Rapport de TP Lamport

ADAM Baptiste
JOBE Ophélie

Sommaire

ADAM Baptiste
JOBE Ophélie

Introduction

Lorsque que généralement l'on veut créer un jeu vidéo ou toutes autre applications exigeant plusieurs ordinateurs à coordonner leurs actions, la meilleur solution pour cela est de se tourner vers les horloges logiques scalaires et vectorielles. Ces horloges permettent de vérifier la cohérence d'un coupon envoyé par l'un des ordinateurs ou encore de détecter la violation de la délivrance d'un message d'un ordinateur, privant les autres ordinateurs de leur possibilité d'envoi.

Dans le cadre de notre enseignement, ce TP a pour but de nous faire comprendre le fonctionnement ces horloges logique scalaire et matricielle. Nous verrons tout d'abord comment le réseau entre plusieurs ordinateur se crée et s'organise puis comment fonctionne l'algorithme de Lamport qui fait parmi des algorithme permettant de résoudre ce problème

ADAM Baptiste
JOBE Ophélie

Programme de base

Avant de pouvoir implémenter l'algorithme de Lamport cité plus haut, il nous faut tout d'abord réussir à faire entrer en contact les 3 ordinateurs afin qu'ils puissent communiquer facilement leurs demandes et réponses voulu dans notre exercice.

C'est ce que fait le programme qui nous est donné. Il commence par expliciter le port qui sera utilisé par ce serveur en fonction de sa position dans la liste passé en paramètre. Ensuite, il crée et configure la socket qui nous permettra de recevoir les messages en mode TCP.

```
155   /*CREATION&BINDING DE LA SOCKET DE CE SITE*/
156   PortBase=atoi(argv[1])+GetSitePos(NSites, argv);
157   printf("Numero de port de ce site %d\n",PortBase);
158
159   sock_add.sin_family = AF_INET;
160   sock_add.sin_addr.s_addr= htons(INADDR_ANY);
161   sock_add.sin_port = htons(PortBase);
162
163   if ( (s_ecoute=socket(AF_INET, SOCK_STREAM,0))==-1) {
164     perror("Creation socket");
165     exit(-1);
166   }
167
168   if ( bind(s_ecoute,(struct sockaddr*) &sock_add, \
169       sizeof(struct sockaddr_in))==-1) {
170     perror("Bind socket");
171     exit(-1);
172   }
173
174   listen(s_ecoute,30);
175   /*----La socket est maintenant cre'e'e, binde'e et listen----*/
```

L'étape suivante est de se synchroniser avec les autres serveurs. Le serveur en première position des paramètres est considéré comme le serveur principal et attends donc que les autres serveurs se manifestent, une fois qu'il a reçu un message de tous les serveurs, il leur renvoie une réponse.

Si le serveur en question n'est pas le serveur principal, il commence par se manifester auprès de celui ci puis attends sa réponse.

Une fois que tout le monde est synchronisé avec tout le monde, l'écoute peut commencer.

La socket *s_ecoute* est passé en non bloquant pour ne pas avoir attendre de recevoir un message pour pouvoir passer à la suite à la ligne 199.

Maintenant, le serveur fera 2 choses en boucle : la première sera de toujours rester à l'écoute d'un potentiel message donné par un client externe afin de l'afficher et la deuxième est d'écrire en continuer dans le terminal une succession de point afin que l'on sache qu'il fonctionne en continu.

ADAM Baptiste
JOBE Ophélie

Implémentation du TP

A - Algorithme de Lamport

Maintenant que nous avons compris comment marche le réseau à utiliser il faut maintenant implémenter l'algorithme de Lamport. Cet algorithme indique que la distribution de priorité entre les différents ordinateurs se fait par une file d'attente. en effet pour cela l'algorithme définit 3 types de messages que les ordinateurs s'envoient entre eux :

- Requête : envoyé aux autres ordinateurs quand le site veut passer en section critique et envoyé son message
- Réponse : envoyé au autres ordinateurs quand ils demandent une requête pour passer en section critique
- Libération : envoyé aux autres ordinateurs afin de leur faire savoir que sa section critique est terminé et que la prochaine section critique peut être donnée

Un message complet est sous cette forme :
[type] [position du serveur dans les paramètres] [estampille au moment de l'envoi]

Afin de pouvoir implémenter cette algorithme il faut donc définir 3 tableaux :

