Résolution des problèmes de routage asymétrique et de partage de charge dans les pare-feux

Cédric Aoun, Olivier Paul, Ahmed Serhrouchni

Résolution des problèmes de routage asymétrique et de partage de charge dans les pare-feux

Cédric Aoun * — **Olivier Paul** ** — **Ahmed Serhrouchni** ** *

* *Nortel Networks/ENST Paris*
cedric.aoun@nortelnetworks.com

** *Institut National des Télécommunications*
olivier.paul@int-evry.fr

*** *ENST Paris*
ahmed@enst.fr

RÉSUMÉ. *Les pare-feux (plus connus sous leur nom anglais de firewall) sont devenus indispensables dans la défense du périmètre de sécurité des réseaux. Les règles dynamiques installées sur les pare-feux nécessitent dans la plupart des cas que les paquets (d'un flux) entrants et sortants traversent le même pare-feu, le cas échéant les paquets entrants dans le réseau sont ignorés et donc non acheminés à leur destination. Ce problème est dû au routage asymétrique des paquets (les paquets sortent du réseau en suivant une certaine route non identique à la route de retour). Le partage de charge utilisant les métriques de routage peut induire qu'une règle dynamiquement soit créée sur un pare-feu et que les paquets de données associés traversant un autre pare-feu (ils seront donc ignorés). Dans ce papier les auteurs décrivent les solutions actuelles, leurs limites et proposent une nouvelle solution pour la résolution des problèmes du routage asymétrique et du partage de charge. Les mécanismes proposés sont applicables à tout service nécessitant l'installation dynamique de règles sur un routeur (QoS entre autres).*

MOTS-CLÉS : *pare-feux, firewall, stateful, routage asymétrique, partage d'états.*

CFIP 2005

1. Introduction

Les pare-feux constituent un élément essentiel de la défense du périmètre de sécurité, leur caractéristique principale est qu'ils empêchent des flux indésirables de pénétrer au sein du périmètre de sécurité. Les règles, utilisées par les pare-feux pour empêcher des intrus de communiquer avec des machines internes au réseau, peuvent être dynamiques ou statiques. Une règle est dite dynamique lorsqu'un événement induit sa création en temps réel, cet événement peut être dû à l'arrivée d'un paquet de donnée ou d'un message d'un protocole de signalisation de pare-feux [STI 04]. Un pare-feu utilisant des règles dynamiques est dit « stateful » (mot anglais signifiant que le pare-feu garde en mémoire l'état de la règle).

Dans la plupart des pare-feux la règle utilisée par défaut est d'ignorer les flux ne correspondant pas aux règles pré-installées (qu'elles soient statiques ou dynamiques). Lorsque plusieurs pare-feux sont déployés au(x) périmètre(s) de sécurité d'un réseau, il est très probable que la règle dynamique associée au flux ne soit pas installée sur le pare-feu traversé par le flux. Par conséquent le flux traversant un pare-feu, non configuré avec la règle associée au flux, sera ignoré; ceci pourrait au pire mettre fin à la session applicative ou au mieux la dégrader .

Dans le cas de déploiement de plusieurs pare-feux au niveau d'un périmètre de sécurité, l'inconsistance entre le pare-feu configuré avec la règle adéquate et son flux associé peut être due:

a) Au partage de charge (ou « ECMP » pour Equal Cost Multi-Path) à poids égale entre les firewalls (partage dû à la métrique des protocoles de routage [THA 00]).

b) Au routage asymétrique, dû à une inconsistance des métriques des routes sortantes et des routes entrantes (liés aux accords entre Fournisseurs d'Accès Internet ou autres contraintes techniques ou administratives)

Dans certains cas, l'inconsistance entre le pare-feu sur lequel est installé la règle dynamique et le flux associé à cette dernière peut être liée aux deux raisons mentionnées ci-dessus. Dans certain cas b) peut être dû à un mélange d'inconsistances de métriques de routage et de partage de charge.

Dans la suite de ce document, nous examinons plus en détails ces problèmes. Nous étudions par la suite quelles solutions pourraient leur être apportées. Nous nous penchons en particulier en section 3 sur les protocoles de synchronisation de contextes. Nous montrons quelles sont leurs limitations et comment ils peuvent être étendus en section 4. Pour finir nous concluons en analysant les forces et faiblesses de notre proposition.

2. Description du problème

Dans cette section, deux types de règles dynamiques de pare-feux seront évoquées, les règles créent par la réception de paquets de données applicatives ainsi que les règles créent par des facteurs externes (protocole de signalisation indépendant de l'application générant les paquets de données [STI 04] ou par

analyse des messages de l'application c'est à dire une Application Layer Gateway (ALG).

Indépendamment du facteur contribuant à leur création, si les règles ne sont pas installées sur le pare-feu traversé par les paquets de données de la communication applicative, les paquets seront ignorés.

Dans ce document lorsque partage de charges est évoqué il s'agira toujours d'un partage à charges égales au niveau des protocoles de routage (i.e. ECMP). Pour la suite de ce document il est important que le lecteur ait une connaissance de base sur les implémentations d'ECMP. Ainsi :

- Les mécanismes utilisés par les routeurs implémentant la fonctionnalité ECMP assurent le séquencement des paquets des flux routés.
- La plupart des routeurs ayant la fonctionnalité ECMP utilisent un algorithme se basant sur l'adresse source et l'adresse destination du paquet, certains routeurs utilisent en plus des adresses source et destination les ports source et destination ainsi que le type de protocole de transport.

