Université Grenoble Alpes

Introduction aux Systèmes et Réseaux
Protocoles réseau et outils associés

L'objectif de ce TP est la prise en main de plusieurs outils intéressants pour observer le fonctionnement des réseaux [1].

Remarques :

- Sauf mention contraire explicite, dans cet énoncé, le terme *IP* fait référence à la version 4 du protocole IP (IPv4).
- Pour les différents exercices d'observation proposés, il est recommandé de se référer aux documentations disponibles en ligne sur les protocoles concernés. Voir par exemple les pages Wikipédia (de préférence en version anglaise) qui détaillent généralement le format et le rôle de chaque message/champ (`http://en.wikipedia.org/wiki/Internet_protocol_suite`).
- Pour des pages de manuel Linux en anglais, voir par exemple :
  - `http://linux.die.net/man/`
  - `https://www.kernel.org/doc/man-pages/`

1 Acheminement des paquets

La commande `ping` permet de vérifier qu'une machine est joignable et d'obtenir une mesure du temps d'aller-retour pour la communication avec celle-ci [2]. La commande `traceroute` [3] (vue en cours, ainsi que sa variante `lft` [4]) permet de connaître le chemin parcouru (ou autrement dit, la séquence des routeurs traversés) par les paquets échangés entre deux machines reliées par un ensemble de réseaux interconnectés [5].

Tester (et étudier le résultat de) ces deux commandes avec plusieurs serveurs web universitaires, par exemple :

- l'Université de Genève (`www.unige.ch`) ;
- University of Cambridge au Royaume-Uni (`cam.ac.uk`) ;
- Yale sur la côte Est des USA (`www.yale.edu`) ;
- UCSD (University of California San Diego) sur la côte Ouest des USA (`www.ucsd.edu`) ;
- University of Sidney en Australie (`sydney.edu.au`) ;

1. Certains exercices sont inspirés de l'ouvrage suivant : J. F. Kurose and K. W. Ross, *Computer Networking, A Top-Down Approach*, Addison Wesley.

2. Remarque : une absence de réponse au ping n'implique pas nécessairement que la machine visée est arrêtée. Elle peut-être injoignable en raison d'un problème (matériel ou logiciel) de fonctionnement du réseau ou d'un filtrage délibéré de certains paquets (exemple : pare-feu visant à protéger la machine de certaines attaques).

3. Cette commande s'appelle `tracert` sous Windows.

4. `lft` utilise des techniques plus rapides et plus fiables que `traceroute`. Cependant, dans un environnement virtualisé, ces techniques ne fonctionnent généralement pas correctement.

5. De nombreux serveurs offrent une interface Web permettant de lancer des requêtes traceroute à partir de divers endroits. Voir notamment la liste suivante : `http://www.traceroute.org`

Vérifier que les temps de communication indiqués par les deux commandes sont concordants. Comparer l'impact du nombre de routeurs traversés et de la distance géographique sur la latence des communications. Existe-t-il d'autres paramètres susceptibles d'influer sur cette latence ?

2 Présentation du logiciel Wireshark (analyseur de protocoles réseau)

Cette section décrit brièvement le logiciel Wireshark, qui sera utilisé pour la plupart des exercices de cette séance de TP. Wireshark est un *analyseur de protocoles* : il permet d'observer (et de mémoriser) les communications au niveau d'une ou plusieurs des interfaces réseau d'une machine. Il est composé de deux modules complémentaires : le premier permet de capturer et mémoriser les communications observées ; le second module permet d'interpréter et de visualiser les traces capturées, en utilisant les règles de codage des protocoles concernés. On peut également ajouter les précisions suivantes :

- D'un point de vue fonctionnel, un analyseur de protocoles est en général complètement transparent au sein d'un réseau : il observe passivement les communications entre machines (pas d'ajout, de modification, ni de suppression de messages).
- Un analyseur de protocoles s'exécutant sur une machine M peut observer les communications émises par M ou envoyées à destination de M mais aussi, dans le cas où M est reliée au réseau par un médium partagé avec d'autres machines, toutes les communications véhiculées sur ce médium, même si elles ne sont pas émises par M ni à destination de M.
- Un analyseur de protocoles est parfois appelé *analyseur de paquets* (en référence au fait qu'il est conçu pour les réseaux à commutation de paquets) mais cette dénomination peut être trompeuse car la plupart de ces logiciels (dont Wireshark) prennent en charge des protocoles à différents niveaux de la pile protocolaire de l'Internet :
  - niveau 2 (liaison) : par exemple, des *trames* Ethernet ;
  - niveau 3 (réseau) : par exemple, des *paquets* IP ;
  - niveau 4 (transport) : par exemple, des *segments* TCP ou des *datagrammes* UDP ;
  - niveau 5 (application) : par exemple, des *messages applicatifs (requêtes, réponses)* HTTP, FTP, DNS.

Par abus de langage, le terme *paquet* est souvent employé pour désigner de façon générique un message de n'importe quel niveau capturé par l'analyseur.

Wireshark est un logiciel libre disponible pour la plupart des plateformes, notamment les système Unix (dont Linux et MacOS) et Windows [6]. On peut facilement trouver de nombreux tutoriels et documentations pour apprendre à s'en servir. Cet énoncé ne décrit pas en détails la prise en main du logiciel mais des indications seront données au cours de la séance de TP. On décrit seulement ici les principaux composants de l'interface graphique de Wireshark (de haut en bas) :

6. Voir la page du projet : `http://www.wireshark.org`

Menu de commandes Il permet notamment de lancer ou d'arrêter une capture, et de sauvegarder une trace capturée ou de charger une trace existante pour la visualiser.

Menu de filtrage Il permet d'indiquer des conditions de filtrage qui impactent les paquets affichés (ou non).

Fenêtre de séquence des paquets Cette fenêtre affiche la séquence des paquets capturés (en fonction de critères de filtrage et des critères de tri choisis par l'utilisateur), avec une ligne par paquet. Noter que les deux premières colonnes (numéros de paquet et date de capture) sont des informations ajoutées par Wireshark et qu'elles ne sont pas présentes dans le contenu des paquets.

Fenêtre de détail d'un paquet Affiche les détails sur le contenu du paquet actuellement sélectionné dans la fenêtre de séquence. Pour un paquet de niveau i, on peut notamment obtenir la valeur des différents champs du protocole concerné, ainsi que des informations sur le paquet de niveau $i+1$ qu'il encapsule.