- Estampille[ ] servira à retenir l'horloge logique du serveur afin de pouvoir exprimer correctement les demandes de section critiques.
- Listei[ ] sera, comme son nom l'indique, une liste des demandes de sections critiques des différentes machines. l'indice du tableau correspond à l'indice du serveur. La valeur de la case est l'estampille à laquelle le serveur correspondant a fait sa requête de passage en section Critique. La notion de priorité est donc faite sur les valeurs du tableau (une valeur plus petite est prioritaire sur les autres) et sur les indices lorsque les valeurs sont identiques (l'indice 0 est prioritaire sur l'indice 3).
- Accord[ ] servira à retenir les accords obtenus pour la demande de section critique de la machine courante

Ainsi, avec ce système, nous retrouvons un système de distribution de priorité qui marche en réseau sur plusieurs machines différentes. Les demandes de section critiques et d'accord devrait se faire normalement entre les différentes machines et lorsqu'un machine à tous les accords elle vérifiera dans sa liste que sa requête est prioritaire pour savoir si elle doit effectuer sa section critique. Sinon elle attendra la libération d'un autre ordinateur.

B - Programme final

1. Recevoir un message

Chronologiquement dans la boucle, la première chose effectuée est la vérification de la réception d'un message. Avant le serveur se contentait de l'afficher. Maintenant il faut le traiter pour effectuer les actions correspondante.

En tout premier lieu, il faut scinder les informations du message, pour cela, nous utilisons la fonction *text2tabseq*. C'est une fonction qui nous vient d'un autre TP et que nous utilisons régulièrement, elle permet de scinder une chaîne de caractère selon un caractère passé en paramètre, en l'occurrence " " (espace).

Il faut maintenant faire les actions correspondantes au type de la requête envoyé.

Si c'est une *"requete"*, il faut stocker l'estampille de cette requête dans le tableau *listei* à l'indice du serveur qui a fait la requête (ligne 329). il faut aussi mettre à jour son estampille dans notre horloge logique (ligne 331). Nous affichons ensuite l'état du serveur (son horloge, sa listei et ses accords) pour pouvoir voir l'évolution dans le terminal.

L'étape suivante est d'incrémenter noter estampille pour passer à l'action suivante, à savoir envoyer la réponse. Nous créons donc la réponse avec le type (*"reponse"*), notre position dans les paramètres et notre estampille actuelle (ligne 339). Nous envoyons ce message grâce à la position du serveur que nous avons récupéré dans le message que nous traitons actuellement. Enfin, nous affichons les informations d'envoie pour pouvoir garder une trace de cet envoi dans le terminal.

Si c'est une *"reponse"*, deux possibilités s'offrent à nous :

- Nous avons effectivement fait une requête de passage en SC (*requeteSC = 1*), nous enregistrons donc l'accord dans notre tableau accord et nous mettons à jour l'estampille du serveur qui vient de nous répondre dans notre horloge.
- Nous n'avons pas fait de requête de passage en SC (*requeteSC = 0*), dans quel cas nous n'aurions pas dû recevoir un message de type *"reponse",* ce qui signifie qu'il y a un problème quelque part.

```
343         else{
344           if(!strcmp(tabMessage[0], "reponse")){
345             if(requeteSC == 1){
346               //je veux passer en SC, je viens de recevoir un accord
347               accords[atoi(tabMessage[1])] = 1;
348               //mettre a jour l'horloge logique (estampille)
349               estampille[atoi(tabMessage[1])] = atoi(tabMessage[2]);
350             }
351             else{
352               //je ne veux pas passer en SC, donc je ne suis pas sencé recevoir une réponse. Ce code est cassé
353               printf("erreur envoie de reponse\n");
354               exit(-1);
355             }
356           }
```

Enfin, si c'est une *"liberation"*, nous devons enlever la requête du serveur qui a envoyé ce message, pour cela, nous passons la valeur a INT_MAX. Cette valeur, qui vaut 2147483647, nous assurent qu'elle ne sera jamais la valeur minimum du tableau a tort. Nous mettons ensuite à jour l'estampille du serveur qui à envoyé ce message dans notre horloge.

Quelque soit le type du message, une fois qu'il est traité, nous affichons l'état du serveur dans le terminal pour voir l'évolution et nous incrémentons notre estampille pour passer à l'action suivante.

