Capturer et traiter du trafic réseau

G. Lehmann

Douce et paisible introduction :

Parmi les ambitieux objectifs professionnels qui m'étaient confiés pour ce début d'année, je devais monter une formation des collègues concernant l'utilisation des sondes. Les maîtres du temps et du planning n'ayant pas voulu étendre les journées au-delà de 24 heures, je me vois dans l'obligation de ne pas devoir l'animer. J'ai donc résumé quelques points sur lesquels j'ai travaillé ces derniers mois et qui pourraient être utile.

Je vais me concentrer sur l'outil Wireshark et les autres outils qui font parti du projet (eh oui, il y en a d'autres !). Cet article parle donc de technique, de méthodologie et parfois de sexe.

Rapidement un peu d'histoire :

Au début, Gérald Combs créa un analyseur réseau nommé Ethereal. Puis paf, un jour il s'appella Wireshark (je vous avez dit que ce serait un historique rapide ...). En conclusion, wireshark est la continuité d'Ethereal.

Présentation générale :

Ethereal permet de faire des captures de trafic en appliquant des filtres à la capture, mais également à l'affichage. On retrouve des filtres par adresse IP (source, destination, les 2), par adresses ethernet, par protocole (de n'importe quelle couche), ect.
Quand on parle de capturer le trafic, cela veut dire enregistrer tout ce que remonte la carte réseau brut de fonderie. Etudier le comportement d'un réseau en analysant 750000 trames représentant environ 60 secondes de trafic sur un lien chargé à 10Mb/s, c'est un peu chercher une aiguille dans une botte de foin. Wireshark nous aide un peu :
-- application de filtres d'affichage pour élager petit à petit le « bruit » ;
-- coloration ;
-- affichage sous 3 formes : résumé de trame (ordre de capture, timestamp de capture, adresses IP source et destinataire, protocole de niveau 4 ou 7, informations complémentaires), détail de chaque en-tête sous forme d'arborescence, et enfin la trame sous forme hexadécimale (avec une traduction approximative en code ASCII).

Avant de continuer, 2 messages subliminaux écrits pour ceux qui disent connaître les analyseurs réseaux, mais pas tant que ça en fait :
-- un analyseur réseau est comme un thermomètre ; ce n'est pas en lisant la température que vous déterminerez la maladie. En plus clair, arrêtez de croire que l'analyseur réseau nous écrit en gros caractères « ton problème est sur le switch 192.168.10.20 où le spanning-tree est activé en version 1 alors que selon la directive N55220-89 de la Direction Informatique & Télécom, tu dois le désactiver. Attends un petit moment, je suis en train de te générer un batch pour corriger tout ça ... ». Même un analyseur à 15000 euros ne fait pas ça.
-- un analyseur requiert évidemment des compétences pour l'utiliser, mais surtout pour comprendre ce qu'il nous affiche. C'est un peu plus complexe qu'un thermomètre quand même.

Où placer la sonde :

(Euh ... mon illustration avec le thermomètre atteint ici ses limites !)
Logiquement, là où est suceptible de passer tout le trafic que nous voulons observer et si possible là