Fenêtre de contenu d'un paquet Affiche les détails sur le contenu complet du paquet actuellement sélectionné dans la fenêtre de séquence. Le contenu est affiché sous forme brute (valeurs hexadécimales) ainsi que sous forme ASCII.

Afin de pouvoir contrôler les interfaces réseau d'une machine (pour capturer les communications), Wireshark nécessite des privilèges d'administrateur [7]. Pour contourner les problèmes de sécurité soulevés par cette contrainte, le TP sera effectué sur des machines virtuelles (reliées par un réseau Ethernet virtuel). Des indications sur la mise en place des machines virtuelles seront données pendant la séance de TP.

Remarque : de nombreuses informations sont disponibles sur le wiki de Wireshark (`http://wiki.wireshark.org`), notamment sur les protocoles que l'outil permet d'étudier (`http://wiki.wireshark.org/ProtocolReference`).

3 Configuration des machines virtuelles

On utilisera le logiciel Virtualbox [8]. Virtualbox peut être installé sur les systèmes Windows, Unix et Mac.) pour gérer les machines virtuelles utiles pour ce TP. La prise est main de Virtualbox n'est pas décrite dans cet énoncé mais des indications seront données pendant la séance de TP (et de nombreux tutoriels sont disponibles sur le Web, en complément de la documentation officielle).

Deux machines virtuelles préinstallées sont fournies ; pour simplifier, elles seront nommées A et B dans la suite de ce document. Ces machines sont installées avec une distribution Xubuntu Linux. Selon les expériences, on utilisera une ou deux machines. Ces deux machines sont configurées de façon identique, sauf pour les paramètres suivants : nom, fond d'écran du bureau, adresse MAC de l'interface réseau et parfois le nom de l'interface réseau.

7. Le lecteur soucieux d'installer Wireshark sur sa propre machine en minimisant les problèmes de sécurité trouvera des informations intéressantes dans la page suivante : `http://wiki.wireshark.org/CaptureSetup/CapturePrivileges`.

8. Voir `http://virtualbox.org`.

Les instructions nécessaires pour récupérer et importer les machines dans Virtualbox seront données pendant la séance de TP.

Remarque : dans les premiers exercices, on occultera temporairement la façon dont une machine obtient la configuration de ses principaux paramètres réseau : adresse IP, adresse IP de la passerelle (routeur permettant l'accès à d'autres réseaux), adresse IP du serveur DNS à utiliser. Ces aspects sont liés au protocole DHCP, étudié en section 8. On rappelle par ailleurs que l'adresse MAC associée à une interface réseau est fixée de façon matérielle [9].

Chaque machine est par défaut configurée avec une interface réseau Ethernet [10]. Pour chaque interface, Virtualbox offre différents modes de fonctionnement possibles. Nous ne décrivons ici que les trois modes utiles pour ce TP (se référer à la documentation de Virtualbox pour les autres) :

NAT Dans ce mode, Virtualbox joue le rôle de passerelle NAT [11] vis-à-vis de l'interface virtuelle et lui attribue automatiquement une configuration réseau (IP) via DHCP. Ainsi, si le système d'exploitation hôte (le système qui héberge l'application Virtualbox) est connecté à Internet, il est très simple de permettre à la machine virtuelle d'accéder elle-même à Internet. *Cependant, si on lance plusieurs machines virtuelles, celles-ci ne peuvent pas communiquer directement (Virtualbox les isole les unes des autres, même si leur configuration IP semble indiquer qu'elles appartiennent au même réseau).*

Réseau privé hôte Dans ce mode (appelé *Host-only networking* dans la version anglaise), des interfaces appartenant à différentes machines virtuelles peuvent appartenir même réseau local et peuvent communiquer directement. Elles bénéficient également d'un serveur DHCP (attribution automatique de configuration IP) et peuvent communiquer avec le système d'exploitation hôte (qui dispose d'une interface virtuelle reliée à chaque réseau privé) [12]. La communication avec l'hôte peut être utile pour transférer des données (un serveur SSH est installé sur chaque machine virtuelle fournie pour le TP). *Cependant, ce mode de configuration n'est pas disponible sur les machines de l'UFR IM2AG.*

Réseau interne Ce mode (appelé *Internal networking* dans la version anglaise) est assez proche du mode précédent (réseau privé hôte). La différence majeure est que l'hôte n'est pas raccordé au réseau virtuel : le réseau de machines virtuelles est complètement isolé de l'extérieur et du système d'exploitation hôte. Par ailleurs, *le serveur DHCP n'est pas activé par défaut dans ce mode.* Pour l'activer le serveur dans le réseau interne (ce réseau est nommé *intnet* par défaut [13]), il faut saisir, sur la machine hôte, la ligne de commande suivante dans un shell (Linux) après le

9. Pour les configurations utilisées dans ce TP, les interfaces réseau sont émulées de façon logicielle par Virtualbox.

10. Virtualbox permet de définir jusqu'à 4 interfaces par machine (ce qui peut être utile pour simuler un routeur, par exemple), mais nous n'exploiterons pas cette possibilité.

11. La notion de passerelle NAT a été présentée en cours.

12. L'utilisateur doit choisir le réseau privé hôte associé à chaque interface. On peut créer un nouveau réseau privé hôte via le menu *Préférences/Réseau* de Virtualbox. Ce menu permet également, pour chaque réseau privé hôte, de configurer l'interface virtuelle de l'hôte et le serveur DHCP.

13. Attention : `intnet` signifie ici *internal network* et non pas *Internet*.

démarrage de Virtualbox [14] :
`VBoxManage dhcpserver add --netname intnet --ip 192.168.100.100 --netmask 255.255.255.0 --lowerip 192.168.100.101 --upperip 192.168.100.254 --enable`
Noter que l'on peut arrêter le serveur DHCP avec la commande suivante : `VBoxManage dhcpserver remove --netname intnet`

Quel que soit le mode de configuration choisi, on peut à tout moment indiquer si l'on souhaite que le câble lié à une interface virtuelle soit raccordé au réseau ou débranché.

Suggestion de piste d'approfondissement : Le simulateur de réseaux GNS3 (`http://gns3.com`) permet de définir des structures de réseaux arbitrairement complexes (notamment avec des routeurs). Les machines du réseau peuvent notamment être mises en œuvre sous la forme de machines virtuelles VirtualBox. (Cet outil ne sera pas étudié dans le cadre de ce TP.)

