Flow-Aware Networking (FAN)

L'assurance de garanties de service pour le trafic requiert la compréhension de la relation qui existe entre la capacité du réseau, la demande, et la performance obtenue. Cette relation peut-être établie en considérant une approche du trafic au niveau flot, qui hérite des études du télétrafic pour le réseau téléphonique. L'introduction de mécanismes "Flow-Aware" que nous présentons dans ce chapitre semble une alternative désirable aux architectures classiques et complexes de qualité de service, et suffisante malgré sa simplicité afin de satisfaire les besoins des applications véhiculées par le réseau.

La proposition *Flow-Aware Networking* (FAN) est réalisée pour le cœur de réseau au travers de l'architecture de routeur Cross-Protect. Elle consiste en l'association d'un ordonnancement équitable par flot (*fair queueing*), et d'un contrôle d'admission implicite par flot. Cette combinaison permet d'assurer la performance à la fois des applications temps-réel et des transferts de données, sans nécessiter la distinction de classes de service, ni la propagation de spécifications de trafic par signalisation. L'introduction de Cross-Protect est transparente pour les utilisateurs notamment parce qu'elle conserve l'interface *Best-Effort* du réseau.

Après la présentation de haut niveau faite dans le chapitre précédent, nous entrons ici plus en détail dans le fonctionnement de *Cross-Protect*, et nous décrivons les modèles de trafic sous-jacents. Ces derniers permettent la compréhension de la performance des réseaux intégrant différent types de flux applicatifs. Les résultats pour un lien isolé sont d'abord introduits, puis étendus à un contexte de réseau. Les conditions nécessaires à leur intégration sont ensuite explicitées. Les résultats présentés dans ce chapitre montrent comment l'architecture s'intègre au sein d'une démarche saine d'opération d'un réseau, fondée sur l'instrumentation, le dimensionnement et la gestion en temps réel des ressources.

Les travaux de cette thèse – qui s'appuient sur les hypothèses et les modèles de trafic introduits ici – s'inscrivent dans la continuité des travaux entrepris autour de FAN, qui ont pour objectif la conception d'une architecture de routeur efficace, robuste et correctement dimensionnée. A la fin de ce chapitre, nous présentons l'environnement de simulation et la plateforme d'expérimentation GNU/Linux que nous avons réalisés afin de supporter nos résultats, et de tester et démontrer les mécanismes proposés.

Sommaire

Contributions :

- Extension de l'algorithme PFQ (*Priority Fair Queueing*) pour les besoins des travaux réalisés lors de cette thèse (ajout de nouveaux indicateurs de congestion).
- Réalisation d'un ensemble de modules pour le simulateur de réseau NS-2, parmi lesquels les mécanismes *Cross-Protect* (PFQ + Contrôle d'admission), les modèles de trafic utilisés, etc.
- Dépôt logiciel : *Implémentation sur noyau Linux des mécanismes Cross-Protect V1.0*
- Réalisation d'une plateforme de test Cross-Protect dans un environnement GNU/Linux ayant permis la démonstration de la faisabilité de l'architecture et ses avantages, intégrant des outils de génération de trafic, ainsi que d'analyse et de visualisation des résultats.

Démonstrations :

- Démonstration interne France Télécom
- Démonstration lors d'un séminaire France Télécom sur le trafic (25/03/2005)

3.1 Introduction

3.1.1 Ressources – Demande – Performance

Flow Aware Networking (FAN) [118] propose une approche de la QoS en rupture avec les architectures classiques présentées dans le chapitre précédent. FAN propose de caractériser et de contrôler le trafic au niveau des flots utilisateurs ; on en trouve les prémisses dans les travaux de Roberts *et al.* ([132] et suivants), qui insistent sur la nécessité de caractériser la relation liant les ressources du réseau, la demande utilisateur, et la performance obtenue.

ressources : il s'agit principalement de la capacité disponible sur les différents liens, et du partage réalisé entre les flots. On parle d'allocation de ressources. Les buffers au sein des routeurs IP sont également une ressource importante qu'il convient de considérer, comme nous le verrons dans le Chapitre 4.

demande : la demande en trafic correspond au trafic généré par les utilisateurs qui consomment les ressources du réseau. On peut la caractériser par le type et les caractéristiques essentielles des différents flots qui la constitue. De nombreuses analysent considèrent un nombre fixe de flots permanents ; Roberts et Massoulié [132] ont montré que le trafic Internet est constitué d'une superposition dynamique de flots de taille finie, rythmée par les arrivées de nouveaux flots, et les départs de flots dont le transfert se termine. La demande est généralement caractérisée lors de la période de pointe, en vue du dimensionnement des ressources. Nous verrons plus loin que la connaissance d'un faible nombre de critères comme la charge globale (le "volume" de trafic) et le débit crête des flots peut être suffisante afin d'en déduire la performance des flots.

performance : la performance est une mesure de la qualité, des garanties de service fournies au trafic. La nature aléatoire de ce dernier nous mène à considérer des garanties probabilistes, correspondant aux besoins des différents types de flots : on s'intéressera à l'espérance du débit (ou au temps moyen de réponse) pour les flots élastiques, ainsi qu'à la probabilité de perte ou au délai des paquets *streaming*.

La présence d'un mécanisme de contrôle d'admission permet l'assurance de garanties de services (minimum) à chacun des flots admis. Il est alors censé de s'intéresser à des métriques telles que le taux de blocage des flots.

3.1.2 Modèles de trafic

Les sections qui suivent présentent un ensemble de modèles de trafic permettant l'étude de la performance des flots *streaming* et élastiques. Ces modèles de trafic forment la base d'une *théorie du trafic internet*. Les modèles les plus simples permettent une compréhension qualitative de la performance réalisée par les flots, ainsi que de l'impact de l'introduction de nouveaux mécanismes. Des versions plus raffinées, que nous référencerons uniquement, permettront par contre de meilleurs prédictions. Les chapitres suivants s'appuieront sur de tels résultats.

Une caractéristique essentielle de ces modèles est leur propriété d'insensibilité : la performance obtenue ne dépend pas de caractéristiques détaillées du trafic (par exemple la connaissance de la distribution de la longueur des flots). On trouvera ainsi de nombreuses analogies avec les résultats établis pour le réseau téléphonique (télétrafic), dont la formule d'Erlang offre une très bonne illustration.

