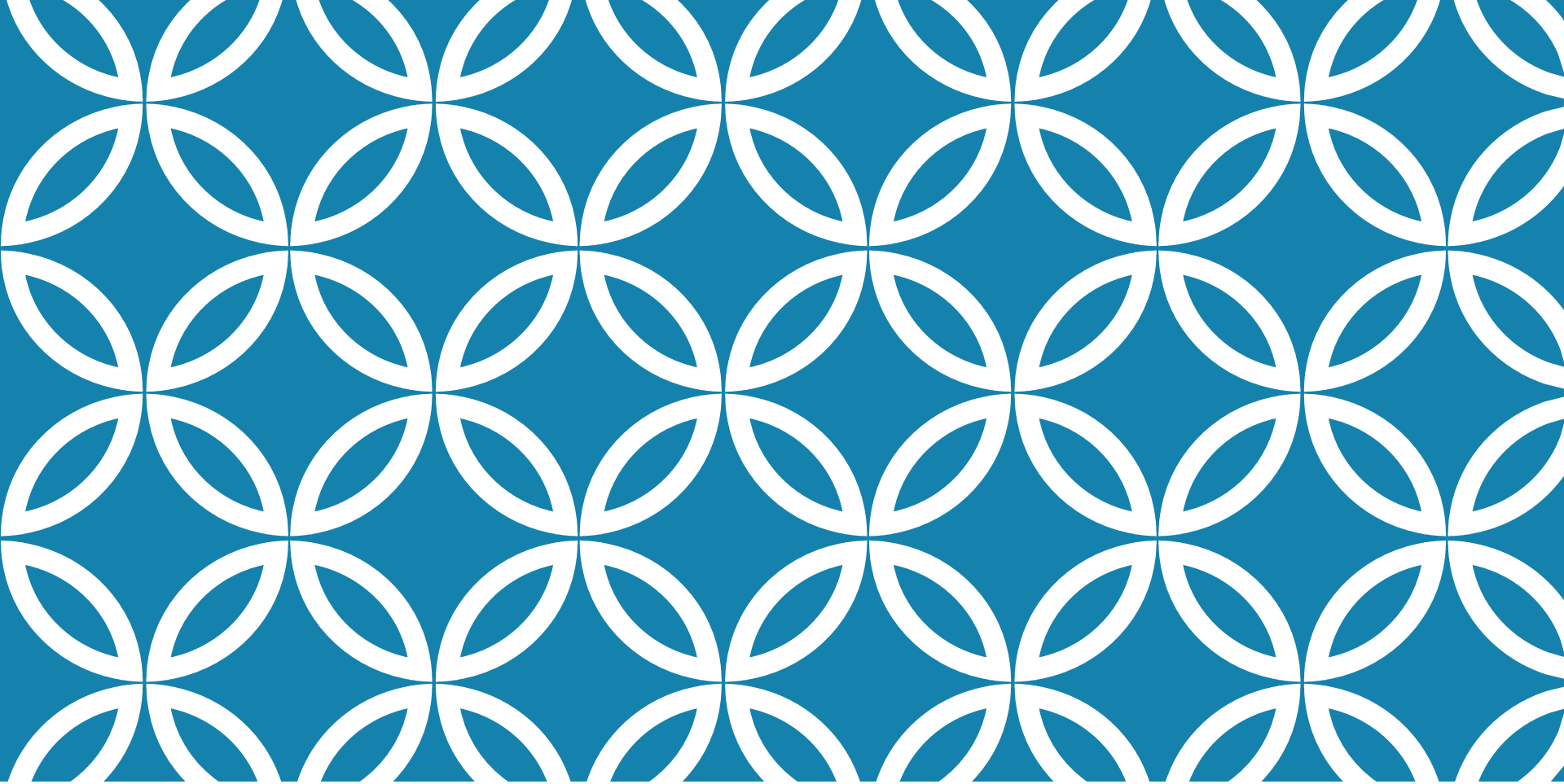


# INTERACTION DISTRIBUÉE

Ph. Truillet
http://www.irit.fr/~Philippe.Truillet
Septembre 2018

UPSSITECH
ÉCOLE D'INGÉNIEURS

# LE MONDE EN RÉSEAU

- **avant-hier**
  - *internet* connecte tous les ordinateurs (ou presque)
- **hier**
  - les terminaux sont omniprésents (*notion d'informatique embarquée*) : smartphones, tablettes, …
- **aujourd'hui et demain**
  - chaque objet physique est connecté (notion *informatique diffuse – pervasive computing, « internet of things » IoT, smart cities, …*)

# UN PEU D'HISTOIRE

En **1982,** une machine à boissons de Carnegie Mellon University est connectée à internet : elle fait un rapport de son stock et donne sa température.

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

# RÉSEAUX

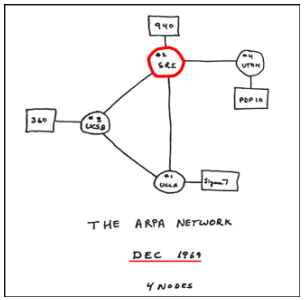
