1}{m}\sum_{n=1}^{+\infty}\frac{1}{m^n}\left[\mathbb{E}\left(e^{-\lambda T_n}\right)\left(1-\mathbb{E}\left(e^{-\lambda T_n}\right)\right)\right.$$
$$\left.+2\sum_{k=0}^{n-1}\mathbb{E}\left[e^{-\lambda T_{n-k-1}}\mathbb{E}\left(e^{-\lambda T_k}\right)^{Z_{n-k}-1}\mathbb{E}\left(e^{-\lambda T_k}\left(1-e^{-\lambda T_k}\right)\right)\right]\right] \quad (8.4)$$

Proposition 30 (Comportement asymptotique de $\lambda\to M_2(\lambda)$ en 0)**.** *Sous la condition* $\mathbb{E}(G\log G) < +\infty$*, on a,*

$$\lim_{\lambda\searrow 0}\frac{M_2(\lambda)}{\lambda\log_m\lambda} = \frac{m(m+1)}{(m-1)^2}. \quad (8.5)$$

8.4 Développement asymptotique dans la cas biaisé

On grade les mêmes notations que dans la section précédente. On suppose dans cette section que $\alpha \in [0,1)$ et, pour $n \geq 0$, le nombre de sondes au niveau n de l'arbre suit une distribution de Poisson de paramètre α^n et que ces sondes sont uniformément distribuées sur tous les noeuds de ce niveau. Par conséquent, le nombre moyen de sondes dans $\mathcal{T}^{n,\ell}$ est $(\alpha/m)^n/(1-\alpha)$ et dans tout l'arbre est $1/(1-\alpha)$. On note S_α le nombre total de noeuds découverts dans tout l'arbre. Comme dans la section précédente, on montre que,

$$\mathbb{E}\left(S_\alpha\right) = \sum_{n=0}^{+\infty}\mathbb{E}(Z_n)\mathbb{P}\left(\mathcal{N}(\mathcal{T}^{\ell,n}) \neq 0\right) = \sum_{n=0}^{+\infty} m^n\left(1-\exp\left(-\left(\frac{\alpha}{m}\right)^n\frac{1}{1-\alpha}\right)\right)$$

L'accroissement du processus d'exploration est donné par

$$\frac{\mathbb{E}\left(S_\alpha\right)}{\mathbb{E}(\mathcal{N}(\mathcal{T}))} = \sum_{n=0}^{+\infty}(1-\alpha)m^n\left(1-\exp\left(-\left(\frac{\alpha}{m}\right)^n\frac{1}{1-\alpha}\right)\right),$$

Proposition 31. *quand* $\alpha \nearrow 1$*, on a l'équivalent suivant*

$$\mathbb{E}\left(S_\alpha\right) \sim \frac{1}{(1-\alpha)}\int_0^{+\infty}\frac{m^{\{\log_{m/\alpha}[1/(u(1-\alpha))]\}}}{u^{\log m/\log(m/\alpha)}}e^{-u}\,du \quad (8.6)$$

ce qui nous permet d'écrire,

$$\frac{1}{m-1} \leq \liminf_{\alpha \nearrow 1} (1-\alpha)^2 \mathbb{E}\left(S_\alpha\right) \leq \limsup_{\alpha \nearrow 1} (1-\alpha)^2 \mathbb{E}\left(S_\alpha\right) \leq \frac{m}{m-1},$$

8.5 Conclusion

On a étendu les résultats obtenus pour les arbres déterministes aux arbres de Galton Watson. Nous avons établi des expressions analytiques pour la moyenne et la variance du nombre total de noeuds découverts. Nous avons développé ces expressions quand la taille du réseau est grande. En particulier, on a montré qu'au premier ordre, un arbre de Galton Watson de degré moyen m peut être remplacé par un arbre régulier de degré constant égal à d.

Bibliographie

[1] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1) :47–97, January 2002.

[2] Krishna B. Athreya and Peter E. Ney. *Branching processes*. Springer-Verlag, New York, 1972. Die Grundlehren der mathematischen Wissenschaften, Band 196.

[3] A. D. Barbour, Lars Holst, and Svante Janson. *Poisson approximation*. The Clarendon Press Oxford University Press, New York, 1992. Oxford Science Publications.

[4] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *j-CACM*, 13(7) :422–6, July 1970.

[5] A. Broder and M. Mitzenmacher. Network applications of bloom filters : A survey. *In Proc of Allerton Conference*, 2002.

[6] Robert C. Chalmers and Kevin C. Almeroth. On the topology of multicast trees. *IEEE/ACM Trans. Netw*, 11(1) :153–165, 2003.

[7] H. Chang, S. Jamin, and W. Willinger. Inferring as-level internet topology from router-level path traces. *In Proceeding of SPIE ITCom 2001, Denver, CO, August*, 2001.

