Module 1

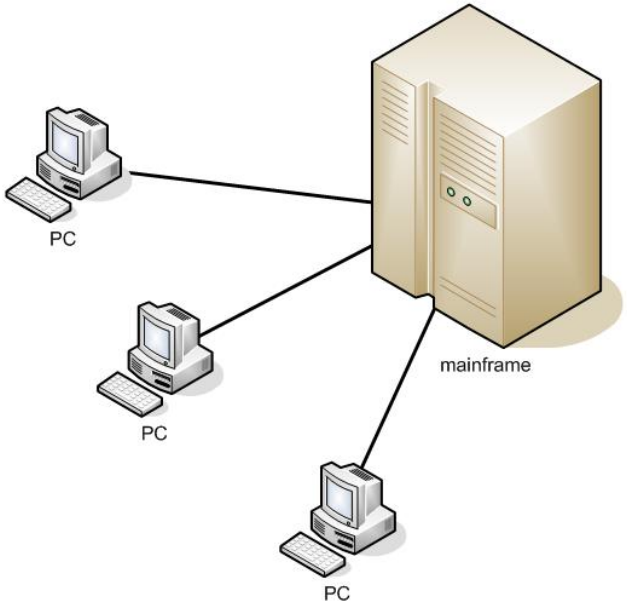
Introduction aux systèmes répartis

Évolution des organisations informatiques

- **Serveur centralisé avec terminaux**
  - Serveur coûteux;
  - Engorgement au serveur.

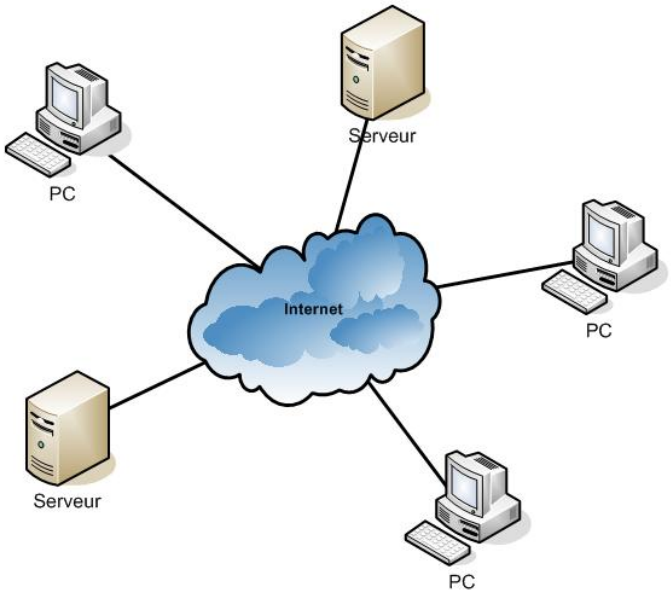
- **Ordinateurs personnels**
  - Faible coût d'achat;
  - Grand choix d'applications;
  - Autonomie mais manque de service et de coordination.

- **Systèmes répartis**
  - Le réseau partout et en continu;
  - Matériel et logiciels modulaires à faible coût;
  - Environnement hétérogène mais protocoles normalisés;
  - Redécoupage des responsabilités client et serveur;
  - Systèmes de plus en plus complexes.