Une constante est *presque* respectée grâce un protocole réseau commun → pile **TCP/IP**

Néanmoins, on a :

- des **couches physiques *hétérogènes***
- des **protocoles** et des **architectures <u>hétérogènes</u>**

# INTERNET ?

Un des pionniers : Douglas Engelbart (1925-2013)



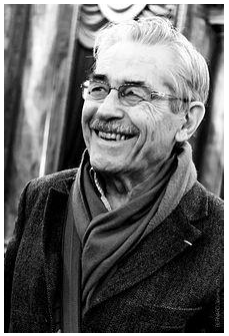

- A l'intuition d'internet[1] dès les années 50 (son laboratoire (SRI) participe à la première liaison en 1969 avec l'UCLA)
- Démontre la première vidéoconférence (1968) « The Mother of All demos » [2]
- Invente la souris (1968)

[1]Augmented Human intellect: http://www.dougengelbart.org/pubs/augment-3906.html
[2]The Mother of All demos, http://www.dougengelbart.org/firsts/dougs-1968-demo.html

# INTERNET ?

D'autres pionniers

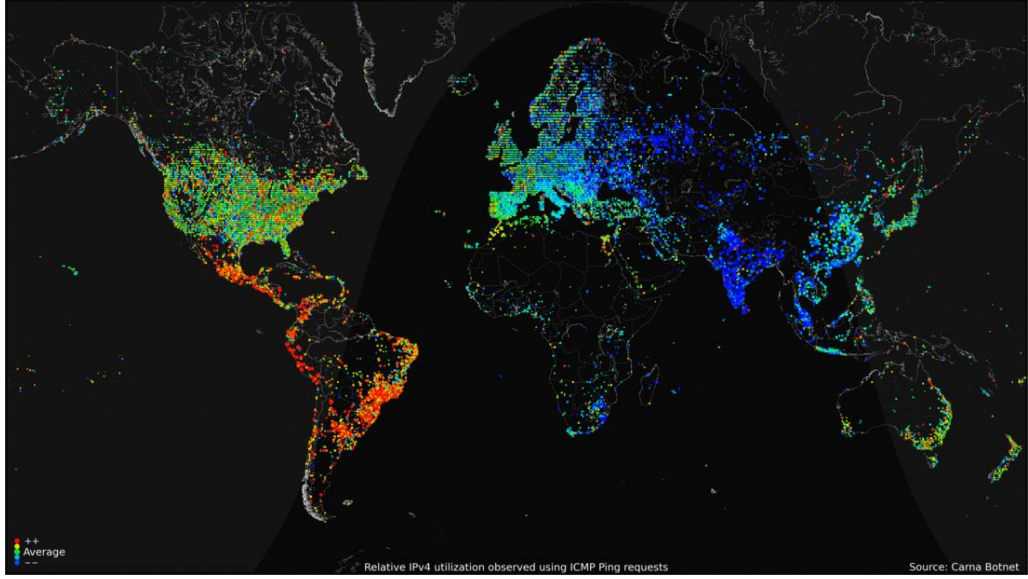
- Louis Pouzin (1931- ?), « inventeur » de la commutation de paquets (IRIA, Projet Cyclades 1971-1978)



