# Couche 3

## Question 1

* 52.176.34.69/23

L’adresse suivante en décimal : 0011 0100 . 1011 0000 . 0010 0010 . 0100 0101

Le masque de sous-réseaux : 1111 1111 . 1111 1111 . 1111 1110 . 0000 0000

Il y a 9 (32-23) bits qui peuvent changer, donc le nombre d’adresses IP possible est de 511 + 1 car ça va de 0 à 511 donc 512 adresses possibles.

La plus petite adresse qu’on peut coder est : 52.176.34.0 ; on met les 9 derniers bits à 0.

La plus grande adresse qu’on peut coder est : 52.176.35.255 ; on met les 9 derniers bits à 1.

* 1.0.0.234/11

L’adresse suivante en décimal : 0000 0001 . 0000 0000 . 0000 0000 . 1110 1010

Le masque de sous-réseaux : 1111 1111 . 1110 0000. 0000 0000 . 0000 0000

Il y a 21 (32-11) bits qui peuvent changer, donc le nombre d’adresses IP possible est de 2 097 151 + 1, car ça va de 0 à 2 097 151 donc 2 097 152 adresses possible.

La plus petite adresse qu’on peut coder est : 1.0.0.0 ; on met les 21 derniers bits à 0.

La plus grande adresse qu’on peut coder est : 1.31.255.255 ; on met les 21 derniers bits à 1.

* 245.255.254.1/0

L’adresse suivante en décimal : 1111 0101 . 1111 1111 . 1111 1110 . 0000 0001

Le masque de sous-réseaux : 0000 0000 . 0000 0000 . 0000 0000 . 0000 0000

Il y a 32 (32-0) bits qui peuvent changer, donc le nombre d’adresses IP possible est de

4 294 967 295 + 1, car ça va de 0 à 4 294 967 295 donc 4 294 967 296 possibles.

La plus petite adresse qu’on peut coder est de 0.0.0.0 ; on met les 32 bits à 0.

La plus grande adresse qu’on peut coder est de 255.255.255.255 ; on met les 32 bits à 1.

## Question 2

La gestion de la fragmentation est différente entre IPv4 et IPv6 ; déjà les routeurs intermédiaires ne font pas de fragmentation pour les paquets IPv6 alors que pour les paquets IPv4 oui, si un paquet IPv6 doit être fragmenter ce ne sont que les hôtes de fin qui qui fragmente et réassemble le tout. Le paquet IPv6 ne contient plus les champs de fragmentation (comme ceux de IPv4 : flag et fragment offset) dans son header principal (le premier) mais dans son header de fragmentation. S’il faut fragmenter le paquet alors il faut inclure un second header (grâce au champ Next Header). Pour cela on utilise la valeur 44 qui permet de mettre le header de fragmentation. La taille minimale pour un paquet IPv6 est de 1280 bytes. Sa taille max est de 65 536 bytes mais peut avoi une taille de presque 4Gb grâce au header « Hop-by-Hop Options header ». Tandis que pour le paquet ipV4 la taille maximum qu’on peut transmettre est de 1500 bytes et la taille minimum est 68 bytes.

