Chapitre 2

Étude de quelques protocoles réseau

Nous allons décrire dans ce chapitre quelques protocoles réseau de l'architecture TCP/IP étendue, plus précisément le protocole Ethernet de la couche d'accès, le protocole de résolution d'adresse ARP, les protocoles IPv4 et ICMP de la couche réseau IPv4, les protocoles UDP et TCP de la couche transport et, enfin, le protocole HTTP de la couche application.

Pour ne pas rester sur une base purement théorique, il est traditionnel depuis [STE–94] d'utiliser un analyseur de trames pour illustrer la description des propotocoles. Un **analyseur de trames** permet de récupérer toutes les trames passant par une interface réseau donnée de façon brute à fin d'analyse. Il s'agit des trames partant de l'ordinateur ou arrivant sur cette interface, que celles-ci soient destinées ou non à l'ordinateur ; il faut l'indiquer explicitement à la carte réseau dans ce dernier cas car, par défaut, celle-ci détruit immédiatement les trames qui ne sont pas destinées à l'ordinateur mais il existe un **mode promiscuité** qui permet de les retenir également. Pour des raisons de sécurité, seul le super-utilisateur peut se servir des analyseurs de trames. Un analyseur de trames peut récupérer (et analyser) les trames en entier ou les premiers octets de celle-ci au choix, ce qui permet de donner des indications sur les en-têtes de protocole sans récupérer l'intégralité des données, ce qui est suffisant pour analyser le réseau et ses problèmes éventuels.

Le premier analyseur de trames à avoir connu un grand succès est **tcpdump**, écrit par Van Jacobson, Craig Leres et Steve McCanne, tous du Laboratoire de Lawrence Berkeley, de l'université de Californie à Berkeley, reposant sur une idée de Jeffrey Mogul [MOG–90] à Stanford. Nous avons choisi d'utiliser **wireshark** (connu précédemment sous le nom d'**ethereal**) qui présente une interface graphique très conviviale et qui analyse de nombreux protocoles. Sa page d'accueil indique d'ailleurs qu'il peut être utilisé à titre pédagogique.

2.1 Les en-têtes des unités d'information

2.1.1 Notion

Une *unité d'information* est composée, d'une part, de *données* et, d'autre part, d'*informations administratives* concernant cette unité d'information. Ces informations de contrôle sont généralement assemblées dans un bloc venant avant les données, ce qui est le cas pour tous les protocoles de TCP/IP. Ces informations de contrôle forment alors ce qui est appelé l'**en-tête de protocole**.

Lorsque l'unité d'information est passée à la couche inférieure, celle-ci ajoute son en-tête à l'unité d'information passée, considérée comme les données de cette couche inférieure. De cette manière, lorsqu'une unité d'information est partie de la couche d'application, au moment où il atteint la couche physique, elle contient quatre en-têtes de protocole dans le modèle TCP/IP (sept avec le modèle OSI).

Pour mieux visualiser ce processus, on peut se représenter l'envoi d'un vote par correspondance : une première enveloppe contient l'adresse du bureau de vote et contient une deuxième enveloppe anonyme. L'intérieur de l'enveloppe est constituée par les données à envoyer. À chaque fois que l'unité d'information passe à travers une des couches, une enveloppe est ajoutée. Lorsqu'elle a parcouru toutes les couches, plusieurs en-têtes de protocole entourent les données initiales. Lorsque l'unité d'information est envoyée à travers toutes les couches en partant du bas (sur une autre machine en général), chaque couche ouvre l'enveloppe, ou plus exactement analyse et enlève l'en-tête de protocole qui lui correspond. Lorsque la couche d'application de destination est atteinte, il ne reste plus que les données initiales.

2.1.2 Noms des unités d'information dans TCP/IP

TCP/IP utilise les dénominations suivantes pour les unités d'information :

Application	Message	
Transport	Segment Datagramme	dans le cas connecté dans le cas non connecté
Réseau	Paquet	
Accès	Trame	

trame se disant *frame* en anglais.

2.2 Protocole Ethernet

Le protocole Ethernet a été défini par Xerox. Il a eu tellement de succès que DEC, Intel et Xerox ont défini en 1978 un standard Ethernet à 10 Mbits/s, appelé le standard **DIX** d'après les initiales des protagonistes. Après deux modifications mineures, il est devenu en 1983 la norme IEEE 802.3, puis plus tard une norme ISO.

2.2.1 Adresse physique MAC

Ethernet [IEEE-802.3] et quelques autres protocoles de la couche d'accès utilisent une adresse physique sur 48 bits, appelée **adresse MAC** (*Media Access Control address* en anglais), dont la structure est donnée sur la figure suivante (dans le cas d'une adresse de destination) :

1 bit	1 bit	22 bits	24 bits
I/G	U/L	Partie OUI du constructeur	Partie déterminée par le constructeur

– Le bit `I/G` le moins significatif de l'adresse, est, uniquement dans le cas d'une adresse de destination, le bit d'adresse d'individu I ou de groupe G. Si ce bit vaut 0, l'adresse est celle d'un individu. S'il vaut 1, le reste du champ de l'adresse identifie une adresse de groupe qui nécessite une résolution supplémentaire.

  Lorsqu'une trame est adressée à un groupe, toutes les stations du groupe la reçoivent. Une diffusion de ce type est appelée **diffusion restreinte**, **diffusion multidestinataire** ou **diffusion multidiffusion** (*multicast* en anglais).

– Le deuxième bit, `U/L`, uniquement dans le cas d'une adresse de destination, est le bit local L ou universel U. S'il vaut 0, l'adresse est déterminée (en partie) par un organisme international pour assurer l'unicité de l'adresse MAC au niveau mondial. S'il vaut 1, l'adresse a été définie localement et l'interface ne doit pas être reliée à Internet.

  L'organisme d'administration universel fut d'abord Xerox (le créateur d'Ethernet) puis ensuite l'IEEE au niveau mondial.

– Dans le cas où le deuxième bit vaut 0, les 24 premiers bits constituent l'**OUI** (pour *Organization Unique Identifier*), attribués à chaque constructeur. La liste des OUI se trouve à :

  `http://standards.ieee.org/regauth/oui/oui.txt`

  Les 24 derniers bits sont déterminés par le constructeur de l'interface réseau, que l'on peut quelquefois changer localement.

Si la totalité de l'adresse est composée de bits à 1, l'adresse a une signification spéciale : elle indique que toutes les stations du réseau local sont considérées comme destinataires. On parle de **diffusion générale** (*broadcast* en anglais).

Un constructeur de cartes réseau est donc identifé par 22 bits. Ce système autorise donc 2^{22} combinaisons – soit 4 millions environ – et il est fort possible qu'au rythme de croissance actuel d'Internet, on se retrouve à court d'OUI. Si une organisation est à court d'adresses physiques, l'IEEE a la possibilité de lui affecter un deuxième OUI.

2.2.2 Les trames Ethernet DIX

Commençons par décrire une trame de l'Ethernet d'origine, appelée **trame DIX** (pour *DEC-Intel-Xerox*). Une trame Ethernet ne comprend pas seulement un en-tête et des données. Elle est suivie d'un suffixe comme le montre le schéma suivant :

Préambule	Adresse destinataire	Adresse émetteur	Type	Données	CRC
64 bits	48 bits	48 bits	16 bits	Longueur variable	32 bits

– Le **préambule** est un ensemble de bits dont la fonction principale est de synchroniser le processus de communication et de mettre en évidence tout bruit parasite affectant les premiers bits envoyés.

  Dans le cas d'une trame de l'Ethernet d'origine, chacun des 8 octets du préambule contient la séquence binaire `10101010`. D'un point de vue physique, le codage utilisé, dit *codage Manchester*, de cette séquence produit un signal de forme rectangulaire de 10 MHz pendant 6,4μs, ce qui permet à l'horloge du récepteur de se caler avec celle de l'émetteur. Les parties doivent ensuite rester synchronisées pour le reste de la trame et utiliser le code Manchester pour identifier les délimitations des bits.

  Le code Manchester ne nous intéressera pas ici car il est pris en compte de façon matérielle par l'interface réseau et ne concerne donc pas la partie logicielle.

– Les adresses MAC du destinataire et de l'expéditeur suivent.

En fait la norme autorise des adresses sur 2 ou 6 octets, mais les paramètres définis pour la norme à 10 Mbit/s en bande de base n'emploient que des adresses sur 6 octets. Nous ne nous occuperons ici que des adresses sur six octets.

– Dans le cas d'une trame DIX, le champ suivant de 16 bits, le **type**, permet au récepteur le démultiplexage au niveau de la couche réseau, autrement dit ce que celui-ci doit faire de la trame. En effet plusieurs protocoles de couche réseau peuvent être employés en même temps sur un même ordinateur donc, lorsqu'une trame Ethernet arrive, le sous-système d'exploitation du réseau doit savoir à quel protocole la remettre. C'est ce champ qui lui permet de le savoir.

Voici quelques valeurs de ce champ :

Valeur décimale	Description
512	Xerox PUP
2048	Internet Protocol (IP)
2051	ECMA
2053	X.25 Level 3
2054	Address Resolution Protocol (ARP)
32821	Reverse ARP
32823	Appletalk

Remarquons que 2048 (IP) et 2054 (ARP) sont les seules valeurs qui intéressent l'architecture TCP/IP.

– Les données suivent l'indicateur de type, dont la longueur varie entre 46 et 1 500 octets, la valeur dépendant du type.

La valeur maximum de 1 500 a été choisie presque de façon arbitraire, principalement parce qu'un transcepteur devait disposer de suffisamment de mémoire pour contenir une trame entière et qu'en 1978 la mémoire RAM était coûteuse.

Outre une longueur maximale, une trame doit respecter une longueur minimale. Bien qu'un champ de données de zéro octet soit parfois utile, cela pose problème. Lorsqu'un transcepteur détecte une collision, il tronque la trame en cours, si bien que des bits ou des fragments de trame égarés apparaissent continuellement sur le câble. Pour faciliter la distinction entre les trames valides et les rebuts parasites, Ethernet exige que les trames valides possèdent au moins 64 octets, du champ d'adresse de destination au champ de somme de contrôle, ces deux champs inclus. Si le champ des données a une taille inférieure à 46 octets, elles sont complétées par des 0 jusqu'à obtenir 46 octets, ce qui constitue le **champ de remplissage**.

– À la fin de la trame se trouvent quatre octets de vérification, (**CRC** pour *Cyclic Redundancy Check*), qui permet de s'assurer plus ou moins que le contenu de la trame n'a pas été modifié lors de la transmission. Toute passerelle sur la route de transmission doit calculer la valeur de CRC pour la trame et la comparer à la valeur qui figure à la fin de celle-ci. Si les deux sont égales, la trame est envoyée un peu plus loin dans le réseau. Sinon, c'est que la trame a été altérée en cours de route et elle doit donc être détruite (elle sera éventuellement retransmise – après expiration d'un temporisateur (*timer*) – par la machine qui l'a envoyée).

2.2.3 Les trames Ethernet 802.3

Nous avons vu qu'Ethernet, conçu à l'origine par Xerox, est devenu en 1983 un standard IEEE. L'**IEEE** (*Institute of Electrical and Electronics Engineers*) est la plus grande institution professionnelle au monde. Elle publie de nombreuses revues, organise des centaines de conférences

par an et dirige une section chargée de développer des normes dans les domaines de l'électronique et de l'informatique. La commission IEEE 802 a notamment standardisé plusieurs types de réseaux locaux.

2.2.3.1 La commission 802 de l'IEEE

Ethernet est le plus célèbre des types de réseaux locaux normalisés par IEEE mais ce n'est pas le seul : il existe aussi, entre autres, les bus à jeton (*token bus*), l'anneau à jeton (*token ring*), tous les deux maintenant pratiquement abandonnés et surtout Wi-Fi (*Wireless Fidelity*, fidélité sans fil) et Bluetooth.

Le tableau ci-dessous présente les groupes de travail dans le cadre du comité IEEE 802 et l'avancement des travaux :

Numéro	Sujet	Avancement
802.1	Vue d'ensemble sur l'architecture des LAN	
802.2	Contrôle logique de liaison (LLC)	↓
802.3	Ethernet	*
802.4	Bus à jeton	↓
802.5	Anneau à jeton	
802.6	DQDB (premier MAN)	↓
802.7	Technologies à large bande	↓
802.8	Fibre optique	†
802.9	LAN en temps réel	↓
802.10	LAN virtuel (VLAN) et sécurité	↓
802.11	LAN sans fil (Wi-Fi)	*
802.12	AnyLAN de Hewlett-Packard	↓
802.13	Numéro non affecté par superstition	
802.14	Modem câble	↓
802.15	Réseaux personnels (PLAN, Bluetoooth)	*
802.16	Sans fil à large bande	*
802.17	RPR (Resilient Packet Ring)	

* : groupe parmi les plus importants, ↓ : groupe en suspens, † : groupe abandonné.

2.2.3.2 Deux sous-couches de la couche liaison

Le standard IEEE 802 (figure 4.1, [WPRMB–02], p. 136) distingue deux sous-couches dans la couche de liaison des données [IEEE-802] :

- la **sous-couche de contrôle d'accès au canal**, ou **sous-couche MAC** (pour l'anglais *Medium Access Control*), ou encore sous-couche 2a, la plus proche de la couche physique comme son nom le laisse entendre, est chargée de la gestion des accès à un canal partagé.
- La **sous-couche de contrôle de liaison logique** (**LLC** pour *Logical Link Control*), ou sous-couche 2b, doit dissimuler les différences spécifiques aux supports et offrir une interface uniforme aux protocoles des couches supérieures. C'est cette sous-couche également qui s'occupe du contrôle de flux et des erreurs.

Ce découpage en deux sous-couches présente des avantages :

- Pour transporter des paquets IP, par exemple, aucune garantie de livraison n'est requise ni attendue. On n'a donc pas besoin d'utiliser la sous-couche LLC.
- La couche LLC est indépendante du type de réseau local : Ethernet ou WiFi.

Lors de l'implémentation :

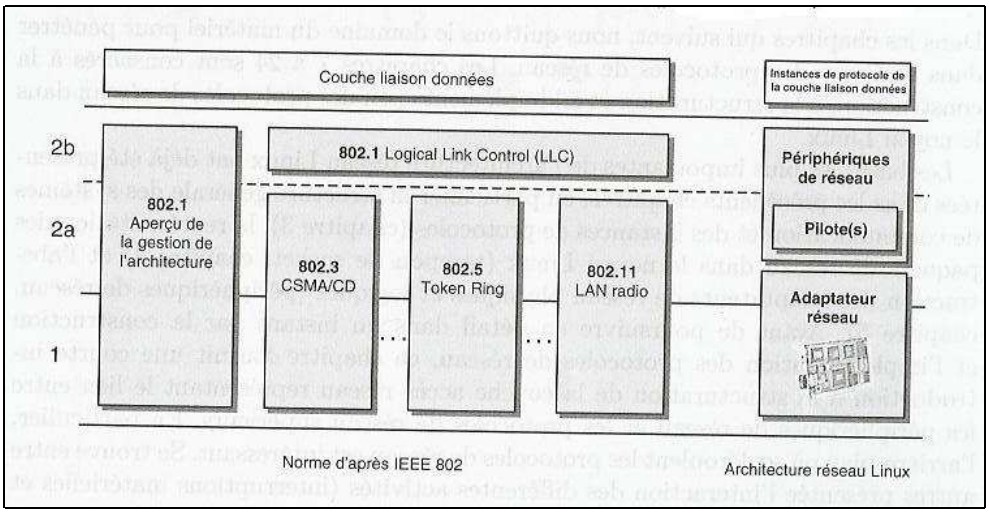


FIG. 2.1 – Sous-couches MAC et LLC

- L'interface avec la couche 1 (couche physique) et la sous-couche 2a (sous-couche MAC) sont souvent implémentées dans les cartes réseau.
- La sous-couche 2b (LLC) est implémentée dans le sous-système réseau du noyau du système d'exploitation.

2.2.3.3 Protocole 802.3 de la sous-couche LLC

Dans le cas de 802.3, le champ données de la trame ne comporte pas d'en-tête de sous-couche 2b ; par contre la signification des champs de l'en-tête Ethernet (*a priori* de la sous-couche MAC) a légèrement changé. On parle alors de **trame Ethernet 802.3**. Il y a deux changements :

- Le premier changement concerne le préambule. Les sept premiers octets restent inchangés ; le huitième octet sert de délimiteur de début de trame (**SFD** pour *Start Frame Delimiter*) afin de rendre le format compatible avec les normes 802.4 et 802.5.
- Le second changement a d'abord transformé le champ de type en champ de **longueur** de trame. Bien sûr, il n'était plus possible pour un récepteur de savoir ce qu'il fallait faire avec une trame entrante, mais ce problème a été résolu par l'ajout d'un petit en-tête dans la portion des données pour apporter cette information.

  Malheureusement, à l'époque où la norme 802.3 fut publiée, il y avait déjà tellement d'équipements et de logiciels en circulation pour l'Ethernet DIX que peu de fabricants et d'utilisateurs ont accueilli avec enthousiasme le changement du champ de type en champ de longueur. En 1997, l'IEEE jeta l'éponge et annonça que les deux formats étaient valides. Heureusement, la plupart des champs de type utilisés avant 1997 étaient d'une valeur supérieure à 1 500 (la longueur maximum des données). Par conséquent, une valeur inférieure ou égale à 1 500 peut être considérée comme étant une longueur et une valeur supérieure à 1 500 comme étant un type.

2.3 Mise en place de Wireshark

La page d'accueil de `Wireshark` (le requin des fils, antérieurement connu sous le nom d'`Ethereal`) est :

http : //www.wireshark.org/

Le menu de téléchargement donne accès à des binaires pour Windows (32 ou 64 bits) ainsi que le code source pour les autres systèmes d'exploitation.