2.1 Problèmes des règles dynamiques créées par des flux de données applicatives

La Figure 1 montre l'impact du routage asymétrique avec TCP, les pare-feux A1 et A2 autorisent uniquement des connections TCP établies par des machines internes au réseau de l'usager Phil. Ainsi sur chacun des pare-feux A1 et A2, un segment TCP ayant pour indicateurs (flags en anglais) SYN et ACK est rejeté s'il ne correspond pas à un segment TCP avec flag SYN envoyé par une machine du périmètre de sécurité. Dans la Figure 1, le segment TCP avec flag SYN passe par le pare-feu A1 mais au retour le TCP SYN ACK passe par le pare-feu A2 à cause du routage asymétrique.

Certains pare-feux peuvent aussi avoir des règles dynamiques pour UDP. Ainsi un paquet UDP émit par une machine du périmètre de sécurité peut créer (en fonction de la configuration du pare-feu) une règle dynamique et cette machine ne pourra recevoir de réponse sur la même socket UDP que par la machine destinataire du paquet initial ayant crée la règle dynamique. Dans ce cas, comme on le constate en Figure 2, à cause du routage asymétrique les paquets émis par le destinataire initial passeront par le pare-feu A2 sur lequel aucune règle n'a été créée ayant pour conséquence le rejet de ces paquets.

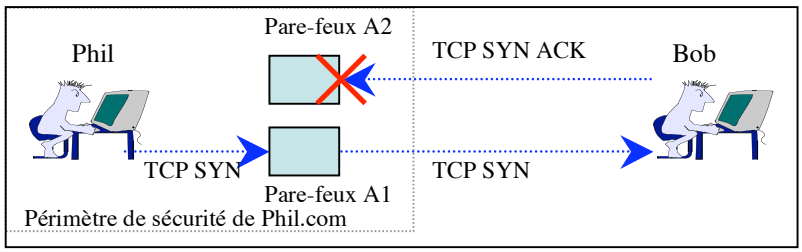

Figure 1. *Problèmes du routage asymétrique pour TCP et les pare-feux « Stateful »*

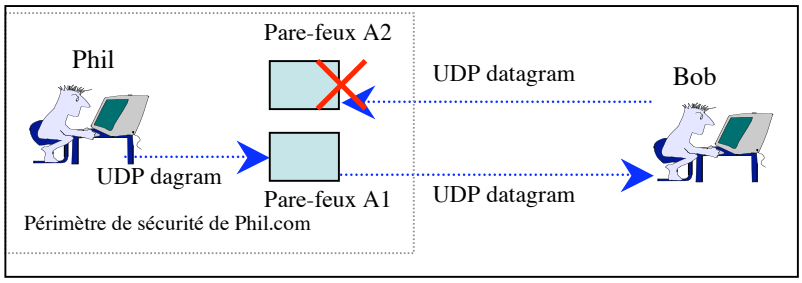

Figure 2. *Problèmes du routage asymétrique pour UDP et les pare-feux "Stateful"*

2.2 Description du problème pour les règles dynamiques créées par des facteurs externes

Lorsque les règles dynamiques ne sont pas créées par les paquets de données applicatives, leur création est due soit au protocole de signalisation associé à la signalisation lorsqu'une ALG est implémentée (par exemple pour SIP [ROS 02]) ou un protocole de signalisation de pare-feux ([SRI 02], [STI 04]). Une ALG est un ensemble de processus consistant à intercepter les messages de signalisation d'une application, à les interpréter pour en déduire comment le pare-feu doit agir sur les flux de données de l'application (ignorer ou acheminer les paquets de données).

L'un des problèmes principaux des ALGs est de nécessiter une connaissance minimale des applications et de pouvoir lire les messages de signalisation des applications. Ainsi lorsque les messages de signalisation de l'application sont chiffrés les ALGs sont inutilisables. D'autres raisons (maintenance, coût de mise à jour, interopérabilité, coût de développement) viennent s'ajouter au problème de confidentialité pour forcer l'industrie à adopter une autre solution : la signalisation des pare-feux.

Parmi les protocoles NSIS [HAN 04], le NSIS NATFW NSLP [STI 04] (par la suite nommé NSIS pour simplifier), développé par l'IETF, fait partie des solutions de signalisation de pare-feux. La principale caractéristique d'NSIS est que les entités communiquant avec le pare-feu n'ont pas besoin de connaître le pare-feu à l'avance, de ce fait cette solution ne nécessite pas la configuration d'informations décrivant la topologie du réseau. En effet NSIS utilise le concept "path-directed", qui consiste à envoyer le message de signalisation vers le destinataire final (et non le pare-feu), les pare-feux sur le chemin du message interceptent le message et créent une entrée dans une table pour acheminer le message de retour de proche en proche. Le routage des messages NSIS est géré par une couche protocolaire commune définie par [SCH 04].

NSIS utilisé de bout en bout, ne subit pas les problèmes liés au routage asymétrique mais peut être impacté par les problèmes de partage de charge (uniquement lorsque l'implémentation d'ECMP utilise les adresses source et destination ainsi que les ports de transport source et destination).

L'une des caractéristiques des protocoles NSIS est de suivre le chemin des données. Cependant dans le cas où ECMP est activé (et que les ports de transport source et destination sont utilisés) sur les routeurs séparant un hôte NSIS des pare-

feux, les messages NSIS ne suivront pas le même chemin que celui des données. La Figure 3 montre l'impact d'ECMP sur les opérations d'NSIS. Par souci de lisibilité les messages NSIS ne sont pas tous affichés.

