Outils de collecte pour réseaux gigabits

Une alternative à la technologie Cisco Netflow

David Rideau
Département Réseau du CICG (Centre Inter-Universitaire de Calcul de Grenoble).
351 Avenue de la Bibliothèque – Domaine Universitaire – 38041 Grenoble Cedex 9
david.rideau@grenet.fr

Date : 7 octobre 2003

Résumé

Cet article présente le projet d'une sonde polyvalente pour réseaux à haut débit. Initié au Centre Interuniversitaire de Calcul de Grenoble (CICG) l'année passée, le développement actuel s'est orienté vers une première application concrète, celle de générer des Netflows à partir des flux de données IP collectés. Ce programme complet et autonome permet aujourd'hui de se détacher d'un matériel spécifique, tout en obtenant le même type de données, utilisables par différentes plateformes de métrologie. Outre les détails d'implémentation, nous verrons que la mise en place de plateformes de tests sur les réseaux grenoblois et valentinois, a permis de valider les informations collectées par notre programme, et de mettre en évidence l'importance du point de collecte du trafic. De ce constat, découlent de nouvelles applications potentielles d'analyse d'un réseau en plusieurs points, sans négliger toutes les possibilités d'analyse du trafic en temps-réel, par recoupement d'informations au sein d'un ensemble de flux de données.

Mots clefs

Netflow, Gigabit, Ethernet, Métrologie, Sécurité

Une alternative à la technologie Cisco Netflow ?

Avons-nous ici la prétention de dire que la technologie Cisco Netflow[1] n'est pas bonne ? Non. Au contraire. Bon nombre d'applications se basent actuellement sur cette technologie pour qualifier et quantifier le trafic IP d'un réseau. Au point que l'Internet Engineering Task Force (IETF) a choisi le nouveau format des Netflows Version 9 pour définir un standard appelé IPFIX (Internet Protocol Flow Information eXport), avec l'objectif de définir un format suffisamment modulable pour qu'il puisse être utilisé sur tout type de matériels. Les applications qui se baseront sur ce format pour collecter les flux IP d'un réseau pourront donc se détacher d'un type de matériel donné.

Cependant, qu'on les nomme Netflows chez Cisco, ou différemment, leur génération sous forme de datagrammes UDP dépend systématiquement d'un équipement de réseau, dont ce n'est pas forcément la fonction première, et pour qui cette tâche supplémentaire peut s'avérer coûteuse. Ainsi naît l'idée de créer une sonde logicielle, embarquée sur une machine linux, que l'on peut aisément placer et déplacer dans un réseau, pour confronter éventuellement des résultats entre plusieurs points de capture, sans ne jamais perturber la fonction première des noeuds de notre réseau. La première mission que l'on confiera à notre sonde sera de générer des tickets Netflows de version 5, afin de pouvoir utiliser les produits de métrologie de la communauté universitaire, à savoir NetMET[2], développé par le Centre Interuniversitaire des Ressources Informatiques de Lorraine (CIRIL) à Nancy, et NetSEC[3], développé par le Centre Interuniversitaire de Calcul de Grenoble. Après quelques mois de labeur, cet objectif est aujourd'hui atteint et validé par comparaison avec les "vrais" Netflows issus du Cisco 6500 qui garde l'entrée du réseau interuniversitaire grenoblois et qui traite quotidiennement environ 300 Gigaoctets de données.

Notre sonde se veut donc autonome, capable de supporter les hauts débits actuels et futurs, et ouverte aux évolutions de format. D'ici là, elle prouvera peut-être qu'elle peut exploiter les données qu'elle collecte à des fins sécuritaires plus immédiates et, pourquoi pas, devenir une alliée potentielle du matériel dont elle s'est détachée.

---

[1] Netflows Cisco (http://www.cisco.com/warp/public/732/Tech/nmp/netflow/netflow_learnabout.shtml)
[2] Network's METrology (http://www.netmet-solutions.org)
[3] Network's SECurity (mailto:netsec@grenet.fr)

1 Définitions et objectifs

Le concept de flow est utilisé par Cisco pour optimiser, au sein de ses équipements de routage, le traitement des flux de données. En effet, l'échange entre deux machines est généralement constitué de plusieurs trames de données, et le calcul effectué pour le premier paquet doit pouvoir être réutilisé pour les paquets suivants, afin de minimiser les temps de calcul du routeur. Un système de cache est donc mis en oeuvre, afin de mémoriser la route calculée. L'article d'Alexandre Simon sur NetMET[1] décrit parfaitement ce mécanisme. Les routeurs Cisco offrent la possibilité d'exporter le contenu de ce cache, sous forme de datagrammes UDP, qui regroupent plusieurs flows. Les plateformes de métrologie peuvent donc réceptionner ces tickets Netflow, comme décrit à la Figure 1.

