Réseaux – TP

RES 101

2014/2015

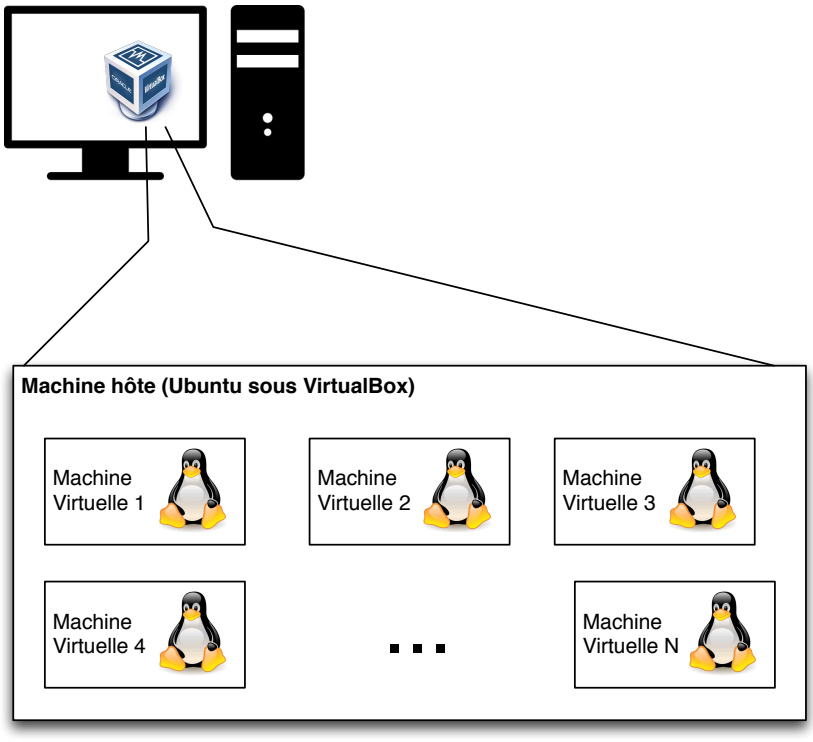
TP 1

Environnement de travail

1.1 Machines virtuelles

Lors de la plupart des travaux pratiques de cette unité d'enseignement, vous allez travailler dans un environnement virtualisé. En d'autres termes, vous allez émuler un réseau complet à l'intérieur de votre PC. Cette technique vous permettra d'avoir un accès administrateur à chacune des machines et à chacun des équipements réseaux virtuels afin de les configurer, d'analyser leur comportement, voire de les éteindre ou les mettre en défaut, chose qu'il serait impossible de faire sur un réseau de production comme celui de l'école.

La configuration qui sera celle de la plupart des TP est schématisée sur la figure ci-dessous.



Sur votre machine physique, vous devrez lancer un logiciel nommé *Virtual Box*[1] (distribué gratuitement par Oracle sous licence GPLv2). Cet outil émule un PC dans lequel vous lancerez *un autre* environnement Linux (Ubuntu) dont la configuration est légèrement différente de celle des machines des salles de TP. Vous avez notamment la possibilité d'exécuter certaines commandes en mode administrateur dans cette machine. Un autre avantage de cette approche est qu'il est possible de copier l'image du

[1] `https://www.virtualbox.org`

système linux de référence et de l'exécuter chez vous quelque soit votre système (Virtual Box existe pour Linux, Microsoft Windows, Mac OS X et Solaris).

C'est à l'intérieur de cette machine virtuelle que vous allez créer, pour chaque TP, un réseau de machines virtuelles légères, qui seront interconnectées entre elles.

Le matériel de TP (l'image de la machine virtuelle) est d'un volume assez conséquent (environ 6 Go pour l'édition 2014), il n'est donc pas à disposition en permanence sur un serveur de l'école. Vous pouvez toutefois en demander une copie afin de l'installer et de l'utiliser sur votre machine personnelle en écrivant à Claude Chaudet (Claude.Chaudet@telecom-paristech.fr).

1.2 Prise en main de l'environnement (Virtualbox / Ubuntu)

À vous :

1. Démarrez VirtualBox (cherchez éventuellement l'outil dans le menu des applications) et sélectionnez l'image RES101.

Une fois que vous avez démarré VirtualBox et sélectionné la bonne machine virtuelle (RES101), le système démarre et ouvre un environnement Linux classique. Vous pouvez, à cet instant, passer en mode plein écran, nous ne retournerons que rarement sur le système de base de la machine physique.

Vous pourrez remarquer sur la gauche du bureau plusieurs icônes, l'une d'entre elles vous permet d'ouvrir un terminal, c'est à partir de là que nous réaliserons la majorité des exercices. Pour ouvrir plusieurs fenêtres, effectuez un clic droit sur l'icône et sélectionnez "nouveau terminal". Une autre icône vous permet de lancer *Wireshark*, un outil d'analyse de trames que nous utiliserons souvent. Cet outil vous permet de capturer le trafic passant sur un réseau et de l'afficher, message par message, dans un format intelligible et interprétable. Il fait partie de la boîte à outils de base.

1.3 Manipulation des machines virtuelles légères (Netkit / UML)

Nous allons maintenant examiner les commandes nous permettant de manipuler les machines virtuelles légères qui nous serviront de base de TP. Ces machines sont crées dans un système de virtualisation qui se nomme *User Mode Linux*[2], grâce à une interface qui se nomme *Netkit* [3], développée par l'Université de Rome. Netkit fournit un certain nombre de commandes permettant de contrôler le réseau virtuel (le démarrer, l'arrêter, agir sur une machine, etc.) que nous allons entrevoir ici.

Nous allons maintenant démarrer notre premier réseau virtuel, très simple, composé de deux machines, A et B, interconnectées par un lien de communication. Ouvrez un terminal et rendez-vous dans le répertoire `TP_1`. Ce répertoire contient 3 fichiers texte et deux répertoires (vides) que vous pouvez examiner :

- Le fichier `lab.conf` contient la description du réseau. Les différentes machines sont identifiées par leur nom (A et B ici) et le fichier contient essentiellement des définitions de paramètres. Prenons l'exemple de la machine A :
  - `A[0] = "L"` signifie que la carte réseau numéro 0 de la machine A est connectée au lien de communication nommé "L". Il est possible de créer autant d'interfaces que nécessaire, par exemple dans ces TP, chacune des machines dispose aussi d'une interface numérotée 9 qui permet d'y accéder depuis le système hôte.
  - `A[con0] = "xterm"` signifie qu'au démarrage de A, le système doit ouvrir une fenêtre de terminal de type xterm sur cette machine.
  - Les autres paramètres (memoire, interface *tap*) sont au delà de ce qu'il est nécessaire de comprendre pour ces TP.

  Ce fichier décrit complètement le réseau. Netkit va le parcourir et configurer les machines ou les liens de communication comme spécifié. Il va, par ailleurs, créer les interconnexions entre machines en fonction des liens de communication indiqués dans ce fichier.

[2] `http://user-mode-linux.sourceforge.net`
[3] `http://wiki.netkit.org/`

- Deux fichiers, `A.startup` et `B.startup` qui sont des scripts exécutés lors du démarrage des machines éponymes. Toutes les machines ont la même configuration et c'est en particulier par le biais de ces fichiers qu'il est possible de distinguer les machines entre elles.

Netkit fournit (entre autres) les commandes suivantes qui vous seront utiles :

Commandes de manipulation des machines légères :

- `lstart` : permet de démarrer un réseau en totalité, machine par machine.
- `lstart A` : permet de ne démarrer que la machine A
- `lhalt` : permet d'arrêter proprement le réseau (commande d'extinction)
- `lhalt A` : permet d'arrêter proprement la machine A (commande d'extinction)
- `lcrash` : permet d'arrêter brutalement (débrancher) le réseau
- `lcrash A` : permet d'arrêter brutalement (débrancher) la machine A
- `vlist` : permet de lister les machines virtuelles démarrées
- `lclean` : permet de supprimer les fichiers temporaires (à exécuter à la fin de chaque session et lorsque l'on veut revenir dans les conditions initiales)

Attention : pour toutes ces commandes, il est très important de se positionner dans le bon répertoire, faute de quoi plusieurs machines seront lancées avec une configuration incorrecte. En cas d'erreur, arrêtez les machines au moyen de la commande `lcrash`, puis `lclean`, positionnez vous dans le bon répertoire et recommencez.

Démarrons maintenant notre premier réseau :

À vous :

2. Assurez-vous d'être dans le répertoire `TP_1` et tapez `lstart`.

Vous verrez deux fenêtres apparaître. Notez que ces fenêtres, une fois que le démarrage est terminé, comportent dans leur barre de titre le nom des machines. Vous pouvez essayer les différentes commandes ci-dessus à partir de la fenêtre principale. Pour revenir dans l'état initial, tapez successivement `lcrash` ; `lclean` et `lstart`.

Les fenêtres "A" et "B" vous fournissent une ligne de commande directe sur chacune des deux machines. Vous pouvez, par exemple, tester l'existence de B depuis A en tapant, dans la fenêtre A : `ping B`. Vous pouvez interrompre le test au moyen d'un Control-C.

Explorons maintenant les possibilités offertes par l'environnement :

- Il est possible de se connecter à chaque machine légère depuis l'environnement Ubuntu, par exemple pour disposer de fenêtres supplémentaires. Pour cela, ouvrez un nouveau terminal et tapez simplement `ssh A` pour vous connecter sur la machine A.
- Vous pouvez capturer le trafic réseau provenant et à destination d'une machine depuis l'environnement Ubuntu. Nous examinerons plus en détails les possibilités de cet outil ultérieurement. Pour capturer, par exemple, le trafic de A :

À vous :

3. Tapez, depuis une fenêtre de terminal Ubuntu, la commande `capture A`. L'outil Wireshark démarre alors.
4. Lancez à nouveau un test de A à B (`ping B` dans la fenêtre de *A*), et observer les messages interceptés qui s'affichent dans la fenêtre de Wireshark.

5. Interrompez le test au moyen d'un Control-C dans la fenêtre de *A*
6. Quittez Wireshark (inutile de sauver le résultat)
7. Interrompez la capture depuis le terminal Ubuntu au moyen d'un Control-C. **Attention, en arrêtant la capture, vous quitterez Wireshark s'il est encore ouvert. Par conséquent ne le faites pas alors que vous êtes en train d'analyser des trames**. Vous pouvez, par contre, sauvegarder le résultat de votre capture dans un fichier *pcap* et le reprendre ultérieurement en lançant Wireshark depuis la barre latérale.

- Il est possible de partager des fichiers entre les machines légères et la machine Ubuntu. Les machines légères disposent d'un répertoire `/hosthome` qui est un pointeur direct vers le répertoire racine (i.e. au dessus de `TP_1`) de la machine Ubuntu. Le répertoire `/hostlab`, quant-à-lui, pointe vers le répertoire du réseau courant (i.e. `TP_1`)

Commandes d'interaction entre l'environnement Ubuntu et les machines légères :

- `ssh A` : permet de se connecter à la machine A depuis l'environnement Ubuntu
- `capture A` : lance Wireshark en mode écoute du traffic de A. Pensez à interrompre la capture explicitement (Control-C) une fois que vous avez terminé.
- `/hosthome` et `/hostlab` : répertoires partagés entre les machines légères et la machine Ubuntu

À vous :

8. Une fois le TP terminé, pensez à arrêter les machines légères, et à nettoyer les fichiers temporaires.

TP 2

Protocoles applicatifs

2.1 Introduction

Dans ce TP, vous allez examiner le fonctionnement de plusieurs protocoles entre les applications classiques de l'Internet et vous familiariser avec la boîte à outils que nous utiliserons tout au long de ces séances. Selon votre aisance avec les environnements Unix, ce TP peut être assez long, il est toutefois important pour la suite du cours. Si vous n'avez pas le temps de le terminer durant la séance, l'environnement que vous utiliserez ne sera pas supprimé des machines, vous pourrez donc continuer ultérieurement. Les objectifs de ce TP sont les suivants :

- Faire vos premiers pas avec **l'environnement de travail** et avec les **outils classiques** permettant d'analyser le fonctionnement des protocoles et des réseaux.
- Comprendre ce qu'est un **protocole de niveau application** et constater quelques bonnes pratiques de mise en œuvre (par exemple : les codes de retour).
- Constater et comprendre le **principe d'encapsulation** (un message de niveau application est envoyé à la couche transport qui ajoute un en-tête avant de le transmettre à la couche réseau, qui à son tour rajoute un en-tête etc.)

À vous :

1. Entrez dans le répertoire TP_2 (`cd TP_2`) et tapez `lstart`.

Quatre terminaux vont s'ouvrir, deux fenêtres vous permettent d'interagir sur une machine virtuelle appelée `PC1` en tant qu'administrateur et deux autres sur une machine appelée `PC2`. Entre les deux machines, un réseau, que vous ne voyez pas, héberge un certain nombre de services.

2.2 Découvrir la boîte à outils réseaux

Pour des raisons de performance (les machines virtuelles ne disposent pas de beaucoup de mémoire), vous n'utiliserez que la ligne de commande sur PC1 et PC2. Vous disposez *a minima* des commandes suivantes que vous pouvez essayer :

- `ifconfig` vous permet de lister les interfaces réseau d'une machine. Notez, sur PC1 comme sur PC2 que vous disposez de deux interfaces (*i.e.* cartes réseau). `eth0` est l'interface principale, utilisant Ethernet et `lo` est une "fausse" interface permettant à un ordinateur de dialoguer avec lui-même. Cette interface, standard et appelée *loopback*, est présente sur tous les systèmes et possède l'adresse IP `127.0.0.1`. Elle permet de lancer un programme client et un programme serveur sur la même machine, de les faire dialoguer comme s'ils étaient sur un réseau, donc sans code spécifique, et ce sans être connecté à un quelconque réseau ni même disposer d'une carte réseau réelle.

