SCIENCES SORBONNE UNIVERSITÉ

AR (4I403) - TD CHORD

CORRECTION

Exercice(s)

Exercice 1 – Ordre cyclique

Soit $\mathbb{Z}$, l'ensemble des entiers relatifs. Soit K un entier strictement positif. On dit que a et b sont *congruents modulo K*, noté $a \equiv b[K]$, si et seulement si il existe λ dans $\mathbb{Z}$ tel que $b = a + \lambda K$. On note $\overline{a}$ l'unique entier dans $[0, K[$ tel que $\overline{a} \equiv a[K]$. Enfin, $d_K(a,b) = min(\overline{a-b}, \overline{b-a})$ est appelée la `distance` entre a et b.

Question 1

Quelle est la distance maximum entre deux entiers a et b, $a, b \in [0, K[$?

Solution:

$\forall a, b \in [0, K[,\ d_K(a,b) \leq \lfloor \frac{K}{2} \rfloor$.

Fin solution

Question 2

Soit a et b dans $[0, K[$. Selon les valeurs de a et de b, exprimer en extension l'ensemble des valeurs de $[a, b[$?

Solution:

$$[a,b[= \begin{cases} \{a, a+1, \ldots, b-1\} \text{ if } a < b \\ \{a, a+1, \ldots, K-1, 0, 1, \ldots, b-1\} \text{ if } a \geq b \end{cases}$$

Fin solution

Question 3

Ecrire la fonction booléenne $app(k,a,b)$ qui vérifie que $k \in [a, b[$.

Solution:

```
int app(int k, int a, int b) {
    if (a < b) return ((k >= a) && (k < b));
    return (((k >= 0) && (k < b)) || ((k >= a) && (k < N)));
}
```

Ou format macro :

```
#define app(k,a,b) ((a)<(b))?((k)>=(a) && (k)<(b)) : \
   ((((k)>=0) && ((k)<(b))) || (((k)>=(a)) && ((k)<N)))
```

Fin solution

Dans la suite, on défini la relation d'*ordre cyclique* notée $\preceq$ sur $[0..K-1]$ de la manière suivante :

$$a \preceq b \text{ ssi } 0 \leq \overline{b-a} \leq \lfloor \frac{K}{2} \rfloor$$

Exercice 2 – DHT

On considère une DHT de type *Chord* pouvant stocker entre 0 et 2^K clés distinctes numérotés de 0 à $2^K - 1$. Une ressource du système est soit une clé, soit un pair. Toutes les resources du système pair-à-pair sont indexées via la DHT. Par simplification, on suppose que la clé d'un pair p_i est égale à i. Tous les calculs dans $[0..2^K - 1]$ sont fait modulo 2^K. Par abus de notation, la relation d'ordre $\leq$ définie sur $\mathbb{Z}$ pourra êrte utilisée en lieu et place de la relation d'ordre cyclique $\preceq$. C_p est un ensemble dans lequel un pair p stocke des clés dont il a la responsabilité. Chaque clé k est stockée sur le plus petit pair p_i tel que $k \preceq i$.

On suppose que la DHT regroupe au moins 2 pairs. Le *successeur* d'un pair p_i est le pair p_j tel que j est la plus petite clé *strictement* supérieur à i. Chaque pair p_i a une table de routage des clés $i + 2^k$, $k \in [0..K-1]$.

Question 1

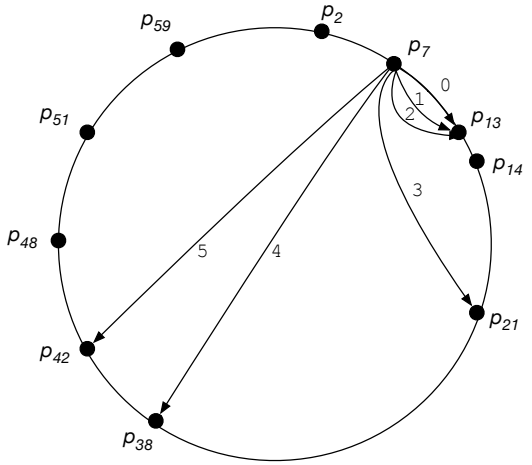
Avec ce type d'overlay, quelle sera la complexité dans le pire des cas d'une requête couvrant un intervalle de valeurs, par exemple, *Trouver l'ensemble des clés dont le numéro est supérieur à 34* ? Justifiez votre réponse.