Système centralisé

VS



Système réparti

Système réparti

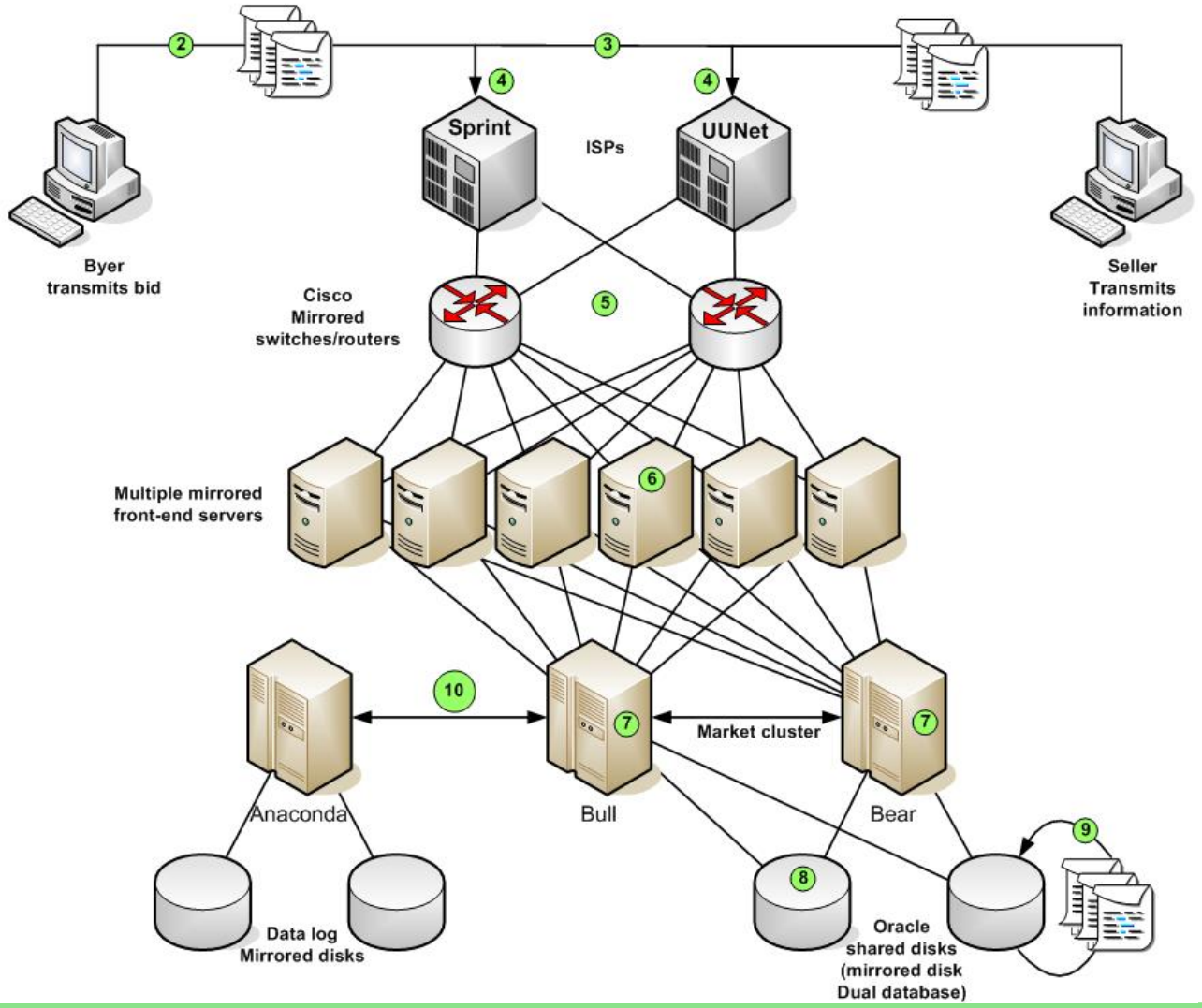
- Système dont les composantes sont réparties sur plusieurs ordinateurs en réseau et qui communiquent entre eux et coordonnent leurs actions uniquement par transmission de messages.

- Un ensemble d'ordinateurs indépendants qui, du point de vue de l'usager, apparaissent comme un système unique et cohérent.

