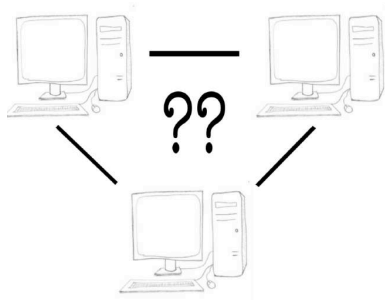


PROGRAMMATION RÉSEAU

Arnaud Sangnier

sangnier@irif.fr

INTRODUCTION



Objectifs du cours

- Ceci n'est pas un cours de réseau
- C'est un cours de **programmation réseau**
- Comprendre les mécanismes réseau
 - Communication
 - Codage de l'information
- Apprendre à programmer en C et en Java
- Programmer des clients d'application déjà existantes
 - Client pour serveur d'envoi de mails
 - Client pour serveur web (qui fait ce que fait un navigateur)
- Développer une application réseau

Quelques informations

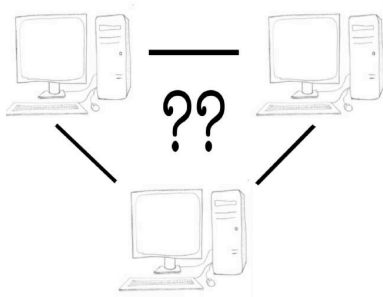
- Un tp par semaine (**Respectez votre groupe de tp**)
- Tps :
 - TP Groupe 1: Mardi 14h00-16h00 - Patrick Lambein-Monette
 - TP Groupe 2: Lundi 10h45-12h45 - Thomas Colcombet
 - TP Groupe 3: Mercredi 14h00-16h00 - Anne Micheli
 - TP Groupe 4: Mardi 8h30-10h30 - Patrick Lambein-Monette
 - TP Groupe 5A: Jeudi 16h15-18h15 - Alliaume Lopez
 - TP Groupe 5B: Vendredi 8h30-10h30 - Giulia Manara
- Page web du cours :
<https://www.irif.fr/~sangnier/enseignement/reseaux.html>
- Évaluation :
 - 1ère session : 40 % Examen + 40 % Projet + 10% TP noté + 10 % QCM
 - Projet donné vers la quatrième semaine
 - Un Tp noté et un QCM
 - 2ème session : 70 % Examen + 30%Projet

À quelles questions ce cours répond

- Comment deux machines peuvent-elles communiquer ?
- Comment connaître la machine avec laquelle on souhaite communiquer ?
- Communiquer pour quoi faire ?
- Quelles sont les différentes façons de communiquer ?
- Quelles informations envoyer ?
- Comment recevoir l'information envoyée ?

Comment communiquer

Généralités réseaux et outils UNIX



- Il faut envoyer de l'information
- Cette information passe d'une application vers une autre
 - Par exemple :
Un navigateur envoie une requête vers un serveur web
- En pratique l'information qui circule est codée en octet
- Cette information peut-être **perdue** ou **erronée**
- L'ordre des messages peut être changé
- Allons nous gérer la perte des messages, le fait que des 'bits' d'information peuvent être changés ?

NON pour le changement des messages

Pour la perte des messages, parfois

Transmission non fiable de données

- Comment peut-on garantir une certaine fiabilité dans l'information envoyée ?
 - Le service de base envoie des données binaires (sous forme de paquets d'octets)
 - Les paquets peuvent être **perdus** ou **dégradés**



Un système de couches

- Comment gère-t-on les pertes d'information ?
 - Mécanisme de détection de pertes
 - Réémission de messages
- Comment gère-t-on la dégradation
 - Redondance de l'information
 - Ajout de bits permettant de savoir les parties du paquet erronées
 - (cf code correcteur d'erreurs)
- Ces mécanismes existent et nous n'avons pas à les programmer
- Comment passe-t-on d'un service avec paquets non fiables à un service fiable ?
 - superposition de couches logicielles chacune avec une mission spécifique

