

Le Peer-to-Peer

Fabien de Montgolfier

Université Paris 7 Denis Diderot - LIAFA
Projet INRIA Gang
fm@liafa.jussieu.fr

M2 Informatique - Univ. Marne-la-Vallée

http://www.liafa.jussieu.fr/~fm/enseignements/P2P_2.pdf

Chapitre 9 :

Stratégie de téléchargement

Quel bloc demander ?

La question

Un pair possède k blocs sur n . Il a l'opportunité d'en télécharger un de plus parmi les $n - k$ manquants. Lequel ? Sachant que tous les pairs n'ont pas tous les blocs manquants !

Push ou pull ?

- ▶ push : l'émetteur choisit le bloc à envoyer
- ▶ pull : le récepteur choisit le bloc à envoyer

La réponse

Stratégies eMule et BitTorrent : pull du bloc le plus rare du client. Pourquoi ?

Stratégies de pull [Mathieu Reynier 2004]

Donnée

matrice M . Lignes = pairs. Colonnes = blocs demandés
($M(p, b) = 1$ si p possède b , 0 sinon)

Hypothèse du modèle

Chaque pair peut télécharger un bloc à chaque round

Choix stratégiques

- ▶ Ordre des choix :
 1. Choix du pair, puis choix du bloc
 2. Choix du bloc, puis choix du pair
- ▶ Quel choix :
 1. Uniformément au hasard (U)
 2. Le plus rare (R)

Stratégies de pull [Mathieu Reynier 2004]

Stratégies

- ▶ Random : case à 1 prise uniformément dans M (Rand)
- ▶ PUBU : choix du pair uniformément, puis choix du bloc unif (NB : \neq Rand!)
- ▶ BUPU : choix du bloc unif., puis du pair unif. (NB : \neq Rand et \neq PUBU)
- ▶ PUBR : choix du pair unif., puis son bloc le plus rare
- ▶ PRBR : choix du pair ayant le bloc le plus rare, puis ce bloc
- ▶ BRPU : choix du bloc rare, puis unif. entre les pairs l'ayant
- ▶ aussi : PRBU, BUPR, BRPR

Modèle [Mathieu Reynier 2004]

- ▶ Une source
- ▶ Tout le temps k téléchargeurs
- ▶ Personne ne reste source après complétion

Torpeur

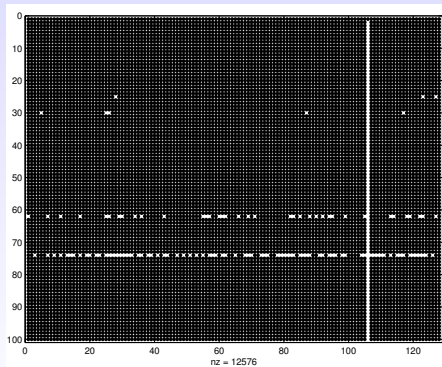
État où (presque) tous les pairs possèdent tous les blocs sauf 1
Un seul pair par round termine son téléchargement

État normal

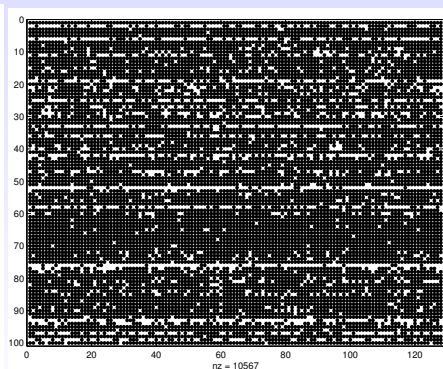
(presque) tous les pairs contribuent
Une importante fraction de pairs terminent à chaque round
La distributions de tous les blocs est (en gros) la même

Visuellement :

Tropeur :



État normal :



Résultats empiriques [Mathieu Reynier 2004]

Très mauvaises stratégies

PUBU et BUPU donnent toujours une torpeur

Mauvaise stratégie

Rand donne une fois sur deux une torpeur, une fois sur deux un état normal

Bonnes stratégies

Les autres stratégies (incluant une recherche de rareté) donnent des états normaux

La stratégie du bloc rare

Stratégie (eMule, BitTorrent, ...)

Télécharger le bloc le plus rare (d'après la vue locale) possédé par l'émetteur

Avantages

On peut prouver que, sous de bonnes hypothèses, la nombre de copies de chaque bloc est en gros le même. Mieux : la fonction donnant le nombre de blocs en fonction du nombre de copies est une **gaussienne**.

Donc plus de "mort" du fichier par disparition de bloc

Inconvénient

Un client qui n'a que des blocs très fréquents n'intéresse personne

Avalanche

Avalanche [Gkantsidis et Rodriguez 2005]

On n'échange non plus des blocs mais des combinaisons de blocs.
Chaque combinaison est susceptible d'intéresser tout le monde.

Codage

Soit \mathbb{K} un corps fini. Un bloc B est vu (i.e. codé) comme une suite $b_1..b_n$ d'éléments de \mathbb{K}

Combinaisons

Un pair qui possède k blocs B_1, \dots, B_k tire (uniformément indépendamment) x_1, \dots, x_k dans \mathbb{K} et transmet

$$x_1 B_1 + \dots + x_k B_k = \begin{pmatrix} x_1 b_{11} + \dots + x_k b_{k1} \\ \vdots \\ x_1 b_{1n} + \dots + x_k b_{kn} \end{pmatrix} \text{ avec } B_i = b_{1i}..b_{ni}$$

Avalanche

Protocole

- ▶ Fichier découpé en b blocs (de n éléments chacun)
- ▶ La source donne des combinaisons aléatoires des b blocs.
- ▶ Chaque client donne des combinaisons de ses combinaisons.
- ▶ Avec chaque combinaison il faut fournir les $k \leq b$ coefficients

Difficultés

- ▶ Sur-coût de communication pour donner les coefficients = une ligne en plus dans la matrice. Il faut donc $k \ll n$.
- ▶ Une combinaison doit être téléchargée entièrement depuis le même pair, donc petits blocs (n petit)
- ▶ Comment retrouver les blocs à partir des combinaisons ?
- ▶ Comment vérifier la validité d'une combinaison ?

Avalanche : décodage

décodage

Une fois b blocs récupérés, il faut inverser une matrice $b \times b$ pour retrouver les blocs originaux.

b blocs suffisent-ils ?

Théorème : Si un pair possède $b - 1$ combinaisons linéaires indépendantes, alors une combinaison linéaire aléatoire permet d'obtenir une matrice inversible avec probabilité constante > 0 .

Intuition de preuve : La nouvelle combinaison a une composante nulle sur la normale à l'hyperplan des $b-1$ des combinaisons déjà présentes avec probabilité $1/|\mathbb{K}|$. On prend \mathbb{K} gros. Conséquence : tout pair P qui possède un bloc récupéré d'un pair Q intéresse presque tout les autres pairs (exceptés P et Q).

Avalanche : plus de bloc rare !