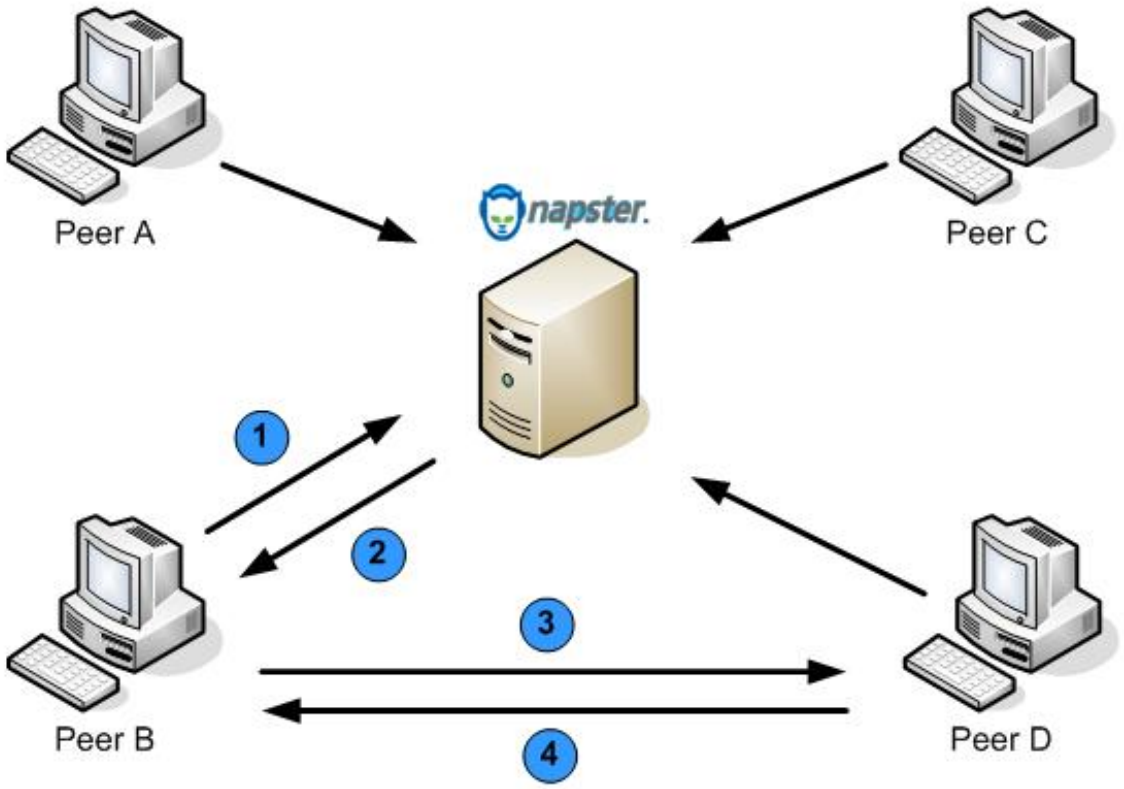
- Une définition alternative par Leslie Lamport (1987):

  "You know you have one when the crash of a computer you've never heard of stops you from getting any work done."

