


TP 6 Réseaux

Commandes et applications réseau (DNS, TELNET, FTP, SMTP, POP3) et simulation Nat/Pat/Firewall

C. Pain-Barre

INFO - IUT Aix-en-Provence

version du 12/4/2011

 Ce TP est à faire depuis Windows.

1 Commandes réseau

1.1 Commande netstat

Cette commande est disponible sur Unix et Windows. Elle donne des renseignements très complets sur les fonctionnalités réseau de la station, notamment sur l'état des connexions TCP, des ports (TCP ou UDP) utilisés par les applications, et bien d'autres choses. Elle admet de nombreuses options comme on peut le constater en consultant le manuel Linux.

Exercice 1 (netstat sur allegro)

 Sous Linux, la commande **netstat** se trouve dans le répertoire `/bin`

Consulter le manuel en ligne de **netstat** sur allegro en tapant :

```
$ man netstat
```


afin de répondre aux questions suivantes :

1. afficher toutes les informations disponibles
2. afficher la table de routage uniquement
3. afficher la table de routage uniquement mais en demandant d'afficher les adresses IP plutôt que les noms
4. l'état des connexions TCP uniquement (regarder le *synopsis* de la commande).

[\[Corrigé\]](#)

Exercice 2 (netstat sur windows)

Ouvrir une invite de commandes MS-DOS (l'interpréteur de commandes Windows) par le *Menu Démarrer* → *Exécuter* puis taper `cmd` et Entrée.

 Il est sûrement inutile de préciser que cette invite de commandes tourne sur le PC et non sur allegro...

Consulter l'aide de **netstat** sous Windows en tapant :

```
C:> netstat /?
```

afin de répondre aux questions suivantes :

1. afficher toutes les informations disponibles
2. afficher la table de routage uniquement
3. l'état des connexions TCP uniquement

[\[Corrigé\]](#)

2 Noms de stations et de domaine

Les adresses IP ne sont pas forcément très lisibles et sont difficiles à retenir. C'est pourquoi, il est possible d'associer à une station, un ensemble de noms qui sont plus faciles à retenir. **Cette association peut être officieuse ou officielle.** Si elle est officieuse, elle ne sera reconnue que par les stations qui ont été configurées pour cela. Si elle est officielle, elle sera reconnue par toute station pouvant faire appel à un serveur de noms (quasiment toutes les machines connectées à Internet). Mais cela nécessite une déclaration au DNS (*Domain Name Service*).

Les configurations officielle et officieuse diffèrent quelque peu dans la flexibilité du nommage. Si officieusement, on peut se laisser aller à quelques fantaisies, il n'en est pas de même officiellement.

2.1 Noms officieux

2.1.1 Sous Unix

Les noms officieux sont simplement renseignés sous Unix dans le fichier `/etc/hosts` pour les noms de stations et dans le fichier `/etc/networks` pour les noms de réseaux. Les noms déclarés dans ces fichiers ne sont connus que de la station qui les héberge. Lorsque le DNS n'existait pas encore, c'était la seule façon d'utiliser un nom d'hôte plutôt que son adresse.

Ces fichiers sont constitués de lignes commençant par une adresse IP suivie d'un ou plusieurs noms. Toutes les commandes réseaux consultent ces fichiers et remplacent lorsqu'il y a lieu les noms qui leur sont passés en arguments par les adresses correspondantes.

Le fichier `/etc/hosts` peut être modifié en utilisant la commande `hostname` (voir [noms officiels](#)).

Exercice 3 (noms officieux sur allegro)

1. Quels sont les noms d'hôtes officieusement reconnus par allegro ?
Pour le savoir, consulter simplement le fichier `/etc/hosts`.
2. Quels sont les noms de réseaux officieusement reconnus par allegro ?
Pour le savoir, consulter `/etc/networks`. S'il n'existe pas, c'est qu'il n'y en a pas.

[\[Corrigé\]](#)

2.1.2 Sous Windows

Windows a adopté l'utilisation de noms officieux à la manière d'Unix, à tel point que les fichiers utilisés portent le même nom. Ces fichiers sont `C:\WINDOWS\system32\drivers\etc\hosts` (ou sur Windows 2000, `C:\WINNT\system32\drivers\etc\hosts`) et `C:\WINDOWS\system32\drivers\etc\networks` (ou sur Windows 2000, `C:\WINNT\system32\drivers\etc\networks`).

Exercice 4 (noms officiels sur windows)

1. Quels sont les noms d'hôtes officiellement reconnus par votre PC ?
2. Quels sont les noms de réseaux officiellement reconnus par votre PC ?

[\[Corrigé\]](#)

2.2 Noms officiels

Le format du nom officiel d'un hôte est *hôte . domaine*. Dans cette écriture, *domaine* ne correspond pas forcément à un réseau. C'est en fait un nom de domaine qui est géré par une organisation. Elle est chargée de mettre à disposition un serveur de noms qui doit répondre aux requêtes provenant du réseau et concernant des caractéristiques de ce domaine : adresse IP d'un hôte de ce domaine, adresse IP du serveur de mail de ce domaine, adresse électronique du responsable de ce domaine...

Le nom d'un domaine est composé de chaînes de caractères, appelées *labels*, séparées par des points. Par exemple, **iut.univ-aix.fr** est le domaine du réseau de l'IUT. Le nom officiel (appelé *nom complètement qualifié* ou FQDN pour *Fully Qualified Domain Name*) de allegro est **allegro.iut.univ-aix.fr**. (avec un point à la fin).

2.2.1 Commande hostname

Les noms officiels d'une machine sont consultables avec la commande **hostname** sous Linux et Windows.

Exercice 5 (hostname sur allegro)



Sous Linux, la commande **hostname** se trouve dans le répertoire `/bin`

Consulter le manuel en ligne de **hostname** sur allegro en tapant :

```
$ man hostname
```

afin de déterminer les options permettant d'obtenir :

1. le nom complet de allegro
2. le nom court de allegro
3. le nom de domaine de allegro

[\[Corrigé\]](#)

Exercice 6 (hostname sur windows)

hostname sur Windows ne reconnaît pas d'option. L'utiliser pour connaître le nom de votre station.

[\[Corrigé\]](#)

2.2.2 Résolution de nom sous Unix et le fichier `nsswitch.conf`

Sous Unix, lorsqu'un nom d'hôte est utilisé, les commandes réseau utilisent généralement d'abord le fichier `/etc/hosts` avant d'effectuer une requête DNS. D'autres méthodes pour "résoudre" des noms d'hôtes peuvent être aussi utilisées (noms *netbios* ou autres systèmes de nommage). C'est le fichier `/etc/nsswitch.conf` qui indique dans quel ordre la résolution doit être faite.

i Le fichier `/etc/nsswitch.conf` provient de Sun Microsystems et de son système Solaris 2 et veut dire "aiguilleur de service de nom" (*name service switch*). Il a été adopté par la majorité des systèmes Unix, il y a déjà un moment.

Il ne sert pas qu'à la résolution de noms, car de nombreuses autres ressources du système peuvent être décentralisées (utilisateurs et mots de passe, groupes d'utilisateurs, alias de messagerie, etc.). Pour chaque ressource, `/etc/nsswitch.conf` contient une ligne qui indique comment la rechercher, par ordre de méthode de recherche.

Pour une *ressource* donnée, la ligne a la simple syntaxe suivante :

ressource : { *service* }

où *service* est le service à utiliser. Lorsqu'il y a plusieurs services, ils sont consultés de gauche à droite jusqu'à trouver une correspondance. Parmi les services possibles il y a :

- **files** : rechercher dans les fichiers locaux. Le(s) fichier(s) à consulter dépend(ent) de la ressource (`/etc/hosts` pour un nom d'hôte, `/etc/passwd` pour un utilisateur, etc.)
- **dns** : rechercher par le service DNS
- **nis** : rechercher par le service NIS (de Sun Microsystems), appelé aussi pages-jaunes (*yellow pages*)
- **ldap** : rechercher dans un annuaire par le protocole LDAP
- **winbind** : rechercher par un serveur Windows (informations utilisateur/groupe, authentification)
- **wins** : rechercher par le service de nommage Windows
- etc.

La *ressource* qui concerne les noms d'hôtes est **hosts**. En principe, pour la résolution de noms d'hôtes, c'est le fichier local `/etc/hosts` qui est préféré avant tout autre service (notamment le DNS).

Exercice 7 (consultation de `/etc/nsswitch.conf` sur allegro)

Consulter le fichier `/etc/nsswitch.conf` de allegro pour savoir comment celui-ci procède pour la résolution de noms d'hôtes.

i Pour en savoir plus sur le fichier `nsswitch.conf`, consulter le manuel de ce fichier en tapant :

```
$ man nsswitch.conf
```

[\[Corrigé\]](#)

2.2.3 Configuration DNS

Sous Unix, afin d'interroger le DNS pour reconnaître les noms officiels de toutes les stations déclarées, il faut configurer le fichier `/etc/resolv.conf`.

Ce fichier comprend un certain nombre de paramètres indiqués par des mots clés suivis d'informations. Les paramètres principaux sont :

- **domain** *domaine-local*
pour préciser le domaine de la machine. La recherche d'un nom court d'hôte (i.e. qui ne comporte pas de `.`) se fera en priorité dans ce domaine. Si ce paramètre n'existe pas, il sera déduit du résultat de **hostname** ;
- **search** *domaine { domaine }*
pour préciser une liste ordonnée de domaines dans lesquels la recherche d'un nom court d'hôte se fera. Ce paramètre est souvent utilisé à la place de **domain** en particulier quand la station n'appartient pas à un domaine particulier ;
- **nameserver** *adresse-ip*
pour indiquer un serveur de noms DNS à contacter pour la résolution. Plusieurs lignes **nameserver** peuvent être indiquées. Leur ordre d'apparition détermine la priorité du serveur : la première désigne le serveur préféré. Celui qui suit n'est utilisé que si le premier ne répond pas, etc. Il est conseillé de renseigner au moins 2 serveurs de noms, voire 3.

