Travaux Pratiques
Étude des protocoles UDP et TCP

Séance de TP entièrement effectuée avec le logiciel Marionnet. Durée estimée : 1h - 1h30.

Prérequis. Cour magistral sur la notion de port (adressage des applications), et sur l'intérêt, le format et le fonctionnement des deux protocoles.

Câblage et configuration du réseau local

On utilise 2 machines, m_1 et m_2, pour réaliser un réseau local $LAN_1 = \{m_1, m_2\}$ en 192.168.1.0/24. L'équipement d'interconnexion pourra être un simple câble croisé.

Distributions GNU/Linux. Utilisez une distribution permettant de lancer `wireshark` (p.e. Debian) sur la machine m_1 et une distribution permettant de lancer un serveur HTTP (p.e. Mandriva ou Pinocchio) sur m_2.

Attribution des IP. Par simplicité, la machine m_i aura l'adresse 192.168.1.i.

Première partie
Étude du protocole UDP

Nous allons étudier UDP en générant des messages DNS, protocole de la couche application qui utilise la sous-couche UDP avec numéro de port conventionnel 53 (côté serveur). Commencez par vérifier que le numéro de port UDP associé au service DNS (`domain`) soit bien le numéro 53 :

```
m1#  grep domain /etc/services
```

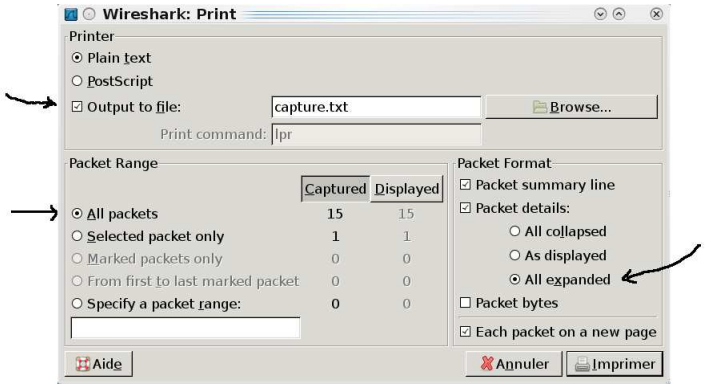
Configurez les interfaces des machines de façon que le LAN_1 soit fonctionnel. Ensuite, tout en capturant les trames avec `wireshark` à partir de m_1, lancez la commande `nslookup` suivante, qui permet de demander la résolution du nom `www.google.fr` au (prétendu) serveur `192.168.1.2` :

```
m1#  nslookup www.google.fr 192.168.1.2
```

La machine m_2 n'ayant pas de serveur DNS activé (sinon tuez-le !, il s'appelle probablement `named`), vous devez obtenir une séquence de capture de la forme suivante :

```
No. Time    Source       Destination  Protocol Info
1 0.000000  192.168.1.1  192.168.1.2  DNS Standard query A www.google.fr
2 0.000552  192.168.1.2  192.168.1.1  ICMP Destination unreachable (Port unreachable)
```

Avec `wireshark` vous pouvez imprimer (CTRL-P) dans un fichier le détail de ces trames par la fenêtre de dialogue suivante :



Une fois imprimé, vous obtenez un fichier dont le contenu a la forme suivante :

```
No. Time    Source       Destination  Protocol Info
1 0.000000  192.168.1.1  192.168.1.2  DNS Standard query A www.google.fr
Frame 1 (73 bytes on wire, 73 bytes captured)
Ethernet II, Src: 02:04:06:00:00:01, Dst: 02:04:06:00:00:02
    ...
    Type: IP (0x0800)
Internet Protocol, Src: 192.168.1.1 (192.168.1.1), Dst: 192.168.1.2 (192.168.1.2)
    Version: 4
    Header length: 20 bytes
    ...
    Total Length: 59
    ...
    Time to live: 64
    Protocol: UDP (0x11)
    Header checksum: 0xb75e [correct]
    Source: 192.168.1.1 (192.168.1.1)
    Destination: 192.168.1.2 (192.168.1.2)
User Datagram Protocol, Src Port: m1 (2049), Dst Port: domain (53)
    Source port: m1 (2049)
    Destination port: domain (53)
    Length: 39
    Checksum: 0x084b [correct]
Domain Name System (query)
    Transaction ID: 0x4524
    Flags: 0x0100 (Standard query)
    ...
    Questions: 1
    ...
    Queries
        www.google.fr: type A, class IN
            Name: www.google.fr
            Type: A (Host address)
            Class: IN (0x0001)
No. Time    Source       Destination  Protocol Info
2 0.000552  192.168.1.2  192.168.1.1  ICMP Destination unreachable (Port unreachable)
...
```

Comparez donc votre fichier avec le contenu ci-dessu : quel champs sont égaux, quels sont différents ?