Architecture répartie de ebay



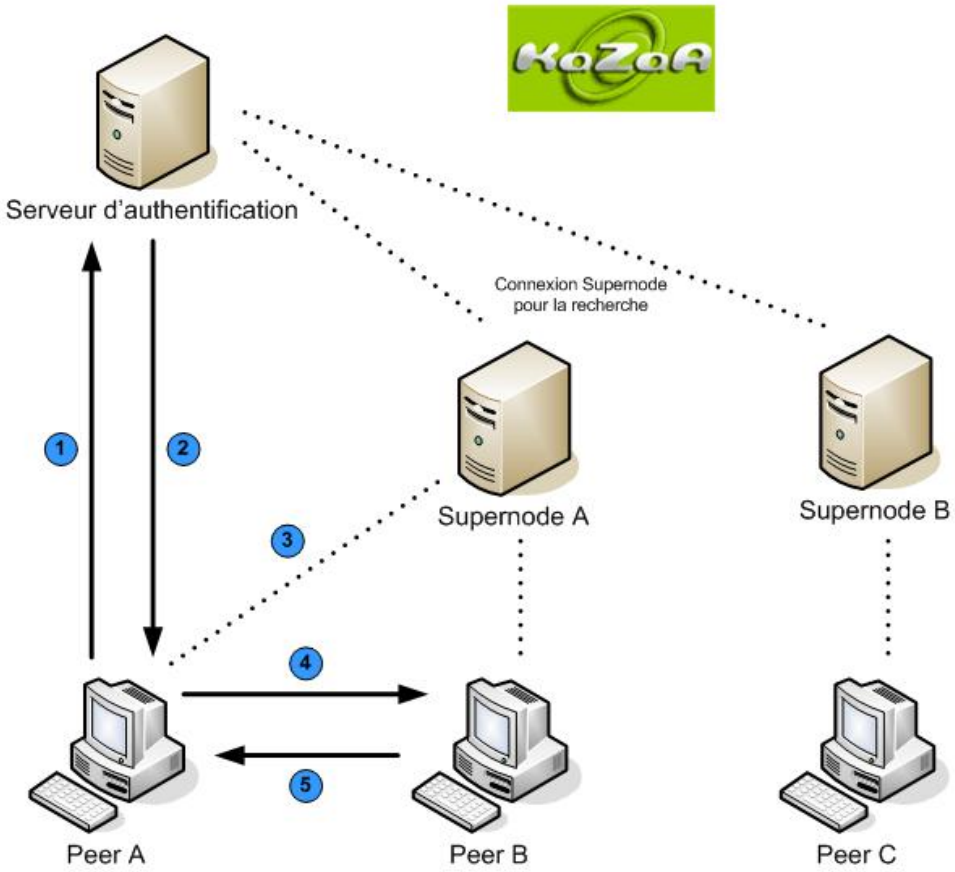
Réseau P2P centralisé http://opennap.sourceforge.net/



1. Recherche un fichier sur le serveur central
2. Serveur renvoit la liste des *peer* qui possèdent le fichier
3. *Peer B* contacte *Peer D* pour obtenir le fichier
4. Transfert du fichier