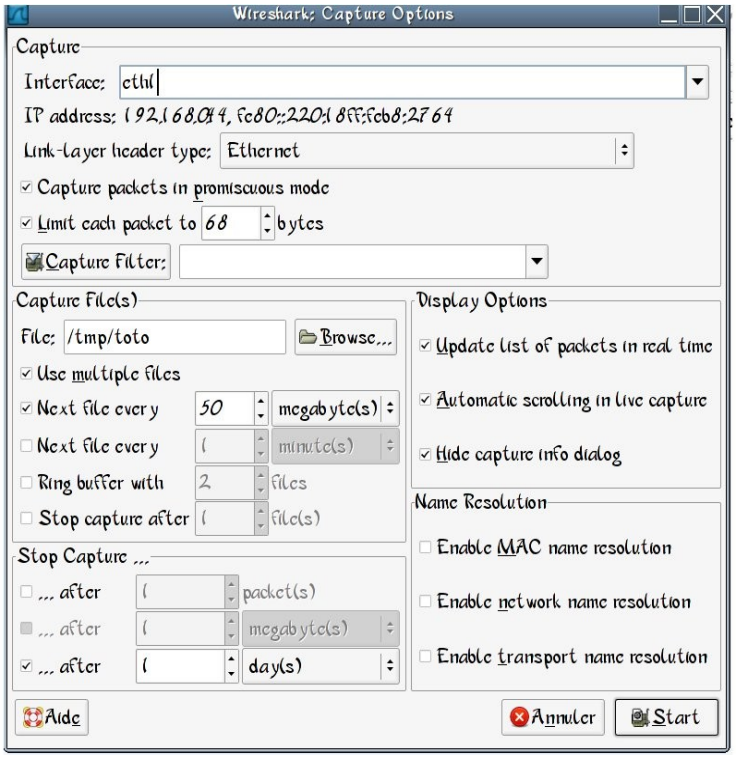
où le trafic dont nous nous fichons ne passe pas. Une fois que nous avons trouvé la bonne planque pour espionner le réseau il reste à placer la sonde. Soit nous l'insérons en coupure, soit si cela n'est pas possible, nous nous connectons au switch où nous voulons surveiller le trafic entrant/sortant d'un port donné, et nous activons la recopie de trafic de ce port vers le port où est connectée la sonde. Il est préférable de se mettre en coupure car plusieurs problèmes peuvent subvenir :
-- le port où est connecté la sonde ne permet pas de monter à un débit aussi élevé que celui généré sur le port surveillé ;
-- la recopie de trafic ne recopie généralement que le trafic valide. Si c'est le cas de votre switch, ne perdez pas votre temps à essayer de capter des trames ethernet erronnées, elles ne vous parviendront pas !
-- sur certains équipements, la recopie de port ne permet pas d'envoyer/recevoir du trafic à/depuis la sonde connectée sur ce port. Si votre sonde ne dispose que d'une carte ethernet qui fait à la fois office d'interface de capture et de point d'entrée pour la PMAD, vous aller vous enquiquiner à chaque fois que vous voudrez voir le résultat de la capture (arrêter la recopie de port et donc perdre quelques secondes ou minutes de capture, pour ensuite faire la PMAD). Impossible donc de voir à distance ce qui est capturé en temp-réel.

Effectuer des captures avec Wireshark :

Vous avez installé Wireshark sur votre PC MS Windows en suivant la procédure très complexe que voici : cliquer sur « Suivant » jusqu'à qu'il n'apparaisse plus, cliquer alors sur « Terminer ».
Sur les unix, c'est différent, mais pas compliqué non plus vu que ce logiciel est empaqueté dans les distributions.

Puis fébrile, ne pouvant vous contenir, vous lancez votre première capture sur le backbone du réseau au plus fort de la journée. Les trames défillent à tout allure, les chiffres s'incrémentent, vous voyez le monde défiler sous vos yeux, votre vue se brouille et vous finissez par vous évanouir submergé par l'émotion de vous sentir l'âme d'un expert réseau.

Quelques minutes voir quelques heures après, vous reprenez vos esprits et vous arrêtez la capture. Si votre PC n'a pas explosé sous la charge RAM induite par Wireshark, vous vous retrouvez avec quelques centaines de milliers de trames et un fichiers de plusieurs centaines de Mo, voir Go. En effet, vous avez lancé une capture n'importe comment et vous vous retrouvez avec un fichier inexploitable car trop volumineux. N'ayons pas honte de le reconnaître, nous sommes tous passés par là, même les meilleurs d'entres-nous.
Eh oui, car quand on lance une capture, il faut programmer certains paramètres :



-- L'interface de la machine sur laquelle réaliser la capture (si plusieurs cartes réseaux ... attention aux interfaces virtuelles comme les VPN IPSec).
-- Analysons-nous tout le trafic qui passe (mode promiscuité activé) ou seulement le trafic à destination ou source de la machine ?
-- Filtre à la capture : intéressant pour alléger plus tard le traitement du fichier, mais seulement dans le cas où déjà nous savons ce que nous cherchons ; donc pas en première approche d'un dépannage.
-- Est-ce que nous nous intéressons aux données transportées ou juste aux en-têtes des trames ? Dans le second cas, inutile de capter les maximum 1500 octets d'une trame ethernet si seulement les adresses IP sources et destination nous intéressent (14 octets pour l'en-tête ethernet, 20 pour IP). Le minimum est de 68 octets. Nos fichiers de captures grossiront moins vite.
-- De quand à quand faisons-nous la capture ? Wireshark ne permet pas de programmer de capture, mais d'autres outils sont utilisables dans des scripts permettant d'effectuer un démarrage différé. Nous verrons ces outils plus tard.
-- Est-ce que nous capturons le trafic dans un fichier ? Sauf capture de quelques minutes et en direct, je le conseille fortement. En effet, si Wireshark plante, nous ne perdrons pas les trames déjà capturées, et cela permet aussi à Wireshark de ne pas garder tout le trafic en RAM (et donc d'exploser votre RAM).
-- Devons-nous enregistrer le trafic dans un fichier circulaire ? C'est utilisé quand nous mettons un analyseur pour étudier un phénomène observable au moment où il se passe, récurrent mais non prévisible. Ainsi, nous n'avons que les X dernières minutes ou dernières heures au moment de l'arrêt de la capture, et non les 10 jours sans intérêt précédant l'évènement.
-- Voulons-nous plusieurs fichiers ou un seul ? Je conseille également, à utiliser le stockage dans un fichier, d'utiliser plusieurs fichiers. A priori notre capture va être volumineuse, donc autant découper pour retraiter ensuite par petits bouts. Viens alors la question suivante :
-- Qu'est-ce qui détermine le passage d'un fichier à l'autre ? La taille (maximum 50Mo par exemple pour une gestion simplifiée) ou le temps (un test dure une heure, puis l'autre démarre en suivant et nous voulons que la trace de chaque test soit dans son propre fichier) ?
-- Est-ce que la capture doit s'arrêter automatiquement ? Si oui, cela peut être après x paquets, ou lorsque nous avons capturé une certaine quantité de trafic ou encore au bout d'un temps donné.