Pour effectuer une requête DNS, l'application la réalisant contacte le serveur de nom afin de lui demander l'adresse IP associée au nom recherché. Si le serveur ne connaît pas cette association, il demandera à un autre serveur ce renseignement.

Exercice 8 (consultation de `/etc/resolv.conf` sur `allegro`)

Consulter le fichier `/etc/resolv.conf` de `allegro` afin de répondre aux questions suivantes :

1. Dans quel(s) domaine(s) par défaut les noms courts d'hôtes sont-ils recherchés par `allegro` ?
2. Quels sont les serveurs de noms utilisés par `allegro` ?



Pour en savoir plus sur le fichier `resolv.conf`, consulter le manuel de ce fichier en tapant :

```
$ man resolv.conf
```

[\[Corrigé\]](#)

2.2.4 Interrogation du DNS avec `host`, `dig` et `nslookup`

Certaines commandes permettent d'interroger des serveurs DNS. Il s'agit notamment de **host**, **dig** et de **nslookup**. Ces commandes ne sont pas toujours disponibles sur Windows. Elles admettent un bon nombre de paramètres, notamment le serveur de nom à utiliser pour l'interrogation. Historiquement, **nslookup** était la plus utilisée. Elle a l'avantage d'être interactive.

La commande host

host est une commande non interactive qui s'utilise en ligne de commandes. Son synopsis le plus basique est le suivant.

Synopsis

```
host [-r] [-t type] nom [serveur]
```

où :

- *nom* est le nom que l'on veut résoudre. Ce peut être une adresse IP, dans ce cas, **host** effectue une **résolution inverse**
- *serveur* est le serveur de noms qui sera utilisé pour la résolution. S'il n'est pas spécifié, le serveur utilisé est celui par défaut (le premier **nameserver** de `/etc/resolv.conf`)
- **-r** désactive la recherche récursive. La recherche récursive est activée par défaut et demande au serveur de noms, si celui-ci ne sait pas résoudre le *nom* (pour le *type* de question posée), de contacter un serveur adéquat qui saura le faire (ou qui contactera lui-même un autre serveur, etc.), et d'afficher la réponse
- **-t type** précise le type d'information que l'on souhaite obtenir. Au cours de ce TP, les informations qui nous intéressent sont demandées par les *types* suivants (la casse importe peu) :
 - ◇ **A** : adresse IPv4 correspondant à *nom* (il existe aussi le type **AAAA** pour les adresses IPv6)
 - ◇ **NS** : (Name Server) serveur de domaines ayant autorité sur le domaine *nom*. Il faut dans ce cas que *nom* soit un domaine (pas un hôte)
 - ◇ **MX** : (Mail eXchanger) serveur SMTP du domaine *nom*. Il faut dans ce cas que *nom* soit un domaine (pas un hôte)
 - ◇ **ANY** : tout type d'information disponible pour ce *nom*

La commande dig

dig est une commande non interactive qui s'utilise en ligne de commandes. Son synopsis le plus basique est le suivant.

Synopsis

```
dig [@serveur] nom type
```

où :

- *serveur* est le serveur de noms à contacter
- *nom* est le nom que l'on veut résoudre
- *type* est le type de question posée (comme pour **host**)

La commande nslookup

À la différence des deux commandes précédentes, **nslookup** est une commande interactive. Son synopsis le plus basique est le suivant.

Synopsis

```
nslookup
```

En l'exécutant, un prompt s'affiche et l'on peut paramétrer le solveur de nom avec les commandes suivantes :

- **set [no] rec** pour désactiver (**norec**) ou activer (**rec**) le mode récursif. Il est activé par défaut

- **server** *serveur* pour indiquer que l'on veut interroger le serveur de noms *serveur*. Sans cette commande, le serveur utilisé est celui par défaut de l'ordinateur (voir `/etc/resolv.conf`)
- **set q=type** pour préciser le *type* de question posée. Le *type* par défaut est **A**
- **nom** provoque l'émission d'une requête DNS au serveur choisi précédemment. Elle demande les informations correspondant au *type* choisi et concernant *nom*. Si *nom* ne se termine pas par un point, il n'est pas complètement qualifié (FQDN) et si la recherche échoue pour ce *nom*, des requêtes supplémentaires seront émises en ajoutant à *nom* les domaines de recherche de la machine (lignes **domain** et **search** dans `/etc/resolv.conf`) jusqu'à obtenir une réponse
- **exit** pour quitter

Exercice 9 (interrogations DNS sur Linux)

 Sous Linux, **host**, **dig** et **nslookup** se trouvent dans le répertoire `/usr/bin`

Consulter le manuel en ligne de **host**, **dig** et **nslookup** sur allegro en tapant :

```
$ man host dig nslookup
```

puis :

1. Utiliser la commande **host** pour connaître l'adresse IP de `www.lsis.org`
2. Utiliser la commande **host** pour connaître le serveur de noms du domaine `lsis.org`
3. Utiliser **dig** pour connaître le serveur de mail de `univmed.fr`
4. Le but de l'exercice est de vous faire interroger directement des serveurs de noms. Pour cela, on va utiliser la commande **nslookup** et désactiver le mode récursif (par **set norec**). Ainsi, lorsqu'on interrogera un serveur, s'il ne connaît pas la réponse il ne tentera pas de l'obtenir auprès d'un autre serveur. Il répondra simplement qu'il ne connaît pas la réponse mais nous indiquera un ou plusieurs serveurs de noms qui pourraient nous aider. On interrogera alors un de ces serveurs et on continuera ce processus jusqu'à interroger un serveur de noms qui fait autorité pour le domaine recherché.

Exécuter **nslookup** et désactiver le mode récursif pour :

- a. obtenir le nom et l'adresse d'un serveur de noms du domaine `pasteur.fr`. La requête formulée auprès de notre serveur **doit échouer** (voir remarque ci-dessous), mais nous indiquera qui contacter pour "avancer" dans notre quête.



Si notre serveur de noms connaît la réponse (quelqu'un lui a adressé récemment une requête récursive pour ce domaine), l'exercice perd tout son intérêt. Dans ce cas, il faut rechercher pour un autre domaine, inconnu de notre serveur. Essayer avec `lecanardenchaine.fr`, `icaps-conference.org`, `fdf.org`, `timesonline.co.uk`, `iht.com`, `guardian.co.uk`, `wwf.fr`, `greenpeace.org`, etc.

- b. sélectionner ce serveur avec la commande **server ip-du-serveur**



Ne pas utiliser le nom du serveur mais son adresse IP! Si besoin, il faudra la rechercher.

- c. l'interroger afin de connaître le nom et l'adresse IP du *Mail Exchanger* par défaut de ce domaine. Le *Mail Exchanger* par défaut est celui qui a la plus petite priorité.

[Corrigé]

Exercice 10 (interrogations DNS sur Windows)

Refaire la question 4 avec le client **nslookup** de Windows.

[Corrigé]

2.2.5 WHOIS : informations sur les gestionnaires d'un domaine

En formulant une "requête whois" à une "autorité compétente", on obtient en réponse un certain nombre de renseignements sur les gestionnaires d'un nom de domaine.

L'autorité compétente pour les domaines se terminant par **.fr** et **.re** est l'**AFNIC**. C'est une association chargée d'attribuer et de gérer les domaines **.fr** et **.re**. Elle fournit la possibilité d'interroger en ligne sa base de données via l'URL <http://www.afnic.fr/outils/whois>.

Un grand nombre de domaines tels que **.arpa**, **.biz**, **.com**, **.edu**, **.org**, et autres sont gérés par l'**Internic** bien que **.com** et **.net** soient en fait l'exclusivité de la société **VeriSign**. On peut aussi interroger en ligne ces gestionnaires via l'URL <http://www.internic.net/whois.html>.

Enfin, on peut utiliser un site comme <http://network-tools.com> qui regroupe plusieurs des services que nous avons étudiés au cours de ce TP.



Sous Linux, on peut interroger les serveurs *whois* en utilisant la commande **whois**.

Exercice 11 (interrogation whois sur Linux)

Utiliser la commande **whois** sur **allegro** pour obtenir des renseignements sur différents domaines tels que : **univ-aix.fr.**, **univ-mrs.fr.**, **free.fr.**, **wanadoo.fr.**, **gouv.fr.**, et autres domaines de votre choix.

[Corrigé]

3 TELNET

3.1 Terminal virtuel

TELNET ([RFC 854](#) et [RFC 855](#)) est à l'origine conçu comme un *Terminal Virtuel*, c'est à dire qu'il sert à se connecter à distance sur une station. Le terminal d'où est exécuté le client **telnet** sert alors aux entrées-sorties avec un shell de la station distante. Tout se passe comme si l'utilisateur était installé devant cette station distante. TELNET fonctionne en mode texte. Par ailleurs, il normalise les échanges de texte qui sont effectués afin que les différences de codage entre les deux stations ne soient pas gênants. Il est possible de lancer des applications graphiques lorsqu'on est logé avec TELNET sur une station. Mais il faut alors disposer d'un serveur X (afficheur de fenêtres) actif en local et paramétrer quelques variables et autorisations d'accès. Nous étudierons cette possibilité au cours d'un prochain TP.

