Théorie des applications réparties

Les réseaux pair à pair

Remarques préliminaires

Pourquoi les réseaux pair à pair ?

On étudie un cas particulier, mais beaucoup de systèmes répartis mettent en œuvre des concepts proches de ce que nous allons voir durant cette séance

Théorique ou pratique ?

Ce module présente principalement des concepts théoriques sur les systèmes répartis, n'hésitez surtout pas à aller regarder les documents du module de C. Attiogbé pour avoir une vision différente de ces systèmes

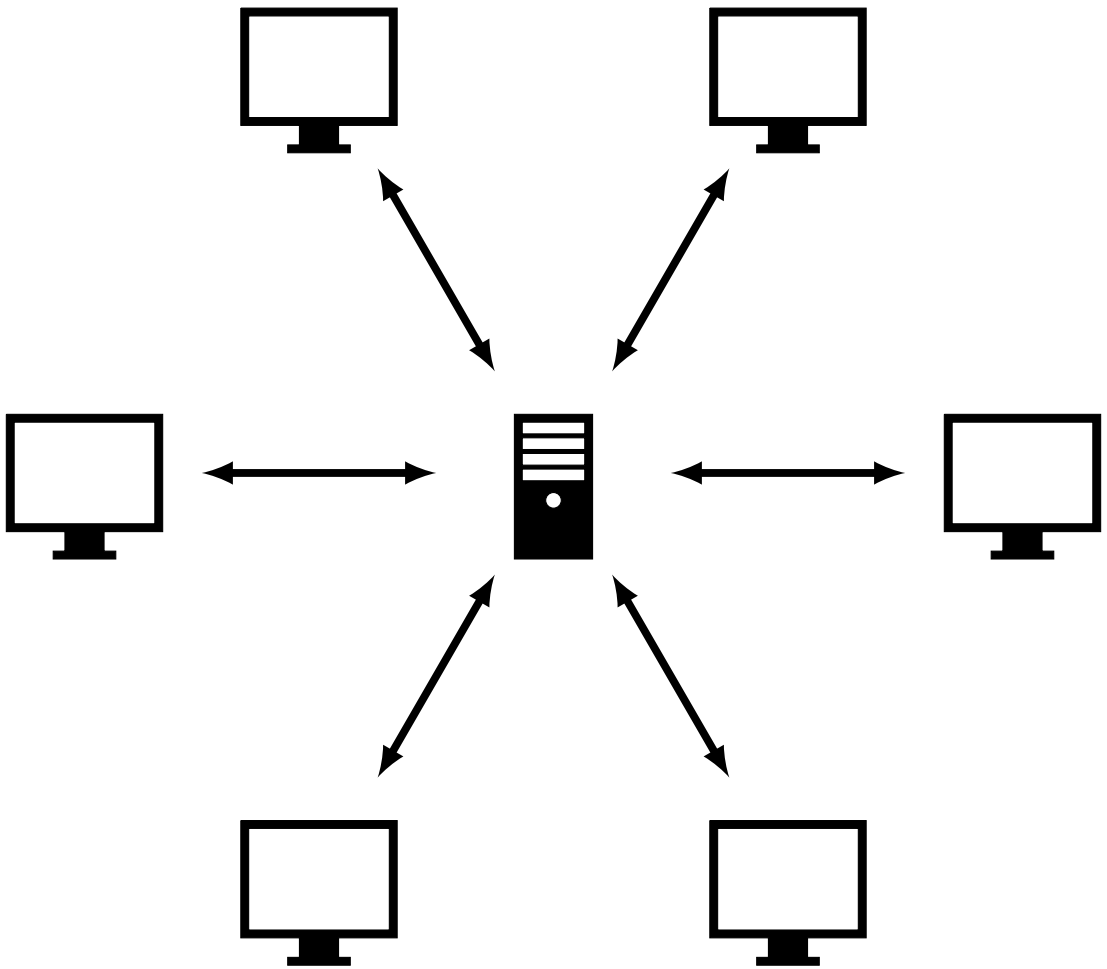
