

Optimisation du contrôle d'admission et développement d'outils d'analyse de performance

Rapport de stage de fin d'étude

Par Sébastien MICHEL
Version 0.1, 13 décembre 2002

Copyright ©2000 Sébastien Michel et France Télécom R&D
All rights reserved.

Remerciements

James W. ROBERTS <james.roberts@rd.francetelecom.com>

Chef de l'URD DAC/OAT/TLB à France Télécom R&D

Sara OUESLATI-BOULAHIA <sara.oueslati@rd.francetelecom.com>

Maître de stage

Nabil BENAMEUR <nabil.benameur@rd.francetelecom.com>

Maître de stage

Alexandre PROUTIERE <alexandre.proutiere@rd.francetelecom.com>

Thomas BONALD <thomas.bonald@rd.francetelecom.com>

Philippe OLIVIER <phil.olivier@rd.francetelecom.com>

Gwenael REGNIE <gwenael.regnie@rd.francetelecom.com>

Slim BEN FREDJ <slim.benfredj@rd.francetelecom.com>

Laurent VALEYRE <laurent.valeyre@rd.francetelecom.com>

Christophe LASORNE <christophe.lasorne@rd.francetelecom.com>

Sébastien QUEGUINER <>

Stagiaire 2002 pour l'URD AFM

Denis SACCHET <denis.sacchet@libertysurf.fr>

Stagiaire 2001 pour l'URD TLB

Benoit SIBAUD <benoit.sibaud@rd.francetelecom.com>

et le reste de l'équipe DAC/OAT <>

Table des matières

Remerciements	iii
1 Un cadre propice à l'innovation	1
1.1 Quelques chiffres	1
1.2 L'organisation interne	2
1.3 Le cadre du stage	2
1.4 A propos de ce document	3
2 Contexte du stage	5
2.1 Travaux de l'équipe TLB	5
2.2 Le projet du contrôle d'admission	6
2.2.1 Pourquoi un contrôle d'admission ?	6
2.2.2 Un contrôle d'admission ok...mais comment ?	7
2.2.3 Un projet déjà bien avancé	9
2.2.4 L'environnement de travail	9
2.2.5 Objectifs du stage	9
3 Optimisation du <i>contrôle d'admission</i>	13
3.1 Architecture logicielle de départ	13
3.2 Étude de nouvelles solutions	14
3.3 Nouvelle architecture logicielle	16
3.4 Évaluation du <i>contrôle d'admission</i>	18
4 Mécanismes d'estimation de bande passante	19
4.1 Amélioration du TCP/fantôme	19
4.1.1 Économie de ressource réseau	20
4.1.2 Atténuation du phénomène d'oscillation	23
4.2 Estimation basée sur le taux de perte	24
4.2.1 <i>sting</i>	24
4.2.2 SNMP	26
5 Monitoring et performance	29
5.1 Les motivations	29
5.2 Indicateurs de performance	29
5.3 Description technique de la solution	30

5.4	Le résultat graphique	32
6	Apports personnels	35
6.1	En programmation	35
6.2	En recherche d'information	35
6.3	En théorie des réseaux	36
6.4	En systèmes	37
6.5	Maîtrise de nouveaux outils	37
6.6	Intégration et travail en groupe	37
A	Manuels	39
A.1	cac	39
A.2	estima	43
A.3	gen	45
A.3.1	gen_host.conf	47
A.3.2	gen_param.conf	48
A.4	ExecGen	50
A.5	monitor_gen	54
A.6	monitor_estima	57
B	Procédure de lancement d'une expérience	61
C	Exemples de programmes multi threads	67
C.1	communication par sémaphores	67
C.2	modèle de purge à la demande	68
C.3	utilisation de mutex	70
D	Glossaire	73
	Bibliographie	81

Chapitre 1

Un cadre propice à l'innovation

1.1 Quelques chiffres

France Télécom R&D¹, leader européen de la recherche et du développement en télécommunications, regroupe l'ensemble des activités R&D² de France Télécom. Son objectif est de créer un maximum de valeur pour l'opérateur. Ses missions consistent à anticiper les révolutions technologiques, les ruptures d'usage et à innover pour offrir à ses clients le meilleur des activités télécoms présentes et à venir.

Avec 3700 ingénieurs et chercheurs répartis sur 12 sites, dont un aux États Unis et un au Japon, France Télécom R&D représente un véritable atout pour l'opérateur France Télécom. En effet, cette filiale initie et développe près de 80% des produits et services commercialisés par l'opérateur. De plus, son organisation transverse et décentralisée permet une ouverture internationale du groupe et ainsi de s'impliquer auprès des grands groupes industriels et de la communauté scientifique mondiale.

Preuve de l'activité grandissante du groupe, France Télécom détient 5028 brevets au niveau mondial ; au mois de juin 2002, France Télécom a déposé 82 logiciels et fait 126 demandes de brevets. Mais France Télécom R&D, c'est aussi 90% des clients satisfaits, une dizaine de start-ups internes créatrices d'une centaine d'emplois, environ 600 étudiants en stage ou en thèse ...

¹Anciennement CNET, Centre National d'Etudes des Télécommunications

²Recherche et Développement

1.2 L'organisation interne

France Télécom R&D répartit ses activités dans plusieurs directions dont la DAC³. La DAC est chargée des produits et systèmes destinés aux nouvelles générations de réseaux de France Télécom pour la France et l'international dans le cadre de la convergence fixe/mobile/internet, ainsi qu'à l'interconnexion du réseau de France Télécom avec des réseaux tiers. Elle veille à la compatibilité des réseaux d'accès et dorsaux, assure l'évolution des systèmes de commutation de troisième génération, vérifie la qualité des services fournis, étudie de nouvelles méthodes de signalisation, d'adressage et de numérotation.

Chaque direction contient des laboratoires. La DAC dirige notamment les laboratoires OAT⁴ et CPN⁵. Le laboratoire OAT a l'objectif général de conduire les études innovantes qui permettent à France Télécom de concevoir l'architecture générale, le dimensionnement et la planification de ses réseaux, d'en optimiser les coûts et d'en maîtriser la qualité de services. Le laboratoire CPN a pour but d'accompagner le déploiement des réseaux dorsaux en mode paquet par France Télécom et ses filiales.

Enfin, chaque laboratoire comporte un certain nombre d'unités de recherche. Le laboratoire OAT comprend notamment l'unité de recherche TLB⁶. Les principales missions relatives à cette unité de recherche sont de :

- Caractériser les différentes classes de trafic IP et ATM en faisant mener si nécessaire des campagnes de mesures.
- Etudier, en terme de performances, les mécanismes et protocoles proposés pour le partage de ressources entre classes de services différentes et entre les flots d'une même classe de service, dans les réseaux de paquets large bande utilisant les nouvelles technologies IP, ATM, ... et proposer de nouveaux modèles de services répondant aux besoins de France Télécom.
- Etudier les diverses politiques de routage permettant d'assurer des qualités de services différenciées.
- Etudier l'impact de l'utilisation de caches sur l'ingénierie du trafic des réseaux IP.
- Participer à la définition des règles d'ingénierie et de trafic dans les réseaux large bande de France Télécom.
- Etudier les principes de dimensionnement des réseaux de paquets large bande et définir les mesures de trafic nécessaires.

1.3 Le cadre du stage

J'ai effectué mon stage dans le laboratoire DAC/OAT dans l'unité de recherche TLB. J'ai travaillé au sein d'une équipe d'une dizaine de personnes sur le site d'Issy-les-Moulineaux sur le sujet *Contrôle d'admission dans les réseaux*

³Direction de l'Architecture, de l'intégration et de la Commande des réseaux

⁴Optimisation, Architectures et Trafic

⁵Coeurs de réseaux Paquets pour le NGN (cf. glossaire page 76)

⁶Modélisation du Trafic et performance des réseaux Large Bande

multiservices.

J'ai été chargé de poursuivre l'implémentation logicielle du concept de contrôle d'admission sur plate-forme GNU/Linux que Denis Sacchet avait initiée en collaboration avec DAC/CPN. Pour cela, j'avais à ma disposition une station de développement ainsi qu'une plate-forme de test composée d'une demi-douzaine de stations. Toutes les stations tournaient sous les systèmes d'exploitation GNU/Linux (2.2.21 et 2.4.xx) ou Solaris (2.5.1 et 2.7).

L'équipe DAC/OAT/TLB est composée de permanents :

James ROBERTS
Nabil BENAMEUR
Thomas BONALD
Sara OUESLATI-BOULAHIA
Philippe OLIVIER
Alexandre PROUTIERE
Hidega Hidega TIKU

mais il y a aussi d'autres intervenants :

Slim Ben FREDJ (thésard sur le contrôle d'admission dans les réseaux IP)
Gwénaél RÉGNIE (thésard sur l'écoulement et l'intégration des trafics temps réel et élastique)
Denis SACCHET (ancien stagiaire, 2001)
Sebastien MICHEL (stagiaire, 2002)

1.4 A propos de ce document

Dans ce document, j'essaie d'être le plus simple possible en évitant les termes techniques ; néanmoins, en raison du sujet du stage, je suis parfois amené à employer des termes techniques qui peuvent paraître compliqués ou obscurs. C'est pour cette raison que j'inclus à la fin de ce document en Annexe D un glossaire technique regroupant quelques définitions et explications sur des notions abordées dans ce rapport.

Chapitre 2

Contexte du stage

L'objectif de l'équipe est de caractériser le trafic Internet afin de mieux le comprendre et mieux le modéliser. L'équipe travaille aussi sur l'évaluation de performance de mécanismes de QoS (Quality of Service) tels que les différentes techniques de routage et d'équilibrage de charge, le *contrôle d'admission*, l'ordonnancement...

2.1 Travaux de l'équipe TLB

Nous pouvons distinguer plusieurs échelles de trafic Internet. La plus élevée est la session et la plus basse est le paquet. L'originalité de la modélisation du trafic réseau de l'équipe TLB est leur vision à l'échelle du flot ou de la session. Des modélisations plus classiques travaillent plutôt à l'échelle du paquet. La session correspond à une activité de l'utilisateur (exemple : transaction e-commerce, navigation web...). Une session est composée de flots (exemple : les pages web successives d'une même session de navigation), eux même composés de paquets. La figure 2.1 illustre ces trois échelles de trafic. La modélisation du trafic à l'échelle du paquet est difficile (propriété d'auto-similarité du trafic). Les sessions sont souvent mal définies et difficiles à identifier en pratique (par exemple, de nombreux services permettent de ne pas fermer explicitement une session ouverte). La QoS perçue par l'utilisateur s'exprime à l'échelle du flot. Elle correspond par exemple au temps de transfert d'une page web ou d'un fichier par FTP. La modélisation à l'échelle du flot présente l'avantage d'être à la fois simple et appropriée.

D'une manière générale, un flot est un ensemble de paquets groupés dans le temps et relatif à une instance d'application, cela peut par exemple correspondre au transfert d'un fichier ou d'un flux audio. On peut distinguer deux grandes catégories de flot :

Les flots élastiques : ils correspondent au transfert d'un document (page web, fichier mp3). Ils sont caractérisés par leur taille. Le débit¹ d'un flot

¹ $D = \frac{L}{T}$, avec D : le débit du transfert, L : la taille du transfert, T : le temps de transfert

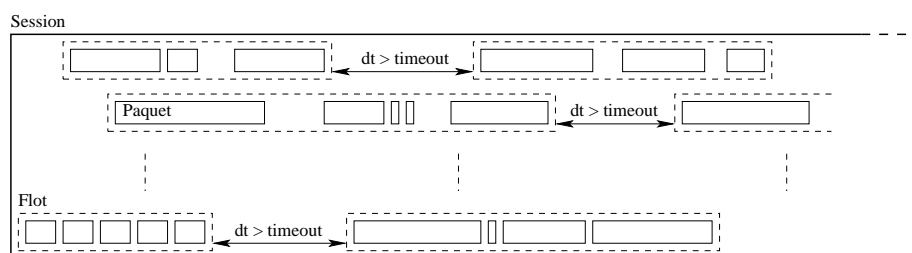


FIG. 2.1 – Les différents niveaux de trafic

élastique s'adapte à la bande passante disponible sous le contrôle de TCP dans les réseaux IP actuels. Leur QoS est quantifiée par leur débit ou leur temps de transfert.

Les flots streaming : ils correspondent à une génération de flux temps réel pour les applications audio ou vidéo. Un flot streaming a un débit et une durée intrinsèques qui ne dépendent pas de la bande passante disponible. Leur QoS s'exprime en terme de délai maximum et de taux de perte des paquets.

Dans notre étude, le flot est composé d'une suite de paquets ayant le même quadruple {adresse IP source, port source, adresse IP destination, port destination} et groupés dans le temps. Ainsi si deux paquets possèdent le même quadruple mais qu'ils sont séparés par un certain temps appelé *timeout*, ils ne seront pas considérés comme faisant partie du même flot (figure 2.1).

La nécessité du *contrôle d'admission* pour les flots streaming est généralement reconnue. L'originalité du *contrôle d'admission* proposé ici est qu'il s'applique aux flots élastiques qui représentent 85% des flots IP.

L'intérêt d'appliquer le *contrôle d'admission* à cette catégorie des flots est exposé dans l'article [3]. Bien qu'une implémentation du *contrôle d'admission* pour les flots élastiques et les flots streaming ait été proposée [4], dans le reste du rapport on se focalise sur les flots élastiques.

2.2 Le projet du contrôle d'admission

2.2.1 Pourquoi un contrôle d'admission ?

Le contrôle d'admission est né dans l'idée d'assurer une QoS² sur les réseaux TCP/IP, UDP/IP (multiservices) actuels car aujourd'hui aucune garantie de qualité sur les réseaux IP existe.

Pour pouvoir assurer une QoS acceptable les opérateurs ont recours à l'ATM qui permet de faire de la réservation de ressources. Ce système de réservation n'est pas optimal car les ressources consommées par les utilisateurs sont souvent inférieures à celles réservées. Il y a donc une perte de ressources par sous-

²En Anglais, Quality of Service, ou en Français, Qualité de Services

utilisation des infrastructures du réseau et donc un manque à gagner pour l'opérateur. De plus le système de réservation de ressource ne convient pas à tout type de service. En effet, le trafic IP est complexe à décrire par de simples paramètres et est souvent composé de petits flots. Une autre solution consiste à effectuer ce *contrôle d'admission* à partir de mesures de bande passante disponible (comme nous le verrons dans la suite de ce rapport).

Enfin un *contrôle d'admission* permettrait d'éviter les écroulements de performance des flots en situation de congestion. Une telle situation se produit quand le trafic offert excède la capacité du réseau. Le volume utile de données transmises se dégrade ; il y a beaucoup de retransmissions et d'abandons prématurés des flots dus à des débits trop faibles. Il en résulte une utilisation inefficace des ressources du réseau et une QoS médiocre.

2.2.2 Un contrôle d'admission ok...mais comment ?

Principe

Le *contrôle d'admission* opère à l'échelle du flot au niveau d'un lien. Il est placé en coupure sur un lien et capture tous les paquets qui transitent par ce lien (figure 2.3). L'implémentation proposée ne requiert ni signalisation, ni réservation explicite de ressources. Comme les nouveaux flots ne sont pas explicitement signalés, ils sont identifiés "au vol" en scrutant l'entête de chaque paquet transmis sur ce lien. Un nouveau flot est admis si le débit réalisable par ce dernier ainsi que les autres flots en cours est supérieur à un seuil prédéfini (une plage de seuils adéquats a été identifiée [3]). Le débit disponible, critère d'admission, est mesuré en temps réel. La mise en place de ce *contrôle d'admission* ne nécessite pas de modification de la couche protocolaire existante. Par ailleurs, même si le *contrôle d'admission* agit de manière locale (sur un lien, par exemple), il permet d'améliorer la performance globale du réseau. Ainsi, il est facile à mettre en place et peut être introduit de façon progressive dans l'ensemble du réseau.

Aujourd'hui, le contrôle d'admission tourne sur un PC sous GNU/Linux. À terme, il pourrait être intégré dans un boîtier ou dans un routeur (cf. Glossaire page 78).

Algorithme

Voici l'algorithme qui est utilisé pour la réalisation du *contrôle d'admission* : A l'arrivée d'un paquet, ce dernier est analysé, les informations pour l'identification du flot sont extraites de l'entête du paquet ({adresse IP source, port source, adresse IP destination, port destination}), elles sont comparées à une table contenant les identifiants des flots en cours, si le paquet appartient à un flot établi, il est transmis, sinon, un mécanisme de décision d'admission pour ce nouveau flot est activé. La décision se base sur la comparaison de la mesure du

débit effectué par une méthode de TCP/fantôme³ et un seuil fixé⁴. Lorsqu'un nouveau flot est accepté, il est ajouté à la table des flots en cours et le temps de passage du premier paquet observé est initialisé. Lorsqu'il s'agit d'un paquet appartenant à un flot connu (qui existe dans la table des flots), le temps de passage de ce paquet (dernier paquet observé du flot) et le nombre d'octets pour ce flot sont mis à jour dans la table. Si au contraire le flot est refusé, un compteur est mis à jour et le paquet est ignoré.

