MobiTicket: Application mobile de ventes aux enchères de billets de spectacles

Préparé par Christina Braz
Laboratoires Universitaires Bell (Bell Canada)
Innovations technologiques

Rapport émis en : 4 août 2004

Table de Matières

1. Introduction

Le projet d'enchère électronique mobile MobiTicket consiste à faire l'évaluation d'une opportunité de mise en marché de billets de spectacles en utilisant certaines fonctionnalités de la plate-forme de négociation générique (GNP2) de l'initiative "Towards Electronic Marketplaces" (TEM) du CIRANO. La particularité de ce projet est d'offrir la réservation des places en envoyant un prix d'offre au travers du service "SMS Push"[1] sur les téléphones mobiles des utilisateurs inscrits en implémentant une application mobile de ventes aux enchères de billets de spectacles.

Le but principal de ce projet est de concevoir et d'implémenter une application d'enchère électronique mobile en utilisant les fonctionnalités de la plate-forme GNP2. Pour cela nous avons réalisé un prototype fonctionnel du l'offre de service afin d'illustrer l'interaction de l'application MobiTicket avec des différents acteurs tels que les utilisateurs et leurs téléphones mobiles, la plate-forme GNP2 au CIRANO et le service de routage de messages SMS. Il ne s'agit pas encore d'un produit à vocation commerciale mais d'un prototype pouvant entrer dans une démarche d'étude de faisabilité. L'application se nommera "MobiTicket 1.0"[2].

Note : L'emploi du masculin dans ce document est utilisé sans discrimination et dans le seul but d'alléger le texte.

2. Description générale

2.1. Définition de l'application MobiTicket

MobiTicket est une application de ventes aux enchères électronique mobile basée sur des messages SMS (Short Message Service) [VIS04] texte pour la vente de billets de spectacles. Ces messages SMS [GSM04] sont envoyés aux téléphones mobiles des utilisateurs déjà inscrits gratuitement sur un site Web d'un producteur des spectacles (e.g. Place des Arts) ou des entreprises de vente de billets (e.g. Admission, Vitrine culturelle).

Le ASP (Application Service Provider, en français FAH pour Fournisseur d'applications hébergées)[3] [ASP03] qu'ici pourrait être le ASP de "Place des Arts" [PLA04] peut offrir à ses utilisateurs mobiles des ventes aux enchères des forfaits promotionnels (billets des spectacles privilégiés couplés avec des offertes promotionnels exclusifs).

[1] Service Push SMS: c'est un service mobile de télécommunications permettant de fournir, après une activation réussie du service par les utilisateurs mobiles eux-mêmes, des informations payantes sur un thème demandé au moyen de SMS. Les utilisateurs peuvent par exemple s'informer sur les actualités, les prévisions météorologiques, la bourse, etc.

[2] Le terme "mobiTicket" n'est pas enregistré comme marque de commerce au Canada (Source: Industrie Canada, 2003).

[3] "Le FAH administre et délivre des solutions applicatives à plusieurs entités, depuis un centre serveur et à travers un réseau de télécommunication étendu WAN-Wide Area Network" (définition World Wide Web Consortium: http://www.w3.org/Glossary). Il s'agit alors de l'externalisation de l'hébergement d'une application ou d'un service en ligne.

2.2. Fonctions de l'application

L'utilisateur mobile recevra initialement une annonce d'une vente aux enchères "forfaits promotionnels" par le marchand (e.g. Place des Arts) selon la Figure 1. Le "forfait promotionnel" se constitue d'une visite au back-stage, d'un souper et d'un siège privilégié dans la salle de spectacle. Selon les saisies faites par l'utilisateur mobile, celui-ci recevra des messages SMS suivants:

- d'acquittement pour une mise valide.
- des messages d'erreurs pour des mises non-valides et pour des mises faites après la clôture de l'enchère.

- de l'état d'avancement d'enchère dans le cas où l'utilisateur mobile remporte ou pas l'enchère.

Après avoir être inscrit gratuitement, l'utilisateur pourra définir ses abonnements au service de ventes aux enchères "forfaits promotionnels". La gestion des abonnements se fera exclusivement via Internet. Suspendre le service pendant une période donnée (vacances, etc.), de refuser de recevoir des SMS d'information les jours fériés ou encore, définir précisément le type d'informations souhaitées et l'horaire de réception des SMS et, surtout, l'utilisateur pourra également définir un nombre maximum de SMS qu'il désire recevoir sur base mensuelle.

Nous présentons un exemple ci-dessous d'une "instanciation standard" d'une enchère [ROB&BER03] "Forfait promotionnel" qui a été utilisé pour la démo du prototype en utilisant les paramètres de la plate-forme GNP2 dans le Tableaux 1.

Tableau 1: Instanciation standard d'une enchère "Forfait promotionnel"

Type d'enchère	Enchère Vickery (deuxième prix offre cachée)
Type de service	Push SMS[4]
Description de l'enchère	Étape 1): Annonce d'une vente aux enchères "forfait promotionnel" par le marchand sur le téléphone de l'utilisateur mobile : *Jean Leloup 29 juil.04 Place des Arts 20 h 2 billets mezzanine + backstage + souper Mise minimale: 300$ Clôture: 16h15 EDT*
Date de début	29.07.2004
Date de fin	29.07.2004
Nombre de Spectacles	1
Nombre des Phases	1
Nombre des Rounds	1
Price Allocation Rule	Le gagnant est celui dont la mise est la plus élevée mais ce dernier payera la deuxième plus haute mise.
Règles de Négociation [ROB&BER03]:	
Reference Side	SELL
Min_Active_Products	1
Min_Active_Parties	1
Ordre_Initial_Vente	300$[5]
Règles de Phase :	
Annonceur Adjudication Price Evolution	FORWARD
Pricing Rule	Second_Price
Order Line Item Ranking	Price_Descending
	Time_Descending
Phase End Conditions	Duration_Timeout = 450 secondes[6]
	Instant_Timeout = 04:15[7]
	Max_Rounds = 1
Round End Conditions	Duration_Timeout = 450 secondes
Règles de adjudication	L'utilisateur mobile recevra des messages SMS étant dans l'un des cas suivants : Étape 2): Si l'utilisateur mobile a fait une mise valide (SMS d'acquittement): *Merci! Votre mise a été acceptée! Attendez un message de l'état d'avancement!* Étape 2a): Si l'utilisateur mobile remporte l'enchère : *Bravo! Vous avez gagné l'enchère! 8575[8] Jean Leloup 400$ Valide jusqu'à 19h30 ET Salle Wilfrid-Pelletier 175 Rue St-Catherine T: 842-2112* Étape 2b): Si l'utilisateur mobile perds l'enchère : *8575 Jean Leloup. Votre mise a été insuffisante. Merci pour votre participation!* Étape 3): Si l'utilisateur mobile a misé plus d'une fois : *Désolé, vous avez déjà fait une mise!* Étape 4): Si l'utilisateur mobile a misé une valeur inférieure ou égal à la mise minimale, ou a saisit des caractères autres que des lettres: *Désolé, vous n'avez pas fait une mise valide!* Étape 5): Si l'utilisateur mobile a misé après l'heure de la fin d'enchère: *Désolé, cette enchère est déjà terminée!*

[4] Service Push SMS: c'est un service mobile de télécommunications permettant de fournir, après une activation réussie du service par les clients mobiles eux-mêmes, des informations payantes sur un thème demandé au moyen de SMS. Les clients peuvent par exemple s'informer sur les actualités, les prévisions météorologiques, la bourse, etc.

[5] Prix de réserve : Le prix de réserve est le prix le plus bas accepté par l'annonceur. Ce prix est inclus dans l'ordre initial de l'annonceur.

[6] La durée moyenne d'une enchère est de 450 secondes (15 minutes). Horaire : Entre 12h et 20h (weekdays) et entre 12h et 22h (weekends).

[7] Le moment exact de la fin de la phase.

[8] Code du spectacle (utilisateur).

Un document est créé par l'encanteur en faisant la mise à jour du l'état du marché (allocations finales) selon le Tableau 2 ci-dessous. Ensuite, l'encanteur doit finalement communiquer aux participants le résultat de l'enchère (publication des adjudications).

Tableau 2: Document sur la mise à jour du l'état du marché.

`OnClosePhase()`[9]	Début de la production de l'adjudication
Cardinalité	Un par produit (spectacle) transigé
Acheteur	Utilisateur mobile
Vendeur	Marchand (e.g. Place des Arts)
Produit	Spectacle Jean Leloup
Quantité	2 billets
Prix	`Second_Price`
Acteurs autorisés à lire	Privée (les participants concernés)
Acteurs autorisés à modifier	Aucun : une fois sérialisée, l'adjudication ne peut être modifié.
Historique Base de données	Oui

Les captures d'écran des ventes aux enchères "forfaits promotionnels" sont présentes dans la Figure 1 où nous pouvons remarquer que l'application en soit représente juste l'envoi des 3 messages SMS au total selon les étapes 1, 2 et 2a. Alors du côté utilisateur, cette application représente alors un envoi de message SMS, que soit, la mise faite pour lui.

Si l'utilisateur remporte l'enchère, il paiera les billets au guichet du théâtre lors de sa remise en présentant le code de confirmation (e.g. 8575) qui est affiché dans l'annonce envoyée à leur téléphone mobile ce que assure au marchand que la réservation est authentique (présence de l'appareil et du numéro de confirmation combiné).

Dans les cas où l'utilisateur commet des erreurs de saisie nous avons soumis aux validations suivantes:

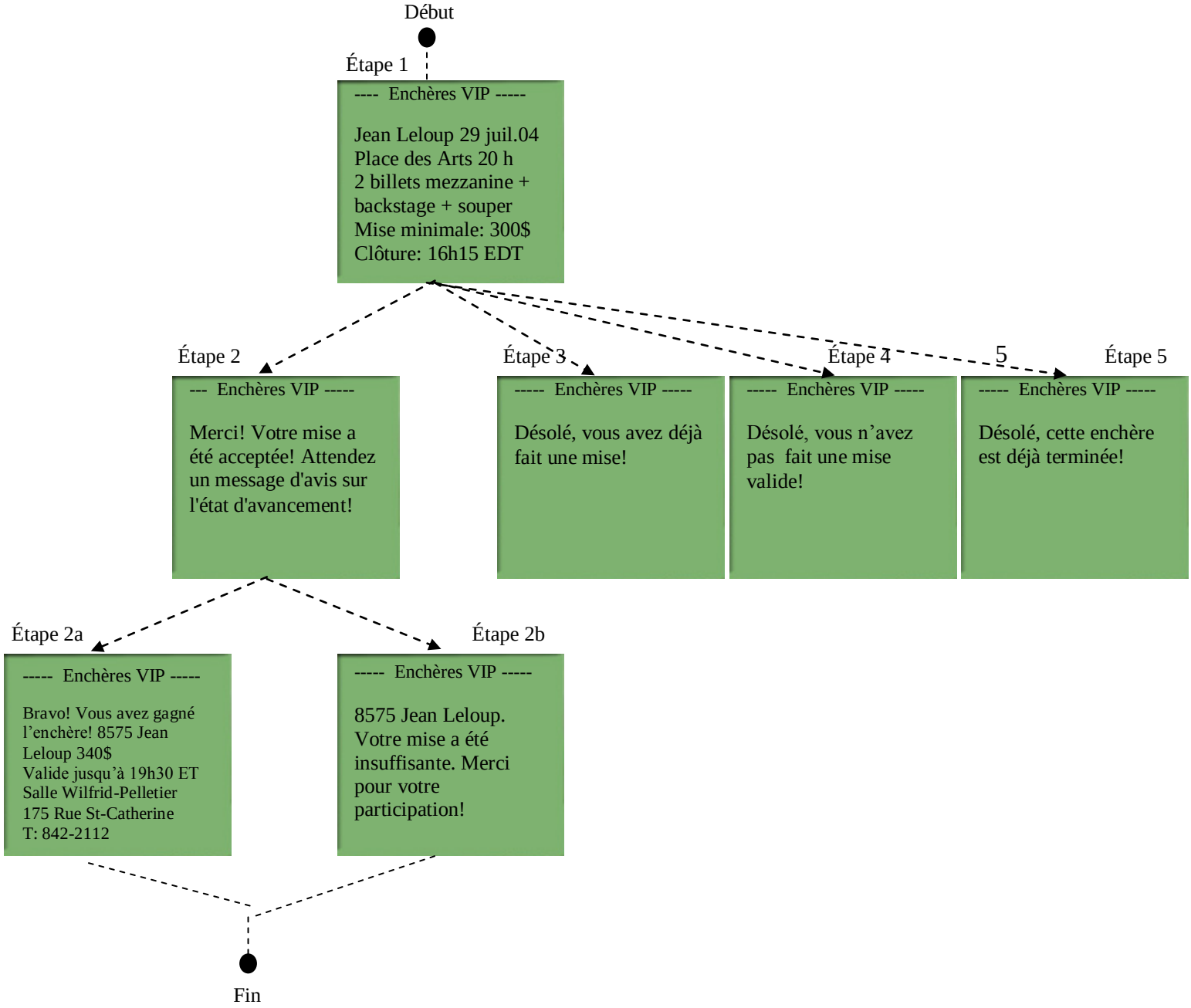
Étape 3): Si l'utilisateur mobile a fait une mise plus d'une fois (le protocole de cette enchère particulière l'utilisateur ne peut soumis une mise qu'une fois);

Étape 4): Si l'utilisateur mobile a a fait une mise avec une valeur inférieure ou égal à la mise minimale, ou a saisit des caractères autres que des lettres;

Étape 5): Si l'utilisateur mobile a fait une mise après l'heure de la fin d'enchère.

[9] L'appel à la méthode onClosePhase(). L'encanteur y fait le traitement de fin de phase. C'est la dernière méthode à être appelée, mettant ainsi fin la boucle de l'encanteur.

Figure 1



Captures d'écran des ventes aux enchères "forfaits promotionnels".

Les messages échangés entre le marchand et l'utilisateur mobile sont démontrés dans la Figure 2.

Figure 2

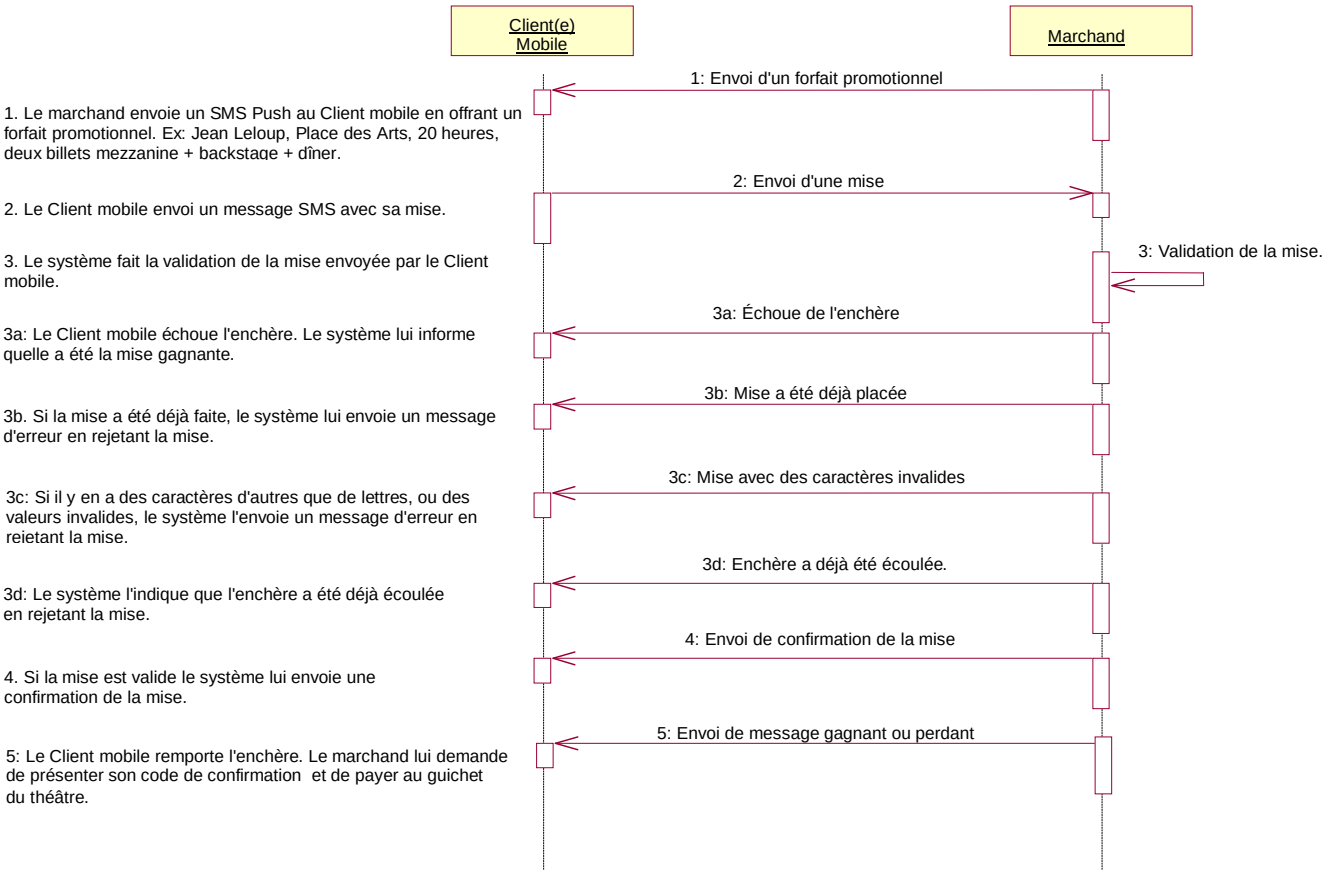


Diagramme de séquence "Front-End" (Client mobile et Marchand).