Deuxième partie

Étude du protocole TCP

Nous allons étudier TCP en générant des messages HTTP, protocole de la couche application qui utilise la sous-couche TCP avec numéro de port conventionnel 80 (côté serveur). Commencez par vérifier que le numéro de port TCP associé au service HTTP (`www`) soit bien le numéro 80 :

```
m1#  grep http /etc/services
m1#  grep www /etc/services
```

Lancez un serveur HTTP sur m_2 :

```
m2#  /etc/init.d/apache2 start
```

(remplacez `apache2` par `apache` ou `httpd` si nécessaire). Puis, tout en capturant les trames avec `wireshark` (toujours sur m_1), tentez une connexion cliente HTTP :

```
m1#  lynx http://192.168.1.2
```

(remplacez `lynx` par `links` ou `firefox` ou autre butineur installé sur m_1 si nécessaire). Vous devez obtenir une séquence de capture de la forme suivante :

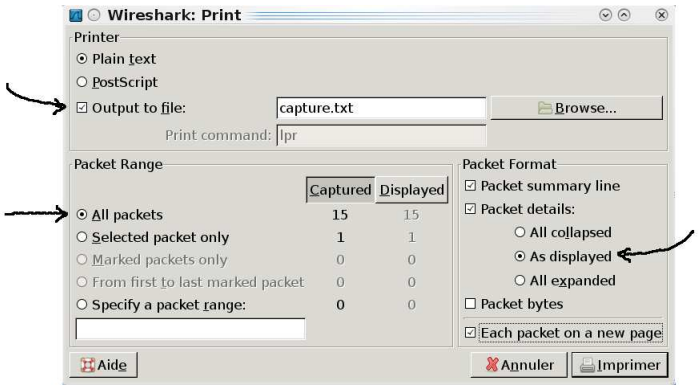
```
No.  Time     Source        Destination   Protocol Info
 1 0.000000   192.168.1.1   192.168.1.2     TCP     m1 > http [SYN] Seq=0 ...
 2 0.000476   192.168.1.2   192.168.1.1     TCP     http > m1 [SYN, ACK] Seq=0 Ack=1 ...
 3 0.000534   192.168.1.1   192.168.1.2     TCP     m1 > http [ACK] Seq=1 Ack=1 ...
 4 0.021186   192.168.1.1   192.168.1.2     HTTP    GET / HTTP/1.0
 5 0.022686   192.168.1.2   192.168.1.1     TCP     http > m1 [ACK] Seq=1 Ack=216 ...
 6 0.023423   192.168.1.2   192.168.1.1     HTTP    HTTP/1.1 200 OK  (text/html)
 7 0.023440   192.168.1.1   192.168.1.2     TCP     m1 > http [ACK] Seq=216 Ack=297 ...
 8 0.023425   192.168.1.2   192.168.1.1     TCP     http > m1 [FIN, ACK] Seq=297 Ack=216 ...
 9 0.065447   192.168.1.1   192.168.1.2     TCP     m1 > http [ACK] Seq=216 Ack=298 ...
10 0.087474   192.168.1.1   192.168.1.2     TCP     m1 > http [FIN, ACK] Seq=216 Ack=298 ...
11 0.087886   192.168.1.2   192.168.1.1     TCP     http > m1 [ACK] Seq=298 Ack=217 ...
```

Question 1.

Comparez avec votre séquence de capture : y a t-il des différences concernant :
- l'ordre des trames ?
- les numéros de séquence ou acquittement ?
- comment les expliquez-vous (ou le expliqueriez-vous éventuellement) ?
- pourquoi y-a-t-il autant de trames ? pourrait-il y en avoir plus ? moins ?

Question 2.

