GASP: un intergiciel pour les jeux en réseaux multijoueurs sur téléphones mobiles

PELLERIN Romain

Projet de Recherche: CNAM-Cedric, GET-INT et InfraWorlds

Plan

1. Le jeu sur mobile
2. Le jeu multijoueur sur mobile
3. Fonctionnalités d'une plateforme de jeux
4. Etat de l'art des plateformes de jeux pour mobile
5. Gaming Services Platform (GASP)
   - Modèle de données
   - Architecture
   - Interfaces
   - Modèle Evénementiel
   - Protocole MooDS
6. Conclusion et Perspectives

1. Le jeu sur mobile

- Langages de développement sur mobile

➢ **C++ / SymbianOS**

➢ **Java (J2ME)**

▪ **CLDC**

❖ **MIDP 1.0 et 2.0**

❖ **DoJa 1.5 et 2.5**

▪ **CDC**

❖ **Personal Java**

POUR	CONTRE
❖ **Performance** (2x plus rapide que MIDP)	❖ **Pas de portabilité** Uniquement sur Nokia series 60 (très homogène)
❖ **Standard/Ouvert** (Nokia, Siemens, Sony-Ericssion, Sharp, Alcatel, SAGEM, Samsung, etc.) tous les opérateurs	❖ **1.0 : Non homogénéité** Norme incomplète => Librairies supplémentaires non-standard => Multiplication des versions ❖ **Piratage** ❖ **2.0 : Marché + restreint** ❖ **Petit marché en Europe**
❖ **Homogénéité** Norme pragmatique et complète Contrôle sévère des implémentations par DoCoMo ❖ **Pas de piratage** ❖ **En avance de phase**	❖ **Spécifique/Fermé DoCoMo/Alliance imode** NEC, Mitsubishi, Panasonic, SAGEM, Alcatel, Samsung Bouygues/Telefonica/Wind/Base/KPN/Cosmote/O2
❖ **Machine Virtuelle plus puissante** float, double ❖ **APIs plus nombreuses**	❖ **Sur PDAs et téléphones haut de gamme, voitures** PalmOS, P800/P900, PocketPC

