# 6: Communiquer en réseau

Martin Quinson



# Épisodes précédents

- Les cours de C sont terminés. Maintenant, on utilise le C mais on a dit tout ce qu'on voulait dire
- Sur la partie réseau, l'objectif est d'obtenir la culture générale nécessaire à n'importe quel informaticien qui se respecte.
  - C'est au programme de l'agreg d'info, donc c'est important (hihi)
  - On ne va pas apprendre à cabler la maison, ni même à configurer un réseau déjà cablé
  - On veut comprendre le monde pour savoir quels sont les problèmes intéressants à résoudre pour les futurs chercheurs que vous êtes.
- Un peu de bibliographie : les livres sont utiles pour accompagner ce cours
  - Le cours est dans un ordre qui, je l'espère, simplifie la compréhension. C'est plus pédagogique.

  - Les livres sont un ordre mieux rangé, plus logique. Mais ça demande de savoir ce qu'il y a dedans pour le lire (ou de lire plusieurs fois).
  - Il y a des liens sur la page web, et les livres suivants sont à la bibliothèque
- Sur le C:
  - Braquelaire: un bon livre pédagogique. Pour être honnête j'ai appris avec, y'a bien longtemps
  - Kernighan et Ritchie: un guide de référence, qui explique (mal) le C. A proscrire pour les apprenants. "man 3 printf" plus utile que le K&R, au final.
  - Nebra: un bon livre pédagogique, et moderne (cours en ligne associé sur OpenClassrooom)
- Sur le réseau:
  - Pujol: 1500 pages en bottom up. C'est la bible francophone, mais ça reste un peu indigeste
  - Kurose: un bon livre top-down. C'est ce que je suis pour ce cours
  - Bonaventure: un livre top-down très honnête, et intégralement disponible en ligne.
- Sur les deux à la fois, et sur le système (qui constitue la suite logique)
  - Computer Systems: A programmer perspective, par Bryant & O'Hallaron. Carnegie Mellon.

# I) Motivation pour l'étude des réseaux

- La première motivation pour étudier le réseau, c'est de permettre à ses programmes de communiquer.
- Mais le design de l'internet que nous avons est également intéressant, en tant que système bien pensé
  - Peu d'évolutions techniques, protocolaires, mais nb machines d'un millier en 1984 à des dizaines de milliards maintenant
  - Qu'est ce qui a rendu ce design si robuste ? Quelles étaient les alternatives ?
  - Répondre à ces questions sont l'autre motivation d'étudier le réseau, celle qui pousse à regarder dans la boite
- 3ieme motivation: L'informatique distribuée a longtemps été la dernière frontière (en date) de l'informatique
  - Nos Euclide s'appelle Vint Cerf et Leslie Lamport, et ils ne sont même pas à la retraite

- Il reste beaucoup de low hanging fruits en recherche, pour qui voudra bien s'en saisir

# II) Programme de l'agreg en réseau

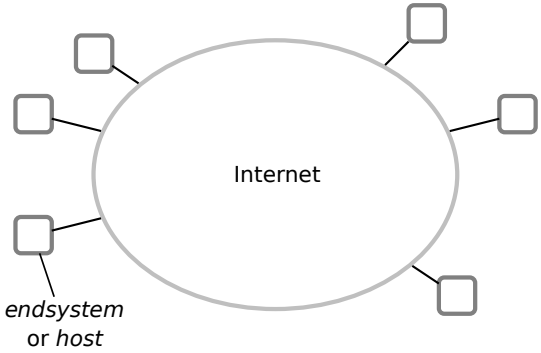
- Caractéristiques des réseaux et performances associées :
  - réseaux d'accès, réseaux de coeur ;
  - topologies de réseaux : point à point, à diffusion ;
  - performances : débit de transmission, délai, taux de perte.
- Modélisation en couches : TCP/IP, encapsulation.
- Transmission :
  - Adressage : adresses MAC, IPv4, IPv6 ;
  - Routage : principes ; routage à vecteur de distance (algorithme de Bellman-Ford), routage par état de liens (algorithme de Dijkstra) ;
  - Solutions de transport : principes ; TCP, UDP.
- Programmation réseau : API Sockets en Python et en C à l'aide d'un aide-mémoire fourni.
- Leçon 2022:
  - 21. Échanges de données et routage. Exemples.

