Capturer les paquets qui passent sur le réseau

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 17 décembre 2008. Dernière mise à jour le 27 mars 2009

https://www.bortzmeyer.org/capture-paquets.html

Lire les paquets qui passent sur le réseau est une activité indispensable pour l'étudiant qui apprend les réseaux informatiques, pour l'administrateur système qui débogue un problème, pour le chercheur qui rassemble des statistiques ou simplement pour le curieux qui veut s'instruire. Mais avant de pouvoir les lire et les analyser, il faut les capturer. Quels sont les outils disponibles ?

Tout le monde connait et utilise tcpdump, qui permet à la fois de capturer et de décoder les paquets. Pour un coup d'œil rapide, c'est parfait. Mais pour des études plus lourdes, il est souvent recommandé de séparer la **capture** des données (qui doit pouvoir tourner longtemps, sans trop charger la machine) de leur **analyse** (souvent très consommatrice en ressources et qui peut parfaitement être faite hors-ligne). Cet article se focalise sur les outils de capture, l'analyse étant faite ailleurs, avec un programme d'exploration comme Wireshark ou bien avec un programme qu'on a développé soi-même, par exemple en Python <https://www.bortzmeyer.org/libpcap-python.html>.

Un bon logiciel de capture doit pouvoir :

— Tourner longtemps, tout en écrivant dans des fichiers, de préférence en sachant passer d'un fichier à l'autre lorsque les fichiers de capture deviennent trop gros, ou bien à l'issue d'un certain délai.
— Pouvoir filtrer les événements intéressants, de façon à ne pas stocker n'importe quoi sur le disque. À cette fin, tous les logiciels de capture disposent d'un mini-langage permettant de dire quels sont les « événements intéressants ». La puissance expressive de ce langage est un des principaux critères qui distinguent ces logiciels.
— Ne pas trop charger la machine (c'est une autre raison pour laquelle le filtrage est important mais attention, il peut lui-même consommer beaucoup de ressources), au cas où elle fasse tourner des services de production, et afin d'éviter de perdre des paquets (ce que font les logiciels de capture lorsqu'ils n'arrivent plus à suivre).

Presque tous les logiciels de capture enregistrement au format « pcap », celui de la libpcap.

Voyons les logiciels possibles. L'ancêtre tcpdump, très connu de tous les administrateurs réseaux, reste parfaitement utilisable pour cette tâche. Son option `-w` permet d'enregistrer les événements, qui pourront ensuite être lus par tout logiciel capable de lire du pcap (y compris tcpdump lui-même, avec l'option `-r`).

```
% tcpdump -i eth2 -w mydata.pcap
tcpdump: listening on eth2, link-type EN10MB (Ethernet), capture size 96 bytes
[Control-C]
41 packets captured
41 packets received by filter
0 packets dropped by kernel
```

Et `mydata.pcap` peut être analysé ensuite à loisir. tcpdump dispose de nombreuses options qui peuvent être utiles pour la capture comme `-c` (s'arrêter après un nombre donné de paquets) ou `-C` (donne une taille maximale au fichier), même si elles sont moins perfectionnées que celles des logiciels récents.

tcpdump dispose d'un langage de filtrage, nommé BPF et qui permet d'exprimer des choses simples comme « uniquement les paquets venant de `192.0.2.3` » ou « uniquement les paquets à destination du port 80 » ou encore « uniquement les paquets à destination de `2001:db8:1::deca:fbad`, **et** venant du port 53 ». Voici cette dernière :

```
% tcpdump -i eth2 -w mydata.pcap \
            dst host 2001:db8:1::deca:fbad and \
            src port 53
```

L'analyseur Wireshark a des mécanismes de décodage des paquets bien plus riches que ceux de tcpdump. Il vient avec un programme en ligne de commande, `tshark` qui peut être utilisé pour la capture :

```
% tshark -i eth2  -w mydata.pcap
Capturing on eth2
340
[Control-C]
```

Notez l'affichage en temps réel du nombre de paquets capturés, bien pratique. Ses options de capture sont plus perfectionnées que celles de tcpdump, par exemple `-a` permet de stopper celle-ci après qu'un temps donné se soit écoulé. En mode « plusieurs fichiers », les noms des fichiers sont plus parlants qu'avec tcpdump, etc.

tshark dispose d'un langage de filtrage plus perfectionné, documenté dans "*Display Filter Reference*" <http://www.wireshark.org/docs/dfref/>. Par exemple, si on regarde les requêtes DNS, tshark peut filtrer avec des règles comme « uniquement les réponses <http://www.wireshark.org/docs/dfref/d/dns.html> ». Mais attention, il ne s'agit que de filtre d'affichage, le langage des filtres de capture est bien plus sommaire, c'est le même que celui de tcpdump (on peut trouver plein d'exemples sur le site de Wireshark <http://wiki.wireshark.org/CaptureFilters>). La raison de cette limitation est probablement que la capture doit aller vite, sans nécessiter de copier des paquets depuis le noyau et que le langage BPF, lui, peut tourner entièrement dans le noyau.

