



INTERACTION DISTRIBUÉE

Ph. Truillet

<http://www.irit.fr/~Philippe.Truillet>

Septembre 2021

UN MONDE EN RÉSEAU

■ avant-hier

- *internet* connecte tous les ordinateurs (ou presque)

■ hier

- les terminaux interactifs sont omniprésents (*notion d'informatique embarquée*) : smartphones, tablettes, ...

■ aujourd'hui et demain

- chaque objet physique est connecté (*notion informatique diffuse ou ubiquitaire – pervasive computing, « internet of things » IoT, smart cities, ...*)



UN PEU D'HISTOIRE

En **1982**, « *The internet Coke Machine* », une machine à boissons de Carnegie Mellon University est connectée à des terminaux via une liaison série (Michael Kazar) : elle fait un rapport de son stock et donne sa température sur le réseau.

En **1992**, la « *Trojan Room Coffee Pot* » à Cambridge University (fermée le 22/08/2001)



Le terme “***Internet of Things***” est utilisé la première fois par Kevin Ashton en **1999**.

RÉSEAUX

TCP



UDP



Quand on parle de réseaux, on l'associe souvent à un protocole réseau généralisé depuis les années 90 → la pile de protocoles **TCP/IP**

Néanmoins, on a :

- des **couches physiques *hétérogènes***
- des **protocoles** et des **architectures hétérogènes**

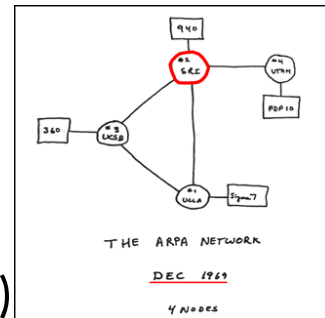
Et bien évidemment, de nombreux protocoles non-IP (LoRa, SigFox, Zigbee, ...) !

INTERNET ?

The augmentation of Doug Engelbart - <https://youtu.be/7ZtISeGyCY>
<https://www.dougelbart.org>



Un des pionniers : **Douglas Engelbart** (1925-2013)



- A l'intuition d'internet¹ dès les années 50 (son laboratoire (**SRI**) participe à la première liaison en 1969 avec l'UCLA)
- Démontre la première vidéoconférence (1968) « **The Mother of All demos** »²
- Invente la souris (1968)



¹Augmented Human intellect:

<http://www.dougelbart.org/pubs/augment-3906.html>

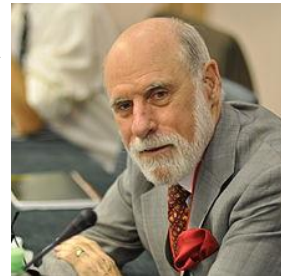
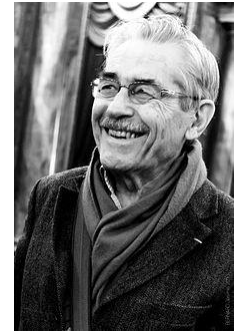
²The Mother of All demos,

<http://www.dougelbart.org/firsts/dougs-1968-demo.html>

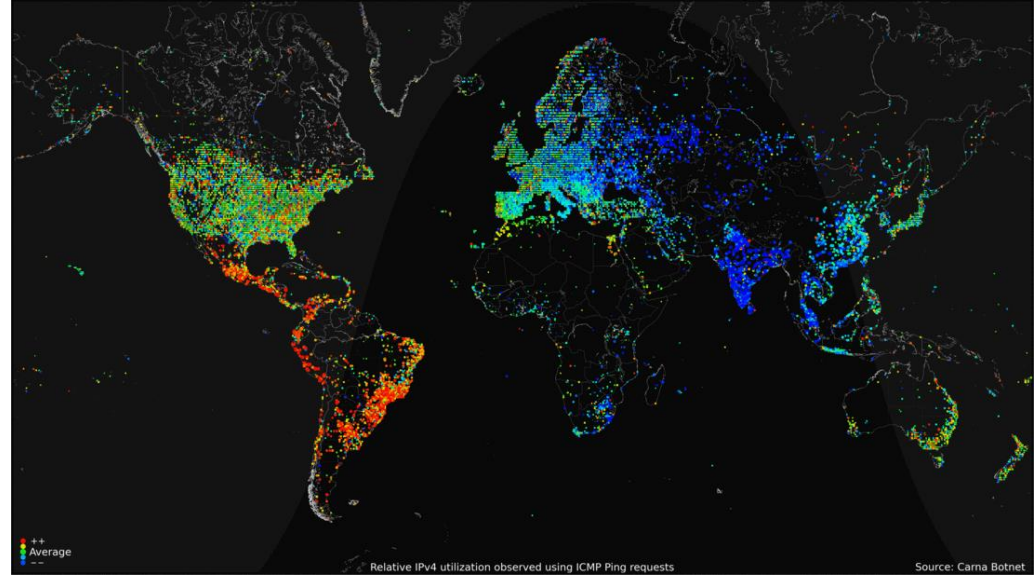
INTERNET ?

D'autres pionniers :

- **Louis Pouzin** (1931- ?), « *inventeur* » de la commutation de paquets (IRIA, Projet Cyclades 1971-1978)
- **Vint Cerf** (1943- ?), « co-inventeur » du protocole TCP/IP, « *Chief Internet Evangelist* » chez Google depuis 2005
- ...

