UNIVERSITE DE NICE-SOPHIA ANTIPOLIS - UFR SCIENCES

Ecole Doctorale STIC

THESE

Présentée pour obtenir le titre de

Habilitation à Diriger des Recherches

de l'Université de Nice Sophia Antipolis

Spécialité : Informatique

par

Chadi BARAKAT

Solutions efficaces pour la métrologie de l'Internet

Soutenue publiquement le 22 Janvier 2009 devant le jury composé de :

M.	Michel	Riveill	UNSA/CNRS/I3S	Président
M.	Walid	Dabbous	INRIA	Tuteur
M.	Nazim	Agoulmine	U. Evry	Rapporteur
M.	Guy	Leduc	U. Liège	Rapporteur
M.	Patrick	Senac	ISAE	Rapporteur
M.	Guillaume	Urvoy-Keller	Eurecom	Examinateur

(10 :30 - INRIA)

Remerciements

Cette thèse est le fruit d'un long travail qui a eu lieu principalement à l'INRIA Sophia Antipolis au sein de l'équipe Planète et lors de mes séjours à l'EPFL Lausanne et à Intel Research Cambridge. Mes premiers remerciements ne peuvent aller qu'à mes collègues dans ces trois laboratoires et aux équipes de support administratif et informatique qui m'ont accompagné. Sans eux cette thèse n'aurait pas vu les jours. J'exprime aussi ma gratitude à mes étudiants qui ont été tout au long des ces dernières années mon premier support et ma grande joie.

Je remercie chaleureusement les membres de mon jury qui ont accepté de rapporter sur cette thèse et de se déplacer pour la présentation. Leur venue à Sophia Antipolis est un grand honneur pour moi.

Finalement, c'est à ma femme Joumana que je dédie ce travail. Sa présence à mes côtés et ses encouragements ont toujours été ma première source d'inspiration.

Table des matières

Chapitre 1

Introduction

1.1 Généralités sur l'Internet

L'Internet a été créé comme un réseau simple, ouvert, flexible, où le seul service offert aux utilisateurs est le "routage au mieux" des paquets (en anglais Best Effort). Ce choix a été délibéré pour la simple raison de faciliter l'interconnexion des réseaux, la création des applications et l'ajout des services à l'Internet. Un des principes fondamentaux de l'Internet est que le réseau doit rester le plus simple possible et que l'intelligence doit se trouver dans les machines des utilisateurs qui effectuent elles les opérations classiques de type recouvrement des erreurs, adaptation de débit, localisation des ressources, et ainsi de suite. C'est le fameux principe de bout-en-bout de l'Internet (en anglais end-to-end). Ce principe permet à toute application de l'Internet désirant communiquer, d'envoyer des paquets à n'importe quelle adresse IP et n'importe quel numéro de port. Le réseau fait ensuite de son mieux pour acheminer ce paquet à la bonne destination. Si la destination n'est pas trouvée, la source est notifiée, sinon le paquet est livré pourvu qu'il n'a pas été rejeté ou corrompu le long du chemin. Tout ceci se fait sans le besoin de réserver des ressources dans le réseau ou de demander l'autorisation de certaines entités administratives comme ce qui se fait dans les réseaux téléphoniques. Cette possibilité de communiquer en temps réel avec plusieurs entités réseaux distribuées partout dans le monde a été un facteur important pour la création d'intéressants services comme le WEB, le mail, le Peer-to-Peer, les overlays, et ainsi de suite. Et c'est principalement à elle que revient le grand succès qu'a connu l'Internet.

Malheureusement, rien n'est gratuit dans la vie. L'ouverture du réseau au trafic complique la tâche de l'administration de l'Internet et de son ingénierie et conduit à des comportements néfastes à la fois pour les utilisateurs et pour les opérateurs réseaux. Un utilisateur de l'Internet peut envoyer autant de paquets que son interface réseau et la vitesse de sa machine le permettent. Une partie de ce trafic est désirable, mais le reste peut consister en un trafic complètement indésirable comme les attaques qui visent à faire tomber des serveurs ou bien les emails du type SPAM qui font déborder les boîtes aux lettres. Il n'y a aucun mécanisme réseau qui empêche les utilisateurs de l'Internet d'envoyer du trafic indésirable ! Ils ont tous la possibilité d'utiliser au maximum leurs bandes passantes disponibles, et les routeurs du réseau font de leur mieux

pour relayer leurs trafics. Les opérateurs réseaux de leur côté voient un flux de paquets arrivant à chacun de leurs routeurs et tâchent de le relayer à travers leur réseau aux réseaux voisins. Ils exercent aucun contrôle à priori sur ce trafic et ne peuvent rien apprendre des observations faites par l'un et l'autre. Un opérateur réseau doit agir tout seul en se basant sur ce qu'il observe dans son réseau, et le maximum qu'il peut faire pour limiter le dégât en cas de comportement anormal (e.g, forte et subite croissance du trafic) est de le bloquer à la périphérie de son réseau. Mais pour le bloquer il faut pouvoir le mesurer, le comprendre, et l'isoler, des tâches qui peuvent être faciles dans un petit réseau, mais quand il s'agit d'un grand réseau de transit où des centaines de Mbps circulent, ces tâches deviennent difficiles à réaliser voire impossibles. Actuellement, les opérateurs du coeur de l'Internet se contentent la plupart de temps d'un simple forwarding des paquets aux réseaux voisins et laissent la tâche de détection et blocage du trafic indésirable aux réseaux destinataires. Localement, des décisions simples sont prises comme le choix des sorties qui permettent de mieux répartir le trafic, et le reroutage d'une partie du trafic en cas de surcharge ou de rupture d'un des liens.

Les conséquences naturelles de l'absence du contrôle fin dans le coeur du réseau et de la collaboration entre les opérateurs (autre que l'échange d'informations sur le routage) sont :

- La prolifération des attaques dans l'Internet (par exemple celles de type DDoS qui visent à tomber un serveur, ainsi que les vers et les virus) et du trafic indésirable (le mail SPAM) que seules les machines et réseaux destinataires peuvent traiter et éliminer. Ce trafic indésirable est chiffré de nos jours à une dizaine de pourcents du volume total du trafic Internet avec un effet néfaste pouvant dépasser le ratio de ce trafic.
- Comme les attaques et le trafic indésirable ne sont pas bloqués à leurs sources suite à l'absence de collaboration entre les opérateurs, les paquets correspondant traversent le réseau pour être rejetés à la fin, causant ainsi une perte de ressources et une dégradation des services.
- L'absence de collaboration entre les opérateurs (autre que l'échange des informations sur le routage) rend difficile le diagnostic des problèmes réseaux. Un opérateur peut à la limite et s'il dispose de méthodes efficaces connaître ce qui se passe dans son réseau, mais il a aucun moyen de savoir ce qui se passe dans les autres réseaux ni la provenance des problèmes. Les exemples typiques sont la dégradation des performances suite à l'augmentation du délai et la chute de la bande passante dont l'opérateur réseau ne peut pas connaître l'origine exacte en dehors de son réseau (C'est déjà une tâche difficile à réaliser dans son réseau). Ce problème de manque de vision globale s'applique bien naturellement aux utilisateurs qui eux aussi n'ont pas la possibilité de connaître l'origine des problèmes dans l'Internet.
- La prolifération des Overlays (réseaux de recouvrement) qui cherchent à résoudre les problèmes de l'Internet et à combler le manque en services fournis. RON [10] est un exemple typique d'un réseau Overlay qui redirige le trafic à travers des proxys afin de contourner les problèmes de routage dans l'Internet. Les systèmes de coordonnées comme Vivaldi [34]

ont été proposés pour mettre à la disposition des applications et des entités réseaux un moyen pour se localiser topologiquement les unes par rapport aux autres. Les plateformes de mesure comme Skitter de CAIDA [31] sont venues combler le manque de vision globale et collaboration entre opérateurs. Elles permettent avec des mesures de bout en bout et des informations sur les tables de routage collectées en certains routeurs ouverts de l'Internet de donner à tout le monde une idée de comment le réseau se porte à un instant donné en termes de performances et topologie.

1.2 La métrologie et ses intérêts

Les opérateurs de l'Internet et les utilisateurs se trouvent alors à eux tous seuls pour s'adapter aux variations et aux anomalies de ce grand réseau qui s'appelle l'Internet où les seuls services qui existent sont le routage paquet par paquet et le contrôle de congestion de bout en bout fourni par TCP que malheureusement beaucoup d'utilisateurs peuvent arrêter pour les besoins de leurs applications. Tout le reste est inconnu et à découvrir : la topologie des réseaux et la manière avec laquelle ils sont connectés, les caractéristiques des chemins et liens de l'Internet, les politiques de routage adoptées par les autres opérateurs réseaux, les applications que les utilisateurs utilisent, l'intention des utilisateurs, le comportement des utilisateurs, l'origine des problèmes réseaux, et ainsi de suite. Tout ceci ne peut pas se faire sans une métrologie du réseau que ca soit par les opérateurs ou par les utilisateurs. De nos jours, la pratique la plus utilisée est que chacun procède à sa manière pour comprendre au mieux ce qui se passe. Un opérateur mesure son réseau et essaie de comprendre le mieux possible sur ce qui se passe dedans et sur ce qui arrive de l'extérieur. Les utilisateurs effectuent à leur tour des mesures de bout en bout pour comprendre les caractéristiques des différents réseaux de l'Internet dans la limite du possible (un opérateur peut très bien bloquer le trafic de mesure d'un utilisateur final). Et finalement, il y a la communauté scientifique qui s'efforce d'effectuer des opérations de mesure à grande échelle (comme Skitter de CAIDA, AMP de NLANR, IPMON de Sprintlabs, et des projets de recherche comme Metropolis, ETOMIC, et MOME) et de mettre les résultats et les observations effectuées à la disponibilité des utilisateurs et des opérateurs.

Parmi les intérêts de la métrologie des réseaux, on peut citer les suivants :

- La métrologie sert à détecter les problèmes de fonctionnement (par exemple les attaques, les ruptures des liens, les baisses de débit) et par conséquence à y remédier dans la mesure du possible.
- La métrologie sert à mieux comprendre l'Internet (e.g., son trafic, sa topologie) et à proposer des modèles théoriques qui permettent de prédire son évolution et de reproduire les résultats dans des simulations et des plateformes expérimentales.
- La métrologie sert à caractériser les applications et les comportements des utilisateurs. Ceci permet d'évaluer l'impact de toute nouvelle application (e.g., le Pair-à-Pair) et de

tout nouveau comportement (e.g., le VPN) sur le réseau et son trafic.
- La métrologie permet aux opérateurs de mieux dimensionner leurs réseaux et les administrer (connectivité, ingénierie de trafic, filtrage, facturation) et aux utilisateurs de mieux concevoir leurs applications (les rendre par exemple adaptatives à la topologie du réseau et ses performances).

1.3 Les difficultés de la métrologie

Il est évident qu'en l'absence d'un plan de contrôle dans l'Internet qui permet aux utilisateurs et aux opérateurs d'agir ensemble pour mieux opérer le réseau, des solutions pour la métrologie sont à développer pour réaliser les fonctions citées ci-dessus. Ces solutions doivent se baser sur le peu de fonctions disponibles dans le coeur du réseau comme les politiques de scheduling FIFO dans les routeurs, les messages ICMP [93] (probablement) renvoyés par les routeurs et machines en cas de rejet de paquets, et les tables de routage (surtout les tables BGP [97]). D'un côté la métrologie peut se faire en collectant le trafic sur les liens de l'Internet et en l'analysant après. Ceci est souvent appelé l'approche passive pour la métrologie (pas d'injection de trafic). De l'autre côté, la métrologie peut se faire en envoyant des paquets et en mesurant la réaction du réseau à ces paquets. Ceci est souvent appelée l'approche active dans la métrologie. Les paquets peuvent être dédiés à la mesure comme ils peuvent être des paquets de données. La mesure elle même se fait de bout en bout entre des machines ordinaires ou des sondes dédiées. Et que ce soit dans le cas passif ou actif, une fois les mesures effectuées, il faut passer par une autre phase qui est celle de l'analyse des résultats et de l'inférence de ce qui se passe dans le réseau. Dans le cas passif, il s'agit de comprendre ce qui se passe au point de mesure et ailleurs. Dans le cas actif, il s'agit de comprendre ce qui se passe tout au long du chemin traversé par les paquets de mesure. Ce qu'on mesure de bout en bout n'est que l'accumulation de tout ce que rencontrent les paquets dans les différents liens et routeurs du réseau. Les mesures peuvent être effectuées en plusieurs points ensuite corrélées ensemble pour obtenir plus de résultats. Elles peuvent aussi être reproduites dans le temps périodiquement ou selon un processus stochastique pour permettre une étude de l'évolution temporelle du réseau et la détection de tout changement dans le comportement et les performances.

L'intérêt de la métrologie est clair mais les difficultés qui l'accompagnent sont nombreuses. Ces difficultés proviennent principalement de la grande taille de l'Internet et de son ouverture, et du grand volume de trafic qui y circule. Les conséquences de celles-ci sur les deux approches de métrologie sont différentes mais elles se rejoignent sur la difficulté de la tâche :

- Pour les mesures actives, un grand réseau veut simplement dire une phase d'inférence compliquée puisque divers phénomènes peuvent apparaître tout le long du chemin suivi par les paquets. Pour chaque métrique (e.g., taux de pertes, bande passante), il devient alors important de bien choisir la manière avec laquelle les paquets sont envoyés afin d'obtenir

le meilleur résultat possible. Malheureusement cette optimisation n'est pas souvent facile à faire. Ce qui est bon sur certains chemins peut s'avérer mauvais sur d'autres chemins. Un grand réseau veut aussi dire un grand volume de paquets sondes à envoyer et plus de machines à faire intervenir pour atteindre une bonne couverture du réseau.

- Pour les mesures passives, le problème est principalement dans le grand volume du trafic à collecter sur les liens du réseau pour avoir une vision claire de ce qui se passe. Et si on duplique ces points de mesure passive, le volume de trafic à collecter augmente aussi. Sur le lien d'accès d'une machine, quelqu'un peut se contenter de collecter tout le trafic et de l'analyser. Le volume de ce trafic va rester dans la limite du raisonnable. Malheureusement sur des liens réseaux bien chargés (e.g., à des centaines de Mbps par exemple), ceci devient impossible. Il s'agit alors de trouver des algorithmes d'agrégation du trafic qui réduisent le volume de données à remonter après à l'unité centrale où le traitement sera effectué. Clairement ces algorithmes sont dépendants de l'application ; un algorithme d'agrégation général est celui qui permet aux données agrégées d'être utilisées par le plus grand nombre d'applications (l'agrégation est une opération irréversible ! !). Et ceci n'est pas le seul problème. Parce que même si un algorithme de compression efficace est trouvé, une unité de mesure passive peut être dans l'incapacité de traiter tous les paquets en temps réel et d'accéder à la mémoire pour chacun de ces paquets afin de mettre à jour les champs et registres correspondants. Il s'agit alors de trouver des algorithmes qui permettent de construire une image du trafic à partir d'un sous-ensemble de paquets, ce qui nous ramène à tout un large domaine qui est celui de l'échantillonnage du trafic et de son impact sur la métrologie.
- Le tout naturel problème de l'interprétation des résultats qui provient de la complexité de l'Internet et du grand nombre de facteurs qui décident de ce qui se passe. Certaines métriques sont faciles à calculer comme par exemple le volume du trafic, mais d'autres comme le nombre des attaques ou la bande passante disponible sont beaucoup plus difficiles à inférer. Ces dernières métriques requièrent un certain modèle pour l'Internet (e.g., modèle pour le trafic, modèle pour les chemins) ce qui est malheureusement difficile à obtenir.
- Les mécanismes que les utilisateurs et opérateurs mettent en place pour se protéger les uns des autres. Ces mécanismes rendent difficile la métrologie de l'Internet voire impossible. Comme exemple de ces mécanismes on trouve le cryptage du trafic et la désactivation de l'envoi des messages de contrôle ICMP que certains opérateurs pratiquent. La solution à ces problèmes passent souvent par des méthodes statistiques et des heuristiques qui essaient d'approcher la réalité en partant du peu qui a pu être mesuré.

Les difficultés dans la métrologie se résument alors aux points suivants :

- Le placement des points de mesure, appelé souvent échantillonnage spatial. Pour un opérateur, ceci revient à choisir les points de son réseau où les mesures sont à effectuer. Pour les utilisateurs, ceci revient à trouver les machines vers lesquelles envoyer les paquets de mesure et les machines sources. Malheureusement et ce qui est le cas actuellement, les opé-

rateurs et utilisateurs ne collaborent pas pour la réalisation de cette tâche ni pour l'échange des résultats obtenus par chacun d'entre eux.

- L'agrégation du trafic pour réduire le volume du trafic à traiter. Cette agrégation doit être faite d'une manière intelligente afin de préserver le maximum d'informations qui peuvent servir au maximum d'applications.
- L'échantillonnage (temporel pour le distinguer du spatial) qui permet de réduire le volume du trafic à collecter. Tout comme l'agrégation, l'échantillonnage ne doit pas trop induire en erreur les résultats. Pour chaque application de mesure, une étude est alors à mener pour trouver l'impact de l'échantillonnage et identifier la meilleure méthode pour inférer le trafic orignal à partir du trafic échantillonné. Cette dernière procédure est appelée *inversion*.
- La collecte des résultats de mesure doit se faire le plus rapidement possible sans congestionner le réseau. La mesure peut être la collecte du trafic sur un lien comme elle peut être une requête envoyée à une entité réseau (ou une machine distante) pour obtenir certaines informations, par exemple le débit sur le lien ou l'adresse IP de l'entité. Une des caractéristiques de cette collecte est le grand nombre d'entités réseaux qui peuvent intervenir. Une autre caractéristique c'est la fiabilité qui peut varier entre une entière fiabilité (tout collecter) et une fiabilité partielle (cas d'une information agrégée comme par exemple le débit moyen sur les liens, le nombre de flots sur un lien, le nombre de paquets sur un lien, le nombre de machines connectées à un réseau, et ainsi de suite).
- La manière avec laquelle envoyer les paquets sondes pour caractériser un chemin du réseau. Le grand nombre de facteurs qui décident ce qui se passe sur un chemin (pensez à tous les paquets qui passent par ce chemin et leur provenance) rend cette caractérisation difficile voire impossible. Chaque métrique nécessite une manière spécifique pour envoyer les sondes. Ce qu'il faut faire pour mesurer le délai moyen est différent de ce qu'il faut faire pour mesurer la bande passante disponible ou la bande passante du lien le plus lent tout au long du chemin.
- Des solutions pour contourner les barrières mises en place par les utilisateurs et les opérateurs de l'Internet afin de se protéger les uns des autres. Une méthode de mesure efficace doit être capable de fonctionner dans tous les scénarios. Une solution par exemple qui fonctionne que sur du trafic non-crypté ou qui suppose que les routeurs du réseau coopèrent en renvoyant des messages ICMP est moins efficace qu'une autre solution n'exigeant pas ces conditions même si la dernière est un peu moins efficace en termes de performance.

1.4 Le passage à l'échelle de la métrologie

Malgré tous les travaux effectués dans la littérature, dont un aperçu général sera présenté dans le chapitre suivant, beaucoup des points durs dans la métrologie restent sans solution claire et très probablement vont le rester vue la complexité de l'Internet et son constante évolution. Parmi

ces problèmes figure principalement le passage à l'échelle de la métrologie. En effet, certains des points durs cités dans la section précédente nécessitent souvent des solutions protocolaires qui ne souffrent pas si la taille du réseau augmente. On trouve dans cette catégorie par exemple les solutions statistiques qui classent le trafic crypté de l'Internet et celles qui cherchent la meilleure manière avec laquelle envoyer les paquets pour mesurer une certaine métrique relative à un chemin de l'Internet. Ce genre de problèmes n'est pas un handicap en cas d'augmentation de la taille du réseau. Par contre les autres points durs dans la métrologie ont leur difficulté directement liée à la taille du réseau ce qui rend difficile le passage à l'échelle. L'exemple typique est celui de l'observation de ce qui se passe dans un grand réseau que ce soit par des mesures passives ou des mesures actives. Ce genre de problèmes nécessite beaucoup d'effort de la part de chercheurs pour arriver à des solutions qui passent à l'échelle sans trop impacter la précision des mesures.

Dans ce manuscrit, nous focalisons sur les aspects de métrologie qui souffrent de problèmes de passage à l'échelle. Nous avons aussi mené des activités sur l'autre catégorie de problèmes, mais elles ne seront pas présentées dans ce manuscrit par souci d'homogénéité de la présentation. Nous regroupons ces aspects de la métrologie en deux catégories :

- D'un côté se trouve le problème du grand volume de données à collecter pour réaliser les mesures. Ceci va du grand volume du trafic à la grande taille du réseau. Nos contributions dans cette direction visent à réduire le volume de données à traiter par l'intermédiaire de l'échantillonnage à la fois spatial et temporel. En parallèle, une étude sera menée pour évaluer l'impact de l'échantillonnage sur la qualité de la mesure.
- De l'autre côté se trouve le problème du grand volume de données à envoyer sur le réseau. Ces données peuvent être les résultats de mesure passive comme ils peuvent être les paquets sondes et requêtes envoyés à des entités réseaux et les machines des utilisateurs. Ce problème a été souvent ignoré dans la littérature. Nous soulevons ce problème dans ce manuscrit et nous proposons une solution protocolaire efficace qui permet d'envoyer un grand volume de données à partir d'un grand nombre de sources sans congestionner le réseau.

A chacun de ces problèmes nous présentons des solutions et nous les validons par modélisation analytique, simulations et expérimentations. Nos solutions, par leur efficacité et leur simplicité, présentent des réponses à certains des points durs de la métrologie et constituent par conséquence une intéressante avancé dans ce domaine. Elles ont la caractéristique d'apporter une nouvelle vision sur certains des aspects de la métrologie et de fonctionner en cas de grands réseaux et de grandes quantités de données à traiter.

Les chapitres suivants présentent une description de nos contributions. Chapitre 3 traite le problème de la métrologie passive des grands réseaux par l'échantillonnage à la fois spatial et temporel. L'échantillonnage n'est pas gratuit et a un coût. Chapitre 4 prend une métrique qui est celle de la détection des utilisateurs les plus gourmands en termes de trafic et montre l'impact de l'échantillonnage dessus. Il s'agit d'un exemple d'une étude que nous devons mener chaque fois qu'il s'agit d'échantillonnage. Chapitre 5 fait une étape supplémentaire en montrant sur un

autre exemple comment on peut réduire le bruit introduit par l'échantillonnage en considérant la corrélation temporelle dans les variations de la taille de la population échantillonnée. Ceci permet bien évidemment des taux d'échantillonnage encore plus réduits. Chapitre 6 traite le problème de la métrologie active et de collecte d'informations. Il présente notre solution TICP (TCP-friendly Information Collection Protocol) qui permet d'envoyer des données et des requêtes dans un grand réseau sans le congestionner et sans nuire au reste du trafic (notamment le trafic TCP d'où le terme TCP-friendly dans le titre).

Avant de commencer à décrire nos solutions et pour mieux les motiver, nos résumons dans le chapitre suivant l'évolution de la métrologie de l'Internet et les principaux travaux dans la littérature. Le manuscrit sera conclu dans le Chapitre 7 par notre point de vue sur le domaine et son évolution ainsi que les pistes que nous comptons suivre par la suite.

Chapitre 2

L'évolution du domaine de la métrologie

L'Internet a été créé sans infrastructure de mesure standard qui permet aux opérateurs et aux utilisateurs d'observer ce qui se passe et d'échanger les résultats entre eux. Comme dit dans le chapitre précédent, le coeur de l'Internet fournit un service simple consistant en un routage au mieux des paquets, et toute l'intelligence est repoussée à la périphérie du réseau (dans les machines). Personne à un moment donné n'a une idée de comment est formé l'Internet ni quelles sont les communications en cours. Une machine peut envoyer et recevoir des paquets autant que la bande passante le permet, et les opérateurs relaient les paquets en provenance des autres opérateurs la plupart de temps suivant des politiques locales qui ne tiennent pas compte des performances globales. Par conséquence, chaque opérateur et chaque utilisateur (ou groupes d'utilisateurs) déploient leurs propres solutions de mesure afin d'optimiser le réseau de leur point de vue.

Néanmoins, certains mécanismes ont accompagné l'Internet depuis sa création et ont formé une bonne base de départ pour la métrologie. Nous commençons par décrire ces mécanismes, après nous présentons les solutions qui ont apparu depuis en les classant en fonction de leur nature : *passive* ou *active*. Nous rappelons qu'une solution passive est une solution qui n'injecte pas de trafic dans le réseau mais seulement se contente d'observer le trafic qui circule sur un ou plusieurs liens de l'Internet. A l'opposé, se trouvent les solutions actives qui consistent à envoyer des paquets dans le réseau dédiés à la mesure et à l'observation de la réaction du réseau à ces paquets sondes. La réaction du réseau à un paquet se manifeste par son temps de livraison, égal à l'infini quand le paquet est rejeté.

2.1 Les mécanismes de base de mesure

Malgré la simplicité de la couche réseau de l'Internet, certains mécanismes ont été introduits pour assurer un minimum de fonctions. A la tête de ces mécanismes se trouvent les messages ICMP (Internet Control Message Protocol) [93] envoyés par les routeurs de l'Internet et ses

machines quand des paquets reçus présentent des anomalies. En effet, quand un paquet arrivant à la couche IP d'une machine ou d'un routeur présente une anomalie, un message ICMP est envoyé (ou plutôt devrait l'être) à la source du message, bien évidemment si l'entête IP du message est jugé ne pas contenir des erreurs, parce que sinon le message ICMP risque d'être envoyé à la mauvaise source. Ces messages ICMP changent en fonction de la nature de l'anomalie. Un code est associé à chaque anomalie et est inclus dans l'entête du message ICMP. En lisant ce code, la source réceptrice du message ICMP est au courant du problème qui, on le rappelle, concerne seulement le paquet correspondant. Et pour davantage d'informations à la source, une partie du paquet anormal est copiée dans le message ICMP.

Le protocole ICMP est le seul moyen qui permet d'avoir un retour de la part des routeurs de l'Internet. Si l'adresse IP du routeur est connue, ICMP permet de l'interroger directement par un message de type *echo* pour avoir un retour de type *echo reply*. De cette façon, un administrateur ou un utilisateur peuvent savoir si le routeur est bien opérationnel ou pas. Cette opération peut aussi se faire avec toute machine et serveur de l'Internet implémentant le protocole ICMP. Maintenant, si l'adresse IP du routeur n'est pas connue, le seul moyen qui reste pour avoir un retour c'est d'envoyer un paquet et de faire de la sorte que le paquet soit anormal en arrivant au routeur, et c'est là que le retour répond pour signaler l'anomalie. Dans sa réponse le routeur met l'adresse IP de son interface de sortie.

Les anomalies apparaissent souvent quand le routeur se trouve obligé de rejeter le paquet. C'est en ce moment qu'il notifie la source de ce rejet. Et les raisons pour rejeter un paquet sont diverses, comme par exemple, destination inconnue, le champ TTL a atteint zéro, et le paquet est trop grand pour que le routeur puisse le transmettre. Une règle de base dans ICMP c'est qu'un message ICMP rejeté par un routeur ne génère pas un autre message ICMP afin d'éviter une avalanche de messages ICMP. Les routeurs relaient les messages ICMP les un des autres comme des paquets IP réguliers de telle sorte qu'un feedback est possible à obtenir de tout routeur et machine de l'Internet. Malheureusement dans la pratique les choses ne sont pas si joyeuses que ca [61]. Certains routeurs de l'Internet désactivent complètement la fonctionnalité ICMP. D'autres limitent le débit des messages ICMP. Et il y a certains routeurs qui bloquent les messages ICMP générés par les autres routeurs. Tout ceci complique les tâches de mesure basées sur les mécanismes de ICMP.

ICMP a inspiré plusieurs chercheurs à développer des outils pour le diagnostic et la métrologie de l'Internet. Ces outils permettant pour la plupart de temps le diagnostic d'un chemin entre deux machines ou entre une machine et un routeur, ont été après utilisés pour une métrologie à grande échelle de l'Internet. Parmi les outils à base de ICMP on trouve (des variantes à ces outils existent sous d'autres appellations) :

- *ping* utilise les messages ICMP *echo* et *echo reply* pour vérifier si une adresse IP est active ou pas.
- *traceroute* envoie des paquets UDP avec un TTL limité et attend le retour ICMP du routeur

qui rejette le message quand son TTL expire. Pour un TTL égal à n au départ et pour un n plus petit que le nombre de sauts le long du chemin, *traceroute* collecte l'adresse IP du routeur qui se trouve au $n$ème saut avec le délai aller-retour entre la source et ce routeur. En commençant de 1 et en augmentant n de 1 à chaque fois, *traceroute* peut de cette manière collecter les adresses IP de tous les routeurs le long d'un chemin ainsi que les délais aller-retour cumulatifs. *traceroute* arrête d'augmenter n quand un retour de la machine destinatrice est noté et ceci est rendu possible par l'utilisation d'un numéro de port sur la machine qui n'est pas alloué ce qui provoque un retour de type *port unreachable*. L'idée de *traceroute* de contrôler le TTL a été ensuite reprise par divers outils qui cherchent à avoir un feedback des routeurs le long d'un chemin mais qui permettent de connaître un peu plus que *traceroute*. On trouve parmi ces outils à base de traceroute : *pathchar* [57] pour le débit des liens, *pathneck* [54] pour la localisation du goulot d'étranglement, *paris traceroute* [12] pour la détection des répartiteurs de trafic le long d'un chemin, et *AS traceroute* pour le repérage des domaines le long d'un chemin et pas seulement des routeurs.

– *path MTU discovery* [83] qui permet de détecter le MTU (Maximum Transmission Unit) le plus petit le long d'un chemin. Ceci se fait en essayant plusieurs tailles de paquet en commençant par des grands paquets et en désactivant chaque fois la fragmentation (bit DF dans l'entête IP mis à 1). Le retour d'un message ICMP de type "Fragmentation needed and DF set" est une indication qu'on n'a pas encore atteint la taille de MTU la plus petite.

ICMP ne répond pas à tous les besoins des opérateurs qui veulent connaître les caractéristiques du trafic sur leurs liens afin de pouvoir réagir à toute augmentation de trafic, détecter des anomalies, et planifier des futures mises à jour de leur réseau. Pour cela il y a eu la création du protocole SNMP qui permet à un opérateur d'interroger les routeurs de son réseau et de lire à distance par SNMP le contenu de certains registres dans chaque routeur. Dans ces registres, appelés MIB (Management Information Database) suivant la terminologie SNMP, sont stockées des informations agrégées sur le routeur et sur le trafic qui passe à travers. Les routeurs calculent des statistiques agrégées sur le trafic comme le débit moyen et le taux des pertes sur l'interface de sortie, et l'opérateur collecte ces statistiques à distance avec le protocole SNMP, ce qui lui permet d'avoir une idée de haut niveau de l'état de son réseau et de ses liens. Malheureusement avec SNMP il n'est pas possible d'avoir une information plus fine sur le trafic ni une information sur les réseaux des autres opérateurs.

SNMP permet la mesure passive de haut niveau des différents routeurs et liens d'un réseau. Mais les utilisateurs réguliers de l'Internet ont aussi besoin de connaître le trafic qui circule sur leurs interfaces réseaux pour diverses applications qui touchent aux performances et à la sécurité. La méthode la plus classique de le faire c'est d'accéder à la carte réseau et de la mettre en mode "promiscuous", le mode qui permet la lecture de tous les paquets qui partent de la machine et qui l'atteignent même ceux qui ne sont pas adressés à la machine même. Une fois la carte dans ce mode, il s'agit de copier les paquets au niveau du OS (Operating System),

de les filtrer, et de les remonter au niveau applicatif pour un traitement ultérieur. L'utilisateur peut effectuer cette intervention lui même, mais il existe aussi des librairies standards pour le faire comme les librairies *libpcap* [71] et BPF [80]. Une fois au niveau applicatif, les paquets peuvent être affichés d'une manière appropriée et stockés pour davantage d'analyse. Divers outils standards ont été proposés pour cet objectif, les plus connus étant *tcpdump* et *ethereal*. Ce genre d'outil permet une capture de tout le trafic présent sur l'interface réseau d'une machine, mais malheureusement est seulement orienté vers les stations de travail et ordinateurs et non pas les routeurs. Ces derniers possèdent une architecture différente et nécessitent alors un autre type d'outils. En plus, le trafic qui traverse un routeur atteint des valeurs beaucoup plus élevées que sur les interfaces des machines, ce qui nécessite des techniques de capture, d'agrégation, et de filtrage plus sophistiquées. *NetFlow* [85], l'outil standard pour le capture de trafic dans les routeurs Cisco et la référence dans ce domaine, sera exposé dans la section liée à l'évolution de la métrologie passive des réseaux.

Une autre information de valeur dans l'Internet c'est le contenu des tables de routage, surtout les tables BGP [97]. Dans les routeurs internes aux domaines, on trouve pour chaque préfixe appartenant au domaine l'interface de sortie et la distance à parcourir pour atteindre ce préfixe. Un préfixe est un ensemble d'adresses IP contiguës représenté par les bits de poids le plus fort communs à ces adresses (par exemple le préfixe 138.96.0.0/16 pour toutes les adresses qui ont les mêmes premiers 16 bits que 138.96.0.0). Pour les préfixes extérieurs au domaine, des sorties par défaut au niveau de chaque routeur interne peuvent être associées. Une autre possibilité est d'associer à ces préfixes externes une distance égale au nombre de sauts qu'il faut pour sortir du domaine plus une valeur indiquant la longueur du chemin restant une fois le paquet quitte le domaine. Certains protocoles de routage intra-domaine comme OSPF diffusent la carte entière du domaine à tous les routeurs, d'autres comme RIP diffusent seulement la longueur du chemin le plus court pour chaque préfixe. Que ce soit pour ce genre de protocole intra-domaine ou l'autre, les tables des routeurs à l'intérieur d'un domaine fournissent une information importante à l'opérateur sur l'état de son réseau. Malheureusement cette information ne permet pas d'avoir une idée sur ce qui se passe en dehors du réseau. Il est à ajouter que les tables de routage des routeurs internes à un domaine ne sont lisibles que par l'opérateur lui-même.

Pour avoir une idée sur le routage dans tout l'Internet à un moment donné, il faut recourir aux tables de routage BGP. En effet, dans un routeur BGP, se trouve en général et à tout moment l'ensemble de préfixes dans l'Internet, et pour chaque préfixe les domaines à parcourir pour y arriver. C'est une sorte d'arbre de chemins les plus courts pour aller d'un domaine vers tous les autres domaines, avec la répartition des préfixes sur les différents domaines. En combinant les tables de plusieurs routeurs BGP, une carte de l'Internet au niveau domaine peut être obtenue. La question intuitive qui se pose est si ces tables de routage sont rendues publiques. Certains opérateurs ont choisi d'ouvrir leurs routeurs BGP, et comme un routeur BGP accepte des connexions à distance des autres routeurs BGP, il suffit alors de se connecter à un de ces

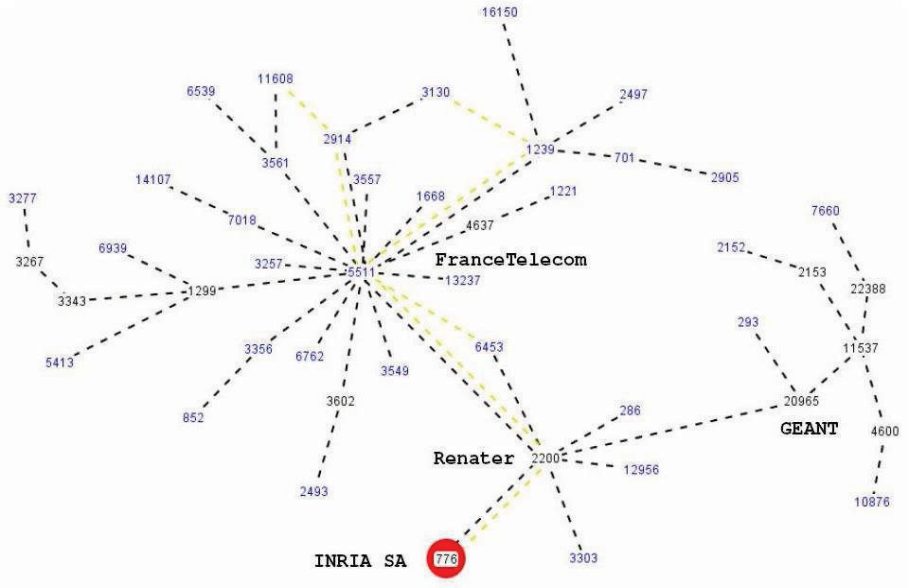

FIG. 2.1 – Carte de l'Internet au niveau domaine retournée par routeviews

routeurs ouverts, de faire comme s'il s'agit d'un routeur BGP voisin et de lui demander la liste des préfixes et routes qu'il détient. Certaines initiatives ont été lancées dans cette direction et ont abouti à une sorte de base de données pour les préfixes de l'Internet, le mapping entre les préfixes et les domaines, et la manière avec lesquelles les domaines de l'Internet sont connectés. L'initiative la plus connue est *routeviews* [101]. Parmi les applications fournies par routeviews une applet java qui trace pour chaque adresse IP, la carte de l'Internet au niveau domaine vue par cette adresse. Figure 2.1 montre un exemple de cette carte relative à une machine à l'INRIA Sophia-Antipolis.

La combinaison des tables de routage BGP donne une certaine image de la carte de l'Internet au niveau domaine. Un lien entre deux domaines apparaît sur cette carte si ce lien figure sur une route choisie par un des routeurs BGP considérés. Un lien qui n'apparaît sur aucune route ne peut pas être détecté par cette méthode. Intuitivement, la précision de la carte augmente avec le nombre des tables collectées et combinées ensemble. Il a été montré dans [17] qu'une dizaine de tables collectées à des endroits différents est suffisant pour avoir une bonne précision de l'Internet. Au delà de ce seuil le gain en considérant plus de tables est marginal.

Les tables de routage BGP peuvent aussi présenter un autre avantage quand observées dans le temps et non pas à un moment donné. En effet, quand un lien de l'Internet tombe en panne, tous les préfixes dont les routes passent par ce lien disparaissent pour peut être réapparaître plus tard si une autre route existe (ou le lien est rétabli). La corrélation entre les temps de disparition et d'apparition des préfixes est donc une bonne indication qu'ils se trouvent proches les un des autres. Cette corrélation a été observée dans [11] sur des grands intervalles de temps, pour conclure que c'est un outil efficace pour décider du regroupement des préfixes dans l'Internet.

2.2 L'évolution de la métrologie passive

La métrologie passive du réseau n'a pas cessé d'évoluer depuis la création de l'Internet. Les travaux dans cette direction peuvent être groupés en cinq classes selon leur objectif :

- Accélérer la vitesse de capture et stockage en mémoire des paquets.
- A partir du trafic collecté en un ou plusieurs points, inférer ce qui se passe ailleurs dans le réseau.
- De nouvelles méthodes d'agrégation qui permettent de compresser le trafic collecté sans perdre trop d'informations.
- Des méthodes d'échantillonnage qui permettent de réduire le volume de trafic à collecter et à traiter tout en restant capable d'inférer les caractéristiques du trafic original.
- Analyse et modélisation du trafic collecté.

Les sections suivantes décrivent brièvement les travaux effectués dans chacune des cinq classes.

2.2.1 Méthodes plus efficaces pour la capture des paquets

Le problème des outils comme *tcpdump* et *ethereal* est qu'ils nécessitent pour chaque paquet capturé, de le remonter à travers le système d'exploitation pour pouvoir le stocker dans la mémoire. Cette procédure est couteuse ce qui ralentit la vitesse de capture et cause des pertes de paquets. Pour accélérer cette procédure, la technique qui a été adoptée par plusieurs cartes spécialisées dans les mesures comme la carte DAG d'Endace [38] a été d'accéder directement à la mémoire pour l'écriture des paquets sans devoir passer par le système d'exploitation. Ceci a constitué une importante avancé dans ce domaine en permettant la capture des paquets à la vitesse que le bus de la station de travail et la mémoire la permettent.

Une autre solution utilisée a été celle de la capture d'une partie des paquets comme par exemple l'entête TCP/IP plus les quelques premiers octets de données. La plateforme de mesure passive IPMON de SprintLabs [46] a adopté cette solution. Le fait que les paquets capturés sont ramenés à une plus petite taille a permis de les écrire en mémoire à une vitesse plus élevée. Pour la plupart des applications, connaître l'entête est suffisant. Malheureusement, si pour une application quelconque comme par exemple l'identification du trafic par son contenu, l'utilisateur ou l'opérateur a besoin de plus de données, les obtenir ca sera impossible étant donné le caractère irréversible de l'approche.

L'introduction des fonctions de hachage [73] pour la capture et l'analyse des paquets a aussi permis de gagner considérablement en performance. Avec les fonctions de hachage appliquées à certains champs dans les paquets, il est devenu facile d'accéder directement dans la mémoire à la case qui contient les informations sur chaque paquet. Ceci a permis de calculer facilement sur un trafic des métriques comme le nombre de paquets par adresse IP ou par réseau. Dans le cas où la vitesse d'accès à la mémoire est considérée toujours inacceptable, des filtres de type Bloom [23] peuvent être utilisés pour effectuer un premier filtrage rapide des informations, suivi par un accès

à la mémoire. L'exemple le plus typique est celui de la mesure des adresses IP sources dans un trafic. Dans un premier stade un filtre Bloom est utilisé pour savoir si une adresse IP source a déjà était vue, et si ce n'est pas le cas, la mémoire est accédée pour l'enregistrement de cette adresse. Les travaux basés sur les fonctions de hachage et les filtres Bloom sont souvent présentés dans la littérature sous le titre "Data Streaming".

2.2.2 Inférence des performances réseau

La métrologie passive du réseau consiste par définition en la capture des données en certains points du réseau. Ceci permet facilement le calcul des métriques liées à ces points comme par exemple la répartition du trafic entre les différentes applications, le volume du trafic, le nombre de flots, la distribution de la taille des paquets, etc. Par contre, il est difficile de calculer des métriques de nature bout-en-bout, c'est-à-dire des métriques liées à un chemin de l'Internet plutôt qu'à un ou plusieurs points de mesure. L'inférence des caractéristiques d'un chemin nécessite le déploiement des points de mesure passive partout dans le réseau, mais malheureusement ceci n'est pas faisable, ou au moins, il est impossible d'imaginer que ces points de mesures même si déployés par les différents opérateurs, puissent partager leurs mesures. Pour des raisons de sécurité, les opérateurs ne permettent pas à des extérieurs d'effectuer des mesures sur leur réseau, et n'acceptent pas de partager les résultats de leurs mesures.

Malgré ces obstacles, des efforts ont été entrepris pour inférer certaines caractéristiques des chemins à partir des mesures passives. Dans [59], une étude a été menée pour inférer les caractéristiques d'une connexion TCP à partir des mesures effectuées en un seul point. Des statistiques comme le débit, la version TCP, l'évolution de la fenêtre et du RTT (Round-Trip Time), ont été calculées. Ceci est rendu possible par la nature bidirectionnelle de TCP, où l'observation simultanée des paquets de données et des acquittements permet de comprendre ce qui se passe au niveau de la source et de la destination. [65] a essayé de calculer les débits physiques des liens le long d'un chemin de l'Internet à partir de l'histogramme du temps qui sépare les paquets d'une connexion TCP traversant ce chemin. [109] utilise les paquets d'une connexion TCP dans les deux sens pour inférer la bande passante disponible le long d'un chemin. Finalement, plusieurs travaux (e.g., [106]) ont essayé d'inférer ce qu'on appelle la matrice de trafic d'un réseau (le trafic entre chaque point d'entrée et chaque point de sortie) à partir des mesures passives effectuées à l'intérieur du réseau. Le point commun à tous ces travaux est la difficulté de la procédure et le grand bruit dans le résultat obtenu.

Le placement des points de mesures passives est un problème en soi. D'un côté ces points ont un certain coût et consomment des ressources que ce soit pour collecter le trafic où pour le transporter. De l'autre côté, certains de ces points de mesure peuvent être complètement redondants avec d'autres ou complètement inutiles pour la métrique considérée. Par exemple, pour calculer le trafic qui transite entre deux réseaux, il suffit de mettre un point de mesure sur le lien de transit au lieu de mesurer tous les liens du réseau et identifier le trafic qui transite par ce

lien. Mesurer seulement le lien de transit est par contre insuffisant pour connaître la répartition de la charge à l'intérieur du réseau.

Divers travaux ont regardé le problème de placement, souvent du point de vue du calcul de la charge sur le réseau (e.g., [108]). La question posée était comment placer le minimum de moniteurs pour capturer au moins une fois tous les paquets qui traversent un réseau. Il s'agit d'un problème classique de la théorie des graphes, qui peut être résolu avec des heuristiques voire une recherche exhaustive dans le cas de petits réseaux. De notre côté, nous avons proposé dans [**?**] une solution générale adaptative qui profite du fait que les routeurs de l'Internet disposent déjà d'outils de mesure comme les MIBs de SNMP et NetFlow [85]. Nous avons bâti sur ces outils pour le développement d'une solution adaptive qui, en fonction de la tâche de mesure à effectuer, fait activer les bons routeurs pour la collecte d'informations sur le trafic dans le réseau. Le Chapitre suivant décrit en détail notre approche. En plus de la simple activation ou désactivation d'un routeur, notre approche tient en compte le fait que les routeurs peuvent participer différemment à une tâche de mesure en collectant le trafic à des vitesses différentes (ce qui correspond à des taux d'échantillonnage différents). Ceci permet un degré de liberté de plus ce qui améliore davantage les mesures passives et les rend plus adaptatives à la tâche de mesure qu'un opérateur désire effectuer.

2.2.3 Agrégation des informations sur le trafic

Collecter et reporter tous les paquets est une opération très couteuse qui ne passe pas à l'échelle. Pour cela il est important de trouver des méthodes d'agrégation du trafic qui permettent de réduire le volume des données à reporter et à traiter sans perdre trop d'informations. Ce qui pousse en faveur de ces méthodes c'est que les applications dans la métrologie n'ont pas besoin de connaître tout sur le trafic. Par exemple, les données applicatives dans les paquets sont rarement utilisées. Pareil pour certains champs comme le checksum, la version de IP, les options, le numéro de séquence TCP, etc. Une méthode d'agrégation qui regroupe les paquets en fonction de certains critères et qui transmet que des informations agrégées comme le nombre d'octets et le nombre de paquets, est souvent suffisante. Pour les applications qui demandent plus, malheureusement il sera trop tard et il faudra refaire la mesure mais cette fois-ci avec une agrégation différente.

La méthode d'agrégation la plus connue, dite la méthode de base ou la méthode la plus fine, est celle qui regroupe les paquets en utilisant les cinq champs suivants dans l'entête : adresse IP source, adresse IP destination, numéro de port source, numéro de port destination, et protocole de transport. L'ensemble des paquets groupés qui en résultent correspond à une connexion TCP ou bien à un flot UDP. Cet ensemble est souvent appelé flot 5-tuple ou flot défini à base de quintuple. Pour chaque flot 5-tuple, un ensemble d'informations peuvent y être associé comme le nombre d'octets, le nombre de paquets, le temps de début et de fin, les bits du champ *TCP flags*, les bits du champ *type of service*, etc. Avec cette méthode, un opérateur collecte donc des flots avec des informations sur chaque flot et essaie d'utiliser ce retour pour comprendre ce qui se passe

dans son réseau. Des statistiques comme l'utilisateur le plus gourmand, les réseaux générant le plus de trafic, le nombre de requêtes TCP envoyées à un serveur ou à un réseau, etc, peuvent être facilement calculées à partir des flots 5-tuples. Il s'agit souvent d'une agrégation des flots 5-tuples à un niveau plus élevé. Par exemple, une agrégation */24 adresse source* est équivalente à mettre ensemble tous les flots 5-tuples dont les adresses IP sources partagent les mêmes 24 bits en partant de gauche.

Un mot important à dire à propos des flots est leur durée. A l'exception des flots TCP dont le début et la fin sont bien définis par les paquets SYN et FIN, un flot en général est associé à un certain intervalle de temps. Au bout de cet intervalle de temps, le flot est considéré comme étant fini et un autre flot est créé même si des paquets continuent à transiter. Ceci est fait pour éviter que certains flots ne durent pas indéfiniment et pour limiter le temps de retour des mesures.

L'agrégation en flots est supportée par les nouveaux routeurs pour permettre aux opérateurs une connaissance plus fine que celle de SNMP de ce qui se passe dans leur réseau. L'exemple le plus typique est NetFlow [85] de Cisco qui est sorti en plusieurs versions. La version 5 permet une agrégation à base de quintuple ; la version 8 permet plus de contrôle en proposant 11 niveaux d'agrégation ; et la version 9 permet une agrégation flexible et extensible pour supporter de nouveaux champs et technologies. NetFlow dispose d'un certain cache pour le stockage des flots. Un flot est reporté quand il est déclaré fini. Le débordement du cache provoque aussi le rapportage des flots les plus âgés.

2.2.4 L'échantillonnage dans la métrologie

Il s'agit d'un domaine d'un grand intérêt dans la réduction du volume du trafic à collecter et à traiter. Ce domaine a gagné beaucoup d'importance dans les dernières années et a été motivé par le fait que certaines métriques dans la métrologie ne nécessitent pas la capture de tous les paquets. Par conséquence, il a été proposé de réduire le volume du trafic collecté pour ces métriques afin de réduire la charge sur les routeurs et permettre aux systèmes de mesure d'opérer à des débits encore plus élevés. Le calcul se fait alors sur la partie collectée du trafic. Clairement, et comme dans chaque application où l'échantillonnage est pratiqué, il faut trouver une méthode pour inverser les observations et calculer la valeur de la métrique relative au trafic original à partir du trafic échantillonné. Il faut aussi trouver des méthodes intelligentes pour effectuer l'échantillonnage sans trop induire en erreur le résultat. Dans certains cas où la métrique est relative à une grande population comme par exemple le volume moyen du trafic, le calcul est simple et des taux d'échantillonnage réduits peuvent être pratiqués. Par contre, dans les autres cas où la métrique est dépendante de populations de petites tailles comme les petits flots, l'échantillonnage et l'inversion demandent un effort non négligeable souvent avec l'aide d'outils mathématiques et stochastiques.

Le choix des points de mesure peut aussi être vu comme un problème d'échantillonnage. C'est plus de l'échantillonnage spatial que de l'échantillonnage temporel. Au lieu de répondre à la

question combien de paquets il faut prendre pour calculer une certaine métrique, l'échantillonnage spatial s'intéresse à répondre à la question d'où il faut capturer les paquets. Dans la littérature ces deux aspects de l'échantillonnage (spatial et temporel) ont été abordés indépendamment l'un de l'autre. Notre travail a consisté en une grande partie à les combiner ensemble pour obtenir une architecture qui permet, en fonction de la tâche de mesure désirée, de choisir où il faut collecter les paquets et à quel taux, avec la possibilité d'avoir des taux d'échantillonnage différents dans les différents routeurs. Nous avons voulu que notre architecture soit capable de s'adapter aux variations du trafic et à la diversité des tâches de mesure qui peuvent être entreprises dans un réseau. Notre architecture, en combinant les deux aspects, est capable d'atteindre de meilleures performances.

L'échantillonnage est utilisé par les opérateurs mais souvent sans une étude solide derrière qui indique comment il faut faire pour avoir des résultats pas très erronés. Certains opérateurs utilisent des taux très bas qui peuvent convenir qu'à un ensemble restreint de métriques. Cisco propose d'ajouter l'échantillonnage à NetFlow avec une méthode d'inversion simple ; pour un taux d'échantillonnage $p\%$ (seulement $p\%$ des paquets sont collectés), les flots sont premièrement formés en partant du trafic échantillonné, après chaque flot a son volume multiplié par $100/p$ pour parvenir à une estimation de son volume original avant l'échantillonnage. Ces pratiques sont loin de l'optimal (il faut penser au cas des petits flots) et de la recherche est encore nécessaire pour mieux comprendre l'impact de l'échantillonnage et comment l'optimiser. Les trois chapitres suivants sont consacrés à nos contributions dans ce domaine. Dans Chapitre 3 (détaillé dans Annexe A), nous présenterons notre architecture qui optimise l'échantillonnage dans un réseau étant donnée une tâche de mesure. Chapitres 4 et 5 (détaillés dans Annexes B et C respectivement) expliquent sur deux exemples concrets l'impact de l'échantillonnage sur les résultats de mesure et comment faire pour inverser les informations échantillonnées. Cette étude est loin d'être exhaustive, cependant la méthodologie utilisée dedans peut être appliquée à d'autres problèmes comme la détection des anomalies et la prédiction du trafic.

2.2.5 Analyse et modélisation du trafic collecté

La collecte du trafic a diverses applications. Les opérateurs s'en servent pour connaître l'état de leurs réseaux. Ceci peut se faire en termes du volume du trafic sur les différents liens en bits/s ou bien en termes des applications et des identifiants des utilisateurs qui participent à la génération du trafic. Ces observations servent à la fois à mieux administrer le réseau et à détecter et réagir à des anomalies et des situations de crises (ce domaine s'appelle le Trafic Engineering). Une autre application est la planification des mises à jour du réseau en fonction de l'évolution de la charge. Malheureusement, certaines de ses observations sont difficiles à obtenir que ce soit parce que les utilisateurs cryptent leurs paquets et donc empêchent l'opérateur au milieu d'accéder à leur contenu [20], ou bien parce que certaines de ces anomalies et attaques sont subtiles que les opérateurs ne peuvent pas les détecter sans une étude minutieuse du trafic [69].

Les scientifiques à leur tour sont intéressés par l'analyse du trafic de l'Internet dans le but de comprendre sa composition et son évolution, et de proposer des modèles analytiques qui peuvent servir à la fois pour prédire le trafic et pour le générer synthétiquement dans des simulations et des expérimentations contrôlées. Ce dernier point est très important pour la validation des protocoles de l'Internet et de toute proposition visant à les améliorer. C'est aussi important pour la validation de tout nouveau système dans des conditions qui peuvent se reproduire et qui sont les plus réalistes possibles. La validation des nouvelles versions de TCP et des mécanismes de contrôle de congestion en est une bonne illustration.

La modélisation du trafic de l'Internet passe d'abord par une étude statistique pour comprendre et quantifier les caractéristiques principales du trafic. Ensuite des modèles analytiques sont proposés pour capturer ces caractéristiques. Les premiers travaux dans cette direction faisaient des hypothèses simples sur le trafic de l'Internet (des d'hypothèses du sorte Markoviennes), après il s'est avéré que le trafic de l'Internet présente un comportement longue mémoire (appelé en anglais long range dependent) qui demande des nouveaux modèles [70]. La différence entre un trafic longue mémoire et un trafic Markovien est que la mémoire du premier décroît polynomialement avec le temps tant dis que la mémoire du deuxième décroît exponentiellement. Cette propriété longue mémoire a été attribuée à la propriété queue lourde (ou heavy tail en anglais) de la durée, volume et nombre des fichiers et flots dans l'Internet (propriété très connue en informatique par rapport à tout ce qui est humain et qui présente un comportement plutôt Markovien ou exponentiel) [33]. Par conséquence les modèles qui sont proposés pour le trafic de l'Internet sont en grande partie hiérarchiques qui procèdent par la superposition des flots à propriété queue lourde e.g. [16, 74, 111]. Il existe cependant d'autres modèles basés sur des équations de récurrence stochastique et qui permettent à partir des valeurs de trafic dans le passé et des valeurs d'autres variables aléatoires et processus stochastiques, de trouver une valeur pour le trafic à l'instant actuel de telle sorte que le trafic total généré satisfait certaines conditions bien définies en termes de débit moyen, variabilité et corrélation. Le mouvement brownien fractionnaire et les processus ARMA et ARIMA en sont un exemple.

Dans [16] nous avons contribué à ce domaine en proposant un modèle à base de flots pour le trafic de l'Internet. Nous avons lié les propriétés du trafic total à la variabilité du débit de chaque flot. Dans un travail en cours, nous sommes en train d'aller plus loin pour proposer des modèles statistiques des flots générés par les différentes applications de l'Internet dans le but de les classer ultérieurement en utilisant seulement des informations sur la taille et instants d'arrivée des paquets sans devoir se fier seulement aux informations dans les entêtes. Ceci a un grand intérêt en cas de trafic crypté.

2.3 L'évolution de la métrologie active

Les opérateurs de l'Internet peuvent mesurer tous les liens de leurs réseaux d'une manière passive pour comprendre ce qui se passe, cependant ils sont incapables d'exécuter la même fonction dans les réseaux des autres opérateurs. Quant aux utilisateurs et aux administrateurs des réseaux de recouvrement, tout ce qu'ils peuvent exécuter c'est une fonction de capture du trafic sur l'interface réseau de leurs machines. Cette dernière capture donne une information très peu utile pour la compréhension de l'état du réseau et les performances des applications. Il y a donc un fort besoin de collecter des informations sur l'état du réseau au delà de l'interface réseau et ceci ne peut se faire que par une approche active pour ceux qui ont pas accès aux liens du réseau. D'ici vient l'intérêt de cette approche. Son grand problème cependant est le manque de précision étant donnés la complexité du réseau et le grand nombre d'applications et utilisateurs.

D'après le principe de la métrologie active, des paquets sont envoyés dans le réseau et c'est en fonction de la réponse du réseau à ces paquets (i.e., délai qu'ils prennent, débit, pertes) que quelqu'un peut inférer ce qui se passe à l'intérieur du réseau et sur le chemin entre la machine qui envoie les paquets de mesure et celle qui les reçoit et qui renvoie les acquittements si jamais. Notez ici que la mesure active peut se faire sans acquittement si c'est la machine destinatrice qui mesure et si les horloges sont synchronisées. A noter aussi que les paquets de mesure peuvent être des paquets dédiés avec des données de bourrage (dummy packets) ou bien des vrais paquets de données comme ceux des transferts de fichiers.

Les travaux sur les mesures actives peuvent être classés en deux catégories, la deuxième catégorie se base en grande partie sur des outils de la première catégorie :

- Travaux visant à évaluer les performances d'un chemin entre deux machines.
- Travaux visant à mesurer tout un réseau en corrélant plusieurs mesures actives. Le réseau à mesurer peut être l'Internet lui même au niveau de ses routeurs et domaines, comme ça peut être au niveau d'un réseau Overlay (réseau au niveau applicatif ou réseau superposé).

Nous résumons dans la suite les principaux travaux dans la littérature dans chacune de ces deux catégories ainsi que nos contributions dans le domaine.

2.3.1 Métrologie d'un chemin de l'Internet d'une manière active

Les outils de base de type ping ou traceroute permettent aux applications de tester la disponibilité d'un chemin de l'Internet. Cependant les applications sont aussi sensibles à d'autres paramètres, typiquement le délai, la bande passante et le taux de pertes des paquets. Le taux de sensibilité dépend de la nature de l'application qui peut se traduire sous la forme d'une fonction d'utilité liant la qualité perçue par l'utilisateur de l'application à l'ensemble des métriques caractérisant le chemin réseau emprunté par les paquets de l'application. Pour les applications de type transfert de fichiers (dites élastiques [19]), la fonction d'utilité varie principalement en fonction de la bande passante disponible, mais pour les courts transferts et à cause de la phase Slow Start

de TCP, elle est connue comme étant fonction aussi du délai aller-retour [30]. En effet, au début d'un transfert de fichier, il faut un certain temps pour atteindre la phase stationnaire et ce temps augmente linéairement avec le délai. Une fois la phase stationnaire atteinte, il est logique de dire que TCP essaie d'utiliser pleinement la bande passante disponible (si le récepteur le permet), et donc le temps restant dans la vie d'un transfert est principalement le rapport volume de données restant sur bande passante.

Pour les applications multimédia qui utilisent souvent le protocole UDP, la situation est différente. D'un côté une application multimédia n'a pas toujours besoin de toute la bande passante disponible. De l'autre côté, et en plus de la bande passante, une application multimédia a sa qualité dépendante du délai aller-retour (surtout s'il s'agit d'une application temps réel) et du taux de pertes De paquets [21]. Pour réduire le taux de pertes et ne pas être agressif, il a été fortement recommandé que les applications multimédia transmettent au même débit que TCP (le fameux principe de TCP-friendliness [44]). La connaissance de ce débit est très importante pour l'application afin qu'elle puisse choisir son codec et la quantité de redondance à ajouter au flux pour corriger toute perte éventuelle de paquets.

Cette dépendance de la qualité applicative en l'état du réseau a motivé un grand nombre de recherche sur des outils de mesure efficaces pour la métrologie des chemins de l'Internet. Fournir une liste exhaustive de ces outils de mesure est une tâche impossible étant donné leur grand nombre. Certains sites comme celui de CAIDA [31] présentent un descriptif des principaux outils de mesure qu'il convient de consulter avant le développement de son propre outil. Dans cette section cependant nous allons nous contenter de citer les principales approches dans la métrologie d'un chemin de l'Internet.