  `ifconfig`, vous fournit plusieurs informations pour chaque carte réseau, notamment :

  - L'adresse MAC de la carte réseau (`HWaddr`, en hexadécimal)
  - L'adresse IPv4 de la carte réseau (`inet addr`) et son (ses) adresse(s) IPv6 (`inet6 addr`, en hexadécimal)
  - L'adresse de diffusion du réseau local (`Bcast`) qui est une adresse IP qui vous permet de joindre toutes les machines présentes sur le réseau local.

- `ping` vous permet de tester si une machine est active en spécifiant son nom ou son adresse IP, ainsi que de mesurer le délai d'aller-retour. Attention, certaines machines sont configurées pour ne pas répondre à cette sollicitation. Dans certaines configurations, vous pouvez envoyer une telle requête à l'adresse de diffusion mentionnée plus haut. Dans ce cas, toutes les machines pourront vous répondre (c'est le principe de la diffusion). Notez que la réponse à ce type de sollicitation en diffusion est généralement désactivée. En effet, elle permet de connaître les machines présentes sur le réseau, première étape d'un certain nombre d'attaques. Pour réaliser une interrogation d'une interface de diffusion, ajoutez l'option `-b` (`ping -b`). Essayez d'identifier les machines présentes sur le réseau à partir de PC1 ou de PC2.

- `nslookup` et `dig` vous permettent d'interroger un serveur d'annuaire (DNS), qui fournit la correspondance entre le nom d'une machine (par exemple pc1.res101.telecom-paristech.fr) et son adresse IP, (par exemple 10.0.1.1). Cette interrogation est, le plus souvent, réalisée pour vous automatiquement par le système. `dig` vous fournit beaucoup plus d'informations que `nslookup` et sa sortie peut être plus difficile à décoder au premier abord.

Vous manipulerez, dans la suite de ce TP, d'autres commandes et outils. En plus de ces différents outils, certains fichiers de configuration définissent les paramètres du réseau. Vous pouvez en afficher le contenu avec les commandes `cat` ou `less`. En particulier :

- `/etc/hostname` vous indique le nom de votre machine.

- `/etc/resolv.conf` spécifie les adresses IP des serveurs d'annuaire (DNS), le nom du domaine (i.e. du réseau) sur lequel votre machine est connectée. Le nom complet de votre machine, tel qu'elle est identifiée sur Internet, est formé du nom présent dans `/etc/hostname`, suivi du nom de domaine présent dans ce fichier, séparés par un point (par exemple `PC1.res101.telecom-paristech.fr`). La ligne `search` dans ce fichier indique le nom de domaine par lequel compléter une adresse incomplète (par exemple, lorsque vous tapez `ping pc1`, votre système va envoyer une requête à `pc1.res101.telecom-paristech.fr`).

- `/etc/services` liste l'ensemble des ports standard affectés aux différents services, avec leur nom. Un port est un nombre qui identifie un service sur une machine, permettant de différentier les applications et de renvoyer le trafic provenant du réseau au bon processus. Par exemple, au moyen des outils d'interrogation de la base DNS, notez l'adresse de `ftp.res101.telecom-paristech.fr` et de `www.res101.telecom-paristech.fr`. Que remarquez vous ? Les numéros de port les plus courants sont standardisés (attention, il ne s'agit pas d'une norme mais bien d'un usage courant), aussi on retrouvera très souvent le serveur Web sur le port 80 d'une machine, le serveur d'envoi d'e-mails sur le port 25, etc. Rien n'interdit toutefois de ne pas tenir compte de ces bonnes pratiques et d'utiliser le numéro qu'il vous plaît, même si certains usages sont fortement déconseillés.

- `/etc/hosts` contient une liste de correspondances entre adresses IP et noms de machines pour des machines d'usage courant. Si ce fichier est souvent relativement vide, il est possible d'y insérer quelques entrées de façon statique afin de ne pas avoir à interroger un serveur d'annuaire à chaque fois. Il faut toutefois noter que cette pratique présente des défauts : un changement dans l'adresse d'un serveur ne sera pas pris en compte.

Ces commandes sont semblables sous la plupart des systèmes. Sous un système Windows, `ifconfig` se nomme `ipconfig` et `dig` n'est pas installé par défaut. Les fichiers d'information sont souvent accessibles via d'autres chemins (ex : `C:\WINDOWS\system32\drivers\etc\hosts`). Notez qu'il existe de nombreuses interfaces graphiques vous permettant d'afficher ces informations et d'exécuter ces commandes.

2.3 Le dialogue client-serveur

Les applications réparties dans le réseau dialoguent en utilisant un ensemble de messages pré-définis afin d'obtenir un service. Par exemple l'affichage d'une page web statique (un fichier HTML) nécessite plusieurs étapes, dont en particulier :

1. Localisation du serveur, à partir d'un nom ou d'une adresse (IP). Si le nom seul est spécifié, il faudra d'abord déterminer l'adresse IP correspondante.
2. Ouverture d'un dialogue avec le serveur
3. Demande d'une page
4. Réception du contenu de la page ou d'un code d'erreur
5. Fin de la communication et affichage de la page

Le serveur n'est ni plus ni moins qu'un programme fonctionnant sur un ordinateur distant. Comme plusieurs programmes peuvent fonctionner sur la même machine, il est nécessaire, afin de s'assurer que l'on dialogue avec la bonne application, de spécifier, au moment de l'ouverture du dialogue, deux paramètres :

- le *nom du serveur* (par exemple `www.res101.telecom-paristech.fr`), qui identifie l'ordinateur sur lequel s'exécute le programme serveur que vous cherchez à joindre, ou son *adresse IP*.
- le *numéro de port* identifiant l'application sur le serveur.

L'exercice suivant consiste à dialoguer manuellement avec trois types de serveurs. Le premier serveur met en œuvre le protocole HTTP, base du Web. Le deuxième utilise le protocole SMTP utilisé lors de l'envoi d'un e-mail. Le troisième et dernier utilise le protocole FTP, destiné au téléchargement de fichiers.

2.4 Trafic Web : HTTP

HTTP (*Hyper Text Transfer Protocol*) est un protocole de niveau application destiné à la récupération de pages Web. Sa version la plus récente est décrite dans la RFC 2616, disponible publiquement sur le Web[1]. Pour cet exercice nous n'utiliserons qu'une toute petite partie de cette spécification. Commençons donc par récupérer une page simple, au moyen de la commande `telnet`. Cette commande permet d'ouvrir un dialogue avec un serveur et laisse, une fois cette première étape réalisée, le soin à l'utilisateur d'effectuer le reste du dialogue. C'est donc à l'utilisateur de saisir les commandes et d'interpréter les réponses.

À vous :

2. Affichez tout d'abord la page web fournie par le serveur `www.res101.telecom-paristech.fr` en utilisant le navigateur en ligne de commande `lynx` depuis PC1 en tapant : `lynx http://www`. Notez que vous pouvez utiliser le nom abrégé de la machine (`www`) ou son nom complet (`www.res101.telecom-paristech.fr`), indifféremment. En effet, les machines appartenant au même réseau, le système complètera automatiquement le nom abrégé. La page d'accueil du serveur s'affiche. Vous pouvez fermer le navigateur au moyen de la touche "Q" puis en confirmant.
3. Dans une fenêtre de terminal sur PC1 ou sur PC2, tapez `telnet www 80`
4. Une fois connecté, tapez `GET / HTTP/1.1` suivi d'un retour chariot, puis `Host: www.res101.telecom-paristech.fr` suivi de *deux* retours chariots. Ce faisant, vous demandez au serveur de vous envoyer la page d'accueil `/`, du site répondant au nom `www.res101.telecom-paristech.fr`.

Le texte qui s'affiche est le contenu de la page d'accueil du site, en code HTML. Le travail d'un navigateur, à ce stade, est alors d'interpréter ce code pour afficher la page formatée.

[1] `http://www.ietf.org/rfc/rfc2616.txt`

2.4.1 Capture et analyse du trafic HTTP

Nous allons maintenant répéter l'opération en insérant un analyseur de trame entre le client et le serveur afin de capturer les messages. Il existe plusieurs programme capables de réaliser cette tâche, tels que `tcpdump`, ou `wireshark`. Ces outils nécessitent un accès administrateur pour des raisons de sécurité que nous entreverrons plus bas[2].

À vous :

5. Démarrez la capture sur les interfaces de PC1 au moyen de la commande `capture` présentée dans le chapitre d'introduction.

6. Depuis une fenêtre de PC1, lancez le navigateur web afin de récupérer la page précédemment affichée. L'échange qui a eu lieu entre le client et le serveur s'affiche alors dans la fenêtre de l'analyseur.

   Dans la liste qui s'affiche, vous pouvez remarquer 3 types de paquets (colonne *protocol*) : des paquets correspondant à un échange DNS initial, des paquets estampillés "TCP" qui concernent l'établissement et la terminaison d'une connexion, et des paquets HTTP qui contiennent le dialogue entre le client et le serveur. La simple requête HTTP génère donc un nombre de messages beaucoup plus important que ce que vous voyez au niveau applicatif direct.

7. Sélectionnez un paquet de type HTTP dans la liste. La deuxième partie de la fenêtre vous affiche les en-têtes complets du paquet ainsi que son contenu et la troisième partie vous affiche tout le contenu du paquet, en hexadécimal sur la gauche et en ASCII sur la droite. Dans la deuxième partie, vous pouvez "déplier" les différents en-têtes pour en voir le détail, décodé. Cliquer sur une entrée vous indique, en surbrillance, les octets correspondants dans la partie basse.

En examinant cette liste d'en-têtes et leur positon dans la trame en dessous, vous pouvez constater un principe fondamental des réseaux IP : l'encapsulation. L'application (le navigateur pour ce message) génère le message que vous pouvez afficher en dépliant la rubrique "Hypertext Transfer Protocol". Elle passe ensuite cette suite d'octets à la couche transport (TCP) qui rajoute un en-tête contenant diverses informations telles que les ports source et destination (voir la rubrique "Transmission Control Protocol"). Cet en-tête est rajouté *avant* le paquet dans l'ordre d'émission des octets sur le réseau. TCP passe ensuite cet ensemble de données à IP qui rajoute les en-têtes de niveau 3 (cf. rubrique *Internet Protocol*), puis vient le tour d'Ethernet et enfin de la couche physique. L'opération inverse est réalisée à la destination, chaque couche examinant l'en-tête la concernant avant de le retirer pour passer le reste de la trame à la couche supérieure.

Vous devez comprendre, à ce stade, que les en-têtes du niveau n ne sont normalement utilisées **que** par le niveau n. En d'autres termes, TCP ne doit pas connaître l'en-tête Ethernet et vice-versa.

Notez aussi qu'avant de remplir un en-tête, une couche doit disposer de toutes les informations. Aussi, si l'interface utilisateur préfère manipuler les noms de machines aux adresses IP, il est indispensable de réaliser la correspondance entre nom et adresse *avant* de traiter le paquet. C'est pourquoi l'échange DNS survient avant les échanges applicatifs, de façon automatique.

À vous :

8. Wireshark dispose d'un ensemble d'outils de présentation vous permettant d'analyser un trafic complexe. Par exemple, il est possible d'afficher un dialogue complet (TCP) en effectuant un clic droit sur un paquet puis en sélectionnant "Follow TCP stream". Une nouvelle fenêtre s'ouvre, présentant l'échange décodé. Lorsque vous retournez sur la fenêtre principale, seule une partie des paquets capturés est affichée car Wireshark a automatiquement filtré les paquets faisant partie de l'échange sélectionné. Vous pouvez voir le filtre défini dans la boîte de texte au dessus de la liste des trames et supprimer ce filtre au moyen du bouton `clear`. Wireshark fonctionne essentiellement avec ce type de filtres, que vous aurez à spécifier vous-même pour

[2] Par conséquent il vous sera impossible de les utiliser sur les machines standard des salles de TP, hors des environnements virtualisés.

tirer pleinement parti de cet outil. Plusieurs didacticiels sont disponibles en ligne.

9. Sélectionnez maintenant la commande `Flow Graph` du menu `Statistics` (pour obtenir le menu, amenez votre pointeur tout en haut de l'écran). Dans le dialogue s'ouvrant alors, sélectionnez les options `All packets` ; `General Flow` et `Standard source / destination address`. Vous obtenez alors un chronogramme de la communication. Vous pouvez utiliser cet outil pour comprendre les échanges entre un client et un serveur, et donc suivre le protocole applicatif. Attention à bien distinguer ce qui relève de votre protocole, ici par exemple vous verrez probablement s'afficher des échanges DNS préalables.

10. Fermez la fenêtre de chronogramme et sélectionnez maintenant la commande `IO Graph` dans le menu `Statistics`. Une fenêtre s'affiche, changez la résolution de l'axe des abscisses (`tick interval`) pour passer à $1\,ms$, l'unité de l'axe des ordonnées pour sélectionner `bytes / tick` et changez le style de la première courbe (la seule) en passant en mode `FBar`. Vous obtenez ainsi un graphique représentant l'évolution du débit de la transmission, ce qui peut vous aider à analyser la performance d'une application.

11. Vous pouvez maintenant fermer Wireshark et interrompre la capture.

2.5 Service de messagerie électronique : SMTP

Maintenant, intéressons-nous au protocole SMTP destiné à l'envoi d'e-mails. Un serveur pour le protocole SMTP fonctionne généralement sur le port numéro 25 d'une machine dédiée. Nous utiliserons le serveur `smtp.res101.telecom-paristech.fr`. L'échange nécessaire à l'envoi d'un e-mail est le suivant :

- Le client ouvre une communication sur le port 25
- Le serveur, en cas de succès, renvoie un code de retour numéro 220, suivi de quelques renseignements dont nous ne tiendrons pas compte ici.
- Le client envoie ensuite la chaîne de caractères `HELO` suivie immédiatement du nom de sa machine. Le nom de la machine sur laquelle vous êtes connecté peut être obtenu, dans une fenêtre de terminal, en tapant la commande `hostname`
- Le serveur répond à cette commande par un message de retour commençant par 250 et suivi de son nom, indiquant qu'il accepte l'ouverture de la session.
- Le client indique ensuite l'adresse de l'émetteur du message (la votre) au moyen de la commande `MAIL FROM:` suivie de l'adresse e-mail complète (avec le @) de l'émetteur. Attention à respecter les espaces et les absences d'espace (il n'y a pas d'espace entre le FROM et les deux-points).
- Le serveur répond à cette commande par un message de retour commençant par 250, indiquant qu'il accepte cette adresse d'émetteur.
- Le client indique ensuite l'adresse destination, précédée par la commande `RCPT TO:` ; Indiquez ici comme destinataire `guest@res101.telecom-paristech.fr` le serveur répond une fois encore par le code 250.
- Le client saisit ensuite la commande `DATA` suivie immédiatement d'un retour chariot, le serveur répond alors par un code 354. Le client saisit alors le message à envoyer, ligne par ligne. La fin de la saisie est indiquée par un point (`.`) seul sur une ligne.
- Une fois le point final saisi, le serveur confirme la mise en file d'attente du message (en indiquant un numéro identifiant le message dans la file) au moyen d'un code 250 et le serveur est alors prêt à traiter un nouveau message.
- Le client met fin à la connexion en tapant la commande `QUIT`, à laquelle le serveur répond par un code 221.