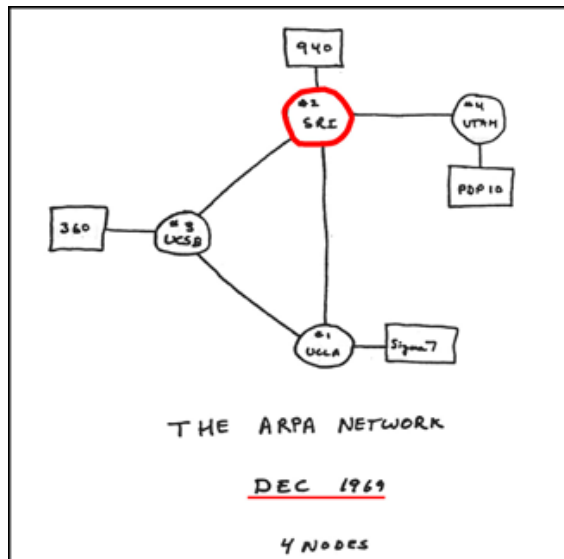


INTERNET ?



<http://motherboard.vice.com/blog/this-is-most-detailed-picture-internet-ever>

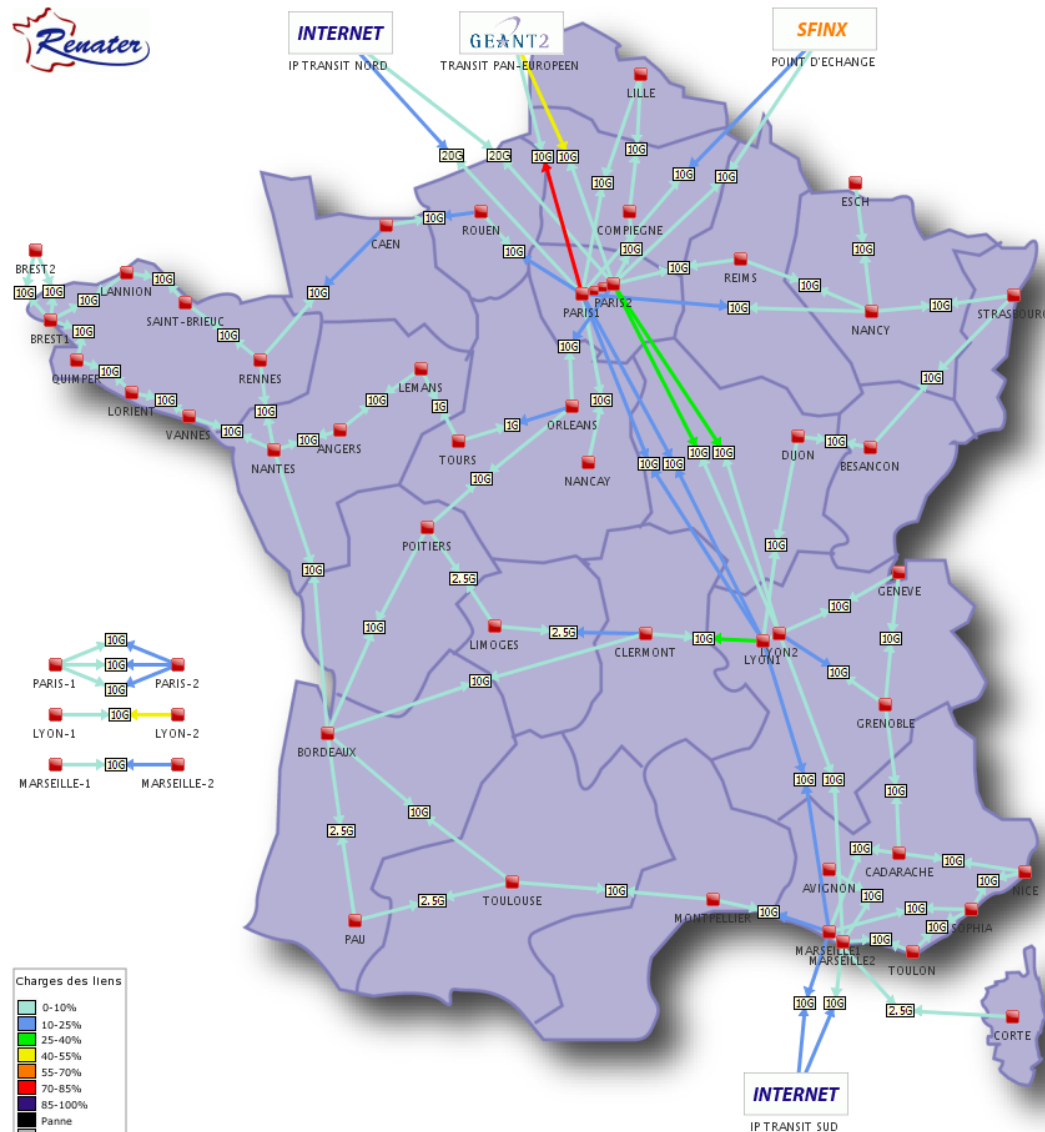
Internet est ... un réseau de réseaux (hétérogènes)



29 OCT 69	2100	LOADED OP. PROGRAM CSK	
		FOR BEN BARKER	
		BBV	
22:30		Talked to SRI	CSK
		Host to Host	
		Left op. program CSK	
		running after sending	
		a host dead message	
		to imp.	