Mesure du délai

La mesure du délai d'un chemin de l'Internet a beaucoup intéressé les chercheurs. Malheureusement et en l'absence d'horloges parfaitement synchronisées, la mesure exacte du délai unidirectionnel est une tâche difficile si ce n'est pas impossible. Beaucoup de protocoles comme TCP [56] et TFRC [45] se contentent de la mesure du délai aller-retour. Cette information peut être obtenue exactement si les récepteurs renvoient rapidement leurs acquittements ou au moins informent les sources du temps passé avant le renvoi des acquittements. Dans le cas des chemins parfaitement symétriques, le délai unidirectionnel peut être considéré comme la moitié du délai aller-retour, ce qui n'est pas malheureusement le cas dans l'Internet où une grande partie des chemins souffrent d'une asymétrie. La seule solution restante serait de synchroniser les horloges autant que possible avec des protocoles comme NTP voire du GPS, et ensuite d'estampiller les paquets à l'envoi et à la réception. Certains travaux dans la littérature ont focalisé sur l'élimination de toute différence dans les fréquences des horloges qui peut causer un décalage des horloges avec le temps e.g. [84].

Mesure du taux de pertes

Le calcul du taux de pertes de paquets le long d'un chemin a aussi intéressé les chercheurs surtout les développeurs des applications multimédia qui ont à optimiser la quantité de redondance (FEC) à ajouter aux paquets dans le but de pouvoir les reconstruire sans devoir les retransmettre. Il est connu que le taux de redondance et la manière dont elle est codée sont fortement dépendants de comment les pertes se produisent [22]. Le taux de pertes est souvent calculé sur une certaine fenêtre pour à la fois donner une valeur significative et pour que cette valeur puisse adapter à tout changement dans l'état du réseau. Le choix de cette fenêtre est souvent un compromis entre précision et réactivité [45]. En plus du taux moyen de pertes, le profile des pertes de paquets est aussi d'une certaine importance. Le fait que les paquets se perdent souvent en rafales dans l'Internet [22] a motivé le développement des mécanismes FEC spécifiques [9, 22]. L'analyse du profile de pertes passent souvent par une collecte de statistiques sur les paquets avec les identifiants de ceux reçus et de ceux perdus, et ensuite par le calibrage d'un modèle statistique à ce profile. Un exemple d'outil à utiliser est la chaîne de Markov à deux ou plusieurs états comme nous l'avons fait dans [8].

Mesure de la bande passante

C'est la métrique de chemin qui a suscité le plus de recherche parce que finalement la bande passante est ce qui intéresse le plus l'utilisateur et le fournisseur de service. Au début, la bande passante d'un chemin a été définie comme étant le débit physique du lien le plus lent le long d'un chemin de l'Internet. Et c'est dans cette perspective que des outils comme *pathchar* [57] et *packet-pair* [66] ont été développés. Le premier utilise le délai pour inférer la bande passante et le deuxième utilise l'espacement entre paquets pour le même objectif. Finalement, seul le délai peut être mesuré de bout en bout et toute l'intelligence revient à trouver le lien entre la bande passante dans le coeur du réseau et le délai des paquets, après il s'agit d'un simple problème d'inversion et filtrage pour trouver la valeur la plus probable pour la bande passante.

Le débit du lien le plus lent (appelé souvent le goulot d'étranglement) est une métrique importante parce qu'elle donne une idée sur la borne supérieure que le débit peut atteindre. *pathchar* est même capable de localiser l'emplacement de ce lien. Cependant le chemin emprunté par les paquets est très probablement chargé par le trafic d'autres utilisateur et donc la bande passante qu'une application peut réaliser peut être bien inférieure à celle du débit du lien le plus lent. La notion de *la bande passante disponible* a alors été introduite et a été définie comme étant la partie de la bande passante non utilisée le long d'un chemin à un instant donné. C'est le débit maximum qu'une application peut réaliser le long de ce chemin à cet instant, en ne considérant pas bien sûr le cas d'une application agressive qui bombarde le réseau avec du trafic UDP et qui pousse le reste du trafic TCP à reculer. A la différence du débit du goulot d'étranglement, la bande passante disponible est une quantité qui varie avec le temps. [1] En plus, la bande passante

[1] Suite au changement de route, le goulot d'étranglement peut aussi varier mais ceci arrive à une fréquence

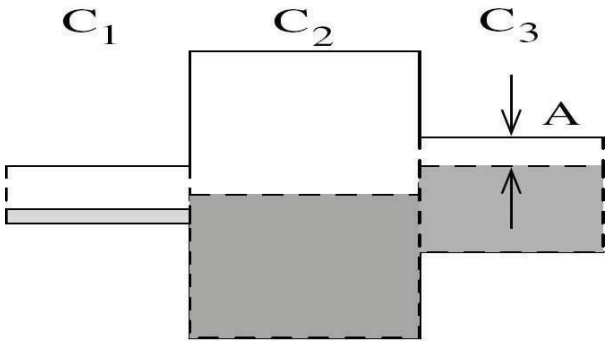

FIG. 2.2 – Goulot d'étranglement (C1) et bande passante disponible (A sur C2)

disponible peut être imposée par un autre lien que celui du goulot d'étranglement comme c'est le cas par exemple dans Figure 2.2.

Un grand nombre d'outils de mesure de bande passante disponible a été proposé dans la littérature e.g. [32, 53, 58, 82, 107]. Ces outils appartiennent à deux classes. La première classe est celle qui consiste à estimer l'intensité du trafic réseau, et connaissant le débit du goulot d'étranglement (ou le débit physique du lien), elle calcule la bande passante disponible comme étant la différence entre le débit physique et l'intensité du trafic. Le débit physique du lien peut être obtenu par une méthode à base de packet-pair comme expliqué ci-dessus. Le calcul du trafic réseau des autres utilisateurs se fait souvent par l'envoi des pairs de paquets proches les uns des autres sans qu'ils soient collés, et ensuite par le calcul de l'espacement de ces paquets à la sortie du réseau, un espacement qui ne peut résulter que des paquets des autres utilisateurs qui viennent s'insérer entre eux. Avec des techniques de filtrage de bruit, le débit moyen du trafic peut alors être calculé. Des outils comme [72, 53, 107] entrent dans cette catégorie. Un des avantages de cette approche est qu'elle ne demande pas l'envoi de beaucoup de paquets et donc par conséquence elle ne perturbe pas le réseau, son problème cependant est qu'elle nécessite la connaissance du débit physique du lien limitant la bande passante disponible.

L'autre classe d'outils est basée sur le principe d'envoi de train de paquets à un débit élevé qui crée une congestion momentané dans le réseau, et ensuite de mesurer le débit de sortie de ces paquets. Le premier travail dans cette direction a été *cprobe* [32] qui a fait l'hypothèse que le débit de sortie des paquets du réseau est celui de la bande passante disponible. Autrement dit, sur un chemin qui a une bande passante A, tout train de paquets envoyé à un débit plus grand que A quitte le réseau à un débit égal à A avec bien évidement un certain bruit à enlever. Si le débit des paquets sondes est inférieur à A, ils quitteront le réseau au même débit d'entrée. Le problème avec un tel outil est qu'il suppose que le trafic réseau ne recule pas face aux paquets sondes. Ceci a un sens quand les paquets sondes sont envoyés à un débit légèrement supérieur à la bande passante disponible, mais quand ils sont envoyés à un débit très élevé, ils vont finir par monopoliser les liens du réseau et passent devant le reste du trafic ce qui fait qu'ils peuvent

beaucoup moins inférieure que la bande passante disponible.

quitter le réseau à un débit beaucoup plus grand que A en fonction de leur agressivité. Ceci vient du fait que le réseau traite tous les paquets de la même manière et n'isole pas le trafic réseau des paquets sondes. Le cas limite c'est l'envoi des paquets sondes collés qui arrivent collés au goulot d'étranglement, qui attendent dans le buffer FIFO (First-In First-Out) du routeur les uns derrière les autres, et qui sortent après du goulot collés espacés par le débit physique du goulot. Dans ce cas, le débit mesuré est celui du débit physique du lien et non pas la bande passante disponible.

Quelques années après *cprobe* la notion de mesure de bande passante a été revue pour tenir compte de l'agressivité des paquets sondes [58, 82]. Supposons que le lien limitant la bande passante a un débit physique C et que la bande passante disponible est A. Le trafic réseau est donc en moyenne égal à $C - A$. Prenons R le débit d'émission des paquets sondes et étudions comment le débit de réception des paquets sondes varie avec R. Comme on le voit dans Figure 2.3, tant que le débit R ne dépasse pas A, le réseau n'est pas congestionné et les paquets sondes quittent en principe au même débit d'émission. Les paquets sondes congestionne le réseau quand ils sont envoyés à plus que A. C'est à partir de ce point que le réseau devient incapable de servir les paquets sondes à leur débit d'émission et se trouve obligé de les ralentir. Il est connu qu'un routeur implémentant une politique FIFO repartit le trafic à sa sortie proportionnellement au débit d'entrée. Par conséquence les paquets sondes se trouvent servis à un débit $CR/(C-A+R)$ tant dis que le reste du trafic réseau se trouve servi à un débit $C(C-A)/C-A+R)$ de telle sorte que la somme des deux débits est égale à la capacité du lien C. En augmentant R au delà de A, les paquets sondes deviennent de plus en plus agressifs et passent de plus en plus devant les paquets du trafic. Leur débit de sortie par conséquence augmente sous linéairement jusqu'à atteindre C pour un débit R très grand (le cas de paquets collés). Le débit de réception des paquets sondes est alors égal à A dans le seul cas où R est égal à A, ce qu'un outil de mesure cherche à calculer d'où la difficulté du problème.

D'après cette approche, la bande passante disponible A peut se mesurer de deux manières. Premièrement en inversant la courbe dans la Figure 2.3 e.g. [82]. La difficulté réside dans le fait que cette courbe demande la connaissance de C. Dans le cas où le lien goulot d'étranglement est le même que celui limitant la bande passante, cette information peut être obtenue avec un certain effort raisonnable, par contre quand les deux liens sont différents, la connaissance de C est loin d'être évidente puisqu'elle demande la connaissance de la bande passante disponible ce qui est la quantité à rechercher. L'autre solution pour calculer la bande passante disponible serait de repérer le point de début de la zone de congestion (point où R est égal à A). C'est en ce point là que le débit de réception des sondes est égal à celui de la bande passante disponible. La détection de ce point peut se faire en balayant l'espace de débit d'émission R et en traquant l'évolution du débit de réception des sondes. Le problème c'est que à cause du bruit réseau et de l'erreur de mesure, la détection du changement de comportement du linéaire au sous-linéaire peut manquer de précision. La méthode la plus efficace et qui a été adoptée par *pathload* est la détection du

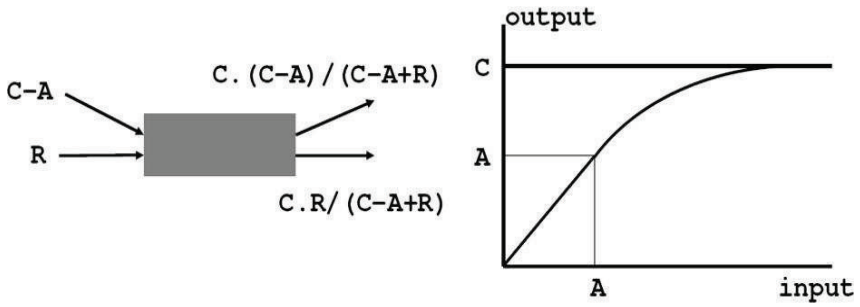

FIG. 2.3 – Débit d'émission des paquets sondes et bande passante disponible

début de la zone de congestion par l'augmentation du délai. En effet, dès que R dépasse la bande passante disponible, le buffer dans le routeur commence à se remplir et la file d'attente à se former ce qui provoque une augmentation continue du délai des paquets de probe. Il suffit alors de détecter cette augmentation pour conclure que nous nous trouvons à droite du point désiré (zone de congestion). A gauche du point désiré cette augmentation du délai des paquets sondes ne doit pas avoir lieu. Avec une telle méthode pour détecter si nous nous trouvons à droite ou à gauche du point désiré et une méthode dichotomique pour l'adaptation du débit R, il devient facile de localiser rapidement et précisément le point début de congestion. Une simple mesure du débit d'émission ou de réception en ce point (les deux égaux) est suffisant pour donner la valeur de la bande passante disponible.

Dans notre travail sur la bande passante les dernières années nous n'avons pas cherché à trouver des meilleurs outils mais plutôt à mieux interpréter les résultats obtenus par des outils existants. A partir des mesures de bande passante effectuées entre un certain nombre de machines, nous avons voulu comprendre si une corrélation *spatiale* existe entre les différentes mesures et s'il y a moyen de tracer une carte de bande passante de l'Internet qui permet d'inférer la bande passante entre deux machines à tout instant sans devoir la mesurer directement [76, 77]. Nous avons voulu vérifier si la corrélation spatiale qui existe pour une métrique comme le délai existe aussi pour la bande passante. Malheureusement, ceci s'est avéré difficile à atteindre, à moins qu'un grand nombre de proxies de mesure de la bande passante est déployé dans le réseau [76, 77]. Nos expérimentations sur PlanetLab [91] ont montré que pour inférer la bande passante entre deux machines, il faut pratiquement déployer un proxy à l'entrée du réseau local de chacune des intuitions partenaires à PlanetLab. La mesure de bande passante entre ces proxies peut alors servir à approximer la bande passante entre les deux machines en question. Les résultats pour le délai sont complètement différents [63]. Le fait que le délai forme une métrique dans un espace Euclidien permet un plongement efficace des délais des chemins de l'Internet à partir d'un sous ensemble de mesures. Ce plongement peut alors servir pour l'inférence des délais entre machines par le simple calcul de la distance Euclidienne entre ces machines dans l'espace Euclidien. Le plongement de la bande passante nécessite pratiquement toute la matrice de mesures et est par

conséquence peu efficace.

2.3.2 Métrologie d'un réseau d'une manière active

En parallèle à la caractérisation d'un chemin de l'Internet connectant une paire de machines, un grand nombre de travaux a focalisé sur la corrélation des mesures actives effectuées par un ensemble de machines pour la caractérisation du réseau les connectant. Cette corrélation des observations a un grand gain puisqu'elle permet de tirer des conclusions sur la topologie du réseau et sur la localisation des incidents. Par exemple, une machine qui parle avec un grand nombre de machines et qui constate que le délai est plus grand que la normale vers toutes ces machines peut conclure que le problème vient de son lien d'accès. De même, une machine qui sonde un certain nombre de machines et qui constate que pour deux des machines la corrélation des pertes de paquets est plus forte que pour les autres peut conclure que ces deux machines se trouvent dans le même voisinage dans le réseau. Dans la littérature, les travaux dans cette direction sont souvent présentés sous le titre "tomographie de l'Internet" ou bien parfois "inférence de la topologie de l'Internet".

Caractérisation des liens d'un réseau

Il s'agit d'inférer les propriétés des liens du réseau à partir de mesures de bout en bout sans faire intervenir les noeuds intermédiaires. Parmi les propriétés à inférer se trouvent le délai et le taux de pertes de paquets. C'est un problème qui normalement n'a pas de solution sur un seul chemin. En effet, ce qu'on mesure de bout en bout sur un chemin c'est la convolution des distributions de la métrique (e.g. délai) sur les différents liens du chemin, et la déconvolution dans ce cas est impossible comme il y a plusieurs distributions liées par une seule équation. Par contre dès qu'on parle du réseau, la déconvolution devient possible grâce à la corrélation qui existe entre les mesures. Le travail le plus connu dans ce domaine et qui cherche à caractériser les liens d'un réseau s'appelle MINC (Multicast Inference of Network Characteristics) [2]. La première version de MINC a visé à caractériser le délai et le taux de pertes des liens le long d'un arbre de diffusion multicast. Les noeuds de l'arbre sont les routeurs multicast ou des proxies multicast au niveau applicatif. A partir des rapports renvoyés par les récepteurs sur les paquets reçus et sur leur délai de réception, la source est capable avec MINC de caractériser les liens de l'arbre de bout-en-bout sans que les noeuds de l'arbre (routeurs ou proxies) aient besoin de se sonder entre eux. Ceci a un grand intérêt puisqu'une source de diffusion multicast peut alors identifier de bout-en-bout les liens du réseau qui présentent un problème et réagir en fonction par exemple en réorganisant l'arbre de diffusion ou bien en séparant les récepteurs en plusieurs groupes et en envoyant du contenu différent à chacun des groupes.

Prenons l'exemple suivant pour comprendre la problématique de MINC et l'intérêt de la corrélation des observations. Dans la Figure 2.4, une source diffuse en multicast à travers un noeud intermédiaire (routeur ou proxy). p, p_1 et p_2 sont respectivement le taux de pertes des

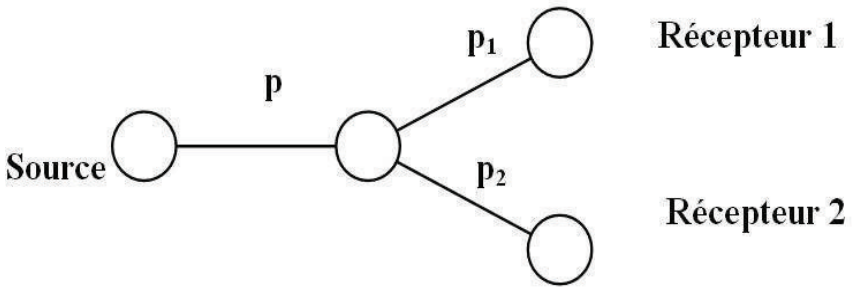

FIG. 2.4 – Exemple simple montrant l'intérêt de la corrélation des observations lors de la métrologie d'un réseau par l'approche active

paquets sur le lien commun, le lien propre au récepteur 1 et le lien propre au récepteur 2. Le taux de pertes mesuré par récepteur 1 seul est égal à $1-(1-p)(1-p_1)$ et par récepteur 2 seul est égal à $1-(1-p)(1-p_2)$. Par conséquence les observations individuelles n'ont aucune chance de calculer les taux de pertes des trois liens séparément. Il manque une équation pour pouvoir inverser le problème. C'est là que la corrélation des observations devient intéressante. Un paquet bien reçu en 1 et perdu en 2 (ou inversement) exclut automatiquement la perte du paquet sur le lien en commun, ce qui est une information précieuse. Ceci peut se traduire sous la forme d'un nouveau taux de pertes qui s'appelle le taux de pertes joint (paquet perdu par les deux récepteurs en même temps) et qui vaut : $p+(1-p)p_1p_2$. Nous avons alors trois équations que nous pouvons facilement inverser pour calculer les trois quantités manquantes.

Le principe de MINC peut facilement s'étendre à des réseaux plus compliqués. Il peut aussi s'appliquer au cas de l'inférence de la distribution du délai sur chacun des liens. Certains travaux ont essayé de l'étendre au cas unicast e.g. [36, 49]. Au lieu d'envoyer un paquet multicast qui est dupliqué par le routeur intermédiaire, deux paquets collés sont envoyés un au premier récepteur et un autre au deuxième récepteur. Le fait qu'ils soient collés ces deux paquets vont vivre très probablement la même expérience au niveau du routeur intermédiaire. En effet, si le routeur est dans un état de congestion, la perte des deux paquets est très probable. Il y a toujours la possibilité qu'un des deux paquets se perd et pas l'autre mais la probabilité de cet événement est considérée comme plus petite que la première.

Caractérisation de la topologie du réseau

L'Internet est construit autour de quelques grands opérateurs de transit (Tier-1) qui s'interconnectent entre eux et qui assurent la connectivité des autres opérateurs de niveau 2 (national ou régional). Ces derniers à leur tour offrent la connectivité aux entreprises et fournisseurs de service. Les opérateurs de n'importe quel niveau peuvent à leur tour et à tout moment singer des accords entre eux pour échanger leur propre trafic sans passer par les opérateurs de niveau supérieur (passage souvent payant) voire offrir les uns aux autres des solutions de backup en cas de problème. Ceci rend très compliqué et très imprévisible la topologie du réseau global que

personne ne contrôle et maitrise. Les opérateurs connaissent bien leurs domaines, à qui ils sont connectés et à qui ils offrent le service, mais au delà de ça tout ce qu'ils connaissent sur le reste de l'Internet ce sont les informations BGP qui indiquent la liste des domaines à parcourir pour atteindre un certain préfixe réseau. Cette liste peut même être incomplète dans le cas où un opérateur intermédiaire décide de ne pas annoncer les noms des domaines qui le suivent.

Ce manque d'informations sur la topologie du réseau a suscité la curiosité des chercheurs qui ont essayé de développer des solutions pour inférer la topologie de l'Internet dans le but de connaitre la connectivité globale et de localiser les adresses IP les unes par rapport aux autres. Ceci a diverses applications que ce soit pour les utilisateurs ou pour les fournisseurs des services ou même pour les opérateurs. Quelque soit le joueur, la connaissance de la topologie de l'Internet peut aider à optimiser les applications en choisissant mieux par exemple le serveur ou le pair à contacter ou le réseau à connecter. Ceci peut aussi servir à diagnostiquer le réseau et à comprendre le comportement des opérateurs. Finalement la connaissance de la topologie peut servir à développer des modèles théoriques qui permettent de générer des topologies des réseaux qui ressemblent le plus possible aux topologies réelles. Ceci est très important pour la précision des simulations et pour la validation des protocoles.

Dans la suite nous commentons les principaux travaux dans cette direction.

Caractérisation du graphe de connectivité C'est l'approche la plus classique. Elle consiste à tracer une carte pour l'Internet où les points sont les routeurs ou les domaines et où les arrêtes du graphe sont les liens les interconnectant. Pour la carte au niveau routeurs, elle peut être obtenue par des traceroutes envoyés par un ensemble de machines vers d'autres machines, ensuite par une représentation des chemins obtenus en un seul graphe e.g. [17, 35, 61, 104, 105]. Pour la carte au niveau domaines, elle peut être obtenue de deux manières : en agrégeant les tables de routage BGP prises dans différents routeurs de l'Internet [101], ou en partant d'une carte au niveau routeurs et en formant les domaines à partir des informations sur les associations (préfixe IP, nom du domaine) e.g. [79]. Mais quelque soit la nature de la carte, diverses difficultés l'accompagnent comme par exemple le nombre de points de mesure (tout en sachant qu'on ne peut pas en prendre autant qu'on en veut), la localisation de ces points de mesure, l'ordre par lequel les mesures sont à faire, la réduction de la redondance, la détection des alias (cas d'un domaine ou d'un routeur avec plusieurs noms et interfaces), etc.

Un des résultats les plus importants de ces travaux a été l'observation faite sur la distribution du degré de connectivité d'un noeud que ce soit routeur ou domaine. En effet, jusqu'à l'année 1999, cette distribution a toujours été supposée suivre une loi exponentielle au moins dans sa queue. En 1999, un travail pionnier par les trois frères Faloustas [41] a conclu que cette distribution suit une loi queue-lourde et par conséquence tous les modèles de graphes utilisés à cette époque pour la modélisation des réseaux comme le modèle ER, de Waxman ou GT-ITM, ont été considérés insuffisants pour refléter cette propriété. En effet, la propriété queue-lourde veut

tout simplement dire que certains noeuds de l'Internet sont fortement connectés par rapport à la moyenne ce que les modèles à queue exponentielle n'arrive pas à reproduire. Ceci a été à l'origine d'un grand nombre de travaux sur l'explication de la source de ce phénomène [13] et sur la proposition des nouveaux modèles théoriques pour la topologie du réseau comme par exemple le modèle *power law random graph* [3] et le modèle *general linear preference* [26]. Souvent, la théorie du *fort devient plus fort* est donnée comme explication à ce phénomène.

Plongement de l'Internet dans un espace virtuel de coordonnées Localiser les machines les unes par rapport aux autres à partir d'une carte n'est pas la solution la plus efficace. D'un côté il est difficile d'obtenir cette carte et de l'étendre à de nouvelles adresses ou de la mettre à jour, et de l'autre côté la carte est d'un grand volume qu'il est difficile de manipuler. Pour cela les efforts se sont concentrés sur la modélisation de l'Internet par un espace de coordonnées, souvent un espace Euclidien à plusieurs dimensions, et la modélisation des machines et des routeurs par des points dans cet espace. Les coordonnées des machines sont choisies de telle sorte que la distance géométrique entre elles dans l'espace est une approximation du délai réseau entre les machines. De cette manière il suffit de connaitre les coordonnées de deux machines pour pouvoir estimer le délai entre elles sans devoir aller sur une des machines et sonder l'autre. La grande question est sans doute si cette idée est faisable et si on peut obtenir les coordonnées d'une manière simple et efficace. Comme le délai satisfait en principe les propriétés d'une métrique, il y a forte chance que ca soit le cas. Ce problème est connu dans la littérature comme étant le problème de plongement des délais de l'Internet ou bien le problème de calcul des coordonnées virtuelles.

Plusieurs approches existent. Le premier travail a consisté en le déploiement d'une infrastructure de balises appelées *landmarks* [40]. Les balises mesurent le délai entre elles et formule un problème d'optimisation qu'elles résolvent pour trouver les meilleures coordonnées pour elles qui minimisent l'erreur quadratique moyen résultant du plongement de leurs délais. Ensuite, et étant données les coordonnées des balises, une machine quelconque les sonde et formule à son tour un problème d'optimisation non-linéaire pour trouver ses meilleures coordonnées qui minimisent l'erreur quadratique moyen résultant du plongement de ses délais dans l'espace. Cette deuxième tâche est réalisée parallèlement et indépendamment par les machines. En procédant comme ceci, toutes les machines réussissent à avoir leurs coordonnées et peuvent ensuite les utiliser dans les applications. Une utilisation typique serait par exemple d'envoyer une requête à un serveur DNS en demandant l'inversion d'une URL. Le serveur DNS au lieu d'envoyer une liste d'adresses IP, il effectue une estimation du délai entre la machine demandeuse et les différentes adresses IP de la même URL et renvoie par exemple l'adresse IP la plus proche.

Un des grands avantage des approches à base de balises comme [40, 96] est leur aspect non intrusif. A l'exception des balises, les machines ne se font pas sonder par d'autres machines dans le but de calculer les coordonnées. En plus, les travaux ont montré qu'un petit nombre de balises est nécessaire pour plonger les délais de l'Internet avec une bonne précision ce qui réduit

considérablement la charge comparée à une approche de type caractérisation totale de la carte du réseau. Cependant, la précision du plongement peut souffrir de certaines irrégularités dans le routage Internet qui ne peuvent pas être capturées dans un espace de coordonnées. Par exemple, une des propriétés d'une distance dans un espace est que la somme des distances entre noeud A et noeud B et noeud B et noeud C doit être supérieure à la distance entre A et C. Dans l'Internet malheureusement nous pouvons trouver des cas où cette règle n'est pas satisfaite et où le délai A-C est supérieur à la somme des délais de A-B et B-C. C'est le fameux problème de violation de l'inégalité triangulaire [40, 110]. La solution à ce problème est loin d'être évidente. Une possibilité serait d'introduire des noeuds intermédiaires modélisant les routeurs mais cette solution risque d'augmenter considérablement la complexité de l'opération. Néanmoins, même en l'absence de cette solution à l'inégalité triangulaire, les systèmes de coordonnées restent une idée intéressante puisque à un coût relativement faible ils peuvent donner une bonne idée sur la topologie du réseau et sur la proximité des machines.

Les coordonnées peuvent aussi se calculer par une approche complètement distribuée sans le besoin des balises. Cette idée a été introduite par Vivaldi [34], un système de coordonnées distribué dynamique très intéressant qui trouve son chemin au déploiement dans diverses applications comme le client BitTorrent de Azureus ou la plateforme OneLab, la version européenne de PlanetLab. Au lieu de calculer les coordonnées à partir des mesures de délai vers les balises, Vivaldi utilise des mesures de délai entre les machines elles mêmes pour l'ajustement des coordonnées. Chaque machine commence par une supposition sur la valeur de ses coordonnées, ensuite à chaque mesure de délai qu'elle effectue vers une autre machine, elle compare la distance Euclidienne au délai mesuré. Si la distance est plus grande que le délai mesuré, elle s'approche de l'autre machine. Dans le cas contraire elle s'éloigne d'elle, et à égalité elle reste à sa place. L'autre machine effectue la même opération de son côté. Et cette opération est répétée vers d'autres machines. Pour trouver le meilleur ajustement des coordonnées et prouver la convergence du système, Vivaldi a fait l'analogie entre un système de coordonnées et un système de ressorts où les machines sont les points d'attachement des ressorts et où les chemins entre machines correspondent aux ressorts mêmes. La longueur d'un ressort au repos représente le délai réseau du chemin. Accrochés dans l'espace aux points qui représentent les machines et lâchés, il est connu que les ressorts vont osciller dans toutes les directions jusqu'à converger vers un régime stationnaire qui minimise l'énergie totale. L'énergie totale n'est autre que la somme de l'erreur quadratique entre délais et distances Euclidiennes. Par conséquence, il suffit que les machines ajustent leurs coordonnées à la manière les ressorts le font pour converger vers un état optimal en termes d'erreur quadratique. Dans Vivaldi, une machine avance ou s'éloigne d'une autre machine avec un pas proportionnel à la différence entre délai mesuré et distance calculée.

Vivaldi est un système très intéressant qui a toutes les chances d'être déployé un jour comme service universel. Les mesures de délai peuvent se faire hors-bande dans un Overlay dédié, ou bien dans la bande en exploitant les communications en cours. Nous donnons beaucoup d'intérêt à ce

système dans notre recherche. Dans un récent travail [63], nous avons étudié la sécurité du système par l'évaluation des dégâts que des machines malveillantes peuvent causer en annonçant de fausses coordonnées ou bien en ne pas ajustant leurs coordonnées correctement. Nous avons proposé une solution à base de Surveyors (machines de confiance) pour vérifier le bon fonctionnement du système. Dans un autre travail en cours nous étudions comment un système comme Vivaldi peut servir dans la détection de tout changement topologique dans le réseau et dans l'identification des raisons de ce changement.

2.4 Conclusions

Nous avons voulu donné dans ce chapitre un aperçu général de l'évolution du domaine de métrologie au cours des dernières années. C'est un domaine qui a attiré beaucoup de recherches et va continuer à en attirer surtout avec la prolifération des attaques et du trafic indésirable et l'augmentation de la taille et l'hétérogénéité du réseau. Le grand défi pour la communauté scientifique va certainement être l'utilisation des résultats et des observations déjà faites dans le processus de construction d'un Internet de Futur plus sûr, plus adaptatif, et plus efficace que celui de nos jours. Un grand objectif pour la prochaine période est certainement la construction d'une architecture distribuée permettant l'observation de l'état du réseau et le diagnostic de ses problèmes. Cette architecture peut très bien faire partie de l'Internet du Futur en fournissant les informations nécessaires au bon fonctionnement du routage, du transport et des applications.

Dans ce domaine nous avons principalement contribué dans des solutions efficaces qui passent à l'échelle et qui permettent une meilleure compréhension des performances des réseaux. Les chapitres suivants donneront une idée de nos contributions sur l'approche passive, sur le problème d'échantillonnage et sur le contrôle de congestion pour les sondes actives. Pour les autres contributions que nous ne citons pas par manque d'espace, elles sont en grande partie portées sur la mesure des performances et la modélisation du protocole TCP e.g. [7, 16], la métrologie vue par les applications e.g. [77], la métrologie des réseaux sans fil e.g. [14, 29, 78] et la métrologie des applications multimédias e.g. [9, 95].

Chapitre 3

La métrologie passive

détaillé dans Annexe A

L'explosion de l'Internet en termes de taille (nombre de machines, routeurs et utilisateurs) et de trafic (volume de paquets, nombre de flots, nombre d'applications) pose pour les opérateurs un sérieux problème d'observation de ce réseau. Historiquement, les observations étaient limitées à une simple collecte de trafic en bits par seconde sur les différents liens du réseau par le protocole de gestion des réseaux SNMP [68]. Depuis, et pour diverses raisons comme la tarification, la détection des anomalies, et l'ingénierie de trafic, il y a eu fort besoin d'une observation un peu plus fine du réseau. Par exemple, il est devenu nécessaire de connaître le contenu du trafic et sa répartition entre les différentes applications et les différents utilisateurs ainsi que des statistiques qui permettent de mieux comprendre les performances du réseau et des applications des utilisateurs. Une observation fine, et au contraire d'une observation de haut niveau à la SNMP, demande de capturer les paquets, de regarder les valeurs des champs dans leurs entêtes et de mettre à jour des tables stockées dans la mémoire et contenant les statistiques sur le trafic. Une telle opération peut se dérouler sans problème pour des volumes de trafic pas très importants, mais quand le volume de trafic atteint de grandes valeurs de l'ordre de centaines de Mbps, cette observation détaillée des paquets et de leur contenu deviennent très difficile voire impossible pour deux raisons principales [37, 39, 94, 28, 15] :

- La difficulté d'un accès rapide à la mémoire pour mettre à jour les différents champs relatifs à la mesure du trafic.
- Le grand volume d'information généré par une mesure complète du trafic.

A ceci s'ajoute un autre défit lié au choix des points où effectuer les mesures [28, 18, 52, 60, 108, 86]. Prenons comme exemple la simple tâche de connaître à un instant donné tout le trafic qui va du site X au site Y. Si les paquets suivent un seul chemin, il suffit alors de mesurer le trafic en un seul point de ce chemin. Pour éviter de collecter d'autres trafics non intéressants, ce point est à choisir de telle manière que le trafic entre X et Y soit majoritaire. Il est évident que ce choix de l'emplacement des points de mesure devient de plus en plus compliqué avec la complexité de la tâche à effectuer et la taille du réseau. Dans la pratique plusieurs tâches de mesure peuvent être imaginées comme par exemple mesurer tout le trafic qui quitte un site, tout

le trafic destiné à un site, tout le trafic d'une application donnée, et comme cas extrême tout le trafic du réseau.

3.1 Échantillonnage temporel et spatial du réseau

Le problème qu'un opérateur confronte dans la mesure de son réseau est comment effectuer une tâche de mesure de la manière la plus précise possible étant donné le grand volume du trafic qui y circule (volume en octets, volume en octets/sec). D'après l'Introduction, il est clair qu'une observation du réseau à certains points bien choisis réduit considérablement le volume du trafic à collecter, à traiter et à stocker. Il reste cependant le problème de collecte de paquets aux points de mesure sélectionnés et l'adaptation de cette collecte au débit élevé du trafic. Le choix optimal des points de mesure n'est malheureusement pas suffisant pour résoudre ce problème.

Afin de pouvoir gérer le débit élevé du trafic en un point de mesure, une direction importante à suivre, qui est d'ailleurs souvent adoptée par les opérateurs et supportée par les constructeurs, est celle de l'échantillonnage temporel [37, 39, 94]. Plusieurs formes d'échantillonnage temporel peuvent être envisagées. L'idée de base de l'échantillonnage temporelle est de traiter une partie des paquets du trafic au lieu de tous les paquets. Le ratio des paquets collectés et traités est ce qu'on appelle le taux d'échantillonnage, souvent exprimé en pourcent. Pour un taux d'échantillonnage de $100.p\%$, une technique simple consiste à collecter un paquet avec une probabilité p et de l'ignorer (vis-à-vis de la métrologie) avec une probabilité $1-p$, et donc c'est seulement pour $100.p\%$ des paquets que la mémoire est accédée pour la mise à jour des champs contenant des statistiques sur le trafic. Ceci réduit par un facteur de $1/p$ le nombre d'accès mémoire, et réduit aussi le volume des résultats de mesure à stocker et à transporter à travers le réseau aux unités centrales chargées du traitement. Une autre manière de faire l'échantillonnage est de collecter et traiter un paquet tous les $1/p$ paquets. Il a été démontré que pour un trafic important stationnaire, toutes les techniques donnent pratiquement le même résultat [37]. La différence entre une technique et une autre est dans sa capacité à détecter des phénomènes transitoires et des activités périodiques ainsi que dans la facilité d'implémentation.

Une question qui se pose est comment une unité de mesure est capable d'appliquer l'échantillonnage à tous les paquets d'un trafic volumineux sans qu'elle soit capable de traiter tous ces paquets. En effet, l'échantillonnage se fait souvent au niveau physique (hardware) par des fonctions de hachage qui sont indépendantes du contenu des paquets. Cette opération est rapide et n'est pas bloquante pour une unité de mesure. La procédure bloquante est par contre l'accès à la mémoire (e.g., [39]) qui en cas d'échantillonnage n'est fait que pour les paquets sélectionnés d'où le gain important en vitesse (trois ordres de grandeur pour un taux d'échantillonnage de 0.001%).

L'échantillonnage est une technique bien connue dans d'autres domaines comme par exemple la théorie du signal et la théorie du sondage. A chaque opération d'échantillonnage correspond une

autre opération dite d'inversion [51]. En effet, seuls les paquets échantillonnés sont traités pour le but de la métrologie et les autres paquets sont seulement relayés sans traitement par l'unité de mesure. Les statistiques sur le trafic sont ensuite calculées à partir des paquets échantillonnés. La question fondamentale est combien ces statistiques sont fidèles au trafic original et comment pour une technique d'échantillonnage donnée et un taux d'échantillonnage donné, inférer les statistiques du trafic original à partir de celles du trafic échantillonné. La dernière opération est celle dite " inversion " et l'erreur qui l'accompagne est l'erreur causée par l'échantillonnage.

Pour certaines statistiques l'inversion est facile et l'erreur est faible voire négligeable [37, 51]. Ceci est le cas des statistiques calculées sur un grand nombre de paquets. Un exemple typique est de calculer le volume du trafic en octets ou en paquets, le nombre de paquets par protocole de transport, le nombre de paquets par application, etc. C'est le résultat intuitif de la loi des grands nombres qui permet d'inférer le volume d'une grande population à partir du volume d'une partie de cette population tirée au hasard. Il y a cependant des statistiques pour lesquelles l'opération d'inversion est difficile voire impossible à taux d'échantillonnage réduit. On trouve dans cette catégorie toutes les statistiques qui concernent un petit volume de paquets comme par exemple le volume des flots, la durée des flots, les phénomènes transitoires et anomalies, etc. La recherche dans ce domaine est grande ouverte et cible de plus en plus de statistiques avec leurs méthodes d'échantillonnage et d'inversion optimisées (e.g., [37, 28, 15, 51]). Dans le chapitre suivant nous illustrons la difficulté d'inversion sur une métrique simple mais d'un grand intérêt pour les opérateurs qui est celle de la détection des flots les plus gourmands et de leur classement les uns par rapport aux autres [15]. Nous montrons par des calculs de probabilité validés sur des traces réelles que cette inversion nécessite un taux d'échantillonnage élevé, au moins plus élevé que les taux d'échantillonnage actuellement pratiqués par les opérateurs. Dans le chapitre d'après, nous montrons sur une autre métrique comment l'impact de l'échantillonnage peut être réduit en tenant compte de l'auto-correlation qui puisse exister dans la quantité mesurée.

3.2 L'échantillonnage comme un problème joint

Un opérateur désirant observer son réseau possède donc deux marges de manoeuvre pour minimiser le volume de paquets à collecter et à traiter. D'un côté il peut faire de l'échantillonnage spatial pour mieux choisir les points de mesure. De l'autre côté il peut configurer le taux d'échantillonnage temporel en fonction de la capacité de ses unités de mesure et de la précision désirée. Ces deux dimensions peuvent être considérées séparément l'une de l'autre : mieux placer les points de mesure ensuite mieux choisir le taux d'échantillonnage. Elles peuvent aussi être combinées ensemble de telle façon à former un problème d'optimisation globale qui cherche comme solution le meilleur placement des points de mesure et le meilleur taux d'échantillonnage minimisant l'erreur d'échantillonnage, étant données bien sûr les contraintes du système en termes de capacité des points de mesure et de volume maximum du trafic à collecter. L'optimisation

jointe des deux problèmes a l'avantage d'identifier divers points de mesure faiblement chargés, au lieu d'un nombre faible de points de mesure qui sont surchargés par les opérations de mesure. Clairement, pour que cette optimisation jointe trouve son intérêt, l'opérateur doit disposer d'un grand nombre de points de mesure à activer et à programmer avec un taux d'échantillonnage optimal qui correspond à la tâche de mesure désirée. C'est une hypothèse qui est tout à fait réaliste étant donné qu'actuellement la plupart si ce n'est pas la totalité des routeurs du réseau d'un opérateur sont équipés par des outils de mesure qui échantillonnent le trafic et le rapportent aux unités centrales (l'exemple typique est les routeurs CISCO équipés par l'outil de mesure passive NetFlow [85]).

Dans la suite de cette section, nous allons considérer une tâche de mesure et nous allons calculer sa précision à partir du trafic échantillonné temporellement à tous les points de mesure disponibles dans le réseau. Notons cette précision par $M(p)$, où p est le vecteur des taux d'échantillonnage dans les différents points de mesure (ou routeurs). Le problème revient alors à trouver les meilleures valeurs de p qui maximisent $M(p)$.

Les contraintes sur le problème d'optimisation sont les suivantes. Premièrement, p est un vecteur à éléments positifs. Un élément de p égal à zéro veut tout simplement dire que le point de mesure en question n'est pas activé pour cette tâche de mesure. Deuxièmement, les taux d'échantillonnage dans les points de mesure (routeurs) ne doivent pas dépasser une certaine valeur décidée par le matériel. Cette borne supérieure peut théoriquement aller jusqu'à 100%. Troisièmement, le volume du trafic globalement collecté ne doit pas excéder une certaine valeur prédéfinie par l'opérateur en fonction de ses capacités de stockage et de transport. Le volume du trafic globalement collecté est simplement la somme du trafic collecté dans les différents points de mesure. Nous possédons alors un problème d'optimisation classique qu'un opérateur peut résoudre pour trouver le meilleur vecteur p étant donnée la tâche de mesure à effectuer. Si par exemple la tâche de mesure est l'estimation du trafic entre le site X et le site Y, l'opérateur peut :

- partir d'une estimation de ce trafic original,
- calculer pour un vecteur p quelconque la précision de l'inversion,
- trouver la meilleure valeur de p,
- configurer les points de mesure avec cette valeur de p et collecter du trafic pendant un certain temps,
- mesurer le volume du trafic échantillonné entre X et Y,
- ensuite trouver une meilleure estimation pour le volume du trafic original entre X et Y étant donné le volume du trafic collecté,
- retrouver une autre valeur optimale pour p en utilisant la nouvelle estimation du volume du trafic original entre X et Y,
- et ainsi de suite jusqu'à ce que le problème converge.

Nos expérimentations ont montré que très peu d'itérations sont suffisantes pour que le système converge vers la valeur désirée.

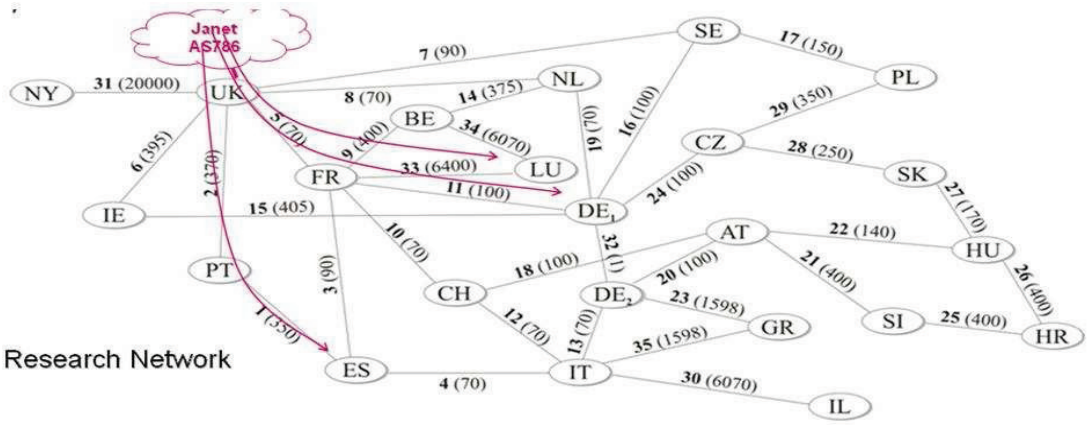

FIG. 3.1 – La topologie du réseau européen GEANT

3.3 Performances de l'optimisation jointe

Nous avons appliqué la méthode d'optimisation jointe au trafic collecté sur le réseau européen GEANT dont la topologie est illustrée dans Figure 3.1. Notre tâche consistait à estimer tout le trafic originaire du/destiné au réseau de recherche JANET au Royaume Uni. Ce trafic est la somme de tous les trafics échangés entre le royaume uni et chacun des autres pays européens. Chacun de ces trafics forme ce qu'on appelle un OD pair (orgin-destination pair) i.e. les lignes rouges dans Figure 3.1. Par exemple, UK-FR représente le trafic échangé entre le royaume uni et la France. Dans ce cas, la fonction $M(p)$ de notre problème d'optimisation n'est qu'une somme sur la précision d'estimation du volume de chacun de ces trafics OD. Pour évaluer la précision nous avons considéré l'erreur quadratique moyenne, voir Annexe A pour plus de détails. Nous avons ensuite tourné notre algorithme pour trouver les meilleurs taux d'échantillonnage dans les différents routeurs du réseau GEANT, et nous avons calculé la précision obtenue pour chacun des OD pairs. Tout ceci a été fait dans un simulateur alimenté par le vrai trafic du réseau GEANT avec la vraie topologie du réseau. Tous les routeurs du réseau étaient supposés implémenter des outils de mesure à taux d'échantillonnage configurable.

Pour bien comprendre les performances de notre méthode, nommée méthode-globale, nous l'avons comparé à une version simplifiée. Au contraire de la méthode-globale où nous faisons un échantillonnage à tous les routeurs du réseau GEANT, cette version simplifiée consiste simplement à observer le trafic aux interfaces du routeur d'accès du Royaume Uni (avec un taux d'échantillonnage optimisé localement). Notons cette version simplifiée par méthode-locale. Les résultats obtenus sont résumés dans la Figure 3.2.

La courbe "Average over all OD pairs" montre la précision moyenne sur tous les OD pairs pour la méthode-globale. Le "Best OD pair" et "Worst OD pair" montrent respectivement la meilleure et la plus mauvaise précision, aussi pour la méthode-globale. Les trois autres courbes montrent la même chose mais ceci pour la méthode-locale, la version simplifiée de la méthode-

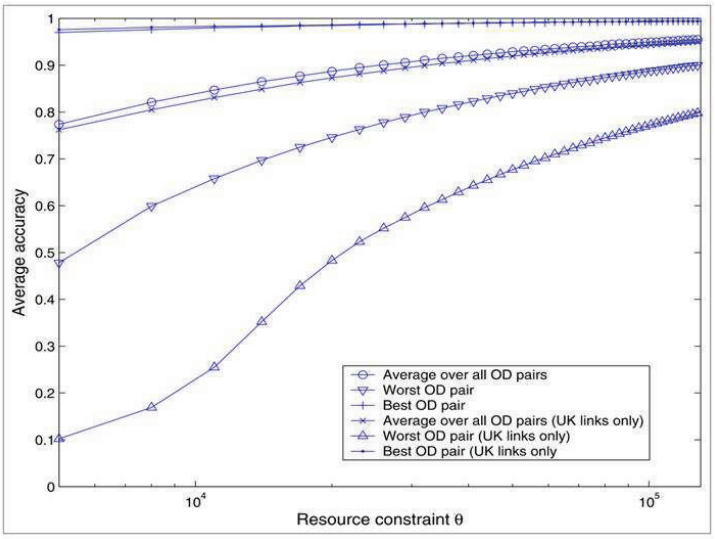

FIG. 3.2 – Les précisions (moyenne, pire et meilleure) obtenues sur tous les OD pairs

globale. Notez que l'axe des x dans la figure correspond à la contrainte sur le volume de trafic collecté en paquets/sec.

Nous voyons clairement dans la figure comment notre méthode-globale donne en moyenne d'excellentes performances pour différentes valeurs de la contrainte sur le trafic collecté. Cette précision moyenne est légèrement supérieure à celle obtenue par la méthode-locale où l'observation est faite seulement aux interfaces du routeur d'accès UK. Les deux méthodes globale et locale donnent pratiquement les mêmes performances pour les OD pairs ayant la meilleure précision. Cependant les résultats sont différents pour les OD pairs ayant les plus faibles précisions. En effet, un OD pair avec une faible précision est formé d'un trafic faible en volume pénalisé par une méthode de mesure simplifiée se limitant à très peu de points de mesure. En l'observant à des endroits chargés (comme les interfaces du routeur d'accès du royaume uni), le taux d'échantillonnage choisi est bas (parce que le trafic est grand) et donc ce petit OD pair souffre énormément. Les petits OD pairs ont besoin d'être observés sur des liens où ils sont seuls ou avec très peu d'autres trafics pour pouvoir augmenter le taux d'échantillonnage et améliorer la précision de leur métrologie, quelque chose que notre méthode-globale le permet.

Figure 3.2 montre aussi le résultat intuitif que la précision de notre méthode-globale s'améliore en relaxant la contrainte sur le volume du trafic collecté.

Concernant la convergence de notre algorithme en l'absence d'une connaissance précise sur le volume du trafic original, nos expérimentations ont montré qu'il suffit d'un très peu nombre d'itérations pour atteindre les bonnes valeurs pour les taux d'échantillonnage et par conséquence les meilleures estimations pour le trafic réseau. Dans Figure 3.3 nous montrons comment cette précision évalue en fonction du nombre d'itérations. L'axe des x dans la figure montre les différentes possibilités pour le taux d'échantillonnage au départ et l'axe des y la précision de l'estimation du

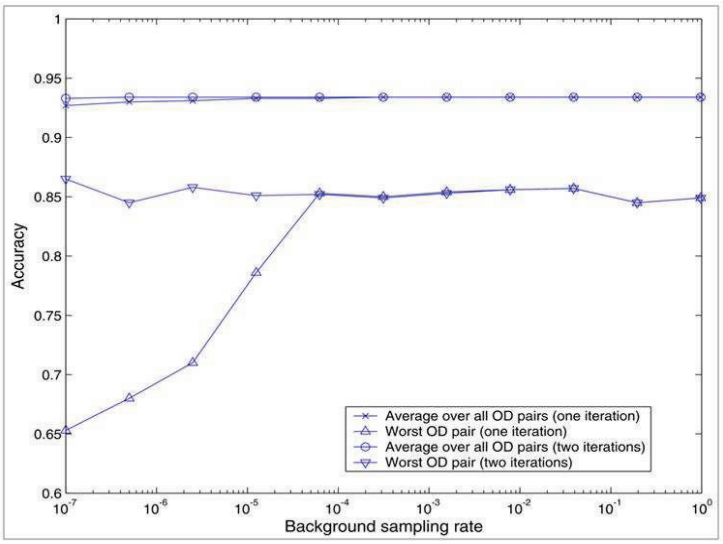

FIG. 3.3 – La précision en fonction du nombre d'itérations

volume de l'OD paire le plus difficile et la précision moyennée sur tous les OD pairs. Il est clair qu'à partir de deux itérations nous atteignons une précision indépendante du taux d'échantillonnage pris au départ. Il est probable que ce nombre d'itérations change d'un contexte à l'autre et d'une métrique à l'autre mais nous croyons qu'il reste toujours faible. Le prouver théoriquement fait partie de nos travaux futurs dans cette direction.

3.4 Conclusions

Dans ce chaptire, nous avons présenté notre architecture pour le choix du taux d'échantillonnage dans les routeurs d'un réseau afin d'optimiser une certaine tâche de mesure. L'architecture a montré son intérêt dans l'estimation du volume des OD pairs, même ceux les plus faibles, et ceci à volume du trafic collecté égal. Plus de détails sur ce travail se trouvent dans Annexe A. Nous poursuivons nos recherches dans cette direction et nous sommes en particulier intéressés par prouver l'efficacité de notre architecture pour d'autres tâches de mesure et par la convergence de l'algorithme en cas d'informations manquantes. Ces travaux se feront dans le cadre du projet européen ECODE qui démarrera en Septembre 2008.

Chapitre 4

Le bruit introduit par l'échantillonnage : détection et classement des flots volumineux

détaillé dans Annexe B

Notre architecture pour la métrologie passive et le choix des taux d'échantillonnage repose sur une fonction importante, celle de l'erreur qui accompagne l'inversion d'un trafic échantillonné. Dans le chapitre précédant nous avons montré l'intérêt de notre approche sur une métrique simple à inverser, celle du volume du trafic entre routeurs sources et routeurs destinations. En effet, la meilleure estimation du volume d'un trafic échantillonné peut être obtenue par une simple division par le taux d'échantillonnage et l'erreur commise est limitée et peut facilement être chiffrée. Cependant pour d'autres métriques l'exercice est loin d'être facile surtout quand ceci touche à des flots de faible volume comme par exemple lors de l'estimation du volume moyen des flots, du nombre de flots ou de la distribution des volumes des flots [37, 51, 99]. La difficulté est dans le fait que beaucoup de flots passent inaperçus à cause de leur faible volume. Par exemple pour un taux d'échantillonnage de 1/100 et un flot de 10 paquets (ce qui est la taille moyenne d'un flot dans l'Internet), la chance que ce flot soit capturé est d'environ 10% ce qui veut dire que 90% de ces petits flots passent inaperçus. La difficulté à connaitre leur nombre réside dans le fait que ceci nécessite la connaissance des volumes originaux des flots pour récompenser la perte causée par l'échantillonnage ce qui est souvent l'information à rechercher. Des solutions protocolaires comme par exemple l'investigation des entêtes des paquets à la recherche des paquets SYN qui annoncent le début des flots peut dans certaines cas améliorer la précision. La détection des anomalies souffrent aussi de l'échantillonnage. A l'exception des anomalies causées par une augmentation brusque du trafic, les anomalies en général sont causées par un grand nombre de petits flots qui visent à saturer un serveur ou à chercher un port disponible sur une machine. La détection se fait souvent par une identification d'augmentation dans le nombre de flots. Mais comme ils s'agit souvent de flots de faible volume, un grand nombre d'entre eux passent sans être vus et la capacité à détecter l'anomalie est alors réduite [24, 75]. Les méthodes basées sur le

calcul d'entropie et pas le nombre de flots ont montré une meilleure résistante à cet égard [24].

Dans ce chapitre nous illustrons l'impact de l'échantillonnage du trafic sur un exemple concret, celui de la détection et classement des flots volumineux dans un réseau (appelés souvent *heavy hitters*). Les operateurs s'intéressent à ces flots pour diverses raisons y compris l'ingénierie de trafic (en anglais *traffic engineering*), la facturation et la sécurité. Un flot peut être défini comme étant un simple transfert caractérisé par le fameux quintuple (adresse IP source, port source, adresse IP destination, port destination, protocole) comme il peut être défini par un préfixe réseau source ou destination (par exemple tous les paquets en provenance de tel préfixe /24 ou destinés à tel préfixe /24). C'est une application qui n'a pas été couvert dans la littérature comme telle puisque les operateurs considèrent souvent que les flots sont volumineux et par conséquence l'échantillonnage n'a pas un grand impact. Nous allons montrer que contrairement à cette pensée, le fait que les flots soient volumineux n'exclut pas les erreurs de détection et de classement. Le volume d'un flot volumineux est estimé avec une erreur relative plus faible mais en termes de valeur absolue, l'erreur est grande et par conséquence le flot peut facilement passer pour un autre flot volumineux si le taux d'échantillonnage n'est pas choisi convenablement. Malheureusement nous trouvons qu'un taux d'échantillonnage élevé (10% voire plus) est nécessaire pour détecter et classer les flots volumineux. Ce taux baisse d'un ordre de grandeur si seulement il s'agit d'une détection des flots volumineux sans classement ou si d'autres informations sont disponibles comme par exemple le numéro de séquence dans les entêtes des paquets.

4.1 Méthodologie

L'étude de l'échantillonnage nécessite souvent des outils théoriques pour pouvoir estimer l'erreur moyenne lors de l'inversion et le lier au taux d'échantillonnage. Les simulations peuvent aider dans cette étude mais étant donné le grand volume de données, elles deviennent très compliquées à mettre en oeuvre. Elles sont plutôt utilisées pour valider les résultats théoriques. Pour notre application nous sommes partis des hypothèses suivantes :

- Un grand nombre de flots dont le volume suit une certaine loi de probabilité. Pour les valeurs numériques, nous avons pris la loi de Pareto avec un volume moyen de 4.8 Kbytes pour les flots 5-tuple et 16.6 Kbytes pour les flots définis à base du préfixe destination /24. La loi de Pareto est connue pour autant une bonne représentation du volume des flots dans l'Internet et dans l'informatique en général [33].
- L'échantillonnage est supposé uniforme indépendant du contenu des paquets et d'un taux moyen p.
- Un operateur est intéressé par détecter et classer les t flots les plus volumineux. Cette hypothèse sera relaxée ultérieurement.

Pour évaluer l'impact de l'échantillonnage, il nous faut une mesure de performance. Nous la définissons comme ceci. Nous formons toutes les paires de flots dont le premier est un des t flots

volumineux et le deuxième un flot quelconque. A chacune de ces paires, nous associons un poids 1 si les deux flots sont mal classés l'un par rapport à l'autre après l'échantillonnage et un poids 0 s'ils sont bien classés. Notre mesure de performance est alors tout simplement la somme des poids de toutes les paires ainsi formées. Si la détection et classement se passent sans problème, alors la mesure de performance est égale à zéro, sinon sa valeur donne une idée sur l'importance du désordre dans la liste des t flots causé par l'échantillonnage. Par exemple une grande valeur veut simplement dire qu'un des flots volumineux est devenu trop petit suite à l'échantillonnage.

Cette mesure de performance a le grand avantage qu'elle peut se calculer analytiquement. Normalement chaque réalisation du processus d'échantillonnage donne lieu à une valeur pour cette métrique. Nous sommes intéressés par sa valeur moyenne sur toutes les réalisations. Pour cela nous procédons comme ceci :

- Nous calculons d'abord la probabilité qu'un flot de taille quelconque soit dans la liste des t flots les plus volumineux.
- Ensuite nous calculons la probabilité qu'un flot dans la liste des t flots soit mal classé par rapport à un autre flot de taille quelconque.
- Il suffit alors de multiplier cette probabilité par le nombre total de paires pour trouver la mesure qu'on cherche. Ce dernier nombre est simplement une fonction de t et du nombre de total des flots disponibles dans la trace. Pour un nombre total de flots égal à N, le nombre de paires est égal à $(2N - t - 1)t/2$.

La détection des t flots les plus volumineux sans le besoin de les classer les uns par rapport aux autres se fait pratiquement de la même manière. Il faut faire attention à ce que le nombre de paires par lequel il faut multiplier est maintenant $N.t$ et donc est inférieur au cas précédent.