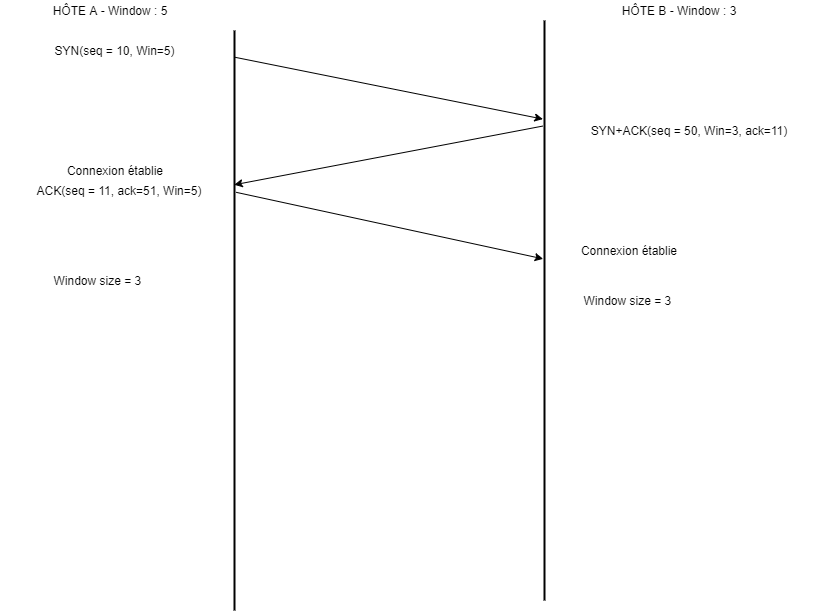
# Couche 4

## Question 1

Le protocole ICMP (Internet Control Message Protocol) sert à envoyer des messages de contrôles à la source (celui qui envoie un paquet) lui informant d’un problème lors de la transmission d’un paquet. Traceroute l’utilise aussi pour connaitre le chemin de la source à la destination. Pour cela il utilise un champ dans les paquets IP, qui est le TTL. Traceroute commence à transmettre un paquet avec un TTL commençant à 1. Le premier routeur va recevoir le paquet et décrémenté le TTL. Le paquet aura donc un TTL qui est égal à 0, le paquet ne peut donc pas être transmis au prochain routeur. Le routeur crée un message ICMP avec comme erreur time exceed. Traceroute va continuer d’envoyer des paquets avec des TTL de plus en plus en incrémentant de 1 à chaque fois jusqu’à ce qu’il atteigne la destination finale. Donc chaque routeur va envoyer à la source des ICMP messages. Traceroute va lire le message ICMP et grâce au champ « Source ip adresse » il aura l’ip du routeur qui lui à envoyer le message. C’est comme ça qu’on aura le chemin entier de la source à la destination.

## Question 2

### 2.1



Pour établir une session TCP ; on utilise le three-way handshake. Le client envoie un paquet au serveur pour établir une connexion avec celui-ci. Il envoie un segment avec le flag SYN mis à 1. Le serveur répond en lui envoyant un segment SYN-ACK, le flag de SYN est toujours à 1 car on établie toujours la connexion et il comporte en plus un numéro de ACK qui correspond au numéro du paquet reçu + 1. La connexion est établie pour le client, il renvoie un dernier segment ACK au serveur avec le numéro du ACK est défini selon le numéro du segment précédent reçu +1. La taille de la « sliding window » est négocié pendant tout l’échange de segment entre le client et le serveur. Chaque hôte respectif envoie la taille de sa fenêtre glissante permettant ainsi de choisir la taille la plus petite entre les deux. Pour cela il y a un champ dans le segment tcp appelé window où la taille peut être insérer. Donc à chaque échange entre le client et le serveur pendant qu’il établisse la connexion et pendant l’échange de données après la connexion ; ils s’échangent leur taille de fenêtre pour trouver la plus petite.

### 2.2

Un paquet avec le flag RST est envoyé.

### 2.3

La sliding-window permet à ce que le client envoie plusieurs segments en même temps au serveur et que le serveur en reçoit plusieurs aussi. Mais ces paquets doivent être comprise dans la fenêtre. Et à chaque fois qu’un segment est bien reçu la fenêtre glisse vers l’avant en attendant les prochains paquets à recevoir si c’est le premier segment de la fenêtre. C’est la même chose pour le client à chaque fois que le client reçois un ACK d’un segment transmis alors il fait glisser sa fenêtre si c’est le premier paquet de la fenêtre sinon il attend que le premier segment soit bien reçu. Il faut toujours que la fenêtre de l’envoyeur soit plus petite ou égale que celle du receveur sinon des segments ne seront pas pris en compte par le receveur et seront donc transmis pour rien.