2.3.1 Installation

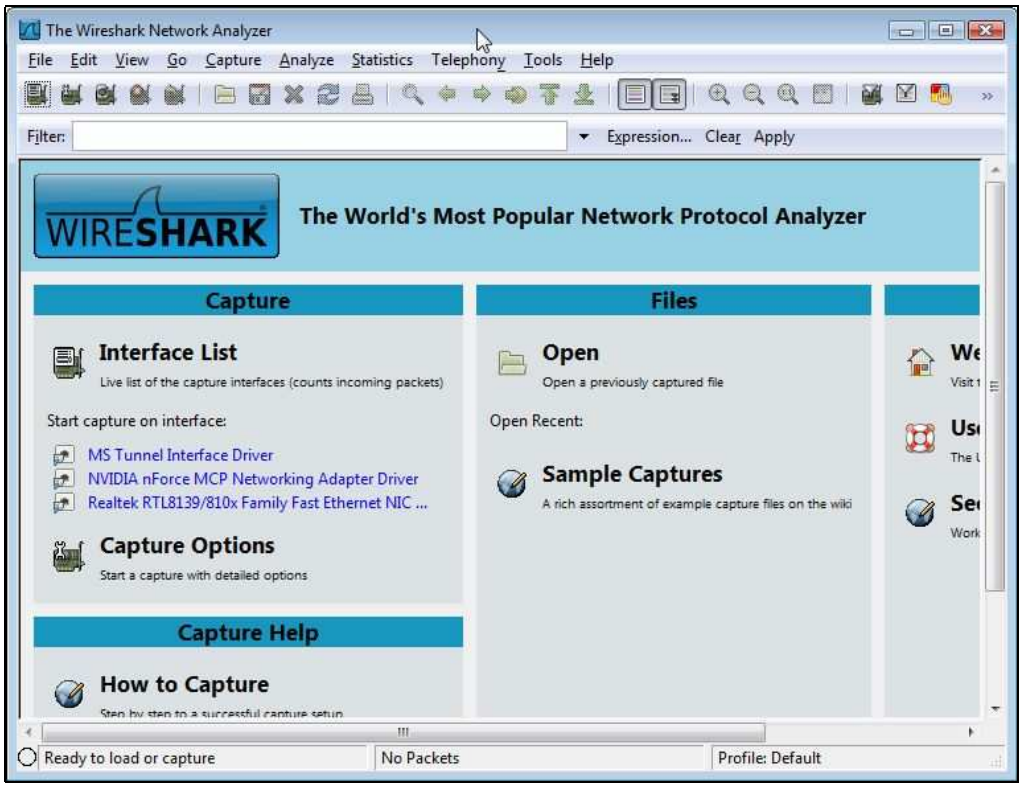


FIG. 2.2 – Fenêtre d'accueil de Wireshark

2.3.1.1 Cas de Windows

Il suffit de télécharger la dernière version (1.2.1 en septembre 2009 de 17,4 Mo ; 77,3 Mo une fois décompressé) et de l'exécuter (en tant que super-utilisateur).

2.3.1.2 Cas de Linux

`Wireshark` est présent sur la plupart des distributions. Il suffit donc de demander d'installer ce logiciel.

2.3.2 Utilisation

Sous Windows, l'appel en cliquant sur l'icône de `wireshark.exe` fait apparaître la fenêtre de la figure 2.2. Sous Linux, il suffit de taper `wireshark &` en ligne de commande, ce qui donne accès directement à ce que l'on voit sur la figure 2.3.

En cliquant sur le nom de l'interface réseau désiré de la liste d'interface apparaissant dans la fenêtre `Capture`, on fait apparaître la fenêtre de la figure 2.3.

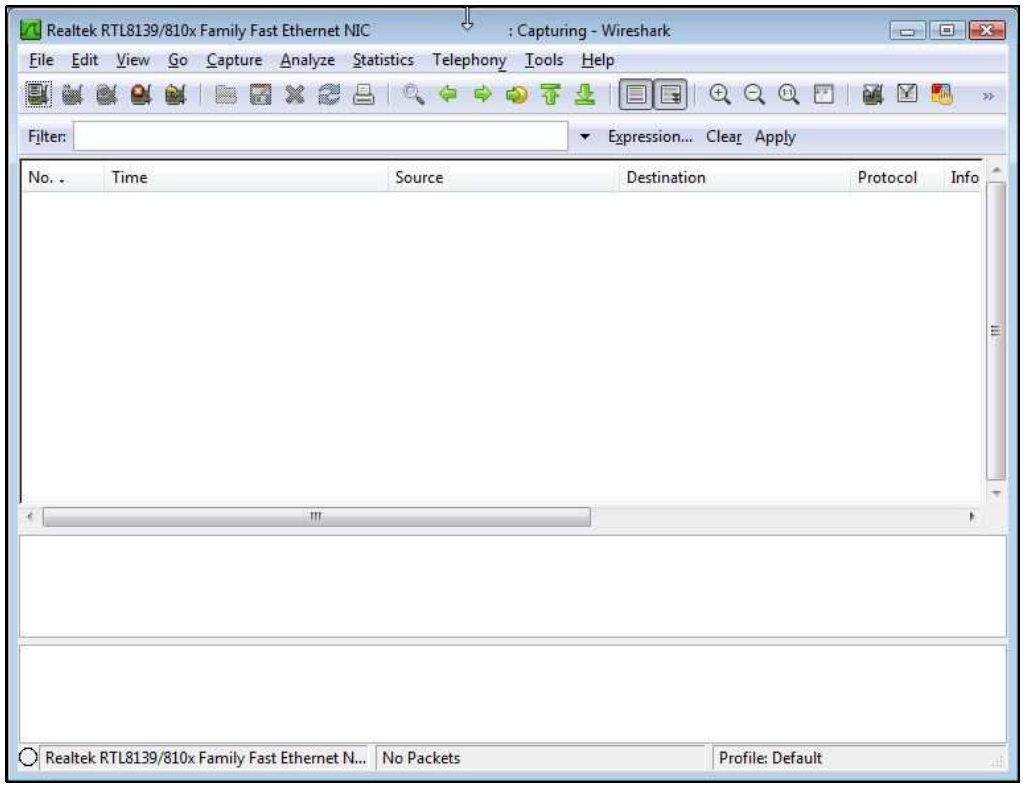


FIG. 2.3 – Fenêtre d'annonce de capture de Wireshark

Mon ordinateur est relié à Internet grâce à un routeur ADSL. Sur mon réseau local ainsi constitué, l'adresse IP du routeur ADSL est classiquement 192.168.1.1 et celle de mon ordinateur 192.168.1.12.

Sous Windows, utilisons `putty` pour effectuer un `telnet`, comme montré sur la figure 2.4.

La fenêtre de capture indique alors que que deux trames (au moins) ont été capturées. Appuyons sur le bouton `arrêter` de celle-ci. La première fenêtre prend alors l'aspect de la figure 2.5.

On y voit trois zones :

- Dans une première zone apparaissent les trames capturées avec un numéro de trame, l'heure à laquelle elle a été capturée (par défaut depuis le lancement de la capture mais on peut demander l'heure réelle), l'adrese réseau source de la trame (adresse IP en général, mais pas toujours), l'adresse réseau de la destination, le protocole de plus haut niveau détecté et quelques autres informations.
- Dans une deuxième zone le nombre d'octets de la trame ainsi que le nombre d'octets cap-

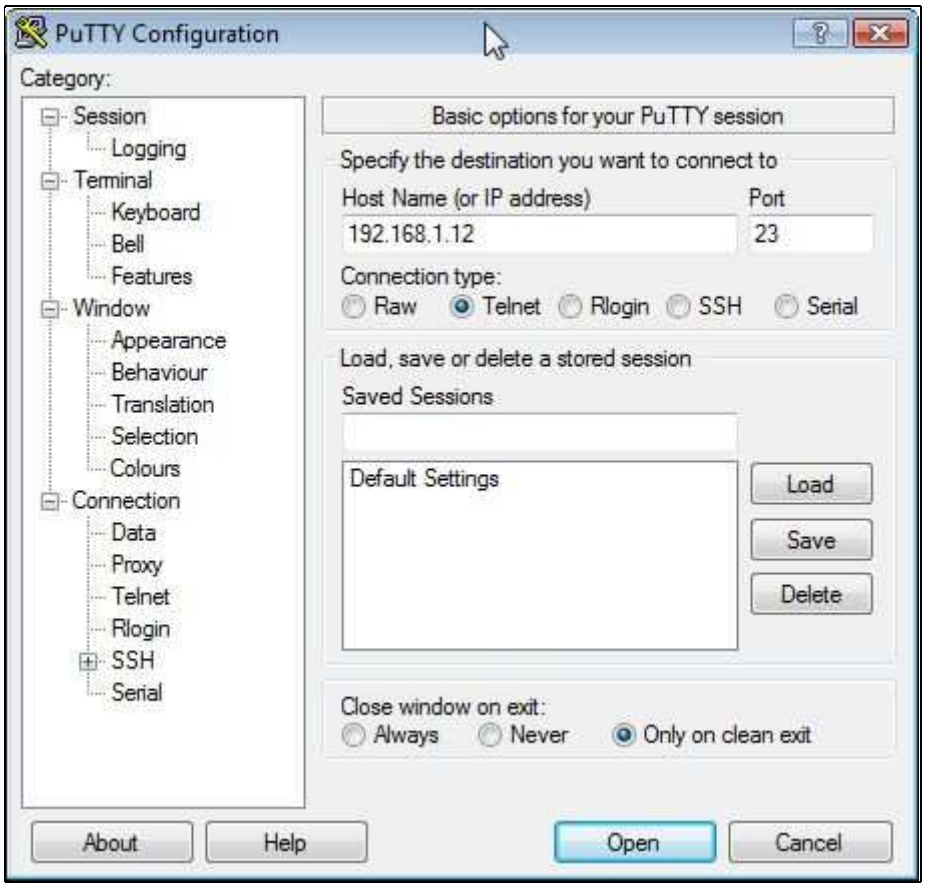


FIG. 2.4 – Utilisation de Putty

turés sont indiqués, suivi du détail des en-têtes de tous les protocoles jusqu'au protocole final (ici Ethernet et ARP).

– Dans la troisième zone apparaissent les octets capturés de la trame. Chaque ligne contient seize octets. Elle commence par le numéro (en héxadécimal) du premier octet de la ligne, suivi de deux groupes de huit octets en hexadécimal, suivi du caractère ASCII correspondant (remplacé par un point si celui-ci n'est pas affichable).

2.4 Analyse des en-têtes Ethernet

Cliquons sur le petit carré contenant un signe plus au début de ligne Ethernet de la deuxième zone. On obtient l'aspect de la figure 2.6 : d'une part on a un petit plus de renseignements sur l'en-tête de la trame ; d'autre part les octets de la trame concernant cet en-tête est surchargée en bleu foncé (que ce soit en hexadécimal ou en ASCII).

Les trames transitant sur le support physique sont récupérées par l'interface réseau (une carte Ethernet). Le préambule et le champ CRC ne sont pas placés dans le tampon de l'interface (puisqu'ils sont censés correspondre à ce qu'il faut) et donc ne sont pas récupérés par le sous-système réseau du système d'exploitation.

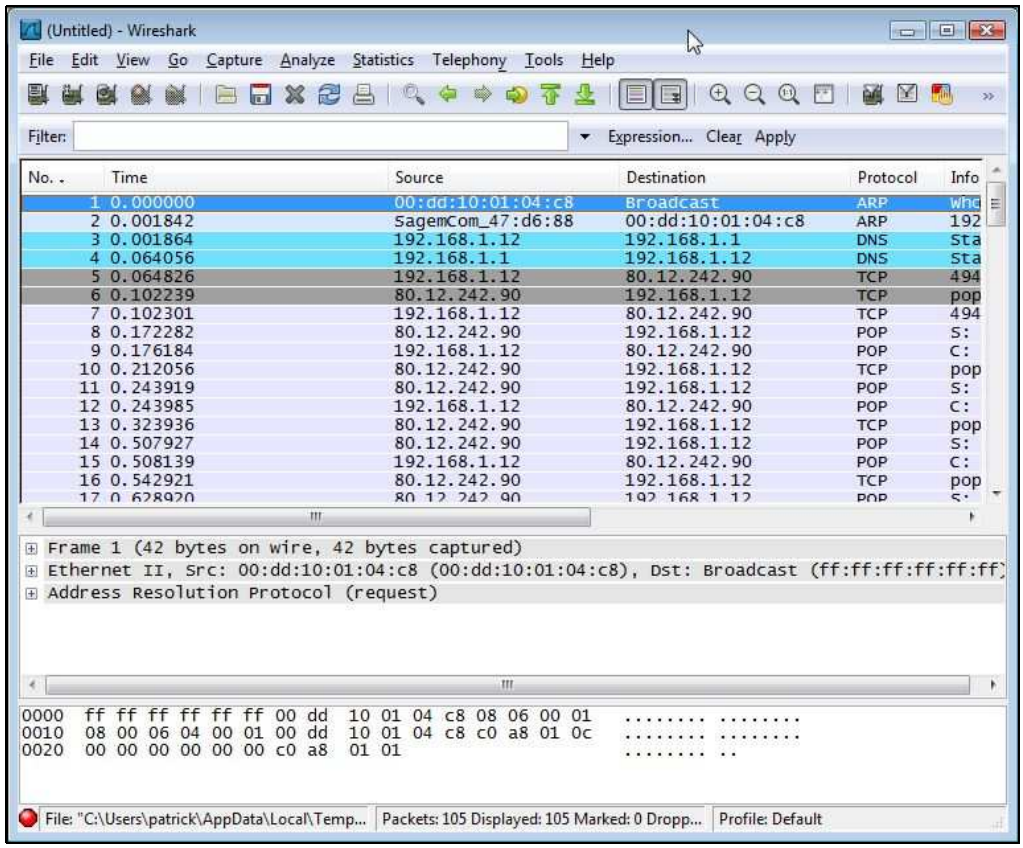


FIG. 2.5 – Liste des trames capturées

– L'en-tête de la trame commence donc par les 48 bits (ou six octets) de l'adresse MAC du destinataire. Il s'agit ici d'un envoi à toutes les interfaces du réseau local. Puisqu'il est habituel de noter les adresses MAC en hexadécimal, il n'est pas difficile de la traduire : `ff:ff:ff:ff:ff:ff`.

– Cette adresse MAC est suivie des 48 bits (ou six octets) de l'adresse MAC de l'expéditeur. Il s'agit ici de notre ordinateur, le seul qui soit sur notre réseau local (à part le routeur). Ce n'est pas toujours le cas dans un vrai réseau local avec plusieurs hôtes lorsqu'on utilise le mode promiscuité : on peut récupérer et analyser des trames qui ne nous sont pas destinées ou que nous n'avons pas envoyées. L'adresse MAC de notre carte Ethernet est donc $00 : dd : 10 : 01 : 04 : c8$.

  Une recherche dans le document `oui.txt` montre que l'adresse du constructeur n'y est pas référencée. Il s'agit d'une carte chinoise dont le constructeur *realteck*) n'a visiblement pas demandé d'OUI.

– Les seize bits (ou deux octets) suivants indiquent le type du protocole de la couche supérieure. On a ici `0806h`, soit 2054, qui spécifie le protocole réseau ARP.

– Les reste est constitué des données Ethernet, commençant par l'en-tête ARP sur lequel nous reviendrons ci-dessous.

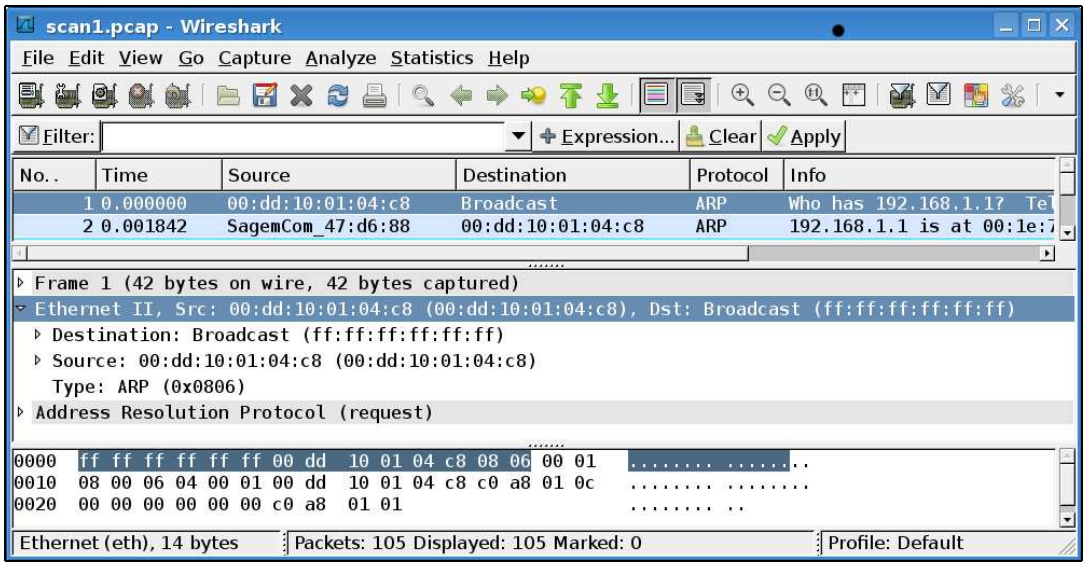


FIG. 2.6 – En-tête Ethernet

2.5 Le protocole de résolution d'adresse ARP

Les en-têtes correspondant à chaque protocole contiennent un certain nombre de paramètres. Le paramètre le plus important pour la couche d'accès est l'**adresse physique** qui permet de déterminer à quel ordinateur du réseau local la trame est destinée. Il s'agit de l'adresse MAC dans le cas d'Ethernet. Le paramètre le plus important pour la couche réseau est l'**adresse réseau** qui permet de déterminer à quel ordinateur de l'interréseau le paquet est destiné. Il s'agit de l'adresse IP dans le cas de IP. La conversion des adresses entre la couche d'accès et la couche réseau représente une tâche essentielle lors de l'identification claire des ressources dans un réseau informatique.

Pour pouvoir envoyer une unité d'information à une instance IP de l'ordinateur cible ou du routeur le plus proche, l'adresse MAC du prochain saut doit être déterminée dans l'instance de protocole émettrice. Ceci s'effectuait à l'aide de tables statiques dans chaque ordinateur au début de l'ARPANET. Toutefois le développement de l'ARPANET a rendu cette méthode trop rigide et trop onéreuse en mémoire. C'est pourquoi la [RFC 826] a introduit l'ARP (*Address Resolution Protocol*) en vue de la conversion des formats d'adresses.