Plan du cours

1 Quelques définitions, quelques remarques préliminaires

2 Quelques exemples, historique

3 Les défis des réseaux pair à pair (sur un exemple)

Retour sur le dernier TP : le modèle client serveur



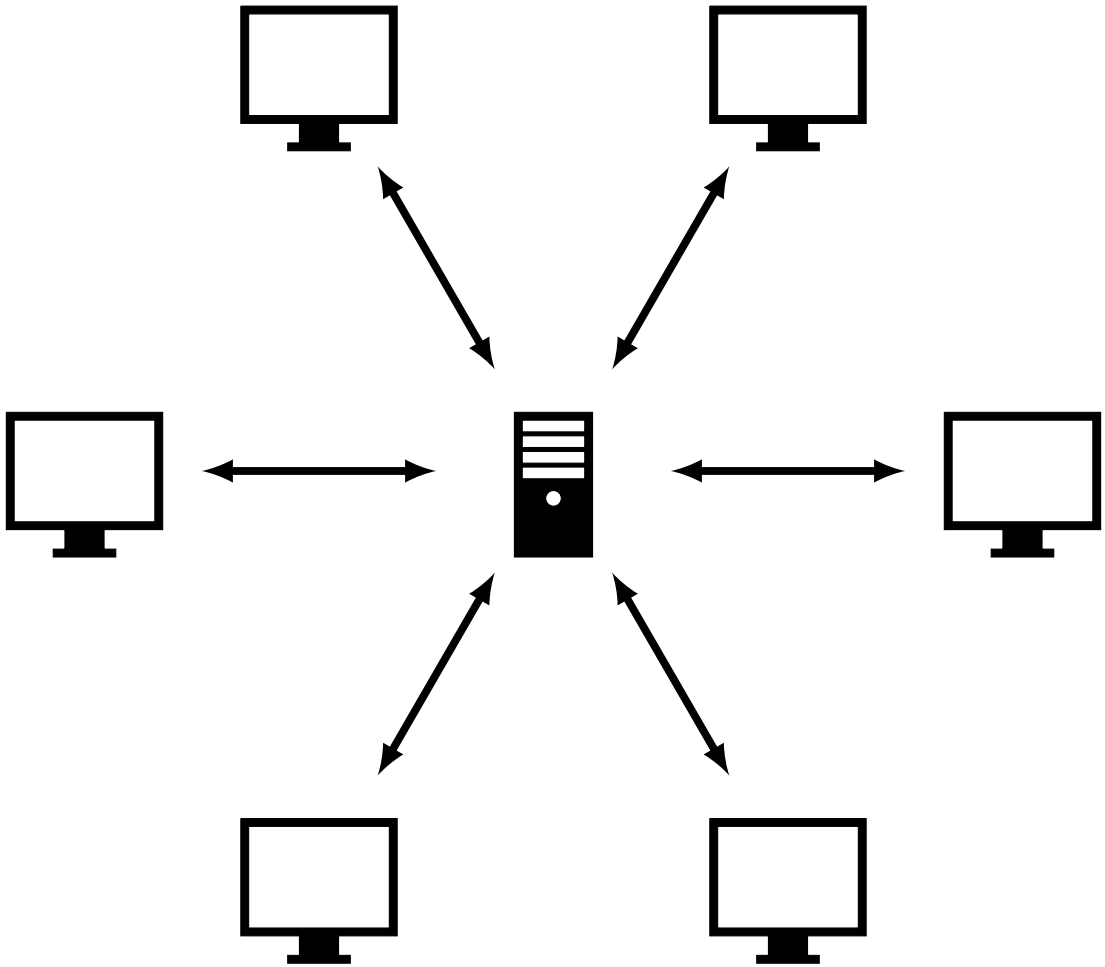
Principe

Les clients ne communiquent qu'à travers le(s) serveur(s)

Transfert de fichiers

Et si tous les clients téléchargent en même temps ?

Retour sur le dernier TP : le modèle client serveur



Principe

Les clients ne communiquent qu'à travers le(s) serveur(s)