Nous avons mis assez des efforts dans la conception de MobiTicket de suivre aux règles standard d'ergonomie des systèmes et des interfaces-utilisateur[ERG04] dans la communication utilisateur/appareil mobile, la facilité d'utilisation au même titre que nous nous s'assurons de la performance du service. Ainsi, nous considérons MobiTicket du côté utilisateur comme une application simple (l'utilisateur fait juste une mise => un reply dans tous les échanges des messages), très commode et pratique (l'utilisateur reçoit des service SMS Push) et quand même une application de divertissement (l'utilisateur "joue aux enchères").

Partant du constat que notre but a été de développer initialement un prototype pour l'application, nous ne seront pas capable d'effectuer des testes d'utilisabilité et d'interaction homme-ordinateur afin d'effectivement identifier des problèmes d'utilisabilité auprès des utilisateurs réels (*end-users*), d'analyser les causes, d'élaborer et de mettre en place des solutions dans la version suivante jusqu'à ce que les critères d'ergonomie soient validés. Ces testes pourront être réalisés dans un programme futur de travail.

2.3. Développement du code informatique

MobiTicket a été développée en utilisant des langages de programmation basés sur des standards telles que Java [JAV04] et XML (eXtensible Mark-up Language) pour la communication avec la plate-forme de GNP2.

Lipso, notre partenaire pour la routage des messages SMS, nous a fourni une DTD (Document Type Definition)[LIP01] à être respecté pour l'envoi et la réception des messages SMS. XML nous permet d'utiliser un fichier afin de vérifier qu'un document XML est conforme à une syntaxe donnée dans notre cas selon à la syntaxe fournie par Lipso. La norme XML définit ainsi une définition de document appelée DTD, c'est-à-dire, une grammaire permettant de vérifier la conformité du document XML.

XML a été utilisé pour les échanges de messages entre le téléphone mobile, le routeur de messages (Lipso) et le programme hébergé sur le serveur Toja (CIRANO) appelé `lipso.java`. En fait, les informations sont encadrées par des balises XML qui sont à son tour encastrées dans un programme Java selon l'extrait de code suivant (voir l'Annexe 1 pour le code complète de `lipso.java`):

```
public void SUBMIT_SM (String message, List adrList)
      {
            request_id_send++;

            XML_Request = headMessage;
            XML_Request = XML_Request + '"';
            XML_Request = XML_Request+request_id_send;
            XML_Request = XML_Request + '"';
            XML_Request = XML_Request.concat(">");

            XML_Request = XML_Request.concat ("<SUBMIT_SM SERVICE_TYPE=");
            XML_Request = XML_Request + '"';
            XML_Request = XML_Request.concat ("");
            XML_Request = XML_Request + '"';

            XML_Request = XML_Request.concat(" SOURCE_ADDR=");
            XML_Request = XML_Request + '"';
            XML_Request = XML_Request.concat(source_adr);
            XML_Request = XML_Request + '"';
            XML_Request = XML_Request.concat (" OVERFLOW_ACTION=");
            XML_Request = XML_Request + '"';
            XML_Request = XML_Request.concat ("SPLIT");
            XML_Request = XML_Request + '"';

            for (int i = 0; i < adrList.size(); i++)
                  {
                        String adresse = (String)adrList.get(i);
                        XML_Request = XML_Request.concat("><DESTINATION ADDRESS=");
                        XML_Request = XML_Request + '"';
                        XML_Request = XML_Request.concat(adresse);
                        XML_Request = XML_Request + '"';
                        XML_Request = XML_Request.concat("/>");
                  }

            XML_Request = XML_Request.concat("<SHORT_MESSAGE>");
            XML_Request = XML_Request.concat(message);
            XML_Request = XML_Request.concat("</SHORT_MESSAGE>");
            XML_Request = XML_Request.concat("</SUBMIT_SM></MCP>");
            XML_Request = entete(Integer.toString(XML_Request.length()
)).concat(XML_Request);
```



```
            try
                  {
                        System.out.println("Envoi de SUBMIT : "+ XML_Request);
                        theWriter.print(XML_Request);
                        theWriter.flush();
                      outFile.println("\n  SUBMIT_SM Send "+ XML_Request );
                        SUBMIT_SM_RESP_trait(request_id_send);
                  }
            catch (Exception e)
                  {
                        e.printStackTrace();
                  }
      }
```

Pour la communication avec GNP2 nous avons conçu le programme `Bridge.java` qui se charge de toute la communication avec GNP2 selon l'extrait de code suivant (voir l'Annexe 2 pour le code complète de `Bridge.java`):

```
public static void main(String[] args) {

        // Creation and Initialisation: DEMO

            String MobileUsers[] = {"15145771965","15145772406"};

        String users[] = {"u1","u2"};
        int durationTimeout = 180;  //300 secondes = 5 minutes!
        String productName = "toto";
        double reservePrice = 300.0;
        int quantity = 2;

      // Fin Initialisation: DEMO


            if (com.init(users,MobileUsers) ==false)
                  {
                  System.out.println("Initialization of the Lipso communication failed.
End of program.");
                        return ;
                  }
        Bridge api = null;

        java.util.Timer scheduler = new java.util.Timer();

        try {

            // Fetch a Context
            Context ctx = ContextUtil.getNamingContext();

            // Get a negotiator session home, for communication with GNP2
            NegotiatorSessionRemoteHome negoHome =
(NegotiatorSessionRemoteHome)ctx.lookup("cirano.gnp.ejb.remote.NegotiatorSessionHome");
            NegotiatorSessionRemote negoSession = negoHome.create();
            AdministratorSessionRemoteHome adminHome  =
(AdministratorSessionRemoteHome)ctx.lookup("cirano.gnp.ejb.remote.AdministratorSessionH
ome");
            AdministratorSessionRemote adminSession   = adminHome.create();
            api = new Bridge(negoSession,adminSession);
            System.out.println("[" + new Date(System.currentTimeMillis()) + "]
Submitting announce...");
```



```
            // Dispatch of the announce.

api.submitAnnounce(users,durationTimeout,productName,reservePrice,quantity);

            // Send The push Request To Lipso
            com.push(users,durationTimeout,productName,reservePrice,quantity,api);

            // make sure the adjudications are asked for only after round terminates
            // scheduler.schedule(new AskForAdjudicationsTask(api,users,com),
durationTimeout * 60000);
            scheduler.schedule(new AskForAdjudicationsTask(api,users,com),
(durationTimeout * 1000)+90000);

            }
        catch (IllegalStateException e)
            {

                  System.out.println("[" + new Date(System.currentTimeMillis()) + "]
ERROR task already scheduled: " + e);

            }
        catch (Exception e)
            {

                  System.out.println("[" + new Date(System.currentTimeMillis()) + "]
ERROR while talking to GNP: " + e);
                  e.printStackTrace();

                  if (api != null)
                        {
                        System.out.println("[" + new Date(System.currentTimeMillis()) +
"] Closing negotiation...");
                        try {
                                    api.closeNegotiation();
                              }
                        catch (Exception e2)
                              {
                                    System.out.println("[" + new
Date(System.currentTimeMillis()) + "] Unable to close negotiation.");
                                    e2.printStackTrace();
                              }

                        }
            }
    }
```

Lors de l'exécution de l'application MobiTicket, vous trouverez les messages affichés par l'application selon ci-dessous:


```
/*
 * EXÉCUTION DE L'APPLICATION AVEC LA SAISIE DES MISES VALIDES DE 500$ ET 800$
*/

toja 9:38am > cd gnp2
toja 9:38am > ./exec &
[1] 13281
toja 9:39am > Buildfile: build.xml

exec-Bridge:
     [java] Ouverture d'une nouvelle session
     [java] Lipso Started
     [java] Envoi de Bind  : 0000102<?xml version="1.0"?><MCP
CLIENT_REQUEST_ID="10001"><BIND CLIENT_ID="FIDO" PASSWORD="f1d0test"/></MCP>
     [java] [Tue Jul 27 09:39:29 EDT 2004] Submitting announce...
     [java] Creating Announce...
     [java] Added 2 users.
     [java] Product 8575 Jean Leloup added.
     [java] Initial order created
     [java] Message reçu de Lipso 0000110

      <?xml version="1.0"?>
      <MCP CLIENT_REQUEST_ID="684949">
            <BIND CLIENT_ID="mobiticket" PASSWORD="mobilipso"/>
      </MCP>

     [java] Bind_trait will call Bind_resp
     [java] The Bind_Resp sent
     [java] Announce sent
     [java] Jean Leloup 5 mar.04 Place des Arts 20h. 2 billets mezzanine + backstage +
souper Mise minimale : 300.0$ Cloture:Tue Jul 27 09:42:48 EDT 2004 ET: Envoyer un SMS

     [java] Envoi de SUBMIT : 0000387

      <?xml version="1.0"?>
      <MCP CLIENT_REQUEST_ID="10002">
            <SUBMIT_SM SERVICE_TYPE="" SOURCE_ADDR="454776" OVERFLOW_ACTION="SPLIT">
                  <DESTINATION ADDRESS="15145771965"/>>
                  <DESTINATION ADDRESS="15145772406"/>
                  <SHORT_MESSAGE>Jean Leloup 5 mar.04 Place des Arts 20h. 2 billets
                  mezzanine + backstage + souper Mise minimale : 300.0$ Cloture:Tue Jul
                  27 09:42:48 EDT 2004
                  </SHORT_MESSAGE>
            </SUBMIT_SM>
      </MCP>

     [java] Acknowlegement received from Lipso for SUBMIT_SM: 0000125

      <?xml version="1.0"?>
      <MCP CLIENT_REQUEST_ID="10002">
            <SUBMIT_SM_RESP COMMAND_STATUS="0" MESSAGE_ID="3"></SUBMIT_SM_RESP>
      </MCP>

     [java] Our message was delivered
     [java] Message reçu de Lipso 0000213




      <?xml version="1.0"?>
      <MCP CLIENT_REQUEST_ID="684950">
```



```
      <SUBMIT_SM SERVICE_TYPE="" SOURCE_ADDR="15145772406"
      OVERFLOW_ACTION="SPLIT">
            <DESTINATION ADDRESS="454776"/>
            <SHORT_MESSAGE>500</SHORT_MESSAGE>
      </SUBMIT_SM>
 </MCP>

[java] Mise lue a partir de 500
[java] User u2 bids for 2.0 units at 500.0$.
[java] u2
[java] Envoi de SUBMIT : 0000294

 <?xml version="1.0"?>
 <MCP CLIENT_REQUEST_ID="10003">
      <SUBMIT_SM SERVICE_TYPE="" SOURCE_ADDR="454776" OVERFLOW_ACTION="SPLIT">
            <DESTINATION ADDRESS="15145772406"/>
            <SHORT_MESSAGE>Merci! Votre mise a ete acceptee! Attendez un message
            d'avis sur l'etat d'avancement!</SHORT_MESSAGE>
      </SUBMIT_SM>
 </MCP>

[java] Acknowlegement received from Lipso for SUBMIT_SM: 0000125

 <?xml version="1.0"?>
 <MCP CLIENT_REQUEST_ID="10003">
      <SUBMIT_SM_RESP COMMAND_STATUS="0" MESSAGE_ID="4"></SUBMIT_SM_RESP>
 </MCP>

[java] Our message was delivered
[java] Message reçu de Lipso 0000213

 <?xml version="1.0"?>
 <MCP CLIENT_REQUEST_ID="684951">
      <SUBMIT_SM SERVICE_TYPE="" SOURCE_ADDR="15145771965"
      OVERFLOW_ACTION="SPLIT">
            <DESTINATION ADDRESS="454776"/>
            <SHORT_MESSAGE>800</SHORT_MESSAGE>
      </SUBMIT_SM>
 </MCP>

[java] Mise lue a partir de 800
[java] User u1 bids for 2.0 units at 800.0$.
[java] u1
[java] Envoi de SUBMIT : 0000294

 <?xml version="1.0"?>
 <MCP CLIENT_REQUEST_ID="10004">
      <SUBMIT_SM SERVICE_TYPE="" SOURCE_ADDR="454776" OVERFLOW_ACTION="SPLIT">
            <DESTINATION ADDRESS="15145771965"/>
            <SHORT_MESSAGE>Merci! Votre mise a ete acceptee! Attendez un message
            d'avis sur l'etat d'avancement!</SHORT_MESSAGE>
      </SUBMIT_SM>
 </MCP>

[java] Acknowlegement received from Lipso for SUBMIT_SM: 0000125

 <?xml version="1.0"?>
 <MCP CLIENT_REQUEST_ID="10004">


 <SUBMIT_SM_RESP COMMAND_STATUS="0" MESSAGE_ID="5"></SUBMIT_SM_RESP>     </MCP>
[java] Our message was delivered
```