Pour des options de capture encore plus riches, on peut regarder pcapdump <http://packages.debian.org/pcaputils>. Il fait partie d'un groupe de programmes utilisant pcap, développés spécialement pour Debian, mais qui peuvent être compilés sur d'autres systèmes. Par exemple, sur Gentoo, il faut installer la bibliothèque Judy <http://judy.sourceforge.net/> (`emerge judy`) puis simplement utiliser `make` pour compiler. Ensuite, on copie les exécutables (dans `src/`) à la main.

Les forces de pcapdump, un logiciel spécialisé dans la capture, sont notamment ses capacités de tourner en démon (l'option n'est pas évidente à trouver, c'est `-P`, par exemple `pcapdump -P /var/run/pcapdump.opi -C $(pwd)/dns.pcapdump`), de changer de fichier sur plusieurs critères, de nommer les fichiers de capture selon un modèle qu'on choisit, d'échantilloner en n'enregistrant qu'un paquer sur N, etc. Il peut aussi lire ces options dans un fichier de configuration, dont voici un exemple qui lit les paquets DNS (port 53) et change de fichier tous les jours :

```
device=eth0
bpf="udp and port 53"
interval=86400
snaplen=512
promisc=0
filefmt=/var/tmp/pcapdump/eth0.%Y%m%d.%H%M.%S.pcap
mode=0600
owner=smith
group=root
```

Si on veut des filtres de capture très sophistiqués (tous les logiciels précédents se limitent au langage BPF), il faut les programmer soi-même ou bien utiliser un logiciel spécialisé comme dnscap <https://www.dns-oarc.net/tools/dnscap>. Ce programme dispose de filtres de capture spécifiques au DNS, par exemple on peut sélectionner uniquement les réponses et seulement celles qui sont négatives (NXDOMAIN, ce domaine n'existe pas).

dnscap n'a pas de documentation en ligne, tout est dans la page de manuel incluse dans la distribution. Sa compilation n'est pas toujours évidente. Sur Gentoo ou Debian, il faut ajouter dans le Makefile :

```
PORTLIBS= /usr/lib/libresolv.a
BINDLIB=-lbind9
```

Voici un exemple d'utilisation de dnscap, pour ne capturer que les paquets de réponses (-sr), et seulement si la réponse est négative (nom de domaine inexistant, -ex) et juste si la question concernait le domaine sources.org :

```
% dnscap -x 'sources\.org' -i eth0 -sr -g -ex
```

Comme pcapdump, il permet de changer automatiquement de fichier de capture en cours de route (par exemple avec -t 86400 pour tourner tous les jours), ce qui le rend utilisable pour des études sur le long terme.

Contrairement aux filtres BPF, le filtrage fait par dnscap est effectué dans l'espace utilisateur, pas dans le noyau. Il a donc fallu copier le paquet dans ledit espace et effectuer des opérations parfois complexes avant de décider de le garder ou pas. Les capacités de capture peuvent donc s'en ressentir.

Une fois les paquets capturés, il existe plusieurs programmes tout faits pour afficher des informations sur ces captures ou pour les modifier. Par exemple, Wireshark est livré avec deux utilitaires en ligne de commande très pratiques. editcap permet de sélectionner une partie d'une capture, en indiquant les numéros d'ordre des paquets qu'on veut garder :

```
% editcap -r large.pcap small.pcap 1-1000
Add_Selected: 1-1000
Inclusive ... 1, 1000
```

gardera les mille premiers paquets du fichier large.pcap. capinfos permet d'obtenir des informations sur un fichier de trace :

```
% capinfos small.pcap
...
File encapsulation: Ethernet
Number of packets: 1000
File size: 210439 bytes
Data size: 371542 bytes
Capture duration: 23.338821 seconds
Start time: Mon Mar  2 21:41:57 2009
End time: Mon Mar  2 21:42:21 2009
...
```

Et si on développe un programme soi-même ? C'est assez facile, dans plusieurs langages de programmation. On peut ainsi choisir les critères de capture à volonté mais il faut prendre garde aux performances : beaucoup de paquets peuvent arriver en très peu de temps.

En langage C, libpcap, maintenue par les responsables de tcpdump, est une solution très répandue, très bien documentée, et qui marche bien. Écrire un programme de capture est assez simple.

Voici un exemple du squelette d'un tel programme (montrant la capture mais pas l'écriture sur disque), (en ligne sur `https://www.bortzmeyer.org/files/sniff-only.c`). Il se compile, par exemple, avec :

```
% gcc -Wall -o sniff -lpcap sniff-only.c
```

Un bon article d'introduction à la libpcap, et à la définition de filtres de capture est "*Programming with Libpcap - Sniffing the network from our own application*" <`http://recursos.aldabaknocking.com/libpcapHakin9LuisMartinGarcia.pdf`>.

Pour la programmation en C, une alternative à pcap est ncap <`https://www.dns-oarc.net/tools/ncap`>, que je n'ai pas testé. Il semble que son format de stockage soit incompatible avec pcap.

Il existe d'autres programmes de capture que je n'ai pas testés :

— Grok <`http://gitorious.org/grok`>
— dumpcap <`http://www.wireshark.org/docs/man-pages/dumpcap.html`> qui, comme tshark, fait partie de Wireshark.