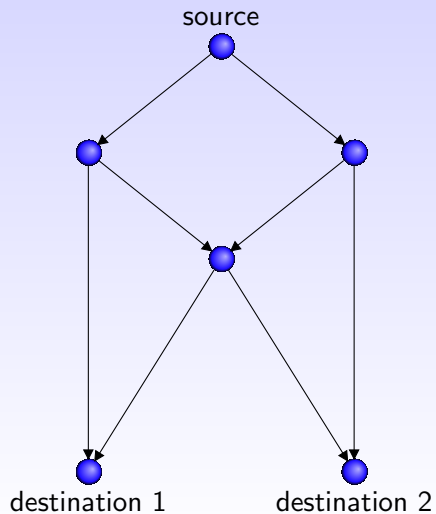
Quel bloc télécharger depuis quel pair ?

Sous de bonnes hypothèses, chaque nouvelle combinaison (nouvelle ligne) augmente de 1 le rang de la matrice des blocs possédés. On peut donc télécharger toute combinaison de tout voisin

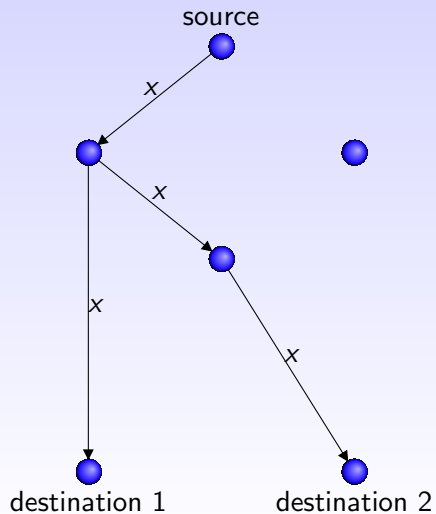
Limite

sauf que le voisin ne doit pas émettre plus vite qu'il ne reçoit

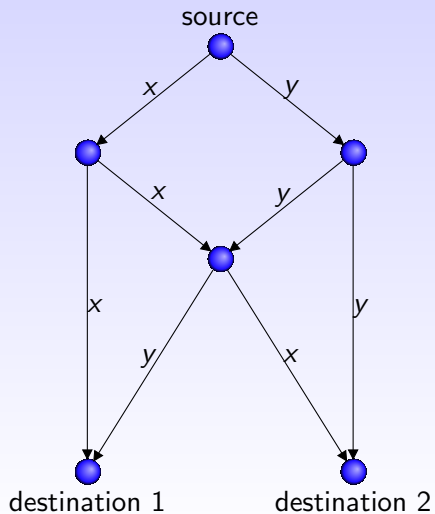
Network coding



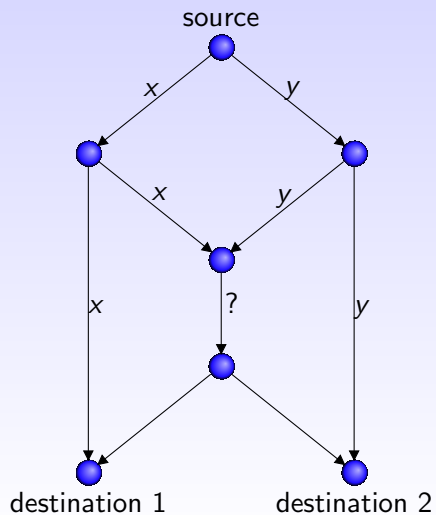
Network coding



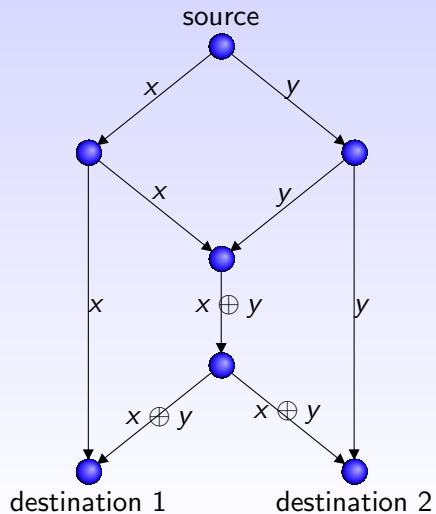
Network coding



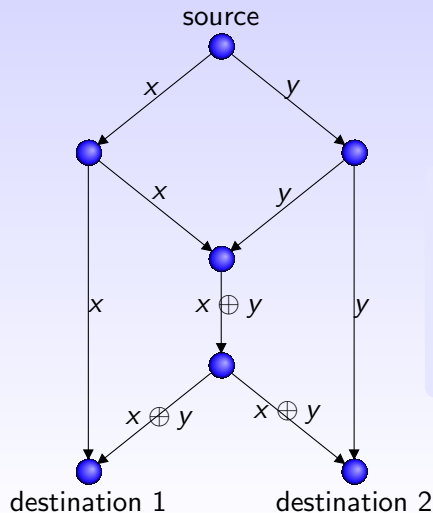
Network coding



Network coding



Network coding



Théorème [Ahlswede & al. 2000]

Pour tout graphe et tout couple source, destination, il existe un codage permettant d'atteindre le flot maximal.

Efficacité du codage

Theoreme[Ahlsvede et al. 2000, Koetter et al. 2003]

Algorithmes polynomiaux pour trouver les combinaisons permettant d'atteindre le flot maximal f pour r destinations avec $|\mathbb{K}| = fr$.

Theoreme [Jaggi, Sander, Chou, Ho, Zongpeng, ... 2003-2004]

- ▶ Algorithmes avec $|\mathbb{K}| = r$.
- ▶ Prendre des combinaisons aléatoires marche dans un graphe avec m arcs avec probabilité supérieure à $1 - \frac{mr}{|\mathbb{K}|}$.

Rateless [Maymounkov et Mazières 2003]

Contraintes

- ▶ Les clients ne peuvent pas recombinaison leurs combinaisons.

Génération d'une combinaison

- ▶ Tirer un degré δ selon une loi bien choisie.
- ▶ Prendre δ blocs au hasard.
- ▶ Donner leur XOR

Similaire à Digital Fountain [Byer & al 1998], LT-codes [Luby].

Décodage

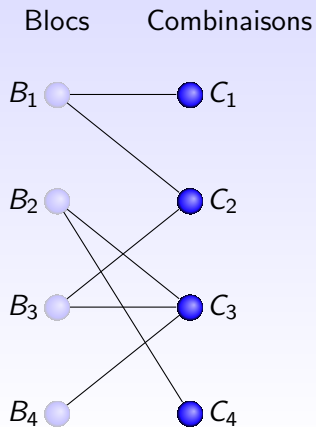
- ▶ Trouver un bloc codé dont tous les blocs dont il est le XOR sont connus sauf un.
- ▶ Le bloc inconnu est obtenu en XORant le bloc codé avec les $i - 1$ blocs connus.

Ordre de décodage

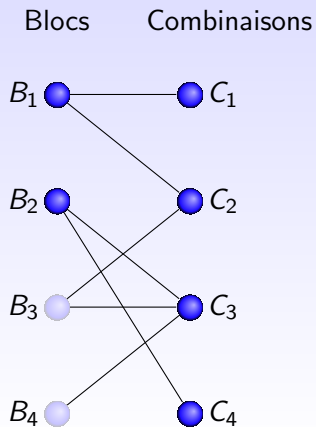
exploration du graphe biparti blocs codés / blocs du message :

- ▶ Trouver un bloc codé de degré 1, calculer le bloc associé et retirer toutes les arêtes de ce bloc associé.
- ▶ Travailler sur le sous-graphe des sommets à 2^l sauts d'un bloc à décoder suffit (avec $l = O(1)$).