Viennent ensuite quelques autres options secondaires que vous découvrirez par vous-mêmes.

Ayant pris le temps de poser les choses cette fois-ci, on se dit que ce que l'on veut vraiment, c'est connaître le débit global, les couples IP les plus bavards et que la période d'observation soit une journée entière.
Nous lançons donc la capture :
-- interface en mode promiscuité sur l'interface eth1 ;
-- seuls les 68 premiers octets sont gardés ;
-- stockage dans des fichiers, sans le mode « ring-buffer » ;
-- nous voulons des fichiers de 50Mo maximum, sans limite de nombre (attention à la taille disque !). Nous choisissons 50Mo seulement car ce qui charge Wireshark ce n'est pas la taille du fichier, mais le nombre de trames, même incomplètes, qu'il contient) ;
-- nous stoppons la capture au bout de 1 jour.

Le lendemain nous nous retrouvons avec des fichiers suffisamment petits pour les traiter au fur et à mesure. Dans le menu « statistiques » nous avons un ensemble d'outil permettant d'afficher, entre autres, les couples IP les plus consommateurs, une courbe du trafic global ou correspondant à un filtre particulier.
Le problème c'est que nous voulions le couple IP le plus bavard sur la totalité de la journée et non par petit bout. Nous sélectionnons tous les fichiers et on fait un glisser-déposer dans l'interface graphique pour concaténer toutes les captures en une seule à partir de laquelle seront calculées les statisques. Et là, l'affichage se fige, le disque dur se met à gratter et au bout de quelques minutes, Wireshark nous insulte pour tentative d'obésité-sation et nous quitte sans même un dernier regard. Eh oui, si un fichier de 1Go est trop gros, ce n'est pas 200 fichiers de 50Mo qui vont mieux passer. Nous atteignons là une première limite de Wireshark (Wireshark plante sur MS Windows, mais pas sous GNU/Linux. Sur ce dernier il consomme toute la RAM puis la swap jusqu'à figer la machine ... guère mieux). Nous verrons plus tard comment contourner cela sans devoir piller les barrettes RAM du magasin informatique du quartier (héhé !).

Autre limite que vous avez surement constaté lors de la capture, c'est que pour passer d'un fichier à l'autre, il ferme la fenêtre de capture en cours et en ouvre une autre pour le fichier suivant. Or si la machine est très sollicitée, il n'a pas toujours le temps de « bien fermer » la fenêtre. Cela n'est pas génant en soit si ce n'est que l'on a une fenêtre ouverte en plus pour rien. Or, au bout de 24h, si 50% des fenêtres ne se sont pas closes, on se retrouve avec 100 fenêtres ouvertes, chacune consommant sa part de RAM. Si la capture est arrivée à son terme, nous pouvons les fermer à la main (en forçant parfois la fermeture), mais il arrive que cela fasse planter wireshark avant la fin et on se rend compte le lendemain que l'on n'a que les 5 premières heures de captures sur les 24 désirées. Nous verrons plus tard comment coutourner cet autre problème (re héhé). En fait non, nous allons le voir desuite avec le paragraphe suivant.