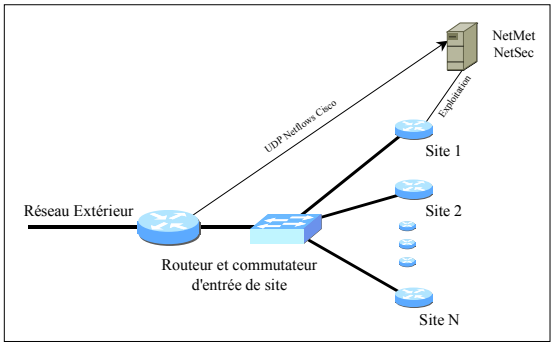


Figure 1 – *Architecture du réseau*

1.1 Qu'est-ce qu'un flow ?

Définissons tout d'abord le *flow*, ou *flux de données* en français, comme un échange de données entre deux machines d'un réseau IP. Il est unidirectionnel, ce qui implique que l'échange complet entre deux machines est caractérisé par deux flows symétriques, chacun identifié de manière unique par la clé formée des cinq informations suivantes :

- **`SrcAddr`** et **`DstAddr`**, les adresses IP sources et destinations des deux machines concernées,
- **`Protocol`**, le protocole utilisé (TCP, UDP, ICMP, etc.), information contenue dans la couche IP,
- **`SrcPort`** et **`DstPort`**, les ports sources et destinations, pour les segments TCP et les datagrammes UDP.

Le routeur y adjoint plusieurs informations : certaines d'entre elles pour mieux router les paquets concernés, avec notamment les champs **`Input`** et **`Output`** (index SNMP des interfaces d'entrée et de sortie du flow) ; et d'autres qui sont bien utiles pour nous, métrologues :

- **`DPkts`**, le nombre de paquets comptabilisé pour ce flow,
- **`DOctets`**, le volume d'octets de niveau 3 (couche IP) qu'a représenté ce flow,
- **`First`** et **`Last`**, respectivement le début et la fin du flow, en millisecondes depuis le démarrage du routeur.

Le routeur initie un flow lorsqu'il traite le premier paquet, et cumule ensuite sur ce flow tous les autres paquets, en mettant à jour les champs `dPkts`, `dOctets`, et `Last`. Le flow durera tant que le routeur n'aura pas détecté sa fin dans la couche transport. Cependant, il est possible de lui demander d'exporter l'état courant du flow, lorsqu'un certain temps depuis sa création s'est écoulé. Le routeur marquera alors ce flow de manière spéciale, en mettant le champ `Last` à `-1`. C'est grâce à ce système de "timeout" qu'il est possible de se baser sur les flows pour obtenir une vue précise du trafic dans le temps.

1.2 Le Netflow Version 5

Il reste ensuite à encapsuler le maximum de flows possible dans un datagramme UDP qui ne dépasse pas la taille limite d'une trame de la couche liaison (en ethernet 1500 octets), afin d'éviter qu'il soit segmenté. Dans l'exemple des Netflows de version 5*[2]*, l'en-tête du Netflow, décrit à la Figure 2, représente 24 octets.

Outre les données techniques nécessaires au bon décodage du Netflow, comme le champ **`Version`** et le champ **`Count`**, l'en-tête nous renseigne sur la référence de temps du routeur, avec les champs **`SysUptime`** et **`UnixSecs`**, informations qui nous permettront de déduire à quel moment s'est déroulé chaque flow. Enfin, nous vérifierons la cohérence des différents Netflows reçus, grâce au champ **`FlowSequence`**, puisque le protocole UDP n'offre aucun moyen de l'assurer.

Octets	Nom du Champ	Description
0-1	Version	Numéro de Version de Netflow
2-3	Count	Nombre de flows exportés dans ce paquet (1-30)
4-7	SysUptime	Temps en millisecondes depuis le démarrage de la machine
8-11	UnixSecs	Nombre de secondes écoulés depuis le 1er janvier 1970 : Timestamp Unix
12-15	UnixNSecs	Nombre de nanosecondes résiduelles depuis le 1er janvier 1970
16-19	FlowSequence	Compteur du nombre total de flows exportés
20-23	Reserved	Octets réservés

Figure 2 – *En-tête d'un Netflow V5*

1.3 Générer des Netflow V5

Nous optons pour le format de Netflow V5, car il est le plus "rentable" vis à vis du nombre de flows exporté par datagramme UDP. Par ailleurs, ce format est supporté par NetMET et par NetSEC, ce qui assurera la compatibilité des programmes. Chaque flow représente 48 octets, comme le montre la Figure 3, dans laquelle les champs que nous prendrons plus tard en considération sont grisés.