Cet échange de message est spécifié dans le document RFC 821[3], disponible publiquement sur le Web. Il s'agit du dialogue effectué par votre client de messagerie lors de l'envoi d'un message électronique.

À vous :

12. Lancez une capture sur les interfaces de PC1.
13. Réalisez cet échange en utilisant `telnet` à partir de PC1 à destination du serveur `smtp`, en faisant attention aux codes de retour
14. Essayez, lors de l'échange, de taper une commande inexistante. Qu'obtenez-vous ?
15. Expliquez pourquoi la fin du corps du message est un point seul sur une ligne et n'est pas un retour chariot comme pour les autres commandes. Quel problème ce type de solution peut-il introduire ? Imaginez-vous des alternatives à cette solution ?
16. Pour consulter l'e-mail, sur PC1 ou sur PC2, lancez le programme `alpine`. Si le logiciel vous demande un mot de passe, il s'agit de `guest`. Sélectionnez alors Message Index pour voir s'afficher le message que vous avez envoyé. Pour quitter `alpine`, appuyez sur "Q" et confirmez.
17. Arrêtez la capture sous Wireshark et tracez le chronogramme représentant uniquement l'échange des messages SMTP entre le client et le serveur.

2.5.1 Transfert de fichier avec FTP

Le troisième et dernier protocole que nous allons examiner dans cet exercice est décrit dans la RFC 959[4] et est destiné au téléchargement de fichiers. Si le protocole HTTP peut s'acquitter de cette tâche, FTP est dédié à de plus gros transferts, utilisant les ressources du serveur pendant une période de temps plus longue. Aussi, afin de ne pas monopoliser le serveur pour une simple connexion, la plupart des serveurs FTP proposent de réaliser un échange initial de commandes sur le port dédié à ce protocole (le port 21), puis d'effectuer le transfert du fichier sur un port aléatoire, dédié à la communication.

Le protocole, pour transférer un fichier entre d'un serveur vers un client, est le suivant :

- Le client ouvre une communication vers le port 21 du serveur.
- Le serveur, en cas de succès, renvoie un message d'accueil spécifiant les conditions d'utilisation du serveur, suivi d'un code de retour numéro 220, lui-même suivi de quelques renseignements additionnels.
- Le client est ensuite supposé s'identifier. Il envoie la commande `USER`, suivie du nom d'utilisateur au serveur qui répond par un code 331 s'il accepte ce type de connexions.
- Le client transmet ensuite le mot de passe de l'utilisateur au moyen de la commande `PASS` suivie du mot de passe. Le serveur répond alors par un code de retour 230. L'utilisateur a alors accès à un système de fichier à la manière d'une ligne de commande Unix, les fonctions disponibles étant plus limitées.
- Une première commande affiche l'endroit de l'arborescence où l'on se trouve. Cette commande, comme pour une ligne de commande unix, est la commande `PWD`.
- Une fois que l'on sait où l'on se trouve, on peut vouloir lister les fichiers disponibles dans le répertoire courant. Cette opération s'effectue au moyen de la commande `LIST`. Lorsque vous essaierez cette commande, vous remarquerez qu'un message d'erreur indique qu'aucune connexion donnée n'a pu être établie. En effet, FTP utilise, pour renvoyer les résultats, une *seconde* connexion parallèle sur un numéro de port différent.

[3] `http://www.ietf.org/rfc/rfc821.txt`
[4] `http://www.ietf.org/rfc/rfc959.txt`

- Pour connaître ce numéro de port **qui change à chaque commande**, le client envoie la commande `PASV`. Il obtient alors plusieurs nombres entre parenthèses, dont les 4 premiers indiquent l'adresse IP du serveur auquel il êtes connecté et les deux derniers représentent le numéro de port sur lequel s'effectuera *le prochain* transfert de données.

  Pour identifier ce numéro de port, il considère les deux derniers nombres retournés par la commande PASV comme les octets de poids fort et de poids faible d'un nombre sur 2 octets. Il multipliez le premier par 256, puis ajoute le second. On obtient un nombre entier inférieur à 65536. Il ouvre alors une connexion sur le numéro de port indiqué.

- Le client affiche le résultat de la commande dans la 2e connexion, et envoie un code de retour 226 sur la première.

- Pour se déplacer dans l'arborescence, on utilise la commande `CWD` (*Change Working Directory*), suivie du nom du répertoire visé, spécifié à la manière des répertoires Unix.

- Pour un fichier dans le répertoire courant, le client envoie la commande `RETR`, suivie du nom du fichier. Le contenu du fichier ne sera pas transmis par la connexion principale, il lui faudra à nouveau envoyer une commande `PASV` au préalable et ouvrir une seconde connexion.

- Une fois le transfert terminé, le client ferme la connexion au moyen de la commande QUIT.

À vous :

18. Lancez une capture sur PC1.
19. Utilisez `telnet` pour démarrer l'échange à partir de PC1 à destination du serveur `ftp.res101.telecom-paristech`
20. Identifiez-vous en utilisant comme nom d'utilisateur `guest` et comme mot de passe `guest`
21. Affichez le chemin courant dans l'arborescence.
22. Utilisez la 2e fenêtre de terminal ouverte sur PC1 pour afficher la liste des fichiers dans le répertoire courant en utilisant la commande `LIST`. N'oubliez pas qu'il vous faudra, pour cela, ouvrir une seconde connexion vers le serveur sur un numéro de port que vous devrez calculer à la main (vous disposez d'une calculatrice sous Ubuntu).
23. Rendez-vous dans le répertoire `test`. En cas de succès, vous obtiendrez le code de retour 250.
24. Récupérez le fichier `TP.txt` dans ce répertoire, puis fermez la ou les connexions restées ouvertes.
25. Fermez la connexion.
26. Examinez votre capture réalisée sur PC1 sous Wireshark. Localisez la ligne correspondant à l'envoi du mot de passe. Que pouvez-vous en déduire quant aux risques liés à l'utilisation d'outils de capture par des utilisateurs ne disposant pas des droits administrateur ?
27. Voyez-vous, dans Wireshark, la liste des fichiers en réponse à une commande LIST ou le transfert du fichier ? Pourquoi ?

2.6 Jouer le rôle d'un serveur

Nous avons vu dans les sections précédentes le coté client de la connexion. Nous avons reproduit le dialogue tel que le mettrait en œuvre un navigateur web, un client de messagerie ou un client de transfert de fichiers. Nous allons maintenant jouer le rôle d'un serveur en répondant aux requêtes d'un navigateur Web. Pour cela vous allez utiliser un programme appelé `netcat` qui est le pendant de `telnet` coté serveur.

La différence fondamentale entre les deux modes de fonctionnement (client ou serveur), est qu'un client, lorsqu'il démarre, va émettre une requête, et donc envoyer des messages dans le réseau et s'attendre

à une réponse. Un serveur, en règle générale, est silencieux lors de son démarrage. Il se contente de réserver des ressources au niveau du système qui l'héberge et d'attendre les connexions auxquelles il répondra.

À vous :

1. Tapez la ligne de commande de PC1 `nc -l -p12345` sur la ligne de commande. L'option `-l` indique que netcat doit se mettre en mode écoute (*listen*) et l'option `-p` permet de spécifier le numéro de port sur lequel écouter. Ce faisant, vous indiquez à PC1 qu'il faut écouter tout ce qui arrivera sur ce le port 12345 et l'afficher.
2. Testez en tapant, dans une fenêtre de PC2, `telnet pc1 12345` puis en envoyant quelques mots dans un sens et dans l'autre (depuis la fenêtre `telnet` et depuis la fenêtre `nc`).
3. Assurez-vous que le programme `nc` est toujours en fonction sur PC1 (sinon relancez-le) puis ouvrez sur PC2 le navigateur en ligne de commande `lynx` avec comme argument `http://pc1:12345`. Que constatez-vous au niveau du navigateur et au niveau de `nc` ? Commentez.
4. Tapez maintenant dans la fenêtre de `nc` les commandes suivantes, à la suite les unes des autres, en terminant par un saut de ligne (sans espaces au début des lignes). Observez à chaque étape ce qu'affiche le navigateur :

```
HTTP/1.1 200 OK
Content-Location: index.html.fr
Accept-Ranges: bytes
Content-Length: 42
Content-Type: text/html
Content-Language: fr

<html><body><h1>Bravo !</h1></body></html>
```

5. Quittez ensuite le programme serveur au moyen d'un `Control-C`.
6. Bravo, vous venez de jouer le rôle d'un serveur Web. Observez les différentes lignes de la réponse que vous avez envoyée au navigateur et notez que la réponse comporte plus que le simple code de la page HTML. Ces différentes lignes servent à indiquer au navigateur le type de réponse qu'il obtient, le protocole HTTP pouvant être utilisé de différentes manières.

2.7 Pour aller plus loin : analyse du traffic DNS

Vous devez maintenant avoir acquis une bonne maîtrise des outils de capture de trafic et d'analyse. Essayez de comprendre le fonctionnement du protocole DNS en lançant une capture sur PC1, puis en interrogeant explicitement le DNS en tapant dans l'autre fenêtre de PC1 `nslookup pop`. Une fois la réponse obtenue, arrêtez la capture et ouvrez le fichier correspondant sous Wireshark.

À vous :

1. Trouvez la requête DNS et la réponse
2. Quelles est l'adresse IP du serveur DNS ?
3. Quel est le format de la requête ?
4. En examinant la trame "brute" (en hexadécimal et en ASCII à coté), pouvez-vous déterminer la façon dont est codé le nom de domaine transmis au serveur DNS ? Examinez en particulier les valeurs remplaçant les points et n'hésitez pas à vous aider des ressources web.
5. Quel est le format de la réponse ? Que remarquez-vous ?

Attention

Pensez à arrêter les machines virtuelles (`lcrash`) et à nettoyer les fichiers temporaires (`lclean`) une fois que vous avez terminé afin de préserver l'espace disque.

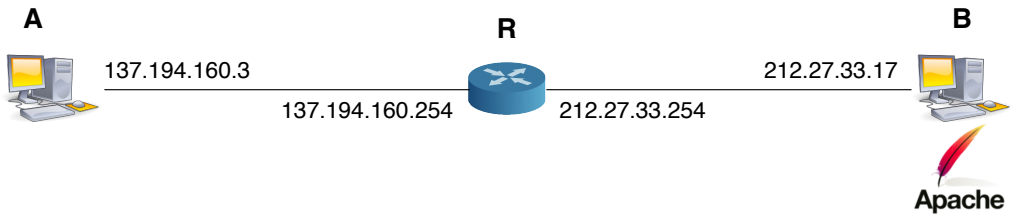
TP 3

Protocoles de transport

Introduction

Ce TP porte sur le transport de bout en bout. Dans un premier temps, vous ajouterez à votre arsenal plusieurs outils faisant partie de l'attirail classique des réseaux : `tcpdump` et `iperf`, que vous utiliserez pour examiner le comportement de TCP entre deux machines.

Commencez par lancer l'environnement virtualisé et positionnez-vous dans le répertoire `TP_3`. Lancez les machines virtuelles, vous obtenez des fenêtres d'interface sur les machines du réseau représenté ci-dessous :



Lors de cette première partie, vous allez évaluer le fonctionnement du protocole de transport TCP. Pour cela, vous téléchargerez, depuis la machine A, un fichier fourni par un serveur Web situé sur la machine B tout en capturant le trafic sur A et en le sauvegardant. Nous pouvons pour cela procéder comme précédemment avec Wireshark et sauvegarder le résultat de la capture dans un fichier au format PCAP ou utiliser un outil en ligne de commande directement depuis les machines légères : *tcpdump*.

3.1 Capturer le trafic avec tcpdump

Plusieurs outils d'analyse de trame existent, Wireshark est l'un des plus répandus car il existe sur de nombreuses plate-formes. Toutefois, il vous faut disposer d'une interface graphique pour l'utiliser, ce qui n'est pas toujours le cas. Par exemple, lorsque vous administrez un serveur à distance, le débit de la connexion ne vous permet pas forcément d'afficher le bureau de la machine distante.

C'est ici que des outils tels que `tshark` ou `tcpdump` vous seront utiles. Ces outils vous affichent les trames au fur et à mesure dans le terminal, en mode texte. Il est possible de les paramétrer pour filtrer les trames intéressantes uniquement ou pour afficher plus ou moins de détails sur les paquets transmis (le contenu, tel ou tel en-tête, etc.).

Ces outils peuvent aussi être configurés pour sauvegarder le résultat de la capture dans un fichier dans un format standard, `pcap`, qui peut alors être analysé a posteriori en utilisant des outils graphiques. Par exemple, vous pouvez, sur une machine virtuelle légère, taper la commande suivante :

```
tcpdump -i eth0 -s 0 -w /hosthome/capture.pcap port 80 or port 53.
```

Vous lancez alors le programme d'écoute `tcpdump` auquel vous donnez l'instruction d'écouter tout ce qui

se passe sur les ports 80 (HTTP) et 53 (DNS) de l'interface `eth0` (option `-i`) et de le sauvegarder dans le fichier `/hosthome/capture.pcap`. Il sera alors directement accessible dans le répertoire racine de la machine virtuelle Ubuntu et vous pourrez l'ouvrir avec Wireshark.