La galaxie des logiciels du projet Wireshark :

Wireshark, tout le monde connaît. C'est joli, c'est rapide, mais c'est limité.
En s'aidant de la libpcap (sous les Unix) ou winpcap (sous MS Windows), nous pouvons créer un outil de capture basique en moins de 50 lignes de code Perl, environ une centaine en C. Quelques centaines de lignes plus tard nous rajoutons le stockage dans plusieurs fichiers. Wireshark, qui fait des miliers de lignes de code intègre donc tout un tas de code, exécuté à son chargement, mais non utile pour la capture. Ce code sera utile pour les statistiques, dessiner des courbes, gérer les boutons et menus déroulants de l'interface graphique, ect.
C'est là que 2 outils arrivent et vont nous intéresser. Le premier est Tshark. C'est Wireshark en mode console. Il est disponible sous les Unix comme un outil à part de Wireshark. Je ne sais pas s'il est disponible sous MS Windows. Nous pouvons appliquer des filtres à l'affichage, stocker le

résultat dans un ou plusieurs fichiers et tout et tout comme Wireshark. Il affiche également en console le résultat courant de la capture. C'est moins joli et coloré, mais si nous avons uniquement accès à la sonde en telnet ou ssh, c'est tout de même bien utile. On peut exporter le résultat dans un fichier texte ou postscript ou ...

Le second outil est dumpcap. Il est à part (dumpcap.exe) sous MS Windows, mais inclus dans le paquet Wireshark sous Unix (concernant GNU/Linux Debian du moins). Il est aussi en mode console, permet de faire des filtres à la capture, de capturer seulement x octets de chaque trame (minimum 68 si la trame fait plus de 68 octets), de mettre tout cela dans plusieurs fichiers avec une rotation ou pas, ect. La seule différence avec Tshark, c'est que le résultat est écrit dans le format pcap, donc non lisible par l'homme (sauf Néo et toute sa clique qui ont fait héxadécimal en seconde langue en sortant de la matrice) et que pendant la capture nous ne voyons pas grand chose d'autre que le compteur de trames s'incrémentant laconiquement.

Il existe d'autres outils que je ne détaillerai pas mais qui sont utiles pour manipuler des fichiers de capture sans passer par l'interface et donc sans mettre à genoux le quadriprocesseurs doubles-coeurs que vous idôlatrez. De plus, Wireshark n'étant pas multi-threadé, inutile d'avoir des multi-processeurs ou des processeurs double-coeur ... Donc ces outils dont je ne vous parlerai pas sont :
-- editcap : Editer les fichiers de capture, inclus dans Wireshark. Mode console. Sa principale fonction est de supprimer les paquets du fichier de capture (équivalent à appliquer un filtre de capture à postériori) ou pour convertir le fichier dans un autre format.
-- mergecap : Concaténer plusieurs fichiers en un.
-- text2pcap : Convertir un fichier de capture au format ASCII hexadécimal dans le format pcap.
-- capinfos : Donner des informations générales sur le fichier de capture (nombre de trames, taille du fichier, la durée de la capture, ...).

Je vous laisse aller faire un tour à l'adresse
http://www.wireshark.org/docs/wsug_html_chunked/AppTools.html
pour avoir plus d'infos sur ces différents outils, ainsi que les arguments qu'ils apprécient de voir renseignés.

Capture avec dumpcap :

Après cette grande appartée, revenons à notre capture d'une journée. Déjà, pour s'assurer qu'elle ne plante pas vicieusement en pleine nuit suite à un nombre trop important de fenêtres ouvertes, nous allons utiliser dumpcap.
Sous GNU/Linux :
dumpcap -i eth0 -w /tmp/tutu -a duration:86400 -b filesize:50000 -s 68
En bon français, nous enregistrons le trafic passant sur l'interface eth0 dans des fichiers qui commencent par tutu, dans le répertoire /tmp. Ils seront nommés par l'outil comme, par exemple : tutu_00004_20090114230857
où 00004 signifie que c'est le 4ième fichier généré lors de la capture,
et 20090211230857 la date de génération de ce fichier (11 février 2009 à 23h08 et 57 secondes, argh, je devrais être couché à cette heure moi !).
La durée de capture s'exprime en secondes, donc 24 heures = 86400 secondes.
La taille de chaque fichier de capture en Ko, donc 50000Ko dans notre cas.
Enfin nous capturons seulement les 68 premiers octets. Pour ceux qui sont inquiets de savoir comment calculer le débit si l'on n'archive que les 68 premiers octets qu'ils se rassurent : il est stocké dans le fichier la taille initiale du paquet avant que celui-ci se fasse retailler et biensûr wireshark prend en compte dans son calcul la taille originale. Ouf, vous avez eu peur, hein ?

