TP-5 Application

Objectifs pédagogiques

Se familiariser avec les principaux protocoles d'application en utilisant les commandes basiques réseau de linux et l'outil `wireshark`.

1 Consignes

General : Lisez TOUT l'énoncé avant de taper une seule ligne de code.

Binômes : Vous pouvez travailler en binôme ou seul.

Ressources : Vous pouvez utiliser les matériels du cours (slides de CM, TPs passés, ...) ainsi que toute ressource disponible sur Internet. `<nom_auteur_1>_<nom_auteur_2>.pdf`.

2 DNS

2.1 Configuration local d'une machine UNIX

Le réseau n'utilise que les adresses IP dans les échanges sur Internet. Il faut donc un service qui permette de faire la correspondance entre adresse IP et nom et vice-versa. Ce service, appelé *résolveur*, est implicitement appelé par les applications chaque fois que cela est nécessaire (p. ex. par votre navigateur lorsque vous surfez sur l'internet). Ce *résolveur* commence sur UNIX ou Linux par lire le fichier `/etc/nsswitch.conf` pour savoir qui peut fournir la réponse et dans quel ordre s'il y a plusieurs possibilités. Par example, parmi ces différentes possibilités il y a :

- un fichier local appelé `/etc/hosts` qui permet de faire une résolution locale
- les serveurs de nom spécialisés (DNS)
- ou, les serveurs non spécialisés (NIS par exemple pour une résolution locale à un réseau).

Dans une machine UNIX, comme la vôtre, la procédure suivie par le *résolveur* pour obtenir la résolution de nom directe est la suivante :

(1) Le *résolveur* consulte le fichier `/etc/nsswitch.conf` afin de connaitre qui peut fournir la réponse et dans quel ordre. P. ex., on assume que le contenu du fichier est :

```
hosts : files, dns, nis, ldap
```

(2) En suivant l'exemple, le *résolveur* consulte d'abord le fichier local `/etc/hosts`. Ce fichier donne la correspondance entre des noms canoniques et des adresses IP (locales). P. ex. il peut contenir entre autres les lignes suivantes :

```
5.6.7.8 machine3
1.2.3.4 machine1
```

(3) Puis, en cas d'échec, le *résolveur* fait appel au serveur de noms en suivant les consignes de recherche données dans le fichier `/etc/resolv.conf`, qui fournit les adresses de serveurs dns (jusqu'à 6). P. ex.

```
domain domain1.dom
search domain1.dom domain2.dom
nameserver 10.10.10.10
```

Question 1 : *Vérifiez la configuration de votre machine et faites un schéma représentant l'enchainement d ?interrogation des différents fichiers et entités de votre machines lorsque celle-ci veut faire une résolution de nom. Quel est le contenu du fichier `/etc/resolv.conf` et la signification de chaque ligne ?*

Application → Résolveur → `/etc/nsswitch.conf` → `/etc/hosts` → `/etc/resolv.conf` → serveur local DNS 134.59.2.6

```
domain unice.fr # nom du domaine local
search unice.fr # nom du domaine de auto-complétion
nameserver 134.59.2.6 # IP du serveur dns
nameserver 134.59.1.7 # IP du serveur dns
nameserver 134.59.1.1 # IP du serveur dns
```

2.2 Commandes `nslookup` et `dig`

Les commandes `nslookup` et `dig` sur une machine UNIX permettent de générer des requêtes DNS afin d'obtenir l'adresse IP correspondant à un nom. Vous pouvez trouver des descriptions détaillées sur son usage sur https://linux.die.net/man/1/nslookup, http://www.madboa.com/geek/dig et sur https://linux.die.net/man/1/dig.

Question 2 : *Ouvrez une fenêtre de terminal et tapez `nslookup www.unice.fr`. Quelle est l'adresse IP du serveur web de l'université ? Et l'adresse IP du serveur DNS qui a répondu ?*

Adresse IP du serveur web de l'université : 134.59.204.9
Adresse IP du serveur DNS qui a répondu : 134.59.2.6

Question 3 : *Examinons maintenant une réponse plus détaillée avec la option `debug` : `nslookup -debug www.unice.fr`. Expliquez les différentes parties de la réponse.*

Il y a 3 parties : (i) cordonnées du serveur DNS qui répond (@IP#53), (ii) le registre RR demandé (la requête du registre (QUESTIONS) et le réponses correspondante (ANSWERS) ainsi que des registres *AUTHORITY RECORDS et ADDITIONAL RECORDS* si nécessaire, (iii) la resolution demandée (l'adresse IP demandée) avec l'indication si la réponse est une *authoritative answer* si elle procède d'un serveur qui fait autorité pour le domaine *unice.fr*

Maintenant, on va utiliser la commande `dig`. Par défaut elle utilise le mode récursif de résolution. Donc, vous aller forcer l'utilisation du mode itératif de résolution (avec la option `+trace`) dans la résolution de nom directe du serveur web de l'université `www.unice.fr`.