2. Faire une action

Pour faire une action aléatoire, nous créons un nombre random dont nous gardons le reste de la division entière par 100 pour garder un nombre entre 0 et 100.

Pour une valeur entre 0 et 10, le serveur veux passer en Section Critique. Il faut cependant vérifier que nous n'avons pas déjà fait une requête de passage en SC. Si c'est le cas, nous ne faisons rien. Sinon, nous faisons les actions correspondantes.

En premier lieu, il faut passer à l'état "requête en cours" qui correspond à requeteSC=1. Il faut aussi nous donner notre propre accord (ligne 383) et ajouter notre requête dans notre *listei* (ligne 385). Il faut aussi envoyer un message de type "requete" aux autres serveurs. C'est que nous faisons dans la boucle for ligne 387. il faut malgré tout ne pas s'envoyer de requête à soi même, d'où la vérification lignes 389 et 390.

```
371
372     /*************************************/
373     /*       choisir l'action a faire       */
374     /*************************************/
375     int random = rand()%100;
376     if(random < 10){
377       //je veux rentrer en SC : envoyer une requetes aux deux autres
378       //  si j'ai deja une requete de faite, ne rien faire
379       if(requeteSC == 0){
380         printf("\n******* je veux entrer en SC  *******\n");
381         requeteSC = 1;
382         //me donner mon accord
383         accords[GetSitePos(NSites, argv)] = 1;
384         //ajouter ma requete a ma liste
385         listei[GetSitePos(NSites, argv)] = estampille[GetSitePos(NSites, argv)];
386         //envoyer ma requete aux autres sites
387         for(int i = 0 ; i<NSites ; i++){
388           if(i == GetSitePos(NSites, argv))
389             continue;
390           sprintf(reponse, "requete %d %d ",GetSitePos(NSites, argv), estampille[GetSitePos(NSites, argv)]);
391           SendMessage(argv[2+i], atoi(argv[1])+i, reponse);
392           printSend(reponse, "requete", i, estampille[GetSitePos(NSites, argv)]);
393         }
394         printf("--- Estampille : (%d,%d,%d)\n",estampille[0], estampille[1], estampille[2]);
395         printf("--- Listei : (%d,%d,%d)\n",listei[0], listei[1], listei[2]);
396         printf("--- Accords : (%d,%d,%d)\n",accords[0], accords[1], accords[2]);
397
398         estampille[GetSitePos(NSites, argv)]++; //incrémenter l'horloge car envoie (d'une requete)
399       }
400     }
```

Enfin, nous affichons l'état du serveur dans le terminal et nous incrémentons notre estampille pour passer à l'action suivante.

Pour une valeur entre 10 et 20, le serveur veux sortir de Section Critique. Encore une fois, il faut d'abord vérifier que nous somme bien en SC, ce qui correspond a SC=1. Si nous n'étions pas en SC, nous ne faisons rien, sinon nous faisons les actions correspondantes.

Pour signifier que nous ne somme plus en SC, nous passons la valeur de SC à 0. Nous passons aussi la valeur de requeteSC à 0 puisqu'il nous est maintenant de nouveau possible de faire une requête de passage en SC.

Il faut ensuite vider nos accords en repassant toutes les valeurs du tableau à 0 (lignes 409 et 410), enlever notre requête de notre *listei* en passant la valeur à INT_MAX et envoyer un message du type *"liberation"* à tous les autres serveurs. Nous affichons aussi dans le terminal que nous sortons de SC.

Enfin, nous affichons aussi l'état du serveur dans le terminal et nous incrementons notre estampille pour passer à l'action suivante.

Pour une valeur au dessus de 20, nous ne faisons rien.

3. Vérification de l'entrée en SC

La dernière étape d'un tour de boucle est de vérifier son toutes les conditions sont réunies pour passer en SC. Il faut donc vérifier que nous avons tous les accords. La variable *accord* nous donne l'état de nos accords, 1 si nous les avons tous, 0 sinon. Nous parcourons donc notre tableau d'accord et passons *accord* a 0 si nous trouvons un accord non donné.

Il faut ensuite vérifier que notre requête est bien prioritaire sur les autres.Nous parcourons donc le tableau *listei* et mémorisons l'indice de la valeur minimale qui correspond à la requête la plus ancienne.