Cette formule, proposée par Erlang en 1917, permet de calculer la probabilité de blocage $B(\cdot)$ d'un appel en fonction du nombre de circuits n et de la demande moyenne E générée par les appels (taux d'arrivée des appels x temps moyen de communication) :

$$B(E,n) = \frac{\frac{E^n}{n!}}{\sum_{i=0}^{n} \frac{E^i}{i!}}$$

Elle nécessite seulement que les arrivées des appels suivent un processus de Poisson, et possède la propriété d'être insensible à la distribution de la longueur des appels. C'est la raison pour laquelle elle peut toujours être utilisée pour le dimensionnement des réseaux téléphoniques actuels, malgré l'évolution des usages et de leurs implications sur le trafic généré.

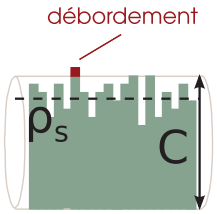
3.2 Modélisation du trafic streaming

Lors de leur génération par les applications audio et vidéo, les flots *streaming* possèdent des profils de débits divers : on trouve des flots à débit constant (CBR, *Constant Bit Rate*) ou variable (VBR, *Variable Bit Rate*). Les principales applications possèdent des débits limités ($<$ 64kb/s pour la téléphonie, 4 à 16Mb/s pour les flux vidéos, etc.) relativement faibles par rapport aux capacités des liens considérés pour le cœur de réseau. Les garanties de QoS pour ces flots s'expriment au niveau paquet, par de faibles délais et un taux de pertes négligeable.

Plusieurs modèles proposent de contrôler la charge moyenne du trafic afin qu'elle ne dépasse pas la capacité du lien. La gigue acquise par les flots est problématique : les multiplexages successifs au sein des files d'attente des routeurs créent des rafales de paquets qu'il est difficile de caractériser. Ces modèles ne sont pas adaptés pour offrir des garanties de service, sauf à considérer des situations très conservatrices avec des tailles de buffer surdimensionnées. Nous considérons ce problème de dimensionnement de buffer dans le Chapitre 4.

Nous nous intéressons ici à une proposition alternative n'utilisant pas de buffer, sauf pour absorber les arrivées simultanées de paquets ou les excédents transitoires de trafic.

3.2.1 Multiplexage statistique "sans buffer" : cas d'un lien isolé



Le modèle de multiplexage dit *sans buffer*, *Rate Envelope Multiplexing* [138, 19], propose l'utilisation d'un contrôle d'admission afin de conserver la transparence du lien. En d'autres termes, la probabilité que la charge du trafic *streaming* ρ_s dépasse la capacité du lien C doit rester négligeable : $P[\rho_s > C] \leq \epsilon$, comme illustré sur la figure ci-contre. Les délais de paquets restent alors faibles, et la probabilité de perte l est donnée approximativement par : $l = E[(\rho_s - C)^+]/E[\rho_s]$. Elle dépend uniquement de la distribution stationnaire du débit d'entrée et n'est pas sensible à d'autres caractéristiques détaillées du processus d'arrivée de paquets (telles que les corrélations lors des rafales).

Le taux d'utilisation atteint est plus faible que si l'on se permettait d'utiliser le buffer. Il reste toutefois très satisfaisant dès lors que l'agrégat de trafic présente une variabilité limitée, lorsque les flots ont des débits faibles par rapport à la capacité du lien (par exemple moins de 1% de C). C'est en pratique le cas sur un réseau comme l'Internet, et la capacité restante peut avantageusement être utilisée pour le transfert des flots élastiques.

Contrôle d'admission pour les flots *streaming*

Les algorithmes de contrôle d'admission basés sur des mesures sont particulièrement adaptés à une telle réalisation, puisqu'ils peuvent se contenter de descripteurs de trafic minimums. L'algorithme présenté dans Gibbens *et al.* [62] requiert par exemple uniquement la connaissance du débit crête du flot, associée à une estimation de la charge moyenne du lien. Un très grand nombre d'algorithmes ont été proposés, et nous passons en revue les plus adaptés à notre contexte dans le Chapitre 6, qui considère un réseau intégrant les flots *streaming* et élastiques.

Dimensionnement

Considérons des flots de débit r et de durée moyenne σ_s arrivant sur le lien selon un processus de Poisson d'intensité λ_s. La demande moyenne en trafic est $\rho_s = \lambda_s \sigma_s r$ (en b/s). En supposant un

contrôle d'admission limitant le nombre de flots, la probabilité de blocage p_b dans ce modèle est alors approximée par la formule d'Erlang, où ρ_s/r erlangs de trafic sont offerts à C/r circuits :

$$p_b = B(\rho_s/r, C/r) \qquad \text{avec} \qquad B(E,m) = \frac{E^m/m!}{\sum_{i=0}^{m} E^i/i!} \tag{3.1}$$

Il est alors possible de déterminer la capacité C nécessaire pour un blocage minimal. Des généralisations de ces modèles existent pour des débits d'accès différents, ou des flots à débit variable, voir par exemple [131].

3.2.2 Garanties de performance de bout en bout

Bonald *et al.* [33] montrent que lorsque les flots *streaming* ont un débit crête limité et qu'ils sont traités en priorité non-préemptive, il est possible de donner des bornes statistiques sur les délais et la gigue, qui dépendent alors uniquement de la charge de ces flots. Notamment, ils n'acquièrent jamais une gigue suffisante pour avoir une performance moins bonne que des arrivées de paquet de taille MTU suivant un processus de Poisson (conjecturé). Il est ainsi possible en limitant la charge des flots par un contrôle d'admission sur chaque lien, de garantir une performance convenable de bout en bout.

3.3 Modélisation du trafic élastique

Les modèles de partage statistique de bande passante (*Statistical Bandwidth Sharing*) permettent de mieux comprendre comment les flots TCP se partagent les ressources disponibles, ainsi que l'impact des caractéristiques du trafic sur la performance obtenue. Dans cette section, nous suivons l'approche de Roberts [134] qui propose une vue d'ensemble de ces modèles et de leur application à l'étude des réseaux de données. Certains résultats permettent d'expliquer le comportement des connexions TCP au niveau paquet. Padhye [120] a par exemple montré que le débit moyen d'une connexion TCP est lié au taux de pertes subi. Cependant, l'abstraction au niveau flot présente l'avantage de simplifier l'analyse, notamment au vu des propriétés d'autosimilarité du trafic, et de s'abstraire d'une version précise de TCP.