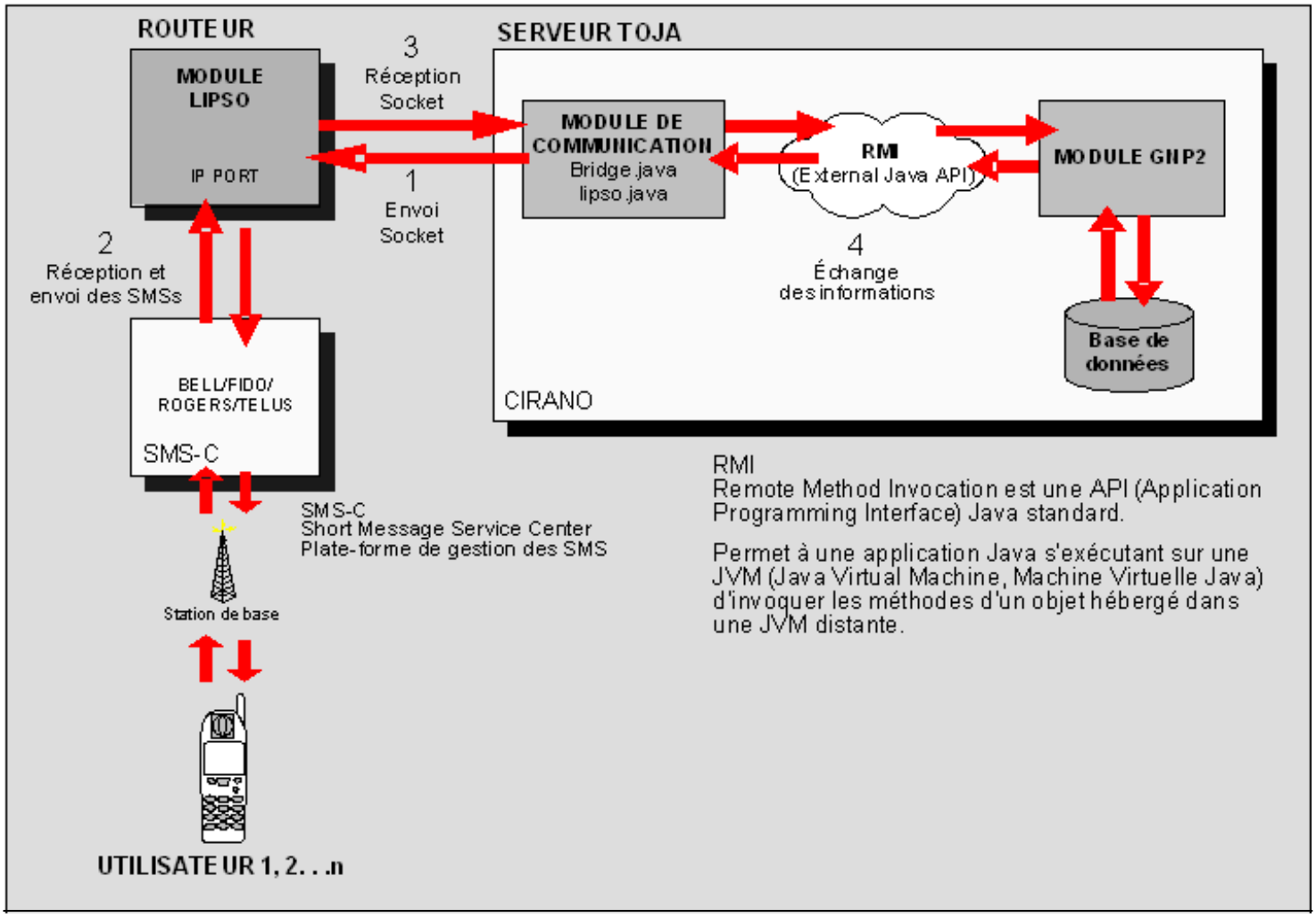


```
[java] [AskForAdjudicationsTask:run]
[java] [Tue Jul 27 09:44:18 EDT 2004] Reading quote
[java] Quote -> negoGPK = 31522816
[java]       -> status = status.closed
[java]       -> subStatus = substatus.receiving
[java]       -> negotiationName = API Nego
[java]       -> negotiationDescription = This is a test Negotiation
[java]       -> currentPhaseName = competition
[java]       -> currentPhaseNumber = 0
[java]       -> currentRound = 2
[java]       -> negotiationStartTime = Tue Jul 27 09:39:41 EDT 2004
[java]       -> negotiationCloseTime = null
[java]       -> nextTimeoutDate = Tue Jul 27 09:42:41 EDT 2004
[java]       -> biddingSide = BUY
[java]       -> nextView = simpleSynchAdj
[java]          properties     -> Property -> number_of_active_parties : 2
[java]       -> QuoteLineItem -> productGPK = 31522817
[java]          properties     -> Property -> number_of_active_parties : 2
[java]          properties     -> Property -> is_leader_price_moving : 1
[java]          properties     -> Property -> leader_name : u1
[java]          properties     -> Property -> leader_price : 800.0
[java] [Tue Jul 27 09:44:18 EDT 2004] Reading adjudications........2
[java] u1 is active
[java] User u1 wins 2.0 units at 800.0$
[java] Envoi de SUBMIT : 0000340
 <?xml version="1.0"?>
 <MCP CLIENT_REQUEST_ID="10005">
       <SUBMIT_SM SERVICE_TYPE="" SOURCE_ADDR="454776" OVERFLOW_ACTION="SPLIT">
             <DESTINATION ADDRESS="15145771965"/>
             <SHORT_MESSAGE>Bravo! Vous avez gagne l'enchere! 8575 Jean Leloup
             800.0$ Valide jusqu'a 19h30 ET Salle Wilfrid-Pelletier 175 St-
             Catherine T:8422112</SHORT_MESSAGE>
       </SUBMIT_SM>
 </MCP>
[java] Acknowlegement received from Lipso for SUBMIT_SM: 0000125
 <?xml version="1.0"?>
 <MCP CLIENT_REQUEST_ID="10005">
       <SUBMIT_SM_RESP COMMAND_STATUS="0" MESSAGE_ID="6"></SUBMIT_SM_RESP>
 </MCP>
[java] Our message was delivered
[java] u2 is active
[java] Envoi de SUBMIT : 0000297
 <?xml version="1.0"?>
 <MCP CLIENT_REQUEST_ID="10006">
       <SUBMIT_SM SERVICE_TYPE="" SOURCE_ADDR="454776" OVERFLOW_ACTION="SPLIT">
             <DESTINATION ADDRESS="15145772406"/>
             <SHORT_MESSAGE>Desole, votre mise a ete insuffisante! 8575 Jean
             Leloup. Merci pour votre participation!
             </SHORT_MESSAGE>
       </SUBMIT_SM>
 </MCP>
[java] Acknowlegement received from Lipso for SUBMIT_SM: 0000125
 <?xml version="1.0"?><MCP CLIENT_REQUEST_ID="10006">
 <SUBMIT_SM_RESP COMMAND_STATUS="0" MESSAGE_ID="7"></SUBMIT_SM_RESP></MCP>
[java] Our message was delivered
[java] Envoi de UNBind -> 0000068
 <?xml version="1.0"?>
 <MCPCLIENT_REQUEST_ID="10007"><UNBIND /></MCP>
[java] Lipso Stopped
```


2.4. Architecture générale de l'environnement MobiTicket

L'architecture de l'application est constituée de trois modules: Module GNP2, Module Communication et Module Lipso [VAC03] selon la Figure 3.

Figure 3



Architecture générale de l'environnement MobiTicket.

2.4.1. Module Communication

Il s'agit d'un modèle permettant la communication inter processus (IPC - Inter Processus Communication) afin de permettre à divers processus de communiquer aussi bien sur une même machine qu'à travers un réseau TCP/IP[VAC03]. Nous allons utiliser le mode connecté (comparable à une communication téléphonique) utilisant le protocole TCP/IP. Dans ce mode de communication, une connexion durable est établie entre les deux processus, de telle façon que le Socket de destination n'est pas nécessaire à chaque envoi de données.

Ce module est constitué des programmes suivants:

- `Bridge.java`: Initialement, un programme appelé `ExternalAPI.java` nous a été fournie par la responsable de GNP2, Isabelle Therrien au CIRANO, lequel nous a servi comme base pour la conception du nouveau programme appelé `Bridge.java` afin d'établir la communication et l'intégration avec GNP2 et la base de données installés sur le serveur "Toja" selon la Figure 3.

- `lipso.java`: Ce programme a pour objectif de réaliser la réception des sockets provenant à une adresse IP et un port bien spécifiques, l'envoi des sockets et l'extraction des informations du fichier XML qui a été reçu à travers le socket (échange de messages SMS).

2.4.2. Module GNP2

Ce module est constitué de la plate-forme GNP2 (Generic Negotiation Platform) qui est hébergée sur le server Toja au CIRANO. GNP2 peut être divisé en quatre entités conceptuelles: des interfaces humain-machine, une base de données, un contrôleur et un encanteur. Chacune de ces composantes joue un rôle bien précis et ensemble, elles forment la plate-forme GNP2. Pour des informations détaillées sur la plate-forme GNP2, veuillez consulter le document "GNP2 : Le guide du développeur version 2.0" [BER03].

2.4.3. Module Lipso

Le Module Lipso se constitue essentiellement d'une connexion à un environnement de laboratoire chez Lipso Inc. [LIP04] dans lequel nous établissons seulement le routage des messages SMS que soit l'envoi et la réception de ces messages (Voir la section 3. Solution mobile de connectivité). Cette connexion est établie dans un mode boucle où tous ce que nous transmettrons vers Lipso nous sera retourné sur notre port. Ainsi, un numéro abrégé 454776 (comme si c'était un numéro de téléphone mobile ordinaire) nous a été fourni par Lipso afin de envoyer et recevoir des messages SMS.

L'envoi et la réception des messages SMS ont été faites à travers des cartes pré-payées utilisant des téléphones mobiles "Fido"[FID04]. Ceci étant une contrainte posée par Lipso puisqu'ils en avaient un partenariat avec l'opérateur sans fil Fido.

3. Solution mobile de connectivité

Pour les échanges des messages (routage d'envoi et de la réception des messages SMS (Short Message Service)[LIP03], entre l'utilisateur mobile, le Module Lipso, notre Module de communication (`Bridge.java` et `lipso.java`) et Module GNP2 (Figure 4), nous avons utilisé les services de Lipso Inc. (Solutions mobiles de connectivité). Ces services ont été rendus gratuitement grâce au partenariat conclu avec Lipso.

3.1. Contraintes de développement

Lipso nous a établi la connexion avec notre serveur Toja, mais nous a fallu maintenir constamment cette connexion entre nos systèmes et leurs systèmes. Si nous devions mettre fin à la connexion pour une mise à niveau de notre système, cela devrait se faire dans une fenêtre de 15 minutes. Si nous étions dans l'impossibilité de remettre la connexion à l'intérieur de cette fenêtre, nous ne devrions pas faire notre mise à niveau. Si pour quelques raisons que ce soit nous aurions dû mettre fin à la connexion pour une période de plus de 15 minutes, nous a fallu contacter Lipso immédiatement, soit par courriel, soit par téléphone. Lipso alors mettait fin à la connexion pour un minimum de 24 heures.

Les tests en dehors des heures normales de travail ont été accordés en vertu d'une entente mutuelle entre Lispo et CIRANO. Dû à la nature des tests que nous faisions, que soit développement sur des équipements de production, nous a fallu fournir à Lipso une *schédule* complète et détaillée de nos tests selon l'exemple présenté dans la Figure 5.

3.2. Contraintes de performance

Présentement, la maille faible dans la majorité du déploiement des applications mobiles concerne fréquemment aux réseaux sans fils. Ainsi, le transport de messages SMS peut dépendre de facteurs en dehors du notre contrôle en incluant, mais sans s'y limiter, aux facteurs affectant l'opération des opérateurs mobiles participants ou surtout d'autres facteurs tels que indiqués ci-dessous:

- Délais ou *"non-performances"* (e.g. messages SMS) qui sont dû à des événements quelconques en dehors du contrôle d'ASP.
- Limites des systèmes d'autres fournisseurs de services d'application mobile.
- Délai d'envoi des messages SMS aux des failles techniques du réseau ou des failles techniques de n'import quelle tierce partie dans laquelle ASP a la confiance en fournir le service mobile.

Figure 4

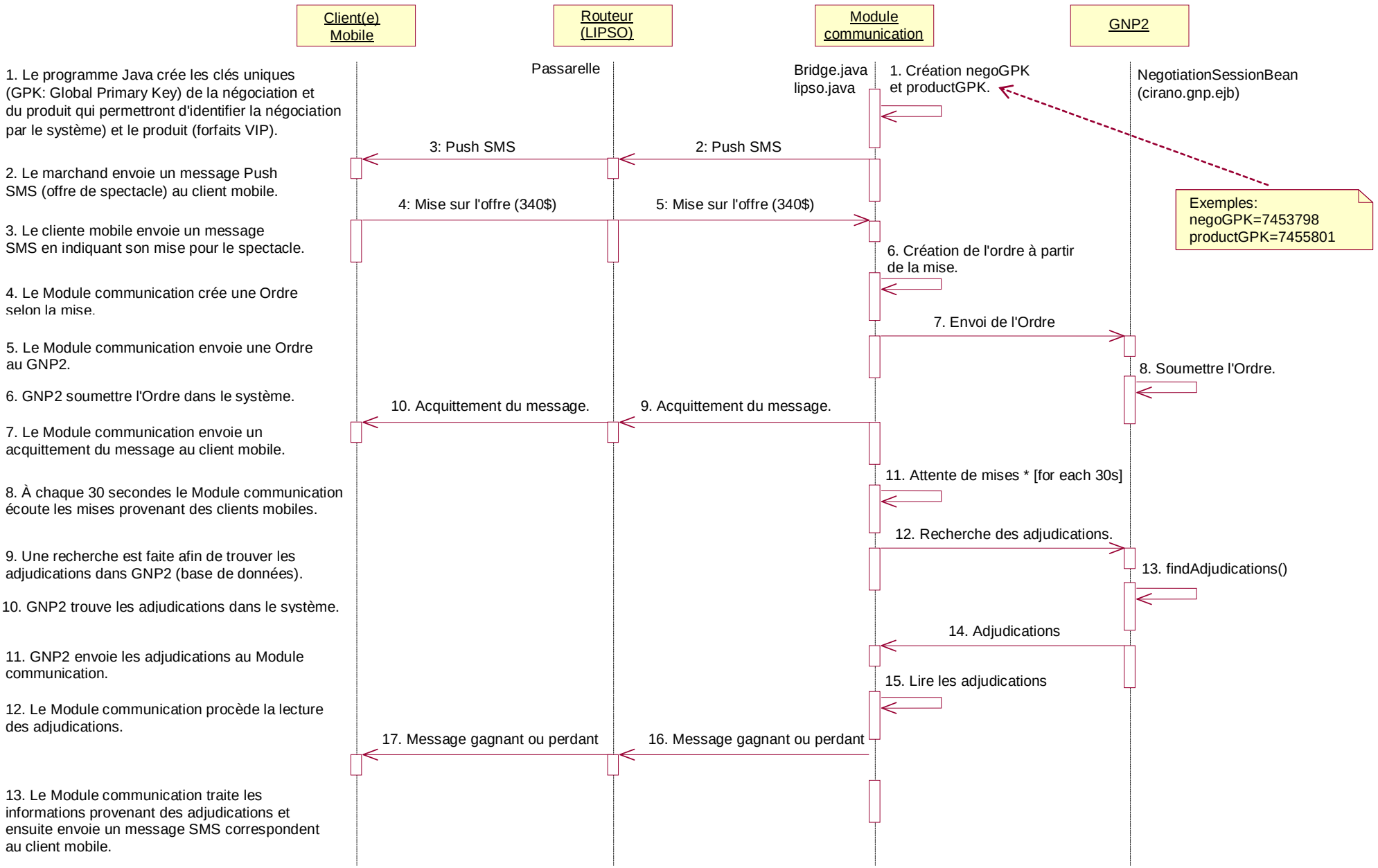


Diagramme de séquence "Back-End" de l'application "Enchères forfaits promotionnels (Côté ASP)".

Figure 5

RAPPORT HEBDOMADAIRE
PROJET D'ENCHÈRE MOBILE MOBITICKET

Mise à jour : 26-juillet-04

ID	Description	Résultat du test	Début des tests/ Heure	Jul 2004
				27-6 / 4-7 / 11-7 / 18-7 / 25-7
1	. Envoi de l'annonce de l'enchère.	. User u1 wins 2.0 units at 800.0$. User u2 lose the auction (mise insuffisante)	2004-07-19 10:33:00	
2	. Envoi de l'annonce de l'enchère.	. Negotiation is not CLOSED. PROBLEM!!!	2004-07-19 15:01:00	
3	. Envoi de l'annonce de l'enchère.	. User u1 wins 2.0 units at 800.0$. User u2 lose the auction (mise insuffisante)	2004-07-19 15:41:00	
4	. Envoi de l'annonce de l'enchère.	. Mise lue a partir de 200 . Mise rejette 300.0 sa mise est 200	2004-07-19 16:01:00	
5	. Envoi de l'annonce de l'enchère.	. User u1 wins 2.0 units at 800.0$. User u2 lose the auction (mise insuffisante)	2004-07-19 16:08:00	
6	. Envoi de l'annonce de l'enchère.	Mise faite après la fin de l'enchère: . Erreur à la lecture du stream . Session problem to receive BIND_RESPn	2004-07-19 16:15:00	
7	. Envoi de l'annonce de l'enchère.	Mise faite avec la lettre "T": Mise lue à partir de "T"; Java.lang.NumberFormatException: T [java] at java.lang.Integer.parseInt(Integer.java:409) [java] at java.lang.Integer.parseInt(Integer.java:458) [java] at cirano.gnp.tests.lipso.SUBMIT_SM_trait(lipso.java:1039) [java] at cirano.gnp.tests.lipso$deamonreceiveMessage.switchMessage(lipso.java:1299)	2004-07-19 16:41:00	
8	. Envoi de l'annonce de l'enchère.	. Not an Active Mobile User u1; . Not an Active Mobile User u2	2004-07-20 15:16:00	
9	. Envoi de l'annonce de l'enchère.	. Not an Active Mobile User u1; . Not an Active Mobile User u2	2004-07-20 15:31:00	
10	. Envoi de l'annonce de l'enchère.	. Not an Active Mobile User u1; . Not an Active Mobile User u2	2004-07-21 10:56:00	
11	. Envoi de l'annonce de l'enchère.	Failed to connect to toja.cirano.qc.ca:1199	2004-07-21 11:10:00	
12	. Envoi de l'annonce de l'enchère.	. Not an Active Mobile User u1; . Not an Active Mobile User u2	2004-07-21 11:14:00	
13	. Envoi de l'annonce de l'enchère.	Mise faite après la fin de l'enchère: . Erreur à la lecture du stream . Session problem to receive BIND_RESPn	2004-07-21 11:32:00	
14	. Envoi de l'annonce de l'enchère.	Failed to connect to toja.cirano.qc.ca:1199	2004-07-21 12:12:00	
15	. Envoi de l'annonce de l'enchère.	. Mise lue à partir de Non600; [java] Mise rejette 300.0 sa mise est 0 . User u1 wins 2.0 units at 900.0$	2004-07-21 12:19:00	
16	. Envoi de l'annonce de l'enchère.	. Not an Active Mobile User u1; . Not an Active Mobile User u2	2004-07-21 16:45:00	
17	Envoi de l'annonce de l'enchère.	. User u1 wins 2.0 units at 800.0$. User u2 lose the auction (mise insuffisante)	2004-07-21 17:12:00	
18	Envoi de l'annonce de l'enchère.	User u1 bids for 2.0 units at 800.0$; User u2 bids for 2.0 units at 500.0$; Erreur à la lecture du stream.	2004-07-23 11:23:00	
19	Envoi de l'annonce de l'enchère.	. User u1 wins 2.0 units at 800.0$; . User u2 lose the auction (mise insuffisante)	2004-07-23 15:34:00	
20	Envoi de l'annonce de l'enchère.	Mise Lu a partir de 800 [java] Submit order failed for u1	2004-07-23 15:45:00	
21	Envoi de l'annonce de l'enchère.	Mise Lu a partir de 900 [java] Submit order failed for u1	2004-07-23 15:51:00	
22	Envoi de l'annonce de l'enchère.	User u1 bids for 2.0 units at 800.0$; . Not an Active Mobile User u2	2004-07-23 15:59:00	
23	Envoi de l'annonce de l'enchère.	. Not an Active Mobile User u1; . Not an Active Mobile User u2	2004-07-23 16:28:00	
24	Envoi de l'annonce de l'enchère.	Mise faite après la fin de l'enchère: . Erreur à la lecture du stream . Session problem to receive BIND_RESPn	2004-07-23 16:49:00	

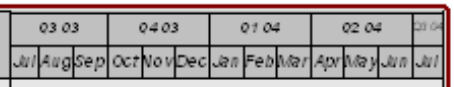
Rapport hebdomadaire des tests (CIRANO).