Octets	Nom du Champ	Description
0-3	SrcAddr	Adresse IP Source
4-7	DstAddr	Adresse IP Destination
8-11	NextHop	Adresse IP du prochain routeur
12-13	Input	Index SNMP de l'interface d'entrée
14-15	Output	Index SNMP de l'interface de sortie
16-19	DPkts	Nombre de paquets dans le flow
20-23	DOctets	Nombre total d'octets de couche 3 dans les paquets du flow
24-27	First	SysUptime au début du flow
28-31	Last	SysUptime à la fin du flow
32-33	SrcPort	Port Source TCP/UDP ou équivalent
34-35	DstPort	Port Destination TCP/UDP ou équivalent
36	pad1	Non utilisé (zero)
37	TcpFlags	OU cumulé des flags TCP
38	Prot	Type de Protocole IP (exemple TCP=6 ; UDP=17)
39	Tos	Type de service IP
40-41	SrcAS	Numéro d'AS de la Source
42-43	DstAS	Numéro d'AS de la Destination
44	SrcMask	Masque de l'adresse Source
45	DstMask	Masque de l'adresse Destination
46-47	Pad2	Non utilisé (zero)

Figure 3 – *Corps d'un Netflow V5*

Ainsi, un datagramme UDP de 1464 octets correspondant à une trame ethernet de 1500 octets, permettra de concentrer 30 échanges de la couche IP. Si, dans le pire des cas, chacun de ces échanges ne correspond qu'à un seul paquet, soit au minimum une trame de la couche liaison, alors nous obtenons déjà un taux de réduction de 1 pour 30. En désignant par n, le nombre moyen de paquets cumulés sur un même flow, nous définissons le taux de réduction tr = 1/30n.

Toutes les informations dont la sonde a besoin pour générer des Netflow V5 sont rassemblées dans les couches réseau et transport, auxquelles on accède par "désencapsulages" successifs de la trame ethernet capturée. Les champs `Input` et `Output`, qui donnent une information sur le sens du trafic, sont les seuls à ne pas pouvoir être obtenus de manière immédiate. Mais nous ferons intervenir la couche ethernet pour y pallier. Par contre, pour cette première version de notre sonde, nous considérons qu'il nous sera difficile d'analyser davantage les différents flows pour détecter leur fin. En effet, le temps consacré au traitement de chaque trame capturée doit rester limité pour supporter de hauts débits. Nous opterons donc pour une exportation des flows qui sera uniquement fonction du temps, l'équivalent du "timeout" sur le routeur Cisco. Ce choix n'a pas d'impact, ni sur la volumétrie, ni sur le début et la fin de chaque flow ; il retarde simplement l'export de certains d'entre eux. Nous étudierons l'impact de différents paramétrages afin de s'approcher au plus près du comportement d'un routeur Cisco.

2 Mise en oeuvre

Après avoir étudié différentes positions de la sonde sur le réseau, nous présentons le fonctionnement multi-tâche du programme, ainsi que la structure de données implémentée pour répondre efficacement aux requêtes de chaque processus.

2.1 Architecture matérielle

Comme le montrait la Figure 1 au paragraphe 1, nous travaillons généralement sur un réseau qui a une topologie en étoile, avec un lien vers l'extérieur et plusieurs liens intérieurs. Pour que la sonde reçoive une copie des trames qui transitent sur le lien extérieur, nous utilisons les techniques de "mirroring" disponibles sur la plupart des commutateurs actuels.

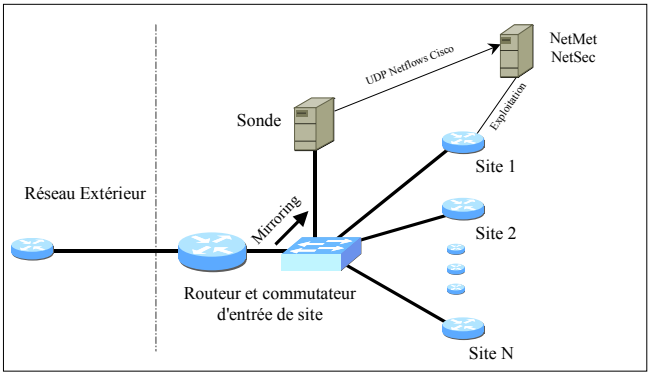


Figure 4 – *Mise en place de la sonde à l'intérieur du réseau*

À La Figure 4, nous décrivons un premier cas d'architecture, lorsque nous sommes en présence d'un routeur d'entrée de site, auquel on adjoint un commutateur pour réaliser l'étoile. Sur le commutateur, nous dupliquons donc le trafic du port qui relie le commutateur au routeur.

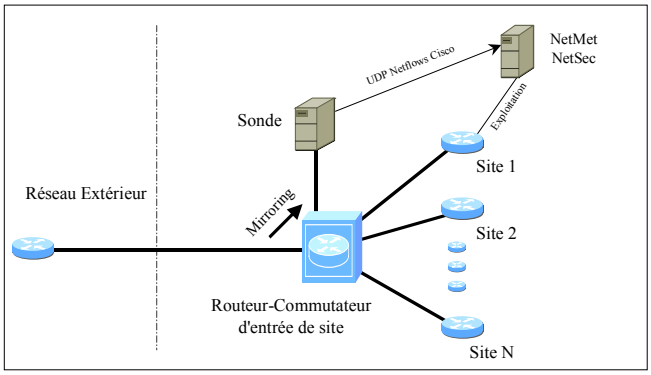


Figure 5 – *Mise en place de la sonde à l'extérieur du réseau*

Dans le deuxième cas d'architecture décrit à la Figure 5, un seul équipement rassemble les deux fonctionnalités de routage et de commutation. Dans ce cas, c'est le port correspondant au lien avec le réseau extérieur que nous devons dupliquer. Cette nuance sur la position du point de collecte est capitale au regard de l'activité première du routeur, qui, en plus de router les flows, en élimine certains.

