Migration des pare-feu Internet de l'Université de Rennes 1 sous OpenBSD / Packet-Filter

Patrick Lamaizière
Direction du Système d'Information / Université de Rennes 1
263, Av. du Général Leclerc

35042 Rennes cedex

Résumé

L'accès Internet de l'Université de Rennes 1 est protégé depuis bientôt trois ans par deux routeurs / pare-feu Packet-Filter (PF) sous OpenBSD, installés sur des serveurs standards (x86). Cet article présente la solution mise en place sous forme d'un retour d'expérience :

- *L'architecture et le fonctionnement de la redondance des pare-feu en mode master/backup à l'aide du protocole CARP (Common Address Redundancy Protocol)*
- *La gestion de BGP avec le réseau Renater (double attachement avec Renater)*
- *La migration des règles existantes de filtrage (depuis du matériel Cisco) vers PF*
- *Le monitoring et métrologie mis en place*
- *Un retour sur les performances obtenues*

Les perspectives d'évolutions de solutions basées sur PF seront abordées en fin d'article.

Mots-clefs

Pare-feu, sécurité, réseau.

1 Introduction

Jusqu'à fin 2010, la connexion Internet de l'UR1 était protégée par un routeur Cisco 7204 et deux pare-feu Cisco PIX. Ces équipements n'étant plus à même d'assurer la bande passante nécessaire (plafonnement à 200 Mbit/s), il a été envisagé un remplacement par des routeurs/pare-feu sur du matériel standard (serveurs x86). Une première étude a évalué un remplacement par la distribution Pfsense (qui est basée sur FreeBSD et Packet Filter). À l'issue d'une deuxième étude, un remplacement par OpenBSD a été retenu puisqu'il contenait tous les outils nécessaires.

1.1 OpenBSD

OpenBSD est un système d'exploitation libre de type Unix dérivé de 4.4BSD. Le but du projet est de fournir un système multi-plateforme et libre, avec un effort mis sur la sécurité : *free, functionnal and secure*. Les contributions les plus connues du projet sont OpenSSH, Packet Filter et CARP.

1.2 Packet Filter

Packet Filter (PF) est un pare-feu logiciel développé par OpenBSD en remplacement de IPFilter, à l'origine pour des problèmes de licence. Il est également intégré dans d'autres systèmes dérivés de BSD (FreeBSD, NetBSD), en général dans des versions moins récentes offrant moins de fonctionnalités.

PF est un pare-feu à *états* : lorsqu'une règle autorise un paquet, un état est automatiquement créé qui autorise les paquets suivants et les paquets retours. Dans le cadre de TCP qui est un protocole connecté, l'état est créé à l'initiation de la connexion (drapeau TCP SYN à 1 et drapeau ACK à 0). Pour UDP, un état est également créé qui autorise une réponse

pendant un cours laps de temps. Le filtrage de l'ICMP est automatique en fonction des connexions, par exemple si un état autorise un paquet TCP vers un hôte, les éventuels messages ICMP (port unreachable, ...) sont autorisés en retour. Les états sont stockés dans une table d'états et expirent au bout d'un laps de temps dépendant du protocole.

D'un point de vue pratique, cela simplifie l'écriture des règles (on n'a pas besoin de s'occuper des réponses) et permet de n'ouvrir que le strict nécessaire. D'un point de vue performance, il est beaucoup plus rapide de parcourir la table d'états que d'évaluer des règles. Ainsi un paquet qui fait partie d'une connexion déjà établie, ce qui constitue la majeure partie du trafic, est traité rapidement [1] .

PF n'est pas un pare-feu applicatif, il ne contrôle pas que le contenu des paquets correspond au protocole (de niveau 7) utilisé. Par exemple il ne vérifie pas qu'une connexion TCP sur le port 80 (http) correspond bien au protocole HTTP. Les ouvertures dynamiques de ports en fonction du protocole peuvent être implémentées par des *proxy* mais seul le protocole FTP est reconnu.