Ci-dessous le résultat de *dumpcap -help* :

```
Dumpcap 1.0.2
Capture network packets and dump them into a libpcap file.
See http://www.wireshark.org for more information.

Usage: dumpcap [options] ...

Capture interface:
  -i <interface>          name or idx of interface (def: first non-loopback)
  -f <capture filter>     packet filter in libpcap filter syntax
  -s <snaplen>            packet snapshot length (def: 65535)
  -p                      don't capture in promiscuous mode
  -y <link type>          link layer type (def: first appropriate)
  -D                      print list of interfaces and exit
  -L                      print list of link-layer types of iface and exit
  -S                      print statistics for each interface once every second
  -M                      for -D, -L, and -S produce machine-readable output

Stop conditions:
  -c <packet count>       stop after n packets (def: infinite)
  -a <autostop cond.> ...  duration:NUM - stop after NUM seconds
                           filesize:NUM - stop this file after NUM KB
                             files:NUM - stop after NUM files
Output (files):
  -w <filename>           name of file to save (def: tempfile)
  -b <ringbuffer opt.> ... duration:NUM - switch to next file after NUM secs
                           filesize:NUM - switch to next file after NUM KB
                             files:NUM - ringbuffer: replace after NUM files
Miscellaneous:
  -v                      print version information and exit
  -h                      display this help and exit

Example: dumpcap -i eth0 -a duration:60 -w output.pcap
"Capture network packets from interface eth0 until 60s passed into output.pcap"

Use Ctrl-C to stop capturing at any time.
```

Analyse avec un super nouvel outil :

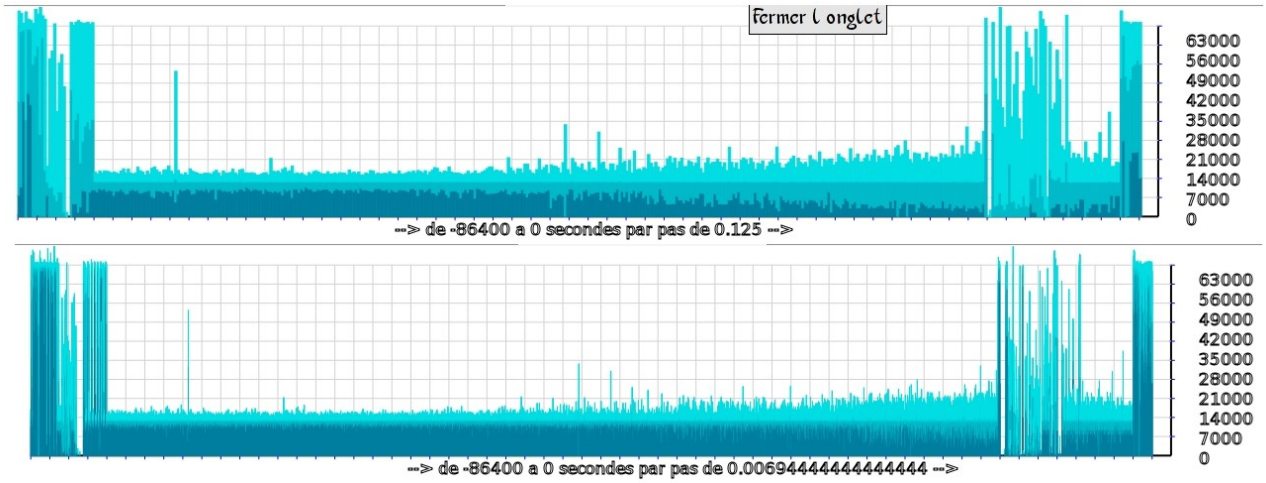
C'est bien beau d'avoir réussi à stocker son fichier sans que sa barrette de RAM parte en sublimation, mais on n'est toujours pas capable d'exploiter l'ensemble de la capture d'un coup. Soit ca vous va de travailler par petits bouts, soit vous travaillez sur chaque fichier pour l'alléger et ne garder que les trames qui vous intéressent (pas de chance dans notre cas, voulant voir tout le trafic, nous devons tout garder) et finalement réduire chaque fichier pour que la somme de tous soit raisonnable pour Wireshark. Vous pouvez également effectuer les traitements et les statistiques à l'aide de Tshark.
Enfin, dernière solution, coder votre propre outil ... je sens un mouvement de foule d'un seul coup, j'ai dit quelque chose qui dérange ?

J'ai codé un outil que je vais vous présenter rapidement. Après consultation des plus grands stratèges logiciels du monde, j'ai décidé de l'appeller « Sonde.pl ».
Nous lui donnons un fichier de capture en entrée (capture faite par Wireshark ou par dumpcap). Il met tout cela dans une base de données, fait des stats de base comme :