- Vint Cerf (1943- ?), « co-inventeur » du protocole TCP/IP, « Chief Internet Evangelist » chez Google depuis 2005

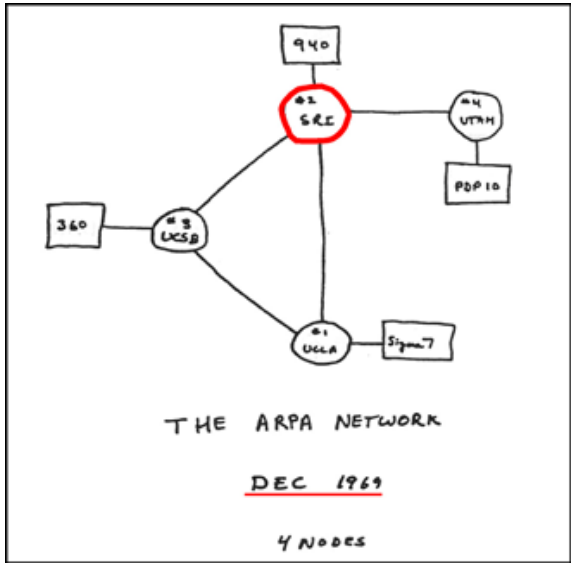
- …

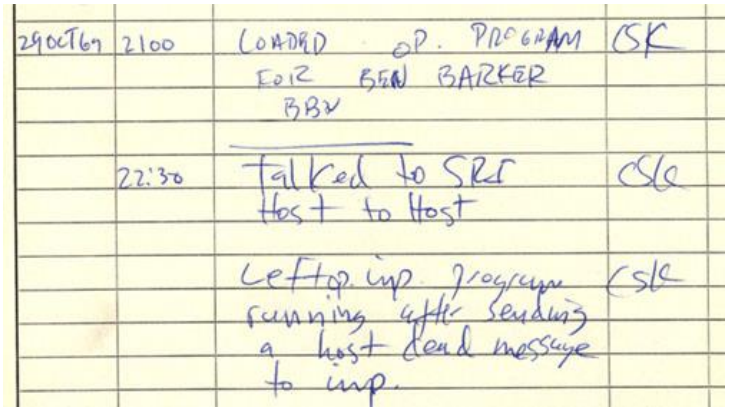
# INTERNET ?