1. Le jeu sur mobile (2)

- Environnements de développement

➢ **C++ / SymbianOS**

➢ **Java (J2ME)**

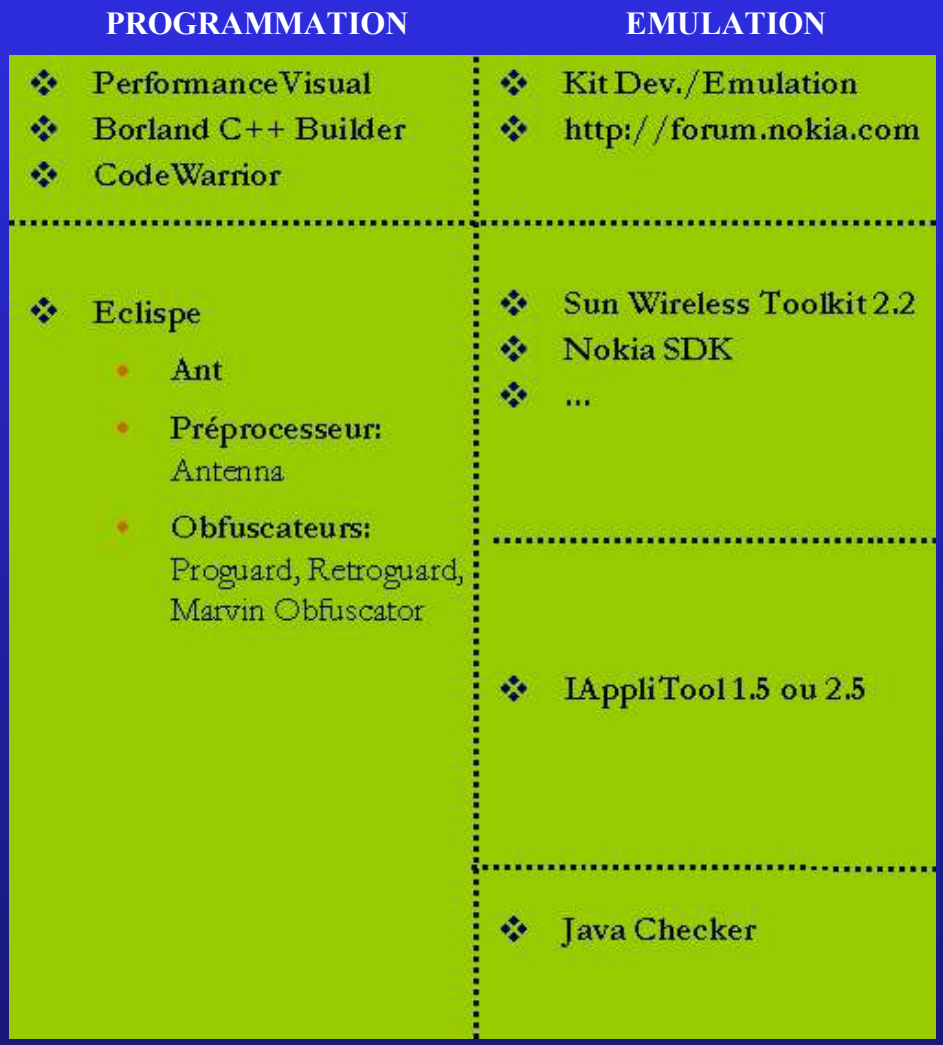
▪ **CLDC**

❖ **MIDP 1.0 et 2.0**

❖ **DoJa 1.5 et 2.5**

▪ **CDC**

❖ **Personal Java**



1. Le jeu sur mobile (3)

- Java 2 Micro Edition (J2ME)

MIDP / DoJa || Personal Java

CLDC || CDC

KVM || JVM

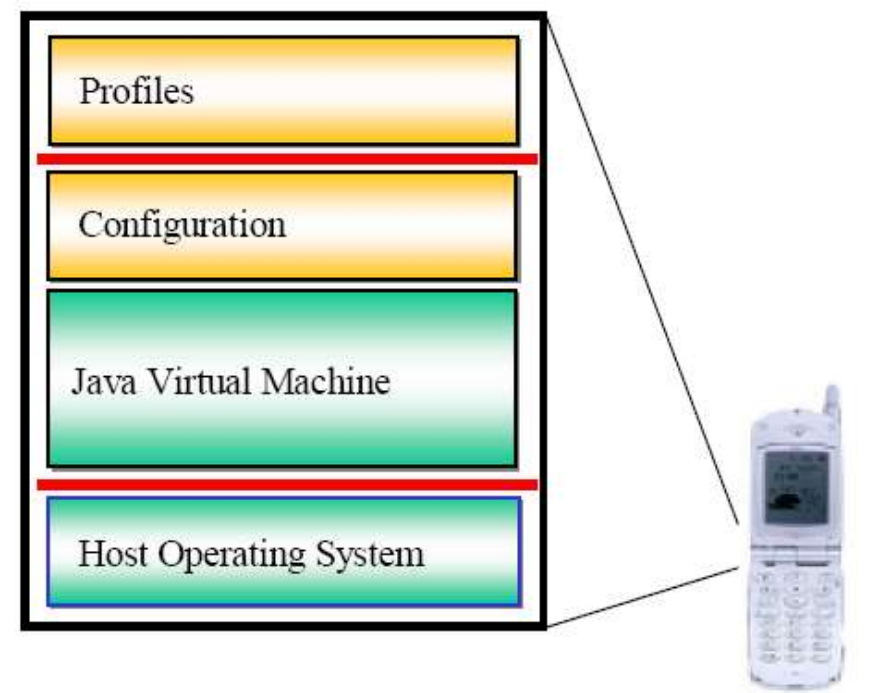


Figure 1-3 J2ME software layer stack

1. Le jeu sur mobile (4)

- Contraintes de développement en MIDP et DoJa
  - Machines virtuelles allégées (pas de FPU: ni float, ni double)
  - Capacités d'exécution réduites
  - Librairies Java réduites ou expurgées (ex: pas de sérialisation d'objets)
  - Limitation en termes de taille de code (.jar)
    - MIDP : 100-200 Ko
    - DoJa: 30 Ko