PF est assez riche en fonctionnalités, comme le montre la FAQ : http://www.openbsd.org/faq/pf/fr/.

1.3 Common Address Redundancy Protocol (CARP)

Le protocole CARP permet la mise en œuvre de pare-feu redondants, simplement en assignant l'adresse virtuelle comme passerelle pour les machines derrière ces pare-feu. Ces pare-feu font donc également office de routeurs.

CARP permet à un groupe d'hôtes présents sur le même segment de réseau de partager une adresse IP (IPv4 ou IPv6) virtuelle. Le groupe contient un maître qui « prend » l'adresse IP pendant que les autres hôtes sont à l'état d'esclave, prêts à récupérer l'adresse en cas de défaillance du maître.

CARP est également implémenté sur les dérivés de BSD, mais le protocole n'est pas normalisé et les implémentations ne sont pas forcément compatibles entre elles.

Il est également possible de faire de la redondance de pare-feu sans CARP, en utilisant des mécanismes de niveau deux [2] .

1.4 pfsync

Pfsync(4) permet de synchroniser les tables d'états entre plusieurs pare-feu utilisant PF. C'est le complément indispensable à la redondance pour maintenir les connexions ouvertes en cas de bascule d'un pare-feu à un autre.

2 Mise en œuvre de la solution

La configuration du réseau et les impératifs de production rendaient difficile une migration au fil du temps. Il a été décidé que la nouvelle solution serait déployée en réel, mais hors du réseau existant, et testée. Puis que le basculement sur les nouveaux pare-feu se ferait d'un coup lors d'une intervention planifiée en soirée, simplement en débranchant les anciens équipements pour mettre à la place les nouveaux. La configuration testée est donc en tout point identique à la configuration finale, ce qui limite le risque d'erreur. D'autre part il était facile de revenir rapidement sur l'ancienne infrastructure, en cas de gros soucis.

2.1 Architecture

La nouvelle infrastructure utilise deux pare-feu redondants en lieu et place d'un routeur Cisco CS7204 et de deux pare-feu Cisco PIX. La connexion avec Renater est doublée pour se prémunir de la panne d'un lien. Actuellement les deux liens sont raccordés sur le même routeur Renater mais sur des cartes différentes, avec en prévision de séparer réellement ces liens. D'un point de vue logique, on considère qu'on est connecté à deux routeurs.

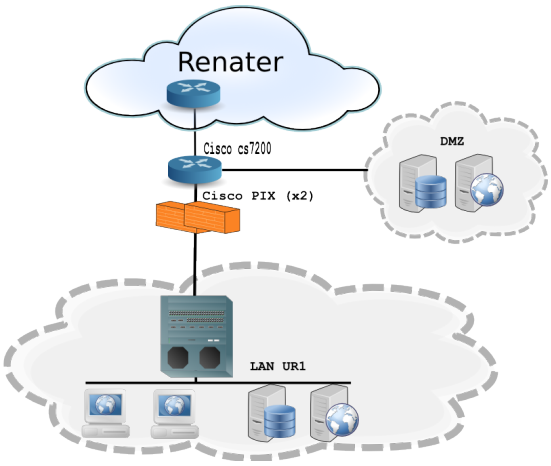

Figure 1: connexion internet 2011

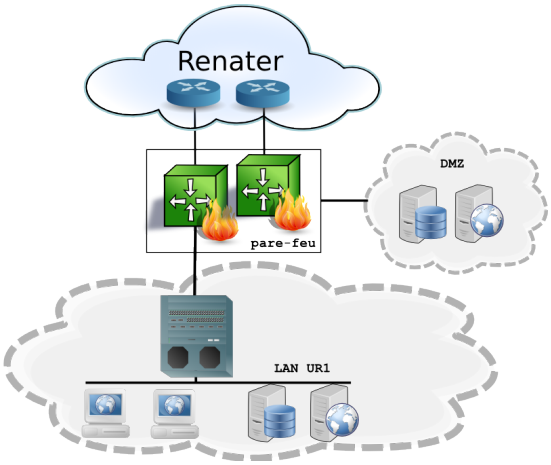

Figure 2: connexion PF

Les connexions avec Renater sont directes, sans switch, et utilisent un lien cuivre et un lien en fibre. Il n'y a pas de redondance CARP sur ces liens. Les liens internes sont raccordés via un switch, chaque paire de lien étant redondée avec CARP. Tous les liens sont en gigabits.