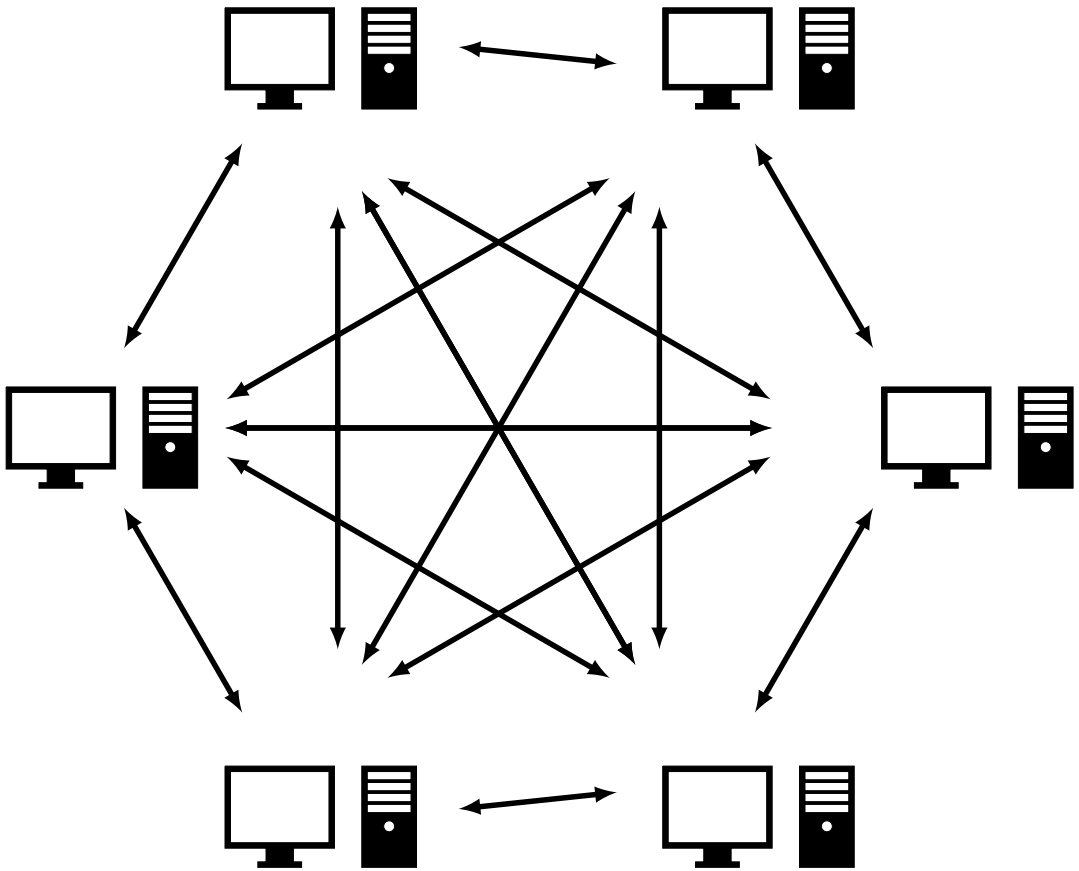
Transfert de fichiers

Et si tous les clients téléchargent en même temps ?

Autres limitations

Pannes, répartition de la charge, ...