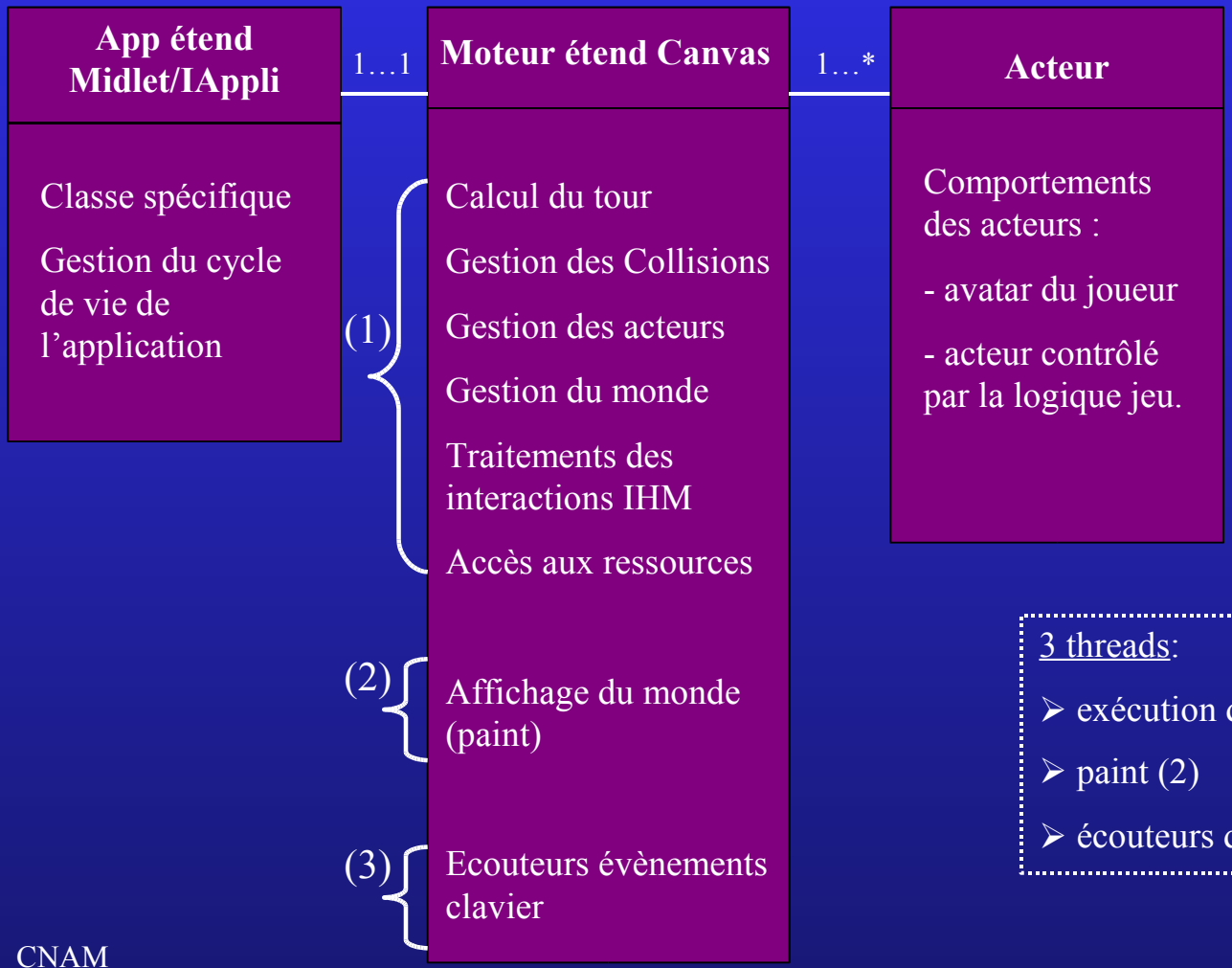
    ⇒ impossibilité d'embarquer des APIs propriétaires lourdes.
  - Limitation en termes de taille de ressources (graphismes, sons, maps,…)
    - MIDP : 4Mo en 1.0 , toute la mémoire disponible du téléphone en 2.0
    - DoJa: 100 Ko en 1.5, 200Ko en 2.5