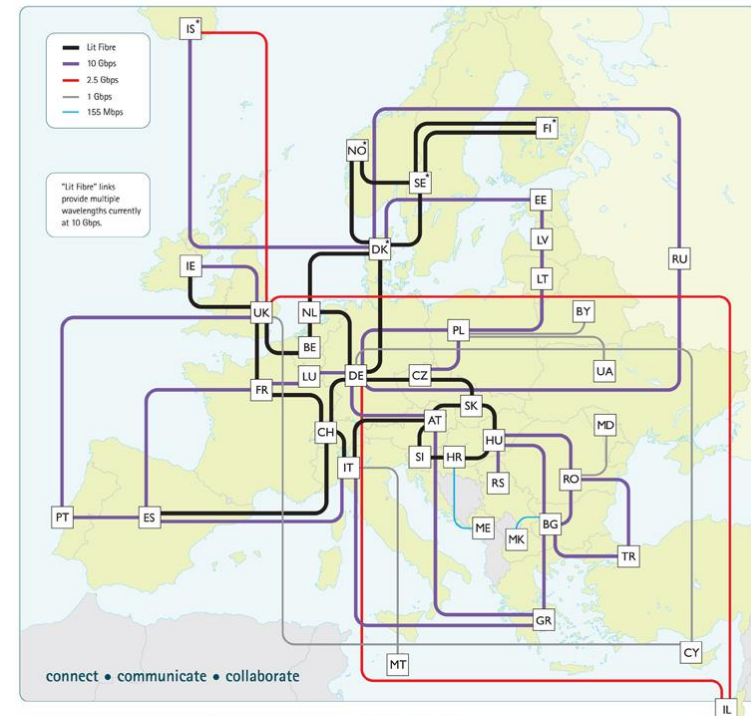
http://www.computerhistory.org/internet_history

INTERNET ?



GÉANT the pan-European research and education network

Transforming the way users collaborate



Backbone topology as at March 2012. GÉANT is operated by DANTE on behalf of Europe's NRENs.

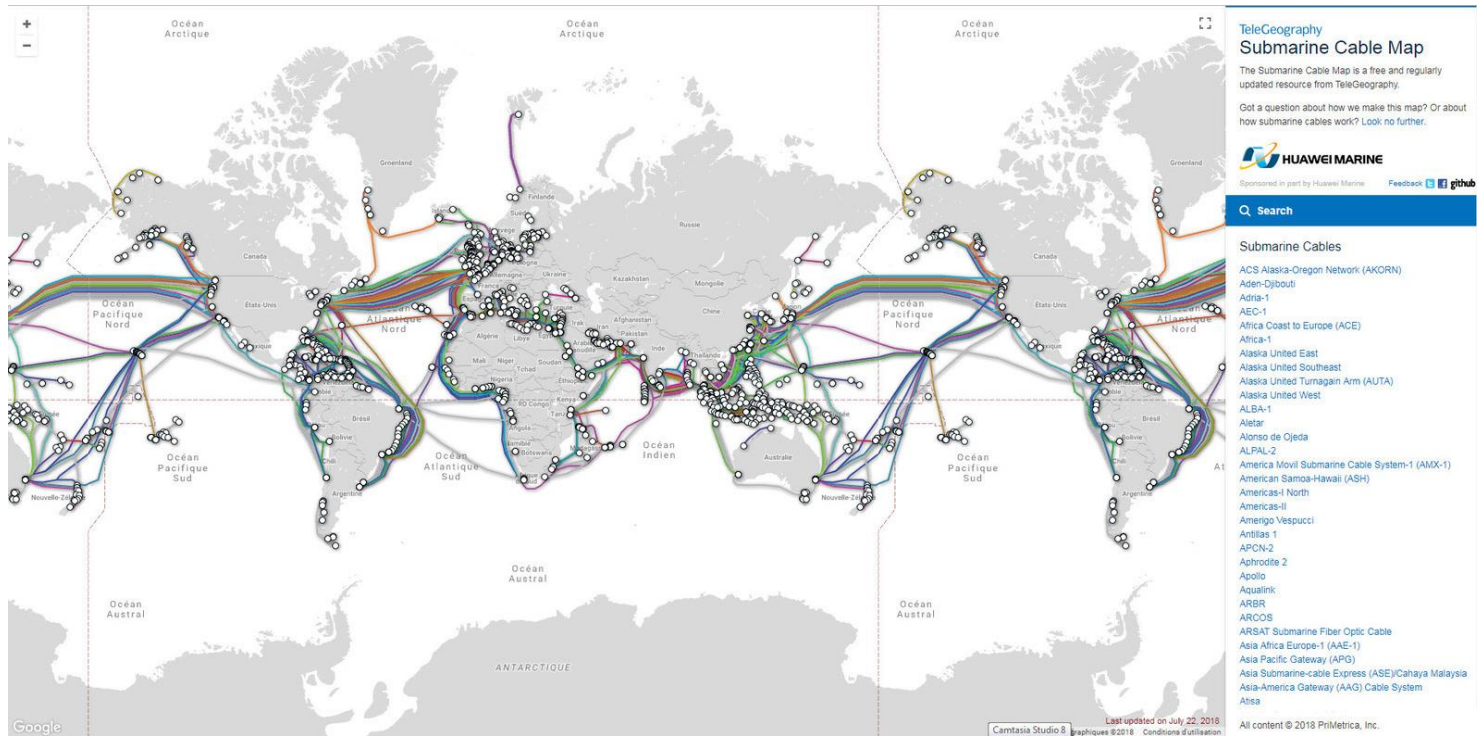


GÉANT is co-funded by the European Commission within its 7th R&D Framework Programme.