4 Étude du protocole ARP

Pour cette expérience, configurer dans Virtualbox l'interface de la machine virtuelle A en mode *réseau interne*. Démarrer la machine virtuelle A. Utiliser la commande `ifconfig` pour vérifier la configuration de l'interface Ethernet : repérer l'adresse IP et l'adresse MAC associées à cette interface. Faire de même avec la machine virtuelle B.

Les machines A et B appartiennent au même réseau et peuvent donc communiquer directement via le protocole de niveau 2 utilisé au sein du réseau local (ici, Ethernet). Mais les protocoles de plus haut niveau utilisent des adresses IP (niveau 3). Avant de pouvoir initier une communication avec une machine du même réseau, il faut donc déterminer la correspondance entre son adresse IP et son adresse MAC ; c'est le rôle du protocole ARP, qui fait l'objet de cette expérience.

Sur chaque machine, commencer par vider le cache ARP. Pour cela, on peut utiliser :
— la commande `arp -d adresse`, où `adresse` correspond à l'adresse IP de la machine dont on veut oublier l'adresse MAC
— la commande `ip -s neigh flush all`, qui permet de vider d'un seul coup l'intégralité du cache.

Vérifier ensuite que le cache ARP a bien été vidé (en utilisant la commande `arp`, sans argument).

Sur la machine A lancer Wireshark et activer la capture sur l'interface Ethernet. Lancer ensuite une commande qui déclenche une communication vers la machine B (`ping` par exemple). Une fois la réponse de la machine B obtenue, stopper la capture. Consulter à nouveau le contenu du cache ARP sur chacune des machines (via la commande `arp`).

14. Sous Windows, il faut se placer préalablement dans le bon répertoire (dans le volume `C:` avec la commande `cd "Program Files\Oracle\VirtualBox"`) et remplacer `VBoxManage` par `VBoxManage.exe`.

Analyser la trace Wireshark en essayant de comprendre la séquence de messages ARP et les valeurs de chacun de leurs champs.

Remarques :
— La commande `arp` sert uniquement à consulter/modifier la table ARP d'une machine. Elle ne déclenche pas directement l'envoi de messages du protocole ARP.
— Pour la signification du booléen *"Is gratuitous"* indiqué par Wireshark pour une requête ARP, voir notamment `http://wiki.wireshark.org/Gratuitous_ARP`.
— Les informations de la page suivante peuvent éventuellement être utiles : `http://www.erg.abdn.ac.uk/users/gorry/course/inet-pages/arp.html`.
— Pour des informations sur la façon dont Linux gère la table ARP d'une machine, voir la page de manuel : `man 7 arp` (à ne pas confondre avec le manuel de la commande `arp` mentionnée précédemment : `man 8 arp`).

Dans un second temps, sélectionner l'un des messages ARP et étudier la trame Ethernet qui l'encapsule (cf. `http://fr.wikipedia.org/wiki/Ethernet#Trame_Ethernet_II`). Comprendre la signification des différents champs de la trame et repérer où se trouvent les octets du message ARP encapsulé.

5 Étude des protocoles IP et ICMP

Pour cette expérience, arrêter la machine virtuelle A, choisir la configuration réseau *NAT* sous Virtualbox, puis redémarrer la machine virtuelle A.

5.1 IP(v4)

Commencer par vérifier l'adresse IP et l'adresse MAC de la machine à l'aide de la commande `ifconfig`. À l'aide de la commande `route`, repérer également l'adresse IP de la passerelle (*gateway*) par défaut, c'est-à-dire le routeur à contacter pour communiquer avec des machines hors du réseau local.

Lancer Wireshark et démarrer une capture sur l'interface Ethernet. Exécuter ensuite la commande `ping 195.83.24.194`[15]. Après avoir obtenu plusieurs échanges requête-réponse, arrêter la commande `ping` (en pressant ctrl-c) mais laisser la capture active. Relancer alors la commande `ping` en changeant la taille du message envoyé : `ping -s 3500 195.83.24.194`. Après avoir capturé plusieurs requêtes ICMP[16], arrêter la commande `ping` ainsi que la capture Wireshark. Dans un premier temps, on considèrera ces messages seulement au niveau IP ; le niveau ICMP sera étudié dans un second temps (cf. 5.2).

15. Cette adresse IP est celle de la machine qui héberge le serveur Web de l'Université Grenoble Alpes.
16. De nombreux serveurs et routeurs filtrent les requêtes ICMP qui dépassent une certaine taille, soit parce qu'ils ne peuvent pas les prendre en charge, soit pour se protéger contre des attaques par saturation des ressources (déni de service). Ceci peut expliquer pourquoi certaines requêtes ICMP qui excédent la taille standard (56 octets de données ajoutées aux 8 octets d'en-tête) ne reçoivent pas de réponse.

Regarder le contenu du premier message (*echo request*), au niveau du paquet IP correspondant. Repérer et comprendre notamment les informations suivantes :

Numéro de version du protocole IP Où est-il placé dans l'en-tête ? Quelle est la raison qui motive cet emplacement ?

Internet Header Length (IHL) Quels sont la signification et l'intérêt de ce champ ?

Total length Mêmes questions que pour le champ IHL.

Header checksum Quel est son rôle ? Comment cette valeur est-elle calculée ?

Adresses IP Quelles sont les adresses IP source et destination du paquet ?

Étudier également la trame Ethernet qui encapsule le paquet IP. Repérer notamment le type de contenu (paquet IP) et les adresses MAC source et destination. À quelle machine l'adresse MAC destination correspond-t-elle [17] ?

Regarder ensuite le contenu d'un autre message *echo request* généré par la seconde exécution de la commande `ping`. Étudier notamment les champs *Identification*, *Flags* et *Fragment Offset* (et comparer leurs valeurs à celles du paquet étudié au début de la question). Observer également combien de trames Ethernet sont associées au paquet IP. Pour chacune de ces trames, déterminer combien d'octets du paquet IP elle encapsule.

5.2 ICMP

ICMP (*Internet Control Message Protocol*) fait partie des protocoles centraux de l'Internet. ICMP est utilisé pour détecter et signaler les problèmes d'acheminement de messages au sein d'un ensemble de réseaux interconnectés. Un routeur renvoie par exemple un message d'erreur ICMP à l'émetteur d'un paquet qui ne peut pas être acheminé (route inexistante), dont la durée de vie a expiré, ou encore, dont l'en-tête IP est incorrect. Conceptuellement, on peut considérer ICMP comme un sous-protocole d'IP mais, en pratique, ICMP est un protocole distinct, dont les messages sont encapsulés dans des paquets IP.