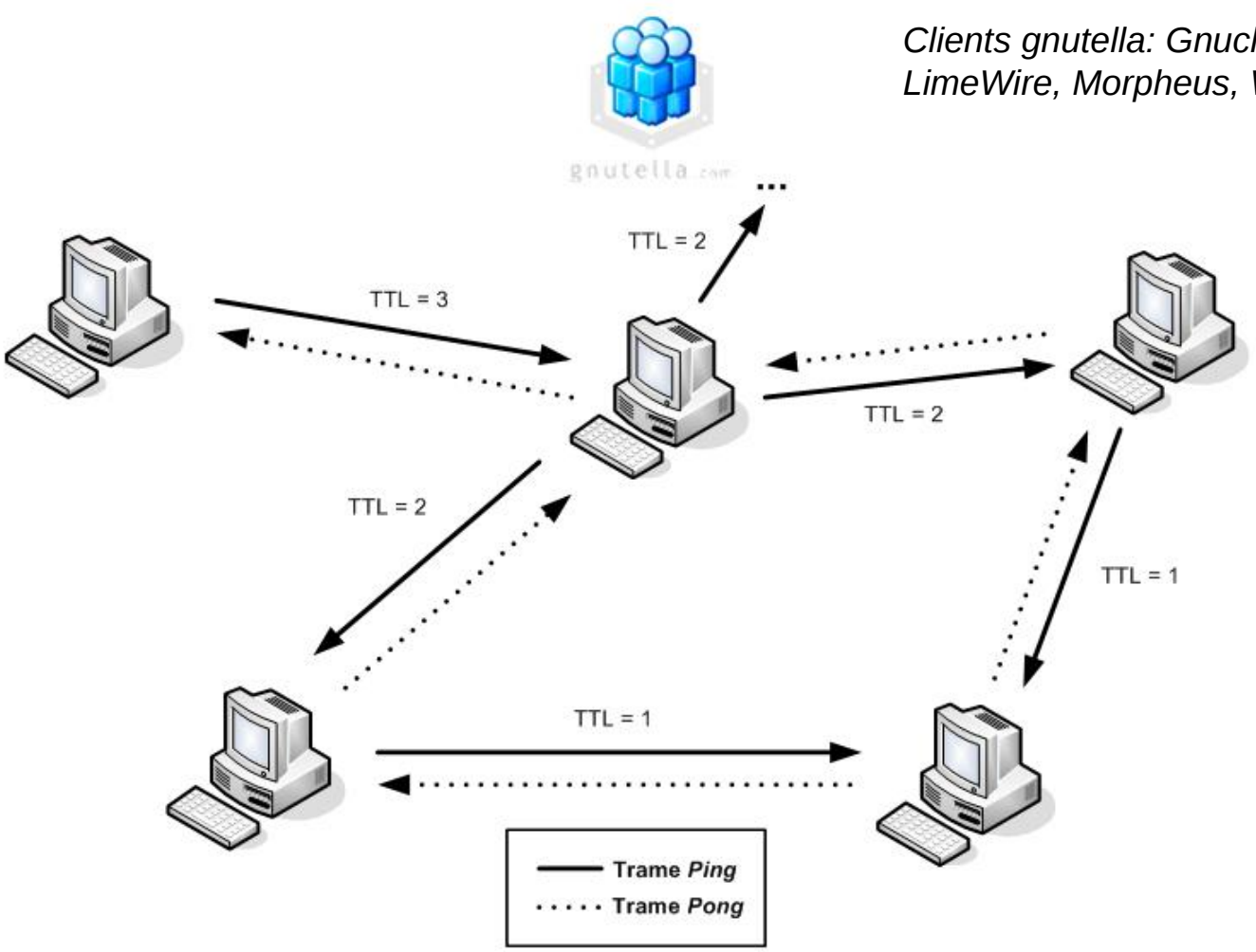
Réseau P2P hybride

http://www.kazaa.com/us/help/glossary/p2p.htm

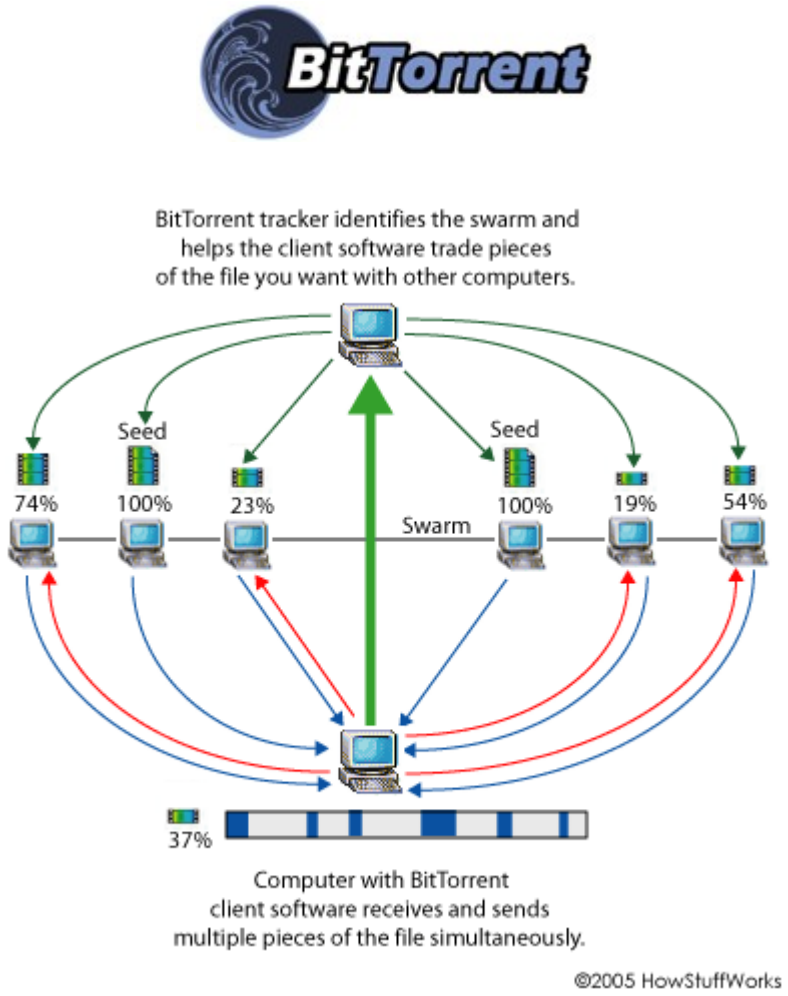


Réseau P2P décentralisé http://rfc-gnutella.sourceforge.net/

Clients gnutella: Gnucleus,BearShare, LimeWire, Morpheus, WinMX, etc.