[8] Baek-Young Choi, Jaesung Park, and Zhi-Li Zhang. Adaptive random sampling for load change detection. *SIGMETRICS '02 : Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 272–273, 2002.

[9] Saar Cohen and Yossi Matias. Spectral bloom filters. *SIGMOD '03 : Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 241–252, 2003.

[10] Louis Comtet. *Advanced combinatorics*. D. Reidel Publishing Co., Dordrecht, enlarged edition, 1974. The art of finite and infinite expansions.

[11] L. Dall'Astra, I. Alvarez-Hameli, A. Barrat, A. Vásquez, and A. Vespignani. A statistical approach to the traceroute exploration of networks : theory and simulations. Available at arXiv :cond-mat/0406404, June 2004.

[12] Nick Duffield, Carsten Lund, and Mikkel Thorup. Charging from sampled network usage. *IMW '01 : Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, pages 245–256, 2001.

[13] Nick Duffield, Carsten Lund, and Mikkel Thorup. Estimating flow distributions from sampled flow statistics. *SIGCOMM '03 : Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 325–336, 2003.

[14] Cristian Estan and George Varghese. New directions in traffic measurement and accounting. *SIGCOMM '02 : Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 323–336, 2002.

[15] Cristian Estan, George Varghese, and Mike Fisk. Bitmap algorithms for counting active flows on high speed links. *IMC '03 : Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 153–166, 2003.

[16] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. *SIGCOMM*, pages 251–262, 1999.

[17] Li Fan, Pei Cao, Jussara Almeida, and Andrei Z. Broder. Summary cache : a scalable wide-area Web cache sharing protocol. *IEEE/ACM Transactions on Networking*, 8(3) :281–293, 2000.

[18] W. Fang and L. Peterson. Inter-AS Traffic Patterns and Their Implications. *IEEE Globecom*, December 1999. Brazil.

[19] Flajolet. On adaptive sampling. *COMPUTG : Computing (Archive for Informatics and Numerical Computation), Springer-Verlag*, 43, 1990.

[20] Philippe Flajolet, Danièle Gardy, and Loys Thimonier. Birthday paradox, coupon collectors, caching algorithms and self-organizing search. *Discrete Appl. Math.*, 39(3) :207–229, 1992.

[21] Philippe Flajolet and G. Nigel Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, 31(2) :182–209, 1985.

[22] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang. Idmaps : A global internet host distance estimation service. *IEEE/ACM Transactions on Networking*, 9(5) :525–540, October 2001.

[23] G. H. Gonnet and R. Baeza-Yates. *Handbook of algorithms and data structures.* Addison-Wesley, Boston, MA, USA, 1991.

[24] Ramesh Govindan and Anoop Reddy. An analysis of internet inter-domain topology and route stability. *INFOCOM (2)*, pages 850–857, 1997.

[25] Ramesh Govindan and Hongsuda Tangmunarunkit. Heuristics for internet map discovery. *IEEE INFOCOM 2000*, pages 1371–1380, March 2000.

[26] Chris A. Klaassen. Dixie cups : sampling with replacement from a finite population. *Journal of Applied Probability*, 31(4) :940–948, 1994.

[27] Abhishek Kumar, Minho Sung, Jun (Jim) Xu, and Jia Wang. Data streaming algorithms for efficient and accurate estimation of flow size distribution. *SIGMETRICS 2004/PERFORMANCE 2004 : Proceedings of the joint international conference on Measurement and modeling of computer systems*, pages 177–188, 2004.

[28] Abhishek Kumar, Jun (Jim) Xu, Li Li, and Jia Wang. Space-code bloom filter for efficient traffic flow measurement. *IMC '03 : Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 167–172, 2003.

[29] Anukool Lakhina, John Byers, Mark Crovella, and Peng Xie. Sampling biases in ip topology measurements. *IEEE Infocom*, 2003.

[30] Will E. Leland, Murad S. Taqq, Walter Willinger, and Daniel V. Wilson. On the self-similar nature of Ethernet traffic. *ACM SIGCOMM*, pages 183–193, 1993.

[31] Russel Lyons and Yuval Peres. *Probability on Trees and Networks.* Preprint, 2005.

[32] Ratul Mahajan, Steven M. Bellovin, Sally Floyd, John Ioannidis, Vern Paxson, and Scott Shenker. Controlling high bandwidth aggregates in the network. *SIGCOMM Comput. Commun. Rev.*, 32(3) :62–73, 2002.

[33] M. D. McIlroy. Development of a spelling list. *IEEE Trans. on Communications*, 30 :91–99, 1982.

[34] Tatsuya Mori, Masato Uchida, Ryoichi Kawahara, Jianping Pan, and Shigeki Goto. Identifying elephant flows through periodically sampled packets. *IMC '04 : Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 115–120, 2004.

[35] M.E.J. Newman and D.J. Watts. Random graph models of social networks. *Proc. Nat. Acad. Sci USA*, 99 :2566–2572, 2002.