3.3.1 Modèle de sessions Poisson

Alors qu'il est reconnu que les arrivées de sessions suivent un processus de Poisson [125], ce n'est généralement pas le cas des arrivées de flots. Bonald *et al.* [32, 15] ont montré que l'on peut se ramener à un tel contexte pour l'étude de la performance du trafic. Ils considèrent pour cela un modèle assez général de sessions, constituées d'une alternance de flots et de périodes d'inactivités, de distributions générales, et où les arrivées de flots peuvent éventuellement être corrélées. Il est par exemple possible de considérer des distributions à queue lourde pour le nombre de flots par session, ce qui contribue à générer un trafic autosimilaire [98, 124].

3.3.2 Modèle processeur partagé : cas d'un lien isolé

Le modèle à processeur partagé, noté PS pour *Processor Sharing*, permet de modéliser le partage des ressources réalisé par TCP au niveau flot. Il suppose un partage instantané et équitable. En réalité, le partage réalisé par TCP n'est pas instantané (lenteur de l'algorithme AIMD), et est biaisé par le démarrage en mode *slow start* et des différences de RTT entre les connexions. Ce modèle illustre cependant bien l'impact du nombre de flots en cours sur la performance.

Nous verrons dans le Chapitre 4 comment adapter ce modèle au cas où la performance de TCP est affectée par la présence de petits buffers. Le Chapitre 5 montrera dans quelle mesure l'introduction du *fair queueing* dans le réseau permet de relaxer ces hypothèses.

Propriété d'insensibilité

Considérons un lien de capacité C partagé par un ensemble de flots. La taille et la constitution de cet ensemble varient dans le temps au fil des arrivées et départs des divers flots. Lorsque les flots sont générés selon le modèle de sessions Poisson, on peut montrer que les mesures de performance telles que le débit moyen des flots sont insensibles à des caractéristiques détaillées du trafic comme la distribution de la taille des flots, ou le nombre de flots par session. Les caractéristiques essentielles du

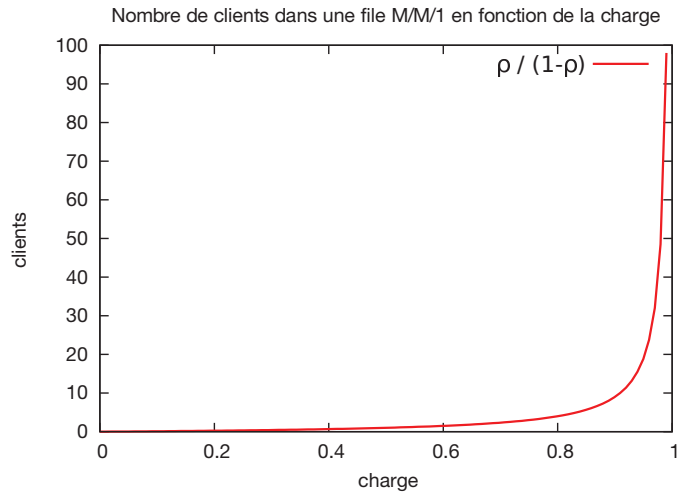


FIGURE 3.1 – Nombre moyen de flots en cours dans une file M/M/1, en fonction de la charge ρ.

trafic sont la **charge du lien** ρ, égale au taux d'arrivée des flots x taille moyenne des flots / capacité du lien, et le **débit crête des flots**, qui est le débit maximum qu'un flot peut atteindre indépendamment du lien considéré. Une telle propriété d'insensibilité n'est pas garantie dans le cas FIFO.

Composition du trafic

Lorsque tous les flots peuvent individuellement atteindre la capacité du lien, le nombre de flots en cours dans le modèle fluide PS idéal correspond à un processus de naissance et de mort. Il possède une distribution géométrique de moyenne $\rho/(1-\rho)$. représentée sur la figure 3.1.

Malgré la simplicité du modèle, c'est une bonne indication que le nombre de flots en cours à n'importe quel instant est généralement plutôt faible (par exemple moins de 20 avec une probabilité de .99 pour une charge de 80%). Or, en pratique, le nombre de flots observé sur un lien de cœur de réseau est nettement plus élevé (jusqu'à plusieurs dizaines de milliers sur un lien gigabit), ce qui induit que **la plupart ne sont pas *bottlenecked***. En effet, de très nombreux flots sont contraints par leur débit d'accès, par un lien saturé ailleurs dans le réseau, ou encore parce qu'ils ne parviennent pas à quitter le mode *slow start* et donc n'atteignent pas leur débit crête.

Le nombre de flots *bottlenecked* est très grand seulement lorsque la charge du lien est proche de 1. Une confirmation empirique du nombre limité de flots *bottlenecked*, indépendamment de la capacité du lien, est présentée dans Kortebi *et al.* [94], où les auteurs analysent plusieurs traces de trafic. Ce résultat renforce les premières observations de Suter *et al.* [146] qui montrent que la population des flots est très dynamique dans le buffer.

Débit des flots

Nous illustrons la performance du partage statistique de bande passante par la mesure de débit des flots γ = taille moyenne des flots / durée moyenne des flots. La mesure γ peut également être interprétée comme l'espérance du débit instantané d'un flot [26]. Nous considérons un modèle PS généralisé où les flots partagent un taux de service dépendant du nombre de flots en cours. Nous notons $\phi(i)C$ le taux de service du lien lorsque i flots sont en cours, et $\Phi(i) = \prod_{j=1}^{i} \phi(j)$. Nous avons alors :

$$\gamma = \frac{\sum \rho^i / \Phi(i)}{\sum i\rho^{i-1} / \Phi(i)}. \tag{3.2}$$

La figure 3.2 représente γ en fonction de ρ lorsque les flots n'ont aucune contrainte de débit crête ($\phi(i) = 1$), ainsi que lorsqu'ils possèdent un débit crête limité $p = 0.1C$ ($\phi(i) = \min(ip, 1)$). **Lorsque les flots ont un débit crête limité p, le lien est transparent pour la performance du débit jusqu'à de fortes charges (proches de $1 - p/C$).**

Contrôle d'admission pour les flots élastiques

Lorsque la charge du lien est trop proche de 1, le nombre de flots en cours augmente rapidement et les débits tendent vers 0 (figure 3.2). Le comportement de la file d'attente est instable et le temps de

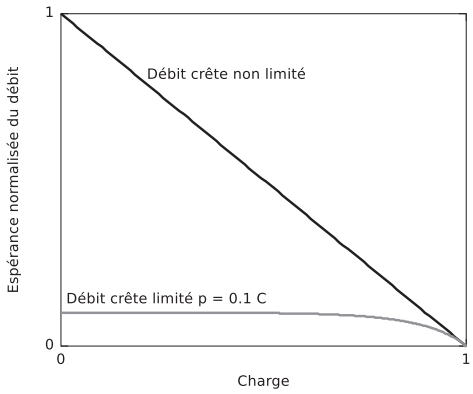


FIGURE 3.2 – Débit normalisé des flots γ en fonction de la charge ρ lorsque les flots n'ont aucune limite de débit, et lorsqu'ils possèdent un débit limité (ici égal à $.1C$).

réponse des flots n'est plus borné. On notera qu'au niveau paquet, avec des buffers de taille finie, le lien est toujours stable en raison des pertes de paquets.

