

Initiation à l'analyse réseau avec Wireshark

Table des matières

1	Introduction	2
2	Interface de Wireshark	3
2.1	Barres de menu et d'actions	4
2.2	Liste des trames	6
2.3	Détails des protocoles reconnus dans une trame	7
3	Filtres de capture et d'affichage	8
3.1	Les filtres d'affichage	8
3.1.1	Syntaxe d'un filtre d'affichage	8
3.1.1.a	Filtre par protocole	9
3.1.1.b	Filtre selon la valeur d'un champ	10
3.1.1.c	Test d'un champ booléen	11
3.1.1.d	Connecteurs	11
3.1.2	Extrait des identificateurs et champs disponibles pour les filtres d'affichage	12
3.1.2.a	Identificateur et champs pour Ethernet	12
3.1.2.b	Identificateur et champs pour IP	12
3.1.2.c	Identificateur et champs pour TCP	13
3.1.2.d	Identificateur et champs pour UDP	13
3.1.2.e	Identificateur et champs pour ICMP	14
3.2	Filtres de capture de la libpcap	15
3.2.1	Primitives pour Ethernet	15
3.2.2	Primitives pour IP	16
3.2.3	Primitives communes à UDP et TCP	16
3.2.4	Cas particulier des primitives spécifiques à ICMP	17

1 Introduction

Au cours des TP de réseaux, il vous aurez à capturer, analyser et commenter du trafic réseau. Dans ce domaine, l'outil le plus populaire est **wireshark**, un logiciel libre. Des versions pour de très nombreuses plateformes sont disponibles à partir de son site Web officiel (<http://www.wireshark.org>).

Wireshark a de nombreuses fonctionnalités parmi lesquelles :

- la capture de trames circulant sur le réseau. Cette opération nécessite **les droits d'administration** pour placer l'interface réseau en **mode promiscuous** afin de garder toutes les trames qui y parviennent. La capture est réalisée via la bibliothèque **libpcap** (sous Linux) ou son équivalent sous d'autres plateformes. Les trames capturées peuvent être sauvegardées dans un fichier au format PCAP (ou l'un des multiples formats proposés), pour une analyse ultérieure. **Des filtres de capture permettent de limiter les trames qui seront capturées.** Nous y reviendrons dans la section 3.2;

i On ne peut capturer une trame que si elle parvient à l'interface réseau. L'utilisation d'un switch plutôt qu'un hub limite les possibilités d'écoute du réseau. Cependant, certains switch permettent d'activer une fonction de monitoring sur un port, vers lequel sera dupliqué toutes les trames transitant par le switch.

- l'analyse des trames capturées ou chargées depuis un fichier. Pour plus de commodité, **des filtres d'affichage très pratiques** (section 3.1) permettent de travailler sur un sous-ensemble de trames capturées. L'analyse ne s'arrête pas au niveau trame (couche 2), mais continue aux niveaux supérieurs tant que **wireshark** reconnaît les protocoles (et il en reconnaît des centaines). Par exemple, si une trame Ethernet contient un datagramme IP, il sera aussi analysé. De même, si le datagramme IP contient un segment TCP, il sera aussi analysé. Enfin, si ce segment contient un message HTTP, il sera aussi analysé;
- l'affichage des informations des protocoles d'une trame donnée;
- le suivi de "dialogues" qui lie entre elles les trames d'un même dialogue, et qui permet d'établir de nombreuses statistiques sur le trafic et les discussions, y compris les débits et temps de réponse, des graphes, etc.

On comprend que c'est un des outils privilégiés des pirates. Notamment, il permet de capturer les identifiants et mots de passe transmis en clair (sans cryptage) sur le réseau où il est utilisé.

Mais c'est aussi un outil indispensable pour les administrateurs réseaux car il aide à la détection des causes d'un dysfonctionnement réseau (trames perdues/dupliquées, temps de réponse anormalement élevés, etc.) ou d'un éventuel trafic suspect.

2 Interface de Wireshark

Wireshark est souvent installé avec un ensemble d'outils en ligne de commande, mais il est principalement utilisé via son interface graphique. Sur notre machine virtuelle, c'est la commande **wireshark** qu'on peut lancer depuis le menu *Applications* → *Internet* → *Wireshark*.

Cette interface (voir figure 1) permet de réaliser toutes les opérations dont vous aurez besoin. Elle est composée principalement de 4 parties qui seront décrites dans les sections suivantes.

