NFA009 - TP 1

LE MODÈLE OSI (1/2/3/4) / ANALYSE DE TRAMES
ETUDE des PROTOCOLES utilisés par
ARP-PING-TRACEROUTE-DNS
ETUDE du CLIENT-SERVEUR IPERF3

A - Le Modèle OSI

1. Indiquer ce qu'on appelle une "pile" réseau.
2. Donner les composants essentiels de la pile OSI/ISO. Indiquer pour chacune des couches leurs fonctionnalités.
3. Donner les composants essentiels de la pile internet.
4. Pourquoi appelle-t-on cela une pile ?

Réponses :

Une pile est une suite de couches et de protocoles qui permettent de transmettre des messages entre machines informatiques.

Pour le pile OSI/ISO, les composants sont, du plus bas au plus haut : physique, liaison, réseau, transport, session, présentation, application.

La couche 1 physique traite des problèmes de physiciens (fréquence, tension électrique, communication radio, ...(représentation d'un bit) => carte hardware.
La couche 2 liaison structure la couche physique, codifie les informations, gère le transfert de ces messages (trame = frame) => carte hardware.
La couche 3 réseau gère le routage c'est à dire les chemins à suivre pour arriver à destination (paquet, datagramme) => inclus dans votre système.
La couche 4 transport gère la communication de bout en bout, éventuellement les pertes et le bon ordre d'arrivée des messages (message) => inclus dans votre système.
La couche 5 session gère une suite de messages qui appartiennent à une même discussion => inclus dans votre système.
La couche 6 présentation gère le format des données => inclus dans votre système.
La couche 7 application est la couche qui fournit les applications (ex clients, serveurs, …) des utilisateurs.

Pour la pile internet, les composants sont, du plus bas au plus haut :
- physique et liaison sont traités par des protocoles comme Ethernet, Wi-Fi, Token Ring, ...
- réseau est traité par le protocole IP (Internet Protocol)
- transport est traité par TCP (Transmisson Control Protocol) ou UDP (User Datagram Protocol). TCP contrôle le bon acheminement des parties de messages. UDP ne fait pas de contrôles.
- session, présentation, application sont souvent regroupées en une seule appelée couche application.

Ce sont des piles, car tout élément fournit un service à son élément supérieur et utilise les services de son élément inférieur.

B - Utilisation du logiciel wireshark de capture et d'analyse de paquets réseau (analyseur de protocoles/trames)

1. Quels sont les usages de ce logiciel ?
2. Expliquer la différence entre les capture_filters et les display_filters ?
3. Capturer 1000 trames sur le réseau.
4. Quelles seraient les @sources et @destinations reçues par votre analyseur si vous n'utilisiez pas de filtres ?
    i. Si les équipements réseau de niveau 2 de la topologie du réseau du CNAM sont des hubs.
    ii. Si les équipements réseau de niveau 2 de la topologie du réseau du CNAM sont des switchs.
5. Au regard des trames obtenues, à votre avis quels types d'équipements sont utilisés au CNAM ?
6. Quelle est la valeur du filtre de capture que vous devez utiliser pour ne capturer que les flux échangés avec une autre machine ?

Réponses :

1. debug, pédagogique, exploratoire, statistique
2. Wireshark propose 2 types de filtres l'un au moment de la capture, l'autre au moment de l'affichage. Plus précisément :
    a. les filtres de capture sont les filtres qui sélectionnent les données à enregistrer dans les journaux. Ils sont définis au démarrage de la capture,
    b. les filtres d'affichage sont utilisés pour rechercher à l'intérieur des données capturées. Ils peuvent être modifiés pendant que des données sont capturées.
    c. Ainsi un filtre d'affichage est utilisé pour rechercher à l'intérieur des données récoltées avec un filtre de capture.
3. 80% de Broadcast, 19 % de Multicast, quelques @unicast dont la celle de la machine
4. Hub/Switch
    a. Hub : Toutes @MAC des machines connectées sur mon LAN, @MAC broadcast, @MAC Multicast,
    b. Switch : Mon @MAC, @MAC broadcast, @MAC Multicast, les (quelques) @MAC inconnues par le switch (si @MAC inconnue => diffusion sur tous les ports)
5. Switch
6. « Ether host @votre_MAC or Ether host @MAC_du_voisin »

C - En utilisant l'analyseur wireshark ; étude du protocole ARP (Address Resolution Protocol) et des autres protocoles utilisés par la commande ping

Pour communiquer sur internet (le WAN), on utilise les adresses IP. Ce sont ces adresses IP qui sont utilisées par les routeurs pour envoyer, envoyer par exemple, un message de France en Australie.
Lorsque le message arrive dans une réseau local (le LAN) contenant votre ordinateur, on utilise les adresses MAC (Media Access Control). Par exemple le protocole Ethernet utilise les adresses MAC.