Au niveau matériel, la machine utilisée doit être dimensionnée pour recevoir tout le trafic du brin principal. Il faut donc qu'elle dispose du même type d'interface réseau que celle du routeur d'entrée de site. Par ailleurs, il est nécessaire de vérifier que les bus internes de la machine sont capables de transférer les données suffisamment rapidement. Cette précision est importante car, actuellement, même si on peut trouver des cartes ethernet au Gigabit, tous les bus ne supportent pas un tel taux de transfert. Il en va de même pour la puissance de traitement du processeur et de la capacité mémoire de la machine, puisque son objectif est de stocker des flows dont le nombre variera probablement en fonction de l'importance du trafic.

2.2 Séparation des tâches

Comme le montre la Figure 6, nous allons devoir effectuer deux traitements indépendants : d'un côté, la collecte des paquets qui doit rechercher, créer et mettre à jour les flows ; de l'autre, l'envoi de tickets Netflows V5 en supprimant les flows exportés dans l'ordre où ils ont été créés.

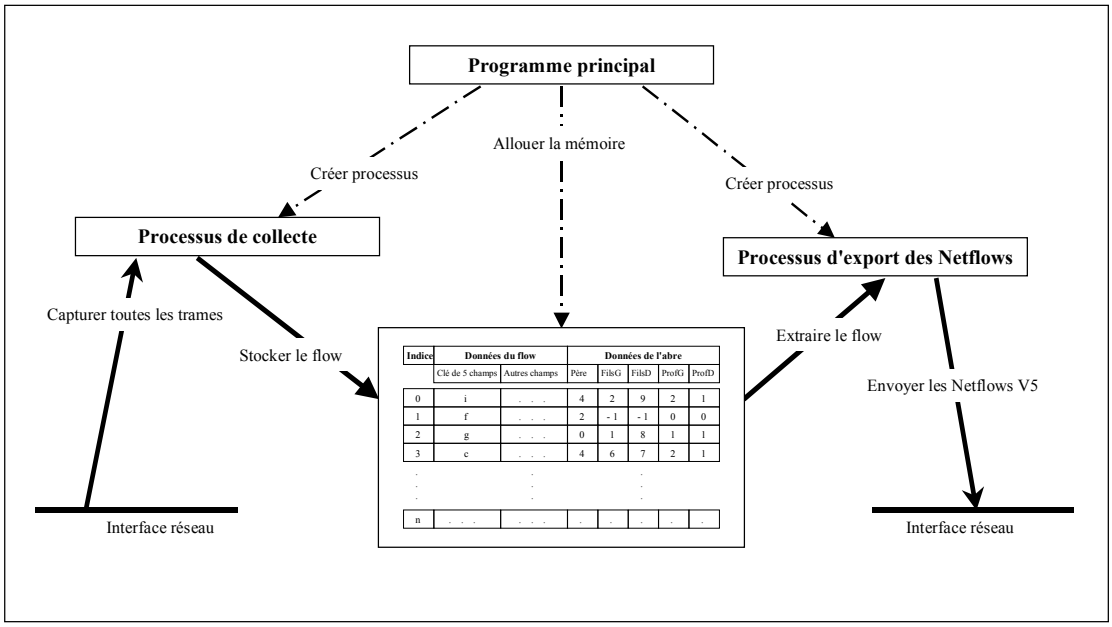


Figure 6 – *Architecture du programme*

Le processus de collecte utilise la librairie libpcap[4] pour ouvrir une socket de type RAW, qui capture toutes les trames de niveau 2. Cela implique que l'exécution du programme soit opérée par le compte utilisateur *root*. Pour tous les paquets IP (le concept de flow n'a pas de sens pour les autres protocole de niveau 3), il faut extraire le paquet IP encapsulé dans la couche liaison pour, d'une part relever les adresses IP source et destination, et le protocole de niveau 4 (ICMP, TCP, UDP, autres), et d'autre part calculer le nombre d'octets de niveau 3 contenus dans ce paquet. Pour les paquets UDP et les segments TCP, nous irons chercher dans la couche transport les ports source et destination. Pour les autres protocoles de niveau 4, ces champs seront mis à zéro. C'est dans l'en-tête du paquet remonté par la librairie libpcap que nous trouverons la référence du moment où la trame a été capturée. Le processus testera ensuite l'existence d'un flow identique. Si celui-ci existe, il cumulera le nombre d'octets de ce nouveau paquet et mettra à jour le champ `Last` correspondant à la fin du flow ; et s'il n'existe pas, il demandera sa création.