This document has been produced with the financial assistance of the European Union. The contents of this document are the sole responsibility of DANTE and can under no circumstances be regarded as reflecting the position of the European Union.



INTERNET ?



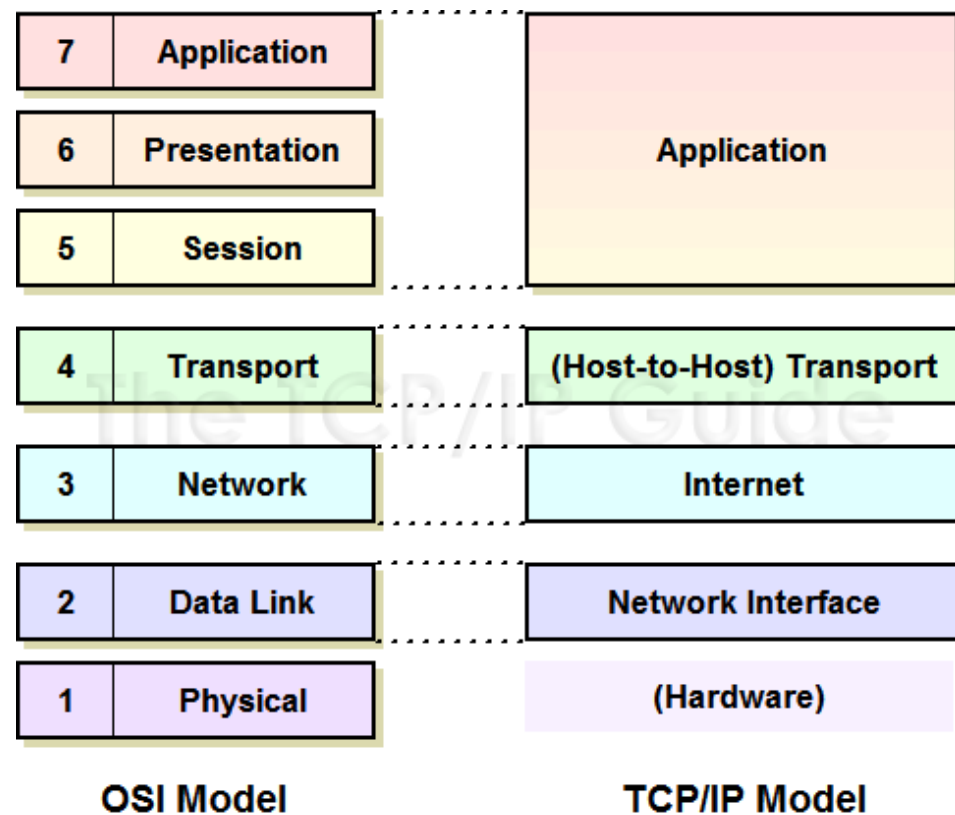
<https://www.submarinecablemap.com>

Câble le plus long : 39 000 km (SEA ME WE 3)

Au total, 800 000 km (20 fois le tour de la terre) et 99% des communications mondiales

INTERNET

internet ... est un ensemble de protocoles (1969/1972)



<http://www.tcpipguide.com/free>

IP “Internet Protocol” (couche réseau)

échange les données entre ordinateurs hôtes

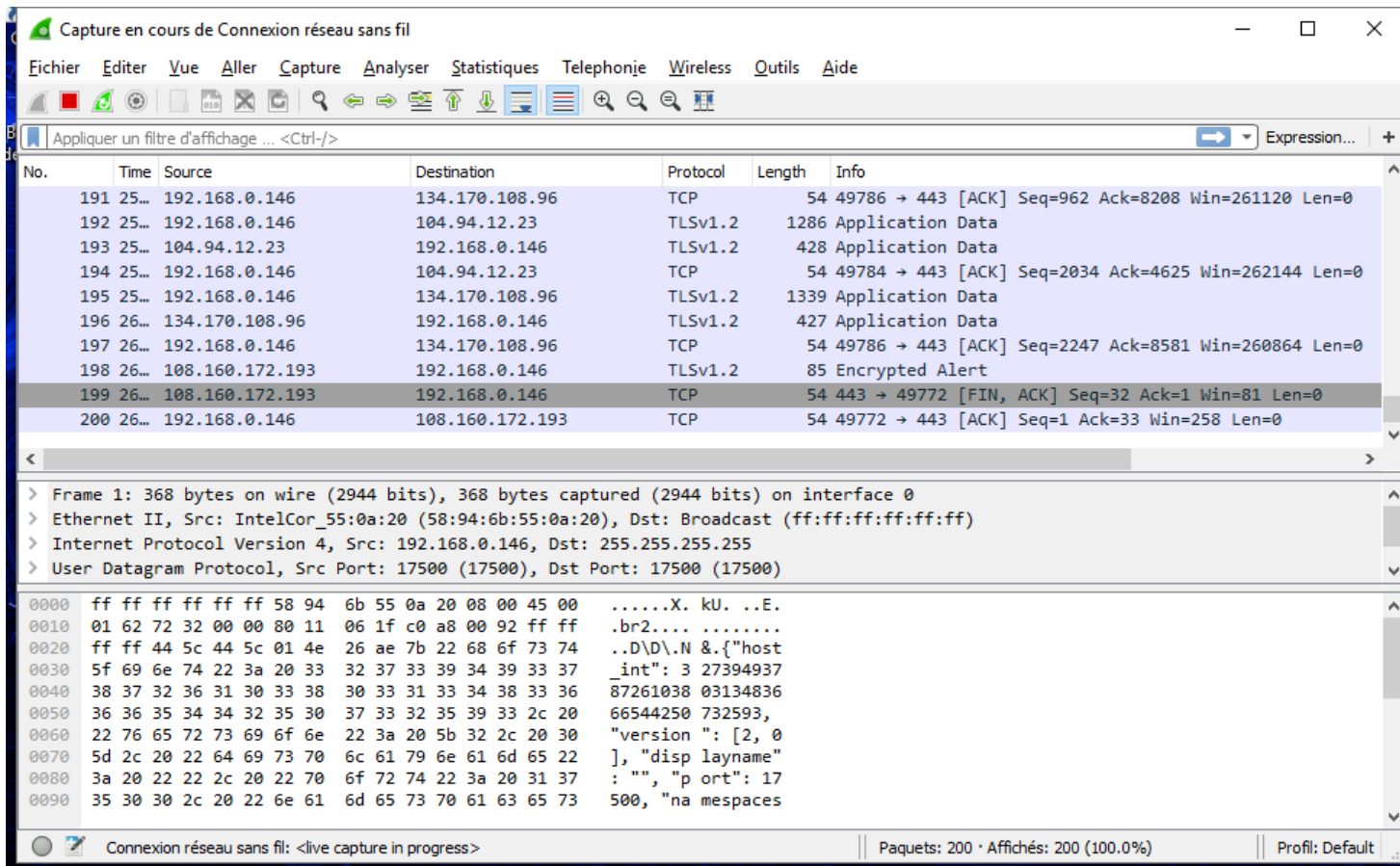
TCP “Transport Control Protocol” (couche transport)