1. Expliquer comment fonctionne le protocole Ethernet.

Indication : C'est un protocole de la forme CSMA/CD c'est-à-dire Carrier Sense Multiple Access with Collision Detection (écoute de porteuse avec accès multiples et détection de collision)

2. Indiquer le rôle joué par une adresse MAC. Peut-il y avoir des ordinateurs sur la planète qui ont la même adresse MAC ?
3. Il va donc falloir traduire les adresses IP en adresse MAC. Le protocole qui le fait est le protocole ARP (Address Resolution Protocol) (pour la version IPv4). Indiquer dans quelle couche réseau est situé le protocole ARP (bon sens).
4. Quelles sont les @IP et les @MAC de votre machine (=> commenter le résultat de la commande « /sbin/ifconfig ».
5. Indiquer comment fonctionne le protocole ARP (c'est-à-dire indiquer comment dans un sous réseau, ce protocole s'y prend pour trouver l'adresse MAC d'un ordinateur (ou périphérique) ayant une adresse IP)

Réponses :

Un machine dite « Ethernet » écoute le médium (câble coaxial, fibre optique, paire torsadée,..). Si il est disponible, elle transmet. Si tout c'est bien passé le transfert est fait. Si une autre machine à décider de transférer au même moment, il a une collision et les machines émettrices, qui écoute le support en sont informées. Elles attendent alors un temps aléatoire (pris dans une plage de plus en plus grande) pour recommencer l'algorithme à son début.

Le terme, « d'adresse MAC » est en fait un identificateur d'un élément informatique réseau. De ce fait deux éléments informatiques réseau distincts doivent avoir des adresses MAC distinctes, car il risque de se retrouver sur le même site .Une adresse MAC et sur 6 octets décrit par une suite de 6 chiffres hexadécimaux séparés par : (linux) ou – (windows)

ARP est évidemment un protocole qui prend des informations de couche réseau (@IP) et les transforment en information de la couche liaison (@MAC) donc est un protocole de couche liaison.

Sur la machine lire01 =>
/sbin/ifconfig -a
em1 Link encap:Ethernet HWaddr C8:1F:66:B4:78:3D
inet adr:163.173.230.81 Bcast:163.173.231.255 Masque:255.255.252.0
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:38641 errors:0 dropped:112 overruns:0 frame:0
TX packets:489 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 lg file transmission:1000
RX bytes:4973898 (4.7 Mb) TX bytes:85468 (83.4 Kb)
Interruption:20 Mémoire:f7c00000-f7c20000

lo Link encap:Boucle locale
inet adr:127.0.0.1 Masque:255.0.0.0
UP LOOPBACK RUNNING MTU:65536 Metric:1
RX packets:2 errors:0 dropped:0 overruns:0 frame:0
TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 lg file transmission:0
RX bytes:100 (100.0 b) TX bytes:100 (100.0 b)

=> Interface em1 de Lire01 = son interface Ethernet
Lire02 @MAC Ethernet = C8:1F:66:B4:78:3D
Lire02 @IP= 163.173.230.81
Lire02 Masque = 255.255.252.0 => /22
Lire02 Broadcast IP = 163.173.231.255

Sur la machine lire02 =>
/sbin/ifconfig -a
em1 Link encap:Ethernet HWaddr C8:1F:66:B4:45:D9
inet adr:163.173.230.82 Bcast: 163.173.231.255 Masque:255.255.252.0
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:163101 errors:0 dropped:0 overruns:0 frame:0
TX packets:19982 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 lg file transmission:1000
RX bytes:59048162 (56.3 Mb) TX bytes:2535183 (2.4 Mb)
Interruption:20 Mémoire:f7c00000-f7c20000

lo Link encap:Boucle locale
inet adr:127.0.0.1 Masque:255.0.0.0
UP LOOPBACK RUNNING MTU:65536 Metric:1
RX packets:12 errors:0 dropped:0 overruns:0 frame:0
TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 lg file transmission:0
RX bytes:600 (600.0 b) TX bytes:600 (600.0 b)

=> Interface em1 de Lire02 = son interface Ethernet
Lire02 @MAC Ethernet = C8:1F:66:B4:45:D9
Lire02 @IP= 163.173.230.82
Lire02 Masque = 255.255.252.0 => /22
Lire02 Broadcast IP = 163.173.231.255

Le premier message ARP est diffusion (broadcast) sur tout le sous réseau et demande "quel est le numéro MAC de l'organe informatique d'adresse IPv4 XXX.YYY.ZZZ.TTT ? ". La réponse provient de l'organe qui a cette adresse IP. C'est un message à destination de demandeur (unicast) qui indique son adresse MAC. Par la suite les échanges entre les deux machines se feront à l'aide de cette adresse MAC.