Ben Fredj *et. al* [15] remarquent que l'accumulation des flots est plus lente lorsque la distribution de leur taille possède une queue lourde, les flots de petite taille parvenant à terminer, au détriment des flots plus longs. Les auteurs modélisent également l'impatience des utilisateurs (des applications) qui stabilisent le système en interrompant les connexions lorsque le temps de réponse devient trop important. Cette argumentation diffère de celle reposant sur les modèles classiques d'utilité pour les flots élastiques où celle-ci est strictement concave, même lorsque le débit tend vers 0 [140].

Les abandons de flots entraînent un gaspillage des ressources du réseau, qu'il convient de prévenir au plus tôt. Le même problème de congestion se produit dans les architectures Diffserv qui protègent uniquement la performance des classes les plus prioritaires, en reportant la congestion sur la classe *Best effort* [20]. Ben Fredj *et al.* [14] proposent un contrôle d'admission pour les flots élastiques en limitant le nombre de flots en cours se partageant la bande passante.

Lorsque les flots n'ont pas de limite de débit et que le partage est équitable, une proposition simple revient à limiter les flots à un nombre maximal m, afin que leur débit moyen soit d'au moins C/m. Le système peut être représenté par une file M/G/1/m/PS. Lorsque $\rho < 1$, la probabilité de blocage p_b est négligeable, et pour $\rho > 1$:

$$p_b(\rho, m) = \frac{\rho^m(1-\rho)}{1-\rho^{m+1}} \xrightarrow[m\to\infty]{} 1 - 1/\rho$$

Le choix du seuil d'admission n'est pas critique pour la performance dès que $m > 50$ [14]. Une valeur de m plus importante n'a pas d'impact sur la probabilité de blocage des flots, mais entraîne un gaspillage des ressources. Il est donc bénéfique de choisir une valeur la plus petite possible, d'où le compromis $m = 100$ choisi dans [91].

Avec des flots de débit limité r, on peut choisir $m = C/r$. La probabilité de blocage devient alors : $p_b(\rho, m) \approx B(\rho m, m)$, où $B(.)$ est la formule d'Erlang présentée en (3.1). Le choix de $m = 100$ fait précédemment convient également dans ce cas.

Dimensionnement

Ces formules permettent le dimensionnement d'un lien en vue d'offrir un blocage négligeable aux flots à charge nominale. La présence de flots à débit limité ne permet pas de limiter directement le nombre de flots en cours, mais il convient plutôt de considérer le débit qu'aurait un nouveau flot non goulotté sur le lien. Nous verrons comment ce contrôle d'admission est mis en pratique dans Cross-Protect dans la Section 3.5.3.

L'ajout d'un contrôle d'admission permet également de conserver l'insensibilité du modèle. Des extensions de ces résultats existent pour des débits d'accès différents, où dans le cas d'un partage non-équitable (modèle processeur partagé discriminatoire, ou DPS, *Discriminatory Processor Sharing*) ; l'insensibilité est alors approximative.

3.3.3 Extension à un réseau de données

Considérons un réseau formé d'un ensemble $\mathcal{L}$ de liens de capacité $C_l, l \in \mathcal{L}$. K classes sont définies, telles que les flots de classe k suivent la route $r_k \subset \mathcal{L}$ et ont un débit d'accès limité a_k. Nous considérons les allocations donnant un débit équitable aux flots de la même classe.

Allocation de ressources

Considérons temporairement des flots permanents de K classes, dont le nombre est représenté par le vecteur $X = (n_1, \cdots, n_K)$. Le vecteur des débits à long terme de chaque classe est $\Phi = (\phi_1, \cdots, \phi_K)$, tel que le débit de chaque flot de classe k est ϕ_k / n_k.

La région des débits réalisable $\mathcal{R}$ représente l'ensemble des vecteurs Φ possibles ; c'est un polytope convexe [1] défini par les contraintes posées par chaque lien [2]. Une allocation est efficace si Φ se situe sur le bord de $\mathcal{R}$. Elle est Pareto-efficace (et donc efficace) si elle consomme toutes les ressources [3].

Les allocations usuelles sont basées sur une fonction concave d'utilité $U(.)$, et se ramènent au problème d'optimisation suivant :

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & \sum_{k=1}^{K} n_k U\left(\frac{\phi_k}{n_k}\right) \\ \text{s.c.} \quad & \left(\sum_{k \backslash l \in r_k} \phi_k \leq C_i\right) (\forall l \in \mathcal{L}) \\ & \phi_k / n_k \leq a_k, \forall k \in [\![1, K]\!] \end{aligned}$$

La première contrainte correspond aux capacités des liens, et la seconde aux débits d'accès.

On retrouve les allocations classiques : *max throughput* ($U(\cdot) = \cdot$), *proportional fairness* [84] ($U(\cdot) = log(\cdot)$), *minimum potential delay* [136] ($U(\cdot) = 1/\cdot$), et *max-min* [22] qui est un cas limite d'une mesure d'utilité [110]. Les auteurs de [110] regroupent ces allocations, dites *α-fair*, sous une formulation mathématique commune. Lan *et al.* [96] généralisent plusieurs notions d'équité (dont les allocations *α-fair* et l'indice de Jain) et donnent un aperçu du compromis réalisé par les allocations *α-fair* entre équité et performance.

Ce problème d'optimisation se prête notamment bien à la réalisation distribuée d'algorithmes de contrôle de congestion de type TCP. Sauf retour explicite du réseau, la saturation des liens est typiquement déterminée de manière heuristique, en se basant par exemple sur les pertes de paquet pour les versions classiques de TCP.

