TP 1 — Fonctionnement de ARP et ICMP, fragmentation IP

Le TP est à faire en binôme. Même si le TP n'est pas noté, chaque étudiant préparera un compte-rendu à étudier avant l'examen final. On appellera ethX l'interface initialement connectée à Internet, ethY l'autre.

L'objectif de ce TP est de créer un petit réseau puis étudier le fonctionnement des protocoles ARP et ICMP. Le TP montre également comment la fragmentation IP fonctionne.

Rappels :

— Vous utiliserez l'image `mageia4`. Vous vous connecterez en utilisant le compte suivant :
identifiant : `etudiant`
mot de passe : `etudiant`

— La plupart des commandes que nous utiliserons ne peuvent être exécutées que par l'administrateur du système.
commande permettant d'ouvrir une session sous l'identité root : `su -`
mot de passe : `iutparis13`

— Dans tout le TP on travaillera uniquement sur l'interface réseau ethY. L'autre interface, ethX, restera connectée au réseau de l'IUT.

Contrairement à ce qui est fait dans le module R101, nous allons utiliser la commande `ip` afin de configurer le réseau statiquement. Cela comprend l'attribution des adresses aux interfaces, ainsi que le remplissage des tables de routage. Pour le fonctionnement avancé de `ip`, voir le site du cours :

`https://lipn.univ-paris13.fr/~petrucci/R102/`

Exercice 1 — L'interface ethX

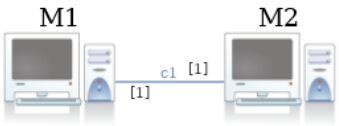
L'interface ethX des deux machines a été configurée par le serveur DHCP du réseau. Nous allons analyser sa configuration à l'aide de la commande `ip`.

Q. 1.1 Afficher l'adresse MAC de l'interface ethX de la machine. Spécifier si l'adresse obtenue est une adresse individuelle ou de groupe. Spécifier s'il s'agit d'une adresse universelle ou locale. Si possible, donner le code du constructeur.

Q. 1.2 Afficher l'adresse IP de l'interface ethX. Spécifier la classe et le masque associé.

Exercice 2 — Câblage et configuration des machines

Connecter les interfaces ethY des deux machines avec un câble (croisé) pour réaliser le réseau suivant :



Q. 2.1 Attribuer l'adresse MAC `00:00:00:aa:aa:aa` à l'interface ethY de M1, et `00:00:00:bb:bb:bb` à l'interface ethY de M2. Vérifier que le changement a eu lieu.

Q. 2.2 Le MTU (Maximum Transmission Unit) désigne la taille maximale de champ données d'une trame Ethernet envoyé par l'interface. Quelle est la valeur par défaut ? Fixer cette valeur à 1 000 octets pour l'interface ethY.

Q. 2.3 On souhaite attribuer à ce réseau l'adresse IP 10.0.0.0. Quelle est la classe de cette adresse ? Quel est le masque réseau associé ? Choisir comme adresses pour les interfaces ethY des deux machines M1, M2 les premières adresses disponibles. Activer ensuite ethY.

Exercice 3 — Étude du protocole ARP

Exécuter en tâche de fond l'analyseur de trames `wireshark` de sorte à capturer le trafic engendrés par l'interface ethY des deux machines. À l'aide de la commande `ip -s neigh flush all`, vider la table ARP de M1 et M2.

Q. 3.1 Exécuter sur M1 la commande `ping -c1 10.0.0.2`. Quel est le trafic observé ? Expliquer.

Q. 3.2 Sélectionner une trame et donner les PCI de niveau Ethernet, en faisant apparaître le nom et la taille de chaque champ. Quelles sont les informations manquantes ?
Q. 3.3 Quelle est l'adresse MAC de destination de la trame contenant ARP-Request ? Justifier.
Q. 3.4 Dessiner la structure d'une trame contenant un message ARP.
Q. 3.5 Sur les deux machines M1 et M2 afficher le contenu des tables ARP. Expliquer.
Q. 3.6 Exécuter la même commande ping sur M1 et relever à nouveau le trafic. Justifier le trafic observé.
Q. 3.7 Comparer la trame ARP envoyée et celle reçue. Seulement l'en-tête Ethernet de la deuxième contient du bourrage, pourquoi ?
Q. 3.8 Une trame de taille inférieure à 72 octets peut-elle effectivement circuler dans le réseau ?