Sur la figure ci-dessous, les dénominations « emX » désignent le nom des interfaces utilisées (sous OpenBSD les noms d'interfaces dérivent du pilote de périphérique : em pilote Intel), les liens en rouge sont en fibre optique et ceux en orange en cuivre. Les pare-feu sont asymétriques au niveau de la connexion coté Renater (em3 pour l'un, em7 pour l'autre).

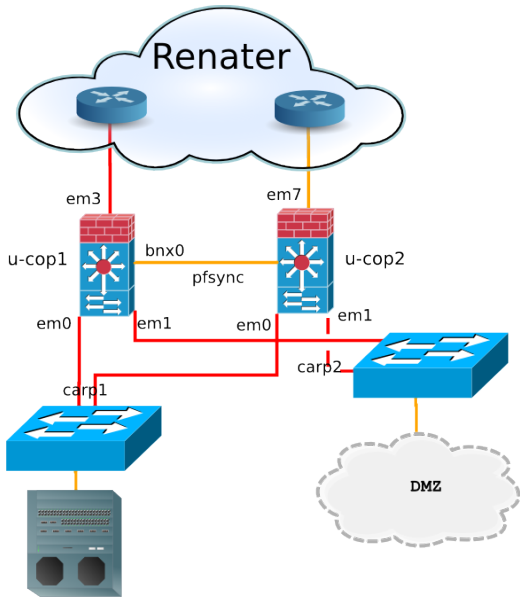

Figure 3: synoptique détaillé

2.2 Fonctionnement de la redondance des pare-feu

L'architecture ci-dessus ne peut se baser uniquement sur le seul mécanisme de CARP pour assurer la redondance. Du fait que CARP n'est utilisé que sur les pattes internes des pare-feu, il faut également surveiller les interfaces coté Renater. OpenBSD intègre à cet usage un démon *ifstated(8) Interface State daemon*, qui permet de réagir à la perte d'un lien, d'un changement d'état de CARP ou d'un test spécifique (surveillance de BGP par exemple).

D'autre part se pose le problème du routage : comment faire savoir à Renater sur quel pare-feu envoyer les paquets ? La solution consiste à utiliser les communautés BGP spécifiées par Renater : la communauté 2200:610 pour le chemin

primaire et la communauté 2200:590 pour le chemin secondaire (voir https://services.renater.fr/connectivite/routage_d_adresses).

Pour simplifier la solution, nous avons opté pour un fonctionnement de type normal/secours. Le pare-feu 1 (nommé u-cop1) est normalement le pare-feu actif : il est maître (CARP) et il annonce via BGP la communauté primaire. Le pare-feu 2 (u-cop2) est en secours, il est normalement esclave (CARP) et il annonce la communauté secondaire. Si pour une raison quelconque, on bascule sur le pare-feu en secours (u-cop2), u-cop1 cesse d'annoncer sa communauté (en tuant le démon BGP). Renater aiguille donc les paquets vers u-cop2.

Lorsque le secours est actif, u-cop1 *se dégrade* lui même en utilisant le compteur de dégradation CARP « *carp demote count* ». Ce compteur permet à un hôte CARP d'indiquer qu'il n'est pas prêt à passer maître. Nous avons fait le choix, une fois sur le secours, de ne pas repasser automatiquement sur le pare-feu normal mais par une intervention humaine. Ceci pour éviter d'éventuels problèmes de *flip-flop* entre les deux pare-feu.