Stabilité

La région de stabilité S est définie par l'ensemble des vecteurs $\rho = (\rho_1, \cdots, \rho_K)$ tels que le réseau est stable, c'est-à-dire que les flots terminent en un temps fini, i.e. $X(t)$ est borné. Lorsque R est convexe, alors S est exactement $\mathring{R}$, la frontière de R. Il suffit ainsi que la charge de tous les liens soit inférieure à leur capacité [28], ce qui peut-être assuré par un contrôle d'admission. Les allocations basées sur l'utilité (ainsi que l'allocation balancée que nous allons introduire) maximisent la région de stabilité, et sont ainsi une bonne approche pour les réseaux filaires.

Allocation balancée

Comme nous l'avons vu, le nombre de flots en cours est un processus aléatoire. On note $X(t)$ le vecteur représentant le nombre de flots de chaque classe, ν_i le taux d'arrivée des flots sur le nœud i, $p_{i,j}$ la probabilité de routage de i à j et μ_i le taux de service moyen de la file i. Les flots de classe k arrivent selon un processus de Poisson (sans perte de généralité dans le cadre d'arrivées de sessions Poisson) d'intensité λ_k et de tailles exponentielles i.i.d de moyenne σ_k. La charge correspondante est $\rho_k = \lambda_k \sigma_k$ bits/s.

Dans un tel contexte dynamique, Massoulié et Roberts [132] précisent qu'à l'échelle de temps des flots, qui est celle perçue par un utilisateur, la performance dépend autant du processus du nombre de flots en cours $X(t)$ que de l'équité de l'allocation. L'utilité d'un flot est plus justement mesurée par

1. Cela est valable pour un réseau filaire, ce n'est généralement pas le cas, par exemple pour un réseau sans fil avec interférences [103]

2. On a un polytope convexe variable dans le temps si l'on considère des mécanismes de priorité, des chemins multiples (*multipath*), ou encore des pannes de lien

3. Ces deux notions sont équivalentes si la bordure de $\mathcal{R}$ ne contient aucun segment parallèle à l'axe d'une classes de flots.

des métriques telles que leur temps de complétion. Il suffit que l'allocation soit suffisamment équitable pour que la performance des flots soit satisfaisante [28].

Le problème majeur des allocations *α-fair* est qu'elles sont sensibles à des caractéristiques détaillées du trafic et ne permettent pas une modélisation efficace de la performance des flots. Bonald et Proutière [30] introduisent la notion d'allocation balancée (BF, pour *balanced fairness*), qui est l'unique allocation optimale et insensible possible dans un réseau. Sa formulation repose sur les réseaux de Whittle, qui sont une extension des réseaux de Jackson où le taux de service dépend de l'état. Ce modèle est une extension directe du modèle PS à un réseau de files d'attente. Les auteurs ont montré que l'insensibilité de la distribution stationnaire du nombre de flots [4] correspondait à la réversibilité partielle du processus Markovien $X(t)$. Cela se transcrit dans le réseau par l'équilibrage des débits par une fonction positive $\Phi(x) : \phi_k(x) = \Phi(x - e_k)/\Phi(x)$, pour $k = 1, \cdots, K$.

L'allocation balancée correspond au choix d'une telle fonction respectant les contraintes de capacité et maximisant l'utilisation des ressources. Elle est définie de manière récursive avec $\Phi(0) = 1$ et

$$\Phi(x) = \max\left(\max_{l=1,\ldots,L}\left(\frac{1}{C_l}\sum_{i:l\in r_i}\Phi(x-e_i)\right), \max_{k:x_k>0}\left(\frac{1}{a_k x_k}\Phi(x-e_k)\right)\right)$$

avec $\Phi(x) = 0$ si $x \notin \mathbb{N}^N$.

Bornes stochastiques de performance

Malgré sa propriété intéressante d'insensibilité pour un dimensionnement robuste des liens, l'allocation balancée reste complexe à évaluer dans un contexte pratique, puisqu'elle dépend de la demande sur toutes les routes et des capacités des liens. Bonald et Proutière [31] proposent des bornes stochastiques sur le temps de réponse des flots, permettant d'évaluer de manière conservatrice la performance de l'allocation à partir d'informations locales aux liens uniquement. Les auteurs suggèrent un comportement similaire pour les autres allocations équitables de type *max-min* ou *proportional fair*. Kortebi *et al.* [94] donnent notamment un exemple d'application de l'allocation balancée, afin d'estimer la performance de *max-min*.

L'architecture *Flow-Aware Networking* réalise *max-min* grâce à l'emploi de *fair queueing* [132], ce qui semble être un choix acceptable et dont on peut approximer la performance à l'échelle du réseau grâce à l'allocation balancée.

3.4 Intégration des flots *streaming* et élastiques

Les besoins très différents de performance entre les flots *streaming* et élastiques permettent à [15, 91] de proposer un ordonnancement approprié qui bénéficie de leur différenciation : les flots *streaming* sont servis en priorité, tandis que flots élastiques utilisent la capacité laissée disponible. Le lien est ainsi transparent pour le trafic *streaming*, du moment que leur charge reste contrôlée. Nous nous intéressons ici à la performance réalisée par les flots lors d'une telle intégration.

Si [88] montre que la stabilité d'un réseau avec des flots *streaming* adaptatifs n'est pas affectée, ce n'est généralement pas le cas avec des flots non-adaptatifs. Les articles [116], [46] et [89] évaluent la performance d'un réseau où les flots *streaming* non-adaptatifs sont traités en priorité. Delcoigne *et al.* [46] donnent notamment des explications qualitatives sur la performance réalisée : la performance des flots élastiques peut être affectée par un phénomène d'instabilité locale, quand bien même la charge globale reste inférieure à la capacité du lien. Un contrôle d'admission approprié pour les flots *streaming* permet de garantir la performance dans un tel contexte, ainsi que de fournir des bornes de performances insensibles pour le temps de réponse d'un flot. Les bornes sont très étroites dans des conditions réalistes de trafic.

La relation entre ressources, demande et performance, introduite dans la section 3.1.1 est représentée pour FAN en figure 3.3. Etant données les capacités des liens, et une caractérisation simple de la demande en trafic (charge et débit crête pour les différents types de flots), l'architecture FAN apporte des garanties de service pour les flots *streaming* (délais, taux de pertes et gigue négligeables), et élastique (débit moyen minimum garanti).