6. Capturer que les trames liées à la commande "ping -c 1 @IP_du_voisin"

Réponses :

```
lire01:~ # ping -c 1 163.173.230.82
PING 163.173.230.82 (163.173.230.82) 56(84) bytes of data.
64 bytes from 163.173.230.82: icmp_seq=1 ttl=64 time=0.403 ms
```

Sur Lire02 installation du Filtre 'ether host C8:1F:66:B4:78:3D ' dans wireshark

Puis taper la commande 'ping -c 1 163.173.230.82' surLire01
4 Trames reçues ou émises sur Lire02 (frames) :

1. ARP , c8:1f:66:b4:78:3d (lire01) , Dst: Broadcast (ff:ff:ff:ff:ff:ff)
   ARP request => Who is 163.173.230.82 ?
2. ARP, c8:1f:66:b4:45:d9 (lire02), Dst: c8:1f:66:b4:78:3d (lire01)
   ARP reply => Je suis 163.173.230.82 (lire02)
3. IPv4, Src: 163.173.230.81 (lire01), Dst: 163.173.230.82 (lire02)
   ICMP (Echo request) => envoyer une serie d'octets
4. IPv4, Src: 163.173.230.82 (lire02), Dst: 163.173.230.81 (lire01) ICMP
   ICMP (Echo reply) => retourner la serie d'octet

Trame 1 => broadcast pour de lire01 sur le réseau LAN pour connaître l'@MAC de lire02
Trame 2 => lire02 dit c'est moi
Trame 3 : lire01 envoie une série d'octet différent a lire02
« 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 36 37 »
Trame 4 : lire02 répond en envoyant la même série (echo)

Lire01 évalue le temps d'aller-retour (RTT) de la cette série
=> **time=0.403 ms**

Détails :

```
Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
Ethernet II, Src: Dell_b4:78:3d (c8:1f:66:b4:78:3d), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: Dell_b4:78:3d (c8:1f:66:b4:78:3d)
    Sender IP address: 163.173.230.81 (163.173.230.81)
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 163.173.230.82 (163.173.230.82)
```



```
Frame 2: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
Ethernet II, Src: Dell_b4:45:d9 (c8:1f:66:b4:45:d9), Dst: Dell_b4:78:3d (c8:1f:66:b4:78:3d)
Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: Dell_b4:45:d9 (c8:1f:66:b4:45:d9)
    Sender IP address: 163.173.230.82 (163.173.230.82)
    Target MAC address: Dell_b4:78:3d (c8:1f:66:b4:78:3d)
    Target IP address: 163.173.230.81 (163.173.230.81)

Frame 3: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
Ethernet II, Src: Dell_b4:78:3d (c8:1f:66:b4:78:3d), Dst: Dell_b4:45:d9 (c8:1f:66:b4:45:d9)
Internet Protocol Version 4, Src: 163.173.230.81 (163.173.230.81), Dst: 163.173.230.82
(163.173.230.82)
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x7af1 [correct]
    Identifier (BE): 2842 (0x0b1a)
    Identifier (LE): 6667 (0x1a0b)
    Sequence number (BE): 1 (0x0001)
    Sequence number (LE): 256 (0x0100)
    Timestamp from icmp data: Apr  5, 2016 10:11:53.000000000 CEST
    [Timestamp from icmp data (relative): -25.752731000 seconds]
    Data (48 bytes)

0000  64 56 02 00 00 00 00 00 10 11 12 13 14 15 16 17   dV..............
0010  18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 26 27   ........ !"#$%&'
0020  28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 36 37   ()*+,-./01234567
      Data: 6456020000000000101112131415161718191a1b1c1d1e1f...
      [Length: 48]

Frame 4: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
Ethernet II, Src: Dell_b4:45:d9 (c8:1f:66:b4:45:d9), Dst: Dell_b4:78:3d (c8:1f:66:b4:78:3d)
Internet Protocol Version 4, Src: 163.173.230.82 (163.173.230.82), Dst: 163.173.230.81
(163.173.230.81)
Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0x82f1 [correct]
    Identifier (BE): 2842 (0x0b1a)
    Identifier (LE): 6667 (0x1a0b)
    Sequence number (BE): 1 (0x0001)
    Sequence number (LE): 256 (0x0100)
    [Request frame: 7]
    [Response time: 0,011 ms]
    Timestamp from icmp data: Apr  5, 2016 10:11:53.000000000 CEST
```

[Timestamp from icmp data (relative): -25.752720000 seconds]
Data (48 bytes)

```
0000  64 56 02 00 00 00 00 00 10 11 12 13 14 15 16 17   dV..............
0010  18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 26 27   ........ !"#$%&'
0020  28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 36 37   ()*+,-./01234567
      Data: 64560200000000001011121314151617181 91a1b1c1d1e1f...
      [Length: 48]
```

7. Que fait cette commande ? et expliquer les résultats obtenus lors de l'exécution de cette commande.

Trame 1 => broadcast pour de lire01 sur le réseau LAN pour connaître l'@MAC de lire02
Trame 2 => lire02 dit c'est moi
Trame 3 : lire01 envoie une série d'octet différent a lire02
« 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 36 37 »
Trame 4 : lire02 répond en envoyant la même série (echo)

Lire01 évalue le temps d'aller-retour (RTT) de la cette série d'octet.