D'autres fonctions existent, notamment un mécanisme de détection des flots expirés suivant un certain *timeout* et un module pour réaliser des statistiques.

La figure 2.2 présente le schéma correspondant à cet algorithme.

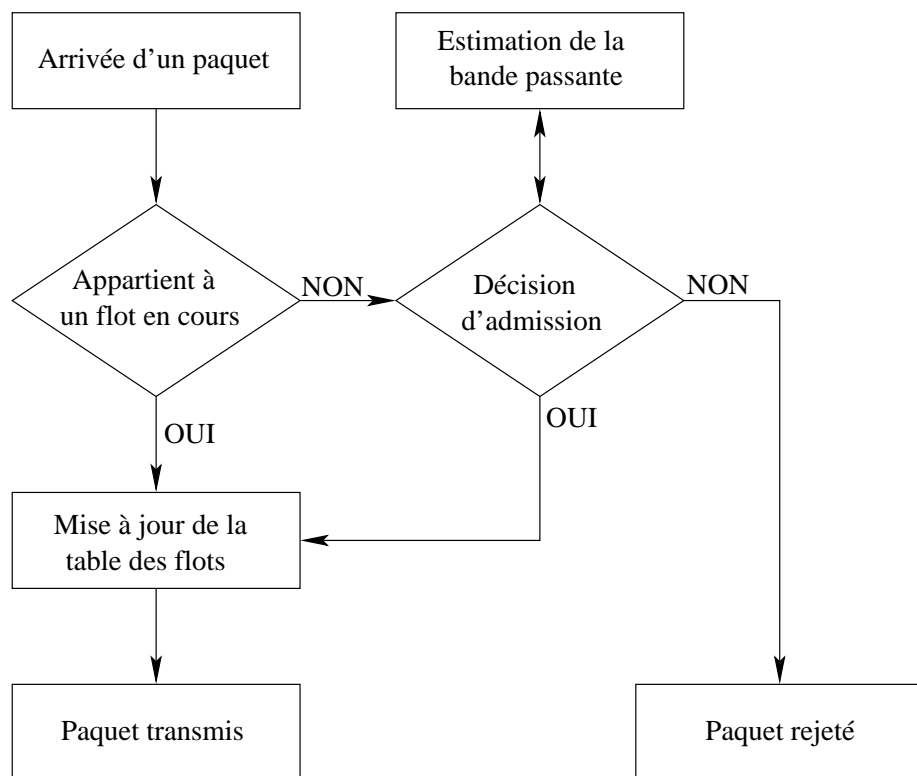


FIG. 2.2 – Algorithme du *contrôle d'admission*

³Méthode active de mesure de bande passante disponible. Appelée 'méthode active' car elle implique une injection de trafic en plus dans le système. Ce type de méthode est opposé aux méthodes de mesures passives qui permettent une mesure sans rajout de trafic dans le système.

⁴Ce seuil doit être fixé autour de 1% de la capacité totale [3].

2.2.3 Un projet déjà bien avancé

L'équipe souhaitait mettre en place une plate-forme de démonstration du contrôle d'admission ayant pour objectif de démontrer l'efficacité d'un tel mécanisme et sa faisabilité en pratique. Pour cela ils ont fait appel à Denis Sacchet en 2001 dans le cadre de son stage de deuxième année de cycle ingénieur EFREI. Son stage a débouché sur l'implémentation du principe du *contrôle d'admission*. Par manque de temps, Denis Sacchet n'a pas pu optimiser l'application, ni faire une analyse des résultats de son application faute d'outils d'évaluation.

Je suis donc venu continuer le travail de Denis Sacchet. Dans un premier temps, je devais optimiser les applications existantes, éventuellement les modifier. Dans un second temps, en créer de nouvelles pour améliorer le rendu visuel des résultats de performance et en faciliter l'analyse.

2.2.4 L'environnement de travail

Le schéma 2.3 illustre la plate-forme où est implémentée le *contrôle d'admission*. La machine qui fait office de contrôle d'admission est placée en coupure sur un lien qui relie le réseau LAN de l'entreprise et des machines puits via un hub 10Mb. Les machines puits servent principalement pour le Générateur de trafic. Le programme de génération de trafic (Générateur) a été développé par Denis Sacchet [5]. Il nous permet de tester le *contrôle d'admission* et d'évaluer la performance des flots. Une ou plusieurs instances de Générateur sont lancées sur une ou plusieurs machines du LAN. Chaque instance envoie des sessions de paquets vers une machine puits selon un processus d'arrivée de Poisson. Dans les sessions, le nombre de flots (ou pages) transférés ainsi que leur taille suivent une distribution paramétrable (Pareto, déterministe, exponentielle, uniforme). Pour générer une situation de surcharge nous lançons plusieurs instances de générateur sur différentes machines⁵.

2.2.5 Objectifs du stage

A mon arrivée le contrôle d'admission était fonctionnel, mais il restait limité en performance par rapport au nombre de flots et au taux d'arrivée⁶ des flots. Des outils d'analyse (monitoring, script de calcul de performance...), devenaient également nécessaires afin d'automatiser le lancement des expériences et d'exploiter les résultats obtenus. Plus précisément, voici les objectifs de mon stage.

- **Comprendre les travaux existants :**

Dans un premier temps, j'ai étudié le rapport de stage de Denis Sacchet qui décrivait l'architecture du *contrôle d'admission* et ses principes. Dans un second temps, j'ai pris connaissance des outils existants (le module de *contrôle d'admission*, le module de génération de trafic) ainsi que leurs

⁵Une machine est limitée en nombre de connexions ouvertes par son système d'exploitation. La commande `ULIMIT` (ou `UNLIMIT`) permet de repousser cette limite

⁶Nombre de nouveaux flots par unité de temps.

codes sources.

- **Optimisation du contrôle d'admission existant :**

L'objectif premier de mon stage était d'améliorer les performances du contrôle d'admission.

- **Automatiser un maximum de tâches :**

Le grand nombre d'applications à mettre en marche lors d'une expérience (activation du *contrôle d'admission*, lancement d'une instance de Générateur par machine, gestion de l'archivage des résultats...) implique un minimum d'automatisation afin de travailler plus vite et plus simplement.

- **Modularisation de l'estimateur de congestion :**

Après réflexion, il nous a paru nécessaire de séparer les modules de l'estimation de bande passante et du contrôle d'admission pour des raisons que j'exposerai plus tard.

- **Monitoring :**

La plate-forme de démonstration doit afficher des résultats de performance parallèlement au déroulement de l'expérience. Il est par exemple utile de suivre en temps réel l'évolution du nombre de flots en cours de transfert, le débit réalisé par chaque flot terminé ainsi que les variations du débit disponible estimé (critère d'admission). C'est pourquoi, le développement d'outils graphiques pour le suivi et l'analyse des performances constituait aussi un objectif important de mon stage.

- **Autres :**

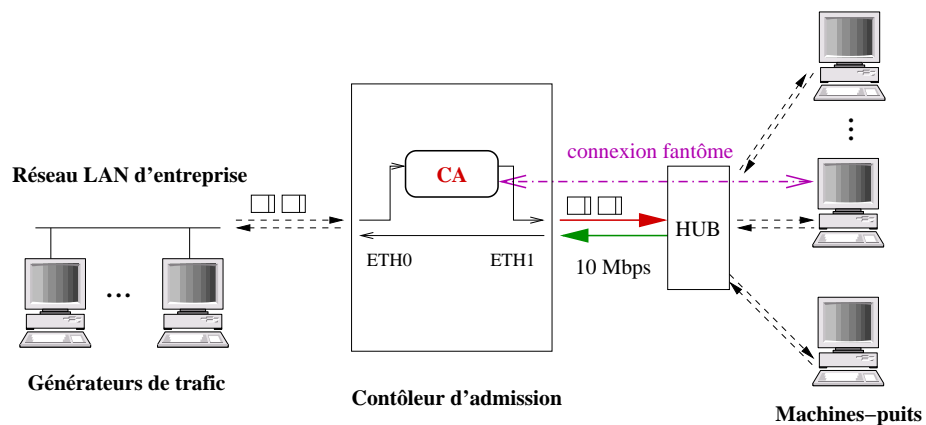


FIG. 2.3 – Plate-forme de démonstration

Pour accomplir ces tâches, on m'a demandé un minimum de créativité, d'esprit critique et d'initiative. Ainsi j'ai dû chercher des solutions de substitution pour améliorer le mécanisme d'estimation qui repose sur le TCP/fantôme et d'évaluer des solutions alternatives.

Chapitre 3

Optimisation du *contrôle d'admission*

3.1 Architecture logicielle de départ

Ce paragraphe décrit l'architecture logicielle de l'application de *contrôle d'admission*. Celle-ci comporte les modules suivants :

Gestion du trafic : L'application de *contrôle d'admission* doit être capable de récupérer tout le trafic du réseau Ethernet ¹ sur lequel elle est placée et surtout celui qui ne lui est pas adressé, afin d'être transparent pour les stations du réseau². Ensuite, il ne faut pas que tout ce trafic soit traité par la pile de protocole du système d'exploitation, il faut le récupérer tel quel (capture de paquet en mode raw -voir glossaire page 78). Enfin, il faut pouvoir réémettre également tel quel, ce trafic s'il est autorisé. Cela constitue trois modules, un premier récupérant le trafic sur le réseau et le passant au module de décision, un deuxième récupérant le trafic validé par le module de décision et le remettant sur le réseau, et un dernier recopiant directement le trafic sans traitement. Le *contrôle d'admission* est appliqué dans un seul sens, dans l'autre sens les paquets sont directement recopiés.

Estimateur de bande passante : L'application de *contrôle d'admission* doit pouvoir estimer la bande passante disponible sur le lien afin d'obtenir le paramètre sur lequel se base la décision d'admission.

Table de flot : L'application de *contrôle d'admission* doit tenir à jour une liste des flots en cours afin de déterminer si un paquet appartient à un flot en cours ou à un nouveau flot. Chaque trame récupérée sur le réseau

¹voir définition d'*Ethernet* page 74

²Dans ce cas le *contrôle d'admission* joue le rôle d'un composant réseau appelé bridge ou pont.

représente un paquet d'un flot.

Module de décision : Il faut un module capable de savoir si une trame appartient à un flot qui est déjà établi ou non, afin de savoir s'il doit être accepté ou rejeté en fonction des paramètres du *contrôle d'admission*. Ce module effectue aussi la purge de la table de flot en vérifiant si chaque flot a expiré ou pas.

L'architecture et l'organisation des modules de départ sont représentées dans la figure 3.1 :

P_ESTIMA : c'est le processus qui correspond au module *estimateur de bande passante*.

P_ETH2SOC, P_SOC2ETH et P_RECOPY : ce sont les processus correspondant aux modules *gestion du trafic*.

P_TABLE : c'est le processus qui correspond au module *table de décision*.

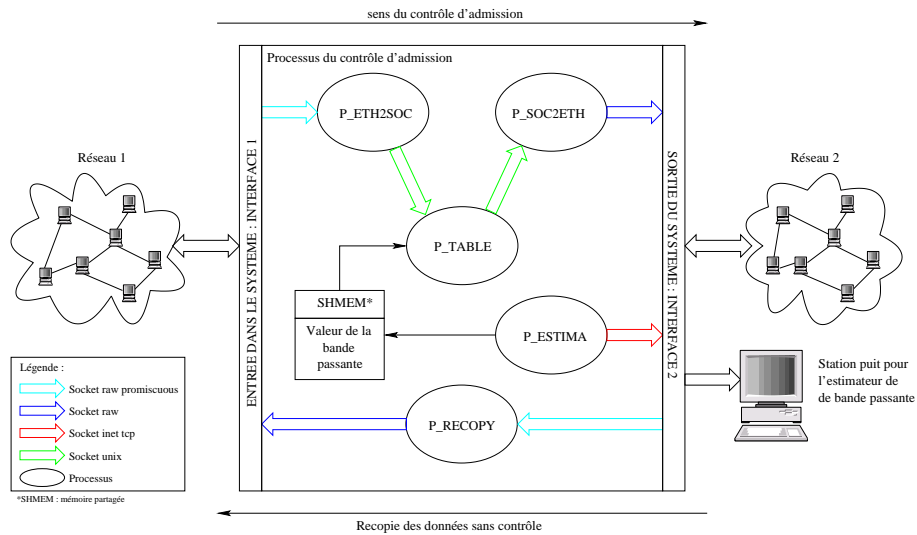


FIG. 3.1 – Architecture du programme de *contrôle d'admission*

3.2 Étude de nouvelles solutions

La nouvelle architecture logicielle a pour but d'accélérer le processus d'admission des flots afin d'augmenter le nombre de flot simultanés et leur taux d'arrivée à travers le *contrôle d'admission*.

Le plus gros problème à résoudre est le blocage causé par le mécanisme de purge qui est exécuté régulièrement par le module *table de décision* de manière séquentielle par rapport aux autres tâches notamment la gestion du trafic. Au-

trement dit, si l'arrivée d'un paquet dans le module *table de décision* coïncide avec le début de la purge, ce paquet est retardé de plusieurs dizaines de millisecondes. De plus, les paquets suivants s'accumulent dans les buffers des cartes réseau. Cette accumulation est probablement à l'origine de la réinitialisation des cartes réseau en cas de dépassement des buffers.

La solution retenue est d'introduire la notion de *multithread* et de modifier l'architecture logicielle du *contrôle d'admission* en conséquence. De cette manière la purge constitue un thread indépendant qui exécute une purge à la demande (grâce à la notion de *sémaphore* -cf. glossaire page 78) du module *table de décision*. L'avantage de la notion de thread est la possibilité de partager des structures de données dynamiques, comme le partage de la table des flots, entre les threads T_PURGE, T_DISPLAY et le processus P_TABLE.

La notion de *multithread* est également utilisée pour réaliser les opérations de calcul de statistiques et d'affichage. Cela permet d'alléger considérablement le module *contrôle d'admission*. Le schéma 3.2 illustre l'interaction entre les processus et les threads. Le processus P_TABLE fait une demande de purge (flèche 1), ce qui a pour effet l'activation du thread T_PURGE. En fin d'activité le thread T_PURGE invoque le thread T_DISPLAY (flèche 2) qui calcule et affiche les statistiques relatives à la table des flots (La description de la structure de la table des flots est disponible dans le rapport de Denis Sacchet [5]).

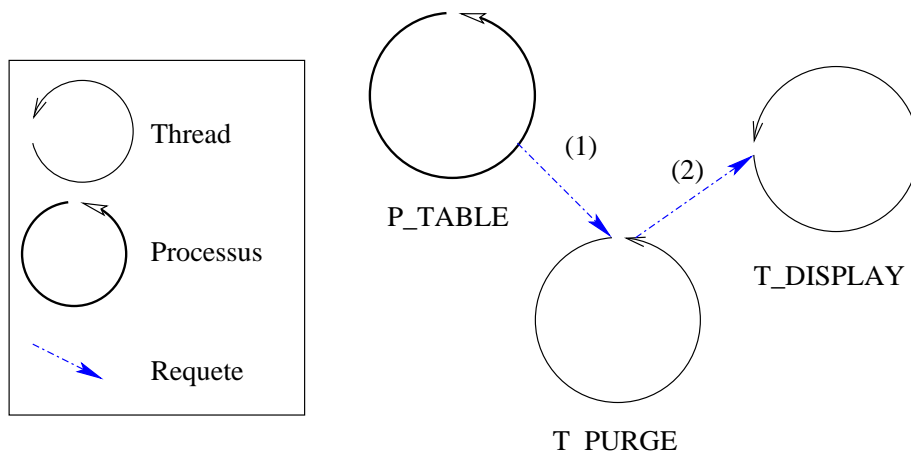


FIG. 3.2 – Interactions entre threads

P_TABLE

```

// * Initialisation des sémaphores pour      *
// * communiquer avec le thread T_DISPLAY *
if (sem_init(&sem_display, 0, 0) < 0)
    perror("sem_init");
// * Initialisation des sémaphores pour      *
// * communiquer avec le thread T_PURGE *
if (sem_init(&sem_purge, 0, 0) < 0)
    perror("sem_init");
while (1)
{
    ...
// * Demande de purge * (flèche 1)
    sem_post(&sem_purge);
}

```

T_PURGE

```

while (1)
{
// * Attente de demande de purge * (flèche 1)
    sem_wait(&sem_purge);
    ...
// * Demande d'affichage * (flèche 2)
    sem_post(&sem_display);
}

```

T_DISPLAY

```

// * Attente de demande d'affichage * (flèche 2)
while (1)
{
    sem_wait(&sem_display);
    ...
}

```

3.3 Nouvelle architecture logicielle

L'architecture découle en grande partie de la version antérieure du *contrôle d'admission*. Voici les différences :

- La disparition du module P_SOC2ETH qui n'est pas utile car le goulet d'étranglement de l'application se situe avant (au niveau de P_ETH2SOC).
- L'apparition du thread T_DISPLAY permettant de faire un affichage de statistiques (durée des flots les plus gros, nombre de flots identifiés, nombre

de flots actuellement dans le système, nombre de flots refusés...).