1. Le jeu sur mobile (5)

- Guidelines pour le développement sur mobile
  - Réduction de la taille du code
    - Limiter le nombre de classes dès la conception (gourmand en code)
    - Obfusquer son code (ex: utiliser ProGuard)
      - réécriture des identificateurs au plus court (variables, fonctions)
      - in-lining des constantes
      - suppression des codes morts …
    - Eviter l'utilisation des packages
    - Sortir les constantes du code => utilisation de fichiers binaires
  - Amélioration des performances
    - Réduire au strict minimum l'instanciation d'objets (new), *malloc* et *garbage collector* déficients
    - Limiter le nombre de threads, implémentations variables en fonction des KVM

1. Le jeu sur mobile (6)

- Modélisation typique d'un jeu sur mobile J2ME



3 threads:

- exécution de la logique de jeu (1)
- paint (2)
- écouteurs d'évènements clavier (3)

2. Le jeu multijoueur sur mobile

- Contraintes des réseaux de téléphonie 2/2.5/3G
  - Latence élevée (dépendante des conditions du réseau)
    - GPRS/EDGE: > 800 ms
    - UMTS: > 350 ms
  - Faible bande passante (dépendante des conditions du réseau)
    - GPRS: 20 (down) / 20 (up) Kbps
    - EDGE: 59 (down) / 59 (up) Kbps
    - UMTS: 384 (down) / 384 (up) Kbps
  - Pas d'adressage IP => pas de connexion serveur vers client possible

2. Le jeu multijoueur sur mobile (2)

- Contraintes protocolaires en MIDP et DoJa
  - Protocoles réseaux utilisables au niveau applicatif:
    - HTTP (MIDP 1.0/2.0 et DoJa 1.5/2.5)
    - HTTPS (MIDP 2.0 et DoJa 1.5/2.5)
    - Sockets (MIDP, utilisation controlée par l'opérateur)