Techniquement, **telnet** fonctionne en mode client/serveur. Nous n'étudierons pas le serveur au cours de ce TP. Sachez simplement qu'il accepte des connexions et lance un shell qui reçoit ses commandes de la connexion et qui envoie ses sorties sur la connexion.

Pour lancer un client **telnet** afin de se logger sur une station distante disposant d'un serveur TELNET, il faut simplement taper la ligne de commande :

```
$ telnet hôte
```

où *hôte* est l'adresse IP ou un nom officiel de la station serveur. Le client **telnet** tente alors d'établir une connexion TCP sur le port 23 de la machine *hôte*, qui est réservé aux serveurs TELNET. Le shell lancé demandera alors à l'utilisateur de s'identifier (nom et mot de passe).

Le rôle du client est de servir d'interface entre l'utilisateur et le serveur de manière à ce que (presque) tout ce que l'utilisateur tape soit envoyé au serveur et que (presque) tout ce que le serveur répond soit affiché sur le terminal de l'utilisateur et ce, jusqu'à ce que la connexion soit rompue.

Cependant, il est possible de taper des commandes que seul le client **telnet** doit interpréter, soit pour le paramétrer, soit pour terminer une connexion ou en ouvrir une autre. Pour cela, lorsque le client **telnet** est lancé, il faut taper la séquence de caractères d'échappement pour rentrer dans **le mode commande du client**. Cette séquence est indiquée en début de session TELNET et consiste souvent à maintenir appuyée la touche `Control`, suivie du caractère `]` (crochet fermant que l'on saisit à l'aide de la touche `AltGr`), ou parfois du `$` (cas du client **telnet** de Windows). On entre alors dans le mode de commande du client **telnet**. On peut ensuite taper **help** pour avoir une liste de commandes disponibles, et il suffit de taper deux fois sur la touche `Entrée` pour revenir au terminal virtuel.

Exercice 12 (Utilisation de telnet depuis Linux)

Sur allegro :

1. à partir d'un shell, se connecter à allegro (ben oui, on n'a pas d'autres machines disponibles pour ça) en utilisant **telnet**.
2. s'identifier
3. taper une commande quelconque (telle que `ls -l`)
4. rentrer dans le mode de commande du client (en tapant les 3 touches `AltGr` + `Ctrl` + `]`). Si vous êtes connecté sur allegro par PUTTY, la séquence à utiliser est `Ctrl` + `$`.
5. taper **help**
6. taper deux fois sur `Entrée` (pour revenir à la session)
7. taper à nouveau une commande ;
8. terminer la connexion soit en tapant `Ctrl` + `D`, soit **exit**, soit en entrant dans le mode de commande et en tapant **close** ou **quit**.

[\[Corrigé\]](#)

i TELNET est un service non sécurisé ni crypté. En particulier le nom de l'utilisateur et son mot de passe circulent "en clair" dans le réseau. Par conséquent, le service TELNET est de moins en moins autorisé par les administrateurs réseaux qui le remplacent par SSH (*secure shell*). Notamment, il n'est pas possible de se connecter sur un machine du Département Informatique par TELNET, à partir de l'extérieur de l'IUT. Nous étudierons en détail SSH au cours d'un prochain TP.

Exercice 13 (Utilisation de telnet depuis Windows)

À partir de l'invite de commandes MS-DOS sur Windows :

1. exécuter **telnet sans se connecter à un serveur.**



Parfois, le client **telnet** sous Windows est graphique. Il ne dispose pas du mode commandes, mis à part les menus de la fenêtre qu'il ouvre. Si c'est le cas pour la version installée sur votre poste, explorer ces menus. Sur les versions non graphiques, on remarque que le caractère d'échappement est `Ctrl+$`.

2. taper **help**
3. se connecter à allegro
4. s'identifier
5. taper quelques commandes (**pwd, cd, ls, ...**) sur la session TELNET sur allegro
6. entrer dans le mode commandes de **telnet**
7. taper **set ?**
8. utiliser **set** pour demander un **echo local**.
9. taper `Entrée` pour sortir du mode commandes de **telnet**
10. taper à nouveau quelques commandes (**pwd, cd, ls, ...**), et observer. Les caractères tapés sont écrits 2 fois. Cette option est utile lorsqu'on contacte un serveur autre que TELNET en utilisant le programme **telnet** (voir ci-après) et que les caractères tapés n'apparaissent pas à l'écran.
11. quitter **telnet**

[\[Corrigé\]](#)

3.2 Telnet comme simple client TCP

Le programme client **telnet** peut être utilisé dans un autre but. En effet, il permet de communiquer avec des serveurs (en utilisant le protocole approprié) qui utilisent TCP comme protocole de transport. Ceci parce que **telnet** peut se comporter comme un client TCP ordinaire. Il suffit de lui indiquer après l'*hôte*, le **port** utilisé par le serveur par la ligne de commande suivante :

```
$ telnet hôte port
```

où *port* est le numéro du port de l'application à contacter, ou le nom (ou un alias) du service rendu par cette application tel qu'il apparaît dans le fichier `/etc/services` sur Linux et `C:\WINDOWS\system32\drivers\etc\services` (ou `C:\WINNT\system32\drivers\etc\services`) sur Windows.

Le client **telnet** va alors établir une connexion TCP avec le serveur lié au port *port* sur l'hôte *hôte*. Tout ce que l'utilisateur tapera sera transmis au serveur qui l'interprétera généralement comme un requête. Tout ce que le serveur enverra sera alors affiché sur le terminal de l'utilisateur et correspondra généralement à une réponse à une requête.

Un grand nombre de serveurs utilisent les caractères ASCII dans leurs échanges avec les clients et peuvent être donc contactés avec **telnet**. De plus, ces serveurs utilisent ou requièrent généralement que les fins de lignes soient exprimées par la séquence des deux caractères '`\r`' (*Carriage Return* ou *CR*) et '`\n`' (*Line Feed* ou *LF*), ce que fait justement **telnet**.

i Pour ne citer que quelques protocoles majeurs qui utilisent des caractères ASCII dans leurs échanges, il y a : HTTP, FTP, SMTP, POP3.

Exercice 14 (Utilisation de telnet en client TCP)

Les différents serveurs à contacter dans cet exercice utilisent un port que l'on a indiqué dans le cours de réseaux sur TCP et que l'on peut trouver aussi dans le fichier `/etc/services`. Les serveurs peuvent être contactés à partir de Windows ou de `allegro`, comme bon vous semble.

1. Contacter le serveur *daytime* de `infodoc` en utilisant **telnet**. Celui-ci se contente de renvoyer la date et l'heure qu'il possède puis ferme la connexion ou reste muet. Entrer dans le mode de commandes de **telnet** pour terminer la connexion si le serveur ne l'a pas fait. Essayer aussi de contacter le serveur *daytime* d'`allegro`, d'`oralinux` et de `paprika`. En revanche, il y a très peu de chances de contacter avec succès un serveur situé à l'extérieur de l'IUT car ce genre de services est généralement désactivé (ou filtré).
2. Contacter le serveur *echo* de `infodoc`. Celui-ci renvoie les caractères qu'il reçoit jusqu'à ce que la connexion soit rompue. Taper quelques lignes puis terminer la connexion (en passant par le mode commandes de **telnet**).

[Corrigé]

4 nmap : scanner un réseau

Un des logiciels favoris des pirates (☹) est un **scanner de ports**. En effet, les pirates cherchent les services qui sont susceptibles de posséder une faille de sécurité afin de pénétrer un système. Pour cela, ils ont besoin d'un bon scanner dans leur trousse à outil. Le plus célèbre est certainement **nmap** (téléchargeable sur le site <http://www.insecure.org>).

Mais le scanner ne sert pas seulement aux pirates !! Il permet aux administrateurs réseaux (☺) de vérifier les ports ouverts sur leurs systèmes et s'il n'y a pas de service inutile actif ni de *backdoor*.

i **nmap** est aussi disponible pour Windows (voir site <http://www.insecure.org>). De plus, une interface graphique nommée **zenmap** est maintenant disponible.

Exercice 15 (Utilisation de nmap sur Linux)

1. Utiliser **nmap** pour vérifier les services ouverts sur `allegro`. Comparer avec le résultat de `netstat -atn` exécutée sur `allegro` (l'état d'un serveur en écoute et donc actif est LISTEN ou LISTENING).
2. Utiliser **nmap** pour vérifier ceux ouverts sur `infodoc`.
3. Utiliser **nmap** pour vérifier ceux ouverts sur votre machine personnelle si elle est connectée à l'Internet.

[Corrigé]

5 netcat : un client/serveur à tester

Il existe d'autres logiciels que **telnet** pour contacter un serveur quelconque, notamment le logiciel **netcat** (et son évolution **socat**). Sous Linux, l'exécutable se nomme **nc** et est généralement situé dans `/usr/bin`.

Comparé par à **telnet**, l'avantage de **nc** est qu'il permet non seulement d'envoyer des caractères ASCII sur une connexion mais aussi des données binaires. Il peut aussi servir de client UDP (alors que **telnet** n'utilise que TCP). Enfin, il peut aussi servir de serveur, c'est à dire qu'il peut être "lié" à un port (TCP ou UDP selon le choix fait) et attendre que des clients lui envoient des données.

i Il existe aussi un portage de **netcat** sur Windows, que l'on peut trouver sur le site <http://www.vulnwatch.org/netcat/>.