Si nous avons tous nos accords (*accord=1*), que l'indice de la requête la plus ancienne est le même que la position de notre serveur, que nous ne somme pas encore en Section critique (*SC=0*) et que nous avons bien une requête de passage en SC en cours (*requeteSC=1*), alors nous pouvons passer en Section Critique.

Nous passons SC à 1, nous affichons dans le terminal que nous entrons en SC et nous incrementons notre estampille pour passer à l'action suivante.

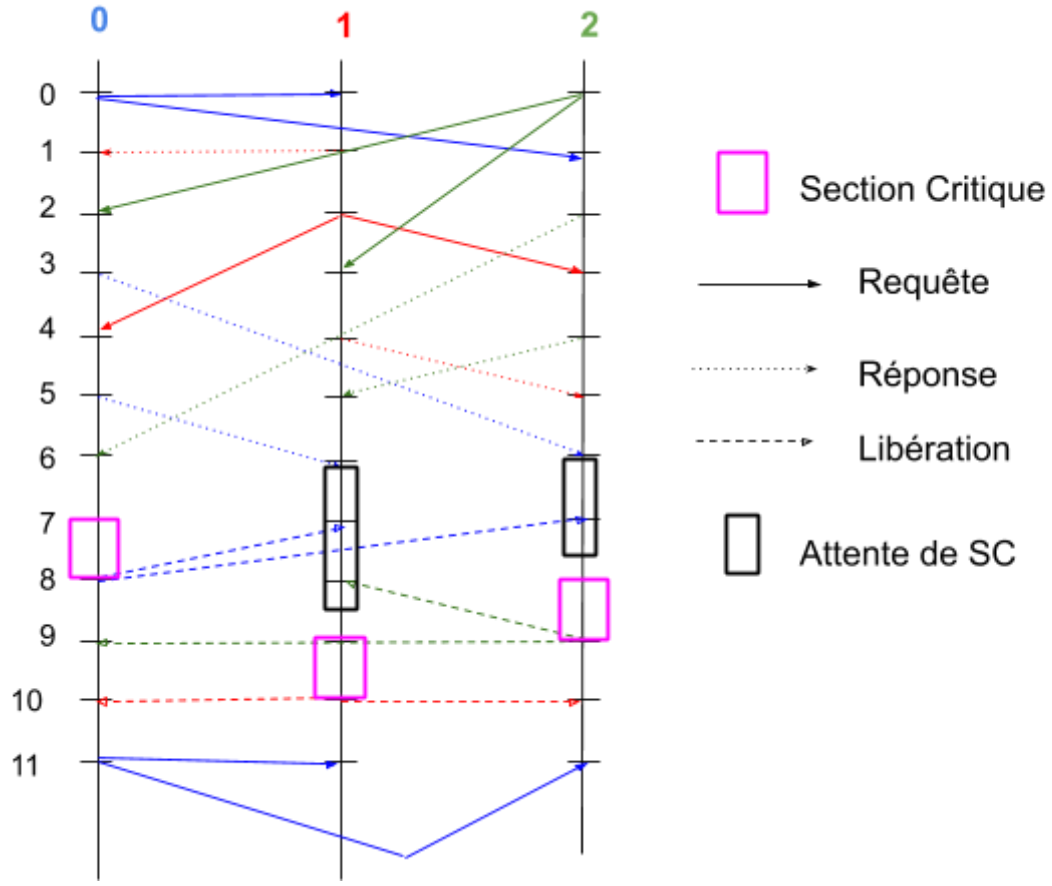
```
432     /*****************************************/
433     /*       rentrer en SC si c'est possible      */
434     /*****************************************/
435     //verifier que j'ai tous les accords
436     accord = 1;
437     for(int i = 0 ; i < NSites ; i++){
438       if(accords[i] == 0)
439         accord = 0;
440     }
441
442     indice = 0;
443     //verifier que ma requete est prioritaire
444     for(int i = 1 ; i < NSites ; i++){
445       if(listei[i] < listei[indice])
446         indice = i;
447     }
448
449     if((accord == 1) && (indice == GetSitePos(NSites, argv)) && (SC == 0) && (requeteSC == 1)){
450       //PASSAGE EN SC
451       SC = 1;
452       printf("\n**********\n********** Passage en SC **********\n**********
453       estampille[GetSitePos(NSites, argv)]++; //incrémenter l'horloge car action de passage en SC
454     }
```

La toute dernière action de la boucle est l'affiche du fameux point pour signifier que le serveur est en fonctionnement et un sleep pour allonger le temps entre deux itérations de boucle pour garder le flux du terminal intelligible .