# III) Un réseau de réseaux

- En matière d'internet, il y a les choses qu'on ne pouvait pas faire autrement, et les choses qui sont le résultat de choix
- On va commencer par voir les aspects nécessaires d'un réseau de réseaux

## III.1) Constitution du réseau

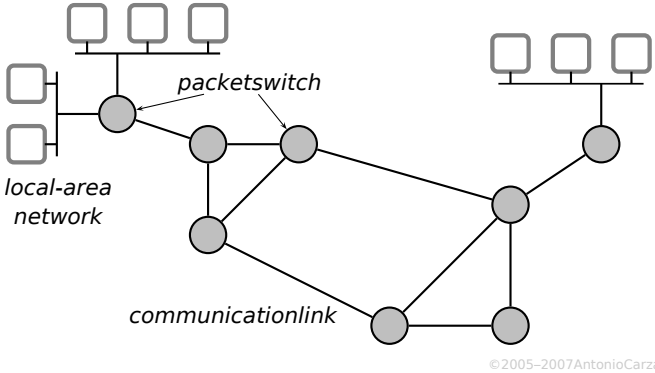
- Qu'est ce qu'Internet?
- Les éléments en bordure sont la raison d'être du réseau




### III.1.a) Terminaux

- Ce sont les choses connectés à Internet, en bordure, pour profiter du service de connexion
- Ordinateurs et maintenant smartphones. Mais aussi objets connectés (frigo)
- Il parait que l'avenir de l'humanité passe par l'IoT, qui demande des milliards de sondes et des centaines de millions d'actuateurs
  - Borne température et chaudière
  - Panneau et feux de circulation, Sondes sur la voie TGV, surveillance réseau élec

### III.1.b) Réseaux informatiques




- ne pas parler de contenu du réseau, car "contenu" désigne habituellement les données
- Équipements réseau: routeurs = switch (bien différents si on est au coeur ou en bordure)

- Liens entre tout ça
  - cuivre: permet de reprendre le réseau téléphone, qui est une paire cuivre (+: existance infra; -: perte en ligne)
    - A l'origine, réseau commuté (voix ou données, mais pas en même temps). ADSL= partage de fréquence grâce à des modems de chaque coté
    - Le dernier kilomètre est un réseau en étoile autour du DSLAM
  - fibre optique: infra nouvelle, mieux adaptée, mais couteuse à déployer.
    - Donc branchement en série sur une fibre, et multiplexage fréquence (une couleur par maison)
    - La SNCF et les sociétés d'autoroute sont des opérateurs d'internet: c'est eux qui posent les fibres opérées par d'autres
  - Sans fil: bcp techno.
    - wifi: 802.11; Bluetooth
    - GSM (techno téléphone). Avec les générations, portée baisse, débit monte. 5G pas raisonnable du tout. On a pas d'usage (sauf netflix 4k dans l'ascenseur) et ça coute une énergie de dingue.
    - Lora: débit assez faible, portée en kilomètres