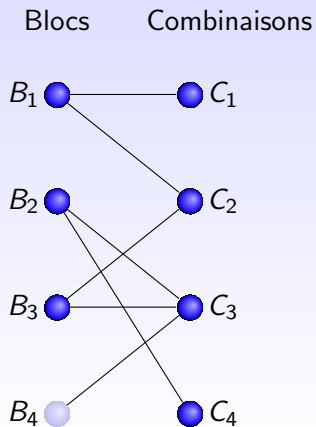
Décodage



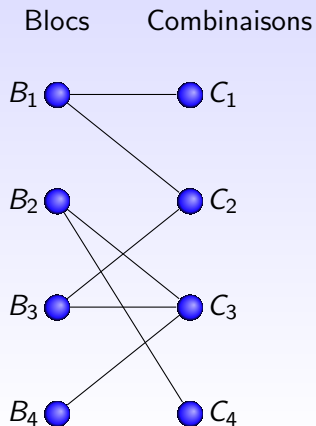
Décodage



Décodage



Décodage



Efficacité de Rateless [Maymounkov et Mazières 2003]

Théorème

$\forall \epsilon \exists \delta$ tq $(1 + \delta)n$ combinaisons permettent de récupérer $(1 - \epsilon)n$ blocs en $O(n)$ XORs, et en temps $O(nf(\delta, \epsilon))$ avec forte probabilité

Théorème

$(1 + \epsilon)n$ blocs codés permettent d'obtenir le message en temps $O(n \log 1/\epsilon)$ avec forte probabilité

Conclusion sur Rateless

remarque sur la distribution des combinaisons

Il faut donc avoir beaucoup de combinaisons de faibles degré. Mais aussi quelques unes de fort degré, pour la redondance.

Dans Rateless, la probabilité que le degré soit d est en $O(1/d^2)$.

Avantages

- ▶ Le problème du bloc rare est éliminé
- ▶ Réciprocité incitant à l'émission (à la BitTorrent) possible
- ▶ Le décodage est plus facile qu'avec Avalanche

Inconvénients

- ▶ Indépendant du graphe de connaissances (\neq Avalanche)
- ▶ Seule les sources créent des combinaisons
- ▶ signature d'intégrité par hascode ?

Chapitre 10 :

Mariages stables et Peer-to-peer

Plan du chapitre

1. Théorie des mariages stables
2. Réseaux P2P sans incitation au téléchargement
3. Réseaux P2P à incitation
 - 3.1 Le cas des préférences acycliques
 - 3.2 Simulations diverses et variées
 - 3.3 Preuve de stratification

Mariages stables : le problème

- ▶ Hypothèse : hommes et femmes ont des préférences (ordre total sur un sous-ensemble de l'autre sexe)
- ▶ Chaque homme classe les femmes
- ▶ Chaque femme classe les hommes
- ▶ Qui va épouser qui ?

Exemple

Préférences

- ♀ Alice : Anselme Bertrand Charles
- ♀ Bianca : Anselme Charles Bertrand
- ♀ Caroline : Bertrand Charles Anselme
- ♂ Anselme : Caroline Bianca Alice
- ♂ Bertrand : Alice Bianca Caroline
- ♂ Charles : Alice Bianca Caroline

Exemple

Préférences

- ♀ Alice : Anselme Bertrand Charles
- ♀ Bianca : Anselme Charles Bertrand
- ♀ Caroline : Bertrand Charles Anselme
- ♂ Anselme : Caroline Bianca Alice
- ♂ Bertrand : Alice Bianca Caroline
- ♂ Charles : Alice Bianca Caroline

Si les couples suivants se forment :

Alice-Anselme

Bianca-Bertrand

Caroline-Charles

Exemple

Préférences

- ♀ Alice : Anselme Bertrand Charles
- ♀ Bianca : Anselme Charles Bertrand
- ♀ Caroline : Bertrand Charles Anselme
- ♂ Anselme : Caroline Bianca Alice
- ♂ Bertrand : Alice Bianca Caroline
- ♂ Charles : Alice Bianca Caroline

Si les couples suivants se forment :

Alice-Anselme

Bianca-Bertrand

Caroline-Charles

Bianca préférerait être avec Anselme... Anselme préférerait être avec Bianca... Adultère !

Définitions

Adultère

couples (f, g) et (f', g') où f préfère g' et g' préfère f

- ▶ Le couple (f, g') se forme
- ▶ f' et g deviennent célibataires

Solution stable (mariage stable)

aucun adultère possible

Exemple

Préférences

- ♀ Alice : Anselme Bertrand Charles
- ♀ Bianca : Anselme Charles Bertrand
- ♀ Caroline : Bertrand Charles Anselme
- ♂ Anselme : Caroline Bianca Alice
- ♂ Bertrand : Alice Bianca Caroline
- ♂ Charles : Alice Bianca Caroline

Mariage stable

- ▶ Bianca ♡♡ Anselme
- ▶ Alice ♡♡ Bertand
- ▶ Caroline ♡♡ Charles

Algorithme du bal de Gale & Shapley

- ▶ Les garçons invitent les filles à danser
- ▶ Chaque garçon propose aux filles de danser, par ordre de préférence
 - ▶ Soit sa demande est refusée : il passe à la suivante
 - ▶ Soit sa demande est acceptée : il dance avec elle
 - ▶ Mais sa compagne peut le plaquer ! Car :
- ▶ Chaque fille accepte une proposition d'un garçon
 - ▶ Seulement si elle a classé ce garçon, et
 - ▶ si elle est célibataire,
 - ▶ ou si le garçon qui l'invite est mieux pour elle que celui avec qui elle danse

Théorème de Gale-Shapley

on calcule ainsi une solution stable

Exemple

Préférences

♀ Alice : Anselme Bertrand Charles

♀ Bianca : Anselme Charles Bertrand

♀ Caroline : Bertrand Charles Anselme

♂ Anselme : Caroline Bianca Alice

♂ Bertrand : Alice Bianca Caroline

♂ Charles : Alice Bianca Caroline

Tous sont célibataires

Exemple

Préférences

- ♀ Alice : Anselme Bertrand Charles
- ♀ Bianca : Anselme Charles Bertrand
- ♀ Caroline : Bertrand Charles Anselme
- ♂ Anselme : Caroline Bianca Alice
- ♂ Bertrand : Alice Bianca Caroline
- ♂ Charles : Alice Bianca Caroline

1. Anselme invite Caroline

- ▶ Anselme ♡♡ Caroline
- ▶ Alice, Bianca, Bertrand, Charles : célibataires

Exemple

Préférences

- ♀ Alice : Anselme Bertrand Charles
- ♀ Bianca : Anselme Charles Bertrand
- ♀ Caroline : Bertrand Charles Anselme
- ♂ Anselme : Caroline Bianca Alice
- ♂ Bertrand : Alice Bianca Caroline
- ♂ Charles : Alice Bianca Caroline