Exercice 4 — Étude de l'encapsulation

Dans cet exercice, nous allons analyser le mécanisme d'encapsulation, tel qu'il est implanté dans la suite de protocoles TCP/IP. **Attention à la terminologie :** message ICMP $\neq$ paquet IP $\neq$ trame Ethernet.

Q. 4.1 Quelle est la structure d'un message ICMP ?
Q. 4.2 Quel protocole encapsule le protocole ICMP ? Quel protocole encapsule ARP ? Justifier.
Q. 4.3 Relever la structure d'un paquet IP.
Q. 4.4 Illustrer avec un dessin, la trame Ethernet qui véhicule un message ICMP.

Exercice 5 — Étude de la fragmentation / assemblage

Sur `wireshark`, dans l'onglet `Edit -> Preferences -> Protocols -> IPv4`, décochez le paramètre `Reassemble fragmented IPv4 datagram`.

Q. 5.1 Sur M1 exécuter la commande suivante `ping -c1 -s 1000 10.0.0.2`. Capturer et expliquer le trafic induit.
Q. 5.2 Combien de fragments sont générés ?
Q. 5.3 Pour l'ensemble des fragments identifiés, justifier les valeurs des champs : DF, MF, OFFSET.
Q. 5.4 Comment peut-on identifier qu'un paquet IP reçu est un paquet entier et non fragmenté ?

TP 2 — Routage et ICMP

Le TP est à faire par groupe de 3. Chaque étudiant préparera un compte-rendu individuel, à reviser avant l'examen. On appellera ethX l'interface initialement connectée à Internet, ethY l'autre.

L'objectif de ce TP est de réaliser le réseau de la figure 1 composé de trois machines. R jouera le rôle de routeur entre M1 et M2. Les interfaces et les tables de routage seront configurées statiquement à l'aide de la commande `ip`.

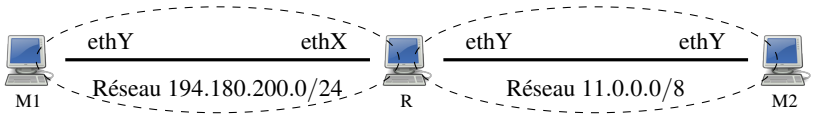


FIGURE 1 – Le réseau à réaliser

Une fois que le réseau a été configuré correctement, nous allons étudier plus en détail le protocole ICMP, permettant de signaler les erreurs de fonctionnement du protocole IP.

Exercice 1 — Configuration statique des interfaces

Q. 1.1 *Câblage.* Réaliser le câblage de la figure 1. Quel type de câble faut-il utiliser pour connecter deux ordinateurs ?

Q. 1.2 Désactiver l'interface ethX des trois machines.

Q. 1.3 Proposer et réaliser le plan d'adressage à appliquer pour le montage demandé. On attribuera aux machines les premières adresses IP disponibles sur le réseau. Par convention, le routeur aura la dernière adresse disponible.

Q. 1.4 *Test de la connexion.* En utilisant la commande `ping`, vérifier que les deux connexions (M1↔R et M2↔R) sont actives.

Exercice 2 — Routage statique

Q. 2.1 *Configuration de R comme routeur.* La commande `sysctl` permet de visualiser ou de modifier certains paramètres réseau (et d'autres). Voici deux façons de l'utiliser :

— pour voir la valeur d'un paramètre : `sysctl <nom-du-parametre>`

— pour modifier la valeur d'un paramètre : `sysctl <nom-du-parametre>=<valeur>`

Le paramètre `net.ipv4.ip_forward` indique ce que la machine doit faire lorsqu'elle reçoit un paquet qui ne lui est pas destiné. S'il vaut 1 alors elle essaiera de le router. S'il vaut 0 le paquet sera ignoré. Changer ce paramètre afin que R accepte de router les paquets.