Le programme client-serveur `ping` s'appuie sur le protocole ICMP [18]. Le client ping envoie un message ICMP de type *Echo request*, auquel un serveur ICMP répond (s'il le souhaite) par un message *Echo reply*.

Reprendre la trace obtenue en 5.1 et étudier les différents champs d'une requête ping et d'une réponse. Observer également le champ *Protocol* d'un paquet IP qui encapsule un message ICMP.

17. On peut confirmer cette hypothèse en consultant la table ARP via la commande `arp`.

18. Plus précisément, la commande `ping` correspond effectivement à un programme client, qui s'exécute sous la forme d'un processus. En revanche, les routeurs et la plupart des systèmes d'exploitation disposent d'un serveur ICMP implémenté directement au sein de leur pile de protocoles réseau. C'est pourquoi on ne trouve généralement pas de processus serveur ICMP.

Le programme `traceroute` s'appuie également sur ICMP mais d'une manière un peu différente de celle de ping. Traceroute envoie une série de paquets IP de durée de vie (logique) variable à destination de la machine cible. Les requêtes sont envoyées sous forme de messages UDP [19] (le protocole UDP est étudié en section 6) avec un numéro de port improbable (ayant peu de chances d'être utilisé par la machine cible). Lorsqu'un routeur reçoit un paquet IP encapsulant une requête traceroute, il décrémente sa durée de vie et, si cette dernière atteint zéro, le routeur renvoie un message d'erreur ICMP. Des détails supplémentaires sont disponibles dans la page de manuel de traceroute (`man 8 traceroute`).

Démarrer une capture sur la machine A, exécuter la commande `traceroute 131.111.150.25` puis arrêter la capture lorsque la commande se termine [20]. Étudier la trace en observant notamment (i) le champ TTL des paquets IP contenant les requêtes, (ii) l'adresse IP de l'émetteur des paquets IP contenant les réponses ICMP, (iii) le type et le code des messages ICMP envoyés par les routeurs intermédiaires, (iv) le reste du contenu du paquet IP associé à la réponse d'un routeur intermédiaire. Observer également les particularités du contenu des trois derniers messages ICMP reçus. On ignorera les traces de type DNS pour cette expérience.

6 UDP

Reprendre la trace obtenue dans l'expérience précédente (section 5.2) et étudier le contenu d'une requête traceroute. Observer notamment les points suivants : (i) le champ *protocol* dans le paquet IP qui encapsule un message UDP et (ii) les différents champs de l'en-tête UDP. Faire le lien entre la taille des champs associés aux numéros de ports et la plage de valeurs autorisées pour ces numéros.

Remarque : On peut s'interroger sur l'intérêt de disposer d'une somme de contrôle dans un en-tête de message UDP alors que l'en-tête d'un paquet IP dispose également d'une somme de contrôle. Il y a plusieurs raisons qui justifient ce choix. Tout d'abord, au niveau IP, la somme de contrôle ne concerne (protège) que le contenu de l'en-tête IP alors que la somme de contrôle présente dans un en-tête de message UDP concerne l'ensemble du message UDP. De plus, un paquet IP peut encapsuler un protocole autre que UDP et, inversement, un paquet UDP peut potentiellement être acheminé par un protocole autre qu'IP. Ces remarques sur la somme de contrôle d'un en-tête de message UDP s'appliquent également à celle d'un en-tête de segment TCP.

7 TCP

Remarque : De nombreux aspects de TCP ne sont pas étudiés dans ce TP (notamment ceux liés au contrôle de flux et au contrôle de congestion, ainsi que les détails sur la gestion des connexions, des erreurs et de la fiabilité des transmissions). Le lecteur intéressé pourra se référer aux ouvrages cités dans la bibliographie ainsi qu'aux nombreuses

19. Sous Windows, le programme `tracert` envoie des requêtes ICMP *echo* plutôt que des requêtes UDP.
20. Vous pouvez éventuellement remplacer cette adresse IP par l'une des autres étudiées dans l'exercice sur l'utilisation de traceroute en section 1

documentations disponibles.

Pour cette série d'expériences, arrêter les machines virtuelles A et B et modifier leur configuration réseau Virtualbox pour les placer en mode *réseau interne* sur le même réseau. Redémarrer ensuite les deux machines virtuelles.

7.1 Observations avec Wireshark

Lancer une capture Wireshark sur la machine A. Sur la machine B, lancer le serveur `echo` (cf. TP réseau sur la programmation client-serveur avec sockets) sur un port au choix. Sur la machine A, lancer l'exécution du client `echo` et saisir quelques lignes à envoyer au serveur. Puis fermer la connexion (ctrl-d) et arrêter la capture.

Observer la séquence de messages TCP échangés. Retrouver l'adresse IP et le numéro de port de la machine source et de la machine destination dans chaque segment TCP. Les flags utilisés dans les trois premiers messages (SYN, SYN/ACK, ACK) correspondent à l'établissement de la connexion entre le client et le serveur. De même, les séquences de flags FIN et ACK échangées dans chaque sens à la fin du dialogue correspondent à la fermeture de la connexion. Étudier également le rôle des champs *Sequence number* et *Acknowledgment number* dans l'en-tête TCP (à l'aide d'une documentation externe sur TCP [21]).

Sur la machine B, lancer le mini-serveur Web (cf. TP sur le mini-serveur HTTP), par exemple sur le port 8080 [22]. Sur la machine A, démarrer une capture Wireshark puis ouvrir un navigateur Firefox et saisir l'URL : `http://192.168.56.102:8080/dudh.txt` (à modifier en fonction de l'adresse IP de la machine B, supposée ici égale à 192.168.56.102) [23]. Sur la machine A, arrêter la capture une fois la page affichée dans le navigateur.

Retrouver dans la trace la requête HTTP GET et la réponse (200 OK) [24]. On cherche maintenant à étudier plus précisément les segments TCP associés au transport de ces deux messages applicatifs. On peut notamment remarquer les points suivants :

- La requête GET est de petite taille et elle est contenue dans un seul segment TCP.
- La réponse OK est plus volumineuse : l'en-tête HTTP généré par Tiny nécessite 93 octets et le contenu du fichier dudh.txt fait 12058 octets (comme indiqué dans le champ `Content-length` dans l'en-tête de la réponse). Ce message applicatif est en fait envoyé en plusieurs segments ; la raison est expliquée ci-après. TCP s'appuie sur IP (un segment TCP est encapsulé dans un paquet IP). Lorsque la couche TCP reçoit une séquence d'octets à envoyer (de la part de la couche applicative),

21. Voir par exemple :
    - `http://en.wikipedia.org/wiki/Transmission_Control_Protocol#Reliable_transmission`
    - `http://packetlife.net/blog/2010/jun/7/understanding-tcp-sequence-acknowledgment-numbers/`

22. Wireshark associe le port 8080 au nom symbolique `http-alt` : port alternatif du protocole HTTP (le port standard étant 80).