4. Milestones du projet MobiTicket

Depuis le début du projet jusque la présentation et la démo, nous présentons les *milestones* du projet dans la Figure 6.

Figure 6

ID	Activité	Début	Fin	Durée	Resp.
1	Réunion d'information (briefing) sur l'application des enchères électroniques mobiles.	2003-07-07	2003-07-07	2h	JR
2	Établissement des comptes sur le serveur Toja, des arrangements sur les ressources nécessaires, des exemples de codes GNP, etc.	2003-07-11	2003-08-07	20d	CB/IT
3	Étude initiale du hardware et des possibilités d'intégration de la "Passerelle SMS->GNP"	2003-08-04	2003-09-05	25d	CB
4	Étude/Définition sur le "routeur" à être utilisé pour l'envoi et la réception de messages SMS.	2003-09-08	2003-10-21	32d	CB/JR
5	Développement des applications "Bridge.java" en langage Java (passerelle SMS->GNP2) pour la communication avec GNP2.	2003-10-22	2003-11-12	16d	CB
6	Continuation du développement des applications "Bridge.java" et en intégrant le module de communication SMS "lipso.java".	2003-12-01	2004-02-06	50d	CB
7	Réalisation des tests initiaux et des corrections des bugs des applications (Serveurs Lipso et Toja/GNP2).	2004-02-09	2004-03-31	38d	CB/FB/IT
8	Réalisation des tests et des corrections des bugs des applications (Serveurs LIPSO et Toja/GNP2) avec les téléphones mobiles.	2004-04-01	2004-07-19	78d	CB/FB/IT
9	Présentation et démo du prototype	2004-07-28	2004-07-28	1d	CB



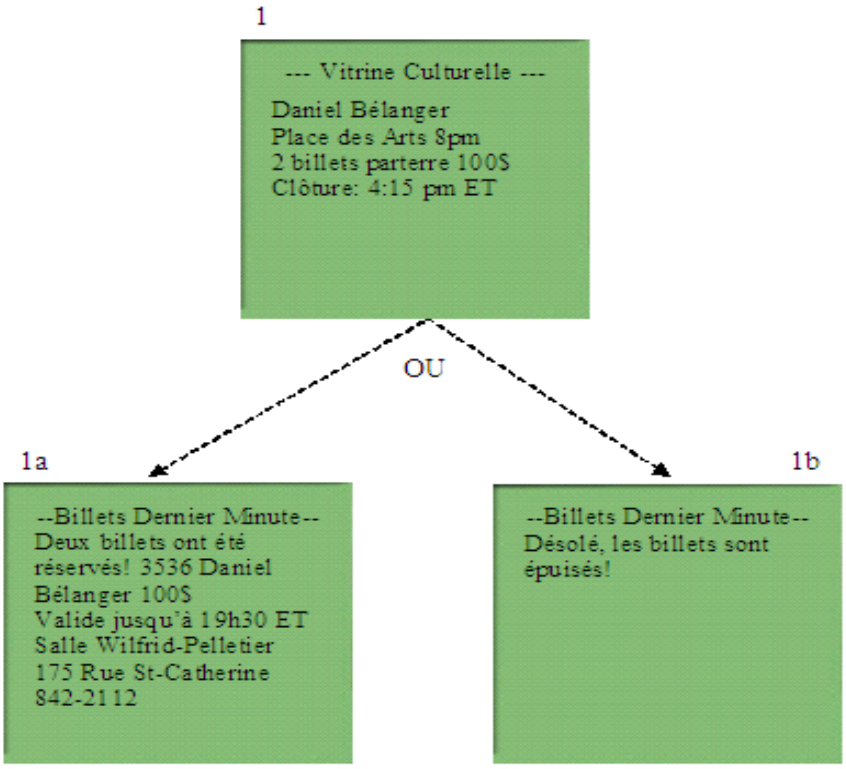
FB= Francis BEAULIEU (LIPSO); CB= Christina BRAZ, JR= Jacques ROBERT, IT= Isabelle THERRIEN (CIRANO).
Notes:
1. CB est parti en vacances pendant le mois de novembre 2003.
2. Période de fêtes: 23-déc.-03/04-janv.-04 (15 jours).
3. JR est parti en vacances pour une durée de trois semaines pendant le mois de juillet 2004.

5. Conclusion et travaux futurs

Nous avons accompli notre objectif de viabiliser des applications d'enchères électroniques mobiles en utilisant la plate-forme GNP2 et en présentant à la fin un prototype fonctionnel de l'application MobiTciket. En outre, nous envisageons également l'implémentation des services suivants:

- Les ventes à la dernière minute de billets des spectacles invendus à travers du Service "SMS PUSH" selon les captures d'écran dans la Figure 7 et le diagramme de séquence dans la Figure 8.

Figure 7



Captures d'écran des ventes à la dernière minute de billets des spectacles invendus.

Figure 8

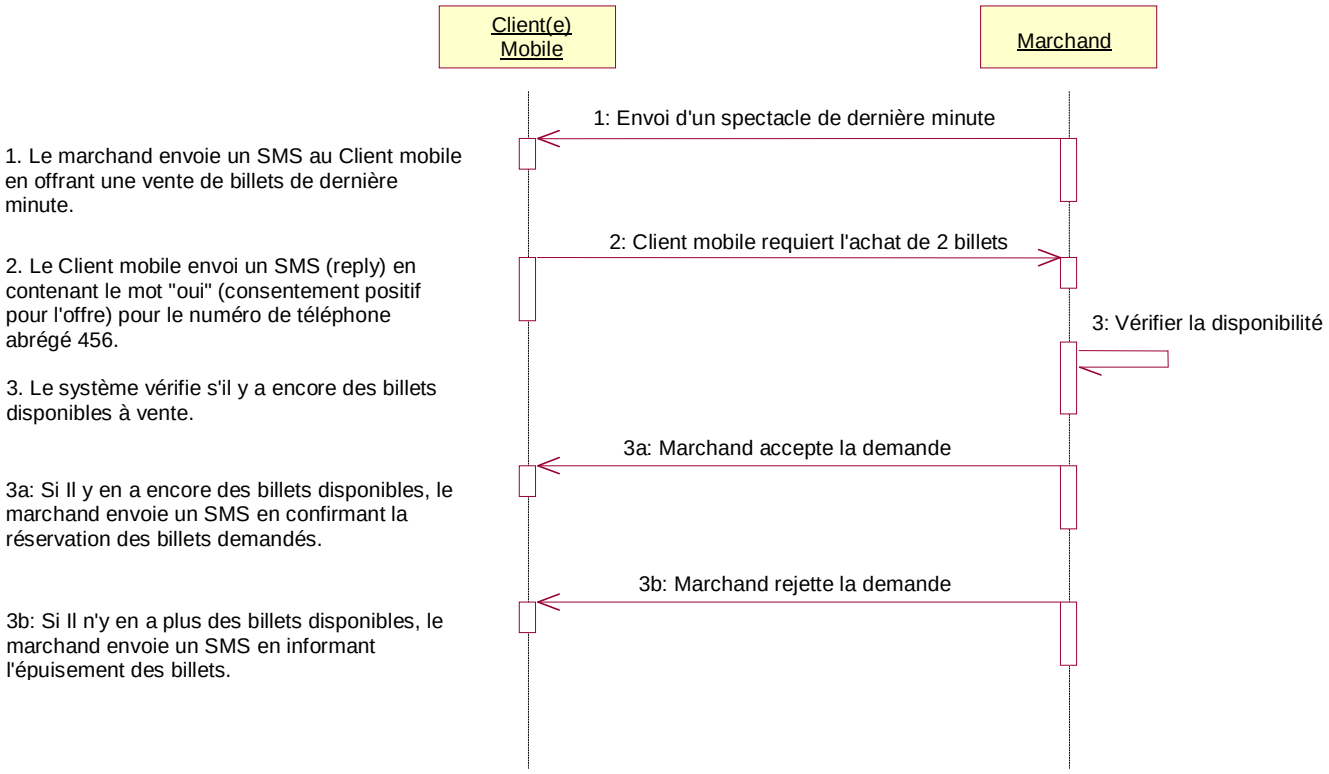


Diagramme de séquence des ventes à la dernière minute de billets des spectacles invendus.

- Les ventes des billets des spectacles à travers du service "SMS PULL" selon les captures d'écran dans la Figure 9 et le diagramme de séquence dans la Figure 10.

Figure 9

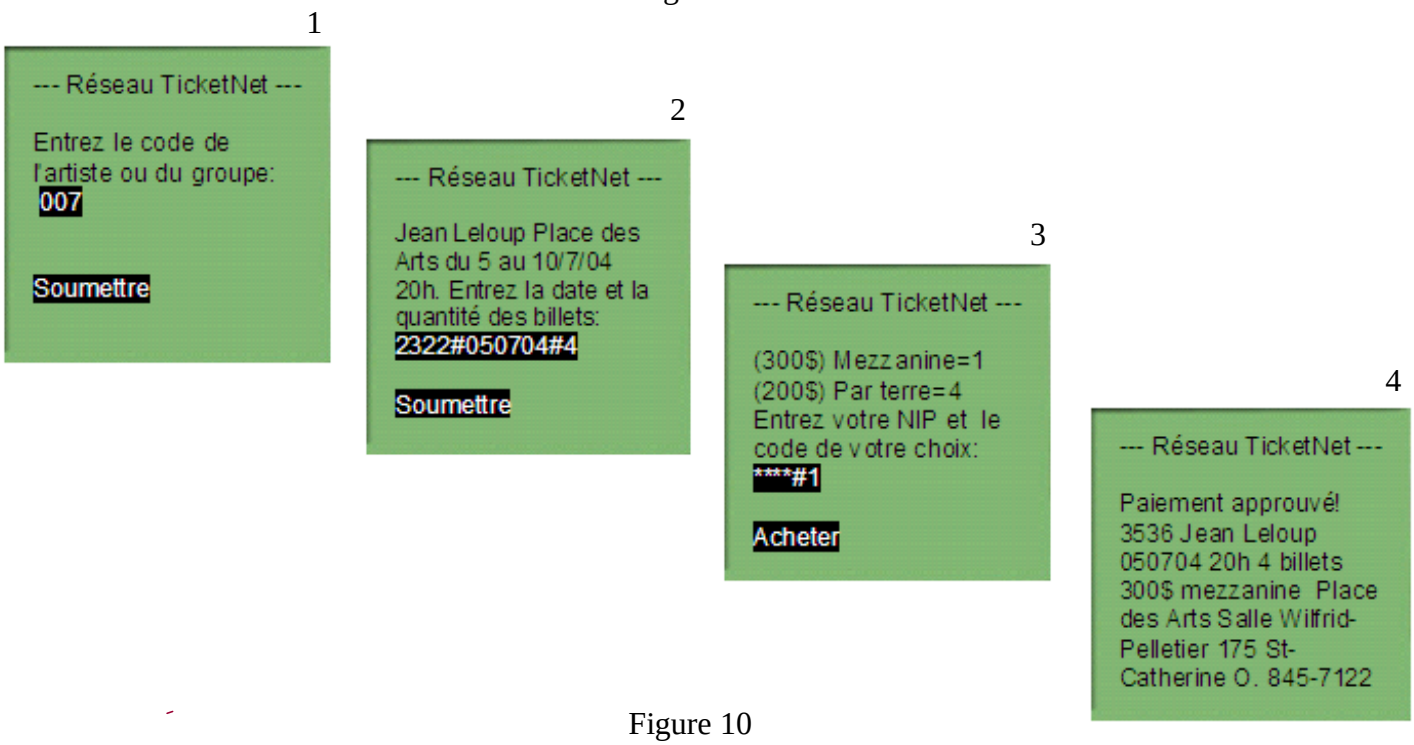


Figure 10

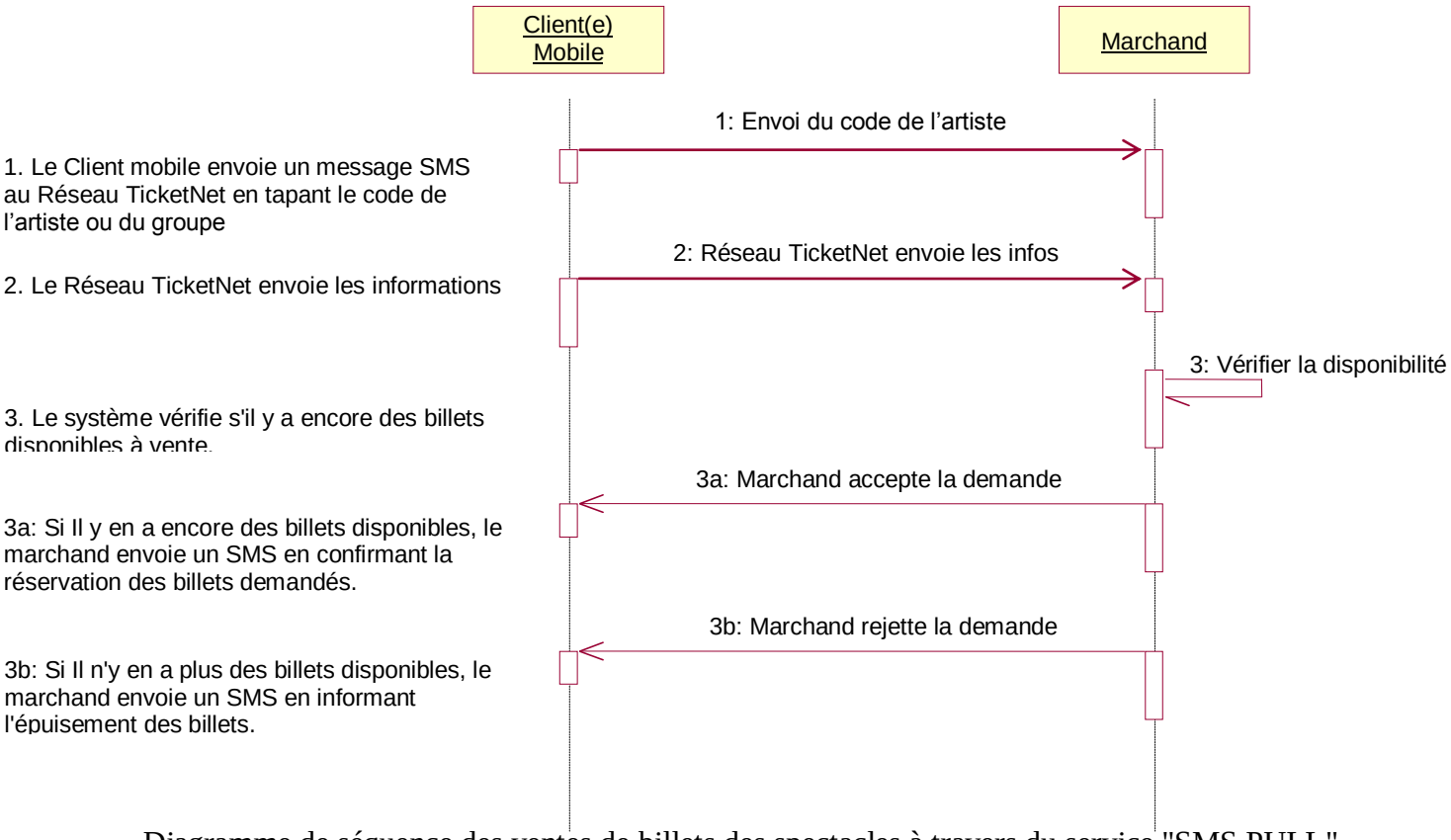


Diagramme de séquence des ventes de billets des spectacles à travers du service "SMS PULL".