Bien que la famille de protocoles TCP/IP soit devenue la norme de premier plan pour la presque totalité des réseaux informatiques, l'ARP n'a pas seulement été mis au point pour l'attribution spécifique d'adresses IP et MAC.

2.5.1 Adresse IP

2.5.1.1 Définition

IPv4 utilise une adresse sur 32 bits pour identifier une machine sur l'inter-réseau, appelée **adresse IP** ou **adresse Internet**.

Les adresses IP sont attribuées par le NIC dont nous avons déjà parlé. Un réseau qui n'est pas connecté à Internet peut déterminer son propre adressage mais, pour tous les accès Internet, l'adresse doit être enregistrée auprès du NIC, pour éviter les doublons.

Le NIC désigne, pour chaque pays, un représentant et une plage d'adresses. Le représentant est chargé d'attribuer les adresses dans ce pays, par exemple :

`http://www.afnic.fr`

pour la France.

2.5.1.2 Représentation utilisateur

Puisqu'il est difficile de lire un nombre de 32 bits, on représente souvent ces nombres dans les applications sous la forme de quatre nombres décimaux de 8 bits, séparés par des points, par exemple :

`127.40.8.72`

127 représentant dans ce cas l'octet de poids fort.

2.5.2 Requête et cache ARP

Lorsqu'un ordinateur A veut envoyer un paquet à l'ordinateur (ou routeur) B, situé dans le même réseau local et dont il connaît l'adresse IP mais pas l'adresse MAC, il envoie une demande (**requête ARP**) à tous les ordinateurs du réseau local (en diffusion générale d'adresse MAC `FF:FF:FF:FF:FF:FF`). L'ordinateur recherché reconnaît à l'adresse IP placée dans le paquet ARP que la requête lui est destinée et envoie une réponse à l'ordinateur demandeur A, lui communiquant alors son adresse MAC.

Pour ne pas être contraint de demander à nouveau l'adresse MAC lors des paquets suivants, A enregistre l'adresse MAC de B dans une table locale, appelée **cache ARP**. L'ordinateur B peut également extraire l'adresse MAC de l'ordinateur A à partir de la requête et l'enregistrer à titre conservatoire dans son cache ARP.

2.5.3 Structure des en-têtes ARP

Les requêtes et les réponses ARP ont une structure identique. Elles sont différenciées par le champ `opération`. Cette structure est la suivante :

<table>
<tr><td>0 7</td><td>8 15</td><td>16 31</td></tr>
<tr><td colspan="2">Type matériel</td><td>Type de protocole</td></tr>
<tr><td>Longueur d'adresse couche 2 (n)</td><td>Longueur d'adresse couche 3 (m)</td><td>Opération</td></tr>
<tr><td colspan="3">Adresse de l'émetteur (couche 2) : n octets</td></tr>
<tr><td colspan="3">Adresse de l'émetteur (couche 3) : m octets</td></tr>
<tr><td colspan="3">Adresse cible (couche 2) : n octets</td></tr>
<tr><td colspan="3">Adresse cible (couche 3) : m octets</td></tr>
</table>

- Le **type matériel** spécifie le protocole de la couche 2 utilisé. Il s'agit par exemple de la valeur 1 pour le protocole Ethernet.
- Le **type de protocole** spécifie le protocole de la couche 3 utilisé, par exemple `0x08 00` pour IPv4.
- La **longueur de l'adresse de la couche 2** spécifie la longueur n, en octets, de l'adresse de la couche 2 pour le protocole utilisé. Dans le cas d'une adresse MAC de 48 bits, on a donc $n = 6$.

- La **longueur de l'adresse de la couche 3** spécifie la longueur m, en octets, de l'adresse de la couche 3 pour le protocole utilisé. Dans le cas d'une adresse IPv4 de 32 bits, on a donc $m = 4$.
- Le champ **opération** indique le type de l'unité de données de protocole ARP : 1 pour une requête ARP, 2 pour une réponse ARP.
- Les champs **adresse de la couche 2 de l'expédieur** et **adresse de la couche 2 du destinataire** sont constitués chacun de n octets.
- Les champs **adresse de la couche 3 de l'expédieur** et **adresse de la couche 3 du destinataire** sont constitués chacun de m octets.

Ces commandes sont encapsulées entre un en-tête et un suffixe de couche 2 et constituent ainsi une trame qui est envoyée en diffusion générale.

Exemple. L'ordinateur A, d'adresse MAC 49 : 72 : 16 : 08 : 64 : 14 et d'adresse IP `129.25.10.72`, qui veut rechercher l'adresse MAC de l'ordinateur d'adresse IP `129.25.10.11`, envoie l'unité suivante à $FF : FF : FF : FF : FF : FF$:

0 7	8 15	16 31
0x00 01		0x08 00
6	4	0x00 01
49 72 16 08 64 14		
129 25 10 72		
00 00 00 00 00 00		
129 25 10 11		

L'ordinateur B, d'adresse MAC `49 :78 :21 :21 :23 :90` et d'adresse IP `129.25.10.11`, répond alors à `49 :72 :16 :08 :64 :14`, c'est-à-dire à l'ordinateur A :

0 7	8 15	16 31
0x00 01		0x08 00
6	4	0x00 02
49 72 16 08 64 14		
129 25 10 72		
49 78 21 21 23 90		
129 25 10 11		

2.5.4 Analyse des messages ARP

2.5.4.1 Cas d'une requête

Nous avons vu que la première trame capturée ci-dessus est un message ARP (figure 2.6). Analysons-le. Nous avons ce qui apparaît à la figure 2.7.

- L'en-tête de la trame Ethernet a déjà été étudié ci-dessus. Passons donc aux données de cette trame, qui contient le message ARP.
- Les deux premiers octets indiquent le type de la couche d'accès : il s'agit ici de 1 pour Ethernet. Les deux octets suivants indiquent le type de la couche réseau : il s'agit ici de `800h`, ou 2 048, pour le protocole IPv4.
- L'octet suivant indique la longueur d'une adresse de la couche d'accès : on a ici 6 octets, soit 48 bits, ce qui est bien la longueur d'une adresse MAC. L'octet suivant indique la longueur d'une adresse de la couche réseau : on a ici 4 octets, soit 32 bits, ce qui est bien la longueur d'une adresse IPv4.

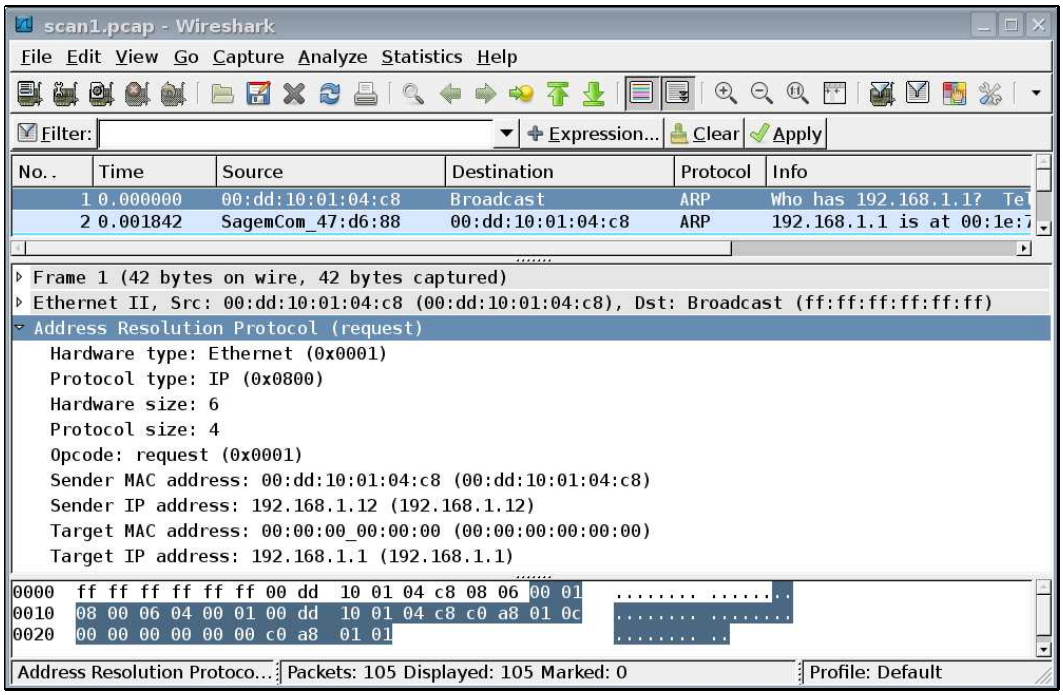


FIG. 2.7 – Requête ARP

- Les deux octets suivants indiquent le type d'opération : puisque c'est 1, il s'agit d'une requête ARP.
- Les six octets (d'après ce que nous avons déterminé ci-dessus) suivants spécifient l'adresse MAC de l'expéditeur : on retrouve ici l'adresse MAC de notre ordinateur. Les quatre octets (d'après ce que nous avons déterminé ci-dessus) suivants spécifient l'adresse IP de l'expéditeur : on retrouve ici l'adresse IP de notre ordinateur.
- Les six octets suivants spécifient l'adresse MAC que nous recherchons : elle est donc initialisée à zéro. Les quatre octets suivants spécifient l'adresse IP de l'hôte dont on recherche l'adresse MAC : on trouve ici l'adresse IP de ce qui est certainement le routeur pour ce réseau (192.168.1.1).

2.5.4.2 Cas de la réponse

La deuxième trame de notre capture, qui apparaît à la figure 2.8, constitue la réponse.

Commençons par remarquer que la taille de la trame capturée est de soixante octets alors que la trame précédente était de quarante-deux octets. Ceci est dû au fait que la longueur minimale d'une trame Ethernet (sans préambule et sans CRC) est de 60 octets pour éviter de prendre en compte les débuts de trame qui subissent une collision. Les 18 derniers octets, tous à zéro, constituent le **remplissage** (*trailer* en anglais). Le champ de remplissage de la trame précédente était vide car la trame a été récupérée avant d'être envoyée à l'interface réseau, or c'est celle-ci qui ajoute le remplissage éventuel.

- Le type de l'en-tête Ethernet est encore `0806h`, soit 2 054, pour le protocole ARP. Les données de la trame constituent le message ARP.

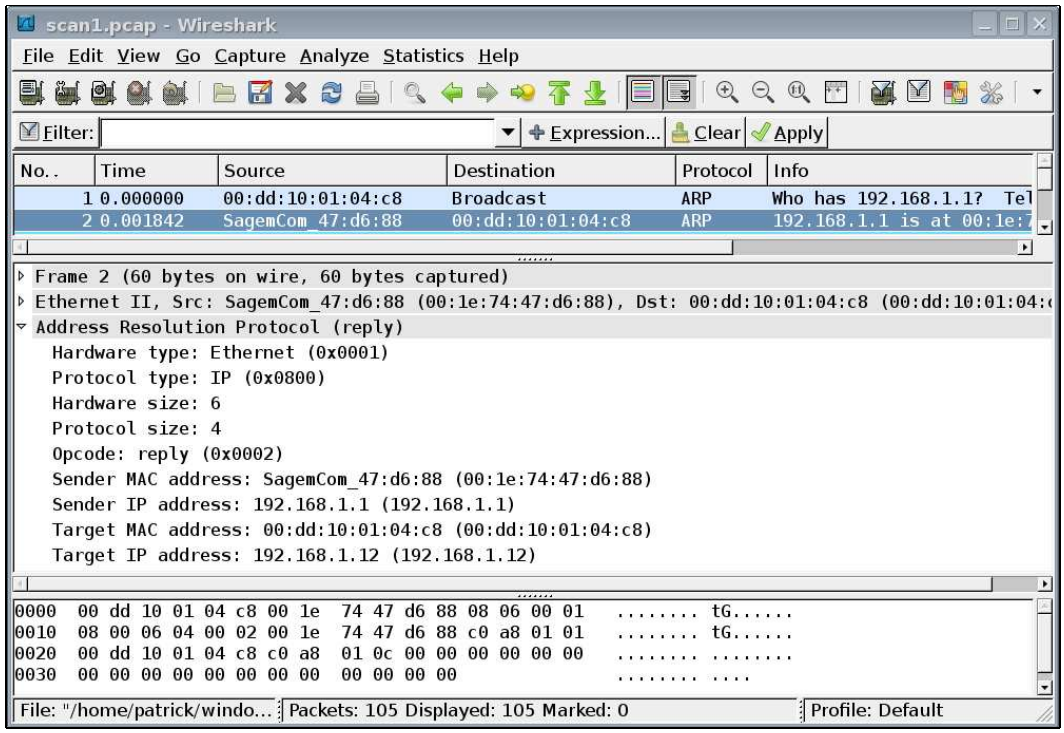


FIG. 2.8 – Réponse ARP

- Les deux premiers octets du message ARP indiquent le type de la couche d'accès : il s'agit ici de 1 pour Ethernet. Les deux octets suivants indiquent le type de la couche réseau : il s'agit ici de `800h`, ou 2 048, pour le protocole IP.
- L'octet suivant indique la longueur d'une adresse de la couche d'accès : on a ici 6 octets, soit 48 bits, ce qui est bien la longueur d'une adresse MAC. L'octet suivant indique la longueur d'une adresse de la couche réseau : on a ici 4 octets, soit 32 bits, ce qui est bien la longueur d'une adresse IPv4.
- Les deux octets suivants indiquent le type d'opération : il s'agit ici de 2 pour une réponse ARP.
- Les six octets (d'après ce que nous avons déterminé ci-dessus) suivants spécifient l'adresse MAC de l'expéditeur : il s'agit de l'adresse MAC recherchée. Les quatre octets (d'après ce que nous avons déterminé ci-dessus) suivants spécifient l'adresse IP de l'expéditeur.
- Les six octets suivants spécifient l'adresse MAC du destinataire : on retrouve l'adresse MAC de notre ordinateur. Les quatre octets suivants spécifient l'adresse IP du destinataire : on retrouve ici l'adresse IP de notre ordinateur.

2.6 Le protocole de couche réseau IPv4

2.6.1 Étude générale de la couche réseau

2.6.1.1 Ce que fait la couche réseau

La couche réseau gère le déplacement des paquets dans un réseau, c'est-à-dire sur des machines qui ne sont pas nécessairement adjacantes. Elle **fragmente** éventuellement les données de la couche transport, pour obtenir des paquets d'une taille standard, et est responsable du **routage** des paquets, c'est-à-dire de la détermination du chemin à suivre par le paquet en proposant des itinéraires de substitution en cas de problème.

2.6.1.2 Ce que ne fait pas la couche réseau

Nous venons de voir que la couche réseau s'occupe de la fragmentation et du routage. Elle ne se préoccupe pas de la fiabilité de la livraison des paquets. Cette dernière n'est pas garantie car le paquet peut être retardé, mal routé ou altéré lors du fractionnement et du réassemblage des fragments. Il n'existe pas de fonctionnalité permettant de vérifier qu'un paquet envoyé a été correctement reçu : comme nous le verrons, l'en-tête de la couche réseau dispose d'une somme de contrôle du contenu de l'en-tête d'un paquet mais pas de son contenu. Elle peut essayer de deviner le meilleur itinéraire jusqu'au prochain nœud sur le chemin, mais ne vérifie pas que le chemin choisi est le plus rapide ou le plus efficace.

2.6.1.3 Les tribulations d'un paquet IP

Hôte de départ.- Lorsqu'une application doit envoyer un paquet sur le réseau, la partie du sous-système réseau chargée du protocole IPv4 de la couche réseau effectue un certain nombre de tâches simples :

- Elle vérifie tout d'abord si le paquet doit être fragmenté. IP autorise une taille de paquet maximale de 65 535 octets, ce qui est bien supérieur à ce que peuvent traiter la plupart des couches inférieures. Si besoin est, elle crée les fragments.
- Elle construit ensuite, pour chaque fragment, le paquet IP en respectant les longueurs valides, telles qu'elles sont spécifiées par l'implémentation IP locale. Une **somme de contrôle** est calculée pour les données, puis l'en-tête IP est construit.
- Ensuite commence le routage. Il est nécessaire de déterminer le premier **saut** (*hop* en anglais), c'est-à-dire la première machine de la route vers la destination, afin de router le paquet vers la machine de destination : la machine elle-même ou une autre machine du réseau local, la machine de destination ou une **passerelle** si on utilise un inter-réseau. Si un routage précis est prédéterminé (à titre de contrôle en général), les informations sur celui-ci sont ajoutées à l'en-tête au moyen d'une **option**.
- Enfin le paquet est transmis à la couche inférieure pour être envoyé sur le réseau physique.

Passerelles et routeurs.- À mesure qu'un paquet traverse l'inter-réseau, chaque passerelle effectue une série de tests :

- Une fois que la couche inférieure a enlevé l'en-tête de la trame reçue, la couche IP de la passerelle calcule la somme de contrôle et vérifie l'intégrité du paquet. Si les sommes de contrôle ne correspondent pas, le paquet est détruit et un message d'erreur est envoyé au composant émetteur *via* ICMP.

- Ensuite, un champ de **durée de vie** (TTL pour *Time To Live* en anglais) de l'en-tête IP est décrémenté puis examiné. Si la durée de vie du paquet a expiré, le paquet est écarté et un message d'erreur est envoyé au composant émetteur *via* ICMP.
- Après avoir déterminé le prochain saut de la route, en analysant l'adresse de destination ou à partir du routage spécifique indiqué dans le champ option de l'en-tête IP, le paquet est reconstruit avec une nouvelle valeur de TTL et donc une nouvelle somme de contrôle.
- Si une fragmentation est nécessaire, soit parce que la longueur du paquet a augmentée, soit du fait d'une limitation du logiciel d'envoi de la passerelle, le paquet est divisé, et de nouveaux paquets, contenant les informations d'en-tête adéquates, sont créés.
- Si un routage ou une estampille temporelle est nécessaire, il est ajouté.
- Enfin, le paquet est renvoyé à la couche inférieure.