23. Si l'expérience est répétée plusieurs fois, penser à vider le cache du navigateur entre chaque essai.

24. Remarque : dans cette expérience, on ne s'intéresse pas aux détails du protocole HTTP, qui seront étudiés ultérieurement (cf. section 9).

celle-ci essaye d'éviter la fragmentation IP [25] En effet, la fragmentation IP présente au moins deux inconvénients. D'une part, elle consomme des ressources au niveau des routeurs (réassemblage d'un paquet avant retransmission, potentiellement elle-même fragmentée) [26]. D'autre part, elle rend les mécanismes de retransmission de TCP inefficaces : en cas de perte d'un seul fragment, il est nécessaire de retransmettre tous les fragments qui composent le paquet. Sur un réseau Ethernet avec des trames de 1500 octets, TCP borne ainsi la quantité de données applicatives par segment à 1460 octets [27]. On peut d'ailleurs observer une négociation de cette taille maximum dans les segments échangés pour établir une connexion TCP (cf. champ *Maximum segment size* dans la partie *Options* de l'en-tête TCP). En pratique, seulement 1448 des 1460 octets sont généralement disponibles pour les données applicatives, car 12 octets optionnels d'en-tête sont utilisés par la plupart des implémentations de TCP (notamment dans les systèmes Unix). On comprend ainsi pourquoi au moins 9 segments TCP sont nécessaires pour véhiculer les 12151 octets de la réponse du serveur HTTP.

— Un certain nombre de segments TCP ne contiennent aucune donnée de niveau applicatif, notamment du client vers le serveur après l'envoi du GET (voir les lignes de Wireshark qui contiennent la mention `[ACK]`.). Ces segments servent à véhiculer des acquittements de données reçues (Wireshark indique le segment auquel l'acquittement fait référence) [28].

7.2 Observations avec Telnet

Le programme client `telnet` permet notamment de se connecter à distance sur une machine, en contactant (via le protocole TCP) un serveur telnet, qui écoute par défaut sur le port n°23. Telnet est aujourd'hui obsolète et supplanté par `ssh`, qui offre les mêmes fonctionnalités et davantage de garanties en matière de confidentialité des informations échangées (les données transmises sur le réseau sont chiffrées alors que telnet les envoie en clair).

Telnet reste cependant utile pour effectuer des tests : en précisant (en paramètre) un numéro de port différent du port par défaut (23), il est possible de se connecter à n'importe quel serveur basé sur le protocole TCP. Ceci permet notamment d'observer en détails le contenu (brut) des réponses d'un serveur (pour les protocoles utilisant des messages en mode texte).

25. La fragmentation IP a été étudiée en section 5.1. En résumé, un paquet IP(v4) peut contenir au maximum 65535 octets au total (en-tête et données encapsulées) car le champ de l'en-tête contenant cette information est codé sur 16 bits. Cependant, les couches de niveau inférieur utilisées pour transmettre le paquet peuvent imposer des contraintes de taille plus fortes (par exemple, dans le réseau Ethernet utilisé pour ce TP, un trame peut au maximum contenir 1500 octets de niveau supérieur). Ainsi, si la taille d'un paquet IP excède la taille maximum disponible dans une trame, celui-ci doit être fragmenté en plusieurs trames.

26. Remarque : Contrairement à IPv4, IPv6 n'autorise d'ailleurs pas la fragmentation et les réassemblage des paquets IP au niveau des routeurs.

27. 1460 octets = 1500 octets moins (20 octets pour l'en-tête IP + 20 octets pour l'en-tête TCP).

28. Comme l'expérience précédente l'a montré, un segment qui contient des données peut également véhiculer un acquittement.

Lancer le serveur echo sur la machine B. Sur la machine A, utiliser `telnet` (plutôt que le programme `echoclient`) pour se connecter et interagir avec le serveur (voir `man telnet` pour les détails d'utilisation).

Selon le même principe, utiliser ensuite `telnet` (sur la machine A) pour dialoguer avec le serveur web Tiny (sur la machine B). Une fois la connexion établie, saisir dans `telnet` la ligne suivante, terminée par deux appuis sur la touche *entrée* :

```
GET /dudh.txt HTTP/1.0
```

Observer la réponse obtenue.

Remarque : À chaque fois que la touche `entrée` est pressée, le programme `telnet` lit la ligne saisie et lui ajoute le caractère `CR` (*carriage return*) suivi du caractère `LF` (*line feed*), représentés respectivement par `\r` et `\n` en langage C). Ceci est cohérent avec le protocole HTTP, qui spécifie que chaque ligne (d'en-tête) doit être terminée par une paire `<CR><LF>`.

7.3 Observations avec TcpCatcher

Remarque préliminaire : Le logiciel TcpCatcher décrit ci-dessous (et utilisé dans certaines des sections suivantes) ne semble plus être disponible/téléchargeable actuellement. Pour les exercices décrits ci-après, il est possible d'utiliser à la place le logiciel Fiddler (disponible pour Windows, MacOs et Linux), qui fournit globalement les même fonctionnalités que TcpCatcher. Le proxy Fiddler écoute sur le port TCP 8888. Fiddler (préinstallé dans la machine virtuelle pour ce TP) est disponible sur la page suivante : `https://www.telerik.com/fiddler`.

Le logiciel TcpCatcher [29] est un analyseur de protocoles spécialisé pour les protocoles TCP et offre des fonctionnalités avancées pour HTTP. Comparé à Wireshark :

- il couvre moins de protocoles (focus sur les couches transport/TCP et application) mais il offre certaines fonctionnalités avancées pour HTTP et les formats de documents du Web ;
- sa présence n'est pas transparente pour les applications observées (il se présente sous la forme d'un serveur mandataire/*proxy* que les clients doivent contacter) ;
- il ne donne pas beaucoup de détails sur les échanges au niveau TCP (on peut voir les flux d'octets applicatifs échangés mais pas les segments TCP distincts, ni leurs en-têtes) ;
- il est facilement à utiliser et ne nécessite pas de privilèges administrateur ;
- il permet d'intercepter les flux d'octets (ou messages HTTP) envoyés et de les modifier si nécessaire avant de les retransmettre (fonctionnalité utile pour le débogage d'applications réparties).