Dans le même temps, le processus d'export des Netflows vérifiera régulièrement si certains flows doivent être exportés, en fonction de leur durée de vie ("timeout"), définie par l'utilisateur. Tous les flows à exporter seront marqués comme tels, afin qu'ils ne soient plus utilisés par le processus de collecte concurrent. Ils seront ensuite assemblés dans une structure de Netflows que le processus d'export pourra transmettre au client, au moyen d'une socket DGRAM (mode non connecté UDP). Le contrôle, effectué périodiquement au moyen d'une attente passive, permet ainsi de garantir l'export des Netflows en fonction du temps.

[4] http://www.tcpdump.org

Enfin, nous demanderons au programme principal de générer régulièrement des informations statistiques sur les différentes étapes de traitement des flows. En effet, c'est à partir de ces données que nous pourrons, au paragraphe 3, analyser le comportement de notre programme. Mais déjà, on anticipe aisément la nécessité d'optimiser le temps d'accès à l'information, compte-tenu du volume de données auquel nous allons faire face.

2.3 Structure de données adaptée

Comme l'a décrit la Figure 6 auparavant, l'espace mémoire nécessaire est alloué, en mémoire partagée IPC (Inter Processus Communications) par le programme principal, avant d'être transmis au contexte des deux processus fils qui effectueront chacun des opérations sur le même espace mémoire. Grâce à un système de sémaphores, certains accès se dérouleront en exclusion mutuelle pour garantir la cohérence des données. Notons qu'il vaut mieux exclure une structure de données "totalement" dynamique, dont les requêtes d'allocation mémoire au système sont très coûteuses. Le programme principal a donc la charge de créer un espace mémoire de taille fixe, dans lequel nous devrons tourner en boucle.

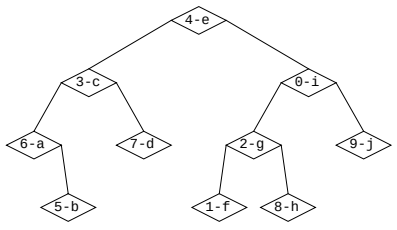
Sur ce modèle, nous pouvons implémenter le principe de tableau circulaire que nous utiliserons pour stocker des flows créés au fil du temps, pour ensuite les exporter dans ce même ordre. Malheureusement, tester l'existence d'un flow nécessite de le comparer avec les flows existant, selon la clé d'identification définie en Figure 7. Or, les flows stockés en mémoire ne sont pas triés selon ce critère.

Octets	Nom du Champ	
0-3	`SrcAddr`	8 octets
4-7	`DstAddr`	
8-9	`SrcPort`	8 octets
10-11	`DstPort`	
12	`Prot`	
13	`Pad8`	
14-15	`Pad16`	

Figure 7 – *Clé d'identification d'un flow*

Nous pouvons réduire à deux le nombre de comparaisons nécessaire en arrondissant artificiellement la taille de la clé, à l'aide des champs `Pad8` et `Pad16`, qui seront toujours nuls. Mais nous ne pouvons pas envisager de tester linéairement tous les flows existant pour chaque paquet capturé. D'où la nécessité d'adjoindre, à chaque flow du tableau circulaire, des informations de chaînage de type père-fils qui feront référence aux indices des flows dans ce même tableau, afin de créer un arbre binaire de recherche, dans lequel les flows seront triés selon la clé de la Figure 7 ci-dessus. En implémentant les techniques d'équilibrage par rotation des arbres AVL[4], toutes les opérations sur les flows s'effectueront désormais en un temps logarithmique dans le pire des cas. Seul l'indice du noeud racine doit être conservé en dehors du tableau circulaire, comme dans l'exemple de la Figure 8, où la racine est le flow numéro 4.

Indice	Données du flow		Données de l'abre				
	Clé de 5 champs	Autres champs	Père	FilsG	FilsD	ProfG	ProfD
0	i	. . .	4	2	9	2	1
1	f	. . .	2	-1	-1	0	0
2	g	. . .	0	1	8	1	1
3	c	. . .	4	6	7	2	1
4	e	. . .	-1	3	0	3	3
5	b	. . .	6	-1	-1	0	0
6	a	. . .	3	-1	5	0	1
7	d	. . .	3	-1	-1	0	0
8	h	. . .	2	-1	-1	0	0
9	j	. . .	0	-1	-1	0	0



avec a < b < c < d < e < f < g < h < i < j

Figure 8 – *Tableau de flows et représentation de l'arbre AVL associé*

3 Premiers résultats

Pour analyser le fonctionnement de notre programme, nous commencerons par quantifier les ressources qu'il utilise pour collecter les données d'un réseau de grande envergure, comme celui des campus grenoblois. Outre l'impact sur la montée en charge, ces tests démentiront certaines idées reçues sur le rapport entre le volume de trafic capturé et le nombre de flows gérés par le programme. Dans un deuxième temps, nous utiliserons NetSEC sur le réseau valentinois pour qualifier le trafic et mettre au jour les écarts constatés entre la collecte effectuée par le routeur d'entrée de site Cisco, et celle de notre programme. Enfin, nous validerons la compatibilité de notre programme avec la plateforme NetMET, et nous comparerons, de la même manière qu'avec NetSEC, les données issues des deux outils de collecte.