Le(s) modèle(s) pair à pair



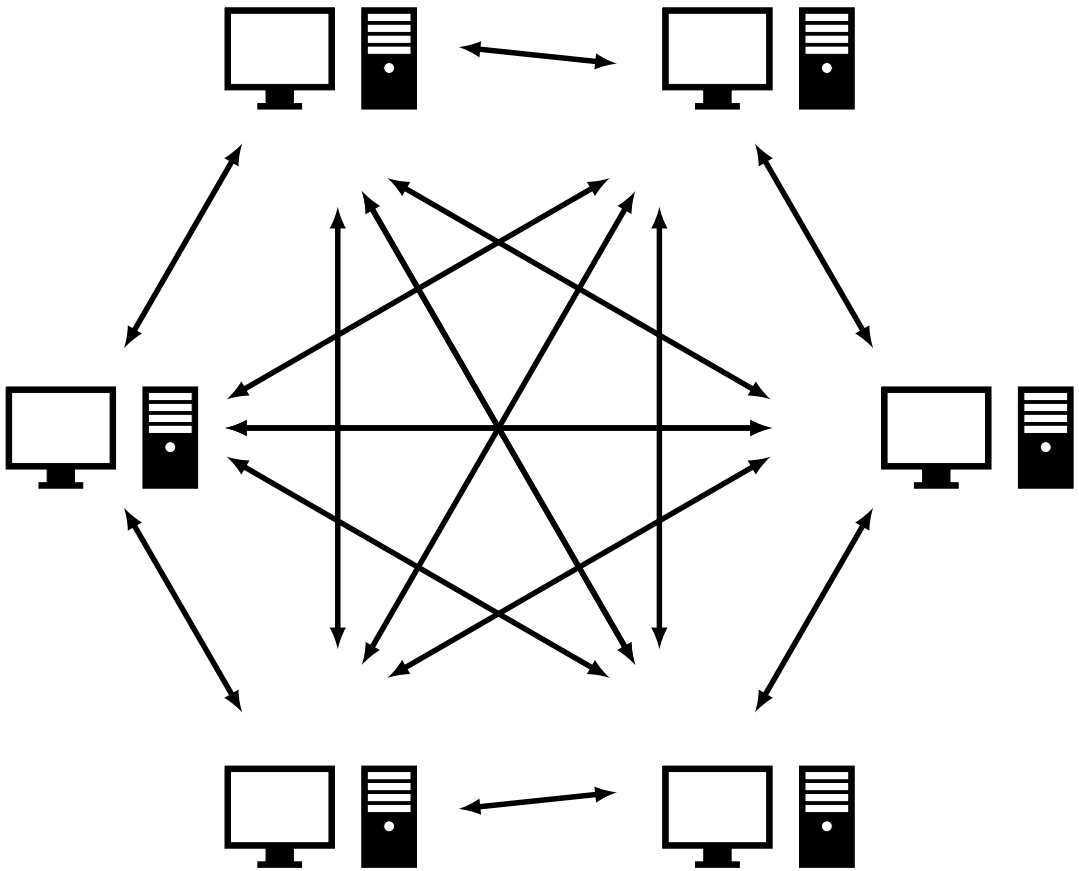
Principe

Chacun est à la fois client et serveur

Transfert de fichiers

Et si tous les clients téléchargent en même temps ?

Le(s) modèle(s) pair à pair



Principe

Chacun est à la fois client et serveur

Transfert de fichiers

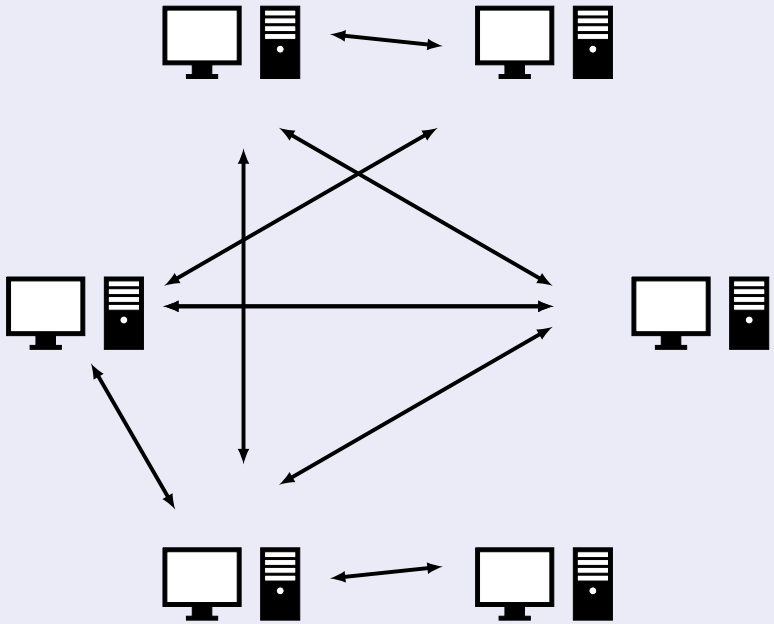
Et si tous les clients téléchargent en même temps ?

Avantages

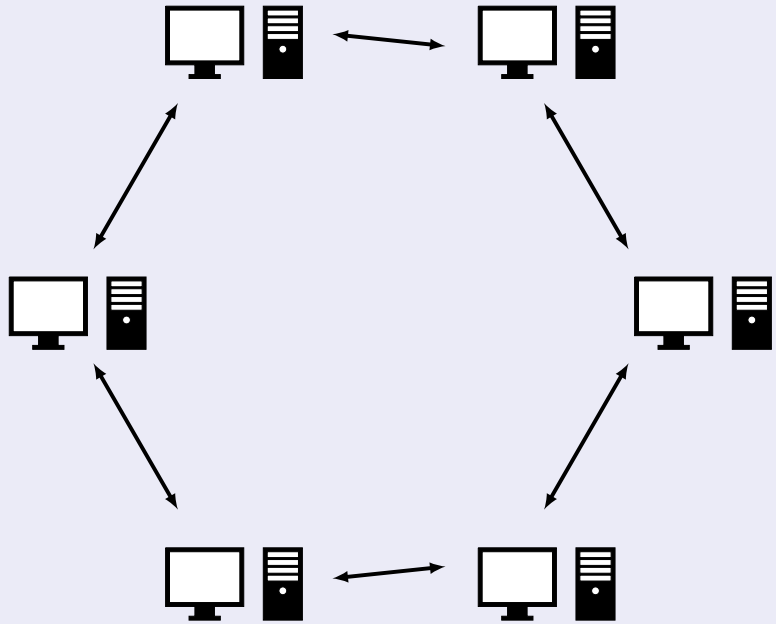
Pannes, répartition de la charge, ...