À vous :

1. Sur la console de A, démarrez `tcpdump` **en tâche de fond** pour capturer le trafic et le sauvegarder, par exemple, dans `/hostlab/capture.pcap`. Récupérez ensuite le fichier `test.pdf` depuis la machine B en tapant sur la console de A, `wget http://B/test.pdf`. Notez que l'utilisation de tcpdump bloque votre terminal, il vous faudra soit le lancer en tâche de fond, soit utiliser le programme `screen` pour le lancer dans un terminal virtuel (regardez le Web pour trouver la façon d'utiliser ce programme) ou alors vous connecter à la machine A depuis la machine hôte en tapant `ssh A`.

   `wget` se substitue au navigateur pour effectuer un échange HTTP et sauvegarder le fichier téléchargé sur le disque, vous affichant à l'issue de l'échange le débit moyen atteint.

2. Interrompez `tcpdump` au moyen d'un Control-C.

3.2 Exploitation de la capture

Une fois la capture réalisée, nous allons maintenant l'analyser au moyen de Wireshark, ainsi que d'un outil spécifique appelé `tcptrace`. Ces deux outils sont disponibles sur la machine Ubuntu, nous n'aurons pas besoin d'interagir ici avec les machines légères.

À vous :

3. Localisez, sur la machine hôte, l'endroit où a été sauvegardé le fichier `capture.pcap` (dans le répertoire du TP si vous avez indiqué `/hostlab` comme racine précédemment, dans `/home/student` si vous avez indiqué comme `/hosthome`).

4. Ouvrez le fichier sous Wireshark (menu fichier ; ouvrir). Localisez le dialogue d'ouverture de session. Vous pouvez les identifier manuellement dans la liste, ou utiliser des filtres :

   - L'expression `tcp.flags.syn == 1` vous permet de ne sélectionner que les paquets pour lesquels le drapeau SYN est positionné
   - L'expression `tcp.flags.ack == 1` vous permet de ne sélectionner que les paquets pour lesquels le drapeau ACK est positionné
   - L'expression `tcp.flags.fin == 1` vous permet de ne sélectionner que les paquets pour lesquels le drapeau FIN est positionné
   - Vous pouvez combiner ces filtres au moyen des opérateurs `&` ("et") et `|` ("ou").

5. Quels sont les numéros de séquence échangés dans ces paquets ? Pourquoi les numéros de séquence initiaux du client et du serveur sont-ils différents ? Comment évoluent ces numéros de séquence durant l'échange initial ?

6. Quel est le premier message relatif à un protocole applicatif ? Quel est le numéro de séquence initial de ce segment ? Quelle est la différence entre ce numéro de séquence et le numéro initial tiré par le client ? Pourquoi ?

7. Localisez le premier acquittement émis par le serveur. Quel est son numéro de séquence et quelle est la différence entre ce numéro de séquence et le numéro de séquence initial tiré par le serveur ? Pourquoi ?

8. Comparez le numéro de séquence final du segment contenant la requête du client avec le numéro de l'acquittement correspondant. Que constatez-vous ? Que représente, en fait, le numéro d'acquittement ?

9. Localisez les fermetures des connexions. Le dialogue s'effectue-t-il en 4 messages comme indiqué en cours ? Que se passe-t-il en réalité ici ?

Affichage des numéros de séquence sous Wireshark :

Le numéro de séquence initial d'une connexion TCP est tiré aléatoirement. Vous avez pu constater ici que l'analyse du comportement de TCP était fastidieuse. Wireshark, par défaut, représente non pas les numéros de séquence réellement échangés lors d'un échange TCP, mais des numéros de séquence relatifs, démarrant à 0 pour le premier message SYN. Cet affichage simplifié a été désactivé dans les TP pour vous permettre de constater les numéros réellement échangés.

Vous pouvez activer ou désactiver cet affichage relatif en vous rendant dans les options de l'outil, dans la section `Protocoles` puis `TCP` et en cochant ou décochant la case *Relative Sequence Numbers*.

3.3 Débit de bout en bout

Plusieurs outils permettent de tester la performance d'un réseau. L'un d'entre eux, nommé `iperf`, permet d'évaluer le débit d'une connexion de bout en bout. Pour tester la connexion de A à B, il faut commencer par lancer un programme serveur sur B. Une fois ce serveur lancé, on peut lancer, à partir de A, plusieurs tests.

Usage de iperf :

Pour des tests en utilisant TCP :

- Lancer le serveur : `iperf -s`
- Démarrer le test vers un serveur localisé sur B : `iperf -c B`
- Démarrer un test dans les deux directions : `iperf -c B -d`

Pour des tests en utilisant UDP, il est nécessaire de spécifier le débit à émettre. En effet, sous TCP il n'est pas nécessaire de limiter le débit du flux étant donné que TCP opèrera une régulation naturellement. Pour UDP, iperf ne peut pas décider tout seul du débit qui correspondra le mieux au scénario à évaluer. Par conséquent, il choisit une valeur par défaut d'1 Mb/s, ce qui peut être en dessous du débit d'un lien effectif. Si l'on choisit un débit supérieur à la capacité du lien iperf s'adaptera, ne cherchez donc pas à émettre un flux à 100 Mbit/s sur un lien à 1Mb/s.

- Lancer le serveur (UDP) : `iperf -s -u`
- Démarrer le test vers un serveur localisé sur B au débit par défaut : `iperf -c B -u`
- Démarrer le test vers un serveur localisé sur B avec un flux à 10mb/s : `iperf -c B -u -b 10m`

À vous :

12. Comparez la performance de TCP et d'UDP (utilisez un débit de 10 Mb/s) sur le lien de A à B. Comment expliquez-vous le résultat ? Pensez-vous que TCP ait besoin d'être optimisé pour mieux utiliser les ressources ou qu'il s'agit d'un choix délibéré ?

TP 4

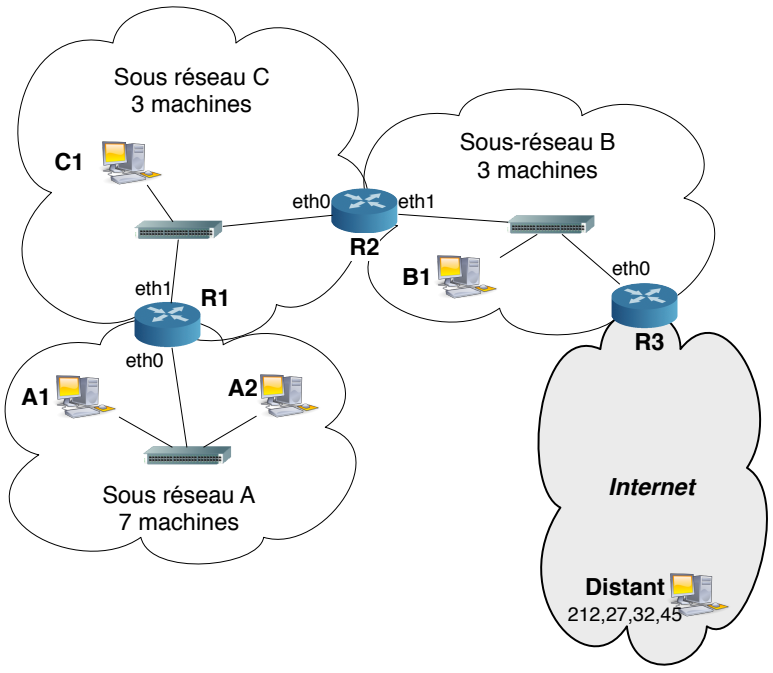
Adressage IP

Introduction

Durant ce TP, prévu pour une séance d'environ 2h, vous aurez à concevoir un plan d'adressage pour un réseau de petite taille et à configurer les différents équipements pour que l'acheminement des paquets se passe correctement.

4.1 Création d'un plan d'adressage

Commencez par lancer l'environnement virtualisé, et positionnez-vous dans le répertoire TP_4. Lancez les machines virtuelles légères au moyen de `lstart`. Le réseau représenté sur la figure suivante démarre (vous obtenez une console vers chacune des machines intéressantes et deux sur $A1$) :



Votre première tâche sera de concevoir le plan d'adressage du réseau qui vous est confié, c'est-à-dire les différentes adresses et masques à affecter à chaque sous-réseau et à chaque machine. Ce réseau comprend les 3 sous réseaux, A, B et C représentés sur la figure. Vous n'avez pas à vous préoccuper de la partie "Internet", ni du PC *Distant* qui sont déjà configurés tel qu'indiqué sur la figure.

Pour cela, il vous faudra évaluer la taille de l'espace d'adressage dont vous disposez et le "découper" en parts suffisantes. D'après la façon dont sont définies les adresses des sous-réseaux, vous ne pouvez créer que des plages de 2^k machines (2, 4, 8, 16, 32, etc.). Par conséquent, il vous faudra découper votre plage d'adresses en blocs de taille supérieure au nombre de machines, quitte à laisser de l'espace vide. Il vous faudra, en outre, prendre en compte le fait que :

- L'une des adresses est réservée pour représenter l'adresse du réseau et ne peut correspondre à aucune machine (il s'agit de l'adresse commençant par le préfixe du sous-réseau et dans laquelle tous les bits de l'identifiant d'interface sont à 0).
- L'une des adresses est réservée pour la diffusion (*broadcast*) et ne peut correspondre à aucune machine (il s'agit de l'adresse commençant par le préfixe du sous-réseau et dans laquelle tous les bits de l'identifiant d'interface sont à 1).
- Les équipements opérant au niveau de la couche réseau (les routeurs) doivent disposer d'une adresse par interface réseau active, cette adresse faisant partie de la plage du réseau dans lequel l'interface est branchée.
- Les équipements opérant au niveau de la couche liaison (commutateurs) n'ont pas besoin d'adresse IP. Certains en possèdent une pour permettre d'y accéder à distance afin de les configurer, mais nous supposerons ici que ces équipements n'en disposent pas.

À vous :

1. Quelles sont les tailles *minimales* des plages d'adresses à affecter à chacun des sous-réseaux de la figure ci-dessus ?
2. Quelle est, par conséquent, la taille minimale de la plage d'adresse globale à affecter au réseau ?
3. Vous avez, à votre disposition, la plage d'adresses 137.194.23.32/27 pour l'ensemble de votre réseau. Etablissez le plan d'adressage en divisant cette plage en sous-plages pour les sous-réseaux A, B et C. Indiquez, pour chaque sous-réseau, son adresse, son masque, l'adresse minimale et l'adresse maximale à affecter à des machines. Combien de configurations distinctes sont possibles ?
4. Une fois le plan d'adressage défini, mettez-le en place sur toutes les machines, *y compris les routeurs*, en fonction de leur sous-réseau d'appartenance. Pour configurer une adresse sur une machine, utilisez la commande `ifconfig` (voir l'encadré ci-dessous).

   Les bonnes pratiques consistent à affecter aux routeurs les adresses les plus "hautes" de l'espace d'adressage et aux machines les plus basses. Cependant, il ne s'agit que d'un usage et cela n'est pas obligatoire.
5. Essayez, depuis la machine $A1$, d'émettre une requête `ping` à destination de la machine $A2$ (en utilisant son adresse IP, il n'y a pas de service DNS). Essayez maintenant de joindre la machine $B1$. Que se passe-t-il ?

Configurer une interface réseau :

Configurer une interface réseau sous Unix peut être réalisé via la commande `ifconfig` qui nous permettait précédemment d'afficher les paramètres d'une telle interface. La syntaxe, pour l'interface eth0 est la suivante :

```
ifconfig eth0 A.B.C.D netmask E.F.G.H
```

où `A.B.C.D` est l'adresse IP de l'interface et `E.F.G.H` est le masque de sous-réseau représenté sous forme décimale (par exemple, pour un réseau /25, le masque est noté 255.255.255.128)

4.2 Configuration des routes

Votre réseau n'est pas encore fonctionnel car les routeurs ne sont pas configurés. En effet, dans cette configuration, les routeurs ne mettent en œuvre aucun protocole de routage et sont donc incapables d'auto-configuration. Vous aurez par conséquent à configurer ces différents routeurs manuellement en saisissant les entrées pertinentes de la table de routage.

Vous pouvez afficher la table de routage d'un PC ou d'un routeur en tapant la commande `route -n`. Vous noterez qu'un ensemble d'entrées sont affichées sous la forme d'un tableau comportant plusieurs colonnes, dont notamment :

- *Destination* correspond à l'adresse du réseau que cette entrée concerne.
- *Genmask* indique le masque du réseau que cette entrée concerne.
- *Gateway* contient l'adresse du prochain routeur, jouant le rôle de passerelle vers ce réseau.

À vous :

6. À partir de ces informations, expliquez pourquoi la machine $A1$ pouvait joindre la machine $A2$ dans la question précédente.
7. Dans l'état actuel, les routeurs sont suffisamment bien configurés pour permettre un certain niveau de communication locale. Les terminaux, quant-à-eux, peuvent joindre les machines appartenant à leur propre sous-réseau mais ne savent pas comment sortir de ce sous-réseau. Il leur manque une *route par défaut* qui indique vers quelle passerelle diriger les paquets pour lesquels aucune entrée explicite dans la table de routage ne correspond. Pour configurer la route par défaut de chaque terminal, il vous suffit de taper `route add default gw A.B.C.D`, où `A.B.C.D` est l'adresse IP de la passerelle. Configurez les passerelles par défaut des différents terminaux et routeurs ($R2$ pour C, $R1$ pour A, $R3$ pour B).
8. lancez `tcpdump -e -t` sur la machine C1. Essayez d'émettre une requête `ping` de $A1$ à $C1$. `ping` utilise un simple échange requête-réponse : la machine initiatrice envoie une requête *Echo request* en utilisant le protocole ICMP et la machine réceptrice y répond simplement au moyen d'un message *Echo reply* faisant aussi partie d'ICMP. Que constatez-vous ? Quelle interprétation pouvez-vous fournir ?

