**Peer-to-Peer**

Applicable à :

* Distribution de contenu
* Communication
* Calcul distribué
* Collaboration

Concepts :

* Sans contrôle central
* Partage de ressources/fichiers
* Les peers sont tous égaux
* Partage volontaire
* Nombre important de pairs dans le réseau

Propriétés :

* Pas d’autorité supérieure, les pairs sont indépendants
* Flexible mais système pas forcément sûr
* Résistant aux attaques : comment le faire tomber
* Gros volumes de données

Définition :

* Un réseau P2P est comme une couche réseau
* Ce sont les utilisateurs qui forment le réseau (TCP/UDP pointent sur des IP avec check périodique si utilisateur toujours connecté)
* Cette couche réseau peut être considérée comme un service

Client-Serveur VS P2P :

* Le P2P est plus rapide quand il y a beaucoup de client (Diapo 16 pour visuel) car un serveur a une bande passante limitée

Partage de fichiers :

* Napster :
  + Serveur centralisé pour les fichiers, mais échange de pair à pair (test via ping pour meilleur tps réponse)
  + Récupération du fichier directement chez le client (mais que 1 seul client)
* Gnutella : (diapo 21-29 pour images)
  + Pas de serveur(s)
  + Communication via TCP (donc fiable niveau qualité)
  + Requête :
    - Lancée aux voisins
    - Ces voisins répercutent la requête => FLOOD
    - Les réponses sont renvoyés à l’emetteur
  + Pour ajouter un nouvel utilisateur, il faut l’adresse d’un autre user déjà membre du réseau => ensuite on apprend la topologie du réseau
    - Les réseaux ne sont donc pas public, il faut y être « invité »
    - Cela permet de maitriser la dimension
      * Suivant la bande passante des pairs, il peut y avoir saturation
  + PROBLEME : trop de traffic ping-pong (requêtes identiques entre voisins)
* Kazaa :
  + Logiciel propriétaire
  + Architecture hiérarchique mais sans serveur
  + Téléchargements parallèles en option
  + Certains pairs sont élus « super node » quand ils ont une bande passante élevée
    - Chaque super nœud agit comme un serveur Napster en collectant contenu et @IP associée des descendants
  + Mise à jour de la liste des supers nodes si jamais un ne répond plus (choix parmis 5 qui répondent)
  + Recherche : pair envoie la requete au supernode qui repond tout de suite s’il a qqch ou alors transmet à une liste de supernode qui répondront à leur tour.
  + Exploite l’hétérogénéité
* BitTorrent :
  + Partage de gros fichiers
  + Recopie rapidement vers un grand nombre d’utilisateurs
  + Fonctionnement : Client demande au WebServer le fichier .torrent et ensuite se connecte au Tracker pour les adresses des pairs qui ont le fichier
  + Contenu : split into pieces and each piece contains blocks
    - Pieces : plus petite unite de retransmission
    - Blocks : Unité de transmission
  + Seeder : il a le fichier entier
  + Leecher : a une partie du fichier
  + Session (= réseau en fait) : en gros il y a un réseau par torrent
    - Chaque torrent est donc indépendant des autres
  + Rarest first policy quand on télécharge un gros morceaux
  + Lent au démarrage mais plus les pairs ont de morceaux, plus ça va vite
  + Un plus s’il y a des seeders
  + Faiblesses :
    - Morceaux de 256KB => rarest first a besoin de plein de morceaux
    - Tout le monde doit contribuer => qu’en est-il des petites bandes passantes ?
    - Impact sur le réseau : quels pairs doit-on choisir ? les plus proches ou ceux qui débitent le plus ?
* Topologies du P2P :
  + Non-structurée :
    - pas de topologie explicite (Gnutella, Freenet)
    - pas de relation explicite entre information et localisation dans le réseau
  + Structurée :
    - un type est respecté (anneau/arbre/…)
    - une info est à un endroit donné (plus facile a trouver du coup)
  + cela dépend donc de ce que l’on cherche
* DHT :Distributed Hash Table
  + Idée : un dictionnaire de structure de données
  + Associer une clé à un item (data object / document / …)
    - Cela a pour but qu’un item soit toujours identifiable
  + Dimensionné pour des centaines de milliers de nœuds
  + Permet de prendre en compte l’arrivée et le départ des nœuds (pairs)
* Voir le reste car pas tout compris