Mini-projet: Implémentation de CHORD

Enoncé

Le but de cette série de TPs est d'implémenter un système de type Chord.

On procédera par étapes

1. Mise en place du cercle initial
    a. Ajout d'un noeud : mise à jours uniquement du précédent et suivant
    b. Utilisation du cercle pour faire les fonctions `get` et `update`
2. Gestion du véritable voisinage lors des recherches
    a. Ajout d'un noeud : mise à jours des tables de voisinage en mode *cercle*
    b. Utilisation de la table de voisinage lors des get et update
3. Gestion Chord complète
    a. Ajout d'un noeud : mise à jours des tables de voisinage en utilisant les tables de voisinages inverses
    b. Utilisation de la table de voisinage lors des `get` et `update`
4. Bonus
    a. K-réplication
    b. Départ d'un noeud

Dans tous les cas, on aura une commande pour obtenir le nombre de communication moyen par type de requêtes (get, update) ainsi que pour la gestion de l'infrastructure (communications dues à l'arrivée/au départ des noeuds). Il y a donc 3 valeurs moyenne qu'il est nécessaire de pouvoir récupérer.

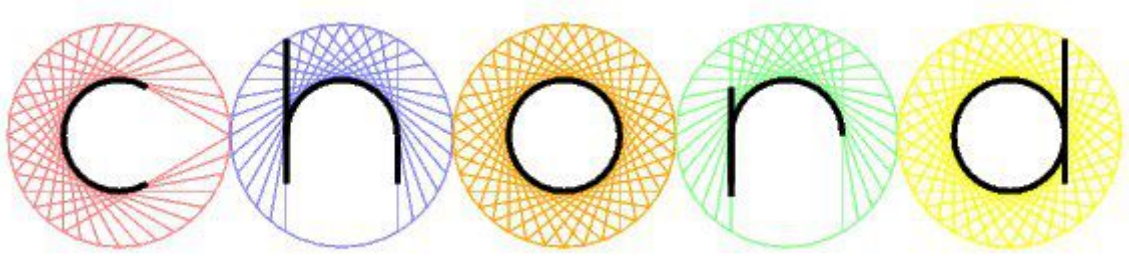
Exemple d'utilisation du code demandé

```
./chord-v1 10000
./chord-v1 12000 127.0.0.1 10000
```

Il sera nécessaire de faire un second programme capable d'envoyer les requêtes aux noeuds Chord. Ce code simulera des `get/put` sur le système Chord.

On demande aussi que des tests entre les différents codes aient lieu. Une partie de l'évaluation concerne la capacité de votre code à fonctionner avec celui des autres et donc la qualité du protocole mis en place.

On demande aussi que vous testiez la performance de votre code, en traçant les 3 graphes (`get`/`put`/gestion) des différentes versions pour un nombre croissant de noeuds (1, 2, 4, 16, 32).



Rappels

On gèrera les couples `<clé,valeur>` par une table de hachage, les clés seront des int comme les valeurs. Les clés seront entre 0 et 65535. Le noeud responsable de la valeur i sera le noeud dont la clé est égale ou plus proche supérieur (modulo 65536).
Pour la première partie, on aura le routage montré dans la figure à gauche.

Pour les seconde et troisième parties, on rappel que chaque table de voisinage est constituée comme suit: Le noeud `i` a comme voisin

- i + 65536/2 % 65536
- i + 65536/4 % 65536
- i + 65536/8 % 65536
- .
- .
- i + 1 % 65536