**INF4402-SYSTÈMES RÉPARTIS SUR L'INTERNET**

**RAPPORT DU LABORATOIRE 2**

# Développement d'un marché d'enchères électronique

**PRÉSENTÉ PAR :**

Diderot FOUDJIO TCHOFO 1227161 (groupe2)
Elisa MIFOUM 1170562 (groupe2)

**PRÉSENTÉ À :**

Méral SHIRAZIPOUR

École Polytechnique de Montréal
Mercredi le 02 Novembre 2005

## INTRODUCTION

Avec les avancées technologiques, il est très commun aujourd'hui de faire des achats par Internet. En effet, ce mode d'achat présente plusieurs avantages dont celui (non négligeable) d'éviter aux adeptes du "magasinage" des longues files d'attente dans les magasins.
Il est question dans le présent laboratoire d'implémenter un système de vente aux enchères électroniques. Ce système permet à divers clients de se procurer des articles à prix avantageux. Ce système doit fonctionner selon le principe de la vente aux enchères : un article est proposé avec un certain prix de base; puis, les acheteurs intéressés doivent proposer des prix pour l'acquérir; l'article reviendra au client ayant misé la somme la plus forte avant la fermeture de l'enchère. Vu que les différentes transactions se font entre les clients, il est impératif que ce système soit efficace et sécuritaire. Pour ce faire, nous utiliserons divers concepts relatifs aux systèmes distribués. Il s'agit de la synchronisation et des états globaux, la coordination des services de temps, le contrôle de la concurrence et les transactions réparties.

## SCHÉMA LOGIQUE DU SYSTÈME

### PolyEbay

Le serveur PolyEbay permet de gérer les stocks d'articles, leurs échéances et de coordonner l'évolution des enchères afin d'assurer la sécurité et détecter les clients malicieux. Il est implanté grâce aux classes :

- « PolyEbayRemote » : qui représente l'interface de PolyEbay
- « PolyEbayImpl » : qui fait l'implémentation de notre serveur distant
- « PolyEbayTimer » : qui permet de mettre à jour les échéances de chaque article et qui met à jour l'affichage du tableau présentant l'état des articles ainsi que l'état actuel du serveur
- « PolyEbayClient » : qui permet l'instanciation du Client dans PolyEbay; le serveur crée cette instance pour communiquer avec le client.

Il utilise par agrégation les objets suivants :

- « PolyPaypal » : Instance du serveur PolyPaypal lui permettant de communiquer afin de vérifier les identités et les soldes des clients
- « Client » : Instance du client distant lui permettant d'envoyer les informations aux clients

**Client :**

Cet Objet permet de représenter un client et de gérer les communications avec les différents serveurs. Il est implanté grâce aux classes :

- « Client » : C'est cette classe qui crée l'objet client; elle initialise les objets et démarre l'application
- « ObjectClient » : Cette classe permet de représenter l'objet client
- « ClientImp » : C'est la classe qui implémente l'interface ClientRemote; elle permet la communication entre le client et les serveurs PolyEbay et PolyPaypal
- « FrameClient » : Cette classe crée l'applet permettent au client d'interagir avec les serveurs et de faire ses achats
- « ClientRemote » : Cette classe est une interface qui permet de représenter le client du coté du serveur.

Il utilise par agrégation les objets suivants :

- « PolyPaypal » : Instance du serveur PolyPaypal lui permettant d'avoir toutes les informations sur son compte et son crédit
- « PolyEbay » : Instance du serveur PolyEbay lui permettant d'avoir toutes les informations sur les articles (échéancier, prix, etc.) et d'effectuer les mises

**PolyPaypal**

Le serveur PolyPaypal permet aux clients de gérer leurs comptes de crédits et de l'utiliser pour effectuer leurs achats. Il est implanté grâce aux classes :

- « PolyPaypalRemote » : qui représente l'interface de PolyPaypal
- « PolyPaypalImpl » : qui fait l'implémentation de notre interface PolyPaypalRemote
- « PolyPaypalClient » : qui permet l'instanciation du Client dans PolyPaypal; le serveur crée cette instance pour communiquer avec le client.