Le retour sur le pare-feu normal se fait en démarrant le démon bgpd (ce qui fait aiguiller les paquets venant de Renater vers u-cop1), puis en remettant le compteur de dégradation à zéro (ce qui provoque une bascule de CARP vers u-cop1). Pendant un petit laps de temps, on a donc un routage asymétrique : les paquets sortent par u-cop2 et entrent par u-cop1.

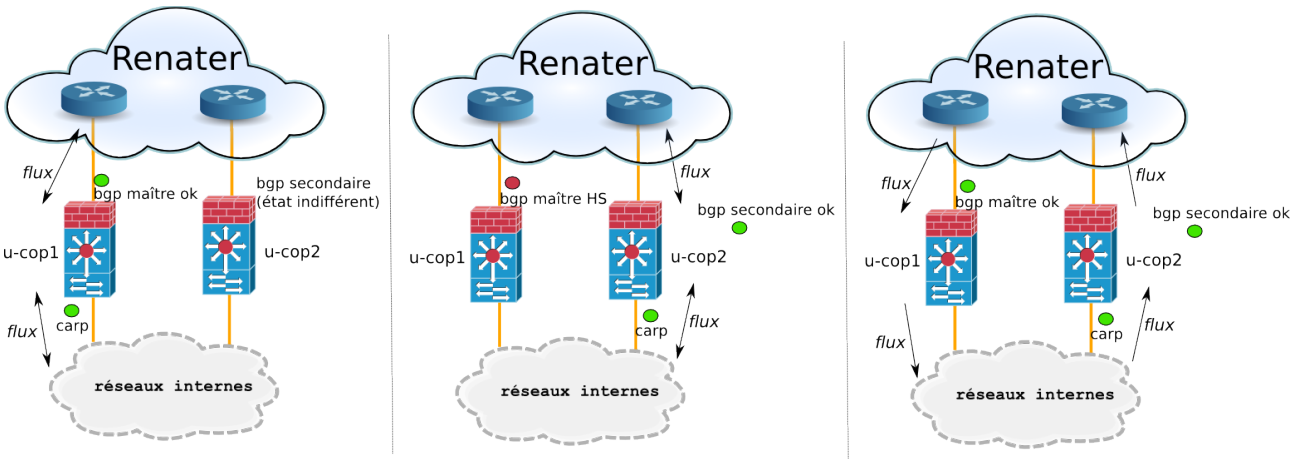

Figure 4: routage en fonction de BGP et CARP

Il n'y a jamais besoin d'intervenir sur le pare-feu secours, CARP suffi à la redondance. Sur le pare-feu normal, la surveillance est effectuée par ifstated(8) (surveillance bgpd, sessions BGP, interfaces).

2.3 Migration des règles de filtrage

La migration des règles de filtrages et les tests nécessaires sont des opérations lourdes. On ne souhaitait pas imposer aux administrateurs réseaux de maintenir deux jeux de règles, en raison du risque d'erreur et de la charge de travail supplémentaire.

La migration a donc été effectuée de manière semi-automatique avec les contraintes suivantes : les règles du routeur 7204 (en amont) sont migrées en entrée (in) sur l'interface reliée à Renater. Les règles du PIX (en aval) sont migrées en sortie (out) sur l'interface coté UR1.

À noter que ce placement des règles a l'inconvénient de surtout filtrer en sortie, ce qui n'est pas recommandé puisque le filtrage des paquets est effectué deux fois et charge inutilement le pare-feu. On devrait rejeter les paquets au plus tôt.

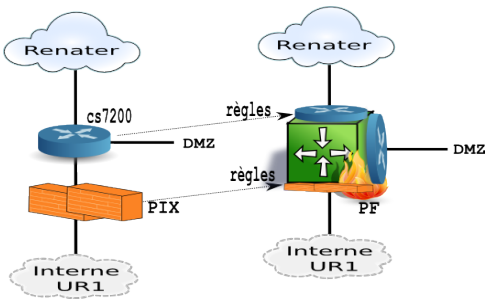

Figure 5: placement des règles

La conversion des règles a été effectuée avec une variante de l'outil lsfw [3] qui disposait déjà d'un interpréteur pour les règles Cisco. Il était assez facile de le modifier pour exporter les règles dans la syntaxe de PF. Les règles posant problèmes (par exemple une règle comportant un masque réseau non CIDR, ce que ne supporte pas PF) sont exportées telles quelles, mais mises en commentaire avec une balise. L'utilitaire patch(1) est ensuite utilisé pour corriger le jeu de règles obtenu.