Réseau logique (overlay)

Non-structuré



Structuré

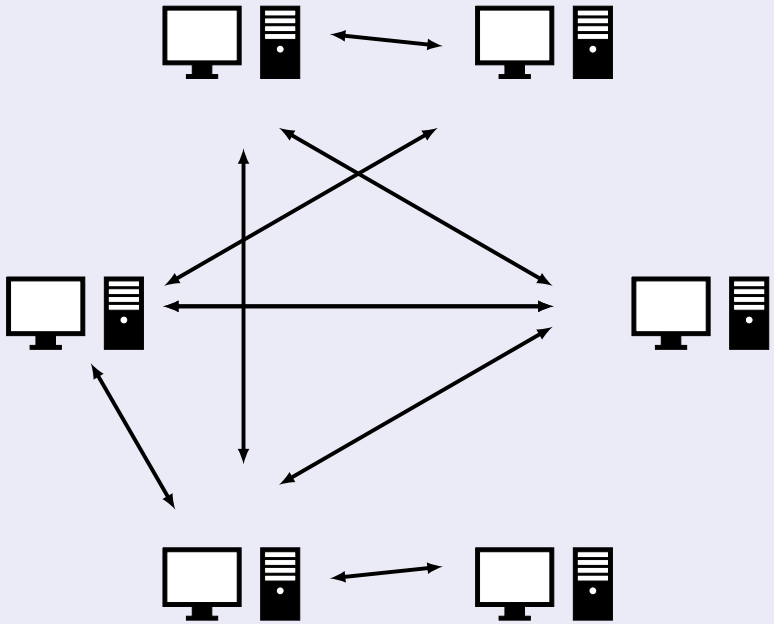


Éléments de comparaison

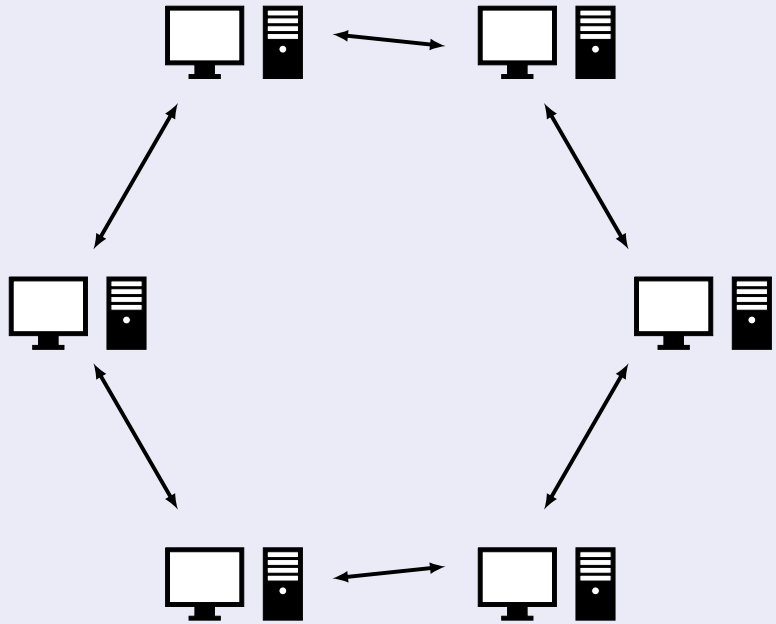
Des idées ? Arrivée dans le réseau, recherche,...

Réseau logique (overlay)

Non-structuré



Structuré



Éléments de comparaison

Des idées ? Arrivée dans le réseau, recherche,...

Remarque : il existe aussi des réseaux logiques hiérarchiques

Principaux défis

Équité et charge

Comment répartir la charge entre les pairs ? équitablement ? selon les ressources de chacun ? comment s'assurer que les pairs ne trichent pas

Tolérance aux pannes

Comment assurer la disponibilité et l'intégrité des données malgré les pannes ? quels types de pannes ?