⇨ **time=0.403 ms**

8. Quel est le nouveau protocole utilisé dans cette capture et son rôle.

La commande ping permettant d'envoyer une requête **ICMP** Echo (request) d'un ordinateur à un autre pour tester si cet ordinateur hôte est accessible par le réseau et le temps mis pour recevoir un ICMP Echo (reply) du message envoyé vers cet hôte.

9. Pourquoi le protocole ARP n'est pas utilisé lors de chaque ping

La machine à l'origine de la requête ARP reçoit la réponse et met à jour son cache ARP et peut donc envoyer à l'ordinateur concerné le message qu'elle avait mis en attente. Ce cache sera utilisé par des nouvelles commandes ping lancées. Les entrées dans ce cache ARP ont une durée de vie limitée (quelques minutes). Ainsi, quand une entrée de ce cache vient à expiration, une nouvelle requête ARP devra être initiée.

10. Capturer que les trames liées à la commande "ping -c 1 -s 8000 @IP_d'un_autre_voisin ".

```
lire01:~ # ping -c 1 -s 8000 163.173.230.82
PING 163.173.230.82 (163.173.230.82) 8000(8028) bytes of data.
8008 bytes from 163.173.230.82: icmp_seq=1 ttl=64 time=1.08 ms
```

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	163.173.230.81	163.173.230.82	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=1403) [Reassembled in #6]
2	0.000017000	163.173.230.81	163.173.230.82	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=1403) [Reassembled in #6]
3	0.000019000	163.173.230.81	163.173.230.82	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=2960, ID=1403) [Reassembled in #6]
4	0.000021000	163.173.230.81	163.173.230.82	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=4440, ID=1403) [Reassembled in #6]
5	0.000023000	163.173.230.81	163.173.230.82	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=5920, ID=1403) [Reassembled in #6]
6	0.000025000	163.173.230.81	163.173.230.82	ICMP	642	Echo (ping) request id=0x0eb4, seq=1/256, ttl=64 (reply in 12)
7	0.000092000	163.173.230.82	163.173.230.81	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=0, ID=4240) [Reassembled in #12]
8	0.000097000	163.173.230.82	163.173.230.81	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=4240) [Reassembled in #12]
9	0.000099000	163.173.230.82	163.173.230.81	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=2960, ID=4240) [Reassembled in #12]
10	0.000101000	163.173.230.82	163.173.230.81	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=4440, ID=4240) [Reassembled in #12]
11	0.000127000	163.173.230.82	163.173.230.81	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, off=5920, ID=4240) [Reassembled in #12]
12	0.000129000	163.173.230.82	163.173.230.81	ICMP	642	Echo (ping) reply id=0x0eb4, seq=1/256, ttl=64 (request in 6)

11. Expliquer les différences entre les 2 séquences de trames obtenues ? Pourquoi ce nouvel échange a-t-il nécessité de la fragmentation ?

La taille du datagramme IP (message) échangé est supérieure à taille de la MTU (1480 = 1500 – 20 (taille de l'entête IP)). **Une fragmentation datagramme est donc nécessaire**, lors de l'envoie, en **paquet de taille maximale de 1480 octets**. **Un ré-assemblage des fragments du datagramme IP sera effectué** sur la machine hôte.

Soit un datagramme IP de 8008 octets (dont entête ICMP=16) + 20 (entête IP) = 8028 octets
Fragmenté en 6 paquets = (5x1480+628) = 8028
Remarques :
Il faut rajouter 14 pour l'entête Ethernet, pour retrouver les tailles de la **colonne length** de wireshark
La machine retourne aussi des datagrammes fragmentés, car sa taille de MTU est aussi = 1500
Le temps d'aller-retour est supérieur de l'échange est supérieur au précédent (**time=1.08 ms**), car le nombre d'octets transféré est **x125 (2x 8028 = 2x 125 x 64)**

12. Comment une machine peut-elle "usurper l'identité" d'une autre en utilisant ce protocole ? (Proposez des parades)

Lors de l'échange utilisant le protocole ARP entre une machine A et une machine B. Une autre machine C pourrait usurper l'identité de celle de B, en répondant au broadcast ARP initial émis par la machine A.

13. Capturer les trames liées à la commande "ping -c 1 www.google.fr"

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	163.173.230.82	163.173.128.6	DNS	73	Standard query 0x22e8 A www.google.fr
2	0.001237000	163.173.128.6	163.173.230.82	DNS	235	Standard query response 0x22e8 A 216.58.208.195
3	0.001501000	163.173.230.82	216.58.208.195	ICMP	98	Echo (ping) request id=0x0a0c, seq=1/256, ttl=64 (reply in 6)
6	0.003218000	216.58.208.195	163.173.230.82	ICMP	98	Echo (ping) reply id=0x0a0c, seq=1/256, ttl=54 (request in 3)

14. Quelle est l'@IP du destinataire, du serveur DNS et du routeur sur le LAN ?

Destinataire = > www.google.fr: type A, class IN, addr 216.58.208.195
Serveur DNS =>163.173.128.6 (serveur DNS consulté dans la 1ere trame)
routeur sur le LAN => commande « ip route » : default via 163.173.228.2 dev em1

15. Quel est le nouveau protocole utilisé dans cette capture et son rôle.

DNS => Le Domain Name System (est un service permettant de traduire un nom de domaine en informations de plusieurs types qui y sont associés, notamment en adresses IP de la machine portant ce nom.