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

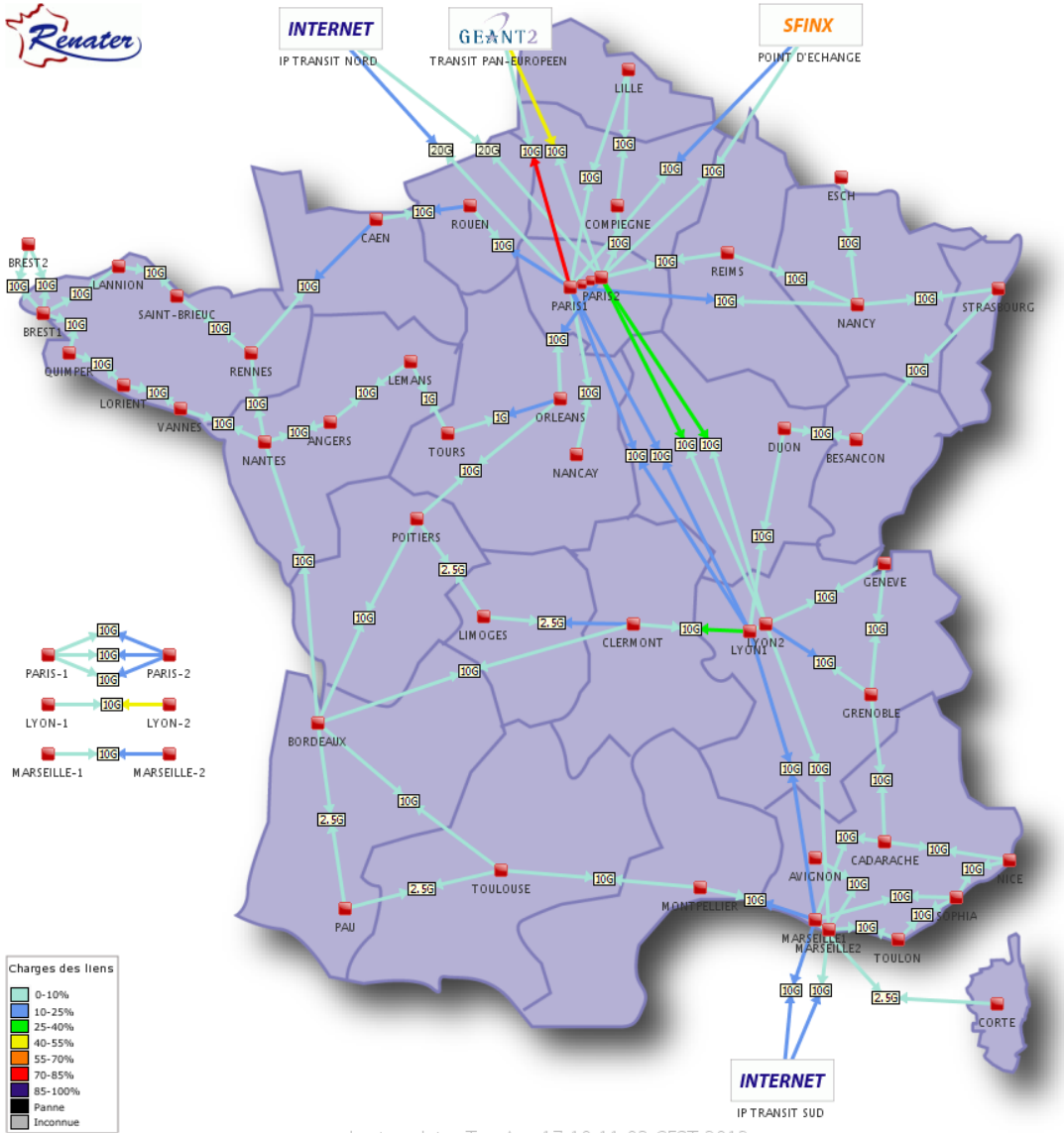
**Internet est … un réseau de réseaux (hétérogènes)**




http://www.computerhistory.org/internet_history

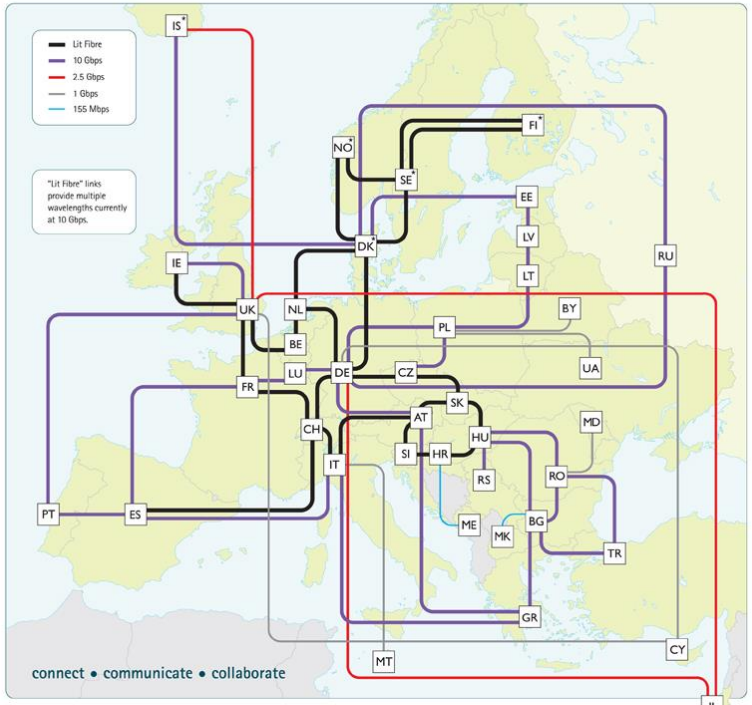
# INTERNET ?




