EI-2I TP2

POLYTECH SORBONNE

1. Contrôle de flux dans TCP

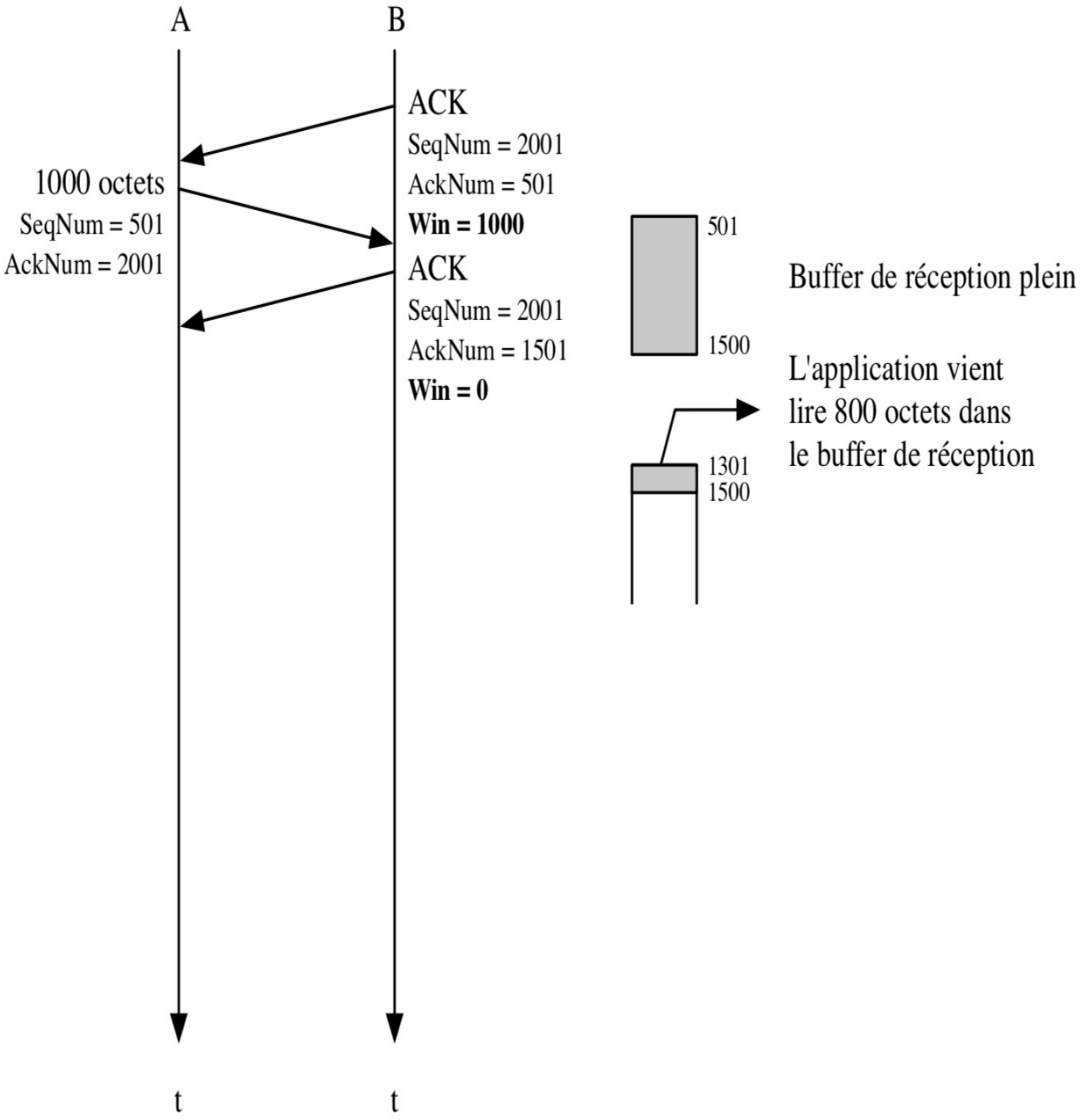
Pour son contrôle de flux, TCP utilise un mécanisme de fenêtre coulissante de taille variable (contrairement aux protocoles de niveaux inférieurs vus jusqu'à présent qui utilisent une fenêtre de taille fixe). La taille courante de la fenêtre est communiquée dans le champ « Window » (également appelé « Advertised Window ») du segment et correspond au nombre d'octets de données que l'émetteur est prêt à recevoir à partir du numéro d'acquittement inclus dans le segment, avant de recevoir plus d'autorisations du récepteur.

La taille de la fenêtre est susceptible de prendre momentanément une valeur nulle. Le récepteur met alors la valeur 0 dans le champ Window du ACK qu'il envoie à l'émetteur. Ce dernier va ensuite sonder périodiquement le récepteur pour déterminer quand la fenêtre de contrôle de flux devient positive (par envoi d'un segment contenant 1 octet de données) ; si le ACK du récepteur indiquant une réouverture de fenêtre se perd, une sonde ultérieure de l'émetteur va provoquer un duplicata de cet ACK. C'est la solution qui a été retenue pour TCP, dans le cadre de sa philosophie « smart sender / dumb receiver » (i.e. avoir une algorithmique aussi simple que possible au niveau du récepteur).