Principaux défis

Équité et charge

Comment répartir la charge entre les pairs ? équitablement ? selon les ressources de chacun ? comment s'assurer que les pairs ne trichent pas

Tolérance aux pannes

Comment assurer la disponibilité et l'intégrité des données malgré les pannes ? quels types de pannes ?

Maintient de la structure

Plus le réseau logique est structuré, plus c'est difficile

VS

Accès aux données

Plus le réseau logique est structuré, plus c'est facile

Principaux défis

Équité et charge

Comment répartir la charge entre les pairs ? équitablement ? selon les ressources de chacun ? comment s'assurer que les pairs ne trichent pas

Tolérance aux pannes

Comment assurer la disponibilité et l'intégrité des données malgré les pannes ? quels types de pannes ?

Maintient de la structure

Plus le réseau logique est structuré, plus c'est difficile

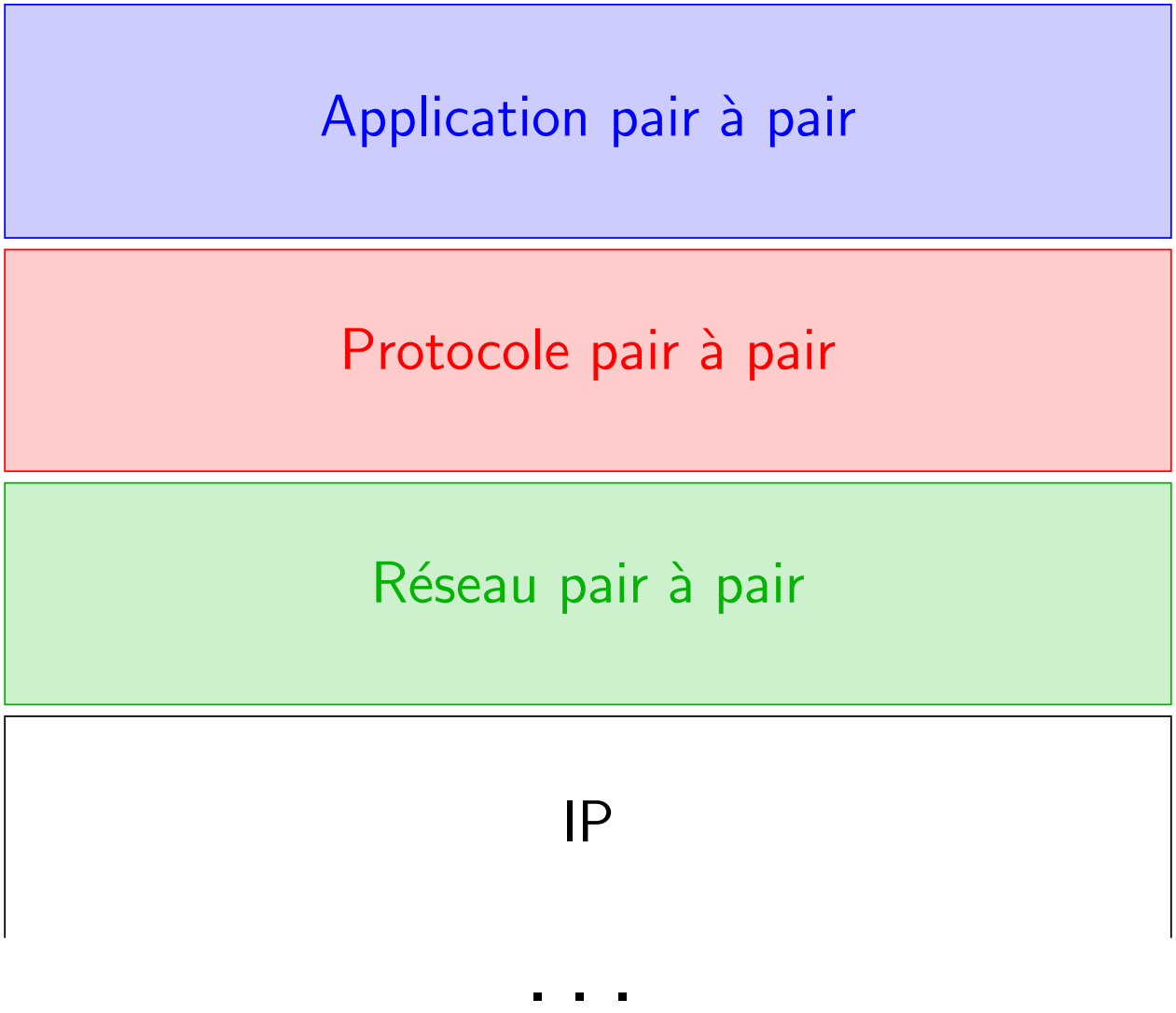
VS

Accès aux données

Plus le réseau logique est structuré, plus c'est facile

Et d'autres : anonymat, intimité/secret (privacy), sécurité, légalité...

Remarque : réseau, protocole, application



Plan du cours

1 Quelques définitions, quelques remarques préliminaires

2 Quelques exemples, historique

3 Les défis des réseaux pair à pair (sur un exemple)

Vous connaissez des applications pair à pair

Vous connaissez des applications pair à pair

Vous connaissez des applications pair à pair

Vous connaissez des applications pair à pair

Vous connaissez des applications pair à pair

Bref historique

Au commencement était Napster (juin 1999 à juillet 2001)

Philosophiquement pair à pair, mais des aspects centralisés

Échanges de fichiers musicaux

Record de fréquentation en février 2001 : 26,4 millions d'utilisateurs, 2,79 milliards de fichiers échangés dans le mois

Mais ça n'a pas duré longtemps

Procès dès décembre 1999 : Recording Industry Association of America

Fermeture par décision de justice en juillet 2001

Bref historique, suite

2000 - 2002, vers BitTorrent

2000 : Gnutella, les pairs se chargent aussi du routage

Un peu plus tard la même année : Kazaa, réseau logique hierarchique

BitTorrent (à partir de 2002)

Aujourd'hui la majeur partie du trafic pair à pair est générée par BitTorrent