Modèle de référence

- Modèle ISO/OSI (Open System Interconnection) en 7 couches

Layer	Application/Example	Central Device/ Protocols	DOD4 Model
Application (7) Serves as the window for users and application processes to access the network services.	End User layer Program that opens what was sent or creates what is to be sent Resource sharing • Remote file access • Remote printer access • Directory services • Network management	User Applications SMTP	Process
Presentation (6) Formats the data to be presented to the Application layer. It can be viewed as the "Translator" for the network.	Syntax layer encrypt & decrypt (if needed) Character code translation • Data conversion • Data compression • Data encryption • Character Set Translation	JPEG/ASCII EBDIC/TIFF/GIF PICT	
Session (5) Allows session establishment between processes running on different stations.	Synch & send to ports (logical ports) Session establishment, maintenance and termination • Session support - perform security, name recognition, logging, etc.	Logical Ports RPC/SQL/NFS NetBIOS names	Host to Host
Transport (4) Ensures that messages are delivered error-free, in sequence, and with no losses or duplications.	TCP Host to Host, Flow Control Message segmentation • Message acknowledgement • Message traffic control • Session multiplexing	TCP/SPX/UDP	
Network (3) Controls the operations of the subnet, deciding which physical path the data takes.	Packets ("letter", contains IP address) Routing • Subnet traffic control • Frame fragmentation • Logical-physical address mapping • Subnet usage accounting	Routers IP/PIX/ICMP	Internet
Data Link (2) Provides error-free transfer of data frames from one node to another over the Physical layer.	Frames ("envelopes", contains MAC address) [NIC card — Switch — NIC card] (end to end) Establishes & terminates the logical link between nodes • Frame traffic control • Frame sequencing • Frame acknowledgement • Frame delimiting • Frame error checking • Media access control	Switch Bridge WAP PPP/SLIP	Network
Physical (1) Concerned with the transmission and reception of the unstructured raw bit stream over the physical medium.	Physical structure Cables, hubs, etc. Data Encoding • Physical medium attachment • Transmission technique - Baseband or Broadband • Physical medium transmission Bits & Volts	Hub Land Based Layers	

Les couches matérielles

1. Couche Physique

- Rôle : Transmission des données sous forme binaire
- Protocoles : Fibre, câble radio, ...

2. Couche Liaison

- Rôle : Gère la communication entre machines, adresse physique MAC
- Protocoles : Ethernet, Wifi, ...

3. Couche Réseau

- Rôle : Détermine le parcours des données et l'adressage logique (adresse IP)
- Protocoles : IPv4, IPv6, ...

Les couches segments et données

4. Couche Transport

- Rôle : Connexion bout à bout, contrôle de flux
- Protocoles : TCP, UDP

5. Couche Session

- Rôle : Communication points à point
- Protocoles : TLS

6. Couche Présentation

- Rôle : Chiffrement et déchiffrement des données
- Protocoles : SSL, WEP

7. Couche Application

- Rôle : Point d'accès aux services réseaux
- Protocoles : SMTP, IRC, HTTP, SSH

Modèle Internet

- Modèle plus simple à 4 couches

1. Liaison

- Protocoles : ARP, Ethernet

2. Internet

- Protocoles : IPv4, IPv6

3. Transport

- Protocoles : TCP, UDP

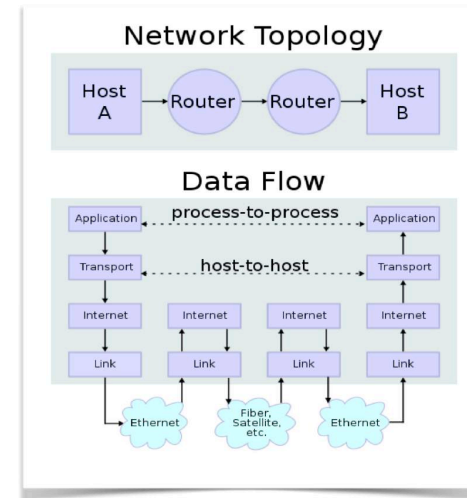
4. Application

- Protocoles : FTP, HTTP, IMAP, POP, SMTP

Ce qui va nous intéresser

- Développer des applications ou services
 - Ce que nous développerons se situera à la couche **Application**
 - Nous utiliserons les protocoles de la couche **Transport**
- Quels protocoles sous-jacents allons-nous utiliser :
 - **TCP (modèle par flux)**
 - **UDP (modèle par paquet)**
- Pour utiliser ces services, nécessité de savoir le nom ou l'adresse des machines (adresse IPv4, IPv6)