2. Charles invite Alice

- ▶ Anselme ♡♡ Caroline
- ▶ Charles ♡♡ Alice
- ▶ Bianca, Bertrand : célibataires

Exemple

Préférences

- ♀ Alice : Anselme Bertrand Charles
- ♀ Bianca : Anselme Charles Bertrand
- ♀ Caroline : Bertrand Charles Anselme
- ♂ Anselme : Caroline Bianca Alice
- ♂ Bertrand : Alice Bianca Caroline
- ♂ Charles : Alice Bianca Caroline

3. Bertand invite Alice, qui plaque Charles

- ▶ Anselme ♡♡ Caroline
- ▶ Bertand ♡♡ Alice
- ▶ Bianca, Charles : célibataires

Exemple

Préférences

- ♀ Alice : Anselme Bertrand Charles
- ♀ Bianca : Anselme Charles Bertrand
- ♀ Caroline : Bertrand Charles Anselme
- ♂ Anselme : Caroline Bianca Alice
- ♂ Bertrand : Alice Bianca Caroline
- ♂ Charles : Alice Bianca Caroline

4. Charles invite Alice → elle le refuse

- ▶ Anselme ♡♡ Caroline
- ▶ Bertand ♡♡ Alice
- ▶ Bianca, Charles : célibataires

Exemple

Préférences

- ♀ Alice : Anselme Bertrand Charles
- ♀ Bianca : Anselme Charles Bertrand
- ♀ Caroline : Bertrand Charles Anselme
- ♂ Anselme : Caroline Bianca Alice
- ♂ Bertrand : Alice Bianca Caroline
- ♂ Charles : Alice Bianca Caroline

5. Charles invite Bianca

- ▶ Anselme ♡♡ Caroline
- ▶ Bertand ♡♡ Alice
- ▶ Charles ♡♡ Bianca

Exemple

Préférences

- ♀ Alice : Anselme Bertrand Charles
- ♀ Bianca : Anselme Charles Bertrand
- ♀ Caroline : Bertrand Charles Anselme
- ♂ Anselme : Caroline Bianca Alice
- ♂ Bertrand : Alice Bianca Caroline
- ♂ Charles : Alice Bianca Caroline

On a un mariage stable

- ▶ Anselme ♡♡ Caroline
- ▶ Bertand ♡♡ Alice
- ▶ Charles ♡♡ Bianca

Déroulement du bal

Non-répétition des rateaux

Il est inutile de ré-inviter une fille qui vous a déjà refusée, car elle danse avec mieux que vous

Évolution des danseurs

- ▶ Les filles dansent avec des garçons de plus en plus haut dans leur liste
- ▶ Inversement, les garçons dansent avec des filles de plus en plus bas dans la liste

Théorème de Gale-Shapley

Théorème

- ▶ La solution est stable
- ▶ Elle ne dépend pas de l'ordre des invitations faites
- ▶ Dans aucune solution stable, un garçon ne peut trouver une fille mieux (pour lui) que celle qu'il obtient avec l'algorithme du bal
- ▶ Inversement, dans aucune solution stable, une fille ne peut trouver un garçon pire (pour elle) que celui qui l'invite en dernier !

Démonstration

- ▶ Si un garçon propose à une fille mieux : rateau → stable
- ▶ Il a déjà proposé à toute fille mieux et fut plaqué → ces couples étaient instables

Le quart d'heure américain

Préférences

- ♀ Alice : Anselme Bertrand Charles
- ♀ Bianca : Anselme Charles Bertrand
- ♀ Caroline : Bertrand Charles Anselme
- ♂ Anselme : Caroline Bianca Alice
- ♂ Bertrand : Alice Bianca Caroline
- ♂ Charles : Alice Bianca Caroline

Et si les filles invitent ?

Le quart d'heure américain

Préférences

- ♀ Alice : Anselme Bertrand Charles
- ♀ Bianca : Anselme Charles Bertrand
- ♀ Caroline : Bertrand Charles Anselme
- ♂ Anselme : Caroline Bianca Alice
- ♂ Bertrand : Alice Bianca Caroline
- ♂ Charles : Alice Bianca Caroline

1. Bianca invite Anselme

- ▶ Anselme ♡ Bianca
- ▶ Alice, Caroline, Bertrand, Charles : célibataires

Le quart d'heure américain

Préférences

- ♀ Alice : Anselme Bertrand Charles
- ♀ Bianca : Anselme Charles Bertrand
- ♀ Caroline : Bertrand Charles Anselme
- ♂ Anselme : Caroline Bianca Alice
- ♂ Bertrand : Alice Bianca Caroline
- ♂ Charles : Alice Bianca Caroline

2. Alice invite Anselme puis Bertand

- ▶ Anselme ♡♡ Bianca
- ▶ Bertand ♡♡ Alice
- ▶ Caroline, Charles : célibataires

Le quart d'heure américain

Préférences

- ♀ Alice : Anselme Bertrand Charles
- ♀ Bianca : Anselme Charles Bertrand
- ♀ Caroline : Bertrand Charles Anselme
- ♂ Anselme : Caroline Bianca Alice
- ♂ Bertrand : Alice Bianca Caroline
- ♂ Charles : Alice Bianca Caroline

3. Caroline invite Bertand puis Charles

- ▶ Anselme ♡♡ Bianca
- ▶ Bertand ♡♡ Alice
- ▶ Charles ♡♡ Caroline

Le quart d'heure américain

Préférences

- ♀ Alice : Anselme Bertrand Charles
- ♀ Bianca : Anselme Charles Bertrand
- ♀ Caroline : Bertrand Charles Anselme
- ♂ Anselme : Caroline Bianca Alice
- ♂ Bertrand : Alice Bianca Caroline
- ♂ Charles : Alice Bianca Caroline

Solution **stable** et **différente** de

- ▶ Anselme ♡♡ Caroline
- ▶ Bertand ♡♡ Alice
- ▶ Charles ♡♡ Bianca

Cycle de Kischnik

Deux solutions (mariages stables) S_1 et S_2

- ▶ f_1 est mariée à g_1 dans S_1
- ▶ g_1 est marié à f_2 dans S_2
- ▶ f_2 est mariée à g_3 dans S_1
- ▶ etc jusqu'à g_k
- ▶ On reboucle toujours sur f_1 car couplage
- ▶ Donc g_k marié à f_1 dans S_2

Si f_1 préfère g_1 à g_k alors g_1 préfère f_2 à f_1 (sinon : adultère) et f_2 préfère g_2 à g_1 etc

les filles préfèrent S_1 et les garçons S_2

Revendicateurs

Théorème des revendicateurs

- ▶ Soient S_1 et S_2 deux solutions stables différentes
- ▶ Si g est avec f_1 dans S_1 et avec f_2 dans S_2
- ▶ Et en supposant que g préfère f_1 à f_2
- ▶ Alors f_1 préfère son mari dans S_2 à g (son mari dans S_1)

dans un couple, personne ne préfère jamais la même solution

Ici g préfère S_1 et f préfère S_2

Démonstration

Construire le cycle de Kischnik

L'ensemble des solutions

Relation d'ordre

$S_1 \prec S_2$ si les garçons préfèrent S_2 et les filles S_1

Le treillis des solutions

L'ordre partiel \prec est un **treillis**

(l'ensemble des majorants d'une partie admet un plus petit élément, et celui des majorants un plus grand élément)

Gale-Shapley calcule le maximum (unique) du treillis

Célibataires

Théorème des célibataires

L'ensemble des célibataires est le même dans toutes les solutions !
En d'autres termes, être marié est une propriété intrinsèque : si quelqu'un est marié dans une solution il est marié dans toutes

Démonstration

La encore, on fait le cycle de Kischnik

Le mensonge

Mentir

Faire semblant d'avoir d'autres préférences que les "vraies"

Les garçons sont honnêtes

Les garçons n'ont aucun intérêt à mentir sur leurs préférences : puisque le bal donne la meilleure solution pour eux tous, tout menteur serait victime d'un adultère

Mais...