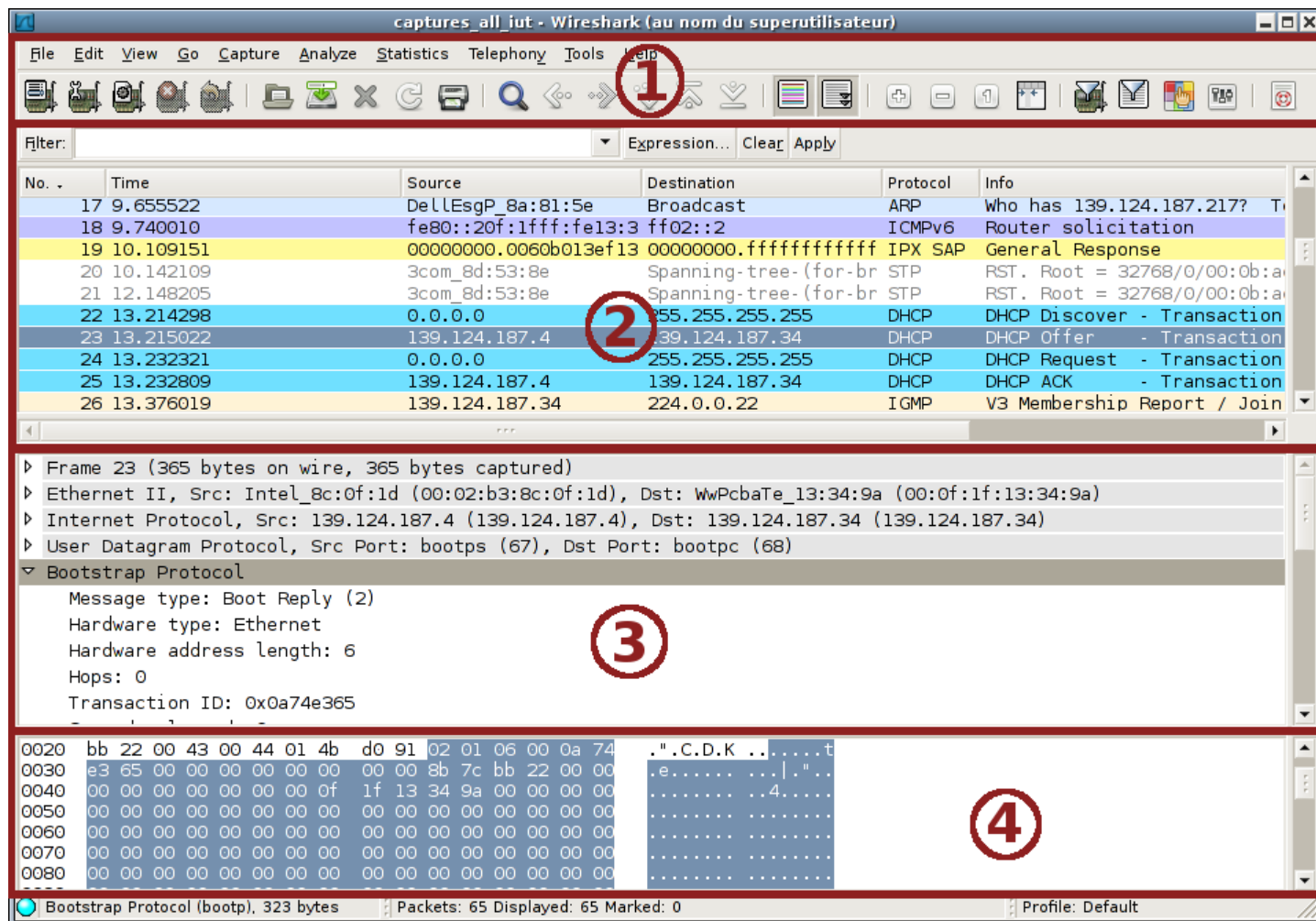


FIGURE 1 – Interface de Wireshark

2.1 Barres de menu et d'actions

Dans la zone marquée ① de la figure 1, se trouve en haut la barre de menu qui permet d'accéder à toutes les fonctions. En particulier, le menu *File* permet de sauver/charger tout ou partie d'une capture, le menu *Capture* permet de démarrer/arrêter/planifier une capture en précisant les interfaces à utiliser, et le menu *Analyze* permet de créer des filtres d'affichage.

Sous la barre de menu, on trouve une **barre d'actions rapides** (figure 2) regroupant les fonctionnalités les plus utilisées :

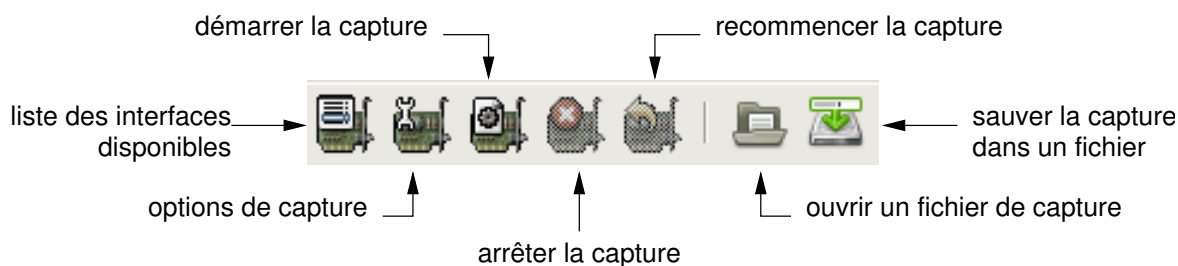



FIGURE 2 – Extrait de la barre d'actions

-  correspond au menu *Capture* → *Interfaces*. Ce bouton ouvre une boîte de dialogue (figure 3) montrant la liste des interfaces disponibles. On peut choisir sur quelle interface démarrer la capture en cliquant

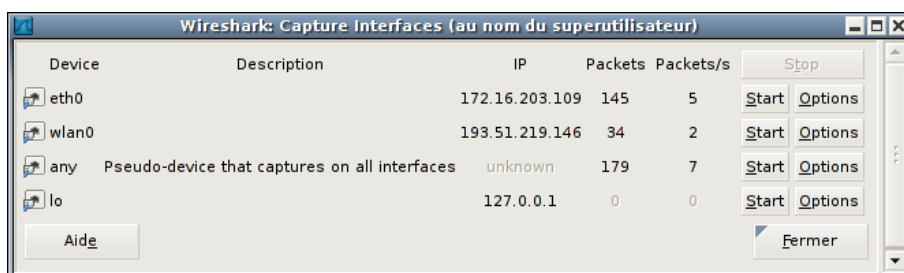


FIGURE 3 – Boîte de dialogue listant les interfaces

sur le bouton *Start*, ou définir des options de capture en cliquant sur *Options*. Notons que pour capturer des trames sur toutes les interfaces disponibles, il faut utiliser l'interface virtuelle **any**.