# Couche application

## Question 1

### 1.1

Quand l’utilisateur va sur le site une requête https est faite avec comme méthode Get pour récupérer la page du site en question. Le serveur répond en envoyant la page et le code 200 disant que la requête c’est bien passé. Comme sur le site il y a une image il y a aussi une requête pour aller récupérer l’image sur un autre serveur, en utilisant encore la méthode Get. Les 2 autres requêtes essayent de récupérer des ressources mais ne les trouve pas ; il y a donc une erreur 404. (La ressource n’a pas été trouvé). Donc à chaque fois qu’on a besoin d’une ressource on fait une nouvelle requête http. La première requête qui retourne un code erreur 404 n’utilise pas https mais bien http, la communication n’est donc pas sécurisé

## 1.2

Le premier protocole de la couche application utilisé est le DNS pour avoir les adresses IP des serveurs. Une requête DNS est faite à chaque fois qu’on a besoin de l’adresse ip d’un serveur ; donc lorsqu’on se connecte sur le site et lorsqu’on demande les ressources se trouvant sur un autre serveur. On peut avoir un cache DNS évitant de faire trop de requête DNS, celui-ci permet d’aller plus vite.

Le Deuxième protocole utilisé est http pour transférer des ressources entre le serveur et le client (navigateur). Donc à chaque requête, le protocole http est utilisé.

## Question 2

Les cookies servent à garder des informations sur l’utilisateur. Il est créé par le serveur et envoyé à l’utilisateur. Le navigateur le stocke sur le pc de l’utilisateur. Il permet de stocker des informations à propos de l’utilisateurs. A chaque fois que l’utilisateur se connecte sur le site il envoie son cookie au serveur. Il permet de facilité la navigation car il sauvegarde les données de connexion pour le site en question. Le problème de donner son cookie à une autre personne c’est qu’il pourra se connecter à notre session sur le site et ainsi avoir accès à nos données confidentielles.

# Questions Générales

## 1

Pour passer du domaine à l’adresse IP, il faut utiliser le DNS. Pour passer du IP à l’adresse MAC il faut utiliser le protocole ARP. La différence entre ces 3 adresses c’est qu’elles ne proviennent pas de la même couche. Il est plus facile pour l’être humain de se souvenir d’un nom de domaine que d’une adresse IP. Lorsqu’un utilisateur veut se connecter au site de [Google](http://Google) il va écrire [www.google.com](http://www.google.com) et pas l’adresse IP de Google. Mais les serveurs sont identifiés par une adresse IP ; pour cela DNS à été mis en place pour traduire le nom de domaine en adresse IP. Ensuite quand deux machines veulent s’échanger des paquets ; ils le font grâce au protocole IP. Mais pour que le paquet circulent entre les câbles il faut faire appels à la couche liaison qui permet d’encapsuler un paquet IP dans une frame. Et comme la couche liaison ne comprend pas les adresses IP. Les hôtes et les routeurs ont des adresses de la couche liaison. Il faut donc trouver l’adresse physique (MAC) de celui à qui transmettre le paquet. Pour cela, il existe un Protocol qui permet s’appel ARP et qui permet de trouver l’adresse MAC de celui qui possède l’adresse IP. Pour cela il broadcast à tout le monde dans son sous réseaux en demandant à qui appartient cette adresse IP. La machine concernée répond en lui envoyant son adresse MAC. Le frame pourra donc circuler jusqu’à la machine ayant l’adresse MAC correspondante.

## 2.

L’encapsulation permet d’inclure des données d’un protocole de la couche précédente dans le protocole actuelle. Par exemple inclure un segment TCP dans un paquet IP. Dans le payload de du paquet ip, il y aura tout le segment TCP (avec son header et son payload). Chaque niveau de couche est encapsulé dans sa couche avant. Tout est au final encapsuler dans un frame qui appartient au protocole le plus bas. Et à chaque fois qu’on a besoin d’un protocole de la couche en haut on décapsule.

## 3.

Ce problème apparait lorsqu’il y a au moins deux liens cassés dans le réseau de routeur. Et que des routeurs ne sont plus du tout accessible.

## 4.

Le minage consiste à calculer le proof of work pour un bloc contenant des transactions Le proof of work est le chiffre qui donne un hash avec 60 zéros au début. Ça requiert une énorme puissance de calculs ; car il faut tester toutes combinaisons possibles. C’est de la brute force. Comme ça utilise beaucoup de calculs les GPU tourne à fond et du coup demande beaucoup d’électricité.

# Question Bonus

theflagis\_NZuiSnarRl6B3APrx7GPCBAU0HjcVjw\_94.224.82.173