Avec `wireshark` vous pouvez imprimer dans un fichier le détail des trames capturées en montrant seulement les informations détaillées à l'écran (partie "ouvertes" avec la souris). Il s'agit de l'option `As displayed`, qui est cochée dans la capture d'écran suivante :



Sélectionnez donc une trame quelconque, ouvrez seulement la partie TCP et imprimez. Vous devez obtenir un fichier dont le contenu ressemblera à celui des pages suivantes. Comparez-le avec votre fichier : quel champs sont égaux, quels sont différents ?

```
No.  Time     Source        Destination  Protocol Info
 1 0.000000   192.168.1.1   192.168.1.2    TCP     m1 > http [SYN] Seq=0 ...
Frame 1 (74 bytes on wire, 74 bytes captured)
Ethernet II, Src: 02:04:06:00:00:01, Dst: 02:04:06:00:00:02
Internet Protocol, Src: 192.168.1.1 (192.168.1.1), Dst: 192.168.1.2 (192.168.1.2)
Transmission Control Protocol, Src Port: m1 (2657), Dst Port: http (80), Seq: 0, Len: 0
    Source port: m1 (2657)
    Destination port: http (80)
    Sequence number: 0    (relative sequence number)
    Header length: 40 bytes
    Flags: 0x02 (SYN)
    Window size: 5840
    Checksum: 0x42c8 [correct]
    Options: (20 bytes)
```

```
No.  Time     Source        Destination  Protocol Info
 2 0.000476   192.168.1.2   192.168.1.1    TCP     http > m1 [SYN, ACK] Seq=0 Ack=1 ...
Frame 2 (74 bytes on wire, 74 bytes captured)
Ethernet II, Src: 02:04:06:00:00:02, Dst: 02:04:06:00:00:01
Internet Protocol, Src: 192.168.1.2 (192.168.1.2), Dst: 192.168.1.1 (192.168.1.1)
Transmission Control Protocol, Src Port: http (80), Dst Port: m1 (2657), Seq: 0, Ack: 1, Len: 0
    Source port: http (80)
    Destination port: m1 (2657)
    Sequence number: 0    (relative sequence number)
    Acknowledgement number: 1    (relative ack number)
    Header length: 40 bytes
    Flags: 0x12 (SYN, ACK)
    Window size: 5792
    Checksum: 0xe453 [correct]
    Options: (20 bytes)
    [SEQ/ACK analysis]
```

```
No.  Time     Source        Destination  Protocol Info
 3 0.000534   192.168.1.1   192.168.1.2    TCP     m1 > http [ACK] Seq=1 Ack=1 ...
Frame 3 (66 bytes on wire, 66 bytes captured)
Ethernet II, Src: 02:04:06:00:00:01, Dst: 02:04:06:00:00:02
Internet Protocol, Src: 192.168.1.1 (192.168.1.1), Dst: 192.168.1.2 (192.168.1.2)
Transmission Control Protocol, Src Port: m1 (2657), Dst Port: http (80), Seq: 1, Ack: 1, Len: 0
    Source port: m1 (2657)
    Destination port: http (80)
    Sequence number: 1    (relative sequence number)
    Acknowledgement number: 1    (relative ack number)
    Header length: 32 bytes
    Flags: 0x10 (ACK)
    Window size: 5840 (scaled)
    Checksum: 0x1e52 [correct]
    Options: (12 bytes)
    [SEQ/ACK analysis]
```

```
No.  Time     Source        Destination  Protocol Info
 4 0.021186   192.168.1.1   192.168.1.2    HTTP    GET / HTTP/1.0
Frame 4 (281 bytes on wire, 281 bytes captured)
Ethernet II, Src: 02:04:06:00:00:01, Dst: 02:04:06:00:00:02
Internet Protocol, Src: 192.168.1.1 (192.168.1.1), Dst: 192.168.1.2 (192.168.1.2)
Transmission Control Protocol, Src Port: m1 (2657), Dst Port: http (80), Seq: 1, Ack: 1, Len: 215
    Source port: m1 (2657)
    Destination port: http (80)
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 216    (relative sequence number)]
    Acknowledgement number: 1    (relative ack number)
    Header length: 32 bytes
    Flags: 0x18 (PSH, ACK)
    Window size: 5840 (scaled)
    Checksum: 0xd82e [correct]
    Options: (12 bytes)
Hypertext Transfer Protocol
```

```
No.  Time     Source        Destination  Protocol Info
 5 0.022686   192.168.1.2   192.168.1.1    TCP     http > m1 [ACK] Seq=1 Ack=216 ...
Frame 5 (66 bytes on wire, 66 bytes captured)
```

```
Ethernet II, Src: 02:04:06:00:00:02, Dst: 02:04:06:00:00:01
Internet Protocol, Src: 192.168.1.2 (192.168.1.2), Dst: 192.168.1.1 (192.168.1.1)
Transmission Control Protocol, Src Port: http (80), Dst Port: m1 (2657), Seq: 1, Ack: 216, Len: 0
    Source port: http (80)
    Destination port: m1 (2657)
    Sequence number: 1    (relative sequence number)
    Acknowledgement number: 216    (relative ack number)
    Header length: 32 bytes
    Flags: 0x10 (ACK)
    Window size: 6864 (scaled)
    Checksum: 0x1b77 [correct]
    Options: (12 bytes)
    [SEQ/ACK analysis]
```

```
No.  Time     Source        Destination  Protocol Info
 6 0.023423   192.168.1.2   192.168.1.1    HTTP    HTTP/1.1 200 OK  (text/html)
Frame 6 (362 bytes on wire, 362 bytes captured)
Ethernet II, Src: 02:04:06:00:00:02, Dst: 02:04:06:00:00:01