Last update: Tue Apr 17 10:11:03 CEST 2012

## 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.


DANTE
www.dante.net

*Connections between these countries are part of NORDUnet (the Nordic regional network)
**Associate NRENs

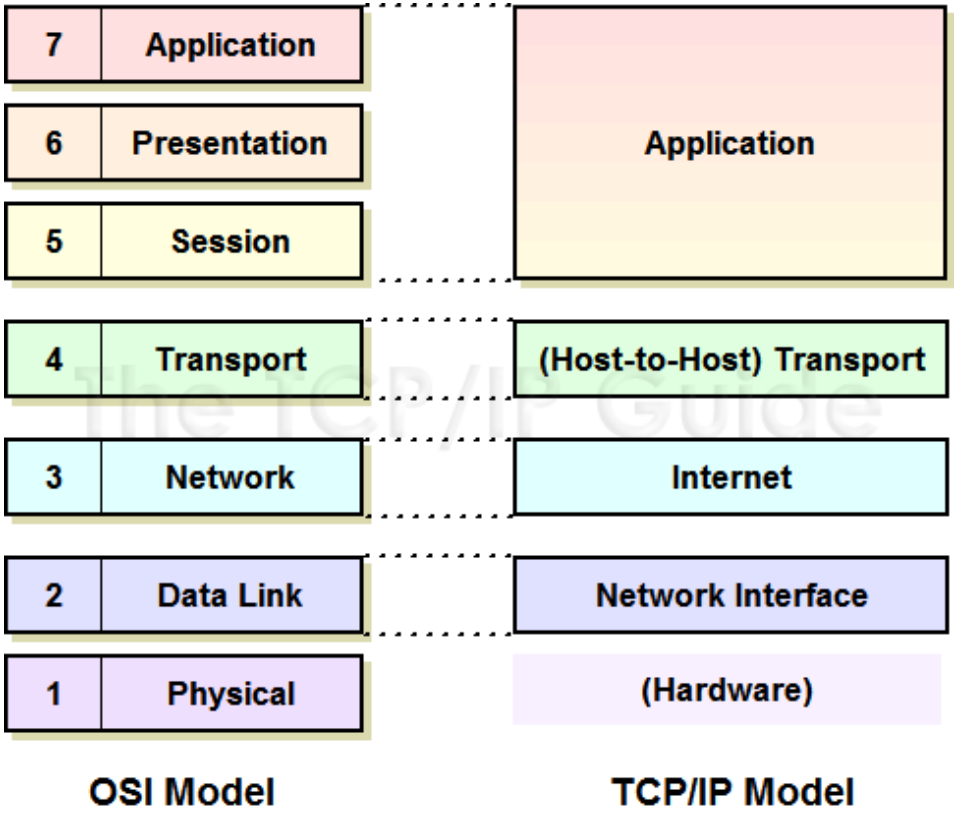
GÉANT is co-funded by the European Commission within its 7th RtD Framework Programme.

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.