Il utilise par agrégation les objets suivants :

- « CreditCheck » : Instance du serveur CreditCheck lui permettant de vérifier l'identité et les soldes des clients
- « Client » : Instance du client distant lui permettant d'envoyer les notifications ou alertes aux clients

**CreditCheck**

Le serveur CreditCheck contient les comptes de crédits des clients et communique régulièrement avec le serveur PolyPaypal pour la vérification des identités et des soldes des clients. Il est implanté grâce aux classes :

- « CreditCheckRemote » : qui représente l'interface de CreditCheck
- « CreditCheckImpl » : qui fait l'implémentation de notre interface CreditCheckRemote

## IDENTIFICATION DES PROBLÈMES DE CONCURRENCE

Nous avons ci-dessous les cas spécifiques d'accès concurrents aux ressources que nous avons identifié dans notre implémentation.

- PolyEbay & PolyPaypal

Partage des ressources sur les clients

- PolyEbay & PolyEbayTimer & PolyPaypal

Partage des ressources articles et les clients

- Accès aux méthodes des interfaces distantes

Partage des méthodes entre les serveurs et les clients

## SOLUTION À CES PROBLÈMES DANS NOTRE CODE

La programmation multithread (plusieurs files d'exécutions) avec partage de ressources implique en général un problème d'accès concurrent qui peu provoquer une instabilité ou un blocage du programme.

Dans notre programme de gestion des enchères, nous avons une application qui est distribuée sur plusieurs clients avec partage d'informations telles que les articles ou les informations sur les clients.  Description des problèmes :

- PolyEbay & PolyPaypal

Nous avons ici deux serveurs gérant respectivement les articles et l'argent des clients qui partage certaines informations sur ces clients, en effet avec la situation ou deux voir plusieurs clients essayent de se connecter sur le système, on a des accès concurrents aux ressources comme la liste des clients ou la mise à jour de la liste des articles.

- PolyEbay & PolyEbayTimer & PolyPaypal

PolyEbayTimer est une file d'exécution permettant de gérer le timer des articles (mise à jour du temps et l'évaluation des échéanciers). Elle s'exécute en parallèle avec les serveurs PolyPaypal et PolyEbay, lorsqu'elle a une échéance d'un article, elle identifie le client ayant remporté les enchères à travers PolyEbay et vérifie le solde du client à travers PolyPaypal. Ceci implique une situation d'accès concurrente aux informations des clients.

- Accès aux méthodes des interfaces distantes

Nous avons des interfaces pour nos serveurs et elles permettent aux clients de voire les méthodes disponibles ou implémentées sur le serveur. Mais aussi le client présente aussi une interface nécessaire aux serveurs pour effectuer les notifications ou afficher les messages chez les clients. Tous ceci entraine pour une situation ou plusieurs clients accèdent au serveur un accès concurrentiel aux méthodes du serveur et nous pouvons avoir une corruption ou mauvaise lecture de données

Pour résoudre ce problème afin de préserver l'intégrité des données, nous avons eu le choix entre deux  méthodes de résolutions : Les sémaphores et les moniteurs Java.
Nous avons utilisé les moniteurs à cause de leur simplicité d'utilisation et aussi comme ils sont intégrés au langage Java.
Bien que  nous obtenions quelque chose qui fonctionne parfaitement, l'utilisation des moniteurs ou sémaphores pour assurer les accès exclusifs aux ressources entraine un autre problème : **l'inter blocage**
Et ce problème n'a pas été traité dans notre programme, malgré qu'il peu fausser complètement les résultats attendus.

## DESCRIPTION DE L'ENVIRONNEMENT DE TEST

Notre environnement de test est Windows car sous linux, nous avions des problèmes de mémoire avec la machine virtuelle java. Pour effectuer nos tests, nous avons élaboré quatre scénarios nous permettant de tester les différentes situations notées.