4.2 Résultats numériques

Pour la validation, nous sommes partis des traces réelles et nous avons formé les flots dedans. Il s'agit des traces au niveau paquet collectées respectivement sur un lien du réseau américain ABILENE [1] et un lien d'accès du réseau français RENATER [98]. Pour chaque trace, nous avons calculé le nombre de flots, leur volume moyen et le paramètre de la loi Pareto qui décrit la queue (paramètre β). Ces valeurs ont été par la suite utilisées dans notre modèle pour trouver le nombre moyen de paires de flots mal classés, ce qui constitue notre mesure de performance pour la précision de la détection et du classement après échantillonnage. Cette opération a été répétée pour plusieurs valeurs du taux d'échantillonnage et pour plusieurs valeurs de t (le nombre de heavy hitters en question). En parallèle, nous avons appliqué l'échantillonnage directement sur la trace au niveau paquet et nous avons calculé notre métrique directement sans passer par le modèle. Tous les résultats sont décrits dans Annexe B dans des figures qui montrent l'erreur moyenne en fonction du taux d'échantillonnage sur une échelle log-log. Dans ce travail, nous avons considéré un grand nombre de combinaisons : classement vs. détection seule, flots TCP

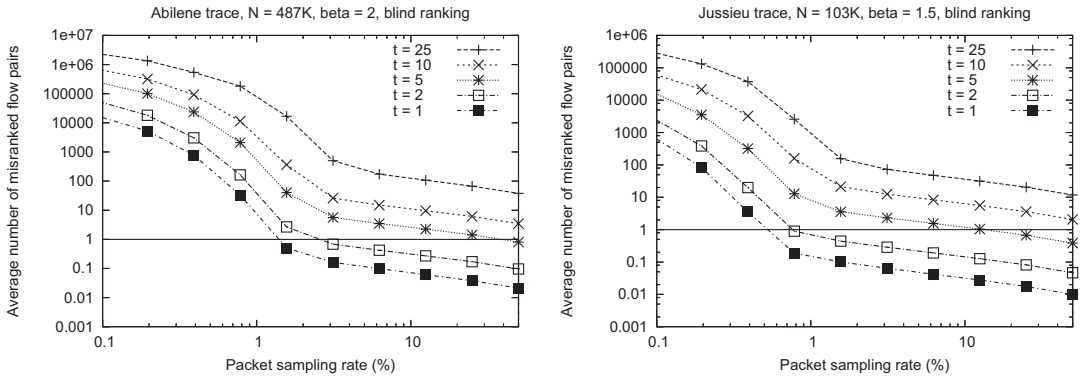

FIG. 4.1 – Erreur moyenne de détection et classement des flots volumineux

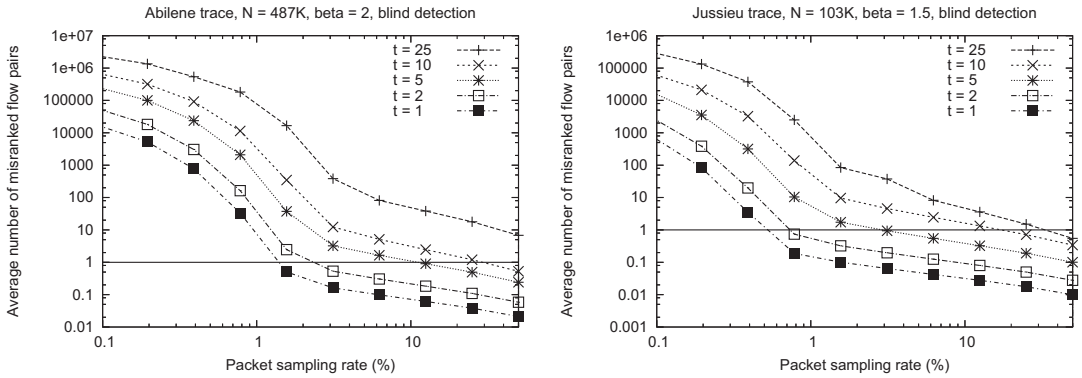

FIG. 4.2 – Erreur moyenne de détection seule des flots volumineux

vs. flots /24, différents nombres de flots, différentes valeurs pour le paramètre de la loi Pareto, etc. Figure 4.1 montre un exemple de ces résultats pour le cas de flots TCP et détection des flots volumineux. La figure à gauche correspond à la trace Abilene et celle à droite à la trace Renater. Figure 4.2 montre le même exemple mais dans le cas d'une détection seule sans classement. Nos conclusions peuvent se résumer comme suit :

- Un taux d'échantillonnage élevé est souvent requis et ce taux augmente avec le nombre de flots à détecter et/ou classer. Pour le flot ou les deux flots les plus volumineux un taux d'échantillonnage de l'ordre de 1% peut suffire à les détecter et les classer. Dès que ce nombre augmente, le taux d'échantillonnage requis augmente d'une façon exponentielle. Par exemple la détection et classement des dix flots les plus volumineux nécessite un taux d'échantillonnage de l'ordre de 50% voire plus. Ceci veut dire tout simplement l'inefficacité de l'échantillonnage pour ce genre d'applications.
- La détection seule réduit le taux d'échantillonnage requis d'un ordre de grandeur pour le même nombre de flots par rapport à la détection et classement. Mais ce taux reste quand même très élevé.
- Augmenter le nombre de flots augmente en général la précision puisque la chance d'avoir

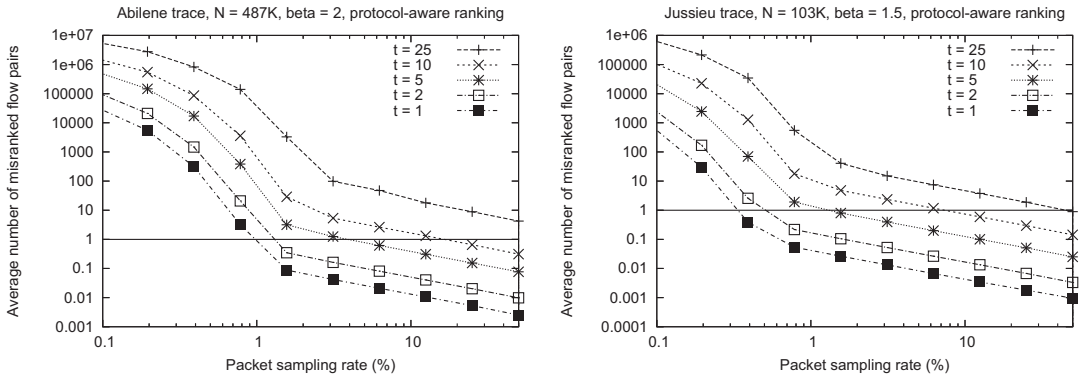

FIG. 4.3 – Erreur moyenne de classement des flots volumineux en utilisant le numéro de séquence de la couche transport

des flots encore plus volumineux augmente. Pratiquement toutes les lois de distribution de volume satisfont cette propriété. Les lois qui ne satisfont pas cette propriété sont celles qui tronquent le volume des flots volumineux.

– La précision augmente quand la queue de la loi de distribution du volume des flots devient plus lourde. Ceci est une conséquence naturelle du fait que les flots deviennent plus volumineux et plus espacés en termes de volume.

4.3 Nouvelle méthode de classement/détection basée sur des informations protocolaires

L'utilisation des informations protocolaires a toujours été considérée comme un moyen efficace pour réduire l'erreur causée par l'échantillonnage. Par exemple le fait de compter les paquets TCP qui portent le SYN flag et de supposer que chaque paquet SYN correspond à un nouveau flot permet une estimation rapide et efficace du nombre de flots dans le réseau. Dans cette même direction nous avons regardé si la connaissance du numéro de séquence disponible dans l'entête transport peut aider notre application. En soustrayant le numéro de séquence du premier paquet échantillonné de celui du dernier paquet, on trouve le nombre des paquets envoyés entre les deux et on utilise ce nombre pour le classement et la détection. L'erreur dans cette estimation vient des paquets non vus par l'échantillonnage au début et à la fin du flot. Apres ajustement du modèle à cette nouvelle méthode nous avons vu qu'elle était capable de réduire le taux d'échantillonnage d'un ordre de grandeur comparé à la méthode précédente. La Figure 4.3 le montre pour les deux traces et pour le cas détection et classement des flots volumineux.

Malgré son intérêt, cette méthode souffre malheureusement de plusieurs problèmes qui peuvent la mettre en cause. D'un côté, elle requiert l'accès à l'entête transport ce qui viole le principe des couches et par conséquence l'empêche de fonctionner en cas de paquets cryptés. De l'autre côté, elle présente des problèmes de fonctionnement en cas de retransmissions, mauvais ordonnance-

ment des paquets dans le réseau et réutilisation du numéro de séquence. Comme on le montre dans l'Annexe B, des algorithmes peuvent être introduits pour détecter ces derniers problèmes et réduire leur impact.

4.4 Conclusions

Dans ce chapitre nous avons montré sur un exemple concret l'impact de l'échantillonnage sur la qualité d'une tâche de mesure importante qui est celle de la détection et classement des flots volumineux. Nous avons mené une étude analytique accompagnée de simulations alimentées par des traces réelles. Cette étude faite sur un lien du réseau peut s'étendre à l'intégralité du réseau en profitant du cadre de travail introduit dans le chapitre précédant. C'est un de nos objectifs pour le futur. Un autre objectif est l'étude de l'impact de l'échantillonnage sur d'autres applications comme la détection d'anomalies ou la classification des applications à partir des données statistiques sur le trafic.

Chapitre 5

La corrélation temporelle au service de l'échantillonnage

détaillé dans Annexe C

La corrélation est une donnée importante dans la métrologie. Elle permet de tirer des conclusions importantes en comparant des mesures effectuées sur des quantités corrélées. Cette comparaison peut donner des informations sur la composante liant ces quantités corrélées, comme elle peut donner une idée sur la précision des mesures effectuées. Les applications sont diverses, nous citons les suivantes :

- *La compréhension de l'état du réseau à partir de mesures actives.* En effet, et comme dit dans les chapitres précédents, les sondes actives mesurent des chemins entiers. Et comme les chemins de l'Internet se chevauchent, il y a l'apparition d'une corrélation qui peut servir à mieux expliquer les résultats des mesures. Par exemple MINC [2] utilise cette propriété pour caractériser les liens d'un réseau à partir des mesures actives. On peut parler dans ce cas de corrélation spatiale. Le reste des applications exploitent la corrélation temporelle.
- *La prédiction* qui permet à partir des mesures actuelles et des informations sur la corrélation d'estimer et de prédire le futur. Ceci permet d'anticiper avec les réactions convenables. Le *trafic engineering* (l'ingénierie du trafic) effectué par les opérateurs peut profiter de cette application en partant des modèles bien connus pour le trafic e.g. [19, 70].
- *La détection d'anomalies* qui utilise des techniques similaires à celle de la prédiction. Pour la prédiction, il s'agit de trouver la valeur la plus probable dans le futur. Pour la détection d'anomalies, il s'agit de trouver la valeur la plus probable pour le présent et de la comparer à la valeur mesurée. Avec une idée sur le taux d'erreur moyen calculé pendant une phase de calibrage, il devient alors facile de comparer la valeur mesurée à la valeur estimée et de conclure s'il s'agit d'une grande déviation ou pas. Une grande déviation est assimilée à une anomalie. Cette méthode peut s'appliquer au trafic de l'Internet pour détecter les attaques contre les machines [25, 68] comme elle peut s'appliquer à tout autre système pouvant présenter un comportement anormal. Un exemple typique est notre contribution récente sur la détection des attaques contre les réseaux de coordonnées virtuelles [63].

– *La correction des mesures.* Le filtre de Kalman est l'exemple typique d'une telle application [64]. Chaque opération de mesure est généralement accompagnée d'une certaine erreur. Cette erreur peut être causée par un bruit extérieur comme c'est le cas lors de la mesure de la bande passante. Mais l'erreur peut aussi venir du fait qu'on observe une partie du système et qu'on essaie d'inférer le global à partir d'observations partielles. L'échantillonnage du trafic appartient à cette deuxième catégorie. Pour éliminer le bruit et corriger les mesures, deux classes de méthodes existent. La première consiste à répéter la même mesure plusieurs fois et à procéder par des méthodes statistiques pour ne garder que les valeurs les plus probables ou les moyennes. Cette méthode s'applique souvent à des populations qui ne varient pas en fonction de temps. L'autre méthode, qui est celle adoptée par le filtre de Kalman et les autres filtres linéaires ou non-linéaires, consiste à profiter de la corrélation du système à mesurer pour corriger les mesures. A la différence de la première méthode, la deuxième méthode profite des mesures précédentes pour améliorer la précision de la mesure actuelle. L'idée est simple. Quand une métrique est corrélée avec elle même (auto-correlation) ou avec une autre métrique, sa valeur est en partie décidée par les valeurs passées, et ceci dépend de l'importance de la corrélation. Par exemple dans le cas d'une corrélation positive, si le passé est grand il y a forte chance que le présent soit aussi grand et l'inverse est aussi vrai. Par conséquence, le fait de connaître les valeurs passées aide à mieux estimer la valeur de la métrique dans le présent comparé au cas où on ne connaît rien sur le passé et où le seul moyen disponible est d'estimer la métrique par sa valeur moyenne ou sa distribution marginale. Souvent les outils procèdent comme suit :
  – A partir des mesures et des estimations précédentes et de la corrélation, estimer une valeur pour le présent.
  – Mesurer le présent et raffiner l'estimation en prenant une partie du présent et une partie du passé. Le poids entre le passé et le présent est parfois appelé le gain de l'estimateur et dépend de l'objectif à atteindre (minimiser l'erreur quadratique moyenne, éliminer certaines fréquences et en garder d'autres, etc).
  – Ajouter la mesure et estimation du présent à l'historique et revenir au point 1.

  Pour mieux comprendre cette méthode, prenons le cas simple de l'estimation du délai-aller retour moyen dans TCP (souvent appelé RTT pour Round-Trip Time). Le RTT est une métrique qui varie à deux échelles de temps : Une petite échelle de temps où la variation peut être considérée comme aléatoire non corrélée causée par la nature paquet des réseaux. Et une autre grande échelle de temps décidée par l'évolution de la congestion dans le réseau et qui introduit une certaine corrélation dans le processus. TCP vise à calculer une valeur moyenne du RTT qui suit la partie corrélée. Pour cela il procède par l'utilisation d'un filtre passe bas de type moyenne glissante avec poids exponentiel (Exponentially Weighted Moving Average) [56]. Lors de chaque nouvelle mesure de RTT l'estimateur du RTT moyen forme une nouvelle estimation en prenant le 7/8 de l'ancienne estimation et le 1/8 de la

mesure actuelle et en calculant ensuite la somme. Ceci permet d'éliminer les oscillations de courtes périodes et de garder celles de longues périodes. Si cette corrélation n'existait pas, TCP aurait fini par calculer une valeur moyenne simple qui ne change pas beaucoup en fonction de temps et que si elle change, son changement est beaucoup moins significatif. Notez ici que la meilleure estimation d'un processus non corrélé est sa valeur moyenne, une valeur qui peut être bien loin des valeurs mesurées surtout si le processus présente une forte variabilité. Les processus corrélés se prêtent mieux à l'estimation.

Il s'agit d'un domaine très important qui se situe à la limite de la théorie et de la pratique. Des outils sophistiqués existent dont la théorie du signal, des processus stochastiques, du contrôle et d'autres, et qui peuvent avoir un grand intérêt lors de la conception et optimisation des solutions efficaces pour la métrologie. De notre coté nous avons contribué dans ce domaine à travers trois applications, que nous nous contentons dans ce chapitre d'en expliquer une. Les deux autres applications traitent respectivement de l'estimation du délai dans les playout buffers des applications audio [95] et de l'estimation des coordonnées nominales d'une machine dans un réseau de coordonnées soumis à des attaques [63]. Les coordonnées nominales permettent après de détecter si des machines intentionnellement annoncent des coordonnées très distantes de ce qu'elles devraient faire.

5.1 Estimation de la taille de grandes populations

Nous avons étudié ce problème dans le cadre d'applications de diffusion qui désirent connaître à tout instant le nombre de personnes qui les reçoivent. C'est le cas par exemple d'un serveur diffusant de la vidéo ou de la télé en multicast avec du RTP à un grand nombre de participants. La diffusion peut se faire sur des satellites ou sur des lignes ADSL. Cette connaissance du nombre de participants en temps réel peut servir à évaluer la popularité des différents canaux de diffusion (e.g. chaines TV) et à mieux dimensionner le réseau et les services par conséquence. Dans certains cas elle peut aussi servir à équilibrer redondance et retransmissions pour assurer la fiabilité de la diffusion (plus de participants c'est plus de retransmissions et par conséquence plus de charge ce qui pourrait justifier l'utilisation de la redondance). Cependant d'autres applications peuvent aussi profiter du calcul de la taille de grandes populations. Nous citons par exemple le calcul du nombre de pairs qui participent à un partage de contenu avec BitTorrent (e.g. le cas des utilisateurs mettant à jour leur version de MS Windows), le calcul du nombre de flots sur un lien ou dans un réseau, le calcul du nombre de machines qui sont réveillées dans une grande plateforme, etc.

Dans notre travail nous étions confrontés au grand volume de données qui peut résulter d'un calcul exact de la taille d'une grande population. Un calcul exact nécessite que chaque machine envoie périodiquement un message "hello" à la source et que la source fasse la somme de tous ces messages. Ceci peut causer une implosion à la source quand les récepteurs sont nombreux. Une

TAB. 5.1 – Description des traces vidéos

Trace	Longueur de la trace	ρ	$1/\mu(s)$
$video_1$	$3^d\,13^h\,33^m\,20^s$	94.7	18316
$video_2$	$11^d\ \ 1^h\,46^m\ \ 8^s$	14.1	16476
$video_3$	$50^d\,22^h\,13^m\,20^s$	8.1	66823
$video_4$	$29^d\,16^h\,43^m\,13^s$	17.9	83390

solution pourrait être l'espacement des messages dans le temps mais ceci conduit à un manque de précision causé par l'évolution temporelle de la population. Afin de résoudre ce problème, nous sommes passés par une solution probabiliste dans laquelle les récepteurs sont demandés d'envoyer périodiquement leurs messages "HELLO" mais avec une certaine probabilité qui peut être choisie en fonction de la précision voulue et de la taille de la population. Appelons cette probabilité p et appelons S la période d'envoi. Alors toutes les S secondes, chaque récepteur tire de son coté un nombre aléatoire entre 0 et 1 et envoie son message "HELLO" si ce nombre tiré est inférieur à p (ceci est équivalent à ce qu'on appelle envoi avec probabilité p). En procédant comme ceci, le nombre de messages envoyés est réduit par un facteur de $1/p$. La question à comprendre est quel impact a cette réduction sur la précision de l'estimation et comment faire pour inverser le problème avec le minimum d'erreur. Notre travail publié dans [5, 6] et détaillé dans Annexe C essaie de répondre à cette question. Il s'agit d'un problème typique d'échantillonnage dont l'inversion peut profiter pleinement de l'auto-corrélation dans les variations de la taille de la population. Dans ce cas l'auto-corrélation vient simplement du fait que les participants restent un temps non négligeable dans la session et par conséquence ceux qui sont présents à un instant donné affectent certainement le nombre total de participants à tout autre instant précédant leur départ.

5.2 Modèle et Analyse

Avant de présenter le modèle parlons des traces que nous avons utilisées pour la validation. Il s'agit de quatre traces vidéos collectés sur le MBone [4] entre Août et Septembre 2001. Les traces ont des durées allant de quelques jours pour l'une à une vingtaine de jours pour l'autre. Nous connaissons pour chaque trace exactement le nombre de participants, leurs instants d'arrivée et leurs instants de départ. Appelons λ le taux moyen d'arrivée des participants par trace, $1/\mu$ le temps moyen qu'il passe dans la session par trace, et $\rho = \lambda/\mu$ la charge. Il est connu d'après le théorème de Little [112] que ρ représente le nombre moyen de participants présents simultanément dans chaque session. Dans la suite appelons $N(t)$ le nombre de participants à un instant donné t (ou la taille de la population). Le Tableau 5.1 donne les valeurs de ces différents paramètres que nous avons calculées pour les différentes traces.

Le problème posé est alors comment estimer la taille de la population $N(t)$ à partir du sous-

ensemble des messages reçus. Prenons un instant $t_n = nS$ et appelons Y_n le nombre de messages "HELLO" reçus à cet instant. Rappelons que chaque récepteur envoie un message ou non avec une probabilité p. La solution la plus simple serait de diviser le nombre de messages reçus par p et de prendre $N(t_n) = Y_n/p$. C'est la solution qui ne tient pas compte de la corrélation. Elle fonctionne, voire même elle est la meilleure, quand la population change tellement vite que la corrélation disparaisse entre les instants de mesure. Par contre en présence de la corrélation, elle est loin d'être l'optimale puisque quelqu'un peut profiter des estimations faites dans le passé pour le calcul d'une meilleure estimation dans le présent. Le gain serait à la fois une meilleure précision et une plus petite valeur de p. Une manière pour tenir compte de la corrélation serait l'utilisation de ce qu'on appelle un filtre de moyenne glissante avec poids exponentiel (Exponentially Weighted Moving Average EWMA). C'est un filtre passe-bas très connu utilisé dans divers protocoles de l'Internet particulièrement le protocole TCP [56] pour lisser des quantités variables. L'idée serait d'estimer $N(t_n)$ en partie en utilisant Y_n/p et en une autre partie en utilisant l'ancienne estimation $N(t_{n-1})$. Un paramètre $\alpha \in [0, 1]$ est utilisé pour définir l'importance d'une partie sur l'autre. Ceci donne lieu à un estimateur de cette forme : $\hat{N}(t_n) = \alpha\hat{N}(t_{n-1}) + (1 - \alpha)Y_n/p$. En augmentant la valeur de α de 0 à 1, cet estimateur donne plus d'importance au passé au déterminant du présent. Bien évidement il y a une valeur optimale de α et cette valeur optimale dépend de l'importance de la corrélation. Pour un processus fortement corrélé il faut prendre une valeur de α proche de 1, cependant pour un processus non corrélé, α doit être proche de 0. Dans ce dernier cas il y n'a aucun gain dans la considération de l'historique de la taille de la population. Figure 5.1 montre sur une trace collectée lors d'une session audio comment l'estimateur EWMA se comporte pour différentes valeurs de α. Ce genre d'estimateur est très intéressant mais il a deux problèmes majeurs : (*i*) d'un coté il est incapable d'estimer des systèmes qui présentent une dynamique autre que linéaire auto-regressive d'ordre 1 (i.e. seule dépendance de l'instant précédant), (*ii*) et de l'autre côté il a le problème du choix de α, un choix qui dépend de la variabilité et de l'auto-corrélation de la taille de la population.

Dans ce travail, nous sommes allés plus loin en cherchant le meilleur estimateur linéaire qui tient compte de la corrélation dans le processus de la taille de la population. Un estimateur linéaire a la propriété que l'estimation de la taille de la population à l'instant actuel est une fonction linéaire de la mesure actuelle, des estimations dans le passé et des mesures dans le passé. WIENER e.g. [50] a développé une théorie pour le calcul de cet estimateur optimal. L'optimalité est en termes d'erreur quadratique moyenne. Le résultat majeur de la théorie de WIENER est que il suffit de connaître l'auto-corrélation du processus à estimer, i.e., $E[N_n N_{n-k}]$, et la covariance entre les mesures et le processus à estimer, i.e., $E[N_n Y_{n-k}]$, pour pouvoir trouver le meilleur estimateur linéaire. Le calcul cependant est compliqué et nécessite le plus souvent des hypothèses fortes pour pouvoir parvenir à des expressions explicites. Nous avons fait pour cela l'hypothèse que les membres arrivent suivant un processus Poisson d'intensité constante λ et que chacun reste dans le système un temps exponentiellement distribué de valeur moyenne $1/\mu$. Ces hypothèses

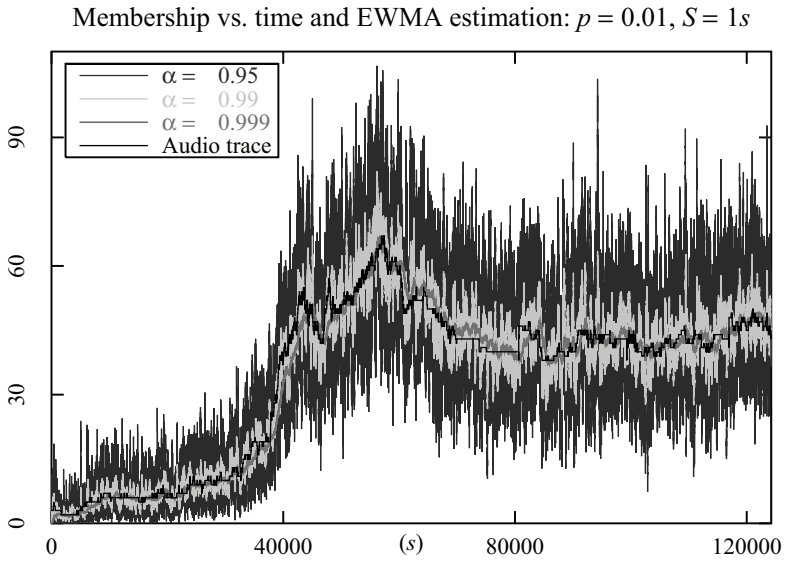

FIG. 5.1 – Evolution de la taille de la population et estimation EWMA

ne sont pas souvent valides, mais malgré cela, elles permettent d'avoir un estimateur sous forme explicite et une grande précision d'estimation comme nous allons le voir sur certaines traces. Il reste qu'à calculer l'auto-corrélation du processus de la taille de la population qui est bien connue dans la théorie des files d'attente : $Cov(N_n, N_{n-k}) = (\lambda/\mu)e^{-\mu k S}$. La corrélation entre le processus des mesures et le processus à estimer quant à elle et sous nos hypothèses d'uniformité et d'indépendance dans les réponses, peut s'écrire comme suit : $Cov(N_n, Y_{n-k}) = p^2 Cov(N_n, N_{n-k})$ pour $k \neq 0$ et $Cov(N_n, Y_{n-k}) = pCov(N_n, N_{n-k})$ pour $k = 0$. Un simple plongement de ces résultats dans les formules dérivées par WIENER (voir Annexe C pour plus de détails) permet de trouver la meilleure forme linéaire pour notre estimateur de la taille de la population ainsi que les meilleurs coefficients :

$$\hat{N}_n = A\hat{N}_{n-1} + BY_n + (\lambda/\mu)(1 - A - pB).$$

A et B sont deux coefficients constants fonction des paramètres du modèle et dont les expressions se trouvent dans Annexe C. Sous nos hypothèses, cet estimateur est le meilleur parmi tous les estimateurs linéaires. Il est clairement autorégressif d'ordre 1.

Notez bien qu'au lieu de chercher le meilleur estimateur linéaire, quelqu'un peut partir d'un estimateur de forme particulière, par exemple le filtre de Kalman comme nous l'avons fait dans [6], et essayer de trouver les meilleurs paramètres de l'estimateur qui répondent à certain objectif comme par exemple la minimisation de l'erreur quadratique moyenne.

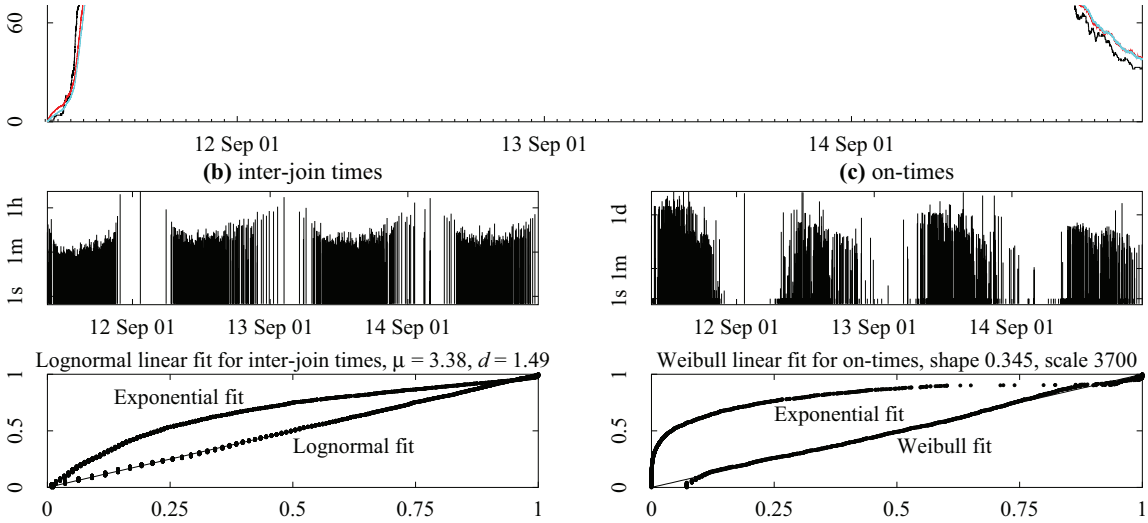

FIG. 5.2 – Performance de l'estimateur plus statistiques sur les connexions des membres

5.3 Validation et application

Sur nos quatre traces citées ci-dessus, nous avons vérifié la précision de notre estimateur. Les résultats se trouvent dans l'annexe, nous nous contentons ici de présenter les résultats pour la première trace. Figure 5.2 montre comment notre estimateur est capable de bien traquer la taille de la population malgré le fait que les statistiques sur les participants ne correspondent pas vraiment aux hypothèses faites dans le calcul (les deux petites figures en bas). Nous notons en particulier le faible volume des messages HELLO envoyés ; c'est une population qui fait environ et en moyenne une taille de 100 et p est choisi de telle façon qu'un message HELLO est envoyé toutes les S seconds ($S = 2.5$ secondes dans la figure). Les autres figures montrent des comportements similaires et une erreur relative moyenne autour de 4%.

Dans la pratique et afin d'utiliser notre estimateur, il faut l'accompagner par une autre estimation du taux d'arrivée des membres et de la durée moyenne de leur inscription. Ceci peut se faire en demandant à chaque membre d'envoyer dans son message HELLO des statistiques sur son temps de participation. Apres, ces statistiques peuvent être combinées ensemble pour former le temps de participation moyen. Le taux d'arrivée λ n'est autre que la taille moyenne de la population estimée divisée par le temps moyen de participation. Une autre solution pourrait être l'envoi par chaque nouveau participant à son arrivée et à son départ un message d'abonnement et de désabonnement avec une probabilité fixe bien définie. De cette manière les paramètres

peuvent être estimés à partir de ses statistiques. Mais que ce soit pour cette solution ou l'autre, il s'agit encore des pistes ouvertes à explorer. Le plus dur sera de combiner l'estimation de ces paramètres avec l'estimation de la taille de la population elle même pour former un problème joint calculant le tout à partir des messages HELLO seulement.

5.4 Conclusions

Dans ce chapitre nous avons montré sur un exemple concret l'intérêt de la corrélation dans l'estimation et la prédiction. La difficulté réside dans le fait que la considération de la corrélation demande des statistiques sur le processus à estimer lui même, des statistiques plus difficiles à obtenir que dans le cas sans corrélation. C'est toute la difficulté du problème. Le mieux est de passer par un modèle du système qui permet de réduire le nombre de paramètres (comme dans notre cas la réduction en deux paramètres λ et μ), ensuite l'estimation de ces paramètres par une procédure auxiliaire. L'utilisation d'une estimation jointe de tous les paramètres est dans certains cas possibles mais demande beaucoup d'effort à mettre en oeuvre.

Chapitre 6

La métrologie active

détaillé dans Annexe D

Un des problèmes avec l'approche active de la métrologie est la charge qu'elle cause sur le réseau qui dans certains cas peut fausser les résultats des mesures. Par exemple, dans le cas d'un collecteur qui envoie des traceroutes pour découvrir la topologie du réseau, le nombre des traceroutes générés et les messages reçus en retour peuvent être à l'origine d'une congestion autour du collecteur ou dans le coeur du réseau et d'une perte des paquets de mesure. Dans la pratique, les traceroutes sont envoyés à un débit constant et faible pour ne pas congestionner le réseau [104]. La question est toujours comment régler ce débit des traceroutes et comment s'assurer qu'il n'est pas trop faible pour ralentir la mesure. Le but est souvent d'avoir une mesure du réseau le plus rapidement possible avant que son état change. Le même problème existe dans d'autres applications comme par exemple le *ping* d'un grand nombre de machines pour l'inférence de leur état ou bien la collecte des statistiques d'un grand nombre de machines réparties à travers le réseau par une approche active. Quant à la congestion que certains outils de mesure causent sur un chemin spécifique de l'Internet comme par exemple lors de la mesure de la bande passante disponible ou le délai, cette congestion peut se régler avec des protocoles simples à la TCP ou la TFRC [45] quand c'est nécessaire. Dans ce dernier cas il faut faire attention à ce que le contrôle de congestion introduit ne perturbe pas la mesure. Certains outils comme *pathchirp* [100] génèrent les paquets avec un profile bien défini que l'introduction d'une boucle de contrôle de bout-en-bout peut altérer leur fonctionnement et par conséquence fausser leurs résultats.

Dans un travail récent [102] nous nous sommes penchés sur le problème après avoir remarqué l'absence d'un protocole de contrôle de congestion et de flux qui permet de régler le débit des sondes actives envoyées dans le réseau par un collecteur donné à un grand nombre de machines. Le mot *collecteur* dans la suite est utilisé pour designer une machine sondant le réseau et collectant les résultats pour un usage ultérieur. L'utilisation de TCP dans ce contexte n'est pas efficace étant donné le petit volume de paquets échangés entre le collecteur et chacune des machines. En plus, le problème du choix du nombre des connexions TCP à établir et de la façon avec laquelle elles sont lancées se pose. Il y a alors besoin de trouver un nouveau protocole qui permet de régler

le débit de ces sondes actives en fonction de l'état du réseau, et de s'assurer que les sondes ont bien atteint leurs destinations et que les réponses sont bien reçues par le collecteur. En cas de non réponse, il faut s'assurer qu'une sonde est renvoyée jusqu'à la réception de la réponse ou jusqu'à ce que la décision que la machine destinatrice n'est pas disponible soit prise. Nous voulons éviter des solutions qui impliquent le réseau dans la procédure, e.g., [27], ou bien celles qui peuvent se contenter d'un petit ensemble de réponses (e.g., cas des solutions qui cherchent à savoir si une machine n'a pas reçu une certaine information sans connaitre l'identité de la machine [87]). Nous visons une solution qui :

- soit basée sur le principe de bout-en-bout suivant lequel le réseau n'est pas impliqué.
- contrôle la congestion du réseau et retransmet les sondes en cas de non réception de l'information.
- amical avec le trafic Internet existant, ce dernier passant pour la plupart de temps par l'intermédiaire de TCP.
- soit générale indépendante de la nature de l'information à collecter des machines sondées. Tout comme TCP a été conçu pour récupérer toute sorte de données, nous voulons que notre solution soit capable d'effectuer toute sorte de sondage et de récupérer toute sorte d'informations de machines distribuées à travers l'Internet sans le devoir d'établir une connexion TCP avec chacune de ces machines.
- soit fiable dans le sens que toutes les sondes réussissent. Ceci ne doit pas empêcher le changement du niveau de fiabilité si besoin.

En vue de toutes ses conditions, nous avons conçu notre solution en s'inspirant largement des mécanismes de TCP mais en les élargissant pour qu'ils puissent fonctionner dans ce cas particulier du point-à-multipoint. La solution est appelée TICP pour TCP-friendly Information Collection Protocol. TICP a été conçu comme un protocole transport avec une version collecteur à installer dans la machine qui sonde et une autre version toute simple qui s'appelle *source* et qui s'installe dans toute machine désirant participer à la session. Dans TICP, toute l'intelligence se trouve au niveau collecteur qui lui assure le contrôle de congestion et le contrôle d'erreurs. La source se contente simplement de répondre aux sondes du collecteur en envoyant immédiatement ses données. Dans certains cas la présence de TICP côté source n'est pas nécessaire si la source est déjà doté d'un mécanisme qui lui permet de répondre aux sondes. L'exemple d'un collecteur TICP envoyant des pings à des machines répondant par ICMP en est un bon exemple. Dans ce chapitre nous présentons une brève description de TICP et de ces mécanismes. Les détails se trouvent dans [102] ainsi que dans Annexe D.

6.1 Le contrôle de congestion et le contrôle d'erreurs dans TICP

Lors de la conception de TICP nous nous sommes largement inspirés du protocole TCP [56]. Tout comme dans TCP, le contrôle de congestion dans TICP se fait à base de fenêtre glissante,

à la différence que la fenêtre de TICP est mesurée en nombre de sondes envoyés dans le réseau et pour lesquelles une réponse n'est pas encore parvenue. La fenêtre augmente tant que les réponses parviennent correctement et est divisée par deux comme dans TCP quand des réponses ne parviennent pas. La fenêtre augmente exponentiellement au début de la session suivant une phase *Slow Start*, et passe à la phase *Congestion Avoidance* après le premier signal indiquant la congestion du réseau.
Le contrôle de congestion et le contrôle d'erreurs dans TICP reposent sur les principes suivants :

Estimation d'une borne supérieure sur le temps aller-retour (RTO)

Tout comme dans TCP, le protocole TICP mesure le délai aller-retour réseau et calcule ainsi une estimation du délai moyen et de la déviation moyenne du délai. La borne supérieure sur le temps aller-retour est estimée comme la valeur moyenne plus quatre fois la déviation moyenne. Cette borne supérieure RTO est une estimation conservatrice sans excès du temps que prend une sonde pour atteindre sa destination et pour que la réponse correspondante atteigne le collecteur. Une réponse qui n'arrive pas pendant ce temps RTO est une bonne indication de la perte de la sonde ou bien de la réponse sur le chemin de retour.

Congestion du réseau

Le collecteur dans TICP, source de sondes, envoie autant de sondes que la fenêtre le permet. Les réponses correspondantes à une fenêtre de sondes sont supposées arriver pendant un temps égal à RTO. De ce fait, le collecteur TICP mesure le nombre de réponses reçues pendant le prochain RTO suite à une fenêtre de sondes envoyées dans le présent RTO et calcule ainsi le taux de pertes de sondes/réponses. Si ce taux est plus élevé qu'un certain seuil, le réseau est considéré comme congestionné et TICP divise alors sa fenêtre par deux. Si ce rapport est plus élevé qu'un autre seuil plus important, le réseau est considéré comme sévèrement congestionné et TICP passe en mode Slow Start.

Retransmission d'erreurs

TICP est supposée connaître la liste des machines à sonder. Cette liste est parcourue d'une façon cyclique. Dans le premier cycle les machines sont sondées successivement. L'ordre suivant lequel elles sont sondées est discuté dans la section suivante. Dans le deuxième cycle les machines qui n'ont pas répondu lors du premier cycle sont sondées. Le sondage continue jusqu'à ce que toutes les machines répondent. Ceci garantit la fiabilité et laisse suffisamment de temps aux réponses en retard pour arriver au collecteur.

Equité avec TCP

TICP dispose d'un paramètre qui indique la granularité de son contrôle de congestion. Ce paramètre, appelé `RS` pour Request Size, est l'équivalent du taille de paquet dans TCP. Dans

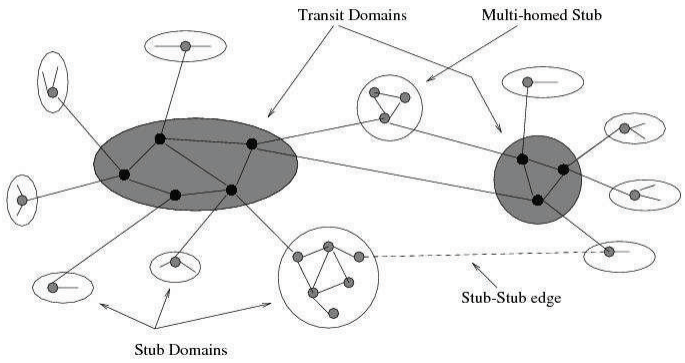

FIG. 6.1 – Topologie du réseau pour la validation de TICP

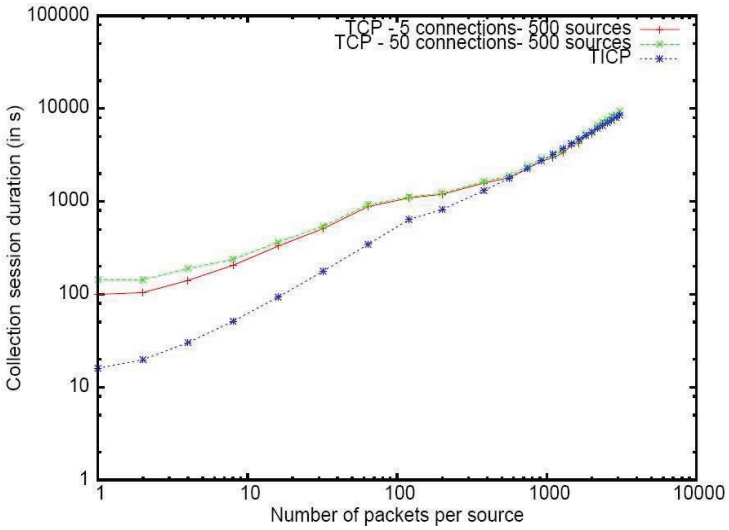

FIG. 6.2 – Comparaison entre TICP et TCP

TICP, la granularité est en nombre de sondes. Pendant la phase de Congestion Avoidance, la fenêtre de TICP augmente de `RS` sondes tous les RTTs. Dans la phase Slow Start, la fenêtre double chaque RTT. Du coup, ce paramètre `RS` décide de l'équité entre TICP et TCP. En raisonnant en termes de paquets/s, le débit d'une session TICP peut être vu comme étant `RS` fois le débit d'une connexion TCP qui tourne dans les mêmes conditions.

Les détails de ces deux contrôles dans TICP sont décrits dans [102] ainsi que dans l'Annexe D. Le protocole a été implémenté dans le simulateur des réseaux ns-2 [88] ainsi qu'en C++ pour tourner sur linux en dessus de UDP. Les performances de TICP ont été validées dans différents scenarios à la fois par simulations et par expérimentations PlanetLab. Le résultat le plus intéressant était à propos de la comparaison entre TCP et TICP. En simulant TICP dans ns-2 sur la topologie décrite dans Figure 6.1 et pour 500 sources de données distribuées aléatoirement dans

le réseau ainsi que le collecteur, nous avons pu tracer les performances illustrées dans Figure 6.2. Cette dernière figure montre la durée de la session de sondage en fonction du volume d'informations à collecter de chaque source sondée. La figure montre aussi les performances obtenues en utilisant des connexions TCP en parallèle (5 et 50). Pour le cas des connexions TCP, dès qu'une connexion se termine une autre est établie avec une autre source de données afin de récupérer ses informations. Il est à noter dans cette figure les excellentes performances de TICP (un ordre de grandeur plus que TCP) quand il s'agit de sources avec peu de données à remonter au collecteur. Ceci s'explique par le fait que TICP agrège les données en provenant des différentes sources et évite par conséquence les phases Slow Start des connexions TCP. Cependant, quand le volume d'informations par source devient important, TICP se comporte identiquement aux connexions TCP, ces dernières pouvant maintenant atteindre leurs phases congestion avoidance.

6.2 Parcours des machines dans TICP

Les machines à sonder peuvent se trouver réparties dans le réseau et situées derrière des goulots d'étranglement différents. Demander à TICP de les sonder indépendamment de leur localité c'est compliquer la tâche pour le protocole puisqu'il aura à gérer plusieurs goulots d'étranglement en parallèle et avec une seule fenêtre de congestion. TICP a alors du mal à différencier les goulots congestionnés de ceux non congestionnés ce qui résulte en un contrôle de congestion sous-efficace et des ressources réseaux mal utilisées. Pour cela nous avons enrichi TICP avec une solution pour le regroupement des machines à sonder. Les machines sont groupées avec un mécanisme de type coordonnées virtuelles ou noms DNS ou adresses IP, et sont après sondées en fonction de leur localité en allant des machines proches de la source vers les machines distantes. Le plus important est que les machines proches les unes des autres soient sondées ensemble avant de passer à un autre groupe de machines à côté et ceci jusqu'à parcourir toute la liste des machines. Ceci a plusieurs avantages :

- En partant du plus près au plus loin, on assure un maximum de collecte au début de la session.
- En ordonnant les machines dans la liste à sonder en fonction de leur localité, on permet au contrôle de congestion de traiter simultanément avec les machines derrière le même goulot d'étranglement ce qui conduit à une utilisation efficace des ressources.
- Finalement, en classant les machines de plus près au plus loin, on assure un changement en douceur des conditions réseau ce qui permet aux paramètres de TICP de s'ajuster rapidement à ce changement sans pénaliser les performances.

Nous avons testé l'idée du regroupement des machines en utilisant des coordonnées virtuelles. Chaque machine est supposée dotée de coordonnées dans un espace à deux dimensions représentant la topologie de l'Internet. Après les sources sont groupées en des cellules de forme carrée et les cellules sont parcourues par TICP dans le sens opposé des aiguilles d'une montre en allant de

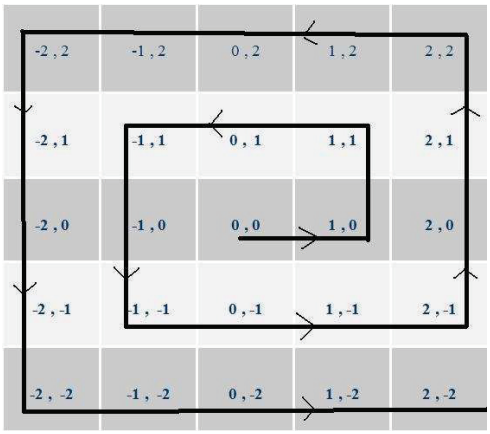

FIG. 6.3 – Le découpage de l'espace et le regroupement des machines dans TICP

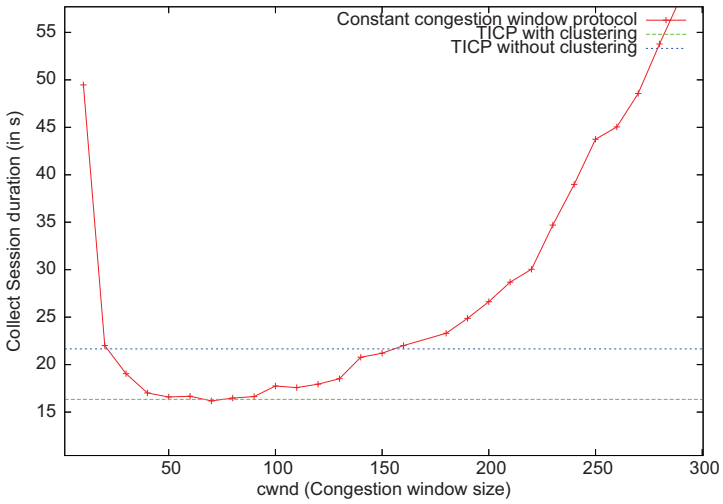

FIG. 6.4 – TICP avec regroupement des machines comparé au cas sans regroupement

plus près au plus loin, voir Figure 6.3. La durée moyenne d'une session TICP avec regroupement des machines est comparée dans Figure 6.4 à la durée de la session sans regroupement. On voit bien une réduction de la durée de presque 50%. La figure montre aussi une comparaison à un protocole de sondage à fenêtre constante et qui change sur l'axe des X dans la figure. Pour ce dernier protocole et en augmentant la fenêtre, la durée de la session baisse au début jusqu'à atteindre la fenêtre optimale à utiliser, ensuite se dégrade puisque la fenêtre devient trop grande pour que le réseau puisse la supporter. TICP avec regroupement est capable de prédire la valeur optimale de la fenêtre et de donner lieu aux mêmes performances pratiquement.

Dans [102] nous avons aussi regardé la taille optimale de la cellule à utiliser et l'impact de cette taille. Clairement cette taille a une importance. Quand elle est trop grande, TICP fonctionne comme dans le cas sans regroupement des machines. Et en réduisant la taille de chaque cellule,

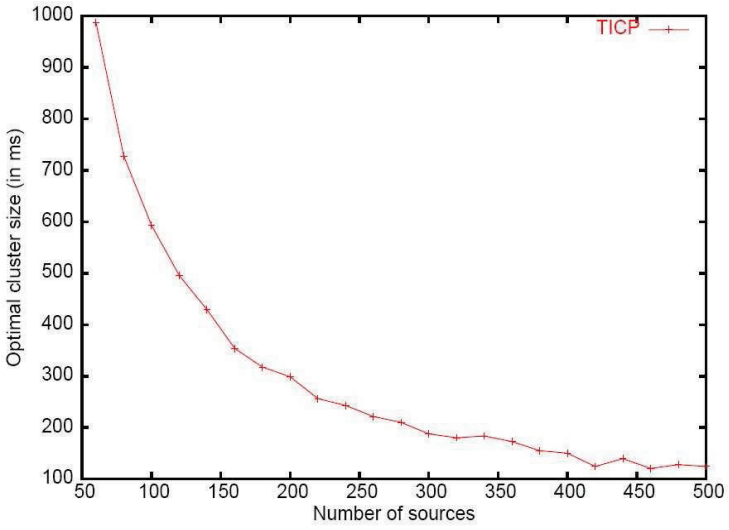

FIG. 6.5 – Clusterisation optimale sur PlanetLab vue par TICP

les performances de TICP s'améliorent jusqu'à un certain point optimal, après elles se dégradent puisqu'on revient au cas sans regroupement pour de très petites cellules. Cette taille optimale est clairement fonction du nombre de sources, mais une de nos observations dans ce travail c'est que à partir d'un certain nombre de sources, la taille optimale devient presque invariable déterminée par la distribution des goulots d'étranglement plus que par la distribution des sources. Figure 6.5 illustre ce résultat sur PlanetLab où on voit un découpage optimal en des carrés de 100 ms d'arrête au delà d'un certain nombre de sources.

6.3 Conclusions

Notre protocole TICP propose une solution à un problème qui était jusqu'à lors sous-estimé, celui du contrôle de congestion et du contrôle d'erreurs pour les applications sondant un grand nombre de machines dans l'Internet. Ce travail est en cours et a eu une première validation avec ns-2 et sur PlanetLab. Il reste à le valider avec des vraies applications comme par exemple le traçage de la topologie de l'Internet ou la vérification de l'état de fonctionnement d'une plateforme quelconque formée d'un grand nombre de machines. L'étude d'autres méthodes de regroupement des machines est aussi importante pour le futur de cette solution.

Chapitre 7

Conclusions

Cette thèse présente un ensemble de solutions pour une métrologie efficace de l'Internet. C'est un domaine en pleine évolution étant donnée la croissance exponentielle de l'Internet en termes de machines et réseaux couplée à l'absence de solutions standard pouvant répondre aux besoins des utilisateurs et des opérateurs. Notre objectif le long de cette recherche a été le développement des solutions qui passent à l'échelle et qui réduisent la charge sur le réseau sans pour autant compromettre la précision des résultats obtenus. Nous avons agi pour cela sur deux fronts. Sur le front des mesures passives, nous avons proposé une architecture pour la capture du trafic à travers un réseau d'opérateur. Au lieu de collecter tout le trafic et de l'analyser ultérieurement, notre architecture permet aux opérateurs de définir une tâche de mesure et de programmer les routeurs de telle sorte que chacun d'eux collecte la quantité de trafic nécessaire sans dépasser un certain seuil sur le volume du trafic global à remonter aux unités centrales. Cette réduction du volume du trafic collecté s'effectue par le biais de l'échantillonnage spatiale et temporelle. L'échantillonnage a un impact négatif sur la précision de la mesure, un impact qui peut être réduit par des techniques de filtrage et des solutions protocolaires. Nous avons illustré ceci sur deux exemples concrets, un celui de la détection des gros flots utilisateurs de trafic et un autre celui de la métrologie de grandes populations variant au cours du temps. Pour les deux exemples nous avons mené une étude pour évaluer l'impact de l'échantillonnage suivi par des solutions réduisant l'erreur lors de l'inversion des informations échantillonnées. Ensuite, nous avons abordé l'approche active dans la métrologie et avons présenté une solution appelée TICP qui permet de contrôler le débit des sondes et d'assurer la fiabilité quand il s'agit d'une application sondant un grand nombre de machines distribuées à travers le réseau. TICP peut être vu comme un multiplexage de plusieurs connexions TCP partant d'une même machine et destinées à des machines différentes en une seule connexion point-à-multipoint. Toutes nos solutions ont été validées par des simulations et des traces réelles.

Les travaux dans cette thèse ne sont qu'un exemple de nos activités de recherche dans le domaine de la métrologie en particulier et des réseaux en général. Nous avons parallèlement contribué à la métrologie des réseaux sans fil [29] et au développement des solutions protocolaires

qui permettent l'échange d'informations entre des appareils sans fils que ce soit des ordinateurs portables ou des ordinateurs de poche [67, 103]. Nous avons aussi contribué à l'étude des réseaux Pair-à-Pair e.g. [103] et à la sécurité et au dimensionnement des réseaux informatiques [89]. Le long de toute cette recherche nous avons adopté une approche de développement des solutions optimisées qui permettent une meilleure utilisation des réseaux et qui introduisent de nouveaux services. L'aspect performances des solutions a toujours était présent dans nos travaux, couplé à une validation des solutions par des traces réelles et des expérimentations.

Les travaux en cours et futurs continuent à focaliser sur la métrologie puisque nous croyons que c'est un domaine très important pour l'avenir de l'Internet. Afin de pouvoir proposer de nouvelles architectures il faut avoir une bonne compréhension des limitations de l'architecture actuelle. Une grande partie des services proposés actuellement pourra se voir intégrée dans l'Internet de futur comme par exemple les services de calcul de coordonnées virtuelles pour les machines ou les architectures Pair-à-Pair et DHT. En plus de ces services, l'Internet de futur sera certainement doté de nouveaux services qui permettent la détection des problèmes et leur diagnostique. Les routeurs de l'Internet de futur se verront aussi dotés de mécanismes intelligents leur permettant de dialoguer entre eux et de prendre les bonnes décisions pour une métrologie efficace du réseau au lieu de se contenter comme c'est fait actuellement de tout envoyer à une unité centrale qui fait le traitement. Nous croyons que ce sont deux directions très intéressantes auxquelles nous allons concentré la plupart de nos efforts. Pour un meilleur Internet ces efforts devront certainement s'accompagner de nouvelles applications qui passent à l'échelle et qui communiquent avec les services de métrologie et de sécurité du réseau pour un meilleur fonctionnement et une meilleure qualité pour tous les acteurs.

Bibliographie

[1] Abilene : Advanced networking for leading-edge research and educatio n. http ://abilene.internet2.edu.

[2] A. Adams, T. Bu, R. Caceres, N. Duffield, T.Friedman, J. Horowitz, F. Lo Presti, S.B. Moon, V. Paxson, D. Towsley, "The Use of End-to-end Multicast Measurements for Characterizing Internal Network Behavior", IEEE Communications Magazine, May 2000.

[3] W. Aiello, F. Chung, L. Lu, Random evolution in massive graphs, Handbook on Massive Data Sets, (Eds. James Abello et al.), Kluwer Academic Publishers, (2002) 97-122.

[4] K. C. Almeroth and M. H. Ammar. MListen, 1995. http ://www.cc.gatech.edu/computing/Telecomm/mbone/.

[5] S. Alouf, E. Altman, C. Barakat, P. Nain, "Estimating Membership in a Multicast Session", in proceedings of ACM SIGMETRICS, Jun. 2003. Available in Annex C.

[6] S. Alouf, E. Altman, C. Barakat, P. Nain, "Optimal Estimation of Multicast Membership", IEEE Transactions on Signal Processing - Special Issue on Signal Processing in Networking, vol. 51, no. 8, pp. 2165-2176, August 2003.

[7] E. Altman, K. Avrachenkov, C. Barakat, "A Stochastic Model of TCP/IP with Stationary Random Losses", IEEE/ACM Transactions on Networking, vol. 13, no. 2, pp. 356- 369, April 2005.

[8] E. Altman, K. Avrachenkov, C. Barakat, P. Dube, "Performance Analysis of AIMD Mechanisms over a Multi-State Markovian Path", Computer Networks Journal, vol. 47, no. 3, pp. 307-326, February 2005.

[9] E. Altman, C. Barakat, V. Ramos, “Queueing Analysis of Simple FEC Schemes for IP Telephony”, IEEE INFOCOM, Apr. 2001.

[10] ANDERSEN, D. G., BALAKRISHNAN, H., KAASHOEK, M. F., AND MORRIS, R., "Resilient Overlay Networks". In Proc. 18th ACM SOSP (Banff, Canada, Oct. 2001), pp. 131-145.

[11] D. G. Andersen, N. Feamster, S. Bauer, H. Balakrishnan, "Topology Inference from BGP Routing Dynamics", ACM SIGCOMM Internet Measurement Workshop, Marseille, Nov. 2002.

[12] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira - "Avoiding traceroute anomalies with Paris traceroute", in proceedings of ACM Internet Measurement Conference, October 2006).

[13] A. L. Barabasi and R. Albert, "Emergence of Scaling in Random Networks", Science, vol. 286, pp. 509-512, 1999.

[14] C. Barakat, A. Al Fawal, " Analysis of link-level hybrid FEC/ARQ-SR for wireless links and long-lived TCP traffic", Performance Evaluation Journal, vol. 57, no. 4, pp. 423-500, August 2004.

[15] C. Barakat, G. Iannaccone, and C. Diot, "Ranking flows from sampled traffic", In Proceedings of CoNext, Toulouse, October 2005. Available in Annex B.

[16] C. Barakat, P. Thiran, G. Iannaccone, C. Diot, P. Owezarski, "Modeling Internet backbone traffic at the flow level", IEEE Transactions on Signal Processing - Special Issue on Signal Processing in Networking, vol. 51, no. 8, pp. 2111-2124, August 2003.

[17] P. Barford, A. Bestavros, J. Byers, and M. Crovella, "On the marginal utility of network topology measurements," in ACM Internet Measurements Workshop, (San Francisco, CA, USA), November 2001.

[18] Y. Bejerano and R. Rastogi. "Robust monitoring of link delays and faults in IP networks". In Proceedings of IEEE Infocom, Apr. 2003.

[19] S. Ben Fredj, T. Bonald, A. Proutiere, G. Regnie, J. Roberts, "Statistical Bandwidth Sharing : A Study of Congestion at Flow Level", *ACM SIGCOMM*, Aug. 2001.

[20] L. Bernaille, R. Teixeira, and K. Salamatian - "Early Application Identification", in Proc. of CoNEXT, Decembre 2006.

[21] J-C. Bolot, T. Turletti, "Experience with rate control mechanisms for packet video in the Internet", Computer Communication Review, vol. 28, no. 1, January 1998.

[22] J-C. Bolot, S. Fosse-Parisis, D. Towsley, “Adaptive FEC-Based error control for Internet Telephony”, IEEE INFOCOM, Mar. 1999.

[23] Bloom, Burton H. (1970), "Space/time trade-offs in hash coding with allowable errors", Communications of the ACM 13 (7) : 422-426

[24] D. Brauckhoff, B. Tellenbach, A. Wagner, A. Lakhina, M. May, "Impact of Packet Sampling on Anomaly Detection Metrics", ACM Internet Measurement Conference 2006, IMC 2006, Rio de Janeiro, Brazil, October, 2006.

[25] Brutlag J.D., "Aberrant Behavior Detection in Time Series for Network Monitoring", in Proc. 14th Systems Administration Conference (LISA 2000), New Orleans, LA, USA, 2000.

[26] T. Bu and D. Towsley, "Fixed Point Approximation for TCP behavior in an AQM Network", Proceedings of ACM SIGMETRICS 2001.

[27] Calvert (K. L.), Griffioen (J.), Mullins (B.), Sehgal (A.), Wen (S.), "Concast : Design and Implementation of an Active Network Service". IEEE Journal on Selected Area in Communications (JSAC), vol. 19, no. 3, March 2001.

[28] G. R. Cantieni, G. Iannaccone, C. Barakat, C. Diot, P. Thiran, "Reformulating the monitor placement problem : Optimal Network-wide Sampling", a short version is invited to the Conference on Information Science (CISS), Princeton NJ, March 2006. A long version in proceedings of CoNext, Lisboa, Portugal, December 2006. Available in Annex A.

[29] G. R. Cantieni, Q. Ni, C. Barakat, T. Turletti, "Performance Analysis of Finite Load Sources in 802.11b Multirate Environments", Computer Communications Journal, special issue on Performance Issues of Wireless LANs, PANs, and Ad Hoc Networks, vol. 28, no. 10, pp. 1095-1109, June 2005.

[30] N. Cardwell, S. Savage, T. Anderson, "Modeling TCP Latency", *IEEE INFOCOM*, Mar. 2000.

[31] Cooperative Association for Internet Data Analysis (CAIDA), http ://www.caida.org/

[32] Carter, R. L. and Crovella, M. E. 1996. "Measuring bottleneck link speed in packet-switched networks". Perform. Eval. 27-28 (Oct. 1996), 297-318.

[33] M. Crovella, A. Bestavros, "Self-Similarity in World Wide Web Traffic : Evidence and Possible Causes", *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 835-846, Dec. 1997.

[34] F. Dabek, R. Cox, F. Kaashoek and R. Morris, "Vivaldi : A decentralized network coordinate system", in Proceedings of the ACM SIGCOMM, Portland, Oregon, August 2004.

[35] B. Donnet, P. Raoult, T. Friedman, M. Crovella, "Deployment of an Algorithm for Large-Scale Topology Discovery", in IEEE Journal on Selected Areas in Communications (JSAC), Dec. 2006, vol. 24, no. 12.

[36] N.G. Duffield, F. Lo Presti, V. Paxson, D. Towsley, "Inferring link loss using striped unicast probes", Proc. IEEE Infocom 2001, Anchorage, Alaska, April 22-26, 2001.

[37] N. G. Duffield, C. Lund, and M. Thorup. "Estimating flow distributions from sampled flow statistics". In Proceedings of ACM Sigcomm, Aug. 2003.

[38] ENDACE. DAG Network Monitoring Interface Card. http ://endace.com/networkMCards.htm

[39] C. Estan and G. Varghese. "New directions in traffic measurement and accounting". In Proceedings of ACM Sigcomm, Aug. 2002.

[40] T. S. Eugene Ng, Hui Zhang, "Predicting Internet Network Distance with Coordinates-Based Approaches", INFOCOM 2002.

[41] M. Faloutsos, P. Faloutsos and C. Faloutsos, "On Power Law Relationships of the Internet Topology", SIGCOMM 1999.

[42] A. Feldmann, A. C. Gilbert, P. Huang, and W. Willinger. "Dynamics of IP traffic : A study of the role of variability and the impact of control". In Proc. ACM SIGCOMM'99, pp. 301–313, 1999.

[43] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True. "Deriving traffic demands for operational ip networks : Methodology and experience". In Proceedings of ACM Sigcomm, Sept. 2000.

[44] S. Floyd, K. Fall, "Promoting the Use of End-To-End Congestion Control in the Internet", IEEE/ACM Transactions in Networking, vol. 7, no. 4, pp. 458-472, Aug. 1999.

[45] S. Floyd, M. Handley, J. Padhye, "Equation-based congestion control for unicast applications", in proceedings of ACM SIGCOMM, Aug. 2000.

[46] Fraleigh, C., Diot, C., Lyles, B., Moon, S., Owezarski, P., Papagiannaki, D., Tobagi, F. : "Design and deployment of a passive monitoring infrastructure". Lecture Notes in Computer Science 2170 (2001).

[47] M. Fullmer and S. Romig. "The OSU flow-tools package and Cisco NetFlow logs". In Proceedings of Usenix LISA, Dec. 2000.

[48] GEANT. http ://www.dante.net.

[49] K. Harfoush, A. Bestavros and J. Byers. "Robust Identification of Shared Losses Using End-to-End Unicast Probes", Proc. of the 8th IEEE International Conference on Network Protocols (ICNP), Osaka, Japan, November 2000, pp. 22-33.

[50] S. Haykin. Modern Filters. Macmillan, New York, 1989.

[51] N. Hohn and D. Veitch. "Inverting sampled traffic". In Proceedings of ACM Internet Measurement Conference, Oct. 2003.

[52] J. Horton and A. Lopez-Ortiz. "On the number of distributed measurement points for network tomography". In Proceedings of ACM Internet Measurement Conference, Nov. 2003.

[53] N. Hu and P. Steenkiste. "Evaluation and Characterization of Available Bandwidth Techniques". IEEE JSAC Special Issue in Internet and WWW Measurement, Mapping, and Modeling, 2003.

[54] N. Hu, L. E. Li, Z. M. Mao, P. Steenkiste, and J. Wang, "Locating internet bottlenecks : algorithms, measurements, and implications", In SIGCOMM 2004 : Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 41-54, New York, NY, USA, 2004, ACM Press.