Les filles ont tout intérêt à mentir (collectivement) !

Intérêt du mensonge féminin

Préférences

- ♀ Alice : Anselme Bertrand Charles
- ♀ Bianca : Anselme Charles Bertrand
- ♀ Caroline : Bertrand Charles Anselme
- ♂ Anselme : Caroline Bianca Alice
- ♂ Bertrand : Alice Bianca Caroline
- ♂ Charles : Alice Bianca Caroline

Solution stable du bal

- ▶ Anselme ♡♡ Caroline
- ▶ Bertand ♡♡ Alice
- ▶ Charles ♡♡ Bianca

Intérêt du mensonge féminin

Préférences

- ♀ Alice : Anselme Bertrand Charles
- ♀ Bianca : Anselme Charles Bertrand
- ♀ Caroline : Bertrand Charles Anselme
- ♂ Anselme : Caroline Bianca Alice
- ♂ Bertrand : Alice Bianca Caroline
- ♂ Charles : Alice Bianca Caroline

Les filles veulent : (quart d'heure américain)

- ▶ Anselme ♡♡ Bianca
- ▶ Bertand ♡♡ Alice
- ▶ Charles ♡♡ Caroline

Intérêt du mensonge féminin

Mensonge sur les préférences

♀ Alice : Anselme Bertrand Charles

♀ Bianca : Anselme ~~Charles~~ Bertrand

♀ Caroline : Bertrand Charles ~~Anselme~~

♂ Anselme : Caroline Bianca Alice

♂ Bertrand : Alice Bianca Caroline

♂ Charles : Alice Bianca Caroline

unique solution stable

▶ Anselme ♥♥ Bianca

▶ Bertand ♥♥ Alice

▶ Charles ♥♥ Caroline

Comment faire un joli mensonge ?

Algorithme de mensonge

- ▶ Les filles se réunissent avant le bal
- ▶ Calculent la solution du quart d'heure américain
- ▶ Refuseront tout autre partenaire (fausse liste de préférence de longueur 1)

Riposte des garçons

Les filles doivent connaître les préférences des garçons !

Alors que dans le bal normal cette information peut être cachée.

→ Les garçons doivent se taire avant...

Lien avec le P2P ?

La théorie des mariages :

- ▶ Des hommes, des femmes
- ▶ Chacun désire le(s) meilleur(s) partenaire(s) possible(s)
- ▶ Chacun classe tous ses partenaires acceptables

Les réseaux P2P :

- ▶ Des pairs qui sont clients et serveurs
- ▶ Chacun désire un service optimal (le plus souvent)
- ▶ Le service est fourni par les autres pairs
- ▶ Un pair accepte ou non une connection proposée

Exemple

Modèle

Quatre pairs. Chacun a deux films et veut les deux autres.

Possessions et demandes des pairs

Pair1 : Alien, Batman Dracula, Emmanuelle	Pair2 : Alien, Conan Dracula, Fight Club
Pair3 : Dracula, Emmanuelle Alien, Conan	Pair4 : Dracula, Fight Club Alien, Batman

Chaque pair-client se connecte a un pair-serveur

Sans incitation directe, upload et download sont indépendants

Exemple

Possessions et demandes des pairs

Pair1 : Alien, Batman Dracula, Emmanuelle	Pair2 : Alien, Conan Dracula, Fight Club
Pair3 : Dracula, Emmanuelle Alien, Conan	Pair4 : Dracula, Fight Club Alien, Batman

Préférences :

Comme client	Comme serveur :
♂Pair1 : ♀Pair3, ♀Pair4	♀Pair1 : ♂Pair4, ♂Pair3
♂Pair2 : ♀Pair4, ♀Pair3	♀Pair2 : ♂Pair3, ♂Pair4
♂Pair3 : ♀Pair2, ♀Pair1	♀Pair3 : ♂Pair1, ♂Pair2
♂Pair4 : ♀Pair1, ♀Pair2	♀Pair4 : ♂Pair2, ♂Pair1

Exemple

Préférences :

Comme client	Comme serveur :
♂Pair1 : ♀Pair3, ♀Pair4	♀Pair1 : ♂Pair4, ♂Pair3
♂Pair2 : ♀Pair4, ♀Pair3	♀Pair2 : ♂Pair3, ♂Pair4
♂Pair3 : ♀Pair2, ♀Pair1	♀Pair3 : ♂Pair1, ♂Pair2
♂Pair4 : ♀Pair1, ♀Pair2	♀Pair4 : ♂Pair2, ♂Pair1

le client à l'initiative

Pair1 télécharge chez Pair3, Pair3 chez Pair2, Pair2 chez Pair4, Pair4 chez Pair1.

Les réseaux généreux

- ▶ eDonkey, Gnutella, KaZaa...
- ▶ Chacun est indépendamment client et serveur
- ▶ Les clients recherchent les meilleurs serveurs
- ▶ Les serveurs recherchent les meilleurs clients (à définir)
- ▶ Le graphe sous-jacent est biparti
 - ▶ Serveur \longleftrightarrow femme
 - ▶ client \longleftrightarrow homme
- ▶ Donc : théorie des mariages...
- ▶ ...ou de la polygamie (extension des mariages)

Application aux réseaux généreux

Il existe une solution stable au problème

On peut la trouver vite et de façon distribuée (requêtes)

Les clients/hommes n'ont pas intérêt à mentir

donc tout va bien !

Asymétrie

Gros avantage à l'initiative (aux hommes = aux clients)

Réseaux à incitation au téléchargement

Ce qui change dans la modélisation

- ▶ BitTorrent : corrélation entre upload et download
- ▶ Donc plus de séparation pair-client de pair-serveur
- ▶ Plus qu'un seul sexe !

Exemple

Possessions et demandes des pairs

Pair1 : Alien, Batman Dracula, Emmanuelle	Pair2 : Alien, Conan Dracula, Fight Club
Pair3 : Dracula, Emmanuelle Alien, Conan	Pair4 : Dracula, Fight Club Alien, Batman

Préférences :

Pair1 : Pair3, Pair4, Pair2 Pair3 : Pair2, Pair1, Pair4

Pair2 : Pair4, Pair3, Pair1 Pair4 : Pair1, Pair2, Pair3

Exemple

Possessions et demandes des pairs

Pair1 : Alien, Batman Dracula, Emmanuelle	Pair2 : Alien, Conan Dracula, Fight Club
Pair3 : Dracula, Emmanuelle Alien, Conan	Pair4 : Dracula, Fight Club Alien, Batman

Préférences :

Pair1 : Pair3, Pair4, Pair2 Pair3 : Pair2, Pair1, Pair4

Pair2 : Pair4, Pair3, Pair1 Pair4 : Pair1, Pair2, Pair3

Si Pair1 et Pair2 prennent l'initiative...