ADAM Baptiste
JOBE Ophélie

4. Analyse

Voici le schéma des interactions entre 3 serveur basé sur une simulation que nous avons réalisé. Vous trouverez en annexes des screenshots des terminaux des 3 serveurs. Le serveur d'indice 0 est en bleu, le serveur d'indice 1 est en rouge et le serveur d'indice 2 est en vert.



Conclusion

Pour conclure, nous aurons donc vu dans ce TP une des manières possibles pour que 3 ordinateurs puissent communiquer leur informations, sans pour autant surcharger le port qu'ils utilisent tous en même temps, grâce à l'aide de l'algorithme de Lamport. Cette algorithme produit une distribution de section critique pour les différents ordinateurs grâce à une liste d'attente fait à partir d'une liste de demande et d'une liste de réponse mis à jour très régulièrement.

Annexe

Serveur en 1ere position (serveur principal) :

```
pc5207h:~/E4/INF 4201C - Distributed programing/TP2> ./dist 5000 pc5207h pc5207g pc5207f
```

```
Numero de port de ce site 5000
WaitSync : **SYNCHRO**
WaitSync : **SYNCHRO**
Toutes les synchros recues
...........
******* je veux entrer en SC  *******

Message envoye : requete 0 0
Type : requete
Cible : site numero 1
Estampille : 0

Message envoye : requete 0 0
Type : requete
Cible : site numero 2
Estampille : 0
--- Estampille : (0,0,0)
--- Listei : (0,2147483647,2147483647)
--- Accords : (1,0,0)
.
Message recu : reponse 1 1
Type : reponse
Origine : site numero 1
Estampille : 1
--- Estampille : (1,1,0)
--- Listei : (0,2147483647,2147483647)
--- Accords : (1,1,0)
.
Message recu : requete 2 0
Type : requete
Origine : site numero 2
Estampille : 0
--- Estampille : (2,1,0)
--- Listei : (0,2147483647,0)
--- Accords : (1,1,0)

Message envoye : reponse 0 3
Type : reponse
Cible : site numero 2
Estampille : 3
--- Estampille : (3,1,0)
--- Listei : (0,2147483647,0)
--- Accords : (1,1,0)
.
Message recu : requete 1 2
Type : requete
Origine : site numero 1
Estampille : 2
--- Estampille : (4,2,0)
--- Listei : (0,2,0)
--- Accords : (1,1,0)

Message envoye : reponse 0 5
Type : reponse
Cible : site numero 1
Estampille : 5
--- Estampille : (5,2,0)
--- Listei : (0,2,0)
--- Accords : (1,1,0)
```

```
.
Message recu : reponse 2 2
Type : reponse
Origine : site numero 2
Estampille : 2
--- Estampille : (6,2,2)
--- Listei : (0,2,0)
--- Accords : (1,1,1)

**********************************
********** Passage en SC **********
**********************************
........
**********************************
********** Sortie de SC  **********
**********************************

Message envoye : liberation 0 8
Type : liberation
Cible : site numero 1
Estampille : 8

Message envoye : liberation 0 8
Type : liberation
Cible : site numero 2
Estampille : 8
--- Estampille : (8,2,2)
--- Listei : (2147483647,2,0)
--- Accords : (0,0,0)
....
Message recu : liberation 2 9
Type : liberation
Origine : site numero 2
Estampille : 9
--- Estampille : (9,2,9)
--- Listei : (2147483647,2,2147483647)
--- Accords : (0,0,0)
....
Message recu : liberation 1 10
Type : liberation
Origine : site numero 1
Estampille : 10
--- Estampille : (10,10,9)
--- Listei : (2147483647,2147483647,2147483647)
--- Accords : (0,0,0)
.................
******* je veux entrer en SC  *******

Message envoye : requete 0 11
Type : requete
Cible : site numero 1
Estampille : 11

Message envoye : requete 0 11
Type : requete
Cible : site numero 2
Estampille : 11
--- Estampille : (11,10,9)
--- Listei : (11,2147483647,2147483647)
--- Accords : (1,0,0)
.
```