[55] IP Flow Information eXport Working Group. Internet Engineering Task Force. http ://www.ietf.org/html.charters/ipfix-charter.html.

[56] V. Jacobson, "Congestion avoidance and control", ACM SIGCOMM, Aug. 1988.

[57] Van Jacobson. "pathchar-a tool to infer characteristics of Internet paths". Presented at the Mathematical Sciences Research Institute(MSR1) ; slides available from ftp ://ftp.ee.lbl.gov/pathchar/, April 1997.

[58] Jain, M. and Dovrolis, C. 2002. "End-to-end available bandwidth : measurement methodology, dynamics, and relation with TCP throughput". SIGCOMM Comput. Commun. Rev. 32, 4 (Oct. 2002), 295-308.

[59] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, D. Towsley, "Inferring TCP Connection Characteristics Through Passive Measurements", IEEE Infocom, Hong Kong, March 2004.

[60] S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang. On the placement of internet instrumentation. In Proceedings of IEEE Infocom, Apr. 2000.

[61] Xing Jin, W.-P. Ken Yiu, S.-H. Gary Chan and Yajun Wang, "Network Topology Inference Based on End-to-End Measurements", IEEE JSAC special issue on sampling the Internet, Vol. 24, No. 12, pp. 2182-2195, Dec. 2006.

[62] Juniper Traffic Sampling. http ://www.juniper.net. Netflow-compatible implementation.

[63] M. A. Kaafar, L. Mathy, C. Barakat, K. Salamatian, T. Turletti, W. Dabbous, "Securing Internet Coordinate Embedding Systems", in proceedings of ACM SIGCOMM, Kyoto, Japan, August 2007.

[64] R.E. Kalman, A New Approach to Linear Filtering and Prediction Problems, in Transactions of the ASME - Journal of Basic Engineering Vol. 82 : pp. 35-45, 1960.

[65] S. Katti, D. Katabi, C. Blake, E. Kohler, and J. Strauss, "MultiQ : Automated Detection of Multiple Bottlenecks Along a Path", ACM IMC 2004.

[66] S. Keshav, “A control-theoretic approach to flow control, ACM SIGCOMM, Sep. 1991.

[67] A. Krifa, C. Barakat, T. Spyropoulos, "Optimal Buffer Management Policies for Delay Tolerant Networks", in proceedings of the 5th IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON 2008), San Francisco, June 2008.

[68] LAKHINA, A., CROVELLA, M., AND DIOT, C. "Diagnosing network-wide traffic anomalies". In Proceedings of ACM SIGCOMM (2004).

[69] A. Lakhina, M. Crovella and C. Diot. "Mining Anomalies Using Traffic Feature Distributions". Appears in ACM SIGCOMM, Philadelphia, PA, August 2005.

[70] W. Leland, M. Taqq, W. Willinger, D. Wilson, "On the self-similar nature of Ethernet traffic," ACM SIGCOMM,Sep. 1993.

[71] Libpcap. http ://www.tcpdump.org.

[72] Machiraju, S., Veitch, D., Baccelli, F., and Bolot, J. C. "Adding definition to active probing". SIGCOMM Comput. Commun. Rev. 37, 2 (Mar. 2007), 17-28.

[73] Litwin, W. (1980). "Linear hashing : A new tool for file and table addressing". Proc. 6th Conference on Very Large Databases : pages 212-223.

[74] B. Mah. "An Empirical Model of HTTP Network Traffic", Proceedings of INFOCOM '97, Kobe, Japan, April 1997.

[75] J. Mai, A. Sridharan, C-N. Chuah, T. Ye, and H. Zang, "Impact of Packet Sampling on Portscan Anomaly Detection," IEEE Journal on Selected Areas of Communications - Special Issue on Sampling the Internet, vol. 24, no. 12, pp. 2285-2298, December 2006.

[76] M. Malli, C. Barakat, W. Dabbous, "Application-level versus Network-level Proximity", in proceedings of the AINTEC conference, Bangkok, December 2005.

[77] M. Malli, C. Barakat, W. Dabbous, "CHESS : An application-aware space for enhanced scalable services in overlay networks", Computer Communications Journal, vol. 31, no. 6, pp. 1239-1253, April 2008.

[78] N. Moller, C. Barakat, K. Avrachenkov, and E. Altman, "Inter-protocol fairness between TCP New Reno and TCP Westwood+", in proceedings of NGI 2007 (Conference on Next Generation Internet Networks), Trondheim, Norway, May 2007.

[79] Z. M. Mao, J. Rexford, J. Wang, and R. H. Katz. "Towards an accurate AS-level traceroute tool". In SIGCOMM 2003.

[80] S. McCanne and V. Jacobson. "The BSD packet filter : A new architecture for user-level packet capture". In Proceedings of the USENIX Winter Conference, January 1993.

[81] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot. "Traffic matrix estimation : Existing techniques and new directions". In Proceedings of ACM Sigcomm, Pittsburgh, Aug. 2002.

[82] B. Melander, M. Bjorkman, and P. Gunningberg. "A new end-to-end probing and analysis method for estimating bandwidth bottlenecks". In IEEE Globecom - Global Internet Symposium, San Francisco, November 2000.

[83] Mogul, J. and S. Deering, " Path MTU discovery", RFC 1191, November 1990.

[84] S. Moon, P. Skelly, and D. Towsley, "Estimation and Removal of Clock Skew from Network Delay Measurements," Proceedings IEEE Infocom, NY, Mar. 1999.

[85] Cisco Netflow, http ://www.cisco.com/ en/US/products/ps6601/products_ios_protocol_group_home.html

[86] H. Nguyen and P. Thiran. "Active measurement for multiple link failures diagnosis in IP networks". In Proceedings of PAM Workshop, Apr. 2004.

[87] J. Nonnenmacher and E. Biersack. "Scalable feedback for large groups". IEEE/ACM Trans. on Networking, 7(3) :375-386, June 1999.

[88] The LBNL Network Simulator, *ns-2*, http ://www.isi.edu/nsnam/ns/

[89] A. Nucci, N. Taft, C. Barakat, P. Thiran, "Controlled Use of Excess Backbone Bandwidth for Providing New Services in IP-over-WDM Networks", IEEE Journal on Selected Areas in Communications, vol. 22, no. 9, pp. 1692-1707, November 2004.

[90] P. Phaal, S. Panchen, and N. McKee. "InMon corporation's sFlow : A method for monitoring traffic in switched and routed networks". RFC 3176, Sept. 2001.

[91] An open, distributed platform for developing, deploying and accessing planetary- scale network services, see http ://www.planet-lab.org/.

[92] D. Plonka. "Flowscan : A network traffic flow reporting and visualization tool". In Proceedings of Usenix LISA, Dec. 2000.

[93] Postel, J., "Internet Control Message Protocol", RFC 777, USC/Information Sciences Institute, April 1981.

[94] Packet Sampling Working Group. Internet Engineering Task Force. http ://www.ietf.org/html.charters/psamp-charter.html.

[95] V. Ramos, C. Barakat, E. Altman, "A Moving Average Predictor for Playout Delay Control in VoIP", in proceedings of the Eleventh International Workshop on Quality of Service (IWQoS 2003), Monterey, CA, June 2003.

[96] S. Ratnasamy, M. Handley, R. Karp, S. Shenker, "Topologically-Aware Overlay Construction and Server Selection", IEEE Infocom 2002.

[97] Y. Rekhter et T. Li, "A Border Gateway Protocol (BGP-4)". RFC 1771, Mars 1995.

[98] Renater. http ://www.renater.fr.

[99] B. F. Ribeiro, D. F. Towsley, T. Ye, J. Bolot, "Fisher information of sampled packets : an application to flow size estimation". Internet Measurement Conference 2006 : 15-26

[100] V.J. Ribeiro, R.H. Riedi, R.G. Baraniuk, J. Navratil, L. Cottrell, "PathChirp : Efficient Available Bandwidth Estimation for Network Paths", Passive and Active Measurement Workshop, 2003.

[101] University of Oregon Route Views Project. http ://www.routeviews.org/

[102] K. Sbai, C. Barakat, "Experiences on enhancing data collection in large networks", INRIA Tech Report inria-00140937, to appear in Computer Networks. Extended version of two papers : one in Annals of Telecommunications, vol. 61, no. 1-2, pp. 167-192, January-February 2006, and the other in proceedings of NTMS 2007 (Conference on New Technologies, Mobility and Security), Paris, May 2007. Available in Annex D.

[103] M. K. Sbai, C. Barakat, J. Choi, A. Al Hamra, T. Turletti, "Adapting BitTorrent to wireless ad hoc networks" in proceedings of 7th International conference on ad hoc networks and wireless 2008 (AD-HOC NOW), Sophia Antipolis, France, September 2008.

[104] Skitter infrastructure by CAIDA, http ://www.caida.org/tools/measurement/skitter/

[105] N. Spring, R. Mahajan, D. Wetherall, "Measuring ISP Topologies with Rocketfuel", ACM SIGCOMM 2002.

[106] A. Soule, A. Lakhina, N. Taft, K. Papagiannaki, K. Salamatian, A. Nucci, M. Crovella and C. Diot. "Traffic Matrices : Balancing Measurements, Inference and Modeling". In ACM Sigmetrics. June 2005.

[107] Strauss, J., Katabi, D., and Kaashoek, "A measurement study of available bandwidth estimation tools". In Proceedings of the 3rd ACM SIGCOMM Conference on internet Measurement (Miami Beach, FL, USA, October 2003).

[108] K. Suh, Y. Guo, J. Kurose, and D. Towsley. "Locating network monitors : Complexity, heuristics and coverage". In Proceedings of IEEE Infocom, Mar. 2005.

[109] M. Zangrilli and B. B. Lowekamp, "Applying Principles of Active Available Bandwidth Algorithms to Passive TCP Traces," 6th Workshop on Passive and Active Measurements (PAM2005), LNCS 3431, March 2005.

[110] H. Zheng, E. K. Lua, M. Pias, T. G. Griffin. "Internet Routing Policies and Round-Trip-Times", Passive and Active Measurement Workshop (PAM) 2005.

[111] WAGON : A Web Server Benchmarking Tool, INRIA, Mistral research group.

[112] R.W. Wolff. Stochastic Modeling and the Theory of Queues. Prentice- Hall, Englewood Cliffs, NJ, 1989.

CV détaillé

Name	: Chadi BARAKAT
Birth	: Saida (Lebanon), 19th June 1974
Family Status	: Married to Joumana Maallawi
Citizenship	: French
Address	: INRIA - 2004, route des Lucioles - 06902 Sophia Antipolis - France
Phone (Office)	: +33 4 92 38 75 96
Phone (Secretariat)	: +33 4 92 38 78 25
Fax (Office)	: +33 4 92 38 79 78
Email	: Chadi.Barakat@sophia.inria.fr
Web Page	: http ://planete.inria.fr/chadi

Profession :

- 03/02 - : Research scientist, Planète project-team, INRIA Sophia Antipolis - Méditerranée, France.
- 03/04 - 08/04 : Invited Faculty Member, Intel Research Cambridge, UK.
- 05/01 - 03/02 : Postdoctoral fellow, LCA - FIC - EPFL, Lausanne, Switzerland.
- 09/98 - 05/01 : PhD student, Mistral project-team, INRIA, France.
- 03/98 - 07/98 : Intern, networking department, ENST-Paris, France.

Education :

- April 2001 : PhD degree in Computer Sciences :
  - *University :* Nice - Sophia Antipolis, France.
  - *Laboratory :* Mistral project-team, INRIA, France.
  - *Advisor :* Eitan Altman.
  - *Subject :* Performance evaluation of TCP congestion control.
- June 1998 : DEA (Diplôme d'Études Approfondies, the french equivalent of the Master degree) in Networking and Distributed Systems, University of Nice - Sophia-Antipolis, France.
- July 1997 : Electrical and Electronics Engineering degree, Lebanese University, Beirut, Lebanon.

Awards :

- Best paper award for my paper "Optimal Buffer Management Policies for Delay Tolerant Networks", IEEE SECON conference, San Francisco, June 2008.
- Best PhD student paper award for my paper "On ACK filtering on a slow reverse channel", first international workshop on Quality of future Internet Services, Berlin, September 2000.
- Hariri foundation award, top 100 funded by the foundation in Lebanon and abroad, five times from 1993 to 1997.

Research Interests : Internet measurement and traffic analysis, congestion and error control in computer networks, performance evaluation of wireless networks, voice over IP, Peer-to-Peer architectures, network coding, Delay Tolerant Networking.

Research Projects :

- Network diagnosis by collaborative monitoring. Supported by the CMON ANR project.
- Content sharing in wireless networks. Supported by the ITEA Expeshare project.
- Adaptive traffic sampling. Supported by the ECODE FP7 project.
- Understanding the Properties of the BitTorrent Overlay. Initially supported by the OSCAR ANR project.
- Scheduling and ressource control in Delay Tolerant Networks.
- Scalable solutions for data collection. Initially supported by Hitachi Europe.
- Performance evaluation of congestion control in the Internet.

Professional activities :

- Co-organizer of the francophone summer school ResCom 2007.
- Guest editor for a JSAC special on Sampling the Internet : Techniques and Applications.
- Area editor for ACM Computer Communication Review (CCR), 2005 - present.
- Steering committee member for PAM workshop (Passive and Active Measurement).
- Program Committee co-chair for the Sampling 2005 workshop (Paris).
- General chair for WiOpt 2005 workshops (Riva Del Garda, Italy).
- General chair for PAM 2004 (Juan-les-Pins, France).
- Technical Program Committee member for Algotel 2009 (France), ITC 2009 (Paris), Infocom 2009 (Rio), Broadnets 2008 (London), WNS2 2008 (Athens), WinMee 2008 (Berlin), PFLDnet 2008 (Manchester), NSTools 2007 (Nantes), SECON 2007 (San Diego), WWIC 2007 (Coimbra), IWQoS 2006 (New Haven), WWIC 2006 (Bern), WONS 2006 (Lyon), WiNMee 2006 (Boston), IMC 2005 (Berkeley), ICNP 2005 (Boston), PAM 2005 (Boston), WONS 2005 (St. Moritz), INFOCOM 2005 (Miami), ASWN 2004 (Boston), GI&NGN symposium at Globecom 2004 (Dallas), PAM 2004 (Juan-les-Pins), INFOCOM 2004 (Hong Kong), ASIAN 2002 (Hanoi), ICNP 2002 (Paris).
- Session chair at IMC 2005 (Berkeley), WiOpt 2003 (Sophia Antipolis), ICNP 2002 (Paris).

- Invited speaker at Brunel University (UK) - 2008, COST 285 final symposium (Surrey, UK) - 2007, Asian school (Bangkok) - 2005, E-next school (Louvain) - 2005, KTH (Stockholm) - 2005, ENSI (Tunis) - 2005, MIT (Boston) - 2004, UMASS (Amherst) - 2004, Boston University (Boston) - 2004, Intel Research Cambridge - 2004, ENSI (Tunis) - 2004, ASIAN school (Bangkok) - 2003, Ecole Normale Supérieure (Paris) - 2003, Ecole d'été Internet Nouvelle Génération (Porquerolles) - 2003, ENSI (Tunis) - 2003, EPFL Summer Research Institute (Lausanne) - 2002, IBM (Zurich) - 2001.
- Speaker in many international conferences : SECON 2008 (San Francisco), CoNext 2006 (Lisboa), CoNext 2005 (Toulouse), WiOpt 2003 (Paris), PFLDnet 2003 (Geneva), IMW 2002 (Marseille), ICN 2001 (Colmar), INFOCOM 2001 (Alaska), QofIS 2000 (Berlin), SIGCOMM 2000 (Stockholm), SIGMETRICS 2000 (Santa Clara), NETWORKING 2000 (Paris), GLOBECOM 1999 (Rio).
- Referee for many international conferences (SIGCOMM, INFOCOM, Co-next, SIGMETRICS, ICNP, GLOBECOM, NETWORKING, WiOpt, ISCC, ASIAN, Net-Con,...) and journals (Transactions on Networking, Transactions on Multimedia, Transactions on Wireless Communications, JSAC, Performance Evaluation, IEEE Networks, Communication Magazine, ACM CCR, Computer Networks, Computer Communications,...).
- Member of the directory board for the Networking Master at the University of Nice-Sophia Antipolis, responsible of the internships for the same Master, and external member in the hiring committee at the computer sciences department of the University of Nice-Sophia Antipolis.
- Member of the E-next European Network of Excellence. 2003-2006.
- (Co)Organizer of Mistral seminars (1999-2001), of Planète seminars (2003 -), of "seminar croisé Réseau" (October 2003) and of INRIA ComB evaluation seminar (November 2007).

Industrial collaborations :

- Partner of the CMON ANR French project led by Thomson.
- Partner of the ECODE STREP European project led by Alcatel Belgium.
- Partner of the Expeshare ITEA European project involving over 40 partners (Nokia, Philips, VTT, Siemens, Telefonica, etc).
- Partner of the OSCAR French RNRT led by FranceTelecom and involving Mitsubishi.
- A visit of six months to Intel Research Cambridge.
- A collaboration of one year with Alcatel Marcoussis on Internet Topology Inference.
- TRANSAT : ESA Project led by Alcatel space on improving satellite access networks.
- TICP : Project with Hitachi on collecting information in wide area networks.
- Collaboration with Sprint ATL and EPFL on traffic modeling and protection.
- Expertise of two companies at Sophia Antipolis for their Crédit D'Impôt Recherche.

Software :

- BitHoc : Enables file sharing over wireless ad hoc networks.
- TICP : Enables data collection in large networks.
- A set of scripts for TCP analysis in the ns-2 simulator.

Selected Talks :

- On message drop and scheduling in Delay Tolerant Networks, Brunel Univ, London, April 2008.
- Issues related to the modelling of the TCP protocol, E-next school, Louvain, November 2005.
- Scalable solutions for information collection, KTH, Stockholm, September 2005.
- On TCP performance in heterogeneous networks, Summer Research Institute, Lausanne, July 2002.

Teaching activities :

- Course on Understanding Networks, 30 hours, Master IFI, University of Nice Sophia Antipolis, 2008-present.
- Course on Internet Measurement and Traffic Analysis, 15 hours, (i) Networking and Distributed Systems Master at the University of Nice Sophia Antipolis, 2004-2007, and (ii) Master RIM, ENSI, Tunis, 2003-present.
- General course on TCP/IP, 20 hours, Licence Professionnelle LPSIL, IUT of the UNSA, 2006-present.
- Course on local area networks, 15 hours + 7.5 hours practical work, first year, IUT of the University of Nice Sophia Antipolis, 2007-2008.
- Course on Voice over IP, 7 hours, (i) Ecole d'Ingénieur Internet, Marseille, 2003, (ii) Master TIM, UNSA, 2007-2008, (iii) Master RTM, IUP Avignon, 2008.
- Course on Inference of Internet Topology, 7 hours, Master RTM, IUP Avignon, 2006-present.
- Exercises on Performance Evaluation in Networking, 6 hours, Institut Eurecom, Sophia Antipolis, 2003.
- Course on TCP/IP, 6 hours, Maîtrise Informatique, University of Nice - Sophia Antipolis, France, 2003.
- Exercises on Networking, Doctoral School, DSC-EPFL, Lausanne, Switzerland, 2001.
- Course with exercises on `ns-2`, 7 hours, (i) Networking and Distributed Systems Master, University of Nice - Sophia Antipolis, France, 2001-2003, (ii) Master RTM, IUP Avignon, 2008.
- Course with research projects on Internet over Satellites, 6 hours, Networking and Distributed Systems DEA, University of Nice - Sophia Antipolis, France, 2000.

- Courses in physics in some Lebanese high schools, 1997.

PhD students :

- *Imed Lassoued* : Started in September 2008. Thesis title : Adaptive Monitoring and Management of Internet Traffic.
- *Mohamed Karim Sbai* : Co-supervised with Walid Dabbous. Started in June 2007. Thesis title : Architecture for data sharing in mobile wireless networks.
- *Mohamad Jaber* : Co-supervised with Philippe Nain. Started in Octobre 2007. Thesis title : Detection and troubleshooting of Internet anomalies.
- *Mohamad Malli* : Graduated in Septembre 2006. Thesis title : Inferring Internet topology from application point of view.
- *Victor Ramos :* Co-supervised with Eitan Altman. Graduated in December 2004. Thesis title : Reliable and robust transmission of multimedia contents over the Internet.

Trainees :

- *Emna Salhi :* Content lookup in wireless networks, 2008, Diplôme d'Ingénieur en Informatique, ENSI, Tunis.
- *Amir Krifa :* Realistic Simulation of Delay Tolerant Networks, 2007, Diplôme d'Ingénieur en Informatique, ENSI, Tunis.
- *Mouna Abdelmoumen :* Support of transport functionalities in Delay Tolerant Networks, 2006-07, Engineer in Telecommunications, SupCom, Tunis.
- *Karim Sbai :* Collecting Information from a large number of network entities, 2006, Diplôme d'Ingénieur en Informatique, ENSI, Tunis.
- *Niels Moller :* Analysis of TCP Westwood+, 2006, PhD student at KTH, Sweden. *Roman Hangartner* : Design and analysis of protocols for multimedia applications over P2P networks, 2004, engineering and master degree of EPFL, Lausanne.
- *Gion-Reto Cantieni* : Analysis of the MAC layer in 802.11 WLAN, engineering degree of EPFL, Lausanne.
- *Raja Abdelmoumen* : Study of a link-level FEC/ARQ scheme for wireless links and short-lived TCP traffic, 2003, Diplôme d'Ingénieur en Informatique, ENSI, Tunis.
- *Alaeddine Al Fawal :* Study of a link-level FEC/ARQ scheme for long-lived TCP traffic, 2002, Networking and Telecommunications DEA, Lebanese University - Saint Joseph University, Lebanon.
- *Sekou Sidibe :* Analysis of TCP/IP over asymmetric satellite links, 1999, Master of the French National Institute for Telecommunications (INT), Evry, France.
- *Emmanuel Laborde :* Markovian analysis of TCP/IP, 1999, DEA of the University of Paris VI, France.

– *Victor Ramos :* Queuing analysis of simple FEC schemes for IP Telephony, 2000, Networking and Distributed Systems DEA, University of Nice Sophia-Antipolis, France.

Liste de publications

Edited Journals and Books

1. Eitan Altman, Konstantin Avrachenkov, Chadi Barakat, Kartikeya Chandrayana, "A Multilevel Approach for the Modelling of Large TCP/IP Networks", Chapter 4 in "Recent Advances in Modeling and Simulation Tools for Communication Networks and Services" published by Springer, September 2007.
2. Chadi Barakat, Gianluca Iannaccone, Jim Kurose, and Darryl Veitch, "Sampling the Internet : techniques and applications", IEEE Journal of Selected Areas in Communications, vol. 24, no. 12, December 2006.
3. Chadi Barakat, Ian Pratt : Passive and Active Network Measurement, 5th International Workshop, PAM 2004, Antibes Juan-les-Pins, France, April 19-20, 2004, Proceedings. Lecture Notes in Computer Science 3015 Springer 2004, ISBN 3-540-21492-5.

Papers in Refereed Journals

1. Karim Sbai and Chadi Barakat, "Experiences on enhancing data collection in large networks", to appear in Computer Networks.
2. Mohammad Malli, Chadi Barakat, Walid Dabbous, "CHESS : An application-aware space for enhanced scalable services in overlay networks", Computer Communications Journal, vol. 31, no. 6, pp. 1239-1253, April 2008.
3. T. Tansupasiri, K. Kanchanasut, Chadi Barakat, Philippe Jacquet, "Using Active Networks Technology for Dynamic QoS", Computer Networks Journal, vol. 50, no. 11, pp. 1692-1709, August 2006.
4. Chadi Barakat, Mohammad Malli, Naomichi Nonaka, "TICP : Transport Information Collection Protocol", Annals of Telecommunications, vol. 61, no. 1-2, pp. 167-192, 2006.
5. Eitan Altman, Kostya Avrachenkov, Chadi Barakat, Arzad-Alam Kherani, Balakrishna Prabhu, "Analysis of MIMD Congestion Control Algorithm for High Speed Networks", Computer Networks Journal, vol. 48, no. 6, pp. 972-989, August 2005.

6. Eitan Altman, Chadi Barakat, Victor M. Ramos R., "Analysis of AIMD protocols over paths with variable delay", Computer Networks Journal, vol. 48, no. 6, pp. 960-971, August 2005.
7. Imad Aad, Qiang Ni, Chadi Barakat, Thierry Turletti, "Enhancing IEEE 802.11 MAC in Congested Environments", Computer Communications Journal, vol. 28, no. 14, pp. 1605-1617, September 2005.
8. Gion Reto Cantieni, Qiang Ni, Chadi Barakat, Thierry Turletti, "Performance Analysis under Finite Load and Improvements for Multirate 802.11", Computer Communications Journal, special issue on Performance of Wireless LANs, PANs, and Ad Hoc Networks, vol. 28, no. 10, pp. 1095-1109, June 2005.
9. Eitan Altman, Kostya Avrachenkov, Chadi Barakat, "A Stochastic Model of TCP/IP with Stationary Random Losses", IEEE/ACM Transactions on Networking, vol. 13, no. 2, pp. 356- 369, April 2005.
10. Eitan Altman, Kostya Avrachenkov, Chadi Barakat, Parijat Dube, "Performance Analysis of AIMD Mechanisms over a Multi-state Markovian Path", Computer Networks Journal, vol. 47, no. 3, pp. 307-326, February 2005.
11. Antonio Nucci, Nina Taft, Chadi Barakat, Patrick Thiran, "Controlled Use of Excess Backbone Bandwidth for Providing New Services in IP-over-WDM Networks", IEEE Journal on Selected Areas in Communications, vol. 22, no. 9, pp. 1692-1707, November 2004.
12. Chadi Barakat, Alaeddine Al Fawal, "Analysis of link-level hybrid FEC/ARQ-SR for wireless links and long-lived TCP traffic", Performance Evaluation Journal, vol. 57, no. 4, pp. 423-500, August 2004.
13. Chadi Barakat, Eitan Altman, "A Markovian Model for TCP Analysis in a Differentiated Services Network", Telecommunication Systems, vol. 25, no. 1,2, pp. 129-155, January 2004.
14. Sara Alouf, Eitan Altman, Chadi Barakat, Philippe Nain, "Optimal Estimation of Multicast Membership", IEEE Transactions on Signal Processing - Special Issue on Signal Processing in Networking, vol. 51, no. 8, August 2003.
15. Chadi Barakat, Patrick Thiran, Gianluca Iannaccone, Christophe Diot, Philippe Owezarski, "Modeling Internet backbone traffic at the flow level", IEEE Transactions on Signal Processing - Special Issue on Signal Processing in Networking, vol. 51, no. 8, August 2003.
16. Chadi Barakat, Eitan Altman, "On ACK filtering on a slow reverse channel", International Journal of Satellite Communications, vol. 21, pp. 241-258, 2003.
17. Eitan Altman, Kostya Avrachenkov, Chadi Barakat, Rudesindo Núñez-Queija, "State-dependent M/G/1 type queueing analysis for congestion control in data networks", Computer Networks Journal, vol. 39, no. 6, pp. 789-808, August 2002.
18. Eitan Altman, Chadi Barakat, Victor Ramos, "Queuing analysis of simple FEC schemes for voice over IP", Computer Networks, vol. 39, no. 2, pp. 185-206, June 2002.

19. Chadi Barakat, Eitan Altman, "Bandwidth tradeoff between TCP and link-level FEC", Computer Networks Journal, vol. 39, no. 2, pp. 133-150, June 2002.
20. Chadi Barakat, "TCP modeling and validation", IEEE Network, vol. 15, no. 3, pp. 38-47, May 2001.
21. Eitan Altman, Kostya Avrachenkov, Chadi Barakat, "TCP in presence of bursty losses", Performance Evaluation, vol. 42, no. 2-3, pp. 129-147, October 2000.
22. Chadi Barakat, Eitan Altman, Walid Dabbous, "On TCP Performance in a Heterogeneous Network : A Survey", IEEE Communication Magazine, vol. 38, no. 1, pp. 40-46, January 2000. A more detailed version : INRIA Research Report RR-3737, July 1999.

Papers in Proceedings of Refereed Conferences

1. Amir KRIFA, Mohamed Karim SBAI, Chadi BARAKAT, Thierry TURLETTI, "BitHoc : A content sharing application for Wireless Ad hoc Networks", demo description to appear in proceedings of the IEEE Percom conference, Galveston, Texas, March 2009.
2. Mohamed Karim Sbai, Chadi Barakat, Jaeyoung Choi, Anwar Al Hamra, and Thierry Turletti, "Adapting BitTorrent to wireless ad hoc networks", in proceedings of the AdHoc-Now Networks and Wireless conference, Sophia Antipolis, September 2008.
3. Amir Krifa, Chadi Barakat, Thrasyvoulos Spyropoulos, "An Optimal Joint Scheduling and Drop Policy for Delay Tolerant Networks", in proceedings of the WoWMoM Workshop on Autonomic and Opportunistic Communications, Newport Beach (CA), June 2008.
4. Amir Krifa, Chadi Barakat, Thrasyvoulos Spyropoulos, "Optimal Buffer Management Policies for Delay Tolerant Networks", in proceedings of the 5th IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON 2008), San Francisco, June 2008. **Best Paper.**
5. Mohamed Ali Kaafar, Laurent Mathy, Chadi Barakat, Kave Salamatian, Thierry Turletti, Walid Dabbous, "Securing Internet Coordinate Embedding Systems", in proceedings of ACM SIGCOMM, Kyoto, Japan, August 2007.
6. Niels Moller, Chadi Barakat, Konstantin Avrachenkov, Eitan Altman, "Inter-protocol fairness between TCP New Reno and TCP Westwood+", in proceedings of NGI 2007 (Conference on Next Generation Internet Networks), Trondheim, Norway, May 2007.
7. Karim SBAI, Chadi BARAKAT, "Transport Information Collection Protocol with clustering of information sources", in proceedings of NTMS 2007 (Conference on New Technologies, Mobility and Security), Paris, May 2007.
8. G. Boggia, P. Camarda, A. D'Alconzo, L.A. Grieco, S. Mascolo, E. Altman, C. Barakat, "Modeling the AIADD Paradigm in Networks with Variable Delays", in proceedings of CoNext, Lisboa, Portugal, December 2006.

9. Gion Reto Cantieni, Gianluca Iannaccone, Chadi Barakat, Christophe Diot, Patrick Thiran, "Reformulating the monitor placement problem : Optimal Network-wide Sampling", in proceedings of CoNext, Lisboa, Portugal, December 2006.

10. Marc Liberatore, Brian Levine, Chadi Barakat, "Maximizing Transfer Opportunities in DTN", in proceedings of CoNext, Lisboa, Portugal, December 2006.

11. Anwar Al Hamra, Chadi Barakat, Thierry Turletti, "Network Coding for Wireless Mesh Networks : A Case Study", in proceedings of WoWMoM, Buffalo-NY, June 2006.

12. Mohammad Malli, Chadi Barakat, Walid Dabbous, "Application-level versus Network-level Proximity", in proceedings of the AINTEC conference, Bangkok, December 2005.

13. Chadi Barakat, Gianluca Iannacconne, Christophe Diot, "Ranking flows from sampled traffic", in proceedings of CoNEXT, Toulouse, October 2005.

14. Hossein Manshaei, Gion Reto Cantieni, Chadi Barakat, Thierry Turletti, "Performance Analysis of the IEEE 802.11 MAC and Physical Layer Protocol", in proceedings of WoWMoM, Taormina, June 2005.

15. Imad Aad, Qiang Ni, Chadi Barakat, Thierry Turletti, "Enhancing IEEE 802.11 MAC in Congested Environments", in proceedings of the Applications and Services in Wireless Networks workshop (AWSN), Boston, August 2004.

16. Sara Alouf, Eitan Altman, Chadi Barakat, Philippe Nain, "On the dynamic estimation of multicast group sizes", in proceedings of the International Symposium on Mathematical Theory of Networks and Systems (MTNS), Leuven, Belgium, July 2004.

17. Fatma Louati, Chadi Barakat, Walid Dabbous, "Adaptive Class-based Queuing for handling Two-Way traffic in Asymmetric Networks", in proceedings of the conference on High Speed Networks and Multimedia Communications (HSNMC), Toulouse, France, July 2004.

18. Eitan Altman, Kostya Avrachenkov, Chadi Barakat, Arzad-Alam Kherani, Balakrishna Prabhu, "Analysis of Scalable TCP", in proceedings of the conference on High Speed Networks and Multimedia Communications (HSNMC), Toulouse, France, July 2004.

19. Mohammad Malli, Qiang Ni, Thierry Turletti, Chadi Barakat, "Adaptive Fair Channel Allocation for QoS Enhancement in IEEE 802.11 Wireless LANs", in proceedings of IEEE ICC, Paris, June 2004.

20. Raja Abdelmoumen, Chadi Barakat, "Analysis of TCP latency over wireless links supporting FEC/ARQ-SR for error recovery", in proceedings of IEEE International Conference on Communications (ICC), Paris, June 2004.

21. Hung Nguyen, Patrick Thiran, Chadi Barakat, "On the correlation of TCP traffic in backbone networks", in proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), Vancouver, Canada, May 2004.

22. Chadi Barakat, Gion Reto Cantieni, "A distance-aware model of 802.11 MAC layer", in proceedings of WiOpt workshop : Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (extended abstract), Cambridge (UK), March 2004.

23. Eitan Altman, Chadi Barakat, Victor M. Ramos R., "Analysis of AIMD protocols over paths with variable delay", in proceedings of IEEE INFOCOM, Hong Kong, March 2004.

24. Qiang Ni, Imad Aad, Chadi Barakat, Thierry Turletti, "Modeling and Analysis of Slow CW Decrease for IEEE 802.11 WLAN", in proceedings of the International Symposium on Personnel, Indoor and Mobile Radio Communications (PIMRC), Beijing, China, September 2003.

25. Urtzi Ayesta, Kostya Avranchenkov, Eitan Altman, Chadi Barakat, Parijat Dube, "Multilevel Approach for Modeling Short TCP Sessions", in proceedings of ITC-18, Berlin, September 2003.

26. Victor Ramos, Chadi Barakat, Eitan Altman, "A Moving Average Predictor for Playout Delay Control in VoIP", in proceedings of the Eleventh International Workshop on Quality of Service (IWQoS 2003), Monterey, CA, June 2003.

27. Sara Alouf, Eitan Altman, Chadi Barakat, Philippe Nain, "Estimating Membership in a Multicast Session", in proceedings of ACM SIGMETRICS, San Diego, CA, June 2003.

28. Alaeddine Al Fawal, Chadi Barakat, "Simulation-based study of link-level hybrid FEC/ARQ-SR for wireless links and long-lived TCP traffic", in proceedings of WiOpt'03 : Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, Sophia Antipolis, France, March 2003.

29. Rares Serban, Chadi Barakat, Walid Dabbous, "Dynamic Resource Allocation in Core Routers of a Diffserv Network", in proceedings of the Asian Computing Science Conference (ASIAN), Hanoi, Vietnam, December 2002.

30. Chadi Barakat, Patrick Thiran, Gianluca Iannaccone, Christophe Diot, Philippe Owezarski, "A flow-based model for Internet backbone traffic", in proceedings of ACM SIGCOMM Internet Measurement Workshop, Marseille, France, November 2002.

31. Chadi Barakat, Patrick Thiran, Gianluca Iannaccone, Christophe Diot, "On Internet backbone traffic modeling", extended abstract in proceedings of ACM SIGMETRICS, Marina Del Rey, CA, June 2002.

32. Eitan Altman, Kostya Avrachenkov, Chadi Barakat, "TCP network calculus : The case of large bandwidth-delay product", in proceedings of IEEE INFOCOM, NY, June 2002.

33. Eitan Altman, Chadi Barakat, Victor Ramos, "On the utility of FEC mechanisms for audio applications", in proceedings of the international workshop on Quality of future Internet Services (QofIS), Coimbra, Portugal, September 2001.

34. Eitan Altman, Kostya Avrachenkov, Chadi Barakat, Rudesindo Nunez Queija, "TCP modeling in the presence of nonlinear window growth", in proceedings of ITC-17, Salvador da Bahia, Brazil, September 2001.
35. Chadi Barakat, Eitan Altman, "Bandwidth Tradeoff between TCP and Link-Level FEC", in proceedings of IEEE International Conference on Networking, Colmar, France, July 2001.
36. Eitan Altman, Kostya Avrachenkov, Chadi Barakat, Rudesindo Nunez Queija, "State-dependent M/G/1 Type Queueing Analysis for Congestion Control in Data Networks", in proceedings of IEEE INFOCOM, Anchorage, Alaska, April 2001. A more detailed version : CWI report PNA-R0005, July 2000.
37. Eitan Altman, Chadi Barakat, Victor Ramos, "Queueing Analysis of Simple FEC Schemes for IP Telephony", in proceedings of IEEE INFOCOM, Anchorage, Alaska, April 2001.
38. Eitan Altman, Chadi Barakat, Emanuel Laborde, Patrick Brown, Denis Collange, "Fairness Analysis of TCP/IP", in proceedings of IEEE Conference on Decision and Control, Sydney, Australia, December 2000.
39. Eitan Altman, Kostya Avrachenkov, Chadi Barakat, Parijat Dube, "TCP over a Multi-State Markovian Path", in proceedings of the international conference on the Performance and QoS of Next Generation Networking (P&QNet), Nagoya, Japan, November 2000.
40. Chadi Barakat, Eitan Altman, "On ACK Filtering on a Slow Reverse Channel", in proceedings of the first international workshop on Quality of future Internet services, Berlin, Germany, September 2000. **Best PhD Student Paper.**
41. Chadi Barakat, Eitan Altman, "A Markovian Model for TCP Analysis in a Differentiated Services Network", in proceedings of the first international workshop on Quality of future Internet services, Berlin, Germany, September 2000.
42. Eitan Altman, Kostya Avrachenkov, Chadi Barakat, "A Stochastic Model of TCP/IP with Stationary Random Losses", in proceedings of ACM SIGCOMM, Stockholm, Sweden, August 2000.
43. Eitan Altman, Kostya Avrachenkov, Chadi Barakat, " TCP in presence of bursty losses", in proceedings of ACM SIGMETRICS, Santa Clara, California, June 2000.
44. Chadi Barakat, Eitan Altman, "Performance of Short TCP Transfers", in proceedings of Networking 2000 (Performance of Communication Networks), Paris, May 2000.
45. Chadi Barakat, Eitan Altman, "Analysis of TCP with Several Bottleneck Nodes", in proceedings of IEEE GLOBECOM (Global Internet Symposium), Rio, Brazil, December 1999. A more detailed version : INRIA Research Report RR-3620, February 1999.
46. Chadi Barakat, Eitan Altman, "Analysis of the Phenomenon of Several Slow Start Phases in TCP", extended abstract in proceedings of ACM SIGMETRICS, Santa Clara, California, June 2000. A more detailed version : INRIA Research Report RR-3574, December 1998.

47. Chadi Barakat, Nesrine Chaher, Walid Dabbous, Eitan Altman, "Improving TCP/IP over Geostationary Satellite Links", in proceedings of IEEE GLOBECOM (General Conference), Rio, Brazil, December 1999. A more detailed version : INRIA Research Report RR-3573, December 1998.

Papers in Proceedings of Non-Refereed Conferences and Magazines

1. Eitan Altman, Konstantin Avrachenkov, Chadi Barakat, Kartikeya Chandrayana, "Multi-level approach for the modeling of large TCP/IP networks", invited to the Symposium on Recent Advances for Modeling and Simulation Tools for Communications Networks and Services, organized by COST 285 in Surrey, UK.
2. Chadi Barakat, "Evaluation des performances des réseaux WIFI 802.11", Techniques de l'Ingénieur, Lettre réseaux sans fil, Numéro Novembre-Décembre 2006.
3. Gion Reto Cantieni, Gianluca Iannaccone, Chadi Barakat, Christophe Diot, Patrick Thiran, "Reformulating the monitor placement problem : Optimal Network-wide Sampling", invited to the Conference on Information Science (CISS), Princeton NJ, March 2006.
4. E. Altman, C. Barakat, S. Mascolo, N. Moller, and J. Sun, "Analysis of TCP Westwood+ in high speed networks", in proceedings of PFLDnet'06 : workshop on Protocols for Fast Long-Distance networks, Nara, Japan, February 2006.
5. Rares Serban, Chadi Barakat, Walid Dabbous, "Enhancing CBQ with a Dynamic Bandwidth Allocation Module", in proceedings of Setit 2003, Tunisia, March 2003.
6. Urtzi Ayesta, Kostya Avranchenkov, Eitan Altman, Chadi Barakat, Parijat Dube, "Multilevel Approach for Modeling TCP/IP", in proceedings of PFLDnet'03 : workshop on Protocols for Fast Long-Distance networks, CERN-Geneva, Switzerland, February 2003.
7. Kostya Avratchenkov, Urtzi Ayesta , Eitan Altman, Philippe Nain, Chadi Barakat, "The effect of router buffer size on the TCP performance", in proceedings of LONIIS workshop on Telecommunication Networks and Teletraffic Theory, St. Petersburg, Russia, January 2002.
8. Chadi Barakat, Eitan Altman, "Impact of Network Buffers on TCP Start-Up", in proceedings of IEEE Middle East Workshop on Networking, Beirut, Lebanon, November 1999.
9. Chadi Barakat, Jean-Louis Rougier, "Optimization of Hierarchical Multicast Trees in ATM Networks", in proceedings of IFIP ATM'98, Ilkley, UK, July 1998.

Research Reports, Submitted Papers

1. Mohamad Jaber, Chadi Barakat, "Iterative Packet-Based Approach For Early Application Identification", submitted for publication, December 2008.

2. Anwar Al Hamra, Arnaud Legout, Chadi Barakat, "Understanding the properties of the overlay generated by BitTorrent", submitted for publication. INRIA report available at http ://hal.inria.fr/inria-00140937.
3. Amir KRIFA, Mohamed Karim SBAI, Chadi BARAKAT, Thierry TURLETTI, "BitHoc : A content sharing application for Wireless Ad hoc Networks", submitted for publication, November 2008.
4. Mohamed Karim Sbai, Emna Salhi, Chadi Barakat, "Adaptive overlay for P2P membership management in MANET", INRIA technical report inria-00342691, November 2008.
5. Mohamed Karim SBAI, Chadi BARAKAT, "Revisiting content sharing in wireless ad hoc networks", submitted, December 2008.

Thesis

1. Chadi Barakat, “Performance Evaluation of TCP Congestion Control”, PhD thesis, April 2001.
2. Chadi Barakat, "Le Routage Multicast Hiérarchique", DEA Thesis, ESSI, University of Nice Sophia-Antipolis, France, June 1998.
3. Chadi Barakat, Rani Makki, Hayan Mahfouz, "Design and Implementation of a Video Tracker", Electrical and Electronics Engineering Degree, Lebanese University, Beirut, Lebanon, July 1997.

Annexe A

Reformulating the Monitor Placement Problem : Optimal Network-Wide Sampling

Gion Reto Cantieni, Gianluca Iannaccone, Chadi Barakat, Christophe Diot, Patrick Thiran

A short version was invited to the Conference on Information Science (CISS), Princeton NJ, March 2006. A long version appeared in proceedings of CoNext, Lisboa, Portugal, December 2006.

Reformulating the Monitor Placement Problem: Optimal Network-Wide Sampling

Gion Reto Cantieni, Gianluca Iannaccone, Chadi Barakat, Christophe Diot, Patrick Thiran *

ABSTRACT

Confronted with the generalization of monitoring in operational networks, researchers have proposed placement algorithms that can help ISPs deploy their monitoring infrastructure in a cost effective way, while maximizing the benefits of their infrastructure. However, a static placement of monitors cannot be optimal given the short-term and long-term variations in traffic due to re-routing events, anomalies and the normal network evolution. In addition, most ISPs already deploy router embedded monitoring functionalities. Despite some limitations (inherent to being part of a router), these monitoring tools give greater visibility on the network traffic but raise the question on how to configure a network-wide monitoring infrastructure that may contain hundreds of monitoring points.

We reformulate the placement problem as follows. Given a network where all links can be monitored, which monitors should be activated and which sampling rate should be set on these monitors in order to achieve a given measurement task with high accuracy and low resource consumption? We provide a formulation of the problem, an optimal algorithm to solve it, and we study its performance on a real backbone network.

1. INTRODUCTION

Network operators perform traffic measurements as part of their day by day network management activities that include traffic engineering, anomaly detection, accounting and capacity planning. There exist several ways to perform traffic measurements. Some involve router support (e.g., SNMP counters, Netflow [4]), while others require additional equipment to be installed in the network to perform passive or active measurements.

The various solutions present a trade-off between the accuracy of the measurement and the amount of computing resources they require. SNMP counters, for example, represent a very low cost solution (in terms of router processing resources) but give the operator only a rough idea on the traffic that is traversing the network. The aggregate counters are of little use to operators interested in users' perceived performance [16] or in estimating network traffic demands [20]. At the other extreme, passive monitoring equipment, that captures every packet on a link, allows extremely accurate measurements, but scales very poorly for large networks, given the high unit cost for deployment and maintenance. In addition, space and power constraints may prevent operators from deploying this equipment where it would be the most useful, while, once deployed, routing or network changes may render it useless.

The solution that many network operators have adopted is then to use Netflow [4] or similar solutions [23, 19]. NetFlow allows a ubiquitous deployment of traffic monitors and provides a detailed enough view of the traffic streams. Netflow is today widely used by Internet Service Providers (ISPs) and several tools exist to process, analyze and visualize NetFlow data [9, 24]. However, enabling NetFlow can have an impact on packet forwarding performance. To address this problem, router vendors have introduced versions of Netflow that sample the incoming packets and update the flow information only with sampled packets.

Network operators then face two options: (i) enable Netflow on all routers but using very low sampling rates to minimize potential network impact, or, (ii), enable Netflow on a chosen set of routers where the sampling rates are set depending on the measurement task and the target accuracy.

Currently the first option is the one followed by ISPs because no automated method exists for the second. This work aims at filling this gap.

The contributions of the paper are threefold. First, we define a general framework to approach the problem of sampling traffic data in large IP networks. Our framework allows to combine and solve in one step the selection of traffic monitors and the setting of the sampling rates for each monitor. We show how this framework can be applied to a general class of measurement tasks. Second, we provide an optimal algorithm to solve the sampling and placement problem. We validate the algorithm using network data collected from the GEANT's backbone network [10]. Finally, we discuss how to deploy our solution in a real backbone network and introduce additional methods that allow to adapt the sampling rates to changes in the traffic due to time-of-day effects, failures or anomalies.

*G.Cantieni and P.Thiran are with EPFL, Switzerland. G.Iannaccone is with Intel Research, Cambridge UK. C.Barakat is with INRIA Sophia Antipolis, France. C. Diot is with Thomson Research, Paris, France.

The paper is organized as follows. In Section 2, we describe the objectives and challenges of this work. Section 3 presents related work. Sections 4 and 5 formally define the problem and our approach to solve it. Section 6 evaluates the performance of our algorithm in the GEANT backbone network. Section 7 describes how to deploy our solution in a real backbone and how to deal with traffic changes when the measurement task runs in a continuous fashion. Section 8 concludes the paper.

2. OBJECTIVES AND CHALLENGES

The nature of the measurement task has a clear impact on the choice of traffic monitors to be used. For example, in order to estimate the traffic demands, a network operator would ideally monitor all ingress links in the network or at least all peering links [8]. However, when the network operators want to focus on an individual network prefix or Autonomous System (AS), they may require a very different layout of monitors in the network.

Very often, network operators do not have prior knowledge of the measurement tasks the monitoring infrastructure will have to perform. This is particularly true with security applications. For example, a specific network prefix that is "below the radars" for traffic engineering purposes may play an important role in the early detection of anomalies.

Furthermore, network traffic demands are subject to short term variations due to failures and other anomalous events as well as longer term variations due to the addition of new customers, peering links, Points of Presence, etc. These changes quickly make a static placement of traffic monitors perform sub-optimally.

For these reasons, Internet Service Providers (ISP) prefer widespread monitoring infrastructures that provide visibility over the entire network. Netflow [4] is a perfect example of such monitoring system. Embedded into the routers, it maintains a list of flow records that describe the traffic forwarded by the router. The flow records are then exported to a collector for analysis and storage. Netflow has become today the de-facto standard of flow monitoring solutions: it is the most widely deployed and various router manufactures support compatible monitoring applications [19]. A standardization effort is also in progress at the IETF [15].

ISPs configure Netflow on all routers to the same "safe" sampling rate — router vendors usually recommend a value of 1/1000 packets — while little attention is paid to the accuracy of the results. Low sampling rates reduce the stress on the routers but introduce large errors in the measurement.

Clearly, this approach leads to an inefficient use of the resources both in the routers and in the collector, and to a less than desirable measurement accuracy (we will quantify this in Section 6.3). Our objective is to find a method that, given a measurement task and a target measurement accuracy, has the following properties:

- It selects the monitors that need to participate in the measurement. This reduces the processing overhead on the collector side where it is desirable that a large proportion of the records received, processed and stored are actually relevant to the measurement task.
- It provides a set of sampling rates for the active monitors to guarantee an optimal use of the resources while making sure that the sampling rates are low enough to not be a concern for the operator.
- The resource consumption, as measured by the total number of sampled and processed packets, is minimal and adapts to the changes in the traffic. The objective is to maintain the overall resource consumption stable while keeping the accuracy close to target.
- The method requires a minimal amount of configuration from the network operator and is robust to incorrect input parameters.

In addition, our method should support a general class of monitoring applications and be easily adapted to new measurement tasks. We choose one example of such tasks to illustrate the contributions of this paper. We find the sampling rates to estimate the amount of traffic flowing among a set of origin-destination pairs selected by the network operator. In our terminology, origin and destination could refer to any combinations of end-hosts, network prefixes, Autonomous Systems, etc. We have chosen this task because it helps in illustrating our contribution and it is, at the same time, a canonical measurement task for classical traffic engineering, security and accounting applications.

3. RELATED WORK

Identifying the strategic locations for traffic monitors is a hard problem that has attracted significant interest in the literature. Several solutions have been proposed for different contexts. For example, in [17], the authors focus on the placement of measurement devices for active monitoring (more specifically for the construction of distance maps). Others have addressed the placement problem in an active monitoring infrastructure to measure delays and detect link failures [1, 13, 21].

In the passive monitoring domain, Suh et al. [28] address the problem of placing monitors and set their sampling rates in order to maximize the fraction of IP flows being sampled. They propose a two phase approach where they first find the links that should be monitored and then run a second optimization algorithm to set the sampling rates. Their approach bears some similarities with our work, but the analysis is limited to a generic monitoring goal (maximize the overall sampled traffic) and only considers a static placement of monitors. Their formulation leads to a set of heuristics that find near-optimal solutions. Our approach, instead, allows to indicate whether a solution corresponds to the global optimum.

Many researchers have also explored ways to improve NetFlow to automate some of the features and help network operators configure the routers. Estan et al. [7] propose a set of techniques to let a router running NetFlow adapt the sampling rate in order to keep a fixed resource consumption. The adaptation is a local and independent decision of the router, and is not tied to any measurement objective. Our work is complementary to [7] in the sense that it provides a global sampling strategy for a specific monitoring goal. The individual routers could then apply local decisions in order to minimize their memory usage.

There is a large body of literature that addresses the problem of inversion of traffic properties from sampled traffic [6, 5, 12]. Duffield et al. [6] show that periodic and random sampling provide roughly the same result on high speed links. Random sampling can thus be used in the mathematical analysis for its appealing features.

Traffic matrix estimation techniques (e.g., [20, 29, 26]) address a measurement task similar to the one we are considering as example. However, the focus in those works is the inference of the traffic matrix from partial information (e.g., link loads). Indeed, they often use sampled Netflow data to validate their methods.

4. PROBLEM FORMULATION

In this section we formalize the placement problem. Our approach to solve it will be described in the next section.

We represent the network by a directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ corresponds to the set of nodes and $\mathcal{E}$ is the set of edges. The traffic load on edge $e \in \mathcal{E}$ is denoted by U_e. The routing of each origin-destination pair (OD pair) is specified by the routing matrix R, whose entries $r_{i,j} = 1$ if the OD pair i traverses edge j and 0 otherwise.

The measurement tasks are defined over a set $\mathcal{F} = \{1, 2, \ldots, F\}$ of OD pairs. We indicate the subset of links traversed by the OD pairs in $\mathcal{F}$ by $\mathcal{L} \subseteq \mathcal{E}$. The optimization framework we propose in this paper is general and can be applied to any definition of node (e.g., end-host, network prefix, Autonomous System, etc.).

The quality of a measurement for an OD pair $k \in \mathcal{F}$ is computed via a utility function $M : \mathbb{R}_+ \to \mathbb{R}_+$, whose argument ρ_k is the *effective* sampling rate of OD pair $k \in \mathcal{F}$, defined as the probability that a packet of the kth OD pair is sampled at least once by at least one monitor deployed in the network. Note that this definition assumes that we have means to discern whether the same packet is sampled at multiple locations in the network. In Section 6, we will address this aspect of the monitoring infrastructure in more detail. Assuming that the packets are sampled in an i.i.d. (independent and identically distributed) fashion at each monitor, with p_i denoting the packet sampling probability of the monitor deployed on link i, and that the sampling processes of different monitors are statistically independent, we have that

$$\rho_k = 1 - \prod_{i \in \mathcal{L}} (1 - p_i)^{r_{k,i}}. \tag{1}$$

Clearly, the larger the effective sampling rate, the more information it brings. However, the marginal rate of information is usually smaller for large values of ρ_k than for small values. These two observations lead us to reasonably assume that $M(\rho_k)$ is an increasing and strictly concave function of ρ_k.

Our objective is to choose the vector of sampling rates $\boldsymbol{p} = (p_i)_{i \in \mathcal{L}}$ that maximizes

$$\sum_{k \in \mathcal{F}} M\left(\rho_k\left(\boldsymbol{p}\right)\right), \tag{2}$$

under the constraints

$$p_i \geq 0 \quad \text{for all } i \in \mathcal{L} \tag{3}$$

$$p_i \leq \alpha_i \quad \text{for all } i \in \mathcal{L} \tag{4}$$

$$\sum_{i \in \mathcal{L}} p_i U_i \leq \theta, \tag{5}$$

where θ is the *capacity* of the system, defined as the maximum total number of packets that can be sampled in the entire network, and α_i represents the maximum sampling rate that can be applied to the individual link i.

All constraints are linear and therefore define a convex solution space Ω defined by $\{\boldsymbol{p} \mid \sum_{i \in \mathcal{L}} p_i U_i \leq \theta, \quad 0 \leq p_i \leq \alpha_i \quad \forall i \in \mathcal{L}\}$. As the utility function M is strictly concave, the optimization problem, given by (2), (4) and (5), has a unique maximizer, that we denote $\boldsymbol{p}^*$ (see e.g., [3, Chapter 2]).

Note that an alternative objective could be to maximize the minimum of the utilities, i.e., $min_{k \in \mathcal{F}} M(\rho_k(\boldsymbol{p}))$. The two formulations have their advantages and limitations. Maximizing the sum of utilities gives us more flexibility in setting the sampling rates. Indeed, we can compensate the poor accuracy of one OD pair increasing the accuracy of another. We will discuss the implications of this in greater detail in Section 6. On the other hand, maximizing the minimum utility may lead to increase significantly the sampling rate on those links that carry small OD pairs. Furthermore, the minimum of the utilities is not a differentiable function over the whole parameter space and this may impact the convergence of the algorithm (see Section 5.4). We leave the study of alternative objective functions for future work.

5. METHOD

In this section we first reformulate the optimization problem using Lagrange multipliers. Second, we introduce and comment on the assumptions that simplify the computation of the optimal solution. Third, we discuss the choice of a suitable utility function $M(\cdot)$. Last, we briefly present the algorithm used to solve the optimization problem and discuss its advantages and pitfalls.

5.1 Our approach

The most common approach to solve a constrained optimization problem, such as the one in (2), (3), (4) and (5), is to define the corresponding Lagrangian:

$$\begin{aligned} L(\boldsymbol{p}, \lambda, \boldsymbol{\mu}, \boldsymbol{\nu}) \quad = \quad & \sum_{k \in \mathcal{F}} M\left(\rho_k(\boldsymbol{p})\right) - \lambda \left(\sum_{i \in \mathcal{L}} p_i U_i - \theta \right) \\ & - \sum_{i \in \mathcal{L}} \mu_i (p_i - \alpha_i) + \sum_{i \in \mathcal{L}} \nu_i p_i, \end{aligned} \tag{6}$$

where $(\lambda, \boldsymbol{\mu}, \boldsymbol{\nu}) = (\lambda, \mu_i, \nu_i)_{i \in \mathcal{L}}$ is the set of Lagrange multipliers. Each Lagrange multiplier enforces the satisfaction of one of the constraints (3), (4) and (5) with an equality sign. A constraint met with an equality sign is called an *active constraint*. For example, if $p_i = \alpha_i$, then the ith constraint (4) is active, whereas if $p_i < \alpha_i$, it is inactive.

In order to find the unique maximizer $\boldsymbol{p}^*$, we can solve the system of equations provided by the Karush-Kuhn-Tucker (KKT) conditions [11, Chapter 5]. As the solution space is a convex hull and the objective function is concave, the KKT conditions are sufficient for optimality (see e.g., [3, Chapter 5.5]). The difficulty in using the KKT conditions to solve the problem given by (2), (3), (4) and (5) is that it requires to know the set of active and inactive constraints in advance, which is not possible. Therefore, we need to rely on an iterative procedure, such as a gradient projection method to explore the solution space. Our approach to find the optimal solution $\boldsymbol{p}^*$ is given in more details further below.

5.2 Assumptions

Let us recall the assumptions made so far: the utility function M is strictly increasing and concave; each monitor sampling process is i.i.d. and is statistically independent from the sampling process of the other monitors.

From a practical perspective, we expect to obtain sampling rates that are in the order of 0.01 and lower. Moreover, we rarely expect to have more than one or two monitors observing the traffic of the same OD pair. This allows us to approximate the effective sampling rate (1) by

$$\rho_k = \sum_{i \in \mathcal{L}} r_{k,i} p_i. \tag{7}$$

This set of assumptions allows to make the optimization problem more tractable and, as we show in Section 6.2, have a minor impact on the performance of our method.

Next, we force the constraint (5) to be met with an equality sign. This assumption is pretty straightforward to make, as there is no practical interest not to use all the provided resources. Hence (5) becomes

$$\sum_{i \in \mathcal{L}} p_i U_i = \theta. \tag{8}$$

5.3 Choosing the utility function

The choice of the utility function $M(\rho_k)$ is dictated by the following conditions. First, the utility function has to quantify the information provided by the measurement for each OD-flow in $\mathcal{F}$. Second, the function has to comply with the requirements set by the optimization framework: as mentioned earlier, it is strictly increasing and concave. Without loss of generality, we assume that $M(0) = 0$, i.e. that the utility is zero if no packet is sampled at all. Third, the function M must be easy to use in our optimization algorithm. This requires the function to be twice continuously differentiable. We next derive a possible function which combines the above specified properties.

A first straightforward choice for the utility function M is to use the relative error between the actual metric of interest that we want to measure (in the present case, the flow size) and its value estimated from the sampled packets. Let S_k be the actual size of the kth OD pair (number of packets of the kth OD pair) in a given time interval, and X_k be the number of sampled packets from this OD pair in the same time interval. Because of the assumptions that the sampling processes at different monitors are independent and that packets are sampled at most once (Section 5.2), the distribution of X_k conditionally to S_k is binomial with parameters (S_k, ρ_k). In this paper, we consider the squared relative error (SRE) between the estimated size X_k/ρ_k and the actual size of the flow S_k, i.e.

$$\text{SRE} = \left(\frac{X_k/\rho_k - S_k}{S_k} \right)^2, \tag{9}$$

whose expected value is

$$\mathbb{E}[\text{SRE}](\rho_k) = \int_0^\infty \frac{1-\rho_k}{\rho_k s} d\mathbb{P}(S_k \leq s) = \mathbb{E}\left[\frac{1}{S_k}\right]\left(\frac{1}{\rho_k} - 1\right).$$

Define $A(\rho_k) = 1 - \mathbb{E}[\text{SRE}](\rho_k)$, that we call the mean squared relative accuracy. A possible candidate for $M(\rho_k)$ would be to take it equal to $A(\rho_k)$, since it is a strictly increasing, concave function of ρ_k. However, this function is not yet adequate to be used as utility function M. The problem is that the function $A(\rho_k)$ is not defined at the origin. This is required, as we expect to have zero utility for zero sampling. To fix this problem, we divide the interval $[0, 1]$ in two intervals $[0, x_0]$ and $[x_0, 1]$. We take $M(\rho_k) = 1 - \mathbb{E}[\text{SRE}](\rho_k)$ if $x_0 \leq \rho_k \leq 1$ and $M(\rho_k) = A'(\rho_k)$ for $0 \leq \rho_k < x_0$, where A' is defined in such a way that it is strictly concave and increasing and twice differentiable over $[0, x_0]$, with $A'(0) = 0$, $A'(x_0) = A(x_0)$, $\partial A'(x_0)/\partial \rho_k = \partial A(x_0)/\partial \rho_k$ and $\partial^2 A'(x_0)/\partial \rho_k^2 = \partial^2 A(x_0)/\partial \rho_k^2$. A function that complies with these requirements is the quadratic expansion of $A(\rho_k)$ at x_0, that reads

$$A'(\rho_k) = A(x_0) + (\rho_k - x_0)\frac{\partial A}{\partial x}(x_0) + \frac{(\rho_k - x_0)^2}{2}\frac{\partial^2 A}{\partial x^2}(x_0),$$

where x_0 is given by the relation $A'(0) = 0$, that is,

$$A(x_0) - x_0 \frac{\partial A}{\partial x}(x_0) + \frac{x_0^2}{2}\frac{\partial^2 A}{\partial x^2}(x_0) = 0.$$

To summarize, we define the utility function M as

$$M(\rho_k) = \begin{cases} A'(\rho_k) & \text{if } 0 \leq \rho_k \leq x_0 \\ A(\rho_k) & \text{if } x_0 \leq \rho_k \leq 1. \end{cases} \tag{10}$$

We plot the function $M(\rho_k)$ for $\mathbb{E}[1/S_k] = 0.002$ and for $\mathbb{E}[1/S_k] = 0.0002$ in Figure 1. We emphasize that this manipulation on the function has the sole purpose of making it suitable for our optimization framework. The interval $0 \leq \rho_k < x_0$, on which we use the quadratic expansion $A'(\rho_k)$, is of little practical interest because it corresponds to low values of utility. Hence, we expect that this necessary manipulation does not affect much the optimization results. We will validate this observation in Section 6.