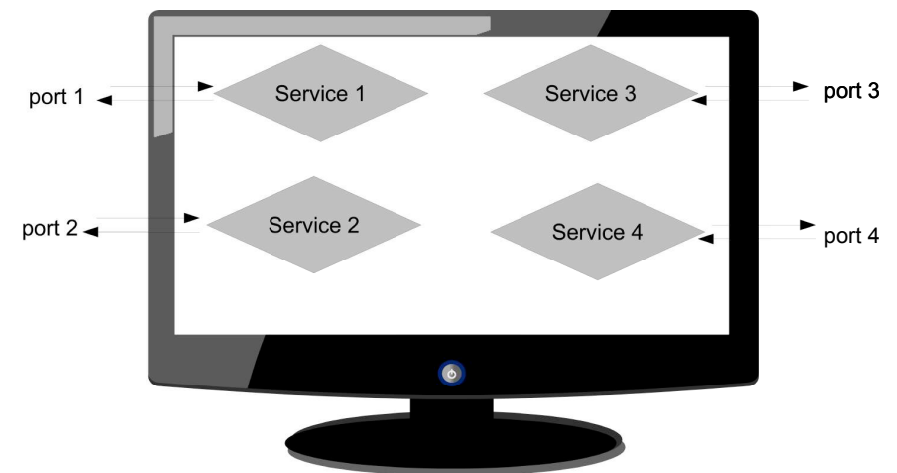
Résumé graphique



Les services en réseau

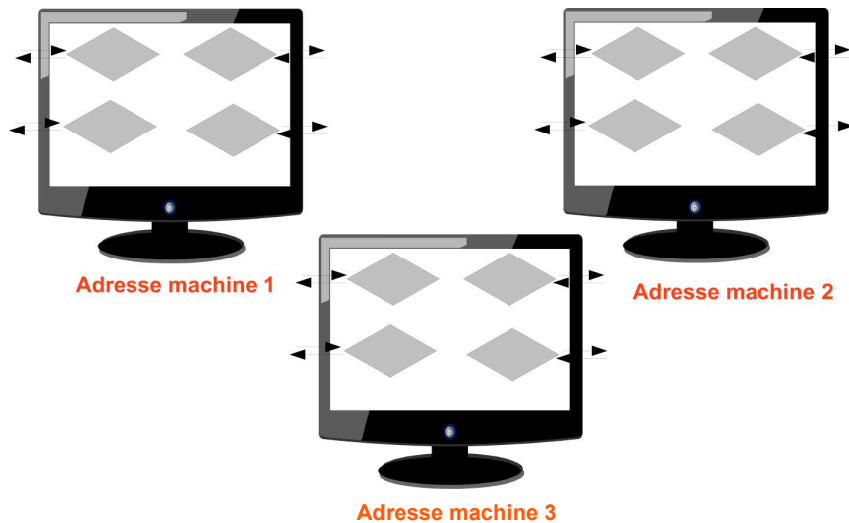
- La **couche transport** nous fournit des services de communication
 - Envoi de données
 - Réception de données
 - Connexion à une machine
- Pour communiquer ces machines doivent se connaître
 - Mécanisme de **nommage** des machines
- Comment identifier la machine
 - elle est identifiée par une **adresse**
- Comment trouver une application ou un **service** sur une machine
 - Chaque service est identifiée par un **port**

Sur une machine



UNE MACHINE AVEC UNE ADRESSE

Sur le réseau



PR - INTRODUCTION

17

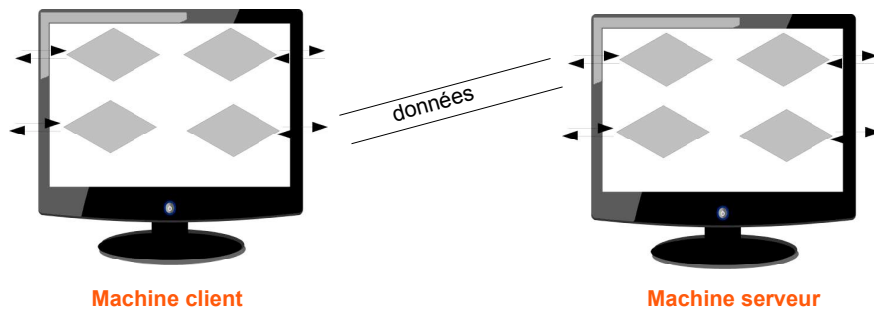
Informations pour communiquer

- Un couple (**adresse, port**) est un point de communication
- Pour communiquer il faut deux points de communication
 1. (adresse1, port1) d'un côté
 2. (adresse2, port2) de l'autre côté
- Exemple :
 - Quand on fait `http://www.google.com` dans le navigateur
 - Connexion à une des machines correspondant à `www.google.com`
 - Sur le port 80 qui correspond au service http
 - Dans ce cas, notre port de sortie n'est pas important ni notre adresse
- Souvent quand on fera une architecture client-serveur, notre port pour le client sera attribué à une valeur automatique

PR - INTRODUCTION

18

Communication entre deux machines



- Du côté du client, un port et une adresse
- Du côté du serveur
- Pour certaines applications, le 'programmeur' n'a pas besoin de connaître le port côté client
- Connaître le port côté serveur est nécessaire pour se connecter

PR - INTRODUCTION

19

Identification d'une machine

- Par un **nom internet** (pas forcément nécessaire)
 - Par exemple : `www.google.com`, `www.informatique.univ-paris-diderot.fr`
 - Une machine peut posséder plusieurs noms
 - Un nom peut correspondre à plusieurs machines
- Par une **adresse internet** (obligatoire pour toute machine sur le réseau)
 - en fait, adresse d'un dispositif réseau sur une machine
 - donc plusieurs adresses possibles pour une machine
 - une adresse par dispositif
 - plusieurs dispositifs par machine
 - Adresse (organisation structurelle) -> mieux pour les machines
 - Nom (organisation logique) -> mieux pour les humains

PR - INTRODUCTION

20

Question ?

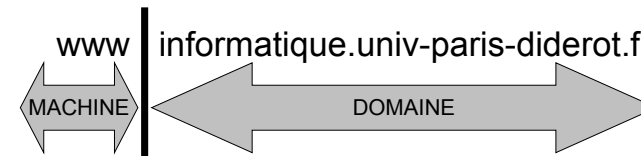


ah ouais !
et c'est quoi alors l'adresse
de www.google.com ?