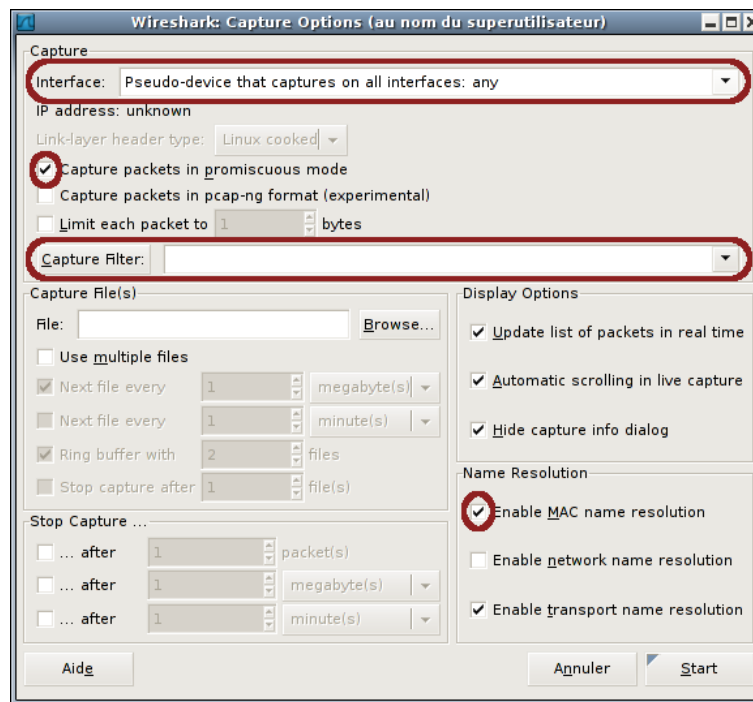








FIGURE 4 – Boîte de dialogue des options de capture

-  correspond au menu *Capture* → *Options* et ouvre une boîte de dialogue (figure 4) permettant de configurer la capture. Si, en principe, la majorité des options ont une valeur par défaut satisfaisante, certaines méritent une attention particulière :
 - ◇ dans la zone *Interface* une liste déroulante permet de choisir l'interface sur laquelle la capture doit se faire (ou **any** pour toutes les interfaces) ;
 - ◇ la case "*Capture packets in promiscuous mode*" **doit être cochée** pour capturer toutes les trames qui arrivent à l'interface, sinon seules celles dont la destination correspond à l'interface seraient capturées ;
 - ◇ la zone *Capture Filter* permet de limiter la capture aux trames satisfaisant un filtre. Une liste de filtres prédéfinis est proposée dans la liste déroulante et le bouton *Capture Filter* permet de définir un filtre à partir d'expressions. Nous y reviendrons dans la section 3.2 ;
 - ◇ la case *Enable MAC name resolution*, si cochée, active la résolution d'adresses MAC. Notamment, pour Ethernet, cela fait afficher Broadcast plutôt que l'adresse ff:ff:ff:ff:ff:ff. De plus, les 3 premiers octets des adresses Ethernet codant le constructeur, ils sont remplacés par le constructeur en question. Par exemple, l'adresse 00:18:8b:1e:b6:cc sera affichée Dell_1e:b6:cc. Cette fonctionnalité ne rend pas forcément l'affichage plus lisible...
-  correspond au menu *Capture* → *Start* et commence la capture avec les options définies. Les trames capturées sont analysées en temps réel. Celles respectant le filtre d'affichage sont affichées au fur et à mesure dans la zone ② de la figure 1, et la première trame affichée est détaillée dans les zones ③ et ④. Ces zones sont décrites dans les sections 2.2 et 2.3.
-  n'est cliquable que lorsqu'une capture est en cours. Il correspond au menu *Capture* → *Stop* et arrête la capture en cours.
-  n'est cliquable que lorsqu'une capture est en cours. Il correspond au menu *Capture* → *Restart* et a pour effet d'effacer les trames capturées jusque là et de continuer la capture.

-  correspond au menu *File* → *Open* et permet de charger un fichier contenant une capture (efface les trames courantes).
-  correspond au menu *File* → *Save* et permet de sauver les trames selon quelques critères. Cela ouvre la boîte de dialogue de la figure 5 dans laquelle on peut indiquer si l'on veut **sauver les trames capturées ou affichées**, sélectionnées, marquées ou un intervalle de trames. Le format du fichier par défaut est PCAP.