Internet Protocol, Src: 192.168.1.2 (192.168.1.2), Dst: 192.168.1.1 (192.168.1.1)
Transmission Control Protocol, Src Port: http (80), Dst Port: m1 (2657), Seq: 1, Ack: 216, Len: 296
    Source port: http (80)
    Destination port: m1 (2657)
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 297    (relative sequence number)]
    Acknowledgement number: 216    (relative ack number)
    Header length: 32 bytes
    Flags: 0x18 (PSH, ACK)
    Window size: 6864 (scaled)
    Checksum: 0x0f1e [correct]
    Options: (12 bytes)
Hypertext Transfer Protocol
Line-based text data: text/html
```

```
No.  Time     Source        Destination  Protocol Info
 7 0.023440   192.168.1.1   192.168.1.2    TCP     m1 > http [ACK] Seq=216 Ack=297 ...
Frame 7 (66 bytes on wire, 66 bytes captured)
Ethernet II, Src: 02:04:06:00:00:01, Dst: 02:04:06:00:00:02
Internet Protocol, Src: 192.168.1.1 (192.168.1.1), Dst: 192.168.1.2 (192.168.1.2)
Transmission Control Protocol, Src Port: m1 (2657), Dst Port: http (80), Seq: 216, Ack: 297, Len: 0
    Source port: m1 (2657)
    Destination port: http (80)
    Sequence number: 216    (relative sequence number)
    Acknowledgement number: 297    (relative ack number)
    Header length: 32 bytes
    Flags: 0x10 (ACK)
    Window size: 6912 (scaled)
    Checksum: 0x1a37 [correct]
    Options: (12 bytes)
    [SEQ/ACK analysis]
```

```
No.  Time     Source        Destination  Protocol Info
 8 0.023425   192.168.1.2   192.168.1.1    TCP     http > m1 [FIN, ACK] Seq=297 Ack=216 ...
Frame 8 (66 bytes on wire, 66 bytes captured)
Ethernet II, Src: 02:04:06:00:00:02, Dst: 02:04:06:00:00:01
Internet Protocol, Src: 192.168.1.2 (192.168.1.2), Dst: 192.168.1.1 (192.168.1.1)
Transmission Control Protocol, Src Port: http (80), Dst Port: m1 (2657), Seq: 297, Ack: 216, Len: 0
    Source port: http (80)
    Destination port: m1 (2657)
    Sequence number: 297    (relative sequence number)
    Acknowledgement number: 216    (relative ack number)
    Header length: 32 bytes
    Flags: 0x11 (FIN, ACK)
    Window size: 6864 (scaled)
    Checksum: 0x1a4e [correct]
    Options: (12 bytes)
    [SEQ/ACK analysis]
```

```
No.  Time     Source        Destination  Protocol Info
 9 0.065447   192.168.1.1   192.168.1.2    TCP     m1 > http [ACK] Seq=216 Ack=298 ...
Frame 9 (66 bytes on wire, 66 bytes captured)
Ethernet II, Src: 02:04:06:00:00:01, Dst: 02:04:06:00:00:02
```

```
Internet Protocol, Src: 192.168.1.1 (192.168.1.1), Dst: 192.168.1.2 (192.168.1.2)
Transmission Control Protocol, Src Port: m1 (2657), Dst Port: http (80), Seq: 216, Ack: 298, Len: 0
    Source port: m1 (2657)
    Destination port: http (80)
    Sequence number: 216    (relative sequence number)
    Acknowledgement number: 298    (relative ack number)
    Header length: 32 bytes
    Flags: 0x10 (ACK)
    Window size: 6912 (scaled)
    Checksum: 0x1a32 [correct]
    Options: (12 bytes)
    [SEQ/ACK analysis]
```

```
No.  Time     Source        Destination  Protocol Info
10 0.087474   192.168.1.1   192.168.1.2    TCP     m1 > http [FIN, ACK] Seq=216 Ack=298 ...
Frame 10 (66 bytes on wire, 66 bytes captured)
Ethernet II, Src: 02:04:06:00:00:01, Dst: 02:04:06:00:00:02
Internet Protocol, Src: 192.168.1.1 (192.168.1.1), Dst: 192.168.1.2 (192.168.1.2)
Transmission Control Protocol, Src Port: m1 (2657), Dst Port: http (80), Seq: 216, Ack: 298, Len: 0
    Source port: m1 (2657)
    Destination port: http (80)
    Sequence number: 216    (relative sequence number)
    Acknowledgement number: 298    (relative ack number)
    Header length: 32 bytes
    Flags: 0x11 (FIN, ACK)
    Window size: 6912 (scaled)
    Checksum: 0x1a2e [correct]
    Options: (12 bytes)
```

```
No.  Time     Source        Destination  Protocol Info
11 0.087886   192.168.1.2   192.168.1.1    TCP     http > m1 [ACK] Seq=298 Ack=217 ...
Frame 11 (66 bytes on wire, 66 bytes captured)
Ethernet II, Src: 02:04:06:00:00:02, Dst: 02:04:06:00:00:01
Internet Protocol, Src: 192.168.1.2 (192.168.1.2), Dst: 192.168.1.1 (192.168.1.1)
Transmission Control Protocol, Src Port: http (80), Dst Port: m1 (2657), Seq: 298, Ack: 217, Len: 0
    Source port: http (80)
    Destination port: m1 (2657)
    Sequence number: 298    (relative sequence number)
    Acknowledgement number: 217    (relative ack number)
    Header length: 32 bytes
    Flags: 0x10 (ACK)
    Window size: 6864 (scaled)
    Checksum: 0x1a3e [correct]
    Options: (12 bytes)
    [SEQ/ACK analysis]
```