- Il y en a plusieurs, par exemple : 173.194.40.144
- On verra :
 - comment trouver un nom à partir d'une adresse
 - comment trouver une adresse à partir d'un nom

Comment marchent les noms ?

- Un nom représente une structure hiérarchique
- Par exemple : www.informatique.univ-paris-diderot.fr
 - la machine **www**
 - dans le sous-domaine **informatique**
 - qui est lui-même dans le sous-domaine **univ-paris-diderot**
 - qui est dans le domaine **fr**
- Deux parties dans le nom :



À propos des domaines

- Le nom de domaine caractérise la hiérarchie des responsabilités
 - Par exemple : l'ufr d'informatique de l'université Paris Diderot située dans le domaine français
- Le domaine le plus à droite est appelé **domaine de premier niveau** (top level domain)
 - On distingue en gros deux types :
 - 1) Génériques (par exemple : .com, .edu, ...)
 - 2) Nationaux (par exemple : .fr, .tz, ...)

Analyse d'un nom

- Pour www.informatique.univ-paris-diderot.fr
 - **fr** est le domaine national attribué par l'ICANN (Internet Corporation for Assigned Names and Numbers) à la France avec délégation à l'AFNIC (Association Française pour le Nommage Internet en Coopération)
 - **univ-paris-diderot** est le sous-domaine attribué par l'AFNIC à l'université Paris Diderot avec délégation à la DSI de l'université
 - **informatique** est le sous-domaine attribué par la DSI à l'UFR d'informatique avec délégation au service informatique de l'UFR
 - **www** est le nom d'une des machines sous la responsabilité de l'UFR d'informatique

À propos des adresses

- Exemple d'adresse pour www.informatique.univ-paris-diderot.fr :
 - **194.254.199.98**
- Les adresses aussi sont structurées
- La structure des adresses est un reflet de la structure physique du réseau (tout du moins en théorie)
- Dans ce cours nous nous intéresserons pas à la structure des adresses
- MAIS nous utiliserons les adresses (et pas seulement les noms)
- **ATTENTION** : les adresses correspondent à des machines et la plupart du temps pas à des domaines

Les adresses IPv4

- Elles sont codées sur 4 octets (donc 32 bits)
- par exemple : 173.194.66.106
- Actuellement encore les plus utilisées
- Certaines adresses IP sont réservées à un usage particulier :
 - **127.0.0.1** : adresses pour l'hôte local **localhost** (en fait adresses comprises entre 127.0.0.1 et 127.255.255.255)
 - **192.168.0.0/16** : adresses privées
 - **224.0.0.4** : adresses pour la multidiffusion
 - **255.255.255.255** : adresse de diffusion
- Les adresses privées ne sont pas routées par Internet

Les adresses IPv6

- Elles sont codées sur 16 octets (donc 128 bits)
- Par exemple : **2a00:1450:400c:c02:0:0:93**
- On les écrit habituellement comme 8 groupes de deux octets
- Chaque octet est écrit en hexadécimal (valeur allant de 0 à F)
- On supprime parfois les 0 consécutifs par ::
- L'exemple précédent devient : **2a00:1450:400c:c02::93**
- Comme pour IPv4, certaines adresses IP sont réservées à un usage particulier
- Il n'existe pas de correspondance automatique entre adresses IPv4 et IPv6
- Les réseaux IPv4 et IPv6 cohabitent

Liens entre noms et adresse

- **Service de nom** : service permettant de faire la traduction d'un nom en une adresse
 - Il faut penser à un **annuaire**
 - Le système le plus répandu aujourd'hui est le **DNS (Domain Name Service)**
 - Il s'agit d'un **annuaire distribué** (il y a donc plusieurs serveurs DNS)
 - Le DNS contient aussi d'autres informations liées à un nom de domaine
 - Exemple d'informations fournies par le DNS :
 - L'adresse IPv4 (**A record**)
 - L'adresse IPv6 (**AAAA record**)
 - Les serveurs de courrier électronique pour le domaine (**MX record**)
 - Les serveurs DNS de ce domaine (**NS record**)

Interrogation des services de nom

- Différentes commandes LINUX :
 - **host**
 - **nslookup** (souvent considérée comme obsolète)
 - **dig**
- Commande pour connaître le nom de sa machine :
 - **hostname**
- Ces commandes peuvent être utilisées pour connaître l'adresse IP à partir d'un nom
- Parfois aussi pour connaître le nom à partir d'une adresse IP (peut être plus difficile à utiliser/comprendre)

Exemple d'utilisation de host



```
sangnier — bash — 80x24
Last login: Fri Jan 15 10:18:49 on ttys000
eduroam-prg-hf-1-7-157:~ sangnier$ bash
bash-3.2$ host www.google.com
www.google.com has address 216.58.204.100
www.google.com has IPv6 address 2a00:1450:4007:80a::2004
bash-3.2$
```