Il ne suffit pas de configurer les routes par défaut sur les différents terminaux. En effet, dans ce cas, tout paquet destiné à un sous-réseau différent sera émis à la passerelle par défaut. Par conséquent la réponse à la requête `ping` sera envoyée à $R2$ qui ne saura pas quoi en faire. Il vous faut configurer, sur les différents routeurs, les chemins vous permettant de joindre les différents sous-réseaux.

La commande route :

La commande `route` permet d'interagir avec la table de routage locale d'une machine. Le plus souvent, cette table ne contiendra qu'une ou deux entrée : une route par défaut, ainsi qu'une directive identifiant toutes les machines qui n'ont pas besoin de routage pour être atteintes, c'est-à-dire celle appartenant au même sous-réseau.

Pour afficher les routes, tapez simplement :

```
route -n
```

L'option -n désactive la résolution des adresses en nom (DNS inversé) et accélère donc souvent l'affichage.

Pour ajouter une telle entrée dans la table de routage d'un routeur, il vous faut taper

```
route add -net A.B.C.D netmask E.F.G.H gw I.J.K.L dev ethX
```

où `A.B.C.D` est l'adresse du réseau visé, `E.F.G.H` le masque dudit réseau, `I.J.K.L` l'adresse du prochain routeur et `ethX` l'interface permettant de joindre `I.J.K.L`.

À vous :

9. Configurez si nécessaire les routes permettant de joindre C et A dans les routeurs $R1$, $R2$ et $R3$. Certaines routes sont directement configurées lors de l'initialisation de l'interface.

10. Affichez la table de routage du routeur $R2$. Remarquez les entrées correspondant à chacun des réseaux auquel ce routeur est directement connecté, les entrées correspondant à un réseau distant et l'entrée correspondant à la route par défaut. Notez que parfois, une adresse `0.0.0.0` est utilisée.

11. Expliquez, pour le cas de la route par défaut, comment cette adresse est utilisée dans la procédure d'acheminement *longest prefix match*. Dans quel ordre les entrées sont-elles examinées dans la table ?

TP 5

Address Resolution Protocol

Dans ce TP, prévu pour une séance d'environ une heure, nous allons examiner le fonctionnement du protocole ARP permettant de réaliser la correspondance entre adresses IP et adresses MAC. Pour ce TP, nous allons reprendre le réseau que nous avons configuré précédemment.

Positionnez-vous dans le répertoire TP_5 avant de lancer les machines légères. Ce répertoire contient une configuration fonctionnelle du réseau que nous avons créé au TP précédent. Si votre réseau fonctionnait précédemment, vous pouvez tout à fait réaliser ce TP sur votre propre configuration.

5.1 Portée des adresses IP et MAC

Vérifiez que votre réseau est opérationnel en réalisant un échange `ping` entre plusieurs couples de stations appartenant aux mêmes réseaux et à des réseaux différents.

À vous :

1. Utilisez wireshark pour capturer le trafic correspondant à une requête `ping` entre $A1$ et $A2$. Quelles sont les adresses IP et MAC source et destination des paquets ? À quelles stations correspondent elles (vous pouvez vous aider de `ifconfig` pour déterminer les adresses MAC des différentes stations) ?

2. Répétez l'expérience entre $A1$ et $B1$, puis entre *Distant* et $B1$. Que remarquez-vous à propos des adresses IP et MAC ? Expliquez.

3. Envoyez maintenant une requête `ping` à partir de $A1$ en *broadcast* sur le réseau A en examinant le trafic au niveau de $A2$. Il vous faudra, pour réaliser cette opération, taper `ping -b A.B.C.D` où `A.B.C.D` est l'adresse de diffusion du sous-réseau. Que remarquez-vous sur les adresses IP et MAC destination des paquets émis ? Quelles sont les adresses IP et MAC destination des réponses à la sollicitation ?

4. Essayez enfin d'émettre une requête `ping` à partir de $A1$ à destination d'une machine n'existant pas dans le réseau A en examinant le trafic émis et reçu sur la seconde fenêtre de $A1$. Que se passe-t-il ? Essayez maintenant d'envoyer une requête à destination d'une machine n'existant pas dans le réseau B. Qui envoie le message d'erreur ?

Nous venons de constater que les adresses IP dans les paquets ne changeaient pas (du moins dans ce scénario qui correspond au cas général), alors que les adresses MAC, elles, changeaient. Les adresses IP identifient la source originelle et la destination finale des données, alors que les adresses MAC servent à franchir les différents réseaux et sous-réseaux traversés en identifiant les points d'entrée et de sortie de ces réseaux.

Lorsqu'un paquet à destination d'une adresse IP `A.B.C.D` arrive au niveau d'un routeur, ce dernier examine sa table de routage pour déterminer l'adresse du prochain routeur et le réseau sur lequel envoyer le paquet pour joindre ce prochain routeur. Il forme ensuite une trame de niveau 2 en spécifiant comme destination l'adresse MAC du prochain routeur. Il doit donc être capable d'établir une correspondance

entre l'adresse IP contenue dans la table de routage et l'adresse MAC. C'est le travail du protocole ARP (*Address Resolution Protocol*). Notez qu'ARP ne se limite pas aux échanges entre routeurs mais réalise toutes les correspondances entre adresses IP et adresses MAC dans un sous-réseau.

ARP fonctionne en examinant les paquets qui sont reçus, mais aussi et surtout au moyen d'un mécanisme de requêtes et de réponses et d'une table (un cache) qui stocke les résultats des requêtes récentes. Cette table est souvent appelée *cache ARP*.

À Vous :

5. Affichez le cache ARP de chaque machine au moyen de la commande `arp`. Les tables en question doivent être vides, si elles ne le sont pas, vous pouvez supprimer des entrées de la table au moyen de la commande`arp -d`. Vous noterez que la table, dans ce cas, n'est pas vide mais que les adresses MAC sont invalidées.

   Les entrées dans cette table expirent et disparaissent au bout de 3 minutes sans activité réseau. Si vous souhaitez vider réellement le cache ARP, sans patienter, il vous faudra redémarrer le TP en tapant, dans la fenêtre qui vous a servi à lancer le TP la commande `lcrash` suivie de `lstart`.

6. Depuis la machine $A1$, envoyez un seul paquet `ping` à destination de la machine $A2$ en tapant `ping -c 1` suivi de l'adresse IP de $A2$. Affichez ensuite le cache ARP de $A1$ et comparez l'adresse MAC affichée à l'adresse MAC de la carte `eth0` de $A2$. Affichez maintenant le cache ARP de $A2$, que remarquez-vous ? Expliquez à quel moment chacune des deux machines a pu avoir connaissance de l'adresse de l'autre.

7. Testez maintenant, depuis $A2$, si la machine $A1$ est active. Examinez le cache ARP des deux machines et du routeur $R1$ et indiquez les entrées correspondant à cette communication. Commentez et expliquez les différentes entrées.

8. Videz tous les caches ARP. Vous pouvez, pour cela, taper la commande `ip -s neigh flush all` sur *chacune des machines* (A1 et A2 en tout cas) ou en redémarrant les machines virtuelles en tapant, dans la fenêtre initiale, `lcrash` suivi de `lstart`.

9. Avant d'exécuter la moindre commande utilisant le réseau (`ping`, etc.) lancez plusieurs capture avec wireshark : sur le routeur $R1$ ainsi que sur $A1$ et sur $R2$.

10. Depuis la machine $B1$, envoyez des requêtes `ping` à destination de $A1$. Identifiez les premières trames ARP sur les différentes machines (avant le premier paquet émis par `ping`), représentez graphiquement et expliquez cet échange.

11. Re-exécutez la commande `ping` précédente deux fois successives. Pourquoi n'y a-t-il pas de message ARP échangé entre les deux échanges ICMP ?

TP 6

Architecture de l'Internet

6.1 Internet et systèmes autonomes

L'objectif de ce TP est d'explorer une partie de la topologie d'Internet au moyen de méthodes expérimentales et de se familiariser avec les outils et bases de données disponibles à cet effet sur le Web. Les différentes questions vous amèneront à déterminer la topologie environnant le système autonome de l'école (AS 1712) et le système autonome numéro 2607 (*Slovak Academy Network*), choisi car il met à disposition des outils libres d'utilisation et possède un environnement intéressant.

Lors de cette étude, vous utiliserez un navigateur web et l'outil `traceroute` depuis votre poste de travail directement. Ce TP ne se déroulera pas en environnement virtualisé. Même si cela pourrait avoir un intérêt (vous pourriez observer les requêtes qui sont envoyées par `traceroute`), les machines virtuelles n'ont pas d'accès au réseau global. Si vous êtes sur une machine Windows, l'outil `traceroute` se nomme `Tracert` et est accessible via la ligne de commande DOS.

Vous chercherez à découvrir une partie de la topologie de l'Internet au niveau de l'interconnexion des systèmes autonomes. Vous aurez à disposition un certain nombre d'outils pour ceci, et notamment des bases de données manipulées par le protocole BGP (*Border Gateway Protocol*).

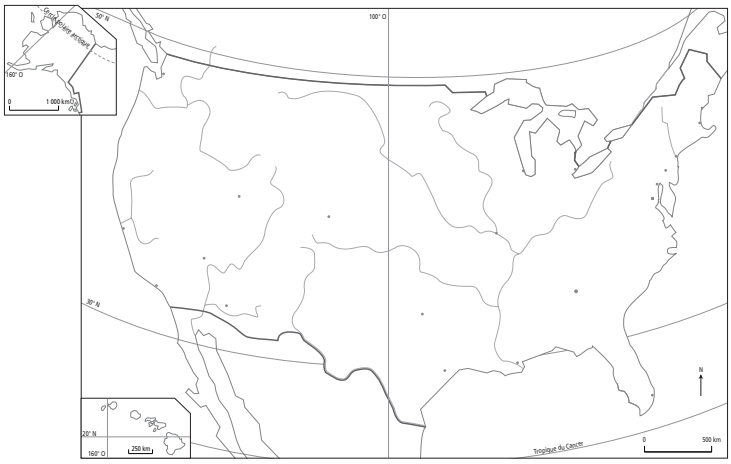
6.1.1 Questions préliminaires

`traceroute` fonctionne en exploitant le champ *Time to Live* de l'en-tête IP. Des paquets sont envoyés vers la destination avec une valeur positionnée tout d'abord à 1. Le premier routeur décrémente ce compteur et, le voyant arriver à 0, renvoie un message d'erreur à la source. `traceroute` voyant ce message d'erreur arriver, il est capable d'identifier le premier routeur qui en est l'émetteur. `traceroute` recommence alors avec un *Time to Live* égal à 2 etc.

À vous :

1. Depuis votre station de travail, déterminez le chemin emprunté par les paquets à destination du domaine `stanford.edu` (Stanford University, Californie) Pour ce domaine, vous obtiendrez une liste des routeurs traversés par les paquets ICMP successifs. Chaque routeur est identifié par son adresse IP et son nom (premier champ de la ligne). Les routeurs ont souvent (mais pas toujours) des noms contenant un indice sur leur localisation : une abréviation de la ville dans laquelle ils se situent (par exemple *atla* pour Atlanta) ou un code de trois lettres (par exemple `jfk`), correspondant aux codes IATA des aéroports proches.

2. Essayez de représenter une partie du chemin suivi par les paquets sur la carte suivante en utilisant un site de cartographie et en vous aidant, si nécessaire, d'une base de données en ligne des codes aéroport.



6.1.2 Ressources publiques mises à disposition

Internet est séparé en systèmes autonomes (AS) et découvrir la topologie interne d'un AS est difficile si elle n'est pas publiée par l'exploitant. Les routeurs, quant-à-eux, utilisent le protocole de routage BGP pour identifier les AS proches et les destinations qu'ils permettent d'atteindre. C'est une partie de cette information qui est rendue publique par certains sites Web.

Par exemple, la page `http://www.robtex.com/as/as1712.html` vous donne des informations relatives à l'AS numéro 1712 correspondant à l'école (ENST). En navigant dans la page, vous accéderez à diverses informations :

- la section *Graph* vous montre les connexions entre l'AS 1712 et ses voisins jusqu'à plusieurs sauts.
- la section *Peer* vous donnera des informations sur les liens entre l'AS 1712 et ses voisins directs. Vous pouvez noter qui sont les fournisseurs d'accès de l'école actuellement.
- Enfin, la section *BGP*, vous affiche les informations envoyées par cet AS à ses voisins par le protocole de routage BGP. Attention, il ne s'agit ici que des informations concernant l'AS lui-même. En explorant les différents AS proches de l'AS 1712, vous pourrez constater qu'un AS peut être composé de plusieurs réseaux IP.

Un second type d'outils mis à disposition par les opérateurs et les administrateurs des AS eux-mêmes se nomment les *Looking Glasses*. Il s'agit de simples pages Web (ou d'outils accessibles en ligne de commande via `telnet`), permettant d'explorer les informations relatives à BGP et d'exécuter un certain nombre de commandes telles que `traceroute` depuis un point distant. Le but initial de ces outils est de faciliter la résolution de problèmes de routage liés à BGP, par exemple lors de l'achat d'un nouveau routeur de bordure. Vous pouvez, de cette manière, examiner ce que votre routeur annonce au reste du monde et résoudre quelques problèmes.

Le site Web `http://www.bgp4.as/looking-glasses` liste plusieurs de ces *looking glasses*. En visitant quelques uns de ces sites, vous remarquerez que plusieurs d'entre eux vous permettent d'effectuer un `traceroute` depuis leur réseau, facilitant de cette manière la découverte de la topologie d'Internet. Certains *looking glasses* vous fournissent le résultat d'une commande appelée `lft` qui est une version de traceroute améliorée pour afficher les numéros d'AS

Ce type d'outils peut être utilisé pour découvrir une partie de la topologie d'Internet à un niveau de précision supérieur par rapport aux AS. Il est possible d'identifier une partie des routeurs. Pour établir une telle cartographie, il est cependant nécessaire de lancer des mesures depuis plusieurs points de la planète afin d'avoir une vision la plus complète possible.