Hôte de destination.- Lorsque le paquet est finalement reçu par le composant récepteur :

- Le système effectue un calcul de la somme de contrôle et vérifie que celle-ci concorde avec le champ adéquat de l'en-tête. Si ce n'est pas le cas, un message ICMP est envoyé à l'émetteur.
- Il regarde ensuite s'il y a des fragments supplémentaires. L'opération inverse de la fragmentation s'appelle le **réassemblage**. Si d'autres paquets sont nécessaires pour réassembler le paquet originel, le système attend, après avoir lancé un **minuteur de réassemblage**, pour ne pas être bloqué si les paquets suivants mettent trop de temps à arriver.
- Si le composant ne peut réassembler toutes les parties du paquet originel avant que le minuteur n'atteigne 0, la partie du paquet arrivée est détruite et un message d'erreur est envoyé à l'émetteur *via* ICMP.

  Une des conséquences de cette étape est qu'un paquet fragmenté a moins de chances d'arriver à bon port qu'un paquet non fragmenté, ce qui explique pourquoi la plupart des applications essaient d'éviter autant que possible la fragmentation.
- Enfin l'en-tête IP est enlevé, le paquet originel est reconstruit (s'il a été fragmenté) et il passe à travers les couches pour arriver à la couche d'application.

2.6.1.4 Fragmentation

Pour pouvoir envoyer des paquets IP sur tous les types de réseau physique, le protocole IP doit être en mesure d'adapter la dimension de ceux-ci au type de réseau en place. Nous avons déjà vu que tout réseau physique comporte une taille de trame maximale, qualifiée de *MTU* (pour l'anglais *Maximum Transfert Unit*, unité de transfert maximale). Seuls des trames de cette taille peuvent être envoyées sur le réseau.

Si la MTU d'un support de transfert est inférieure à la taille d'un paquet à expédier, ce dernier doit être divisé en paquets IP plus petits, appelés **fragments**.

Il ne suffit pas que les protocoles de la couche de transport n'expédient que des paquets convenant à la MTU, tout au moins dans le cas non connecté. Un paquet peut traverser plusieurs réseaux différents avec des MTU différentes sur le trajet de l'hôte source à l'hôte cible. C'est pourquoi il faut appliquer une procédure plus flexible capable également de générer des paquets plus petits au sein d'un système de transmission (routeur) sur la couche IP. Cette procédure est appelée **fragmentation**.

La fragmentation sous-entend que le protocole IP de tout ordinateur IP (que ce soit un routeur ou un système d'extrémité) doit être en mesure de reconstituer ces fragments pour obtenir le paquet d'origine (*réassemblage*).

Chaque fragment d'un paquet IP découpé est un paquet IP complet qui contient un en-tête IP. Le paquet originel auquel appartient un fragment se reconnait grâce au champ identificateur

de l'en-tête IP. Ce champ seul ne suffit pas à déterminer le fragment. On utilise également les champs adresse de l'émetteur, adresse cible et protocole.

Les différents fragments d'un datagramme peuvent emprunter des itinéraires différents pour parvenir à l'ordinateur cible et être également fragmentés plusieurs fois sur leurs trajets. La position des données d'un fragment au sein du datagramme IP d'origine est identifiée grâce au champ `décalage de fragment` de leur en-tête IP. Les sous-paquets sont repérés par un décalage et non par un numéro parce qu'un sous-paquet peut être fragmenté à tout moment le long de sa route. On ne peut pas revenir en arrière pour renuméroter les fragments.

Tous les fragments, jusqu'à l'avant-dernier, comportent un bit `MF` (pour l'anglais *More Fragments*) pour indiquer que d'autres fragments vont suivre.

La figure 2.9 ([WPRMB–02], p. 280) montre l'exemple d'un paquet IP qui doit être fragmenté plusieurs fois.

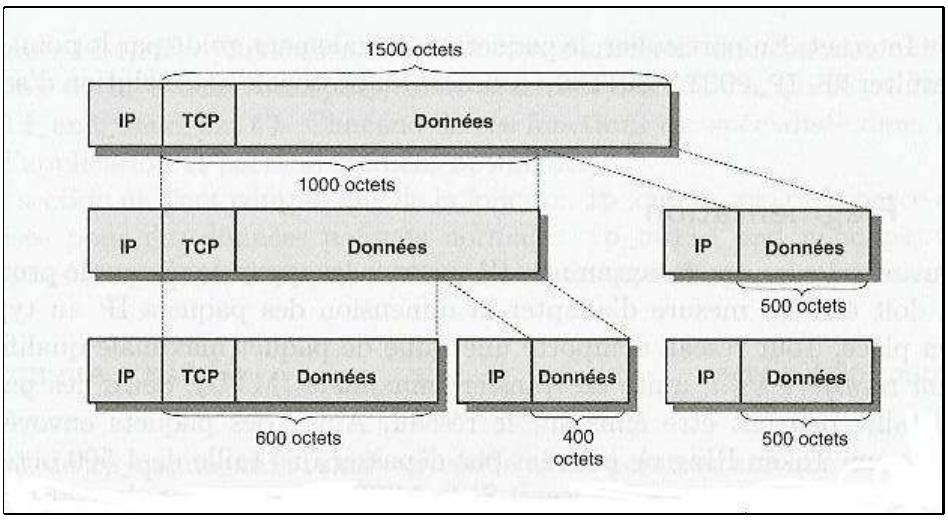


FIG. 2.9 – Fragmentation d'un paquet IP

2.6.1.5 Le routage

Notion.- Le **routage** (*routing* en anglais) est l'une des tâches de la couche réseau. Il s'agit de déterminer la route à suivre pour aller d'un point A à un point B. Ceci est l'objet des ordinateurs spécialisés appelés **routeurs** (*router* en anglais) et non des hôtes d'extrémité.

Le routage comprend deux modules :

- La **redirection** (*forwarding* en anglais) recherche, pour chaque paquet, à partir de la destination de celui-ci dans une table l'interface sur laquelle transmettre ce paquet. Cette table est appelée **table de redirection** (*forwarding table* en anglais).
- La tâche de routage proprement dit consiste à construire et à maintenir la table de redirection.

La redirection est une procédure du noyau alors que la construction de la table est une procédure utilisateur.

En gros le rôle de la table de redirection est le suivant : on lui fournit une adresse IP et elle renvoie le prochain saut (*next hop* en anglais, abrégé en `nh`), sous la forme de l'interface réseau sur laquelle envoyer le paquet (ce qui détermine le réseau local) et une adresse IP (ce qui

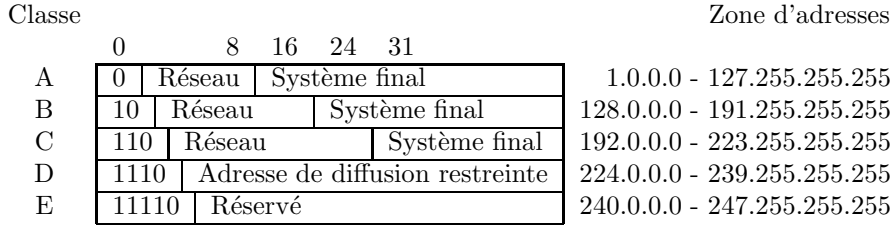
détermine l'élément de ce réseau local). La détermination de l'interface réseau est importante car un routeur possède au minimum deux interfaces réseau et en moyenne quatre à cinq.

Routage et adresse IP.- La distribution des adresses IP simplifie la conception du routage. En effet les adresses IPv4 ne sont pas de simple identifiants de nœuds réseau. Elles sont distribuées de façon à faciliter le routage. Les adresses IP sont structurées de manière hiérarchique et sont formées de deux parties distinctes : une **adresse de sous-réseau** (*network part* en anglais) et un **identifiant de système terminal** (*host part* en anglais). L'adresse de sous-réseau est identique pour tous les éléments présents dans un sous-réseau donné. Quant à l'identifiant de système terminal, il est destiné à distinguer les différents éléments au sein d'un sous-réseau donné.

Dans le cas du routage, l'identifiant du système terminal peut être ignoré jusqu'à ce que le paquet soit parvenu dans le bon sous-réseau. De ce fait, les routeurs n'ont pas besoin de connaître les identifiants des systèmes terminaux, ce qui permet ainsi de réduire de manière notable la quantité d'informations à enregistrer, grâce au découpage de l'adresse IP en deux parties.

Il existe deux façons de savoir quelle est la taille de la partie de l'adresse IP correspondant au sous-réseau et celle correspondant à l'identifiant de système terminal, comme nous allons le voir. Dans les deux cas la partie sous-réseau d'une adresse IP se trouve toujours au début, d'où le synonyme **préfixe réseau** que l'on rencontre quelquefois.

Classes d'adresses.- Le premier système ([RFC 791]) date de 1981. Il consiste à distinguer trois **classes d'adresses**, comme le montre la figure ci-dessous :



Les classes d'adresses A, B et C comprennent une partie sous-réseau respectivement de 7, 14 et 21 bits et un identifiant de système terminal de 24, 16 et 8 bits. L'appartenance à l'une de ces classes est déterminée par les premiers bits (forts) de l'adresse.

Ce schéma d'adressage a été conçu de manière à permettre à chaque réseau physique d'obtenir un identifiant sous-réseau appartenant à l'une des trois classes A, B ou C en fonction de la taille du réseau physique. L'idée était qu'il pouvait y avoir 256 sous-réseaux de classe A, tels que ARPANET et SATNET, 65 536 sous-réseaux de classe B, tels que Berkeley, le MIT, UCLA et UCL, et 16 777 216 sous-réseaux de classe C.

Cependant, il est très rapidement apparu que cette approche épuiserait très rapidement les préfixes réseau disponibles. Par ailleurs, les classes présentes se sont fréquemment montrées peu adaptées : un réseau de classe A pourrait compter près de 2^{24}, c'est-à-dire 16 777 216 systèmes terminaux, une valeur qui dépasse les besoins des plus grandes entreprises – indépendamment du fait qu'aucune technique réseau n'est en mesure de gérer un aussi grand nombre de systèmes terminaux. Les sous-réseaux de classe C – qui sont très nombreux – ne peuvent en revanche gérer que 254 systèmes terminaux, ce qui est généralement insuffisant.

Adressage sans classe.- C'est la raison pour laquelle des techniques visant à une meilleure utilisation de l'espace d'adressage disponible ont été développées. L'idée fondamentale est de permettre que la limite entre le préfixe réseau et l'identifiant du système terminal soit placé à n'importe

quelle position de bit, au lieu de ne l'autoriser qu'aux trois positions prévues par les classes d'adresses A, B et C. On utilise actuellement sur Internet le système **CIDR** (*Classless Inter-Domain Routing*) défini dans [RFC 1518] et [RFC 1519].

Les préfixes réseau peuvent avoir une longueur quelconque. L'information concernant la longueur de l'identifiant du sous-réseau ne peut plus être déterminée à partir des premiers bits de l'adresse. Cette information doit être transmise par ailleurs. On a le choix pour cela entre deux notations utilisateur :

- Dans la première notation, le nombre de bits correspondant au préfixe du sous-réseau est indiqué sous forme d'un nombre décimal, séparé de l'adresse par une barre oblique. Ainsi, l'adresse 192.168.152.0/21 désigne un sous-réseau dont le préfixe est représenté par les 21 premiers bits de l'adresse IP.
- La seconde notation consiste à ajouter à l'adresse IP un masque de bits de même longueur, le **masque réseau**, dans lequel tous les bits appartenant au préfixe du sous-réseau dans l'adresse IP ont la valeur 1. Par exemple le réseau évoqué ci-dessus est noté :

  192.168.152.0/255.255.248.0.

2.6.2 En-tête IPv4

Rappelons que l'unité d'information utilisée par IP est appelée **paquet**, ou plus précisément **paquet Internet** ou **paquet IP**. Un paquet est constitué d'un en-tête IP suivi des données. La figure suivante montre l'agencement de l'en-tête IP, telle que définie par [RFC 791] (« *Internet Protocol* » écrite par Postel en 1981) :

Version (4 bits)	Longueur (4 bits)	Type (8 bits)	Longueur du paquet (16 bits)		
Identificateur				DF	MF	Décalage
TTL		Protocole	Somme de contrôle		
Adresse émetteur					
Adresse destinataire					
Options	Remplissage				

Cette RFC a été mise à jour par la [RFC 1349] (*Type of Service in the Internet Protocol Suite*), maintenant obsolète et replacée par la [RFC 2474] (*Definition of the Differentiated Services Field (DS Field) in th IPv4 and IPv6 Headers*), mise à jour par la [RFC 3168] (*The Addition of Explicit Congestion Notification (ECN) to IP*) et la [RFC 3260] (*New Terminology and Clarifications for Diffserv*).

Commentons les champs de cet en-tête un à un :

- Le **numéro de version** est un champ de 4 bits contenant le numéro de version IP que le paquet utilise.

  La version la plus utilisée est la version 4 mais quelques systèmes effectuent des tests avec la version 6. Nous décrivons la version 4 dans la suite.

  Ce champ est *a priori* nécessaire pour que le logiciel IP récepteur sache décoder le reste de l'en-tête, qui varie à chaque nouvelle version de protocole IP. En fait nous avons vu que la distinction se fait par le champ *protocole* de l'en-tête Ethernet au niveau de la couche MAC (`0x800` pour IPv4, `0x86DD` pour IPv6).
- La **longueur de l'en-tête** (`IHL` pour *Internet Header Length*) est un champ de 4 bits qui indique la longueur totale de l'en-tête IP construit par la machine émettrice. Cette longueur permet de déterminer où commencent les données puisqu'il n'existe pas de marqueur de début des données.

Les spécifications d'IP (ainsi que de la plupart des autres protocoles de la suite TCP/IP) définissent comme unité de longueur le **mot**, un mot représentant 32 bits, soit 4 octets.

À cause du champ `Options`, nous ne pourrions pas connaître la longueur de l'en-tête sans ce champ. L'en-tête IP a une longueur maximale de quinze mots, soit 60 octets. Le plus court en-tête autorisé par IP est celui sans aucune option, soit cinq mots ou 20 octets.

- Le champ suivant, de 8 bits (un octet), spécifiait à l'origine le **type de service** (`ToS` pour *Type Of Service*), qui indiquait aux routeurs comment traiter correctement le paquet.

  Les 8 bits de ce champ étaient répartis de la façon suivante :

Priorité	Délai	Débit	Fiabilité	non utilisé

  - Les trois premiers bits, 0 à 2, indiquaient l'**ordre de priorité** (*precedence* en anglais) du paquet, avec des valeurs comprises entre 0 (normal) et 7 (contrôle réseau). Plus cette valeur est grande, plus le paquet est important, ce qui, en théorie du moins, le fera arriver plus vite à destination.
  - Les trois bits suivants sont des drapeaux d'un bit qui contrôlent le délai (*Normal/low delay* pour le bit 3), le débit (*Normal/High throughput* pour le bit 4) et la fiabilité (*Normal/High reliability* pour le bit 5) du paquet. Un bit positionné à 1 indique un délai bas, un grand débit et une haute fiabilité respectivement.
  - Les deux derniers bits (6 et 7) étaient réservés pour des utilisations futures et devaient être mis à zéro.
    Dans la mise à jour [RFC 3168], ces deux bits sont utilisés pour traiter la congestion des paquets (*ECN*).

  En pratique, la plupart des implémentations des routeurs TCP/IP ignoraient ce champ et traitaient tous les paquets avec les mêms valeurs de priorité, de délai, de débit et de fiabilité. Il prenait donc la plupart du temps la valeur 0.

  La signification de ce champ a été changée dans [RFC 2474] pour devenir `DS` (pour *Differentiation Services*). Il indique maintenant le comportement de transmission utilisé, ce qui n'intéresse que les routeurs. Les bits 0 à 5 constituent le `DSCP` (*Differentiated Services Code Point*) et les deux bits restants 6 et 7 sont encore non utilisés.

  [RFC 3168] utilise les deux bits 6 et 7 comme `ECN` (*Explicit Congestion Notification*) pour avertir les points terminaux de la congestion des routeurs, ce qui risque de faire disparaître des paquets. Ces deux bits sont appelés `ECT` (pour *ECN Capable Transport*) et `CE` (pour *Congestion Experienced*).

  [RFC 3260] change la terminologie et clarifie les utilisations du `DS`.

- La **longueur du paquet** (*Total Length*) est un champ de 16 bits, soit deux octets, qui indique la longueur totale du paquet, en-tête compris, en octets (et non en mots).

  On a donc des paquets d'au plus 65 535 octets.

- L'**identificateur de paquet originel** (*Fragment-ID*) est un champ de 16 bits, soit deux octets, qui contient un identificateur unique créé par le nœud émetteur pour chaque paquet, avant que celui-ci ne soit éventuellement fragmenté. Ce numéro est nécessaire lors du réassemblage des fragments, afin de garantir que les fragments d'un paquet ne soient pas mêlés à ceux d'un autre.

- Le champ **drapeaux** (*Flags* en anglais) est un champ de trois bits qui contrôle la gestion des paquets lorsqu'une fragmentation est requise. Le premier bit n'est pas utilisé, les deux bits restants permettent de spécifier des drapeaux appelés `DF` (pour *Don't Fragment*, ne pas fragmenter) et `MF` (pour *More Fragments*, encore des fragments) :

  - Si le drapeau `DF` vaut 1, le paquet ne peut en aucun cas être fragmenté. Si la couche de logiciel IP ne peut pas envoyer le paquet à une autre machine sans le fragmenter et que ce bit vaut 1, le paquet est détruit et un message d'erreur doit être envoyé au composant émetteur *via* ICMP.
  - Si le drapeau `MF` vaut 1, c'est que le paquet en cours est suivi d'autres paquets (parfois appelés **sous-paquets**) qui devront être réassemblés pour former le paquet originel. Le dernier fragment d'un paquet envoyé doit avoir son drapeau `MF` égal à 0, de manière à ce que le composant récepteur sache qu'il doit cesser d'attendre de nouveaux sous-paquets.

    Comme l'ordre d'arrivée des fragments ne correspond pas forcément à leur ordre d'envoi, le drapeau `MF` est utilisé en conjonction avec le champ suivant pour indiquer à la machine réceptrice à quoi ressemble le paquet originel.
- Si le drapeau `MF` est égal à 1, ce qui indique un paquet fragmenté, le champ `décalage de fragment` (*Fragment Offset* en anglais) de 13 bits contient la position, l'unité étant deux mots ou huit octets ou 64 bits, au sein du paquet complet, du début des données contenues dans le paquet en cours. Ceci permet à IP de réassembler les paquets fragmentés dans leur ordre correct.