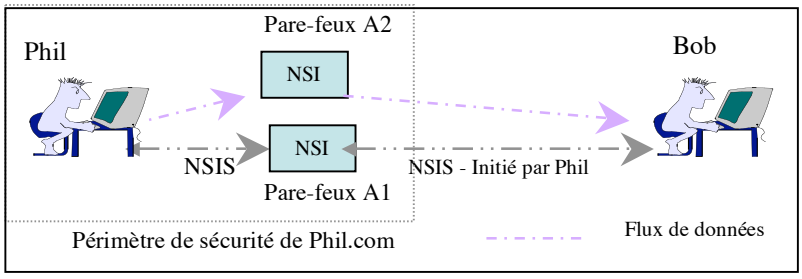

Figure 3. *Problème de NSIS de bout en bout avec ECMP*

Dans certains scénarios de déploiement, NSIS est utilisé en mode proxy; ceci est notamment le cas lorsqu'une des machines hôtes n'a pas d'implémentation NSIS ou dans le cas où la signalisation NSIS est terminée localement au sein d'un périmètre de sécurité. Lorsque NSIS est utilisé en mode proxy, les messages NSIS risquent d'être impactés par les problèmes de routage asymétrique et de routage ECMP.

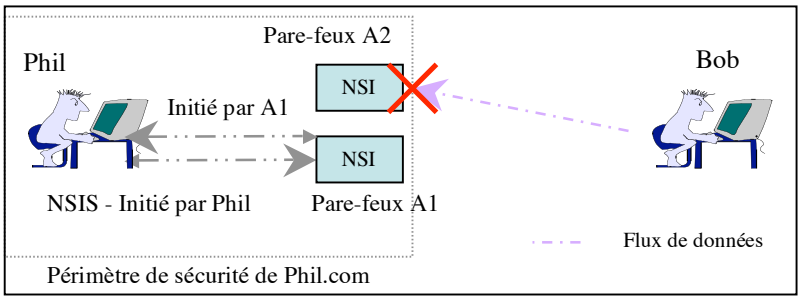

Figure 4. *Impact du routage asymétrique sur NSIS utilisé en mode proxy*

Le routage ECMP impacte les messages NSIS en mode proxy puisque le pare-feu (et non la machine communiquant avec Phil) est la source des messages NSIS destinés à l'hôte Phil dans la Figure 5.

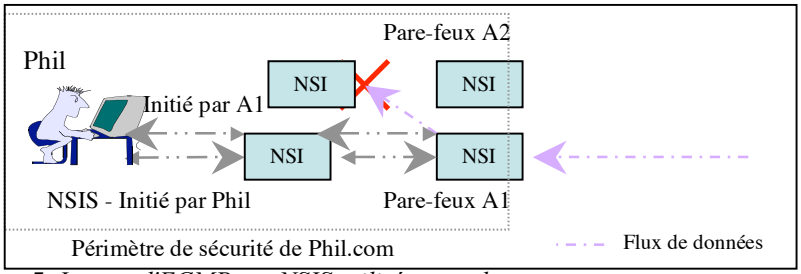

Figure 5. *Impact d'ECMP sur NSIS utilisé en mode proxy*

Les problèmes évoqués dans cette section peuvent être résolu par trois types de solutions :

a) Ancrage du flux par utilisation d'option IP de type 'source route'.
b) Ancrage du flux par modification de l'adresse de destination. Dans ce cas un traducteur d'adresse est utilisé ([SRI 99]).
c) Synchronisation d'états entre les pare-feux du périmètre de sécurité.

a) et b) ne sont pas des solutions transparentes pour le client applicatif, dans le cas de b) la solution est transparente uniquement lorsque ECMP n'est pas utilisé. De plus la solution a) nécessite l'utilisation d'une option du paquet IP, cette utilisation est très souvent critiquée par les administrateurs réseaux se souciant de dévoiler certaines parties de la topologie de leur réseau. L'utilisation d'un traducteur d'adresse impacte énormément d'applications [HOL 01], en particulier celles transportant des adresses et ports dans leurs paquets ; de plus cette utilisation ne peut pas être justifiée dans le cadre de réseau ayant suffisamment d'adresses publiques.

La solution c) apporte plus de transparence vis-à-vis des applications et ne dévoile pas d'information sur la topologie du réseau. La suite du document décrit les techniques de synchronisation d'états recommandées pour résoudre les problèmes décrits dans cette section.

3. Techniques de synchronisation d'états

Ces dernières années ont vu un travail important sur la définition de protocoles de gestion de panne. Des protocoles comme HSRP (Hot Standby Router Protocol) [LI 98], VRRP (Virtual Router Redundancy Protocol) [HIN 04] ou CARP (Common Address Redundancy Protocol) [BSD 03] permettent la détection de pannes ainsi que l'élection d'un routeur associé à une adresse IP servant de routeur/pare-feu par défaut.

Ces protocoles permettent lors de la panne du pare-feu "maître" de re-diriger les nouvelles communications vers un pare-feu "secondaire". Cependant ces protocoles ne permettent pas toujours de faire en sorte que les communications existantes puissent être acheminées par le pare-feu "secondaire". Dans le cas d'une re-direction du trafic, l'absence de ces informations contextuelles empêche le nouveau pare-feu de traiter les paquets des communications existantes de manière satisfaisante comme nous l'avons montré en section précédente.

Une technique permettant de limiter ce problème consiste à répliquer les états d'un pare-feu à l'autre de telle sorte que les contextes soient présents dans le cas d'une re-direction du trafic. Il n'existe pas à ce jour de protocole standardisé pour l'échange d'informations d'états. Cependant de nombreux pare-feux implantent un protocole de synchronisation d'état.