Dans cet exercice, nous étudierons seulement TcpCatcher en mode *proxy TCP* (générique) ; le mode *proxy HTTP* sera étudié en section 9.

29. Voir `http://www.tcpcatcher.org/`.

Sur la machine B, lancer le serveur echo (cf. 7.1). Sur la machine A, lancer TcpCatcher avec la ligne de commande suivante : `java -jar TcpCatcher.jar`. Dans la fenêtre *Settings*, choisir un port d'écoute (8200 par exemple), sélectionner le mode *TCP proxy*, indiquer l'adresse IP de la machine B (*Server Host*) et le port d'écoute choisi pour le serveur echo sur la machine B (*Server Port*). Puis activer le proxy TcpCatcher en cliquant sur *Apply Settings and Start*. Toujours sur la machine A, lancer un client (`telnet` ou `echoclient`, au choix) en indiquant `localhost` pour le nom du serveur et 8200 pour le port. Échanger quelques lignes de dialogue avec le serveur et observer le résultat dans TcpCatcher.

8 DHCP

Le protocole DHCP (Dynamic Host Configuration Protocol) permet à une machine d'obtenir automatiquement une configuration IP au sein d'un réseau lorsqu'elle y est raccordée. Par configuration IP, on entend : une adresse IP pour la machine, le masque de réseau ainsi que l'adresse IP de la passerelle (le routeur) du réseau ainsi que (de manière optionnelle) les adresses IP des serveurs DNS du réseau. Ce protocole ne nécessite aucune connaissance préalable de la configuration du réseau de la part de la nouvelle machine [30].

Pour cette expérience, commencer par éteindre la machine virtuelle A. Sous Virtualbox, pour la configuration réseau, choisir *réseau interne* et décocher la case *câble branché*. Démarrer ensuite la machine virtuelle et lancer une capture Wireshark. Après le démarrage de la capture, retourner dans la configuration de Virtualbox (sans arrêter la machine) pour rebrancher le câble virtuel. Attendre quelques secondes jusqu'à ce qu'une fenêtre du bureau Linux indique que la configuration réseau est opérationnelle. Utiliser ensuite la commande `ifconfig` pour vérifier que la machine dispose d'une configuration IP valable. Arrêter la capture Wireshark.

Étudier les messages de type DHCP présents dans la trace Wireshark. Quelles sont les adresses IP et les numéros de ports utilisés dans le premier message et comment peut-on les expliquer ? Quel est le rôle de chacune des quatre phases du protocole ?

Remarques :
- Les quatre phases à étudier sont liées à la séquence de messages suivante : (i) *DHCP Discover*, (ii) *DHCP Offer*, (iii) *DHCP Request*, (iv) *DHCP ACK*.
- On observe parfois une séquence préalable (*DHCP Request* suivie de *DHCP NAK*) que l'on pourra ignorer. Ce préambule s'explique de la façon suivante : la machine cliente commence par demander une adresse IP qu'elle a déjà obtenue dans le passé et le NAK (*negative acknowledgement* : acquittement négatif) correspond à un refus de la part du serveur.
- On observe aussi parfois une séquence plus courte (*DHCP Request* suivie de *DHCP ACK*). Dans ce cas, la machine cliente commence par demander une adresse IP

30. Mais cette nouvelle machine peut éventuellement être soumise à une phase d'admission préalable, basée sur un mécanisme d'authentification (par exemple, l'utilisation d'une clé de connexion pour un réseau sans fil protégé).

qu'elle a déjà obtenue dans le passé et le serveur accepte sa demande. Pour déclencher l'apparition d'une séquence complète à quatre phases, une possibilité est de redémarrer la machine virtuelle en configurant son interface réseau dans un autre mode (*NAT* si elle était configurée en *réseau interne*, ou inversement). En effet, les configurations réseau (notamment les adresses IP) allouées par le serveur DHCP de Virtualbox ne sont pas les mêmes dans ces deux modes.

9 HTTP

Le but de cet exercice est d'étudier le protocole HTTP par le biais d'interactions entre un client et un serveur HTTP réalistes.

Pour le serveur Web (HTTP) cible, deux options sont possibles :

— Utiliser le serveur Web Apache qui est installé sur les machines virtuelles A et B (et lancé automatiquement au démarrage du système). Le répertoire racine de ce serveur est `/var/www` (il est possible d'y déposer des fichiers et des sous-répertoires). Si le client et le serveur sont exécutés sur la même machine, il n'y a pas de contrainte particulière sur la configuration réseau à sélectionner dans Virtualbox. Sinon, utiliser la même configuration qu'en section 7.

— Utiliser un ou plusieurs serveurs Web de l'Internet. Dans ce cas, choisir la configuration réseau *NAT* dans Virtualbox pour la machine virtuelle qui héberge le client HTTP. De plus, si le réseau hôte filtre les communications HTTP vers l'Internet (comme à l'Université Grenoble Alpes par exemple), il est nécessaire d'indiquer à TcpCatcher le proxy Web à utiliser (voir détails ci-dessous).

Pour observer les communications, nous utiliserons TcpCatcher (présenté en section 7.3), configuré en mode *proxy HTTP*. Sur la machine virtuelle A (qui hébergera aussi le client), lancer TcpCatcher avec la ligne de commande suivante : `java -jar TcpCatcher.jar`. Dans la fenêtre *Settings*, choisir un port d'écoute (8200 par exemple) et sélectionner le mode *HTTP proxy*. Si le réseau hôte impose un proxy Web pour les communications vers l'Internet, cocher la case *Target corporate web Proxy* et indiquer le nom et le port du proxy (exemple pour l'UGA : `www-cache.ujf-grenoble.fr:3128`) [31]. Puis activer le proxy TcpCatcher en cliquant sur *Apply Settings and Start*.

Pour le client HTTP, lancer le navigateur Firefox sur la machine A. La première chose à faire est de configurer le navigateur pour lui indiquer qu'il doit contacter un proxy (TcpCatcher) plutôt que le serveur cible. Pour cela [32], aller dans le menu *Edit/Preferences/Advanced/Network/Settings*. Choisir *Manual proxy configuration* et, pour la ligne *HTTP Proxy*, indiquer `localhost` et le numéro de port attribué à TcpCatcher (8200 dans l'exemple ci-dessus). Penser aussi à supprimer `localhost` et `127.0.0.1` de la case *No Proxy for*.