  Puisque ce champ comporte 13 bits et que les décalages sont calculés en unités de 8 octets, on retrouve bien la longueur de paquet maximale de 65 535 octets. Puisque ce champ a une taille de 13 bits, seulement 8 192 fragments au maximum peuvent appartenir à un paquet IP originel. Tous les fragments, sauf le dernier, doivent avoir une longueur multiple de 8 octets, l'unité élémentaire de fragmentation.
- Le champ **durée de vie** (`TTL` pour *Time To Live*), de taille un octet, est un compteur qui indique le nombre maximal de systèmes intermédiaires (routeurs ou plutôt sauts, *hop* en anglais) que le paquet peut traverser avant d'être détruit par le nœud en cours avec, dans ce cas, un message ICMP envoyé à la machine émettrice (*Time exceeded*).

  Cette durée de vie permet d'éviter que des paquets IP circulent sans fin sur le réseau et ne saturent celui-ci. Elle est fixée par le nœud émetteur lors de l'assemblage du paquet.
- Le **protocole de transport** est un champ d'un octet qui contient le numéro d'identification du protocole de transport duquel provient ou sera confié le paquet.

  Les numéros sont définis par le IANA (*Internet Assigned Numbers Authority*, où ils peuvent être trouvés sur leur site Web). Il existe actuellement environ 50 protocoles affectés d'un numéro de transport pour la partie serveur, codifiés dans [RFC 1700]. Les quatre protocoles les plus importants sont ICMP, qui porte le numéro 1, TCP le numéro 6, UDP le numéro 17 et IGMP le numéro 2.
- La **somme de contrôle** (*header checksum*) ne porte que sur l'en-tête de protocole (et non sur les données) pour accélérer les traitements.

  La somme de contrôle est égale à l'opposé en complément à un de la somme en complément à un sur 16 bits de tous les mots de 16 bits de l'en-tête IP sauf le champ somme de contrôle.

  Pour vérifier la somme de contrôle, il suffit donc d'ajouter cette somme à la valeur du champ *somme de contrôle*, ce qui devrait donner 0, soit `FFh`.

  Comme le champ `TTL` est décrémenté lors du passage de chaque routeur, la somme de contrôle doit être recalculée par chaque routeur par lequel passe le paquet.
- Les champs **adresse de l'émetteur** (*Source address* et **adresse de destination** (*Destination address*) contiennent les adresses IP à 32 bits des composants émetteur et récepteur.
- Le champ **Options** est optionnel et, dans le cas où il existe, est composé de plusieurs codes de longueurs variables. Si plus d'une option est utilisée dans un paquet, les options apparaissent les unes à la suite des autres dans l'en-tête IP.

Toutes les options sont contrôlées par un octet, généralement divisé en trois champs : un *drapeau de copie* sur un bit, une *classe d'options* sur deux bits et un *numéro d'option* sur cinq bits.

- Le **drapeau de copie** est utilisé pour indiquer la manière dont l'option est traitée lorsqu'une fragmentation est nécessaire dans une passerelle. Lorsque ce bit vaut 0, l'option doit être copiée sur le premier fragment mais pas sur les suivants. Si le bit vaut 1, elle doit être copiée sur tous les fragments.
- Les deux bits de la **classe d'options** permet quatre classes. Pour le moment il n'existe que deux classes. La classe 0 indique que l'option s'applique au contrôle du paquet ou du réseau. La classe 2 indique que l'option a trait au débogage ou à l'administration.
- Les valeurs actuellement prises en charge pour les classes et numéros d'options figurent dans le tableau suivant :

Classe	Numéro	Description
0	0	Marque la fin de la liste d'options
0	1	Aucune option (utilisée pour le remplissage de caractères)
0	2	Options de sécurité (utilisation militaire seulement)
0	3	Routage tolérant
0	7	Active l'enregistrement du routage (ajoute des champs)
0	9	Routage strict
2	4	Marquage de la date activé (ajoute des champs)

Les options les plus intéressantes sont celles qui activent l'enregistrement du routage (*record route*) et de l'heure (*internet timestamps*). Elles permettent de tracer le passage d'un paquet au travers de l'interréseau, ce qui peut être utile pour émettre des diagnostics.

Le **routage tolérant** (*loose routing* en anglais) fournit une série d'adresses IP par lesquelles le paquet doit passer, mais il permet de prendre n'importe quelle route pour parvenir à chacune de ces adresses (correspondant à des passerelles en général).

Le **routage strict** n'autorise aucune déviation de la route spécifiée. Si cette route ne peut pas être suivie, le paquet est détruit et un message d'erreur est envoyé *via* ICMP. Le routage strict est fréquemment utilisé pour tester les routes mais rarement pour transmettre des paquets utilisateur.

- Le contenu de la **zone de remplissage** (*padding*) dépend des options sélectionnées. On utilise le remplissage pour s'assurer que la longueur de l'en-tête est bien un multiple de mots, c'est-à-dire de 32 bits.

2.6.3 Analyse d'un en-tête IP par Wireshark

Si nous revenons à notre capture, nous nous apercevons que la troisième trame comprend un en-tête IP, plus exactement un en-tête Ethernet, un en-tête IP, un en-tête UDP et un message DNS (voir figure 2.10). Rappelons que c'est l'en-tête Ethernet qui nous indique, grâce à son champ `type`, que l'en-tête Ethernet est suivie d'un en-tête IP. La longueur totale de celui-ci sera indiquée dans l'en-tête lui-même.

- Les quatre premiers bits (le premier demi-octet) rappellent qu'il s'agit de la version quatre et donc d'un en-tête IPv4.
- Les quatre bits suivants (le demi-octet suivant) indiquent que la longueur de l'en-tête est de cinq mots, soit vingt octets. Il s'agit ici de la longueur minimum d'un en-tête sans option.
- L'octet suivant, interprété comme *Differentiated Services Field* par `Wireshark`, est nul, ce qui sera le plus souvent le cas.

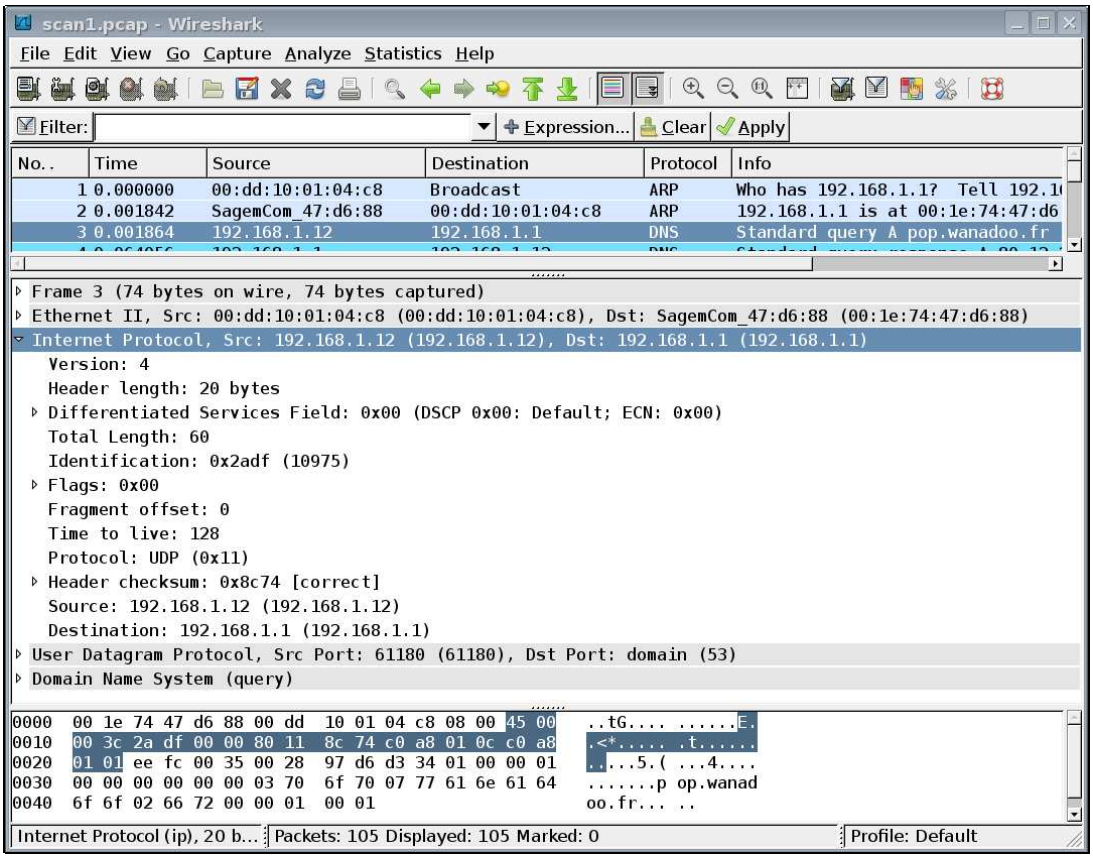


FIG. 2.10 – Analyse d'un en-tête IP

- Les deux octets suivants précisent la longueur du paquet, ici `38h`, soit 60 octets. Avec les 14 octets de l'en-tête Ethernet, on retrouve bien les 74 octets de la trame.
- L'identificateur de paquet de deux octets (déterminé de façon presque aléatoire, rappelons-le) est ici `2ADFh`, soit 10 975.
- Les trois bits suivants `000b` constituent les drapeaux. Le premier bit est un bit réservé pour une utilisation future qui doit toujours être nul ; c'est bien le cas. Le bit suivant, `DF`, positionné à 0, indique que l'on peut fragmenter le paquet. Le bit suivant, `MF`, positionné à 0, indique qu'il n'y a pas de paquet suivant portant le même identificateur.
- Les treize bits suivants, tous égaux à 0, spécifient la position au sein du message complet. Nous en sommes donc au début et, puisque `MF` est égal à zéro, le paquet contient le message complet.
- L'octet suivant, de valeur `80h` ou 128, indique la durée de vie (`TTL`) du paquet.
- L'octet suivant spécifie le protocole de transport. On a ici `11h`, soit 17. Le protocole de transport est donc UDP.
- Les deux octets suivants constituent la somme de contrôle. On a ici `8C74h`. `Wireshark` indique qu'elle est correcte. Je vous laisse le vérifier.
- Les quatre octets suivants donnent l'adresse IP de l'émetteur. On a ici `C0A8010Ch`, soit 192.168.1.12. On retrouve bien l'adresse IP de mon ordinateur.

– Les quatre octets suivants donnent l'adresse IP du destinataire. On a ici `C0A80101h`, soit 192.168.1.1. On retouve bien l'adresse IP de mon routeur.

2.7 Le protocole de couche de transport UDP

Le protocole UDP (*User Datagram Protocol*), décrit dans [RFC 768], a le minimum de ce que peut avoir un protocole de transport.

2.7.1 Multiplexage au niveau transport

Le protocole UDP possède la fonctionnalité essentielle des protocoles de la couche de transport, celle du multiplexage/démultiplexage : il transmet les datagrammes aux différentes applications d'un système terminal donné, grâce à la spécification des numéros de port.

2.7.1.1 Notion de port

Un ordinateur peut communiquer en même temps avec plusieurs autres (en utilisant des processus différents et du pseudo-parallélisme), par exemple recevoir une page web et du courrier tout en envoyant un fichier par `ftp`. Les paquets qui arrivent, au niveau de la couche réseau donc, sont triés :

– ceux qui ne correspondent pas à l'adresse IP sont rejetés, comme nous l'avons vu à propos de IP ;
– les messages des autres, donc après avoir enlevé les en-têtes de la couche réseau et de la couche transport, sont envoyés vers telle ou telle (instance d') application.

Comment déterminer l'instance d'application vers laquelle on doit envoyer le message ? L'en-tête de la couche de transport (que ce soit TCP ou UDP) contient un champ indiquant celle-ci. Il s'agit d'une entier de 16 bits, donc compris entre 0 et 65 535, appelé **port**.

2.7.1.2 Modèle client/serveur

La manipulation des datagrammes se fait toujours dans le cadre du modèle client/serveur :

– le **client** est celui qui demande quelque chose ; ce quelque chose est appelé une **requête** ;
– le **serveur** est celui qui attend que l'on demanque quelque chose. On appelle **réponse** ce qu'il transmet. Il faut donc prévoir pour le serveur un procédé pour cette attente, ne serait-ce qu'une simple boucle.

2.7.1.3 Attribution des numéros de port aux serveurs

Si un client situé sur un ordinateur A veut envoyer une requête à un serveur situé sur un ordinateur B, comment peut-il connaître son port ? Chaque **service** peut-il choisir son port de façon arbitraire ? L'attribution des ports est-elle centralisée ? Comment une application peut-elle obtenir le numéro de port d'une autre application ?

Un organisme, appelé **IANA** (pour *Internet Assigned Number Authority*), attribue des ports fixes aux serveurs des applications connues. On parle des ports bien connus (*well bnown ports*), auxquels chaque application peut se référer lors d'une requête. Au départ, les valeurs situées entre 0 et 256 étaient réservées à l'attribution des ports normalisés. Mais, en 1994, ce quota a été étendu aux valeurs contenues entre 0 et 1 023, en raison de la demande toujours croissante de ports réservés. La liste des ports normalisés peut être consultée dans [RFC 1700] de 1994. La figure 5.1 montre quelques-uns des numéros de ports les plus utilisés par les serveurs.

Application	Port
daytime (heure du jour)	13
ftp	21
telnet	23
smtp	25
time (heure)	37
DNS	53
finger	79
web	80
pop3	110
nntp (news)	119
snmp	161
irc (Internet Relay Chat)	194

FIG. 2.11 – Numéros de port bien connus des serveurs

2.7.1.4 Attribution dynamique pour les clients

Si les services bien connus ont des numéros fixés par IANA, les clients par contre se voient toujours attribuer un numéro de port de façon dynamique par le sous-système réseau du système d'exploitation au moment où ils ont besoin (supérieurs évidemment à 1 024).

2.7.2 Utilisation d'UDP

Le protocole UDP est utilisé, d'une part, pour les applications orientées **transaction** telles que DNS, dans lesquelles seules une requête et une réponse associée doivent être transmise, de telle sorte qu'il est inutile d'établir une connexion pour cela. D'autre part, le protocole UDP est également utilisé dans les contextes où la rapidité de la transmission des données prédomine sur la fiabilité. C'est ainsi que dans le cas des flux audio, qui sont transmis sous forme de petits messages, il n'est pas excessivement gênant que quelques messages isolés se perdent ; par contre, un contrôle automatique de flux et des erreurs (assorti de la retransmission des messages perdus) s'avérerait souvent très pénalisant pour la régularité du flux.

2.7.3 L'en-tête UDP

L'en-tête UDP est défini dans [RFC 768]. Sa structure, très simple, n'occupe que huit octets, comme le montre la figure suivante :

Port source (16 bits)	Port Destination (16 bits)
Longueur (16 bits)	Somme de contrôle (16 bits)
Données	Remplissage

- Le **port source** (*source port*) est un champ facultatif contenant le numéro de port de l'expéditeur, compris entre 1 et 65 535. Si aucun numéro de port n'est spécifié, le champ est mis à 0. Ce champ est par contre nécessaire au destinataire s'il doit renvoyer des données.
- Le **port de destination** (*destination port*) est le numéro de port sur la machine de destination.
- La **longueur** (*length*) est la taille du datagramme, exprimée en nombre d'octets, comprenant en-tête et données. Sa valeur minimum est 8 (taille de l'en-tête UDP) alors que le

datagramme UDP le plus long peut transporter 65 535 - 8 = 65 527 octets de données utiles.

- la **somme de contrôle** (*checksum*) est un champ facultatif sur lequel nous allons revenir ci-dessous. De nombreuses applications n'y ont pas recours. S'il n'est pas utilisé, sa valeur doit être zéro.
- Les **données** sont de longueur variable.
- Le **remplissage** assure que le datagramme a une longueur multiple de 16 bits.

2.7.4 Sécurisation optionnelle par somme de contrôle

UDP permet une connexion non fiable dans le sens où il n'existe aucun mécanisme permettant de reconnaître et de traiter les datagrammes perdus ou dupliqués. Les datagrammes peuvent cependant être protégés, de manière optionnelle, des erreurs à l'aide d'une somme de contrôle. Contrairement au protocole IP, cette somme de contrôle ne se rapporte pas uniquement à l'en-tête du datagramme mais également aux données utiles de celui-ci.

Si le datagramme expédié comporte une somme de contrôle, celle-ci est le complément à 1 de 16 bits du complément à 1 de la somme du datagramme (en-tête et données utiles) ainsi que d'un **pseudo en-tête**, dont la structure est montrée à la figure 2-12 ([TJ–97], p.117).

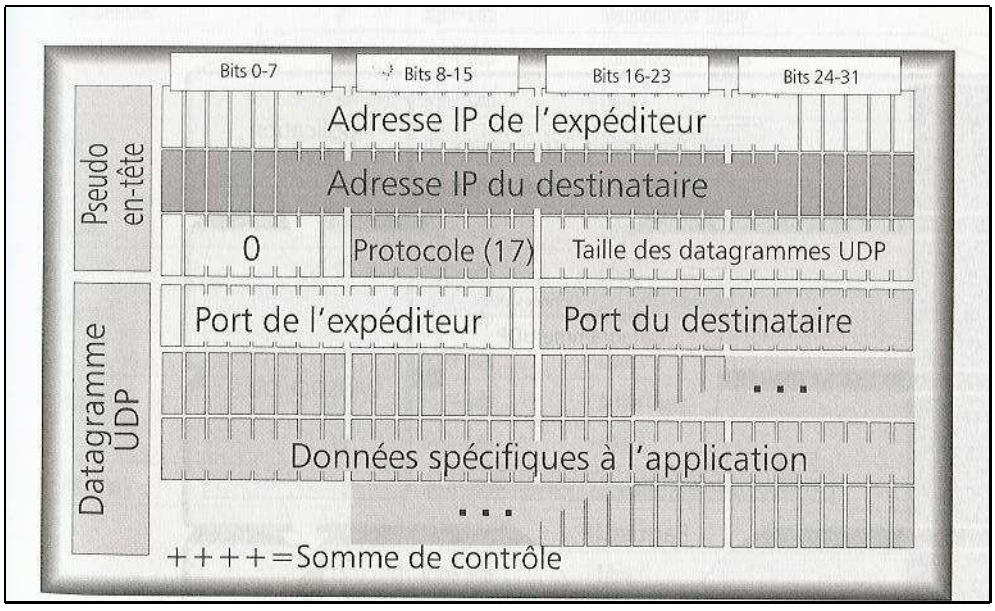


FIG. 2.12 – Pseudo en-tête UDP

Le pseudo en-tête contient les adresses IP de l'expéditeur et du destinataire, la taille du datagramme UDP et le code de protocole IP pour UDP (à savoir 17). Toutes ces informations proviennent de la couche réseau.