Réseau P2P ??? http://wiki.theory.org/BitTorrentSpecification



- **D’autres exemples…**
  - L’Internet et tous ses protocoles, DNS, SMTP, BGP;
  - L’intranet d’une entreprise;
  - Le système I*nterac*;
  - Google, Facebook...

- **Pourquoi les systèmes répartis**

  - Partage des ressources (données, périphériques...);
  - Accès à des ressources distantes;
  - Augmentation modulaire de la capacité du système;
  - Possibilité de tolérance aux pannes.

- **Inconvénients des systèmes répartis**
  - Plusieurs points de défaillance;
  - Sécurité;
  - Difficulté pour le système d’avoir un état global;
  - Complexité accrue.

- **Caractéristiques des systèmes répartis**

  - **Les composantes du système :**
    - Sont réparties matériellement et/ou géographiquement;
    - Sont autonomes;
    - Sont concurrentes;
    - Peuvent défaillir indépendamment;
    - Possèdent des horloges asynchrones;
    - Communiquent par envoi de message sur le réseau.

- **Conséquences**
  - Nombreux points de défaillance possibles;
  - Décalage de temps entre les horloges de chaque système;
  - Pas d'état global;
  - Pas de garantie que les messages sont reçus;
  - Messages peuvent être interceptés, modifiés, ajoutés.

- **Historique**

  - Xerox DFS, Xerox PARC, Xerox Alto, Ethernet, 1977;
  - Cambridge Distributed Computing Services (DCS), M68000, Cambridge ring, 1979;
  - Système d'exploitation Locus de UCLA, VAX, Ethernet, 1980;
  - Apollo Domain, Token Ring, 1980;
  - Grapevine, Xerox PARC, Xerox Alto, Ethernet, 1981 (replicated distributed application-oriented database service);
  - Cedar, Xerox PARC, Xerox Dorado, Ethernet, 1982, development environment for office and personnal systems;
  - Amoeba, Vrije University, VAX/M68000..., Ethernet, 1984, distributed system based on capabilities;
  - Unix BSD 4.2 + SUN RPC/NFS, Sun, Vax/SUN, Ethernet, 1985;
  - Mach, CMU, VAX/Sun, Ethernet, 1986, système d'exploitation réparti basé sur un micro-noyau.

- **Historique (suite)**
  - World Wide Web, HTTP sur TCP/IP, 1992;
  - Groupe OMG, CORBA (Common Object Request Broker Architecture) 1992;
  - Langage Java, RMI (Remote Method Invocation), 1995;
  - Google, 1998;
  - VMWare, 1998;
  - Langage C#, Remoting, 2001;
  - Facebook, 2004;
  - Amazon EC2, 2006;
  - iPhone, Android, 2007, 2008;
  - OpenStack, 2010;
  - Docker, 2013;
  - Kubernetes et la Cloud Native Computing Foundation, 2015;

- **Les principaux problèmes à résoudre:**
  - Répartition de l'application;
  - Hétérogénéité des équipements et technologies, besoin d'interopérabilité;
  - Ouverture de système;
  - Sécurité;
  - Évolutivité;
  - Tolérance aux fautes et la fiabilité/ Détection et isolation des fautes/défaillances;
  - Concurrence, Synchronisation et Interblocage;
  - Transparence;
  - Validation et tests;

Répartition de l’application:

- Partitionnement de l'appplication en différents composants;
- Equilibrer la charge de l’application à travers différents composants répartis (client, nœuds de la grappe), statiquement ou dynamiquement;
- Architecture simple, propice à l'évolutivité et au maintien de la sécurité;

Hétérogénéité:

- Réseaux et protocoles utilisés;
- Matériel;
- Systèmes d’exploitation;
- Langages de programmation;
- Implémentations;
- Représentations internes.