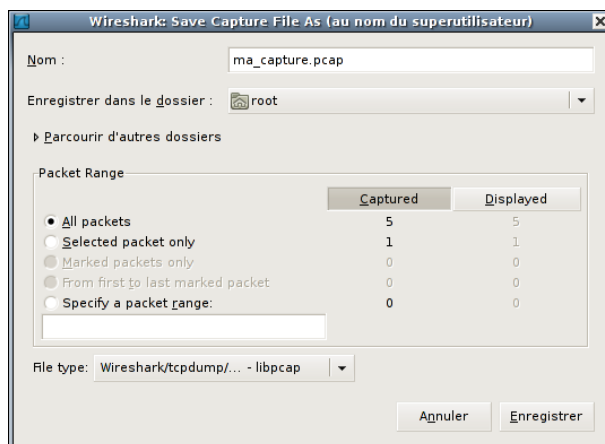


FIGURE 5 – Boîte de dialogue de sauvegarde de trames

2.2 Liste des trames

La liste des trames capturées et respectant le filtre d'affichage figure dans la zone ② de la figure 1, reprise dans la figure 6. En haut de cette zone se trouve le filtre d'affichage spécifié. En dessous, figure la liste des trames respectant les critères de ce filtre, à raison d'une trame par ligne. Les lignes sont colorées en fonction des protocoles reconnus dans les trames. Les colonnes affichées sont configurables via le menu *Edit* → *Preferences*. Par défaut, il y en a 6 : le numéro de la trame capturée, la date de capture depuis son début, sa source, sa destination, le protocole reconnu de plus haut niveau, et une présentation humainement compréhensible de l'information transportée pour ce protocole.

Filter: <input type="text"/> Expression... Clear Apply					
No.	Time	Source	Destination	Protocol	Info
17	9.655522	DellEsgP_8a:81:5e	Broadcast	ARP	Who has 139.124.187.217? T
18	9.740010	fe80::20f:1fff:fe13:3	ff02::2	ICMPv6	Router solicitation
19	10.109151	00000000.0060b013ef13	00000000.ffffffffffff	IPX SAP	General Response
20	10.142109	3com_8d:53:8e	Spanning-tree-(for-br	STP	RST. Root = 32768/0/00:0b:a
21	12.148205	3com_8d:53:8e	Spanning-tree-(for-br	STP	RST. Root = 32768/0/00:0b:a
22	13.214298	0.0.0.0	255.255.255.255	DHCP	DHCP Discover - Transaction
23	13.215022	139.124.187.4	139.124.187.34	DHCP	DHCP Offer - Transaction
24	13.232321	0.0.0.0	255.255.255.255	DHCP	DHCP Request - Transaction
25	13.232809	139.124.187.4	139.124.187.34	DHCP	DHCP ACK - Transaction
26	13.376019	139.124.187.34	224.0.0.22	IGMP	V3 Membership Report / Join

FIGURE 6 – Filtre d'affichage et liste des trames correspondantes

2.3 Détails des protocoles reconnus dans une trame

En sélectionnant une trame dans la liste précédente, comme la trame 23 sur la figure 6, les informations qui en ont été extraites sont présentées dans la zone ③ de la figure 1, reprise dans la figure 7. Dans cette zone, les informations apparaissent sous la forme d'une arborescence. En cliquant sur le symbole ▷ en début d'une ligne on la développe sur plusieurs lignes afin d'en avoir plus de détails. Au contraire, en cliquant sur ▽ on réduit l'affiche de l'information à une seule ligne. En effectuant un clic droit sur une information, on accède à un menu permettant notamment de la copier, ou de la sauver dans un fichier.

Sur la figure 7, on voit que dans la trame 23 sont reconnus les protocoles Ethernet-V2, IP, UDP et BOOTP (en réalité DHCP), dont les informations peuvent être détaillées.

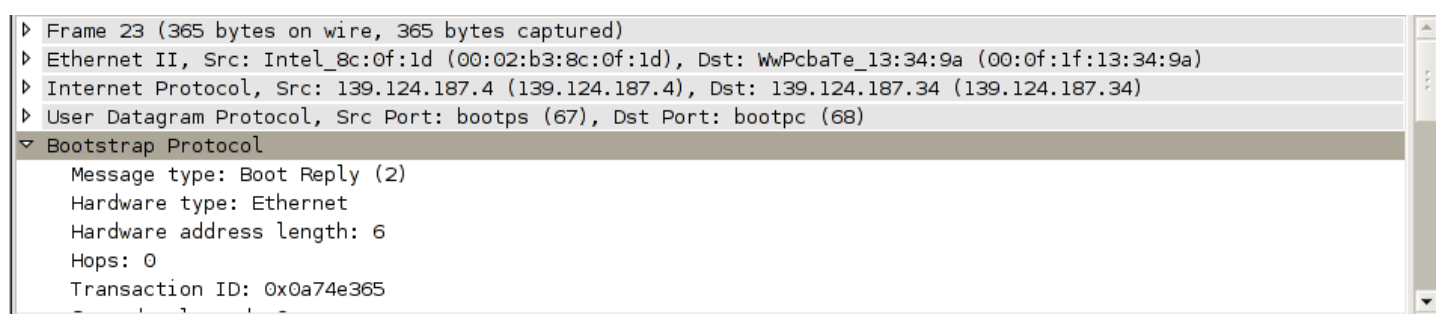


FIGURE 7 – Affichage des détails sur les protocoles reconnus dans une trame

Les octets de la trame sélectionnée sont affichés dans la zone ④ de la figure 1, reprise dans la figure 8.