À vous :

3. Quels types de routeurs êtes-vous, à votre avis, le plus susceptibles d'identifier lorsque vous effectuez un `traceroute` ?

4. En vous basant sur vos connaissances sur l'acheminement des paquets dans un réseau IP, et sur la théorie des graphes, expliquez pourquoi il est nécessaire de disposer de plusieurs points d'observation pour avoir une vision globale de la topologie du réseau. Vous pourrez vous aider d'un exemple de taille limitée.

6.1.3 Chemin entre les AS 2607 et AS 1712

Le système autonome 2607 correspond au réseau *Slovak Academy Network* (SANET), qui fournit un *looking glass* à l'adresse : (`http://www.six.sk/lg.php`).

Exécutez, au moyen de cet outil, un `traceroute` (via l'option *trace*) à destination de la machine `137.194.2.34` appartenant au réseau de l'école. La version de `traceroute` fournie ici n'affiche pas tous les numéros de systèmes autonomes, il vous faudra les déterminer vous-mêmes.

À vous :

5. Notez l'adresse IP du premier routeur appartenant au réseau SANET.
6. Vous pouvez ensuite déterminer le numéro de système autonome auquel appartient l'adresse `A.B.C.D` (lorsque celle-ci n'est pas indiquée) en tapant, dans une fenêtre de terminal (sous Unix), la commande suivante : `whois -h whois.cymru.com A.B.C.D`. Si la commande n'est pas disponible (machines Windows), vous pouvez utiliser le site `http://whois.cymru.com/` qui vous fournira les mêmes informations. Cette commande interroge une base de données administrative, la base `whois`, située sur le serveur `whois.cymru.com`.
7. Dessinez le graphe des AS traversés.
8. Vous noterez, en interrogeant la base `whois` via le Web et en cochant l'option "prefix", que vous pouvez obtenir les adresses et masques de réseaux traversés. Identifiez les réseaux IP traversés par leur adresse réseau en notation CIDR.

Depuis votre machine locale, effectuez un `traceroute` à destination du premier routeur de SANET que vous avez identifié plus haut.

À vous :

9. Comparez les résultats. Les chemins aller et retour sont-ils identiques au regard des routeurs traversés ? Et au regard des AS traversés ? Pensez-vous que ce cas soit représentatif ?

Vous pourrez noter que `traceroute` indique des mesures de délai pour chaque liaison. Ce délai représente le temps nécessaire à un message pour faire l'aller-retour entre le point d'origine et le routeur identifié.

À vous :

10. En considérant que les liens sur la route sont des fibres optiques dans lesquelles le signal se déplace à une vitesse de 300000 km/s, calculez les distances théoriques entre les différents routeurs.
11. Cette information vous semble-t-elle cohérente ? Quels autres paramètres peuvent influencer ce délai ?

6.1.4 Interconnexion entre cinq systèmes autonomes

À vous :

12. Vous allez maintenant utiliser votre machine locale, le *looking glass* de SANET ainsi que d'autres *looking glasses* pour tracer le graphe d'interconnexion au niveau AS des systèmes autonomes suivants : AS 1712 (ENST), AS 2607 (SANET) et AS 513 (CERN, Suisse).

    En utilisant les *looking glasses* que vous jugerez utiles et les informations que vous pourrez trouver sur la page `http://www.cidr-report.org/cgi-bin/as-report?as=AS1712` dans l'adresse de laquelle vous pourrez remplacer AS1712 par n'importe quel numéro d'AS, ajoutez au graphe précédent les connexions directes entre les AS que vous n'avez pas vues en n'utilisant que traceroute.

TP 7

Routage Dynamique

Introduction

Ce TP porte sur les protocoles de routage. Vous allez manipuler un réseau correspondant au système autonome de Télécom ParisTech (AS 1712) dans lequel 9 routeurs sont déployés et deux stations (ENST et INFRES) sont branchées. Ce système autonome est connecté à deux autres systèmes autonomes, Cogent et Renater. Le protocole de routage OSPF est déployé au sein du système autonome et BGP fonctionne sur les routeurs de sortie (Avoranfix et Goudurix).

Commencez par lancer l'environnement virtualisé et positionnez-vous dans le répertoire `TP_7a`. Lancez les machines virtuelles, vous obtenez plusieurs fenêtres, pour les 9 routeurs de l'école, pour 3 routeurs externes et pour deux stations dans l'école.

7.1 Sans routage

Dans un premier temps, le réseau est déployé sans qu'OSPF ne soit lancé. Les routes ne sont donc pas actives.

À vous :

1. En utilsant `ifconfig` et `ping`, dessinez la topologie du réseau. Identifiez les différents sous-réseaux et leurs masques.
2. Si vous effectuez un `ping` d'Avoranfix à Lug (dont vous déterminerez une adresse IP), que se passe-t-il ? Examinez les routes des différents routeurs et expliquez.

7.2 Réseau avec OSPF

Arrêtez les machines virtuelles, nettoyez les fichiers temporaires (`lclean`), placez-vous dans le répertoire `TP_7b` et relancez les machines virtuelles. Il s'agit maintenant du même réseau dans lequel le protocole OPSF est actif. Certaines fenêtres de terminal ne s'ouvriront pas, pour les machines dont vous n'aurez pas besoin, mais les machines sont présentes sur le réseau.

À vous :

3. Testez à nouveau la requête `ping` précédente. Que se passe-t-il ? Examinez les routes des différents routeurs et expliquez le rôle d'OSPF.
4. L'outil `traceroute` vous permet de connaître la liste des routeurs traversés pour atteindre une destination (sa syntaxe est `traceroute dest` où *dest* peut être un nom DNS ou une adresse IP). Son fonctionnement sera étudié dans un prochain TP. Examinez la route d'Avoranfix à Lug et identifiez le ou les routeurs intermédiaires. Matérialisez cette route sur votre schéma.

5. Effectuez une capture du trafic sur l'un des routeurs avec `wireshark` et examinez les messages *Hello* d'OSPF. Quelle est l'adresse de destination de ces messages ? Cette adresse correspond-t-elle à un routeur que vous connaissez ?

   Il s'agit en réalité d'une adresse de diffusion un peu particulière, qu'on appelle une adresse *multicast*. Une adresse *multicast* définit un groupe de noeuds particuliers du réseau (ici tous les routeurs participant à OSPF pour l'adresse 224.0.0.5) et est traité par les routeurs de manière à minimiser les transmissions de messages inutiles (les routeurs essaient de ne pas envoyer 2 fois le même message sur un même lien, contrairement au cas où chaque routeur serait joint individuellement).

6. Examinez maintenant la partie *OSPF Hello* du message, vous remarquerez qu'il y a un certain nombre de champs dont la plupart ne vous intéresseront pas à ce stade. Remarquez les champs `Hello_interval` et `Dead_Interval`. Que signifient, à votre avis, ces deux valeurs ? Que pouvez-vous en conclure ? (vous pouvez vous aider du Web).

7. Trouvez un message émis par le routeur `Avoranfix`. Examinez, dans la partie *Hello* les différents voisins déclarés. Comparez avec la topologie que vous avez déterminée au premier exercice ; que pouvez-vous en conclure sur le fonctionnement d'OSPF ?

8. Exécutez maintenant une capture sur Avoranfix en tâche de fond (suffixe & après la commande) puis effectuez un `ping` d'Avoranfix à Lug. Toujours sur Avoranfix, tapez la commande `mtr A.B.C.D` où A.B.C.D représente l'adresse IP de Lug. L'outil que vous utilisez, `mtr` effectue un `traceroute` en permanence et vous affiche les résultats successifs. Eteignez maintenant le routeur par lequel passe ce trafic en tapant, dans la fenêtre de la machine hôte, `lhalt nomRouteur` où `nomRouteur` est le nom de la machine à éteindre. Attention, la casse des caractères est importante (majuscules / minuscules). Examinez et expliquez l'affichage de `mtr`.

9. Examinez la trace collectée à l'étape précédente avec `wireshark` et recherchez les paquets ICMP autres que `echo` et notez leur date. Examinez ensuite le trafic OSPF autour de ces paquets. Sans rentrer dans les détails, que pouvez-vous dire sur le fonctionnement d'OSPF ? Estimez, à partir de ce que vous voyez le temps de convergence du protocole. Y a-t-il un lien avec les *hello_interval* et *dead_interval* ? Emettez une hypothèse pour expliquer ce phénomène.

10. Relancez la capture, puis `mtr` comme précédemment avant de redémarrer le routeur que vous avez éteint (`lstart routerName`). De la même manière qu'à la question précédente, examinez le scénario et les messages échangés. Pourquoi, à votre avis, la route change-t-elle alors qu'une route valide existe déjà ?

11. Imaginez comment le scénario aurait pu se dérouler dans le cas d'un routage statique. Quel est l'intérêt d'un protocole de routage ?

7.3 Pour référence : examiner les tables BGP

Le protocole BGP est utilisé pour le routage entre AS. Sur la machine `transit`, il vous est possible d'examiner les tables BGP. Pour réaliser cet examen, vous devrez exécuter la commande `vtysh` qui vous amènera dans une ligne de commande particulière, celle mimant le système d'exploitation d'un vrai routeur. Plusieurs commandes sont disponibles, que vous pouvez trouver sur la page `man` de vtysh ou sur le web.

La commande `show ip bgp` vous donne une vision de la table BGP (table de routage au niveau AS) d'un routeur. Examinez la table de `transit` et plus particulièrement le chemin menant à l'école.

Vous pouvez, par exemple, examiner la table de routage de `Toutatis`. L'école dispose de la plage d'adresse `137.194.0.0 / 16`. Remarquez qu'il y a d'autres préfixes annoncés. Vous pouvez aussi comparer les tables de routage de `Lug` et de `Cogent` et remarquer le nombre d'entrées correspondant au réseau de l'école en notant que le niveau de détail est différent car l'un des deux routeurs est plus proche de l'intérieur du réseau de l'école que l'autre.

TP 8

Spanning Tree Protocol

Introduction

Ce TP est prévu pour durer 1h30 et concerne l'étude du fonctionnement du protocole *Spanning Tree* dans un réseau local. En annexe de ce TP, vous trouverez des rappels sur le protocole en lui-même et des informations complémentaires qui n'ont pas été vues en cours qui pourront vous aider lors du TP.

8.1 Effet des *broadcasts* dans un réseau redondant

Placez-vous dans le répertoire TP_8a de la machine hôte et tapez `lstart`. Le réseau représenté sur la figure 8.1 suivante démarre :

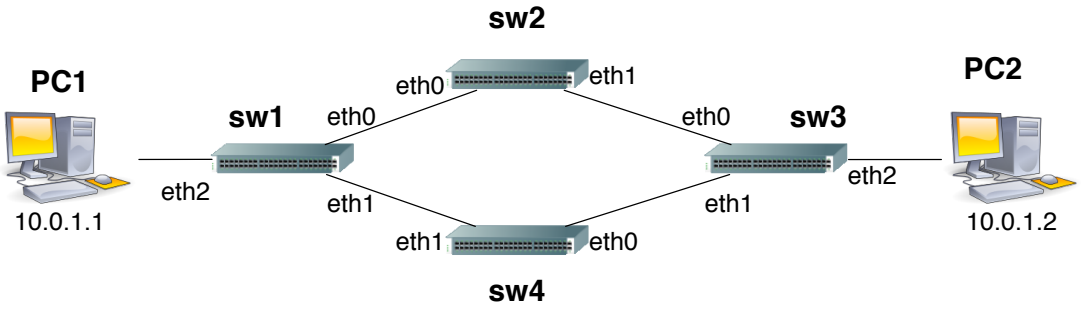


Figure 8.1: Réseau du répertoire `TP_8a`

La topologie ci-dessus est redondante, possédant un cycle. Nous allons tout d'abord examiner l'effet d'un message envoyé en *broadcast* sur une telle topologie.

Ping, quelques otions :

Nous avons déjà utilisé `ping` à de nombreuses occasions pour tester l'existence d'une machine distante. Le programme se comporte en envoyant une série de messages à l'infini et en affichant les réponses avec le délai d'aller-retour. Ping dispose de plusieurs options qui nous seront utiles ici :

- `-c 1` : permet de n'envoyer qu'un seul message au lieu d'une série
- `-b` : nécessaire pour interroger un groupe de machine (lorsque A.B.C.D est une adresse de *broadcast*)

À vous :

1. lancez une capture sur l'un des ponts (`sw1`, `sw2`, etc.).
2. Vous vous trouvez sur le réseau 10.0.1.0/24 ; quelle est l'adresse de *broadcast* de ce réseau ? Envoyez **un unique** message `ping` à destination de cette adresse de *broadcast* alors que la capture est en cours.
3. Arrêtez la capture. Sauvegardez le fichier pcap **avant** d'arrêter les machines virtuelles (`lcrash`, puis `lclean`), sans quoi votre machine pourrait rapidement devenir très lente. Lors de l'arrêt des machines virtuelles, wireshark sera fermé (son exécution est liée à l'existence de la machine surveillée). Relancez donc l'outil et ouvrez la capture précédente pour l'analyser.
4. Examinez le trafic capturé et regardez en particulier les paquets de type ICMP correspondant au trafic généré par `ping`. Vous pouvez créer un filtre en tapant simplement `icmp` dans le champ `filter` avant de presser `Apply`. Vous pouvez aussi utiliser la commande `IO Graphs` dans le menu `Statistics` pour représenter le débit généré par ces paquets. N'oubliez pas de filtrer à nouveau les paquets pour obtenir le graphe représentant le volume de trafic qui vous intéresse et n'hésitez pas à jouer avec les échelles (*tick interval* et *Unit*).
5. Interprétez et expliquez ce qui se passe.

8.2 Fonctionnement du Spanning Tree Protocol

Relançons l'expérience en activant le protocole *Spanning Tree Protocol* entre les différents ponts. Assurez-vous d'avoir correctement nettoyé les fichiers temporaires de la précédente exécution du TP (`lclean` et suppression du ou des fichiers de capture) et relancez le réseau virtuel. Nous allons maintenant configurer les commutateurs afin d'activer le protocole spanning tree.