- L'apparition du thread T_PURGE réalisant l'opération de purge de la table des flots.

Les deux threads T_PURGE et T_DISPLAY sont activés à la demande. Ainsi, lorsque le processus P_TABLE souhaite purger, il envoie une demande de purge en modifiant l'état du sémaphore `sem_purge`. Quant au calcul et à l'affichage des statistiques c'est le thread de purge qui envoie une demande en modifiant l'état du sémaphore `sem_display` lorsqu'il a fini son travail, ce qui permet d'avoir un affichage cohérent.

L'architecture et l'organisation de ces modules sont représentées dans la figure 3.3³ :

P_ESTIMA : c'est le processus qui correspond au module *estimateur de bande passante*.

P_ETH2SOC et P_RECOPY : ce sont les processus correspondants aux modules *gestion du trafic*.

P_TABLE : c'est le processus qui correspond au module *table de décision*.

T_DISPLAY : c'est le thread qui correspond au module d'*affichage* d'informations et de statistiques.

T_PURGE : c'est le thread qui correspond au module *purge* de la table des flots.

Auparavant, les deux derniers étaient intégrés au module *table de décision*.

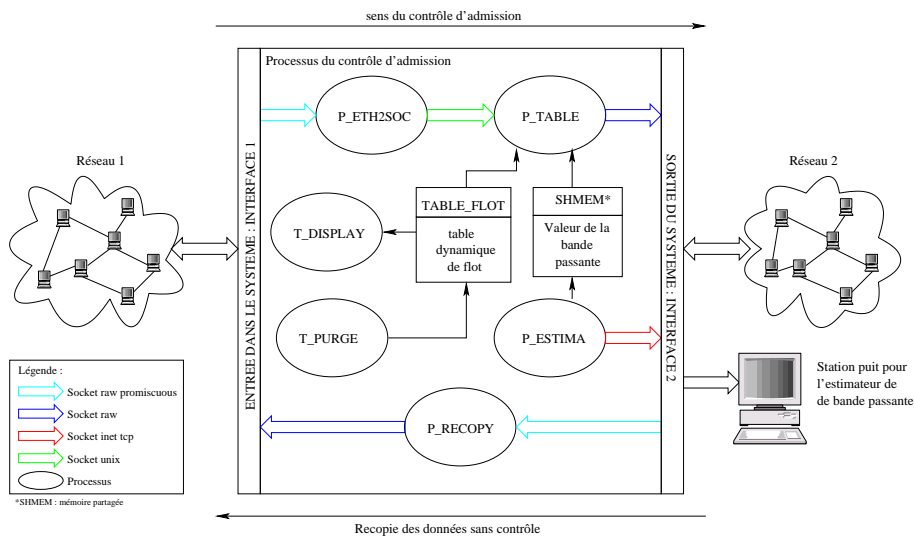


FIG. 3.3 – Nouvelle architecture du programme de *contrôle d'admission*

³Il n'y a aucune différenciation entre les threads et les processus dans la figure 3.3 car ce n'est pas nécessaire

3.4 Évaluation du *contrôle d'admission*

Grâce aux améliorations apportées, le phénomène de réinitialisation des cartes réseau en cas de surcharge a disparu. De plus, le module de *contrôle d'admission* est désormais capable de supporter des surcharges d'au moins 140% et un nombre de flots plus important (au moins 700 flots).

Malgré ces améliorations, la performance du *contrôle d'admission* en tant que pont reste inférieure à celle d'un module *ipchains* ou *iptables* (cf. glossaire page D). Ces modules permettent de transformer la machine en pont (fonction de bridging).

La figure 3.4 illustre une courbe correspondant au débit d'un flot témoin traversant le *contrôle d'admission* pendant un peu moins d'une heure.

Pendant la première partie de l'heure, j'ai fait tourner le *contrôle d'admission* sans connexion fantôme ni trafic supplémentaire. Le débit de la connexion témoin est d'environ 3500 Kbit/s.

Pendant la seconde partie de l'heure, j'ai transformé le *contrôle d'admission* en bridge simple en activant le module *ipchains*. Le débit de la connexion témoin dépasse les 7000 Kbit/s.

La figure 3.4 montre cet écart de performance entre le *contrôle d'admission* (première partie de la courbe) et une translation NAT (deuxième partie de la courbe, pour la définition de NAT voir glossaire page 76) (*ipchains*).

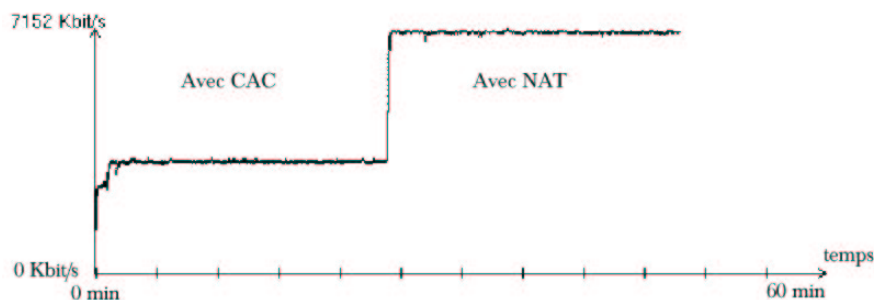


FIG. 3.4 – TCP/fantôme avec translation NAT

Afin d'améliorer la performance du *contrôle d'admission*, une solution consisterait à coder un *contrôle d'admission* en tant que module *ipchains* ou, mieux encore, en module *iptables*.

Chapitre 4

Mécanismes d'estimation de bande passante

Le module d'estimation de bande passante disponible pour un nouveau flot est une partie indispensable du *contrôle d'admission*. Il doit être efficace tout en étant peu gourmand en ressources. Une mesure passive du débit disponible serait idéale mais s'avère difficile à mettre en œuvre. En effet, nous avons exploré les deux grandes approches qui existent pour la mesure du débit disponible, à savoir :

La mesure passive : Elle ne nécessite pas d'injection de trafic supplémentaire. Elle est plutôt basée sur l'observation du trafic qui passe à un point du réseau et peut s'appuyer sur des outils tels que snmp (Simple Network Management Protocol). SNMP met à disposition des statistiques sur les paquets transmis telles que le nombre de paquets perdus, nombre d'octets sortant d'une interface donnée, ...

La mesure active : Elle est basée sur l'envoi de paquets de test dans le réseau. La méthode d'estimation actuellement utilisée dans l'application de *contrôle d'admission* (TCP/fantôme) est une mesure active.

4.1 Amélioration du TCP/fantôme

La méthode de *TCP/fantôme* consiste à établir une connexion TCP entre deux machines et de mesurer son débit. Le débit de cette connexion correspond alors, de façon assez fidèle (à une connexion près), au débit que pourrait avoir une nouvelle connexion. En effet, le protocole TCP partage plus ou moins équitablement la bande passante d'un lien entre les flots en cours de transfert à moins que certains soient contraints ailleurs que sur ce lien (Par exemple, par la vitesse du modem ou la lenteur d'un serveur). En pratique, ce principe est assez simple à implémenter.

Deux problèmes persistent néanmoins. Tout d'abord, ce procédé nécessite deux machines. Cela pose des problèmes pour la transparence du système puis-

qu'il faut configurer une deuxième station, autre que celle qui réalise le contrôle d'admission, pour réaliser l'estimation de la bande passante disponible. La machine puits doit avoir le service Discard (glossaire page 74) ouvert. Le deuxième inconvénient de cette méthode est la génération de trafic artificiel qui encombre le lien.

D'un point de vue implémentation, cette méthode suppose que l'émetteur envoie continuellement des données et qu'il est possible de connaître à chaque instant le débit instantané de la connexion. En pratique, cela est impossible à moins d'avoir un contrôle total sur la couche réseau du système d'exploitation, ce qui n'est pas le cas ici. La méthode retenue est donc la programmation de deux processus concurrents, l'un réalisant l'envoi des données, l'autre réalisant les calculs.

Pour éviter les variations brutales et ponctuelles du débit, le débit instantané est lissé exponentiellement en utilisant la formule suivante :

$$D_n = D_{n-1} \times \alpha + \frac{\text{taille}}{\text{duree}} \times (1 - \alpha), 0 \leq \alpha \leq 1$$

avec :

- D_n est le débit à la n^{ieme} mesure.
- α est le coefficient de lissage (plus la valeur est proche de 0, moins le passé est pris en compte, donc plus le débit réagit vite au changement).
- *taille* est le nombre d'octets envoyés depuis la dernière mesure.
- *durée* est la durée de l'intervalle de mesure.

En pratique, on peut prendre le paramètre α de l'ordre de 0.9 et une durée de l'ordre de quelque centaines de milisecondes.

4.1.1 Économie de ressource réseau

L'inconvénient majeur de la méthode TCP/fantôme est une consommation importante des ressources. En effet, nous avons constaté que le débit du TCP/fantôme était largement supérieur à celui des flots réels. Pour pallier ce problème, nous avons entrepris d'améliorer la méthode du TCP/fantôme.

La formule de Padhye [1] (équation 4.1) montre que le débit moyen réalisé par une connexion TCP, noté D , est inversement proportionnel au délai d'aller/retour ou RTT (cf. glossaire D).

$$D = \frac{w}{RTT} \sqrt{\frac{3}{2p}} \quad (4.1)$$

avec w : taille maximale de la fenêtre de la connexion TCP,
 p : taux de perte des paquets de la connexion,

La connexion TCP/fantôme a un RTT beaucoup plus petit et, par conséquent, un débit plus élevé comparé aux autres flots du système dans la mesure où elle est intégrée au *contrôle d'admission*. En effet, comme le montre la figure 4.1, le *contrôle d'admission* est implémenté dans un nœud qui fait le pont entre les deux parties du réseau. Si le module d'estimation s'exécute au niveau

de ce nœud, alors sa connexion a un parcours moins long à effectuer pour atteindre une machine puits par rapport à une connexion établie par une machine se trouvant de l'autre côté du *contrôle d'admission*.

Dans la nouvelle version, nous avons décidé de séparer les modules d'estimation et de décision et de délocaliser l'*estima*.

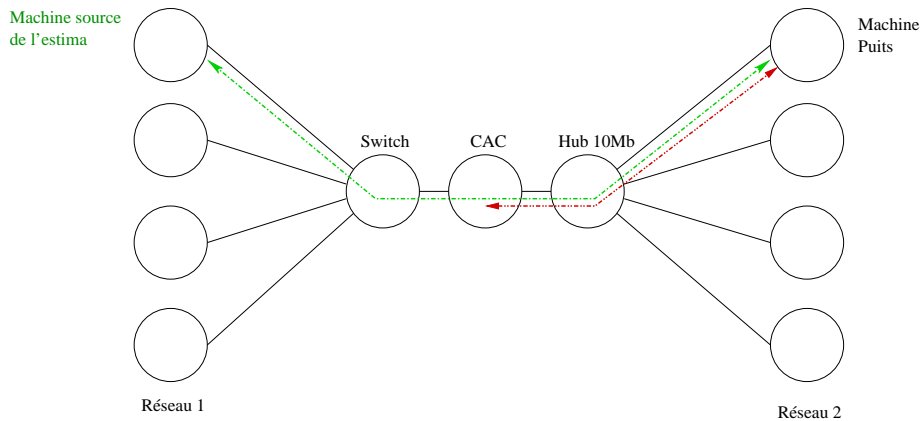


FIG. 4.1 – Mise en évidence des différences de RTT

La séparation des modules d'estimation et de décision présente des avantages pratiques, notamment la possibilité de changer les paramètres de l'*estima* sans devoir interrompre le programme de *contrôle d'admission*. De cette manière, nous pouvons changer la valeur du seuil d'admission et modifier la méthode d'estimation (donc l'expérience en cours) sans couper le réseau. La figure 4.2 illustre la nouvelle architecture.

Nous ne pouvons pas directement comparer les débits estimés dans l'application de *contrôle d'admission* première version et dernière version, car le module d'estimation de la bande passante disponible dans la première version est intégré au *contrôle d'admission* et ne produit pas d'historique (trace des débits estimés en fonction du temps).

Nous pouvons toutefois mesurer l'impact de la modification sur le débit des flots réels. La figure 4.3 trace le débit moyen réalisé par chaque flot terminé (pour plus d'informations sur l'interprétation du graphe, se reporter au chapitre 5). La ligne représente une moyenne des débits des flots. Sur cette figure, nous pouvons distinguer deux phases :

- La première phase correspond au scénario 1, où le *contrôle d'admission* tourne avec la connexion TCP/fantôme en local sur la même machine (petit RTT).
- La seconde phase correspond au scénario 2, où le *contrôle d'admission* tourne avec la connexion TCP/fantôme délocalisée. La connexion est initiée à partir d'une machine source sur l'un des deux réseaux alors que la machine puits se trouve sur l'autre réseau.

Nous observons que les débits des flots sont plus importants dans le deuxième scénario.

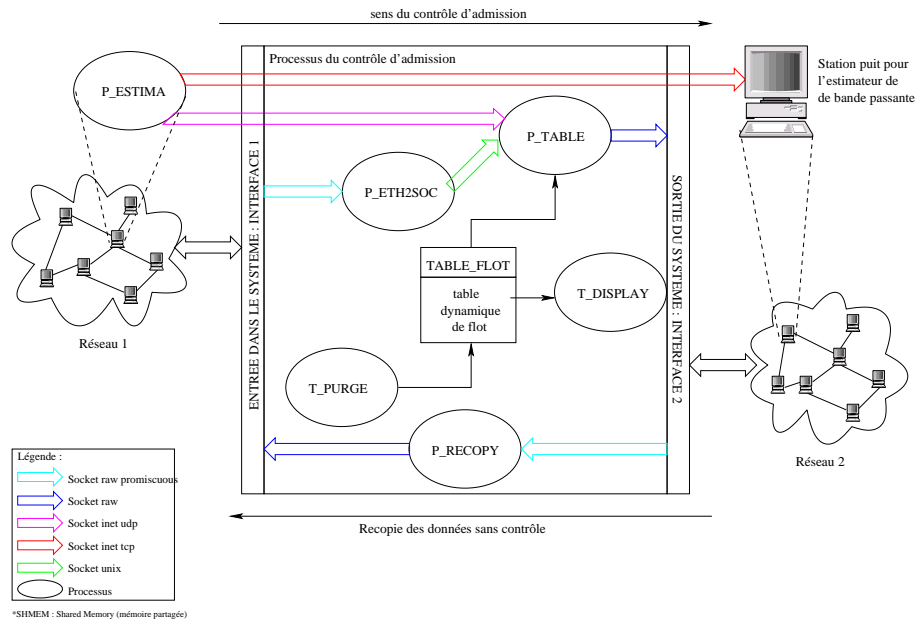


FIG. 4.2 – Nouvelle architecture de *contrôle d'admission* avec la délocalisation de l'*estima*

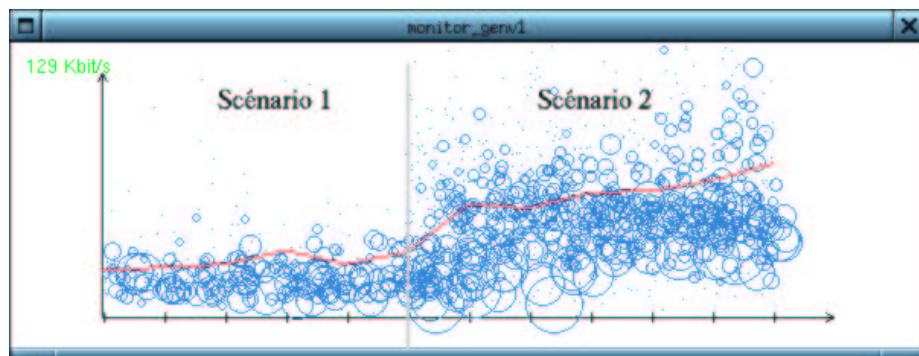


FIG. 4.3 – TCP/fantôme avec le programmes de *contrôle d'admission*

4.1.2 Atténuation du phénomène d'oscillation

Après avoir augmenté le RTT de l'estima nous avons identifié un nouveau problème. En situation de congestion, nous observons une oscillation du débit des flots du système autour du seuil de congestion. Cette oscillation a deux effets néfastes qui s'observent sur la figure 4.4 :

- Périodiquement le débit des flots se dégrade sensiblement, il peut descendre jusqu'à 50% en dessous du seuil de congestion, ce qui peut être gênant pour certains types d'applications (Par exemple, une vidéo). Cette baisse de débit se produit lorsque trop de flots sont admis. Ceci peut conduire à penser que la méthode d'estimation n'est pas suffisamment réactive, autrement dit elle ne refuse pas suffisamment tôt les flots.
- Par période, le système se vide. Autrement dit, trop de flots sont inutilement refusés (le débit de l'estimateur reste trop longtemps sous le seuil de congestion) et le nombre de flots dans le système est faible.

