- TP 2 : Tables de hachage distribuées -

- Exercice 1 - Réseaux Kademia et eDonkey -

Rendez vous sur la page du projet *eMule*.

Dans la section *screenshots*, observez particulièrement les écrans *Serveurs* et *Kademia*. Dans ce dernier, on remarque les vecteurs de bits calculés correspondant à la distance XOR entre les nœuds du réseau.

- Exercice 2 - Simulation du réseau Chord -

Le but de l'exercice est de simuler un réseau pair-à-pair basé sur la table de hachage distribuée Chord (un graphe de Cayley particulier). On veut implémenter principalement 3 opérations : la recherche d'une machine, l'insertion d'une machine et la suppression d'une machine, comme détaillé dans le cours (recherche en $O(\log N)$ et insertion/suppression en $O((\log N)^2)$ en moyenne, où N est le nombre total de clés possibles.

Un code en C++ pré-rempli, et contenant notamment des routines d'affichage, est disponible à l'adresse suivante :

<http://www.lirmm.fr/~bessy/MathWeb/chord.cc>

Le réseau comporte initialement 4 machines placées aux clés 0, 2^{n-2} , 2^{n-1} et $2^{n-1} + 2^{n-2}$, le réseau comportant $2^n - 1$ clés en tout. Dans un premier temps, on prendra $n = 10$.

1. Grâce à la fonction d'affichage, vérifier que le domaine de responsabilité et les voisinages sortant et entrant de chaque machine sont corrects.
2. Ecrire une fonction `bool Localise(int cle, int deb, int fin)` qui renvoie vrai si la valeur `cle` est entre les valeurs `deb` et `fin` dans l'ordre cyclique sur le réseau Chord.
3. Ecrire la fonction de recherche d'une machine responsable d'une clé donnée : la fonction `machine* Recherche_responsable(int valeur_cle, struct machine *mach_dep)` doit renvoyer un pointeur sur la machine responsable de la clé `valeur_cle`, la recherche se faisant à partir de la machine pointée par `mach_dep` (pour permettre les appels récursifs).
4. Tester votre fonction en demandant notamment les machines responsables des clés 1, 257, 513 et 769.
5. Ecrire une fonction `void Ajout_machine(int valeur_cle)`, qui ajoute une machine de clé `valeur_cle` dans le réseau.
6. Ecrire une fonction `void Retrait_machine(int valeur_cle)`, qui retire la machine responsable de la clé `valeur_cle` du réseau. On prendra garde de conserver toujours au moins deux machines sur le réseau ainsi qu'un point d'entrée valide.