Configuration des commutateurs :

Les commutateurs utilisés dans ces TP ne sont pas des équipements réels. Il s'agit de machines (virtuelles) sous Linux configurées en mode *bridge*, c'est-à-dire pour se comporter comme des ponts entre leurs interfaces réseau. Sur une machine, il est possible lier plusieurs interfaces entre elles en créant un tel pont qui se voit doté d'un identifiant d'interface : `br0`, `br1`, etc. La machine qui accueille un tel pont assure alors la tâche d'un commutateur entre les interfaces liées. Il est possible d'agir sur la configuration de chaque pont au moyen de la commande `brctl`, dont vous êtes invités à examiner la page de manuel (`man brctl`) pour comprendre les fonctions exactes que peut réaliser un tel commutateur logiciel.

Les commandes suivantes permettent d'interagir spécifiquement avec le spanning tree sur le pont logiciel `br0` :

- `brctl stp br0 on` : permet d'activer le protocole *Spanning Tree*
- `brctl stp br0 off` : permet de désactiver le protocole *Spanning Tree*
- `brctl showstp br0` : affiche l'état du protocole *Spanning Tree*. Cette commande affiche un ensemble d'informations pour chaque interface (`ethXX`). ; (*Designated root*) identifie la racine de l'arbre selon le commutateur. L'information *Root Port* pour chaque pont vous donne l'interface par laquelle on "remonte" dans l'arbre, c'est-à-dire l'interface qui a été sélectionnée pour aller en direction de la racine. Attention, l'identifiant de l'interface est décalé par rapport à son numéro (`eth0` correspond à l'interface 1).

  L'état de chaque interface (*state*) peut être soit *forwarding* (c'est-à-dire active), soit *blocking* (c'est-à-dire inactive). Les interfaces peuvent être actives d'un coté et désactivées de l'autre, ce qui veut dire que le trafic est bloqué de toutes manières.
- `brctl setbridgeprio br0 XXX` : permet de modifier la priorité d'un pont dans l'algorithme,

utilisez la commande suivante, dans laquelle XXX doit être remplacé par un nombre compris entre 0 et 65535. La plus basse valeur correspond à la plus haute priorité.

À vous :

6. Activez le protocole *Spanning Tree* sur les quatre commutateurs et réitérez l'expérience précédente. Que constatez-vous ?
7. Chaque commutateur possède trois interfaces (`eth0`, `eth1` et `eth2`). Utilisez wireshark afin de déterminer quelle(s) interface(s) ont été désactivée(s) par le protocole.
8. Vérifiez votre résultat en affichant sur chaque commutateur l'état du *Spanning Tree*. Reconstruisez l'arbre en identifiant la racine et les interfaces vers la racine pour chacun des ponts.
9. Modifiez la priorité des différents ponts en désactivant le protocole, changeant la priorité puis en réactivant le protocole. Vous pouvez examiner la topologie ainsi créée et vérifier qu'elle correspond bien à ce que vous attendiez.
10. Manipulez les coûts des interfaces pour définir un arbre dans lequel le lien entre *sw1* et *sw2* est bloqué

8.2.1 Efficacité du protocole

Arrêtez le réseau virtuel nettoyez les fichiers temporaires, positionnez-vous dans le répertoire `TP_8b` et relancez le nouveau réseau au moyen de la commande `lstart`. Le réseau sur lequel vous travaillez maintenant est représenté sur la figure 8.2 (la ligne pointillée représente un lien qui n'est pas activé et ne sera pas pris en compte par le protocole ; les numéros représentent les identifiants d'interface) :

À vous :

11. En vous basant sur votre compréhension de l'algorithme, quel devrait être l'arbre couvrant créé sur ce réseau (tous les coûts des liens sont identiques) ?
12. Vérifiez, en examinant les *root ports* sur les différents ponts, que c'est bien le cas.
13. En insérant des sondes `wireshark` à différents endroits du réseau, quel est le chemin suivi par un `ping` de PC1 (10.0.1.1) à PC2 (10.0.0.2) ?
14. Ce chemin est-il efficace ? Quels sont, à votre avis, les problèmes qui peuvent survenir lorsque le *Spanning Tree* est utilisé ?
15. Quelle serait la meilleure place pour localiser la racine de l'arbre ? Influencez le protocole au moyen des priorités sur les ponts pour que cette racine soit effectivement à sa place.
16. Pouvez-vous régler le problème, quitte à aboutir à une solution efficace mais non-optimale, sans changer la racine et en ne modifiant que les poids des interfaces ?

8.3 Référence : Le Spanning Tree Protocol

L'effet des diffusions sur le réseau est très important en présence de cycles. Cependant, pouvoir construire un réseau comportant des cycles est important pour des questions de fiabilité, car ces derniers introduisent de la redondance. En cas de panne d'un pont ou d'un lien, une topologie redondante peut continuer à fonctionner. C'est pourquoi les ponts peuvent communiquer entre eux pour extraire et utiliser une topologie sans cycle de la topologie redondante. Seuls les liens appartenant à cette topologie extraite sont utilisés, les autres étant désactivés (en réalité ils restent activés mais ne transmettent ni ne reçoivent aucune trame de données) et ce jusqu'à ce qu'une panne survienne et qu'il soit nécessaire de réactiver un ou plusieurs liens de secours.

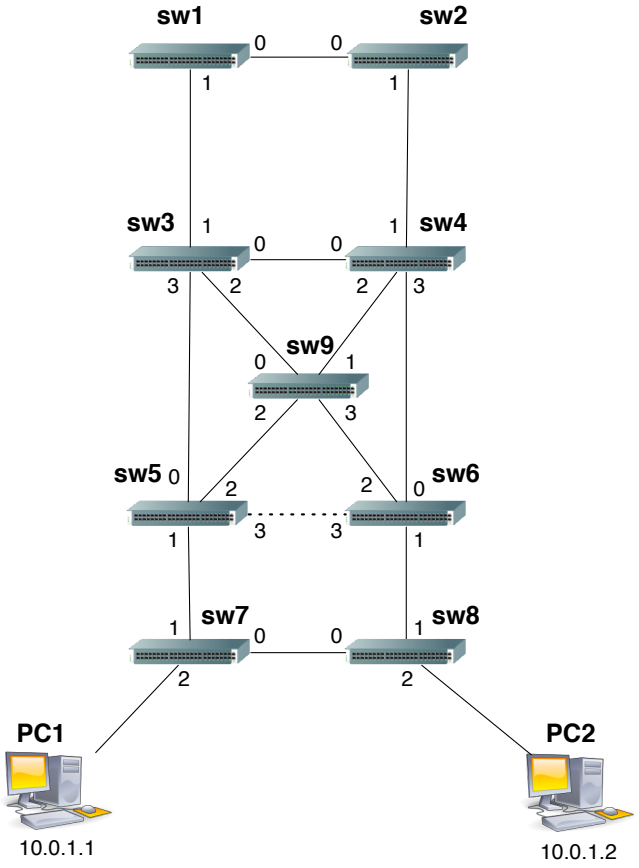


Figure 8.2: Réseau du répertoire `TP_8b`

Le *Spanning Tree Protocol* est le protocole mis en œuvre entre les ponts et permettant d'identifier quels liens doivent rester actifs et quels liens doivent être désactivés. Son but est d'extraire, à partir du graphe représentant la topologie, un sous-graphe sans cycle, c'est-à-dire un arbre. Cet arbre doit connecter tous les éléments du réseau, il est par conséquent qualifié de couvrant (*spanning*). Le protocole fonctionne de manière totalement distribuée, les ponts communicant entre eux et ne se reposant sur aucun serveur central.

8.3.1 Construction de l'arbre couvrant

Le protocole fonctionne en deux phases. Lors de la première phase, des messages `Configuration` sont échangés afin de déterminer quel pont sera considéré comme racine de l'arbre couvrant, c'est-à-dire comme point de référence à partir duquel l'arbre sera construit. Par la suite, tout changement de topologie est détecté et provoque un certain recalcul de l'arbre.

Identification de la racine

Lors de la phase d'identification de la racine de l'arbre, les ponts échangent des messages de type `Configuration` dans lesquels ils indiquent à leurs voisins directs quel pont ils considèrent être la racine. Chaque pont émet régulièrement ce type de message et par conséquent reçoit des informations similaires de chacun de ses voisins. À la réception d'un tel message, un pont compare l'identificateur du pont qu'il considère comme étant la racine de l'arbre à celui qui lui est annoncé dans le message, à la recherche du pont de **plus petit identificateur**.

L'identificateur d'un pont, que l'on peut voir dans les champs `Root Identifier` des paquets `Configuration` est composé par deux données :

Une priorité sur 16 bits valant par défaut 32768 (80:00 en hexadécimal) et pouvant aller de 0 jusqu'à 65535.

L'adresse MAC de la première interface du pont. Les adresses MAC sont, dans ce TP, définies de façon à ce que l'avant-dernier chiffre soit égal au numéro du pont et que le dernier chiffre soit égal au numéro de l'interface. (ex : 00:00:00:00:03:02 identifie l'interface n.2 sur le pont sw3).

L'identificateur est donc basé sur l'adresse MAC, mais le champ priorité permet à un administrateur système de configurer le protocole pour influencer la création de l'arbre. Vous pouvez afficher l'identificateur de chaque pont (en hexadécimal) en tapant `brctl show` dans un terminal.

Si l'annonce reçue contient un pont d'identificateur plus petit que la racine actuellement considérée, le pont remplace cette racine par la nouvelle annoncée, arrête d'émettre des messages `Configuration` et retransmet uniquement les messages `Configuration` annonçant l'existence de cette racine (ou une meilleure). À l'issue de cette phase, seuls les messages émanant du pont de plus petit identificateur sont transmis dans le réseau.

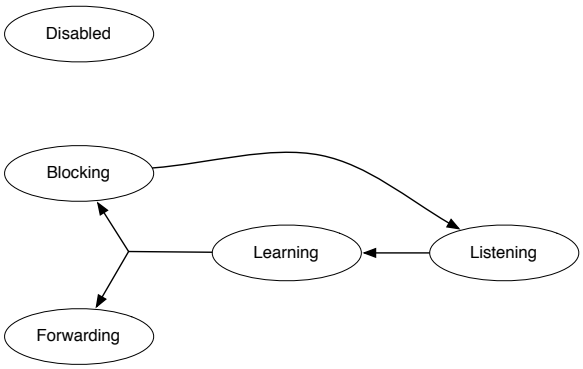
Sélection de l'état des interfaces

Une fois que la racine est identifiée, il reste à construire l'arbre, c'est-à-dire en sélectionnant quelles interfaces seront actives et inactives. Toujours en se basant sur les messages `Configuration`, chaque pont va sélectionner une interface comme *root port*, c'est-à-dire comme interface vers son père dans l'arbre. Cette interface est celle qui lui offrira le meilleur chemin vers la racine au sens :

- du **coût** du chemin : chaque pont définit, pour chacun de ses ports un coût, valeur numérique qu'il est possible d'afficher au moyen de la commande `brctl showstp br0`. Pour chaque interface, un champ *path cost* affiche le coût associé. Il est possible de régler cette valeur au moyen de la commande `brctl setpathcost br0 ethX YY` où ethX est l'identifiant de l'interface et YY le coût. Le protocole sélectionnera comme *root port* l'interface lui offrant le chemin de coût cumulé minimum vers la racine.
- en cas d'égalité de coût, l'**identifiant** du pont parent sera le critère discriminant, on sélectionnera le pont d'identifiant minimum, en prenant en compte, comme pour la sélection de la racine, la priorité.
- en cas d'égalité, les **identifiants** des interfaces pourront être utilisés comme dernier discriminant.

8.3.2 Réactions aux changements dans la topologie

Le protocole *Spanning Tree* fonctionne en permanence, surveillant les liens du réseau et envoyant des messages `Topology Change` lorsqu'il semble nécessaire de mettre à jour l'arbre. Les ponts et les interfaces des ponts ré-exécutent alors la procédure habituelle. En outre, lorsqu'un lien ne répond plus, les interfaces voient leur état (actif, bloquant, etc.) évoluer selon la machine à état représentée ci-dessous :



En fonction de l'avancement et du résultat de l'exécution du protocole, chaque interface d'un pont peut être dans cinq états distincts. À chacun de ces états correspond une politique de gestion des trames de données ainsi que des trames de gestion du protocole *Spanning Tree* (ces trames spécifiques au protocole sont appelées BPDU dans la suite) :

forwarding : le port reçoit, émet et renvoie tous les types de trames. Il est pleinement actif.

blocking : Le port ne renvoie ou n'émet aucune trame de données, il a été désactivé par le protocole *Spanning Tree*. Il n'émet non plus aucune BPDU, mais continue à recevoir les BPDU afin de s'assurer que les liens sont toujours actifs, sans toutefois les prendre en compte (il n'examine pas le contenu des BPDU). Lorsqu'aucune BPDU n'est reçue durant un certain temps (généralement 20 secondes) sur une interface, le pont considère qu'une erreur s'est produite et l'interface passe en mode *listening*.

listening : L'interface ne fait que recevoir les BPDU, mais ces trames sont maintenant examinées et un nouvel arbre est en cours de calcul. Cette phase est transitoire et dure généralement 15 secondes avant que l'interface ne passe en mode *learning*. Aucune trame de donnée n'est prise en compte dans cette état, quelle que soit la direction.

learning : le port accepte maintenant de recevoir des trames de données, mais refuse toujours de les retransmettre. Les BPDU sont émises, reçues et prises en compte, le pont participe pleinement au protocole mais est encore restrictif sur les trames de données avant de passer en mode *forwarding* au bout de 15 secondes à moins d'être désactivé par le protocole, aboutissant dans l'état *blocking*.

disabled : L'interface est explicitement désactivé par l'utilisateur.

Vous pouvez dès lors modifier la topologie en activant ou en désactivant des interfaces et en examinant le fonctionnement du protocole et son résultat au moyen de `wireshark` et `brctl`.