Q. 2.2 *Test de la connexion.* Est-ce que la machine M1 peut pinguer M2 ? Pourquoi ?

Q. 2.3 *Ajout de routes.* Proposer et ajouter dans la table de routage de M1 une route qui lui permette d'envoyer des paquets à M2. Réaliser l'opération symétrique sur M2. À titre de rappel :

Ajout d'une règle de routage	`ip route add [destination] via [gateway] dev [interface]`
Ajout d'une règle par défaut	`ip route add default via [gateway] dev [interface]`
Effacer une règle de routage	`ip route delete [destination] dev [interface]`
Vider la table de routage	`ip route flush`

Q. 2.4 *Test de la connexion.* Vérifier que M1 est maintenant accessible depuis M2 et vice versa.

Exercice 3 — Internet Control Message Protocol

Le protocole ICMP permet de signaler des erreurs de fonctionnement du protocole IP. Parmi les messages d'erreurs prévus par ICMP on trouve les messages suivants :

1. Destination unreachable (Host unreachable) : envoyé par un routeur si la machine destination est introuvable.
2. Destination unreachable (Network unreachable) : envoyé par un routeur si le réseau destination est introuvable.
3. Time to live exceeded : envoyé par un routeur si le champ TTL d'un paquet IP a expiré (a la valeur 0).
4. Destination unreachable (Fragmentation needed) : envoyé par un routeur si un paquet IP nécessite fragmentation mais la fragmentation n'est pas autorisée (drapeau DF positionné à 1).

Q. 3.1 Proposer des scénarios afin que R génère un exemple de chaque message d'erreur cité ci-haut.
Q. 3.2 Sur la machine M1, exécuter la commande `traceroute 11.255.255.254`. Quel est le message ICMP engendré ?

TP 3 — Scapy

Le TP est à faire individuellement. Chaque étudiant préparera un compte-rendu à étudier avant l'examen final.

Ce TP est à faire en utilisant l'image `BOSC-Debian-11`. Choisir comme identifiant / mdp : `student` / `toto`.

L'objectif de ce TP est de se familiariser avec l'outil `scapy` : un outil de forge de paquets réseaux fréquemment utilisé pour l'audit de securité des réseaux. Les outils à utiliser sont : `scapy`, `python` et `wireshark`. Commencer par installer les logiciels nécessaires :

```
sudo apt update
sudo apt-get install python3-scapy
sudo apt install wireshark
```

Dans un interpréteur Python (exécuté en mode `root`) importer le paquetage `scapy.all`

```
# sudo python3
Python 2.7.13 (default, Sep 26 2018, 18:42:22)
[GCC 6.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from scapy.all import *
```

Exercice 1 — Prise en main de Scapy

Q. 1.1 Relever la configuration par défaut des PDU générés par Scapy pour les protocoles suivants : ARP, IP, ICMP, TCP.

Q. 1.2 Donner une instruction qui permette d'envoyer le paquet ICMP généré par défaut.

Q. 1.3 Écrire un script `python+scapy` qui implémente une simple opération de `ping`. Le script accepte un seul argument : l'adresse de destination. Il consiste à envoyer un seul paquet ICMP `Echo Request`. Si la machine cible répond avec un paquet ICMP `Echo Reply` alors le script affiche le message `destination up` sinon il affiche `destination unreachable`.

Q. 1.4 Modifier le script précédent de sorte à utiliser l'opération time stamp request pour réaliser le `ping` (`Time stamp request` est le type 13, la réponse est le type 14).

Q. 1.5 Développer et tester un script `python+scapy` qui permette de tracer la route vers une machine destination (indication : augmenter le TTL d'un paquet ICMP `Echo Request` progressivement)

Exercice 2 — À l'attaque!