Connections bidirectionnelles Pair1—Pair3 et Pair2—Pair4

Exemple

Possessions et demandes des pairs

Pair1 : Alien, Batman Dracula, Emmanuelle	Pair2 : Alien, Conan Dracula, Fight Club
Pair3 : Dracula, Emmanuelle Alien, Conan	Pair4 : Dracula, Fight Club Alien, Batman

Préférences :

Pair1 : Pair3, Pair4, Pair2 Pair3 : Pair2, Pair1, Pair4

Pair2 : Pair4, Pair3, Pair1 Pair4 : Pair1, Pair2, Pair3

Mais Pair3 et Pair4 prennent l'initiative...

Connections bidirectionnelles Pair1—Pair4 et Pair2—Pair3

Exemple

Possessions et demandes des pairs

Pair1 : Alien, Batman Dracula, Emmanuelle	Pair2 : Alien, Conan Dracula, Fight Club
Pair3 : Dracula, Emmanuelle Alien, Conan	Pair4 : Dracula, Fight Club Alien, Batman

Préférences :

Pair1 : Pair3, Pair4, Pair2 Pair3 : Pair2, Pair1, Pair4

Pair2 : Pair4, Pair3, Pair1 Pair4 : Pair1, Pair2, Pair3

Observation : La configuration finale dépend de qui demande

Être le premier à proposer donne un avantage

Changements à apporter au modèle

Exemple canonique : BitTorrent

- ▶ Download accordé aux trois meilleurs uploaders
- ▶ Un download accordé au hasard
- ▶ Donc quatre sockets d'envoi seulement !
- ▶ C'est bien un problème de mariage (de 4-mariage)

Relation client/serveur symétrique

Graphe sous-jacent quelconque !

On passe des mariages aux **collocations**

Théorie des collocations stables

Version unisexe des mariages

tout le monde classe tout le monde

On forme des paires et non des couples

Une théorie plus difficile que les mariages

Pas d'existence d'une solution stable

L'algorithme du bal peut ne pas converger

Exemple instable

Préférences

Alix : Claude Dominique

Claude : Dominique Alix

Dominique : Alix Claude

Aucune colocation stable ne peut se former !

Théorie des collocations stables

Que garder des mariages stables ?

- ▶ L'algorithme du bal reste utilisable
- ▶ Mais ne converge plus vers une solution stable

Mais en P2P ?

- ▶ La stabilité n'est pas forcément utile
- ▶ Chacun cherche son meilleur ! Gale-Shapley implicite
- ▶ On peut donc faire de nouvelles choses

Modélisation du problème

Les données

- ▶ P : ensemble des pairs
- ▶ Chaque pair trie ses connaissances par ordre de préférence

Les actions

Initiatives à la Gale-Shapley : p se met avec le meilleur (pour lui) de ceux qui l'acceptent

- ▶ Initiative active : P peut changer de partenaire (sinon, inactive)
- ▶ Dépend bien sûr de la configuration courante (qui est avec qui)

Préférences acycliques : un cas simple ?

Cycle de préférences

- ▶ 1 préfère 2 à n
- ▶ 2 préfère 3 à 1
- ▶ ...
- ▶ n préfère 1 à $n-1$

Théorème

s'il n'y a pas de cycle de préférence de taille $n > 2$ alors il existe une solution unique

Préférences acycliques : un cas simple ?

Théorème

s'il n'y a pas de cycle de préférence de taille $n > 2$ alors il existe une solution unique

Démonstration

1. On crée un graphe où chaque pair/sommet pointe son pair préféré.
2. Un cycle de ce graphe est un cycle de préférences
3. Donc il existe un cycle de taille deux
4. C'est une **superpaire** : chacun est le préféré de l'autre. Un couple indissociable !
5. On enlève les deux de la superpaire
6. On se ramène à un cas acyclique plus petit et on finit par récurrence.

Atteignabilité de la solution

Théorème

Si les préférences sont acycliques, toute suite d'initiative assez longue donne la solution stable

Démonstration

On considère l'espace (gros !) des configurations, avec les transitions (initiatives) entre

- ▶ Un puits est une solution stable (unique)
- ▶ Un cycle dans ce graphe conduit à un cycle de préférences

Question ouverte

Trouver une borne supérieure du temps de convergence en nombre d'initiatives

Nombre d'initiatives minimal

Dans toute configuration stable (acyclique), il existe une transition (initiative) qui stabilise une paire de pairs.

Il suffit de prendre une superpaire

Conséquence

Toute configuration est stabilisable en $n/2$ initiative

Problème

Il faut ordonner les initiatives (algorithme centralisé)

Nombre moyen d'initiatives

Théorème

Le nombre d'initiatives aléatoires (actif ou passif) dans le cas acyclique pour parvenir à la solution stable est inférieur à

$$\frac{n^2 \log n}{4}$$

avec forte probabilité

Démonstration

- ▶ Il existe toujours une superpaire
- ▶ Borne de Chernoff (tiroir à chaussettes aléatoire) : la paire est stabilisée au bout de $(n/2) \log(n/2)$ initiatives aléatoires (avec forte proba)

Principaux cas acycliques

1. Beauté intrinsèque (préférences globales)

Il existe un classement absolu des pairs

Exemples : bande passante, stockage disponible, CPU...

2. Préférences symétriques

Les pairs se donnent des notes symétriques : $note_i(j) = note_j(i)$

Exemple : latence, niveau des joueur (jeux en ligne), affinité

3. Préférences complémentaires

Tout le monde veut les même ressources.

Exemple : blocs de fichier dans BitTorrent

$note_i(j) = \#blocs(j) - \#communs(i, j)$

Équivalence des cas acycliques

Théorème

- ▶ Les préférences acycliques
- ▶ Les préférences symétriques
- ▶ Les préférences complémentaires

sont des formalismes équivalents

Analyse

On observe facilement que les symétriques sont acycliques (la note augmente strictement le long d'un cycle donc égalité)

Réiproquement, tout système de préférence acyclique peut s'écrire avec des notes (plus dur).

Chapitre 10bis :

Simulations

des préférences acycliques

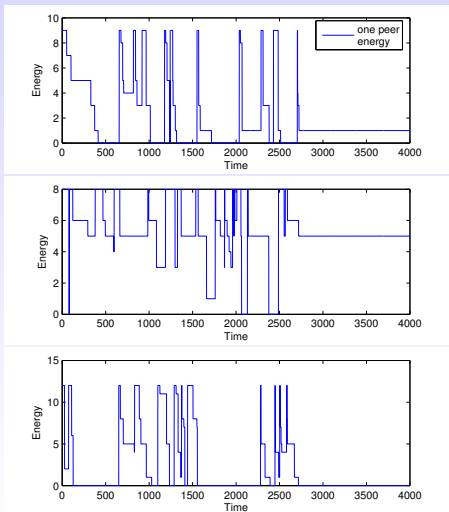
Protocole expérimental

- ▶ On crée une liste de préférences
 - ▶ Taille des listes variable ou non
 - ▶ Préférences random, universelles, symétriques...
- ▶ On décide de qui prend l'initiative
 - ▶ Random
 - ▶ Round robin
 - ▶ Proportionnel au mécontentement
 - ▶ ...
- ▶ On fait tourner
- ▶ On regarde

Que regarder ?