3.1. *Un aperçu d'un protocole de synchronisation: pfsync*

pfsync est le protocole de synchronisation d'états associé à OpenBSD. OpenBSD possède depuis quelques années un module de filtrage propre appelé *pf* [PF] (packet filter). Celui-ci gère les états de connexion en conservant des informations relatives à chaque communication (adresses, ports, protocole, numéros de séquence TCP attendus, timers, ...) ainsi que sur la politique de filtrage à appliquer à la connexion. La gestion des mises à jour dans *pfsync* se fait à chaque modification sur la table

d'états c'est à dire généralement à la réception de chaque paquet. Le protocole utilisé par pfsync est situé dans la pile protocolaire immédiatement au dessus d'IP. Le numéro de protocole associé est 240. Le protocole définit plusieurs opérations de type requête-réponse ou notification permettant à un pare-feu de créer un nouvel état dans un pare-feu secondaire, de mettre à jour un état dans un pare-feu secondaire, de supprimer un état dans un pare-feu secondaire, de demander le transfert de tous les états de manière simultanée ou d'un état en particulier, de transférer tous les états de manière simultanée, de demander l'état du pare-feu opposé.

pfsync utilise plusieurs techniques afin de limiter la quantité de données échangées d'un pare-feu à l'autre.

- Utilisation d'un protocole basé sur le multicast. L'ensemble des pare-feux participant au protocole est associé à une adresse multicast particulière (224.0.0.240). Cette idée permet de limiter le nombre de paquets transportés entre les pare-feux à un nombre proportionnel au nombre de paquets reçus par le pare-feu. Dans le cas de l'utilisation de flux unicast, ce nombre serait multiplié par le nombre de pare-feux du groupe.

- Utilisation de techniques d'agrégation d'états. L'agrégation d'état a deux objectifs. D'une part il s'agit de limiter le nombre de paquets émis en faisant en sorte que les paquets soient le plus remplis vis à vis de la valeur de la MTU entre les équipements. Ceci est réalisé en gardant en mémoire une liste des mises à jour à envoyer jusqu'à ce que la taille totale corresponde à la taille de la MTU. D'autre part comme plusieurs mises à jour concernant une même connexion peuvent être soumises à l'entité *pfsync*, il est possible de supprimer ou de modifier l'information des mises à jour les plus anciennes afin de ne pas stocker les nouvelles. Cette solution a cependant un inconvénient: les mises à jour peuvent être gardées indéfiniment soit car aucune mise à jour n'est reçue par *pfsync* soit car ces mises à jour concernent toujours la même connexion. Afin de régler ces problèmes *pfsync* maintient deux paramètres limitant la durée de rétention d'un paquet de mise à jour. Un compteur est associé à chaque mise à jour contenue dans un paquet, lorsque celui-ci atteint une valeur limite (128), le paquet est envoyé indépendamment de sa taille. D'autre part un timer est maintenu pour chaque paquet, de telle sorte qu'un paquet ne soit pas gardé plus d'une certaine durée (1 seconde).

- Utilisation d'informations d'état simplifiées. Certaines informations sont redondantes entre les mises à jour pouvant apparaître dans des paquets de mise à jour successifs. Le protocole utilise cette particularité pour définir deux types d'états : les mise à jour normales contenant toutes les informations relatives à un état et les mise à jour compressées contenant uniquement les informations généralement modifiées. Afin de créer le lien entre mise à jour compressées et états on utilise un identifiant unique.

En terme d'implantation, l'implantation de *pfsync* possède plusieurs particularités intéressantes. Contrairement à d'autres implantations de protocoles du même type, *pfsync* est implanté dans l'espace noyau. Cette particularité permet d'éviter la double

copie des informations de contexte entre l'espace noyau et l'espace utilisateur. La sélection des communications à dupliquer se fait au travers de la politique de filtrage du module pf. A chaque règle, il est possible d'associer une directive commandant la synchronisation de toutes les communications dérivées.

3.2. *Problèmes dans le cas d'un routage asymétrique*

Nous considérons dans la suite de ce document deux pare-feux M et N pour lesquels les communications subissent un routage asymétrique et utilisent un protocole de synchronisation tel que *pfsync*.

Dans le cas d'un routage asymétrique, des paquets peuvent arriver sur l'interface d'un pare-feu n'ayant pas traité les paquets précédents d'une communication. Ce cas peut se présenter lorsque le paquet reçu par N est une réponse au paquet traité par M. Si le temps d'aller retour entre M, la destination et N ou entre N, la source et M est plus faible que le temps d'installation d'état entre M et N, aucun état ne permettra de valider ce paquet. Un exemple est le cas de l'ouverture de connexion TCP. Si le paquet SYN est reçu par M et le paquet SYN-ACK est reçu par N, un état doit être présent en N afin de valider ce second paquet.

Une approche naïve afin de régler ce premier problème pourrait consister à utiliser une technique de partage de charge telle que nous l'avons présentée dans la section précédente et à lier le rythme d'émission des mises à jour à celui d'arrivée des paquets au lieu de le conditionner à une optimisation du remplissage des paquets de mise à jour. Cependant, cette approche implique que le débit nécessaire pour les mises à jour sera proportionnel au nombre de paquets soumis au groupe de pare-feux. Un paquet contenant une mise à jour sous *pfsync* a une taille d'environ 160 octets. En supposant un trafic typique rencontré sur Internet (taille de paquet de 300 octets) et un volume de trafic de 10Gb/s réparti entre des pare-feux en parallèle, les opérations de synchronisation généreraient environ un volume 5Gb/s de trafic. Ceci n'est généralement pas acceptable.