En septembre 2011 on estimait que plus de 25% du trafic sur internet venait du pair à pair (et environ 40% des paquets transmis)

Un exemple de protocole fameux : BitTorrent

BitTorrent c'est quoi ?

Un protocole de partage de fichiers pair à pair

Principe de mise en ligne d'un fichier

1. Découper le fichier à partager en plusieurs parties
2. Produire un fichier .torrent décrivant le fichier à partager
3. Distribuer le .torrent de façon standard (mail, http, clé usb, . . .)
4. Rendre les parties du fichier disponibles (par exemple sur son ordinateur relié au réseau)

Un exemple de protocole fameux : BitTorrent, suite

Principe de récupération d'un fichier

1. Récupérer le .torrent correspondant
2. À partir des informations du .torrent, récupérer indépendamment les différentes parties du fichiers
3. Dès qu'une partie de fichier est récupérée par un pair, il en devient un nouveau distributeur

Un exemple de protocole fameux : BitTorrent, suite

Principe de récupération d'un fichier

1. Récupérer le .torrent correspondant
2. À partir des informations du .torrent, récupérer indépendamment les différentes parties du fichiers
3. Dès qu'une partie de fichier est récupérée par un pair, il en devient un nouveau distributeur

Quelques points forts du protocole

- Utilisation de la bande passante proche de l'optimale
- Duplication intelligente des données (téléchargement du plus rare d'abord)
- Faible impact d'une panne pendant un téléchargement
- ...

Plan du cours

1 Quelques définitions, quelques remarques préliminaires

2 Quelques exemples, historique

3 Les défis des réseaux pair à pair (sur un exemple)
- Accès aux données réparties
- Maintient d'une topologie de réseau logique (overlay)

Quelques mots sur l'exemple

Chord

Un protocole pair à pair relativement simple, proposé en 2001[1]

- ajout et retrait de pairs
- routage
- recherche de données

Caractéristiques

Passe très bien à l'échelle (les opérations de base dépendent faiblement du nombre de nœuds et de données dans le réseau)

Limitation importante

La découverte du réseau par un pair n'est pas directement possible

[1] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan. *Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications.* SIGCOMM'01.