Le problème d'oscillation a été mis en évidence lors de démonstrations où on visualisait une séquence vidéo transférée en TCP. Pour assurer une bonne qualité, il était nécessaire de fixer le seuil de congestion à une valeur nettement supérieure à celle du débit du codage (au moins 2 fois) afin de compenser les grandes variations de débit. Or, plus le seuil est élevé plus le blocage des nouveaux flots est important.

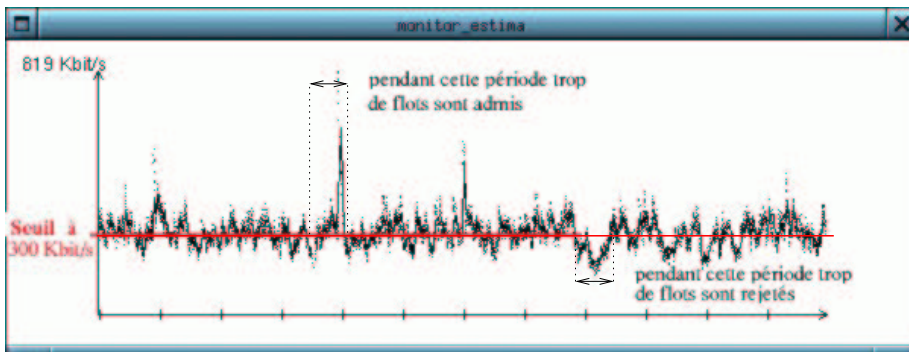


FIG. 4.4 – TCP/fantôme qui oscille autour du seuil de congestion

Pour atténuer ce phénomène d'oscillation plusieurs possibilités ont été étudiées :

Améliorer la réactivité de l'estima : Dans un premier temps, nous avons augmenté la réactivité du *contrôle d'admission* par rapport au signal de congestion envoyé par *estima*. Pour cela, il a simplement fallu augmenter la fréquence d'envoi d'information de congestion de la part de l'*estima* et augmenter la fréquence de lecture de la part du *contrôle d'admission*. Dans un second temps, afin d'anticiper le passage à l'état de congestion, nous avons testé une décision d'admission probabiliste plutôt que binaire.

Nous avons introduit un second seuil et une probabilité de congestion qui dépend du débit estimé. La figure 4.5 trace la probabilité de congestion dans la version initiale, où elle est binaire. Sur la figure 4.6, nous observons que si le débit estimé d'un flot est supérieur au seuil maximum la probabilité de congestion est nulle, tout nouveau flot est alors accepté. Par contre, tout nouveau flot est refusé si le débit estimé est inférieur au seuil minimum.

problème intrinsèque à la méthode de mesure : Nous avons remarqué que malgré ces améliorations le phénomène d'oscillation persistait même s'il a été atténué. Pour l'atténuer davantage, une proposition est de forcer la fenêtre TCP du TCP/fantôme à rester haute afin de supprimer la phase de slow start dans laquelle il bascule de temps à autre suite à des pertes consécutives. Pendant la période d'agrandissement de la fenêtre, le débit du TCP/fantôme est faible, par conséquent des flots sont rejetés alors que le lien n'est pas nécessairement en situation de surcharge.

4.2 Estimation basée sur le taux de perte

Après avoir travaillé sur l'amélioration de la méthode du TCP/fantôme, nous avons voulu explorer une alternative basée sur la mesure du taux de perte. Il s'agit d'une mesure indirecte de la bande passante disponible. En effet, la formule de Padhye (cf. équation 4.1) montre que le débit moyen d'une connexion TCP dépend à la fois de son RTT et de son taux de perte. Plus précisément, le débit est inversement proportionnel à la racine carrée du taux de perte. Dans cette perspective, nous avons étudié les deux possibilités décrites ci-dessous.

4.2.1 sting

`sting`, est le nom d'une application écrite par Stefan Savage en 1999 [2] (<http://www.cs.washington.edu/homes/savage/>). Cette application a pour but de mesurer le taux de perte dans un intervalle de temps donné.

Le programme offre la possibilité de paramétrer le nombre de paquets émis dans l'intervalle de mesure, le temps qui sépare l'envoi de 2 paquets successif (l'interpaquet), etc.

Même si `sting` reste une méthode de mesure active, le volume de données transmis est moins important qu'avec une connexion TCP/fantôme.

La mise en place de cet outil a nécessité des changements de configuration système car le programme faisait appel à des modules très particuliers qui existaient en 1999 mais qui ne sont plus forcément d'usage. Ensuite, il a fallu modifier `sting` pour qu'une mesure en continu du taux de perte soit faite et pour que les informations nécessaires soient envoyées au *contrôle d'admission*. J'ai donc reconfiguré une partie des machines de la plate-forme pour qu'elles puissent héberger l'application `sting` modifiée. Enfin, il a fallu calibrer le logiciel pour trouver un bon compromis entre la réactivité de l'estima et la fiabilité

Probabilités de congestion en fonction du débit estimé :

FIG. 4.5 – estima première version

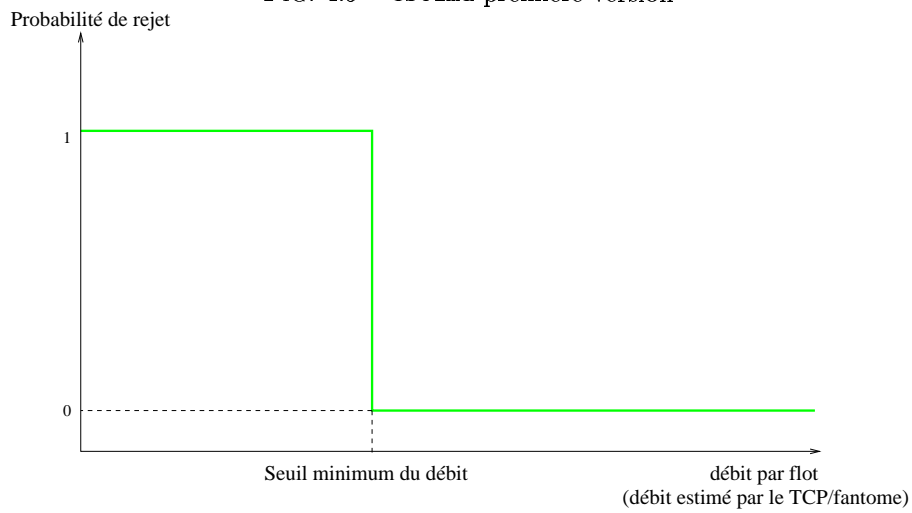
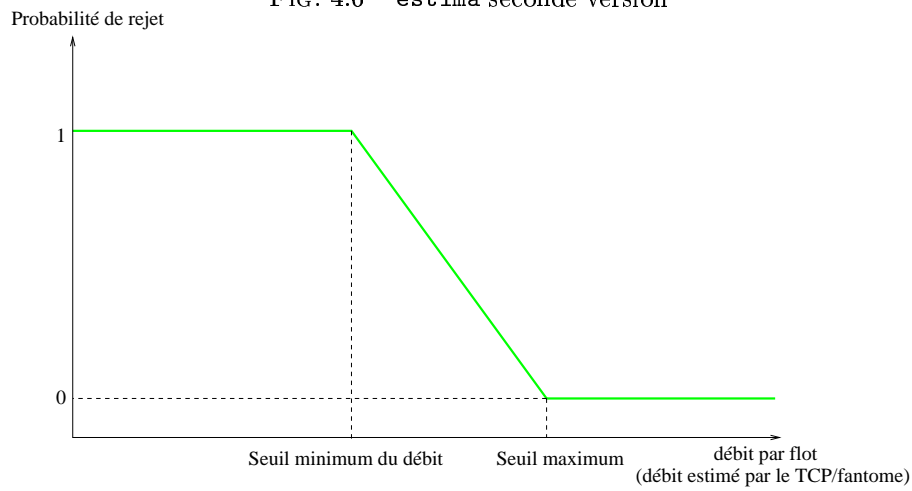


FIG. 4.6 – estima seconde version



de la mesure.

L'évaluation de cet outil a révélé un problème de réactivité. En effet, malgré un calibrage permettant à `sting` d'être le plus sensible possible, le taux de perte de paquets était trop faible par rapport au débit estimé. Ainsi, `sting` réagissait avec un certain retard par rapport à l'occurrence d'une congestion. Ce problème nous a poussé à chercher d'autres solutions.

4.2.2 SNMP

Le principe d'SNMP est de rendre disponible un certain nombre de statistiques du système sur le réseau. SNMP se présente comme un service réseau. Le serveur SNMP est appelé agent. Il génère la MIB (Management Information Base [6]). Cette MIB est consultable par le biais d'un client SNMP tel que `snmpwalk` ou `snmpget`.

Voici un exemple de consultation de MIB :

```
# Consultation de la partie 'ip' de la MIB.

root@p-poisson:~/tmp# snmpwalk -c CAC.ORG localhost ip

IP-MIB::ipForwarding.0 = INTEGER: forwarding(1)
IP-MIB::ipDefaultTTL.0 = INTEGER: 64
IP-MIB::ipInReceives.0 = Counter32: 31232798
IP-MIB::ipInHdrErrors.0 = Counter32: 0
IP-MIB::ipInAddrErrors.0 = Counter32: 0
IP-MIB::ipForwDatagrams.0 = Counter32: 172213
IP-MIB::ipInUnknownProtos.0 = Counter32: 0
IP-MIB::ipInDiscards.0 = Counter32: 0
IP-MIB::ipInDelivers.0 = Counter32: 1622541
IP-MIB::ipOutRequests.0 = Counter32: 106297492
IP-MIB::ipOutDiscards.0 = Counter32: 0
IP-MIB::ipOutNoRoutes.0 = Counter32: 0
IP-MIB::ipReasmTimeout.0 = INTEGER: 0
IP-MIB::ipReasmReqds.0 = Counter32: 8608
IP-MIB::ipReasmOKs.0 = Counter32: 3068
IP-MIB::ipReasmFails.0 = Counter32: 0
IP-MIB::ipFragOKs.0 = Counter32: 0
IP-MIB::ipFragFails.0 = Counter32: 0
IP-MIB::ipFragCreates.0 = Counter32: 45219
IP-MIB::ipAdEntAddr.10.193.163.123 = IpAddress: 10.193.163.123
IP-MIB::ipAdEntAddr.10.193.163.134 = IpAddress: 10.193.163.134
IP-MIB::ipAdEntAddr.127.0.0.1 = IpAddress: 127.0.0.1
IP-MIB::ipAdEntIfIndex.10.193.163.123 = INTEGER: 2
IP-MIB::ipAdEntIfIndex.10.193.163.134 = INTEGER: 3
IP-MIB::ipAdEntIfIndex.127.0.0.1 = INTEGER: 1
IP-MIB::ipAdEntNetMask.10.193.163.123 = IpAddress: 255.255.255.0
IP-MIB::ipAdEntNetMask.10.193.163.134 = IpAddress: 255.255.255.0
```

La seconde solution étudiée pour la détection de congestion est la mesure du taux de perte par SNMP. Cette mesure est passive, car nous exploitons les statistiques calculées par SNMP au niveau d'une interface réseau. En effet, une quantité importante d'informations sous forme de compteurs est disponible grâce à SNMP. Cette solution a été abandonnée faute de temps. Le problème que nous avons rencontré est délicat à traiter. Les compteurs prévus pour les pertes de paquets restent à zéros. Malgré, des recherches, des questions sur la mailing list e2e (end2end-interest) nous n'avons pas pu identifier la source du problème. Il est possible que le compteur de paquets perdus ne soit pas activé. Cette solution reste à explorer.

Chapitre 5

Monitoring et performance

5.1 Les motivations

L'application de *contrôle d'admission* a été développée avec le double objectif d'expérimenter et de démontrer sa nécessité. Dans cette perspective, il est important de disposer d'outils permettant de bien comprendre le fonctionnement du *contrôle d'admission* et ses effets sur la performance des flots.

J'ai donc été chargé de développer une application de monitoring qui permet l'affichage graphique d'indicateurs de performance pendant le déroulement des expériences. L'outil doit également pouvoir rejouer des scénarios existants.

5.2 Indicateurs de performance

Deux types de résultats nous intéressent :

- Les variations du débit disponible mesuré par *estima*, mesure de l'état de congestion sur le lien.
- Le débit moyen réalisé par les flots au fur et à mesure qu'ils se terminent. Ce paramètre est fourni par l'application *Générateur*.

Pour ce faire, deux applications distinctes ont été développées :

- L'une trace une courbe des débits disponibles estimés. L'affichage se présente sous forme d'un nuage de points et d'une courbe. Chaque point correspond à un débit disponible estimé à un temps t . La courbe représente la moyenne de ces débits par intervalle de temps (par défaut une minute).
- La deuxième application fournit un '*diagramme à bulles*' et une courbe. Chaque bulle représente un flot terminé. L'ordonnée du centre de la bulle indique son débit moyen, et l'abscisse correspond au temps de fin du flot. Le diamètre de la bulle est proportionnel à la taille du flot. Ceci permet d'accorder plus d'importance (au niveau visuel) au débit des grands flots. La courbe (ligne brisée) trace les débits moyens des flots sur des intervalles des temps successifs (par défaut cinq minutes).

5.3 Description technique de la solution

Choix du langage :

Dans un premier temps, j'ai voulu développer l'apparence graphique en GTK/Perl (cf. définition page 74). Perl, est un très bon langage en ce qui concerne le traitement de chaînes de caractères et la gestion de fichiers. Mais j'ai très vite été bloqué à cause du manque de maturité de Perl en ce qui concerne le multithread. Le multithread est nécessaire dans le développement de ces outils d'analyse pour lire le fichier de log tout en gérant l'affichage graphique des résultats. Nous avons donc choisi l'environnement GTK1.2 avec les bibliothèques C.

Algorithme :

Le traitement comporte deux parties qui tournent en parallèle :

Extraction des données : L'idée est de lire le fichier de log (historique d'information) généré par l'application *estima* ou *Générateur*. Nous pouvons relire des logs anciens ou lire "au vol" les logs générés en temps réel par l'application. Le principe est simple : consulter un fichier régulièrement et en extraire les nouvelles informations.

Au préalable, j'ai dû modifier les applications (*estima* et *Générateur*) afin de disposer d'un fichier de log correctement formaté. Chaque instance de *Générateur* produit son propre fichier log. Comme, plusieurs instances de *Générateur* sont lancés sur différentes machines, une centralisation de l'ensemble des statistiques était nécessaire. J'ai exploité les montages de disques en NFS (partage de disque en réseau) pour faire en sorte que toutes les instances écrivent dans le même fichier de résultat.

Le programme de monitoring doit vérifier régulièrement si il y a de nouveaux résultats à lire dans le fichier de log (issu d'*estima* par exemple), auquel cas il vérifie leur validité. Une fois validés, ces résultats sont stockés dans la table d'informations qui sera consulté par le module d'affichage.

La restitution graphique : À l'initialisation du programme, un graphique vierge est généré. C'est sur ce graphique que les courbes sont tracées à chaque rafraîchissement. Le traçage de ces courbes se fait en une seule lecture de la table d'informations mise à jour par le module d'extraction des données. Lors de cette passe, chaque élément est lu, comptabilisé dans la moyenne si nécessaire, puis tracé. Dans le cas de l'*estima* le traçage ne comporte qu'un point par relevé. Par contre, pour le *Générateur* le traçage demande un calcul de diamètre de la bulle en fonction de la taille du flot. L'affichage de la moyenne est faite sous forme d'une ligne brisée.

Description de la structure :

Les figures 5.1 et 5.2 illustrent les structures utilisées pour le stockage des données respectivement dans le moniteur de l'*estima* puis du *Générateur*. Le stockage du débit est suffisant dans le cas du moniteur d'*estima*.