Exercice 16 (Utilisation de netcat sur Linux)

1. Utiliser **nc** pour contacter le serveur TCP daytime de infodoc
2. Utiliser **nc** pour contacter le serveur UDP echo d'infodoc. On peut voir les services UDP actifs sur une machine si l'on y est logé, en utilisant **netstat -au**. Ce n'est pas possible avec **nmap** si on n'est pas *root*.
3. Utiliser **nc** pour lancer un serveur sur un port que vous choisirez (au dessus de 10000). Sur un autre terminal, utiliser **telnet** pour vous connecter à ce serveur.
4. Utiliser **nc** pour lancer un serveur UDP sur un port que vous choisirez (au dessus de 10000). Essayer d'utiliser **telnet** pour le contacter (ça ne marchera pas). Le contacter en lançant un client **nc**.

[Corrigé]

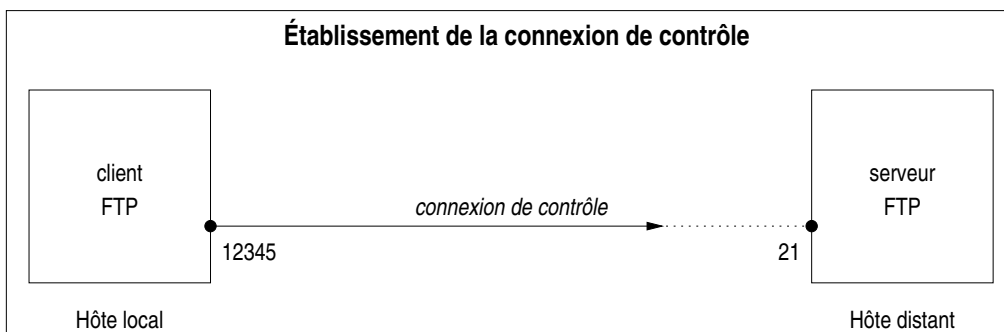
6 FTP

File Transfer Protocol ([RFC 959](#)) est un (vieux) protocole de la couche application de TCP/IP, permettant de transférer des fichiers entre des stations. Le programme **ftp** sur *Linux* et *Windows* est un **client** FTP permettant de contacter un **serveur** FTP.

En tapant :

```
$ ftp hôte
```

où *hôte* est l'adresse IP ou le nom d'une station, le client tente d'ouvrir une connexion TCP sur le port 21 de l'*hôte* indiqué :



Si le serveur FTP est actif et répond favorablement, une **connexion de contrôle** est établie. Une "session" FTP vient alors de démarrer et le serveur attend généralement que l'utilisateur s'authentifie en fournissant un **nom d'utilisateur** et un **mot de passe**.

6.1 Authentification FTP

Il y a deux possibilités d'authentification :

- soit on dispose d'un compte sur le serveur et l'on veut transférer des fichiers depuis ou vers ce compte. Dans ce cas, on saisit son nom d'utilisateur et son mot de passe ;
- soit on ne dispose d'aucun compte sur le serveur mais on sait qu'il met à disposition certains fichiers que l'on peut télécharger ou un répertoire dans lequel on peut déposer (*upload*) des fichiers. Dans ce cas, il faut utiliser le nom d'utilisateur "**anonymous**" (parfois "**guest**") et n'importe quoi est souvent accepté comme mot de passe (parfois, il n'est même pas demandé).

❗ La règle de bonne conduite sur Internet veut cependant qu'on indique son adresse e-mail comme mot de passe, en particulier si un message du serveur le précise lors de l'authentification. Il n'y *a priori* aucune raison de ne pas le faire...

6.2 Commandes du client FTP

Une fois le *login* accepté, de nombreuses commandes sont disponibles pour transférer des fichiers (mais pas des répertoires).

❗ Pour obtenir la liste des commandes disponibles proposées par le client **ftp**, il faut taper :

```
ftp> help
```

ou simplement :

```
ftp> ?
```

Pour obtenir une description sur une commande en particulier, taper :

```
ftp> ? commande
```

Passons en revue quelques unes des commandes disponibles à partir du client FTP.


6.2.1 Commandes relatives au transfert de fichiers

- **binary** et **ascii** : permettent de choisir une méthode de transfert selon la nature du fichier à transférer.

```
ftp> binary
ftp> ascii
```

Avant de transférer un fichier, il faut en connaître le type : **fichier texte** ou **fichier binaire**. FTP ne fournit pas de commande permettant de connaître le type d'un fichier avant de le transférer. Soit on sait quel est son type, soit on se base sur son extension ou sur son nom. Par exemple, les fichiers `.h`, `.cxx`, `.txt`,

.uu, .ps, .java, readme devraient être des fichiers textes, alors que les fichiers .tar, .tgz, .Z, .gz, .exe, .avi, .rar, .zip, .gif, .mp3, .deb, .rpm, devraient être des fichiers binaires.

 Pourquoi les distinguer ? Parce que les fichiers textes doivent subir une modification pour adapter leur codage à celui du système qui les télécharge, alors que les fichiers binaires ne doivent pas être modifiés par le transfert.

Ainsi, **binary** demande que le transfert qui suit soit binaire alors que **ascii** demande à ce qu'il soit en mode texte.

- **passive** : permet de basculer du mode actif au mode passif, et inversement (voir **get** ci-après).

```
ftp> passive
```



Le client FTP sur Windows ne reconnaît pas cette commande. Ce client est donc généralement inutilisable sur un ordinateur qui se trouve derrière un *firewall* ou une *Natbox*. Voir explications ci-dessous sur les modes actif et passif.

- **get** : permet de demander le transfert d'un fichier du serveur vers l'hôte local.

```
ftp> get fic-source [fic-destination]
```

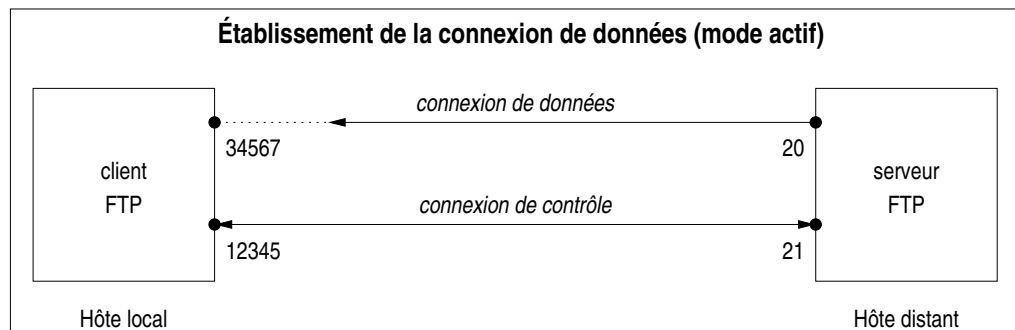
où *fic-source* est la référence du fichier à transférer et *fic-destination* est le nom qu'il portera localement. Le transfert du fichier est opéré sur une *connexion de données* établie pour l'occasion.

Il y a deux possibilités pour l'établissement de la connexion de données :

- ◇ en **mode actif**, le client choisit un port (ou se le fait attribuer par TCP) sur lequel il se place en écoute puis, par l'intermédiaire de la connexion de contrôle, indique au serveur sur quel port il est joignable. Le serveur se connecte alors au client pour établir la connexion de données. Côté serveur, le port utilisé pour cette connexion est le port 20.



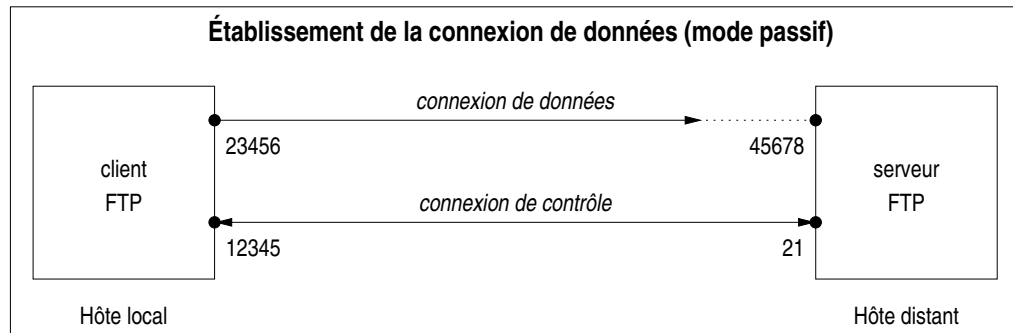
Pour la connexion de données en mode actif, le client devient le serveur et le serveur devient le client.



Le mode actif est problématique lorsque le client se trouve derrière un NAT/PAT. En effet, il faudrait établir une règle de redirection pour le port écouté par le client. Bien souvent, cette solution n'est pas envisageable. Derrière un routeur qui filtre les connexions entrantes (firewall), le problème est similaire.

i Certaines *Natbox* évoluées règlent toutefois le problème du transfert actif en établissant d'elles mêmes la règle de redirection nécessaire, lorsqu'elles détectent une initiation de ce type de transfert.

- ◇ en **mode passif**, c'est le client qui se connecte au serveur pour établir la connexion de données. Le port utilisé pour cette connexion par le serveur est communiqué au client par l'intermédiaire de la connexion de contrôle.



i Chaque transfert de fichier doit se faire sur une connexion de données différente.

i Le mode passif est aussi problématique lorsque le serveur se trouve derrière un NAT/PAT ou un firewall. En effet, il faudrait établir une règle de redirection pour le port écouté par le serveur.

i Là encore, certaines *Natbox* évoluées règlent toutefois le problème du transfert passif (côté serveur) en établissant d'elles mêmes la règle de redirection nécessaire, lorsqu'elles détectent une initiation de ce type de transfert.