— Le **ping de la mort** est une attaque historique de type « déni de service » réalisé par l'envoi de paquets `ping` malformés dont la taille dépasse la taille maximale autorisée qui est de 65 535 octets. Certains systèmes n'étaient pas en mesure de traiter correctement de tels paquets et pouvant provoquer un crash de la machine cible.

— Une autre attaque connue est l'attaque **Tear Drops** qui consiste à envoyer un ensemble de fragments IP avec des informations erronées qui impliquent un chevauchement dans les offsets

Q. 2.1 Développer et tester un script `python+scapy` qui permette de réaliser l'attaque « ping de la mort ».

Q. 2.2 Développer et tester un script python/scapy qui permet de réaliser une attaque de type « Tear drops » :

Annexe 1 — Scapy — Description générale

`Scapy` est une bibliothèque de manipulation de paquets réseaux écrite en Python. Elle permet notamment de :

— Forger des PDU des principaux protocoles : Ethernet, IP, ICMP, ARP, UDP, TCP, etc. L'extension pour représenter d'autres PDU est possible. L'outil donne accès à tous les champs de données de contrôle d'un PDU permettant ainsi de générer des PDU (corrects ou erronés).
— Capturer et analyser du trafic réseau.
— Écrire des scripts permettant de générer, capturer et analyser des grandes quantités de PDU. Ceci est souvent utile pour mettre en place des attaques réseau sophistiquées. Ces scripts peuvent servir lors des études d'audit de la sécurité d'un système.

Annexe 2 — Scapy — mini guide d'utilisation

— La génération d'un PDU passe par l'invocation d'un constructeur du protocole demandé. Par exemple : `ip =IP()` génère un paquet IP avec la configuration par défaut.
— La commande `ls()` liste l'ensemble des protocoles supportés par `scapy`.
— La commande `ls(PDU())` montre les champs disponibles d'un PDU (par exemple `ls(IP())` donne les champs d'un paquet IP).
— L'accès à un champ d'un PDU passe par la notation objet usuelle ou a lieu lors de la construction. Exemple :

```
ip = IP()
ip.ttl = 70

ip = IP(ttl=70)
```

— La méthode `show` permet d'afficher la configuration actuelle d'un PDU.

```
ip = IP()
ip.ttl = 70
ip.show()

###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 70
  proto     = ip
  chksum    = None
  src       = 127.0.0.1
  dst       = 127.0.0.1
  \ options \
```

— L'encapsulation se fait par l'opérateur /. Exemple : `IP()/ICMP()`
— `send(pkts, inter=0, loop=0, verbose=None)` permet d'envoyer un paquet au niveau 3, `inter` est le temps entre deux paquets et `loop` permet d'envoyer les paquets en boucle si la valeur est différente de 0.
— `sendp` est l'équivalent de send mais au niveau 2.
— `sr`, `stp`, `sr1`, `srp1` sont des primitives d'envoi et de réception de réponses aux niveaux 3 et 2. Les primitives avec 1 donnent seulement le premier paquet réçu comme réponse. En cas d'absence de réponse, le résultat est `None`.
— `srloop` : pour envoyer et recevoir en boucle.
— `sniff` permet de capturer des paquets.

Annexe 3 — Python — Écrire un script

Un script Python est un simple fichier texte dont, par convention, l'extension est .py. Par exemple, on pourra créér un fichier `test.py` et l'exécuter en utilisant la commande `python3 test.py`.

On peut aussi passer des arguments à la commande d'exécution du script avec, par exemple, la commande `python3 test.py input`. Ici, `input` est un argument supplémentaire qui sera utilisé par le script `test.py`. On peut récupérer ce genre d'argument directement depuis le code avec le module `sys` et son attribut `argv`. Cet attribut est une liste, son deuxième élément contiendra "input". Exemple :

```
import sys

# Lecture de l'argument passé à la ligne de commande
inputfile = sys.argv[1]

print(inputfile)
```