3.1 Montée en charge

C'est sur la plaque grenobloise que nous avons effectué les premiers tests en situation de trafic réel, à l'aide d'une machine de type PC bi-processeur à 2Ghz, équipée de 2 Gigaoctets de mémoire vive, et de deux interfaces ethernet gigabit. L'architecture du réseau grenoblois est basée sur le modèle décrit à la Figure 5 du paragraphe 2.1. Le routeur d'entrée de site grenoblois est un switch-routeur Cisco Catalyst 6500.

Le graphique de la Figure 9 ci-dessous représente l'évolution du nombre de flows traités au cours d'une journée, où le trafic moyen fut de 47 Mbits/s et le maximum atteint fut de 155 Mbits/s (cette courbe n'est pas représentée sur le graphique, mais son allure est identique à celle du nombre de flows traités, à un facteur près). Notre programme a donc fait face à un rythme moyen de 10 000 flows/sec, et un pic maximum de 25 000 flows/sec. Cependant, nous savons qu'il n'a jamais dû traiter plus de 270 000 flows simultanément, ce nombre étant davantage fonction de la durée de vie des flows paramétrée par l'utilisateur que du volume du trafic.

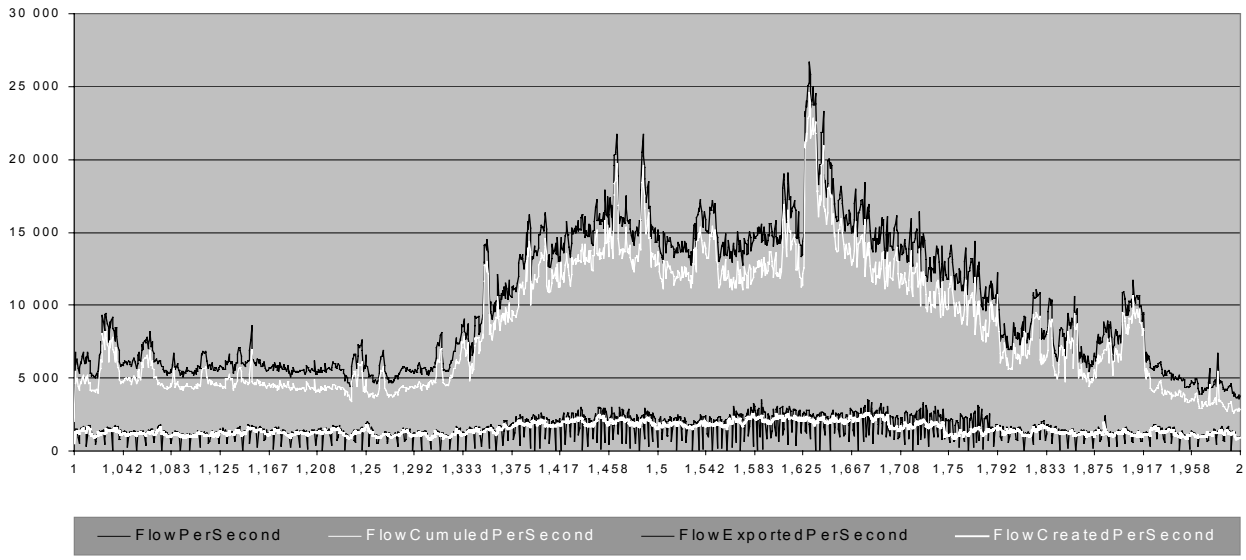


Figure 9 – *Evolution du traitement des flows sur une journée*

Les deux courbes du haut, respectivement en noir et blanc, représentent le nombre total de trames ethernet traitées et le nombre de trames pour lesquelles le flow existait déjà. Les deux courbes du bas, quant à elles, représentent, en blanc le nombre de flow créés, en noir le nombre de flows exportés. Les brusques variations sur la courbe des flows exportés proviennent d'un effet de résonance entre le déclenchement de l'export des flows toutes les minutes et le relevé des statistiques toutes les minutes également.

Sachant que l'addition des deux courbes blanches équivaut à la courbe noire du haut, on constate que le nombre de flow créés ne varie presque pas en fonction du nombre de paquets capturés. Une brusque augmentation du nombre de flows créés pourrait ainsi être corrélée à l'usage de certains outils de découverte de réseau, qui interrogent très rapidement une grande quantité de machines différentes, créant ainsi autant de flows distincts.

