TP Réseau n°1

Table des matières

*Un compte-rendu au format texte (**UTF-8**) devra être rédigé et envoyé à l'adresse*
tvaira@free.fr
*La convention de nommage pour les compte-rendus est la suivante : **tp-1-nom.txt***

Manipulations

Objectifs

– prendre en main le fonctionnement de netkit
– observer la communication entre deux machines dans un même réseau local

Remarque : il est conseillé de lire la FAQ Netkit.

Mise en situation

1. **Solution n°1** : Vous devez disposer d'un PC possédant une distribution Linux (sur une partition spécifique, sur une clé USB bootable, sur un Live CD ou encore à l'aide d'un logiciel de virtualisation du type *VMware* ou *VirtualBox*). Le logiciel de virtualisation **Netkit** doit être installé sur la machine Linux ainsi que le programme `uml_dump`. Évidemment, le logiciel `wireshark` doit être installé sur votre système.

   – **Site de NetKit :** www.netkit.org
   – **Site pour `uml_dump` :** kartoch.msi.unilim.fr

2. **Solution n°2** : utilisez un **Live CD/DVD/USB Netkit**. Vous pouvez aussi utiliser l'image ISO à l'aide d'un logiciel de virtualisation du type *VMware* ou *VirtualBox*.

   – **Site du Netkit live DVD/USB :** tocai.dia.uniroma3.it
   – **Site du Netkit4TIC live DVD :** tocai.dia.uniroma3.it
   – **Site du Live CD Raizo :** www.utec-tic.org

Installation du TP

Le TP1 est disponible dans l'archive : `/home/user/Memos/tp/tp1.tgz`

```
host> cd /home/user
host> mkdir tp
host> cd tp
host> tar zxvf ../Memos/tp/tp1.tgz
host> cd tp1
```

Remarques
– le prompt `host>` indique que la commande doit être tapée dans le terminal de votre machine réelle (par opposition aux terminaux ouverts par les machines virtuelles).
– le prompt `name :~#` représente le terminal de la machine virtuelle name.

Démarrage des machines virtuelles

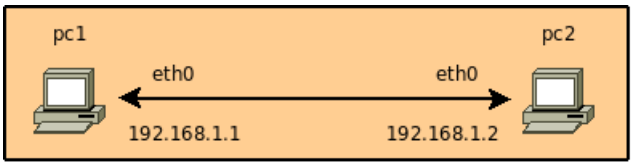
Démarrer le premier tp en lançant la commande `lstart` dans le répertoire du premier lab :

```
host> cd /home/user/tp/tp1
host> lstart -s
```

Dans chaque machine, vous possédez une interface réseau :

```
pc1 :~# ifconfig eth0 up
pc1 :~# ifconfig eth0
```

Les deux machines vont se partager la même plage IP : **192.168.1.0/24**
- pc1 : 192.168.1.1
- pc2 : 192.168.1.2



Pour configurer l'interface `eth0` de chaque machine, on utilise la commande `ifconfig` en précisant l'adresse de diffusion générale (*broadcast*) et le masque réseau (*netmask*).

```
pc1 :~# ifconfig eth0 192.168.1.1 netmask 255.255.255.0 broadcast 192.168.1.255
pc2 :~# ifconfig eth0 192.168.1.2/24
```

Pour voir si les machines peuvent communiquer, la commande `ping` permet de faire un test de communication :

```
pc1 :~# ping 192.168.1.2 -c 1
```

pc1 envoie un message ICMP de type *echo request* auquel la machine **pc2** va répondre par un message ICMP *echo reply*.

Capture

Pour connecter `wireshark` (qui permet de sniffer un réseau) avec le domaine de collision **A** du lab, il faut taper :

```
host> vdump A | wireshark -i - -k
```


Travail demandé

Objectifs

– Configurer les paramètres IP d'une machine sous Linux
– Étudier le fonctionnement des protocoles de la couche réseau (IP, ICMP, ARP)
– Utiliser les commandes de configuration IP et de test (ping, traceroute)

Sujet

Question 1. Sur un réseau **192.168.1.0/24** :

a) Quelle est la plage d'adresse **IP** disponible ?

b) Quelle est sa classe d'adresse ?

c) Justifier la valeur du masque.