- **hash** : activer/désactiver la visualisation de l'activité de transfert

```
ftp> hash
```

a pour effet de faire afficher par le client un dièse (ou autre caractère selon le client) pour chaque bloc de fichier transféré. La taille du bloc est indiquée en réponse à **hash**.

i les dièses ne sont pas transmis par le serveur ; c'est le client qui les écrit en fonction de la quantité d'information transférée.

- **mget** : permet de demander le transfert de plusieurs fichiers du serveur vers l'hôte local.

```
ftp> mget motif {motif}
```

où *motif* est un motif de nom de fichier. Pour chaque fichier qui correspond au(x) motif(s), l'utilisateur est invité à confirmer le transfert, à moins qu'il ait demandé un mode sans invite (voir **prompt** ci-après).

i Encore une fois, chaque transfert de fichier doit se faire sur une connexion de données différente.

- **prompt** : permet de basculer du mode avec invite au mode sans invite.

```
ftp> prompt
```

- **put** : permet de demander le transfert d'un fichier de l'hôte local vers le serveur.

```
ftp> put fic-source [fic-destination]
```

où *fic-source* est la référence du fichier à transférer et *fic-destination* est le nom qu'il portera sur le serveur. Il faut bien évidemment avoir les autorisations nécessaires pour déposer un fichier sur le répertoire cible. Le transfert du fichier est opéré sur une *connexion de données* établie pour l'occasion.

- **mput** : permet de demander le transfert de plusieurs fichiers de l'hôte local vers le serveur.

```
ftp> mput motif {motif}
```

où *motif* est un motif de nom de fichier. Pour chaque fichier qui correspond au(x) motif(s), l'utilisateur est invité à confirmer le transfert, à moins qu'il ait demandé un mode sans invite.

6.2.2 Commandes de déplacement et de manipulation de fichiers

- **cd** : permet de se placer dans un répertoire sur le serveur. Cette commande s'utilise comme sur Unix.

```
ftp> cd répertoire
```

- **lcd** : permet de se placer dans un répertoire sur l'hôte local. Cette commande s'utilise comme sur Unix.

```
ftp> lcd répertoire
```

- **ls** ou **dir** : permettent d'obtenir des informations sur un fichier ou le contenu d'un répertoire.

```
ftp> ls [-l] [référence] [fichier-sortie]
```

```
ftp> dir [référence] [fichier-sortie]
```

dir est équivalent à **ls -l**. *référence* est le fichier ou le répertoire sur lequel appliquer la commande et *fichier-sortie* est une référence d'un fichier dans lequel on veut placer le résultat de la commande.

Certains sites (comme `ftp.inria.fr`) conseillent fortement d'utiliser **ls -l** ou **dir** plutôt qu'un simple **ls** (j'avoue ne pas savoir pourquoi, alors si vous le savez, dites-le moi en m'envoyant un [mail](#)).



Les informations fournies par **ls (dir)** le sont sur une *connexion de données*.

- **chmod** : permet de changer les permissions d'un fichier (ou répertoire) sur le serveur

```
ftp> chmod permissions référence
```

- **rename** : permet de renommer un fichier sur le serveur

```
ftp> rename ancien nouveau
```

demande de renommer le fichier (ou répertoire) *ancien* en *nouveau*.



On ne déplace pas un fichier avec cette commande. De plus, si le fichier nouveau existe déjà, il est écrasé. Si ancien et nouveau sont des répertoires (existants), alors nouveau sera écrasé s'il est vide, sinon cela produit une erreur.

- **delete** : permet de supprimer un fichier sur le serveur

```
ftp> delete référence
```

Cette commande ne s'applique pas aux répertoires.

- **mdelete** : permet de supprimer plusieurs fichiers sur le serveur

```
ftp> mdelete motif {motif}
```

À nouveau, cette commande ne s'applique pas aux répertoires.

- **mkdir** : permet de créer un répertoire sur le serveur

```
ftp> mkdir référence
```

- **rmdir** : permet de supprimer un répertoire sur le serveur

```
ftp> rmdir référence
```

où *référence* est un répertoire vide.

- **!** : permet d'exécuter une commande localement.

```
ftp> ! [commande]
```

La *commande* est exécutée dans un shell et son résultat est affiché sur la sortie standard. Cela permet par exemple de pouvoir visualiser un fichier sans quitter la session FTP, de créer localement un répertoire, renommer un fichier, etc. La syntaxe de *commande* est propre au shell du système du client. Si aucune commande n'est fournie, le client FTP exécute un shell et lui cède la place. Lorsque le shell est terminé (par exemple, en tapant une commande shell de type **exit**), le client FTP reprend la main.

6.2.3 Commandes de gestion de la session

- **bye** ou **exit** ou **quit** : terminer la session FTP et se déconnecter du serveur.

```
ftp> bye
ftp> exit
ftp> quit
```

- **close** : permet de terminer la session FTP en cours.

```
ftp> close
```

Cette commande ne fait pas quitter du client FTP.

- **open** : permet d'ouvrir une session FTP

```
ftp> open serveur
```

- **user** : permet de commencer la phase d'authentification

```
ftp> user nom
```

Peut s'utiliser si l'utilisateur ne s'est pas encore logé (par exemple après un échec d'authentification). Suite à cette commande, il faudra généralement fournir un mot de passe pour s'authentifier en tant qu'utilisateur *nom*.

6.3 Limitations de FTP

FTP n'est pas conçu pour télécharger ou déposer une hiérarchie de répertoires (arborescence). Plus simplement, il n'est pas possible de demander le transfert d'un fichier répertoire (avec son contenu). Cela est très gênant. Pour contourner le problème, les arborescences doivent être archivées dans un fichier unique (et binaire) qui pourra être transféré.

Sous Unix (et Linux), les archives sont créées avec l'utilitaire **tar** (*tape archiver*) et portent généralement l'extension `.tar`, `.tgz` (si compressée en même temps par **gzip**) ou `.tgZ`, `.taz` (si compression par **compress**). Parfois, l'archive est compressée après avoir été créée, et porte une extension de type `.tar.gz` ou `.tar.Z`. Sous Windows, les archives sont le plus souvent créées (avec compression automatique) en utilisant les logiciels WinZip ou Winrar, et portent l'extension `.zip` ou `.rar`.

i Certains clients FTP (comme la plupart des clients graphiques) permettent de transférer des arborescences. Pour cela, ils parcourent eux-même récursivement l'arborescence à télécharger, en faisant une requête **LIST** pour chaque répertoire rencontré.

Si vous n'avez pas encore vu l'utilitaire **tar** en cours de systèmes (ni les utilitaires de compression/décompression), vous ne saurez pas télécharger pour le moment une arborescence sans passer par un client graphique.

6.4 Clients FTP graphiques

Il existe pléthore de clients FTP graphiques plus ou moins ergonomiques, notamment en offrant une visualisation côte-à-côte du répertoire distant et du répertoire local et en permettant le *drag and drop*. De plus, ils permettent généralement de transférer des arborescences (ils se chargent de se déplacer dans l'arborescence et d'en transférer le contenu). Parmi ce genre d'outils, on peut citer :

- [WS-FTP](#) sous Windows édité par [ipswitch](#)
- [Filezilla](#) sous Windows et Linux
- [gftp](#) sous Linux/Unix

6.5 Exercices

Exercice 17 (Session FTP avec ftp.rfc-editor.org depuis Linux)

En utilisant **ftp** à partir d'**allegro**, se connecter en tant qu'utilisateur anonyme au serveur FTP du site officiel des dépôts des RFC, d'adresse `ftp.rfc-editor.org` (ou au serveur `ftp.pasteur.fr`). Aller dans le répertoire `in-notes` (sur le serveur `pasteur.fr`, ce répertoire est dans `pub/computing/rfc`) et récupérer le fichier `rfc1939.txt` en mode **passif** (transfert de fichier texte) avec visualisation de l'activité de transfert. Lorsque le fichier a été transféré, fermer la connexion en quittant **ftp** en tapant **quit** (ou **bye**).

[Corrigé]

Exercice 18 (Dépot d'un fichier sur un serveur FTP)

1. Depuis Windows, déplacer (donc en supprimant l'original) le fichier `rfc1939.txt` pour le mettre sur le *Bureau*
2. Sur l'invite de commandes MS-DOS, se placer sur le *Bureau* en tapant :


```
Z:> C:
C:> cd "\\Documents And Settings\nnnppp\Bureau"
```

 où *nnnppp* est votre nom d'utilisateur, puis taper `dir` pour vérifier que le fichier est bien là.
3. Exécuter ensuite le client FTP en se connectant à *allegro* en utilisant votre vrai nom d'utilisateur. Une fois fait, créer un répertoire `rfc` dans `tp` à partir de la session FTP et y déposer le fichier `rfc1939.txt`. Enfin quitter le client FTP et vérifier que le fichier est bien présent sur *allegro*.

[Corrigé]

Exercice 19 (Utilisation de gftp depuis Linux)

En utilisant `gftp` depuis *allegro*, récupérer la RFC 959 depuis le serveur `ftp.rfc-editor.org`. Cette RFC décrit le protocole FTP. Y jeter un coup d'œil puis la mettre de côté.

[Corrigé]

i Les administrateurs réseaux ferment de plus en plus les services non sécurisés (cryptés) comme FTP. Dans ce cas, il faut utiliser de sous-modules de `ssh` (*secure shell*) offrant des services de type FTP-crypté. Les clients de ces sous-modules sont par exemple `scp` (*secure copy*), `sftp` (*secure FTP*) ou `lftp` sous Linux. Sous Windows, il existe un logiciel gratuit nommé **PSFTP** que l'on trouve sur le site consacré à PUTTY (un client SSH pour Windows) : <http://www.chiark.greenend.org.uk/~sgtatham/putty/>.