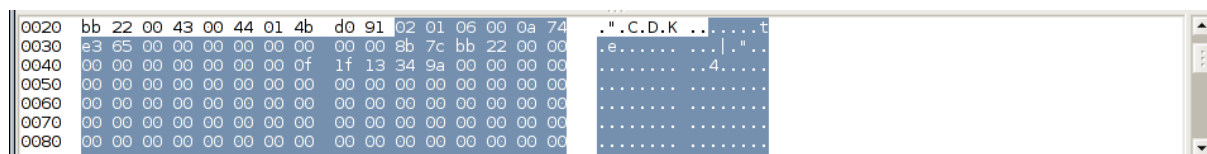


FIGURE 8 – Mise en évidence des octets d'une information

Cette zone est constituée de 3 parties :

- au centre, les octets de la trame sont affichés en hexadécimal à raison de 2 séries de 8 octets par ligne (soit 16 octets en tout, écrits avec 32 digits hexadécimaux) ;
- à droite, ces octets sont affichés sous la forme de caractères ASCII, en remplaçant les caractères non imprimables par un point (.)
- à gauche, un nombre en hexadécimal donne le numéro du premier octet de la ligne.

Lorsqu'on sélectionne une information dans la zone ③, les octets qui la constituent sont mis en évidence dans la zone ④. Sur les figures 7 et 8, l'information sélectionnée est le message DHCP et les octets qui le constituent sont mis en évidence. Inversement, lorsqu'on clique sur des octets (hexadécimaux ou ASCII) de la zone ④, l'information correspondante est développée dans la zone ③.

❗ Notons qu'en sélectionnant un protocole tel qu'Ethernet, IP, UDP, TCP, qui transporte un message d'un protocole de niveau supérieur, seul son en-tête est mis en évidence, puisque les données sont des informations de cet autre protocole.

3 Filtres de capture et d'affichage

L'utilisation conjointe de filtres de capture et de filtres d'affichage a son intérêt :

- le **filtre de capture** permet de limiter le volume de trames capturées **en ignorant les trames reçues qui ne respectent pas les critères** du filtre. On précise optionnellement un filtre de capture dans la boîte de dialogue de capture (voir figure 4). Bien qu'il soit en principe plutôt conseillé de ne pas filtrer les trames à capturer pour ne pas passer à côté de problèmes éventuels, l'absence de filtrage peut se traduire par une trop grande quantité de données à enregistrer, surtout si la capture dure longtemps ;
- le **filtre d'affichage permet de sélectionner**, parmi les trames capturées, **les trames à afficher** dans la zone ② (et *a fortiori* ③ et ④). Il permet de se concentrer sur un sous-ensemble des trames capturées, ce qui rend l'analyse de celles-ci plus confortable. Au cours de l'analyse d'une capture, on peut utiliser successivement plusieurs filtres d'affichage selon les informations que l'on cherche et que l'on découvre. Un filtre d'affichage se précise juste en dessous de la barre d'actions (voir *Filter* : en haut de la zone ② de la figure 1). Une fois entré, seules les trames satisfaisant les critères du filtre apparaissent dans la zone ②, les autres étant masquées.

Ce qui peut paraître surprenant et ennuyeux est que la syntaxe des filtres de capture est différente de celle des filtres d'affichage. Cela est dû au fait que **wireshark** utilise la bibliothèque **libpcap** pour la capture de trames qui ne reconnaît qu'un certain langage pour ses filtres. Ce langage n'est pas forcément très ergonomique mais doit être utilisé par de nombreux outils basés sur **libpcap**, tels que **tcpdump**.

Pour définir un filtre des trames à afficher (réalisée, en cours ou chargée), il faut utiliser le langage propre à **wireshark**, qui est à la fois plus ergonomique et plus puissant que celui de la **libpcap**. Nous commencerons par ces filtres.

3.1 Les filtres d'affichage

3.1.1 Syntaxe d'un filtre d'affichage

Les filtres d'affichage de **wireshark** sont plus "orientés objets" que les filtres de capture de **libpcap**. Ce sont des critères sur des protocoles ou sur les valeurs des champs de protocoles. Le critère ne peut être vrai que si la trame contient effectivement un PDU du protocole spécifié et dont les champs satisfont les conditions exprimées.

Un critère est exprimé en indiquant le nom du protocole suivi éventuellement d'un champ et de la condition qu'il doit vérifier. Le nom du protocole doit correspondre à l'identificateur que reconnaît ¹ **wireshark**, par exemple, **eth**, **ip**, **arp**, **icmp**, **udp**, **tcp**, **http**, **smtp**, **bootp**, **dns**...

1. En réalité, **wireshark** utilise des dissecteurs qui sont des plugin reconnaissant les protocoles et fournissant les moyens d'exprimer des conditions les concernant.

3.1.1.a Filtre par protocole


Si l'on ne veut afficher que les messages d'un certain protocole pour lequel un identificateur existe, il suffit d'indiquer cet identificateur.