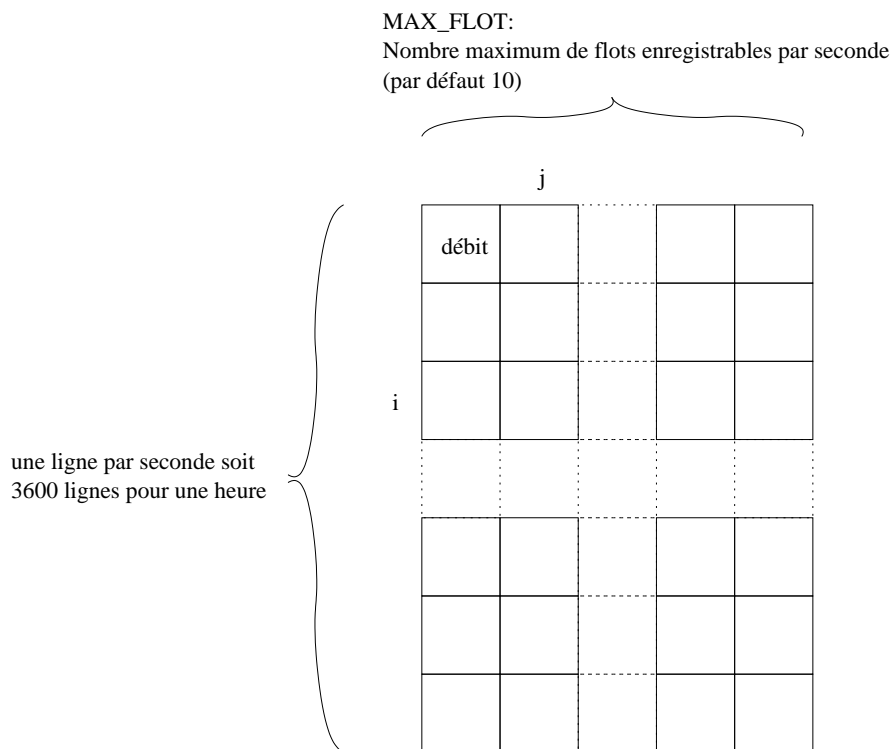


FIG. 5.1 – Structure de données utilisée par le moniteur d'estima

5.4 Le résultat graphique

Sur la figure 5.3, chaque point représente une estimation du débit du TCP/fantôme fournie par l'estima. L'axe des abscisses correspond au temps, le graphe représente par défaut une heure d'expérience. L'axe des ordonnées, nous avons le débit en Kbit par seconde. La ligne correspond à la moyenne des débits par intervalle de 60 secondes.

Sur la figure 5.4, chaque cercle représente un flot terminé que le Générateur a créé. La taille du cercle est proportionnelle à la taille du flot. Lorsque le flot est trop petit pour être représenté par un cercle il est représenté par un point. En abscisse, nous avons le temps. En ordonnée, nous avons le débit en Kbit par seconde.

Pour les deux applications, l'échelle au niveau des ordonnées est ajusté automatiquement en fonction des données à afficher.

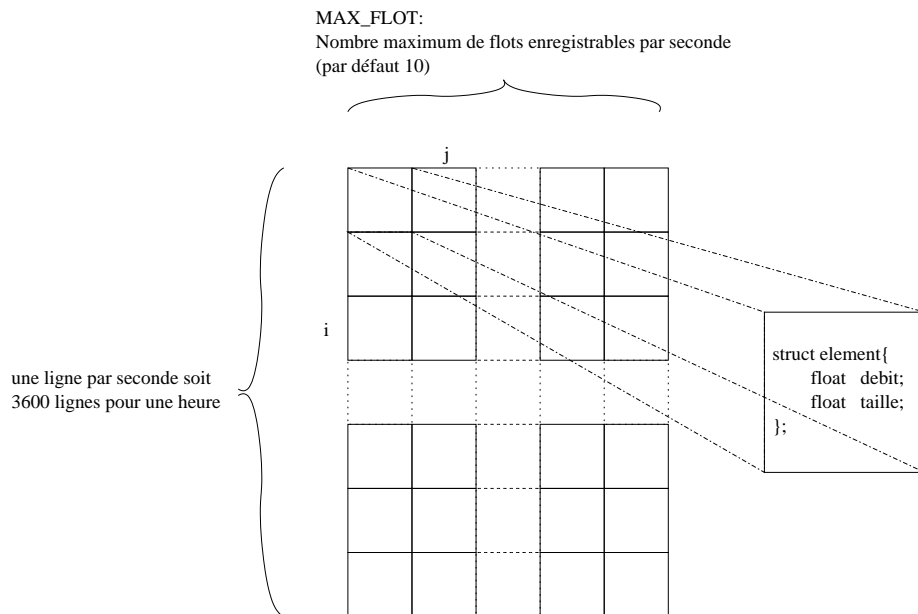


FIG. 5.2 – Structure de données utilisée par le moniteur de Générateur

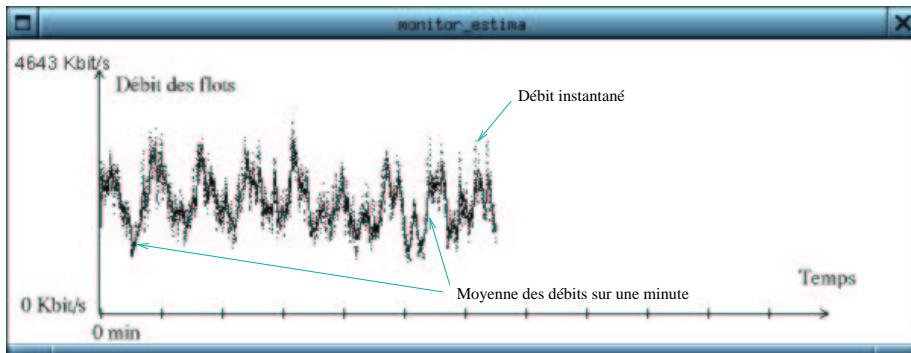


FIG. 5.3 – Rendu graphique de l'estima

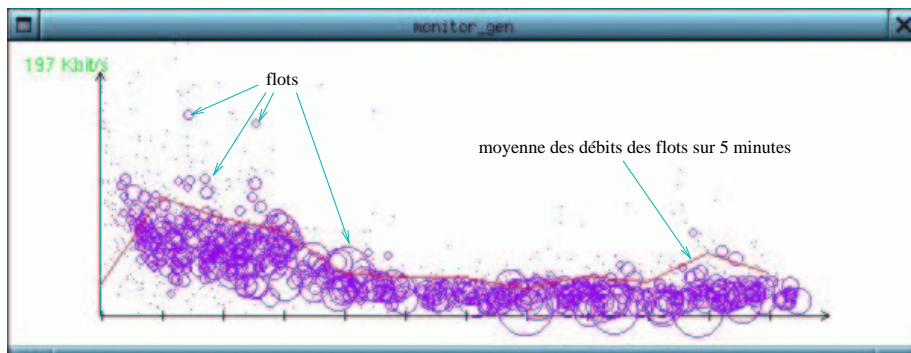


FIG. 5.4 – Rendu graphique du Générateur

Chapitre 6

Apports personnels

6.1 En programmation

J'ai dû écrire ou reprendre des applications au cours de mon stage, le contrôle d'admission, l'estimateur de bande passante, le générateur de trafic, sting et les moniteurs, toutes en environnement Unix. Pour certaines applications, il a fallu faire attention au portage entre les différents systèmes d'exploitation de type Unix (Linux, Solaris)

La programmation de ces logiciels m'a demandé énormément de recherches, notamment pour la programmation système et GTK sous Unix. Les cours et les TPs d'Unix m'ont donné une excellente base et ce stage m'a permis d'approfondir mes connaissances. Pour le développement de ces applications, j'ai également appris à utiliser pkg-config, gcc, make, vim . . .

Pour mener à bien le codage de ces applications, j'ai également étudié plusieurs codes sources de logiciels existants et disponibles en *Open Source*¹. Ces logiciels constituent une référence en ce qui concerne la façon de coder, aussi bien sur la forme (commentaires abondants, présentation claire, lisibilité, . . .) que sur le fond (ingéniosité et élégance des solutions, utilisation astucieuse des appels système, . . .).

Cette expérience dans la programmation Unix ne peut être que positive pour mon avenir professionnel, que je souhaite orienter vers le logiciel libre.

6.2 En recherche d'information

Comme déjà mentionné au paragraphe précédent, j'ai dû effectuer beaucoup de recherches pour trouver les solutions aux problèmes posés. Je pense que dans le travail d'ingénieur, la recherche efficace d'information est un atout considérable. Il est important de bien se documenter avant de se lancer dans un projet. Cela va de pair avec la faisabilité : analyser l'existant, analyser le potentiel des outils mis à disposition, imaginer des solutions alternatives.

¹Voir définition d'*open source* page 77

Les sources de documentation sont nombreuses à France Télécom R&D. Tout d'abord, le site d'Issy-les-Moulineaux dispose d'une bibliothèque technique recensant plusieurs milliers d'ouvrages sur tous les domaines de l'informatique, des réseaux, des télécommunications et des sciences en général.

La deuxième source de documentation est, bien sûr, Internet. Le réseau mondial regorge d'informations et c'est à la fois sa force et sa faiblesse. L'étude du code source de logiciel libre m'a appris énormément de choses, notamment la manière de programmer avec les bibliothèques GTK et GDK. Les forums de discussion et leurs archives ont apporté des réponses à plusieurs questions grâce à l'expérience de personnes ayant déjà eu à résoudre les mêmes problèmes.

Mais cette profusion d'informations peut également être source de confusion, il faut savoir où chercher les informations . . . Et pour cela, un conseil qui s'applique dans 99 % des cas, c'est de remonter à la source. Par exemple, vous recherchez une information sur une fonctionnalité du noyau, il faut d'abord regarder qui s'occupe de cette branche, puis trouver l'email et le site personnel de cette personne ; à partir de là, il est possible de retrouver la documentation, soit écrite par l'auteur lui-même, soit conseillée par lui, ou s'il n'existe pas de documentation, de contacter l'auteur pour discuter avec lui des difficultés rencontrées.

Enfin la troisième source d'information, et pas la moindre, est l'équipe dans laquelle j'ai travaillé. Celle-ci est très disponible, jamais avare d'explications, patiente et tolérante à l'égard de mes ignorances. Elle a autant envie d'apprendre que de faire apprendre. C'est très agréable de travailler dans cette ambiance d'entre-aide constante. Cela permet d'avancer beaucoup plus vite dans son travail car tout le monde a une expérience et il est toujours bénéfique, pour le donneur comme pour le receveur, de la faire partager.

6.3 En théorie des réseaux

Mon environnement de travail, un service de recherche spécialisé dans l'optimisation et le dimensionnement des réseaux, m'a permis d'améliorer ma culture générale des réseaux. Que ce soit au niveau des protocoles réseaux, des services offerts, de l'architecture, les connaissances de mes collègues sont très nombreuses. Au-delà de mon sujet de stage, qui concernait les réseaux IP essentiellement, j'ai pu aborder le protocole ATM, les domaines MPLS (norme intervenant dans le routage), et au cours de discussions informelles, le GPRS, l'UMTS, l'ADSL, l'avenir de la téléphonie mobile et des réseaux WLAN (Wireless LAN), . . .

J'ai également eu à mettre en pratique des cours théoriques enseignés à l'EFREI, tels que les probabilités. Il s'avère que dans le domaine des réseaux, le calcul probabiliste est utile pour la caractérisation du trafic, l'évaluation de la performance des mécanismes de contrôle de trafic. C'est bien de voir des cours théoriques dispensés à l'EFREI mis en pratique dans un domaine concret comme celui-ci.

6.4 En systèmes

J'ai pu perfectionner mes connaissances sur plusieurs distributions Linux, à savoir la Slackware, la Debian et la RedHat. Ces trois distributions, quoique très similaires dans leur utilisation, utilisent trois philosophies différentes quant à leur installation et leur configuration. Alors que la Slackware est la plus spartiate (l'installation d'un programme passe pratiquement toujours par une compilation des sources et la configuration se fait directement dans les fichiers), la Debian et la RedHat utilisent chacune un système de packages qui facilitent l'installation (.deb pour la Debian et .rpm -Redhat Package Management- pour la RedHat).

6.5 Maîtrise de nouveaux outils

Pour effectuer ce rapport, j'ai également appris à utiliser des logiciels très puissants comme Latex, Xfig ou Gimp :

- Latex est un standard dans le monde scientifique. Il génère des documents d'une très grande qualité typographique à partir d'instructions passées dans un fichier texte. Il est assimilé à la classe des traitements de texte, mais contrairement à des logiciels comme Microsoft Word, la personne qui rédige un document sous Latex ne fournit que le contenu, Latex se chargeant tout seul de la forme, alors que sous Word, l'auteur du document doit s'occuper de l'apparence (c'est lui qui fait la mise en page, qui choisit les polices, ...), et en général, il oublie un peu le fond, la structure du document ...
- Xfig est un outil de dessin vectoriel. Il permet de réaliser des illustrations de grande qualité. Contrairement à un logiciel de dessin bitmap où l'unité est le pixel, un document de dessin vectoriel contient des objets : des lignes, des ellipses, des points, du texte ... Chaque objet possède des propriétés : point de départ, points intermédiaires, point final, inclinaison, type de figure, taille de la police ... A partir de toutes ses informations, l'illustration finale est construite. L'avantage de ce mode de dessin est qu'il permet de faire des mises à l'échelle sans perte de qualité. Toutes les figures de ce rapport ont été réalisées avec Xfig.
- Gimp est un logiciel de traitement d'image. Il m'a notamment permis de réaliser des retouches de capture d'écran

6.6 Intégration et travail en groupe

Étant donné mon travail, j'ai passé la plupart de mon temps en salle machine plutôt que dans mon bureau. Le cadre est toutefois resté propice aux échanges dans la mesure où je partageais le laboratoire avec un autre stagiaire (Sébastien Queguiner) et deux ingénieurs d'étude faisant partie d'une autre équipe de recherche (Laurent Valeyre et Christophe Lasorne de OAT/AFM).

En plus des échanges au sein de mon équipe, j'ai pu avoir des échanges avec des personnes très impliquées dans différentes communautés (linuxfr.org, FSF,

APRIL, ABUL...) de logiciel libre.

J'ai également pu assister à une thèse sur le *contrôle d'admission*. J'ai participé à des démonstrations du *contrôle d'admission* notamment pour Alcatel et la direction de France Telecom R&D

Tous ces échanges m'ont permis de mieux comprendre la R&D et le métier de l'opérateur.

Annexe A

Manuels

A.1 cac

Nom

`cac` - Application de contrôle d'admission

Synopsys

```
cac [OPTIONS] hôte
```

Description

CAC est une application qui réalise un contrôle d'admission entre deux réseaux Ethernet. Un réseau est branché sur l'interface `int1` et l'autre sur l'interface `int2`. Le contrôle d'admission est effectué entre l'interface 1 et l'interface 2. Dans l'autre sens, tout le trafic est transmis.

Le contrôle d'admission prend des arguments supplémentaires optionnels (il existe des valeurs par défaut).

Voici l'extrait de l'option `'-help'` avec les valeurs par défaut de chaque option :

```

Usage: cac3 [OPTION] host

General options :
-----
-d, --debug           log to syslog more information (default no)
-f, --force           allow overwriting of existing files (default no)
-h, --help            print this message and exit
-M, --min_nb_packet  minimum number of packets in a flow to display it
                    (default 10)
-V, --version         display version information and exit
-v, --verbose         print some comments during execution (default no)

Options for bandwidth estimator :
-----
-p, --port            port to connect to on host (default 9)

Options for admission control :
-----
-i, --int1            first interface (default eth1) (1)
-o, --int2            second interface (default eth2) (1)
-l, --limit           limit below which the program refuses new flow in Kb/s
                    (default 0.0100 Kb/s)
-t, --timeout         duration after which a flow is considered finished
                    (default 20 s)

Options for files :
-----
-s, --file_stat       file to write stat (default '/tmp/stats')
-c, --file_socket_check file which represents socket check on local filesystem
                    (default '/tmp/check')
-y, --file_socketok   file which represents socket ok on local filesystem
                    (default '/tmp/check')

(1) Admission control will be performed between interface 1 and interface 2.
    Data will be recopied without test between interface 2 and interface 1.

```

Options

Options générales :

-d, --debug	enregistre dans syslog et affiche plus d'informations
-f, --force	permet d'écraser des fichiers existants
-h, --help	imprime un message résumant les options
-M, --min_nb_packet	nombre de paquets minimum que doit contenir un flot avant d'être affiché dans la table récapitulative
-V, --version	affiche la version du programme
-v, --verbose	affiche des informations durant l'exécution (résumé des options, tables récapitulatives des flots, ...)

Options de l'estimateur de bande passante :

-p, --port	port qui accueille la connexion d'un hôte distant sur lequel tourne l'estimateur fournissant l'information de congestion
-r --refresh	intervalle en millisecondes entre deux lectures de l'état de congestion que l'estimateur envoie

Options pour le contrôle d'admission :

-i, --int1	première interface
-o, --int2,	deuxième interface
-t, --timeout	temps d'inactivité au bout duquel un flot est considéré comme expiré

Options pour les fichiers :

-s, --file_stat	fichier pour écrire les statistiques sur les flots expirés
-c, --file_socket_check	fichier qui représente le socket check (doit se trouver sur un système de fichier local)

Bugs connus

version 1 :

Plusieurs bugs restent à corriger. Notamment sous forte charge, la carte réseau qui est connecté sur le deuxième réseau subit une remise à zéro. Le problème a été identifié et mis sur la liste des améliorations à mettre en place. Pour le moment, il faut donc choisir des cartes réseaux qui ne sont pas trop sensibles à des remises à zéro et qui effectuent celles-ci très rapidement (en particulier, il faut éviter les cartes DLINK).

version 2-3 :

Malgré le gain de performance grâce au passage en multithread il y a une grande perte de performance par rapport à un forward de packet classique effectué par le noyau ou un module. Délocalisation du module `estima`.