6. Annexes

6.1. Code source complète `lipso.java`

```
/**
*
* Nom du fichier: lipso.java
* But: Envoi et réception des messages SMS
* Date: 23 juillet 2004
* Projet: Enchère électronique mobile MobiTicket
* Auteur: Christin Braz
* Institution: CIRANO.
*
**/

package cirano.gnp.tests;
import java.util.*;
import java.net.*;
import java.io.*;
import java.io.File;

public class lipso {

//************************************************
//
//   Connection parameters
//
//************************************************

      /**
      * Mobitcket Configuration
      */

private String          password                = "f1d0test";
private String          client_id               = "FIDO";

private String          source_adr              = "454776";
private int             request_id_send         = 10000;
private String          my_SESSION_ID           = "0000";
private String          marchandIp              = "207.162.57.23";
private int             marchandPort            = 9317;

      /**
      * Lipso Configuration
      */
private String          lipsoIp                 = "www.ilipso.com";
private int             lipsoPort               = 10010;
private String          SESSION_ID_LIPSO        = "0000";  // Session number to affect
to LIPSO


private String          headMessage             = "<?xml version=" + '"' + "1.0" + '"'+
"?><MCP CLIENT_REQUEST_ID=";
private String          headMessageRec          = "<?xml version=" + '"' + "1.0" + '"'
+ "?><MCP";
```



```
private String                BIND_RESP               = "><BIND_RESP COMMAND_STATUS=";
private String                UNBIND_RESP             = "><UNBIND_RESP
COMMAND_STATUS=";
private String                SUBMIT_SM_RESP          = "><SUBMIT_SM_RESP
COMMAND_STATUS=";
private String                ERROR                        = "><ERROR REASON=";
private String                ENQUIRE_LINK_RESP       = "><ENQUIRE_LINK_RESP
COMMAND_STATUS=" ;
private String                BIND                    = "><BIND CLIENT_ID=";
private String                UNBIND                       = "><UNBIND/";
private String                SUBMIT_SM               = "><SUBMIT_SM SERVICE_TYPE=" ;
private String                ENQUIRE_LINK            = "><ENQUIRE_LINK/>";
private ServerSocket    mobiticketSocket;
private Socket                theSocket;                        // Used to send a
message to Lipso
private PrintWriter     theWriter;                        // Used to send a
message to Lipso
private BufferedReader  theReader;                        // Used to recieve
acknowlegement
private PrintWriter     theServerWriter;
private BufferedReader  theServerReader;
private String          XML_Request;

public  static boolean  connected;                        // default is false
public  static double   bidmin;                                 // pourquoi
public ?
public  static int            quantity;                         // pourquoi
public ?

private static String   MobileUsers[];
private static String   users[];
private static int            MobileActivUsers [];
private static PrintWriter outFile;

private Bridge _api;
private Calendar rightNow;

/**
* connected to Lipso server
*
*/
public boolean connected()
      {
            return connected;
      }


/**
*  constructeur par defaut initialise les variables
*
*/
public void lipso ()
      {
      // Nothing to do
      }

/**
*
*/
public boolean init (String users_ids[],String MobileUsers_num[])
      {
            try
```



```
                {

                /*  Neither this method nor any of its variants will return the same
abstract
                      pathname again in the current invocation of the virtual
machine. */
                      File file   = File.createTempFile("lipso", ".log");

                outFile
                      = new PrintWriter(new BufferedWriter (new
FileWriter("lipso.log")));
                /*
                if (file !=null)
                {
                      String file_name = file.toString() ;
                      System.out.println("Tmp file created is " + file_name);
                      FileWriter fr = new FileWriter (file);
                  }
                else{
                System.out.println("Creation of tmp file failed" );
                      }
                */
                }
            catch (Exception e)
                {
                e.printStackTrace();
            }

            new deamonreceiveMessage().start();

        initSocket();

            /* Test du port

            Socket theSocketTmp;
            PrintWriter theWriterTmp;  // Used to send a message to Lipso
            BufferedReader theReaderTmp; //Used to recieve acknowlegement

            try {
                  theSocketTmp = new Socket(marchandIp,marchandPort );
                  theWriterTmp = new PrintWriter(theSocketTmp.getOutputStream(), true);
                  theReaderTmp = new BufferedReader(new
InputStreamReader(theSocketTmp.getInputStream()));

                    System.out.println("Envoi de Message sur l'adresse "+ marchandIp +
" au port " + marchandPort);
                  theWriterTmp.print("Un message a mon serveur ");
                  theWriterTmp.flush();

          }
        catch (Exception e)
            {
                  // We got a problem...
                  System.out.println("Echec de connecion a Lipso Server "+ lipsoIp+ " :
"+ lipsoPort);
                e.printStackTrace();

            }
        */


            // Debut de lancement de         ENQUIRELINK. Inverser les commentaires !
```



```
        BIND();
            /*
            boolean flag = ENQUIRE_LINK();
            System.out.println(" EnquireLink falg is "  + flag);
            if (flag ==false)
                  {
                  BIND();
                        System.out.println("Envoie de BIND due to ENQUIRELINK
Failure");
                        outFile.println(" Envoie de BIND due to ENQUIRELINK Failure ");
                  }
            */

                //Fin de Lancement de ENQUIRELINK

            MobileActivUsers  = new int         [MobileUsers_num.length];
            users                         = new String      [users_ids.length];
            MobileUsers             = new String      [MobileUsers_num.length];

        for (int i = 0; i < MobileUsers_num.length ; i++)
            {
                  MobileActivUsers [i]    = 0;
                        users[i]                        = users_ids[i];
                        MobileUsers[i]                = MobileUsers_num[i];
            }

            connected   = true;
            return true;
      }

/**
*
*
*/
public void end_auction ()
      {
            UNBIND();
            try
                  {
                        outFile.close();
                  }
            catch (Exception e)
                  {
                  System.out.println("Failed to close the log file");
                  e.printStackTrace();
            }

        connected=false;

      }

/**
*
*
*/
public void push (String[] users, int durationTimeout, String productName, double
reservePrice, int quant, Bridge api)
      {
            _api =api;
            bidmin = reservePrice ;
            quantity = quant;
```



```
            List maListe = new ArrayList();
            String reponse ;
            for (int i=0; i < MobileUsers.length; i++)
                  {
                        String mob_user =MobileUsers[i];
                        maListe.add(mob_user);
                  }
            /*
            Date deadline     = new Date(System.currentTimeMillis());
            int minutes       = deadline.getMinutes();
            int hours         = deadline.getHours();
            minutes           += durationTimeout ;
            int nbr_heures    = durationTimeout / 60 ;
            int nbr_minutes   = durationTimeout - ( nbr_heures * 60);
            minutes           += nbr_minutes ;
            hours                   = hours + nbr_heures;
            */
            rightNow = Calendar.getInstance();
            rightNow.add(Calendar.SECOND,durationTimeout);

            System.out.println("Jean Leloup 5 mar.04 Place des Arts 20h. " + quant +"
billets mezzanine + backstage + souper Mise minimale : " + reservePrice +"$ Cloture:"+
rightNow.getTime().toString() + " ET: Envoyer un SMS ");
            reponse           ="Jean Leloup 5 mar.04 Place des Arts 20h. " + quant +"
billets mezzanine + backstage + souper Mise minimale : " + reservePrice +"$ Cloture:" +
rightNow.getTime().toString() ;

            SUBMIT_SM(reponse, maListe);
      }

/**
*
*
*/
public void send_sms (String message, int user_id) {

      List maListe = new ArrayList();
      maListe.add(MobileUsers[user_id]);
      SUBMIT_SM(message, maListe);
      return; // return ??
      }

/**
*
*
*/
public void listen (Bridge api)
      {

            _api =api;
        System.out.println("Start Listen");
        // submit orders
        System.out.println("[" + System.currentTimeMillis() + "] Submitting
orders....");
        //for (int i = 0; i  MobileUsers.length; i++) {
            if (connected)
                  {
                  for (int i = 0; i < 2 ; i++)
                        {
                              if (!is_activ (i))
                                    {
                                          try {
```



```
                                                      // Valider la mise
                                                            System.out.println("Merci,
votre mise a ete acceptee. Attendez un message d'avis sur l'etat d'avancement !");

      _api.submitOrder(users[i],(double) quantity,(double) (12+i));
                                                      activate_user (i);
                                                      }
                                                catch (Exception e)
                                                      {

                                                System.out.println("Submit order failed
for " + users[i]);
                                                e.printStackTrace();
                                                      }

                                    }
                                    else{
                                                System.out.println("Send Desole, Vous
avez deja fait une mise");

                                          }
                        }
                  System.out.println("[" + System.currentTimeMillis() + "] Finished
submitting orders....");
                  }
            else
                  {
                        System.out.println("Send Desole, Enchere deja termine");
                  }
      }

/**
*
*
*/
public boolean is_activ (int i)
      {
            if ( MobileActivUsers [i] == 1)
            {
                  return true ;
            }
            else{ // else inutile
                  return false;
            }
      }

/**
*
*
*/
public void activate_user (int i)
      {
            MobileActivUsers [i] = 1;

      }

/**
*
*
*/
public void BIND()
      {
```



```
            request_id_send++;

            XML_Request = headMessage;
            XML_Request = XML_Request + '"';
            XML_Request = XML_Request +request_id_send;
            XML_Request = XML_Request + '"';
            XML_Request = XML_Request.concat(">");
            XML_Request = XML_Request.concat ("<BIND CLIENT_ID=");
            XML_Request = XML_Request + '"';
            XML_Request = XML_Request.concat(client_id);
            XML_Request = XML_Request + '"';
            XML_Request = XML_Request.concat(" PASSWORD=");
            XML_Request = XML_Request + '"';
            XML_Request = XML_Request.concat(password);
            XML_Request = XML_Request + '"';
            XML_Request = XML_Request.concat("/></MCP>");
            XML_Request =
entete(Integer.toString(XML_Request.length())).concat(XML_Request);

            try
                  {
                        System.out.println("Envoi de Bind  : " + XML_Request);
                        theWriter.print(XML_Request);
                        theWriter.flush();

                        outFile.println(" BIND Send " + XML_Request);
                        BIND_RESP_trait(request_id_send);
                  }
            catch (Exception e)
                  {
                        e.printStackTrace();
                  }

      }

/**
*
*
*/
public void BIND_RESP_trait(int request_id_send)
      {
            String messageReceived = readBuffer();
            if (messageReceived.equals(""))
                  {
                        my_SESSION_ID="0000";
                        System.out.println("\n\n\t Session problem to receive
BIND_RESPn\n");
                  }
            else{
                        messageReceived=messageReceived.substring(7);
                        if (messageReceived.startsWith(headMessageRec)==false)
                              {
                                    System.out.println("\n\n\t--------- > Message not
correct ! :"+ messageReceived );
                              }
                        else{
                                    outFile.println(" BIND Response received" +
messageReceived);

                                    String cote                   = ""+'"';
                                    StringTokenizer t             = new StringTokenizer
(messageReceived,cote);
```

```
                                    int nbr_token                 = t.countTokens();
                                    t.nextToken();
                                    t.nextToken();
                                    t.nextToken();
                                    int request_nbr         =
Integer.parseInt(t.nextToken());
                                    String tag1                   = "";
                                    String command_status   = "";
                                    String tag2                   = "";
                                    String session_nbr            = "";
                                    if (nbr_token>4)
                                          {
                                            tag1                        =
t.nextToken();
                                          }
                                    if (nbr_token>5)
                                          {
                                            command_status  = t.nextToken();
                                          }
                                    if (nbr_token>6)
                                          {
                                            tag2                        =
t.nextToken();
                                          }
                                    if (nbr_token>7)
                                          {
                                            session_nbr           = t.nextToken();
                                          }

                                    if ((tag1.equals(ERROR)) && (tag2.equals("
REASON=")) )
                                          {
                                                System.out.println("BIND_RESP_trait->
ERROR MESSAGE RETURNED THE REASON IS  " +session_nbr);
                                          }
                                    else{
                                                if ((tag1.equals(BIND_RESP)) &&
(tag2.equals(" SESSION_ID=")) )
                                                      {
                                                            if
(request_id_send!=request_nbr)
                                                                  {

      System.out.println("Request code received unknown : " + request_nbr + " The
actual request is :" + request_id_send);
                                                                  }
                                                            else{
                                                                        if
(command_status.equals("0"))
                                                                              {

      my_SESSION_ID=session_nbr;
                                                                              }
                                                                        else{
                                                                                    if
(command_status.equals("81"))

      {

            System.out.println("Session already opened");

      }
```



```
	else{
		if (command_status.equals("89"))
			{
				System.out.println("Authentification failure");
			}
		else{
				if (command_status.equals("-1"))
					{
						System.out.println("Error unknown, contact Lipso");
					}
				else{
						System.out.println("Error not handled; Status = " +
command_status);
					}
			}
	}
                                                                        }
                                                                }
                                                        }
                                                else{
                                                                System.out.println("\n\n\t-
-------- > Bind_Resp_Trait -> Message not expected ! :\n\n" + messageReceived );
                                                        }
                                                }
                                        }
                                }
                        }
                }
        }

/**
*
*
*/
public void UNBIND()
        {
                request_id_send++;

                XML_Request = headMessage;
                XML_Request = XML_Request + '"';
                XML_Request = XML_Request + request_id_send;
                XML_Request = XML_Request + '"';
                XML_Request = XML_Request.concat(">");
                XML_Request = XML_Request.concat ("<UNBIND /></MCP>");
                XML_Request =
entete(Integer.toString(XML_Request.length())).concat(XML_Request);
                try
                        {
```



```
                        System.out.println("Envoi de UNBind -> "+ XML_Request);
                        theWriter.print(XML_Request);
                        theWriter.flush();
                        UNBIND_RESP_trait(request_id_send);
                  outFile.println("\n  UNBIND Sent "+ XML_Request );
                }
            catch (Exception e)
                {
                        e.printStackTrace();
                        System.out.println("UNBind Failed "+ XML_Request);
                  outFile.println("\n  Failed UNBIND "+ XML_Request );
                }
      }

/**
*
*
*/
public void UNBIND_RESP_trait(int request_id_send)
      {

            String messageReceived  = readBuffer();
            if (messageReceived.equals(""))
                {
                        my_SESSION_ID     = "0000";
                        System.out.println("\n\n\t Session problem to receive
BIND_RESPn\n");
                }
            else
                {
                        outFile.println("\n  UNBIND Response received"+
messageReceived);

                        my_SESSION_ID     = "0000";
                        messageReceived=messageReceived.substring(7);

                        if (messageReceived.startsWith(headMessageRec)==false){
                                    System.out.println("\n\n\t--------- > Message not
correct ! :\n\n"+ messageReceived );
                              }
                        else{
                                    String cote = ""+'"';
                                    StringTokenizer t = new StringTokenizer
(messageReceived,cote);
                                    int nbr_token= t.countTokens();
                                    t.nextToken();
                                    t.nextToken();
                                    t.nextToken();
                                    int request_nbr = Integer.parseInt(t.nextToken());
                                    String tag1= "";
                                    String command_status="";
                                    if (nbr_token>4)
                                          {
                                            tag1= t.nextToken();
                                          }
                                    if (nbr_token>5)
                                          {
                                            command_status= t.nextToken();
                                          }
                                    if (tag1.equals(ERROR))
                                          {
                                                System.out.println("\n ERROR MESSAGE
```



```
RETURNED in  UNBIND_RESP_trait Line 1176 : THE REASON IS  " + command_status);
                                        }
                                  else{
                                              if (tag1.equals(UNBIND_RESP))
                                                    {
                                                          if
(request_id_send!=request_nbr)
                                                                {

      System.out.println("Request code received unknown : " +request_nbr + " The actual
request is :" + request_id_send);
                                                                }
                                                          else
                                                                {
                                                                      if
(command_status.equals("0")==false)
                                                                            {

      System.out.println("Command Status unknown in UNBIND_RESP request :
"+command_status);
                                                                            }
                                                                }
                                                    }
                                              else{
                                                          System.out.println("\n\n\t-
-------- > UNBind_Resp_Trait -> Message not expected ! :\n\n"+ messageReceived );
                                                    }
                                        }
                                  }
                        }
            endSocket();
      }

// appel fonction envoie  message (content, liste d'adresse);
/**
*
*
*/
public void SUBMIT_SM (String message, List adrList)
      {
            request_id_send++;

            XML_Request = headMessage;
            XML_Request = XML_Request + '"';
            XML_Request = XML_Request+request_id_send;
            XML_Request = XML_Request + '"';
            XML_Request = XML_Request.concat(">");

            XML_Request = XML_Request.concat ("<SUBMIT_SM SERVICE_TYPE=");
            XML_Request = XML_Request + '"';
            XML_Request = XML_Request.concat ("");
            XML_Request = XML_Request + '"';

            XML_Request = XML_Request.concat(" SOURCE_ADDR=");
            XML_Request = XML_Request + '"';
            XML_Request = XML_Request.concat(source_adr);
            XML_Request = XML_Request + '"';
            XML_Request = XML_Request.concat (" OVERFLOW_ACTION=");
            XML_Request = XML_Request + '"';
            XML_Request = XML_Request.concat ("SPLIT");
            XML_Request = XML_Request + '"';
      /*
```



```
            XML_Request = XML_Request.concat("><DESTINATION ADDRESS=");
            XML_Request = XML_Request + '"';
            XML_Request = XML_Request.concat("16472260270");
            XML_Request = XML_Request + '"';
            XML_Request = XML_Request.concat("/>");
      */

            for (int i = 0; i < adrList.size(); i++)
                  {
                        String adresse = (String)adrList.get(i);
                        XML_Request = XML_Request.concat("><DESTINATION ADDRESS=");
                        XML_Request = XML_Request + '"';
                        XML_Request = XML_Request.concat(adresse);
                        XML_Request = XML_Request + '"';
                        XML_Request = XML_Request.concat("/>");
                  }


            XML_Request = XML_Request.concat("<SHORT_MESSAGE>");
            XML_Request = XML_Request.concat(message);
            XML_Request = XML_Request.concat("</SHORT_MESSAGE>");
            XML_Request = XML_Request.concat("</SUBMIT_SM></MCP>");
            XML_Request =
entete(Integer.toString(XML_Request.length())).concat(XML_Request);

            try
                  {
                        System.out.println("Envoi de SUBMIT : "+ XML_Request);
                        theWriter.print(XML_Request);
                        theWriter.flush();
                    outFile.println("\n  SUBMIT_SM Send "+ XML_Request );
                        SUBMIT_SM_RESP_trait(request_id_send);
                  }
            catch (Exception e)
                  {
                        e.printStackTrace();
                  }
      }

/**
*
*
*/
public void SUBMIT_SM_RESP_trait(int request_nbr_send)
      {

      //       System.out.println("\n\t --------------- Attente de
SUBMIT_SM_RESP_trait");
      //Test = 1;
      String messageReceived = readBuffer();
      if (messageReceived.equals(""))
            {
                  my_SESSION_ID     = "0000";
                  System.out.println("\n\n\t Session problem to recieve BIND_RESPn\n");
            }
      else
            {
                  outFile.println(" SUBMIT_SM Response received" + messageReceived);
                  System.out.println("\n\tAcknowlegement received from Lipso for
SUBMIT_SM: " + messageReceived);
                  messageReceived   = messageReceived.substring(7);
```



```
      if (messageReceived.startsWith(headMessageRec)==false)
            {
                  System.out.println("\n\n\t--------- > Message not correct ! :
SUBMIT_SM_RESP_trait\n\n"+ messageReceived );
            }
            else{
                  String cote                   = ""+'"';
                  StringTokenizer t             = new StringTokenizer
(messageReceived,cote);
                  int nbr_token                 = t.countTokens();
                  t.nextToken();
                  t.nextToken();
                  t.nextToken();

                  int request_nbr         = Integer.parseInt(t.nextToken());
                  String tag1                   = "";
                  String command_status   = "";
                  if (nbr_token > 4)
                        {
                          tag1                        = t.nextToken();
                        }
                  if (nbr_token > 5)
                        {
                          command_status  = t.nextToken();
                        }
                  if (tag1.equals(ERROR))
                        {
                              System.out.println("ERROR MESSAGE RETURNED THE REASON IS
" + command_status);
                        }
                  else{
                              if (tag1.equals(SUBMIT_SM_RESP))
                                    {
                                          if (request_id_send != request_nbr)
                                                {
                                                      System.out.println("Request code
received unknown : " +
                                                      request_nbr + " The actual
request is :" +
                                                      request_id_send);
                                                }
                                          else{
                                                if (command_status.equals("0"))
                                                      {
                                                            System.out.println("Our
message was delivred");
                                                      }
                                                else{
                                                            System.out.println("Unkmown
status for acknlowdgement dilevry : " + command_status);
                                                      }
                                                }
                                    }
                              else{
                                          System.out.println("\n\n\t--------- >
SUBMIT_SM_Resp_Trait -> Message not expected ! :\n\n" + messageReceived );
                                    }
                        }
                  }
            }
      }
```



```
/**
*
*
*/
public void BIND_trait(String id, String passwd, int request_nbr)
	{

		System.out.println("Bind_trait will call Bind_resp" );
		BIND_RESP('0',request_nbr);
		//else
		//System.out.println("Mot de passe ou compte invalide : " + passwd+ " : " +
client_id);
	}


/**
*
*
*/
public void ENQUIRE_LINK_trait(int request_nbr)
	{
		ENQUIRE_LINK_RESP("0", request_nbr);
	}

/**
*
*
*/
public boolean ENQUIRE_LINK()
	{

		request_id_send++;

		XML_Request = headMessage;
		XML_Request = XML_Request + '"';
		XML_Request = XML_Request+request_id_send;
		XML_Request = XML_Request + '"';
		XML_Request = XML_Request.concat(">");
		XML_Request = XML_Request.concat ("<ENQUIRE_LINK/>");
		XML_Request = XML_Request.concat("</MCP>");
		XML_Request =
entete(Integer.toString(XML_Request.length())).concat(XML_Request);

		try
			{
				System.out.println("Envoi de  ENQUIRE_LINK : "+ XML_Request);
				theWriter.print(XML_Request);
				theWriter.flush();
				  outFile.println(" ENQUIRE_LINK Sent"+ XML_Request);
				boolean flag = ENQUIRE_LINK_RESP_trait(request_id_send);
				return flag;
			}
		catch (Exception e)
			{
				e.printStackTrace();
				outFile.println(" Failed ENQUIRE_LINK "+ XML_Request);
				return false;
			}

	}

/**
```



```
*
*
*/
public boolean ENQUIRE_LINK_RESP_trait(int request_id_send)
      {
            String messageReceived = readBuffer();