Les objets utilisés dans les règles sont convertis avec les correspondances suivantes :

Objet Cisco	*Objet PF*
name	macro
service-group	macro
network-group	table

Pour contrôler la conformité des règles, un échantillon de règles jugées pertinentes a été sélectionné, puis testé automatiquement, toujours avec lsfw. Cet échantillon a également été testé à la main, en réel sur les pare-feu.

Sans entrer dans le détail de la conversion, il faut prendre en compte le fait que PF est un pare-feu à états, il n'est donc pas nécessaire de traiter les règles autorisant le retour des flux. Le routeur 7204 n'utilisant pas de règles à états, les retours de flux pour TCP étaient autorisés par une simple règle « permit tcp ACK », règle qu'il suffit de supprimer. Pour UDP il est difficile de différencier les règles qui autorisent un flux de celles qui autorisent leur retour, les règles ont donc été converties telles quelles. Ceci ne concernait que les règles issues du routeur puisque les PIX sont aussi des pare-feu à états. La politique au niveau de l'ICMP a également été retranscrite en l'état.

Même si les règles ainsi obtenues sont conformes aux anciennes, à terme il est prévu une réécriture pour éviter le filtrage en sortie du pare-feu, et profiter pleinement des capacités de PF.

3 Exploitation

3.1 Gestion des systèmes

La configuration des deux pare-feu est quasiment identique, mais avec quelques divergences (adresses IP, configuration BGP, ...). Pour réduire le risque d'erreur, il n'est pas souhaitable de maintenir ces deux configurations en parallèle. La configuration est gérée de la manière suivante : une machine virtuelle sous OpenBSD sert de template, et les fichiers de configuration de base sont utilisés tels quels. Les fichiers de configuration nécessaires aux pare-feu peuvent, par l'intermédiaire d'une marque « ##PAREFEU## », spécifier les lignes de configuration propres à chaque pare-feu. Les fichiers sont ensuite passés à un petit script shell qui produit les fichiers de configuration finaux.

Par exemple le fichier de configuration de l'interface bnx0, où seules les adresses IP diffèrent est décrit par :

```
# CRI plamaiziere 11/10/2010
 #@@template@@

description "PFSYNC"
##UCOP1## inet 192.168.255.253/30
##UCOP2## inet 192.168.255.254/30
 group "IFPFSYNC"
up
```

Des images de clefs USB sont montées sur le système et mises à jour en y copiant le système et ces fichiers de configuration. Il suffit ensuite de copier ces images sur des clefs USB et les installer sur les pare-feu.

Le projet OpenBSD maintient les deux dernières versions en publiant des correctifs, et sort une version tous les six mois. Une version est donc supportée un an au maximum. Nous avons fait le choix de suivre la version « -1 » de OpenBSD, en espérant que les problèmes majeurs ont été corrigés par des erratas.

La mise à jour des pare-feux s'effectue en production, en commençant par le pare-feu « normal ». Il faut tenir compte de la compatibilité entre les versions sur le pilote *pfsync(4)* de transfert des états. En cas d'incompatibilité, les sessions seront perdues.

Le changement des règles de pare-feu s'effectue par un script à l'usage des administrateurs réseau qui, via ssh, envoie le fichier des règles *pf.conf(5)* sur les pare-feu, vérifie la syntaxe, charge les règles et effectue une sauvegarde sur la machine virtuelle qui sert de template.