Exemple 1

Voici quelques exemples de filtres utilisant seulement des identificateurs de protocoles :

- **arp**
n'affiche que les trames véhiculant un datagramme ARP
- **tcp**
n'affiche que les trames véhiculant un segment TCP (lui-même contenu dans un datagramme IP)
- **http**
n'affiche que les trames véhiculant du trafic HTTP



 Notons que les (PDU des) protocoles ne sont recherchés (et reconnus) que dans les conditions normales d'utilisation. Notamment, un datagramme ARP n'est recherché que si le champ *EtherType* de la trame vaut 0x0806. De même, les messages HTTP ne sont recherchés que dans les segments TCP dont le port source ou destination vaut 80.

3.1.1.b Filtre selon la valeur d'un champ

Pour exprimer des conditions sur les champs des PDU, on utilise la forme :

protocole.champ


où *champ* est l'identificateur du champ (ou de la propriété) à tester. La section 3.1.2 présente un extrait des champs disponibles. La liste complète des champs est accessible via l'URL <http://www.wireshark.org/docs/dfref/>.

Le champ peut être comparé à une valeur en utilisant l'un des opérateurs ==, !=, <, <=, >, >=. Le type d'opérateur permis ainsi que les valeurs à comparer dépendent du champ.


Exemple 2

Voici quelques exemples de filtres selon la valeur d'un champ :

- **eth.addr == 00:1f:4b:c8:05:e0**
n'affiche que les trames Ethernet dont l'adresse MAC source ou destination est 00:1f:4b:c8:05:e0


 On peut remarquer que **eth.addr** est une propriété dans la mesure où ce champ n'existe pas en réalité et représente une alternative entre deux champs réels.

- **eth.type == 0x0800**
n'affiche que les trames Ethernet transportant un datagramme IP
- **arp.opcode == 1**
n'affiche que les trames véhiculant une **requête** ARP
- **ip.hdr_len > 20**
n'affiche que les trames véhiculant des datagrammes IP comportant des options (en-tête de taille supérieure à 20 octets)

 On peut remarquer que **ip.hdr_len** contient la taille réelle de l'en-tête, c'est à dire la valeur du champ *HLEN* du datagramme, multipliée par 4

- **ip.src == 10.0.0.5**
n'affiche que les trames véhiculant un datagramme IP avec pour adresse source 10.0.0.5
- **ip.src != 10.0.0.0/8**
n'affiche que les trames véhiculant un datagramme IP ne provenant pas du réseau 10.0.0.0/8
- **http.request.method == "POST"**
n'affiche que les trames véhiculant des requêtes HTTP de type POST

□

 Comme on le voit dans le dernier exemple, on peut aussi tester des sous-champs pour certains protocoles.

3.1.1.c Test d'un champ booléen


Lorsque le champ a une valeur booléenne, il n'y a pas de comparaison à faire : il faut seulement le nommer (comme on le ferait pour une variable booléenne). Seules seront affichées les trames pour lesquelles ce champ existe et est vrai.

Exemple 3

Voici quelques exemples de filtres avec un champ booléen :

- **`ip.flags.df == 1`**
n'affiche que les trames véhiculant un datagramme IP dont le bit *Don't Fragment* est positionné
- **`tcp.flags.syn == 1`**
n'affiche que les trames véhiculant un établissement de connexion TCP
- **`tcp.checksum_bad == 1`**
n'affiche que les trames véhiculant un segment TCP dont le checksum est incorrect



 Comme on le voit dans le dernier exemple, certains "champs" booléens sont fictifs et expriment une qualité sur la valeur d'un champ effectif.


3.1.1.d Connecteurs

Enfin, on peut construire des expressions composées en utilisant les connecteurs ! (négation), && (et), || (ou), ainsi que les parenthèses.

Exemple 4

- **`eth.src == 00:1f:4b:c8:05:e0 && arp.opcode == 1`**
n'affiche que les trames dont l'adresse MAC source est 00:1f:4b:c8:05:e0 et contenant une requête ARP
- **`ip.src != 10.0.0.0/8 && ! http`**
n'affiche que les trames véhiculant un datagramme IP ne provenant pas du réseau 10.0.0.0/8 et qui ne concernent pas du trafic HTTP
- **`ip && ! (ip.addr == 10.0.0.5)`**
n'affiche que les trames contenant des datagrammes IP dont ni la source ni la destination est 10.0.0.5.



 Dans le dernier exemple, il aurait été incorrect de remplacer le test **`! (ip.addr == 10.0.0.5)`** par seulement **`ip.addr != 10.0.0.5`**, car cette dernière expression est vraie si l'une ou l'autre des adresses source ou destination est différente de 10.0.0.5.
Aussi, l'expression **`! (ip.addr == 10.0.0.5)`** seule est insuffisante car elle serait vraie aussi pour les trames ne contenant pas de datagramme IP (le champ **`ip.addr`** n'existe pas, donc l'égalité est fausse, et sa négation est vraie) qui seraient donc aussi affichées.

3.1.2 Extrait des identificateurs et champs disponibles pour les filtres d'affichage

3.1.2.a Identificateur et champs pour Ethernet

L'identificateur de protocole **eth** qualifie les trames Ethernet-V2 et IEEE 802.3. La liste complète de ses champs est disponible à l'URL <http://www.wireshark.org/docs/dfref/e/eth.html>.