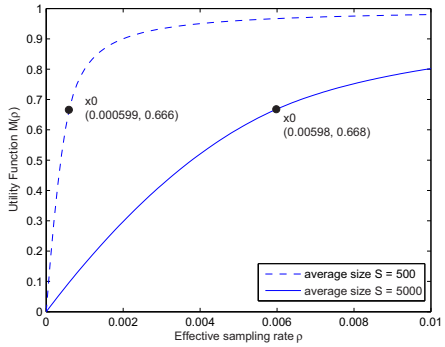

Figure 1: The utility function M with respect to the effective sampling rate ρ_k. The utility function is defined over the interval [0,1], such as given in (10).

5.4 The algorithm

We solve our optimization problem by use of the gradient projection method for constrained optimization (Refer to [11, Chapter 5]). At each iteration step n, this method consists in projecting first the gradient of the objective function onto the subspace spanned by the active constraints (that is, the set of points $\boldsymbol{p}$ that satisfy the active constraints). This projected gradient gives the search direction $\boldsymbol{s}(n)$, along which the current feasible solution $\boldsymbol{p}(n)$ is moved until either the objective function is maximized along this line or an inactive constraint is hit. In the latter case, this inactive constraint has to be activated and incorporated into the next computation of the gradient projection. In the former case, the maximization of the objective function along $\boldsymbol{s}(n)$

reduces to a one-dimensional search. We choose Newton's method (refer to [3], chapter 9.5) to perform this search. Newton's method shows fast convergence but requires the objective function to be twice continuously differentiable. Once the new solution $\boldsymbol{p}(n+1)$ that maximizes the objective along $\boldsymbol{s}(n)$ is found, a new search direction $\boldsymbol{s}(n+1)$ has to be computed. The new search direction will be orthogonal to the previous one. The successive search directions $\boldsymbol{s}(n), \boldsymbol{s}(n+1), \ldots$ form therefore a zigzag path in the subspace spanned by the active constraints, which may result in a poor convergence depending on the shape of this subspace. A better approach is to add, with some weighting factor, the previous search direction to the new one. This weighting factor is usually chosen according the Polak-Ribiere rule. More details can be found e.g. in [18].

We start our search with a feasible solution $\boldsymbol{p}(0)$ arbitrarily chosen on the plane defined by the active constraints (5). We then follow the search direction $\boldsymbol{s}(0)$, that coincides with the projection of the gradient of the objective function (2) on the subspace spanned by the active constraints, until we hit the constraint $p_1 \geq 0$. This forces us to activate this constraint, i.e. to set $p_1 = 0$, and to recompute next the search direction $\boldsymbol{s}(1)$. With Newton's method we finally find $\boldsymbol{p}(2)$.

We mentioned earlier that in order to find the optimal solution, we eventually have to know the set of active and inactive constraints of our optimization problem. That is, we have to know which monitors i are going to be used (whose corresponding constraint $p_i > 0$ is inactive) and which ones not (whose corresponding constraint $p_i = 0$ is active). This combinatorial explosion is unavoidable as the monitor placement problem is NP-hard. Probing each possible combination is not feasible in practice, hence we use the above introduced gradient method to find iteratively the optimum. However, we do not have any guarantee that the gradient projection method always converges to the optimum. It might well happen that the projection of the gradient $\boldsymbol{s}$ converges to a zero vector at some point, say $\boldsymbol{p}'$, which forces us to stop the search, even though $\boldsymbol{p}'$ is not optimal. This means that the algorithm has not generated the optimal set of active and inactive constraints, and thus that some monitors are turned on/off on suboptimal links. Given the feasible solution $\boldsymbol{p}'$ we have obtained, we apply the Karush-Kuhn-Tucker (KKT) conditions [11, Chapter 5] that determine whether this solution is optimal or not. To do so, we have to compute the Lagrange multipliers defined in (6). Having some of them negative indicates that the algorithm cannot converge to the optimum with the current set of active constraints. It may be possible to continue the search by removing (making inactive) some of the currently active constraints.

A strategy is to remove (make inactive) the subset of active constraints associated with negative Lagrange multipliers (a similar strategy is given in [11, Chapter 5]). After this operation, we recompute the gradient projection on the subspace spanned by the *new* set of active constraints and proceed with the search. We continue until we either reach a point that satisfies the KKT conditions and therefore is the optimal solution by KKT's theorem, or we abort the search because the number of iterations (a new iteration starts each time a new search direction has to be computed) exceeds a threshold we set on the maximum number of iterations. In our experiments described in Section 6, this threshold is set to 2000 to keep the execution time of the algorithm in the order of few seconds. We observe that in 98.6 percent of cases the optimum is found in less than 2000 iterations.

The performance of the gradient projection method can also be measured by the number of times we end up in the situation where we have to remove (make inactive) the active constraints associated with negative Lagrange multipliers. To provide a number, we conducted 200 independent executions of the algorithm, each time with a different set of input parameters (different OD pair sizes, different link loads, different capacity θ). The input parameters are given from the data set discussed in Section 6. Even though we cannot have a formal upper bound on the number of times we have to remove some active constraints, we observe an average of 1.64 situations per execution of the algorithm, where we have to remove (make inactive) the constraints associated with negative Lagrange multipliers, with a standard deviation of 1.17. These low numbers, that are specific to the monitoring problem we solve, justify our choice of the gradient projection method to solve efficiently our optimization problem.

6. EVALUATION

We study the performance of our method by defining and simulating a measurement task on GEANT, the European Research network [10]. Our goal is to verify the following properties of the solution we obtain with the method described in Section 5: (*i*) the sampling rates are low and validate the approximation in (7); (*ii*) the solution results in a fair allocation of resources to each OD pair; (*iii*) the solution is clearly superior to other solutions that could be derived without running any optimization algorithm.

6.1 Data

We use sampled Netflow data collected on all the 23 routers of the GEANT network. The links have varying speeds from OC-3 (155 Mbps) to OC-48 (2.5 Gbps). We also collect in a continuous fashion BGP and ISIS updates and use them to build GEANT routing matrix.

Every GEANT router has NetFlow-compatible monitoring capabilities [19] enabled with a sampling rate of 1/1000. The packets are classified by the 5-tuple (source and destination IP address, source and destination port number and protocol number) and the flow records are exported every minute by the routers. Each record contains the following information (in addition to the 5-tuple) that is relevant to this study: (*i*) *Flow start and end time*. The start time is the timestamp of the first sampled packet of the flow. The end time is the timestamp of the last packet of the flow. Flow termination is triggered either by a FIN packet or by an idle timeout (set to 30 seconds). (*ii*) *Sampled packets and bytes*. The total number of sampled packets in the flow and their cumulative size in bytes. (*iii*) *Source and Destination Autonomous System (AS)*. The AS numbers to which the source and destination IP addresses belong. (*iv*) *Input and output interface*. The index of the router interface on which the flow was received and sent.

Before using NetFlow data, we need to perform some additional post-processing of the records. First, we aggregate all flow records (exported by the routers every minute for all active flows) in 5 minutes bins according to their start time. We choose 5 minutes as our measurement interval to reduce the impact of synchronization issues that could have arisen when collecting flow data from different routers. Then, we

adjust the sampled packet and byte count by multiplying them by the inverse of the sampling rate (1000 in our case).

Henceforth, we consider the post-processed NetFlow data to represent the *actual traffic* traversing the GEANT network. Our experiments then consist in simulating a sampling process with the rates set by the optimal algorithm and comparing the results with the post-processed data. Although our validation does not depend on a perfect reconstruction of the traffic dynamics, the sampled Netflow data present a potential bias against small flows that can affect the relative contribution of each OD pair of interest. Unfortunately, this is the only type of data available today for research and we are not aware of any other public dataset that contains full unsampled information for a network of the size of GEANT.

6.2 Results

For the evaluation we choose to estimate the traffic sent by JANET (UK Research Network, AS number 786) to each individual GEANT PoP through the UK PoP.

This task gives us a set $\mathcal{F}$ of 20 OD pairs. The OD pairs traverse 22 of the 72 unidirectional links of GEANT. We use flow data from November 22nd, 2004 where we associate to each flow record the egress PoP, computed from the destination IP address using the technique presented in [8]. The first two columns of Table 1 present a summary of the OD pairs and their sizes in packets/sec. The results in this section are computed over a single measurement interval.

This task shows that the method operates on OD pairs where origins (i.e., the JANET AS) and destinations (i.e, all the GEANT PoPs) are of a different nature. In addition, the OD pair sizes cover the entire spectrum: JANET to Netherlands (NL) consist of more than 30,000 packets/sec while JANET to Luxembourg is made of a mere 20 packets/sec. As it will become clear later, this is one of the main strengths of the method, i.e., the ability to track indifferently small and large OD pairs.

The remaining columns in Table 1 represent the sampling rates that the optimal solution provides. For each link i, we indicate the sampling rate p_i and the OD pairs that traverse (and are sampled on) that link. All links that are not present in the table have $p_i = 0$, i.e., those monitors do not need to participate in the measurement task. Finally, the last two rows of the table show the load in packets/sec on the links and the relative contribution to the total capacity. In this experiment we have chosen a value of $\theta = 100,000$, that is, at most $100,000$ packets can be sampled in each 5 minutes measurement interval in the entire network. We also set $\alpha_i = 1$ for all links, i.e., we do not define an upper limit for the sampling rates. Thus, we assume the operator has no prior knowledge of the network traffic.

We can immediately observe that the sampling rates are extremely low on most links even if no upper limit was set. Only two links (FR-LU and CZ-SK) need a sampling rates somewhat higher (around 0.9%), but they are lightly loaded links needed to accurately estimate the two smallest OD pairs (JANET-SK and JANET-LU). Furthermore, the low sampling rates we obtain and the fact that each OD pair is sampled in at most two links validate the assumptions on the effective sampling rate made in Section 5.2.

The last two columns in Table 1 show the value of the utility function (10) and the accuracy of the measurement. We define the accuracy of an OD pair size as 1 minus the absolute relative error: $1- \mid x/\rho - s \mid /s$, where s is the actual size of the OD pair, x is the sampled size and ρ is the effective sampling rate as in (7).

We use the accuracy instead of the utility M (10) to validate the impact of two assumptions we made in the definition of M: (i) the quadratic expansion to force M to zero when the sampling rate is zero (see Section 5.3); (ii) the approximation (7) for the effective sampling rate ρ, that may result in overestimating the size of certain OD pairs.

We run 20 sampling experiments on the flow records and compute the average accuracy over the 20 runs. Each sampling experiment consists in simulating a random sampling process on the flow records observed on link i using the sampling rate p_i in Table 1. The values in those two columns demonstrate that the method achieves good fairness among OD pairs. Although the algorithm maximizes the sum of the utilities, the results indicate that the individual utilities are well balanced. Moreover, the accuracy of the measurement is extremely good being on average above 0.89 for any OD pair.

6.3 Comparison with other solutions

The last aspect we want to address is how the optimal solution compares to naïve solutions. Clearly, if any solution performs well enough, then there is no reason to add the complexity of the optimization algorithm we propose.

The first naïve solution would consist in monitoring only the JANET access link to GEANT. This solution has the advantage that every sampled packet would belong to one of the OD pairs of interest. However, in order to track a small OD pair (e.g., JANET to Luxembourg) with a similar accuracy to the one we obtain in Table 1, the network operator would be forced to sample the link at a rate of about 1%, i.e., the effective sampling rate for JANET-LU. Given the high load on that link, this would require the capacity θ to be 70% higher than the one needed by our method to give the same measurement accuracy[1].

Furthermore, apart from being a suboptimal solution, monitoring the access link may not be feasible in all scenarios given that, in corporate networks for example, the edge routers (i.e., the router that is connected to the ISP) is often directly owned and managed by the ISP that provides network connectivity. These routers, usually called CPE (Customer Premise Equipment), can only be accessed by the ISP network operators. Therefore, in order to keep our method general and applicable to a wide range of network scenarios, we do not include the access links in the set of possible links to monitor. Even with this additional constraint, Table 1 shows that our method performs extremely well.

An alternative to the monitoring of the access link is to monitor all links that connect the UK PoP to the other PoPs in GEANT. This solution allows to "balance" the sampling rates over six links, instead of just one, and gives more freedom to reduce the resource consumption.

In order to compare this solution with the optimum we run our algorithm and restrict the choice of available monitors to just the six UK links. Figure 2 shows the comparison in terms of the accuracy over a wide range of values of the capacity θ. With respect to the optimum, this simple solu-

[1]Adding up the values in the second column of Table 1 we obtain 57,933 packets per second. At a sampling rate of 1%, this results in 173,798 sampled packets on average over a 5 minutes interval.

OD pair	pkt/s	p_5 UK-FR	p_7 UK-SE	p_8 UK-NL	p_9 UK-NY	p_{17} SE-PL	p_{30} UK-PT	p_{31} IT-IL	p_{33} FR-BE	p_2 FR-LU	p_{28} CZ-SK	Utility	Avg Accuracy
JANET-NL	30123	-	-	0.0016	-	-	-	-	-	-	-	0.9999	0.993
JANET-NY	9387	-	-	-	0.0002	-	-	-	-	-	-	0.9982	0.965
JANET-DE	4300	-	-	0.0016	-	-	-	-	-	-	-	0.9995	0.982
JANET-SE	4080	-	0.0003	-	-	-	-	-	-	-	-	0.9973	0.960
JANET-CH	4033	0.0013	-	-	-	-	-	-	-	-	-	0.9994	0.979
JANET-FR	1723	0.0013	-	-	-	-	-	-	-	-	-	0.9985	0.969
JANET-PL	1400	-	0.0003	-	-	0.0003	-	-	-	-	-	0.9960	0.950
JANET-GR	1080	-	-	0.0016	-	-	-	-	-	-	-	0.9981	0.964
JANET-ES	1003	0.0013	-	-	-	-	-	-	-	-	-	0.9974	0.959
JANET-SI	913	-	-	0.0016	-	-	-	-	-	-	-	0.9977	0.961
JANET-IT	873	0.0013	-	-	-	-	-	-	-	-	-	0.9971	0.956
JANET-AT	790	0.0013	-	-	-	-	-	-	-	-	-	0.9968	0.954
JANET-CZ	590	-	-	0.0016	-	-	-	-	-	-	-	0.9965	0.952
JANET-BE	490	0.0013	-	-	-	-	-	-	0.0002	-	-	0.9955	0.946
JANET-PT	463	-	-	-	-	-	0.0011	-	-	-	-	0.9935	0.937
JANET-HU	377	-	-	0.0016	-	-	-	-	-	-	-	0.9945	0.940
JANET-HR	237	-	-	0.0016	-	-	-	-	-	-	-	0.9912	0.924
JANET-IL	87	0.0013	-	-	-	-	-	0.0018	-	-	-	0.9877	0.910
JANET-SK	43	-	-	0.0016	-	-	-	-	-	-	0.0092	0.9929	0.932
JANET-LU	20	0.0013	-	-	-	-	-	-	-	0.0090	-	0.9840	0.897
Link Loads (pkt/s)		63603	51833	57756	37286	23680	19950	15213	11173	6133	2600		
Contribution to θ		24.5%	5.1%	26.9%	2.1%	2.1%	6.8%	8.3%	0.7%	16.5%	7.1%		

Table 1: Optimal sampling rates p_i for each link i. For each OD pair, it indicates the size (pkts/sec), the link(s) where it is monitored, their respective sampling rates and the average accuracy. All other links have zero sampling rate.

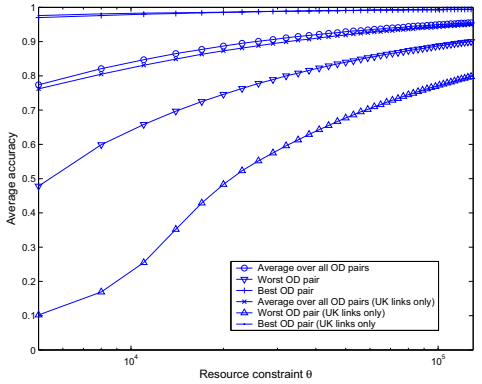

Figure 2: Accuracy of the measurement for two different solutions as a function of θ.

tion has poor performance with respect to small OD pairs. This is expected given that the UK links are heavily loaded and the high sampling rate required to accurately estimate a small OD pair results in a high resource consumption.

In summary, we have seen that both in terms of resource usage and measurement accuracy there is a clear advantage in using our method for setting the sampling rates when compared to other naïve solutions. This advantage comes from the network-wide approach to packet sampling. Indeed, the optimization method finds those links across the entire network, where the small OD pairs "manifest" themselves with a small amount of cross traffic from other large OD pairs. Several studies [2, 8] have shown that this is a general property of current network design.

7. DEPLOYMENT ON A BACKBONE NETWORK

Before deploying our method in a real backbone network, we need to address two additional issues: (i) how to set the input parameters and (ii) how to deal with traffic variations over time.

As described in Section 5, the input parameters consist of the quantities θ, $E[1/S_k]$ and U_i, for all $k \in \mathcal{F}$ and $i \in \mathcal{L}$. In order to obtain these quantities, the network operator needs to have some prior knowledge about the network. In particular, the operator needs to have a rough idea of the sizes, S_k, of the OD pairs that are under study. This is a classical bootstrapping problem and we will present a solution in Section 7.1.

The second deployment issue has to do with the temporal fluctuations of the OD pair sizes and link loads. In the flow data, we observe that these fluctuations may be very large and abrupt. This phenomenon is also more pronounced for small OD pairs and confirms previous measurement studies [27].

It is out of the scope of this paper to investigate the reasons behind these fluctuations. They may be due to normal time-of-day effects, failures in the network, flash crowd events or other anomalies. What is important to observe is that these fluctuations may lead our method to "overshoot" the capacity θ when the link loads increase abruptly, or to miss the target accuracy when an OD pair size decreases. Section 7.2 shows the impact of the fluctuations on the performance of our method, while Section 7.3 introduces a simple heuristic to effectively cope with them.

In the remainder of this section we will refer to flow data collected over a 12 hours period from 8AM to 8PM on November 22nd, 2004.

7.1 Bootstrapping phase

The bootstrapping method we propose is very simple. Given the presence of traffic monitors on every link in the network, an operator can get a rough estimate of the OD pair sizes and link loads by activating all monitors at once. The monitors should be activated at a very low sampling rate, that we call *background sampling rate* ($\tilde{p}$), to guarantee a minimal impact on the performance of the routers and a limited overall resource usage.

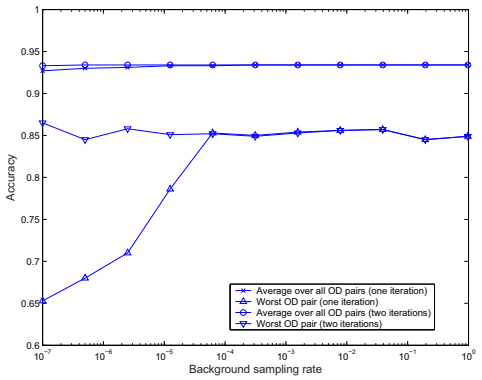

Figure 3: Performance of the bootstrapping algorithm as a function of the background sampling rate.

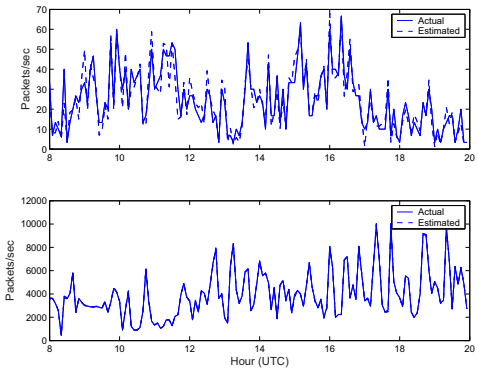

Figure 4: Estimated size of JANET to Luxembourg (top) and JANET to Switzerland (bottom) OD pairs. Sampling rates are fixed over the entire duration of the experiment

We can then use these estimates as input parameters to our method, derive a new set of sampling rates for all the routers (many of which will be set to zero), re-configure all routers, wait one measurement interval and get a new estimate of the input parameters. We iterate this procedure until the set of sampling rates converges and no router needs to change its settings.

This approach, however, raises three questions: (*i*) how do we estimate the size of the OD pairs? (*ii*) what is an appropriate background sampling rate? (*iii*) how many iterations are needed to perform the bootstrapping?

The optimization algorithm needs the values of $\mathbb{E}[1/S_k]$ for all OD pairs of interest. After the first measurement interval, we obtain X_k, the sampled size of each OD pair k. If $X_k > 0$, we replace $\mathbb{E}[1/S_k]$ in (10) with ρ_k/X_k – note that $p_i = \tilde{p}$, for all $i \in \mathcal{L}$. In the case $X_k = 0$, the OD pair has not been sampled at all. This means that the original OD pair size is likely smaller than $1/\tilde{p}$. For our purposes, here, any value below $1/\tilde{p}$ is a good enough estimate of its size. Hence, we replace $\mathbb{E}[1/S_k]$ with $c\tilde{p}$, where c is a constant arbitrarily set to 10 in all our experiments.

Note that finding an accurate estimator for $\mathbb{E}[1/S_k]$ is outside the scope of this paper. In fact, we deliberately choose to use one single sample of the size as the estimator because it is simple and *clearly imperfect*. Our aim is to show that our bootstrapping method is robust to incorrect input parameters and requires no configuration effort.

To answer the last two questions, we run an experiment over the flow data where we vary the value of $\tilde{p}$ from 10^{-7} to 1. We estimate the size of the OD pairs from JANET to all PoPs and the link loads across the entire network. Then, we run the optimization algorithm using this set of input parameters and a value of $\theta = 100,000$ (as in Section 6) to derive a set of sampling rates.

Figure 3 shows the average and worst accuracy over all OD pairs as a function of the background sampling rates. The figure compares the performance after one or two iterations. It is clear that two iterations are sufficient to complete the bootstrapping. Indeed, after two iterations the accuracy of the measurement does not depend anymore on the background sampling rate and thus on the quality of the input parameters. We also observe that a background sampling rates of 10^{-7} is enough if the algorithm is allowed at least two iterations. This demonstrates that the bootstrapping algorithm is very robust against incorrect input parameters.

7.2 Performance over time

We evaluate the performance of our method by estimating the size of all OD pairs originating from JANET over the entire period 8AM to 8PM. As in Section 6, we set a capacity $\theta = 100,000$. We perform the bootstrapping algorithm as described in the previous section at 7.55AM with a sampling rate of 10^{-7}. The sampling rates obtained from the optimization algorithm are then set on all the routers and unchanged for the entire 12 hour period.

Figure 4 compares the estimated and actual size for two OD pairs (JANET to Luxembourg and JANET to Switzerland). The figure shows the result for one run of the sampling process over the 12 hour period. The estimate is extremely accurate and closely follows all fluctuations over time for the JANET to Switzerland OD pair (the two lines overlap completely in the figure). The estimate is somewhat less accurate for JANET to Luxembourg (our worst case) but this is well expected given the small size of this OD pair (it peaks at 70 packets/sec).

In Figure 5 we plot the accuracy of the measurement over 20 experiments. Each experiment consists in one run of the sampling process across the entire 12 hour period. We show the average accuracy over all OD pairs and the accuracy for the worst OD pair over time. The average accuracy is always above 0.95 indicating that we are able to closely track the fluctuations in all OD pairs. Even, the worst OD pair is tracked with an accuracy around 0.90, but in certain measurement intervals, the accuracy drops below 0.85.

A second metric that plays a crucial role in our method is the evolution over time of the measured resource consumption defined as the total number of sampled (and thus processed) packets, $\hat{\theta}(t) = \sum_i p_i U_i(t)$. We are interested in comparing this value with θ^*, the overall capacity. Figure 6 shows the relative difference computed as $(\hat{\theta}(t) - \theta^*)/\theta^*$.

As we can see, the actual resource consumption is always above the constraint and it may exceed the constraint by as much as 120%. This is a consequence of having set the sampling rate early in the morning without taking into account time-of-day effects that lead to an increase in traffic in the later hours of the day. Therefore, the sampling rates are too high for the resource constraint, but result in good

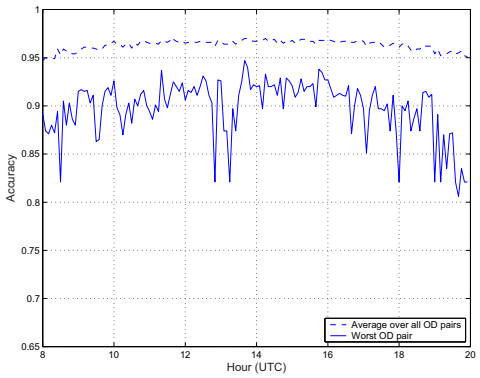

Figure 5: Accuracy over all OD pairs averaged over 20 experiments. Sampling rates are fixed over the entire duration of the experiments

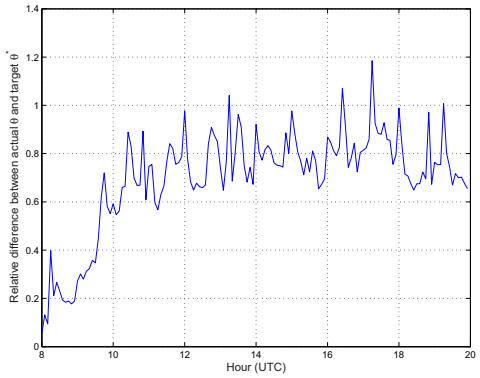

Figure 6: Relative difference between the actual resource consumption $\theta(t)$ and the target constraint θ^*. Sampling rates are fixed over the entire duration of the experiment

accuracy.

7.3 Adapting to traffic fluctuations

Ideally, the network operator would like to define a measurement task and a target accuracy (or a limit on the resource consumption) and then have the monitoring infrastructure that provides the best possible results independently of when the measurement is started or which network events occur during the measurement.

There are three specific events that make a given set of sampling rates suboptimal: (1) The monitored links start to carry a larger amount traffic that is not of interest for the measurement task. This leads to an increase in the resource consumption that does not correspond to an increase in the accuracy (as described in Section 7.2). (2) An OD pair of interest decrease in size requiring a higher sampling rate in order to keep the target accuracy. (3) The OD pairs of interest are routed differently in the network and some monitors are only capturing traffic that is not of interest.

In the following, we present solutions that allow us to cope with each one of these events.

Fluctuations in the link loads. The link loads vary over time for a variety of reasons, including time-of-day effects, anomalies, flash crowds, etc. At peak times the traffic on a link may be orders of magnitude larger than during quiet times. If the sampling rates are fixed and set during quiet times, this may lead to exceed the resource constraint as shown in Figure 6.

A possible solution is to define two thresholds θ_L and θ_H around the capacity θ and then predict the number of sampled packets in the next interval, $\bar{\theta}(t+1)$. When the prediction exceeds those thresholds, the collector can trigger a recomputation of the optimal sampling rates and (potentially) reconfigure the monitors.

At the end of each measurement interval, the collector can estimate $\bar{\theta}(t+1)$ from the number of sampled packets $\hat{\theta}(t)$ derived from the received flow records. Identifying an accurate predictor of the resource usage is outside the scope of this paper. Our goal is to provide an analysis of the performance of this method with the simplest predictor: we use $\hat{\theta}(t)$ as the predictor of the resource usage in the next interval, $\bar{\theta}(t+1)$.

Figures 7 and 8 show the performance of this method when $\theta_L = 0.9\theta$ and $\theta_H = 1.1\theta$. The results are averaged over 20 experiments. Each experiment is started at 8AM with the bootstrapping taking place at 7.55AM (UTC). As we can see this method succeeds in keeping the resource consumption within the desired bounds even if the bootstrapping is performed during quiet times (early morning). With this adaptive method, the network operator can start the measurement at any time and for any duration without incurring the risk of overloading the monitoring infrastructure.

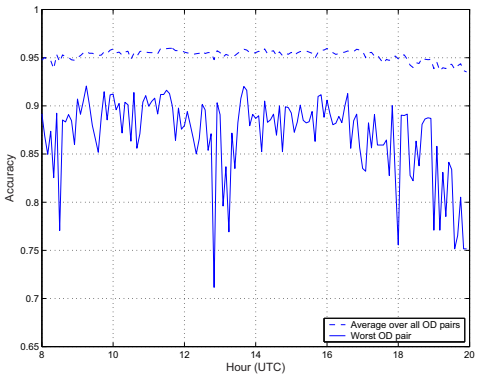

Figure 7: Average accuracy over all OD pairs and worst accuracy. Monitors are reconfigured as soon as the resource usage $\hat{\theta}(t)$ is 10% above or below the capacity θ.

OD pair fluctuations. The case of fluctuations in an OD pair traffic volume is the most difficult to address. Indeed, if the size of an OD pair decreases, in order to preserve the same measurement accuracy, there exists no alternative but to increase the sampling rates. If the OD pair accounts only for a small portion of the traffic on the monitored link, increasing the sampling rate causes a large amount of traffic not relevant for the measurement task to be collected.

We can see this phenomenon in Figure 7, where the accuracy of the worst OD pair (JANET to Luxembourg in our network scenario) drops significantly around 1PM and 7PM.

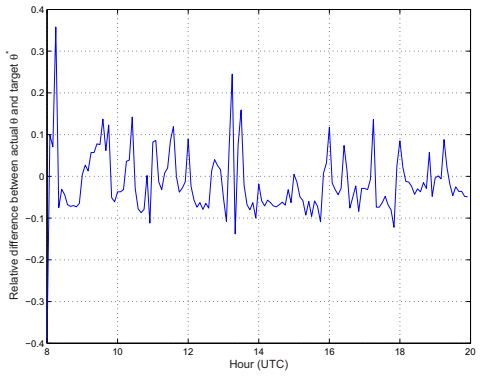

Figure 8: Relative difference between the resource usage $\hat{\theta}(t)$ and the capacity θ. Monitors are reconfigured as soon as $\hat{\theta}(t)$ is 10% above or below θ.

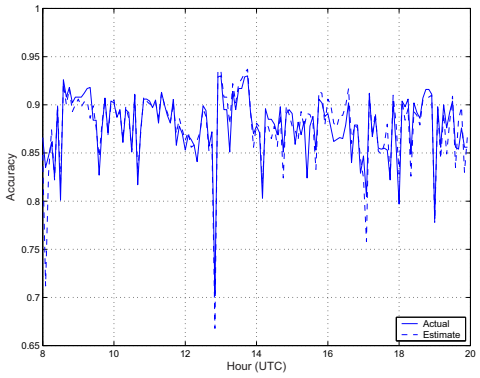

Figure 9: Comparison between estimated and actual accuracy for JANET to Luxembourg OD flow

These drops are consistent with the decrease in size of the JANET to Luxembourg OD pair (see Figure 4), but they do not correspond to equivalent decreases in the link loads.

Therefore, in the cases where the network operator is interested in preserving a target measurement accuracy, the capacity θ (i.e., the total number of sampled packets) needs to adapt to network conditions as well.

We now face a second challenge. Given a target accuracy set by the network operator, how can we estimate if, given the sampled size $X_k(t)$ of each OD pair k, the sampling rate is "good enough" to meet our target in the next intervals?

To this end, we approximate the accuracy as 1 minus the square root of (10). As we did in Section 7.1, we replace $\mathbb{E}[1/S_k]$ by $\rho_k/X_k(t)$, where ρ_k is the effective sampling rate for OD pair k.

Figure 9 compares the estimated accuracy with the actual accuracy from the data for the JANET to Luxembourg OD pair (the smallest OD pair). As we can see, our estimate closely follows the actual accuracy.

We can then use this approximation of the measurement accuracy to trigger a new reconfiguration of the traffic monitors as soon as the worst accuracy drops below the defined target.

We first run the optimization algorithm with the same value of θ used in the previous interval. We then compute

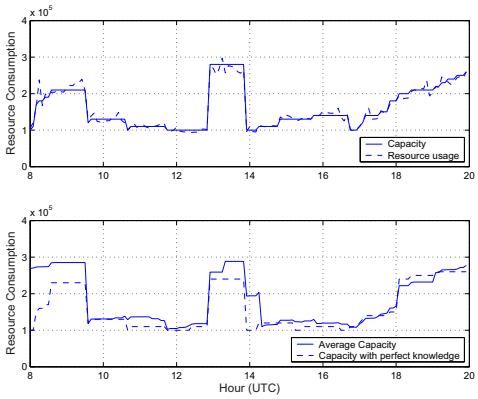

Figure 10: Adapting θ to preserve the target accuracy. Top: actual resource consumption over one experiment. Bottom: average θ over 20 experiments compared with the case of complete knowledge of the traffic.

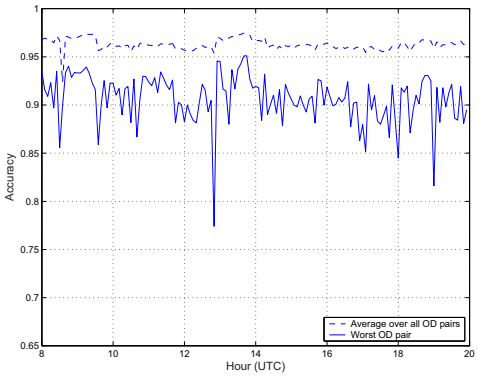

Figure 11: Average accuracy over all OD pairs when θ is adapted to meet target accuracy (0.85)

again the estimate accuracy using the same size estimate $X_k(t)$ but with the new sampling rates. If the estimated accuracy is still below the target, we increase θ by 10% until we reach the target accuracy for all OD pairs.

Furthermore, in order to avoid keeping a θ unnecessary large for long periods, we trigger a new reconfiguration if the estimated worst accuracy is above our target for one consecutive hour.

Figure 10 shows how the capacity θ varies over time when the target accuracy is set to 0.85. The top graph shows, for one experiment, the evolution of the actual resource usage and the capacity. The bottom graph shows the average over 20 experiments and compares it with the capacity that could be set knowing the exact size of each OD pair. Finally, Figure 11 illustrates the performance in terms of accuracy over all OD pairs and worst OD pair.

Around 1PM we observe a significant increase of the capacity θ. This corresponds to a significant drop in accuracy for the worst OD pair. Note that in the previous non-adaptive schemes, the poor accuracy would persist for more than one hour (see Figure 7). Increasing θ allows instead to quickly bring the accuracy above the target. Moreover, θ is decreased as soon as there is no more need of the additional

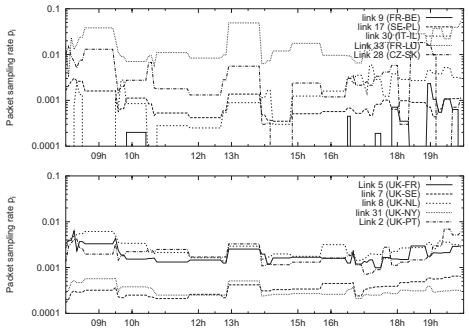

Figure 12: Evolution of the sampling rates over time. Top: lightly loaded links. Bottom: heavily loaded links.

resources.

A last aspect we want to investigate is the evolution of the sampling rates over time. We are interested in having sampling rates that are always low. Moreover, we want to verify that to track effectively OD pairs, a static placement of monitors is not sufficient.

Figure 12 shows the sampling rates for the 10 monitors that are activated during the day. The top graph shows the sampling rates for the lightly loaded links, while the bottom graph shows the sampling rates on the heavily load links.

The number of monitors used to track the 20 OD pairs of interest varies between eight and ten. Often the algorithm includes one or more of the lightly loaded links in order to improve the accuracy or reduce the resource consumption. The sampling rates exceed 1/100 only for the lightly loaded France to Luxembourg link that is instrumental to accurately track the fluctuations of the JANET to Luxembourg OD pair. The heavily loaded links (all links from the UK, a major PoP in GEANT's network) are all set to low sampling rates and experience much less variability over time.

Routing matrix changes. Network failures or BGP updates may modify the path taken by an OD pair under study across the network. When this occurs, some monitors may be recording traffic that is not of interest for the operator. Detecting routing changes is very simple given that routing messages need to propagate to all routers. ISPs often deploy systems to record and process all intra-domain and inter-domain routing messages [14, 25]. The monitoring infrastructure can therefore be aware of routing changes that affect the OD pairs of interest and trigger a new bootstrapping phase. The bootstrap would allow to find where the OD pairs are now entering the network (e.g., for multihomed customers) and get a new estimate of their size. Therefore, after two measurement intervals the monitoring infrastructure will have reconfigured itself with a new set of optimal monitors and sampling rates (see Section 7.1).

7.4 Implementation feasibility

We have shown that it is possible to design a self-configuring, self-adapting monitoring infrastructure in today's backbone networks. Our proposed method can be deployed today in any network running NetFlow (or any other widespread monitoring infrastructure). The network operator has just to select a measurement task (e.g., track any set of OD pairs) and a measurement strategy (i.e., focus on the resource usage or on the accuracy). The proposed mechanisms will then choose the monitors that allow to track those OD pairs with an optimal use of the resources and adapt to changes in the network conditions.

However, we did not look at the time it takes to reconfigure NetFlow on today's routers. All our results assume zero time to run the optimization algorithm (that is instead in the order of a few seconds), and to set the new sampling rates on the routers. Unfortunately, we do not have any figures about the time needed to remotely configure NetFlow, as we did not have any access to the routers in the network but only to the flow records stored at the collector. We leave this task as future work.

Our approach uses only the information stored (or that can be derived) from NetFlow records. This is one of the strengths of our method given that it is generic and can be applied to new NetFlow versions without modifications. Newer versions of NetFlow are constantly increasing the amount of information that is exported in each record. For example, NetFlow v9 records contain BGP next-hop information that would significantly simplify the task of finding the egress PoP for a given network prefix [22]. Moreover, router vendors are reportedly working on ways to support variable sampling rates as in [7]. In this case, our method can be used to define lower bounds on the sampling rates across the entire network, while the individual routers adapt to the local traffic conditions given their available resources.

The method gives the best performance when compared to alternative solutions (Section 6.3) if different network links have different load and carry a diverse set of OD pairs. Indeed, lightly utilized links are candidate for higher sampling rates that allow to compensate for the low sampling rates that we set on heavily loaded links. This is true for the GEANT network and other measurement studies have indicated that this is a common case for other ISP networks as well [2, 8].

Finally, there are a number of ways in which the adaptive methods described in this section can be extended. The choice to increase θ by steps of 10% and to decrease it after an hour is in fact somewhat arbitrary (see Section 7.3). It showed to perform well in practice but one can find smoother ways to adapt θ to the traffic conditions. It is also possible to use different predictors for the resource usage, as well as estimators for the OD pair size. All the results show the baseline performance when the prediction is done using just the previous sample. Even with this simple techniques, our method has demonstrated to work well and achieve the target accuracy. However, other predictors may perform better for different datasets.

8. CONCLUSION

We reformulated the monitor placement problem to adapt it to the reality of network operations and management in the Internet. We have proposed an optimization method to select and configure passive monitors in a backbone network. The method receives as input the network topology, the routing matrix and the set of OD pairs of interest. It returns a set of monitors (and their sampling rates) that is optimal with respect to the measurement task to perform. We have described the performance of our method considering a canonical measurement task, i.e., the estimation of the size of a set of OD pairs.

Although the evaluation in this paper is in terms of estimating OD pair sizes, the optimization method is not specific to them. The method can be applied to a wide range of measurement tasks for which a utility function can be sought. Our ongoing work is centered on defining new expressions for the utility function for applications such as anomaly detection and network performance analysis. We are also studying alternative formulations for the objective function as well as trying to evaluate the performance of our method in larger networks and different datasets.

ACKNOWLEDGMENTS

We wish to thank DANTE for granting us access to the NetFlow, BGP and ISIS dataset of the GEANT network to perform this study.

9. REFERENCES

[1] Y. Bejerano and R. Rastogi. Robust monitoring of link delays and faults in IP networks. In *Proceedings of IEEE Infocom*, Apr. 2003.

[2] S. Bhattacharyya, C. Diot, J. Jetcheva, and N. Taft. Geographical and temporal characteristics of inter-PoP flows: View from a single PoP. *European Transactions on Telecommunications*, 13(1):5–22, Feb. 2002.

[3] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[4] Cisco Systems. NetFlow services and applications. White Paper, 2000.

[5] N. G. Duffield, C. Lund, and M. Thorup. Properties and prediction of flow statistics from sampled packet streams. In *Proceedings of ACM Internet Measurement Workshop*, Nov. 2002.

[6] N. G. Duffield, C. Lund, and M. Thorup. Estimating flow distributions from sampled flow statistics. In *Proceedings of ACM Sigcomm*, Aug. 2003.

[7] C. Estan, K. Keys, D. Moore, and G. Varghese. Building a better NetFlow. In *Proceedings of ACM Sigcomm*, Aug. 2004.

[8] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True. Deriving traffic demands for operational ip networks: Methodology and experience. In *Proceedings of ACM Sigcomm*, Sept. 2000.

[9] M. Fullmer and S. Romig. The OSU flow-tools package and Cisco NetFlow logs. In *Proceedings of Usenix LISA*, Dec. 2000.

[10] GEANT. http://www.dante.net.

[11] R. T. Haftka and Z. Guerdal. *Elements of Structural Optimization*. Kluwer Academic Publishers, third edition.

[12] N. Hohn and D. Veitch. Inverting sampled traffic. In *Proceedings of ACM Internet Measurement Conference*, Oct. 2003.

[13] J. Horton and A. Lopez-Ortiz. On the number of distributed measurement points for network tomography. In *Proceedings of ACM Internet Measurement Conference*, Nov. 2003.

[14] G. Iannaccone, C.-N. Chuah, S. Bhattacharyya, and C. Diot. Feasibility of IP restoration in a tier-1 backbone. *IEEE Network*, 18(2):13–19, Mar. 2004.

[15] IP Flow Information eXport Working Group. Internet Engineering Task Force. http://www.ietf.org/html.charters/ipfix-charter.html.

[16] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley. Inferring TCP connection characteristics through passive measurements. In *Proceedings of IEEE Infocom*, Mar. 2004.

[17] S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang. On the placement of internet instrumentation. In *Proceedings of IEEE Infocom*, Apr. 2000.

[18] A. K. John Hertz and R. G. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley.

[19] Juniper Traffic Sampling. http://www.juniper.net. Netflow-compatible implementation.

[20] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot. Traffic matrix estimation: Existing techniques and new directions. In *Proceedings of ACM Sigcomm*, Pittsburgh, Aug. 2002.

[21] H. Nguyen and P. Thiran. Active measurement for multiple link failures diagnosis in IP networks. In *Proceedings of PAM Workshop*, Apr. 2004.

[22] K. Papagiannaki, N. Taft, and A. Lakhina. A distributed approach to measure IP traffic matrices. In *Proceedings of ACM Internet Measurement Conference*, Oct. 2004.

[23] P. Phaal, S. Panchen, and N. McKee. InMon corporation's sFlow: A method for monitoring traffic in switched and routed networks. RFC 3176, Sept. 2001.

[24] D. Plonka. Flowscan: A network traffic flow reporting and visualization tool. In *Proceedings of Usenix LISA*, Dec. 2000.

[25] A. Shaikh, M. Goyal, A. Greenberg, R. Rajan, and K. Ramakrishnan. An OSPF topology server: Design and evaluation. *IEEE Journal on Selected Areas in Communications*, 20(4), May 2002.

[26] A. Soule, A. Lakhina, N. Taft, K. Papagiannaki, K. Salamatian, A. Nucci, M. Crovella, and C. Diot. Traffic matrices: Balancing measurements, inference and modeling. In *Proceedings of ACM Sigmetrics*, June 2005.

[27] A. Soule, A. Nucci, E. Leonardi, R. Cruz, and N. Taft. How to identify and estimate top largest OD flows in a dynamic environment. In *Proceedings of ACM Sigmetrics*, June 2004.

[28] K. Suh, Y. Guo, J. Kurose, and D. Towsley. Locating network monitors: Complexity, heuristics and coverage. In *Proceedings of IEEE Infocom*, Mar. 2005.

[29] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg. Fast accurate computation of large-scale IP traffic matrices from link loads. In *Proceedings of ACM Sigmetrics*, June 2003.

Annexe B

Ranking Flows from Sampled Traffic

Chadi Barakat, Gianluca Iannaccone, Christophe Diot

Appeared in proceedings of CoNext, Toulouse, October 2005.

Ranking Flows from Sampled Traffic

Chadi Barakat*

INRIA - Planète group
Sophia Antipolis, France
Chadi.Barakat@inria.fr

Gianluca Iannaccone, Christophe Diot

Intel Research
Cambridge, UK
Gianluca.Iannaccone@intel.com,
Christophe.Diot@intel.com

ABSTRACT

Most of the theoretical work on sampling has addressed the inversion of general traffic properties such as flow size distribution, average flow size, or total number of flows. In this paper, we make a step towards understanding the impact of packet sampling on individual flow properties. We study how to detect and rank the largest flows on a link. To this end, we develop an analytical model that we validate on real traces from two networks. First we study a *blind* ranking method where only the number of sampled packets from each flow is known. Then, we propose a new method, *protocol-aware* ranking, where we make use of the packet sequence number (when available in transport header) to infer the number of non-sampled packets from a flow, and hence to improve the ranking. Surprisingly, our analytical and experimental results indicate that a high sampling rate (10% and even more depending on the number of top flows to be ranked) is required for a correct blind ranking of the largest flows. The sampling rate can be reduced by an order of magnitude if one just aims at detecting these flows or by using the protocol-aware method.

Categories and Subject Descriptors

C.4 [**Computer Systems Organization**]: PERFORMANCE OF SYSTEMS—*Measurement techniques*

General Terms

Measurement, Performance, Experimentation

Keywords

Packet sampling, largest flow detection and ranking, performance evaluation, validation with real traces

*This work was done while the author was visiting Intel Research Cambridge.

CoNEXT'05, October 24–27, 2005, Toulouse, France.

1. INTRODUCTION

The list of the top users or applications is one of the most useful statistics to be extracted from network traffic.

Network operators use the knowledge of the most popular destinations to identify emerging markets and applications or to locate where to setup new Points of Presence. Content delivery networks use the popularity of sites to define caching and replication strategies. In traffic engineering, the identification of heavy hitters in the network can be used to treat and route them differently across the network [20, 17, 10]. Keeping track of the network prefixes that generate most traffic is also of great importance for anomaly detection. A variation in the pattern of the most common applications may be used as a warning sign and trigger careful inspection of the packet streams.

However, the ability to identify the top users in a packet stream is limited by the network monitoring technology. Capturing and processing all packets on high speed links still remains a challenge for today's network equipment [16, 9]. In this context, a common solution is to sample the packet stream to reduce the load on the monitoring system and to simplify the task of sorting the list of items. The underlying assumption in this approach is that the sampling process does not alter the properties of the data distribution.

Sampled traffic data is then used to infer properties of the original data (this operation is called *inversion*). The inversion of sampled traffic is, however, an error-prone procedure that often requires a deep study of the data distribution to evaluate how the sampling rate impacts the accuracy of the metric of interest. Although the inversion may be simple for aggregate link statistics (e.g., to estimate the number of packets transmitted on a link, it is usually sufficient to multiply the number of sampled packets by the inverse of the sampling rate), it is much harder for the properties of individual connections or "flows" [9, 11, 8].

For these reasons, in this paper, we address this simple, and so far unanswered, question: *which sampling rate is needed to correctly detect and rank the flows that carry the most packets?*

We define the problem as follows. Consider a traffic monitor that samples packets independently of each other with probability p (random sampling) and classifies them into *sampled flows*. At the end of the measurement period, the monitor processes the list of sampled flows, ranks them based on their size in packets, and returns an ordered list of the t largest flows.

We are interested in knowing (i) whether the ordered list contains all the actual largest flows in the original packet

stream (*detection*), and (*ii*) if the items in the list appear in the correct order (*ranking*).

We build an analytical model and define a performance metric that evaluates the accuracy of identification and ranking of the largest flows. We consider a flow to consist of a single TCP connection. However, our results are general and can be applied to alternative definitions of flow, as well.

We evaluate two approaches to sort the list of flows:

(*i*) *Blind*, where the sampled flows are ranked just based on their sampled size. This method can be applied to any definition of flow.

(*ii*) *Protocol-aware*, where we make use of additional information in the packet header (e.g., the sequence number in TCP packets) to infer the number of non-sampled packets between sampled ones. This method can only be applied to flow definitions that preserve the protocol level details.

The contributions of this work are the following: (1) We perform an analytical study of the problem of ranking two sampled flows and compute the probability that they are *misranked*. We propose a Gaussian approximation to make the problem numerically tractable. (2) We introduce the protocol-aware ranking method that uses protocol level information to complement the flow statistics and render the detection and ranking of the largest flows more accurate. (3) Based on the model for the ranking of two flows, we propose a general model to study the detection and ranking problem, given a generic flow size distribution. We define a performance metric and evaluate the impact of several metric's parameter on the accuracy of the ranking. (4) We validate our findings on measurement data using publicly-available packet-level traces. Our results indicate that a surprisingly high sampling rate is required to obtain a good accuracy with the blind approach (10% and even more depending on the number of flows of interest). As for the protocol-aware approach, it allows to reduce the required sampling rate by an order of magnitude compared to the blind approach.

The paper is structured as follows. Next, we discuss the related literature. In Section 3 and 4, we present our model. Section 5 analyzes the model numerically and Section 6 validates it on real packet-level traces. Section 7 concludes the paper and provides perspectives for our future research.

2. RELATED WORK

The inversion of sampled traffic has been extensively studied in the literature. The main focus has been on the inversion of aggregate flow properties such as flow size distribution [9, 11], average flow size or total number of flows [8] on a given network link. Duffield et al. [8] study the problem of flow splitting and propose estimators for the total number of flows and for the average flow size in the original traffic stream. [9, 11] study the inversion of the flow size distribution with two different methods. They both show that the major difficulty comes from the number of flows that are not sampled at all and that need to be estimated with an auxiliary method. As an auxiliary method, [8, 9] propose the use of the SYN flag in the TCP header to mark the beginning of a flow. [9] shows that periodic and random sampling provide roughly the same result on high speed links, and so random sampling can be used for mathematical analysis due to its appealing features. [4] finds the sampling rate that assures a bounded error on the estimation of the size of flows contributing to more than some predefined percentage of the traffic volume. [14] studies whether the number of sampled packets is a good estimator for the detection of large flows without considering its impact on the flow ranking.

Given the potential applications of finding the list of top users, it does not come as a surprise that there has been a significant effort in the research community to find ways to track frequent items in a data stream [5, 7, 3, 10]. However, this problem has usually been addressed from a memory requirement standpoint. All the works in the literature assume that if the algorithm and the memory size is well chosen, the largest flows can be detected and ranked with a high precision. However, in the presence of packet sampling, even if the methods rank correctly the set of sampled flows, there is no guarantee that the sampled rank corresponds to the original rank. The problem we address in this paper complements these works as it focuses on the impact of sampling on the flow ranking.

3. BASIC MODEL: RANKING TWO FLOWS

In this section, we study the probability to misrank two flows of original sizes S_1 and S_2 in packets. This probability is the basis for the general model for detecting and ranking the largest flows that we will present later. Indeed, the detection and ranking of the largest flows can be transformed into a problem of ranking over a set of flow pairs.

Without loss of generality, we assume $S_1 < S_2$. We consider a random sampling of rate p. Let s_1 and s_2 denote the sizes in packets of both flows after sampling. The two sampled flows are misranked if (*i*) s_1 is larger than s_2, or (*ii*) both flows are not sampled, i.e., their sampled sizes equal to zero. By combining (*i*) and (*ii*), one can see that the necessary condition for a good ranking is to sample at least one packet from the larger flow (i.e., the smaller of the two flows can disappear after sampling). The probability to misrank the two flows can then be written as $P_m(S_1, S_2) = \mathbb{P}\{s_1 \geq s_2\}$. For the case $S_1 = S_2$, we consider the two flows as misranked if $s_1 \neq s_2$, or if both flows are not sampled at all, i.e. $s_1 = s_2 = 0$.

We compute and study the misranking probability of two flows of given sizes in the rest of this section. First, we consider the blind ranking method where only the number of sampled packets from a flow is known. For this method, we express the misranking probability as a double sum of binomials, then we present a Gaussian approximation to make the problem tractable numerically. Second, we consider the protocol-aware ranking method for which we calculate a numerical-tractable closed-form expression of the misraking probability. Note that the misranking probability is a symmetric function, i.e., $P_m(S_1, S_2) = P_m(S_2, S_1)$.

3.1 Blind ranking

With this method, s_1 and s_2 represent the number of sampled packets from flows S_1 and S_2. Under our assumptions, these two variables are distributed according to a binomial distribution of probability p. Hence, we can write for $S_1 < S_2$,

$$P_m(S_1, S_2) = \mathbb{P}\{s_1 \geq s_2\} = \sum_{i=0}^{S_1} b_p(i, S_1) \sum_{j=0}^{i} b_p(j, S_2). \quad (1)$$

$b_p(i, S)$ is the probability density function of a binomial distribution of probability p, i.e., the probability of obtaining i successes out of S trials. We have $b_p(i, S) = \binom{S}{i} p^i (1-p)^{S-i}$ for $i = 0, 1, ..., S$, and $b_p(i, S) = 0$ for $i < 0$ and $i > S$. The

probability to misrank two flows of equal sizes is given by $\mathbb{P}\{s_1 \neq s_2 \text{ or } s_1 = s_2 = 0\} = 1 - \mathbb{P}\{s_1 = s_2 \neq 0\}$
$= 1 - \sum_{i=1}^{S_1} b_p^2(i, S_1)$.

Unfortunately, the above expression for the misranking probability is numerically untractable since it involves two sums of binomials. For large flows of order S packets, the number of operations required to compute such a probability is on the order of $O(S^3)$, assuming that the complexity of the binomial computation is on the order of $O(S)$. The problem becomes much more complex if one has to sum over all possible flow sizes (i.e., $O(S^5)$). For this reason, we propose next a Gaussian approximation to the problem of blind ranking that is accurate and easy to compute. We use this approximation to study the ranking performance as a function of the sampling rate and the flow sizes.

3.1.1 Gaussian approximation to blind ranking

Consider a flow made of S packets and sampled at rate p. The sampled size follows a binomial distribution. However, it is well known that the binomial distribution can be approximated by a Normal (or Gaussian) distribution when p is small and when the product pS is on the order of one (flows for which, on average, at least few packets are sampled) [21, pages 108–109]. We assume that this is the case for the largest flows, and we consider the sampled size of a flow as distributed according to a Normal distribution of average pS and of variance $p(1-p)S$. Using this approximation, one can express the misranking probability for the blind ranking problem in the following simple form.

PROPOSITION 1. *For any two flows of sizes S_1 and S_2 packets ($S_1 \neq S_2$), the Gaussian approximation gives,*

$$P_m(S_1, S_2) \simeq \frac{1}{2} erfc\left(\frac{|S_2 - S_1|}{\sqrt{2(1/p-1)(S_1+S_2)}}\right), \qquad (2)$$

where $erfc(x) = (\frac{2}{\sqrt{\pi}}) \int_x^\infty e^{-u^2} \mathrm{d}u$ is the complementary error cumulative function.

Proof: Consider two flows of sizes S_1 and S_2 in packets such that $S_1 < S_2$. Their sampled versions s_1 and s_2 both follow Normal distributions of averages pS_1 and pS_2, and of variances $p(1-p)S_1$ and $p(1-p)S_2$. We know that the sum of two Normal variables is a Normal variable. So the difference $s_1 - s_2$ follows a Normal distribution of average $p(S_1 - S_2)$ and of variance $p(1-p)(S_1+S_2)$. We have then this approximation for the misranking probability:

$$\begin{aligned} P_m(S_1, S_2) &= \mathbb{P}\{s_1 - s_2 \geq 0\} \\ &\simeq \mathbb{P}\left\{V > \frac{p(S_2 - S_1)}{\sqrt{p(1-p)(S_1+S_2)}}\right\} \\ &= \frac{1}{2}\mathrm{erfc}\left(\frac{S_2 - S_1}{\sqrt{2(1/p-1)(S_1+S_2)}}\right). \quad (3) \end{aligned}$$

V is a standard Normal random variable. Given the symmetry of the misranking probability, one can take the absolute value of $S_2 - S_1$ in (3) and get the expression stated in the proposition, which is valid for all S_1 and S_2. □

For $S_1 = S_2$, one can safely approximate the misranking probability to be equal to 1. This approximation is however of little importance given the very low probability of having two flows of equal sizes, especially when they are large.

3.2 Protocol-aware ranking

Packets can carry in their transport header an increasing sequence number. A typical example is the byte sequence number in the TCP header. Another example could be the sequence number in the header of the Real Time Protocol (RTP) [19]. One can use this sequence number, when available, to infer the number of non-sampled packets (or bytes in the case of TCP) between sampled ones, and hence to improve the accuracy of ranking. The size of the sampled flow in this case is no longer the number of packets collected, but rather the number of packets that exist between the first and last sampled packets from the flow. Although this solution is limited to flows whose packet carry a sequence number, we believe that the study of this ranking method is important given the widespread use of the TCP protocol. Our objective is to understand how the use of protocol-level information can supplement the simple, and more general, blind method and if it is worth the additional overhead it introduces (i.e., storing two sequence numbers per flow record).

In the following, we calculate the misranking probability of two flows of given sizes when using the protocol-aware method. This probability will be used later in the general ranking problem. The main contribution of this section is a closed-form expression for the misranking probability that is numerically tractable, without the need for any approximation.

Let S be the size of a flow in packets. Let s_b, $s_b = 1, 2, ..., S$, denote the (packet) sequence number carried by the first sampled packet, and let s_e, $s_e = S, S-1, ..., s_b$, denote the sequence number carried by the last sampled packet. Given s_b and s_e, one can estimate the size of the sampled flow in packets to $s = s_e - s_b + 1$. The error in this estimation comes from the non-sampled packets that are transmitted before s_b and after s_e. We give next the distribution of s, which is needed for the computation of the misranking probability, then we state our main result. Before presenting the analysis, note that this new flow size estimator only counts the packets that are transmitted with distinct sequence numbers. In the case of TCP, this corresponds to the number of bytes received at the application layer, rather then the number of bytes carried over the network. It is equivalent to assuming that the probability of sampling a retransmitted (or duplicated) packet is negligible. This is a reasonable assumption if the loss rate is low. We will address this aspect in more detail in Section 6.

Consider a flow of size $S \geq 2$ in packets. Using the above definition for s, the sampled flow has a size of i packets, $i \geq 2$, with probability:

$$\mathbb{P}\{s = i\} = \sum_{k=1}^{S-i+1} \mathbb{P}\{s_b = k\}\, \mathbb{P}\{s_e = k + i - 1\}.$$

We have $\mathbb{P}\{s_b = k\} = (1-p)^{k-1}p$, and $\mathbb{P}\{s_e = k+i-1\} = (1-p)^{S-k-i+1}p$. This gives

$$\begin{aligned} \mathbb{P}\{s = i\} &= \sum_{k=1}^{S-i+1} (1-p)^{k-1} p (1-p)^{S-k-i+1} p \\ &= p^2 (1-p)^{S-i} (S - i + 1). \qquad (4) \end{aligned}$$

As for $i = 0$, we have $\mathbb{P}\{s = 0\} = (1-p)^S$ for $S \geq 1$. And for $i = 1$, we have $\mathbb{P}\{s = 1\} = p(1-p)^{S-1}S$ for $S \geq 1$. It is easy to prove that the cumulative distribution of s is the

following for all values of S:

$$\mathbb{P}\{s \leq i \neq 0\} = p(1-p)^{S-i}(S-i+1) + (1-p)^{S-i+1}. \quad (5)$$

We come now to the misranking probability, which we recall is a symmetric function. For $S_1 < S_2$, we have

$$P_m(S_1, S_2) = \mathbb{P}\{s_2 \leq s_1\} = \sum_{i=0}^{S_1} \mathbb{P}\{s_1 = i\} \sum_{j=0}^{i} \mathbb{P}\{s_2 = j\}. \quad (6)$$

And for $S_1 = S_2$, we have

$$P_m(S_1, S_2) = 1 - \sum_{i=1}^{S_1} \mathbb{P}\{s_1 = i\}^2. \quad (7)$$

Our main result is the following.