Présentation de la topologie

Principe de base

- On considère la relation de successeur habituelle sur les entiers de 0 à n, en considérant en plus que 0 est le successeur de n : on a donc un cercle d'entiers
- On considère une fonction de hachage capable d'associer un entier entre 0 et n à chaque pair potentiel (à partir de son IP) et à chaque donnée potentielle
- Le hachage d'une donnée ou de l'IP d'un pair donne sa place sur le cercle

Présentation de la topologie

Principe de base

- On considère la relation de successeur habituelle sur les entiers de 0 à n, en considérant en plus que 0 est le successeur de n : on a donc un cercle d'entiers
- On considère une fonction de hachage capable d'associer un entier entre 0 et n à chaque pair potentiel (à partir de son IP) et à chaque donnée potentielle
- Le hachage d'une donnée ou de l'IP d'un pair donne sa place sur le cercle

Problématique

Comment se fait le routage dans le réseau ? (pour l'accès à un pair ou à une donnée)

Tables de routage

Principe

Chaque pair maintient (on verra plus tard comment) une table de routage contenant les adresses IP et les hachages de :

- au minimum le pair suivant sur le cercle (pourquoi est-ce suffisant ?)
- un petit nombre de pairs plus éloignés (pourquoi est-ce intéressant ?)

Lorsque p_i doit envoyer un message à p_j il lui suffit de l'envoyer au pair le plus proche de p_j dans sa table de routage

Pairs éloignés : les doigts (fingers)

Le pair de hachage k connaît le successeur de $k + 2^1$, celui de $k + 2^2$, etc jusqu'à faire le tour du cercle

Remarque : pour des raisons pratiques, chaque pair connaît aussi son prédécesseur direct sur le cercle

Accès aux données

Idée de départ

On l'a vu, les (hachages des) pairs et les (hachages des) données appartiennent au même espace

Principe

Chaque pair est responsable des données dont il est le premier pair successeur sur le cercle

Accès à une donnée

Il suffit d'envoyer un message au pair qui en est responsable (ce qui se fait simplement grâce aux tables de routage précédentes)

Vocabulaire

On parle de table de hachage répartie (distributed hash table, DHT)

Ajout/retrait d'une donnée

Ajout d'une donnée

Il suffit de calculer son hachage puis de transmettre cette donnée au pair qui doit en être le responsable (en faisant une recherche de cette donnée on trouve le pair en question)

Retrait d'une donnée

Le pair qui en est responsable peut directement la supprimer

Remarque : en pratique les données sont dupliquées et on n'a pas d'intérêt particulier à faire disparaître complètement l'une d'elles

Ajout d'un pair

Ce qu'il faut préserver à tout prix

- La correction de la relation de succession entre pairs
- L'association correcte entre données et pairs

Hypothèse de départ

On considère que le pair k qui veut s'insérer dans le réseau connaît déjà (l'adresse d') un autre pair k'

L'ajout de k en trois étapes

1. Calcul du prédécesseur et du successeur de k
2. Mise à jour des prédécesseurs/successeurs des autres pairs (lesquels ?)
3. Transfert de la responsabilité des données

Ajout d'un pair, suite

Ce qu'il est souhaitable de préserver en plus

- Les tables de routage telles que définies plus haut (ce n'est pas nécessaire pour que le routage soit correct, seulement pour qu'il soit efficace)

Calcul de la table de routage de k

- Récupérer la table de routage du prédécesseur de k
- Vérifier que chaque entrée est correcte, si ce n'est pas le cas on peut corriger rapidement par un calcul de successeur

Mise à jour des tables de routage des autres pairs

Quels pairs en ont besoin ? Est-ce intéressant de le faire à chaque fois ?

En pratique c'est fait une fois de temps en temps par chaque pair, sans se préoccuper des nouvelles arrivées

Retrait d'un pair

Que faut-il corriger ?

La même chose qu'à l'ajout, mais il faut surtout faire attention que le pair disparu n'apparaisse plus dans aucune table de routage

Comment ?

Le principe est le même que lors de l'ajout d'un pair : pour chaque pair dont la table de routage contient le disparu on fait une mise à jour

Difficulté

Si un pair part sans prévenir, que faire ?

Mise en place du réseau

Jusqu'à présent

On a considéré qu'un réseau pré-construit existait

Problème

Comment faire pour créer un réseau de toutes pièces ?

Mise en place du réseau

Jusqu'à présent

On a considéré qu'un réseau pré-construit existait

Problème

Comment faire pour créer un réseau de toutes pièces ?

Solution

On sait facilement construire un réseau à un seul pair !

Pour le prochain TP (i.e après la pause)

Former des groupes :

huit groupes de trois