Extrait des champs susceptibles de nous intéresser :

- **eth.addr** (6-byte hardware address)
adresse MAC source ou destination de la trame
- **eth.dst** (6-byte hardware address)
adresse MAC destination de la trame
- **eth.src** (6-byte hardware address)
adresse MAC source de la trame
- **eth.type** (unsigned 16-bit integer)
champ *EtherType* (uniquement pour Ethernet-V2)
- **eth.len** (unsigned 16-bit integer)
longueur de la trame (uniquement pour IEEE 802.3)

❗ Leur signification est indiquée dans le document décrivant le format des trames Ethernet dans les *Documents Utiles* du cours Ametice.

3.1.2.b Identificateur et champs pour IP

L'identificateur de protocole **ip** qualifie les trames contenant un datagramme IPv4. La liste complète de ses champs est disponible à l'URL <http://www.wireshark.org/docs/dfref/i/ip.html>.

Extrait des champs susceptibles de nous intéresser :

- **ip.addr** (IPv4 address)
adresse IPv4 source ou destination du datagramme
- **ip.dst** (IPv4 address)
adresse IPv4 destination du datagramme
- **ip.src** (IPv4 address)
adresse IPv4 source du datagramme
- **ip.proto** (unsigned 8-bit integer)
champ *Protocole* du datagramme
- **ip.hdr_len** (unsigned 8-bit integer)
taille réelle de l'en-tête du datagramme (valeur du champ HLEN multipliée par 4)
- **ip.len** (unsigned 16-bit integer)
taille totale du datagramme

❗ Leur signification est indiquée dans le document décrivant le format des datagrammes IPv4 dans les *Documents Utiles* du cours Ametice.

3.1.2.c Identificateur et champs pour TCP

L'identificateur de protocole **tcp** qualifie les trames contenant un segment TCP. La liste complète de ses champs est disponible à l'URL <http://www.wireshark.org/docs/dfref/t/tcp.html>.

Extrait des champs susceptibles de nous intéresser :

- **tcp.port** (*unsigned 16-bit integer*)
Port source ou destination du segment
- **tcp.dstport** (*unsigned 16-bit integer*)
Port destination du segment
- **tcp.srcport** (*unsigned 16-bit integer*)
Port source du segment
- **tcp.hdr_len** (*unsigned 8-bit integer*)
taille réelle de l'en-tête du segment (valeur du champ LET multipliée par 4)
- **tcp.len** (*unsigned 16-bit integer*)
taille totale du segment
- **tcp.flags.syn** (*boolean*)
valeur du bit SYN (établissement de connexion)
- **tcp.flags.ack** (*boolean*)
valeur du bit ACK (accusé de réception)
- **tcp.flags.fin** (*boolean*)
valeur du bit FIN (fin de connexion)
- **tcp.flags.reset** (*boolean*)
valeur du bit RST (fin brutale de connexion)

 Leur signification est indiquée dans les transparents du cours consacré à TCP.

3.1.2.d Identificateur et champs pour UDP

L'identificateur de protocole **udp** qualifie les trames contenant un datagramme UDP. La liste complète de ses champs est disponible à l'URL <http://www.wireshark.org/docs/dfref/u/udp.html>.

Extrait des champs susceptibles de nous intéresser :

- **udp.port** (*unsigned 16-bit integer*)
Port source ou destination du datagramme
- **udp.dstport** (*unsigned 16-bit integer*)
Port destination du datagramme
- **udp.srcport** (*unsigned 16-bit integer*)
Port source du datagramme
- **udp.length** (*unsigned 16-bit integer*)
taille totale du datagramme

3.1.2.e Identificateur et champs pour ICMP

L'identificateur de protocole **icmp** qualifie les trames contenant un message ICMP. La liste complète de ses champs est disponible à l'URL <http://www.wireshark.org/docs/dfref/i/icmp.html>.

Extrait des champs susceptibles de nous intéresser :

- **icmp.code** *(unsigned 8-bit integer)*
code du message
- **icmp.type** *(unsigned 8-bit integer)*
type du message pour le code donné
- **icmp.ident** *(unsigned 16-bit integer)*
identificateur du message (présence dépendant du message)

❗ Leur signification est indiquée dans le document décrivant le format des messages ICMP dans les *Documents Utiles* du cours Ametice.

3.2 Filtres de capture de la libpcap

Un filtre de capture se précise dans la zone *Capture Filter* de la boîte de dialogue des options de capture (figure 4). Il s'exprime, selon la syntaxe de la **libpcap**, avec des **primitives**, des **connecteurs** et les parenthèses. La syntaxe complète de ces filtres est décrite dans le manuel de **pcap-filter** (man 7 pcap-filter).

Nous nous contenterons d'une syntaxe allégée, avec des primitives simples. Dans ce contexte, un filtre suit la grammaire suivante :

```

filtre ::= [not] primitive { (and|or) [not] primitive }
primitive ::= ( filtre )
primitive ::= l'une des primitives ci-dessous

```

Exemple 5

Voici un petit aperçu de filtres possibles :

- **ether host 00:21:ab:28:10:c5**
ne retient que les trames dont l'adresse MAC source ou destination est 00:21:ab:28:10:c5
- **tcp port 23 and host 10.0.0.5**
ne retient que le trafic TELNET (car le port TCP 23 est *a priori* réservé aux serveurs TELNET) impliquant l'hôte 10.0.0.5 (en tant que serveur ou client)
- **tcp port 23 and not src host 10.0.0.5**
ne retient que le trafic TELNET, et ne provenant pas de l'hôte 10.0.0.5
- **host 10.0.0.5 and not (port 80 or port 25)**
ne retient que le trafic hors HTTP et SMTP, impliquant l'hôte 10.0.0.5
- **host 10.0.0.5 and icmp[icmptype] == icmp-echo**
ne retient que les messages ICMP de type ECHO REQUEST dont la source ou la destination est 10.0.0.5



Il existe des primitives pour tous les protocoles reconnus par la **libpcap**. Nous nous limiterons à un sous-ensemble des primitives relatives aux protocoles qui nous occupent.