### III.1.c) Topologies

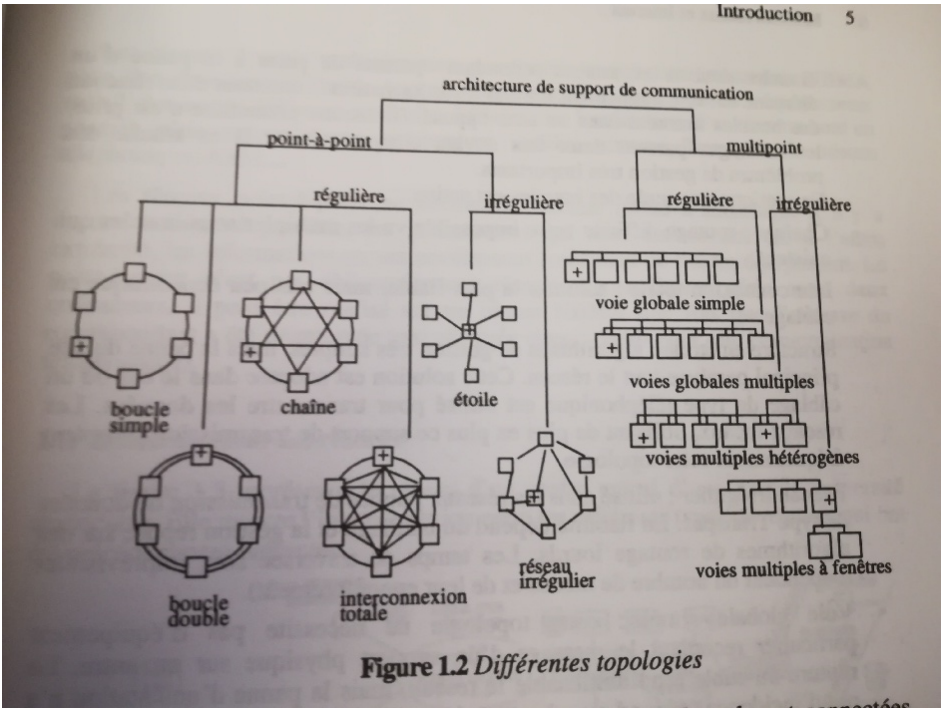



**Figure 1.2** *Différentes topologies*

- On pourrait ajouter les dragonfly et autres joyeusetés

## III.2) Partage de ressources

- Qu'est ce qui se passe quand on envoie un film au bout du monde ? Est-ce que ça voyage en un seul gros bloc ou en petits paquets? Est-ce qu'on réserve une place sur le réseau ?

### III.2.a) Commutation de circuit

- Mot compliqué pour parler de continuité cuivre: le réseau est réservé, alloué pour le flux
- C'est l'organisation du réseau téléphonique à l'origine (avant sa numérisation). X.25 pré-allouait des ressources lors de la connexion (circuit virtuel, plus bas)
- Ca permet des garanties fortes de performance, mais c'est un gaspillage de ressource

  - Pour permettre à N personnes de parler entre Rennes et Paris, faut N paires de cuivres séparées, mais les fils sont inutilisés quand y'a un blanc dans la conversation entre les interlocuteurs.
- Donc non, le vrai Internet n'est pas si dispendieux. On a la ligne que le temps où on l'utilise vraiment, sans pré-réservation
  - Les bandes passantes sur Transpac étaient garanties, et incroyablement faibles

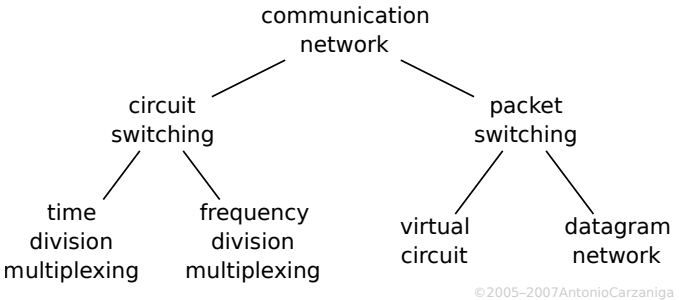
### III.2.b) Commutation de paquets

- Sur Internet, les données sont découpées en petit paquets envoyés les uns derrière les autres et traités séparément par le réseau
  - Pas de coût de connection, meilleur partage des liens
  - Mais pas de garantie de performance, surcoût de traitement de chaque paquet, entêtes plus long
- Faire autrement s'appelle "Store and Forward": Youtube stocke la vidéo sur le premier routeur après, qui l'envoi au routeur suivant.
  - Belle analogie du camion, mais ça marche pas comme ça, pour des raisons de performances sur lesquelles on revient bientôt

### III.2.c) Circuits virtuels

- L'idée est de combiner le meilleur des deux mondes.
- Etablissement d'un circuit virtuel = attribution d'un identifiant de communication
- Les paquets ensuite routés en utilisant cet identifiant, ce qui est plus efficace en temps de traitement et en taille d'entête