[36] J. Pansiot and D. Grad. On routes and multicast trees in the internet. *ACM Computer Communication Review, vol. 28, no. 1, pp. 41–50, Jan*, 1998.

[37] Vern Paxson and Sally Floyd. Wide area traffic : the failure of Poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3) :226–244, 1995.

[38] P. Radoslavov, H. Tangmunarunkit, R. Govindan H. Yu, S. Shenker, and D. Estrin. On characterizing network topologies and analyzing their impact on protocol design. Technical Report 03-782, Univ. South. Cal., 2001.

[39] S. C. Rhea and J. Kubiatowicz. Probabilistic location and routing. *Proc of Infocom-02*, June 2002.

[40] J. Riordan. *An introduction to combinatorial analysis*. John Wiley and Sons, 1958.

[41] Philippe Robert. *Réseaux et files d'attente : méthodes probabilistes*, volume 35 of *Mathématiques et Applications*. Springer-Verlag, Berlin, Octobre 2000.

[42] Philippe Robert. On the asymptotic behavior of some algorithms. *Random Structures and Algorithms*, 27(2) :235–250, September 2005.

[43] Yuval Shavitt and Tomer Tankel. On the curvature of the internet and its usage for overlay construction and distance estimation. *IEEE INFOCOM*, 2004.

[44] Haoyu Song, Sarang Dharmapurikar, Jonathan Turner, and John Lockwood. Fast hash table lookup using extended bloom filter : an aid to network processing. *SIGCOMM '05 : Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 181–192, 2005.

[45] N. Spring, R. Mahajan, and D. Wetherall. Measuring isp topologies with rocketfuel. *Proceedings of ACM/SIGCOMM '02*, August 2002.

[46] Hongsuda Tangmunarunkit, Ramesh Govindan, Scott Shenker, and Deborah Estrin. The impact of routing policy on internet paths. *INFOCOM*, pages 736–742, 2001.

[47] A. Whitaker and D. Wetherall. Forwarding without loops in icarus. *In Proceedings of the Fifth OPENARCH*, pages 63–75, June 2002.

[48] Herbert S. Wilf. *generatingfunctionology*. Academic Press Inc., Boston, MA, 1990.

[49] Walter Willinger, Murad S. Taqqu, Robert Sherman, and Daniel V. Wilson. Self-similarity through high-variability : statistical analysis of Ethernet LAN traffic at the source level. *IEEE/ACM Transactions on Networking*, 5(1) :71–86, 1997.

Résumé

La thématique de cette thèse porte sur l'analyse de grands réseaux comme celui d'Internet. Dans la première partie de la thèse, on s'est intéressé à la caractérisation de la topologie Internet et à l'étude de l'efficacité de traceroute. On a considéré plusieurs architectures d'arbres déterministes (Réguliers, PLI et PLD) et aussi aléatoires de type Galton-Watson. On a étudié la moyenne du nombre de routeurs découverts et aussi la variance de cette quantité pour déduire l'effet de l'emplacement aléatoire des sondes. Dans un second temps, on a développé des asymptotiques quand le réseau devient très grand. Ces développements permettent de mieux comprendre l'évolution des propriétés quantitatives du réseau lors du passage à l'échelle. Nous nous somme intéressés aussi au cas où le nombre de sondes est insuffisant afin de mieux comprendre la vitesse initiale de découverte du réseau. La deuxième partie de la thèse porte sur la détection des grands flots. On a développé deux algorithmes temps réel et à mémoire optimale, le premier se base sur les filtres de Bloom et introduit un mécanisme adaptatif qui permet de libérer régulièrement la mémoire. Le second algorithme s'appuie sur le constat que les éléphants arrivent en rafale et permet donc de déceler les éventuels flots éléphants. Nous nous somme aussi intéressés au lien entre échantillonnage et queue de distribution lourde. Ce travail permet entre autres de déduire les paramètre des distributions originales des tailles des flots.

Mots clés

Topologie, architecture en arbre, arbre de Galton Watson, Filtres de Bloom, éléphants, échantillonnage

Abstract

The purpose of this thesis is the study of large networks such as the Internet. The first part of this thesis is related to Internet Topology discovery by means of traceroute. We considered several deterministic and random trees. We studied the mean and variance of the proportion of discovered nodes. Asymptotic expansions are derived in order to understand the behavior when the network size is large and also when the number of traceroute hosts is small. In the second part, we developed two real time algorithms to detect large flows. The first one is based on Bloom filters and uses a cleaning mechanism to free the memory. The second one exploits the elephants burstiness. We studied also the sampling relationship with heavy tail property.

Keywords

Topology, tree architecture, Galton-Watson trees, Bloom filter, elephants, sampling

INRIA Rocquencourt-Domaine de Voluceau BP105-78153 Le chesnay Cedex