6.6 Messages du protocole FTP

Le protocole FTP fonctionnant en mode client/serveur, le client envoie un message (requête) au serveur via la connexion de contrôle et le serveur lui retourne une réponse sur cette même connexion. Selon la requête, cela produit l'établissement d'une connexion de données et un transfert d'information (fichier, contenu d'un répertoire).

Voici ci-dessous quelques messages que le client peut envoyer ainsi que quelques réponses possibles du serveur, le tout étant précisé dans la RFC 959 que vous avez téléchargée :

💣 Tous les messages (requête ou réponse) se terminent par les deux caractères '`\r`' (*Carriage Return* ou *CR*) et '`\n`' (*Line Feed* ou *LF*). Cependant, les serveurs acceptent généralement les requêtes qui ne se terminent que par *LF*.

USER *nom*

pour indiquer un nom d'utilisateur. Si le serveur a besoin d'un mot de passe pour cet utilisateur, il enverra une réponse commençant par 331 comme :

```
331 User name okay, need password.
```

sinon, il acceptera le login par un message commençant par 230 comme :

```
230 nom user logged in
```

✍ Les réponses des serveurs commencent toutes par un code à trois chiffres suivi d'une explication. L'explication est libre et est destinée aux humains. En revanche le code n'est pas libre car est destiné à être traité par le processus client. Les réponses commencent toujours par un chiffre compris entre 1 et 5. S'il vaut 4 ou 5, c'est que le serveur indique une erreur, sinon il s'agit d'une réponse favorable.

✍ Parfois, la réponse du serveur tient en plusieurs lignes. Dans ce cas, toutes les lignes ont le même code suivi d'un tiret -, sauf la dernière pour laquelle le code est suivi d'un espace.

PASS mot-de-passe

pour indiquer le mot de passe pour cet utilisateur. En réponse, le serveur indique que l'authentification est acceptée par un message commençant par 230 comme :

```
230 User nom logged in.
```

Il indique une erreur d'authentification par un message commençant par 530 comme :

```
530 Login incorrect.
```

CWD répertoire

pour changer de répertoire. En réponse, le serveur indique un succès par un message commençant par 250 comme :

```
250 CWD command successful
```

et une erreur par un message commençant par 550 comme :

```
550 repertoire: No such file or directory
```

PORT $h_1, h_2, h_3, h_4, p_1, p_2$

dans le mode actif, le client indique par cette commande sur quel couple adresse/port le serveur doit se connecter pour établir la connexion de données. Les valeurs h_1, h_2, h_3, h_4 indiquent l'adresse IP $h_1.h_2.h_3.h_4$. Le port P en question est indiqué par p_1 et p_2 , où $P = (p_1 \times 256) + p_2$ (p_1 est la valeur de l'octet de poids fort de P et p_2 celle de l'octet de poids faible).

Une réponse favorable du serveur commencera par 200 comme :

```
200 PORT command successful
```

❗ L'adresse ne correspond pas forcément à celle du client. Ce dernier peut demander au serveur de se connecter à une autre machine, pour peu qu'il y ait un processus sur la machine ciblée, en écoute sur le port indiqué. Cependant, la plupart des serveurs refusent cette opération.

PASV

dans le mode passif, le client indique par cette commande qu'il va demander un transfert, et qu'il a besoin que le serveur lui indique un couple adresse/port pour se connecter.

Une réponse positive commencera par 227 et contiendra le couple adresse/port comme :

```
227 Entering Passive Mode (82,3,4,5,151,37).
```

Ici, le serveur indique qu'il s'est mis en attente sur le port 38693 (= $151 \times 256 + 37$) sur la machine d'adresse 82.3.4.5.

TYPE type

pour spécifier un mode de transfert des données. Parmi les *types* reconnus, il y a **A** (ascii) et **I** (image binaire).

Une réponse positive du serveur commencera par 200 comme :

```
200 Type set to type
```

LIST

demande au serveur de transférer le contenu (avec détails) du répertoire de travail (sur le serveur). Ce transfert se fait en établissant une connexion de données. Pour cela, cette requête doit être précédée d'un message **PORT** ou **PASV**.

Si le serveur accepte, il enverra un message commençant par 150 comme :

```
150 Opening ASCII mode data connection for file list
```

Lorsque le transfert est terminé, le serveur envoie un message commençant par 226 comme :

```
226 Transfer complete.
```

RETR référence

demande le transfert du fichier indiqué depuis le serveur vers le client. Ce transfert se fait dans les mêmes conditions que pour **LIST**.

Si le serveur accepte, il enverra un message commençant par 150 comme :

```
150 Opening ASCII mode data connection for référence (x bytes)
```

Lorsque le transfert est terminé, le serveur envoie un message commençant par 226 comme :

```
226 Transfer complete.
```

STOR *référence*

pour indiquer au serveur que l'on veut transférer des données, qu'il faut sauver dans le fichier *référence*. Ce transfert se fait dans les mêmes conditions que pour **LIST**.

Si le serveur accepte, il enverra un message commençant par 150 comme :

```
150 Opening ASCII mode data connection for référence
```

Lorsque le transfert est terminé, le serveur envoie un message commençant par 226 comme :

```
226 Transfer complete.
```

QUIT

pour terminer la session.

La réponse du serveur commence par 221 comme :

```
221 Goodbye.
```

6.7 Exercices

Exercice 20 (Transfert manuel en mode actif)

Sur allegro :

1. utiliser **telnet** pour ouvrir une connexion de contrôle avec le serveur FTP de allegro
2. vous authentifier avec votre vrai nom d'utilisateur et votre mot de passe. **Attention, il est écrit en clair !!**
3. sur un autre terminal, utiliser **nc** pour se mettre en écoute sur un port que vous choisirez
4. sur la connexion de contrôle, indiquer au serveur sur quel port votre **nc** est en écoute
5. sur la connexion de contrôle, demander le contenu du répertoire de travail. Celui-ci doit être récupéré par votre "serveur" **nc**
6. quitter la session FTP.

[\[Corrigé\]](#)

Exercice 21 (Transfert manuel en mode passif)

Utiliser **telnet** pour ouvrir une connexion de contrôle avec le serveur FTP de `ftp.rfc-editor.org` puis faites le nécessaire afin de récupérer la RFC 821 (fichier `rfc821.txt` du répertoire `in-notes`). La mettre de côté.

[\[Corrigé\]](#)

7 Le courrier électronique

7.1 Clients de messagerie

Un client de messagerie classique est un logiciel permettant de télécharger du courrier électronique, d'en composer et d'en envoyer. Il n'est pas à confondre avec un client *hotmail* ou *caramail* qui utilise les facilités d'un *browser* (navigateur Web). Les clients graphiques de messagerie les plus connus sont **Outlook** (Express), **Eudora**, **Thunderbird** sous Windows, et **Netscape Messenger**, **Evolution** ou **Thunderbird** sous Linux. Pour fonctionner, ces clients ont besoin d'être paramétrés pour les deux protocoles qu'ils utilisent, POP3 et SMTP :

- **POP3** : le client a besoin de connaître le nom de la machine hébergeant le serveur POP3, et éventuellement le numéro du port qu'il utilise (normalement 110). Il a aussi besoin de connaître le nom d'utilisateur de la boîte aux lettres ainsi que son mot de passe. Avec ces informations, il est ensuite possible de télécharger tout le courrier disponible dans la boîte puis de le consulter hors-ligne ;
- **SMTP** : le client a besoin de connaître le nom de la machine hébergeant le serveur SMTP (qui peut être différente de celle hébergeant le serveur POP3) ainsi que le port (normalement 25). Le plus souvent, ce serveur doit être celui mis à disposition par le FAI. Les autres serveurs devraient refuser de relayer les messages à cause des *spams* (ces courriers publicitaires qui pourrissent la vie et appelés aussi "pourriels" au lieu de "courriels"). Il faut aussi indiquer son adresse e-mail afin qu'elle soit indiquée dans les messages envoyés pour que le destinataire puisse répondre.

Exercice 22 (Utilisation de Thunderbird)



Si vous avez un fichier `~/ .forward` sur allegro, le renommer le temps du TP.

Lancer **Thunderbird** sous Windows. Le configurer en utilisant allegro comme serveurs POP3 et SMTP, puis s'envoyer un ou deux messages. Ensuite les récupérer à nouveau avec **Thunderbird**.

[\[Corrigé\]](#)

7.2 Format d'un message électronique

Le format d'un message électronique (transmis via SMTP) a d'abord été défini dans la [RFC 822](#) qui a été remplacée récemment par la [RFC 2822](#). Un message électronique est composé de deux parties : un en-tête et un corps, séparées par une ligne vide. Le corps est libre, puisqu'il contient le message de l'utilisateur. Il doit cependant respecter la règle d'au plus 1000 caractères par ligne, et ne doit contenir que des caractères ASCII. Pour envoyer des données binaires par email, il faut les encoder en ASCII.