Fiabilité des connexions TCP

Exemple de connexion TCP client-serveur

La commande `netcat` est une utilitaire Unix qui permet de transmettre (par défaut) et recevoir (avec l'option *listen* `-l`) des données (lignes de texte) à travers une connexion réseau, en utilisant TCP (défaut) ou UDP. Lire la page de manuel Unix de `netcat`, puis lancer sur m_2 un processus `netcat` qui permettra de recevoir la requête d'un client HTTP lancé sur m_1 (sans cependant y répondre) :

```
m2#  netcat -l -p 80
m1#  lynx http//192.168.1.2
```

Vous devez apercevoir dans la fenêtre de m_2 la réception, de la part de `netcat`, de la requête HTTP. Bien entendu, `netcat` n'étant pas un serveur HTTP, il ne répondra pas à la question posée, laissant le client HTTP `lynx` en attente. À présent, que se passe-t-il si l'on force la terminaison du processus `netcat` (avec, par exemple, un CTRL-C sur le terminal de m_2) pendant cette attente ? Capturez les trames avec `wireshark` pour savoir si `netcat` a la "politesse" d'envoyer un datagramme de fin de connexion TCP.

Fiabilité (contrôle du transport) d'une connexion TCP

Utilisez `netcat` en tant que client su m_1 :

```
m1#  netcat 192.168.1.2 80
```

En simulant des **dysfonctionnements** du câble croisé (par l'onglet `Anomalies` de `marionnet`, en particulier le taux de perte des paquets), essayez de provoquer la retransmission de quelques segments TCP en observant notamment l'évolution des numéros de séquence.

Essayez de comprendre jusqu'à quel taux de perte des paquets (20% ?, 30% ?, 40%,..) la communication aura lieu malgré tout.