3.2 Utilisation avec NetSEC

NetSEC est une solution de métrologie pour la sécurité des réseaux, qui se base sur un système de base de données pour enregistrer les flows en fonction du temps[3]. Si, à l'origine, NetSEC s'est basé sur le collecteur de NetMET pour collecter les tickets Netflows, ce n'est plus le cas aujourd'hui, puisqu'il dispose de son propre collecteur, qui insère les flows directement dans la base de données. Cette structure de stockage s'avère intéressante pour mener une étude comparative entre les tickets Netflows générés par un routeur Cisco et ceux générés par la sonde, comme le montre, à la Figure 10, la maquette mise en place sur le réseau inter-universitaire de Valence.

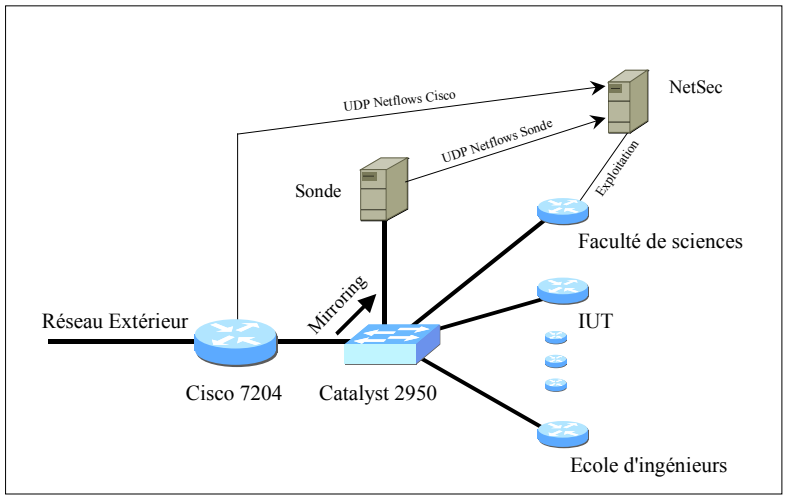


Figure 10 – *Maquette de validation par comparaison*

D'un point de vue qualitatif, nous vérifions aisément, à l'aide de requêtes SQL appropriées, que notre programme crée correctement les mêmes flows que le routeur Cisco. Cependant, il nous faut mettre de côté tous les flows que le routeur filtre en entrée, et pour lesquels il crée tout de même des entrées dans son cache netflow. On peut aisément les identifier, grâce aux index SNMP des interfaces renseignés dans chaque flow. Alors que ces index débutent à 1, un flow filtré sur une interface sera marqué avec 0 comme index d'interface de sortie, ce qui signifie "trame à ne pas router". Nous avons également pris soin de synchroniser la sonde sur le routeur, grâce au protocole ntp[5], ce qui nous permet de constater que les durées de flows sont également correctes par rapport à ce que renseigne le routeur Cisco.

Le deuxième test effectué est de nature plus quantitative. En effet, on demande aux deux bases de données de générer un cumul sur chaque flow, sur une période de temps donnée, et de trier le résultat par ordre de volume. On obtient ainsi l'équivalent d'un top10 de NetMET ou de NetSEC. Là encore, les résultats sont probants puisque l'on obtient les mêmes résultats avec les deux solutions.

[5] http://www.ntp.org

3.3 Utilisation avec NetMET

La plateforme de métrologie NetMET est de plus en plus utilisée par la communauté universitaire française. Nous avons validé la compatibilité de notre sonde avec cette plateforme, en configurant une maquette de test similaire à celle utilisée pour NetSEC. La première difficulté se situe au niveau de l'interrogation SNMP qu'effectue NetMET sur le routeur pour obtenir les index des interfaces. En effet, NetMET utilise ces informations, notamment pour agréger le trafic extérieur et réduire ainsi le volume des données. Ce problème peut être contourné en utilisant le daemon snmpd standard de la librairie net-snmp[6], qui permettra de remonter les numéros d'interfaces des cartes réseaux de la machine, puisque la MIB utilisée est identique à celle qu'utilise le routeur Cisco.

Nous n'avons pas ici la possibilité d'éliminer les flows que le routeur a filtrés. C'est pourquoi nous obtenons de légères différences dans les calculs de volumétrie. Par contre, les données issues de la sonde semblent effectuer un lissage des données dans le temps. On peut le constater en regardant les débits maxima calculés par les deux plateformes NetMET installées à Valence, sur le même modèle que NetSEC. Sur l'exemple de la Figure 11 et de la Figure 12 ci-après, on constate que les deux courbes ont globalement la même allure, mais que les échelles diffèrent.