- Pour désactiver un lien, il vous suffit de désactiver l'interface X correspondante : `ifconfig ethX down`
- Vous pouvez activer un lien en l'insérant dans l'ensemble des liens pris en charge par le pont. Par exemple, entre les ponts n. 5 et n. 6 (voir figure précédente), la ligne pointillée est un lien inactif. Il vous suffit d'activer l'interface en tapant, sur sw5 et sur sw6 `ifconfig eth3 up` puis de l'inclure, sur les deux machines, aux interfaces gérées par le pont logiciel en tapant : `brctl addif br0 eth3`.

Notez que vous pouvez, pour ralentir le fonctionnement du protocole, régler les temporisateurs décrits ci-dessous, vous permettant de voir l'évolution des états d'une interface :

L'intervalle entre *Hello* (Hello Time) : par défaut à 2 secondes, il indique l'intervalle séparant deux envois successifs de messages *Hello* par un pont, signalant sa présence. Ce délai se règle par `brctl sethello br0 XX` où XX est le temps en secondes.

Le délai d'expiration associé à une interface (Max Age) : par défaut à 20 secondes, il définit au bout de combien de temps une interface en mode *blocking* doit passer en mode *listening*. Il définit par conséquent le temps de réaction sur erreur du protocole. Ce délai se règle par `brctl setmaxage br0 XX` où XX est le temps en secondes.

Le temps d'apprentissage (Forward Delay) : par défaut à 15 secondes, il définit le temps passé dans les phases intermédiaires (*Listening* et *Learning*) avant de passer à l'étape suivante. Il définit le compromis entre prudence et réactivité sur événement du protocole. Ce délai se règle par `brctl setfd br0 XX` où XX est le temps en secondes.

TP 9

Programmation réseaux

Introduction

L'objectif de ce TP est de manipuler l'interface de programmation classique dans le monde TCP/IP. Cette API se nomme *Socket* et est similaire pour la plupart des langages de programmation. À l'image des modes de communication inter-processus (tubes, files, etc.), elle permet de faire communiquer plusieurs processus répartis un peu partout sur le réseau IP (i.e. l'Internet).

Située au dessus des protocoles de transport, cette API propose deux modes de communication :

- Transmission simple de messages, sans garantie, dans une seule direction à la fois en utilisant le protocole de transport UDP. UDP assurant simplement le transfert d'un message à la fois, on parle de mode *non connecté*.
- Échange bidirectionnel et fiable de messages en utilisant le protocole TCP. TCP encadrant l'envoi de messages par une phase d'ouverture et de fermeture de connexion, on parle de mode *connecté*.

9.1 Sujet du TP

Le sujet est volontairement peu détaillé, afin que vous soyez confrontés aux différentes étapes nécessaires à la réalisation d'un dialogue au travers du réseau. Pour comprendre les échanges à implémenter, vous pourrez vous référer au TP sur les protocoles applicatifs, mais aussi prendre exemple sur un navigateur réalisant la même opération ou enfin vous baser sur la documentation disponible sur le Web (RFC, etc.).

Le TP nécessitant une connexion à Internet et ne nécessitant aucun accès administrateur, vous le réaliserez a priori hors de l'environnement virtualisé introduit au premier TP.

Dans ce TP, vous aurez à utiliser l'API socket détaillée dans les sections ci-dessous pour écrire deux programmes (*Client* et *Server*) réalisant, en séquence, les opérations suivantes :

À vous :

1. *Client* interrogera le serveur `www.google.com` et récupérera la page Web d'accueil en HTML.
2. *Client* utilisera le serveur SMTP de l'école (smtp.enst.fr) pour envoyer le code de cette page par e-mail à votre propre adresse.
3. *Client* la transmettre à un serveur, *Server*, écoutant sur le port UDP 12345 qui se contentera d'afficher le code de cette page.

9.2 Langage Java

En Java, pour utiliser l'API socket, vous importerez `java.net.*` et `java.io.*`.

9.2.1 Sockets UDP

Dans le mode de communication non-fiable (UDP), un récepteur commence par écouter le trafic réseau sur un port applicatif. Pour réaliser cette opération, un programme commence par créer un objet *socket* décrivant les quelques paramètres de la communication (type d'adresse, numéro de port), puis *enregistre* cette *socket* au niveau du système d'exploitation au moyen d'une fonction généralement nommée `bind`. Le système redirigera alors le trafic provenant du port spécifié à l'application.

```
// Création de de l'objet socket
DatagramSocket s=new DatagramSocket(null);

// Objet représentant l'adresse et le numéro de port sur laquelle on é
    coute
//  Param 1 : Adresse IP sur laquelle écouter (null pour toutes)
//  Param 2 : port sur lequel écouter
InetSocketAddress myAddress = new InetSocketAddress((InetAddress)null,
    80);

// Enregistrement au niveau de l'OS
s.bind(myAddress);

// Réception effective (bloquante) du message
byte[] message = new byte[0x100];
DatagramPacket packet=new DatagramPacket(message, message.length);
s.receive(packet);

// Récupération de l'adresse de l'émetteur
InetSocketAddress senderAddress=(InetSocketAddress)packet.
    getSocketAddress();


// Fermeture de la Socket
s.close();
```

Listing 9.1: Socket UDP en Java : coté serveur

De l'autre coté du réseau, un émetteur réalise les opérations correspondantes, créant un objet *socket* similaire, en spécifiant le numéro de port et l'adresse IP de la destination, puis envoie effectivement ses données. Notez l'utilisation de la fonction `gethostbyname` ou de la méthode `InetAddress.getByName` qui permet d'interroger la base DNS, renvoyant une structure complexe représentant la destination, dont on peut extraire l'adresse.

```
// Interrogation du DNS pour obtenir l'adresse IP de la destination
InetAddress destination = InetAddress.getByName("www.enst.fr");

// Objet représentant l'adresse
//  Param. 1 : adresse IP
//  Param 3 : numéro de port
InetSocketAddress destIPAddr=new InetSocketAddress(destination, 80);

// Création de de l'objet socket
DatagramSocket s=new DatagramSocket(null);

// Préparation du message à envoyer
String message = "Hello␣World\n";
byte[] payload = message.getBytes();
DatagramPacket packet = new DatagramPacket(payload, payload.length,
    destIPAddr);

// Envoi effectif du message
s.send(packet);
```

```
// Fermeture de la Socket
s.close();
```

Listing 9.2: Socket UDP en Java : coté client

9.2.2 Sockets TCP

La communication utilisant TCP se déroule de façon similaire mais comprend de plus une phase d'ouverture de connexion et de fermeture de connexion. Tous les messages faisant partie de ladite connexion sont envoyés et reçus avec une certaine fiabilité, TCP s'arrangeant pour qu'un message perdu sur la route soit retransmis.

Le récepteur, de manière similaire à UDP, doit se mettre en écoute. Cependant vous noterez qu'il est possible de gérer plusieurs connexions simultanées avec TCP, car ce dernier permet, via la fonction `accept`, de créer une nouvelle *socket* pour la transmission des données, laissant la précédente disponible pour de nouvelles tentatives de connexion. TCP gère donc une file d'attente des connexions successives et il est possible de spécifier la taille de cette file lors de l'appel à la fonction `listen`, appelée obligatoirement entre `bind` et `accept` (en C). En Java, cette méthode n'est pas présente est les fonctionnalités associées sont automatiquement invoquées, en fonction du type de *socket*. Vous noterez enfin, dans les exemples de code ci-dessous, que le type de socket est différent du scénario UDP (`SOCK_STREAM` à la place de `SOCK_DGRAM`).

```
ServerSocket s = new ServerSocket(80);

// Création d'une nouvelle socket dédiée à la communication
Socket sData = s.accept();

InetSocketAddress senderAddress=(InetSocketAddress)sData.
    getRemoteSocketAddress();

// Communication au moyen des fonctions write et read
String message = "Hello World\n";
PrintWriter outBuf = new PrintWriter(new BufferedWriter(new
    OutputStreamWriter(sData.getOutputStream())), true);
outBuf.println(message);

BufferedReader inBuf = new BufferedReader(new InputStreamReader(sData.
    getInputStream()));
String inText = inBuf.readLine();

// Affichage, exploitation des données, etc.

// Fermeture DES Sockets
sData.close();
s.close();
```

Listing 9.3: Socket TCP en Java : coté serveur

Le coté de l'émetteur est assez similaire au cas UDP. Vous noterez cependant que le type de socket est différent du scénario UDP (`SOCK_STREAM` à la place de `SOCK_DGRAM`), et qu'une primitive `connect` est maintenant appelée avant de démarrer le dialogue applicatif avec l'autre partie.

```
InetAddress destination = InetAddress.getByName("www.enst.fr");

InetSocketAddress destIPAddr=new InetSocketAddress(destination, 80);

// Création de de l'objet socket
Socket s=new Socket();

// Connexion de la socket
s.connect(destIPAddr);
```

```
// Communication au moyen des fonctions write et read
// Voir exemple au niveau du récepteur

// Affichage, exploitation des données, etc.

// Fermeture de la Socket
s.close();
```

Listing 9.4: Socket TCP en Java : coté client

9.3 Langage C

En C, il vous faudra inclure `<sys/socket.h>` et `<netdb.h>`. Vous pourrez aussi avoir besoin de `<stdio.h>`, `<string.h>` ou `<unistd.h>`.

9.3.1 Sockets UDP

```
// Création de l'objet socket
//  Paramétre 1 : domaine de communication (PF_INET = l'Internet IPv4)
//  Paramétre 2 :  type de socket (SOCK_DGRAM = UDP)
//  Paramétre 3 : protocole a utiliser (si plusieurs choix)
int s = socket(PF_INET, SOCK_DGRAM, 0);

// structure représentant l'adresse et le numero de port sur laquelle on
    écoute
struct sockaddr_in myAddress;

  // Type d'adresse ; AF_INET signifie adresse IP
myAddress.sin_family=AF_INET;

  // Adresse du récepteur - n'importe quelle interface
myAddress.sin_addr.s_addr = htonl(INADDR_ANY);

  // Port sur lequel écouter
myAddress.sin_port=htons(80);

// Enregistrement au niveau de l'OS
//  Param. 1 : objet socket
//  Param. 2 & 3 : adresse et taille de l'adresse
bind(s, (struct sockaddr *)&myAddress, sizeof(myAddress));

// Descripteur de l'adresse de l'émetteur
struct sockaddr_in senderAddress;
socklen_t length = sizeof(senderAddress);

// Réception effective (bloquante) du message
//  Param 1 : socket
//  Param 2 & 3 : tampon de réception et taille de ce tampon
//  Param 4 : drapeaux (inutilisé ici)
//  Params 5 & 6 : adresse source & taille de l'adresse
char message[255];
int nbCars;
nbCars = recvfrom (s, message, 255, 0, (struct sockaddr *)&senderAddress,
    &length);

// Affichage, exploitation des données, etc.
```

Listing 9.5: Socket UDP en C : coté serveur

```
// Création de de l'objet socket
//  Paramétre 1 : domaine de communication (PF_INET = l'Internet IPv4)
//  Paramétre 2 :  type de socket (SOCK_DGRAM = UDP)
//  Paramétre 3 : protocole à utiliser (si plusieurs choix)
int s = socket(PF_INET, SOCK_DGRAM, 0);

// Interrogation du DNS pour obtenir l'adresse IP de la destination
struct hostent *destination = gethostbyname("www.enst.fr");
in_addr_t destIPAddr = *((in_addr_t *)(destination->h_addr));

// structure représentant l'adresse (+ numéro de port) de destination
struct sockaddr_in destAddress;

  // Type d'adresse ; AF_INET signifie adresse IP
destAddress.sin_family=AF_INET;

  // Adresse du récepteur
destAddress.sin_addr.s_addr=destIPAddr;

  // Port visé coté récepteur
destAddress.sin_port=htons(80);

// Préparation du message à envoyer
char message[] = "Hello World\n";

// Envoi effectif du message
//  Param. 1 : identificateur de socket
//  Param 2 & 3 : message & longueur du message
//  Param 4 : drapeaux pour transmissions particuliéres
//  Params 5 & 6 : adresse destination et taille de la structure
sendto(s, message ,strlen(message), 0, (struct sockaddr *)&destAddress,
    sizeof(destAddress));
```

Listing 9.6: Socket UDP en C : coté client

9.3.2 Sockets TCP

```
int s = socket(PF_INET, SOCK_STREAM, 0);

struct sockaddr_in myAddress;
myAddress.sin_family=AF_INET;
myAddress.sin_addr.s_addr = htonl(INADDR_ANY);
myAddress.sin_port=htons(80);

// Enregistrement au niveau de l'OS
bind(s, (struct sockaddr *)&myAddress, sizeof(myAddress));

// Démarrage de l'écoute active (non bloquante) des demandes de connexion
//  Param 1 : socket
//  Param 2 : nombre maximum de connexions en attente (avant refus)
listen(s, 5);

struct sockaddr_in senderAddress;
socklen_t length = sizeof(senderAddress);

// Création d'une nouvelle socket dédiée à la communication
int sData = accept (s, (struct sockaddr *)&senderAddress, &length);

// Communication au moyen des fonctions send et recv
```

```
char msg[] = "Hello World\n";
send(sData, msg ,strlen(msg), 0);

char message[255];
int nbCars;
nbCars = recv (sData, message, 255, 0);


// Affichage, exploitation des données, etc.

// Fermeture DES sockets
close (sData);
close(s);
```

Listing 9.7: Socket TCP en C : coté serveur

```
int s = socket(PF_INET, SOCK_STREAM, 0);

struct hostent *destination = gethostbyname("www.enst.fr");
in_addr_t destIPAddr = *((in_addr_t *)(destination->h_addr));

struct sockaddr_in destAddress;
destAddress.sin_family=AF_INET;
destAddress.sin_addr.s_addr=destIPAddr;
destAddress.sin_port=htons(80);

// Connexion de la socket
connect(s, (struct sockaddr *)&destAddress, sizeof(destAddress));

// Communication au moyen des fonctions send et recv
// Voir exemple au niveau du récepteur

// Affichage, exploitation des données, etc.

close(s);
```

Listing 9.8: Socket TCP en C : coté client