Exemple d'utilisation de host (2)



```
sangnier — bash — 80x24
[bash-3.2$ host www.lemonde.fr
www.lemonde.fr is an alias for s2.shared.global.fastly.net.
s2.shared.global.fastly.net has address 151.101.122.217
bash-3.2$
```

À propos de dig

- Obtenir la liste des serveur de messagerie de google.com
 - **dig MX google.com**
- Pour obtenir l'adresse d'une machine
 - **dig www.informatique.univ-paris-diderot.fr**
- **Attention** : aux requêtes que vous faites
 - google.com est un domaine et www.informatique.univ-paris-diderot.fr est une machine
- Pour obtenir le nom d'une machine à partir d'une adresse IP
 - **dig -x 194.254.61.138**
- Pour avoir une réponse lisible : utiliser l'option **+short**

Exemple d'utilisation de dig

```
bash-3.2$ dig www.informatique.univ-paris-diderot.fr
; <<> DiG 9.10.6 <<> www.informatique.univ-paris-diderot.fr
;; global options: +cmd
;; Got answer:
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 20365
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 3, ADDITIONAL: 6
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.informatique.univ-paris-diderot.fr.      IN A
;; ANSWER SECTION:
www.informatique.univ-paris-diderot.fr. 81969 IN CNAME trotinette.informatique.univ-paris-diderot.fr.
trotinette.informatique.univ-paris-diderot.fr. 81969 IN A 194.254.199.80
;; AUTHORITY SECTION:
informatique.univ-paris-diderot.fr. 11813 IN NS korolev.univ-paris7.fr.
informatique.univ-paris-diderot.fr. 11813 IN NS shiva.jussieu.fr.
informatique.univ-paris-diderot.fr. 11813 IN NS potemkin.univ-paris7.fr.
;; ADDITIONAL SECTION:
shiva.jussieu.fr. 104323 IN A 134.157.0.129
potemkin.univ-paris7.fr. 17827 IN A 194.254.61.141
potemkin.univ-paris7.fr. 20630 IN AAAA 2001:660:3301:8000::1:1
korolev.univ-paris7.fr. 20114 IN A 194.254.61.138
korolev.univ-paris7.fr. 20630 IN AAAA 2001:660:3301:8000::1:2
;; Query time: 8 msec
;; SERVER: 194.254.200.25#53(194.254.200.25)
;; WHEN: Mon Jan 18 08:59:54 CET 2021
;; MSG SIZE  revd: 297
```

33

Exemple d'utilisation de dig (2)

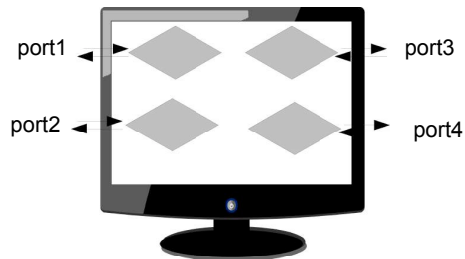
```
bash-3.2$ dig MX u-paris.fr
; <<> DiG 9.10.6 <<> MX u-paris.fr
;; global options: +cmd
;; Got answer:
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 7705
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 3, ADDITIONAL: 8
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;u-paris.fr.      IN      MX
;; ANSWER SECTION:
u-paris.fr.      86400  IN      MX      10 mx1.u-paris.fr.
u-paris.fr.      86400  IN      MX      30 korolev.univ-paris7.fr.
u-paris.fr.      86400  IN      MX      30 potemkin.univ-paris7.fr.
u-paris.fr.      86400  IN      MX      30 mataram.parisdescartes.fr.
;; AUTHORITY SECTION:
u-paris.fr.      19970  IN      NS      potemkin.univ-paris7.fr.
u-paris.fr.      19970  IN      NS      korolev.univ-paris7.fr.
u-paris.fr.      19970  IN      NS      delta.univ-paris5.fr.
```

PR - INTRODUCTION

34

Plusieurs ports sur une machine

- Plusieurs points de communication depuis une machine (**ports**)



Une machine avec une adresse

PR - INTRODUCTION

35

À propos des ports

- Les communication sur différents **ports** peuvent avoir lieu **simultanément**
- Toute communication passe par un port
- Un flux de communication est donc identifié par une adresse **ET** un port
- Les ports sont des numéros
- Il existe trois types de ports :
 - 1) Les ports reconnus (numéros allant de **0 à 1023**)
 - 2) Les ports réservés (numéros allant de **1024 à 49151**)
 - 3) Les ports libres (numéros allant de **49152 à 65535**)