ATTENTION : En cas d'échec, vous devrez probablement sélectionner le serveur de nom à interroger pour forcer les requêtes itératives. Une bonne option est de demander à un serveur publique DNS de Google (à savoir 8.8.8.8 ou 8.8.4.4, voir https://developers.google.com/speed/public-dns/. Vous pouvez le fournir en ajoutant p.ex. `@8.8.4.4` à l'appel de la commande.

Question 4 : *Faites la résolution de nom directe du serveur web de l'université `www.unice.fr` en mode itératif. Quels sont et dans quel ordre les serveurs sollicités ? Quels sont les domaines gérés par chacun des serveurs ? Faites un schéma résumant l'enchaînement des opérations.*

`dig +trace www.unice.fr. @8.8.8.8`
Une solution possible c'est de solliciter les serveurs dns dans l'ordre suivant :
8.8.8.8 → i.root-servers.net → e.ext.nic.fr → taloa.unice.fr

3 Web

3.1 Création des requêtes HTTP

Dans cette section on va utiliser des commandes `telnet` ou `nc` (*netcat*) pour ouvrir des connexions TCP vers un serveur web et construire "à la main" des .requêtes HTTP. On appelle les deux commandes de une

manière similaire : `telnet nom.serveur.com 80` ou `nc nom.serveur.com 80`. Vous pouvez utiliser l'un ou l'autre indistinctement. Vos trouverez plus de détails sur les commandes sur https://linux.die.net/man/1/telnet et sur https://linux.die.net/man/1/nc.

Question 5 : *Dans l'URL http://www.i3s.unice.fr/~raparicio/teaching/intr2netw/labs/Part1/index.html, quelle partie correspond au serveur et quelle partie correspond au chemin de l'objet sur le serveur ?*

Partie qui correspond au serveur : `http://www.i3s.unice.fr/`.
Partie qui correspond au chemin de l'objet sur le serveur :
`/~raparicio/teaching/intr2netw/labs/Part1/index.html`

Question 6 : *Connectez-vous à l'aide de `telnet` ou `nc` sur le serveur. Une fois que le canal TCP est établi, dans un autre terminal, entrez la commande `netstat -punt` (cf. https://linux.die.net/man/8/netstat) afin de visualiser les connexions réseau (sockets) actives par les protocoles (`tcp`/`udp`). Normalement, il y aura 7 colonnes. Explique la signification de chacune pour la connexion TCP que vous venez d'ouvrir.*

Si la connexion TCP ne s'est pas fermée entre temps, les sept colonnes indiquent :

1. **(1) Proto : TCP → Protocole de transport (TCP ou UDP)**
2. **(2) Recv-Q : 0 → Données en attentes d'être lues dans la file de réception du socket par l'application connectée sur le socket.**
3. **(3) Send-Q : 0 → Données dans la file d'envoi du socket en attente d'être acquittées par la machine distante.**
4. **(4) Local Address → Adresse et port utilisés par la machine ayant lancé la commande netstat.**
5. **(5) Foreign Address : 134.59.136.6 : 80 → Adresse et port utilisés par la machine distante à laquelle on se connecte, dans notre cas, le serveur web.**
6. **(6) State : ESTABLISHED → Etat de la connexion TCP.**
7. **(7) PID/Programe name :2208/telnet → Identifiant de processus et nom de programme ayant lancé le socket.**

Question 7 : *Il y a des fortes chances que la connexion TCP vers le serveur web se soit fermée entre-temps. Si c'est le cas, reconnectez-vous à l'aide de `telnet` ou `nc` sur le serveur. Une fois que le canal TCP est établi, entrez dans le console `telnet` ou `nc` une commande `http` GET afin de récupérer l'URL avec une connexion HTTP non persistante. Quelle est la syntaxe exacte de la commande que vous devez taper ?*

> *ATTENTION : Dans la plupart des protocoles de l'IETF, le retour à la ligne s'effectue avec la séquence `"\r\n"` alors que les systèmes unix/linux utilisent simplement le caractère `"\n"`. Si vous ne respectez pas cette convention, le serveur ne traitera pas correctement vos requêtes.*

```
telnet www.i3s.unice.fr 80
GET /~raparicio/teaching/intr2netw/labs/Part1/index.html HTTP/1.0 \r\n
host: www.i3s.unice.fr \r\n
\r\n
```

Question 8 : *Vous obtenez une réponse HTTP à la requête que vous venez de taper. Quelle partie de la réponse correspond aux informations de contrôle (en-tête) et quelle partie correspond aux données ? Quel est l'état de la réponse et que signifie-t-il ?*

Informations de contrôle :
Données :

L'état de réponse est 200 et il signifie OK.