échange les données entre les applications



INTERNET ... ?

Ce fonctionnement « en couches » est intéressant mais pose certains de nombreux problèmes de sécurité.



The image shows a Wireshark network traffic capture window. The title bar reads "Capture en cours de Connexion réseau sans fil". The menu bar includes "Fichier", "Editer", "Vue", "Aller", "Capture", "Analyser", "Statistiques", "Telephonie", "Wireless", "Outils", and "Aide". The toolbar contains various icons for file operations, capture control, and analysis. The filter bar shows "Appliquer un filtre d'affichage ... <Ctrl-/>". The packet list pane displays a table of captured packets:

No.	Time	Source	Destination	Protocol	Length	Info
191	25...	192.168.0.146	134.170.108.96	TCP	54	49786 → 443 [ACK] Seq=962 Ack=8208 Win=261120 Len=0
192	25...	192.168.0.146	104.94.12.23	TLSv1.2	1286	Application Data
193	25...	104.94.12.23	192.168.0.146	TLSv1.2	428	Application Data
194	25...	192.168.0.146	104.94.12.23	TCP	54	49784 → 443 [ACK] Seq=2034 Ack=4625 Win=262144 Len=0
195	25...	192.168.0.146	134.170.108.96	TLSv1.2	1339	Application Data
196	26...	134.170.108.96	192.168.0.146	TLSv1.2	427	Application Data
197	26...	192.168.0.146	134.170.108.96	TCP	54	49786 → 443 [ACK] Seq=2247 Ack=8581 Win=260864 Len=0
198	26...	108.160.172.193	192.168.0.146	TLSv1.2	85	Encrypted Alert
199	26...	108.160.172.193	192.168.0.146	TCP	54	443 → 49772 [FIN, ACK] Seq=32 Ack=1 Win=81 Len=0
200	26...	192.168.0.146	108.160.172.193	TCP	54	49772 → 443 [ACK] Seq=1 Ack=33 Win=258 Len=0

The packet details pane shows the selected packet (No. 199) with the following structure:

- > Frame 1: 368 bytes on wire (2944 bits), 368 bytes captured (2944 bits) on interface 0
- > Ethernet II, Src: IntelCor_55:0a:20 (58:94:6b:55:0a:20), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
- > Internet Protocol Version 4, Src: 192.168.0.146, Dst: 255.255.255.255
- > User Datagram Protocol, Src Port: 17500 (17500), Dst Port: 17500 (17500)

The packet bytes pane shows the raw data in hexadecimal and ASCII:

```
0000  ff ff ff ff ff ff 58 94 6b 55 0a 20 08 00 45 00  ....X. kU. ..E.
0010  01 62 72 32 00 00 80 11 06 1f c0 a8 00 92 ff ff  .br2.... ....
0020  ff ff 44 5c 44 5c 01 4e 26 ae 7b 22 68 6f 73 74  ..D\D\.N &{"host
0030  5f 69 6e 74 22 3a 20 33 32 37 33 39 34 39 33 37  _int": 3 27394937
0040  38 37 32 36 31 30 33 38 30 33 31 33 34 38 33 36  87261038 03134836
0050  36 36 35 34 34 32 35 30 37 33 32 35 39 33 2c 20  66544250 732593,
0060  22 76 65 72 73 69 6f 6e 22 3a 20 5b 32 2c 20 30  "version ": [2, 0
0070  5d 2c 20 22 64 69 73 70 6c 61 79 6e 61 6d 65 22  ], "displayname"
0080  3a 20 22 22 2c 20 22 70 6f 72 74 22 3a 20 31 37  : "", "port": 17
0090  35 30 30 2c 20 22 6e 61 6d 65 73 70 61 63 65 73  500, "namespaces
```

The status bar at the bottom indicates "Connexion réseau sans fil: <live capture in progress>", "Paquets: 200 · Affichés: 200 (100.0%)", and "Profil: Default".