Lorsque le calcul de la somme de contrôle donne la valeur nulle, on indique `0xFFFF`, soit -1, à la place, qui équivaut également à 0 sans toutefois coïncider avec la valeur constante 0 signalant l'absence de somme de contrôle.

La méthode de calcul de la somme de contrôle s'implémente très efficacement pour tous les types de processeurs. Elle est décrite dans les recommandations [RFC 1071], [RFC 1141] et

[RFC 1624].

2.7.5 Analyse d'un en-tête UDP

Continuons à analyser la trame DNS (voir figure 2.13). L'en-tête IP nous a indiqué, grâce à son champ `protocole de transport` que la suite est un en-tête UDP. La longueur de celui-ci est toujours de huit octets.

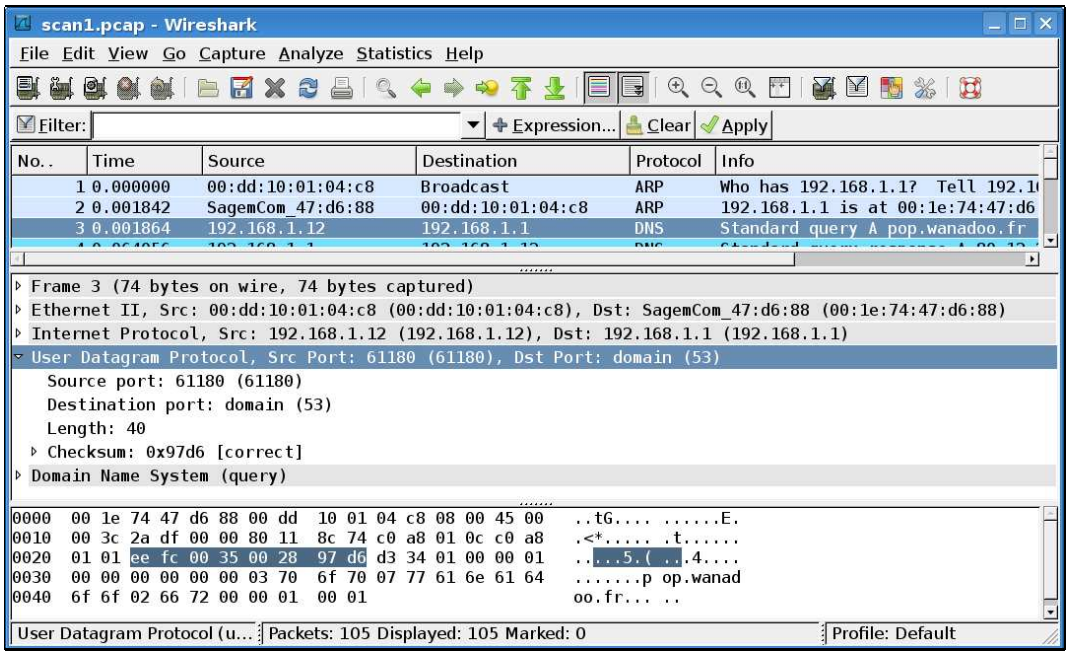


FIG. 2.13 – Analyse d'un en-tête UDP

- Les deux premiers octets de l'en-tête UDP spécifient le numéro de port source. On a ici `EEFCh`, soit 61 180. Rappelons que ce numéro de port est déterminé (presqu'au hasard) par l'implémentation de la couche UDP du sous-système réseau du système d'exploitation de façon à ce que deux instances d'applications différentes n'aient pas le même numéro de port.
- Les deux octets suivants spécifient le numéro de port destination. On a ici `0035h`, soit 53. Il s'agit d'un numéro de port bien connu, celui attribué pour le serveur DNS. Ceci nous indique que le message concerne un message DNS, et même une requête sinon le port serait supérieur à 1 024.
- Les deux octets suivants indiquent la taille du datagramme. Elle est ici de `00 28h`, soit 40 octets. Avec les vingt octets de l'en-tête IP et les quatorze octets de l'en-tête Ethernet, on retrouve bien les 74 octets de la trame.
- Les deux derniers octets constituent la somme de contrôle. On a ici `97 D6h`. `Wireshark` indique qu'elle est correcte ; nous vous laissons le soin de le vérifier.

2.8 Le protocole de couche de transport TCP

2.8.1 Fonctionnalités

Le protocole de transport TCP permet évidemment, comme UDP, le multiplexage/démultiplexage mais il essaie d'aller au-delà en s'intéressant à la fiabilité (transfert des données à nouveau si celles-ci ont été corrompues en cours de route), contrôle du flux (réduction de celui-ci en cas de congestion, augmentation progressive sinon) et connexion.

2.8.1.1 Transfert continu d'octets

TCP transfère un flux continu d'octets dans chaque direction. Il laisse le soin aux applications de grouper les octets dans un *segment de message* devant être envoyé/reçu. Les segments de message peuvent être de taille arbitraire. Afin d'être plus efficace dans la gestion des messages, les connexions TCP négocient généralement une taille maximum de segment lors de la connexion.

TCP numérote chaque octet qu'il envoie. Les octets sont remis à l'application à l'autre extrémité dans leur ordre d'envoi. On appelle cela le **séquencement des octets**. Bien entendu, TCP n'envoie pas les données octet par octet, mais par groupe d'octets ; cependant ce sont bien les octets qui sont numérotés.

Comme TCP envoie les données sous forme de flux d'octets, on ne trouve pas de marqueur de fin de message dans le flux. Pour s'assurer que toutes les données soumises au module TCP ont bien été transmises, une fonction `push()` est nécessaire : elle fait envoyer par TCP toute donnée reçue jusqu'ici en provenance d'une application. Sinon TCP attend que la taille maximum négociée soit atteinte avant d'envoyer un datagramme.

2.8.1.2 Fiabilité

La fiabilité totale est impossible, cependant TCP possède un mécanisme qui essaie de remettre de façon fiable le flux d'octets. Il essaie de restaurer les données endommagées, perdues, dupliquées ou remises en mauvais état.

Il utilise le mécanisme **PAR** (*Positive Acknowledgment Retransmission*, retransmission d'acquittement positif). TCP implémente ce mécanisme en affectant un **numéro de séquence** à chaque octet transmis et en requérant un acquittement positif (**ACK**) du module TCP récepteur. Si cet acquittement ne parvient pas avant expiration d'un certain délai, l'implémentation TCP de l'expéditeur doit retransmettre le datagramme concerné.

Au niveau du module TCP récepteur les numéros de séquence servent à ordonner les segments et à éliminer les doublons.

Pour détecter les données erronées, on se sert d'un champ somme de contrôle. Les segments de données dont la somme de contrôle ne correspond pas sont purement et simplement écartés.

2.8.1.3 Contrôle de flux

Les machines qui émettent et reçoivent les segments de données TCP ne le font pas toutes au même rythme en raison de la diversité des unités centrales et de la bande passante. Il peut donc arriver que l'émetteur envoie ses données beaucoup plus rapidement que le récepteur ne peut les gérer. C'est pourquoi TCP implémente un mécanisme de contrôle des flux de données.

TCP se sert de la technique dite de la **fenêtre glissante** (voir figure 2.14). Comme nous l'avons déjà dit, le flux de données est numéroté octet par octet. Lors de l'ouverture de la connexion, le module TCP du récepteur envoie à l'émetteur un acquittement qui fixe un intervalle pour les numéros de séquence à partir du dernier datagramme correctement reçu. Cet intervalle est appelé **fenêtre** : il indique à l'émetteur le nombre d'octets qu'il peut transmettre sans avoir

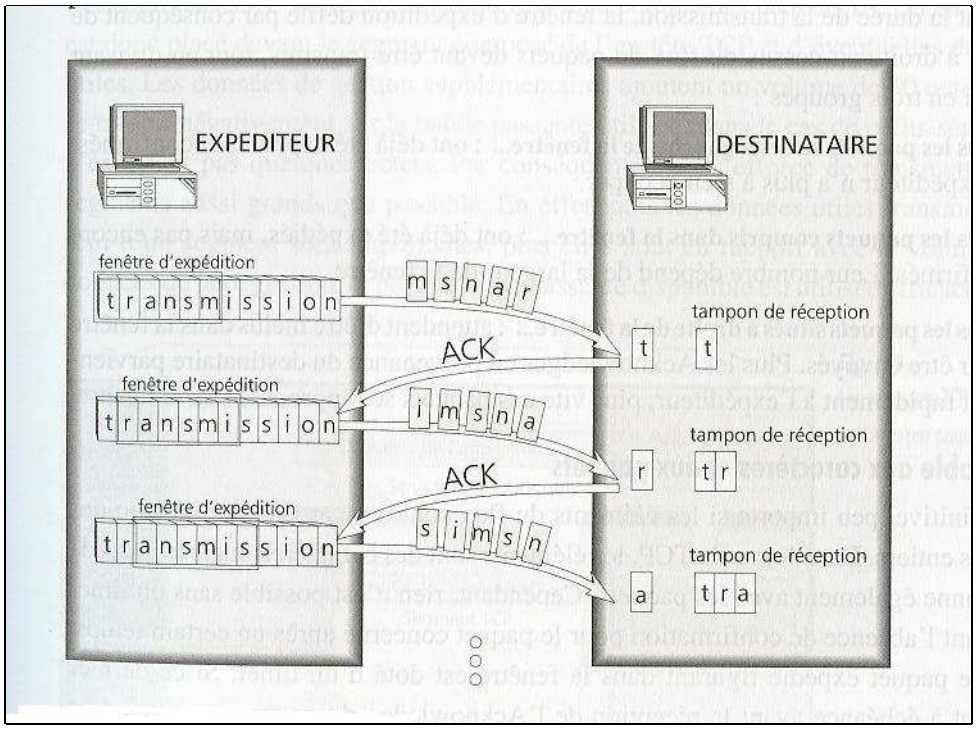


FIG. 2.14 – Fenêtre glissante

besoin d'une autorisation. Voici alors comment fonctionne ensuite le mécanisme de contrôle de flux :

- Les octets qui précèdent la limite inférieure de la fenêtre ont été envoyés et acquittés.
- Les octets se trouvant dans l'intervalle défini par la fenêtre peuvent être envoyés. S'ils ont déjà été envoyés, leur acquittement n'est pas encore parvenu.
- Les octets qui dépassent la limite supérieure de la fenêtre n'ont pas encore été envoyés et ne pourront l'être que lorsqu'ils passeront dans l'intervalle de la fenêtre.

La limite inférieure de la fenêtre est l'octet non acquitté dont le numéro est le plus petit. La fenêtre progresse, on dit de gauche à droite, dès qu'elle reçoit un acquittement pour les données expédiées. Le segment qui contient l'acquittement contient aussi des informations concernant l'amplitude de la fenêtre que l'émetteur peut se permettre.

L'amplitude de la fenêtre dépend de la quantité d'espace libre encore disponible dans le tampon de données du récepteur. Lorsque le récepteur est trop sollicité, il envoie une taille de fenêtre réduite. Dans les cas extrêmes, cette taille peut se réduire à 1. Lorsque le module TCP du récepteur renvoie une taille de fenêtre égale à zéro, cela indique à l'expéditeur que ses tampons sont pleins et qu'il ne faut plus lui envoyer aucune donnée.

2.8.1.4 Connexion

Avant de pouvoir envoyer des données à l'aide de TCP, les processus doivent d'abord établir une connexion. Une connexion s'établit entre le numéro de port de l'émetteur et celui du récepteur. Une **extrémité** (*peer* en anglais) est définie par un couple : l'adresse IP et le numéro de port. On établit une **connexion** entre deux extrémités, autrement dit avec un quadruplet :

(adresse IP 1, numéro de port 1, adresse IP 2, numéro de port 2)

2.8.2 L'en-tête TCP

2.8.2.1 Structure de l'en-tête

La figure suivante montre l'agencement de l'en-tête TCP telle que définie par la [RFC 793] (« *Transmission Control Protocol* » écrite par Postel en 1981) :

<table>
<tr><td colspan="8">Port source (16 bits)</td><td>Destination (16 bits)</td></tr>
<tr><td colspan="9">Numéro de séquence (32 bits)</td></tr>
<tr><td colspan="9">Numéro d'acquittement (32 bits)</td></tr>
<tr><td>Offset
données
(4 bits)</td><td>Réservé
(6 bits)</td><td>U
R
G</td><td>A
C
K</td><td>P
S
H</td><td>R
S
T</td><td>S
Y
N</td><td>F
I
N</td><td>Fenêtre (16 bits)</td></tr>
<tr><td colspan="8">Somme de contrôle (16 bits)</td><td>Pointeur Urgent (16 bits)</td></tr>
<tr><td colspan="7">Options</td><td colspan="2">Octets de remplissage</td></tr>
</table>

- Le `port source` (*Source Port*) est un champ de 16 bits qui identifie l'utilisateur TCP local, comme dans le cas UDP.
- Le `port de destination` (*Destination Port*) est un champ de 16 bits qui identifie l'utilisateur TCP de la machine distante.
- Le `numéro de séquence` (*Sequence Number*) est un champ de 32 bits qui indique la position du bloc en cours dans l'ensemble du message.

  Le `numéro de séquence` indique au destinataire le début des données contenues dans le segment TCP. Pour le premier segment TCP, on indique 0 mais pas toujours. Si l'on transmet 300 octets dans ce segment, le segment suivant débutera à l'octet 301 du flux de caractères et son champ `numéro de séquence` indiquera 301.

  Si le drapeau SYN (voir ci-dessous) vaut 1, ce champ définit le numéro de séquence initial (**ISN**) correspondant à la session en cours.
- Le `numéro d'acquittement` (*Acknowledgement Number*) est un champ de 32 bits. L'expéditeur utilise cette information pour préciser à son correspondant le numéro de séquence qu'il attend dans le segment TCP suivant. Ceci donne une idée des confirmations (acquittements) que l'expéditeur a déjà recues.

  Le destinataire doit tenir compte de ce champ lorsqu'il reçoit un en-tête TCP contenant le drapeau `ACK` (voir ci-dessous). Celui-ci indique que le segment contient, outre les données éventuellement présentes, une confirmation (acquittement) provenant du correspondant de la réception de données. Le champ `numéro d'acquittement` contient le numéro du prochain octet attendu en provenance de l'interlocuteur. Pendant qu'il sert au transport des données du point A vers le point B, le segment signale à B les données déjà reçues par A. Bien que circulant de A à B au sein du segment, cette partie du segment concerne donc la communication entre B et A. C'est assez astucieux car une confirmation peut ainsi être transmise sans occasionner pour autant l'envoi d'un segment TCP comprenant uniquement un en-tête TCP, comme le montre la figure 2.15 ([TJ–97], p.146). En anglais ce procédé est appelé *piggybacking*.

  Un acquittement peut servir à acquitter plusieurs segments TCP. L'acquittement signifie que le module TCP a reçu les données, mais il ne garantit pas que les données sont parvenues à l'application.

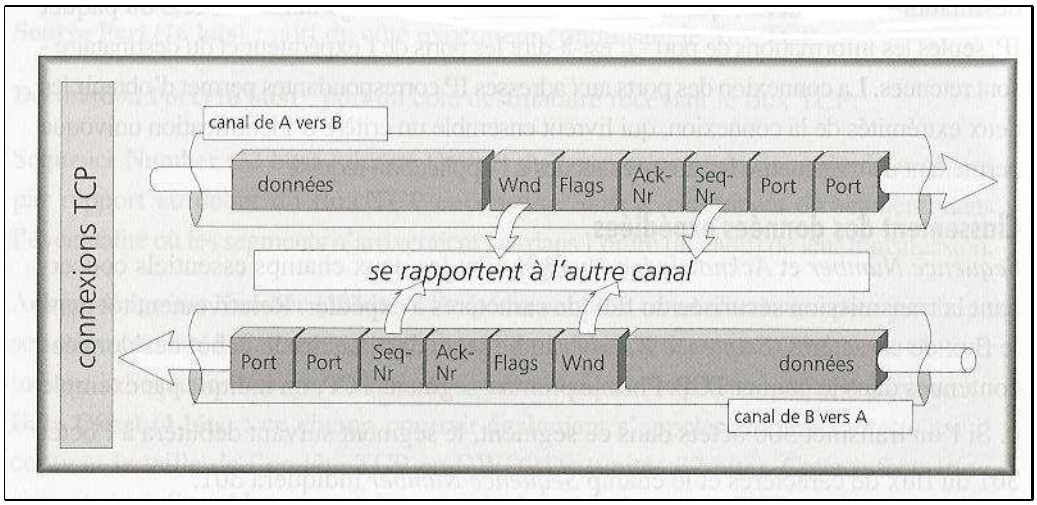


FIG. 2.15 – Piggybacking

- Le `décalage de données` (*Data Offset*) est un champ de quatre bits qui indique à quel mot commencent les données, qui suivent immédiatement l'en-tête. Il pourrait donc aussi s'appeler longueur de l'en-tête (*Header Length*) car il contient le nombre de mots de 32 bits qui se trouvent dans l'en-tête TCP. Cette information est nécessaire parce que le champ options a une longueur variable. Lorsque le champ options est vide, la valeur de décalage des données est cinq (soit 20 octets).
- `Réservé` est un champ de 6 bits réservé pour des utilisations ultérieures. Les 6 bits devaient être positionnés à 0, ce qui pose quelques problèmes maintenant que deux de ces bits ont une signification.

  L'avant dernier de ces bits est devenu le drapeau `CWR` (pour *Congestion Window Reduced*) avec la [RFC 3268]. Il est utilisé par l'expéditeur pour informer le destinataire, s'il est activé (c'est-à-dire de valeur 1), que la fenêtre a été réduite. On doit alors envoyer moins de données par unité de temps.