PR - INTRODUCTION

36

Les ports reconnus

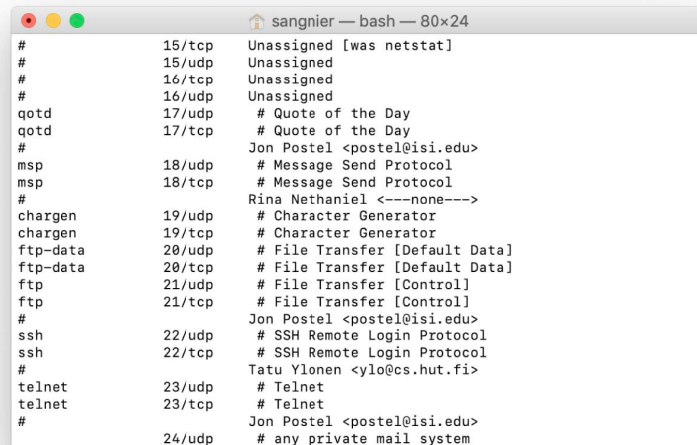
- En anglais : *Well-known ports*
- Ils sont utilisés par des services réseau d'usage général et commun :
- Par exemple :
 - **20** et **21** pour le service FTP
 - **25** pour le service SMTP
 - **80** pour le service HTTP
- Ainsi pour une établir une connexion avec un serveur web, on s'adresse au port 80 de la machine concernée (par exemple le port 80 de www.informatique.univ-paris-diderot.fr)
- Ainsi pour les services que vous développerez, évitez d'utiliser les numéros de ports entre 0 et 1023

Les ports réservés et libres

- Les ports **réservés** :
 - En anglais : *Registered port*
 - Certains correspondent à des services d'usage moins général
 - Par exemple : 3306 port utilisé par les bases de données MySQL
 - N'importe quelle application peut les utiliser
 - Pour les services que vous développerez, vous utiliserez ces ports
- Les ports **libres** :
 - En anglais : *Dynamic, private or ephemeral port*
 - Ports normalement utilisés pour des durées limitées ...

Le fichier /etc/services

- Sur les machines de type Linux donne les numéros de port utilisés et le nom du service correspondant



```
sangnier — bash — 80x24
#          15/tcp  Unassigned [was netstat]
#          15/udp  Unassigned
#          16/tcp  Unassigned
#          16/udp  Unassigned
qotd      17/udp  # Quote of the Day
qotd      17/tcp  # Quote of the Day
#          Jon Postel <postel@isi.edu>
msp       18/udp  # Message Send Protocol
msp       18/tcp  # Message Send Protocol
#          Rina Nathaniel <---none--->
chargen   19/udp  # Character Generator
chargen   19/tcp  # Character Generator
ftp-data  20/udp  # File Transfer [Default Data]
ftp-data  20/tcp  # File Transfer [Default Data]
ftp       21/udp  # File Transfer [Control]
ftp       21/tcp  # File Transfer [Control]
#          Jon Postel <postel@isi.edu>
ssh       22/udp  # SSH Remote Login Protocol
ssh       22/tcp  # SSH Remote Login Protocol
#          Tatu Ylonen <ylo@cs.hut.fi>
telnet    23/udp  # Telnet
telnet    23/tcp  # Telnet
#          Jon Postel <postel@isi.edu>
          24/udp  # any private mail system
```

Quelques commandes utiles

- La commande **ping**
 - Permet de tester si une machine est présente sur le réseau
 - Service d'écho réseau
- La commande envoie une requête *Echo* et attend une réponse *Echo Reply*
- L'envoi est répétée pour des fins statistiques
 - Pour déterminer le taux de perte des paquets et le délai moyen de réponse
- Certains pare-feux (en anglais *firewall*) bloquent les paquets de type Echo

Exemple d'utilisation de ping

```
sangnier — bash — 80x24
bash-3.2$ ping www.google.com
PING www.google.com (216.58.213.164): 56 data bytes
64 bytes from 216.58.213.164: icmp_seq=0 ttl=116 time=3.824 ms
64 bytes from 216.58.213.164: icmp_seq=1 ttl=116 time=9.100 ms
64 bytes from 216.58.213.164: icmp_seq=2 ttl=116 time=9.139 ms
64 bytes from 216.58.213.164: icmp_seq=3 ttl=116 time=5.389 ms
64 bytes from 216.58.213.164: icmp_seq=4 ttl=116 time=9.172 ms
64 bytes from 216.58.213.164: icmp_seq=5 ttl=116 time=9.148 ms
64 bytes from 216.58.213.164: icmp_seq=6 ttl=116 time=9.768 ms
^C
--- www.google.com ping statistics ---
7 packets transmitted, 7 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 3.824/7.934/9.768/2.156 ms
bash-3.2$
```

Quelques commandes utiles (2)

- La commande **ssh**
 - Permet de se connecter à une machine à distance
 - Votre machine doit autoriser ces connexions
 - Exemple d'utilisation :
 - **ssh login@nom_machine**
 - Permet de vous connecter à la machine nom_machine où votre login pour cette machine est login
 - Ensuite on vous demande (si la connexion est protégée) un mot de passe
 - Parfois la connexion ne peut se faire uniquement via clef RSA
- Par exemple pour l'ufr d'informatique :
 - Depuis l'extérieur vous pouvez vous connecter sur **nivose.informatique.univ-paris-diderot.fr** avec mot de passe et sur **lulu.informatique.univ-paris-diderot.fr** avec clef RSA et un jump via **lucy.informatique.univ-paris-diderot.fr**