Un deuxième problème lié au mode de compression des paquets peut se produire lors du filtrage de paquets TCP. Un critère de filtrage des paquets TCP utilisé par toutes les implantations open source porte sur leurs numéros de séquence et d'acquittement [ROO01]. Pour chaque connexion on définit un intervalle de validité pour ces deux variables. Les bornes de ces intervalles (m_s,M_s et m_a, M_a respectivement les numéros de séquence et numéros d'acquittements) pour les segments TCP allant de la source vers la destination dépendent des numéros acquittés par les segments allant de la destination vers la source de la manière suivante:

$$M_s < max(a_d + w_s), \qquad m_s > max(a_d)$$
$$M_a < max(n_s), \qquad m_a > max(n_s) - MAXWIN$$

Où a_d est le numéro d'acquittement des paquets de la destination vers la source, w_s la taille de la fenêtre chez la destination, n_s le numéro de séquence du dernier octet des paquets reçus par la source et MAXWIN dépend de la taille maximale possible pour la fenêtre et de la taille maximale des segments (MSS).

Réciproquement une formule analogue est utilisée pour valider les segments TCP allant de la destination vers la source. Dans le cas d'une mise à jour trop lente (par exemple lorsque le timer est utilisé pour provoquer la mise à jour), les bornes des intervalles restent fixes. En parallèle, le numéro de séquence maximal, des

segments envoyés, peut avancer en fonction du débit. Dans le cas où celui-ci est supérieur à w_s durant la durée du timer, les paquets seront rejetés. Dans le cas de *pfsync* la durée maximale peut atteindre une seconde soit un débit de 500kbits/s environ pour une taille de fenêtre de 65535 octets.

4. Amélioration de la technique de synchronisation

Afin d'essayer d'améliorer la technique de synchronisation existante, regardons d'abord dans quel cas les deux problèmes précédents peuvent se présenter.

Vis-à-vis du premier problème, on peut estimer que le temps d'aller-retour est supérieur au temps de communication entre les pare-feux (le temps d'installation d'un état sur des équipements modérément chargés au travers d'un réseau local non congestionné est de l'ordre de 1ms). De ce fait il est toujours possible de mettre à jour les informations de contexte avant la réception d'un paquet de retour à condition que la mise à jour soit envoyée sans attente. Cette approche pose le problème du volume de trafic de synchronisation comme indiqué en section précédente. Une contrainte sur la méthode de synchronisation est donc d'essayer de minimiser le volume de trafic associé aux mises à jour sans utiliser de méthode d'agrégation telle que celle utilisée par *pfsync*.

4.1. *Mode de validation des paquets dans les pare-feu à états.*

Il nous faut entrer dans le détail des méthodes de validation des communications afin de voir quelles solutions nous pourrions apporter. Le mode de validation des communications dépend du protocole utilisé. Les informations associées aux contextes pour réaliser cette validation sont indiquées dans le Tableau 1.

Tableau 1. Informations utilisées pour le filtrage des paquets.

Protocole	Informations
TCP	Etat courant (automate), borne inférieure/supérieure numéro de séquence émetteur/récepteur, taille de fenêtre émetteur/récepteur, multiplicateur taille de fenêtre émetteur/récepteur, date d'expiration (valeur du timer d'expiration), direction, action à appliquer.
UDP	Date d'expiration (valeur du timer d'expiration), direction, action à appliquer.
ICMP (ind.)	Date d'expiration (valeur du timer d'expiration), direction, action à appliquer.
ICMP (req.-rep.)	Etat courant (automate), Date d'expiration (valeur du timer d'expiration), direction, action à appliquer.

Les informations d'état, de bornes de fenêtres et de date d'expiration sont mises à jour à chaque réception de paquet. Ces informations sont utilisées de la manière suivante:

- En fonction du contenu du paquet et de l'état courant, l'automate change d'état. Les paquets portant des informations non cohérentes vis à vis de l'état courant sont rejetés.

- Dans TCP, la spécification de l'automate TCP possède 11 états mais les automates de validation en possèdent généralement moins. Dans le cas de *pf* 5 états sont utilisés (SYN_SENT, ESTABLISHED, CLOSING, FIN_WAIT2, TIME_WAIT). Un automate est maintenu par direction. Le passage de SYN à SYN_SENT se fait du côté source après la réception du paquet SYN et du côté destination à la réception du SYN/ACK. Le passage de SYN_SENT à ESTABLISHED se fait dès la réception du SYN/ACK du côté destination et à la réception du ACK du côté source. Pour la fermeture de connexion, le passage de ESTABLISHED à CLOSING se fait à la réception d'un FIN du côté source ou destination. Le passage de CLOSING à FIN_WAIT_2 se fait alors à la réception d'un ACK. Enfin l'état TIME_WAIT est atteint à la réception d'un RST à partir de n'importe quel état.
- Dans ICMP, les échanges de type requête-réponse utilisent généralement un état par type de message. Le passage de l'état fermé à l'état ouvert se fait lors de réception d'une requête. L'opération inverse se fait à la réception de la réponse. Les réponses arrivant dans l'état fermé sont rejetées.

- A l'expiration d'un timer, le contexte associé à la communication est détruit. Les paquets arrivant après expiration sont soumis à la politique de filtrage pour décider de la création d'un nouveau contexte.
  - Pour TCP, une valeur différente de timer est associée à chaque état. Ainsi dans *pf* la durée du timer associé à l'état ESTABLISHED est de 24 heures alors que celui associé à l'état SYN_RECEIVED est de 30 secondes.
  - Pour UDP, un timer est généralement utilisé indépendamment de l'état de la communication. Dans *pf* la durée du timer associé aux communications UDP est de 60 secondes.
  - Pour ICMP, la durée du timer dépend du type de message. Dans *pf* les messages de type requête-réponse possèdent une durée de timer de 20 secondes contre 10 secondes pour les messages de type indication.

- Pour TCP, les numéros de séquence, d'acquittement et tailles de fenêtre sont utilisés pour construire des intervalles de validité des numéros de séquence et d'acquittement. les segments arrivant avec des numéros hors des intervalles d'acceptations sont rejetés.