Solution:

Avec ce type d'overlay, le mécanisme de routage n'est utile que pour rechercher des valeurs exactes. Il faut donc faire une recherche en utilisant la structure d'anneau, c'est-à-dire en allant de successeur en successeur. La complexité d'un telle recherche est donc de $O(n)$ (n étant le nombre de pairs dans le système).

Fin solution

Dans la suite de l'exercice, on supposera que $K = 6$ et que le système pair à pair est constitué des 10 pairs suivants : p_2, p_7, p_{13}, p_{14}, p_{21}, p_{38}, p_{42}, p_{48}, p_{51} et p_{59}. La figure ci-dessous représente le système avec la table de routage de p_7 (les étiquettes sur les arcs représentent leurs index respectifs dans la table de routage).



Question 2

Sur le modèle de la table de p_7 ci-dessous, donnez les tables de routage des pairs p_{51} et p_{21} . Il est conseillé de suivre le modèle de réponse suivant :

Table de p_7		
i	$(7+2^i) \mod 2^K$	j
0	8	13
1	9	13
2	11	13
3	15	21
4	23	38
5	39	42

Solution:
Je donne toutes les tables, ça peut servir...

Table de p_2		
i	$2+2^i$	j
0	3	7
1	4	7
2	6	7
3	10	13
4	18	21
5	34	38

Table de p_{13}		
i	$13+2^i$	j
0	14	14
1	15	21
2	17	21
3	21	21
4	29	38
5	45	48

Table de p_{14}		
i	$14+2^i$	j
0	15	21
1	16	21
2	18	21
3	22	38
4	30	38
5	46	48

Table de p_{21}		
i	$21+2^i$	j
0	22	38
1	23	38
2	25	38
3	29	38
4	37	38
5	53	59

Table de p_{38}		
i	$38+2^i$	j
0	39	42
1	40	42
2	42	42
3	46	48
4	54	59
5	$70[K]=6$	7

Table de p_{42}		
i	$42+2^i$	j
0	43	48
1	44	48
2	46	48
3	50	51
4	58	59
5	$74[K]=10$	13

Table de p_{48}		
i	$48+2^i$	j
0	49	51
1	50	51
2	52	59
3	56	59
4	0	2
5	16	21

Table de p_{51}		
i	$51+2^i$	j
0	52	59
1	53	59
2	55	59
3	59	2
4	3	7
5	19	21

Table de p_{59}		
i	$59+2^i$	j
0	60	2
1	61	2
2	63	2
3	3	7
4	11	13
5	27	38

Fin solution

Le principe de la recherche d'un élément de clé k par un pair p_i est le suivant :
- Soit j la plus grande clé de la finger table de p_i tel que $k \in]j;i]$.
- Si j existe, la requête de recherche de k est transmise au pair p_j qui répète récursivement.
- Si j n'existe pas, le successeur de p_i est responsable de k et p_i lui demande directement si k est présent.

Question 3

Décrire les différentes étapes de l'algorithme lorsque le pair p_7 recherche la clé 30.

Solution:
p_7 recherche dans la table $finger[]$ le plus grand pair j tel que $30 \in [j, 7[$: c'est p_{21}. p_7 transmet la requête à p_{21}.

A la réception du message, p_{21} recherche dans la table $finger[]$ le plus grand pair j tel que $30 \in [j, 21[$: il n'en trouve pas. Donc, demande directe au successeur, p_{38}.

Ce dernier cherche 30 dans C_{38} et notifie le résultat à p_7.

Fin solution

Question 4

Même question pour les cas suivants :

- p_7 recherche la clé 0 ;
- p_7 recherche la clé 10 ;
- p_{51} recherche la clé 50 ;
- p_{51} recherche la clé 22.