3.2 Métrologie et surveillance

La métrologie des flux est effectuée par une sonde Netflow, en utilisant le pilote *pflow(4)*. Ce pilote exporte les flux au format Netflow. Les flux sont ensuite collectés sur un serveur utilisant NfSen. Le support des flux IPv6 étant manquant à l'époque de la mise en production des pare-feu, des essais avec des sondes Netflow tournant en mode utilisateur ont montré qu'elles étaient incapables de tenir la charge. Le support des flux IPv6 a été ajouté depuis.

La surveillance s'effectue avec Nagios pour les alarmes et Cacti pour la visualisation. Les statistiques sont remontées en SNMP par le démon *net-snmpd(8)*, en utilisant son mécanisme d'extension ou par les MIB standards (interfaces).

Les statistiques collectées à partir des interfaces sont : le trafic en Mbits/s et en paquets/s, ainsi que les taux d'erreurs (Ierr/Oerr). Le taux d'erreur en entrée indique une éventuelle saturation de la queue en entrée de l'interface.

Les statistiques liés à Packet Filter sont extraites via *pfctl(8)*: le nombre d'états, le nombre de congestions, et le volume traité par PF.

Le nombre d'états est très important car si la table d'états est pleine, tout nouveau flux sera bloqué. Le nombre de congestions indique une saturation du pare-feu. Si la queue en entrée des interfaces est pleine, PF entre pendant un cours laps de temps en mode congestion. En mode congestion les règles ne sont pas évaluées, seuls les états le sont, ceci pour permettre de vider la queue.

3.3 Performances

Les serveurs utilisés sont des Dell R610 (processeur Xeon à 2,27 GHz.), équipés de trois cartes réseaux giga bits : une carte 4 ports fibre Intel PRO/1000 QP (82571EB), une carte 4 ports cuivre Intel PRO/1000 QP (82575GB) et la carte intégrée dans les R610 Broadcom BCM5709. OpenBSD est utilisé en version 64 bits, en mono-processeur du fait que le noyau d'OpenBSD ne supporte pas le multi-processeurs SMP (présence d'un verrou de type « *Giant Lock* » sur le noyau).

Sur l'interface connectée à Renater, où il y a le plus de trafic, le trafic en pointe est de l'ordre de 560 Mbits/s (66 kPaquets/s) en entrée, 180 Mbits/s en sortie (48 kPaquets/s). En sortie, la disproportion entre la bande passante et le nombre de paquets vient du fait qu'il s'agit surtout de paquets TCP ACK (majoritairement du trafic web). Ce qui est important sur un pare-feu n'est pas la bande passante, mais le nombre de paquets traités.

La courbe sur la congestion indique que, parfois, la queue d'entrée de PF sature. La congestion ne semble pas liée au trafic, je suppose qu'elle se produit lors d'une pointe d'activité sur plusieurs interfaces en même temps. Dans ce cas le système sature.

Le nombre d'états en pointe est de l'ordre de 400 000.