            if (messageReceived.equals(""))
                  {
                        my_SESSION_ID="0000";
                        System.out.println("\n\n\t Session problem to recieve
BIND_RESPn\n");
                        return false;
                  }
            else
                  {
                        outFile.println(" ENQUIRE_LINK Response received"+
messageReceived);
                        System.out.println("\n\tENQUIRE_LINK Response  received from
Lipso : " + messageReceived);
                        messageReceived=messageReceived.substring(7);
                      if (messageReceived.startsWith(headMessageRec)==false)
                              {
                                    System.out.println("\n\n\t--------- > Message not
correct ! ENQUIRE_LINK_RESP_trait :\n\n"+ messageReceived );
                                    outFile.println(" Message not correct !
ENQUIRE_LINK_RESP_trait "+ messageReceived);
                                    return false;
                              }
                        else{

                                    String cote                   = ""+'"';
                                    String command_status   = "";
                                    StringTokenizer t             = new StringTokenizer
(messageReceived,cote);

                                    int nbr_token                 = t.countTokens();
                                    t.nextToken();
                                    t.nextToken();
                                    t.nextToken();

                                    int request_nbr         =
Integer.parseInt(t.nextToken());
                                    String tag1                   = "";
                                    if (nbr_token>4)
                                          {
                                                tag1= t.nextToken();
                                          }

                                    if (nbr_token>5)
                                          {
                                                command_status= t.nextToken();
                                          }
                                    if (tag1.equals(ERROR))
                                          {
                                                System.out.println("ERROR MESSAGE
RETURNED THE REASON IS  " + command_status);
                                                outFile.println(" ERROR MESSAGE
RETURNED THE REASON IS  " + command_status);
                                                return false;
                                          }
                                    else{
```

```
                                                if (tag1.equals(ENQUIRE_LINK_RESP))
                                                      {
                                                            if
(request_id_send!=request_nbr)
                                                                  {

      System.out.println("Request code received unknown : " +request_nbr + " The actual
request is :" + request_id_send);

      outFile.println("Request code received unknown : " +request_nbr + " The actual
request is : " + request_id_send);
                                                                        return false;
                                                                  }
                                                            else
                                                                  {

System.out.println("Request code received ENQUIRE_LINK_RESP_trait : " +request_nbr + "
The actual request is :" + request_id_send);

outFile.println("Request code received ENQUIRE_LINK_RESP_trait : " +request_nbr + " The
actual request is : " + request_id_send);
                                                                              return
true;
                                                                  }
                                                      }
                                                else{
                                                            System.out.println("\n\n\t-
-------- >Message not expected ! ENQUIRE_LINK_RESP_trait:\n\n"+ messageReceived );
                                                            outFile.println("Message
not expected ! ENQUIRE_LINK_RESP_trait:\n\n"+ messageReceived );
                                                            return false;
                                                      }
                                          }
                                    }
                              }
      }

/**
*
*
*/
public void BIND_RESP(char status, int request_nbr)
      {
            XML_Request             = headMessage;
            XML_Request             = XML_Request + '"';
            XML_Request             = XML_Request + request_nbr;
            XML_Request             = XML_Request + '"';
            XML_Request             = XML_Request.concat(">");
            XML_Request             = XML_Request.concat ("<BIND_RESP
COMMAND_STATUS=");
            XML_Request             = XML_Request + '"';
            XML_Request             = XML_Request + status;
            XML_Request             = XML_Request + '"';
            XML_Request             = XML_Request.concat(" SESSION_ID=");
            XML_Request             = XML_Request + '"';
            // To affect a sessions ID for Lipso
            // Random Gen = new Random();
            // int session_id =Gen.nextInt();
            SESSION_ID_LIPSO  = "1000";
            XML_Request             = XML_Request.concat (SESSION_ID_LIPSO);
            XML_Request             = XML_Request + '"';
            XML_Request             =        XML_Request.concat("/></MCP>");
```



```
            XML_Request             =
entete(Integer.toString(XML_Request.length())).concat(XML_Request);

            // Test output
            try
                  {
                        theServerWriter.print(XML_Request);
                        theServerWriter.flush();
                        outFile.println("\n  BIND_RESP Sent " + XML_Request );
                        System.out.println("The Bind_Resp sent ");
                  }
            catch (Exception e)
                  {
                        e.printStackTrace();
                        System.out.println("Failed Bind_Resp -> " + XML_Request);
                        outFile.println(" Failed BIND_RESP " + XML_Request );
                  }

      }

/**
*
*
*/
public void UNBIND_RESP(String status, int request_nbr)
      {

            XML_Request = headMessage;
            XML_Request = XML_Request + '"';
            XML_Request = XML_Request +  request_nbr;
            XML_Request = XML_Request + '"';
            XML_Request = XML_Request.concat(">");
            XML_Request = XML_Request.concat ("<UNBIND_RESP COMMAND_STATUS=");
            XML_Request = XML_Request + '"';
            XML_Request = XML_Request + status;
            XML_Request = XML_Request + '"';
            XML_Request = XML_Request.concat("/></MCP>");
            XML_Request =
entete(Integer.toString(XML_Request.length())).concat(XML_Request);

            // Test output
            System.out.println("The UNBind_Resp will be send -> "+ XML_Request);
            try   {

                        theServerWriter.print(XML_Request);
                        theServerWriter.flush();
                        outFile.println(" UNBIND_RESP Sent "+ XML_Request );
                  }
            catch (Exception e)
                  {
                        e.printStackTrace();
                    outFile.println(" Failed UNBIND_RESP "+ XML_Request );
                  }
      }

/**
*
*
*/
public void   SUBMIT_SM_RESP (String status, int request_nbr)
      {
```



```
            XML_Request = headMessage;
            XML_Request = XML_Request + '"';
            XML_Request = XML_Request+ request_nbr;
            XML_Request = XML_Request + '"';
            XML_Request = XML_Request.concat(">");
            XML_Request = XML_Request.concat ("<SUBMIT_SM_RESP COMMAND_STATUS=");
            XML_Request = XML_Request + '"';
            XML_Request = XML_Request + status;
            XML_Request = XML_Request + '"';
            XML_Request = XML_Request.concat("></SUBMIT_SM_RESP></MCP>");
            XML_Request =
entete(Integer.toString(XML_Request.length())).concat(XML_Request);

            try   {
                        theServerWriter.print(XML_Request);
                        theServerWriter.flush();
                        outFile.println("\n SUBMIT_SM_RESP sent"+ XML_Request );
                  }
            catch (Exception e)
                  {
                        e.printStackTrace();
                        outFile.println(" Failed SUBMIT_SM_RESP "+ XML_Request );
                  }

      }

/**
*
*
*/
public void  ENQUIRE_LINK_RESP (String status, int request_nbr)
      {

            XML_Request = headMessage;
            XML_Request = XML_Request + '"';
            XML_Request = XML_Request + request_nbr;
            XML_Request = XML_Request + '"';
            XML_Request = XML_Request.concat(">");
            XML_Request = XML_Request.concat ("<ENQUIRE_LINK_RESP COMMAND_STATUS=");
            XML_Request = XML_Request + '"';
            XML_Request = XML_Request + status;
            XML_Request = XML_Request + '"';
            XML_Request = XML_Request.concat("/></MCP>");
            XML_Request =
entete(Integer.toString(XML_Request.length())).concat(XML_Request);

            try   {
                        theServerWriter.print(XML_Request);
                        theServerWriter.flush();
                        outFile.println(" ENQUIRE_LINK_RESP Sent " + XML_Request );
                  }
            catch (Exception e)
                  {
                        e.printStackTrace();
                        outFile.println(" Failed ENQUIRE_LINK_RESP " + XML_Request );
                  }

            // Test output
            System.out.println("ENQUIRE_LINK_RESP is -> "+ XML_Request);
      }
```



```
/***
 Connector Mode
 */
public void UNBIND_trait(int request_nbr)
	{
		// System.out.println("UNBIND_trait will call UNBIND_RESP ! ");
		if (SESSION_ID_LIPSO.equals("0000"))
			{
				System.out.println("Session not open to handle Unbind ! ");
				UNBIND_RESP("-1",request_nbr);
			}
		else{
				SESSION_ID_LIPSO="0000";
				UNBIND_RESP("1",request_nbr);
			}
	}

/**
*
*
*/
public void SUBMIT_SM_trait(String service,String source_addr,String message, int
request_nbr)
	{

		outFile.println("  SUBMIT_SM received"+ message);
		SUBMIT_SM_RESP("0",request_nbr);
		List maListe	= new ArrayList();
		String reponse	= "";
		maListe.add(service);
		if (!connected)
			{
				reponse ="Desole, Enchere est deja termine";
				SUBMIT_SM(reponse, maListe);
				return;
			}
		else{
				int i =0;
				boolean userfound = false;
				for (int k=0; k < MobileUsers.length ; k++)
					{
						if (service.equals(MobileUsers[k]))
							{
								userfound = true;
								i = k ;
								break;
							}
					}
				if (userfound == false)
					{
						reponse =   "Desole, Vous n'etes pas un user
enregistre" + source_addr + " " + MobileUsers.length ;
						SUBMIT_SM(reponse, maListe);
						return;
					}
				int bid=0;
				System.out.println("Mise Lu a partir de " +   message );

				try   {
						bid   =     Integer.parseInt(message);
						if (bid < 0) throw new
NumberFormatException("erreur nombre         negatif");
```



```
                        }
                  catch (NumberFormatException e) // la mise reçu n'est pas un
chiffre
                        {
                              bid =0;
                        }

                  if (!is_activ (i))
                        {
                              if (bid <= (int)bidmin)
                                    {
                                          reponse="Desole, vous n'avez pas fait
une mise valide !\n"+
                                                "La mise minimale est : " +
bidmin + "$";
                                          System.out.println("Mise rejette " +
bidmin +  "  sa mise est " + bid );
                                          SUBMIT_SM(reponse, maListe);
                                          return;
                                    }
                              else{


                                          if
(Calendar.getInstance().after(rightNow))
                                                {
                                                      reponse = "Desole!
l'enchere est deja terminee";
                                                }
                                          else{
                                                      reponse="Merci! Votre mise
a ete acceptee! Attendez un message d'avis sur l'etat d'avancement!";
                                                }

                                    try {


      _api.submitOrder(users[i],(double)quantity,(double)bid);
                                          System.out.println(""+ users[i]);
                                          activate_user (i);
                                                SUBMIT_SM(reponse,
maListe);
                                                return;
                                    }
                                          catch (Exception e)
                                                {
                                          System.out.println("Submit order failed
for "+ users[i]);
                                          e.printStackTrace();

                                                reponse="Desole! Un
probleme est survenu au GNP, essayer de nouveau!";

                                                SUBMIT_SM(reponse,
maListe);
                                                return;
                                    }
                                    }
                        }
                  else{
                              reponse="Desole, Vous avez deja fait une mise";
                                    SUBMIT_SM(reponse, maListe);
```



```
                                    return;
                                }
                        }
                }

/**
* Connection methods :
*
*/
public boolean initServerSocket()
        {
                try   {
                                mobiticketSocket = new
ServerSocket(marchandPort,15,InetAddress.getLocalHost());
                                return true ;
                                /* A titre de test client
                                System.out.println("\nServer receving on Port: " +
mobiticketSocket.getLocalPort() + mobiticketSocket.toString() +
mobiticketSocket.getInetAddress());
                                // Attendre une connexion entrante
                          System.out.println(">>> Pret");
                        Socket client = mobiticketSocket.accept();
                          System.out.print("   Connexion recue de :");
                          System.out.println(" " + client.getInetAddress());
                      BufferedReader monEntree = new BufferedReader(new
InputStreamReader(client.getInputStream()));
                      PrintWriter maSortie = new PrintWriter  (new
OutputStreamWriter(client.getOutputStream()));
                      String line = monEntree.readLine();
                      System.out.println("> Recu: " + line) ;
                      maSortie.println("Echo: " + line);
                      maSortie.flush();
                      monEntree.close();
                      maSortie.close();

                                A titre de test client */


                        }
                catch (Exception e)
                        {
                                System.out.println(" !!!!!!!!!!!!!!! Erreur a la creation du
socket server : ");
                                e.printStackTrace();
                                endServerSocket();
                                return false;
                        }
        }

/**
*
*
*/
public   void endServerSocket()
        {
                try {
                        //mobiticketSocket.close();
                        theServerWriter.close();
                        theServerReader.close();
                        }
                catch (Exception e)
                        {
```



```
                        System.out.println("Erreur a la fermeture du socket server :
");
                        e.printStackTrace();

                  }
      }

/**
*
*
*/
public void initSocket()
      {
            try {
                        theSocket = new Socket(lipsoIp, lipsoPort);
                        theWriter = new PrintWriter(theSocket.getOutputStream(), true);
                        theReader = new BufferedReader(new
InputStreamReader(theSocket.getInputStream()));
                        System.out.println("Lipso Started");
                  }
            catch (Exception e)
                  {
                        System.out.println("Echec de connecion a Lipso Server "+
lipsoIp+ " : "+ lipsoPort);
                        e.printStackTrace();
                  }
      }

/**
*
*
*/
public void endSocket()
      {
            try {
                  theReader.close();
                  theWriter.close();
                  theSocket.close();
            System.out.println("Lipso Stopped");
                  }
            catch (Exception e)
                  {
                        e.printStackTrace();
                  System.out.println("Failed : Lipso Stopped by the end of
UNBIND_trait");
                  }
      }

      /**
      * innerclass de lipso
      *
      */
      class deamonreceiveMessage extends Thread{

            /**
            *
            *
            */
            public void run()
                  {
                        boolean unbind                = false;
                        boolean              coonectionOk      = initServerSocket();
```



```
                        if (coonectionOk != true )
                              {
                                    System.out.println("\t Opening Server Connection
failed !!!!!! ");
                                    System.exit(0);
                              }

                        while (true)
                              {
                                    try {
                                                unbind                  = false;

                                                System.out.println("\t Ouverture d'une
nouvelle session");
                                                Socket client     =
mobiticketSocket.accept();
                                                theServerReader = new
BufferedReader(new InputStreamReader(client.getInputStream()));
                                                theServerWriter = new
PrintWriter(client.getOutputStream(), true);

                                                while (unbind==false)
                                                      {
                                                            String msg  =
readServerBuffer(); //theServerReader.readLine();

      System.out.println("\tMessage recu de Lipso " + msg );
                                                            unbind            =
switchMessage(msg);
                                                      }

                                                System.out.println("\t Fermeture de
session");
                                                endServerSocket();

                                          }
                                    catch (Exception e)
                                          {
                                                e.printStackTrace();
                                          }
                              }

                  }

            /**
            *
            *
            */
            public boolean switchMessage(String messageReceived)
                  {
                        outFile.println("  Message received" + messageReceived);
                        int tmpMessageaControler1;
                        int tmpMessageaControler2;
                        int tmpMessageaControler3;
                        int tmpMessageaControler4;
                        int tmpMessageaControler5;
                        int tmpMessageaControler6;
                        String request;
                        String val1       = "";
                        String val2       = "";
                        String val5       =  "";
```



```
                        messageReceived   = messageReceived.substring(7);

                        if (messageReceived.startsWith(headMessage)==false)
                              {
                                    System.out.println("\n\n\t--------- > Message not
correct ! :\n\n"+ messageReceived );
                              }
                        else{
                              if (messageReceived.indexOf("ENQUIRE_LINK") > 0)
                                    {
                                          tmpMessageaControler1 =
messageReceived.indexOf("_ID="); // debut du numero de client
                                          tmpMessageaControler2 =
messageReceived.indexOf('"' + ">"); // fin du numero de client

                                          request = composeChaine(messageReceived,
tmpMessageaControler1 + 5, tmpMessageaControler2);

                                          // valider si request est bien un int

      ENQUIRE_LINK_trait(Integer.parseInt(request));
                                    }
                              else{
                                          if ((messageReceived.indexOf("BIND") > 0) &&
(messageReceived.indexOf("PASSWORD=") > 0) )
                                                {
                                                      tmpMessageaControler1   =
messageReceived.indexOf("REQUEST_ID=");
                                                      tmpMessageaControler2   =
messageReceived.indexOf('"' + ">");
                                                      tmpMessageaControler3   =
messageReceived.indexOf("CLIENT_ID=");
                                                      tmpMessageaControler4   =
messageReceived.indexOf('"' + " PASSWORD");
                                                      tmpMessageaControler5   =
messageReceived.indexOf('"' + "/>");
                                                      request                       =
composeChaine(messageReceived, tmpMessageaControler1 + 12, tmpMessageaControler2); //
extrait le request ID
                                                      val1                          =
composeChaine(messageReceived, tmpMessageaControler3 + 11, tmpMessageaControler4); //
extrait le ID client
                                                      val2                          =
composeChaine(messageReceived, tmpMessageaControler4 + 12, tmpMessageaControler5); //
extrait le password
                                                      BIND_trait(val1, val2,
Integer.parseInt(request));

                                                }
                                          else{
                                                      if
(messageReceived.indexOf("UNBIND") > 0)
                                                            {
                                                                  tmpMessageaControler1
      = messageReceived.indexOf("_ID=");
                                                                  tmpMessageaControler2
      = messageReceived.indexOf('"' + ">");
                                                                  request
            = composeChaine(messageReceived, tmpMessageaControler1 + 5,
tmpMessageaControler2); // extrait le ID client
```



```
      //System.out.println("UNBIND : " + request);

                                                                  // valider si request
est bien un int

      UNBIND_trait(Integer.parseInt(request));
                                                                  return true;
                                                            }
                                                            else{
                                                                        if
((messageReceived.indexOf("SUBMIT_SM") > 0) && (messageReceived.indexOf("SOURCE_ADDR=")
> 0))
                                                                                    {

      tmpMessageaControler1 = messageReceived.indexOf("REQUEST_ID=");

      tmpMessageaControler2 = messageReceived.indexOf('"' + ">");

      tmpMessageaControler3 = messageReceived.indexOf("SOURCE_ADDR=");

      tmpMessageaControler4 = messageReceived.indexOf('"', tmpMessageaControler3 + 13);

      tmpMessageaControler5 = messageReceived.indexOf("DESTINATION ADDRESS=");

      tmpMessageaControler6 = messageReceived.indexOf('"' + "/>");

      request = composeChaine(messageReceived, tmpMessageaControler1    + 12,
tmpMessageaControler2); // extrait le request ID

      val1 = composeChaine(messageReceived, tmpMessageaControler3       + 13,
tmpMessageaControler4); // extrait le source Addresse

      val2 = composeChaine(messageReceived, tmpMessageaControler5       + 21,
tmpMessageaControler6); // extrait le destination Addresse

      tmpMessageaControler1 = messageReceived.indexOf("<SHORT_MESSAGE>");

      tmpMessageaControler2 = messageReceived.indexOf("</SHORT_MESSAGE>");

      val5 = composeChaine(messageReceived, tmpMessageaControler1 + 15,
tmpMessageaControler2); // extrait le court message

      SUBMIT_SM_trait(val1, val2, val5, Integer.parseInt(request));
                                                                                    }
                                                                        else{

      System.out.println("\n\n\t--------- > Message not yet traited ! :\n\n"+
messageReceived );
                                                                                    }
                                                                  }
                                                      }
                                          }
                                    }
                              return false;
                        }
            }

/**
*
*
*/
public String readBuffer ()
```



```
	{

		String line	= "";
		char c;
		int tmpChar;

		try {
				while (!theReader.ready())
					{
					//System.out.println("Reponse Lipso : " +
theReader.ready());
						Thread.sleep(60);
					}
				while (((tmpChar = theReader.read()) != -1) )
					{
						c		= (char)
Integer.parseInt(Integer.toString(tmpChar), 10);
						line  = line + c;
						//if (Test == 1)
						//{
						//System.out.println(line);
						//}
						if (!theReader.ready())
							{
								break;
							}
						//System.out.println("\n\tAcknowlegement received
from Lipso : " + line);
					}
				// System.out.println("\n\tAcknowlegement received from Lipso :
" + line);
			}
		catch (Exception e)
			{
				System.out.println("erreur a la lecture du stream ");
				e.printStackTrace();
				line  = "";
			}
		return line;
	}

/**
*
*
*/
public String readServerBuffer ()
	{

		String line = "";
		char c;
		int tmpChar;
		try {

			while (!theServerReader.ready())
				{
				//		System.out.println("Reponse Lipso : " +
theServerReader.ready());
				Thread.sleep(60);
				}
			while (((tmpChar = theServerReader.read()) != -1)  )
				{
```



```
                              c = (char) Integer.parseInt(Integer.toString(tmpChar),
10);
                              line = line + c;

                              if (!theServerReader.ready())
                                    {
                                          break;
                                    }
                        }

                  //System.out.println("Acknowlegement received from Lipso : "+line);
                  }
            catch (Exception e)
                  {
                        //System.out.println("erreur a la lecture du stream ");
                        //e.printStackTrace();
                  }
            System.out.println("Packet received from Lipso : "+line);
            return line;
      }

/**
* fonction qui remplie l'entete de "0" sur une longueur de 7
*
*/
public String entete (String lengthChaine){

      while (lengthChaine.length() < 7) lengthChaine = "0" + lengthChaine;

      return lengthChaine;

/*
      String tmpChaine= "";
      if (lengthChaine.length() == 1)
            {
                  tmpChaine = "000000" + lengthChaine;
            }
      else
            if (lengthChaine.length() == 2)
                  {
                        tmpChaine = "00000" + lengthChaine;
                  }
      else
            if (lengthChaine.length() == 3)
                  {
                        tmpChaine = "0000" + lengthChaine;
                  }
      else
            if (lengthChaine.length() == 4)
                  {
                        tmpChaine = "000" + lengthChaine;
                  }

      else if (lengthChaine.length() == 5)
      {
            tmpChaine = "00" + lengthChaine;
      }
      else if (lengthChaine.length() == 6)
      {
            tmpChaine = "0" + lengthChaine;
      }
      else if (lengthChaine.length() == 7)
```



```
	{
		tmpChaine = lengthChaine;
	}

	*/

	// return tmpChaine;

	}

/**
*
* si on peut tronquer sinon on peut ajouter des espaces à la fin du mot
* if (fin > maChaine.length()) fin = maChaine.length());
* String sortie = maChaine.substring(depart, (fin-depart));
*
*/
public String composeChaine(String maChaine, int depart, int fin)
	{

		String chaineSortie = "";

		for (int i = depart; i < fin; i++)
			{
				chaineSortie = chaineSortie + maChaine.charAt(i);
			}

		return chaineSortie;
	}
}
```