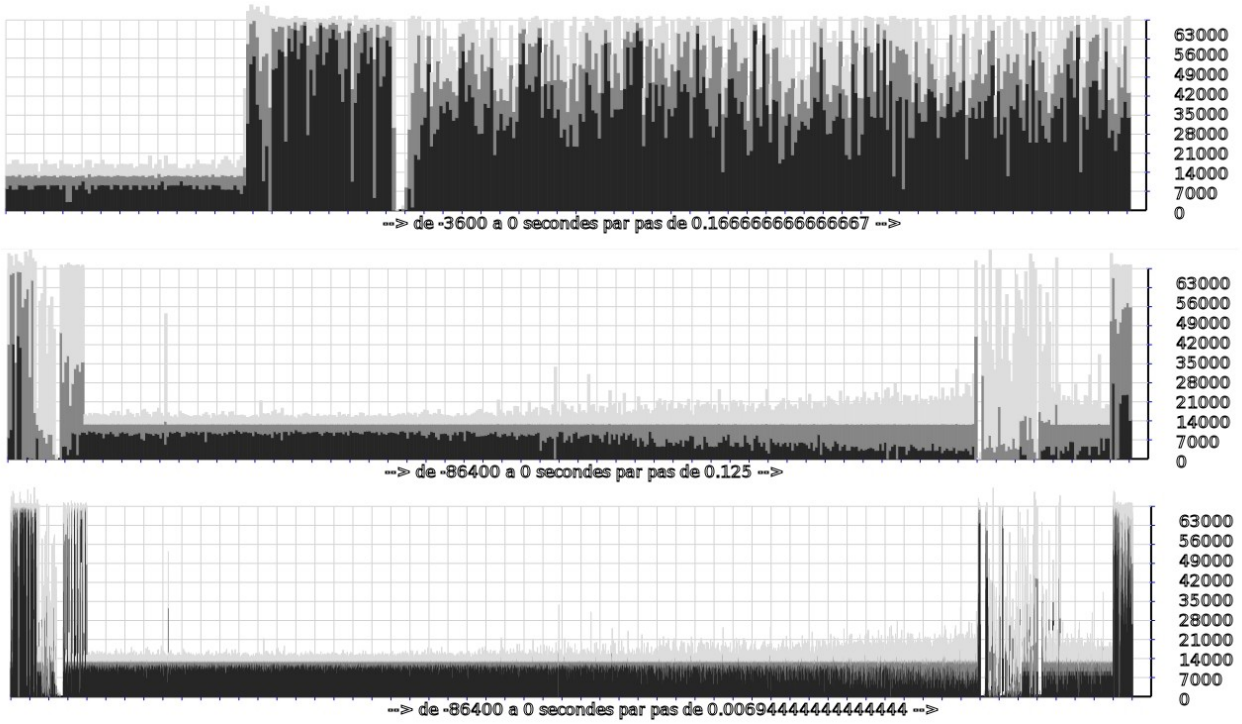
-- classement des couples IP les plus bavards sur la période d'observation ;
-- débit (par seconde) utilisé par chaque couple IP (2 valeurs, une pour chaque sens) ;
-- broadcast ethernet ;
-- ...
Il permet également de faire des histogrammes (format svg, lisible par exemple par Firefox ou Epiphany browser) :
-- trafic TCP, trafic UDP et trafic ICMP ;
-- trafic non IP, trafic IP non TCP/UDP/ICMP ;
-- trafic minimum/moyen/maximum sur une période choisie avec une granularité définie (minimum 10 secondes).

Ci-dessous des exemble de trafic Min (bleu), Moyen (jaune), Max (gris) sur 1 heure et sur 1 jour, avec des granularités de 10 secondes.

Sur 1 jour, même chose avec des granularités différentes et des couleurs différentes.



D'autres couleurs cette fois pour les gothiques ...



Limites :
L'outil ne connait que le trafic :
-- ethernet II ;
-- ethernet 802.3 (traitement des en-têtes Ethernet et LLC) ;
-- ethernet II et IP ;
-- ethernet II et IP et TCP ;
-- ethernet II et IP et UDP ;
-- ethernet II et IP et ICMP ;
-- ethernet II et ARP.

Pas d'analyse du spanning-tree ou d'IPX. Le premier sera traité dans l'Ethernet 802.3, le second sera classé dans le trafic Ethernet II sans plus de précision aux côtés par exemple de SCTP et DCCP. Il est assez aisé pour une personne parlant le Perl d'étendre la liste des protocoles reconnus par l'outil.

Secret de fabrication :
La stabilité de l'outil réside dans la simple idée qu'il vaut mieux décortiquer et stocker le résultat dans une base de données qui gère bien mieux le stockage et les manipulations de gros volumes de données plutôt que dans un fichier ou une variable tableau. Ainsi, pour calculer les statistiques et ensuite en extraire les résultats pour réaliser des graphiques ou des tableaux, nous consultons la base de données (MySQL) en SQL.
Le stockage et la consultation peuvent être réalisés de manière asynchrone.