16. Grâce à la commande « dig » donner l'@IP de la machine liée au nom DNS www.google.fr.

```
lire01:~ $ dig www.google.fr
;;QUESTION SECTION:
      www.google.fr.              IN      A
;; ANSWER SECTION:
      www.google.fr.        58    IN      A       216.58.208.195
```

17. Que se passe-t-il lorsqu'un serveur DNS tombe en panne ou est arrêté ?

On utilise généralement que des noms de domaine. La correspondance nom_de_domaine -> @IP n'est donc plus possible => Les @IP des machines gérées par ce DNS ne sont plus accessibles sur INTERNET.

D - En utilisant l'analyseur wireshark ; étude des protocoles utilisés par la commande « traceroute »

1. Capturer que les trames liées aux commandes "traceroute –I www.elyse.fr"

```
lire02:~ # traceroute -I www.elyse.fr
traceroute to www.elyse.fr (149.202.133.35), 11 hops max, 60 byte packets
 1  lmi35-atm1.cnam.fr (163.173.228.2)  0.495 ms
 2  ursula-big.cnam.fr (163.173.9.57)  186.142 ms
 3  interco-K01-cnam.rap.prd.fr (195.221.126.249)  0.748 ms
 4  pe-jussieu.rap.prd.fr (195.221.125.18)  1.210 ms
 5  vl165-te3-2-jussieu-rtr-021.noc.renater.fr (193.51.181.102)  1.229 ms
 6  te0-1-0-3-paris2-rtr-001.noc.renater.fr (193.51.177.114)  80.321 ms
 7  *
 8  be11-1269.rbx-g1-a9.fr.eu (91.121.128.46)  5.048 ms
 9  po5.rbx-s3-6k.fr.eu (213.186.32.142)  5.636 ms
10  *
```

Source	Destination	Protocol	Time to live	Info
163.173.230.82	149.202.133.35	ICMP	1	Echo (ping) request id=0x0f60, seq=1/256, ttl=1 (no response found!)
163.173.230.82	149.202.133.35	ICMP	2	Echo (ping) request id=0x0f60, seq=2/512, ttl=2 (no response found!)
163.173.230.82	149.202.133.35	ICMP	3	Echo (ping) request id=0x0f60, seq=3/768, ttl=3 (no response found!)
163.173.230.82	149.202.133.35	ICMP	4	Echo (ping) request id=0x0f60, seq=4/1024, ttl=4 (no response found!)
163.173.230.82	149.202.133.35	ICMP	5	Echo (ping) request id=0x0f60, seq=5/1280, ttl=5 (no response found!)
163.173.230.82	149.202.133.35	ICMP	6	Echo (ping) request id=0x0f60, seq=6/1536, ttl=6 (no response found!)
163.173.230.82	149.202.133.35	ICMP	7	Echo (ping) request id=0x0f60, seq=7/1792, ttl=7 (no response found!)
163.173.230.82	149.202.133.35	ICMP	8	Echo (ping) request id=0x0f60, seq=8/2048, ttl=8 (no response found!)
163.173.230.82	149.202.133.35	ICMP	9	Echo (ping) request id=0x0f60, seq=9/2304, ttl=9 (no response found!)
163.173.230.82	149.202.133.35	ICMP	10	Echo (ping) request id=0x0f60, seq=10/2560, ttl=10 (no response found!)
163.173.230.82	149.202.133.35	ICMP	11	Echo (ping) request id=0x0f60, seq=11/2816, ttl=11 (no response found!)
163.173.228.2	163.173.230.82	ICMP	255,1	Time-to-live exceeded (Time to live exceeded in transit)
195.221.126.249	163.173.230.82	ICMP	252,1	Time-to-live exceeded (Time to live exceeded in transit)
163.173.9.57	163.173.230.82	ICMP	253,1	Time-to-live exceeded (Time to live exceeded in transit)
193.51.181.102	163.173.230.82	ICMP	251,1	Time-to-live exceeded (Time to live exceeded in transit)
91.121.128.46	163.173.230.82	ICMP	248,1	Time-to-live exceeded (Time to live exceeded in transit)
213.186.32.142	163.173.230.82	ICMP	245,1	Time-to-live exceeded (Time to live exceeded in transit)
195.221.125.18	163.173.230.82	ICMP	252,1	Time-to-live exceeded (Time to live exceeded in transit)
193.51.177.114	163.173.230.82	ICMP	250,1	Time-to-live exceeded (Time to live exceeded in transit)

