SCIENCES SORBONNE UNIVERSITÉ

Rapport Projet AR DHT Chord

Firas Jebari, Titouan Polit

May 9, 2022

1 VARIABLES UTILISÉES

N : Nombre de sites dans le système
M : Dans chord, les pairs auront des finger table de taille M. De plus la taille de l'anneau chord sera de 2^{M-1}

2 CALCUL DES FINGER TABLE

Dans cet exercice on s'intéresse au calcul des fingers table par les pairs. Le simulateur ne sert qu'à donner aux pairs leurs identifiants chords et leur indiquer si ils sont candidats ou non à l'élection. L'objectif est de réaliser ce calcul avec une complexité en nombre de messages sous-quadratique. Nous présentons l'algorithme que nous avons choisi de mettre en place.

2.1 Description de l'algorithme

Nous avons choisi de réaliser une élection pour qu'un leader fasse le calcul des finger table. Pour élire un leader nous utilisons l'algorithme de Hirschberg et Sinclair. Cet algorithme utilise des jetons. Tous les candidats à l'élection exécute des étapes (ou rondes) tant qu'ils n'ont pas été battus par un autre candidat. Les étapes sont numérotées de 0 à log(N). Une étape consiste en l'émission de deux jetons par le candidat. Un jeton vers son voisin de droite et un vers son voisin de gauche. Ce jeton est étiqueté de l'identifiant de l'émetteur. De la distance à laquelle il doit se propager. Cette distance étant égale à 2^i où i est l'étape dans laquelle est le pair émetteur. Mais aussi d'un intitulé, OUT ou IN en fonction de si il est sur le chemin de l'aller ou du retour. Lorsqu'un pair reçoit un jeton OUT, si il est candidat alors il regarde l'identifiant de l'émetteur que porte le jeton et se dit battu si l'identifiant est supérieur, sinon il supprime le jeton. Ensuite, il regarde la distance que porte le jeton et si elle est supérieure à 0 il la décrémente de un. Enfin, il transmet le jeton à son voisin de gauche si il l'a reçu de la droite et réciproquement. Si la distance est à 0 lors de la réception alors le pair renvoie le jeton avec l'intitulé IN par la où il est arrivé pour qu'il retourne à son émetteur. Un pair est donc élu lorsqu'il reçoit un jeton OUT portant son identifiant, autrement dit, un jeton ayant fait tout le tour de l'anneau.

Le pair ayant gagné l'élection, l'annonce en faisant passer un message dans l'anneau. Tous les pairs connaissent donc l'identifiant mpi du leader. Grâce à cela ils lui envoient un message contenant leurs identifiants chord et mpi pour que le leader puisse commencer le calcul des finger table. Le leader utilise le même algorithme de calcul des finger table que le simulateur dans l'exercice 1. Une fois le calcul des finger table terminé, le pair élu les envoient aux pairs. A partir de ce moment là, les pairs n'ont donc plus connaissance de leurs voisins gauche et droit. Ils ne connaissent que leur successeur dans l'anneau CHORD et les pairs présents dans leur finger table.

2.2 Justification de sa correction

Le simulateur garantit qu'un nombre non nul de pairs seront candidats à l'élection. De plus l'algorithme de Hirschberg et Sinclair garantit l'élection d'un des candidats. On est donc assuré, d'avoir à la fin de la phase d'élection un leader. Grâce à l'annonce, ce leader connaîtra les identifiants de tous les pairs et sera donc en mesure de calculer leurs finger table. Les communications étant fiables on est sûr que les pairs recevront bien leurs finger table. L'algorithme de calcul des finger table de manière distribuée est donc correct.

2.3 Justification de sa complexité

L'algorithme de Hirschberg et Sinclair a une complexité pire cas en N*log(N) messages. N étant le nombre de processus et log(N) étant le nombre d'étapes maximum exécutées. Ensuite, l'annonce par le leader coûte N messages puisque c'est une annonce qui parcourt tout l'anneau. L'annonce des identifiants chord et mpi par les pairs coûte elle N-1 messages puisque le processus élu n'en envoie pas. Après le calcul des finger table le processus élu envoie N messages. Il s'envoie à lui-même sa finger table, c'est un choix que nous avons fait pour que tous les pairs aient le même code d'initialisation. Il n'y a plus de leader lorsque l'on rentre dans la phase CHORD donc ce choix nous semblait judicieux. On a donc N*log(N) + N + (N-1) + N messages dans le pire cas ce qui nous ramène à une complexité sous-quadratique.