31. Quand cette option est activée, TcpCatcher retransmet les requêtes HTTP qu'il reçoit vers le proxy sortant plutôt que vers le serveur cible. On a donc deux proxys en cascade.

32. Attention, la procédure à suivre et l'emplacement du menu pour configurer le proxy varient selon les versions de Firefox.

On peut alors contacter un serveur Web en entrant une URL dans la barre d'adresse du navigateur. Attention : l'URL doit contenir le nom (ou l'adresse IP) du serveur cible (et non pas de la machine qui héberge le proxy TcpCatcher).

Essayer de contacter différents serveurs Web et observer le résultat obtenu dans TcpCatcher.

Remarques :

— En complément de TcpCatcher, on peut éventuellement effectuer une capture avec Wireshark pour avoir une vue plus précise du trafic réseau. Afin de voir l'ensemble du trafic (y compris le trafic interne à la machine, entre le navigateur et TcpCatcher), choisir l'interface *any* (plutôt que l'interface Ethernet) pour la capture.

— Ne pas oublier qu'une requête HTTP ne correspond qu'à un seul objet/fichier mais que lorsqu'un navigateur Web récupère une page Web, celui-ci envoie automatiquement de nouvelles requêtes pour obtenir les images référencées dans la page.

— La plupart des serveurs Web compressent à la volée le contenu des fichiers qu'ils renvoient aux clients, ce qui peut compliquer les observations des messages [33]. On peut désactiver cette fonctionnalité en demandant à Firefox de ne pas indiquer aux serveurs qu'il accepte les contenus compressés. Pour cela, taper `about:config` dans la barre d'adresses pour accéder à un panneau de configuration des options avancées. Chercher l'option `Network.http.accept-encoding :` et supprimer les valeurs `gzip` et `deflate`.

Étudier les champs des en-têtes présents dans les requêtes et les réponses HTTP capturées et essayer de comprendre leur rôle à l'aide de documentations externes. Voir par exemple les liens suivants :

— `http://en.wikipedia.org/wiki/List_of_HTTP_headers`

— spécification HTTP 1.1 (`http://tools.ietf.org/html/rfc2616`)

Essayer de recharger une page en appuyant sur le bouton *reload* du navigateur. Observer les messages capturés avec TcpCatcher. Le serveur renvoie-t-il le contenu complet de la page demandée ?

Le navigateur utilise par défaut des connexions persistantes (prise en charge par la plupart des serveurs Web). Observer les différences de comportement obtenues en utilisant des connexions non persistantes. Pour cela, taper `about:config` dans la barre d'adresses pour accéder à un panneau de configuration des options avancées. Modifier la valeur des options `Network.http.version` et `Network.proxy.version` : 1.0 au lieu de 1.1. Une autre façon de procéder consiste à activer le *catch mode* de TcpCatcher pour intercepter et modifier à la volée le contenu de l'en-tête des requêtes (dans cet exemple, c'est le champ `Connection` qu'il faut modifier).

33. TcpCatcher est cependant capable de décompresser à la volée le contenu des messages.

10 DNS

10.1 Rappels et compléments de cours

Les principes généraux du DNS (Domain Name System) ont été présentés en cours. Par ailleurs, des compléments sur les principales informations disponibles dans les serveurs DNS sont disponibles dans les pages suivantes :

— `http://dns-record-viewer.online-domain-tools.com`
— `http://www.debianhelp.co.uk/dnsrecords.htm`

Noter qu'une machine n'est pas obligée d'envoyer ses requêtes DNS au serveur DNS du réseau auquel elle appartient (même dans le cas d'une configuration via DHCP, on peut fixer manuellement l'adresse du serveur DNS à contacter). Cela peut être intéressant pour deux raisons. D'une part, car on peut éventuellement ne pas faire confiance aux serveurs DNS de certains réseaux (par exemple, dans le cas d'un réseau sans fil public dont on ne sait pas grand chose). D'autre part, car certains serveurs peuvent parfois refuser de traiter certaines requêtes. Il existe des serveurs DNS publics, c'est-à-dire acceptant les requêtes d'interrogation issues de n'importe quelle machine/réseau [34].

L'utilitaire `dig` permet d'envoyer des requêtes DNS. Les lignes de commandes les plus simples pour une requête `dig` sont les suivantes :

— `dig nom` : demande l'adresse IP correspondant à `nom`
— `dig -x ip` : le demande le nom correspondant à l'adresse `ip`

Plus généralement, une requête `dig` est de la forme : `dig @serveur parametre type` où :

— `serveur` correspond au serveur DNS à interroger (par défaut, `dig` utilise le serveur DNS spécifié dans la configuration réseau de la machine cliente — il s'agit typiquement du serveur DNS du réseau auquel appartient la machine cliente)
— `parametre` correspond à l'objet de la requête
— `type` correspond au type de requête (par défaut, `dig` envoie une requête de type `A`, pour obtenir l'adresse associée à un nom).

Quelques précisions supplémentaires :

— La commande `dig NS` permet d'obtenir la liste des serveurs racine.
— L'option `+trace` permet d'afficher toute la chaîne de requêtes qui conduit à l'obtention d'une réponse.
— Les requêtes DNS envoyées par `dig` sont par défaut récursives (cf. schémas du cours). On peut modifier ce comportement avec l'option `+norecurse`. L'option `+trace` désactive automatiquement les requêtes récursives.

De nombreux serveurs proposent également un interface Web pour envoyer des requêtes dig (voir par exemple `http://www.digwebinterface.com`). Cela peut être utile car certains routeurs/pare-feux au sein d'un réseau filtrent parfois les requêtes DNS émises directement vers l'extérieur.

34. On peut notamment citer les serveurs DNS publics suivants :
— Google : adresses IP(v4) `8.8.8.8` et `8.8.4.4`
— OpenDNS : adresses IP(v4) `208.67.222.222` et `208.67.220.220`

10.2 Exercices

Reprendre les exemples d'utilisation de la commande `dig` vus en cours.

Déterminer le nom du serveur DNS principal ainsi que l'adresse électronique du responsable technique pour le domaine `inria.fr`.