Exemple d'utilisation de ssh

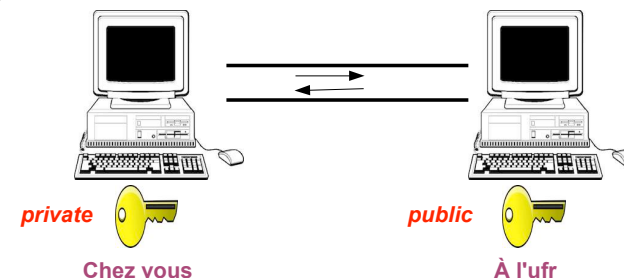
```
sangnier — ssh - ssh -t sangnier@lulu -J sangnier@lucy.informatique.univ-pari...
bash-3.2$ ssh -t sangnier@lulu -J sangnier@lucy.informatique.univ-paris-diderot.fr
Linux lulu 4.19.0-13-amd64 #1 SMP Debian 4.19.160-2 (2020-11-28) x86_64
Debian GNU/Linux 10

Bienvenue sur Lulu, le serveur linux généraliste de l'UFR d'informatique !
Voir http://www.informatique.univ-paris-diderot.fr/wiki/doku.php/wiki/linux

Last login: Mon Jan 18 09:05:30 2021 from fdc7:9dd5:2c66:be86:d294:66ff:fe06:d98e
sangnier@lulu:~$
```

Fonctionnement des clefs RSA

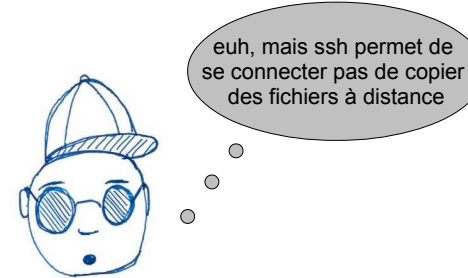
- Il s'agit d'une paire de clef (une clef étant un nombre entier) :
 - 1) Une clef publique, que l'on donne aux autres
 - 2) Une clef privée, que l'on garde
- Au moment de la connexion, le système vérifie si la paire clef publique-clef privée est la bonne
- Ainsi si vous donnez votre clef privée quelqu'un peut se faire passer pour vous



Connexion ssh vers l'ufr

- Générer votre couple (clef privée, clef publique)
 - **ssh-keygen -t rsa**
- Copier votre clef publique sur **nivose**
 - Pour rappel l'accès à nivose se fait via mot de passe
- Comme **nivose**, **lulu** et **lucy** partagent le même système de fichier, votre clef publique est aussi sur **lulu** et **lucy**
- À la fin, vous pouvez vous connecter sur **lulu**
- Pourquoi vouloir se connecter à **lulu** plutôt qu'à **nivose** :
 - Parce que **lulu** a la même configuration que les machines que vous utilisez en tp
 - Ce sera en particulier utile pour les tp à distance et les projets
- **Vous verrez en TP le détail de cette manipulation**

Copies de fichiers à distance



- La commande **scp** (qui marche comme **cp**) permet de copier des fichiers à distance
- Par exemple :
 - **scp test.txt sangnier@nivose.informatique.univ-paris-diderot.fr:~/Test/**
 - copie le fichier **test.txt** de mon répertoire courant dans le répertoire **~/Test/** de mon compte sur nivose
 - Pour l'opération inverse :
 - **scp sangnier@nivose.informatique.univ-paris-diderot.fr:~/Test/test.txt .**

Établir une connexion vers un service

- La commande **telnet** permet d'établir une liaison (**TCP**) **interactive** vers un service
- **Rappel** : pour ce connecter à un service il faut deux informations :
 - 1) Le nom de la machine ou l'adresse
 - 2) Le port du service
- Que veut dire **interactive** :
 - Ce que vous tapez au clavier est envoyé au service
 - Tous les messages que le service envoie sont affichés à l'écran
- Que veut dire **TCP** :
 - Liaison en mode connectée
 - Pensez au téléphone, on se connecte, on communique, puis on se déconnecte
 - Autre modes de liaisons **UDP** que l'on verra plus tard

Comment utiliser telnet

- Syntaxe de la commande
 - **telnet nom_machine port_service**
 - **nom_machine** est le nom de la machine où tourne le service
 - **port_service** est le port du service
- Pour certains services on peut donner le nom à la place du port
- Exemple :
 - **telnet lampe.informatique.univ-paris-diderot.fr dict**
 - Liaison au service **dict** qui tourne sur la machine **lampe.informatique.univ-paris-diderot.fr**
 - **telnet lampe.informatique.univ-paris-diderot.fr 2628**
 - Liaison au service écoutant sur le port **2628** de la machine **lampe.informatique.univ-paris-diderot.fr**
 - Le port 2628 correspond au service dict