Soumission de bugs

Auteurs

Sebastien Michel <0683040182@orange.fr>

Denis Sacchet <denis.sacchet@libertysurf.fr>

James W. Roberts <james.robert@rd.francetelecom.com>

Nabil Benameur <nabil.benameur@rd.francetelecom.com>

Sara Oueslati-Boulahia <sara.oueslati@rd.francetelecom.com>

Copyright

Copyright ©2000 Sébastien Michel et France Télécom R&D

All rights reserved.

A.2 estima

Nom

`estima` - Application d'estimation de bande passante par le biais d'un TCP/fantôme.

Synopsys

```
estima [OPTIONS] hôte
```

Description

`estima` est une application qui permet de déterminer s'il y a ou non de la congestion sur un lien.

L'argument `hôte` spécifié dans la ligne de commande désigne une station acceptant une connexion TCP discard sur un port (par défaut le port 9). Idéalement, cet hôte se trouve sur le réseau se trouvant de l'autre coté du CAC. En effet, celui-ci va permettre d'établir une connexion fantôme et donc d'estimer la bande passante disponible. Cela n'aurait pas de sens de prendre une station se trouvant dans la même partie du réseau que la machine sur laquelle l'`estima` tourne.

L'`estima` prend des arguments supplémentaires optionnels (sinon il existe des valeurs par défaut) qui sont les limites de bande passante en dessous de laquelle les flots sont refusés.

Options

Options générales :

<code>-d, --debug</code>	enregistre dans syslog et affiche plus d'informations
<code>-h, --help</code>	imprime un message résumant les options et sort
<code>-V, --version</code>	affiche la version du programme et sort
<code>-v, --verbose</code>	affiche des informations durant l'exécution (résumé des options, tables récapitulatives des flots, ...)

Options de l'estimateur de bande passante :

<code>-f, --file</code>	Nom du fichier de sortie dans lequel <code>estima</code> écrit un historique de bande passante
<code>-g, --length</code>	taille d'une écriture
<code>-p, --port</code>	port pour la connexion sur l'hôte distant (puits)
<code>-a --alpha</code>	coefficient pour le lissage de la bande passante
<code>-D --delta</code>	intervalle en millisecondes entre deux mesures de la bande passante
<code>-l, --limit1</code>	limite de la bande passante (en Kb/s) en dessous de laquelle le programme commence à refuser les nouveaux flots
<code>-L, --limit2</code>	limite de la bande passante (en Kb/s) en dessous de laquelle le programme commence à accepter les nouveaux flots

note: Entre les deux limites la congestion n'est pas déterministe. elle est proportionnelle à la position de la bande passante entre les limites. Plus on se rapproche de `limit1` plus on estime que l'on est en congestion plus on se rapproche de `limit2` plus on estime que l'on est de plus en plus en état d'acceptation de nouveaux flots.

Bugs connus

Soumission de bugs

Auteurs

Sebastien Michel <0683040182@orange.fr>

Denis Sacchet <denis.sacchet@libertysurf.fr>

James W. Roberts <james.robert@rd.francetelecom.com>

Nabil Benameur <nabil.benameur@rd.francetelecom.com>

Sara Oueslati-Boulahia <sara.oeslati@rd.francetelecom.com>

Copyright

Copyright ©2000 Sébastien Michel et France Télécom R&D

All rights reserved.

A.3 gen

Nom

gen - générateur de trafic

Synopsys

gen [OPTIONS]

Description

gen est un générateur de trafic se basant sur un modèle de sessions. Le trafic à envoyer est découpé en sessions et les sessions elles-mêmes en pages.

Entre chaque début de session, un certain temps est attendu ; chaque session contient un certain nombre de pages ; chaque page fait une certaine taille et un certain temps existe entre deux pages.

Tous ces paramètres (intersession, nombre de pages, taille d'une page et interpage) sont tirés aléatoirement au cours de l'exécution suivant des distributions aléatoires. Ces distributions sont spécifiées dans un fichier de configuration (voir *gen_param.conf*).

Ce générateur de trafic est un émetteur. Il émet des données aléatoires vers des puits. Le programme récupère une liste des puits possibles dans un fichier de configuration (voir *gen_host.conf*).

Options

-d, --debug	Affiche plein de messages (nombre de sessions en cours, valeurs aléatoires tirées, ...)
-h, --help	Affiche une liste des options et sort
-H, --host	Spécifie le fichier contenant les puits possibles
-o, --output	Spécifie un fichier pour l'écriture des résumés des sessions
-P, --param	Spécifie le fichier contenant les paramètres des distributions aléatoires
-p, --prefix	Spécifie le préfix des fichiers de statistiques intermédiaires
-v, --verbose	Affiche un résumé de chaque session sur le terminal
-V, --version	Affiche la version du programme et sort

Auteurs

Denis Sacchet <denis.sacchet@libertysurf.fr>
James W. Roberts <james.robert@rd.francetelecom.com>
Nabil Benameur <nabil.benameur@rd.francetelecom.com>
Sara Oueslati-Boulahia <sara.oeslati@rd.francetelecom.com>

Bugs connus

Théoriquement, le programme est sensé chercher en premier le fichier de configuration spécifié en ligne de commande, puis celui contenu dans le répertoire personnel, puis le fichier par défaut. Alors que le premier et le dernier cas fonctionne très bien, la recherche du fichier contenu dans le répertoire personnel ne fonctionne pas encore.

Soumission de bugs

Copyright

Copyright ©2000 Sébastien Michel et France Télécom R&D
All rights reserved.

Voir aussi

`gen_param.conf` `gen_host.conf` `discardd`

A.3.1 gen_host.conf

Nom

gen_host.conf - fichier de configuration pour les puits possibles pour le générateur de trafic gen

Description

Ce fichier, par défaut `~/gen_host.conf` ou `/etc/gen_host.conf`, est lu par le générateur de trafic gen pour récupérer une liste des puits possibles.

Le fichier est lu ligne par ligne. La fin d'une ligne est indiquée par un retour chariot. Il existe trois types de ligne :

- les lignes de commentaires : celles-ci commencent toujours par le caractère # et il est suivi par n'importe quelle suite de caractères.
- les lignes vides : celles-ci ne contiennent que des espaces ou des tabulations.
- les lignes de puits : celles-ci contiennent le nom ou l'adresse IP d'un hôte ainsi que le plus bas et le plus haut port sur lequel le générateur peut envoyer des données. Chaque champ est séparé par autant de caractères blancs (espace et tabulation) que voulu.

Les lignes ne répondant pas à un de ces trois critères seront indiquées comme invalides par le générateur de trafic. Il est également possible de rajouter des caractères blancs au début et à la fin d'une ligne.

Bugs connus

Le générateur de trafic n'est pas capable de trouver le fichier `~/gen_host.conf` même si celui-ci est présent et conforme dans sa syntaxe.

Voir aussi

gen gen_param.conf discardd

A.3.2 gen_param.conf

Nom

`gen_param.conf` - fichier de configuration pour les distributions aléatoires utilisées dans le générateur de trafic `gen`.

Description

Ce fichier, par défaut `~/gen_param.conf` ou `/etc/gen_param.conf`, est lu par le générateur de trafic `gen` pour récupérer les distributions de chaque paramètre.

Le fichier est lu ligne par ligne. La fin d'une ligne est indiquée par un retour chariot. Il existe quatre types de ligne :

- les lignes de commentaires : celles-ci commencent toujours par le caractère `#` et il est suivi par n'importe quelle suite de caractères.
- les lignes vides : celles-ci ne contiennent que des espaces ou des tabulations.
- les lignes de section : celles-ci contiennent le nom d'une section. Il existe quatre types de sections correspondant chacun à un paramètre du générateur. Les valeurs possibles sont `intersession`, `numberofpage`, `sizeofpage` et `interpage`. Ces valeurs doivent se trouver entre crochets, sans espace entre les crochets et le nom de la section.
- les lignes de paramètre : celles-ci sont constituées du nom d'un paramètre et d'une valeur. Les valeurs possibles pour paramètres sont `distribution`, `parameter1` et `parameter2`. Suivant la valeur de `distribution`, `parameter1` et `parameter2` n'ont pas la même signification. Voir le tableau A.3.2 page 48 pour le détail.

<code>distribution</code>	<code>parameter1</code>	<code>parameter2</code>
PARETO	moyenne	forme
EXPONENTIAL	moyenne	
CONSTANT	constante	
UNIFORM	minimum	maximum

TAB. A.1 – Valeur possible du paramètre `distribution` et signification des paramètres

Toutes lignes ne répondant pas à un de ces quatre critères seront indiquées comme invalides par le générateur de trafic. Il est également possible de rajouter des caractères blancs au début et à la fin d'une ligne. Chaque mot d'une ligne peut également être séparé d'un nombre quelconque de caractères blancs.

Bugs connus

Le générateur de trafic n'est pas capable de trouver le fichier `~/gen_param.conf` même si celui-ci est présent et conforme dans sa syntaxe.

Voir aussi

gen gen_host.conf discardd

A.4 ExecGen

Nom

`cac` - Application écrite en script bash permettant de lancer et d'arrêter un jeu de générateur sur l'ensemble du réseau.

Synopsys

```
ExecGen {start|status|stop}
```

Description

Ce script utilise `rsh` pour pouvoir lancer des commandes à distance. le `rsh` doit être configuré de telle manière à ce qu'il ne soit pas nécessaire d'entrer un mot de passe (cf. man `rsh` -/etc/hosts.equiv, /etc/hosts.deny, `.rhosts`).

Les résultats de l'expérience génère plusieurs fichiers :

- * L'ensemble des chiffres de chaque flot généré dans le fichier `\$FIC_RES` (Exploitation de ce fichier par `monitor_gen` -cf. manuel `monitor_gen` A.5-).
- * Un fichier par instance de générateur stockant la sortie standard de chaque instance.

Ce script nécessite un fichier de configuration (`./ExecGenrc` ou `/etc/ExecGenrc`) voici les options nécessaires à mettre dans ce fichier :

REP_GEN	Répertoire dans lequel se trouve l'ensemble des exécutables repertoriés par système
GEN	Nom de l'exécutable dans les répertoires
REP_CONF	Répertoire dans lequel les fichiers de configuration du générateur se trouvent
FIC_PARAM	Nom du fichier contenant les paramètres nécessaires au lancement des générateurs (cf. manuel gen)
REP_RES	Répertoire dans lequel les résultats de l'expérience sont stockés.
FIC_RES	Nom du fichier regroupant l'ensemble des résultats (informations sur les flots générés) de toutes les instances de générateur.
STAT	Nom des fichier temporaires de statistique nécessaire au bon fonctionnement des générateurs.
ARCH	Si cette option est absente ou a TRUE les résultats sont archivés dans un répertoire nommé avec la date et l'heure au moment de l'arrêt de l'expérience par le biais de l'option <code>stop</code>
OPTIONS	Variable contenant l'ensemble des instances de générateur qui sont lancées ou stoppées avec un certain nombre d'informations nécessaires (machine qui héberge l'instance, nom du fichier <code>gen_host</code> rattaché à cette instance ainsi que le <code>S</code> et enfin le nom du répertoire correspondant au système de la machine hôte dans lequel l'exécutable se trouve) au bon fonctionnement des générateurs (cf. Exemple)

Options

Exemple

```
REP_GEN="/applix/CAC/generateur";
GEN="genv3.0.mon";

FIC_PARAM="gen_param";
REP_CONF="~/docs/prog/gen/exp/";
REP_RES="~/docs/prog/gen/exp/res/";
FIC_RES="res";

STAT="/tmp/stat";
ARCH=TRUE

# OPTIONS format :
# GENERATRICE_MACHINE:GEN_HOST1[,GEN_HOST2]...:S;SYSTEM_REPERTORY
# [GENERATRICE_MACHINE2:GEN_HOST1 [,GEN_HOST2]...:S;SYSTEM_REPERTORY ]...

OPTIONS="p-plum:gen_host_rambo:245;solaris2.7";

# philomene:gen_host_airelle:245;solaris2.7";
# p-pareto:gen_host_rambo:100;solaris2.7
# merida:gen_host_cuenca:100;solaris2.5.1"; philomene:ganesh,aire
# lle,ganesh,airelle;solaris2.7 mure:bambou;solaris2.5.1";
```

Bugs connus

Problème pour la génération du repertoire de résultats impliquant un problème de l'archivage.

Soumission de bugs

Suggestion de refonte de la conception du principe des répertoires de résultat et de configuration.

Permettre au script de créer des répertoires en récursif (mkdir -p) mais c'est dangereux en cas d'erreur de saisi de la part de l'utilisateur.

Auteurs

Sebastien Michel <0683040182@orange.fr>

James W. Roberts <james.robert@rd.francetelecom.com>
Nabil Benameur <nabil.benameur@rd.francetelecom.com>
Sara Oueslati-Bouhahia <sara.oeslati@rd.francetelecom.com>

Copyright

Copyright ©2000 Sébastien Michel et France Télécom R&D
All rights reserved.

A.5 monitor_gen

Nom

monitor_gen - Application de Monitoring de flots.

Synopsis

```
monitor_gen [OPTIONS]
```

Description

monitor_gen est un monitor qui permet d'exploiter le fichier \$FIC_RES d'ExecGen (cf. A.4)

Le monitor_gen prend des arguments supplémentaires optionnels (sinon il existe des valeurs par défaut).

Options

Options générales :

-f, --file	Fichier de log à monitorer (par défaut :res_gen)
-l, --limit	Limite du changement d'échelle afin de pouvoir analyser plus facilement le phase de régime continu d'une courbe de trafic en ignorant le slow start (régime transitoir)
-L, --length	Taille maximum approximative d'un flot du système
-p, --plot	Facteur permettant de régler l'échelle de la taille des cercles représentant les flots
-o, --diametre	Limite de diamètre pour un cercle représentant un flot, en dessous de cette limite le flot sera représenté par un point
-c, --color	Activation ou désactivation de la colorisation du graphique (Si par défaut elle est activé alors l'option -c la désactivera)
-v, --verbose	affiche des informations durant l'exécution
-d, --debug	Enregistre dans syslog plus d'informations
-h, --help	Imprime un message résumant les options et sort
-r, --refresh	Intervalle de rafraichissement de l'image en nombre de flot
-V, --version	affiche la version du programme et sort
-P --period	Période affiché par le graphique (par défaut 3600s) (implementation incomplète)
-s, --scale	Facteur permettant de choisir le format d'affichage du graphique (par défaut c'est fateur 2)
-D --delay	Intervalle en millisecondes entre deux lectures du fichier de log
-m, --moyinter	Taille de l'intervalle dans lequel la moyenne sera calculé.
-M, --limitemoy	Taille minimum des flot pour être considéré dans la moyenne (A noter que quelque soit la limite la moyenne reste souvent la même!)

Bugs connus

Le changement de période n'est pas censé marcher. Malgré le débogage des '*segmentation fault*' peuvent être encore apparaître lors de dépassement de buffer (sscanf ...) et lors de l'ouverture d'un fichier non existant ou non autorisé en lecture.

Soumission de bugs

Auteurs

Sebastien Michel <0683040182@orange.fr>

James W. Roberts <james.robert@rd.francetelecom.com>

Nabil Benameur <nabil.benameur@rd.francetelecom.com>

Sara Oueslati-Boulahia <sara.oeslati@rd.francetelecom.com>

Copyright

Copyright ©2000 Sébastien Michel et France Télécom R&D

All rights reserved.

A.6 monitor_estima

Nom

`monitor_estima` - Application de Monitoring de flots.

Synopsys

```
monitor_estima [OPTIONS]
```

Description

`monitor_estima` est un monitor qui permet d'exploiter le fichier de sorti d'estima (cf. A.2)

Le `monitor_estima` prend des arguments supplémentaires optionnels (sinon il existe des valeurs par défaut).

Options

Options générales :

-f, --file	Fichier de log à monitorer (par défaut :res_estima)
-l, --limit	Limite du changement d'échelle afin de pouvoir analyser plus facilement la phase de régime continu d'une courbe de trafic en ignorant le slow start (régime transitoir)
-L, --length	Taille maximum approximative d'un flot du système
-p, --plot	Facteur permettant de régler l'échelle de la taille des cercles représentant les flots
-v, --verbose	affiche des informations durant l'exécution
-d, --debug	Enregistre dans syslog plus d'informations
-h, --help	Imprime un message résumant les options et sort
-r, --refresh	Intervalle de rafraichissement de l'image en nombre de flot
-V, --version	affiche la version du programme et sort
-P --period	Période affichée par le graphique (par défaut 3600s) (implémentation incomplète)
-s, --scale	Facteur permettant de choisir le format d'affichage du graphique (par défaut c'est facteur 2)
-D --delay	Intervalle en millisecondes entre deux lectures du fichier de log
-m, --moyinter	Taille de l'intervalle dans lequel la moyenne sera calculé.
-M, --limitemoy	Taille minimum des flots pour être considéré dans la moyenne (A noter que quelque soit la limite la moyenne reste souvent la même !)

Bugs connus

Le changement de période n'est pas censé marcher. Malgré le débogage des segmentation fault peuvent être encore apparaître lors de dépassement de buffer (sscanf ...) et lors de l'ouverture d'un fichier non existant ou non autorisé en lecture.