Question 9 : *Créez une requête GET qui est syntaxiquement correcte, mais elle demande un objet non disponible (autre resource que* `index.html`*). Quel est le statut de réponse donné par le serveur et qu'est-ce que cela signifie ?*

```
telnet www.i3s.unice.fr 80
GET /~raparicio/teaching/intr2netw/labs/Part1/index1.html HTTP/1.0 \r\n
host: www.i3s.unice.fr \r\n
\r\n
```

L'état de réponse est 404 et il signifie que la resource n'est pas disponible dans le url indiqué.

Question 10 : *Créez une requête correcte (syntaxe correcte et un objet existant) dans HTTP / 1.1 et ajoutez la ligne suivante à l'en-tête :* `connection: keep-alive \r\n` *Quels sont les changements dans l'en-tête de la réponse HTTP renvoyé et dans la connexion TCP ?*

```
telnet www.i3s.unice.fr 80
GET /~raparicio/teaching/intr2netw/labs/Part1/index.html HTTP/1.1 \r\n
host: www.i3s.unice.fr \r\n
connection: keep-alive \r\n
\r\n
```

Dans l'en-tête de la réponse, le champ `Connection: close \r\n` de la réponse HTTP dans la question 10 est remplacé avec le champ `Connection: keep-alive \r\n`.
En plus, un nouveau champ `Keep-Alive: timeout=15, max=100` est rajouté.
Maintenant, on utilise le mode de connexion HTTP persistent, c.-à.-d., la connexion TCP ouverte par `telnet` n'est pas fermée immédiatement après l'envoi de la requête HTTP

Question 11 : *Maintenant, reentrez la commande* `netstat -punt` *(cf.* `https://linux.die.net/man/8/netstat`*) afin de visualiser à nouveau la connexion TCP. Est-ce que la connexion est encore ouverte ?*

La connexion TCP ouverte par `telnet` sur le port 80 du serveur web `niouze.i3s.unice` est encore ouverte puisque on a utilisé le mode de connexion HTTP persistent dans la dernière requête HTTP.

3.2 Analyse des échanges HTTP

Désormais, on va utiliser les outils du navigateur Mozilla Firefox pour étudier des échanges HTTP. En particulier, allez dans *Firefox* → *Outils de développement* → *Onglet Réseau*.

Question 12 : *Tapez la url* `http://www.i3s.unice.fr/~raparicio/teaching/intr2netw/labs/Part2/index.html` *dans le navigateur. Dans l'onglet* Réseau*, quel type des requêtes HTTP s'affichent et combien ? (Vous devrez probablement effacer l'historique de navigation la première fois.) Si vous rafraîchissez la même url au moyen du bouton (sans effacer l'historique cette fois-là), s'affichent-elles les mêmes requêtes HTTP ? Pourquoi ? Qu'est-ce qui s'est passé entre les deux fois que vous avez tapé la url ?*

La première fois, deux échanges HTTP avec des requêtes de type GET et des réponses avec code 200 s'affichent : le premier échange pour récupérer le `index.html`, le seconde pour récupérer l'image `iut.gif`. On observe que suite à la deuxième requête les 5.41 KB de l'image sont en effet transférés dans la deuxième réponse. La deuxième fois, on a aussi deux requêtes de type GET pour récupérer, d'abord, le `index.html` ; et, ensuite, l'image `iut.gif`. Mais, maintenant, on reçoit des codes 304 `Not modified` dans les réponses et l'image n'est pas transféré. Ce qui se passe c'est que le page web demandée (`index.html` et l'image `iut.gif`) n'a pas subi des modifications entre les deux

fois, et, en conséquence, l'image n'est pas transféré depuis le serveur web afin de réduire l'envoi des données. L'image `iut.gif` qui s'affiche dans la page web c'est un fichier qui a été mis en cache du navigateur après la premiere fois qu'on a demandé la page.