2. Que fait cette commande ? Et expliquer les résultats obtenus lors de l'exécution de cette commande.

Cette commande fait 1 ping sur chacun des routeurs utilisés pour communiquer, ici 11 routeurs : (

- lmi35-atm1.cnam.fr, ursula-big.cnam.fr, interco-K01-cnam.rap.prd.fr,
- pe-jussieu.rap.prd.fr, vl165-te3-2-jussieu-rtr-021.noc.renater.fr,
- te0-1-0-3-paris2-rtr-001.noc.renater.fr, 7 *, be11-1269.rbx-g1-a9.fr.eu,
- po5.rbx-s3-6k.fr.eu, * ,*)

avec une machine (ici de nom DNS www.elyse.fr qui a pour @IP 149.202.133.35).

3. Expliquer les trames obtenues lors de l'exécution de cette commande

La machine émettrice fait en 11 pings avec avec nom de domaine www.elyse.fr d' @IP 149.202.133.35 en changeant le ttl a chaque instance du ping (ttl=1 puis 2 …11). Lorsque la trame ICMP traverse un routeur, son ttl est décrémenté de 1 ; si ce ttl est égal 0; le routeur concerné retourne un message 'ttl exceeded in transit' à la machine émettrice, qui est ainsi informée des @IP des routeurs traversés.

4. Pourquoi certains temps obtenus sont représentés par une '*' ?

Certains routeurs (ici : '*') ont été configurés pour ne pas répondre au ping.

E - Étude du client-serveur iperf3 et des couches de transports TCP et UDP

URL : http://software.es.net/iperf

MANUEL : http://software.es.net/iperf/invoking.html#iperf3-manual-page

Exemple usage serveur tcp et udp =>

$ **iperf3 –s –p 5004**

Exemple usage client tcp =>

$ **iperf3 –c @IP_du_serveur_iperf3 –p 5004**

Exemple usage client udp =>

$ **iperf3 –u –c @IP_du_serveur_iperf3 -b 1000M –p 5004**

L'interface réseau « lo » (boucle locale) permet d'échanger des messages entre 2 applications sur la même machine SANS passer par la carte Ethernet.

Vérification => utiliser la commande : 'ifconfig lo'

Le transport est traité des messages est par le protocole TCP (Transmisson Control Protocol) ou le protocole UDP (User Datagram Protocol). TCP contrôle le bon acheminement des parties de messages. UDP ne fait pas de contrôles.

Échanges TCP

1. Lancer un nouveau terminal et un serveur iperf3 sur votre machine
2. Dans une autre fenêtre, lancer un client tcp iperf3

   iperf3 –c @IP_du_voisin –p 5004
3. Commenter les résultats obtenus

```
lire02:~ # iperf3 -c lire01 -p 5004
Connecting to host lire01, port 5004
[  4] local 163.173.230.82 port 55299 connected to 163.173.230.81 port 5004
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   112 MBytes   939 Mbits/sec    0    238 KBytes
[  4]   1.00-2.00   sec   111 MBytes   934 Mbits/sec    0    238 KBytes
[  4]   2.00-3.00   sec   112 MBytes   937 Mbit/sec     0    238 KBytes
[  4]   3.00-4.00   sec   111 MBytes   932 Mbits/sec    0    238 KBytes
[  4]   4.00-5.00   sec   112 MBytes   938 Mbits/sec    0    238 KBytes
```

```
[  4]   5.00-6.00   sec   111 MBytes   933 Mbits/sec    0    238 KBytes
[  4]   6.00-7.00   sec   111 MBytes   934 Mbits/sec    0    238 KBytes
[  4]   7.00-8.00   sec   111 MBytes   933 Mbits/sec    0    238 KBytes
[  4]   8.00-9.00   sec   111 MBytes   935 Mbit/sec    0    238 KBytes
[  4]   9.00-10.00  sec   111 MBytes   935 Mbit/sec    0    238 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  1.09 GBytes   935 Mbit/sec    0             sender
[  4]   0.00-10.00  sec  1.09 GBytes   934 Mbits/sec                 receiver
```

- ⇨ Sur la machine lire01, une application serveur (couche 7) tcp sur le port = 5004
- ⇨ Sur la machine lire02, une application cliente (couche 7) tcp se connectant sur le port 5004 de la machine lire01.
- ⇨ L'application client évalue le débit tcp entre lire01 et lire02 => 935 Mbit/sec
- ⇨ swich gigabit et adapteur Ethernet gigabit sur chaque machine