Performances :
J'ai fait des tests avec gestion de plus de 2 millions de trames ethernets ce qui représentait un fichier de plus de 1Go. Capture de trames entières à l'aide de dumpcpap. Wireshark ne peut même pas ouvrir le fichier (plantage après plusieurs minutes d'agonie).
Le traitement du fichier, par cet outil codé en perl, a duré 53 minutes. Ensuite le calcul des statistiques a duré environ 20 minutes. La génération de tous les graphiques environ 6 minutes. La CPU était chargée, mais pas la RAM (machine équipée de 768Mo de RAM seulement). Jouer avec la commande « nice » permet de garder un poste toujours réactif malgré les lourds calculs en tâches de fond.
Sur des fichiers moins volumineux, il s'avère que le traitement du fichier est aussi rapide sous

Wireshark que sous Sonde.pl. La génération des graphiques est cependant plus rapide sous Sondes.pl.
Ces tests ont été fait sur une machine monoprocesseur simple coeur. On peut imaginer des gains substanciels sur une machine multi-processeurs et/ou multi-coeurs à partir du moment où nous avons plusieurs fichiers que nous traitons en parrallèle en chargeant une instance de Sonde.pl pour chacun.

Mais pourquoi ?! :
Parce que j'en avais besoin pour retraiter des captures très volumineuses et wireshark tenait difficilement la phase de capture, et pas du tout la phase de traitement.

Où est-il disponible ? :
Il faut me le demander avec la seule contrainte que la phrase commence par « s'il te plait » et finisse par « merci ».

Comment l'utiliser ? :
Voici le manuel qui s'affiche en tapant *./Sonde.pl -h* :

-= MANUEL D'UTILISATION =-

-help : Affichage de cet aide.

==========
=MODES ET OPTIONS POUR CHACUN=
*-**rec** : (Non codé) Utilisation en tant que sonde uniquement (RECord traffic).*
*-**interface** <Nom Interface> : Interface réseau sur laquelle la sonde se met en écoute (mode promiscuité).*
*-**file** <Nom Fichier> : Chemin et nom du fichier dans lequel enregistrer la capture.*

*-**surv** : Surveillance du trafic pour détecter une coupure d'un flux. Entre 2 apparitions de flux, le trafic est enregistré dans un fichier.*
*-**interface** <Nom Interface> : Interface réseau sur laquelle la sonde se met en écoute (mode promiscuité).*
*-**chaine** <RegExp> : Chaîne de caractère identifiant le flux à surveiller.*
*-**file** <Nom Fichier> : Chemin et nom des fichiers où est loggué le trafic lorsqu'il y a un problème. Le programme rajoute une extension pour distinguer les enregistrements s'il y a une série de problèmes.*
*-**freq** <Nombre> : Délai maximum, en seconde, d'apparition de la chaîne de caractère. Si depuis la dernière apparition il s'est passé plus de temps que défini ici, alos le mode alarme est enclenché : le trafic depuis la dernière apparition est enregistré dans un fichier jusqu'à la prochaine réapparition.*

*-**init** : Suppression de tous les enregistrements des tables CAPTURE, STAT et STATHEURE.*

*-**pcap** : Réaliser le stockage des informations des trames (du fichier indiqué par -file) dans la base de données.*
*-**ajout** : Seuls les paquets, stockés dans le fichier de capture, plus récents que le plus récent des paquets stockés dans la BD, sera insérer dans la base de données. Cela permet de rajouter seulement les paquets non traités d'un fichier, sans doublon ni perte de temps à faire ce qui a déjà été fait.*
*-**file** <Nom Fichier> : Chemin et nom du fichier à traiter par -pcap.*
*-**v** : Mode verbeux. Afficher dans la console les en-têtes des trames traitées.*
*-**vv** : Mode très verbeux. Afficher dans la console les en-têtes et le contenu des trames traitées.*

*-**stat** : Générer les statistiques sur les paquets capturés et stockés dans la base de données. Stocket le résultat dans la base de données. On peut indiquer la granularité du calcul (impactant seulement les stats moyennées) :*
*-**ajout** : Seules les statistiques non encore générées sur la période donnée sont calculées. Les autres sont gardées. Généralement utile après un "-**ajout** -**cap**".*
*-**stgranularite** <Nombre> : 10 par défaut. Durée en secondes. Les stats qui sont moyennées seront calculées sur des intervalles allant de (t) à (t + granularité).*

*-**graph** : Générer les graphiques standards (sur les dernières minutes).*
*-**bp** <Nombre> : Bande-passante du lien. Permet de dimensionner l'échelle de l'axe des ordonnées.*