1. ***Un seul client***

Cette situation nous permettait de valider l'état dit «sur» afin de la prendre comme l'idéale pour vérifier les autres tests.

2. ***Plusieurs clients sans timer (pas d'échéance des articles)***

Avec cette situation, nous avons vérifié et validé le problème d'accès concurrent aux listes des clients par les serveurs PolyPaypal et PolyEbay. En effet, nous avons procédé comme suit :

- Connecter le premier client aux serveurs, pendant qu'il est entrain de miser sur un article, connecter tous les autres clients simultanément (pseudo-simultané).
- Effectuer les mises simultanées avec les clients afin de détecter les situations de corruption de données (liste des clients ou des articles).

3. ***Plusieurs clients avec timer (avec échéance des articles)***

Cette situation nous permet de vérifier l'état de synchronisation entre PolyEbay et PolyEbayTimer afin de s'assurer que tous les échéanciers seront respectés sans détérioration des articles. Après avoir paramétré les échéances de certains articles, nous avons procédé comme suit :

- Connecter les clients l'un à la suite de l'autre pour s'assurer de la rapidité de traitement du serveur PolyEbay et sa communication avec le serveur PolyPaypal car à chaque connexion d'un client PolyEbay doit vérifier si ce client est déjà une connexion sur PolyPaypal
- Attendre la fin d'un échéancier afin de vérifier comment PolyEbayTimer gère la traite les données avec PolyEbay pour déterminer la vainqueur des enchères et vérifier son compte avec PolyPaypal (et CreditCheck) afin de le sanctionner le client s'il a misé sans suffisance de fonts.
- Avec tout ceci on vérifie l'accès aux méthodes des interfaces

4. ***Synchronisation des horloges***

Cette situation devait nous permettre de vérifier la synchronisation des horloges des clients avec celles des serveurs. Bien que nous ayons testé tout ceci sur une même machine, ce test est nécessaire dans les applications comme les ventes aux enchères afin de ne pas favoriser un client par sa position par rapport au temps universel (UTC).
Nous avons procédé comme suit après avoir paramétré les échéances des articles :

- Connecter plusieurs clients (l'ordre n'a pas d'importance)
- Note l'heure à laquelle chaque client effectue une mise pour un même article sur le serveur PolyEbay
- Analyser le tableau affiché de manière périodique sur le serveur afin de vérifier les heures de mise pour chaque client. Ceci nous permet de vérifier à quel moment le serveur reçois et affiche les requêtes de mise pour régler tous les horloges selon l'heure du serveur PolyEbay.

## CODE SOURCE

Nous avons choisi les classes PolyPaypalImpl, PolyEbayImpl et ClientImpl car elles implémentent les interfaces de communication pour notre application. En effet PolyPaypalImpl et PolyEbayImpl implémentent respectivement les interfaces PolyPaypalRemote et PolyEbayRemote qui représentent les objets distants fournissant les méthodes des serveurs PolyPaypal et PolyEbay accessibles par les clients pour faire des opérations. La classe ClientImpl implémente l'interface ClientRemote qui permet aux serveurs de faire les notifications ou alertes sur les clients.
Ces fichiers sont joints afin de garder le formatage.

## CONCLUSION

Au terme de ce laboratoire, nous avons assimilé la notion d'application distribuée par le biais d'un programme de vente aux enchères sur internet et avons rencontré ses différentes contraintes. En effet, nous avons eu les problèmes d'accès concurrent entre plusieurs processus et threads aux ressources qui entrainent en général la détérioration des données. Pour résoudre ces problèmes, nous avons utilisé le principe de synchronisation offert par Java à savoir les moniteurs, mais cette solution nous a entrainé le problème d'inter-blocage. Comme la résolution des problèmes d'inter-blocage est plus complexe car elle nécessite d'abord la détection de l'inter-blocage et souvent l'arrêt du programme. Nous avons dû enlever certaines méthodes et ressources du moniteur pour diminuer le nombre de sections critiques et pouvoir continuer l'exécution du programme sans inter-blocage car aucune solution efficace n'a été trouvée.