Solution:

- p_7 recherche dans la table $finger[]$ le plus grand pair j tel que $0 \in [j, 7[$: c'est p_{42}. p_7 transmet la requête à p_{42}.
  A la réception du message, p_{42} recherche dans la table $finger[]$ le plus grand pair j tel que $0 \in [j, 42[$: c'est p_{59}. p_{42} transmet la requête à p_{59}.
  A la réception du message, p_{59} recherche dans la table $finger[]$ le plus grand pair j tel que $0 \in [j, 59[$: il n'en trouve pas. Donc, demande directe au successeur, p_2.
  Ce dernier cherche 0 dans C_2 et notifie le résultat à p_7.
- p_7 recherche dans la table $finger[]$ le plus grand pair j tel que $10 \in [j, 7[$: il n'en trouve pas. Donc, demande directe au successeur, p_{13}.
  Ce dernier cherche 10 dans C_2 et notifie le résultat à p_7.
- p_{51} recherche dans la table $finger[]$ le plus grand pair j tel que $50 \in [j, 51[$: c'est p_{21}. p_{51} transmet la requête à p_{21}.
  A la réception du message, p_{21} recherche dans la table $finger[]$ le plus grand pair j tel que $50 \in [j, 21[$: c'est p_{38}. p_{21} transmet la requête à p_{38}.
  A la réception du message, p_{38} recherche dans la table $finger[]$ le plus grand pair j tel que $50 \in [j, 38[$: c'est p_{48}. p_{38} transmet la requête à p_{48}.
  A la réception du message, p_{48} recherche dans la table $finger[]$ le plus grand pair j tel que $50 \in [j, 48[$: il n'en trouve pas. Donc, demande directe au successeur, p_{51}.
  Ce dernier cherche 50 dans C_{51} et notifie le résultat à p_{51}, i.e., lui-même.
- p_{51} recherche dans la table $finger[]$ le plus grand pair j tel que $22 \in [j, 51[$: c'est p_{21}. p_{51} transmet la requête à p_{21}.
  A la réception du message, p_{21} recherche dans la table $finger[]$ le plus grand pair j tel que $22 \in [j, 21[$: il n'en trouve pas. Donc, demande directe au successeur, p_{38}.
  Ce dernier cherche 22 dans C_{51} et notifie le résultat à p_{51}.

Fin solution

Question 5

Que se passe t-il si la clé recherchée n'est pas dans la DHT ?

Solution:

Idem, mais la fonction de recherche doit renvoyer une valeur spéciale pour dire que la clé n'existe pas, par ex. -1 ou `nil`.

Fin solution

Question 6

Ecrire une fonction booléenne $lookup(k)$ qui permet à un pair de tester si k appartient à la DHT ou non.

Solution:

Fin solution

FIGURE 1 – Algorithm for Process p

Input :
P : set of pairs
$\forall i \in [0..m-1], finger_p[i] \in [0..2^m-1]$: finger table
C_p : set of keys stored by p

Variables :
$next$, q : CHORD identifiers of pair

Routine :
$send(m)$ to q : send the message m to the pair of CHORD identifier q

Function $initiate\_lookup(k \in [0 \dots 2^m-1])$
begin
$lookup(k, p)$
end

Function $lookup(k \in [0 \dots 2^m-1], initiator \in P)$
begin
$next = findnext(k)$
if $next =$ `nil`
then send $($`lastchance`$, initiator, k)$ to $finger[0]$
else send $($`lookup`$, initiator, k)$ to $next$
endif
end

Upon receipt of $($`lookup`$, initiator, k)$ **from** q **do**
begin
$lookup(k, initiator)$
end

Upon receipt of $($`lastchance`$, initiator, k)$ **from** q **do**
begin
if $k \in C_p$
then send $($`succ`$, p)$ to $initiator$
else send $($`succ`$,$ `nil`$)$ to $initiator$
end if
end

Upon receipt of $($`succ`$, id)$ **from** q **do**
begin
return id
end

Function $findnext(k \in [0 \dots 2^m-1])$
begin
for $i = m-1$ **to** 0 **step** -1 do
if $k \in]finger_p[i]; p]$
then return $finger_p[i]$
end if
end for
return `nil`
end