L'architecture Cross-Protect que nous allons présenter réalise un contrôle d'admission implicite dans un tel contexte. L'article original [91] considère des situations où la proportion du trafic *streaming* n'est pas trop importante. Nous nous intéressons à un contexte plus réaliste où la plupart des flots sont limités en débit dans le Chapitre 6.

4. le fait qu'elle ne dépende pas de la distribution de la durée des flots

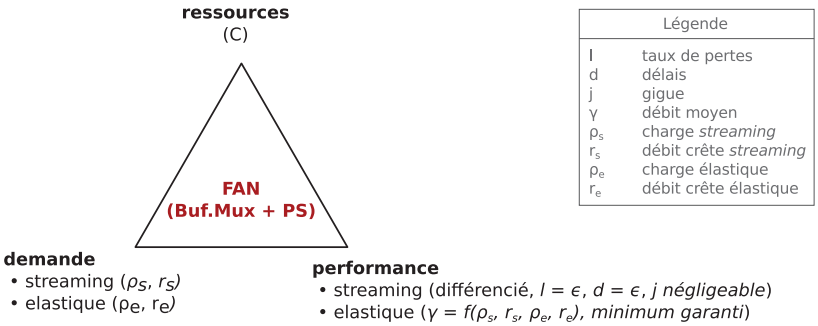


FIGURE 3.3 – Relation fondamentale entre ressources, demande et performance réalisée par l'architecture FAN

.

3.5 L'architecture de différentiation implicite de service Cross-Protect

Nous considérons une réalisation de FAN pour le cœur de réseau, Cross-Protect, qui permet de garantir la performance des flots *streaming* et élastiques. Cross-Protect repose sur la combinaison d'un d'ordonnancement équitable (*fair queueing*) et d'un contrôle d'admission implicite au niveau flot, basé sur des mesures (MBAC). Cette architecture ne nécessite pas de mécanisme complexe de marquage ou de signalisation, et permet ainsi au réseau de conserver son interface *Best Effort*, tout en permettant les évolutions nécessaires au sein du réseau : routage adaptatif, optimisations de TCP, etc. Une déclinaison de ces mécanismes pour le réseau d'accès sera considérée dans le Chapitre 7.

Le nom Cross-Protect vient de la complémentarité des deux mécanismes mis en œuvre. Le contrôle d'admission utilise des mesures de congestion fournies par l'ordonnanceur afin de décider de l'acceptation ou du rejet d'un nouveau flot ; en contrôlant la charge des flots *streaming* et élastique, il permet de conserver leur performance et d'assurer l'extensibilité de l'ordonnanceur (le nombre de flots à maintenir à tout instant est borné).

3.5.1 PFQ : un algorithme de FQ extensible

Le comportement naïf d'un algorithme d'ordonnancement équitable (*fair queueing*, ou FQ) consiste à diviser l'espace du buffer en plusieurs files d'attente (virtuelles), qui reçoivent chacune les paquets d'un seul flot. Il en résulte que FQ est souvent considéré comme non-extensible sur des liens de forte capacité où le nombre de flots en cours est très élevé.

Principe de l'algorithme PFQ

L'algorithme *Priority Fair Queueing* (PFQ) [91] est une extension de l'algorithme Start-Time Fair Queuing (SFQ), qui maintient un temps virtuel et attribue une estampille à chaque paquet déterminant son ordre d'émission. PFQ distingue implicitement les flots selon que leur débit instantané est inférieur ou supérieur au débit équitable. Celui-ci fluctue et est défini à tout instant en fonction du nombre et des caractéristiques des flots en compétition. Il faut noter que le débit de chaque flot n'est pas calculé, mais au lieu de cela PFQ se fonde sur l'évolution de la file d'attente.

L'algorithme est détaillé dans l'article original [91] et illustré en figure 3.4. Les flots dont le débit est supérieur au débit équitable voient leurs paquets s'accumuler dans la file d'attente du routeur jusqu'aux instants de pertes de paquets. L'algorithme d'ordonnancement est responsable du partage de la bande passante laissée disponible entre ces flots. PFQ intègre une gestion de la file d'attente qui oriente les pertes sur les flots de plus fort débit qui ont accumulé le plus grand nombre de paquets : cela protège les flots *streaming* à débit limité et permet au mécanisme de contrôle de congestion de TCP de réagir. Il est suffisant pour un algorithme d'ordonnancement de maintenir une file pour ces flots uniquement, qualifiés de flots actifs. Les flots de débit inférieur ont à tout moment au plus un paquet dans les files d'attente du routeur. Ils ne participent pas activement aux mécanismes de partage de bande passante et peuvent être "ignorés" par l'algorithme. Sans affecter la performance des autres flots, ils peuvent être servis en priorité, tant que leur charge reste limitée. PFQ maintient une file mixte, dite file PIFO (*Push In First Out*), dont le début représente la partie prioritaire et où tous les paquets ont la même estampille, et dont la fin reçoit les autres paquets, ordonnés en fonction de leur estampille.

Nous avons montré précédemment que le nombre de flots goulottés sur un lien est généralement faible jusqu'à de fortes charges. Ce nombre est plus important pour des flots à débit limité, mais [94]

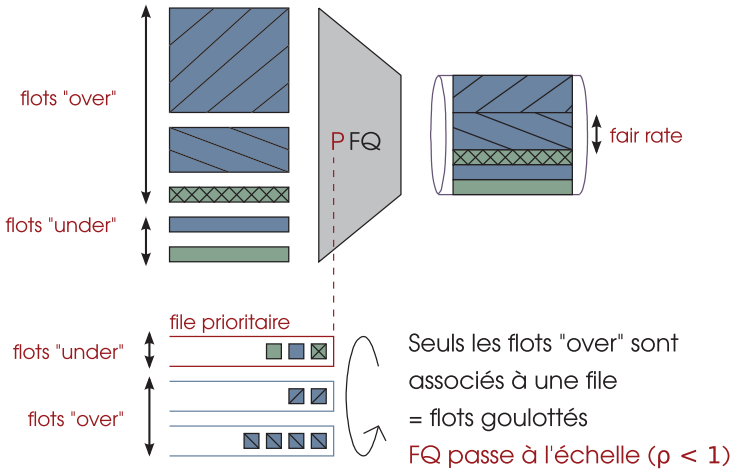


FIGURE 3.4 – Illustration du fonctionnement de l'algorithme PFQ

montre à partir d'un modèle confirmé par des traces de trafic que ce nombre est limité à quelques centaines jusqu'à une forte charge indépendamment de la capacité du lien.