1.1. Une machine envoie un segment TCP avec les valeurs suivantes : SeqNum = 1227, AckNum = 2513 et Win = 3007. Quelle plage d'octets est-elle prête à recevoir ?

On considère l'échange suivant dans lequel la machine A doit envoyer 1500 octets de données à la machine B, mais B a auparavant annoncé à A une fenêtre de taille 1000. A envoie donc seulement 1000 octets qui viennent remplir le buffer de réception de B. Peu de temps après, l'application de B vient lire 800 octets dans le buffer.

1.2. Compléter l'échange.



2. Continuation sur WireShark

Nous allons, lors de ce TP, étudier plus en détail le protocole HTTP et d'autres protocoles correspondents à différents niveaux pour compléter les échanges de HTTP.

Il faut vous essayer d'installer **WireShark** sur votre machine pour réaliser ce TP. Un fichier pré-capturé vous est fourni sur Moodle.
Commencez donc par récupérer le fichier suivant et à le sauvegarder sur votre compte : **capture_http.pcap**

Dans un terminal, exécutez la commande **wireshark** (et avec la commande **man wireshark** obtenez les informations de base). Vous devez lancer l'application en mode **« sans privilège ».** Initialement l'écran est vide car aucune capture n'a été réalisée ou chargée. Cliquez sur le menu **File** et sélectionnez **Open.**

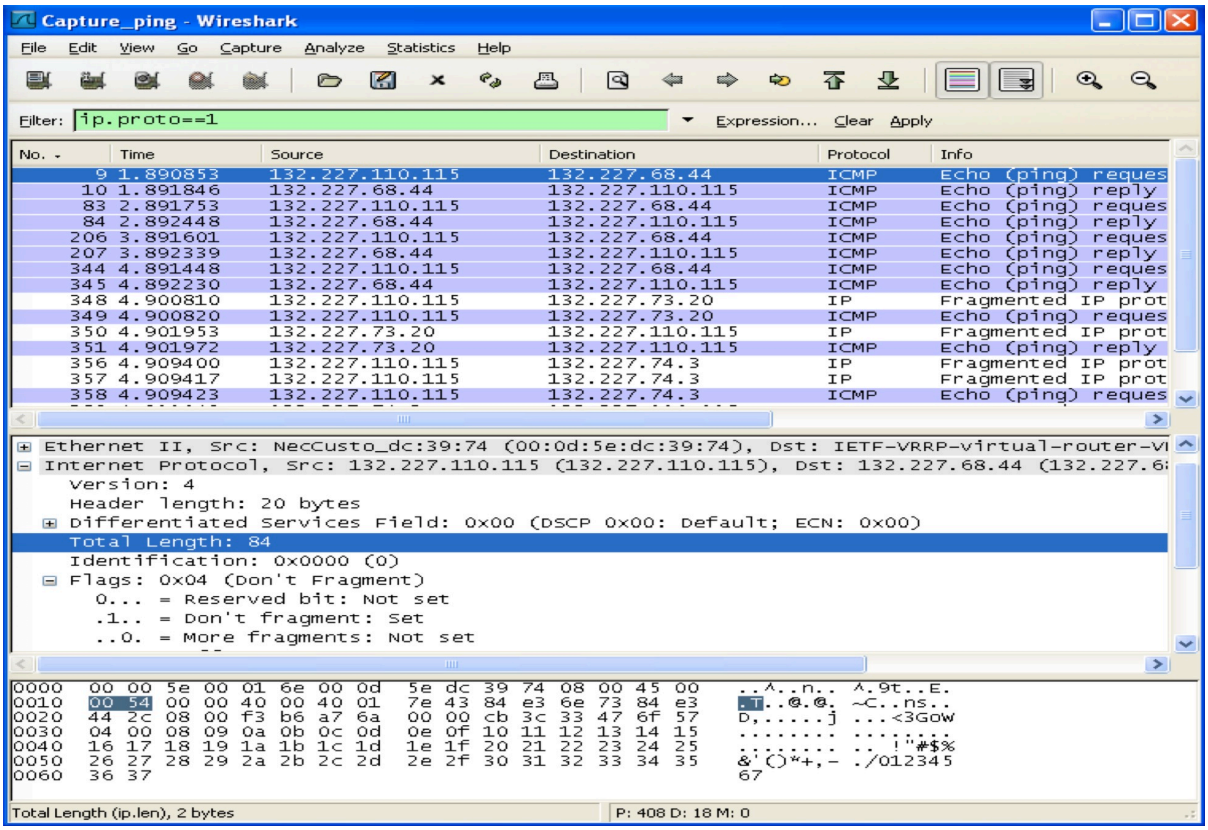
§ Sélectionnez le fichier **capture_http.tcap**

§ Ne pas spécifier de filtres dans le champ **Filter** (nous y reviendrons plus loin).