6.2. Code source complète `Bridge.java`

```
/**
*
* Nom du fichier: Bridge.java
* But: Communication avec la plate-forme de GNP2
* Date: 23 juillet 2004
* Projet: Enchère électronique mobile MobiTicket
* Auteurs: Christina Braz et Isabelle Therrien
* Institution: CIRANO.
*
**/

package cirano.gnp.tests;

import cirano.gnp.ContextUtil;
import cirano.gnp.Constants;
import cirano.gnp.GNPException;
import cirano.gnp.ContextUtil;
import cirano.gnp.Constants;
import cirano.gnp.GNPException;
import cirano.gnp.ejb.NegotiatorSessionRemote;
import cirano.gnp.ejb.NegotiatorSessionRemoteHome;
import cirano.gnp.ejb.AdministratorSessionRemote;
import cirano.gnp.ejb.AdministratorSessionRemoteHome;
import cirano.gnp.auctioneer.core.Adjudication;
import cirano.gnp.auctioneer.core.Announce;
import cirano.gnp.auctioneer.core.Quote;
import cirano.gnp.auctioneer.core.Order;
import cirano.gnp.auctioneer.core.Product;
```



```
import cirano.gnp.auctioneer.core.QuoteLineItem;
import cirano.gnp.auctioneer.core.Rules;
import cirano.gnp.auctioneer.core.NegotiationRules;
import cirano.gnp.auctioneer.core.PhaseRules;
import cirano.gnp.auctioneer.core.Property;
import cirano.gnp.auctioneer.core.SubOrderExpression;
import cirano.gnp.auctioneer.core.Status;
import cirano.gnp.auctioneer.core.OrderLineItem;

import java.util.*;
import java.io.*;
import javax.naming.Context;

public class Bridge {

    private static lipso                  com= new lipso();
    private NegotiatorSessionRemote       _negoSession;
    private AdministratorSessionRemote    _adminSession;
    private long                          _negoGPK;
    private long                          _productGPK;
    private int                           _roundNumber;
    private int                           _phaseNumber;
    private final String                  MARKET_NAME= "BridgeMarket";
    private double                        reservePrice;



      /**
      * Constructeur permet de construire deux sessions une pour l'administration
      * l'autre pour la negociation
      * @param NegotiatorSessionRemote    session1 une session negociation
      * @param AdministratorSessionRemote session2 une autre session admin
      */
    public Bridge(NegotiatorSessionRemote nego, AdministratorSessionRemote admin)
      {
              _negoSession    = nego;
              _adminSession   = admin;
           }

    /**
     * Submit a test announce
     * Set negoGPK and unique productGPK in properties
     * @param users array of usernames
     * @param durationTimeout in seconds, the duration time of this auction
     * @param productName a String that identifies the product
     * @param reservePrice the minimum price for adjudication
     * @param quantity number of units of the product
     * @exception GNPException if an error occurs
     */
    public void submitAnnounce(String[] users, int durationTimeout,
                                       String productName, double reservePrice,
                                       int quantity)

                                       throws GNPException
      {

                this.reservePrice = reservePrice; // initialisation de la valeur de
base.

             try {

                // Create and send the     announce
```

```
                    Announce announce       = createAnnounce(users, durationTimeout,
productName, reservePrice, quantity);
                    _negoGPK                = _negoSession.submitAnnounce(announce,
MARKET_NAME).longValue();

                    System.out.println("Announce sent");

                    // wait 5 seconds (because it was an asynchronous call)
                    // (is it enough?)
                    Thread.sleep(5000);

                    // get the productGPK (for future orders)
                    Quote quote             = _negoSession.getQuote(_negoGPK);
                    _phaseNumber            = quote.getCurrentPhaseNumber();
                    _roundNumber            = quote.getCurrentRound();

                    Set quoteLineItems      = quote.getQuoteLineItems();

                    // here, because I sent only one item in the anounce, I can suppose
this line is correct.
                    QuoteLineItem qli       =
(QuoteLineItem)quoteLineItems.iterator().next();
                    _productGPK             = qli.getProductGPK();

                    // here, with the qli, you could have plenty of information on the
product negotiated:
                    // leader price, leader name, range constraints, etc.

                    }
                catch (javax.ejb.FinderException fe)
                    {
                          throw new GNPException("ERROR: couldn't find a ressource to
build the negotiation",fe);
                    }
                catch (java.rmi.RemoteException re)
                    {
                          throw new GNPException("ERROR: couldn't access foreign
ressource",re);
                    }
                catch (javax.ejb.CreateException ce)
                    {
                          throw new GNPException("ERROR: couldn't create a
PersistentAuctioneer instance",ce);
                    }
                catch (javax.naming.NamingException ne)
                    {
                          throw new GNPException("ERROR: couldn't access foreign
ressource",ne);
                    }
                catch (java.lang.InterruptedException ie)
                    {
                          throw new GNPException("ERROR: Thread sleep interrupted!",ie);
                    }
                catch (cirano.gnp.GNPException gnpe)
                    {
                          throw gnpe;
                    }
      }

    /**
     * Submit an order on the unique product of the negotiation
     *
```



```
     * @param submitter a <code>String</code>: the user
     * @param quantity a <code>double</code>: the quantity
     * @param price a <code>double</code>: the price
     */

    public void submitOrder(String submitter, double quantity, double price) throws
Exception
      {
              // validate values received before processing
              // order
              Order order = Order.create(submitter);
              order.setNegoGPK(_negoGPK);
              order.setSentPhase(_phaseNumber);
              order.setSentRound(_roundNumber);
              order.setSide("BUY");

              // insert root operator
              SubOrderExpression soe = new SubOrderExpression(order);

              // build an OrderLineItem
              OrderLineItem oli = OrderLineItem.create(order, _productGPK);

              oli.setPrice(price);
              oli.setQuantity(quantity);
              soe.addOperand(oli);
              order.addSubOrderExpression(soe);
              _negoSession.submitOrder(order);

              System.out.println("User "+ submitter + " bids for "+ quantity + " units
at "+ price +"$.");
      }

    /**
     * Fonction qui permet de recuperer la Quote en cours selon le numero de negoGPK
     * @return the <code>Quote</code> for the submitted announce or null if there is
none
     * @exception javax.naming.NamingException if quoteHome cannot be found
     */
    public Quote readQuote() throws Exception
      {
            Quote quote = _negoSession.getQuote(_negoGPK);
            return quote;
      }

      /**
      * Tenir compte que si l'enchere n'est pas terminee ne pas faire d'adjudication
      *
      */
      public Adjudication readAdjudicationForUser(String user, int i) throws Exception
            {
            // verifier si l'enchere est bien terminee
            // il faudrait synchroniser alors pour relancer cette operation un peu plus
tard
            // if (readQuote().getStatus() == cirano.gnp.auctioneer.core.Status.CLOSED)
                  // {
                        String mesg     = "";
                        Collection adjs = _negoSession.getAdjudications(_negoGPK,
user);

                        //Mobile User loses VIP auction.
                        if (adjs.size() == 0)
                            {
```



```
                                        mesg = "Desole! Votre mise a ete insuffisante.
8575 Jean Leloup. Merci pour votre participation.";

                                        // Un usager ne connait les mises gagnantes des
autres (size() ==0)
                                        // il faut synchroniser une pile, si il n'est
pas gagnant,
                                        // on le met dans la pile
                                        // aviser d'abord le gagnant et depiler les
perdants ensuite.

                                    com.send_sms (mesg,i);
                                    System.out.println("No adjudication for user " +
user + ".");
                                    return null;

                                }
                            else { //Mobile User wins VIP auction. ici a trouver l'erreur
                                    if (adjs.size() == 1)
                                        {
                                            Adjudication adj =
(Adjudication)adjs.iterator().next();
                                            System.out.println("User " + user + "
wins " + adj.getQuantity() + " units at " + adj.getPrice() + "$");

                                            mesg ="Bravo! Vous avez gagne
l'enchere! 8575 Jean Leloup";
                                            mesg
=mesg.concat(String.valueOf(adj.getPrice()));
                                            mesg =mesg.concat("$ Valide jusqu'a
19h30 ET Salle Wilfrid-Pelletier 175 St-Catherine T:8422112");
                                            com.send_sms ( mesg,i);
                                            return adj;
                                        }
                                    else{
                                            System.err.println("Too many
adjudications for user " + user + " (" + adjs.size() + ").");
                                            return
(Adjudication)adjs.iterator().next();
                                        }
                                }

            //    }
            // else{

            //          System.out.println("l'enchere n'est pas terminee");

            //    }
      }


    /**
     * Read the adjudication of the negotiation and
     * return it to the caller; return the first one if there are many.
     * @return null if no adjudication exist for that user

    public Adjudication readAdjudicationForUser(String user, int i) throws Exception
      {

              String mesg     = "";
              Collection adjs = _negoSession.getAdjudications(_negoGPK, user);
```



```
              //Mobile User loses VIP auction.
              if (adjs.size() == 0)
                  {
                              mesg = "Enchere Jean Leloup. Votre mise a ete insuffisante.
La mise gangante :";
                              mesg = mesg.concat("100000");
                           mesg = mesg.concat("$");
                           com.send_sms (mesg,i);
                           System.out.println("No adjudication for user " + user + ".");
                           return null;

                  }
              else { //Mobile User wins VIP auction. ici a trouver l'erreur
                        if (adjs.size() == 1)
                              {
                                    Adjudication adj =
(Adjudication)adjs.iterator().next();
                                    System.out.println("User " + user + " wins " +
adj.getQuantity() + " units at " + adj.getPrice() + "$");

                                    mesg ="Vous avez gagne l'enchere! 8575 Jean
Leloup";
                                    mesg =mesg.concat(String.valueOf(adj.getPrice()));
                                    mesg =mesg.concat("$ Valide jusqu'a 19h30 ET Salle
Wilfrid-Pelletier 175 St-Catherine T:8422112");
                                    com.send_sms ( mesg,i);
                                    return adj;
                              }
                           else{
                                    System.err.println("Too many adjudications for user
" + user + " (" + adjs.size() + ").");
                                    return (Adjudication)adjs.iterator().next();
                              }
                  }
      }


      /**
      *
      *
      */

    public void closeNegotiation() throws GNPException, java.rmi.RemoteException
      {

            _adminSession.submitNegotiationCommand(_negoGPK,
Constants.CLOSE_PHASE_MSG);

      }

    /**
     * Create an announce where some attributes are hardcoded.
     *
     * @param users array of usernames
     * @param durationTimeout in seconds, the duration time of this auction
     * @param productName a String that identifies the product
     * @param reservePrice the minimum price for adjudication
     * @param quantity number of units of the product
     * @return an <code>Announce</code>
     * @exception GNPException if an error occurs
     */
    private Announce                          createAnnounce(String[] users, int
```



```
durationTimeout, String productName, double reservePrice, int quantity) throws
GNPException
      {

            System.out.println("Creating Announce...");

            Rules rules             = new Rules();
              NegotiationRules nr = new NegotiationRules();

              nr.setReferenceSide("SELL");

              for (int i = 0; i < users.length; i++)
                  {
                        nr.addPartyKey(users[i]);
                  }

              System.out.println("Added " + users.length + " users.");

              // add phase rules to nego rules
              PhaseRules pr         =  new PhaseRules();

              pr.setPhaseName                         ("competition");
              pr.addClassName                         (new Property("AUCTIONEER",
                  "cirano.gnp.auctioneer.core.SimpleAuctioneer"));
              pr.setAnnouncerAdjPriceEvolution("FORWARD");
              pr.setPriceIncrement              (1);
              pr.setPricingRule                       ("FIRST_PRICE");
              pr.addPartiesActivityRule         (new Property("MIN_ACTIVE_PRODUCTS",
            new Integer(1)) );
              pr.addOrderLineItemRanking        ("PRICE_DESCENDING");
              pr.addOrderLineItemRanking        ("TIME_DESCENDING");
              pr.addRoundEndCondition                 (new Property("DURATION_TIMEOUT",
                  new Integer(durationTimeout)));
              pr.addPhaseEndCondition                 (new Property("MAX_ROUNDS",
                  new Integer(1)) );
              pr.addRoundEndOption              (new
Property("SUBMIT_LAST_ROUND_ORDERS",      new Boolean(false)));

              nr.addPhaseRules(pr);
              rules.setNegotiationRules(nr);


              // Product
              Product product             = new Product();
              product.setName (productName);

              Vector products             = new Vector(1);

              products.add(product);
              System.out.println("Product " + productName + " added.");

              // Initial order
              Order order                 = Order.create("announcer");

              order.setSentPhase(1);
              order.setSentRound(0);
              order.setSide("SELL");

              // insert root operator
              SubOrderExpression soe      = new SubOrderExpression(order);

              // build an OrderLineItem
```



```
            OrderLineItem oli           = OrderLineItem.create(order, product);

            oli.setPrice(reservePrice);
            oli.setQuantity(quantity);

            soe.addOperand(oli);
            order.addSubOrderExpression(soe);

            System.out.println("Initial order created");

            // Create Announce
            Announce announce           = Announce.create(rules,products,order);
            announce.setNegotiationName("API Nego");
            announce.setNegotiationDescription("This is a test Negotiation");

            return announce;

      }

      /**
      * test
      *
      */
    public static void main(String[] args) {

        // Creation and Initialisation: DEMO

            /*try  {

             String message;


            BufferedReader stdin = new BufferedReader (new
InputStreamReader(System.in));
                  System.out.println ("Enter a line of text:");

                  message = stdin.readLine();

                  System.out.println ("You entered: \"" + message + "\"");
            }catch (Exception e){

                        e.printStackTrace();
            }
            */

            String MobileUsers[] = {"15145771965","15145772406"};
            //String MobileUsers[] = {"15149286809","1514444442","15144444443"};


        String users[] = {"u1","u2"};

        // int durationTimeout = 5;  //300 secondes=5 minutes!
        int durationTimeout = 180;  //300 secondes = 5 minutes!

        String productName = "toto";

        double reservePrice = 300.0;

        int quantity = 2;

      // Fin Initialisation: DEMO
```



```
            if (com.init(users,MobileUsers) ==false)
                  {
                  System.out.println("Initialization of the Lipso communication failed.
End of program.");
                        return ;
                  }
        Bridge api = null;

        java.util.Timer scheduler = new java.util.Timer();

        try {

            // Fetch a Context
            Context ctx = ContextUtil.getNamingContext();

            // Get a negotiator session home, for communication with GNP2
            NegotiatorSessionRemoteHome negoHome            =
(NegotiatorSessionRemoteHome)ctx.lookup("cirano.gnp.ejb.remote.NegotiatorSessionHome");
            NegotiatorSessionRemote negoSession             = negoHome.create();
            AdministratorSessionRemoteHome adminHome  =
(AdministratorSessionRemoteHome)ctx.lookup("cirano.gnp.ejb.remote.AdministratorSessionH
ome");
            AdministratorSessionRemote adminSession   = adminHome.create();

            api                                                         = new
Bridge(negoSession,adminSession);

            System.out.println("[" + new Date(System.currentTimeMillis()) + "]
Submitting announce...");


            // Dispatch of the announce.


api.submitAnnounce(users,durationTimeout,productName,reservePrice,quantity);

            // Send The push Request To Lipso
            com.push(users,durationTimeout,productName,reservePrice,quantity,api);

            // make sure the adjudications are asked for only after round terminates
            // scheduler.schedule(new AskForAdjudicationsTask(api,users,com),
durationTimeout * 60000);
            scheduler.schedule(new AskForAdjudicationsTask(api,users,com),
(durationTimeout * 1000)+90000);


            }
        catch (IllegalStateException e)
            {

                  System.out.println("[" + new Date(System.currentTimeMillis()) + "]
ERROR task already scheduled: " + e);

            }
        catch (Exception e)
            {

                  System.out.println("[" + new Date(System.currentTimeMillis()) + "]
ERROR while talking to GNP: " + e);
                  e.printStackTrace();
```



```
                  if (api != null)
                        {
                        System.out.println("[" + new Date(System.currentTimeMillis()) +
"] Closing negotiation...");
                        try {
                                    api.closeNegotiation();
                              }
                        catch (Exception e2)
                              {
                                    System.out.println("[" + new
Date(System.currentTimeMillis()) + "] Unable to close negotiation.");
                                    e2.printStackTrace();
                              }

                        }
            }

    }


      public double getReservePrice(){
            return reservePrice;
            }

}

/**
*
*
*/
class AskForAdjudicationsTask extends java.util.TimerTask
      {

          private Bridge _api                   = null;
          private String[] _users               = null;
          private lipso _lipso                  = null;
          private final int SLEEP_TIME    = 3 * 1000;  // time between checks for quote
update at the end of the round
          private final int TIMEOUT             = 60 * 1000;  // maximum time awaiting
for quote to be updated before considering an error occured


      /**
      *
      *
      */
      public AskForAdjudicationsTask(Bridge api, String[] users, lipso com)
            {
              _api=api;
              _users=users;
                  _lipso=com;
          }

      /**
      *
      *
      */
    public void run()
      {

              try {
```



```
                System.out.println("[AskForAdjudicationsTask:run]");

                Quote quote       = _api.readQuote();
                byte status       = quote.getStatus().toByte();

                int timeWaited    = 0;

                while (status != Status.NEGO_CLOSED && timeWaited < this.TIMEOUT)
                      {
                          System.out.println("[" + new
Date(System.currentTimeMillis()) + "] Negotiation is not CLOSED yet, waiting....");
                          Thread.sleep(SLEEP_TIME);
                          timeWaited += SLEEP_TIME;
                          quote  = _api.readQuote();
                          status = quote.getStatus().toByte();
                      } // end of while (status != Status.NEGO_CLOSED)

                System.out.println("[" + new Date(System.currentTimeMillis()) + "]
Reading quote \n"+quote);

                if (status == Status.NEGO_CLOSED)
                      {

                                  int k = _users.length ;
                          System.out.println("[" + new Date(System.currentTimeMillis()) +
"] Reading adjudications........" + k );
                          for (int i = 0; i < _users.length; i++)
                                {
                                        // Only active users.
                                        if (_lipso.is_activ(i)==true)
                                                {

      System.out.println(_users[i] + " is activer "  );

      _api.readAdjudicationForUser(_users[i],i);
                                                }
                                             else{
                                                        System.out.println("Not an
Active Mobile User " + _users[i]);
                                                }
                                }

                      }
                else{
                      System.out.println("[" + new Date(System.currentTimeMillis()) +
"] Negotiation is not CLOSED.  PROBLEM!!!");
                      } // end of else
                }
              catch (Exception e)
                {
                      System.out.println("[" + new Date(System.currentTimeMillis()) +
"] ERROR while talking to GNP****: "+e);
                    e.printStackTrace();
                }

                _lipso.end_auction();
                // System.exit(0);
      }
      }
```