Solutions:

- Protocoles et formats de stockage normalisés;
- Intergiciels d'adaptation (e.g. gRPC, CORBA, Java RMI, .NET).

Systèmes ouverts:

- Possibilité d'évoluer, de re-développer le système en tout ou en partie;
- Interopérabilité avec des systèmes complémentaires;
- Portabilité vers du nouveau matériel;
- Services développés selon des règles normalisées, formalisées à l'intérieur de protocoles, formats de stockage et interfaces de programmation.

Evolution vers les systèmes ouverts:

- Système unique, homogène;
- Développement interne, en plein contrôle;
- Applications commerciales prêtes à utiliser, plus performantes, moins chères;
- Fournisseur unique, perte de contrôle sur le prix et le cycle de mise à jour;
- Systèmes ouverts, interface de programmation (CORBA), protocoles (IIOP) et formats de stockage normalisés, code source ouvert, implémentation de référence libre (Orbit);
- Mélange de logiciels internes, logiciels libres et logiciels commerciaux

Sécurité :

- Transmettre des informations sensibles sur un lien de communication non sécuritaire et non fiable de manière sécuritaire;
- Confidentialité, intégrité, disponibilité.

Évolutivité:

- Le système doit demeurer performant lorsque le nombre d'utilisateurs et/ou de ressources augmente;
- Taille du système;
- Etendue géographique;
- Architecture du logiciel réparti;

Tolérance aux fautes et fiabilité :

- Les fautes et les défaillances sont plus courantes que dans les systèmes centralisés;
- Les défaillances sont habituellement indépendantes;
- Détection des fautes/défaillances;
- Masquage ou tolérance des fautes/défaillances;
- Redondance et réplication;

Concurrence:

- Permettre au système de traiter simultanément plusieurs requêtes à une même ressource;
- Les opérations doivent être sérialisées ou donner un résultat cohérent équivalent.

Transparence:

- Masquer à l'utilisateur tous les aspects reliés à la répartition du système;
- Accès, localisation, concurrence, réplication, défaillance, mobilité, performance, évolutivité.

Validation et tests:

- Comment tester le système complet? Chaque composante?
- Les fautes lors des tests pourraient être masquées par la tolérance aux pannes?
- Validation formelle de certaines portions.
- SPIN, modelchecker développé par Bell Labs, http://spinroot.com/spin/whatispin.html.
- UPPAAL est un modelchecker développé par l'Université Uppsala, en Suède et l'Université d'Aalborg en Danemark, http://www.uppaal.com/. Standard opensource pour la fiabilité et interopérabilité:
- Service Availability Forum (SAF) et Availability Management Framework (AMF), http://www.saforum.org/.

Modèles de systèmes

- Client-serveur.
- Client-serveurs multiples (NIS, DNS).
- Client-proxy-serveur.
- Collègues (peer to peer).
- Client + code mobile - serveur.
- Agents mobiles.
- Ordinateur réseau ou client minimal (X, VNC, Citrix).
- Réseaux spontanés (découverte de ressources DHCP, réseaux infra-rouge, bluetooth).

Modèles de pannes

- Auto-détection;
- Omission;
- Mauvaise réponse plus ou moins aléatoire;
- Erreur byzantine;
- Erreur de synchronisme.

Exemple: World Wide Web

- Format HTML, XML, CSS, XSLT, XSL-FO, Javascript;
- Réseau TCP/IP;
- Convention pour les adresses (URL), et protocole pour les requêtes (HTTP);
- Client qui exécute un fureteur: envoi de requêtes par HTTP, affichage du résultat en XML/CSS, exécution d'applet;
- Serveur: sert des requêtes HTTP à partir de fichiers HTML, de fichiers de script (CGI, PHP, jsp), ou de modules spéciaux (XML, XSLT);
- Cette plate-forme est utilisée pour accéder de l'information, rechercher des documents, consulter des annuaires, faire du courriel, interagir avec des groupes de discussion...