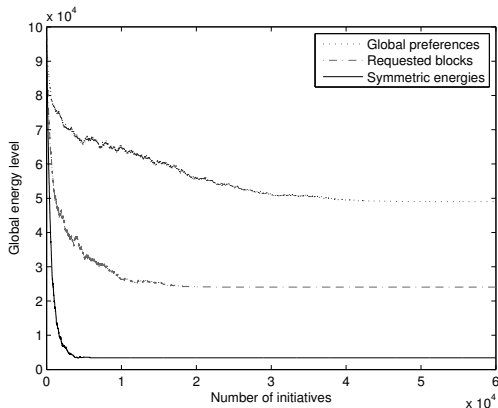
- ▶ Le classement de l'autre
- ▶ La performance brute (bande passante, ping...)
- ▶ On peut regarder à l'échelle locale (temps de célibat, distribution des niveaux d'énergie)...
- ▶ ou à l'échelle globale...

Localement : contentement de quelques pairs



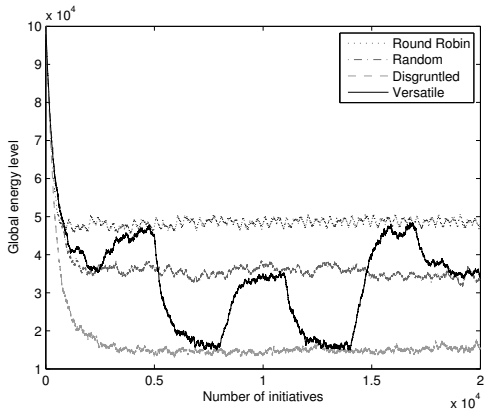
Préférences acycliques

- ▶ Initiatives : aléatoires
- ▶ Mesure : somme des contentenents
- ▶ Ça converge vers l'(unique) solution stable !



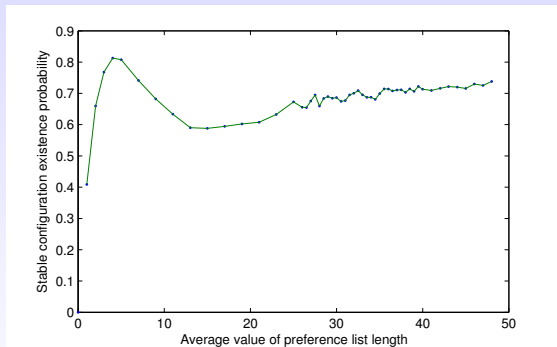
Préférences aléatoires

- ▶ On choisit des stratégies d'initiatives différentes
- ▶ Quand on change de stratégie en cours de route, on change de "niveau d'énergie"
- ▶ Ça ne converge plus
- ▶ Chaque stratégie semble avoir un espace d'états privilégié



Existence d'une solution stable ?

- ▶ Mertens a prouvé qu'en prefs. complètes, la probabilité d'existence d'une solution stable dépend du nombre de paires
- ▶ On observe qu'à nombre de paires fixé, elle dépend de la connaissance du réseau (taille moyenne de la liste)



Chapitre 10ter :

Préférences globales

Préférences globales

Tout le monde a la même liste de préférences

Il y a donc LE meilleur pair, LE deuxième etc

Exemple : BitTorrent (avec la bande passante)

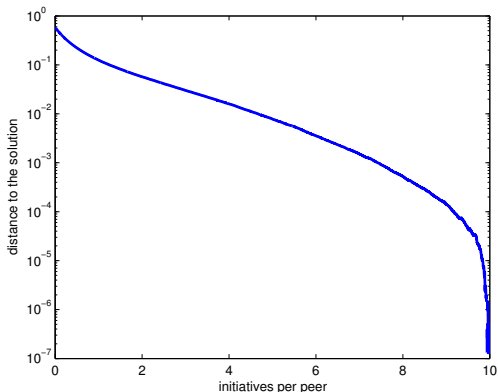
On veut montrer la stratification

les pairs échangent des données avec ceux ayant la même bande passante qu'eux

- ▶ Les puissantes (fibre) avec les puissants
- ▶ Les moyens (ADSL) avec les moyens
- ▶ Les faibles avec les faibles

Convergence empirique

- ▶ n utilisateurs
- ▶ d voisins
(connaissances d'un pair)
- ▶ donc dn initiatives avec forte proba



Préférences globales et connaissance complète

Le graphe des connaissances est une clique

On autorise plusieurs connections par pair (sinon c'est trop facile)

Résultat : des clusters ?



On a des cliques (ici de 3 car 2 connections)

La diffusion des données est compromise...

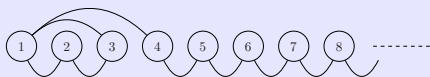
Préférences globales et connaissance complète

Le graphe des connaissances est une clique

On autorise plusieurs connections par pair (sinon c'est trop facile)

Mais c'est instable

Ici on autorise le pair 1 à avoir 3 connections

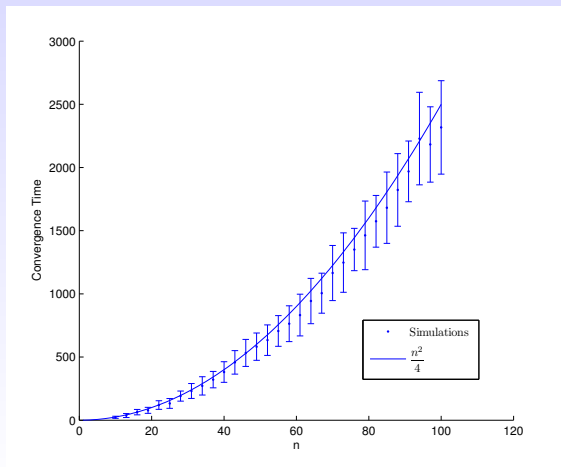


On observe la **stratification**

En général, la structure est déconnectée (clusters) ou linéaire

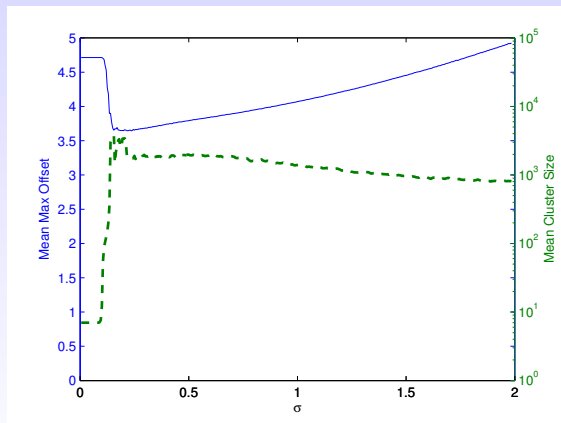
Temps de stabilisation

On ne sait pas calculer mais on conjecture :