  Le dernier de ces bits est devenu le drapeau `ECE` (pour *ECN Echo*) avec la [RFC 3268]. Il est utilisé par le destinataire pour confirmer à l'expéditeur, s'il est activé, qu'il a reçu un drapeau `CWR`.
- La drapeau `Urg` (pour *URGent*), s'il est activé (valeur égale à 1), indique que le segment contient des données urgentes et donc qu'il faut prendre en compte la valeur du champ de pointeur urgent.
- Le drapeau `Ack` (pour *ACKnowledgment*), s'il est activé, indique que le segment contient une confirmation et donc qu'il faut prendre en compte la valeur du champ `Acquittement`.
- Le drapeau `Psh` (pour *PuSH*), s'il est activé, indique qu'il faut faire suivre immédiatement les données reçues (par exemple en faisant appel à la fonction `push()` sous UNIX).
- Le drapeau `Rst` (pour *ReSeT*), s'il est activé, indique que la connexion doit être réinitialisée pour cause d'erreurs irrécupérables. À la réception du drapeau `Rst`, le destinataire doit immédiatement terminer la connexion.
- Le drapeau `Syn` (pour *SYNchronize sequence numbers*), s'il est activé, indique que les numéros de séquence doivent être synchronisés. Ce drapeau est utilisé lors de l'établissement d'une connexion, par le segment qui ouvre la connexion et par le segment SYN/ACK qui lui répond. Il ne doit pas être utilisé par d'autres segments.

- Le drapeau `Fin`, s'il est activé, indique que l'émetteur n'a plus de données à envoyer. C'est l'équivalent d'un marqueur de fin de transmission. Lorsque l'autre extrémité reçoit un segment avec ce drapeau activé, elle doit répondre par FIN/ACK.
  Une fois qu'une extrémité a envoyé un segment contenant ce drapeau, elle ne doit plus envoyer aucun segment. Cependant, l'autre extrémité peut continuer à envoyer des données jusqu'à ce qu'elle ait terminé et envoyer alors un segment FIN et attendre le FIN/ACK final. Après cela, la connexion est close.
- La `longueur de la fenêtre` (*window*) est un champ de 16 bits indiquant le nombre d'octets données que la machine réceptrice est prête à recevoir, compte tenu de l'octet affiché dans le champ `numéro d'acquittement`.
- La `somme de contrôle` (*checksum*) est calculée en prenant le complément à un en 16 bits du complément à un de la somme des mots de 16 bits dans l'en-tête et le texte réunis. Ce champ est le résultat du calcul de la somme de contrôle basée sur la taille du segment entier, comprenant un pseudo en-tête de 96 bits préfixé à l'en-tête TCP lors du calcul. Le **pseudo en-tête** de 96 bits contient l'adresse source, l'adresse de destination, l'identificateur de protocole et la longueur du segment. Ce sont les paramètres qui sont passés à IP lorsqu'une instruction d'envoi est passée, et ceux qui sont lus par IP lorsqu'une livraison est tentée.
- On n'a à tenir compte du champ `pointeur urgent` (*Urgent Pointer*) que si le drapeau `Urg` est activé. Il indique la portion des données du message qui est urgente, en en spécifiant le décalage à partir du numéro de séquence dans l'en-tête.
  Aucune action spécifique n'est entreprise par TCP en ce qui concerne les données urgentes. C'est l'application qui s'en charge éventuellement. Il peut servir pour mettre en place les fonctions `send()` et `recv()` d'Unix.
- `Options` : chaque option consiste en un type d'option (1 octet), suivi éventuellement du nombre d'octets dans l'option (1 octet) puis des données (le nombre d'octets précédent moins deux). Dans la RFC 793, seules trois types sont définis pour TCP :
  - 0 : fin de la liste d'option (*End-of-Options*), option constituée d'un seul octet. Cette option n'est pas nécessaire lorsque la fin des options coïncide avec la fin de l'en-tête TCP.
  - 1 : pas d'opération (*No-Operation*), option constituée d'un seul octet. Ce type sert à aligner le commencement de l'option suivante sur le début d'un mot, mais les émetteurs TCP ne prennent pas toujours cette précaution.
  - 2 : taille de segment maximale (`MMS` pour *Maximum Segment Size*) de longueur 4. Ce champ sert à spécifier la taille maximale de tampon qu'une implémentation TCP réceptrice peut accepter. Comme TCP utilise des zones de données à longueur variable, il peut arriver qu'une machine émettrice crée un segment plus long que ce que peut gérer le logiciel récepteur.
- Le champ `remplissage` (*padding*) est rempli pour être sûr que la longueur de l'en-tête est un multiple de 32 bits.

2.8.2.2 Procédure de négociation d'ouverture d'une session TCP en trois étapes

On ouvre une session TCP à l'aide d'une procédure de négociation en trois temps (*Three-way handshake* en anglais) :

- Le client envoie au serveur un segment TCP d'ouverture de session avec les valeurs suivantes pour les drapeaux : SYN = 1 et ACK = 0. Ces valeurs caractérisent cette première étape. Le numéro de séquence d'ouverture est SEQ = X, appelé **ISS** pour *Initial Sender Sequence*.

On a éventuellement Z octets de données mais en général Z = 0. Le numéro d'acquittement est nul : #ACK = 0.

– Le serveur, s'il accepte d'établir une connexion avec le client, acquitte ce segment d'ouverture par un segment TCP dont les valeurs des drapeaux sont les suivantes : SYN = 1 et ACK = 1. Ces valeurs caractérisent cette seconde étape. Le numéro de séquence d'ouverture de la part du serveur est SEQ = Y, appelé **IRS** pour *Initial Receiver Sequence*. Le numéro d'acquittement doit être #ACK = X + 1 + Z.

  Si le serveur n'accepte pas d'établir une connexion avec le client, il envoie un segment au client avec les drapeaux SYN et RST positionnés.

– Si le serveur a accepté d'établir une connexion alors, dans l'étape trois, le client acquitte le numéro de séquence que lui a envoyé le serveur avec les valeurs suivantes des drapeaux : SYN = 0 et ACK = 1. On doit avoir SEQ = X + 1 + Z et #ACK = Y + 1.

  À ce moment la connexion a été établie, les données peuvent être envoyées.

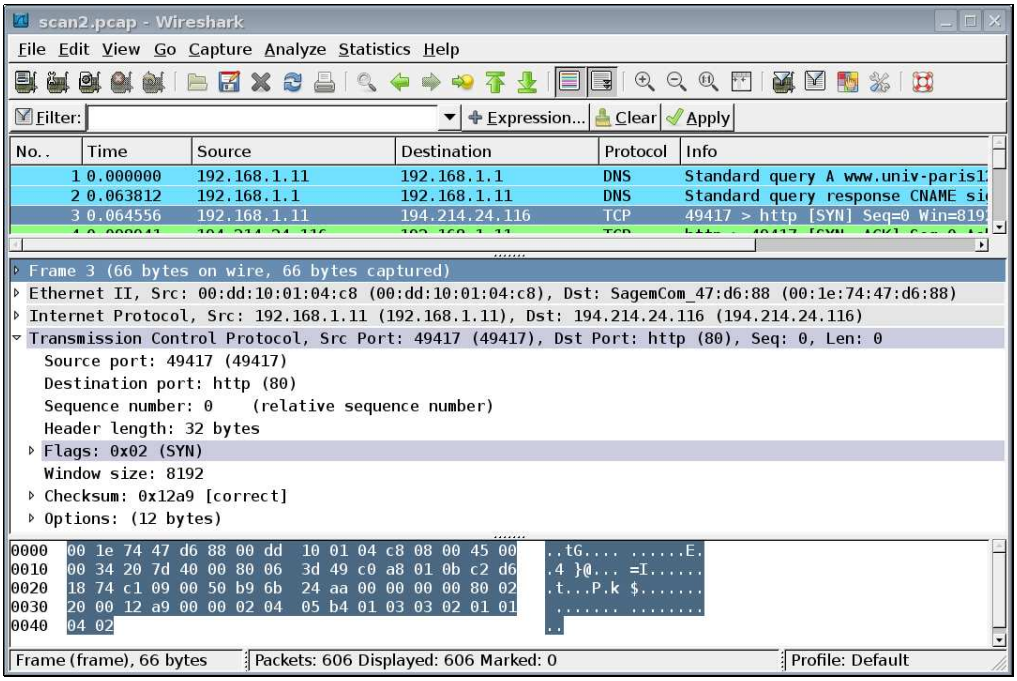


FIG. 2.16 – Analyse d'un en-tête TCP (SYN)

2.8.2.3 Transmission des données et fin de session

Après l'établissement de la connexion TCP, le drapeau ACK des segments doit être positionné pour indiquer que le champ numéro d'acquittement est valide. La transmission des données pourra se faire dans les deux sens, jusqu'à la fermeture de la session.

Pour clore la session, une extrémité envoie le drapeau FIN. Mais la fermeture de la connexion TCP n'est réalisée que lorsque l'autre extrémité envoie répond par les drapeaux FIN/ACK pour accepter la fermeture. Cette façon de faire permet d'éviter les pertes de données que pourrait occasionner une fermeture unilatérale.

2.8.3 Analyse d'un en-tête TCP

Pour utiliser le protocole de transport TCP, chargeons par exemple la page d'accueil de notre université dans un navigateur, après avoir fait démarrer une capture `Wireshark` :

`http://www.univ-paris12.fr`

et arrêtons la capture. On récupère plus de de cinq cents trames, mais nous n'allons en analyser que quelques-unes.

2.8.3.1 Première trame TCP (SYN)

Les deux premières trames concernent DNS, utilisant UDP comme nous l'avons déjà vu. Considérons la première trame TCP de cette capture (voir figure 2.16). Il n'y a pas grand chose à dire sur les en-têtes Ethernet et IP, sinon que le protocole de transport de ce dernier en-tête est `06h`, soit 6, pour indiquer TCP.

– Les deux premiers octets de l'en-tête TCP spécifient le numéro de port source. On a ici `C1 09h`, soit 49 417. Rappelons que, comme pour UDP, ce numéro de port est déterminé (presque au hasard) par l'implémentation de la couche TCP de façon à ce que deux applications différentes n'aient pas le même numéro de port pour la partie client.

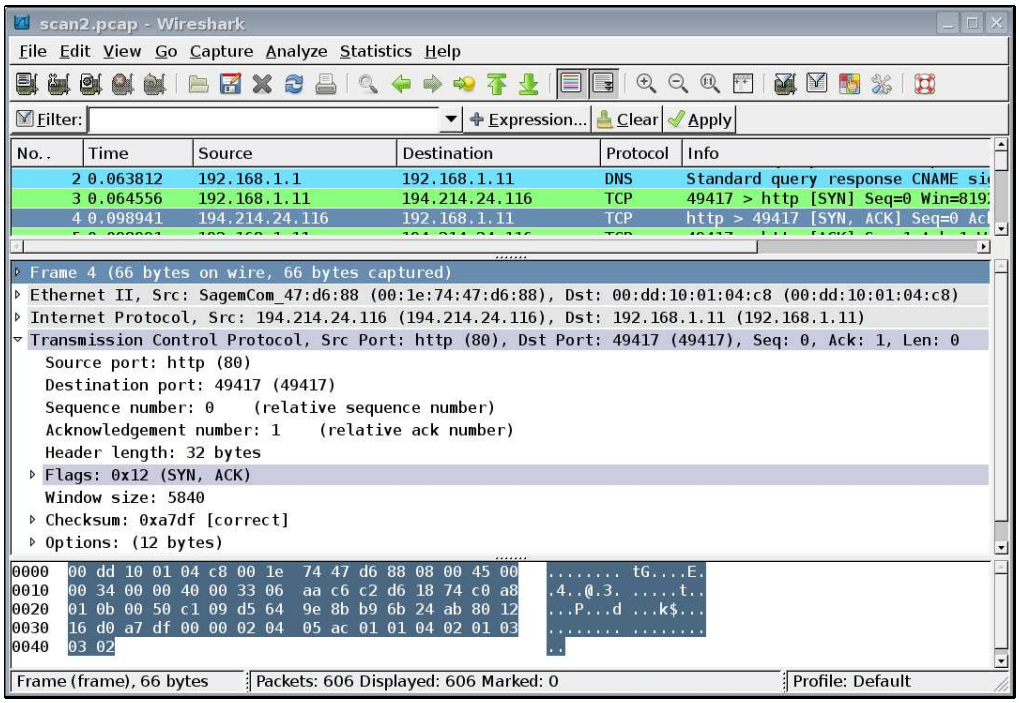


FIG. 2.17 – Analyse d'un en-tête TCP (SYN/ACK)

– Les deux octets suivants spécifient le numéro de port destination. On a ici `00 50h`, soit 80. Il s'agit d'un numéro de port bien connu, celui attribué aux services HTTP. Ceci nous indique que le message concerne *a priori* une requête HTTP ; il s'agit en fait de la première trame d'ouverture d'une session TCP.

– Les quatre octets suivants indiquent le numéro de séquence. On a ici `B9 6B 24 AAh`. Celui-ci est, rappelons-le attribué presque au hasard. Remarquons que l'analyseur se contente de

lui attribuer le numéro relatif 0.

- Les quatre octets suivants indiquent le numéro d'acquittement. On a ici zéro, ce qui est normal puisque le client essaie d'initier une connexion : il n' a donc encore reçu aucun octet du serveur. De toute façon, ce nombre n'a pas à être interprété puisque, comme nous le verrons ci-dessous, le drapeau `Ack` n'est pas positionné.
- Les quatre bits suivants indiquent la longueur de l'en-tête TCP en mots de 32 bits. On a ici `8h`, soit 32 octets : on a donc le minimum de 20 octets et 12 octets d'options.
- Les six bits suivants, tous à zéro, sont réservés pour une utilisation ultérieure.
- Les six bits suivants indiquent la valeur des drapeaux. On a `02h`, soit `000010b` : les drapeaux `Urg`, `Ack`, `Psh`, `Rst` et `Fin` ne sont pas positionnés. Le drapeau `Syn` est positionné : il indique le début d'une connexion TCP.
- Les deux octets suivants indiquent la longueur de la fenêtre. On a ici `20 D0h`, soit 8 192 octets.
- Les deux octets suivants donnent la valeur de la somme de contrôle. On a ici `12 A9h` et l'analyseur dit qu'elle est correcte. Nous vous laissons le soin de le vérifier.
- Les douze octets suivants concernent les options. Ne nous intéressons ici qu'à la première des options spécifiées. Elle est du type 2, donc spécifie la taille maximum du segment, qui est de `05 B4h`, soit 1 460 octets.

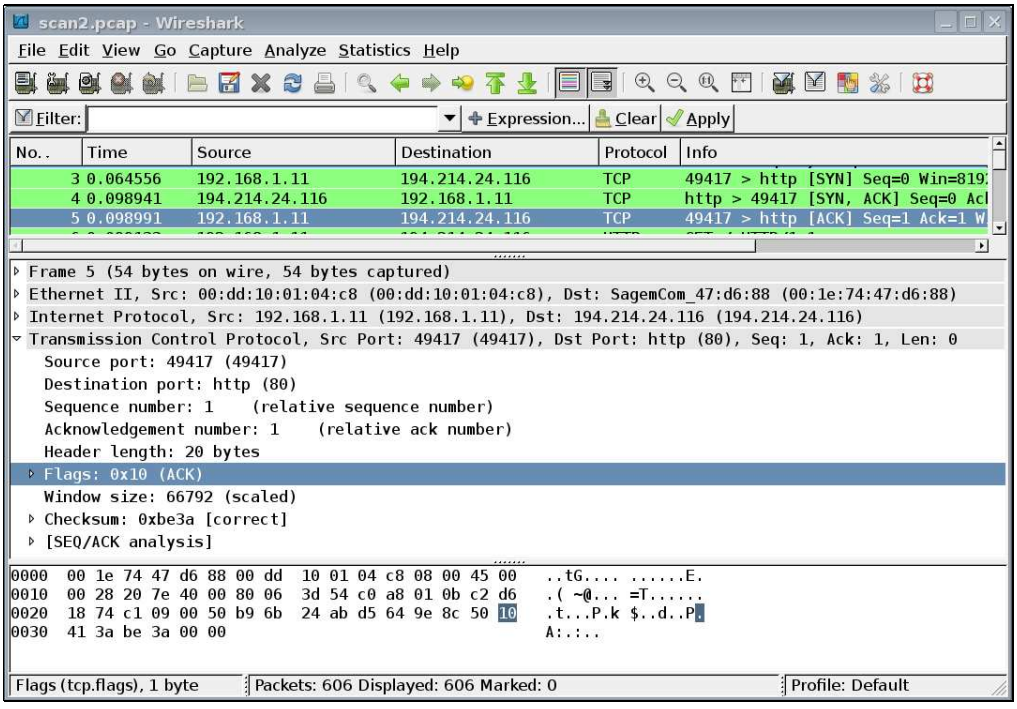


FIG. 2.18 – Analyse d'un en-tête TCP (ACK)

2.8.3.2 Deuxième trame TCP (SYN/ACK)

On peut analyser de la même façon la deuxième de ces trames (figure 2.17). Ne commentons que ce qui est intéressant :

- Le numéro de séquence est ici `D5 64 9E 8Bh`, différent donc de celui de la trame précédente : il n'y a aucune raison que le client et le serveur donnent des numéros de séquence identiques.
- Le numéro d'acquittement est `B9 6B 24 ABh`, soit le numéro de séquence du segment précédent plus un. Le serveur a donc bien reçu tous les octets du segment envoyé par le client.
- La valeur de la suite de drapeaux est `12h`, soit `010010b` : les drapeaux `Urg`, `Psh`, `Rst` et `Fin` ne sont pas positionnés alors que les drapeaux `Ack` et `Syn` le sont. Ils indiquent l'acquittement de la demande d'ouverture d'une connexion connexion TCP.
- La longueur de la fenêtre est `16 D0h`, soit 5 840, qui est bien inférieur à celle du client (8 192 octets).