PROPOSITION 2. *For $S_1 < S_2$, the misranking probability is equal to*

$$\begin{aligned} P_m(S_1, S_2) &= (1-p)^{S_1}(1-p)^{S_2} \\ &+ p(1-p)^{S_1-1}S_1[p(1-p)^{S_2-1}S_2 + (1-p)^{S_2}] \\ &+ p^3 \frac{\partial^2 F(1-p, 1-p)}{\partial x \partial y} + p^2 \frac{\partial F(1-p, 1-p)}{\partial x}, \end{aligned}$$

where

$$\begin{aligned} F(x,y) &= xy^{S_2-S_1+1} + ... + x^{S_1-1}y^{S_2-1} \\ &= xy^{S_2-S_1+1}(1-(xy)^{S_1-1})/(1-xy). \end{aligned}$$

For $S_1 = S_2 = S$, the misranking probability is equal to

$$P_m(S,S) = 1 - p^2(1-p)^{2(S-1)}S^2 - p^4 \frac{\partial^2 G(1-p, 1-p)}{\partial x \partial y},$$

where

$$G(x,y) = xy + x^2y^2 + x^{S-1}y^{S-1} = (xy - (xy)^S)/(1-xy).$$

Proof: One can validate the results by plugging (4) and (5) into (6) and (7). □

Note that the main gain of writing the misraking probability in such a condensed form is a complexity that drops from $O(S^3)$ in (6) to $O(S)$ in our final result. This gain comes from the closed-form expression for the cumulative distribution in (5), and from introducing the two functions $F(x,y)$ and $G(x,y)$. These two latter functions transform two series whose complexity is $O(S^2)$ into a closed-form expression whose complexity is $O(S)$.

We solve the derivatives in the above equations using the symbolic toolbox of matlab, which gives explicit expressions for the misranking probability. These expressions are simple to compute, but span on multiple lines, so we omit them for lack of space.

3.3 Analysis of the misranking probability

3.3.1 The blind case

We use the Gaussian approximation to study how the misranking probability varies with the sampling rate and with the sizes of both flows, in particular their difference. The study of the impact of the flow sizes is important to understand the relation between flow size distribution and ranking of the largest flows.

The misranking probability is a decreasing function of the sampling rate. It moves to zero when p moves to 1 and to 0.5 when p approaches zero[1]. Therefore, there exists one sampling rate that leads to some desired misranking probability, and any lower sampling rate results in larger error.

We study now how the misranking probability varies with the sizes of both flows. Take $S_1 = S_2 - k$, k a positive integer. From (2) and for fixed k, the misranking probability increases with S_1 and S_2 ($\text{erfc}(x)$ is an increasing function in x). This indicates that it is more difficult to rank correctly two flows different by k packets as their sizes increase in absolute terms. The result is different if we take the size of one flow equal to $\alpha < 1$ times the size of the second, i.e., $S_1 = \alpha S_2$. Here, $(S_1 - S_2)/\sqrt{S_1 + S_2}$ is equal to $\sqrt{S_1}(1-\alpha)/\sqrt{1+\alpha}$, which increases with S_1. Hence, the misranking probability given in (2) decreases when S_1 increases. We conclude that, when the two flow sizes maintain the same proportion, it is easier to obtain a correct ranking when they are large in absolute terms.

We can now generalize the result above. One may think that the larger the flows, the better the ranking of their sampled versions. Our last two examples indicate that this is not always the case. The ranking accuracy depends on the relative difference of the flow sizes. In general, to have a better ranking, the difference between the two flow sizes must increase with the flow sizes and the increase must be larger than a certain threshold. This threshold is given by (2): the difference must increase at least as the square root of the flow sizes. This is an interesting finding. In the context of the general ranking problem, it can be interpreted as follows. Suppose that the flow size has a cumulative distribution function $y = F(x)$. As we move to the tail of the distribution[2], the size of the flows to be ranked increases. The ranking performance improves if the difference between flow sizes increases faster than $\sqrt{x}$. This is equivalent to saying that $\mathrm{d}x/\mathrm{d}y$ should increase with x faster than $\sqrt{x}$. All common distributions satisfy this condition, at least at their tails. For example, with the exponential distribution we have $\mathrm{d}x/\mathrm{d}y \propto e^{\lambda x}$ ($1/\lambda$ is the average), while for the Pareto distribution we have $\mathrm{d}x/\mathrm{d}y \propto x^{\beta+1}$ (β is the shape).

3.3.2 The protocol-aware case

The first difference with the blind case is in the estimation error ($S - s = s_b - 1 + S - s_e$), which can be safely assumed to be independent of the flow size for large flows (only dependent on p). This means that if two large flows keep the same distance between them while their sizes increase, their ranking maintains the same accuracy. Their ranking improves if the difference between their sizes increases as well, and it deteriorates if the difference between their sizes decreases. So in contrast to the blind case, the threshold for the ranking here to improve is that the larger flow should have its size increasing a little faster than the smaller one. In the context of the general ranking problem where flow sizes are distributed according to a cumulative distribution function $y = F(x)$, and when the top flows become larger, the protocol-aware ranking improves if the derivative $\mathrm{d}x/\mathrm{d}y$ increases with x. This is equivalent to saying that the function $F(x)$ should be concave, which is satisfied by most common distributions at their tail. For blind ranking, concavity was

[1]The Gaussian approximation does not account for the case $p = 0$ where the misranking probability should be equal to 1 based on our definition.

[2]Because we are more and more focusing on large flows or because the number of available flows for ranking increases.

not enough to obtain a better ranking; the derivative $\mathrm{d}x/\mathrm{d}y$ had to increase faster than $\sqrt{x}$. So in conclusion, the condition to have a better ranking when we move to the tail of the flow size distribution is less strict with the protocol-aware method, which is an indication of its good performance.

The second difference with the blind case is in the relation between the ranking accuracy and the sampling rate. Consider two large flows of sizes S_1 and S_2 in packets, and let s_1 and s_2 denote their sampled sizes. The coefficient of variation of the difference $s_2 - s_1$ is an indication on how well the ranking performs (a small coefficient of variation results in better ranking[3]). It is easy to prove that this coefficient of variation scales as $1/p$ for protocol-aware ranking and as $1/\sqrt{p}$ for blind ranking. This is again an important finding. It tells that when the sampling rate is very small, blind ranking could (asymptotically) perform better than protocol-aware ranking. Our numerical and experimental results will confirm this finding.

4. GENERAL MODEL: DETECTING AND RANKING THE LARGEST FLOWS

We generalize the previous model from the ranking of two flows to the detection and ranking of the top t flows, $t = 1, 2, \ldots, N$. The misranking probability $P_m(S_1, S_2)$ previously calculated is the basis for this generalization. Let $N \geq t$ denote the total number of flows available in the measurement period before sampling. We want the sampled list of top t flows to match the list of top t flows in the original traffic. Two criteria are considered to decide whether this match is accurate. First, we require the two lists to be identical. This corresponds to the *ranking* problem. The second, less constrained, criterion requires the two lists to contain the same flows regardless of their relative order within the list. This corresponds to the *detection* problem. For both problems, the quality of the result is expressed as a function of the sampling rate p, the flow size distribution, the number of flows to rank t, and the total number of flows N.

4.1 Performance metric

In order to evaluate the accuracy of detection and ranking, we need to define a performance metric that is easy to compute and that focuses on the largest flows. A flow at the top of the list can be misranked with a neighboring large flow or a distant small flow. We want our metric to differentiate between these two cases and to penalize more the latter one; a top-10 flow replaced by the 100-th flow in the sampled top list is worse than the top-10 flow being replaced by the 11-th flow. We also want our metric to be zero when the detection and ranking of the top flows are correct.

We introduce our performance metric using the ranking problem. The performance metric for the detection problem is a straightforward extension. Let's form all flow pairs where the first element of a pair is a flow in the top t and the second element is anywhere in the sorted list of the N original flows. The number of these pairs is equal to $N-1+N-2+\cdots+N-t = (2N-t-1)t/2$. We then count the pairs in this set that are misranked after sampling and we take the sum as our metric for ranking accuracy. This sum indicates how good the ranking is at the top of the list. It is equal to zero when the ranking is correct. When the ranking is not correct, it takes a value proportional to the original rank of the flows that have taken a slot in the top-t list. For example, if the top flow is replaced by its immediate successor in the list, the metric will return a ranking error of 1. Instead, if the same flow is replaced by a distant flow, say the 100-th, the metric will return an error of 99. Also, note that our metric does not account for any misranking of flows outside the list of top t flows. For any two flows n and m, such that $n > m > t$, the fact that n takes the position of m does not add anything to our performance metric since our metric requires at least one element of a flow pair to be in the original list of top t flows.

In the detection problem, we are no longer interested in comparing flow pairs whose both elements are in the top t list. We are only interested in the ranking between flows in the top t list and those *outside* the list. Therefore, our detection metric is defined as the number of misranked flow pairs, where the first element of a pair is in the list of top t flows and the second element is *outside* this list (non top t).

The above metrics return one value for each realization of flow sizes and of sampled packets. Given that we want to account for all realizations, we define the performance metrics as the number of misranked flow pairs *averaged* over all possible values of flow sizes in the original list of N flows and over all sampling runs. We deem the ranking/detection as acceptable when our metric takes a value below one (i.e., on average less than one flow pair is misranked).

In addition to the above, our metrics have the advantage of being easily and exactly calculable. Performance metrics based on probabilities (e.g.,[12]) require lot of assumptions that make them only suitable for computing bounds, but not exact values.

4.2 Computation of the performance metric for the ranking problem

Consider a flow of i packets belonging to the list of top t flows in the original traffic (before sampling). First, we compute the probability that this flow is misranked with another flow of general size and general position. Denote this probability by $P_{mt}(i)$, where m stands for misranking and t for top. Then, we average over all values of i to get $\bar{P}_{mt}$[4]. This latter function gives us the probability that, on average, the top t-th flow is misranked with another flow. Thus, our performance metric, which is defined as the average number of misranked flow pairs where at least one element of a pair is in the top t, is equal to $(2N-t-1)t\bar{P}_{mt}/2$. Next, we compute the value of $\bar{P}_{mt}$.

Let p_i denote the probability that the size of a general flow is equal to i packets, and P_i denote the flow size complementary cumulative distribution, i.e., $P_i = \sum_{j=i}^{\infty} p_j$. For a large number of flows N and a high degree of multiplexing, we consider safe to assume that flow sizes are independent of each other (see [2] for a study of the flow size correlation on a OC-12 IP backbone link). A flow of size i belongs to the list of top t flows if the number of flows in the original total list, with a size larger than i, is less or equal than $t-1$. Since each flow can be larger than i with probability P_i independently of the other flows, we can write the probability that a flow of size i belongs to the list of the top t flows

[3]For $S_1 < S_2$, we are interested in $\mathbb{P}\{s_1 \geq s_2\}$. According to Tchebychev inequality, this probability can be supposed to behave like $\mathrm{VAR}[s_1 - s_2]/\mathbb{E}[s_1 - s_2]^2$, which is the square of the coefficient of variation.

[4]Note that the distribution of the size of a flow at the top of the list is different from that of a generic flow.

as $P_t(i,t,N) = \sum_{k=0}^{t-1} b_{P_i}(k, N-1)$, where $b_{P_i}(k, N-1)$ is the probability to obtain k successes out of $N-1$ trials, P_i being the probability of a success. The probability that the t-th largest flow has a size of i packets is equal to $P_t(i) = p_i P_t(i,t,N)/\bar{P}_t(t,N)$. $\bar{P}_t(t,N)$ is the probability that a flow of general size is among the top t in the original total list, which is simply equal to t/N.

Using the above notation, one can write the misranking probability between a top t flow of original size i packets and any other flow as follows

$$P_{mt}(i) = \frac{1}{P_t(i,t,N)} \left(\sum_{j=1}^{i-1} p_j P_t(i,t,N-1) P_m(j,i) + \sum_{j=i}^{\infty} p_j P_t(i,t-1,N-1) P_m(i,j) \right). \quad (8)$$

In this expression, we sum over all possible original sizes of the other flow (the variable j) and we separate the case when this other flow is smaller than i from the case when it is larger than i [5]. $P_m(i,j)$ is the misranking probability of two flows of sizes i and j packets, which we calculated in the previous section for the two ranking methods. $\bar{P}_{mt}$ is then equal to $\sum_{i=1}^{\infty} P_t(i) P_{mt}(i)$.

For protocol-aware ranking, $P_m(i,j)$ is given explicitly in Proposition 2 and can be easily computed. For blind ranking, we use the Gaussian approximation summarized in Proposition 2, which we recall holds when at least one of the two flows to be compared is large.

4.3 Computation of the performance metric for the detection problem

Consider the probability that a flow among the top t is swapped with a flow that does belong to the top t. Let $\bar{P}^{\star}_{mt}$ denote this probability. Following the same approach described in Section 4, we can write

$$\bar{P}^{\star}_{mt} = \frac{1}{\bar{P}^{\star}_t} \sum_{i=1}^{\infty} \sum_{j=1}^{i-1} p_i p_j P^{\star}_t(j,i,t,N) P_m(j,i).$$

To get this expression for $\bar{P}^{\star}_{mt}$, we sum over all possible values for the size of the flow in the top t (index i) and all possible values for the size of the other flow not among the top t (index j). In this expression, p_i and p_j represent the probability that the size of a flow is equal to i or j packets, respectively. $P_m(j,i)$ is the probability that two flows of sizes i and j are misranked – it is given by the Gaussian approximation described in Proposition 1 for the blind method and the result stated in Proposition 2 for the protocol-aware method. $P^{\star}_t(j,i,t,N)$ is the joint probability that a flow of size i belongs to the list of the top t flows while another flow of size j does not belong to it (i.e., it is in the bottom $N-t$ flows). $\bar{P}^{\star}_t$ is the joint probability that a flow of any size belongs to the list of the top t flows while another flow of any size does not belong to this list. It is equal to $t(N-t)/(N(N-1))$.

We now compute $P^{\star}_t(j,i,t,N)$ for $j < i$, i.e., the probability that flow i belongs to the top list while flow j does not. The number of flows larger than i should be smaller than t, while the number of flows larger than j should be larger than t. The probability that a flow size is larger than i is $P_i = \sum_{k=i}^{\infty} p_k$. The probability that it is larger than j is $P_j = \sum_{k=j}^{\infty} p_k$. The probability that a flow size is between j and i given that it is smaller than i is $(P_j - P_i)/(1-P_i)$. We call it $P_{j,i}$. It follows that:

$$P^{\star}_t(j,i,t,N) = \sum_{k=0}^{t-1} b_{P_i}(k, N-2) \sum_{l=t-k-1}^{N-k-2} b_{P_{j,i}}(l, N-k-2).$$

The first sum accounts for the probability to see less than t flows above i packets. The second sum accounts for the probability to see more than t flows above j given that k flows ($k < t$) were already seen above i. For $t = 1$, $P^{\star}_t(j,i,t,N)$ is no other than $P_t(i,t,N-1)$, and both $\bar{P}^{\star}_{mt}$ and $\bar{P}_{mt}$ are equal (i.e., the ranking and the detection problems are the same).

Once $\bar{P}^{\star}_{mt}$ is computed, we multiply it by the total number of flow pairs whose one element is in the top t and the other one is not. This total number is equal to $t(N-t)$. Our metric for the detection problem is the result of this multiplication. As for the ranking problem, we want this metric to be less than one for the detection of the top t flows to be accurate.

Trace	Jussieu	Abilene
Link speed	GigE (1 Gbps)	OC-48 (2.5 Gbps)
Duration	2 hours	30 minutes
TCP connections	11M	15M
Packets	112M	125M

Table 1: Summary of the traces

5. NUMERICAL RESULTS

We analyze now the accuracy of identifying and ranking the largest flows in a packet stream for both the blind and protocol-aware methods. Our metrics require the following input: p_i, the flow size distribution and N, the total number of flows observed on the link during the measurement period.

To derive realistic values for these two quantities, we consider two publicly available packet-level traces. The first trace is Abilene-I collected by NLANR [15] on an OC-48 (2.5 Gbps) link on the Abilene Network [1]. The second trace has been collected by the Metropolis project [13] on a Gigabit Ethernet access link from the Jussieu University campus in Paris to the Renater Network [18]. Table 1 summarizes the characteristics of the two traces.

We model the flow size distribution in the traces with Pareto. We opted for Pareto since it is known to be appropriate to model flow sizes in the Internet due to its heavy tailed feature [6]. Note that it is not our goal to find an accurate approximation of the distribution of flow sizes in our traces, but rather to find a general, well-known, distribution that approaches the actual flow size. In this section we analyze a wide range of parameters while Section 6 focuses on the performance we observe in the two packet-level traces.

The Pareto distribution is continuous with a complementary cumulative distribution function given by $\mathbb{P}\{S > x\} = (x/a)^{-\beta}$. $\beta > 0$ is a parameter describing the shape of the distribution and $a > 0$ is a parameter describing its scale. The Pareto random variable takes values larger than a, and has an average value equal to $a\beta/(\beta-1)$. The tail of the Pareto distribution becomes heavier as β decreases.

We use our traces to derive an indicative value of the shape parameter β. To this end, we compute the empirical complementary cumulative distribution of flow sizes and we

[5] In the case $j \geq i$, at most $t-2$ flows can be larger than i packets if we want the flow of size i to be in the top t.

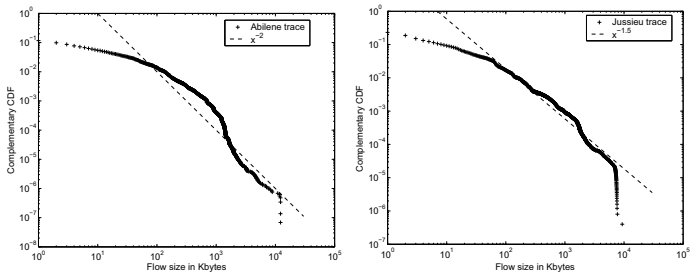

Figure 1: Empirical flow size distribution

plot it on a log-log scale. A heavy-tailed distribution of shape parameter β decays linearly on a log-log scale at rate $-\beta$. The empirical distributions are shown in Figure 1. The plots show that β equal to 2 suits the Abilene trace and β equal to 1.5 suits the Jussieu one. This means that the flow size distribution has a heavier tail in the Jussieu trace.

Then, we compute the average flow size in packets to get the starting point a for the Pareto distribution. As an average flow size we measure 5.76 Kbytes and 7.35 packets on the Abilene trace, and 9.22 Kbytes and 9.9 packets on the Jussieu trace. The total number of flows N is set by taking a measurement interval equal to one minute, then multiplying this interval by the average arrival rate of flows per second on each trace. This gives $N = 487$ Kflows for the Abilene trace and $N = 103$ Kflows for the Jussieu one.

In the rest of this section, all figures plot the ranking metric versus the packet sampling rate p on a log-log scale. We vary p from 0.1% to 50%. Each figure shows different lines that correspond to different combinations of t, β, and N. We are interested in the regions where the value of the metric is below one, indicating that the ranking is accurate on average. To ease the interpretation of results in the figures, we plot the horizontal line of ordinate 1.

5.1 Blind ranking

5.1.1 Impact of the number of flows of interest

The first parameter we study is t, the number of largest flows to rank. The purpose is to show how many flows can be detected and ranked correctly for a given sampling rate. We set β, N, and the average flow size to the values described before. The performance of blind ranking the top t flows is shown in Figure 2 for both traces. We observe that the larger the number of top flows of interest, the more difficult it is to detect and rank them correctly. In particular, with a sampling rate on the order of 1%, it is possible to rank at most the top one or two flows. As we focus at larger values of t, the required sampling rate to get a correct ranking increases well above 10%. Note that with a sampling rate on the order of 0.1%, it is almost impossible to detect even the largest flow. We also observe that the ranking on the Jussieu trace behaves slightly better than that on the Abilene trace. The Jussieu trace has a heavier tail for its flow size distribution, and so the probability to get larger flows at the top of the list is higher, which makes the ranking more accurate. This will be made clear next as we will study the impact of the shape parameter β.

5.1.2 Impact of the flow size distribution

We consider the blind ranking of the top 10 flows varying

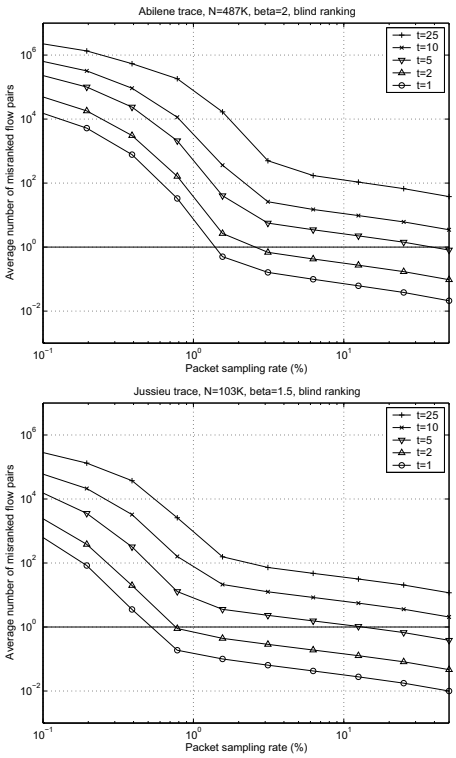

Figure 2: Performance of blind ranking varying the number t of top flows of interest

the shape parameter for the Pareto distribution among five distinct values: 3, 2.5, 2, 1.5 and 1.2. Note that for $\beta \leq 2$ the Pareto distribution is known to be heavy tailed (infinite variance). The other parameters of the model (N and the average flow size) are set as before. The values taken by our metric are shown in Figure 3 for both traces. We can make the following observations from the figure:

- Given a sampling rate, the ranking accuracy improves as β becomes smaller, i.e., the tail of the flow size distribution becomes heavier. Indeed, when the distribution tail becomes heavier, the probability to obtain larger flows at the top of the list increases, and since it is simpler to blindly rank larger flows (for distributions satisfying the square root condition, see Section 3.1.1), the ranking becomes more accurate.

- The ranking is never correct unless the sampling rate is very high. In our setting, one needs to sample at more than 50% to obtain an average number of misranked flow pairs below one for a value of β equal to 1.5 (i.e, heavy tailed distribution), and at more than 10% for a value of β equal to 1.2 (i.e., pronounced heavy tailed distribution). For larger values of β (i.e., lighter tail), the sampling rate needs to be as high as 100%.

5.1.3 Impact of the total number of flows

Another important parameter in the ranking problem is N, the total number of flows available during the measurement period. When N increases, the flows at the top of the list should become larger, and therefore as we saw in Section 3.1.1, the blind ranking accuracy should improve

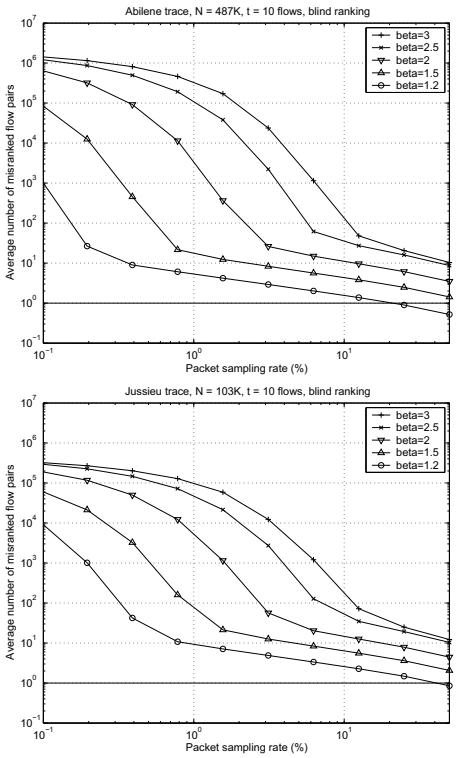

Figure 3: Performance of blind ranking varying the shape parameter of the flow size distribution

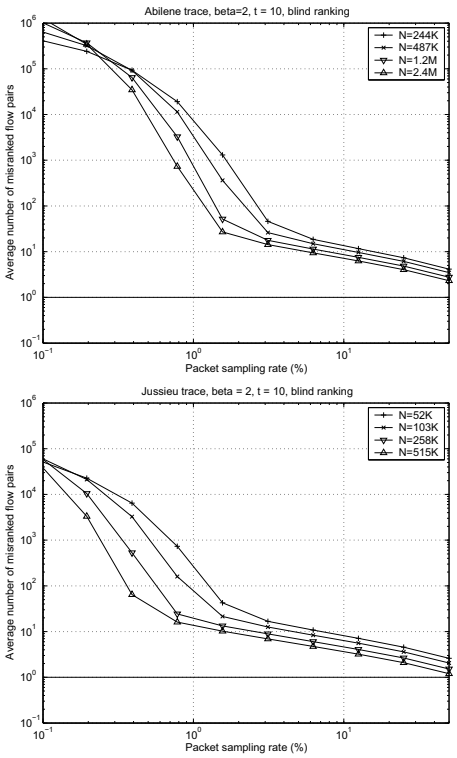

Figure 4: Performance of blind ranking varying the total number of flows

for flow size distributions satisfying the square root condition (in particular the Pareto distribution we are considering here). N varies with the utilization of the monitored link – the higher the utilization, the larger the number of flows. N can also vary with the duration of the measurement period – the longer we wait before ranking and reporting results, the larger the number of flows.

We study the impact of N on the blind ranking accuracy. We take the same value of N used in the previous sections and computed over one minute measurement period (487 Kflows for the Abilene trace and 103 Kflows for the Jussieu trace), then we multiply it by some constant factor ranging from 0.5 (2 times fewer flows) to 5 (5 times more flows). Results are shown in Figure 4. The lines in the figures correspond to a factor value equal to: 0.5, 1, 2.5, and 5. In these figures, we consider the ranking of the top 10 flows with the values of β and average flow size set from the traces. Clearly, the ranking accuracy improves as N increases. However, in our setting, this improvement is still not enough to allow a perfect ranking. One can always imagine increasing N (e.g., by increasing the measurement period) until the top t flows are extremely large and hence, perfectly detected and ranked.

5.2 Protocol-aware ranking

Protocol-aware ranking takes advantage of the information carried in the transport header of the sampled packets to infer the number of non-sampled packets of a flow. We use our model to check whether this improvement exists and to evaluate it. Remember that we are always in the context of low retransmission and duplication rates, which is necessary to remove the discrepancy between carried data volume (throughput) and application data volume (goodput).

Using the previous values for N, β and average flow size, we reproduce Figure 2, but this time for the protocol-aware case. This leads to Figure 5, which illustrates the impact of the number of largest flows to rank. For lack of space, we omit the other figures.

We compare this new figure to its counterpart in the blind case. We make the following two observations:

(i) The protocol-aware method improves the accuracy of the largest flows ranking by an order of magnitude for high sampling rates (above 1%). For example, for the Abilene trace, a sampling rate on the order of 50% was necessary to detect and rank the largest 5 flows with the blind method. Now, with the protocol-aware method, a sampling rate on the order of 5% is sufficient. The same conclusion applies to the Jussieu trace. A sampling rate on the order of 10% is needed. With the protocol-aware method, it becomes on the order of 1%.

(ii) The protocol-aware method does not improve the performance when applied at low sampling rates (above 1%). This can be clearly seen if we compare the plots between both figures for sampling rates below 1%. This results confirms our observations in Section 3.3.2.

5.3 Largest flows detection

To illustrate the difference between ranking and detection, we consider the same scenario as in Section 5.1.1. We plot the detection metric as a function of the sampling rate for different values of t (the number of top flows of interest) and for both Abilene and Jussieu traces. This gives Figure 6 for

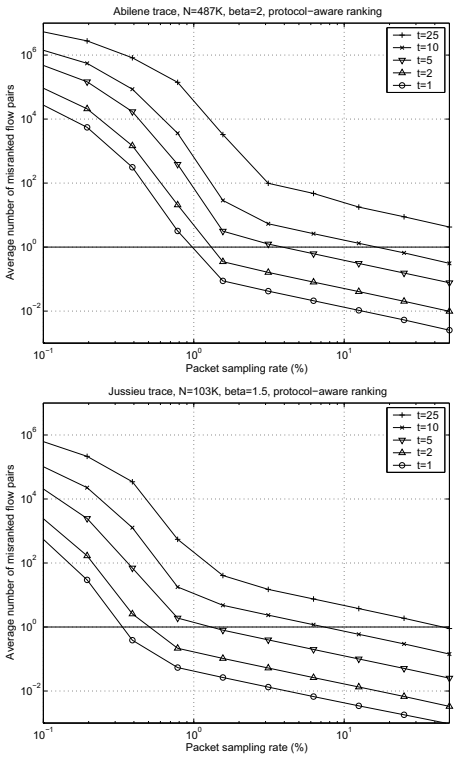

Figure 5: Performance of protocol-aware ranking varying the number t of top flows of interest

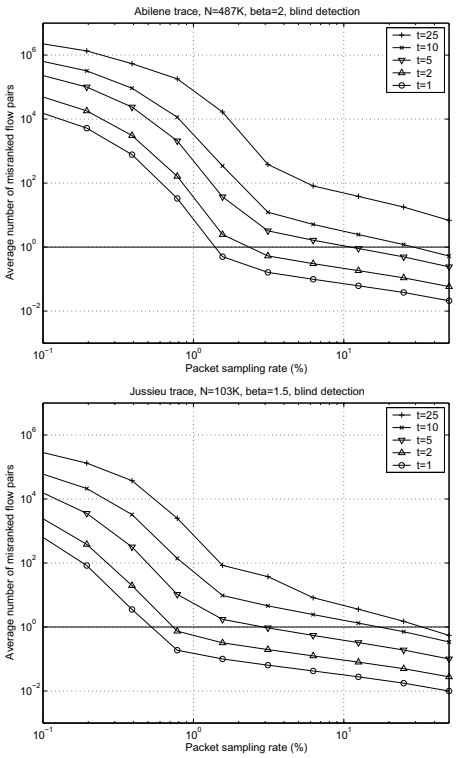

Figure 6: Only detecting the largest flows: Performance of blind ranking varying the number t of top flows of interest

blind ranking and Figure 7 for protocol-aware ranking. A comparison between these results and their counterparts in Figure 2 and 5, respectively, shows a significant improvement in the detection case for both ranking methods. All plots are shifted down by an order of magnitude. For example, in the case of blind ranking, the required sampling rate to correctly rank the top 5 flows was around 50% for the Abilene trace and 10% for the Jussieu trace. Now, with blind detection, it is around 10% and 3%, respectively. Another example is with the protocol-aware method where a sampling rate around 10% was required to rank the largest 10 flows (Figure 5), whereas now, a sampling rate around 1% is sufficient to only detect them. The same gain can be observed if we reconsider the other scenarios in Section 5.1 (not presented here for lack of space). Also, note how in the detection case the protocol aware method allows a better accuracy for high sampling rates when compared to the blind method. For low sampling rates (e.g., below 1%), the accuracy does not improve.

6. EXPERIMENTAL RESULTS

In this section we present the results of running random sampling experiments directly on the packet traces. We use the traces described in Section 5 and compute the performance metrics defined in Section 4.1.

In our traces we consider only TCP packets. Since TCP sequence numbers count bytes, we express the flow sizes in bytes instead of packets throughout this section.

Our experiments are meant to address four major issues that arise when we move from the analytical study to a real network setting: (i) how to deal with invalid TCP sequence numbers in the packet stream; (ii) the importance of flow size distributions and duration of the measurement interval; (iii) the impact of packet loss rates on individual flows – lost packets trigger retransmissions by the TCP senders; (iv) the variability of the detection/ranking performance across multiple bins and packet sampling patterns.

6.1 Implementation of protocol-aware ranking

The protocol-aware method depends on TCP sequence numbers to perform the ranking. For a given flow, it keeps track of the lowest and highest sequence number observed (taking care of packets that wrap around the sequence number space), s_b and s_e respectively.

Note that an actual implementation of this method would just require two 32 bit fields per flow to store the two sequence numbers.

At the end of the measurement period, we compute the difference between the highest and lowest sequence numbers for each sampled flow, and we use the obtained values to rank flows. We then compare this ranking with the one obtained by counting all the bytes each flow transmits in the original non sampled traffic.

In order to discard invalid packets carrying incorrect sequence numbers that would corrupt the ranking, we implement a simple heuristic to update s_e and s_b. A sampled packet with sequence number $S > s_e$ causes an update $s_e \leftarrow S$ if $S - s_e < (\alpha * \text{MTU})/p$. The same rule applies to the updates of s_b. This way we set a limit on the maximum distance in the sequence space between two sampled pack-

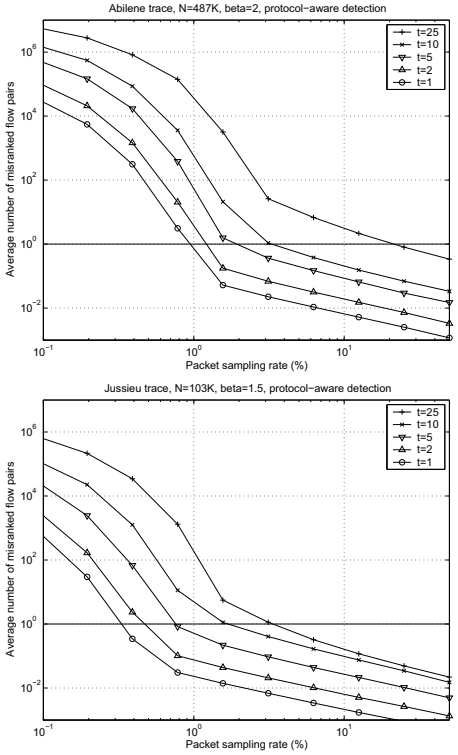

Figure 7: Only detecting the largest flows: Performance of protocol-aware ranking varying the number t of top flows of interest

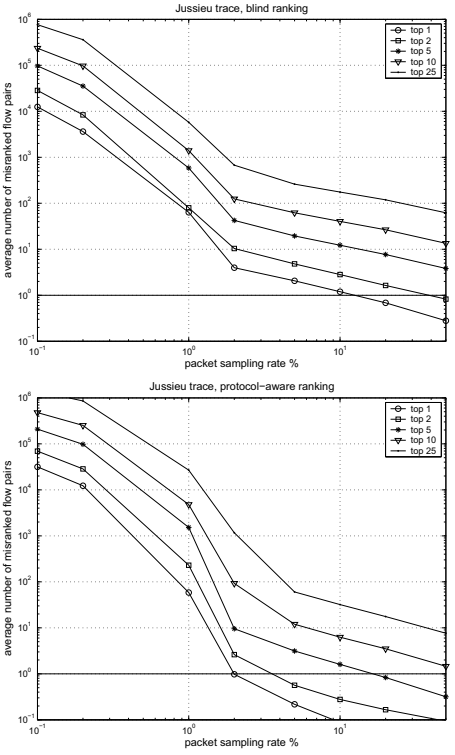

Figure 8: Performance of blind and protocol-aware ranking on Jussieu trace (60s measurement interval).

ets. This distance is inversely proportional to the sampling rate and depends on the Maximum Transmission Unit.

Furthermore, we use the parameter α that allows to make this threshold more or less "permissive" in order to account for the randomness of the sampling process and for other transport-layer events (e.g., packet retransmissions when the TCP window is large). We have run several experiments with different values of α and the results have shown little sensitivity to values of $\alpha > 10$. All the results in this Section are derived with $\alpha = 100$.

6.2 Flow size distribution and measurement interval

As shown in Figure 1, flow size distributions do not follow a perfect Pareto. Furthermore, the measurement interval itself plays a major role in shaping the distribution: it caps the size of the largest flows, that is not unbounded but now depends on the link speed. Indeed, network operators often run measurements using a "binning" method, where packets are sampled for a time interval, classified into flows, ranked, and then reported. At the end of the interval, the memory is cleared and the operation is repeated for the next measurement interval. With this binning method, all flows active at the end of the measurement interval are truncated, so that not all sampled packets of the truncated flow are considered at the same time for the ranking. The truncation may, therefore, penalize large flows and alter the tail of the flow size distribution (where flows are of large size and probably last longer than the measurement interval).

Each experiment consists of the following. We run random sampling on the packet traces and classify the sampled packets into flows. At the end of each measurement interval (set to 1 or 5 minutes), we collect the flows and rank them by the number of bytes sampled for each flow. We compare the ranking before and after sampling using our performance metric (Section 4.1). For each sampling rate we conduct 15 runs and we calculate averages.

The results of the experiments confirm the numerical results of the previous section. In the interest of space, we plot the results of two representative experiments on which we make several observations. The difference between numerical and experimental results, especially at low sampling rates, is caused by the non perfect match of the empirical flow size distribution with Pareto (Figure 1).

Figure 8 shows the performance of ranking flows on the Jussieu trace when the measurement bin is 60s. We consider a wide range of sampling rates from 0.1% to 50% and study the performance when ranking the top 1, 2, 5, 10 and 25 flows in the packet stream. The top graph in Figure 8 is derived using the blind method while the bottom graph shows the performance of the protocol-aware methods. These results are very similar to the numerical results. For sampling rates above 1%, protocol-aware ranking gives approximately an order of magnitude gain on the performance when compared to blind ranking. When the sampling rate is lower than 1%, however, the performance of the two methods is similar. Overall, the blind method requires a sampling rate of 10% to correctly identify the largest flow in the packet stream. The same sampling rate allows to correctly rank the largest 5 flows when using the protocol-aware method.

6.3 Impact of loss rate

In the analysis of the protocol-aware method in Section 3.2, we made the assumption of negligible number of retransmissions for all the flows in the packet stream.

A retransmitted packet may cause inconsistency between the blind and protocol-aware method depending on the location of the monitoring point. Indeed, the blind method counts the total number of bytes sent by the flow while the protocol-aware method considers only the data sent by the transport layer. Therefore, if the packet is lost before the monitoring point, the blind and protocol-aware method will have a consistent view of the number of bytes sent. Instead, if the packet is lost after the monitoring point, the blind method may count this packet twice.

The impact of packet losses on the detection and ranking of the largest flows depends on the metric used to estimate the size of the flows. If flow sizes are estimated according to the total number of bytes sent (i.e., the throughput), then the protocol-aware method may incur in an underestimation error that is independent of the sampling rate (it will occur even if all packets are sampled!). On the other hand, if the flow sizes are estimated according to the transport data sent (i.e., the goodput), then the blind method may incur in an overestimation error independently of the sampling rate.

To illustrate the effect of packet loss rates, we plot in Figure 9 the performance of detecting the largest flows in the Abilene trace when the measurement bin is 5 minutes and the flow sizes are measured using the total number of bytes sent over the link. The top graph shows the performance of the blind method, while the bottom graph presents the results for the protocol-aware method.

We can make the following observations:

- The protocol-aware method keeps performing better than the blind method when the sampling rate is above 1%. At lower sampling rates, the blind method performs better although it presents very large errors.
- For sampling rates above 2%, the curve relative to the detection of the top-25 flows in the protocol-aware method flattens to a value around 70. This is due to the presence of a few flows that experience a high loss rate when compared to other flows. Increasing the sampling rate does not help the protocol-aware method in detecting the largest flows when the volume of bytes sent is used to define the flow size. However, the protocol-aware method can correctly detect the top-25 flows when their size is defined in terms of transport data (see Figure 10).

In summary, the network operator has to choose the metric of interest that depends on the application. For example, for anomaly detection or traffic engineering, a metric that counts the number of bytes sent may be more appropriate. Instead, for dimensioning caches and proxies, the metric that considers the size of the objects transferred may be preferred. This latter metric suits more the protocol-aware method.

6.4 Variability of the results

A last important aspect that we need to address is the variability of the results across multiple measurement intervals and different realizations of the sampling process. Indeed, moving from one measurement interval to another,

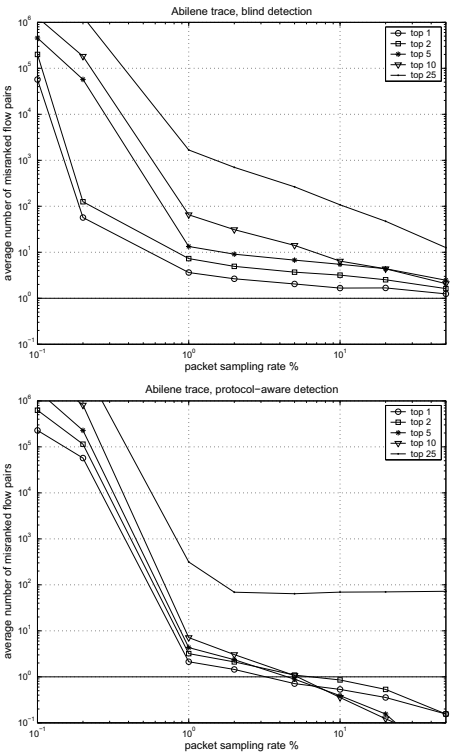

Figure 9: Performance of blind (top) and protocol-aware (bottom) detection on Abilene trace (300s measurement interval).

the composition of flows varies and with it the flow size distribution. Moreover, the sampling process may "get lucky" in certain cases and provide good results. The opposite is also possible.

Figure 11 shows the average performance over 15 sampling experiments of the detection of the top-10 flows in the Abilene trace over the 5-minute measurement intervals. The error bars indicate the standard deviation across the 15 experiments. As usual, the top graph refers to the blind method, while the bottom graph presents the protocol-aware method results.

As we can see the average performance shows limited variability. A sampling rate of 0.1% gives poor results for all bins, while increasing the sampling rates consistently helps. With a sampling rate of 10% the performance metric (i.e., average number of misranked flow pairs) for the blind method is always below 100 while the protocol-aware method is always below 1.

Looking at the standard deviation, we observe large values for the blind method and much smaller values for the protocol-aware method. This indicates that the blind method is more sensitive to the sampling process than the protocol-aware method. The explanation is given in Section 3.3.2 where we showed that that the blind method presents a larger error for large flow sizes (expect when the sampling rate is very low).

7. CONCLUSIONS

We study the problem of detection and ranking the largest flows from a traffic sampled at the packet level. The study is

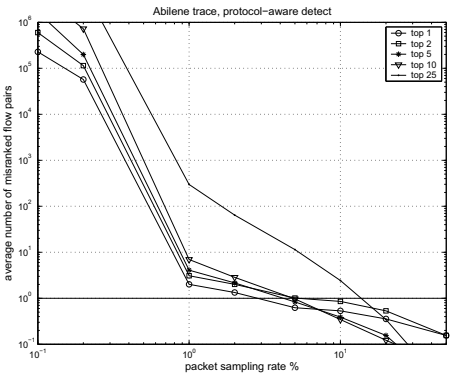

Figure 10: Performance of protocol-aware detection on Abilene trace (300s measurement interval) when using actual amount of data sent by the transport layer application.

done with stochastic tools and real packet-level traces. We find that the ranking accuracy is strongly dependent on the sampling rate, the flow size distribution, the total number of flows and the number of largest flows to be detected and ranked. By changing all these parameters, we conclude that ranking the largest flows requires a high sampling rate (10% and even more). One can reduce the required sampling rate by only detecting the largest flows without considering their relative order.

We also introduce a new method for flow ranking that exploits the information carried in transport header. By analysis and experimentation, we demonstrate that this new technique allows to reduce the required sampling rate by an order of magnitude.

We are currently exploring two possible future directions for this work. First, we want to study the accuracy of the ranking when the sampled traffic is fed into one of the mechanisms proposed in [10, 12] for sorting flows with reduced memory requirements. Second, we are exploring the use of adaptive schemes that set the sampling rate based on the characteristics of the observed traffic.

Acknowledgements

We wish to thank NLANR [15], Abilene/Internet2 [1] and the Metropolis project [13] for making available the packet traces used in this work.

8. REFERENCES

[1] Abilene: Advanced networking for leading-edge research and education. http://abilene.internet2.edu.

[2] C. Barakat, P. Thiran, G. Iannaccone, C. Diot, and P. Owezarski. Modeling Internet backbone traffic at the flow level. *IEEE Transactions on Signal Processing (Special Issue on Signal Processing in Networking)*, 51(8):2111–2124, Aug. 2003.

[3] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. In *Proceedings of ICALP*, 2002.

[4] B. Y. Choi, J. Park, and Z. Zhang. Adaptive packet sampling for flow volume measurement. Technical Report TR-02-040, University of Minnesota, 2002.

[5] G. Cormode and S. Muthukrishnan. What's hot and what's not: Tracking most frequent items dynamically. In *Proceedings of ACM PODS*, June 2003.

[6] M. Crovella and A. Bestravos. Self-similarity in the World Wide Web traffic: Evidence and possible causes. *IEEE/ACM Transactions on Networking*, 5(6):835–846, Dec. 1997.

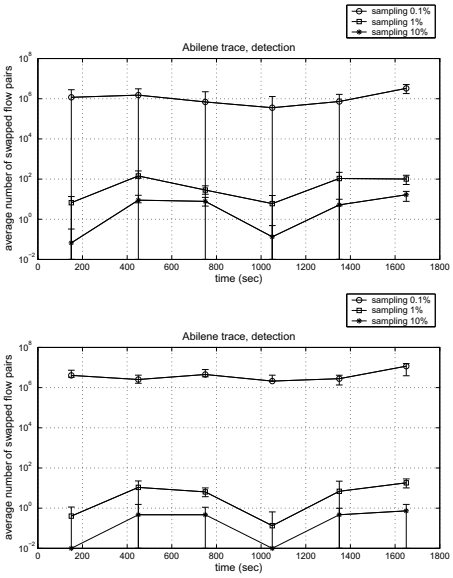

Figure 11: Performance of blind (top) and protocol-aware (bottom) detection over multiple 300s intervals (Abilene trace). Vertical bars show the standard deviation over multiple experiments.

[7] E. Demaine, A. Lopez-Ortiz, and I. Munro. Frequency estimation of internet packet streams with limited space. In *Proceedings of 10th Annual European Symposium on Algorithms*, 2002.

[8] N. G. Duffield, C. Lund, and M. Thorup. Properties and prediction of flow statistics from sampled packet streams. In *Proceedings of ACM Sigcomm Internet Measurement Workshop*, Nov. 2002.

[9] N. G. Duffield, C. Lund, and M. Thorup. Estimating flow distributions from sampled flow statistics. In *Proceedings of ACM Sigcomm*, Aug. 2003.

[10] C. Estan and G. Varghese. New directions in traffic measurement and accounting. In *Proceedings of ACM Sigcomm*, Aug. 2002.

[11] N. Hohn and D. Veitch. Inverting sampled traffic. In *Proceedings of ACM Sigcomm Internet Measurement Conference*, Oct. 2003.

[12] J. Jedwab, P. Phaal, and B. Pinna. Traffic estimation for the largest sources on a network, using packet sampling with limited storage. Technical Report HPL-92-35, HP Laboratories, Mar. 1992.

[13] Metropolis: METROlogie Pour l'Internet et ses services. http://www.laas.fr/ owe/METROPOLIS/metropolis_eng.html.

[14] T. Mori, M. Uchida, R. Kawahara, J. Pan, and S. Goto. Identifying elephant flows through periodically sampled packets. In *Proceedings of ACM Sigcomm Internet Measurement Conference*, Oct. 2004.

[15] NLANR: National Laboratory for Applied Network Research. http://www.nlanr.net.

[16] Packet Sampling Working Group. Internet Engineering Task Force. http://www.ietf.org/html.charters/psamp-charter.html.

[17] K. Papagiannaki, N. Taft, and C. Diot. Impact of flow dynamics on traffic engineering design principles. In *Proceedings of IEEE Infocom*, Hong Kong, China, Mar. 2004.

[18] Renater. http://www.renater.fr.

[19] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A transport protocol for real-time applications. RFC 1889, Jan. 1996.

[20] A. Shaikh, J. Rexford, and K. G. Shin. Load-sensitive routing of long-lived IP flows. In *Proceedings of ACM Sigcomm*, Sept. 1999.

[21] M. Spiegel. *Theory and Problems of Probability and Statistics*. McGraw-Hill, 1992.

Annexe C

Estimating Membership in a Multicast Session

Sara Alouf, Eitan Altman, Chadi Barakat, Philippe Nain

Appeared in proceedings of ACM SIGMETRICS, San Diego, CA, June 2003. A long version was published in IEEE Transactions on Signal Processing - Special Issue on Signal Processing in Networking, vol. 51, no. 8, pp. 2165-2176, August 2003.

Estimating Membership in a Multicast Session

Sara Alouf*
Vrije Universiteit
Department of Mathematics
De Boelelaan 1081a
1081 HV Amsterdam, The Netherlands
alouf@cs.vu.nl

Eitan Altman
INRIA
2004 Route des Lucioles
B.P. 93
06902 Sophia Antipolis, France
altman@sophia.inria.fr

Chadi Barakat
INRIA
2004 Route des Lucioles
B.P. 93
06902 Sophia Antipolis, France
cbarakat@sophia.inria.fr

Philippe Nain
INRIA
2004 Route des Lucioles
B.P. 93
06902 Sophia Antipolis, France
nain@sophia.inria.fr

ABSTRACT

We propose two novel on-line estimation algorithms to determine the size of a dynamic multicast group. We first use a Wiener filter to derive an optimal estimator for the membership size of the session in case the join process is Poisson and the lifetime of participants is distributed exponentially. We next develop the best first-order linear filter from which we derive an estimator that holds for any lifetime distribution. We apply this approach to the case where the lifetime distribution is hyperexponential. Both estimators hold under any traffic regime. Applying both estimators on real traces corresponding to video sessions, we find that both schemes behave well, one of which performs slightly better than the other in some cases. We further provide guidelines on how to tune the parameters involved in both schemes in order to achieve high quality estimation while simultaneously avoiding feedback implosion.

Categories and Subject Descriptors

G.3 [**Probability and Statistics**]: Queueing theory, Stochastic processes; H.4.3 [**Information Systems Applications**]: Communications Applications—*Computer conferencing, teleconferencing, and videoconferencing*

General Terms

Algorithms, Performance, Measurement

*This work was done when the author was at Inria Sophia Antipolis, France.

SIGMETRICS'03, June 10–14, 2003, San Diego, California, USA.

Keywords

On-line estimation, multicast applications, membership size, $M/G/\infty$ queue, Wiener filter, first-order linear filter

1. INTRODUCTION

Since its introduction, IP multicast has seen slow deployment in the Internet. As stated in [6], the service model and architecture do not efficiently provide or address many features required for a robust implementation of multicast. However, the fact remains that IP multicast is very appealing in offering scalable point-to-multipoint delivery specially in satellite communications. Current research efforts tend to propose alternatives to IP multicast like the so-called "application layer multicast" [4, 9, 14, 19], the idea being to deploy multicast at the application layer. Also, new models to support multicast communications in a more effective way have been proposed, such as the EXPRESS multicast [13]. The latter is an extension to IP multicast that provides explicit support for large-scale multicast applications such as real-time stock quote dissemination, live sports video feeds or Internet radio and TV. EXPRESS provides as well a *best-effort* count of the number of subscribers.

This paper is motivated by the conviction that large-scale multicast applications will be widely deployed in the future as soon as the capability becomes available. We believe that membership estimates will be an essential component of this widespread deployment as they can be very useful for scalable multicast. The membership of a session can be used for feedback suppression as it is the case in current protocols such as RTP [21] and SRM [8]. In order to regulate the amount of session/control messages sent by receivers – the idea being not to exceed 5% of overall session bandwidth – these protocols use delay timers that are tuned based on the membership estimates.

The membership of a multicast session can be used for charging the sources in large-scale applications. ISPs traditionally charge their customers on an input-rate basis. An alternative pricing scheme would be to charge sources based

on their audience size which is more profitable in the case of millions of subscribers.

Estimating the size of a multicast session can be quite useful to many applications. Bolot, Turletti and Wakeman [3] use membership estimation to further estimate the proportion of congested receivers as needed in their videoconference system IVS. Future Internet radios and TVs will need to characterize their audience preferences and to follow the fluctuations of the audience size. Dutta, Schulzrinne and Yemini proposed an architecture for Internet radio and TV called MarconiNet [7] that relies on RTCP [21, 20]. Even though RTCP provides an easy mechanism for collecting statistics on the size of the audience, it does not scale well to large multicast session [7]. In such applications, sampling-based techniques are more appropriate.

There has been a significant research effort in devising sampling-based schemes for the estimation of the membership in multicast sessions [3, 17, 18, 10, 16, 2]. The feedback algorithms presented in [3, 17, 18, 10, 16] are all *at-least-one* scenarios in the sense that the membership estimation is based on at least one acknowledgement (ACK) coming from the receivers. In these probabilistic schemes, the receivers send ACKs to the source as a reply to a specific request, either with a certain probability as in [3] or after some random time like in [17, 18, 16]. But what is common to these schemes (except for [17]) is that they all assume that the size of the group does not change during the estimation process. In a recent work [2], we propose a dynamic scheme that tracks the variations of the membership in an optimal way. The estimation algorithm used is quite simple: the source requests from its receivers to send ACKs with probability p every S seconds; it collects the amount of ACKs received at each observation step and filters out these measurements to estimate the membership. To derive the optimal estimator we rely on a diffusion approximation for the heavy-traffic regime. Under the assumptions of Poisson join times and exponentially distributed connection times, the diffusion approximation yields linear dynamics which enables the design of the optimal filter using Kalman filter theory.

In this paper we propose two novel algorithms for estimating the membership based on the polling scheme presented in [2]. Our purpose is to develop an estimator under more general assumptions than the ones used in [2]. Our first approach is based on a Wiener filter, which provides the optimal dynamic estimator among all linear estimators. The dynamics is not required to be linear as in the case of the Kalman filter, which allows us to remove the heavy-traffic assumption made in [2]. Yet, in order to obtain explicit expressions for the parameters of the Wiener filter, we still have to assume that participants join the session according to a Poisson process, and that the time during which they stay in the multicast session, hereafter referred to as *on-time*, has an exponential distribution. Under these assumptions we design the optimal linear estimation scheme that turns out to require a filter of order one. Motivated by this structure, we then design an efficient estimation scheme for generally distributed on-times. To that end, we identify the optimal filter among all linear filters of order one. We illustrate this approach in the case of hyperexponentially distributed on-times. Alike the estimator developed with the Wiener filter, the latter estimator is valid under any traffic regime. Both estimators are then tested on real traces. Despite the fact that these traces violate the assumptions under which the estimators have been derived, very good performance are observed.

REMARK 1.1. *The material presented in this paper does not require the specific use of IP multicast, nor any other multicast protocol. The solutions proposed hereafter are meant to be deployed at the application layer, and only require a multicast delivery of the requests for ACKs. This delivery can be achieved either by IP multicast, or by an application-layer multicast, or even by a* new *multicast technology. It is however assumed that the IP address of the source is available to all receivers in the session.*

The paper is organized as follows: the mathematical model of the membership is introduced in Section 2. The theory of Wiener filters is briefly presented in Section 3 and its application to the $M/M/\infty$ model comes in Section 3.1. The optimal first-order linear filter is developed in Section 4. Section 5 proposes some guidelines on how to choose parameters p and S. The robustness of both estimators is addressed through validations on real traces in Section 6. Finally, open issues are discussed in Section 7 and concluding remarks are given in Section 8.

2. MULTICAST GROUP MODELED AS AN $M/G/\infty$ QUEUE

We consider a multicast group that participants join and leave at random times. Let T_i and $T_i + D_i$ be the join time and leave time, respectively, of the i-th participant. In the following, D_i denotes the on-time of the i-th participant. Let $N(t)$ be the number of participants in the multicast group at time t or, equivalently, the size of the multicast session at time t. Clearly

$$N(t) = \sum_{i \geq 1} \mathbf{1}(T_i \leq t < T_i + D_i) \tag{1}$$

where $\mathbf{1}(E)$ equals 1 if the event E occurs and 0 otherwise.

We shall assume that the join times form a homogeneous Poisson process (with constant intensity $0 < \lambda = 1/\mathbf{E}[T_{i+1} - T_i]$) and that the on-times form a renewal sequence of random variables (rvs) with common probability distribution $\Psi(x) = P(D_i < x)$, $0 < \mathbf{E}[D_i] < \infty$, further independent of the join times. In the following D will denote a generic rv with probability distribution $\Psi(x)$.

In the queueing terminology, $\{N(t),\, t \geq 0\}$ represents the occupation process (number of busy servers) in an $M/G/\infty$ queue [15].

At times $t = nS$, $n = 0, 1, \ldots$, with $S > 0$ a constant, each participant to the multicast session sends an ACK to the source with probability $0 < p < 1$ and does not send any feedback information to the source with probability $1 - p$. We assume that ACKs cannot be lost. However, this assumption can be relaxed if one knows the loss probability as it is possible to incorporate it in our feedback mechanism.

The ACK interval S between two consecutive polling instants has to be larger than the largest round-trip time between a receiver and the source, so that all of the ACKs produced in a round reach the source before the (automatic) start of the next round. Note that in practice the source will have to regularly multicast the pair (p, S) to ensure that each participant will know these values. Throughout the paper, p and S are held fixed (see Section 5 for possible extensions).

Let Y_n be the number of ACKs received by the source at time nS. Based on the knowledge of $Y_1, \ldots, Y_n$, our objective is to find an optimal estimator (in a sense to be defined below) $\hat{N}_n$ for $N_n := N(nS)$, the size of the multicast group at time nS. In filtering parlance, Y_n is an input signal and we want to generate another signal $\hat{N}_n$ that is as close as possible to an unknown signal N_n (e.g. by minimizing the mean square error).

For later use we briefly review some results on the $M/G/\infty$ queue. In steady-state, the number N of busy servers is a Poisson random variable with parameter $\rho := \lambda E[D]$, namely, $P[N = j] = \rho^j \exp(-\rho)/j!$. In particular, both the mean and the variance of the number of busy servers are equal to ρ. The autocovariance function of the stationary version of the process $\{N(t),\, t \geq 0\}$, also denoted by $\{N(t),\, t \geq 0\}$, is given by [5, Eqn (5.39)]

$$\mathrm{Cov}(N(t), N(t+h)) = \lambda \int_{|h|}^{\infty} P(D > u)\, du. \qquad (2)$$

In the following we will denote by $\mathrm{Cov}_X(\cdot)$ the autocovariance function of any second-order discrete-time stationary process $\{X_n, n = 0, 1, \ldots\}$. With this notation and the definition of the process $\{N_n\}_n$, we see from (2) that

$$\mathrm{Cov}_N(k) = \rho\, \gamma^{|k|}, \quad k = 0, \pm 1, \ldots, \qquad (3)$$

with $\gamma := \exp(-\mu S)$, when the on-times $\{D_i\}_i$ are exponentially distributed with mean $1/\mu$.

Throughout the paper, we will assume that

$$\sum_{k \geq 0} \mathrm{Cov}_N(k) < \infty. \qquad (4)$$

In other words, we will exclude the situation where the on-times are heavy-tailed (e.g. Pareto distribution).

3. WIENER FILTER

Our objective is to transform a signal Y_n (noisy observation) into another signal $\hat{N}_n$ (estimator) that is the closest to an unknown signal N_n. By closest we mean that the mean error is zero (i.e. $\mathbf{E}[\hat{N}_n] = \mathbf{E}[N_n]$) and that the mean square error is minimized.

Such a transformation can be achieved by the Wiener filter that identifies the optimal linear filter [11]. This approach gives the transfer function of the linear filter, which can be transformed back to the time domain to obtain the impulse response of the filter. From the impulse response of the filter, the expression of $\hat{N}_n$ as a function of Y_n and, possibly, of $\hat{N}_{n-1}, \hat{N}_{n-2}, \ldots$, can be found. We will detail this procedure below.

Since a filter that minimizes the mean square error when the underlying processes are centered also minimizes the mean square error when the same processes are non-centered, we will derive the Wiener filter for the centered (stationary) versions of processes $\{N_n\}_n$, $\{\hat{N}_n\}_n$ and $\{Y_n\}_n$, denoted by $\{\nu_n\}_n$, $\{\hat{\nu}_n\}_n$ and $\{y_n\}_n$, respectively. We have observed in the previous section that $\mathbf{E}[N_n] = \rho$. On the other hand

$$\mathbf{E}[Y_n] = \mathbf{E}[\mathbf{E}[Y_n \mid N_n]] = \mathbf{E}[p\, N_n] = p\rho. \qquad (5)$$

Therefore $\nu_n = N_n - \rho$, $\hat{\nu}_n = \hat{N}_n - \rho$ and $y_n = Y_n - p\rho$.

Throughout the paper, z is a complex number such that $|z| = 1$. Introduce

$$S_y(z) = \sum_{k=-\infty}^{\infty} \mathrm{Cov}_y(k) z^{-k}$$

the z-transform of the autocovariance function (also called the power spectrum) of $\{y_n\}_n$.

Let $\mathrm{Cov}_{\nu y}(k) = \mathbf{E}[\nu_{n-k}\, y_n]$ be the cross-correlation function of processes $\{\nu_n\}_n$ and $\{y_n\}_n$. We also introduce[1]

$$S_{\nu y}(z) = \sum_{k=-\infty}^{\infty} \mathrm{Cov}_{\nu y}(k) z^{-k}$$

the z-transform of $\mathrm{Cov}_{\nu y}(k)$. We can express $\mathrm{Cov}_y(k)$ and $\mathrm{Cov}_{\nu y}(k)$ in terms of $\mathrm{Cov}_\nu(k)$ as follows

$$\mathrm{Cov}_y(k) = p^2 \mathrm{Cov}_\nu(k) + \mathbf{1}(k=0)\rho p(1-p) \qquad (6)$$

$$\mathrm{Cov}_{\nu y}(k) = p\, \mathrm{Cov}_\nu(k) \qquad (7)$$

where we have used the identity $\mathrm{Cov}_\nu(k) = \mathrm{Cov}_N(k)$. We are now in position to derive the Wiener filter. First, we write $S_y(z)$ as

$$S_y(z) = \sigma G(z) G(z^{-1}) \qquad (8)$$

where σ is a constant. This operation is called the canonical factorization of the power spectrum of $\{y_n\}_n$. The function $G(z)$ is the part of $S_y(z)$ that has all its zeros and poles inside the unit circle. The function $1/G(z)$ is the transfer function of the whitening filter: it transforms $\{y_n\}_n$ into a white noise process with variance σ.