<http://www.wireshark.org>

INTERNET ... ?

La plupart des protocoles utilisés (et pas que dans ce cours !) sont forgés sur **ces couches** et vont permettre de faire une abstraction plus ou moins importante du réseau !

Thread

iamkirkbater and jkjustjoshing



iamkirkbater 🇺🇸 Aug 23rd, 2017 at 9:37 AM
in #www

Do you want to hear a joke about TCP/IP?



7 replies



jkjustjoshing 5 months ago
Yes, I'd like to hear a joke about TCP/IP



iamkirkbater 🇺🇸 5 months ago
Are you ready to hear the joke about TCP/IP?



jkjustjoshing 5 months ago
I am ready to hear the joke about TCP/IP



iamkirkbater 🇺🇸 5 months ago
Here is a joke about TCP/IP.



iamkirkbater 🇺🇸 5 months ago
Did you receive the joke about TCP/IP?



jkjustjoshing 5 months ago
I have received the joke about TCP/IP.



iamkirkbater 🇺🇸 5 months ago
Excellent. You have received the joke about TCP/IP. Goodbye.

SYSTÈMES RÉPARTIS

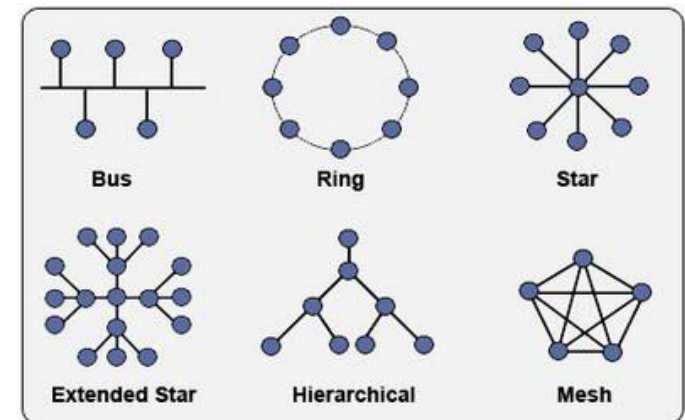
Une définition très large : un système réparti est un système informatique dans lequel les ressources ne sont pas centralisées

Les ressources ont un sens très large :

- stockage (disques, bases de données)
- puissance de calcul
- mais aussi les **utilisateurs**

SYSTÈMES RÉPARTIS

- **But** : permettre à des utilisateurs de manipuler (calculer) leurs données (stocker) sans contrainte sur les localisations respectives des éléments du système
- La plupart du temps, cela correspond à la généralisation et l'amélioration du schéma client/serveur :
 - serveurs multiples (équilibrage de charge, redondance)
 - systèmes multi-couches (*tiers*)
 - *peer to peer*
- Dans notre cas, **réparti** \approx **distribué**



POURQUOI DES SYSTÈMES RÉPARTIS ?

- pour des aspects **économiques**
 - réalisation de systèmes à haute disponibilité
 - partage de ressources (programmes, données, services)
 - réalisation de systèmes à grande capacité d'évolution
- **pour une adaptation** de la structure d'un système à celle des applications
- pour un besoin **d'intégration**
- pour un besoin de **communication** et de **partage** d'information

LES DOMAINES D'APPLICATION

ingénierie

- coopération d'équipes pour la conception d'un produit
- production coopérative de documents
- partage cohérent d'information

gestion intégrée des informations d'une entreprise

- intégration de l'existant

contrôle et organisation d'activités en temps réel

centres de documentation, bibliothèques

- recherche, navigation, visualisation multimédia

systèmes d'aide à la formation (collecticiels, ...)

LES BESOINS DES APPLICATIONS

ouverture

- interopérabilité, portabilité, fédération ; réutilisation de l'existant

coopération, coordination, partage

- vision commune cohérente d'informations partagées (globalement, par groupes)
- interaction en temps réel, support multimédia

transparence

- **accès** (mobilité des usagers avec préservation de l'environnement)
- **localisation** (de l'information, des services, ...)

LES BESOINS DES APPLICATIONS

qualité de service

- disponibilité, délais, coûts, qualité de perception, ... avec niveau garanti

sécurité

- authentification, intégrité, confidentialité, ...

évolutivité, administrabilité

- reconfiguration, gestion dynamique de services

QUELQUES DIFFICULTÉS LIÉES À LA RÉPARTITION

- l'**accès** aux données distantes
- le **maintien** du service (gestion des pannes)
- l'**encodage/décodage des données** (*marshalling, unmarshalling*)
- les **accès concurrents** au(x) service(s)
- la **gestion des droits** à distance
- ...

NOTIONS FONDAMENTALES : CLIENT/SERVEUR ET PROTOCOLES

QUELQUES NOTIONS

ARCHITECTURE CLIENT/SERVEUR

Serveur : celui qui offre un service (doit l'offrir de manière permanente) → « *daemon* »

- Accepte les requêtes, les traite et renvoie une réponse
- Ex : `httpd`, `ftpd`, `telnetd`, ...