4.2. *Diminution du nombre de mise à jour*

Nous considérons par la suite deux pare-feu M et N pour lesquels les communications sont asymétriques.

Le classement précédent nous montre que tous les protocoles n'ont pas des besoins similaires en terme fréquence de la mise à jour. Ainsi si on considère UDP,

une fois l'état installé, une mise à jour ne s'impose que toutes les 60 secondes dans le pire des cas. Ceci nous montre que, vis à vis de la première approche naïve, nous pouvons diminuer le volume de mise à jour en adaptant la fréquence de mise à jour aux événements. D'une manière générale on peut distinguer parmi les événements décrits précédemment, ceux liés à la réception d'un paquet (automate, intervalles des numéros de séquence et d'acquittement) et ceux liés au temps (timers). Parmi ces événements on peut également distinguer ceux ayant une influence sur l'acceptation d'un paquet et ceux n'en ayant pas.
Les événements de type automate ne nécessitent un envoi immédiat qu'en cas de changement d'état (Tableau 2). Ceci est vrai car l'absence de changements d'état n'a pas d'influence sur l'acceptation d'un paquet.

Les événements de type changement d'intervalles ne nécessitent une mise à jour que si le segment suivant le segment reçu ne valide pas les intervalles du pare-feu homologue. C'est toujours le cas pour les segments TCP SYN et SYN-ACK (Tableau 3) puisque ces segments permettent de définir les origines des intervalles.

Tableau 2. Evènements liés aux automates et nécessitant une action.

Evénement	**Action**
Réception TCP SYN/SYN-ACK Réception TCP FIN/FIN-ACK, RST Réception UDP, Etat non existant Réception ICMP requête Réception ICMP réponse	Envoi mise à jour;

Pour les autres segments, cette condition peut être déterminée plus facilement par le pare-feu homologue. Si on se place sur le pare-feu N, pour un paquet portant S, A, W (S numéro de séquence du dernier octet du segment, A numéro de séquence acquitté, W fenêtre) et en utilisant la connaissance des intervalles de validité locaux ($[n_s,N_s]$ $[n_a,N_a]$), il est possible de déterminer les distances $|S-N_a|$ et $|A-N_s|$. Lorsque ces distances sont inférieures à une borne prédéfinie D, une mise à jour est nécessaire. N peut alors demander cette mise à jour à M.

Tableau 3. Evènements liés aux intervalles et nécessitant une action.

Evénement	**Action**
Réception TCP SYN/SYN-ACK	Envoi mise à jour;
Réception TCP ACK S, A [ns,Ns] [na,Na]	Si (S-Ns)<MSS ou (A-Na)<D Demande de mise à jour; Sinon /

Les événements liés aux timers sont de deux types. Les événements conduisant à définir une durée de vie initiale pour une communication et ceux visant à la modifier. Les premiers doivent être envoyés immédiatement car l'absence de durée de vie conduit au rejet des paquets liés à la communication. Parmi les seconds, on peut distinguer les événements conduisant à une augmentation de la durée de vie (passage de l'état SYN-SENT à ESTABLISHED pour TCP) de la communication et ceux la réduisant (passage de ESTABLISHED à CLOSING) comme indiqué en Figure 6.

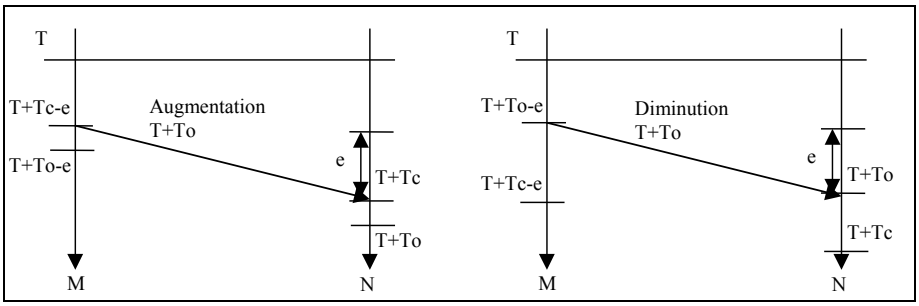

Figure 6. *Mise à jour des timers entre deux pare-feux M et N en fonction de l'opération réalisée et des valeurs des timers.*

Si au temps T on considère la nouvelle durée To et la durée restante Tc pour une communication alors les premiers événements doivent être signalés avant T+Tc afin d'éviter l'expiration du timer en N. Dans le second cas, la mise à jour doit être signalée avant min(T+To,T+Tc) afin de ne pas accorder plus de temps à la communication que la nouvelle valeur du timer (Tableau 4). Il faut par ailleurs considérer le temps de synchronisation (e) de telle sorte que la mise à jour n'arrive pas après l'expiration du timer sur le pare-feu homologue.

Tableau 4. Evènements liés aux timers et nécessitant une action.

Evénement	**Action**
Réception TCP SYN (Tc, To, T) Réception UDP (Tc, To, T) Etat non existant Réception ICMP requête (Tc, To, T)	Envoi mise à jour;
Réception TCP SYN-ACK (Tc, To, T) Réception TCP ACK (Tc, To, T) Réception UDP (Tc, To, T) Etat existant	Envoi mise à jour T+To-e;
Réception TCP FIN/FIN-ACK, RST (Tc, To, T)	Envoi mise à jour à min(T+To,T+Tc)-e;

L'exemple des timers nous montre que d'une manière générale, la décision de mise à jour devrait être séparée du contenu des données envoyées. Ainsi il est possible après avoir programmé une mise à jour à la date T que de nouveaux paquets de la même communication amènent à repousser la date d'expiration du timer. Cependant ces paquets ne doivent pas modifier la date d'envoi de la mise à jour qui dépend de la date d'expiration du timer dans le pare-feu homologue et non de la valeur d'expiration locale. L'action à réaliser à chaque évènement peut être déterminée en utilisant l'action la plus contraignante définie pour chaque type de contrainte (automate, intervalles, timer).