### III.2.d) Taxonomie des réseaux



## III.3) Protocoles

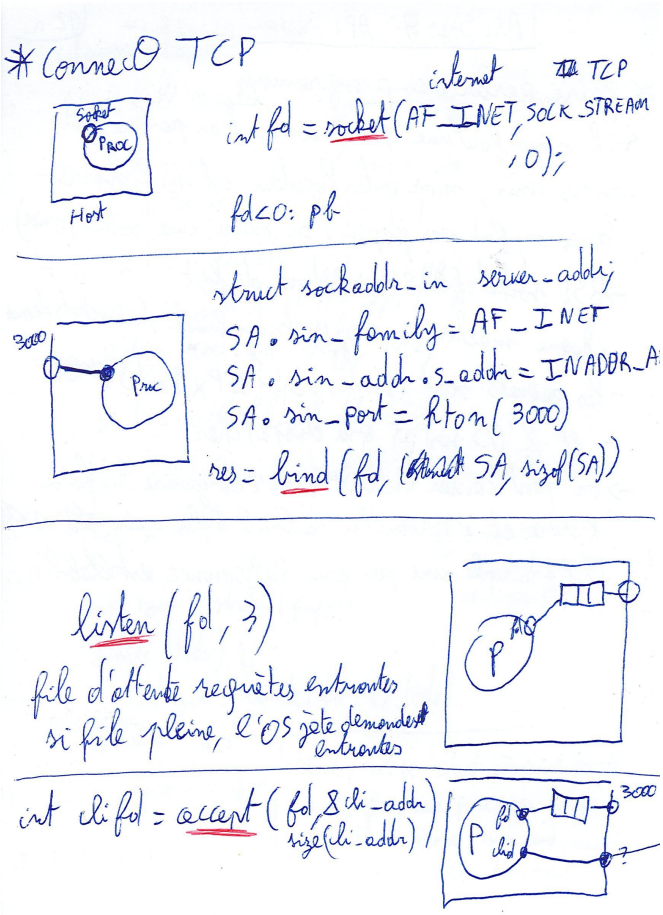
- Le mot "protocole" existait bien avant les ordis.
  - Le protocole, c'est qui a le droit de parler à la reine et quand. Une réponse fort peu protocolaire.
- C'est souvent culturel. Ne pas respecter le protocole $\rightsquigarrow$ incompréhensions et fait passer pour un malpoli
  - En France, quand on décroche le tél (en informatique: handshake), on a:
    - Répondant: Allo?
    - Appellant: Oui bonjour, c'est toto
    - Répondant: Ah bonjour toto, c'est bidule. Qu'est ce qui t'ammène?
  - En Allemagne, c'est un peu différent:
    - Répondant: Bidule.
    - Appellant: Oui bonjour Bidule, c'est Toto
    - Répondant: Ah bonjour Toto. Qu'est ce qui t'ammène?
- En informatique, les protocoles doivent définir le format et l'ordre des message
  - Ça se spécifie souvent avec plusieurs machines à états synchronisées
    - Ca donne des recherches amusantes, où les collègues de Nancy faisaient de l'apprentissage sur la machine à état du téléphone IP, puis attaquaient aux bordures inhabituelles du protocole. Ils ont découvert une série de message étranges à envoyer au téléphone pour décrocher et écouter, sans que le téléphone ne signale quoi que ce soit en facade... Ca fait un "bug" bien pratique et un peu complexe pour que ce soit dû au hasard.
  - traitement des erreurs de transmission, d'accès concurrent au média, time-out/retransmission
  - Un protocole est une sorte de programme distribué
- Autre exemple: le protocole du contrôle aérien
  - Extrêmement rigoureux.
  - Partage du temps de parole (la tour donne la parole), pas d'ambiguité et ACK, gestion des problèmes de transmission et retransmission
  - Multi-partite et sans connexion
- Sur Internet, il faut en plus se mettre d'accord sur leur specs
  - Internet Engineering Task Force (IETF): organisation d'orga et individus qui discutent et spécifient. C'est très américain.
  - Request For Comments: type de doc qui spécifie un protocole en particulier