Next, we form the ratio $S_{\nu y}(z)/G(z^{-1})$. This ratio is interpreted as the transfer function of a linear filter. The impulse response of this filter has values at the left and the right of the time origin (non-causal filter). We look for the transfer function of the part of the impulse response at the right of the time origin. This can simply be done by expanding the transfer function into fractions and then taking only the fractions with zeros and poles in the unit circle. In other words, we transfer the filter from a non-causal filter into a causal one. We denote the transfer function of the causal version of the filter by

$$H(z) = \left[\frac{S_{\nu y}(z)}{G(z^{-1})} \right]_+ .$$

The transfer function of the optimal filter is given by [11]

$$H_o(z) = \frac{H(z)}{\sigma G(z)}.$$

It remains to invert this transfer function back into the time domain to find the desired recurrence between $\hat{\nu}_n$ and y_n and, subsequently, between the non-centered variables $\hat{N}_n$ and Y_n. This procedure is illustrated in Section 3.1 for the case that the underlying model is the $M/M/\infty$ queue.

3.1 Application to the $M/M/\infty$ model

In light of the results reported in Section 3, all what we have to do is to find expressions for $S_y(z)$ and $S_{\nu y}(z)$. This can easily be done when the underlying model is the $M/M/\infty$ queueing model, as shown below.

[1]Observe from (6) and (7) that both $S_y(z)$ and $S_{\nu y}(z)$ are well-defined for $|z| = 1$ under the assumption (4).

Let us first determine $S_y(z)$. By using (6) and (3) together with the property that $\mathrm{Cov}_N(k) = \mathrm{Cov}_\nu(k)$, we find

$$\mathrm{Cov}_y(k) = \begin{cases} p^2\rho\gamma^{|k|}, & \text{for } k \neq 0 \\ p\rho, & \text{for } k = 0. \end{cases}$$

Since $\gamma < 1$ and $|z| = 1$, the z-transform of $\mathrm{Cov}_y(k)$ is

$$S_y(z) = \frac{p\rho\left[\gamma(p-1)z^2 + [1+\gamma^2(1-2p)]z + \gamma(p-1)\right]}{z(1-\gamma z)(1-\gamma z^{-1})}.$$

The second-order polynomial in the variable z in the numerator has two positive real roots given by r and $1/r$, with

$$r = \frac{1+\gamma^2(1-2p) - \sqrt{(1-\gamma^2)[1-\gamma^2(1-2p)^2]}}{2\gamma(1-p)}.$$

Note that $r < 1$. Hence

$$\begin{aligned} S_y(z) &= \frac{\gamma\rho p(1-p)}{r}\left[\frac{(1-rz)(1-rz^{-1})}{(1-\gamma z)(1-\gamma z^{-1})}\right] \\ &= \sigma\, G(z)\, G(z^{-1}) \end{aligned}$$

with

$$\sigma := \frac{\gamma\rho p(1-p)}{r} \quad \text{and} \quad G(z) := \frac{1-rz^{-1}}{1-\gamma z^{-1}}.$$

We now compute $S_{\nu y}(z)$. From (7) and (3) we find

$$\mathrm{Cov}_{\nu y}(k) = p\rho\gamma^{|k|}$$

so that

$$S_{\nu y}(z) = \frac{p\rho(1-\gamma^2)}{(1-\gamma z)(1-\gamma z^{-1})}.$$

The transfer function $H(z)$ is given by

$$H(z) = \left[\frac{S_{\nu y}(z)}{G(z^{-1})}\right]_+ = \frac{\rho p(1-\gamma^2)}{(1-\gamma r)(1-\gamma z^{-1})}$$

and the transfer function $H_o(z)$ of the optimal filter takes here the simple form

$$H_o(z) = \frac{\rho p(1-\gamma^2)}{\sigma(1-\gamma r)(1-rz^{-1})} = \frac{B}{1-Az^{-1}}$$

where

$$A = r, \qquad B = \frac{\rho p(1-\gamma^2)}{\sigma(1-\gamma r)} = \frac{r(1-\gamma^2)}{\gamma(1-p)(1-\gamma r)}.$$

The impulse response of this linear filter is given by the first-order recurrence relation [11]

$$\hat{\nu}_n = A\hat{\nu}_{n-1} + By_n$$

with $\hat{\nu}_n$ the estimator of ν_n. We now return to the original processes $\{N_n\}_n$ and $\{Y_n\}_n$, to finally obtain the optimal linear filter:

$$\hat{N}_n = A\hat{N}_{n-1} + BY_n + \rho(1-A-pB). \tag{9}$$

It is interesting to compare this filter with the Kalman filter derived in [2][2]. They appear to be the same! This result is somehow expected, since both the Kalman filter and the Wiener filter are optimal (among the class of linear filters) in the sense that they minimize the mean square error. The key point is that the Kalman filter used in [2] was derived under a heavy traffic assumption, while the Wiener filter computed in the present paper holds for any value of the model parameters λ and μ. This partly explains why the estimator in [2] behaves well under light or moderate traffic as experimentally observed in that paper.

[2]Recall that a Kalman filter is the optimal filter under the condition of linear dynamics and observation, which does not hold in our case. However, the dynamics does converge to a linear diffusion as the traffic load tends to infinity, allowing us in [2] to obtain a Kalman filter which is optimal for the asymptotic heavy traffic regime.

We conclude this section by computing the mean square error $\epsilon_{min} := \mathbf{E}[(N_n - \hat{N}_n)^2]$ of our estimator. It is known that [11]

$$\epsilon_{min} = \sum_{k=1}^{M} \mathrm{Res}\,[F(z), z_k]$$

with

$$F(z) := \frac{1}{z}\left[S_\nu(z) - H_o(z)S_{\nu y}\left(z^{-1}\right)\right]$$

where $z_1, \ldots, z_M$ are the poles (if any) of the function $F(z)$ inside the unit circle. The notation $\mathrm{Res}\,[F(z), z_k]$ stands for the residue of $F(z)$ at point $z = z_k$, namely, the coefficient of $1/(z - z_k)$ in the Laurent series expansion of $F(z)$ in the vicinity of z_k.

Specializing $F(z)$ to the values of $S_\nu(z)$, $S_{\nu y}(z)$ and $H_o(z)$ found earlier, yields

$$F(z) = \frac{\rho(1-\gamma^2)((1-Bp)z - A)}{(1-\gamma z)(z-\gamma)(z-A)}.$$

This function has two poles inside the unit circle which are located at $z = A$ and $z = \gamma$; the residues of $F(z)$ at these poles are given by $-\rho pAB(1-\gamma^2)/((1-\gamma A)(A-\gamma))$ and $\rho(1+pB\gamma/(A-\gamma))$, respectively. Summing up these residues gives

$$\epsilon_{min} = \rho\left(1 - \frac{Bp}{1-\gamma A}\right).$$

By using the expressions of A and B, we finally obtain

$$\epsilon_{min} = \rho\frac{-(1-\gamma^2) + \sqrt{(1-\gamma^2)(1-\gamma^2(1-2p)^2)}}{2\gamma^2 p}. \tag{10}$$

This expression for ϵ_{min} can be used to tune the parameters p and γ or equivalently S (see Section 5).

4. OPTIMAL FIRST-ORDER LINEAR FILTER

The theory reported in Section 3 applies to any on-time distribution $\Psi(x)$ (with the exception of heavy-tailed distributions). However, it is not easy to identify the function $G(z)$ that appears in the canonical factorization of the spectrum $S_y(z)$ (see (8)) and thereby the optimal filter, except when the on-times are exponentially distributed rvs (see Section 3.1).

In this section we will determine the first-order linear filter that minimizes the mean square error. Observe that, unlike the Wiener filter, the proposed approach will not return the optimal filter among all linear filters but simply the optimal linear filter among all first-order linear filters. We will illustrate this approach at the end of this section in the case where $\Psi(x)$ is an hyperexponential distribution.

Recall the definition of the centered processes $\{\nu_n\}_n$, $\{\hat{\nu}_n\}_n$ and $\{y_n\}_n$ made at the beginning of Section 3.

The methodology is simple: we want to find constants $A \in (0,1)$ and B such that $\epsilon := \mathbf{E}[(\nu_n - \hat{\nu}_n)^2]$ is minimized when the process $\{\hat{\nu}_n\}_n$ satisfies the following first-order recurrence relation

$$\hat{\nu}_n = A\hat{\nu}_{n-1} + By_n. \tag{11}$$

In steady-state this implies that

$$\hat{\nu}_n = B\sum_{k=0}^{\infty} A^k y_{n-k}. \tag{12}$$

The mean square error ϵ is equal to

$$\epsilon = \mathbf{E}\left[\hat{\nu}_n^2\right] + \mathbf{E}\left[\nu_n^2\right] - 2\mathbf{E}\left[\hat{\nu}_n\nu_n\right].$$

We have $\mathbf{E}[\nu_n^2] = \mathbf{E}[(N_n - \rho)^2] = \rho$ (see Section 2). From (12) and (7) we find

$$\mathbf{E}\left[\hat{\nu}_n\nu_n\right] = pB\sum_{k=0}^{\infty} A^k \mathrm{Cov}_\nu(k) = pBg(A)$$

where

$$g(z) := \sum_{k=0}^{\infty} z^k \mathrm{Cov}_\nu(k). \tag{13}$$

The power series $g(z)$ converges for $|z| < 1$ (Hint: $k \to \mathrm{Cov}_\nu(k)$ is nonincreasing) and is therefore differentiable for $|z| < 1$. We will denote by $g'(z)$ its derivative.

It remains to express $\mathbf{E}[\hat{\nu}_n^2]$ in terms of the parameters A and B. Squaring both sides of (11) and then taking the expectation, yields

$$\mathbf{E}\left[\hat{\nu}_n^2\right] = \left(\frac{B}{1-A^2}\right)\left(2A\mathbf{E}[\hat{\nu}_{n-1}y_n] + B\mathbf{E}\left[y_n^2\right]\right).$$

With the identities $\mathbf{E}[y_n^2] = \mathrm{Cov}_y(0) = \rho p$ (see (6)) and $\mathbf{E}[\hat{\nu}_{n-1}y_n] = Bp^2\left(g(A) - \rho\right)/A$ (Hint: use (12), (7) and $\mathrm{Cov}_\nu(0) = \rho$), we obtain

$$\mathbf{E}\left[\hat{\nu}_n^2\right] = \left(\frac{pB^2}{1-A^2}\right)\left(2pg(A) + \rho(1-2p)\right).$$

Finally, the mean square error is

$$\epsilon = \rho - 2pBg(A) + \left(\frac{pB^2}{1-A^2}\right)\left(2pg(A) + \rho(1-2p)\right). \tag{14}$$

In order to minimize ϵ, $A \in (0,1)$ and B must be the solution of the following system of equations:

$$\begin{cases} \dfrac{\partial\epsilon}{\partial A} = \dfrac{2pB}{1-A^2}\left(AB\left[\dfrac{2pg(A)+\rho(1-2p)}{1-A^2}\right] + g'(A)(pB - (1-A^2))\right) = 0 \\ \dfrac{\partial\epsilon}{\partial B} = 2p\left(B\left[\dfrac{2pg(A)+\rho(1-2p)}{1-A^2}\right] - g(A)\right) = 0. \end{cases}$$

The 2nd equation gives

$$B = \frac{g(A)(1-A^2)}{2pg(A) + \rho(1-2p)}. \tag{15}$$

Substituting this value of B into the 1st equation shows that A must satisfy

$$Ag(A)(2pg(A) + \rho(1-2p)) - g'(A)(1-A^2)(pg(A) + \rho(1-2p)) = 0. \tag{16}$$

If this equation has a unique solution $A \in (0,1)$, then substituting this value of A into (15) will give the optimal pair (A, B).

It is shown in Appendix A that (16) has always a unique solution in $[0,1)$ (in particular) if $g'(x) > 0$ in $[0,1)$. This condition will hold as long as $P(D > S) > 0$. In practice, one can always select S such that this condition is true.

The reader can check that the filter defined in (11) with the optimal pair (A, B) is the same as the Wiener filter found in Section 3.1 when the on-times are exponentially distributed.

We now illustrate the approach developed in this section by considering the situation where on-times have a hyper-exponential distribution. More precisely, we assume that

$$\Psi(x) = 1 - \sum_{l=1}^{L} p_l e^{-\mu_l x} \tag{17}$$

with $0 < p_l < 1$, $l = 1, 2, \ldots, L$, and $\sum_{l=1}^{L} p_l = 1$. In this setting the underlying queueing model can be seen as L independent $M/M/\infty$ queues in parallel. The arrival rate to queue l is $p_l\lambda$ and the service rate is μ_l. Define $\gamma_l := \exp(-\mu_l S)$, $\rho_l := p_l\lambda/\mu_l$ so that $\rho = \sum_{l=1}^{L}\rho_l$. The autocovariance function of the process $\{\nu_n\}_n$ is equal to

$$\mathrm{Cov}_\nu(k) = \begin{cases} \rho & \text{for } k = 0 \\ \sum_{l=1}^{L}\rho_l\gamma_l^{|k|} & \text{for } k \neq 0 \end{cases}$$

so that

$$g(A) = \sum_{l=1}^{L}\frac{\rho_l}{1 - A\gamma_l}.$$

Numerical example[3]*:* $L = 2$, $p = 0.0106$ and $S = 2.5s$. Also

$1/\mu_1$	=	$3897s$	ρ_1	=	19.5	γ_1	= 0.999359
$1/\mu_2$	=	$480061s$	ρ_2	=	75.1	γ_2	= 0.999995
$1/\mu$	=	$18316s$	ρ	=	94.7		

The optimal first-order filter is

$$\hat{N}_n = 0.99879456\,\hat{N}_{n-1} + 0.10720289\,Y_n + 0.006540864.$$

For comparison the Wiener filter found in Section 3.1 (for exponential on-times) for these values is

$$\hat{N}_n = 0.99828589\,\hat{N}_{n-1} + 0.14885344\,Y_n + 0.012900081.$$

5. GUIDELINES ON CHOOSING ACKS PARAMETERS

A "good" pair (p, S) should (i) limit the feedback implosion while at the same time (ii) achieve a good quality of the estimator. Of course (i) and (ii) are antinomic and therefore a trade-off must be found. This trade-off will be formalized as follows: we want to select a pair (p, S) so that the mean number of ACKs generated every S seconds (see (5)) and the relative error of the variance of the estimator (denoted as η) are bounded from above by given constants, namely

$$\begin{cases} \mathbf{E}[Y_n] = p\rho \leq \alpha \\ \eta = \dfrac{Var(N_n) - Var(\hat{N}_n)}{Var(N_n)} \leq \beta. \end{cases} \tag{18}$$

[3]The values of the parameters come from the trace called $video_1$ investigated in Section 6.

When $\hat{N}_n$ is optimal then $Var(N_n) - Var(\hat{N}_n) = \mathbf{E}[(N_n - \hat{N}_n)^2]$ and η becomes the "normalized mean square error" [12, p. 202]. Optimality was shown for the $M/M/\infty$ queue, therefore

$$\eta = \frac{\epsilon_{min}}{\rho}$$

with ϵ_{min} given in (10).

For given constants α and β, it is easy to solve the constrained optimization problem defined in (18), provided that η is known.

For the $M/M/\infty$ model, where ϵ_{min} is given in (10), we find that $p = \alpha/\rho$ and that S, or equivalently γ, is the unique positive solution of the equation $\epsilon_{min} = \rho\beta$.

The problem now is to choose constants α and β so that conditions (i) and (ii) are satisfied. We have found that α in the range $[0.5, 1]$ and $\beta \leq 0.15$ give satisfactory results.

We conclude this section with general remarks on how to adapt the parameters p and S to important variations in the membership. The estimation schemes in Sections 3.1 and 4 have been obtained under the assumption that parameters p and S are fixed. However, the filters constructed in Sections 3.1 and 4 can still be used if p and/or S change over time, provided that these modifications do not prevent the system to be most of the time in steady-state. In that setting, a new filter will have to be recomputed after each modification. Such a modification can be called for each time the number of ACKs received during a given period of time significantly deviates from the current expectation (i.e. $p\rho$).

6. VALIDATION WITH REAL TRACES

In this section we apply the estimators developed in Sections 3.1 and 4 to four traces corresponding to video muticast sessions. Two types of estimators will be used: the estimator – denoted as $\hat{N}_n^E$ – found in (9) when the population is modeled as an $M/M/\infty$ queue; the estimator – denoted as $\hat{N}_n^{H_2}$ – derived in Section 4 in case the join times are Poisson and the on-times have a 2-stage hyperexponential distribution ($M/H_2/\infty$ model).

The objective is twofold: we want to investigate the quality of both estimators when compared to real life conditions, and we want to identify the best one.

We have collected four MBone traces – denoted $video_i, i = 1, \ldots, 4$ – between August 2001 and September 2001 using the *MListen* tool [1]. Each trace corresponds to a long-lived video session (see duration of each session in Table 1, where the superscript "d" stands for "days"). We have run both algorithms (estimators) on each trace.

For each trace we have identified the parameters of the $M/M/\infty$ model (parameters λ and μ, or equivalently parameters ρ and μ) and of the $M/H_2/\infty$ model (parameters ρ, μ_1, μ_2, p_1 and $p_2 = 1 - p_1$ – see definitions in Section 4). The values of these parameters are reported in columns 3–8 in Table 1. Details on how these values are retrieved come in Appendix B.

Parameters p and S have been chosen by following the guidelines presented in Section 5, namely $\alpha \in \{0.5, 1\}$ and $\beta \in \{0.1, 0.15\}$. Values of these parameters are listed in columns 9–10 in Table 1. The performance of estimators $\hat{N}_n^E$ and $\hat{N}_n^{H_2}$ are reported in Tables 2 and 3.

Table 2 reports several order statistics (columns 3–7) and the sample mean of the relative error $\frac{|N_n - \hat{N}_n|}{N_n}$ (column 2), where $\hat{N}_n$ is either $\hat{N}_n^E$ or $\hat{N}_n^{H_2}$. All results are expressed in percentages. The first observation is that both estimators perform reasonably well. The sample mean of the relative error is always less than 6.82% and is as low as 3.79%; when averaging over all experiments, this sample mean is less than 4.5% for both $\hat{N}_n^E$ and $\hat{N}_n^{H_2}$ (see last two rows). The last column gives the 95th percentile and reads as follows: the relative error achieved on trace $video_3$ by $\hat{N}_n^E$ (resp. $\hat{N}_n^{H_2}$) is 95% of the time less than 11.00% (resp. 12.56%). The second observation is that no scheme is uniformly better than the other over an entire session but their sample means are very close to each other (see column 2). For instance, $\hat{N}_n^E$ performs better than $\hat{N}_n^{H_2}$ regarding the 90th and the 95th percentiles whereas the result is reversed regarding the 25th percentile. It looks like the relative error on $\hat{N}_n^{H_2}$ is empirically more dispersed around its mean than is the relative error on $\hat{N}_n^E$, and has a longer tail. Across all sessions (see last two rows), 75% of the time $\hat{N}_n^{H_2}$ performs better than $\hat{N}_n^E$. This improvement does not come for free, since it requires the identification of 4 parameters (ρ, μ_1, μ_2 and p_1) instead of 2 (ρ and μ) for $\hat{N}_n^E$.

Table 2: Mean and percentiles of $|N_n - \hat{N}_n|/N_n$

Trace	Mean	25	50	75	90	95
$video_1$ $\hat{N}_n^E$	6.82	1.09	2.42	5.25	11.5	19.4
$\hat{N}_n^{H_2}$	6.12	1.08	2.55	6.31	13.5	20.6
$video_2$ $\hat{N}_n^E$	4.19	1.41	3.08	5.43	8.66	11.9
$\hat{N}_n^{H_2}$	4.12	0.98	2.14	4.41	8.78	12.6
$video_3$ $\hat{N}_n^E$	4.20	1.55	3.26	5.71	8.71	11.0
$\hat{N}_n^{H_2}$	3.98	1.07	2.36	4.83	9.35	12.6
$video_4$ $\hat{N}_n^E$	3.79	1.23	2.57	4.51	7.50	11.0
$\hat{N}_n^{H_2}$	4.06	1.02	2.21	4.39	8.98	14.7
overall $\hat{N}_n^E$	4.44	1.33	2.88	5.22	8.60	12.0
$\hat{N}_n^{H_2}$	4.34	1.02	2.26	4.73	9.61	14.2

Table 3: Empirical mean and variance of $N_n - \hat{N}_n$

Trace	Mean	Variance	ϵ_{min}, ϵ	η
$video_1$ $\hat{N}_n^E$	-0.112	12.664	13.942	0.147
$\hat{N}_n^{H_2}$	-0.047	12.851	12.120	
$video_2$ $\hat{N}_n^E$	0.006	0.495	1.407	0.099
$\hat{N}_n^{H_2}$	0.019	0.785	0.396	
$video_3$ $\hat{N}_n^E$	0.037	0.207	0.737	0.091
$\hat{N}_n^{H_2}$	0.019	0.229	0.208	
$video_4$ $\hat{N}_n^E$	0.052	0.911	1.566	0.087
$\hat{N}_n^{H_2}$	0.065	1.423	0.676	

Table 3 reports the sample mean and the sample variance of the error $N_n - \hat{N}_n$. In the 4th column, we list the theoretical variance. It is given by ϵ_{min} for $\hat{N}_n^E$ (see (10)) and by ϵ for $\hat{N}_n^{H_2}$ (see (14)). The expected average $\mathbf{E}[N_n - \hat{N}_n]$ is zero in both approaches. Both estimators $\hat{N}_n^E$ and $\hat{N}_n^{H_2}$ have almost no bias (see column 2), and their empirical variances closely match the theoretical ones given by ϵ_{min} and ϵ, respectively. It is of interest to point out that for the 4 traces studied, ϵ, the theoretical mean square error provided by $\hat{N}_n^{H_2}$, is smaller than ϵ_{min}, the theoretical mean square error provided by $\hat{N}_n^E$ (however, this result is reversed if we consider the empirical mean square errors). Thus, $\hat{N}_n^{H_2}$ is

Table 1: Parameter identification

Trace	Session lifetime	ρ	$1/\mu$	$1/\mu_1$	$1/\mu_2$	p_1	p_2	p	S	α	β
$video_1$	$3^d\,13^h\,33^m\,20^s$	94.7	18316	3897	480061	0.97	0.03	0.011	2.5	1.0	0.15
$video_2$	$11^d\ \ 1^h\,46^m\ \ 8^s$	14.1	16476	1	226498	0.93	0.07	0.034	3.2	0.5	0.1
$video_3$	$50^d\,22^h\,13^m\,20^s$	8.1	66823	1	900854	0.93	0.07	0.062	20.0	0.5	0.1
$video_4$	$29^d\,16^h\,43^m\,13^s$	17.9	83390	1	473268	0.82	0.18	0.028	10.0	0.5	0.1

Table 4: Distributions that best fitted into the inter-arrivals and on-times sequences.

Trace	Best fit for inter-arrivals sequence	Best fit for on-times sequence
$video_1$	Lognormal with $\mu = 3.38$, $d = 1.49$	Weibull with shape 0.35, scale 3700
$video_2$	Lognormal with $\mu = 5.20$, $d = 1.68$	Weibull with shape 0.26, scale 1400
$video_3$	Weibull with shape 0.65, scale 3500	Lognormal with $\mu = 5.08$, $d = 3.32$
$video_4$	Weibull with shape 0.55, scale 2700	Weibull with shape 0.18, scale 4000

more efficient[4] than $\hat{N}_n^E$ (again, $\hat{N}_n^E$ is empirically more efficient than $\hat{N}_n^{H_2}$). The last column provides the relative error on $Var(\hat{N}_n^E)$, called η ($= \epsilon_{min}/\rho$) in Section 5.

In Fig. 1 (resp. 2, 3 and 4) we plot the variations of membership for session $video_1$ (resp. $video_2$, $video_3$ and $video_4$), together with the estimates returned by $\hat{N}_n^E$ and $\hat{N}_n^{H_2}$. Among all 4 sessions, session $video_1$ presents the highest variations in N_n. Fig. 1(a) (resp. 2(a), 3(a) and 4(a)) displays three curves: the membership of the video session, the estimation returned by $\hat{N}_n^E$, labeled "Exponential", and the estimation returned by $\hat{N}_n^{H_2}$, labeled "Hyperexponential". It is clearly visible, especially at the left-hand side of graph 1(a), that $\hat{N}_n^E$ tracks better the session dynamics than $\hat{N}_n^{H_2}$. Both estimators $\hat{N}_n^E$ and $\hat{N}_n^{H_2}$ have been derived under some specific and restrictive assumptions: Poisson join times for both of them, exponential (resp. 2-stage hyperexponential) on-times for the first (resp. second) one. It is interesting to know whether or not these assumptions were violated in each session $video_i$, $i = 1, \ldots, 4$. We have therefore carried out a statistical analysis of each trace in order to determine the nature of their join time process and of their on-time sequence.

As shown in Table 4 and Figs. 1, 2, 3 and 4, parts (b) and (c), neither is the join time process Poisson nor are the on-times exponentially distributed (or hyperexponentially distributed) for any of the traces. The inter-join times and the on-times appear to follow subexponential distributions (Weibull and Lognormal distributions), a situation quite different from the assumptions under which the estimators have been obtained. Despite these significant differences, the estimators behave well and therefore show a good robustness to assumption violations.

In summary, both estimators perform very well when applied to real traces and are robust to significant deviations from their (theoretical) domain of validity. Estimator $\hat{N}_n^{H_2}$ returns the best global performance for the relative error criterion, but does not track high fluctuations as well as $\hat{N}_n^E$. Overall, we have found that $\hat{N}_n^E$ is a good estimator, both in terms of its performance and its usability since it only requires the knowledge of two parameters: ρ and μ.

[4]An estimator is said to be more efficient if it has a smaller variance.

7. OPEN ISSUES

The main pending issue concerns the knowledge of parameters ρ and μ (or equivalently any couple of parameters among ρ, λ and μ, since $\rho = \lambda/\mu$ in steady-state). When these parameters are not known, the source should estimate them. Again, the source could estimate any two parameters among ρ, λ and μ and infer the third one.

One possible way of estimating λ is to let a newly arrived receiver send a "heartbeat" to the source with a certain (constant) probability q (q should be small enough to avoid overwhelming the source with heartbeats). The source would then use the arrival time t_m of the mth heartbeat to estimate λ. The maximum likelihood estimator is $\hat{\lambda} = m/(qt_m)$. This estimator is unbiased and consistent by the strong law of large numbers ($\lim_{m\to\infty} t_m/m = 1/(q\lambda)$).

In a similar way, the source can estimate μ if receivers probabilistically send a "goodbye" message reporting their lifetime when they leave the session. Let $\tau_{m'}$ be the lifetime indicated in the m'th goodbye message received at the source, then the maximum likelihood estimator of μ is simply $\hat{\mu} = m'/(\sum_{i=1}^{m'} \tau_{m'})$. The estimator $\hat{\mu}$ is unbiased and consistent.

A natural estimator for ρ is $\hat{\rho} = \mathbf{E}[\hat{N}_n]$. As long as there is no estimation of both ρ and μ, it is not possible to compute the filter coefficient A and B. Then only a naive estimator for N_n can be used, defined as the ratio of the number of ACKs received Y_n over the ACK probability p. Notice that $\mathbf{E}[Y_n/p] = \rho$.

One might want to treat the estimation of N_n, μ and ρ as a joint parameter estimation problem. Unfortenately, the problem becomes much more complicated, and we are not able at the moment to say anything about its resolution. Alternatively, one might want to replace ρ with $\hat{\rho} = \mathbf{E}[\hat{N}_n]$ in (9) and study the performance of the resulting equation. It is possible to go even farther by replacing ρ with Y_n/p, instead of using $\mathbf{E}[Y_n/p]$. This will add more burstiness to the dynamics of $\hat{N}_n$ and (9) becomes

$$\hat{N}_n = A\hat{N}_{n-1} + (1 - A)\frac{Y_n}{p}.$$

The latter equation is nothing but an exponential weighted moving average (EWMA) estimator for the membership process N_n. We expect this EWMA estimator to perform better than $\hat{N}_n$, as given in (9), over highly dynamic sessions, as its

auto-regressive equation has been derived by replacing the constant term ρ with the bursty term Y_n/p. It is also expected that this EWMA estimator will react faster to important and sudden changes in the membership process. However, its performance over real sessions still needs to be investigated more carefully. Observe that the use of this EWMA estimator relies on the prior knowledge of $\gamma = \exp(-\mu S)$ solely, and therefore either "goodbye" or "hello" messages are still required to compute $\hat{\mu}$.

The first option is to directly estimate μ as indicated above, but the first estimate will be delayed until the reception of the first goodbye message, which might necessitates the departure of several customers. The second option is to estimate both ρ and λ as indicated earlier in the section and to estimate μ as $\hat{\lambda}/\hat{\rho}$, which is not expected to perform as well as the first option, but will return a first estimate much faster. At the expense of a larger warm-up period, we believe that the first option is preferrable. Another advantage of this option concerns the possible modification of q' by the source, in order to control the volume of goodbye messages sent, which clearly cannot be done with the hello messages.

8. CONCLUSION

The major contribution of this work is the design of two novel estimators for evaluating the membership in multicast sessions. We have designed estimators capable of efficiently tracking the dynamics of multicast sessions while simultaneously avoiding feedback implosion. In contrast to the estimator proposed in [2] which was designed under heavy traffic assumptions, our schemes do not place such restrictions.

Relying on the appealing Wiener filter theory, we have computed the optimal linear estimator for session membership when the underlying model is an $M/M/\infty$ queue. The optimality refers to the unbiasedness of the estimator and to the fact that the mean square error is minimized. We have also developed the optimal first-order linear filter in case the on-time distribution is arbitrary and have derived the associated estimator in case the on-times have a 2-stage hyperexponential distribution.

Both obtained estimators have been validated on real traces. Their performance have been shown to be excellent, one of them showing a good ability to adapt to highly dynamic multicast sessions.

9. ACKNOWLEDGMENTS

The authors wish to thank Prof. Patrick Thiran for helpful suggestions.

APPENDIX

A. EXISTENCE AND UNIQUENESS OF THE SOLUTION

LEMMA A.1. *Define*

$$\begin{aligned} f(x) &:= (2pg(x) + \rho(1-2p))xg(x) \\ &\quad -(pg(x) + \rho(1-2p))(1-x^2)g'(x), \end{aligned}$$

where $g(x)$ is given in (13).

If $g'(x) > 0$ for $x \in [0,1)$, then $f(x)$ has a unique zero in $[0,1)$.

Proof. Write $f(x)$ as $f(x) = f_+(x) - f_-(x)$ where

$$\begin{aligned} f_+(x) &:= [2p(g(x)-\rho)+\rho]\, xg(x) \\ f_-(x) &:= [p(g(x)-\rho)+\rho(1-p)]\,(1-x^2)g'(x). \end{aligned}$$

The derivative of $f_-(x)$ is given by $f'_-(x) = -\alpha x^2 - \beta x + \alpha$ using

$$\begin{aligned} \alpha &:= p\left(g'(x)\right)^2 + [p(g(x)-\rho)+\rho(1-p)]g''(x), \\ \beta &:= 2[p(g(x)-\rho)+\rho(1-p)]g'(x). \end{aligned}$$

Since $g'(x) > 0$ and $g(x) > \rho$ (see (13)), it is seen that $\alpha > 0$ and $\beta > 0$ which implies that $f''_-(x) = -2\alpha x - \beta < 0$ for $x \in [0,1)$. We therefore have that $f'_-(x)$ is strictly decreasing in $[0,1)$, with $f'_-(0) = \alpha > 0$ and $f'_-(1) = -\beta < 0$. Thus, the function $f'_-(x)$ has only one zero in $[0,1)$.

Therefore, and under the assumptions of the lemma, it is seen that:

(i) $f_+(x)$ is continuous and strictly increasing in $[0,1)$ with $f_+(0) = 0$;

(ii) $f_-(x)$ is continuous in $[0,1)$ and $f_-(1) = 0$. There exists $x_0 \in (0,1)$ such that $f_-(x)$ is strictly increasing in $[0,x_0)$, strictly decreasing in $(x_0,1)$ and $f'_-(x_0) = 0$.

We deduce from the above that $f(x)$ has a unique zero in $[0,1)$ if $g'(x) > 0$ in $[0,1)$. This condition will hold as long as $P(D > S) > 0$. In practice, one can always select S such that this condition is true.

B. COMPUTING PARAMETERS FROM TRACE

Each trace records $(T_i, D_i), i \geq 1$. To use the $M/M/\infty$ queue model, we identify $1/\lambda = \mathbf{E}[T_{i+1} - T_i]$ and $1/\mu = \mathbf{E}[D]$, and deduce $\rho = \lambda/\mu$. To use the $M/H_2/\infty$ queue model, λ is computed as before. Identifying μ_1, μ_2, p_1 and p_2 requires the knowledge of the first three moments of D (recall that $p_2 = 1 - p_1$). For a 2-stage hyperexponential distribution, the kth moment is given by

$$\mathbf{E}[D^k] = \sum_{l=1}^{2} \frac{p_l k!}{(\mu_l)^k} = k! \left(\frac{p_1}{(\mu_1)^k} + \frac{p_2}{(\mu_2)^k} \right), \text{ for } k \geq 1.$$

The parameters μ_1, μ_2, p_1 and p_2 are then solution to the following system of four equations, where σ_l stands for $1/\mu_l$ with $l = 1, 2$.

$$p_1 + p_2 = 1 \tag{19}$$

$$p_1\sigma_1 + p_2\sigma_2 = \mathbf{E}[D] \tag{20}$$

$$p_1\sigma_1^2 + p_2\sigma_2^2 = \mathbf{E}[D^2]/2 \tag{21}$$

$$p_1\sigma_1^3 + p_2\sigma_2^3 = \mathbf{E}[D^3]/6. \tag{22}$$

Equations (19) and (20) readily give

$$p_1 = \frac{\sigma_2 - \mathbf{E}[D]}{\sigma_2 - \sigma_1}, \qquad p_2 = \frac{\mathbf{E}[D] - \sigma_1}{\sigma_2 - \sigma_1}. \tag{23}$$

Substituting Equations (23) for p_1 and p_2 into (21) yields

$$\mathbf{E}[D^2]/2 = \mathbf{E}[D](\sigma_2 + \sigma_1) - \sigma_1\sigma_2. \tag{24}$$

Substituting them into (22) yields

$$\mathbf{E}[D^3]/6 = \mathbf{E}[D]((\sigma_2 + \sigma_1)^2 - \sigma_1\sigma_2) - \sigma_1\sigma_2(\sigma_2 + \sigma_1). \tag{25}$$

Introduce now S_σ and P_σ as the sum and the product of σ_1 and σ_2, respectively. Equations (24) and (25) become

$$\begin{aligned}\mathbf{E}[D^2]/2 &= \mathbf{E}[D]S_\sigma - P_\sigma \\ \mathbf{E}[D^3]/6 &= \mathbf{E}[D](S_\sigma^2 - P_\sigma) - P_\sigma S_\sigma \\ &= (\mathbf{E}[D^2]/2)\, S_\sigma - \mathbf{E}[D]P_\sigma\end{aligned}$$

where the latter identity is obtained when using the first one. It then follows that

$$S_\sigma = \frac{3\mathbf{E}[D]\mathbf{E}[D^2] - \mathbf{E}[D^3]}{3(2\mathbf{E}[D]^2 - \mathbf{E}[D^2])}, \tag{26}$$

$$P_\sigma = \frac{3\mathbf{E}[D^2]^2 - 2\mathbf{E}[D]\mathbf{E}[D^3]}{6(2\mathbf{E}[D]^2 - \mathbf{E}[D^2])}, \tag{27}$$

and $\sigma_1 = 1/\mu_1$ and $\sigma_2 = 1/\mu_2$ are the (positive) solutions of $x^2 - S_\sigma\, x + P_\sigma = 0$. Namely,

$$\sigma_{1,2} = 1/2 \times \left(S_\sigma \pm \sqrt{(S_\sigma)^2 - 4P_\sigma}\right).$$

We now can compute p_1 and p_2 as given in Equations (23). It is then possible to calculate $\rho_l = p_l\lambda/\mu_l$ and $\gamma_l = \exp(-\mu_l S)$ for $l = 1, 2$. Last $\rho = \rho_1 + \rho_2$.

C. REFERENCES

[1] K. C. Almeroth and M. H. Ammar. *MListen*, 1995. http://www.cc.gatech.edu/computing/Telecomm/mbone/.

[2] S. Alouf, E. Altman, and P. Nain. Optimal on-line estimation of the size of a dynamic multicast group. In *Proc. of IEEE Infocom '02, New York, New York*, volume 2, pages 1109–1118, June 2002.

[3] J.-C. Bolot, T. Turletti, and I. Wakeman. Scalable feedback control for multicast video distribution in the Internet. In *Proc. of ACM SIGCOMM '94, London, England*, pages 58–67, September 1994.

[4] Y. Chu, S. Rao, and H. Zhang. A case for end system multicast. In *Proc. of ACM SIGMETRICS '00, Santa Clara, California*, pages 1–12, June 2000.

[5] D. R. Cox and V. Isham. *Point Processes*. Chapman and Hall, New York, 1980.

[6] C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen. Deployment issues for the IP multicast service and architecture. *IEEE Network magazine, special issue on Multicasting*, 14(1):78–88, January/February 2000.

[7] A. Dutta, H. Schulzrinne, and Y. Yemini. MarconiNet - an architecture for Internet radio and TV networks. In *Proc. of NOSSDAV '99, Basking Ridge, New Jersey*, June 1999.

[8] S. Floyd, V. Jacobson, S. McCanne, C. Liu, and L. Zhang. A reliable multicast framework for light-weight sessions and application level framing. In *Proc. of ACM SIGCOMM '95, Cambridge, Massachusetts*, pages 342–356, August 1995.

[9] P. Francis. Yoid: Extending the internet multicast architecture. Unrefereed report, April 2000. available at http://www.icir.org/yoid/docs/index.html.

[10] T. Friedman and D. Towsley. Multicast session membership size estimation. In *Proc. of IEEE Infocom '99, New York, New York*, volume 2, pages 965–972, March 1999.

[11] S. Haykin. *Modern Filters*. Macmillan, New York, 1989.

[12] S. Haykin. *Adaptive Filter Theory*. Prentice Hall, 3rd edition, 1996.

[13] H. W. Holbrook and D. R. Cheriton. IP multicast channels: EXPRESS support for large-scale single-source applications. In *Proc. of ACM SIGCOMM '99, Cambridge, Massachusetts*, pages 65–78, September 1999.

[14] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, and J. O'Toole. Overcast: Reliable multicasting with an overlay network. In *Proc. of USENIX OSDI '00, San Diego, California*, pages 197–212, October 2000.

[15] L. Kleinrock. *Queueing Systems: Theory*, volume 1. John Wiley and Sons, 1975.

[16] C. Liu and J. Nonnenmacher. Broadcast audience estimation. In *Proc. of IEEE Infocom '00, Tel Aviv, Israel*, volume 2, pages 952–960, March 2000.

[17] J. Nonnenmacher. *Reliable multicast transport to large groups*. PhD thesis, Ecole Polytechnique Federale de Lausanne, Switzerland, July 1998.

[18] J. Nonnenmacher and E. Biersack. Scalable feedback for large groups. *IEEE/ACM Trans. on Networking*, 7(3):375–386, June 1999.

[19] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel. ALMI: An application level multicast infrastructure. In *Proc. of USENIX USITS '01, San Francisco, California*, pages 49–60, March 2001.

[20] J. Rosenberg and H. Schulzrinne. Timer reconsideration for enhanced RTP scalability. In *Proc. of IEEE Infocom '98, San Francisco, California*, volume 1, pages 233–241, March/April 1998.

[21] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: a transport protocol for real-time applications. RFC 1889, Network Working Group, January 1996.

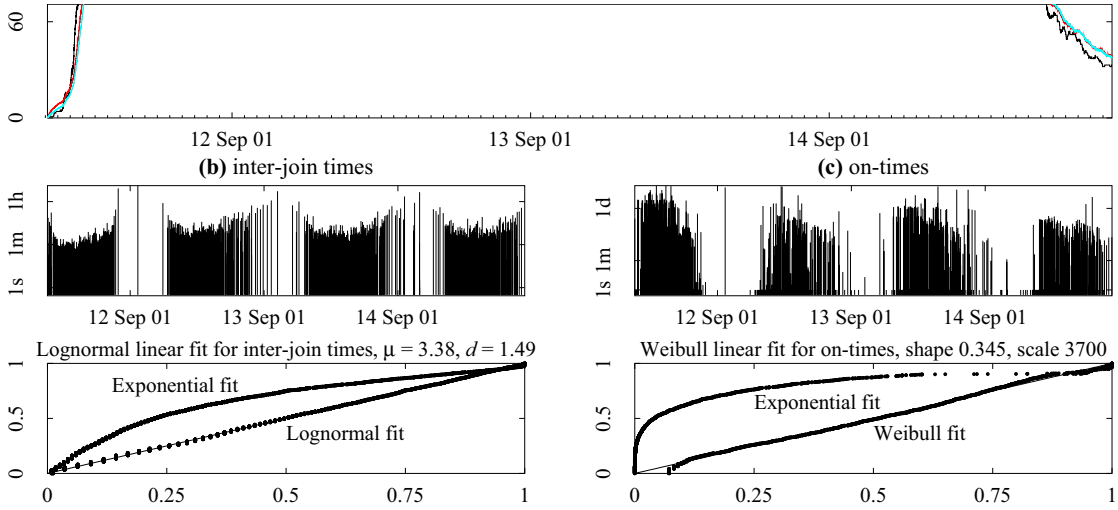

Figure 1: Membership estimation of session $video_1$ and corresponding probability plots

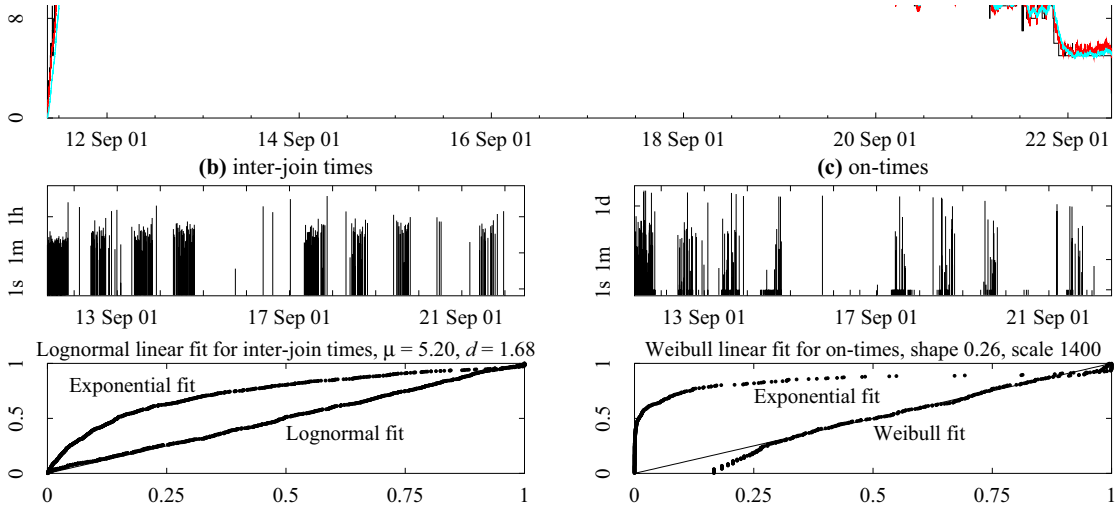

Figure 2: Membership estimation of session $video_2$ and corresponding probability plots

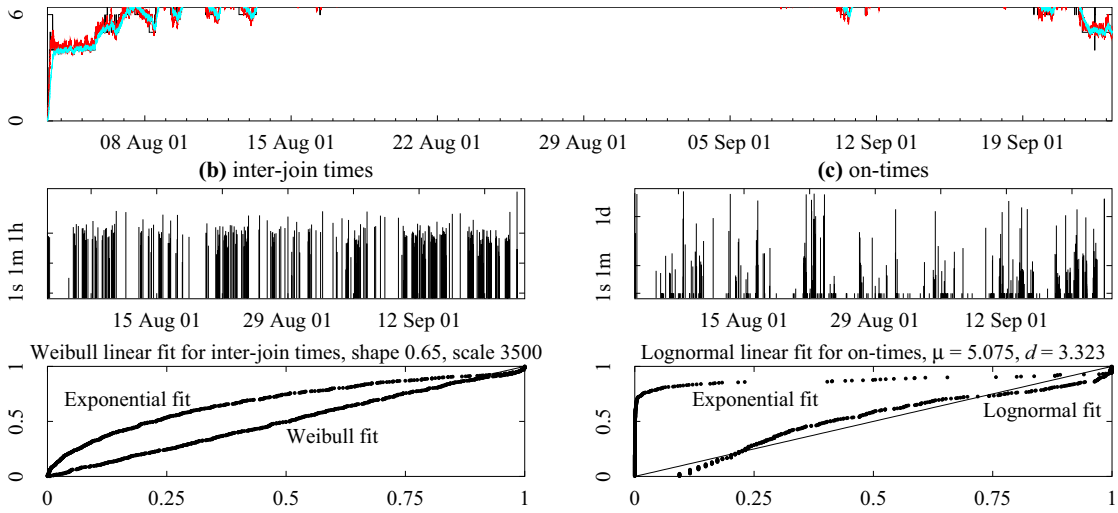

Figure 3: Membership estimation of session $video_3$ and corresponding probability plots

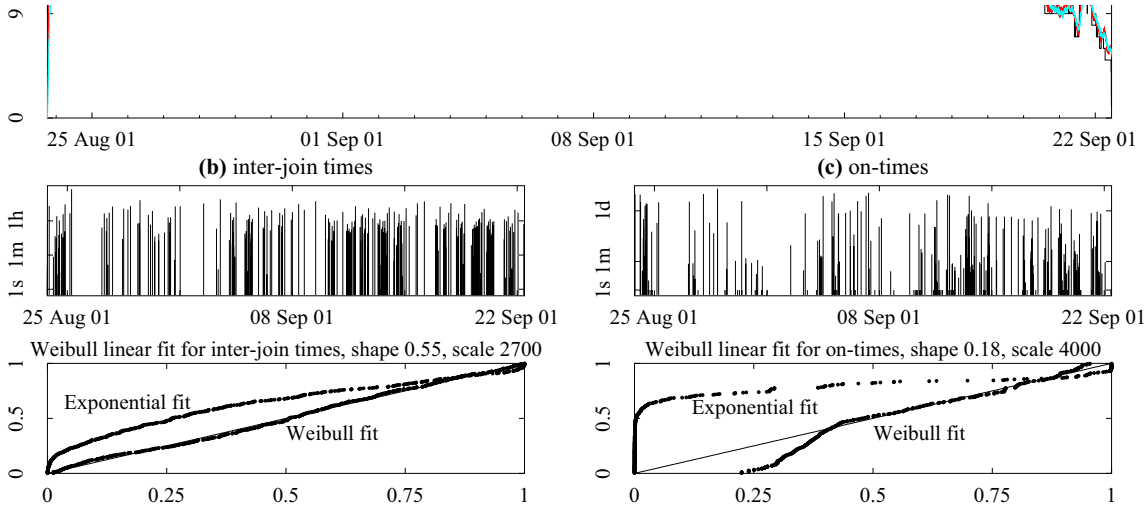

Figure 4: Membership estimation of session $video_4$ and corresponding probability plots

Annexe D

Experiences on enhancing data collection in large networks

Karim Sbai, Chadi Barakat

INRIA Tech Report inria-00324121, to appear in Computer Networks.

Extended version of two papers : one in Annals of Telecommunications, vol. 61, no. 1-2, pp. 167-192, January-February 2006, and the other in proceedings of NTMS 2007 (Conference on New Technologies, Mobility and Security), Paris, May 2007.

Experiences on enhancing data collection in large networks

Mohamed Karim Sbai, Chadi Barakat

Project-Team Planète, INRIA Sophia-Antipolis méditerranée, France

Email: Mohamed_Karim.Sbai@sophia.inria.fr, Chadi.Barakat@sophia.inria.fr

Abstract

We improve and validate TICP [3], our TCP-friendly reliable transport protocol to collect information from a large number of Internet entities. A collector machine sends probes to a set of information sources that reply by sending back their reports. TICP adapts the sending rate of probes in a way similar to TCP for the purpose of avoiding network congestion and keeps requesting reports until they are well received. In a first part of this work, we add to TICP a mechanism to cluster information sources in order to smooth the variation of network conditions during the collection session and to ensure an efficient handling of congestion at network bottlenecks. Simulations in ns-2 and PlanetLab experiments prove the outperformance of TICP over non adaptive solutions and the interest of the clustering mechanism in shortening the duration of the collection session and in decreasing the ratio of lost packets. In a second part, we adapt TICP to collect several packets of information from each data source. By the means of simulations, we compare the performance obtained by TICP to that obtained when using parallel short-lived TCP connections. Our main observation is that TICP yields shorter collection sessions due to its inherent multiplexing capability and that it avoids parallel Slow Start phases. Finally, in a last part, we study the impact of delegating collection to some proxy collectors. We explain our method for delegation and we show by simulations that for a judicious choice of proxy collectors, one can decrease considerably the collection session duration.

Key words: Transport protocol, information collection, clustering, several packets, short-lived TCP connections, delegation

⋆ This paper summarizes our contributions on the TICP protocol. It relies on our two previous publications [3] and [4] and goes beyond by presenting experimental results over PlanetLab and two new and important components, that are the extension to the multiple packets per source case and the delegation of the collector function to intermediate proxies.

1 Introduction

TICP [3] stands for TCP-friendly Information Collection Protocol. It is a TCP-friendly reliable transport protocol designed to collect information from a large number of sources distributed throughout the Internet. TICP is a general purpose end-to-end transport protocol that does not impose any constraint on the type of the collected data and that does not require network collaboration. This generality widens the spectrum of application of the protocol. TICP can be used in all scenarios where entire data collection is needed. The data to be collected can be the availability of network entities, statistics on hosts, routers and network paths, quality of reception in a multicast session, numbering of population, weather monitoring, vote results, etc. It is of particular interest in the area of network monitoring and topology inference (e.g., [5]) where the number of network entities involved is large (hundreds of thousands of machines and routers) and where it is preferable to end up the measurement in a relatively short time without incurring additional load on the network. TICP is an appropriate congestion and error control protocol to be used in this area for the purpose of adapting the rate of probe packets as the pings and the traceroutes.

In TICP, a collector machine sends probes to information sources, which reply by sending back report packets containing their information. Some difficulties come into play when designing a data collection protocol like TICP:

- There is a risk of network congestion due to bandwidth limitation and the large number of sources. Furthermore, all sources are not behind the same bottleneck which makes the congestion control more difficult.
- The collection traffic can be aggressive towards traffic generated by other applications. In particular, attention should be paid to prevent the collection traffic from penalising the concurrent TCP traffic.
- The loss of probes or reports lengthens the duration of the collection session, which urges for an efficient retransmission scheme.

TICP does not only adapt the probing rate as a function of network conditions, but also tries to minimize the collection session duration by implementing an efficient retransmission strategy. Moreover, TICP shares network resources fairly with concurrent traffic, namely TCP traffic, by adapting its probing rate in a similar way to TCP.

TICP was first introduced in [3]. It answers a need for a reliable transport solution to collect information in large IP networks. Current solutions used in network monitoring and management as in Skitter [17] or in the SNMP protocol [15] implement simple periodic probing at a low rate in order not the overload the network. For example in SNMP and to collect average traffic

statistics over 5 minute intervals, routers are probed periodically by no more than once every 5 minutes. For short intervals and large number of routers the overhead on the network can become significant. Other solutions like Concast [14] rely on routers to aggregate the collected information and so do not work in the current Internet.

The present work enhances the protocol with three additional mechanisms to help it achieve better performances and adapt to more situations. This is in addition to a deep analysis of the protocol by the means of extensive simulations and real experiments over the PlanetLab platform [9]. The first mechanism has been introduced in [4]. The second and third mechanisms are novel. The implementation of the code in C++ to run over UDP sockets in the PlanetLab platform is also one of our new contributions in this area.

The collector in the former version of TICP [3] probes information sources in a random order. We start by showing that this strategy causes many problems when moving to large networks, which results in long collection sessions, high loss ratios and out of control traffic. The reason is that only one control at the TICP collector is used to limit the traffic at several bottlenecks simultaneously, which is clearly suboptimal given that the congestion of one bottleneck router can be hidden by the low utilization of another bottleneck router and vice versa. To probe sources behind the same bottleneck together and separately from other bottlenecks, we add to TICP a mechanism to gather information sources into clusters. This mechanism is based on the modeling of the Internet by a two-dimensional Euclidean space and its decomposition into clusters. We use to this end the Global Network Positioning system (GNP) [6] that provides Internet host coordinates. Our new mechanism makes it possible to probe sources from the closest cluster to the collector in terms of RTT (Round Trip Time) to the farthest one. This very likely results in sources behind the same bottleneck probed together before the collector moves to neighboring sources located behind another bottleneck. It is supposed that this behaviour improves the efficiency of the congestion control and ensures a smooth variation of its variables in TICP.

To evaluate the performances of the protocol thus obtained, we run simulations with the ns-2 simulator [7] over realistic and complex network topologies. These simulations show that TICP with the new mechanism of clustering has better performances than without clustering, and that it outperforms other non adaptive data collection solutions. To generalize this result, we implement the protocol in C++ in linux and run real experiments on the PlanetLab testbed. This practical experience tells us that TICP can be easily deployed in the Internet and that the outperformance of the protocol observed in simulations is real.

Another challenge in developing a protocol like TICP is how to extend it to

the case of several packets of information to be collected from each source. This information can be present at the moment the collection is initiated or can be generated later by sources. The old version of TICP [3] ensures collecting only one packet from each information source. In this paper, we explain how we have extended TICP to support collecting several packets per source and we evaluate this extension. The hard issue in this extension is how to handle acknowledgments and sequence numbers of packets and how to enable parallel collection so that to fully utilize the available bandwidth. We implement our extension to TICP into ns-2 and we compare its performance to that of a collection application based upon parallel TCP connections. When the number of packets to be collected from each data source is not important, TCP connections will be short-lived. Simulation results show that TICP has shorter collection session than parallel short-lived TCP connections and this is for any number of parallel TCP connections used. One can explain this result by the fact that TICP multiplexes better the packets coming from all sources. Furthermore, three-way handshake is a problem in short- lived TCP connections and having several Slow Start phases in parallel is sub-optimal. TICP has only one slow start phase for all sources.

We believe that deploying TICP to collect very large amounts of data will not bring much compared to the use of parallel TCP connections. TICP was designed to collect one to several packets from each source. But one can note that even in the case of long-lived TCP connections, TICP does not require adjusting any number of connections since it has only one connection to all sources. Our simulation results will confirm this observation.

Afterwards, we move into studying the impact of delegating collection to a set of sources, called proxy collectors, on TICP performance. A proxy collector plays the role of a TICP collector from the viewpoint of sources on its charge, and plays the role of a regular information source (with several packets of data) when seen by the main collector. Thus, there is a global TICP session connecting the main collector to proxy collectors and local TICP sessions connecting the proxy collectors to sources on their charges. The goal of this delegation is to minimize the duration of the collection session by replicating the collector probing functionality inside the network. This paper explains the method used to choose the proxy collectors as well as the changes made to TICP to support this mechanism of delegation. With the help of ns-2, we vary the number of proxies in different scenarios and we measure the duration of the collection session. As expected, results show that delegation improves performances. However, we observe the existence of an optimal number of delegated proxies otherwise we get a negligible gain compared to the no delegation case.

The paper is organized as follows. In Section 2, we describe the main functionalities of TICP [3]. We explain in Section 3 our approach to cluster information sources and we discuss therein simulation and experimental results. Section

4 shows how we extend TICP to support the collection of several packets of information from each source together with the comparison by simulations to the case of parallel TCP connections. In the fifth section, we present the changes made to TICP to ensure collection delegation and we discuss the choice of proxy collectors. Section 6 overviews the related work and Section 7 ends the paper with some conclusions and perspectives on our future research in this area.

2 TCP-friendly Information Collection Protocol

TICP [3] is a reliable transport information collection protocol implementing diverse functionalities. We overview in this section those related to error recovery and network congestion control.

2.1 Error recovery

The TICP collector has a list of all information sources. The way to obtain this list is out of the scope of TICP. Every source is distinguished by an identifier that can be for example its IP address. Sources whose reports are lost are probed again and are requested to retransmit their reports until they are correctly received by the collector. To make the retransmission of reports in TICP efficient, the collection session is made as a succession of rounds. In the first round, the collector sends request (probe) packets to all sources following their ranking in the list. In a second round, the collector sends requests to sources whose reports were not received in the previous round. The collector continues in rounds until it receives all reports. This behavior in rounds is meant to wait for transitory network congestion to disappear from one round to another and to absorb the excessive delay that some reports may experience.

2.2 Congestion control

To control the rate of requests and reports across the network, TICP implements a report-clocked window based congestion control similar to the TCP one [2]. The collector maintains one variable *cwnd* indicating the congestion window size in number of requests or reports. *cwnd* is then the maximum number of requests that the collector can send without receiving any report packet. New requests are only transmitted when the number of expected reports *pipe* is less than *cwnd*. TICP adapts *cwnd* to the observed loss rate of

reports. It proposes two algorithms to achieve this objective: Slow start and Congestion Avoidance.

2.2.1 Slow start

The collector starts a collection session by setting *cwnd* to *RS* (Request Size, a protocol parameter) and sending *RS* request packets. [1] After some time, reports start to arrive. Some of these reports come on time, others are delayed. A timely report indicates that the network is not congested and that the collector can continue increasing its congestion window: $cwnd = cwnd + 1$. This yields a doubling of the probing rate for every window size of probes. The window continues growing in this way until the network becomes congested. At this point, the collector divides its congestion window by two and enters the congestion avoidance phase. The protocol comes back to slow start whenever a severe congestion appears (to be defined later).

2.2.2 Congestion avoidance

This represents the steady state phase of TICP. During this phase, the collector increases slowly *cwnd* to probe the network for more capacity. We aim a linear increase of the congestion window by *RS* probes every window size of probes. Thus, upon each timely report, the congestion window is increased by: $cwnd = cwnd + \frac{RS}{cwnd}$. When congestion is detected, *cwnd* is divided by two and a new congestion avoidance phase is started.

2.3 Congestion detection mechanism

TICP implements a congestion detection mechanism to compute report loss rates and to decide whether a report is on time, delayed or lost. This mechanism is based upon a timer *TO* scheduled at the beginning of the session and rescheduled every time it expires.

2.3.1 Round-trip time estimator

TICP sets the timer of the mechanism to a conservative estimate of RTT (Round Trip Time), using the samples of RTT seen so far. The value of the

[1] *RS* (Request Size) is the initial number of requests sent without receiving any report packet. It is also the step by which the window size increases. Compared to TCP, this can be seen as the packet size in bytes.

timer is computed using estimates of the average RTT and of its mean deviation. Let *srtt* and *rttvar* be the estimates of the average and the mean deviation of the RTT. Let *rtt* be the measured round-trip time when a report arrives. The collector updates the estimates and the timer (TO) in the following way :

$$rttvar = \frac{3}{4}.rttvar + \frac{1}{4}.|srtt - rtt|$$
$$srtt = \frac{7}{8}.srtt + \frac{1}{8}.rtt$$
$$TO = srtt + 4.rttvar$$

This dynamics and the coefficients it involves are inspired from TCP [8]. The weights for the estimation of *srtt* and *rttvar* are known in TCP to be easy to calculate and to provide good balance between convergence and precision. Setting TO to *srtt* plus four times the *rttvar* has shown its efficiency in setting an upper bound on the Round-Trip Time in TCP. Since our estimator has almost the same target as in TCP, we adopt the same strategy.

2.3.2 Detecting network congestion

TICP computes the report loss ratio over a time window in the past equal to TO. When the timer is scheduled, the collector saves in the variable *torecv* the number of reports to be received before the expiration of the timer. Let *recv* be the number of timely reports received between the scheduling of the timer and its expiration. The collector considers then that $torecv - recv$ reports were lost in the network. Consequently, it estimates the loss ratio to $1 - \frac{recv}{torecv}$.

The network is considered congested if the loss ratio exceeds the Congestion Threshold (CT) and severely congested if the loss ratio exceeds a higher threshold $SCT > CT$ called the Severe Congestion Threshold. CT and SCT are two parameters of our protocol. They can be set to some default values, for example, to 10% for CT and to 90% for SCT. We set them as follows:

$$CT = min(0.1, \frac{RS}{cwnd})$$
$$SCT = max(0.9, \frac{cwnd - RS}{cwnd})$$

The minimum and maximum functions in the expressions of CT and SCT are necessary to ensure that these thresholds do not take unrealistic values when the congestion window is of small size (close to RS). One can use other default values than 0.1 and 0.9. Set as above, CT is equal to $\frac{RS}{cwnd}$ for large congestion windows, which means that congestion is concluded when more than RS reports are not received (resp. severe congestion is concluded when less than RS reports are received in a window). We recall that a report is not received if it is lost or delayed, or if the corresponding request itself is lost or delayed. This way we are compliant with TCP, which considers that the network is congested if at least one packet is lost, and severely congested

when all packets are lost or delayed (i.e., they arrive after the expiration of the timer). Note that we are designing TICP so that a packet in TCP is comparable to *RS* requests/probes in our context. A TICP session behaves like this as *RS* TCP connections running in the same environnement, a parameter that the administrator can tune to whatever desired value.