Serveur en 2eme position :

```
.
Message recu : reponse 0 5
Type : reponse
Origine : site numero 0
Estampille : 5
--- Estampille : (5,6,4)
--- Listei : (0,2,0)
--- Accords : (1,1,1)
........
Message recu : liberation 0 8
Type : liberation
Origine : site numero 0
Estampille : 8
--- Estampille : (8,7,4)
--- Listei : (2147483647,2,0)
--- Accords : (1,1,1)
....
Message recu : liberation 2 9
Type : liberation
Origine : site numero 2
Estampille : 9
--- Estampille : (8,8,9)
--- Listei : (2147483647,2,2147483647)
--- Accords : (1,1,1)

**********************************
********** Passage en SC **********
**********************************
...
**********************************
********** Sortie de SC  **********
**********************************

Message envoye : liberation 1 10
Type : liberation
Cible : site numero 0
Estampille : 10

Message envoye : liberation 1 10
Type : liberation
Cible : site numero 2
Estampille : 10
--- Estampille : (8,10,9)
--- Listei : (2147483647,2147483647,2147483647)
--- Accords : (0,0,0)
..................
Message recu : requete 0 11
Type : requete
Origine : site numero 0
Estampille : 11
--- Estampille : (11,11,9)
--- Listei : (11,2147483647,2147483647)
--- Accords : (0,0,0)
```

ADAM Baptiste
JOBE Ophélie

Serveur en 3eme position :

```
pc5207f:~/E4/INF 4201C - Distributed programing/TP2> ./dist 5000 pc5207h pc5207g pc5207f
Numero de port de ce site 5002
Wait Synchro du Site 0
WaitSync : **SYNCHRO**
Synchro recue de  Site 0
...........
******* je veux entrer en SC  *******

Message envoye : requete 2 0
Type : requete
Cible : site numero 0
Estampille : 0

Message envoye : requete 2 0
Type : requete
Cible : site numero 1
Estampille : 0
--- Estampille : (0,0,0)
--- Listei : (2147483647,2147483647,0)
--- Accords : (0,0,1)
.
Message recu : requete 0 0
Type : requete
Origine : site numero 0
Estampille : 0
--- Estampille : (0,0,1)
--- Listei : (0,2147483647,0)
--- Accords : (0,0,1)

Message envoye : reponse 2 2
Type : reponse
Cible : site numero 0
Estampille : 2
--- Estampille : (0,0,2)
--- Listei : (0,2147483647,0)
--- Accords : (0,0,1)
.
Message recu : requete 1 2
Type : requete
Origine : site numero 1
Estampille : 2
--- Estampille : (0,2,3)
--- Listei : (0,2,0)
--- Accords : (0,0,1)

Message envoye : reponse 2 4
Type : reponse
Cible : site numero 1
Estampille : 4
--- Estampille : (0,2,4)
--- Listei : (0,2,0)
--- Accords : (0,0,1)
.
Message recu : reponse 1 4
Type : reponse
Origine : site numero 1
Estampille : 4
--- Estampille : (0,4,5)
--- Listei : (0,2,0)
--- Accords : (0,1,1)
.
```

```
.
Message recu : reponse 0 3
Type : reponse
Origine : site numero 0
Estampille : 3
--- Estampille : (3,4,6)
--- Listei : (0,2,0)
--- Accords : (1,1,1)
.........
Message recu : liberation 0 8
Type : liberation
Origine : site numero 0
Estampille : 8
--- Estampille : (8,4,7)
--- Listei : (2147483647,2,0)
--- Accords : (1,1,1)

**********************************
********** Passage en SC **********
**********************************
..
**********************************
********** Sortie de SC  **********
**********************************

Message envoye : liberation 2 9
Type : liberation
Cible : site numero 0
Estampille : 9

Message envoye : liberation 2 9
Type : liberation
Cible : site numero 1
Estampille : 9
--- Estampille : (8,4,9)
--- Listei : (2147483647,2,2147483647)
--- Accords : (0,0,0)
.....
Message recu : liberation 1 10
Type : liberation
Origine : site numero 1
Estampille : 10
--- Estampille : (8,10,10)
--- Listei : (2147483647,2147483647,2147483647)
--- Accords : (0,0,0)
.................
Message recu : requete 0 11
Type : requete
Origine : site numero 0
Estampille : 11
--- Estampille : (11,10,11)
--- Listei : (11,2147483647,2147483647)
--- Accords : (0,0,0)
```