Il existe d'autres déclinaisons de l'algorithme PFQ pour Cross-Protect , comme PDRR [92] (*Priority Deficit Round Robin*), qui se base sur DRR (*Deficit Round Robin*) [155] et présente l'avantage d'avoir une complexité en O(1), et d'être plus adaptée à la réalisation au sein d'un routeur commercial incluant déjà des mécanismes tels que DRR. Si elle est similaire en théorie, cette réalisation se révèle un peu moins équitable en pratique en raison de la dynamique des arrivées et départs de flots.

Le fair rate instantané : une caractéristique de la file d'attente

Lorsque tous les flots sont *non-bottlenecked*, les arrivées et départs de paquets définissent des périodes d'activité (*busy period*) et de silence (*idle period*) de la file d'attente, qui permettent de déduire directement le débit équitable instantané, ou *fair rate* instantané (fr_i).

Lorsqu'un ou plusieurs flots sont *bottlenecked*, on définit alors un *busy cycle* comme l'intervalle pendant lequel l'ensemble des flots *non-bottlenecked* sont traités, ainsi qu'un quantum (par ex. 1 MTU) au plus de chaque flot *bottlenecked*. Lorsqu'un paquet d'un flot arrive alors qu'il a déjà été servi dans le même cycle, il est inséré dans la file non prioritaire (*backlogged*) pour être traité dans les cycles suivants.

L'algorithme a ainsi besoin de maintenir une liste des flots actifs pendant chaque cycle ou *busy period* afin de garder en mémoire les flots précédemment émis. Cette liste est vidée lorsque le lien devient inactif, ou du moins purgée des flots terminés lors d'un changement de cycle, qui se manifeste par le changement de l'estampille du paquet en tête de file.

3.5.2 Indicateurs de congestion

Le fonctionnement naturel de l'algorithme PFQ permet la mesure de plusieurs indicateurs de congestion, qui sont utilisés par le contrôle d'admission. Le *fair rate* et le *priority load* sont présents dans la proposition initiale de PFQ.

La charge prioritaire, ou *priority load* (PL), est une mesure lissée exponentiellement du volume de trafic $B(t)$ arrivant dans la file prioritaire pendant un intervalle de temps τ.

$$pl = \Delta B / \tau$$

$$PL = (1 - \alpha) pl + \alpha PL$$

où $\Delta B = B(t + \tau) - B(t)$ et α un paramètre de lissage ($0 < \alpha < 1$).

Le débit équitable, ou *fair rate* (FR), est également une moyenne à long terme de la quantité suivante :

$$fr = MAX(\Delta vt, S.C) / \tau$$

$$FR = (1 - \beta) fr + \beta FR$$

où vt est le temps virtuel de la file, c'est-à-dire l'estampille temporelle du paquet en tête de file, S la durée pendant laquelle la file est vide sur l'intervalle τ, et $\Delta vt = vt(t + \tau) - vt(t)$. FR représente le débit moyen d'un flot fictif qui aurait constamment des paquets à émettre.

Nouveaux indicateurs de congestion

Dans le contexte d'un lien sur lequel les flots sont majoritairement élastiques, des estimations grossières suffisent, notamment pour PL, et les valeurs de τ sont dans les deux cas choisies égales à 100ms. Nous verrons dans le Chapitre 6 le besoin de définir un algorithme plus avancé pour les cas fréquents où le lien transporte majoritairement des flots à débit limité (cas d'un lien ADSL par exemple).

L'adaptation de l'algorithme afin de fournir une mesure de fr_i a été faite pour les besoins de l'algorithme présenté dans le Chapitre 6. Nous ajouterons également une mesure de la variance du PL (en plus de la valeur moyenne), et nous analyserons l'importance d'un choix correct de τ.

3.5.3 Contrôle d'admission basé sur des mesures

Motivations

L'intégration d'un contrôle d'admission est une solution efficace et éprouvée permettant d'agir proactivement afin de garantir la performance des flots en cours sur un lien [139, 24], en rejetant l'excédent de trafic. Il doit être transparent dans des conditions normales de charge.

Contrôle d'Admission Basé sur des Mesures

Les algorithmes de contrôle d'admission basés sur des paramètres de trafic, PBAC (*Parameter Based Admission Control*), offrent des garanties déterministes aux flots en fonction de descripteurs de trafic. Outre le fait que ces derniers sont très complexes à définir en raison du comportement du trafic IP au niveau paquet, ces algorithmes s'avèrent peu efficaces. Ils conduisent à une faible utilisation du lien, et nécessitent des mécanismes de signalisation et de contrôle de conformité du trafic.

Le choix pour Cross-Protect d'un contrôle d'admission basé sur des mesures (MBAC, *Measurement Based Admission Control*) provient de sa simplicité et de son efficacité. Il permet d'offrir aux applications des garanties statistiques (suffisantes) à partir de descripteurs simples (tels que le débit crête des flots) sans recourir à de tels mécanismes. Il permet également une adaptation au trafic et à ses changements.

Un algorithme de MBAC doit être simple, extensible et robuste (notamment aux erreurs de mesure). La complexité réside principalement dans le maintien d'une liste des flots protégés (identification au vol, purge après un certain temps d'inactivité, etc.), puisque le nombre de flots en cours sur un lien peut être élevé. Il est cependant possible d'utiliser des technologies matérielles telles que celles présentées dans la section 2.4.1 du Chapitre 2, ou d'inscrire les flots dans la table de manière probabiliste (ce qui élimine un grand nombre de flots très courts sans trop affecter la performance du lien).

MBAC et différenciation implicite de service

Les deux modèles présentés précédemment pour gérer les flots *streaming* et élastiques reposent sur un moyen de limiter le nombre de flots en cours :

- pour les flots *streaming*, il s'agit de garder le lien transparent afin de préserver les conditions de multiplexage sans buffer ;
- pour les flots élastiques, cela permet d'assurer un débit minimal pour les flots en cours et de garantir le temps de complétion. Un grand nombre de flots en cours cause un grand nombre de pertes et de retransmissions, et peut conduire à l'abandon de connexions (impatience utilisateur ou applicative), gaspillant inutilement des ressources.

Les mesures fournies par l'algorithme PFQ sont adaptées au contrôle de la performance des deux classes de trafic :

- PL permet de contrôler la charge des flots *streaming* (envoyés dans la file prioritaire) avec la performance du multiplexage sans buffer ;
- FR est une mesure du débit à long terme des flots élastiques, ordonnancés selon une discipline FQ.