4. Dans une autre fenêtre , lancer un client tcp iperf3

   iperf3 –c localhost –p 5004

5. Commenter les nouveaux résultats obtenus

```
lire01:~ # iperf3 -c localhost -p 5004
Connecting to host localhost, port 5004
[  4] local 127.0.0.1 port 48557 connected to 127.0.0.1 port 5004
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec  7.54 GBytes  64.7 Gbits/sec    0   1.19 MBytes
[  4]   1.00-2.00   sec  7.84 GBytes  67.3 Gbits/sec    0   1.19 MBytes
[  4]   2.00-3.00   sec  7.83 GBytes  67.3 Gbits/sec    0   1.19 MBytes
[  4]   3.00-4.00   sec  7.75 GBytes  66.6 Gbits/sec    0   1.19 MBytes
[  4]   4.00-5.00   sec  7.84 GBytes  67.4 Gbits/sec    0   1.19 MBytes
[  4]   5.00-6.00   sec  7.82 GBytes  67.1 Gbits/sec    0   1.19 MBytes
[  4]   6.00-7.00   sec  7.68 GBytes  66.0 Gbits/sec    0   1.19 MBytes
[  4]   7.00-8.00   sec  7.86 GBytes  67.5 Gbits/sec    0   1.19 MBytes
[  4]   8.00-9.00   sec  7.82 GBytes  67.1 Gbits/sec    0   1.19 MBytes
[  4]   9.00-10.00  sec  7.71 GBytes  66.2 Gbits/sec    0   1.19 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  77.7 GBytes  66.7 Gbits/sec    0             sender
[  4]   0.00-10.00  sec  77.7 GBytes  66.7 Gbits/sec                  receiver
```

- ⇨ Sur la machine lire01, une application serveur (couche 7) tcp sur le port = 5004
- ⇨ Sur la machine lire01, une application cliente (couche 7) tcp se connectant sur le port 5004 de la machine localhost => lire01.
- ⇨ L'application client évalue le débit tcp entre lire01 et lire01 => 66.2 Gbits/sec
- ⇨ Échanges INTERNES (système) entre le client et le serveur sur la MEME machine => on n'utilise PAS le swich gigabit et l'adapteur Ethernet gigabit de la machine

Échanges UDP

1. Dans une autre fenêtre, lancer un client udp iperf3

   iperf3 –u –c @IP_du_serveur_iperf3_udp -b 1000M –p 5004

2. Commenter les résultats obtenus

```
lire02:~ # iperf3 -u -c lire01 -b 1000M  -p 5004
Connecting to host lire01, port 5004
[  4] local 163.173.230.82 port 60560 connected to 163.173.230.81 port 5004
[ ID] Interval           Transfer     Bandwidth       Total Datagrams
[  4]   0.00-1.00   sec   102 MBytes   856 Mbits/sec  13055
[  4]   1.00-2.00   sec   113 MBytes   951 Mbits/sec  14509
[  4]   2.00-3.00   sec   113 MBytes   951 Mbits/sec  14507
[  4]   3.00-4.00   sec   113 MBytes   951 Mbits/sec  14508
[  4]   4.00-5.00   sec   113 MBytes   951 Mbits/sec  14508
[  4]   5.00-6.00   sec   113 MBytes   951 Mbits/sec  14508
[  4]   6.00-7.00   sec   113 MBytes   951 Mbits/sec  14510
[  4]   7.00-8.00   sec   113 MBytes   951 Mbits/sec  14508
[  4]   8.00-9.00   sec   113 MBytes   951 Mbits/sec  14508
[  4]   9.00-10.00  sec   113 MBytes   951 Mbits/sec  14508
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  4]   0.00-10.00  sec  1.10 GBytes   941 Mbits/sec  0.114 ms  0/143627 (0%)
[  4] Sent 143627 datagrams
```

- ⇨ Sur la machine lire01, une application serveur (couche 7) tcp et UDP sur le port = 5004
- ⇨ Sur la machine lire02, une application cliente (couche 7) UDP se connectant sur le port 5004 de la machine lire01.
- ⇨ L'application cliente évalue le débit UDP entre lire01 et lire02 => 951 Mbit/sec
- ⇨ Débit supérieur au protocole TCP s'expliquant, car le protocole UDP est sans connexion préalable à l'envoi des données et sans garantie de livraison des datagrammes à destination, ni leur ordre d'arrivée.
- ⇨ swich gigabit et adapteur Ethernet gigabit sur chaque machine
- ⇨ pas de perte au niveau adapteur Ethernet et swich (Lost=0%)
- ⇨

3. Dessiner la pile réseau utilisée pour le serveur et les clients

   Machine Serveur iperf3 :

   Couche 7 **Serveur iperf3 avec son protocole**
   Couche 6 vide
   Couche 5 vide
   Couche **UDP port 5004 et TCP port 5004**
   Couche 3 IP (@IP de lire01 : **163.173.230.81**)
   Couche 2 Ethernet Gigabit (@MAC lire01 : **c8:1f:66:b4:78:3d**)
   Couche 1 **Paire torsadée UTP-5e**

Machine Cliente iperf3 (lire02) :