European Commission Information Society and Media

# INTERNET ?

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




http://www.tcpipguide.com/free

**IP** "Internet Protocol" (couche réseau)

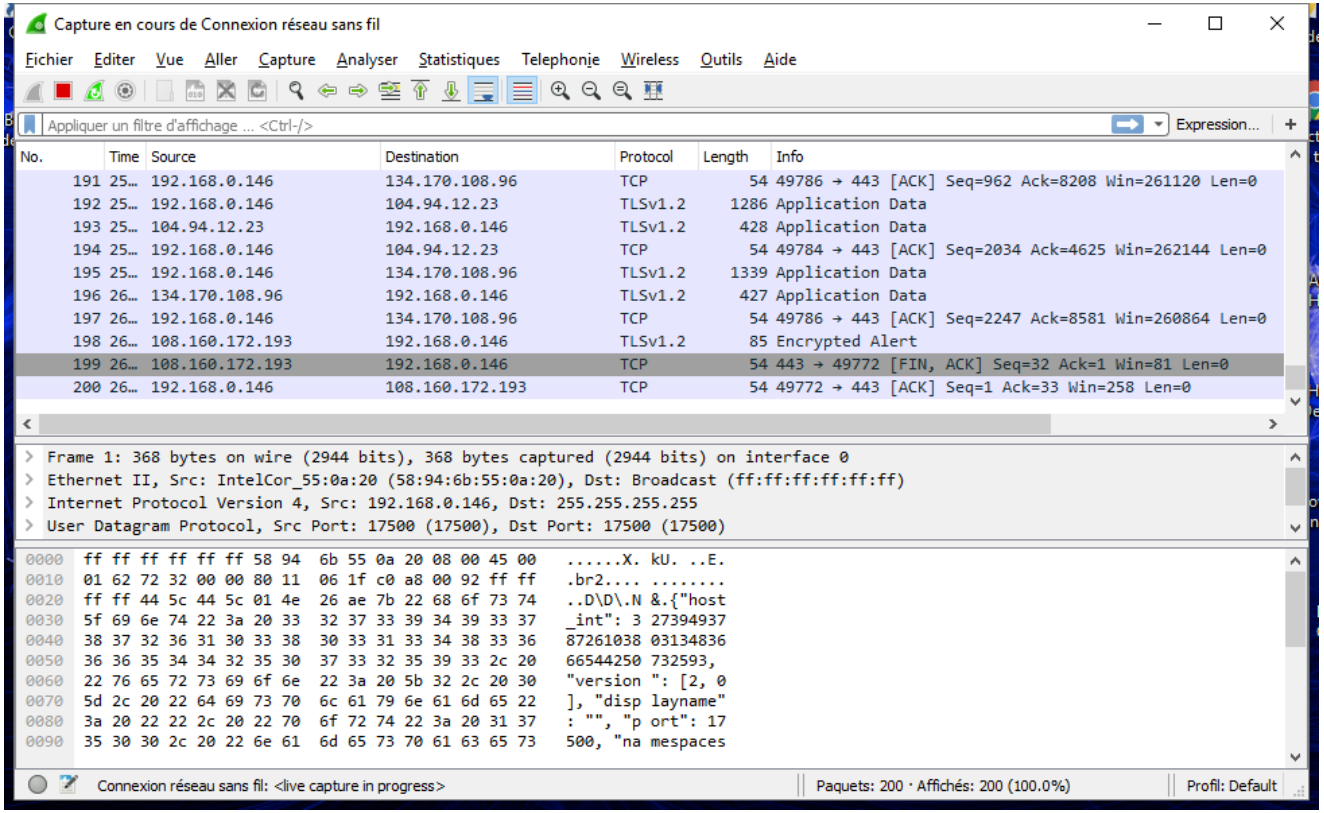
échange les données entre ordinateurs hôtes

**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.




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 !