En revanche, l'en-tête contient une succession de champs, dont certains sont obligatoires, d'autres sont optionnels, et d'autres encore sont ignorés. Un grand nombre de champs sont prévus par la RFC, parmi lesquels :

- **Return-Path**: *<adresse-origine>* identifie (l'utilisateur à) l'origine du message
- **Received**: *informations de transfert* est un champ ajouté par chaque serveur SMTP ou client de relève de courrier par lequel le message est passé
- **From**: *adresse-expéditeur* identifie l'expéditeur du message. Ce champ peut éventuellement être différent de celui indiqué dans **Return-Path** :
- **To**: *adresses-destinataires* identifie les destinataires du message
- **Cc**: *adresses-destinataires-cc* indique des destinataires en copie carbone

- **Reply-To**: *adresse-réponse* indique une adresse électronique à laquelle une réponse à ce message doit être envoyée. Si ce champ n'est pas présent, c'est le champ **To**: qui servira pour la réponse.
- **Subject**: *sujet* indique le sujet du message
- **Message-Id**: *identificateur* indique un numéro de message qui doit être unique pour celui qui le génère (normalement le client de messagerie électronique qui est à l'origine du message)
- **Date**: *date* indique la date à laquelle la transmission du message a commencé

Exemple

Voici un exemple d'un message (très court) envoyé depuis la station d'IP 139.124.208.98, passant par le serveur SMTP de l'Université de la Méditerranée et à destination de ma boîte aux lettres sur allegro :

```
Return-Path: <cyril.pain-barre@univmed.fr>
X-Original-To: cpb@allegro.iut.univ-aix.fr
Delivered-To: cpb@allegro.iut.univ-aix.fr
Received: from univmed.fr (smtp.univmed.fr [139.124.132.120])
    by allegro.iut.univ-aix.fr (Postfix) with ESMTMP id 05C00FC0
    for <cpb@allegro.iut.univ-aix.fr>; Wed, 28 Mar 2007 15:18:25 +0200 (CEST)
Received: from localhost (localhost.localdomain [127.0.0.1])
    by univmed.fr (Postfix) with ESMTMP id 87D0A46A399
    for <cpb@allegro.iut.univ-aix.fr>; Wed, 28 Mar 2007 15:18:23 +0200 (CEST)
X-Virus-Scanned: by amavisd-new at univmed.fr
Received: from univmed.fr ([127.0.0.1])
    by localhost (smtp.univmed.fr [127.0.0.1]) (amavisd-new, port 10024)
    with ESMTMP id 3EYMcU2EpxsQ for <cpb@allegro.iut.univ-aix.fr>;
    Wed, 28 Mar 2007 15:18:23 +0200 (CEST)
Received: from [139.124.208.98] (unknown [139.124.208.98])
    by univmed.fr (Postfix) with ESMTMP id 1E90D46A423
    for <cpb@allegro.iut.univ-aix.fr>; Wed, 28 Mar 2007 15:18:23 +0200 (CEST)
Message-ID: <460A6B1E.7030102@univmed.fr>
Date: Wed, 28 Mar 2007 15:18:22 +0200
From: Cyril Pain-Barre <cyril.pain-barre@univmed.fr>
User-Agent: Thunderbird 1.5.0.7 (X11/20060927)
MIME-Version: 1.0
To: CPB <cpb@allegro.iut.univ-aix.fr>
Subject: exemple de message
X-Enigmail-Version: 0.94.2.0
Content-Type: text/plain; charset=ISO-8859-1
Content-Transfer-Encoding: 8bit

Le corps du message commence ici...

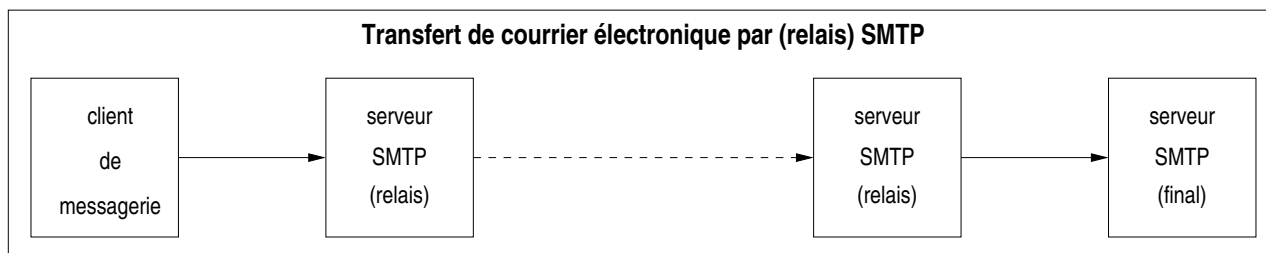
...et se termine là.
```

□

7.3 SMTP

Simple Message Transfer Protocol ([RFC 821](#)) est un (vieux) protocole d'application de TCP/IP dont la spécification remonte à 1982. Il a pour but de transférer des messages tels que le précédent depuis l'expéditeur jusqu'à la boîte aux lettres du (ou des) destinataire(s) du message.

Étant donné un destinataire pour un mél comme `cpb@allegro.iut.univ-aix.fr`, il faut transmettre le mél au serveur qui est en charge de cette boîte aux lettres. Ce serveur est celui spécifié par le champ MX de l'enregistrement DNS pour le domaine de l'adresse (ici `allegro.iut.univ-aix.fr` qui n'a pas d'entrée MX associée donc le message est directement envoyé à `allegro`). Bien qu'un client de messagerie puisse en théorie contacter directement le serveur SMTP qui est en charge de la boîte aux lettres destination, il se contente généralement de délivrer le mél au serveur SMTP qui lui a été indiqué dans sa configuration. Ce serveur va alors servir de relais et va contacter un (ou plusieurs) autre(s) serveur(s) SMTP pour lui(leur) remettre le mél. Dans ce cas, le serveur SMTP relais joue le rôle client SMTP :



Ce protocole est relativement complexe (la RFC fait 68 pages) mais la remise d'un message électronique à un serveur SMTP peut être réalisée très simplement.

Pour cela, le client SMTP doit se connecter au serveur SMTP de l'hôte cible, qui est normalement en écoute sur le port TCP 25. Si un serveur SMTP est effectivement actif sur l'hôte cible, il réagit à l'établissement de la connexion par l'envoi d'un message commençant par 220 comme :

```
220 hôte READY FOR MAIL
```

Si le serveur refuse le client (par exemple un serveur SMTP d'un FAI qui n'accepte que la connexion d'une machine ayant une IP attribuée par le FAI), il peut envoyer un message commençant par 554 comme :

```
554 <adresse client>: Client host rejected: Access denied
```

À l'instar de nombreux protocoles d'application de TCP/IP (et notamment de FTP vu précédemment), le dialogue entre le client et le serveur se fait en mode requête/réponse :

- la requête du client commence par un mot clé suivi d'un espace et d'éventuels paramètres ;
- la réponse du serveur commence toujours par un code à 3 chiffres, suivi d'un espace et d'un texte destiné à être traité par les humains. Le premier chiffre du code est toujours compris entre 1 et 5 : s'il vaut 4 ou 5, le serveur indique une erreur, sinon il s'agit d'une réponse positive.

i À nouveau, les requêtes et les réponses doivent être terminées par la séquence `'\r\n'`, bien que de nombreux serveurs s'accommodent d'un simple `'\n'`.

7.3.1 Messages du protocole SMTP

Voici ci-dessous quelques messages que le client SMTP doit envoyer **dans l'ordre** pour transmettre un message ainsi que quelques réponses possibles du serveur :

```
HELO hôte-client
```

le client se présente au serveur en indiquant par *hôte-client* son nom d'hôte officiel (enregistré au DNS),

ou à défaut son adresse IP.

Une réponse positive du serveur commence par 250 comme :

```
250 hôte serveur
```



Certains serveurs refusent le dialogue avec un client qui ne possède pas d'enregistrement officiel au DNS ou dont le nom indiqué ne correspond pas à l'adresse IP du client.

MAIL FROM: <email-expéditeur>

indique au serveur quel est l'expéditeur du message (son adresse email). Si l'en-tête du message (voir message **DATA**) ne contient pas de champ **From:**, cette adresse sera prise comme celle de l'expéditeur.

Une réponse favorable commencera par 250 comme :

```
250 OK
```

Une erreur possible est la suivante, où l'adresse de l'expéditeur n'est pas acceptée par le serveur (il n'y a pas la suite comme @parici.fr) :

```
501 <toto>: sender address must contain a domain
```

RCPT TO: <email-destinataire>

indique un destinataire pour ce message. Il peut y avoir plusieurs destinataires pour un message et donc plusieurs messages **RCPT**.

Une réponse favorable du serveur commencera par 250 comme :

```
250 OK
```

Si l'adresse du destinataire correspond à un domaine géré par le serveur (comme allegro.iut.univ-aix.fr), mais que le destinataire est inconnu, la réponse du serveur commencera par 550 comme :

```
550 No such user
```



À cause des *spams*, de nombreux serveurs sont configurés pour ne pas accepter de servir de relais, c'est à dire prendre en charge un message destiné à un utilisateur non local (il faudrait alors que le serveur SMTP devienne un client SMTP en contactant le serveur SMTP en charge du domaine¹ de cet utilisateur), si le client SMTP qui lui confie ce message n'appartient pas au même réseau.

Dans ce cas, la réponse du serveur commence par 554 comme :

```
554 <utilisateur@domaine.non.local.fr>: Relay access denied
```


DATA

le client indique au serveur qu'il souhaite lui communiquer le message.

Une réponse favorable du serveur commencera par 354 comme :

```
354 Enter message, ending with <CR><LF>.<CR><LF>
```

Le client est alors invité à transmettre son message, conformément au format décrit précédemment.

 Ainsi qu'il est précisé dans la réponse du serveur ci-dessus, la fin d'un message doit être indiquée par la séquence "\r\n.\r\n", c'est à dire **un point seul sur une ligne**. Pour éviter les ambiguïtés, le protocole veut que si une ligne du message commence par un point, alors il faut le doubler (caractère de transparence).

Lorsque l'indicateur de fin du message est reçu par le serveur, celui-ci envoie une réponse commençant par 250 comme :

```
250 OK
```

QUIT

indique au serveur que la transmission des messages est terminée, et que la connexion peut être rompue.