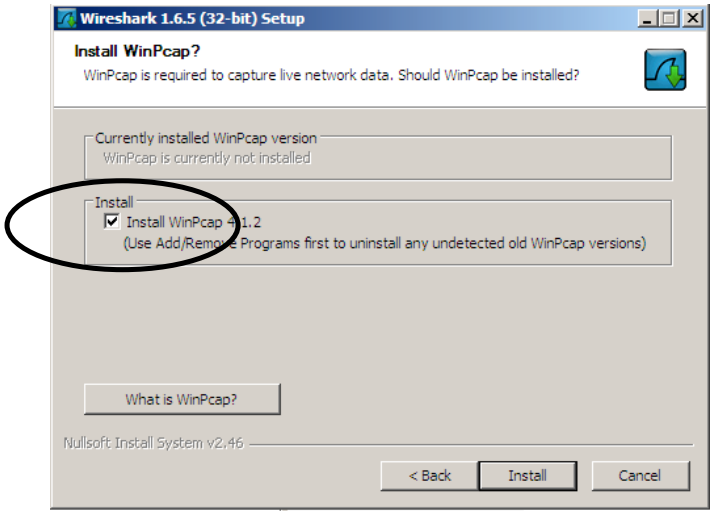
Couche 7 **Client iperf3 avec son protocole**
Couche 6 vide
Couche 5 vide
Couche **UDP port 5004 ou TCP port 5004**
Couche 3 IP (@IP de lire02 : **163.173.230.82**)
Couche 2 Ethernet Gigabit (@MAC : **C8:1F:66:B4:45:D9**)
Couche 1 **Paire torsadée UTP-5e**

INSTALLATION WIRESHARK SUR WINDOWS

Wireshark est un analyseur de protocole réseau. Il permet de visualiser et de capturer les trames, les paquets de différents protocoles réseau, filaires ou pas.
Le site originel est à `http://www.wireshark.org/`. À partir de ce site, on peut lire un tutoriel à `http://www.wireshark.org/docs/wsug_html_chunked/`.
On peut télécharger pour l'installer sur diverses plates-formes (Windows, Linux, Unix et donc Mac OS) à partir de `http://www.wireshark.org/download.html`. Une FAQ est disponible à `http://www.wireshark.org/faq.html`.

Depuis le 10 septembre 2013, la dernière version stable est Wireshark 1.10.2. C'est un produit open source et gratuit.
Il a été écrit par Gerald Combs à partir de 1997. C'est la suite du produit Ethereal (son ancien nom).

On utilise l'environnement Windows 7. Pour ce TP, Wirehark est installé sur les machines. Si vous voulez l'installer sur votre propre machine, c'est possible. Au moment de l'installation, bien cocher la case d'installation de la dll WinPCap :



"The experience capturing your first packets can range from "it simply works" to "very strange problems"." Si problème voir à
`http://wiki.wireshark.org/CaptureSetup`.

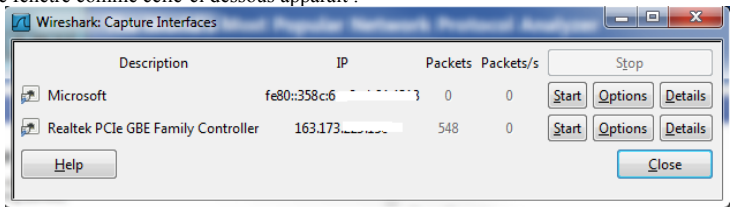
Première approche de Wireshark

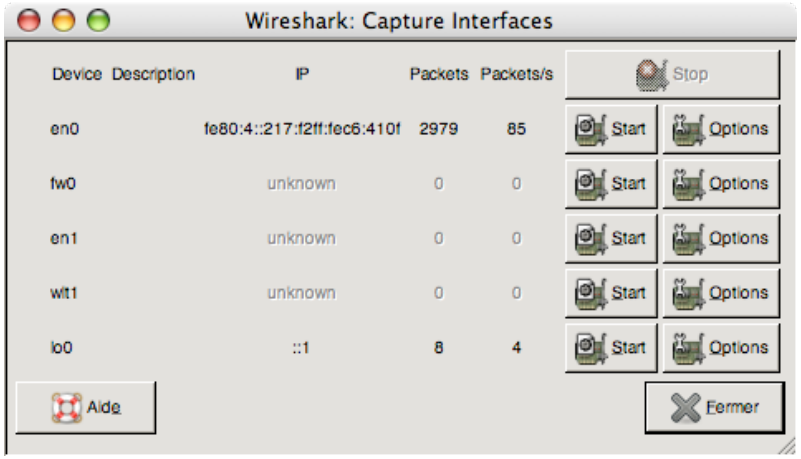
1°) Lancer Wireshark (double clic sur l'icône 

sur le bureau).
La fenêtre

apparaît.

2°) Sélectionner Capture | Interfaces... (ou cliquer sur Interface List ci-dessus).
Une fenêtre comme celle-ci dessous apparaît :





3°) Repérez une entrée ayant du trafic réseau. Cliquer sur son bouton Start. Vous devriez voir une fenêtre similaire à :