# SYSTÈMES RÉPARTIS

**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 :

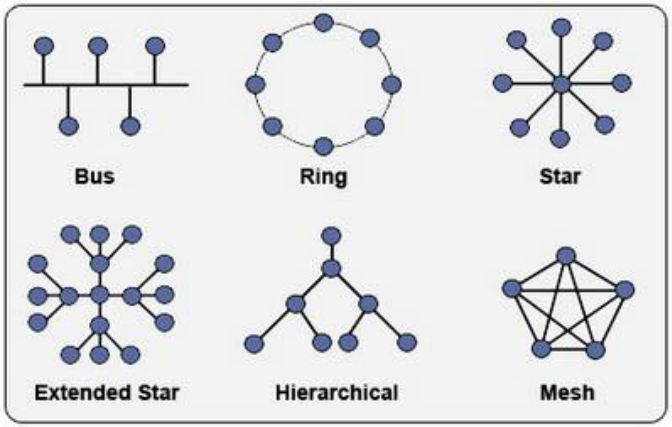
- stockage (disques, bases de données)
- puissance de calcul
- mais aussi les **utilisateurs**

# 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 ≈ <u>distribué</u>**



# 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

# 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, …)

# 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, ...)

# 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

- **accès** aux données distantes
- **maintien** du service (gestion des pannes)

- **encodage/décodage des données** (*marshalling*, *unmarshalling*)

- **accès concurrents** au(x) service(s)
- **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 en renvoie une réponse
- Ex : http**d**, ftp**d**, telnet**d**, …

**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 (Request for Comments)
- FTP → RFC 114 (Avril 1971), HTTP → RFC 1945 (mai 1996), RFC 2616 (juin 1999), …
- IPoAC (RFC 1149) ☺

# 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

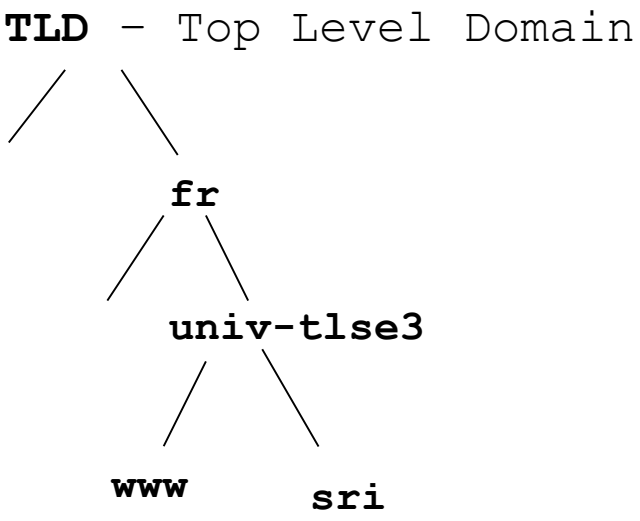
**exemple :** `sri.univ-tlse3.fr` → `195.220.43.54`