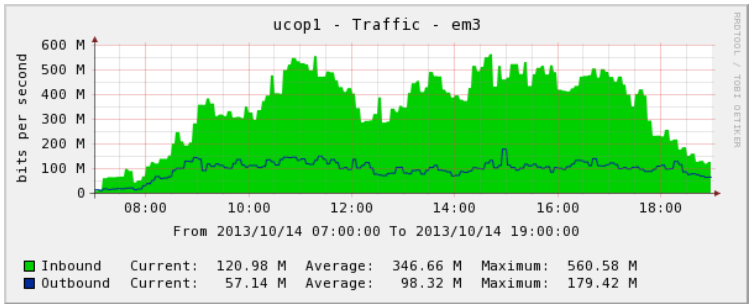

Figure 6: trafic Mbits/s

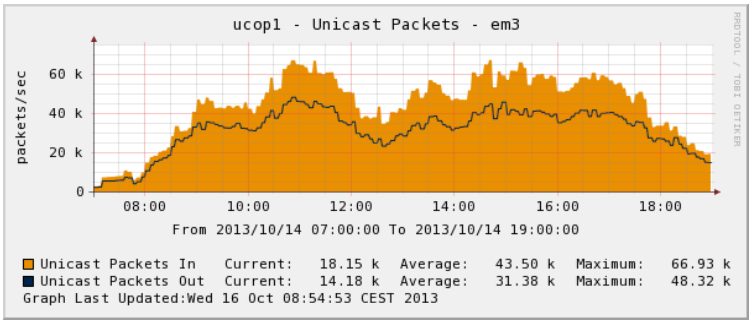

Figure 7: trafic paquets/s

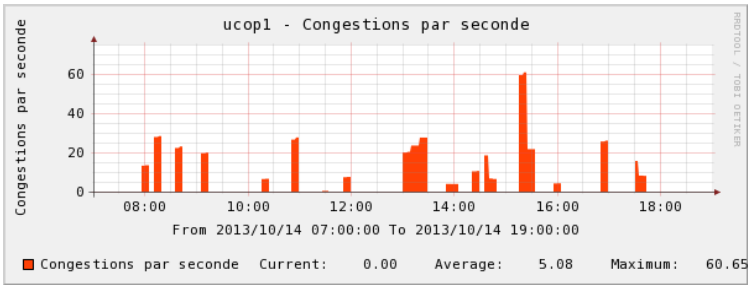

Figure 8: nombre de congestions par seconde

3.4 Évolution

Vu les performances obtenues, il est sûr que nous arrivons à limite du système (le trafic a aussi très sensiblement augmenté depuis la mise en production en 2011).

La première idée serait d'utiliser du matériel plus performant (au niveau UC et cartes réseaux), mais je doute qu'on puisse faire beaucoup mieux que le giga bits sous OpenBSD et ceci tant que le noyau ne sera pas SMP (*Symmetric multiprocessing*). Il y a des travaux en cours sur ce sujet mais les résultats seront longs à venir.

Une autre solution à évaluer est de passer à FreeBSD (ou un dérivé) qui a l'avantage d'être SMP depuis plusieurs années. Mais son implémentation de Packet Filter souffre également d'un verrou global (pour faciliter l'intégration depuis OpenBSD). Une nouvelle version de PF avec un verrouillage fin est présente dans FreeBSD 10, ce qui devrait améliorer sensiblement les performances. FreeBSD 10 est actuellement en version *alpha*.

L'avantage de OpenBSD est d'inclure toutes les fonctionnalités nécessaires à un routeur/pare-feu. Ces fonctionnalités ne sont pas forcément présentes ailleurs. Ci dessous un tableau des fonctionnalités en fonction des systèmes

	OpenBSD	*FreeBSD*
IP virtuelle	carp(4)	carp(4) (réécriture en 10, quelques problèmes sur les versions précédentes)
Synchronisation des états	pfsync(4)	pfsync(4)
Netflow	pflow(4)	ng_netflow(4)
BGP	OpenBGPd(8)	plusieurs démons BGP dans les ports (dont OpenBGPd) Signature TCP MD5 des connexions BGP via ipsec(4)
Surveillance des interfaces	ifstated(8)	devd(8), ifstated(8)

3.5 Conclusion

L'expérience à l'UR1 a montré que OpenBSD est un bon système pour la mise en œuvre de pare-feu redondants. L'architecture est robuste en production et l'exploitation facile de part la syntaxe claire de PF. Le seul bémol est en terme de performance. Si nécessaire une alternative se basant sur FreeBSD (mais pas avant 10.0 voire 10.1) peut être envisagée.

Bibliographie

[1] Daniel Hartmeier. Design and Performance of the OpenBSD Stateful Packet Filter (pf). Usenix 2002. http://www.benzedrine.cx/pf-paper.html

[2] Matthieu Herrb. Outils de sécurité réseau avec OpenBSD et PF. Dans conférence JRES2011, Toulouse, page 3. https://2011.jres.org/archives/130/paper130_article.pdf

[3] Patrick Lamaizière. lsfw : outil de test de règles de pare-feux distribués sur un réseau. Dans Actes du congrès JRES2011, Toulouse, Novembre 2011. https://2011.jres.org/archives/108/paper108_article.pdf