Client : celui qui utilise le service

- Envoie une requête et reçoit la réponse



QUELQUES NOTIONS

ARCHITECTURE CLIENT/SERVEUR

Architecture C/S → description du comportement coopératif
entre
le serveur et les clients

→ fonctionnement général des services internet

→ s'appuie sur des protocoles entre les processus communicants

QUELQUES NOTIONS PROTOCOLE

On nomme **protocole** les **conventions** qui facilitent une communication sans faire directement partie du sujet de la communication elle-même

- Exemples de protocole : FTP, HTTP, ...

Sont compilées dans les RFC (**R**equest **f**or **C**omments).
Peu de RFC sont des normes mais toutes les normes internet sont enregistrées en tant que RFC

- FTP → RFC 114 (Avril 1971), HTTP → RFC 1945 (mai 1996), RFC 2616 (juin 1999), ...
- Les RFC du 1^{er} avril :
 - **IPoAC** (RFC 1149, RFC 6214 pour IPv6) ☺
 - **HTJP** – HyperText Jeopardy Protocol (RFC 8565)
 - **TCP Option to Denote Packet Mood** (RFC 5841)
 - ...

QUELQUES NOTIONS PROTOCOLE

HTTP 1.1

localisation : `http_URL = "http:" "//" host [":" port] [abs_path ["?" query]]`

requête (client) : `Method SP Request-URI SP HTTP-Version CRLF`

Ex : `GET /index.php HTTP/1.1 <entrée>`

réponse (serveur) : `Status-Line`

`*((general-header | response-header | entity-header) CRLF)`

`CRLF [corps de message]`

Ex : `HTTP/1.1 400 Bad Request
Date: Tue, 13 Sep 2011 14:27:22 GMT
Server: Apache
Content-Length: 226
Content-Type: text/html; charset=iso-8859-1
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">`

QUELQUES NOTIONS SERVICES

Une machine offre usuellement plusieurs services accessibles par un numéro de port (de 1 à 65535)

- On doit connaître ce numéro pour accéder au service → notion de ports « *bien connus* »

Ex :

echo	: port 7
daytime	: port 13
ftp	: port 21
http	: port 80

/etc/services sous Linux

EXEMPLE : LE DNS

DNS : Domain Name Server (1984)

Permet de trouver l'adresse IP correspondant au nom de domaine

exemple : `sri.univ-tlse3.fr` →
`195.220.43.54`

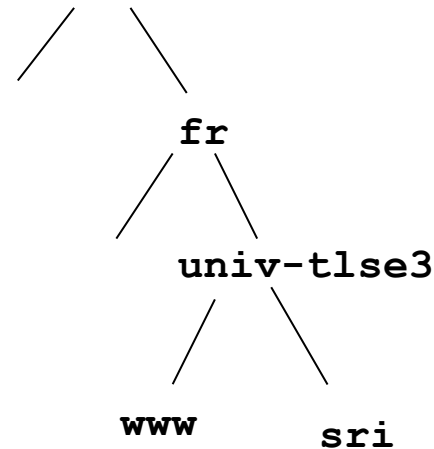
DNS : base de donnée répartie, système hiérarchique

EXEMPLE : DNS

un serveur DNS gère un domaine

le gestionnaire peut déléguer la gestion d'un sous-domaine à une autre gestionnaire

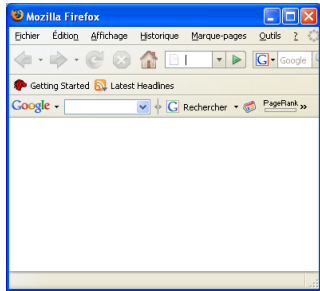
TLD - Top Level Domain



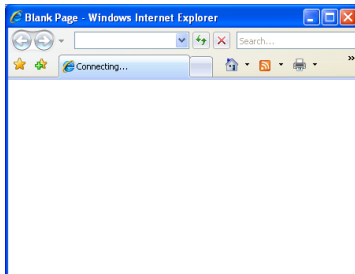
EXEMPLE : HTTP

HTTP : HyperText Transfer Protocol

Protocole client-serveur très simple. Une requête → une réponse



clients

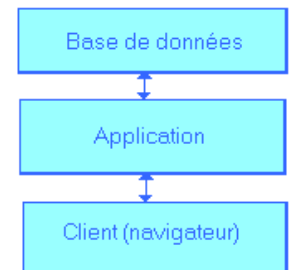


serveur

EXEMPLE : ARCHITECTURE MULTI-COUCHES

Classiquement, 3 couches (extension du modèle Client/Serveur)

- **Présentation** : (Interface) – visible et interactive
- **Application** : (partie fonctionnelle) - couche métier, logique applicative
- **Stockage**