7. Bibliographie

[ASP03] The ASPexchange.com *Advisor for ASP Solutions* (2003) http://www.theaspexchange.com/

[BER03] Bérubé, Jean-François. *GNP2 : Le guide du développeur version 2.0*, Centre interuniversitaire de recherche en analyse des organisations – CIRANO (2003).

[ERG04] Ergonomie On line Web Site. Ce site présente un ensemble de liens vers des sites concernant l'ergonomie des IHM (2002) http://membres.lycos.fr/ergoline/

[FID04] Microcell Télécomunications. *Site Web Fido*, marque de commerce des services numériques sans fil offerts par Microcell Solutions Inc. (2004).

[GSM04] GSM™ WORLD. GSM Technology *SMS (Short Message Service)* (2004) http://www.gsmworld.com/technology/sms/index.shtml

[JAV04] Sun Microsystems Inc. *The Java™ Tutorial*, Tutorials & Code Camps The Source for Developers (2004) http://java.sun.com/docs/books/tutorial/

[LIP01] Lipso Systems Inc. *DTD Document Type Definition* by Lipso. 2001.

[LIP04] Lipso Systems Inc. *Where is Mobile Message headed?* White-paper by Lipso. 2004.

[PLA04] Place des Arts *Place des Arts Site Web* (2004) http://www.pda.qc.ca

[ROB&BER03] Jacques Robert, vice-president E-Commerce Group, CIRANO and IT professor, HEC Jean-François Bérubé, research professional, CIRANO. *Auction Rules Description* (2001).

[VAC03] Vachon, Julie. *IFT6803 : Génie logicielle du commerce électronique (Chapitre 1 – Introduction, Processus de développement)*, Maîtrise en commerce électronique, Université de Montréal (2003) http://www.iro.umontreal.ca/~dift6803/Transparents/Chapitre1/chapitre1.1.pdf

[VIS04] VISUALTron Software Corporation. *Wireless Short Message Services (SMS)* Tutorials (2003) http://www.visualtron.com/wire_sms_topic04.htm