Soumission de bugs

Auteurs

Sebastien Michel <0683040182@orange.fr>
James W. Roberts <james.robert@rd.francetelecom.com>
Nabil Benameur <nabil.benameur@rd.francetelecom.com>
Sara Oueslati-Bouahia <sara.oeslati@rd.francetelecom.com>

Copyright

Copyright ©2000 Sébastien Michel et France Télécom R&D
All rights reserved.

Annexe B

Procédure de lancement d'une expérience

Présentation

Je présente ici l'ensemble des applications dans leur contexte en déroulant un scénario d'expérience. Voici le réseau sur lequel l'expérience se déroule :

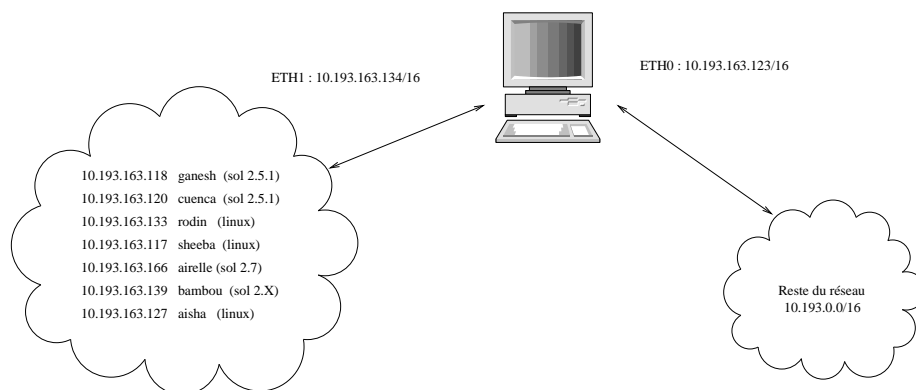


FIG. B.1 – Plate-forme

CAC

Dans un premier temps, il faut activer le contrôle d'admission entre les deux parties de réseau, pour cela il suffit d'exécuter le CAC (cf. manuel CAC A.1) sur la passerelle ainsi que bien configurer les routes de la passerelle. Sur la plate-forme actuelle il s'agit du même réseau des deux côtés de la passerelle, elle a donc une

route par défaut sur une interface ethernet et un ensemble de routes énumérant les machines se trouvant sur l'autre interface, voici un fichier d'exemple avec l'ensemble des routes actuellement sur la machine p-poisson :

```
#!/bin/sh

HOSTNAME='cat /etc/HOSTNAME'

##### LOOPBACK

echo "Configuring loopback device"
/sbin/ifconfig lo 127.0.0.1
/sbin/route add -net 127.0.0.0 netmask 255.0.0.0 lo

##### ETH0

echo "Configuring eth0 as 10.193.163.123/24 (p-poisson)"
/sbin/ifconfig eth0 10.193.163.123 \
    broadcast 10.193.163.255 netmask 255.255.255.0

echo "Add default route by 10.193.163.1 "
/sbin/route add default gw 10.193.163.1

##### ETH1

echo "Configuring eth1 as 10.193.163.134/24"
/sbin/ifconfig eth1 10.193.163.134 \
    broadcast 10.193.163.255 netmask 255.255.255.0

echo "Delete default route for eth1"
/sbin/route del -net 10.193.163.0 netmask 255.255.255.0 dev eth1

echo "Add route for p-rambo by 10.193.163.27"
/sbin/route add 10.193.163.27 gw 10.193.163.134

echo "Add route for p-51mustangb by 10.193.163.145"
/sbin/route add 10.193.163.145 gw 10.193.163.134

echo "Add route for aisha by 10.193.163.127"
/sbin/route add 10.193.163.127 gw 10.193.163.134

echo "Add route for ganesh by 10.193.163.118"
/sbin/route add 10.193.163.118 gw 10.193.163.134

echo "Add route for bambou by 10.193.163.139"
/sbin/route add 10.193.163.139 gw 10.193.163.134
```

```
echo "Add route for sheeba by 10.193.163.117"
/sbin/route add 10.193.163.117 gw 10.193.163.134

echo "Add route for rodin by 10.193.163.133"
/sbin/route add 10.193.163.133 gw 10.193.163.134

echo "Add route for cuenca by 10.193.163.120"
/sbin/route add 10.193.163.120 gw 10.193.163.134

echo "Add route for airelle by 10.193.163.166"
/sbin/route add 10.193.163.166 gw 10.193.163.134
```

voici maintenant un exemple d'une commande de lancement du CAC (sur p-poisson) :

```
/tmp/cac3 -v -F -M 20 -i eth0 -o eth1 -t 10
```

estima

Après le CAC c'est au tour d'estima d'être lancé (généralement, mais pour certaines expériences il est nécessaire de désactiver `estima`). Pour obtenir les meilleurs résultats possible, il est de lancer l'estima (le TCP/Fantôme) entre une machine d'un côté de la passerelle et une autre machine se trouvant de l'autre côté de la passerelle.

Voici maintenant un exemple d'une commande de lancement de l'estima (sur p-zzmonein) :

```
/applix/CAC/estimateur/linux/estimav2 -v -a 0.9 -l 1000 -L 2000 ganesh
```

L'option `-a` permet de lisser l'estimation de la moyenne, l'option `-l` est le premier seuil et l'option `-L` le second seuil en Kb/s `ganesh` est la machine puits ayant le port discard (cf. glossaire page 74) disponible.

execgen

Après avoir correctement rempli les `.ExecGenrc` (ce qui veut dire que vous avez bien choisi le répertoire de config où se trouve l'ensemble des fichiers nécessaires au générateur, le répertoire de résultats ainsi que le nom du fichier de sortie global de l'ensemble des générateurs.

Il suffit alors de taper la commande : `/applix/CAC/scripts/ExecGenv3.2/execgen.sh start`

monitors

Les monitors se lancent facilement mais il faut faire attention. En général le problème rencontré est le problème de droit sur les fichiers de log qui sont générés par root ou un autre utilisateur.

Voici les exemples de lancement des deux types de moniteurs :

```
/applix/CAC/monitor/linux/monitor_genv1 -f res -l 150  
/applix/CAC/monitor/linux/monitor_estimav1 -f estima.log
```

L'argument -f permet de spécifier le fichier de résultats, l'option -l permet de limiter le changement d'échelle (valeur maximale en ordonnée) afin de mieux visualiser le régime continu sans être pollué par le régime transitoire du début de l'expérience (Donne lieu à de grandes valeurs de débit par rapport au régime continu).

Auteurs

Sebastien Michel <0683040182@orange.fr>
Denis Sacchet <denis.sacchet@libertysurf.fr>
James W. Roberts <james.robert@rd.francetelecom.com>
Nabil Benameur <nabil.benameur@rd.francetelecom.com>
Sara Oueslati-Bouahia <sara.oueslati@rd.francetelecom.com>

Copyright

Copyright ©2000 Sébastien Michel et France Télécom R&D
All rights reserved.

Annexe C

Exemples de programmes multi threads

C.1 communication par sémaphores

```
void *produce(void *);
void *consume(void *);
int loopcnt = 5;

n = 1;
if (sem_init(&consumed, 0, 0) < 0) {
    perror("sem_init");
    exit(1);
}
if (sem_init(&produced, 0, 1) < 0) {
    perror("sem_init");
    exit(1);
}
if (pthread_create(&idprod, NULL, produce, (void *)loopcnt) != 0) {
    perror("pthread_create");
    exit(1);
}
if (pthread_create(&idcons, NULL, consume, (void *)loopcnt) != 0) {
    perror("pthread_create");
    exit(1);
}
(void)pthread_join(idprod, NULL);
(void)pthread_join(idcons, NULL);
(void)sem_destroy(&produced);
(void)sem_destroy(&consumed);
}

void *produce(void *arg)
```

```

{
    int i, loopcnt;
    loopcnt = (int)arg;
    for (i=0; i<(loopcnt); i++) {
        fprintf(stderr,"Prod : je produit ?");
        sem_getvalue(&consumed,&cons);
        sem_getvalue(&produced,&prod);
        fprintf(stderr," cons=%d prod=%d!\n",cons, prod);
        sem_wait(&consumed);
        n++; /* increment n by 1 */
        fprintf(stderr,"Prod : je produit 1!");
        sem_getvalue(&consumed,&cons);
        sem_getvalue(&produced,&prod);
        fprintf(stderr," cons=%d prod=%d!\n",cons, prod);
        sem_post(&produced);
        fprintf(stderr,"Prod : 1 produit!");
        sem_getvalue(&consumed,&cons);
        sem_getvalue(&produced,&prod);
        fprintf(stderr," cons=%d prod=%d!\n",cons, prod);
    }
}

void *consume(void *arg)
{
    int i, loopcnt;
    loopcnt = (int)arg;
    for (i=0; i<loopcnt; i++) {
        fprintf(stderr,"Conso : je veux consomme...");
        sem_getvalue(&consumed,&cons);
        sem_getvalue(&produced,&prod);
        fprintf(stderr,"cons=%d prod=%d!\n",cons, prod);
        sleep (1);
        sem_wait(&produced);
        fprintf(stderr,"Conso : n is %d", n); /* print value of n */
        sem_getvalue(&consumed,&cons);
        sem_getvalue(&produced,&prod);
        fprintf(stderr," cons=%d prod=%d!\n",cons, prod);
        sem_post(&consumed);
        fprintf(stderr,"Conso : J'ai consommé...");
        sem_getvalue(&consumed,&cons);
        sem_getvalue(&produced,&prod);
        fprintf(stderr,"cons=%d prod=%d!\n",cons, prod);
    }
}

```

C.2 modèle de purge à la demande

```

{

```

```

int i;
pthread_t idpurge[10]; /* ids of threads */
if (sem_init(&do_purge, 0, 0) < 0) {
    perror("sem_init");
    exit(1);
}
fprintf(stderr, "Main : Je cree le thread!\n");
if (pthread_create(&idpurge[nbt++], NULL, purge, NULL) != 0) {
    perror("pthread_create");
    exit(1);
}
fprintf(stderr, "Main : c'est parti for ever!\n");
while (1){
    sleep(1);
    fprintf(stderr, "Main : je veux purger...");
    sem_post(&do_purge);
    sem_getvalue(&do_purge, &p);
    fprintf(stderr, " nb de tread %d, nb de purge %d, purge=%d!\n", nbt, n, p);
    if (p > 10 & nbt < 10){
        fprintf(stderr, "Main : Je cree un nouveau thread!\n");
        if (pthread_create(&idpurge[nbt++], NULL, purge, NULL) != 0) {
            perror("pthread_create");
            exit(1);
        }
    }
}

for(i=0; i<10; i++)
{
    (void)pthread_join(idpurge[i], NULL);
}
(void)sem_destroy(&do_purge);
}

void *purge(void *arg)
{
while(1)
{
    sleep(3);
    sem_wait(&do_purge);
    fprintf(stderr, "Purge : blablaba\n");
    n++; /* increment n by 1 */
    sem_getvalue(&do_purge, &p);
    if (p < 3 & nbt > 1) {-nbt; pthread_exit(0);}
}
}
}

```

C.3 utilisation de mutex

```

void EndRegion()
{
    pthread_mutex_unlock(&mutex);
}

void IncrementX(char * name)
{
    int Temp; /* local variable */

    BeginRegion(); /* enter critical region */
    Temp = x;
    Temp = Temp + 1;
    if (!strcmp(name,"A")) {
        printf("ZZZZz.. Thread %s is sleeping :)\n", name);
        sleep(2);
    }
    x = Temp;
    EndRegion(); /* exit critical region */
}

void *MyThread(void *arg)
{
    char *sbName;

    sbName = (char *)arg;
    fprintf(stderr,"Thread %s want to increment X\n", sbName);
    IncrementX(sbName);
    printf("X = %d in Thread %s\n", x, sbName);
}

int main()
{
    pthread_t idA, idB; /* ids of threads */
    void *MyThread(void *);

    if (pthread_mutex_init(&mutex, NULL) < 0) {
        perror("pthread_mutex_init");
        exit(1);
    }
    if (pthread_create(&idA, NULL, MyThread, (void *)"A") != 0) {
        perror("pthread_create");
        exit(1);
    }
    if (pthread_create(&idB, NULL, MyThread, (void *)"B") != 0) {
        perror("pthread_create");
    }
}

```

```
    exit(1);
}
(void)pthread_join(idA, NULL);
(void)pthread_join(idB, NULL);
(void)pthread_mutex_destroy(&mutex);
return 0;
}
```


Annexe D

Glossaire

Adresse MAC : C'est une valeur sur 48 bits permettant d'identifier de façon unique un élément actif dans un réseau Ethernet. Pour information, les 24 premiers bits permettent d'identifier le constructeur du composant, les 24 derniers bits étant totalement aléatoires.

ARP : En Anglais, Address Resolution Protocol, ou en Français, Protocole de Résolution d'Adresse. ARP est un protocole qui permet d'associer à une adresse MAC (donc unique), une ou plusieurs adresses IP (l'adresse IP étant une adresse logique, une même station, possédant une seule adresse MAC, peut se voir attribuer plusieurs adresses IP). Une trame Ethernet dite de broadcast (c'est-à-dire à destination de toutes les stations), est envoyée sur le réseau et est capturée par toutes les stations. La station qui reconnaît son adresse IP renvoie une trame de réponse avec son adresse MAC au demandeur.

Bus Ethernet : Support physique sur lequel viennent se brancher les stations. Dû à la particularité du protocole d'accès CSMA/CD, ce bus ne peut pas dépasser une certaine distance (calculable en fonction du débit et de la longueur minimale de la trame) sans rajouter des équipements spécialisés (répéteur, commutateurs). Une distance minimale doit également être respectée entre deux stations. Un bus Ethernet peut également s'appeler un brin Ethernet ou domaine de collision. La dénomination bus provient du premier support physique des réseaux Ethernet, le câble coaxial. Aujourd'hui, la plupart des réseaux Ethernet possèdent une topologie en étoile grâce aux concentrateurs ou commutateurs, ces éléments faisant alors office de bus. La dénomination domaine de collision provient du protocole d'accès CSMA/CD. C'est l'ensemble des stations qui peuvent émettre et donc provoquer des collisions sur un support physique.

CBQ : En Anglais, Class Based Queuing, ou en Français, file d'attente basée sur des classes. À l'origine CBQ a été développé pour la gestion de trafic réseaux sur les routeurs, les files d'attente qui permettent la gestion de

trafic sont utilisés en nombre fixe (typiquement moins de 10) et gérées de manière FiFo (First in, First out). Le trafic est injecté dans la file d'attente dans laquelle il attend que le reste du trafic finisse la transmission. Si le protocole TCP est utilisé, l'émetteur attendra un acquittement avant de renvoyer les autres paquets ; si un autre protocole est utilisé alors les paquets s'accumuleront dans la file d'attente. S'il y a une grande différence sur la vitesse du réseau, ou si la file d'attente est trop (congestion) ou pas assez (famine) pleine, alors le comportement du système a, en général, un comportement négatif. Plusieurs formes de file d'attente incluant le CBQ (avec la classification, les priorités entre les différentes files d'attente) existent. Les files d'attente fournissent une approche limitée et statique de la classification du trafic réseau, tout dépend quelle technologie de file d'attente est utilisée.

Discard : protocole correspondant à la RFC 863, en TCP ce protocole correspond à un puit ; un client envoie des données et le serveur les reçoit sans toute fois envoyer de réponse. Et ceci jusqu'à ce que la connexion soit interrompue par le client.

Domaine de collision : voir *Bus Ethernet*.

Ethernet : Technologie de réseau local reposant sur le protocole CSMA/CD, normalisé par l'IEEE. Cette technologie est très répandue en raison de sa facilité et son faible coût d'installation. L'adressage repose sur les adresses MAC. Toutes les machines possèdent une adresse MAC unique. La trame Ethernet contient l'adresse MAC source (station émettrice de la trame) et l'adresse MAC destination (station ou groupe de stations destinataire de la trame). Chaque station reçoit toutes les trames (en raison du CSMA/CD et de l'écoute du support physique), elle compare alors sa propre adresse MAC à l'adresse MAC destination contenu dans la trame, si cela concorde, elle capture la trame, sinon elle ne fait rien.

fork : Le fork est une technique de duplication de processus dans le monde UNIX.

GDK : GIMP Drawing Kit. Librairie incluse dans GTK permettant de faire des dessins (Rectangles, cercles, points...).

GTK : GIMP ToolKit. Librairie de développement permettant de créer des GUI (fenêtres, boutons, menus...).

GUI : Graphic User Interface, en français, Interface homme/machine Graphique. Permet de faciliter l'interactivité entre les utilisateurs et la machine.

IP : En Anglais, Internet Protocol, ou en Français, Protocole Internet. Le protocole IP est chargé du réseau (niveau 3 du modèle OSI) dans la pile TCP/IP. Actuellement la version 4 est utilisée pour l'adressage sur Internet ainsi que dans les entreprises ; il existe également la version 6 qui est en cours de déploiement. Pour la version 4, l'adressage se fait sur 32 bits. Ce protocole assure une gestion de la fragmentation et est utilisé

conjointement avec le protocole TCP ou le protocole UDP (voir figure D.1 page 75).