*-**gpers** : Générer les graphiques personnalisés (infos concaténées, mais affichées sur plusieurs heures). Temps de génération plus long, utile seulement pour une vision globale. Il est nécessaire d'indiquer les options supplémentaires suivantes :*
*-**nom** <Nom Fichier> : Nom du fichier avec extension. Le programme y rajoute automatiquement "Reporting".*
*-**duree** <Nombre> : Durée à afficher. S'exprime en secondes.*
*-**bp** <Nombre> : Bande-passante du lien. Permet de dimensionner l'échelle de l'axe des ordonnées.*
*-**abscisse** <Nombre> : Taille de l'abscisse en pixels.*
*-**ordonnee** <Nombre> : Taille de l'ordonnée en pixels.*
*-**granularite** <Nombre> : Durée en secondes. Une valeur affichée représente le min/moy/max des valeurs prises entre (t) et (t + granularite).*

==========
=OPTIONS=
*-**abscisse** <Nombre> : Voir le mode -**gpers**.*
*-**ajout** : Voir les modes -**pcap** et -**stat**.*
*-**bp** <Nombre> : Voir les modes -**graph** et -**gpers**.*
*-**chaine** <RegExp> : Voir le mode -**surv**.*
*-**duree** <Nombre> : Voir le mode -**gpers**.*
*-**file** <Nom Fichier> : Voir les modes -**rec,** -**surv** et -**pcap**.*
*-**freq** <Nombre> : Voir le mode -**surv**.*
*-**granularite** <Nombre> : Voir le mode -**gpers**.*
*-**interface** <Nom Interface> : Voir les modes -**rec** et -**surv**.*
*-**nom** <Nom Fichier> : Voir le mode -**gpers**.*
*-**ordonnee** <Nombre> : Voir le mode -**gpers**.*
*-**stgranularite** <Nombre> : Voir le mode -**gpers**.*
*-**v** : Voir le mode -**pcap**.*
*-**vv** : Voir le mode -**pcap**.*

==========
=!! Attention !!=
*-**init** annule l'effet de -**ajout***
*-**cap** (sans -ajout) ne fait pas de nettoyage de la BD. Exécuter 2 fois la commande sur le même fichier va créer des doubons ! Utiliser -**init** la deuxième fois si l'on veut retraiter tout le fichier mais en nettoyant la BD avant.*
*-**stat** (sans -**ajout**) nettoie la base de données. Il n'est donc pas utile de rajouter -**init**.*

Quelques exemples :
Donc pour ajouter une capture dans un base de données que l'on initialise préalablement :
./Sonde.pl -init -pcap -f /tmp/MonFichier.cap
Puis on génère les statistiques :
./Sonde.pl -stat
Enfin on génère tous les graphiques :
./Sonde.pl -graph -gpers -nom NomGraphique.svg -duree 86400 -abscisse 20 -ordonnee 100 -granularite 60 -bp 700000

On peut biensûr cumuler les options dans n'importe quel ordre :
./Sonde.pl -init -pcap -f /tmp/MonFichier.cap -graph -gpers -nom NomGraphique.svg \ -duree 86400 -stat -abscisse 20 -ordonnee 100 -granularite 60 -bp 700000

Pour les machines multi-processeurs, on peut lancer plusieurs processus en même temps. Pour lancer le traitement de 2 fichiers de capture en même temps (fichier2.cap plus récent que fichier1.cap), taper dans une première console :
./Sonde.pl -pcap -f fichier1.cap
Puis dans la seconde :
./Sonde.pl -pcap -f fichier2.cap
Ainsi, la première commande insère dans la base de données le résultat du traitement de fichier1.cap et la seconde commande fait de même pour fichier2.cap même si le traitement de fichier1.cap est toujours en cours. Comme ce sont 2 processus différents et indépendants, ils occuperont chacun une CPU différente exploitant ainsi au mieux les capacités multi-coeurs des nouveaux processeurs. Cela n'est pas limité à 2 CPU !

Autre et dernier cas : vous avez lancé le traitement d'un fichier fichier10.cap que vous avez arrêté en cours de traitement. Soit vous réinitialisez tout et cela peut prendre du temps de tout recommencer, soit vous ne traitez que ce qui manquait à stocker avec l'option *-ajout* :
./Sonde.pl -ajout -cap -f fichier10.cap
Cette option est également utilisable avec *-stat*.

En synthèse :

Je vous ai fait une petite introduction sur la démarche à suivre pour réaliser une capture avec Wireshark. Ensuite j'ai évoqué l'existence d'autres outils liés au projet Wireshark. Enfin j'ai parlé des limites de Wireshark et comment les contourner, en utilisant dumpcap, Tshark et pourquoi pas un petit outil que j'ai codé.
Ah oui, quand je disais en introduction qu'il serait question de sexe dans cet article, bhen ... j'ai menti !