Un nom de machine associé à un domaine (DNS) de premier niveau correspondant à un critère géographique (par exemple, `fr` pour la France, `jp` pour le Japon, `eu` pour l'Europe ...) implique-t-il nécessairement que la machine soit physiquement localisée dans la zone géographique concernée [35] ? Dans le cas contraire, donner un contre-exemple, avec l'aide éventuelle des outils vus aux questions précédentes et des ressources de localisation géographiques disponibles en ligne. À ce sujet, voir par exemple :

- `http://www.slac.stanford.edu/comp/net/wan-mon/traceroute-srv.html` (section *Finding Host Information*).
- `http://www-public.int-evry.fr/~maigron/Internet/Adresses.html` (section *Localisation d'une adresse IP*)

Indice : s'intéresser aux grands sites commerciaux et portails web disponibles dans de nombreux pays.

La figure 1 illustre les différentes étapes nécessaires à l'obtention de l'adresse IP associée à un nom de machine (`css.csail.mit.edu`), dans l'hypothèse (peu probable en pratique) où l'on ne dispose pas de caches et de serveurs dupliqués dans la hiérarchie de serveurs DNS. En utilisant une ou plusieurs requêtes DNS (envoyées grâce à la commande `dig`), compléter la figure avec les questions précises et les réponses obtenues pour chaque étape de l'algorithme de résolution.
Refaire la même expérience avec le nom `www.eecs.mit.edu`.

11 NAT

Pour cette expérience, vérifier que la configuration réseau de la machine virtuelle A est en mode *NAT* sous Virtualbox. Si ce n'est pas le cas, éteindre la machine virtuelle, modifier sa configuration dans Virtualbox puis la redémarrer.

Lancer une capture sous Wireshark. Lancer ensuite le navigateur Web Firefox et demander la page suivante : `http://en.dnstools.ch/show-my-ip.html`. La page renvoyée est générée dynamiquement par un script CGI. Arrêter la capture une fois la page affichée dans Firefox. Comparer les informations affichées sur cette page (adresse IP et numéro de port du client) avec celles trouvées dans la trace Wireshark (au niveau de la première réponse HTTP, qui contient la page HTML).

35. N.B. : Cette question porte exclusivement sur des considérations techniques et non sur des aspects légaux, qui peuvent varier selon les pays et évoluer dans le temps.

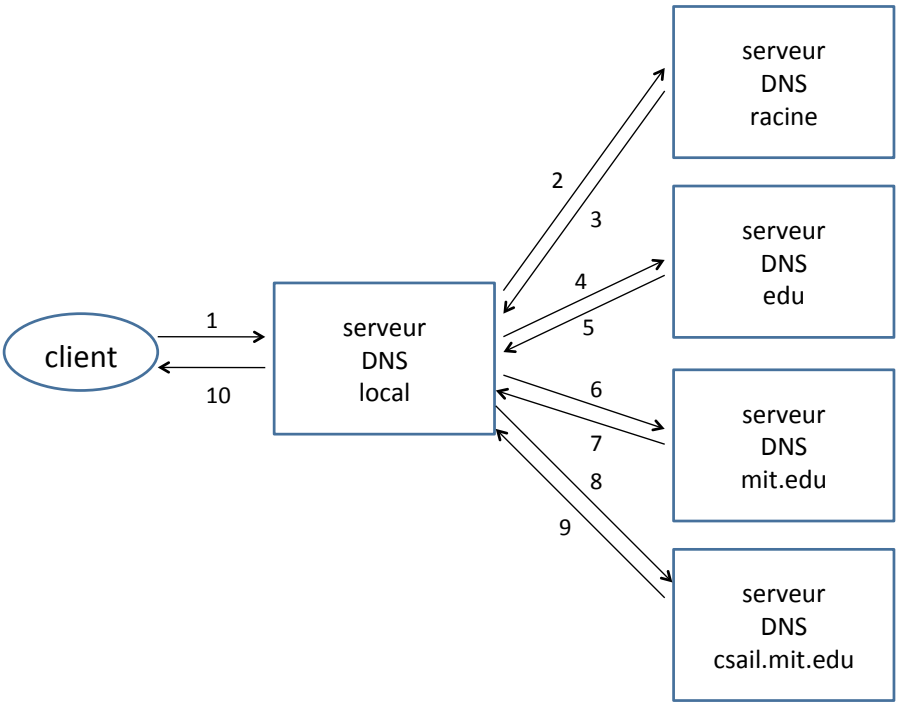

FIGURE 1: Etapes de la résolution DNS du nom `css.csail.mit.edu`

12 Informations sur le propriétaire d'un domaine ou d'une adresse IP

La commande `whois` permet d'interroger des bases de données pour obtenir des informations [36] sur le propriétaire d'un domaine ou d'une adresse IP.

De nombreux sites internet fournissent une interface web pour effectuer des requêtes `whois`. La plupart ne couvrent cependant qu'un sous-ensemble des domaines. Il est recommandé d'utiliser ces interfaces web pour le TP car les requêtes de la commande `whois` sont généralement filtrées par le réseau de l'Université.

- Pour la racine `fr`, voir notamment le site de l'AFNIC : `https://www.afnic.fr/fr/produits-et-services/services/whois/`
- Pour les racines les plus répandues, voir notamment `http://www.networksolutions.com/whois/index.jsp`
- Pour une recherche par adresse IP, voir notamment le site de l'ARIN : `http://ws.arin.net` (ou bien utiliser la commande `dig` pour trouver le nom correspondant puis faire une recherche par nom de domaine ... mais il n'existe pas nécessairement un nom DNS pour chaque adresse IP).

Pour des recherches sur des domaines géographiques, on distingue cinq bases `whois` principales :

36. N.B. : Dans certains cas, les informations obtenues peuvent être incomplètes, obsolètes ou fausses.

— l'AfriNIC pour l'Afrique : *African Network Information Center* (`http://www.afrinic.net`)
— l'APNIC pour l'Asie et le Pacifique : *Asia Pacific Network Information Center* (`http://www.apnic.net`) ;
— l'ARIN pour l'Amérique du Nord : *American Registry for Internet Numbers* (`http://www.arin.net`) ;
— le LACNIC pour l'Amérique Latine et les Caraïbes : *Latin American and Caribbean Network Information Center* (`http://www.lacnic.net`)
— le RIPE pour l'Europe : *Réseaux IP Européens* (`http://www.ripe.net`) ;

Grâce aux indications données ci-dessus, obtenir des informations sur les propriétaires des domaines ou adresses suivants : `univ-grenoble-alpes.fr`, `ujf-grenoble.fr`, `elysee.fr`, `grenoble.com`,
`193.55.76.228`, `209.85.135.104`