2.8.3.3 Troisième trame TCP (ACK)

Ne commentons que ce qui est intéressant pour la troisième trame TCP (figure 2.18) :

- Le numéro de séquence est ici `B9 6B 24 ABh`, soit un de plus par rapport au numéro de séquence du premier segment.
- Le numéro d'acquittement est `D5 64 9E 8Ch`, soit le numéro de séquence du segment précédent plus un. Le client a donc bien reçu tous les octets du segment envoyé par le serveur.
- La valeur de la suite de drapeaux est `10h`, soit `010000b` : les drapeaux `Urg`, `Psh`, `Rst`, `SYN` et `Fin` ne sont pas positionnés alors que le drapeau `Ack` l'est. Il indique l'acquittement du numéro de séquence que le serveur lui a envoyé.

La connexion a donc bien été établie.

2.9 Le protocole ICMP

Les messages ICMP sont utilisés pour indiquer des erreurs d'un hôte à l'autre ou d'un hôte à une passerelle (*gateway* en anglais). De passerelle à passerelle, on utilise un autre protocole : **GGP** pour *Gateway to Gateway Protocol*.

Les messages ICMP sont envoyés au-dessus de IP, avec un en-tête IP dont le type de protocole est 1, comme nous l'avons déjà vu.

2.9.1 En-tête des unités d'information ICMP

L'en-tête des unités d'information ICMP comprend 32 bits :

Type (8 bits)	Code (8 bits)	Somme de contrôle (16 bits)

- Le champ `type` d'un octet spécifie le type de message d'erreur ou de réponse tel que *destination unreachable*, *echo*, *echo reply* ou *redirect message*.
- Le champ `code` d'un octet spécifie des informations supplémentaires si nécessaire.

  Par exemple si le message est du type *destination unreachable*, il existe plusieurs valeurs possibles pour ce code telles que *network unreachable*, *host unreachable* ou *port unreachable*.

  La liste complète des types et des codes des messages ICMP est tenue à jour sur le site Web de l'IANA :

  `http://www.iana.com`
- Le champ `somme de contrôle` (*checksum* en anglais) porte sur le message en entier, y compris l'en-tête IP.

2.9.2 Quelques messages ICMP

Les messages ICMP se répartissent en trois catégories : les messages d'erreur (destinataire inaccessible, temps écoulé qui informent les hôtes des problèmes se révélant au cours de leurs envois ; il ne s'agit pas d'une correction d'erreur, puisque les messages ne sont pas retransmis mais d'un système d'avertissement d'erreur), de contrôle (ralentissement de la source) et d'analyse de réseau (l'écho).

2.9.2.1 Requête et réponse d'écho (types 8 et 0)

Un administrateur réseau souhaitant vérifier si un hôte donné est toujours opérationnel peut lui envoyer un message d'écho (type 8, sans code). Si l'hôte est actif, il lui renvoie une réponse d'écho (type 0, sans code).

L'utilitaire le plus courant pour l'envoi d'un message d'écho et l'analyse des réponses s'appelle **ping** (*Packet INternet Groper*).

Type (8 bits)	0	Somme de contrôle (16 bits)
Identificateur		Numéro de séquence
Données		

- Comme nous l'avons vu, le type est 8 pour une demande d'écho et 0 pour une réponse.
- L'identificateur (*identifier* en anglais) est décidé lors de la requête. Il est renvoyé lors de la réponse.
- Le numéro de séquence (*Sequence Number* en anglais) est décidé par chaque hôte.
- Par défaut le champ des données (*Data* en anglais) est vide mais il peut contenir une quantité spécifiée par l'utilisateur de données aléatoires.

Type	Code	Description	Requête	Erreur	Référence
0	0	Echo Reply	x		RFC 792
3	0	Network unreachable		x	RFC 792
3	1	Host unreachable		x	RFC 792
3	2	Protocol unreachable		x	RFC 792
3	3	Port unreachable		x	RFC 792
3	4	Fragmentation nécessaire mais drapeau Don't F		x	RFC 792
3	5	Source routing failed		x	RFC 792
3	6	Destination network unknown		x	RFC 792
3	7	Destination host unknown		x	RFC 792
3	8	Source host isolateted (obsolete)		x	RFC 792
3	9	Destination network adminsitratively prohibited		x	RFC 792
3	10	Destination host adminsitratively prohibited		x	RFC 792
3	11	Destination unreachable for TOS		x	RFC 792
3	12	Host unreachable for TOS		x	RFC 792
3	13	Communication adminsitratively prohibited by filtering		x	RFC 1812
3	14	Host precedence violation		x	RFC 1812
3	15	Precedence cutoff in effect		x	RFC 1812
4	0	Source quench			RFC 792
5	0	Redirect for network			RFC 792
5	1	Redirect for host			RFC 792
5	2	Redirect for TOS and network			RFC 792
5	3	Redirect for TOS and host			RFC 792
8	0	Echo request	x		RFC 792
9	0	Router advertisement - Normal			RFC 1256
9	16	Router advertisement - Does not route common traffic			RFC 2002
10	0	Route selection			RFC 1256
11	0	TTL equals 0 during transit		x	RFC 792
11	1	TTL equals 0 during reassembly		x	RFC 792
12	0	IP header bad		x	RFC 792
12	1	Required options missing		x	RFC 792
12	2	IP header bad length		x	RFC 792
13	0	Timestamp request (obsolete)	x		RFC 792
14		Timestamp reply (obsolete)	x		RFC 792
15	0	Information request (obsolete)	x		RFC 792
16	0	Information reply (obsolete)	x		RFC 792
18	0	Address mask reply	x		RFC 950
30	0	Traceroute	x		RFC 1393
31	0	Datagram conversion Error		x	RFC 1475

FIG. 2.19 – Types et codes ICMP

2.9.2.2 Destination inaccessible (type 3)

La principale famille de messages d'erreur ICMP regroupe les messages de destination inaccessible (type 3), émis par les routeurs ne parvenant pas à transmettre les paquets qu'ils traitenet à leurs destinataires.

3	Code (8 bits)	Somme de contrôle (16 bits)
Non utilisé		
En-tête Internet + 64 bits du datagramme d'origine		

- Le type *Destination Unreachable* a 16 codes de base :
  - Code 0 (*Network unreachable*) dit que le réseau (local du destinataire) n'est pas accessible actuellement.
  - Code 1 (*Host unreachable*) dit que (le réseau est accessible mais que) l'hôte n'est pas accessible actuellement.
  - Code 2 (*Protocol unreachable*) dit que le protocole (TCP, UDP, etc) n'est pas utilisable actuellement.
  - Code 3 (*Port unreachable*) dit que le port n'est pas accessible actuellement.
  - Code 4 (*Fragmentation needed and DF set*) dit qu'un paquet doit être fragmenté mais que le drapeau *Do not fragment* est présent, d'où ce retour de la part de la passerelle.
  - Code 5 (*Source route failed*) dit que le routage a échoué.
  - Code 6 (*Destination network unknown*) dit qu'il n'existe pas de route vers le réseau indiqué.
  - Code 7 (*Destination host unknown*) dit qu'il n'existe pas de route vers l'hôte indiqué.
  - Code 8 (*Source host isolated*) doit être renvoyé si l'hôte est isolé. Ce code est maintenant obsolète.
  - Code 9 (*Destination network administratively prohibited*) dit que le réseau est bloqué à une passerelle.
  - Code 10 (*Destination host administatively prohibited*) dit qu'on ne peut pas accéder à l'hôte à cause de l'administration du routage.
  - Code 11 (*Network unreachable for TOS*) dit que le réseau n'est pas accessible à cause d'un mauvais positionnement du TOS du paquet.
  - Code 12 (*Host unreachable for TOS*) dit que l'hôte n'est pas accessible à cause du TOS du paquet.
  - Code 13 (*Communication administratively prohibited by filtering*) dit que le paquet a été interdit à cause du filtrage, par exemple à un pare-feu.
  - Code 14 (*Host precedence violation*) est envoyé par le premier routeur pour notifier à un hôte connecté que la priorité utilisée n'est pas permise pour cette combinaison destination/source.
  - Code 15 (*Precedence cutoff in effect*) est envoyé par le premier routeur si le paquet a une priorité trop basse.
- Le message contient une petite partie des données, contenant l'entier IP complet et 64 bits du paquet IP originel, qui devrait comporter les ports.

2.9.2.3 Ralentissement de la source (type 4)

Certains types de messages ICMP servent à altérer le fonctionnement des hôtes ou routeurs du réseau. Un message de demande de ralentissement de la source (*Source Quench* en anglais ; type 4, sans code) sert au **controle de flux**, c'est-à-dire à gérer le débit auquel les hôtes envoient leurs paquets. Un hôte recvant ce type de message est invité à réduire son débit de transmission.

4	0	Somme de contrôle (16 bits)
Non utilisé		
En-tête Internet + 64 bits du datagramme d'origine		

2.9.2.4 Redirection (type 5)

Supposons que vous ayez un réseau (192.168.0.0/24) comportant plusieurs hôtes et deux passerelles : une passerelle vers le réseau 10.0.0.0/24 et une passerelle par défaut pour le reste. Considérons qu'un des hôtes du réseau 192.168.0.0/24 soit paramétré pour envoyer *via* la passerelle par défaut. S'il envoie un paquet à 10.0.0.26, celui-ci passera par la passerelle par défaut, qui connaît bien sûr le réseau 10.0.0.0/24. Mais la passerelle par défaut sait qu'il est plus rapide d'envoyer le paquet directement à la passerelle 10.0.0.0/24. Elle enverra donc un message ICMP *Redirect* à l'hôte, en lui disant la bonne passerelle à utiliser tout en envoyant son paquet vers la passerelle 10.0.0.0/24. L'hôte connaîtra maintenant l'existence de cette passerelle et espérons qu'il l'utilisera dans le futur.

5	Code (8 bits)	Somme de contrôle (16 bits)
Adresse Internet de la passerelle		
En-tête Internet + 64 bits du datagramme d'origine		

- Le champ principal est celui de l'adresse Internet de la passerelle à utiliser.
- Il y a 4 codes possibles :
  - Code 0 (*Redirect for network*) est seulement utilisé pour rediriger vers un réseau en entier (comme dans l'exemple ci-dessus).
  - Code 1 (*Redirect for host*) est seulement utilisé pour rediriger vers un hôte donné.
  - Code 2 (*Redirect for TOS and network*) est seulement utilisé pour rediriger dans le cas d'un type de service et d'un réseau complet donnés.
  - Code 3 (*Redirect for TOS and host*) est seulement utilisé pour rediriger dans le cas d'un type de service et d'un hôte donnés.

2.9.2.5 Temps écoulé (type 11)

Le message ICMP *temps écoulé* (*TTL equals 0* ou *Times Exceeded* en anglais) est envoyé à l'hôte si le champ `TTL` atteint 0 lors du transit dans une passerelle ou lors du réassemblage des fragments sur l'hôte de destination, tout en écartant le paquet.

11	Code (8 bits)	Somme de contrôle (16 bits)
non utilisé		
En-tête Internet + 64 bits du datagramme d'origine		

Deux codes sont utilisés :

– Code 0 (*TTL equals 0 during transit*) est envoyé si le TTL a atteint 0 lors du transit par une passerelle.
– Code 1 (*TTL equals 0 during reassembly*) est envoyé si le TTL a atteint 0 lors du réassemblage des fragments.

2.9.2.6 Problème de paramétrage

Le message ICMP *Parameter problem* est utilisé pour dire à l'expéditeur qu'une passerelle ou l'hôte de destination a des problèmes à comprendre certaines parties de l'en-tête IP.

Type (8 bits)	Code (8 bits)	Somme de contrôle (16 bits)
Pointeur	non utilisé	
En-tête Internet + 64 bits du datagramme d'origine		

– Deux codes sont utilisés :
  – Code 0 (*IP header bad (catchall error)*) est envoyé s'il y a une erreur dans l'en-tête IP. Le champ *pointeur* spécifie la partie de l'en-tête qui pose problème.
  – Code 1 (*Required options missing*) est envoyé si une option requise manque.
– Le champ pointeur (*pointer*) sert dans le cas du code 0.

2.9.3 Analyse de trames ICMP

Effectuons un ping (voir figure 2.20) tout en capturant les trames grâce à `Wireshark`. Nous obtenons un certain nombres de trames de reqêtes d'écho et de réponses d'écho.

FIG. 2.20 – L'utilitaire ping

2.9.3.1 Requête d'écho

Étudions la première trame, qui est une requête d'écho (voir figure 2.21). Il n'y a pas grand chose à dire sur les en-têtes Ethernet et IP, sinon que le protocole de ce dernier en-tête est `01h`, soit 1, pour indiquer un message ICMP. Commentons plus en détail ce message ICMP.

– Le type est `08h`, ce qui indique une requête d'écho.
– Le code est `00h` ; il n'y a pas d'autre valeur possible pour une requête d'écho.

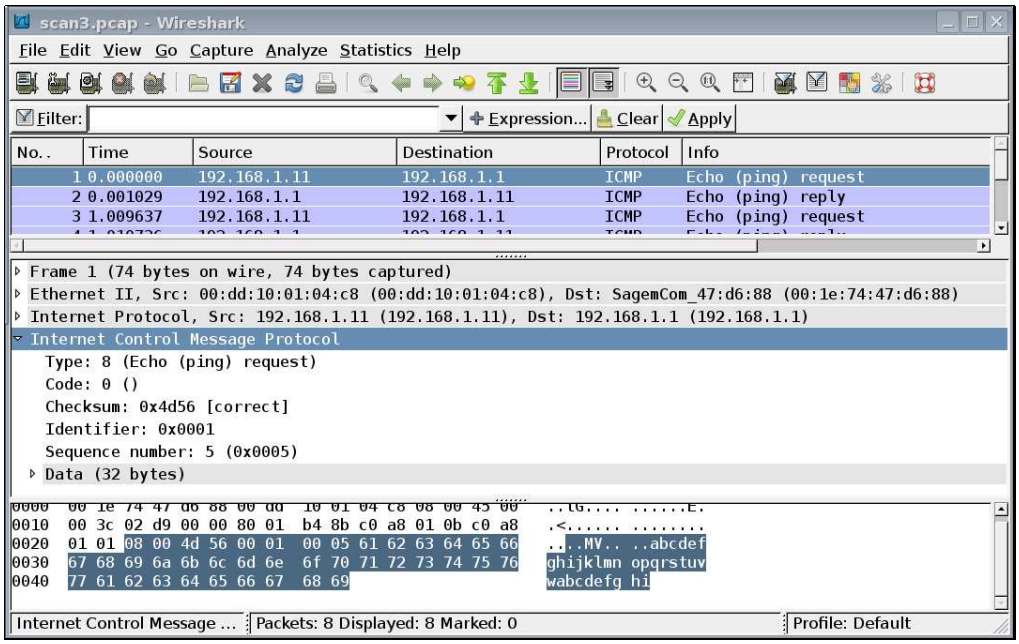


FIG. 2.21 – Requête d'écho

- La somme de contrôle est `4D56h`. `Wireshark` dit que celle-ci est correcte ; nous laissons le soin au lecteur de vérifier.
- L'identificateur est `0001h`, soit 1.
- Le numéro de séquence est `0005h`, soit 5.
- Il y a 32 octets de données, constitués tout simplement du code ASCII des seize premières lettres minuscules de l'alphabet.

2.9.3.2 Reponse d'écho

Étudions également la seconde trame, qui est la réponse à notre requête d'écho (voir figure 2.22). Il n'y a toujours pas grand chose à dire sur les en-têtes Ethernet et IP, sinon que le protocole de ce dernier en-tête est également `01h`, soit 1, pour indiquer un message ICMP. Commentons plus en détail ce message ICMP.

- Le type est `00h`, ce qui indique une réponse d'écho.
- Le code est `00h` ; il n'y a pas d'autre valeur possible pour une requête d'écho.
- La somme de contrôle est `5556h`. `Wireshark` dit que celle-ci est correcte ; nous laissons le soin au lecteur de vérifier.
- L'identificateur est `0001h`, soit 1, ce qui est bien le même que celui de la requête.
- Le numéro de séquence est `0005h`, soit 5.
- Il y a 32 octets de données, constitués tout simplement du code ASCII des seize premières lettres minuscules de l'alphabet.

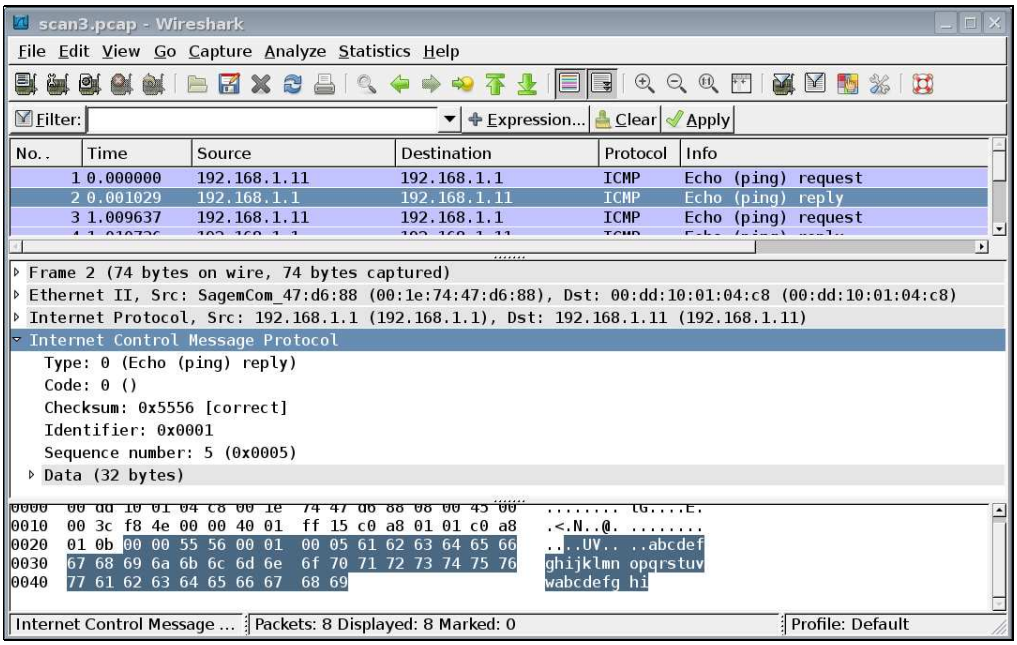


FIG. 2.22 – Réponse d'écho