    ⇒ le seul protocole « universel » sur téléphone java: HTTP

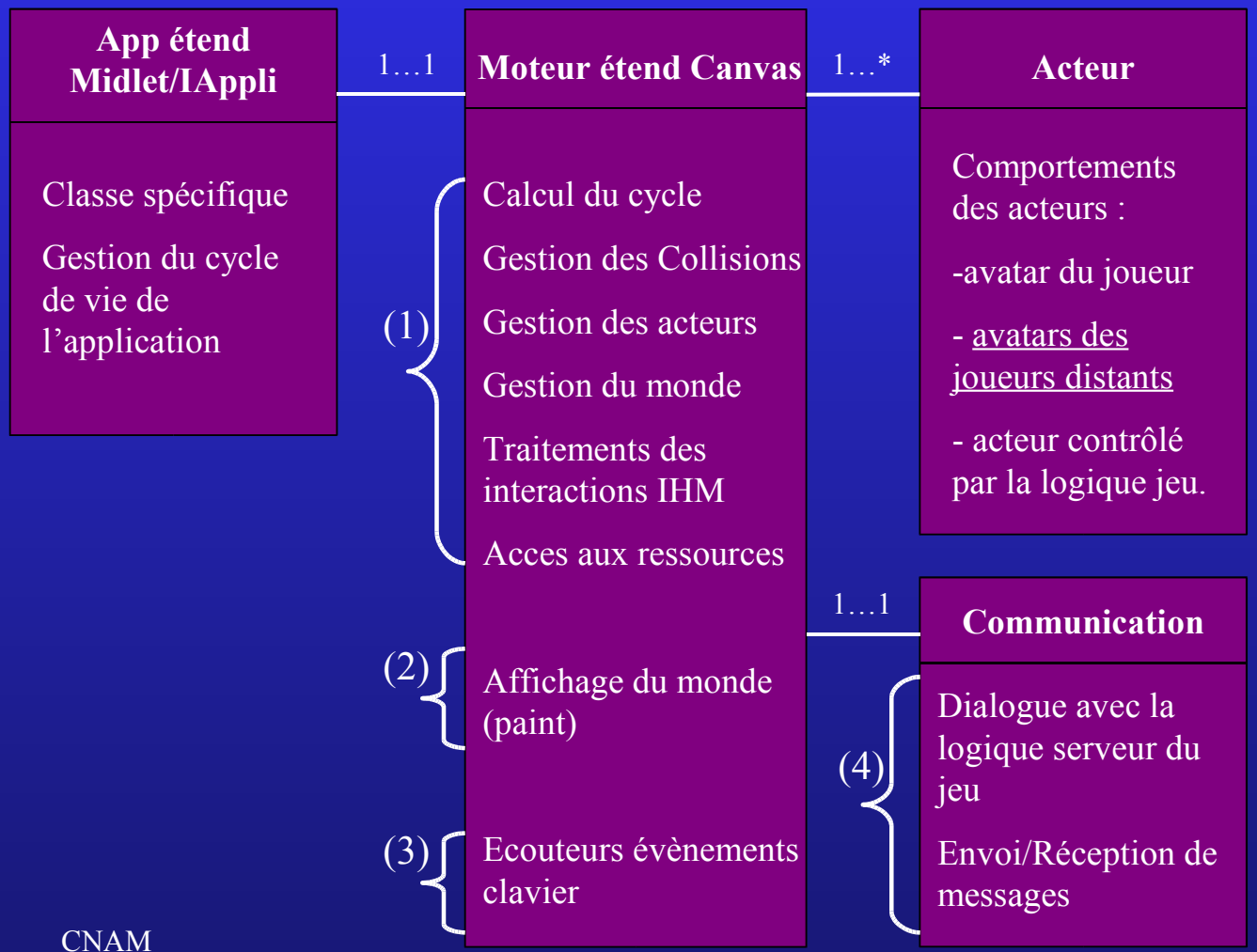
2. Le jeu multijoueur sur mobile (3)

- Impact sur le développement de jeux multijoueurs sur mobile
  - ➢ Pas d'adressage IP => modèle client-serveur
  - ➢Utilisation du protocole HTTP => requêtes périodiques du client vers le serveur pour être informer des évènements du jeu & stockage côté serveur des évènements.
  - ➢ Optimisation requise du protocole d'échange de messages: minimiser la consommation de bande passante et améliorer la vitesse de transmission tout en contrôlant les coûts (facturation au Ko).
  - ➢ Optimisation requise de la taille de code réservée à l'interface cliente de l'intergiciel.
  - ➢ Types de jeux multijoueurs envisageables sur ces réseaux:
    - Tour par tour
    - RTS «Real Time Strategy»

    => jeux d'action temps réel irréalisables (ex: FPS « First Person Shooter » ou simulation) , temps réel maximal acceptable ~ 200ms

2. Le jeu multijoueur sur mobile (4)

- Modélisation typique sur mobile J2ME



4 threads:

- exécution de la logique de jeu (1)
- paint (2)
- écouteurs d'évènements clavier (3)
- communications (4)

3. Fonctionnalités d'une plateforme de jeux

- Services support des jeux pour:
  - **Joueurs**: gestion du compte, récupération des sessions de jeux « lobby », gestion des amis « buddy », gestion des scores
  - **Editeurs**: publication des jeux (incluant les mises à jour), gestion des joueurs, gestion des compétitions
  - **Administrateurs systèmes**: supervision, logging, répartition de charge
- Services systèmes
  - Securité, optimisation des échanges de données, équilibrage de charge, tolérance aux fautes, persistance des données, monitoring, logging, data mining…