4.3. *Amélioration du rythme d'émission*

Afin de limiter le nombre de paquets générés par les mises à jour nous proposons d'adapter le rythme d'émission au type d'évènement. En effet, tous les événements ne nécessitent pas une même réactivité de mise à jour. Ainsi l'expiration des timers

peut être prévue et il importe peu par exemple dans le cas d'UDP que la mise à jour parte dans la milliseconde. Ceci montre que l'agrégation temporelle que l'on peut réaliser sur les états devrait également dépendre du type d'événement à signaler. Contrairement aux événements imprévisibles (ouverture d'une connexion) les événements prévisibles (expiration de timer, dépassement d'intervalles) peuvent être anticipés et agrégés. Nous proposons donc de lier le rythme d'émission des mises à jour au type d'événement protocolaire à signaler. Nous proposons trois files d'émissions.

- Les émissions "express" pour lesquelles le temps de conservation du paquet avant émission est inférieur à 1ms. Cette file est liée aux opérations de mise à jour de l'automate. La contrainte de temps liée à cette file pour deux pare-feux homologues M et N est la valeur minimale entre le temps d'aller retour entre M, la destination et N et le temps d'aller retour entre N, la source et M.
- Les émissions "moyennes" pour lesquelles le temps de conservation du paquet avant émission est de l'ordre de 10ms. Cette file est liée aux opérations de mise à jour de valeurs d'intervalles d'acceptation de segments. Il est à noter que le temps de mise à jour entre deux pare-feux homologues N et M doit être inférieur à la durée de réception par N de $D_a=|S-N_a|$ octets ou de $D_s=|A-N_s|$ acquittements. Pour un temps d'attente de 10ms, on peut représenter les valeurs possibles de D en fonction du débit unidirectionnel B de la connexion et du temps du temps de synchronisation e selon la formule suivante.

  $$B(D,e) = (8.D)/2.(e+10.10^{-3})$$

  En choisissant une valeur de D égale à la moitié de la valeur maximale de la fenêtre (32768), et une valeur de temps de synchronisation égale à e=1ms on obtient une valeur de B de l'ordre de 13Mb/s.
- Les émissions "lentes" pour lesquelles le temps de conservation du paquet avant émission est de l'ordre de 100ms. Cette file est liée aux opérations de mise à jour des timers. La contrainte de temps pour cette file est liée à la durée e et au timer de durée minimale.

4.4. *Evaluation du protocole avant agrégation*

Afin d'évaluer cette évolution du protocole *pfsync* pour la prise en compte du routage asymétrique, nous proposons d'évaluer le volume de trafic généré par le protocole de synchronisation. Nous évaluons tout d'abord le nombre d'événements tels que définis en section 4.2. Afin de définir le trafic associé à chaque protocole, nous utilisons un ensemble de traces NLANR capturées en 2003 et comprenant environ 100 millions de paquets ainsi que l'analyse des trafics Internet réalisée par Sprint dans le cadre du projet IPMON [SPR 04] en 2004. Nous utilisons une distribution des paquets par protocoles de 86% pour TCP, 10% pour UDP, 1% pour ICMP et 3% pour les autres protocoles. Au moyen des traces, nous pouvons définir la proportion de paquets liés aux événements précédemment définis (Tableau 5).

Tableau 5. Calcul du nombre d'évènements en fonction de son type.

Evénement	% paq. par proto.	% paq. total	Evènements dans un réseau 10Gb/s
Réception TCP SYN	5	4.3	172.10^3
Réception TCP SYN-ACK	2	1.7	68.10^3
Réception TCP FIN/FIN-ACK, RST	3	2.5	100.10^3
Réception UDP, Etat non existant	5	0.5	20.10^3
Réception ICMP requête	35	0.3	14.10^3
Réception ICMP réponse	27	0.2	10.10^3

Vis-à-vis des évènements dépendant des modifications d'intervalles, le nombre d'événements dépend d'une part de la valeur du timer associé aux connexions TCP. Cependant ce timer est tellement important dans la pratique (24 heures), que ce premier paramètre peut être considéré comme négligeable. L'autre paramètre est le nombre d'octets transportés W-D ou W est la taille de la fenêtre. Le nombre moyen d'octets transportés par les connexions TCP dans les traces examinées est de l'ordre de 11000 octets avec une taille moyenne des fenêtres utilisées W=25500. Nous examinons donc la proportion de communications transportant plus de données que W-D. Pour les valeurs (D = W/2) ces communications constituent 2% des connexions TCP et 6% des segments. Elles transportent en moyenne 206k octets dans le sens serveur-client et 19k octets dans le sens client-serveur. En utilisant ces paramètres et en supposant une taille de MSS de 1000 octets nous pouvons déterminer le nombre moyen de demandes de mise à jour par seconde. Ce résultat est donné dans le Tableau 6.

Tableau 6. Calcul du nombre d'évènements liés aux modifications d'intervalles.