Le serveur répond toujours favorablement par une réponse commençant par 221 comme :

```
221 hôte serveur Service closing connection
```

7.3.2 Exercice

Exercice 23 (Envoi du message anonyme)

En s'appuyant sur les messages possibles SMTP et la RFC 821 si besoin, envoyer un mail anonyme (en déclarant une fausse identité) en discutant avec le serveur SMTP d'allegro par **telnet**.



L'envoi de messages anonymes contredit totalement la charte du Département Informatique et de l'Université et est répréhensible pénalement.

NE VOUS AMUSEZ SURTOUT PAS À USURPER L'IDENTITÉ DE QUELQU'UN POUR ENVOYER UN MESSAGE, SINON VOUS VOUS EXPOSEREZ À DE GRAVES SANCTIONS VOIRE DES POURSUITES JUDICIAIRES ! MAIL ANONYME NE VEUT PAS FORCÉMENT DIRE NON TRAÇABLE !

Nous faisons cet exercice dans un but purement pédagogique qui a aussi pour objectif de vous faire prendre conscience qu'on ne peut se baser simplement sur l'émetteur annoncé d'un message pour être sûr de son origine réelle. Pour cela, il faudra utiliser la signature numérique qui fera peut-être l'objet d'un prochain TP.

[\[Corrigé\]](#)

1. Rappelons que le serveur SMTP en charge d'un domaine correspond au champ MX (*Mail eXchanger*) de l'enregistrement DNS de ce domaine.

7.4 Étude de la RFC de POP3

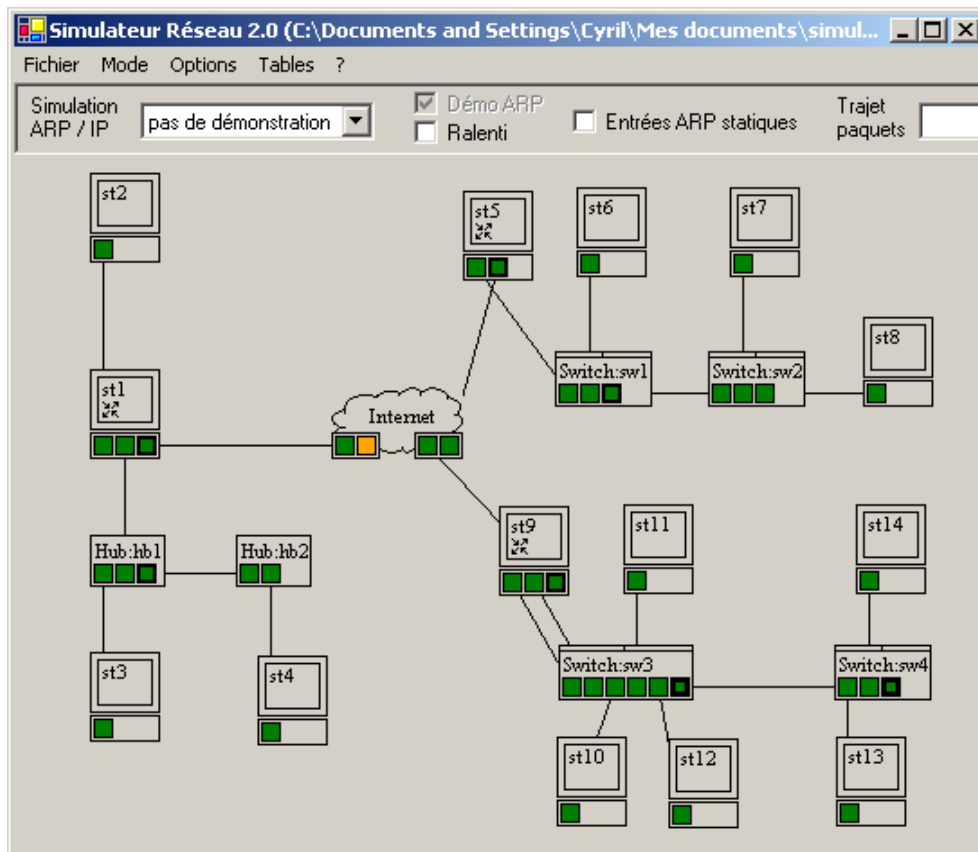
Exercice 24 (récupération de messages avec POP3)

1. S'envoyer 2 mails bien distincts sur allegro (procédure habituelle par **mail** ou **pine** ou tout autre client de messagerie).
2. En se basant sur la RFC de POP3 contenue dans le fichier `rfc1939.txt` téléchargé précédemment, contacter le serveur POP3 d'allegro, récupérer le deuxième message uniquement, puis le supprimer du serveur.
3. Utiliser la commande **mail** pour consulter le premier message et le détruire.

[\[Corrigé\]](#)

8 Simulateur : Nat/Pat et firewall

Soit l'interconnexion de réseaux suivante :



Celui-ci contient 5 réseaux (plus Internet et les FAI) :

- le réseau 192.168.2.0/24 constitué des stations st1 et st2
- le réseau 192.168.0.0/24 constitué des stations st1, st3 et st4
- le réseau 192.168.1.0/24 constitué des stations st5, st6, st7 et st8
- le réseau 192.168.5.0/24 constitué des stations st9, st11 et st13 appartenant au vlan (de niveau 2) 5
- le réseau 192.168.6.0/24 constitué des stations st9, st10, st12 et st14 appartenant au vlan (de niveau 2) 6

Les stations st1, st5 et st9 ont toutes des cartes d'accès distant les reliant à Internet et sont configurées pour servir de routeurs. Cependant les autres stations appartiennent à des réseaux privés et ne peuvent utiliser Internet.

Exercice 25 (Simulation Nat/Pat et firewall)

- Récupérer le fichier `reseau_depart.xml` et le charger dans le simulateur
- En mode *Transport*, activer le **Nat/Pat** sur st1, st5 et st9. Pour cela, effectuer un clic droit sur la station, choisir *Configuration IP* et cocher la case *Nat/Pat* puis sélectionner l'interface publique **ppp** (qui les relie à Internet) :

- Mise en place de serveurs :
 - faire écouter par st2 son port UDP 69 (serveur TFTP). Pour cela, en mode *Transport*, faire un clic droit sur st2 puis choisir *Tables* → *Ports écoutés* puis ajouter le port UDP 69 :

- faire écouter par st3 son port TCP 22 (serveur SSH)
 - faire écouter par st8 son port TCP 21 (serveur FTP)
 - faire écouter par st12 son port TCP 80 (serveur HTTP)
 - faire écouter par st13 son port TCP 22 (serveur SSH)
 - faire écouter par st14 son port TCP 21 (serveur FTP)
- Configuration des redirections de port (translations statiques) :
 - configurer st1 pour que l'accès aux serveurs de st2 et st3 soit possible depuis Internet. Pour cela, en mode *Transport*, faire un clic droit sur st1 puis choisir *Tables* → *Table Nat/Pat* et ajouter les entrées pour les serveurs de st2 et st3. Pour chaque serveur, il faut remplir une ligne de la table :

en spécifiant :

- le protocole de transport TCP ou UDP
- l'adresse IP privée du serveur (ici de st2 ou st3)
- le port privé du serveur (celui sur lequel il est en écoute)
- l'adresse IP publique (donc ici de st1)
- le port public (ici on peut prendre le même que le port du serveur)

(b) configurer st5 pour que l'accès au serveur de st8 soit possible depuis Internet

(c) configurer st9 pour que l'accès aux serveurs de st12, st13 et st14 soit possible depuis Internet

5. Tester l'accessibilité des serveurs depuis n'importe quelle station et que le serveur répond (en mode *Transport*, clic-droit sur la station émettrice puis *envoyer une requête/répondre à une requête*)

i On rappelle que l'envoi de la requête doit se faire vers l'IP publique et le port public du serveur...

6. Configuration d'un firewall (règles de filtrage ou *access lists*). Les règles de filtrage sont entrées sur une station/routeur en mode *Transport* en effectuant un clic droit sur la station/routeur puis choisir *Tables* → *Règles de filtrage*. Chaque règle dicte comment traiter un datagramme UDP ou segment TCP ou message ICMP reçu, et il faut remplir une ligne de la table :

N°	Entrée	Sortie	Prot.	Ip source/n	Pt sce	Ip dest/n	Pt dest	Action
01	*	*	*	*	*	*	*	Bloquer

valider annuler

en spécifiant pour quels datagrammes elle doit être appliquée :

- sur quelle carte en entrée (* pour toutes)
- sur quelle carte en sortie (* pour toutes)
- pour quel protocole parmi UDP, TCP ou ICMP (* pour tous)
- pour quel IP source (* pour toutes)
- pour quel port source (* pour tous)
- pour quel IP destination (* pour toutes)
- pour quel port destination (* pour tous)
- que faire du datagramme qui correspond à ces critères : Accepter ou Bloquer.

Pour un datagramme donné, la première règle qui correspond est celle appliquée.

Configurer les règles de st9 pour que :

- (a) tout le monde puisse atteindre les serveurs de la DMZ (serveur Web sur st12 et serveur FTP sur st14)
- (b) seul st1 puisse atteindre le serveur SSH (de st13)
- (c) laisser passer en entrée tout le trafic ICMP
- (d) bloquer tout autre type de trafic entrant
- (e) laisser le trafic sortant passer

7. Vérifier que ces règles fonctionnent en tentant d'envoyer des requêtes à ces serveurs et en les faisant répondre lorsqu'elles arrivent.

[\[Corrigé\]](#)