3.2.1 Primitives pour Ethernet

Ces primitives ne peuvent être vraies que pour les trames Ethernet, FDDI, Token Ring et Wifi.

- **ether host *ehost***
vrai si l'adresse MAC source ou destination de la trame Ethernet est *ehost*
- **ether dst *ehost***
vrai si l'adresse MAC destination de la trame Ethernet est *ehost*
- **ether src *ehost***
vrai si l'adresse MAC source de la trame Ethernet est *ehost*
- **ether proto *protocol***
vrai si le champ *EtherType* de la trame Ethernet est *protocol*, qui peut être un nombre ou un nom tel que **ip**, **ip6**, **arp**, **rarp**,... Attention toutefois car ces noms sont aussi des mots clés de primitives et il faut les protéger avec un *backslash*

3.2.2 Primitives pour IP

Ces primitives ne peuvent être vraies que si la trame transporte un datagramme IP. Cela rend implicite l'un des critères **ether proto ip** et **ether proto ip6**.

- **host** *host*
vrai si l'adresse source ou destination (IPv4/v6) du datagramme est *host*
- **dst host** *host*
vrai si l'adresse destination (IPv4/v6) du datagramme est *host*, qui peut être une adresse ou un nom d'hôte
- **src host** *host*
vrai si l'adresse source (IPv4/v6) du datagramme est *host*
- **dst net** *net* [**mask** *mask*]
vrai si l'adresse destination (IPv4/v6) du datagramme fait partie du réseau *net* de masque *mask*. Si le masque n'est pas précisé, *net* peut être exprimé par un seul octet (masque 255.0.0.0), deux octets *x.y* (masque 255.255.0.0) ou trois octets *x.y.z* (masque 255.255.255.0)
- **src net** *net* [**mask** *mask*]
vrai si l'adresse source (IPv4/v6) du datagramme fait partie du réseau *net*
- **net** *net* [**mask** *mask*]
vrai si l'adresse source ou destination (IPv4/v6) du datagramme fait partie du réseau *net*
- **ip proto** *protocol*
vrai s'il s'agit d'un datagramme IPv4 dont le champ *Protocole* est *protocol*, qui peut être un nombre ou un nom tel que **icmp**, **udp**, **tcp**,... Attention toutefois car ce sont aussi des mots clés de primitives et il faut les protéger avec un *backslash* !

3.2.3 Primitives communes à UDP et TCP

Ces primitives ne peuvent être vraies que si la trame transporte un datagramme UDP ou un segment TCP (inclus dans un datagramme IP).

- [**udp|tcp**] **port** *port*
vrai si le port source ou destination du segment TCP/datagramme UDP est *port*. Le mot clé optionnel du début (**udp** ou **tcp**) précise s'il doit s'agir d'un datagramme UDP ou d'un segment TCP. En son absence, les deux conviennent. *port* peut être un nombre ou un nom figurant dans le fichier */etc/services*. Si un nom est utilisé et qu'il est spécifique à un TCP ou à UDP, seuls les paquets correspondant à ce protocole seront retenus
- [**udp|tcp**] **dst port** *port*
vrai si le port destination est *port*
- [**udp|tcp**] **src port** *port*
vrai si le port source du segment TCP/datagramme UDP est *port*
- [**udp|tcp**] **portrange** *port₁-port₂*
vrai si le port source ou destination du segment TCP/datagramme UDP est compris entre *port₁* et *port₂*
- [**udp|tcp**] **dst portrange** *port₁-port₂*
vrai si le port destination du segment TCP/datagramme UDP est compris entre *port₁* et *port₂*
- [**udp|tcp**] **src portrange** *port₁-port₂*
vrai si le port source du segment TCP/datagramme UDP est compris entre *port₁* et *port₂*

3.2.4 Cas particulier des primitives spécifiques à ICMP

Ces primitives ne peuvent être vraies que si la trame transporte un message ICMP. La syntaxe est un peu différente des primitives précédentes.

- **icmp**
vrai si la trame transporte un message ICMP
- **icmp[icmptype] (==|!=) valeur**
vrai si le type du message ICMP vaut (==) ou est différent de (!=) *valeur*, où *valeur* peut être un nombre ou un mot clé tel que **icmp-echo** (ECHO REQUEST) ou **icmp-echoreply**
- **icmp[icmpcode] (==|!=) valeur**
vrai si le code du message ICMP vaut (==) ou est différent de (!=) *valeur*, où *valeur* peut être un nombre ou un mot clé

❗ Ainsi qu'il a été dit, il existe de nombreuses autres primitives, et la possibilité d'exprimer des conditions plus fines/complexes mais cela sort du cadre de ce TP. Voir le manuel de **pcap-filter** pour des exemples.