Le fonctionnement du contrôle d'admission et sa complémentarité avec l'ordonnanceur PFQ sont illustrés en figure 3.5.

Seuils d'admission

L'hypothèse générale est que **les flots *streaming* possèdent un débit limité $r < p$ typiquement faible par rapport à la capacité du lien**, et nécessairement inférieur au seuil sur le *fair rate*. Le contrôle d'admission permet de garantir que les flots de débit inférieur ou égal à p sont toujours traités en priorité, tandis que PFQ réagit à la surcharge en différenciant les flots de débits les plus élevés.

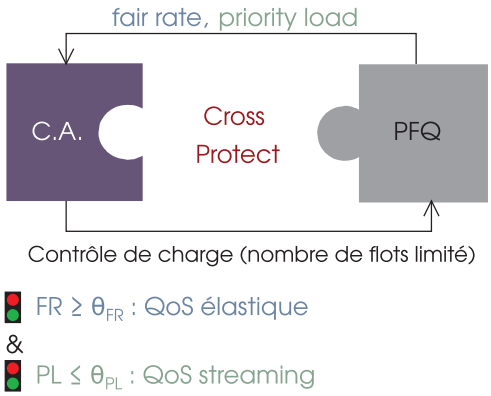


FIGURE 3.5 – Complémentarité du Contrôle d'Admission (C.A.) et de l'ordonnanceur PFQ au sein de Cross-Protect, et mention des critères d'admission.

Nous avons vu précédemment que le seuil sur le FR est peu sensible, et une valeur de 1% est satisfaisante pour la performance des flots en cours (voir Section 3.3.2).

Le seuil sur *PL* n'est pas critique du moment que le trafic est majoritairement élastique sur le lien ; c'est pourquoi la proposition initiale [91] fixe cette valeur à 70% de la capacité du lien. Dans le Chapitre 6, nous reconsidérerons cet aspect lorsque les flots sont majoritairement à débit limité, comme sur un lien de collecte ADSL par exemple.

3.6 Vers la réalisation d'un routeur Cross-Protect

3.6.1 Évaluation de l'architecture en simulation

L'architecture Cross-Protect a initialement été évaluée à l'aide du simulateur NS-2 [105], pour lequel plusieurs modules ont été proposés, afin d'avoir un cadre d'évaluation complet, notamment utilisé dans les chapitres suivants :

- algorithmes de contrôle d'admission, PFQ et de leur combinaison ;
- outils de génération de trafic respectant les modèles que nous avons présentés ;
- modules et outils permettant le suivi des flots ainsi que le calcul de diverses métriques de performance ;
- un module permettant la simulation d'un grand nombre de flots à débit d'accès limité ;
- un module permettant le rejeu de traces au niveau paquet et flot, avec conservation des caractéristiques telles que le débit crête ;
- un module reproduisant le comportement de flots TCP avec retentatives.

3.6.2 Évaluation de l'architecture par expérimentation

En plus d'un module pour le simulateur NS-2, les mécanismes Cross-Protect ont été réalisés au sein de l'architecture de contrôle de trafic du noyau Linux (`iptables` et `traffic control`)[5], afin de construire une plate-forme d'expérimentation et d'évaluation de la solution. Les objectifs d'une telle réalisation sont multiples :

- convaincre de la faisabilité de l'architecture, et notamment du traitement des flots, à haut débit ;
- démontrer le fonctionnement de Cross-Protect dans des conditions réelles,
- analyser l'évolution des différentes structures de données (notamment les tables de flots) ;
- cerner les contraintes d'implémentation de l'architecture, notamment en identifiant les différences entre routeur purement logiciel et routeur spécialisé ;
- mieux comprendre les contraintes posées par le trafic (identification des flots, encapsulation, chiffrement, etc.) et les applications le générant (téléphonie IP, *streaming* vidéo).
- enfin, évaluer différentes propositions pour les briques constituant l'architecture.

Cette réalisation a été limitée aux flots TCP, UDP et ICMP de IPv4, qui sont reconnus par le quintuplet classique (adresses et ports source et destination + protocole, ou, dans le cas d'ICMP, adresses source et destination + type et code ICMP + protocole), et cohérents dans le temps (*timeout*). Une possible utilisation du *flow label* d'IPv6 n'a pas été considérée pour l'évaluation.

5. voir `http://netfilter.org/`

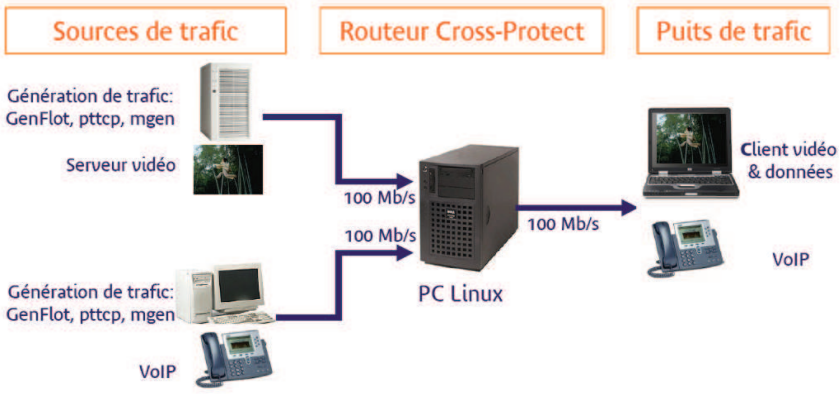


FIGURE 3.6 – Plateforme de démonstration Cross-Protect

Plate-forme d'expérimentation

Cette réalisation a été intégrée au sein d'une plateforme plus complète d'expérimentation [8], illustrée en figure 3.6. Elle est constituée d'un routeur central constituant un lien par lequel transite le trafic entre plusieurs sources et puits de trafic : téléphones IP, serveurs de fichiers ou de vidéo *streaming*, applications de génération de trafic et de visualisation (certaines étant communes avec le simulateur).

Évaluation

Cette évaluation expérimentale de l'architecture Cross-Protect au travers de la plateforme GNU/Linux a pu donner lieu à deux démonstrations. Les évaluations par expérimentation qui ont été effectuées et démontrées sont incluses dans la thèse d'Abdesselem Kortebi et par conséquent non répétées ici. Le document est disponible en ligne [6].

6. `http://jordan.auge.free.fr/files/these_kortebi.pdf`