# IV) Faire communiquer des programmes

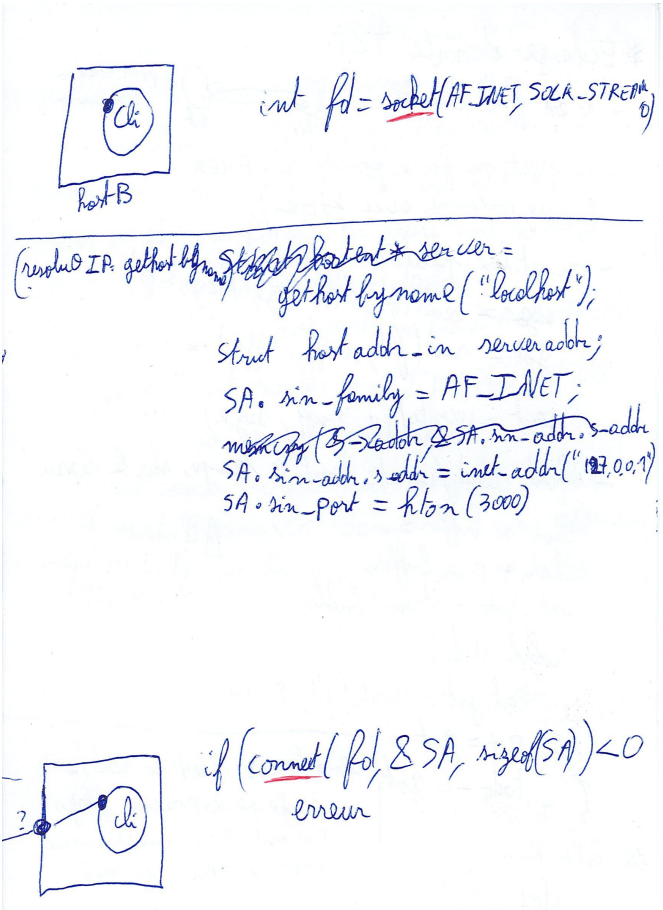
- Pour faire communiquer des programmes informatiques, il nous faut une API: comment on parle à qui.
- L'interface standard de facto, de l'internet que nous avons, c'est **BSD socket**
  - Cette interface date de 1983 (BSD 4.2), c'est à dire qu'UNIX avait déjà 13 ans quand ça a été introduit.
    - Certaines parties mal intégrées (fnctl), et UNIX n'est pas le design le mieux adapté au réseau.
    - Plan 9, par les mêmes, était mieux pensé mais ça n'a jamais pris car UNIX reste good enough dans son design, et ses implems sont très solides.
  - *Socket*, en anglais, ça veut pas dire chaussette, mais prise femelle. Une prise male, c'est *plug*
- En TP et partiel, on utilise le protocole TCP, où il faut se connecter pour avoir un flux bi-directionnel, mais qui permet d'utiliser le réseau sans réfléchir ensuite
  - Il y a une autre façon d'utiliser l'Internet moderne (protocole UDP) mais pas de mise en pratique cette année
- Sous UNIX, mon interlocuteur est un fichier. Ou plutôt, je le manipule comme un fichier. C'est à dire que `fprintf` marche, que mon interlocuteur soit:
  - un clavier/écran (par défaut)
  - un vrai fichier (redirection d'E/S dans le shell)
  - un autre programme en local (tube)
  - un autre programme à distance (socket réseau)
  - Si je représente mon processus comme un ovale, une socket est un rond en surface sur lequel un autre processus peut venir se connecter
- Sur internet, la destination des paquets d'information est donné par un couple {IP, port}
  - L'IP désigne l'ordinateur destination
  - le port désigne une sorte de boite-aux lettres à laquelle les paquets d'information doivent être délivrés. L'idée est qu'il y a un programme donné derrière une boite aux lettres donnée.
  - Si je représente mon ordi comme un carré contenant des ovales-processus, un port est un rond en surface du carré:
    - ouvert vers l'extérieur pour permettre à des processus distants de se connecter dessus

∗ connecté dedans à une socket d'un processus donné

## IV.1) Connexion TCP




1. Le serveur


Cli
host B
int fd = socket(AF_INET, SOCK_STREAM, 0)
(resolve IP: gethostbyname)
gethostbyname("localhost");
struct hostaddr_in serveraddr;
SA.sin_family = AF_INET;
SA.sin_addr.s_addr
SA.sin_addr.s_addr = inet_addr("127.0.0.1")
SA.sin_port = htons(3000)
Cli
if (connect(fd, &SA, sizeof(SA)) < 0
erreur

2. Le client

## IV.2) Echange de données TCP

* Echange donnée TCP
  - C'est full duplex : 

pour chaque connexion
  - on peut pas faire fprintf sans FILE* (ça se contourne avec fdopen)
  - on utilise plutôt
    recv(fd, buff, size, flags=0)
    size = send(fd, buff, size, flags=0)
    got = read(fd, buff, size)
    sent = write(fd, buff, size)
  - short read/write tout le temps sur le réseau

```
char * buffer = "....";
char * p = buffer;
int todo = strlen(buffer);
while (todo) {
    int got = write(fd, p, todo)
    p += got
    todo -= got
}
```

⚠ il manque le coca client/serveur, et un peu P2P ?

* à la fin
  close(fd)

send/recv c'est co read/write mais je peux preciser flags
Dont wait, More sur send/recv
PEER, WAITALL sur recv