Question 2. Relever l'adresse **MAC** de vos deux cartes réseau et indiquer la commande que vous avez utilisée.

Activer une capture `wireshark`.

Question 3. Donner la commande pour tester une communication depuis **pc2** sur **pc1**.

Question 4. Quel est le protocole de niveau réseau utilisé par la commande ? Donner alors la pile de protocoles mise en oeuvre par cette commande.

Question 5. Quelle est la valeur du champ *Protocol* de l'en-tête **IP** ?

Question 6. Quels sont le type et le code des requêtes et réponses échangées par le protocole de plus haut niveau ?

Question 7. Justifier la valeur du *TTL* et du *RTT*.

Question 8. Relever les adresses **IP** source et destination ainsi que les adresses **MAC** source et destination.

Question 9. Vider le cache `arp` de votre machine par une commande `arp -d`. Puis exécuter une commande `ping` depuis **pc1** sur **pc2**.

Question 10. Pourquoi visualisez-vous des trames **ARP** sur votre capture ?

Question 11. Observer l'adresse **MAC** de destination d'une requête et d'une réponse **ARP**. Expliquez leurs valeurs.

Question 12. Quel est maintenant le contenu du cache **ARP** des postes **pc1** et **pc2** ?

Question 13. Éditer le fichier de configuration du poste **pc2** et ajouter la configuration **192.168.1.3** pour `eth0`. Donner la syntaxe exacte des lignes que vous avez ajoutées.

Vous pouvez utiliser l'éditeur de texte `vim`. Pour insérer du texte, il faut taper **i**. Pour sortir du mode édition, appuyez sur **Echap** puis taper `:wq` pour enregistrer et sortir de l'éditeur.

Question 14. Relancer le service réseau en donnant la commande que vous avez utilisé. Vérifier que la nouvelle adresse a bien été prise en compte. Donner la commande.

Question 15. Observer les valeurs des registres **TX/RX** avec la commande `ifconfig`, avant et après avoir exécuté la commande `ping localhost`. Par quelle interface passent les paquets émis par la commande `ping localhost` ? Quel est son nom ?

Question 16. Tester l'état de connexion vers une adresse **IP** inexistante de votre réseau. Puis, vers une adresse **IP** inexistante d'un autre réseau. Donner les commandes et commenter les réponses obtenues.

Question 17. Donner la syntaxe de la commande `ping` qui permet d'envoyer à tous les postes de votre réseau depuis **pc1**. Comment se nomme cette technique ? Fonctionne-t-elle ?

Les options concernant le réseau et donc le protocole **ICMP** sont accessibles depuis le répertoire `/proc/sys/net/` :

```
# find /proc/sys -name icmp | grep ipv4
/proc/sys/net/ipv4/icmp_echo_ignore_all
/proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
/proc/sys/net/ipv4/icmp_ignore_bogus_error_responses
/proc/sys/net/ipv4/icmp_errors_use_inbound_ifaddr
/proc/sys/net/ipv4/icmp_ratelimit
/proc/sys/net/ipv4/icmp_ratemask
```

Pour visualiser la valeur d'une option, on réalise l'opération suivante :

```
# cat /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses
1
```

Pour modifier la valeur d'une option booléenne, on fera (**0=inactif et 1=actif**) :

```
# echo "0" > /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses

# cat /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses
0
```

Question 18. Modifier l'option qui permettra d'envoyer un message **ICMP** à tous les postes de votre réseau depuis **pc1**. Donner la commande et faites la fonctionner.

Question 19. Modifier l'option qui permettra d'interdire tout message **ICMP** de type `echo` pour le poste **pc2**. Donner la commande et vérifier son efficacité.

Question 20. Exécuter une commande `traceroute` depuis **pc1** vers **pc2**. D'après la capture, quels sont les protocoles encapsulant les requêtes de la commande ? Commenter la valeur du *TTL* dans cet échange. Quelle est le nom de la commande équivalente sous Windows ? Sachant que la commande sous Windows utilise le protocole **ICMP**, pourquoi la commande `traceroute` de Linux a-t-elle plus de chance de recevoir une réponse que la commande de Windows ?