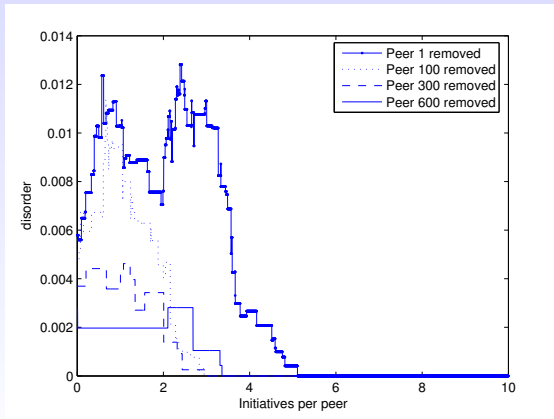
Transition de phase

- ▶ On suppose que le nombre de connections suit une loi normale
- ▶ Taille des clusters et saut moyen dépendent de la variance
- ▶ On observe une transition de phase



Temps de re-stabilisation

On enlève un pair quand c'est stable



Préférences globales mais connaissance partielle

On ne connaît pas tous les pairs

Proba p de connaissance \rightarrow graphe d'Erdős-Reyni

Ainsi le meilleur ne connaît pas forcément le deuxième donc plus de mixité sociale

BitTorrent fait ainsi

En effet le tracker donne une courte liste de pairs

Une formule exacte

- ▶ $D(i, j)$: probabilité que i soit avec j
- ▶ Pour ça, il faut que :
 - ▶ i connaisse j
 - ▶ i ne soit pas avec meilleur que j
 - ▶ j ne soit pas avec meilleur que i
- ▶ Attention, ces deux derniers évènements ne sont pas indépendants

$$D(i, j) = p \left(1 - \sum_{k=1}^{k=j-1} D(i, k) \right) \text{Prob}(\text{Pair}(j) \geq i \mid \text{Pair}(i) \geq j)$$

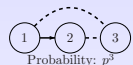
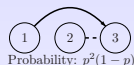
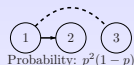
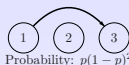
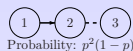
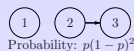
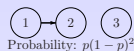
Une formule approchée

- ▶ On néglige les corrélations entre “ i a meilleur que j ” et “ j a meilleur que i ”
- ▶ On a alors une formule plus simple (calcul par récurrence)

$$D(i, j) = p \left(1 - \sum_{k=1}^{k=j-1} D(i, k) \right) \left(1 - \sum_{k=1}^{k=i-1} D(j, k) \right)$$

Erreur d'approximation

Sur trois pairs il y a 8 configurations



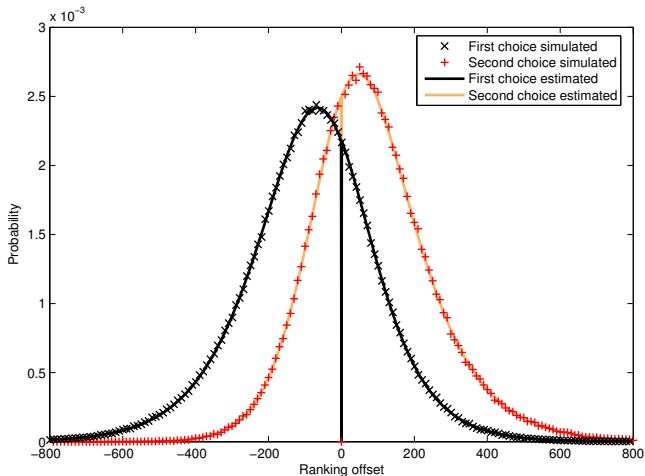
Probas exactes : $D_{exact}(1, 2) = p$; $D_{exact}(1, 3) = p(1-p)$;
 $D_{exact}(2, 3) = p(1-p)^2$

L'approximation donne

$$\begin{aligned}
 D(2, 3) &= p(1 - D(2, 1))(1 - D(3, 1)) \\
 &= p(1 - p)(1 - p(1 - p)) \\
 &= D_{exact}(2, 3) + p^3(1 - p)
 \end{aligned}$$

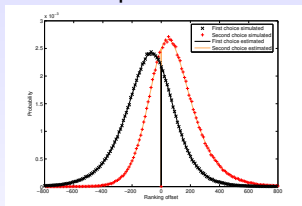
Erreur d'approximation

Elle est complètement négligeable si n grand et p petit



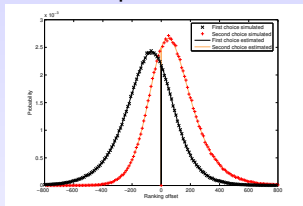
Application à BitTorrent

Pour upload et nombre de connections donné, download ?

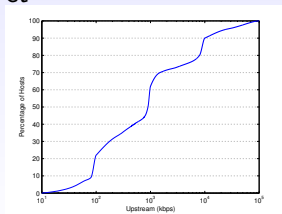


Application à BitTorrent

Pour upload et nombre de connections donné, download ?

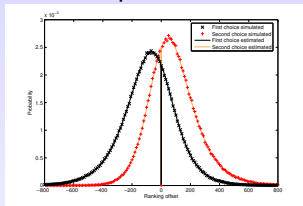


et

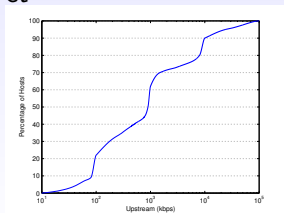


Application à BitTorrent

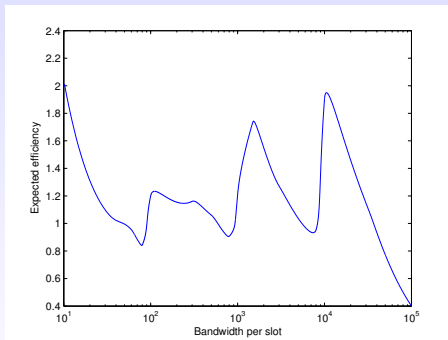
Pour upload et nombre de connections donné, download ?



et



donne



Conclusion générale

Domaine très récent, en plein essor

- ▶ Donc beaucoup de nouvelles applications
 - ▶ partage de fichiers personnels www.peerple.net
 - ▶ radio, télévision
 - ▶ Sauvergarde coopérative
 - ▶ Petites annonces, jeux, VoD etc etc...
- ▶ Et aussi de domaines de recherche en plein essor
 - ▶ DHT (recherches plus puissantes, robustesse...)
 - ▶ Anonymat
 - ▶ Incitations à la coopération (à la BitTorrent)
 - ▶ Protocoles etc etc

Stagiaires, doctorants et chercheurs recherchés !

Remerciements

Ce cours doit beaucoup à Laurent Viennot (en particulier sections sur Avalanche, Rateless et network coding, et plusieurs figures) et à Fabien Mathieu (pour la partie sur les préférences acycliques)