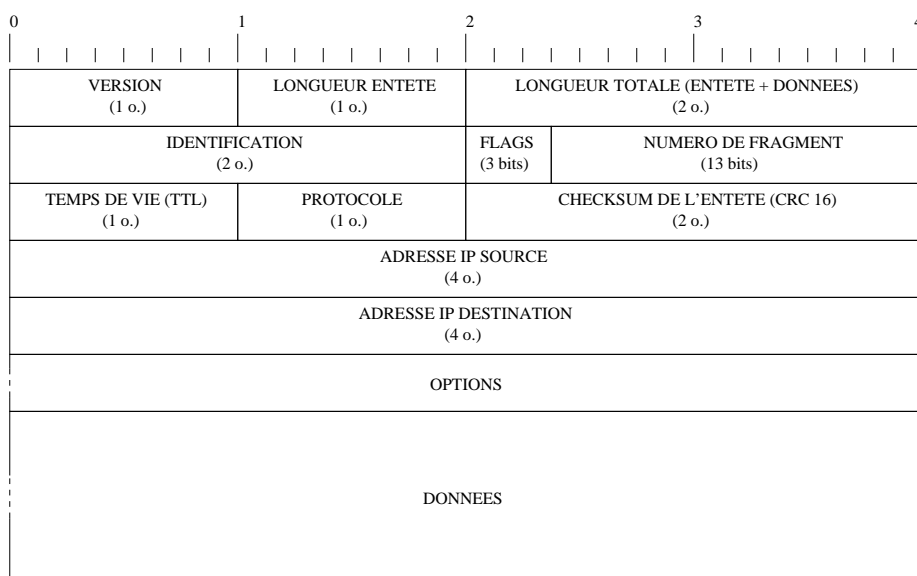


FIG. D.1 – Structure d'un paquet IP

ipchains : Ipchains est utilisé pour configurer, maintenir et vérifier les règles de firewall IP qui se trouvent dans le noyau Linux. Ces règles peuvent être divisées en quatre catégories : Les chaînes IP entrantes, les chaînes IP sortantes, les chaînes IP forwardées, et les chaînes définies par l'utilisateur.

iptables : Iptables a le même rôle qu'ipchains mais il a des fonctions plus avancées telles que le marquage de paquet.

Lien : Un lien (ou lien réseau) est la jonction entre deux nœuds (cf. définition page 76) d'un réseau.

Mode promiscuous : Quand une carte réseau est en *mode promiscuous*, elle capture toutes les trames passant sur le support physique et court-circuite donc la comparaison avec sa propre adresse MAC. Cela permet d'observer tout le trafic passant sur le réseau.

MTU : En Anglais, Maximum Transfer Unit, ou en Français, Unité Maximale de Transfert. C'est la taille maximale des données transportables dans une trame Ethernet. Cette valeur est comprise forcément entre 46 et 1500 pour une interface physique. Dans le cas des interfaces virtuelles (comme la boucle locale), des valeurs plus élevées peuvent être trouvées.

Modèle OSI : Modélisation des protocoles réseaux normalisés. Ce modèle comporte 7 couches (voir figure D.2 page 76), chacune ayant un rôle particulier dans la chaîne de transmission. Les données partent de la

couche 7 et à chaque étape, elles subissent des actions, en général, fractionnement, ajout d'une entête ... Les données se retrouvent donc dans une sorte de *structure en oignon* qui sera *pelé* par le receveur afin de récupérer l'information utile (voir figure D.3 page 76).

	Modèle OSI		Modèle TCP/IP
7	Application	-----	Applications (telnet, ftp, smtp, dns, ...)
6	Présentation		
5	Session		
4	Transport	-----	Transport (TCP, UDP ...)
3	Réseau	-----	Réseau (IP)
2	Liaison	-----	Liaison/Physique (Arpanet, Ethernet ...)
1	Physique	-----	

FIG. D.2 – Comparaison du modèle TCP/IP au modèle de référence OSI

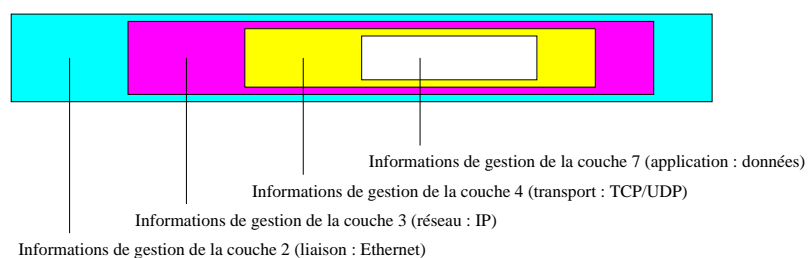


FIG. D.3 – Illustration de la structure en oignon d'une trame

NAT : En Anglais, Network Address Translation, ou en Français, Translation d'Adresse Réseau. Mécanisme par lequel une station est capable de transformer l'adresse source d'un paquet pour faire croire à la destination que le paquet vient d'autre part. Ce mécanisme est notamment utilisé dans les réseaux ne possédant pas d'adresse IP routable ou seulement une seule, pour *masquer* les machines internes.

NGN : Next Generation Network, concept d'un réseau unique permettant le transport de l'ensemble des services voix et données

Noeud : Le noeud réseau est un endroit de rencontre de plusieurs liens. Matériellement c'est un hub, un switch, un routeur, une passerelle...

Noyau : Le noyau d'un système d'exploitation est le cœur de celui-ci. C'est lui qui gère la mémoire vive, les interruptions, les entrées/sorties, les périphériques ... Il prend également en charge les couches réseaux.

Open Source : "En 1998, il est devenu plus difficile de sensibiliser les nouveaux utilisateurs à la notion de liberté dans le logiciel, quand une portion de notre communauté a choisi d'arrêter d'utiliser le terme "Free Software" pour lui préférer la dénomination "Open Source software"

N.d.T. : encore et toujours cette ambiguïté de la langue anglaise. "software" signifie "logiciel". "free" signifie à la fois "libre", sens qui est pertinent ici, et "gratuit", qualité qui n'est qu'un effet de bord des logiciels libres. "open source" signifie "dont le code source est ouvert".", extrait du site www.gnu.org

Pipe : En Français *tuyaux*, ce sont des canaux de communication qui sont utilisés dans les systèmes Unix pour communiquer entre deux processus. L'écriture et la lecture se font séquentiellement, il n'y a pas de gestion de paquets. Si deux écritures successives ont lieu, les données peuvent être récupérées avec une seule lecture.

Pont : Equipement actif d'un réseau agissant au niveau 2. Il sert en général à séparer les domaines de collision d'un réseau Ethernet afin d'augmenter les performances. Il tient à jour une table des adresses MAC présentes sur chacun des deux brins Ethernet qui lui sont reliés. Quand il reçoit une trame d'un des brins, il est émis sur l'autre brin seulement si l'adresse MAC de destination n'est pas présente dans la table du premier brin. Cela permet de limiter le nombre de trames inutiles sur l'un ou l'autre brin et donc d'augmenter très sensiblement les performances

Proxy : Station Serveur proxy : Serveur de proximité ou passerelle internet, qui agit comme un firewall (filtre), gérant le trafic entre un réseau privé et Internet. Physiquement, c'est un vaste espace disque localisé chez le provider qui sert à stocker les pages Web les plus consultées par les abonnés pour qu'elles s'affichent plus rapidement. Un proxy ne s'avère efficace qu'avec un nombre restreint d'utilisateurs. Un proxy est surtout utile pour économiser de la bande passante (le débit vers Internet offert aux abonnés). Enfin, un proxy demande des temps d'accès plus longs pour obtenir une page Web.

Son rôle est le partage des sessions de communication vers Internet (et non le partage des ressources de communication comme plusieurs modem) et il autorise à plusieurs utilisateurs de partager un seul canal de communication (et donc un seul modem) pour accéder au Web.

Le serveur proxy établit la communication avec le provider. C'est au serveur proxy que sont transmises les requêtes émises par les utilisateurs. En retour, les réponses en provenance d'internet transitent par ce serveur avant d'arriver sur la station de travail à l'origine de la requête. Sur le plan technique le serveur proxy transforme les demandes en provenance du réseau local (adresses IP privés) en demandes avec sa propre adresse IP internet. Cette translation d'adresse IP est appelée NAT (Network

Adress Translator). Cette translation d'adresse isole totalement le réseau local du réseau Internet. Les solutions serveurs proxy sont soit matérielles, soit logicielles (Unix, NetWare, Windows NT, et même Windows 95).

QoS : Quality of Service, en français, qualité de service.

Mode Raw : correspondent au domaine PF_PACKET pour récupérer les trames au niveau Ethernet (type SOCK_RAW). Ce type de socket, très puissant, permet de récupérer des trames entières et de les réémettre telles quelles. Pour peu que les fonctions adéquates soient programmées, il est également possible d'émettre des trames construites de toutes pièces.

Routeur : Equipement actif d'un réseau agissant au niveau 3. Il sert à interconnecter des réseaux avec des domaines d'adressage différents. Si une station veut envoyer des informations à une autre station qui n'est pas dans son espace d'adressage, elle passe cette information à son routeur qui s'occupe de la transmettre, soit directement à la destination s'il est connecté directement à son réseau, soit à un routeur voisin dont il sait qu'il sera capable d'atteindre la destination.

RTT : Round-Trip Time, temps d'aller et retour d'un signal réseau. Le débit d'un transfert dépend énormément de la RTT entre les deux machines.

Sémaphore : Système de signalisation utilisé en multithread permettant la communication entre deux threads (système de demande/réponse).

Socket : Notion de programmation réseau introduite sur les systèmes BSD. C'est un ensemble de routines plus ou moins standardisées permettant d'envoyer et de recevoir des données dans un réseau. Les sockets fonctionnent par domaine (par exemple, INET pour les sockets TCP/IP, ATM pour les sockets ATM, UNIX pour la communication inter-processus ...) et par niveau (STREAM ou DGRAM pour le domaine INET permettent d'obtenir une connexion TCP ou UDP ...). Ces paramètres sont passés lors de la création du socket et permettent d'accéder à la pile de protocoles à peu près comme on le désire.

TCP/IP : Pile de protocoles définissant des règles de communication entre stations. Elle s'inspire du modèle OSI pour le rôle des différentes couches. Les protocoles sous-jacent à cette pile sont les protocoles IP, UDP, TCP ainsi que ARP et RARP pour pouvoir utiliser TCP/IP sur un réseau Ethernet (voir figure D.2 page 76).

TCP : En Anglais, Transmission Control Protocol, ou en Français, Protocole de Contrôle de Transmission. Le protocole TCP est un des protocoles chargés du transport (niveau 4 du modèle OSI). Il permet de garantir une communication fiable entre deux hôtes en garantissant l'ordre des paquets. C'est un protocole orienté connexion qui possède également un contrôle de flux. Ce protocole utilise un système de ports pour multiplexer les applications entre deux mêmes hôtes. A chaque port correspond un service (voir figure D.4 page 79).

Thread : Une thread (appelée aussi processus léger ou activité) est un fil d'instructions (un chemin d'exécution) à l'intérieur d'un processus. Contrairement aux processus, les threads d'un même processus partagent le même espace d'adressage, le même environnement (par exemple les mêmes variables d'environnement, des mêmes données, etc.). Ainsi 2 threads d'un même processus communiquent beaucoup plus facilement que 2 processus. Par contre les threads ont leur propre compteur ordinal, leur propre pile. Les programmes qui utilisent plusieurs threads sont dits multithreadés.

UDP : En Anglais, User Datagram Protocole, ou en Français, Protocole de Datagramme Utilisateur. Le protocole UDP est un des protocoles charges du transport (niveau 4 du modèle OSI), l'autre étant TCP. C'est un protocole non-fiable, ne garantissant pas la livraison des paquets, ni leur ordre. C'est un protocole non orienté connexion. Ce protocole utilise un système de ports pour multiplexer les applications entre deux mêmes hôtes. A chaque port correspond un service (voir figure D.5 page 80).

Trame : Plus petit ensemble logique circulant sur le support physique. Une trame est constitué d'un ensemble de bit représentant une information à destination d'une station ou d'un ensemble de stations. Une partie de trame ne sert à rien, il faut avoir l'ensemble pour récupérer l'information, c'est pour cela que c'est le plus petit ensemble logique. Une trame possède une structure bien particulière, elle est découpée en champs. De façon générale, on retrouve une partie pour l'adressage (source et destination), une partie pour les données et une partie redondante (pour vérifier l'intégrité de la trame). Suivant le support

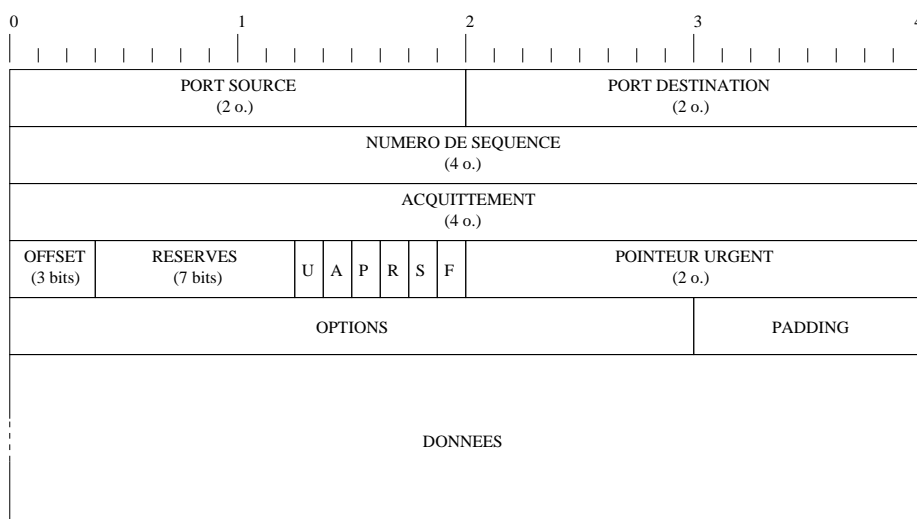


FIG. D.4 – Structure d'un paquet TCP

physique, une partie en début de trame permet souvent de synchroniser l'horloge de la carte réceptrice sur l'horloge de l'émetteur, c'est le préambule (pour le détail de la trame Ethernet, voir figure D.6 page 80).

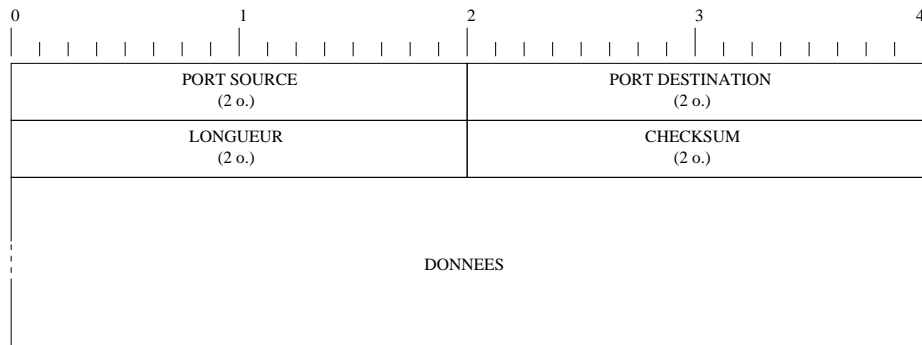


FIG. D.5 – Structure d'un paquet UDP

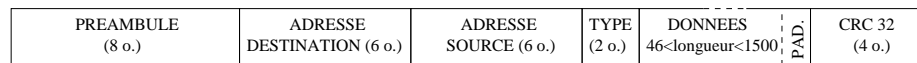


FIG. D.6 – Structure d'une trame Ethernet

Bibliographie

- [1] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, *Modeling TCP throughput : A simple model and its empirical validation*, In proceedings of SIGCOMM, Vancouver, Canada, 1998
- [2] S. Savage *Sting : a TCP-based Network Measurement Tool*, In proceedings of the 1999 USENIX Symposium on Internet Technologies and Systems, pp. 71-79, Boulder, CO, October 1999
- [3] S. Ben Fredj, S. Oueslati-Boulahia, J.W. Roberts, *Measurement-based admission control for elastic traffic*, In proceedings of ITC 17, Salvador, Brazil, September 2001
- [4] N. Benameur, S. Ben Fredj, F. Delcoigne, S. Oueslati-Boulahia and J.W. Roberts *Integrated Admission Control for Streaming and Elastic Traffic*, In proceeding of QofIS 2001 organised by COST 263, Coimbra, Portugal, September 2001.
- [5] D. Sacchet *Rapport de Stage I2, France Telecom R&D*, September 2001.
- [6] K. McCloghrie, M. Rose *Request for Comment 1156 - Management Information Base for Network Management of TCP/IP-based internets*, May 1990.