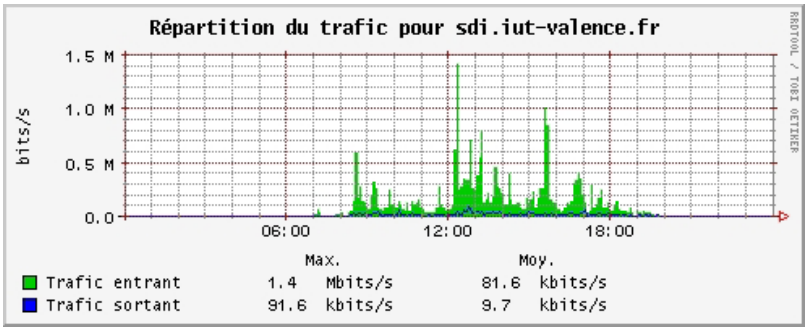


Figure 11 – *Trafic journalier d'une machine calculé à partir des Netflows Cisco*

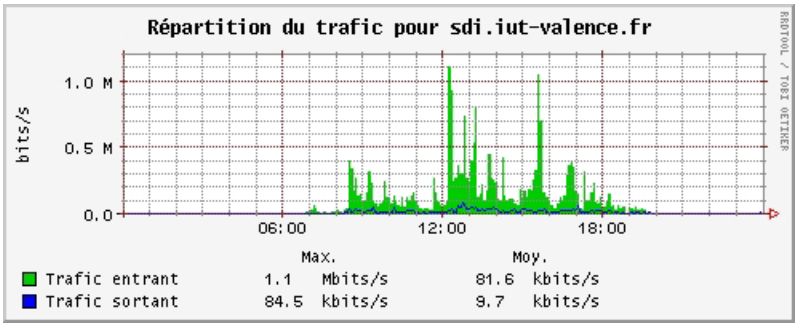


Figure 12 – *Trafic journalier de la même machine calculé à partir des Netflows de la sonde*

Alors que les débits moyens sont parfaitement identiques, ce sont les pics de trafics qui sont atténués, nous le constatons avec les valeurs maximales de trafic entrant et sortant affichées sous la courbe. Ce comportement est, à ce jour, inexpliqué et dépend peut-être de la façon dont NetMET collecte les Netflows. Grâce au dialogue instauré avec le CIRIL de Nancy, nous pourrons analyser plus finement les données de notre sonde.

[6] http://www.net-snmp.org

Conclusion

A ce jour, le programme est fonctionnel et la génération de Netflows V5 a été validée à grande échelle. L'un des premiers résultats a montré que la quantité de mémoire utile pour le programme n'était pas, comme nous le pensions, la ressource critique. C'est davantage la puissance de traitement, et une architecture multi-processeurs qui permet de tirer les plus grands bénéfices de l'organisation du programme. D'autres essais doivent encore être menés pour affiner les résultats comparatifs en fonction des règles de routage et pour vérifier la robustesse du code, avec différentes configurations de trafic.

Le processus de collecte des Netflows bénéficiera prochainement de plusieurs optimisations, en utilisant les possibilités de filtrage de la librairie libpcap, et en limitant la portion de trame capturée. C'est également l'interface utilisateur qui sera améliorée pour faciliter le paramétrage du programme, et générer des informations statistiques les plus pertinentes possibles. Enfin, c'est l'étude sur la durée des flows qui permettra d'optimiser l'utilisation de la mémoire.

Par ailleurs, notre programme continuera de s'enrichir, notamment pour traiter les paquets IPv6. Leur capture s'effectuera, au niveau Ethernet, de la même manière que les autres trames, il suffira donc d'implémenter le décodage de l'en-tête IPv6. Par contre, l'étape suivante consistant à générer des Netflows IPv6, imposera d'implémenter le format de Netflow V9, choisi par l'IETF pour définir le standard IPFIX. De fait, notre volonté de développer pour la communauté nous incitera donc à préférer la voie du standard, lorsqu'il sera clairement spécifié.

Une autre application est actuellement à l'étude, à partir du même noyau de collecte et de stockage des flows. En effet, la structure d'arbre binaire organise les flows en fonction des adresses IP et des ports. Cela permet donc de mettre en évidence certains radicaux communs sur un grand nombre de flows. Typiquement, un "scan" lancé par une machine extérieure sur un de nos réseaux donnera naissance, dans l'arbre binaire, à une série identifiable de flows voisins. Quels moyens peut-on mettre en oeuvre pour détecter ce comportement anormal ? Quel retard avons-nous par rapport au temps réel du trafic ? Quelles actions pouvons nous déclencher pour interrompre cet espion avant qu'il ne soit trop tard ?

Références

[1] Alexandre Simon, NetMET : Une solution de métrologie générale pour les réseaux régionaux, métropolitains et de campus. Dans *Actes du congrès JRES2001*, pages 305-318, Lyon, Décembre 2001.

[2] Cisco. Netflow FlowCollector Installation and User's Guide, Appendix B : Netflow Export Datagram Format

[3] Bernard Martinet et Jean-François Scariot, La métrologie, base pour la sécurité : NetSEC. Dans *Actes du congrès JRES2001*, pages 319-328, Lyon, Décembre 2001.

[4] Arbres AVL (de leurs auteurs Adelson, Velskij et Landis), documentation internet, par exemple http://brassens.upmf-grenoble.fr/IMSS/dciss/Enseignements/PSR/Prog/Java/CoursJava/arbresAVL.htm