DNS : base de donnée répartie, système hiérarchique

# EXEMPLE : DNS

un serveur DNS gère un domaine

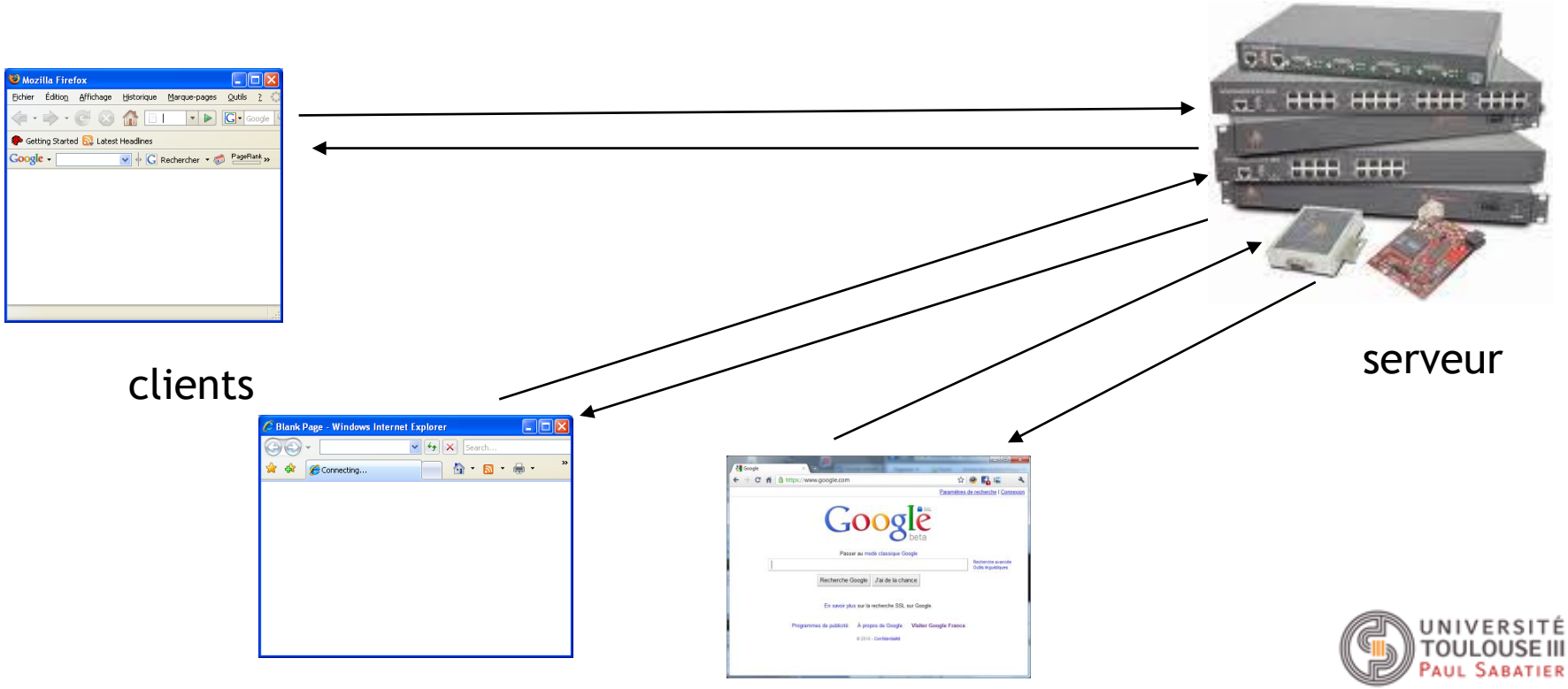
le gestionnaire peut déléguer la gestion d'un sous-domaine à une autre gestionnaire



# EXEMPLE : HTTP

HTTP : HyperText Transfer Protocol

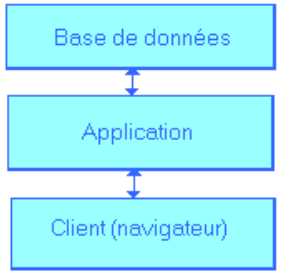
Protocole client-serveur très simple. Une requête → une réponse



# 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
leSocket = new Socket(machine, port);

// Mise en forme du flux de sortie
fluxSortieSocket = new
  PrintStream(leSocket.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** !

- programmation en java et python (la plupart du temps) mais vous pouvez choisir un autre langage si vous le souhaitez

- Objectifs : avoir des notions solides sur les protocoles les plus utilisés.

En tester les avantages et inconvénients de chaque approche

# OUTILS


**KiTTY :**

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


- http://curl.haxx.se

**Terminal :**

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

# TRAVAUX PRATIQUES (6+1 SÉANCES)

- **Echange de messages**
  - *bus à événements (ivy) – 2 h → utile pour le TP IHM*
  - programmation sockets – 2 h
  - MQTT – 4 h (+ raspberry)


- **Appel de méthodes et d'objets à distance**
  - RMI/CORBA - 2 à 4 h
  - JSON / API REST – 2 h


- **SOP : Service Oriented Programming (*support disponible*)**
  - OSGi

# EVALUATION

**UNIQUEMENT** sur un mini-projet technique !

- **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 »

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