2.3.3 Delayed and timely reports

A timely report is a report received before its deadline. The deadline of a report is given by the timer. A report not received before its deadline is assumed to be lost. If it arrives later than its deadline, it is considered to be delayed.

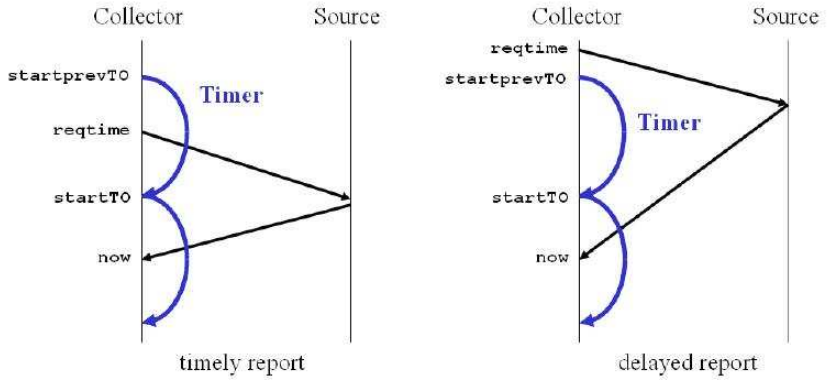

Fig. 1. The two types of reports

Fig. 1 explains how the deadline of a report is set. Let *startTO* be the scheduling time of the timer. Let *startprevTO* be the previous scheduling time of the timer. When a report is received, the collector extracts from its header the timestamp *reqtime* indicating the time by which the corresponding probe has been sent. The report is received on time if and only if $startprevTO < reqtime$. The report is a delayed one in the opposite case.

3 Clustering of information sources

The congestion control mechanism of TICP described above relies on RTT estimation for the calculation of the report loss ratio and for the setting of deadlines to retransmit lost reports. In case sources are probed randomly and independently of their locations, this estimation is errored because of the high variability in the RTT process and its low auto-correlation. The loss ratio and report deadlines are then badly calculated, which impairs the performance of the collection. In this section, we study the impact of clustering

information sources on the performance of TICP. First, we detail the reasons that motivated the addition of the clustering mechanism. Then, we describe briefly the GNP Internet coordinate system[6] that provides the knowledge of sources locations and we explain our approach to cluster information sources. Finally, we validate TICP with clustering by the means of simulations and real experiments.

3.1 Need for clustering of information sources

Does a random ordering of sources yield a good estimation of RTT for the next pairs of request/report? The accuracy of this estimation is essential for the correct functioning of the protocol. An overestimation of RTT results in a delay in the detection of network congestion; the collector waits more than necessary for already lost reports. This delay means a waste of time and an aggravation of network congestion since the probing rate will not be reduced on time. On the other hand, an underestimation of RTT can cause errors in the computation of the report loss ratio due to the premature expiration of the timer. Thus, some reports can be declared lost while they are not. If it is the case, TICP will reduce unnecessarily the size of the congestion window (*cwnd*) which will lengthen unnecessarily the collection session.

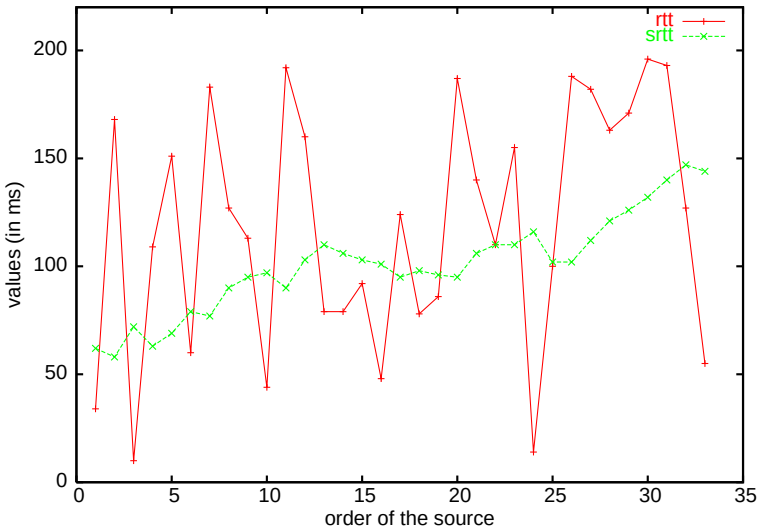

Fig. 2. Estimating RTT with random ordering of sources

To clarify this point, we plot in Fig. 2 a sequence of *rtt* and *srtt* for sources picked randomly from the list of sources without any consideration of topology. The x-axis shows the probing order. Results are obtained from the simulations whose setting is to be described later. The figure illustrates how the real value of the RTT oscillates; it can exceed the estimated average by large values and go much below it from time to time. This certainly implicates a regression in

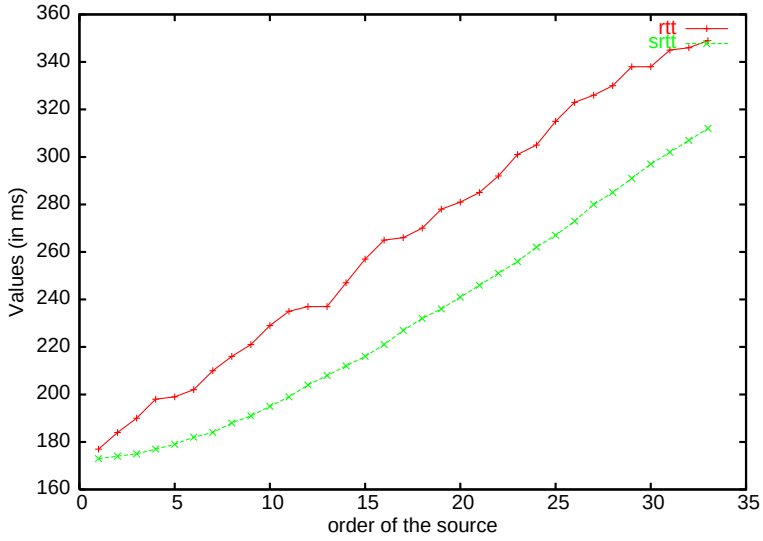

Fig. 3. Estimating RTT with latency-based ordering of sources

the performances of TICP. Our idea is to cluster information sources so as to probe them from the nearest to the collector to the farthest one. This way, the RTT in the network varies smoothly and can be estimated more accurately. Fig. 3 supports this claim. It plots *rtt* and *srtt* when sources are ordered based on their latency to the collector. We can clearly observe how the RTT and the *srtt* vary smoothly with a smooth shift that can be easily compensated by the mean deviation *rttvar*.

There is another gain from clustering. If the collector probes the sources in a random order, the request packets and the reports will circulate everywhere in the network. Hence, the collection traffic will participate to the congestion of several bottlenecks at the same time. But the collector cannot with one control adapt the sending rate of probes to network conditions on several paths in parallel. In case of random ordering, the Internet is viewed by TICP as a sole bottleneck which is far from reality. TICP should treat the bottlenecks one by one. The gathering of information sources into clusters and their probing from the nearest to the collector to the farthest ensures this decoupling of network bottlenecks and adapts correctly the probing rate to these bottlenecks one after the other instead of having a joint adaptation.

3.2 Global Network Positioning System (GNP)

To support the clustering mechanism, we need the knowledge of hosts' locations. Global Network Positioning (GNP) [6] System is a promising and accurate solution to this problem. It models the Internet as an Euclidean space S of dimension n. This space has a well-defined coordinate base and a distance function. An internet host is represented in this space by a point such that

the Round-Trip Time between two hosts can be estimated by computing the distance between their coordinates. To assign coordinates to hosts, GNP first computes the coordinates of a small set of well-known hosts called *landmarks* based on the measurements of delay between them. The coordinates of landmarks are then disseminated to any ordinary host willing to compute its own coordinates. An ordinary host derives its own coordinates based on landmarks coordinates and the measurements of delay between itself and the landmarks. We suppose these coordinates are available to the TICP collector in the way the IP addresses of hosts and their names are. Finally, coordinates are meant to form a part of the ID of a machine in the Internet helping to localize it. The way to get this information is out of scope of TICP. One can do it using an additional channel as the DNS or other at a frequency much lower than the frequency of collecting information.

3.3 The clustering approach

A cluster is a set of hosts located in the same neighbourhood such that the latency to reach a host in the same cluster is generally shorter than the latency to reach a host not belonging to it. If the cluster is well chosen, its hosts very likely meet the same network conditions when they communicate with the collector and their traffic cross the same network bottleneck. This is the basic idea behind our solution to remove (even partially) the bias caused by random probing in TICP. To locate information sources in the Internet, we use GNP as described above. We model the Internet as a two-dimensional Euclidean space. The collector and sources participate to GNP as ordinary hosts in addition to some well-defined landmarks. An information source S has a couple of coordinates (x_S, y_S). The collector's coordinates are (x_{co}, y_{co}).

We define a cluster as being a set of information sources whose representing GNP points are located in a square area. The side length of the square is denoted by a, which is a parameter of the protocol. The central cluster is the square whose centre is the point representing the collector.

A cluster is completely defined by a couple of coordinates (X,Y) being integer values. These coordinates are those of the centre of the corresponding square relatively to the collector coordinates and then normalized by a. An information source S whose coordinates are $S(x_S, y_S)$ belongs to the cluster (X,Y) given by:

- $X = round(\frac{x_S - x_{co}}{a})$
- $Y = round(\frac{y_S - y_{co}}{a})$

TICP probes information sources from the nearest to the collector to the farthest one. This is supposed to smooth traffic transitions between network

bottlenecks while maximizing the volume of collected information at the beginning of the session. The collector begins with the central cluster and traverses the other clusters following a spiral trajectory. Fig. 4 illustrates this trajectory. One can with a simple procedure, find the coordinates of the next cluster during the collection knowing the coordinates of the current cluster.

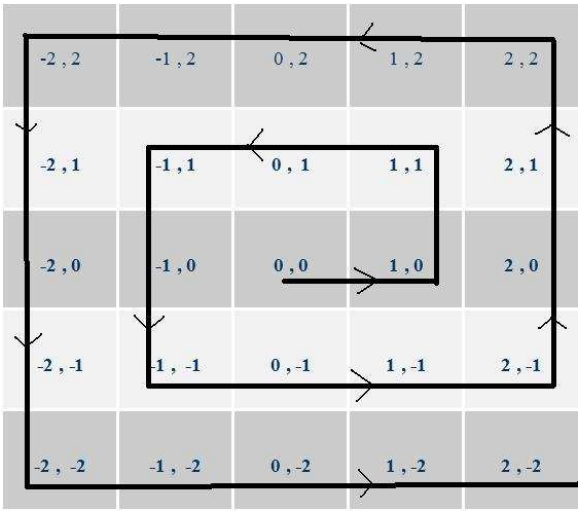

Fig. 4. Order of probing information sources

3.4 Performance evaluation

To assess the impact of clustering sources on the performance of TICP, we run simulations in ns-2 [7] over realistic network topologies as well as real experiments on the PlanetLab testbed using our C++ implementation of the protocol. We are particularly interested in the evaluation of the collection session duration and in the understanding of the impact of the different protocol parameters.

3.4.1 Simulations results

We discuss here the results of our simulations. These simulations are run in ns-2 [7] after the implementation of both TICP and GNP. We generate realistic network topologies for simulations using the GT-ITM tool (Georgia Tech-Internet Topology Modeling) [11]. We choose to work on transit-stub (TS) topologies due to their ability to capture the complexity and the hierarchical structure of the real Internet. TS topologies model the Internet using a 2-level hierarchy of routing domains with transit domains interconnecting lower level stub domains. We assign latencies of 35ms for intra-transit domain links, 10 ms for stub-transit links and 5ms for intra-stub domain links. Fig. 5 shows an example of a TS topology. The size of each topology realization is depicted by

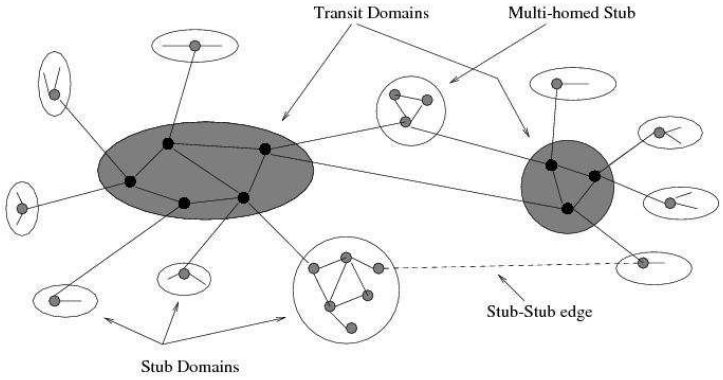

Fig. 5. Transit-stub network topology

the parameter values in Table 1. In each simulation we generate a TS topology and we choose randomly 500 sources of information and a collector among the nodes composing the topology. The parameters of TICP are set as in Table 2. In particular, probe packets are of size 100 bytes and reports can fit in packets of 1500 bytes. With these values, congestion appears on the return path from sources to collector. As for the cluster size a, we set its value to 50 ms as our simulations indicate that this value leads to the best results over our TS topologies. Later in the paper, we change the value of a and we study how this impacts the collection session. Routing tables are calculated at the beginning of simulations to provide shortest paths in terms of number of hops. These tables are used to carry probes and reports in unicast between the collector and the information sources. In contrary to [3] and to make our evaluation of TICP more realistic, multicast routing is disabled in routers.

Table 1
Transit-stub topology parameters

Parameter	Signification	Scenario
T	Number of transit domains	5
Nt	Average Number of nodes / transit domain	7
K	Number of stub domains / transit node	8
Ns	Average number of nodes / stub domain	7

We compare between the both versions of TICP with and without clustering of information sources. The comparison criterion is the collection session duration. Fig. 6 shows this duration for several simulations of TICP without clustering. In each simulation, the order of sources in the list of the collector is different, that is why we obtain each time a different duration for the collection session. For TICP with clustering, the result is the same since the topology does not change, and so the ranking of the sources is the same defined by their coordinates. TICP with clustering finds the good ranking of information sources and guarantees the shortest collection session duration. We save on average 30% of time by moving from TICP without clustering to TICP with

Table 2
TICP parameters

Parameter	Value
RS	10
probe size	100 B
report size	1500 B
a	50 ms

clustering.

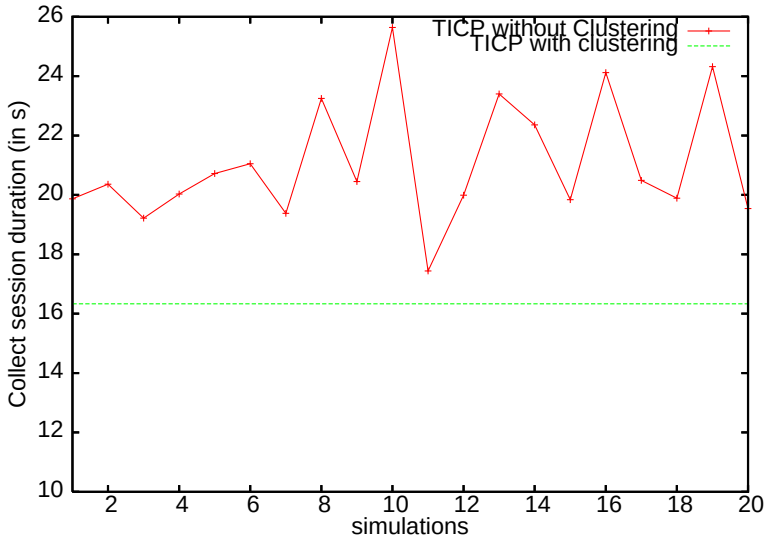

Fig. 6. Collection session duration for different ordering of sources

To evaluate the optimality of TICP, we implement in ns-2 an information collection protocol having a constant window size. For each window size, we run 10 simulations and we record the minimum of the collection session duration over them in order to identify the best combination. Figure 7 plots three curves in the same space. The x-axis represents cwnd, the size of the congestion window for the constant congestion window protocol. The y-axis represents the collection session duration (in seconds) for the constant window protocol. This is the curve having a parabolic convex shape. For small congestion window sizes, the collection session is long because we have a low probing rate. For large window sizes, the network is congested which lengthens the collection session. As for the two horizontal lines, they represent the session duration obtained by TICP and are only drawn for comparison. TICP adapts dynamically cwnd to network conditions. The role of TICP is to find in a dichotomic way the right congestion window size that minimizes the collection session duration. This proves the out-performance of an adaptive solution compared to a non adaptive one. We notice in the figure how TICP with clustering manages to reach the optimum unlike TICP without clustering which yields longer durations.

Then, we vary the cluster size and we study its impact on the collection

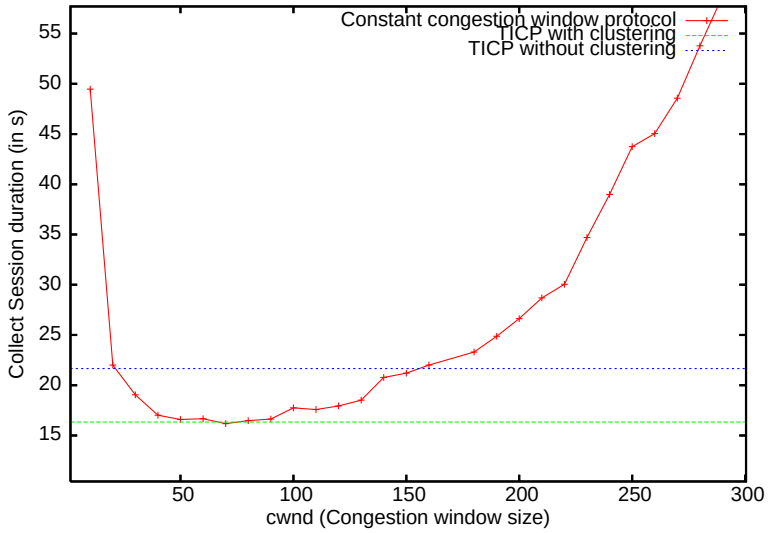

Fig. 7. Optimality of the protocol

session duration. Taking a very large a is equivalent to TICP without clustering since sources will be probed independently of their locations within the large cluster. Taking a very small results in clusters empty or with few number of sources. This is not efficient since there will be no clustering of sources behind common bottlenecks. So, there should be some average a that provides the best performance. Fig. 8 validates this intuition where we can see that over the network topologies we simulate in this work, a value of a around 50ms is optimal. Each point of the curve in the figure is the average over 5 simulation runs on different network topology realizations, all satisfying the characteristics in Table 1. The number of sources is taken equal to 500.

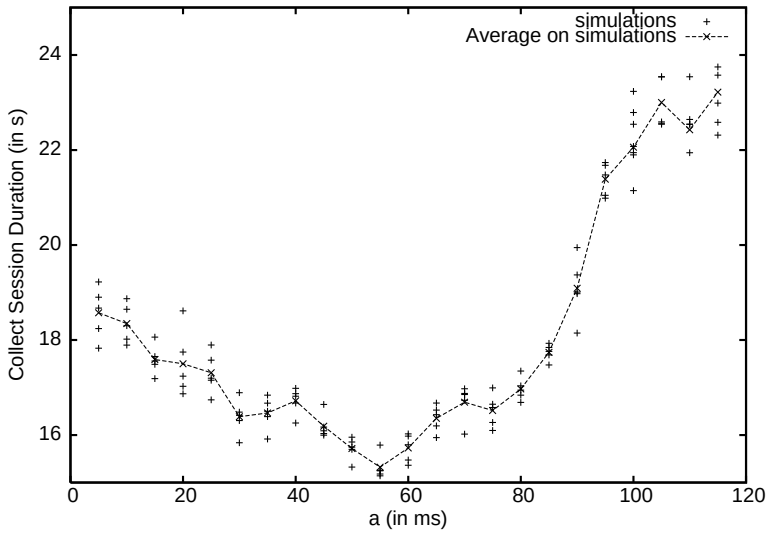

Fig. 8. Impact of a on collection session duration

Fig. 9 studies how the number of sources impacts the choice of optimal a. We can clearly see that the optimal cluster size decreases when the number of

sources increases. Compared to the value used above, the optimal a is equal to 85ms for 300 sources and to 45ms for 700 sources. Indeed, for small number of sources, one needs to increase a to group more sources behind the same bottleneck together. At the opposite, for more sources, one needs to decrease a so that the collector can better probe them depending on their locations. But, if we continue increasing the number of sources, the optimal a will stabilize and become equal to some minimum value dependent of the topology. As mentioned above, a cluster is in somehow the set of sources located behind the same bottleneck. Thus, increasing the number of sources (distributed uniformly in space) does not increase the number of bottlenecks in the network. Once we reach the maximum number of bottlenecks, the optimal size of the cluster will become constant. One can safely use this value for applications collecting data from a very large number of sources. We suggest that in practice, one starts by calculating this value by running multiple collection sessions, then adapts it as a function of the measured session duration to account for any change in the underlying network topology.

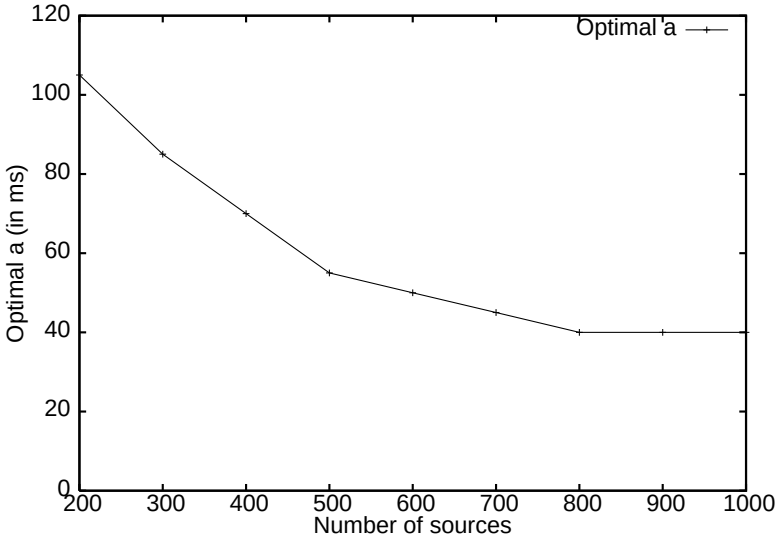

Fig. 9. Optimal a as a function of the number of sources

3.4.2 Experimental results

We implement TICP with clustering in C++ and run real experiments on the PlanetLab testbed [9]. We use one node from those available in our PlanetLab slice as collector and the other nodes as information sources. These experiments on PlanetLab aim to study the efficiency of deploying TICP in the Internet and to validate the simulation results already obtained. We run successfully all scenarios described in our simulations. For lack of space, we include two samples of our results. Fig. 10 plots the collection session duration as a function of the number of sources for the optimal cluster size, a small cluster size (100 ms) and a big cluster size (1000 ms); and Fig. 11 plots the

optimal cluster size as a function of the number of sources.

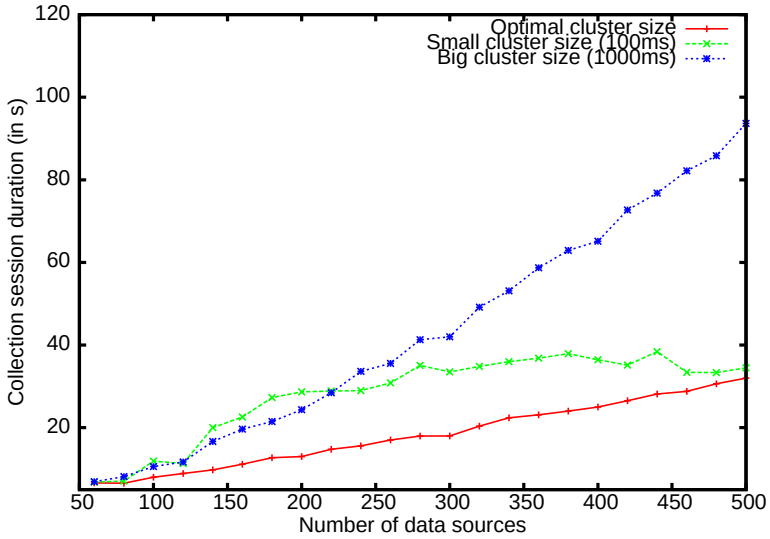

Fig. 10. Collection session duration as a function of number of sources in PlanetLab experiments

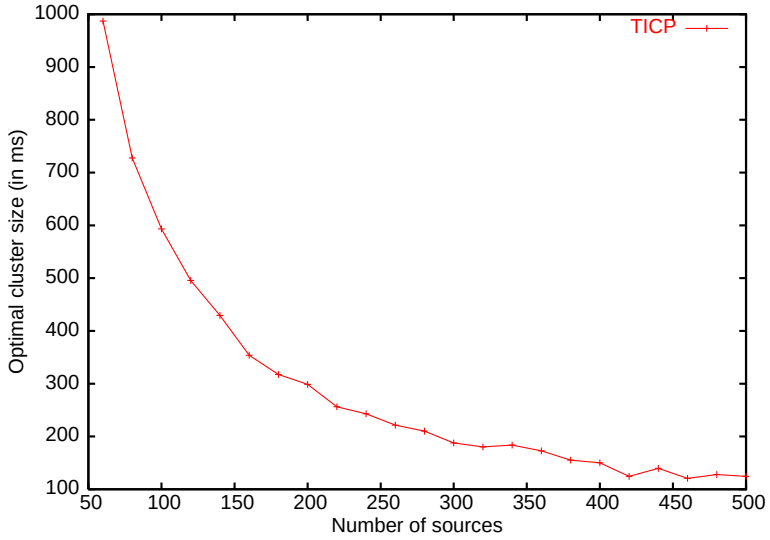

Fig. 11. Optimal cluster size as a function of the number of sources in PlanetLab experiments

Both figures show that experimental results have the same shape compared to simulation ones but clearly different values of the optimums. This is because the real Internet topology is different from the topologies used in our simulations. The main observation we can make from Fig. 10 is that the collection session duration grows almost linearly with the number of sources which means that TICP scales well in practice. Since the large cluster curve corresponds almost to the no clustering case, one can notice that there is an important gain when using TICP with clustering. Moreover, the small cluster curve shows

that having a few number of sources in each cluster is suboptimal. As for Fig. 11, it gives an idea on how to choose the cluster size when collecting information in the real Internet. It seems that clusters of 100ms side length lead to the best performances for a large number of Internet sources.

4 Collecting several packets of data from each source

The first version of TICP [3] supposes that each information source has a small amount of data to send to the collector that can fit in one packet. This is a strong assumption that needs to be relaxed. Some sources might possess more than one packet when the session is initiated, or generate data later during the collection session. In this section, we present our extension to TICP that allows collecting informational reports of several packets. We explain first the new functionalities added to TICP. We then study the scalability of the new version through simulations. Finally, we compare TICP to a collection application based on parallel TCP connections while varying the number of TCP connections.

This extension is far from being an easy task. First, it requires sequence numbers to detect the lost reports from each information source and to ask for retransmissions. Second, it requires parallel collection from each information source when possible to better utilize the network resources. Lastly, it should adapt to the case when information is generated by the sources during the collection session. As to be described later, our extension handles all these issues in an efficient way.

4.1 Behaviour of information sources

An information source subdivides data into packets of equal sizes. Each packet has a unique sequence number *SEQ*. When the collector probes an information source for packet number *N*, the source sends back the data packet having *SEQ* equal to *N*. In general, the role of sources is simple as before and consists in answering the requests for packets sent by the collector.

An information source does not necessarily know the number of packets to send to the collector at the beginning of the session. We want to account for the scenario when data is continuously generated during the collection session. To indicate to the collector that the current packet is the last one, an information source sets a field *MORE* in this packet to 0. When *MORE* is set to 1, the collector understands that the source has more packets to send and that it should keep probing it for more packets.

4.2 Behaviour of the collector

The collector probes sources in the order given by the clustering mechanism. The sending rate of probes respects the congestion window size as before.

In the first round, the collector probes all sources asking them to send their first packet. To make this possible, we add to the request packet a field *RSEQ* indicating the sequence number of the requested packet. In the second round, the collector discovers that some packets requested during the first round have arrived to the collector while others have been delayed or lost. The probing scheme consists in asking a source which has correctly sent her first packet to send its second one and to ask a source whose first packet has been declared lost to retransmit it. As for sources whose first packet has not yet arrived (their deadlines have not expired), an efficient solution is to ask for their second packet, of course if the congestion window allows and if the field *MORE* has not been set to 0 for these sources. By behaving in this way, TICP is able to anticipate the arrival of packets and fill the congestion window which is necessary for an efficient utilization of network resources. This behavior is generalized to the following rounds.

To implement this behavior, we introduce the following structure at the collector that contains all required information about the sources during the collection:

SOURCES_LIST = {*SOURCE*}
SOURCE = *[ORDER, ID_SRC, MORE, REQUESTS, RECEIVED_PACKETS]*
REQUESTS = *[LAST_SENT_REQUEST, REQUESTS_LIST]*
RECEIVED_PACKETS = *[LAST_RECEIVED_PACKET, PACKET_NUMBERS_LIST]*
REQUESTS_LIST = {*REQUEST*}
PACKET_NUMBERS_LIST = {*SEQ*}
REQUEST = *[RSEQ, DEADLINE]*

When it is the turn of an information source, the collector checks the *MORE* field of the object *SOURCE* corresponding to this source. If *MORE* equals 0, then the collector jumps over this source and treats the next source in the list. If all sources have the *MORE* field equal to 0, the collector announces the end of the collection session. In the case when the *MORE* field of a source is still equal to 1, the collector first seeks in the list of sent requests related to this source (*REQUESTS_LIST*) a request packet whose deadline has been reached. We recall that the deadline of a request/data packet is reached when the corresponding request has been transmitted before *startprevTO*, the time at which the previous timer has been scheduled (see Section 2). If such request is found, the collector considers that the corresponding report is lost, cancels this request, retransmits it and jumps to the next source in the list. If

REQUESTS_LIST is empty or the collector has not found any request whose deadline has been reached, it sends a request packet having *RSEQ* equal to *LAST_SENT_REQUEST + 1*. This means that it anticipates requesting the next packet and gives more time to delayed packets to arrive at the collector.

At the beginning of the session, *LAST_SENT_REQUEST* is set to 0. Any newly sent request must be added to *REQUESTS_LIST* and *LAST_SENT_REQUEST* must be incremented. In the case of a retransmitted request, a request packet having the same *RSEQ* is sent and the transmission time is set equal to the current time. The attribute *LAST_SENT_REQUEST* does not change in this case. When the collector receives a data packet, it updates *cwnd* and *pipe* as described earlier (see Section 2), then it deletes the corresponding request from *REQUESTS_LIST*.

4.3 Scalability of the new version of TICP

We want to test the scalability and the efficiency of this new version of TICP when the number of packets per source and the number of sources grow. For this, we run simulations in ns-2 on the same network topology used to validate the clustering approach. The sizes of a request packet and a data packet are respectively taken equal to 100 bytes and 1000 bytes. We fix the number of sources and study the impact of changing the number of packets per source on the performance of TICP. Fig. 12 plots the evolution of the collection session duration as a function of the number of packets per source for three values of the number of sources *N1=50, N2=100, N3=150*. We observe that for a

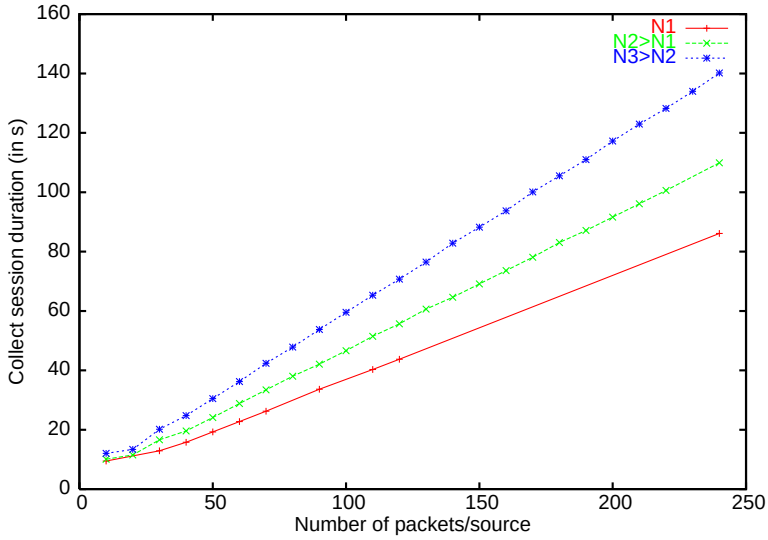

Fig. 12. Impact of the number of packets per source

given number of sources, the collection session duration varies linearly with the number of sources. When the number of sources increases, the slope of the

graph remains constant. This evolution proves the scalability of TICP when the number of packets per source increases. One can interpret this behavior by the fact that the network is seen by TICP as a set of bottlenecks. The collection session duration is the sum of the collection times over the different bottlenecks. We know that the collection time of lets say S bytes of data over a bottleneck having B bps as available bandwidth, is almost linear with S and equal to $\frac{S}{B}$. When adding packets in sources, it is as if we are increasing linearly the number of packets to collect over each bottleneck, which results in this linear behaviour.

4.4 Comparison with a collection application using parallel TCP connections

The intuitive question that one would ask at this level is how TICP compares to a collection application using regular TCP to collect data from sources (one TCP connection per source). As we are in the context of relatively small number of packets from each data source, the TCP connections will be short-lived connections, and hence, TICP will provide better performances for the simple reason that it avoids the three-way handshaking and the slow start phases of the individual TCP connections. Note that for very large amounts of data per source, TICP works as well but its gain compared to the use of TCP is not guaranteed. It is still has the advantage of doing the collection in one session for all sources without the need to establish and schedule a large number of TCP connections.

We compare here TICP to a collection application using parallel TCP connections to gather packets from sources. One TCP connection is opened between the collector and each source to retrieve its information. The collection application using TCP limits the number of parallel connections to lets say m connections. When one connection ends, another connection is established with another source and this continues until all data is retrieved. The main challenge here is to well choose the number of parallel TCP connections that minimizes the collection session duration. Several works have studied the problem of how many parallel TCP connections one needs to open to get the best performances. All agree on that only few connections need to be established in parallel. [1] shows through emulations that for a large number of connections, the network becomes congested and the throughput deteriorates. On the other hand, for m very small, one does not use completely the available bandwidth and so the throughput of reception is not the best. [1] observes that the throughput reaches its maximum for a number of connections between 6 and 8.

Having this in mind, we design the collection application based on parallel TCP as follows. The application begins by opening m connections from the

collector to the first m sources in the list. When a connection finishes retrieving packets from the source to which it connects, another connection is opened with the next available source in the list. The application continues in this way until reaching the last source in the list. We believe this should provide a fair comparison with between TCP and TICP.

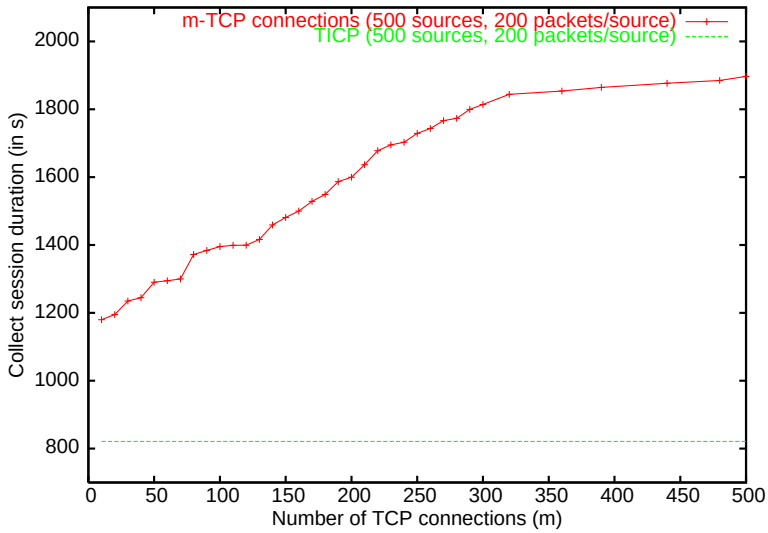

Fig. 13. Parallel TCP connections vs. TICP

We implement this parallel TCP based collection application in ns-2 and we generate topologies similar to those used earlier in this paper. We choose 500 sources and a collector randomly among the nodes of the topology. Each information source is supposed to have 200 packets of 1000 bytes each to send to the collector. First, we run TICP under this configuration. Then, we run the collection application for different values of the number of parallel TCP connections m. Fig. 13 plots two curves in the same space. The first curve is for the TCP based collection application and the second one is for TICP and is present only for comparison. In the figure, the x-axis represents the number of parallel TCP connections (m) for the collection application using TCP. The y-axis represents the collection session duration. As TICP opens one connection to all sources, the duration of its session is independent of the value of m on the x-axis. As expected, we can see that a small number of parallel TCP connections gives the optimal collection session duration. But surprisingly, the figure also shows that the collection session duration for TICP is shorter than the collection session duration for any number of parallel short-lived TCP connections. This proves the efficiency and the importance of TICP for the collection of several packets of data from a large number of sources. We interpret this result by the fact that in case of TCP, at least one slow start phase is needed by each connection, which wastes network bandwidth. In case of TICP, these slow starts are avoided since the collection is done in parallel from all sources with only one congestion window. The other explanation is that TICP collects data in parallel from all sources, which allows an efficient

utilization of network resources. In case of parallel TCP, we are obliged to open connections with a limited number of sources and collect from them entirely before moving to other sources. This is suboptimal since only few bottlenecks are utilized while others are not.

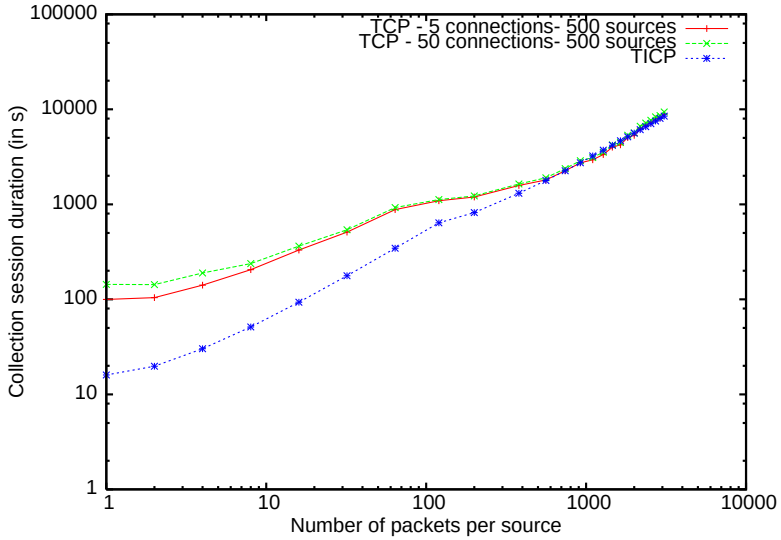

Fig. 14. Impact of the number of packets per source

To study the impact of the number of packets on the performance of both TICP and parallel TCP connections, we do further simulations varying the number of packets per source. For the TCP-based collection, we consider the two cases of 5 and 50 connections. Figure 14 plots on a log-log scale the collection session duration as a function of the number of packets for TICP, TCP-5 connections and TCP-50 connections. As expected, this figure shows the outperformance of TICP when the number of packets to collect is relatively small (from 1 to 500 packets here). We can notice a gain of one order of magnitude due to avoiding the slow starts phases of the individual TCP connections. For large files however, the gain reduces and both TICP and parallel TCP perform equally well. This is because large files are able to bring the individual TCP connections to their stationary regimes, hence reducing the gain introduced by the traffic multiplexing in TICP. For small files this stationary regime is barely reached making the slow start phases dominant and network ressources under utilized if TICP multiplexing across sources is not used.

5 Parallelizing the collection using proxy collectors

Even though TICP has been shown to be efficient, its performances are still limited by the fact that there is only one collector that probes sources distributed worldwide. In this section, we study the impact of delegating information collection to a set of proxy collectors. The goal of this delegation is

to decrease the collection session duration. Proxy collectors are specific well-dimensioned machines responsible of collecting packets from sources in their neighborhood. We assume the more general scenario where proxy collectors are sources of information themselves. The main collector in its turn is responsible of the global collection by gathering packets from proxy collectors. First, we present the method we use to choose the proxy collectors among information sources. Then, we explain how we extend TICP to support delegation. Finally, we discuss the simulation results.

Note that in some cases the delegation of collection can influence the collected data itself. This can be seen for example when part of the collected information is related to the characteristics of the path between the collector and the source of information as the delay and the bandwidth. The delegation in this case should be done with further attention and its impact on the data itself is to be studied carefully before deploying proxies. These specific cases are out of the scope of this work. We only deal here with the network load and we suppose that the collected information is the same whether it is probed from machine A or machine B. Our optimization only aims to reduce TICP traffic and fastens the collection while collecting the entire data. The specific cases where the data to collect is function of the way the collection is done is an important problem to consider but is somehow application specific, so to stay general we leave it for a future research.

5.1 Choice of proxy collectors

The idea is to choose p proxy collectors among the sources in a way to minimize the sum of the latencies from each source to its closet proxy collector. This way we minimize the length of network paths crossed by probes and reports. The choice of proxy collectors is a particular instance of the well-known p-median problem. The p-median problem is defined as follows. Given a set F of l potential facilities, a set U of n users, a distance function d: U x F-> R, and a constant $p \leq l$, determine which p facilities to open so as to minimize the sum of the distances from each user to its closest open facility. This is a well-known NP-hard problem. In our case, the set of information sources represents the facilities as well as the users. The distance function is the Euclidean distance computed using sources' coordinates given by GNP.

To solve this problem, we use the hybrid heuristic proposed in [10]. It is a multi-start iterative method where each iteration consists of a randomized greedy construction of a solution, which is then submitted to local search. A pool of best solutions found in iterations is kept. In each iteration, the solution found with local search is combined with one of elite solutions. After all iterations, there is a post-optimization phase in which elite solutions are

combined with each other. This method gives good solutions in short running time.

5.2 TICP with collection delegation

Three types of actors take part of a TICP session with collection delegation: the main collector, the p proxy collectors and the n-p ordinary information sources. n is the total number of sources. A collection session starts with choosing p information sources as proxy collectors using the above heuristic. Then, the main collector runs a TICP session between itself and the proxy collectors. In its first probe packets to a proxy collector, the main collector sends the identifiers of ordinary sources that are under the responsibility of this proxy collector. Once a proxy collector receives the list of sources for which it is responsible, it updates its data structure to begin immediately a local TICP collection session. During this local session, it plays the role of a regular TICP collector towards ordinary sources in its charge. This behavior is described with details in Section 4.

Now from the point of view of the main collector, a proxy collector behaves like an information source. It considers the packets issued by ordinary sources under its responsibility as its own packets. First, it starts by sending its own packets to the collector. Then, it sends the packets of ordinary sources in the order of their arrivals. It may happen that the main collector asks the proxy collector to send a new packet and that the proxy collector is waiting for new packets to arrive from ordinary sources. In this case, the proxy collector replies with a *persistency* packet instead of the requested packet to tell the main collector that it has nothing to send right now but that more packets will come later. We add this persistency packet in order for the main collector not to consider the absence of a response as an indication of network congestion. Such a persistency packet has the same sequence number as the requested packet but with a *MORE* field set to 2.

We run simulations in ns-2 over network topologies described in Section 3 in order to evaluate the gain obtained when delegating the collection. We vary the number of proxy collectors and we compute for each number, the average duration of the collection session over several simulations. Fig. 15 plots the results for 500 randomly distributed sources, with the x-axis showing the number of proxy collectors. We can clearly observe in the figure an improvement in the performances of TICP when we delegate to the first proxies, then a deterioration in the performances as long as we add more proxies. There is an optimal number of delegated proxies that leads to the shortest session (a gain by two in our settings). Indeed, for a small number of proxy collectors, there is a waste of time and resources since the proxy collectors are in charge of

many sources and so are overwhelmed and not able to answer in real time the probes of the main collector. If we continue increasing the number of proxies, TICP will be able to collect with a high speed from the proxy collectors who themselves are collecting at high speed from the ordinary sources, which explain the best performances. Further increase in the number of proxy collectors lengthens the collection session since the main collector becomes in charge of many proxy collectors which prohibits it from collecting from them at high speed. For the extreme case where the number of proxy collectors is equal to the total number of sources, the proxy collectors have no ordinary sources in their responsibilities, so the collection operation is similar to an ordinary TICP collector gathering data from all sources. Generally speaking, delegating to everyone is equivalent to not delegating at all. The collect session duration for 500 proxy collectors (the point on the very right hand side of the figure) is equal to that for zero proxy collector (the point on the very left hand side of the figure).

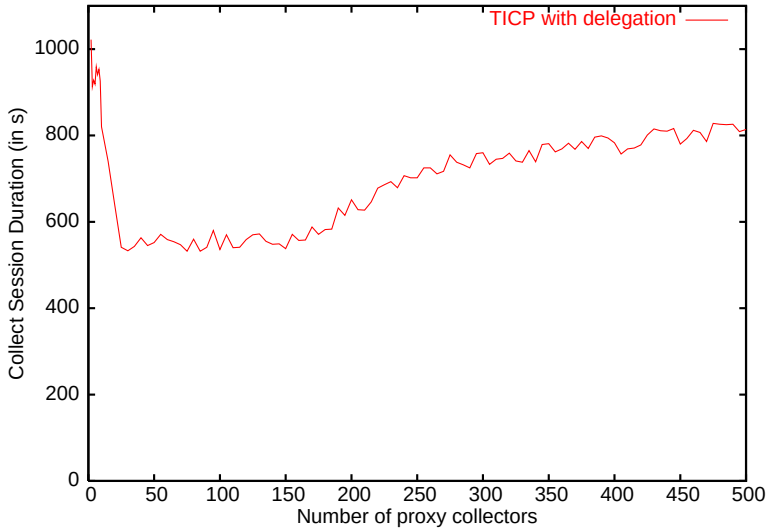

Fig. 15. TICP with delegation for 500 sources

Given the presence of an optimal value for the number of delegated proxies, we want to check how this optimal value varies with the number of ordinary sources. For this, we vary the number of sources and record the number of proxy collectors giving the minimal collection session duration. The results are average over several simulations and are plotted in Fig. 17. We can see that when the number of sources is less than some threshold (140 in our settings), it is better to delegate to all sources (or equivalently not to delegate at all).This observation is illustrated in Fig. 16 where we plot the collection session duration as a function of the number of proxies for 100 sources randomly distributed in the network. As the number of sources is less than the threshold, the performance keeps improving by adding more proxy collectors. In this case, the population is small and one TICP session is able to do a good

job. For a number of sources larger than the threshold, we notice that the optimal number of proxy collectors to consider is almost constant equal to the threshold (140 in our settings). Our interpretation for this behavior is that there is a maximum capacity for the network that can be achieved by some delegation and any further increase of sources beyond this delegation will be equivalent to adding more data per sources and so will not impact the number of delegated proxies. This maximum number of delegated proxies to be used is topology dependent. Our future research will focus on its calculation for different scenarios.

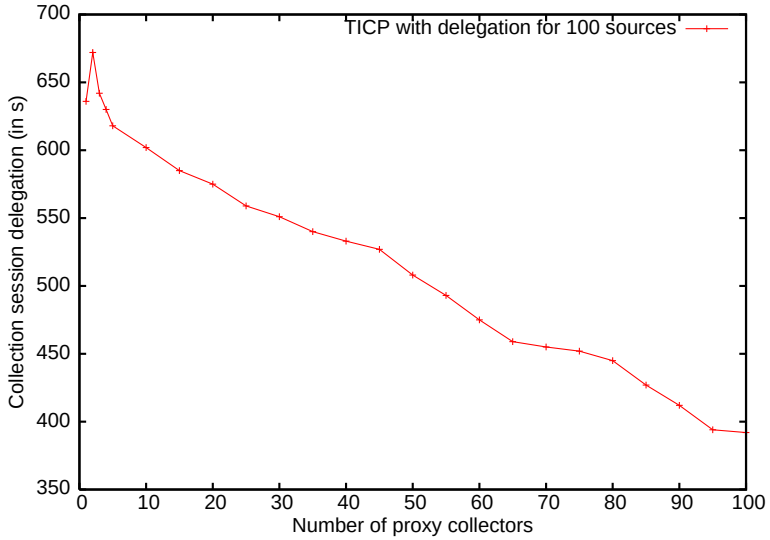

Fig. 16. TICP with delegation for 100 sources

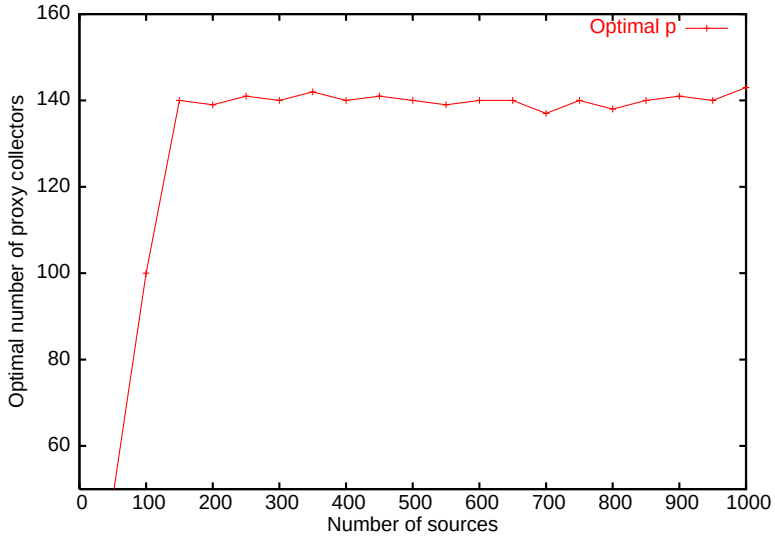

Fig. 17. Optimal number of proxy collectors

6 Related work

There is actually no general stand-alone transport solution to collect information from a large number of network entities on an end-to-end basis while providing congestion and error control. Most if not all existing solutions are application specific that do not answer all the questions we rise in this work. Some solutions provide partial collection because the applications don't need it, others require the collaboration of intermediate nodes and routers and so are not general, and there are some solutions close to ours but use simple congestion control as periodic or exponential probing at a low rate in order not to congest the network.

TCP is a protocol to collect one file from one machine. Ensemble-TCP [20] and HTTP 1.1 [21] are two protocols to collect many files from one machine using one single TCP connection. SNMP [15] and measurement infrastructures as Skitter [17] implement periodic and exponential probing at a low rate to collect information from routers and to probe machines for the purpose of delay and topology measurements. When it comes to data disseminated through the Internet, there is no clear solution to achieve this collection on an end-to-end basis. Though some protocols like Concast [14] use the principle of active networks to program routers so that they can participate to the collection.

There has been some effort in the literature to develop collection solutions for reliable multicast applications. Indeed, in reliable multicast, sources that did not receive a packet need to send a NACK asking the collector for a retransmission of this packet (we keep the terms collector and sources even though the collector in this context is transmitting data and the sources are listening). Sending many NACKs may cause congestion in the network or at the collector. The problem is called NACK implosion and several solutions have been proposed in the literature to collect these NACKs while avoiding congestion. The difference from our context is that the NACK information can be safely filtered; there is no need that a host sends a NACK if another host has already sent a NACK for the same packet. Thus, it was proposed to aggregate NACKs either along a tree that connects all sources or using multicast itself. In [12], it is proposed that leaf sources send their NACKs to a parent source (called designated receiver), which aggregates this information and sends it to its parent until it reaches the collector. In [13,16], a source waits for a random time before sending a NACK, and listens at the same time if another source has sent a NACK for the same packet. If so, the former source cancels its request, otherwise it sends it when the timer expires. Another approach for NACK aggregation is to use the principle of active networks to program nodes of a multicast tree as advocated by [14].

The reliable collection of information has also its application in sensor net-

works. Sensors are sources of information that wake up generally when an event happens and send information about this event to some collecting point called the sink. Some protocols exist in the literature for a reliable collection, e.g., [18], [19]. These protocols agree on that end-to-end transport solutions lead to poor performance in this case given the noisy nature of wireless links connecting the sensors, and the absence of permanent routes caused by the intermittent wake up of sensors and their limited lifetime. Per-hop transport protocols have been advocated for sensor networks. The information is proposed to be reliably sent from one sensor to another until it reaches the sink, with retransmissions done on a hop-per-hop basis. Clearly, such solutions are not optimal in the wired Internet where permanent routes exist and are provided by the IP protocol. Moreover, the round-trip time in the Internet is usually in the order of hundreds of milliseconds and links are of good quality. All this make an end-to-end solution the most appropriate for the Internet, which is what TICP provides.

7 Conclusions and perspectives

TICP is a transport protocol to collect information from a large number of network entities. It aims to control the congestion of the network and to minimize the collection session duration. This work explains and validates three main components of TICP. First, to ensure a smooth variation of the congestion control parameters, we add to TICP a mechanism to cluster information sources. Simulation and experimental results show that this mechanism ameliorates the performance by reducing the loss ratio of reports, and therefore shortens the collection sessions. Second, we extend the protocol to support collecting several packets of data from each source. We prove by simulations that our approach is scalable and that TICP performs better than a collection application using parallel TCP connections. Lastly, we enhance the collection further by delegating it to some intermediate nodes called proxy collectors. We modify TICP to support this delegation and to choose optimally the proxy machines. The simulations showed that for a well-chosen number of proxy collectors, TICP is able to collect faster data from sources. The optimal number of proxies to be used seems to be topology dependent and not function of the number of sources of information.

Our future research will be about the experimental evaluation of TICP in the cases of several packets of data per source and delegated proxies. We are also working towards the integration of TICP in Internet measurement infrastructures for the purpose of controlling the rate of probing packets and for the collection of passive measurements carried out at different links inside the network.

References

[1] M. Allman, H. Kruse, S. Ostermann, "An Application-Level Solution to TCP's Satellite Inefficiencies", Proceedings of the First International Workshop on Satellite-based Information Services (WOSBIS), New York, Nov. 1996.

[2] M. Allman, V. Paxson, W. Stevens, "TCP Congestion Control", RFC 2581, Apr. 1999.

[3] C. Barakat, M. Malli, N. Nonaka, "TICP: Transport Information Collection Protocol", in Annals of Telecommunications, vol. 61, no. 1-2, pp. 167-192, Jan. 2006.

[4] Mohamed Karim Sbai, Chadi Barakat, "Transport Information Collection Protocol with clustering of information sources", in proceedings of NTMS 2007 (Conference on New Technologies, Mobility and Security), Paris, May 2007.

[5] B. Donnet, P. Raoult, T. Friedman, M. Crovella, "Deployment of an Algorithm for Large-Scale Topology Discovery", in IEEE Journal on Selected Areas in Communications (JSAC), Dec. 2006, vol. 24, no. 12.

[6] T. S. Eugene Ng and H. Zhang, "Towards Global Network Positioning", ACM SIGCOMM Internet Measurement Workshop 2001, San Francisco, Nov. 2001.

[7] The Network Simulator ns-2, http://www.isi.edu/nsnam/ns/

[8] V. Paxson, M. Allman, "Computing TCP's Retransmission Timer", RFC 2988, Nov. 2000.

[9] The PlanetLab platform, http://www.planet-lab.org/

[10] M. G. C. Resende and R. F. Werneck, "A hybrid heuristic for the p-median problem", Journal of Heuristics, vol. 10, pp. 59-88, 2004.

[11] E. W. Zegura, K. Calvert and S. Bhattacharjee, "How to Model an Internetwork", IEEE Infocom '96, San Francisco.

[12] Lin (J.C.), Paul (S.), RMTP: A Reliable Multicast Transport Protocol.IEEE INFOCOM, March 1996.

[13] Floyd (S.), Jacobson (V.), McCanne (S.), Liu (C.), Zhang (L.), A reliable multicast framework for light-weight sessions and application level framing. ACM SIGCOMM, August 1995.

[14] Calvert (K. L.), Griffioen (J.), Mullins (B.), Sehgal (A.), Wen (S.), Concast: Design and Implementation of an Active Network Service. IEEE Journal on Selected Area in Communications (JSAC), vol. 19, no. 3, March 2001.

[15] Case, J., M. Fedor, M. Schoffstall, and J. Davin. 1990. A Simple Network Management Protocol (SNMP). RFC 1157, May.

[16] Lacher, M. S., Nonnenmacher, J., and Biersack, E. W. 2000. Performance comparison of centralized versus distributed error recovery for reliable multicast. IEEE/ACM Transactions on Networking. 8, 2 (Apr. 2000), 224-238.

[17] Skitter infrastructure by CAIDA, http://www.caida.org/tools/measurement/skitter/

[18] Stann (F.), Heidemann (J.), RMST: Reliable Data Transport in Sensor Networks. IEEE International Workshop on Sensor Net Protocols and Applications (SNPA), May 2003.

[19] Wan (C.), Campbell (A.), Krishnahmurthy (L.), PSFQ: A Reliable Transport Mechanism for Wireless Sensor Networks. ACM International Workshop on Wireless Sensor Networks and Applications, September 2002.

[20] Eggert, L., Heidemann, J., and Touch, J., "Effects of Ensemble TCP," ACM Computer Comm. Review, January 2000.

[21] RFC 2616, Hypertext Transfer Protocol http://tools.ietf.org/html/rfc2616

Résumé

Cette thèse présente un ensemble de solutions pour une métrologie efficace de l'Internet. La métrologie de l'Internet est un domaine en pleine évolution étant donnée la croissance exponentielle du réseau en termes de machines, réseaux, et applications, couplée à l'absence de solutions standard pouvant répondre à tous les besoins des utilisateurs et des opérateurs. Notre objectif le long de cette recherche a été le développement des solutions qui passent à l'échelle et qui réduisent la charge sur le réseau sans pour autant compromettre la précision des résultats obtenus. Nous avons agi pour cela sur deux fronts. Sur le front des mesures passives, nous avons proposé une solution pour la capture du trafic à travers un réseau d'opérateur. Au lieu de collecter tout le trafic et de l'analyser ultérieurement, notre solution permet aux opérateurs de définir une tâche de mesure et de programmer les routeurs de telle sorte que chacun d'eux collecte la quantité de trafic nécessaire sans que le volume total du trafic collecté dépasse un certain seuil. Cette réduction du volume du trafic collecté s'effectue par le biais de l'échantillonnage spatiale et temporelle. L'échantillonnage a un impact négatif sur la précision de la mesure, un impact qui peut être réduit par des techniques de filtrage et des solutions protocolaires. Nous avons illustré ceci sur deux exemples concrets, le premier celui de la détection des gros flots utilisateurs et le deuxième celui de la métrologie des grandes populations variant au cours du temps. Pour les deux exemples nous avons mené une étude pour évaluer l'impact de l'échantillonnage suivi par des solutions réduisant l'erreur lors de l'inversion des informations échantillonnées. Ensuite nous avons abordé l'approche active dans la métrologie et avons présenté une solution appelée TICP qui permet de contrôler le débit des paquets sondes à injecter dans le réseau et d'assurer la fiabilité quand il s'agit d'une application sondant un grand nombre de machines distribuées à travers le réseau. TICP peut être vu comme un multiplexage d'un grand de connexions TCP partant d'une même machine et destinées à des machines différentes en une seule connexion. Toutes nos solutions ont été validées par des simulations et des traces réelles.

Mots-clés: Métrologie de l'Internet, processus stochastiques, traces, échantillonnage, mesures passives et actives

Abstract

In this thesis we present a set of solutions for a better monitoring of the Internet. This is an interesting area of research evolving quickly to cope with the increase in the size and heterogeneity of the Internet and to fill the gap caused by the absence of global and standard monitoring infrastructure for the Internet. Our main objective during this research is the design and validation of efficient monitoring solutions that are able to scale and to reduce the load on the network without compromising the usefulness and quality of the results. We work in two directions. In one direction we address the passive monitoring approach and we propose a framework for choosing the routers that should contribute to some monitoring task and the amount of traffic to be collected by each router. This is done by writing the passive monitoring problem as an optimization problem and controlling the participation of routers by the means of spatial and temporel traffic sampling. Unfortunately traffic sampling cannot be done without a negative impact on the quality of measurements. This impact is to be evaluated and an optimized inversion method is to be defined for each metric one wants to consider. With the help of two concrete examples, we evaluate the impact of traffic sampling and illustrate how the inversion of the sampled information is to be done. One example is the detection and raking of the largest flows in a traffic, and the second example is the counting and tracking in time of the size of a large and changing population. In parallel, we work on the active approach and we present a protocol solution called TICP for regulating the rate of probes to be injected in the network and for controlling the reliability of the monitoring. TICP can be seen as a solution to multiplex the many TCP connections one needs to use for a network wide active monitoring in one connection. All our solutions during this thesis are supported by analytical models and validated on real traffic traces. The thesis provides an overview and a discussion of the state of the art and perspectives on our future research.

Keywords: Internet monitoring, stochastic processes, traces, sampling, passive and active measurements