Evénement	% paq. par proto.	% paq. Total	% paq. connexions longue	% paq. générant demandes	Evènements dans un réseau 10Gb/s
Réception TCP ACK	90	77	4.6	0.4	15.10^3

Concernant les mises à jour liées aux timers, le nombre de mises à jour dépend du nombre d'états créés pour les communications dont les évènements utilisent des timers. Les traces en notre possession ne nous ayant pas permis de définir une valeur réaliste pour la durée de vie moyenne des communications UDP. Nous utilisons les indicateurs calculés par Chang et al. [CHA 04] donnant une durée de vie moyenne de m=9 secondes. En utilisant la loi de Little, nous pouvons définir à partir de cette durée moyenne et du nombre de nouvelles communications UDP par seconde, le nombre moyen d'états. Celui-ci est de l'ordre de 180000. Celui-ci nous permet par la suite de déterminer le nombre de mises à jour par seconde en prenant une durée de mise à jour To=60s, e=1s (Tableau 7).

Tableau 7. Calcul du nombre d'évènements liés aux timers.

Evénement	Durée vie moyenne	Nouvelles com. par sec.	Nombre moyens états	Evènements dans un réseau 10Gb/s
Réception UDP, Etat existant	9s	$20.\ 10^3$	$180.\ 10^3$	$3.\ 10^3$

4.5. *Evaluation du système d'agrégation*

A partir du nombre d'évènements calculés dans la section précédente, nous pouvons évaluer le nombre de mises à jour par files. Les événements associés à une file sont stockés dans un paquet jusqu'à ce que, soit le paquet soit plein, soit la durée de stockage locale maximale du paquet soit atteinte. Si on suppose une utilisation maximale du réseau, la seconde contrainte n'est jamais atteinte. En supposant une MTU de 1500 octets et une taille de contexte similaire à celle utilisée dans *pf*, on peut alors déterminer le nombre de mise à jour par seconde et le débit global nécessaire.
Comme indiqué dans le Tableau 8, le volume global de trafic est de l'ordre de 550Mb/s pour un réseau de 10Gb/s. Comparé à une méthode naïve, notre approche permet donc une division par 10 du débit nécessaire en limitant le coût de la synchronisation à 5% du débit global.

Tableau 8. Nombre d'évènements et débit associé à chaque file.

File	Evènements par sec.	Evènements par durée de stockage	Paquets IP par durée de stockage	Paquets IP par sec.	Débit
Express	$384.\ 10^3$	384	43	43000	516Mb/s
Moyenne	$30.\ 10^3$	300	33	3300	39Mb/s
Lente	$3.\ 10^3$	300	33	330	4Mb/s

5. Conclusion

Dans ce document nous avons exploré les problèmes que pouvaient poser aux pare-feux le routage asymétrique lorsque le service de filtrage est rendu de manière contextuelle et lorsque celui-ci est composé de plusieurs équipements. Nous avons analysé les solutions possibles avec une attention particulière pour les techniques de synchronisation d'états. Nous avons montré que les protocoles actuels ne permettent pas de résoudre le problème posé par le routage asymétrique de manière adéquate. Nous avons donc montré comment ces protocoles pouvaient être améliorés afin de limiter d'une part la latence de mise à jour et d'autre part le débit généré par le protocole de synchronisation. Dans ce domaine, la prise en compte des types d'évènements dans la définition de l'agenda des envois ainsi que dans leur traitement dans le protocole de transport permet d'obtenir un système de synchronisation plus efficace.

6. Remerciements

Nous remercions Elwyn Davies pour ses commentaires.

7. Bibliographie

[STI 04] Stiemerling M., Tschofenig H., Martin M., Aoun C., NAT/Firewall NSIS Signaling Layer Protocol (NSLP), IETF draft, draft-ietf-nsis-nslp-natfw-04, octobre 2004

[THA 00] Thaler D., Hopps C., Multipath Issues in Unicast and Multicast Next-Hop Selection, IETF Informational document, RFC 2991, novembre 2000

[ROS 02] Rosenberg J., Schulzrinne H., Camarillo, G. , Johnston A., Peterson J., Sparks R., Handley M,, Schooler E., SIP: Session Initiation Protocol, IETF Standards Track, RFC 3261, juin 2004

[SRI 02] Srisuresh P., Kuthan J., Rosenberg J., Molitor A., Rayhan A., IETF Informational document, RFC 3303, Août 2002

[HAN 04] Hancock R., Karagiannis G., Loughney J., van den Bosch S., IETF draft, Next Steps in Signaling: Framework, draft-ietf-nsis-fw-06, juillet 2004

[SCH 04] Schulzrinne H,. Hancock R,, Internet Draft, GIMPS: General Internet Messaging Protocol for Signaling, draft-ietf-nsis-ntlp-04, octobre 2004

[SRI 99] Srisuresh P., Holdrege M., IP Network Address Translator (NAT) Terminology and Considerations, IETF Informational document, RFC 2663, août 1999

[HOL 01] M. Holdrege, P. Srisuresh, Protocol Complications with the IP Network Address Translator, IETF Informational document, RFC 3027, janvier 2001

[PF] PF. The OpenBSD Packet Filter, http://www.openbsd.org/faq/pf/

[LI 98] Li. T, Cole B., Morton P., Li D., Cisco Hot Standby Router Protocol (HSRP), IETF Informational document, RFC 2281, mars 1998

[BSD 03] carp - Common Address Redundancy Protocol, Open BSD man pages, http://www.openbsd.org/cgi-bin/man.cgi?query=carp, octobre 2003

[HIN 04] Hinden R., Virtual Router Redundancy Protocol (VRRP), IETF Standards Track, RFC 3768, avril 2004

[ROO 01] Guido van Rooij, Real Stateful Packet Filtering with IP Filter, Usenix Security 2001.

[SPR 04] Sprint Labs, IP Monitoring Project, disponible à http://ipmon.sprint.com.

[CHA 04] Approximate Packet Classification Caching, Francis Chang, Kang Li, Wu-chang Feng. IEEE INFOCOM 2004.