3 INSERTION D'UN PAIR

Dans cet exercice, on s'intéresse à l'insertion d'un pair lorsque la DHT est dans un état global cohérent. On cherche à réaliser cette insertion avec une complexité en messages linéaire tout en garantissant que la DHT retrouvera un état global cohérent après l'insertion.

3.1 Description de l'algorithme

Comme dans l'exercice 1, c'est le simulateur qui sera chargé de calculer les finger tables. Pour réaliser l'insertion nous aurons besoin de tables reverse. Dans la table reverse d'un pair p1 on retrouve les identifiants chord et mpi de tous les pairs p qui ont un finger qui pointent vers p1. Ces tables reverse seront elles aussi calculées par le simulateur. Pour se faire le simulateur calculera d'abord les finger tables puis les parcourera pour remplir les tables reverse des pairs. Nous avons choisi de donner une taille fixe de N-1 à ces reverse table, cela simplifie grandement leur envoi aux pairs. Les cases non utilisées dans les reverse table seront remplies par des -1.
Après avoir envoyé les identifiants chord, les finger table et les reverse table aux pairs, le simulateur tire un identifiant chord aléatoire non utilisé. Par convention on affecte ce chord ID aléatoire au processus de rang : NBSITE -1. Il associe à ce nouveau pair un contact dans l'anneau. Le nouveau pair ne possède donc qu'un identifiant chord, un identifiant mpi et un contact dans la DHT pour s'insérer.

Le nouveau pair sera responsable d'un sous ensemble des clés de son responsable. Pour cela il doit donc savoir qui est son responsable et de quel ensemble de clés celui-ci détient la responsabilité. Dans un premier temps le nouveau pair envoie ses identifiants à son contact dans la DHT en lui demandant de faire un lookup pour trouver qui est son responsable. Le contact exécute donc le lookup et renvoie au nouveau pair son responsable. Les seules finger table qui seront peut-être amenées à être modifiées par cette insertion sont celles qui avaient un finger qui pointaient sur le responsable du nouveau pair. Le nouveau pair envoie donc un message à son responsable en lui indiquant qu'il faut que les pairs de sa reverse table recalculent leur finger table. Dans ce message est indiqué l'identifiant chord du nouveau pair. Le calcul des finger table par les pairs se déroule de la manière suivante. Ils ont, avant le recalcul, la connaissance des M pairs de leur finger table. A ceci on ajoute le nouveau pair dont l'identifiant chord était dans le message d'invitation au recalcul. Cette liste d'identifiants leur est suffisante pour recalculer leur finger table car les seules modifications possibles sont dans les cases où l'identifiant du responsable du nouveau pair était inscrit. Les pairs ont donc leur finger table à jour. Nous ne nous sommes pas occupé de recalculer les reverse table, il y a une solution mais qui ferait exploser la complexité en nombre de messages. Il faudrait que chaque pair qui a observé un changement dans sa finger table le fasse savoir au pair concerné en lui envoyant un message.
Le nouveau pair doit lui aussi calculer sa finger table. Pour se faire il doit connaître les identifiants chord des pairs présents dans l'anneau. Il lance donc un message qui va faire le tour de l'anneau où les pairs vont rentrer leur identifiant. Le message lui revient et il peut donc calculer sa finger table. Après cela le pair envoie au simulateur que son insertion s'est bien passée.

3.2 Justification de sa correction

Le pair nouvellement inséré pourra calculer sa finger table sans fautes puisqu'il connaît les identifiants chord de tous les pairs.
Du côté des pairs présents dans la reverse table du responsable du nouveau pair. Eux ne peuvent pas observer d'autres changements que la suppression du responsable ou l'ajout du nouveau pair dans leur finger table. Ils n'auront donc besoin que des identifiants chord de leur finger table actuelle et de l'identifiant chord du nouveau pair pour garantir la véracité du calcul de leur nouvelle finger table.
Les autres pairs, ceux qui ne sont pas dans la table reverse du responsable du nouveau pair eux n'ont pas besoin de la recalculer puisque leur finger ne pointe pas sur la zone ou il y a eu des changements de propriétés.

3.3 Justification de sa complexité

Lorsque le nouveau pair indique a son contact qu'il veut connaître son responsable on a 1 message plus log(N) qui correspond à la complexité de lookup. Ensuite le nouveau pair demande au responsable d'indiquer aux pairs de sa reverse table de recalculer leur finger table, ce qui entraîne dans le pire cas l'envoi de N messages. Enfin le message du nouveau pair qui fait le tour de l'anneau pour connaître les identifiants des pairs entraîne N messages à nouveau. On a donc log(N) + N + N ce qui nous donne O(log(N) + N), une complexité sous linéaire.