§ Dans le menu **Preferences,** désactivez les options **Enable MAC name resolution, Enable network name resolution** et **Enable transport name resolution.**

§ Cliquez finalement sur **Ouvrir.**

L'interface de Wireshark est constituée de trois fenêtres principales (voir figure ci-dessous) :



La fenêtre du haut liste les trames capturées et résume leurs caractéristiques. En cliquant sur l'une des trames dans cette fenêtre, le contenu de la trame apparaît en hexadécimal dans la fenêtre du bas, et le contenu de chaque champ est décrit dans la fenêtre du milieu. Ces trois fenêtres sont coordonnées (un clic dans l'une des fenêtres a un impact sur les deux autres).

La fenêtre du milieu décrit précisément chaque champ de la trame. Ces champs, ainsi que diverses informations fournies par le logiciel, y sont présentés dans une structure arborescente.

La fenêtre du bas est constituée essentiellement de la trame en hexadécimal. Les champs sélectionnés dans l'arbre de la fenêtre du milieu sont surlignés dans la trame. Les données sont présentées selon trois colonnes :

§ La première colonne indique, sur 4 caractères hexadécimaux, le rang du premier octet de la ligne courante dans la trame ;

§ La deuxième colonne affiche la valeur hexadécimale de 16 octets ;

§ La troisième colonne représente les caractères ASCII correspondant aux 16 octets de la seconde colonne (la correspondance n'est significative que lorsque du texte lisible se trouve encodé dans ces octets).

Observez le contenu des trois fenêtres proposées par WireShark.

2.2. Dans quels formats sont représentées les données de la fenêtre du bas ?

2.3. Quels sont les différents protocoles que vous pouvez observer dans la capture affichée ?

2.4. Combien de protocoles est capable d'analyser la version de WireShark que vous utilisez ? (à l'aide: www.wireshark.org)

3. Filtres d'affichage WireShark

En plus des trois fenêtres décrites précédemment, il existe un champ fort intéressant, situé juste au dessus de la première fenêtre, et qui est le champ **Filter**. Il sert à sélectionner certaines trames en fonction de champs particuliers (par exemple les adresses, les protocoles, etc.). Après écriture du filtre, il est activé en cliquant sur le bouton **Apply**. Ainsi, seules les trames autorisées par le filtre sont visibles dans les fenêtres. Un clic sur le bouton **Clear** désactive le filtre d'affichage en cours d'utilisation.

La syntaxe d'un filtre simple est la suivante :

```
nom_du_champs relation valeur
```

Par exemple, si vous ne souhaitez voir apparaître que les trames correspondant au protocole http, vous pouvez appliquer le filtre suivant :

```
tcp.port == 80
```

Pour créer des filtres plus complexes, vous pouvez combiner plusieurs filtres simples grâce aux opérateurs logiques **and** et **or**. Pour vous faciliter l'écriture d'un filtre, cliquez sur l'onglet **Expression** à droite de l'espace réservé à l'écriture du filtre. Vous aurez ainsi accès aux différents noms de champs utilisables dans les filtres ainsi qu'à des valeurs prédéfinies. Si vous êtes plus à l'aise, vous pouvez écrire directement le filtre dans l'espace réservé.

3.1. Décrivez et appliquez un filtre qui ne sélectionne que les trames ARP de/vers l'interface ayant l'adresse MAC `00:1b:77:d2:d2:27`.

Supprimez le filtre précédent

4. Trace HTTP

Nous allons maintenant étudier en détail l'échange contenu dans la trace **capture_http.pcap.**

4.1. Quel est l'objectif de l'échange ARP initial ? Qui est le destinataire de la première trame ? Quelles sont les adresses des participants de cet échange ?

4.2. Quel est l'objectif des trames 3 à 6 ? Quel site est visé ? Quelle est la différence entre les trames 3 et 4, d'une part, et 5 et 6, d'autre part ?

4.3. Quel est le protocole de transport associé au protocole applicatif utilisé dans les trames 3 à 6 ? Quelles sont les fonctionnalités apportées par ce protocole de transport ?

4.4. Quel semble être le but de cet échange ?

4.5. Cette trace a été obtenue en tapant l'url http://ratp.fr dans un navigateur. D'après vous, pourquoi observe-t-on plusieurs requêtes (GET) ?

4.6. Quel est le protocole de transport utilisé à partir de la trame 7 ? Quelles sont les fonctionnalités apportées par ce protocole de transport ? Quelles différences pouvez-vous constater dans l'en-tête par rapport au protocole de transport utilisé dans les trames 3 à 6 ?

4.7. Sélectionnez toutes les trames relatives au protocole http (port 80). Quel filtre appliquez- vous ?

4.8. A quoi correspond la trame 45 ? Le fichier correspondant a-t-il été envoyé en une seule fois ?

4.9. Pourquoi y a-t-il une seconde requête DNS (trame 48) ? Est-elle liée au reste de l'échange ?