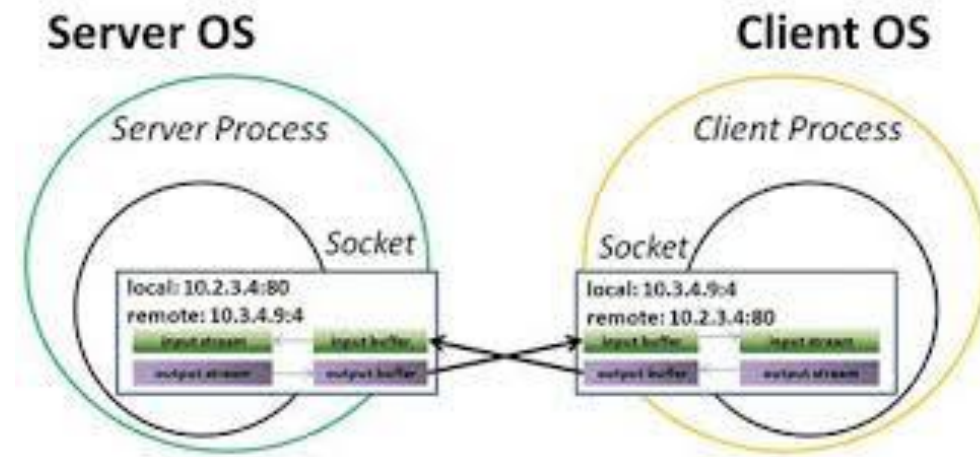


EXEMPLE : P2P

Principe du « pair-à-pair » : chacun est à la fois client et serveur

- p2p « *pur* » : connexions directes entre participants
- p2p « *pratique* » : des serveurs existent entre les clients permettant l'existence d'un service d'annuaire (qui est connecté, qui propose quoi et où ?, ...)

COMMUNICATION



Plusieurs niveaux d'abstraction :

- **Bas-niveau : la socket (échanges de messages)**

...

```
// connexion
```

```
laSocket = new Socket(machine, port);
```

```
// Mise en forme du flux de sortie
```

```
fluxSortieSocket = new
```

```
PrintStream(laSocket.getOutputStream());
```

```
fluxSortieSocket.println("GET /cam_picture HTTP/1.0\r\n");
```

COMMUNICATION

- Appel à des procédures distantes (RPC, SOAP, ...)

...

```
Resultat = calcul_addition_1(&parametre, clnt);
```

```
if (resultat == (reponse *) NULL) {  
    clnt_perror (clnt, "call failed");  
    clnt_destroy (clnt);  
    exit(EXIT_FAILURE);  
}
```

COMMUNICATION

■ Appel à des méthodes distantes (RMI, CORBA, ...)

```
...
ORB orb = ORB.init(argv, null);
// get the root naming context
org.omg.CORBA.Object objRef =
    orb.resolve_initial_references("NameService");

NamingContext ncRef =
    NamingContextHelper.narrow(objRef);
NameComponent nc = new NameComponent("Horloge", " ");

// Resolve the object reference in naming
NameComponent path[] = {nc};

// La classe Helper fournit une fonction "narrow«
// pour obtenir un objet sur lequel on peut invoquer
// les methodes
Horloge horloge = HorlogeHelper.narrow(ncRef.resolve(path));
```

COMMUNICATION

- **Déclenchement d'événements distants (OSGi, ...)**

```
public class AmpouleListener implements ServiceListener {  
    public void serviceChanged(ServiceEvent event) {  
        // ServiceReference ref = e.getServiceReference();  
  
        String[] objectClass = (String[])  
        event.getServiceReference().getProperty("objectClass");  
  
        ...  
    }  
}
```

COMMUNICATION



▪ Publisher/Subscriber (MQTT, ROS, ...)

```
cli = paho.Client(client_id="PiZero2")
cli.on_message = on_message
cli.on_publish = on_publish
cli.username_pw_set("try", password="try")
cli.connect("broker.shiftr.io", 1883, 60);
cli.subscribe("/data",0);
while cli.loop()==0:
    cli.publish('/Bureau/temperature',
                '{0:0.2f}'.format(sensor.read_temperature()))
    cli.publish('/Bureau/pressure',
                '{0:0.2f}'.format(sensor.read_pressure()))
    time.sleep(60) # delay for 1 minute
```

CE QUE L'ON VA FAIRE

- des travaux ... **pratiques (6 séances) !**
- programmation en **java** et **python** (la plupart du temps) mais vous pouvez choisir **un autre langage** si vous le souhaitez

Objectifs :

- acquérir des notions solides sur les abstractions réseau les plus utilisés.
- tester les avantages et inconvénients de chaque approche

(PETITS) OUTILS

PuTTY, KiTTY :

- <http://www.9bis.net/kitty/?zone=fr>



httpie

- <https://httpie.io>



- <http://curl.haxx.se>

Terminal :

- <https://sites.google.com/site/terminalbpp>

TRAVAUX PRATIQUES (6+1 SÉANCES)

■ Echange de messages

- *bus à événements, prototypage*
 - *ivy – 2 h → utile pour le TP IHM*
 - *(ZeroMQ / ingescape)*
- programmation sockets – 2 h
- MQTT – 2 h (+ raspberry)



■ Appel de méthodes et d'objets à distance

- RMI/CORBA – 2-4 h
- JSON / API REST – 2 h



EVALUATION

UNIQUEMENT sur un mini-projet technique (à 2 ou 3) !

- **2013/2014** : projet « myCloud »
- **2014/2015** : projet « pimp my P2P »
- **2015/2016** : projet « Lord of the sensors »
- **2016/2017** : projet « Lord of the sensors (le retour) » : one place to rule them all
- **2017/2018** : projet « CMS – Check My Sensors »
- **2018/2019** : projet « MySQL - My Smart Quiet Locality »
- **2019/2020** : projet « TISSU - Tiny Smart Sensors Unified »
- **2020/2021** : projet « SEW – Sensors EveryWhere »

Sujet donné mi octobre, à livrer mi-janvier (mini-rapport + code + démonstration)

SUIVI

Github



<https://github.com/truillet/upssitech/tree/master/SRI/3A/ID>

Serveur discord



<https://discord.gg/JUrFdw3u5A>