Question 13 : *Encore une fois, à l'aide de l'onglet Réseau dans les outils de développement de Firefox, comparez les opérations de* uploading *effectuées par les URL suivantes :* `http://www.i3s.unice.fr/~raparicio/teaching/intr2netw/labs/Part3/index.html` *et* `http://www.i3s.unice.fr/~raparicio/teaching/intr2netw/labs/Part3/index2.html`*. Est-ce que les deux urls utilisent les memes méthodes HTTP ? Quelles sont les différences ?*

No, les deux urls utilisent des méthodes différentes : `http://www.i3s.unice.fr/~raparicio/teaching/intr2netw/labs/Part3/index.html` utilise la méthode POST (données envoyées vers le page web dans le corps de la requête), et `http://www.i3s.unice.fr/~raparicio/teaching/intr2netw/labs/Part3/index2.html` utilise la méthode GET - procédé URL (données envoyées vers le page web dans le champ de la l'URL spécifié)

.

ATTENTION : Videz l'onglet réseau une fois que la page a été téléchargée afin de voir uniquement la dernière opération effectuée lorsque vous cliquez sur `Submit query`

4 Analyse d'un trace d'une session web

Ouvrez la trace `http_espn.pcap` avec Wireshark (cf. `https://www.wireshark.org/docs/wsug_html/`). Vous pouvez la trouver sur le site du cours. Il enregistre le chargement d'une seule page Web.

Question 14 : *Quels sont les protocoles de transport que vous avez observés dans la trace ? Utilisez la fonction* `protocol hierarchy` *du menu* `statistics`*. Combien de conversations avez-vous sur les protocoles IP, TCP et UDP dans la trace ? Utilisez la fonction* `conversations` *du menu* `statistics`*.*

On a les deux protocoles de transport : TCP et UDP. On a 14 conversations IP, 25 conversations TCP et 14 conversations UDP.

Question 15 : *Concentrons nous sur le trafic DNS. Créez un filtre pour filtrer uniquement les requêtes DNS. Pour faire cela, positionnez le curseur sur l'entrée des commandes* `Filter` *dans la partie supérieure. Tapez le nom de filtre (p. ex.* `dns` *pour le traffic DNS). Puis cliquez avec le bouton droit de la souris sur* `Apply`*. Combien des requêtes d'un registre de type A trouvez-vous ? Maintenant, on considère les requêtes HTTP. Utilisez l'option* `HTTP` → `Requests` *du menu* `statistics`*. Interpréter le résultat et expliquer la raison pour laquelle il est conforme aux requêtes DNS effectuées par le client.*

Il y a 14 requêtes DNS d'un registre de type A (resolution directe DNS). En utilisant l'option `HTTP` → `Requests` du menu `statistics`, on observe qu'on peut classer les requêtes HTTP en 14 groupes par hôte HTTP. Donc, ces deux résultats indique que le client essai d'accéder aux 14 sites web, ce qui implique trouver d'abord leur 14 adresses IP correspondantes par le bias de 14 échanges DNS.

Question 16 : *Revenez sur la fonction* `conversations` *du menu* `statistics`*. En utilisant uniquement des renseignements fournis par cette fonction, identifiez (i) les adresses MAC et IP de la machine locale où la trace a été capturée, (ii) l'adresse IP du serveur dns local, et (iii) les adresses IP des serveurs webs. Justifiez vos réponses. Finalement, pourquoi il y a uniquement deux adresses MAC impliquées dans tous les échanges ? A qui appartient l'autre adresse MAC ?*

(i) les adresses MAC et IP de la machine locale où la trace a été capturée : 00 :21 :70 :c0 :56 :f0 et 172.16.0.122.

(ii) l'adresse IP du serveur dns local : 4.2.2.1
(iii) les adresses IP des serveurs webs : 63.85.36.8, 63.85.36.9, 63.85.36.72, 66.235.139.152, 68.71.208.11, 68.71.208.113, 68.71.208.177, 68.71.209.72 , 199.181.132.250, 205.234.218.129, 205.234.218.82, 205.234.218.112 et 205.234.218.67
Il y a uniquement deux adresses MAC (l'adresse du client et une autre) impliquées dans tous les échanges. Cette autre adresse MAC (00 :26 :0b :31 :07 :33) est l'adresse du router d'access à Internet (présumablement un routeur Cisco : CiscoInc_31 :07 :33) par lequel tous les échanges entre le client et les serveurs DNS et les serveurs web passent. Comme les serveurs ne sont pas dans le même réseaux Ethernet, les paquets en provenance d'Internet sont encapsulés au niveau d'Ethernet par ce routeur.