Exemple d'utilisation

- Communication avec le service tcp dict (port 2628) sur lampe

```
sangnier@lulu:~$ telnet lampe dict
Trying fdc7:9dd5:2c66:be86:4849:43ff:fe49:79be...
Trying 192.168.70.237...
Connected to lampe.
Escape character is '^'.
220 lampe.informatique.univ-paris-diderot.fr dictd 1.12.1/rf on Linux 4.19.0-13-
amd64 <auth.mime> <2.26035.1610958358@lampe.informatique.univ-paris-diderot.fr>
[DEFINE * bonjour
150 1 definitions retrieved
151 "bonjour" fd-fra-eng "French-English FreeDict Dictionary ver. 0.4.1"
bonjour /b53ur/
1. good morning
2. good day
.
250 ok [d/m/c = 1/0/56; 0.000r 0.000u 0.000s]
```

Quelques remarques

- Tous les services ne tournent pas toujours
- Pour les exemples précédents, les services sont ouverts depuis le sein de l'ufr, donc pour reproduire les exemples :
 - Se connecter à lucien
 - Faire telnet depuis lucien
- Quand vous êtes sur une machine de l'ufr, il n'est pas nécessaire de répéter le suffixe **informatique.univ-paris-diderot.fr**, on peut faire :
 - **ssh nivose**
 - **telnet lampe dict**
 - **telnet lampe 2628**
- **telnet** permet aussi de tester si un service est actif

Exemple d'utilisation depuis l'extérieur

```
bash-3.2$ ssh -t sangnier@lulu -J sangnier@lucy.informatique.univ-paris-diderot.fr
Linux lulu 4.19.0-13-amd64 #1 SMP Debian 4.19.160-2 (2020-11-28) x86_64
Debian GNU/Linux 10

Bienvenue sur Lulu, le serveur linux généraliste de l'UFR d'informatique !
Voir http://www.informatique.univ-paris-diderot.fr/wiki/doku.php/wiki/linux

Last login: Mon Jan 18 09:10:58 2021 from fdc7:9dd5:2c66:be86:c294:66ff:fe06:d98e
sangnier@lulu:~$ telnet lampe dict
Trying fdc7:9dd5:2c66:be86:4849:43ff:fe49:79be...
Trying 192.168.70.237...
Connected to lampe.
Escape character is '^'.
220 lampe.informatique.univ-paris-diderot.fr dictd 1.12.1/rf on Linux 4.19.0-13-
amd64 <auth.mime> <4.26037.1610959052@lampe.informatique.univ-paris-diderot.fr>
```

Remarques

- Quand on communique avec un service en TCP
 - Il est important de savoir quelle forme a la communication
 - **Qui commence à communiquer ?**
 - **Comment ont- lieu les échanges ?**
 - **Quel est le format des messages ?**
 - **Comment prend fin la communication ?**
- Quand vous développerez vos services ou vos clients de service
 - Il faudra faire attention aux points ci-dessus
- N'hésitez pas à utiliser **telnet** pour tester le comportement de services

Normalisation

- Dans le monde d'Internet il existe des procédures de Normalisation
- Les **RFC** (Request For Comments)
 - Documents officiels recouvrant tous les aspects d'Internet
 - Voir : <http://www.ietf.org/rfc.html>
 - Ces documents ne concernent pas que la description des services, mais aussi des protocoles, des programmes, parfois des compte-rendus de réunion
- Par exemple :
 - **RFC 867** concernant le Daytime Protocol
 - **RFC 793** concernant le protocole TCP
 - **RFC 882** concernant les noms de domaine

RFC pour le service dict

[\[Docs\]](#) [\[txt\]](#) [\[pdf\]](#) [\[draft-rfced-inf...\]](#) [\[Tracker\]](#) [\[Diff1\]](#) [\[Diff2\]](#) [\[Errata\]](#)

INFORMATIONAL
Errata Exist

Network Working Group
Request for Comments: 2229
Category: Informational

R. Faith
U. North Carolina, Chapel Hill
B. Martin
Miranda Productions
October 1997

A Dictionary Server Protocol

Status of this Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (1997). All Rights Reserved.

Abstract

The Dictionary Server Protocol (DICT) is a TCP transaction based query/response protocol that allows a client to access dictionary definitions from a set of natural language dictionary databases.

Table of Contents

1.	Introduction	
1.1.	Requirements	
2.	Protocol Overview	
2.1.	Link Level	
2.2.	Lexical Tokens	
2.3.	Commands	
2.4.	Responses	
2.4.1.	Status Responses	
2.4.2.	General Status Responses	
2.4.3.	Text Responses	
3.	Command and Response Details	
3.1.	Initial Connection	

1-7/16 (6) (un)enrte/10/10/10/10/10