3. Etat de l'art des plateformes de jeu pour mobile

- Nombreuses plateformes propriétaires (Terraplay Move, InFusio EGE, mFormat…)
- Aucune plateforme open source

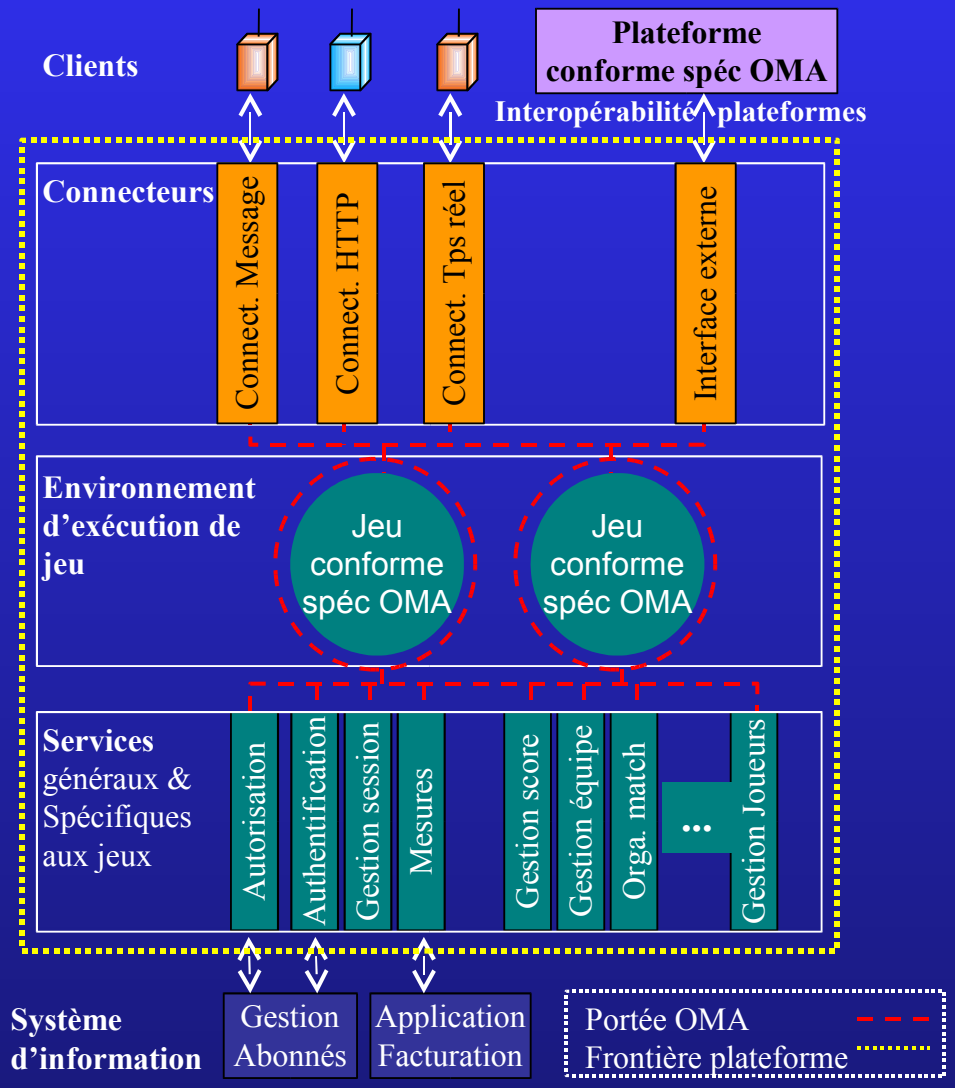
➢Problèmes liés aux spécificités des architectures propriétaires

- fortes contraintes jeu
- non intéropérabilité

⇒ normalisation du domaine

➢ Normalisation: **Open Mobile Alliance** (2002) , groupe de travail « **Games Services** » [OMA GS]

- architecture générale (ci-contre)
- spécifications fonctionnelles

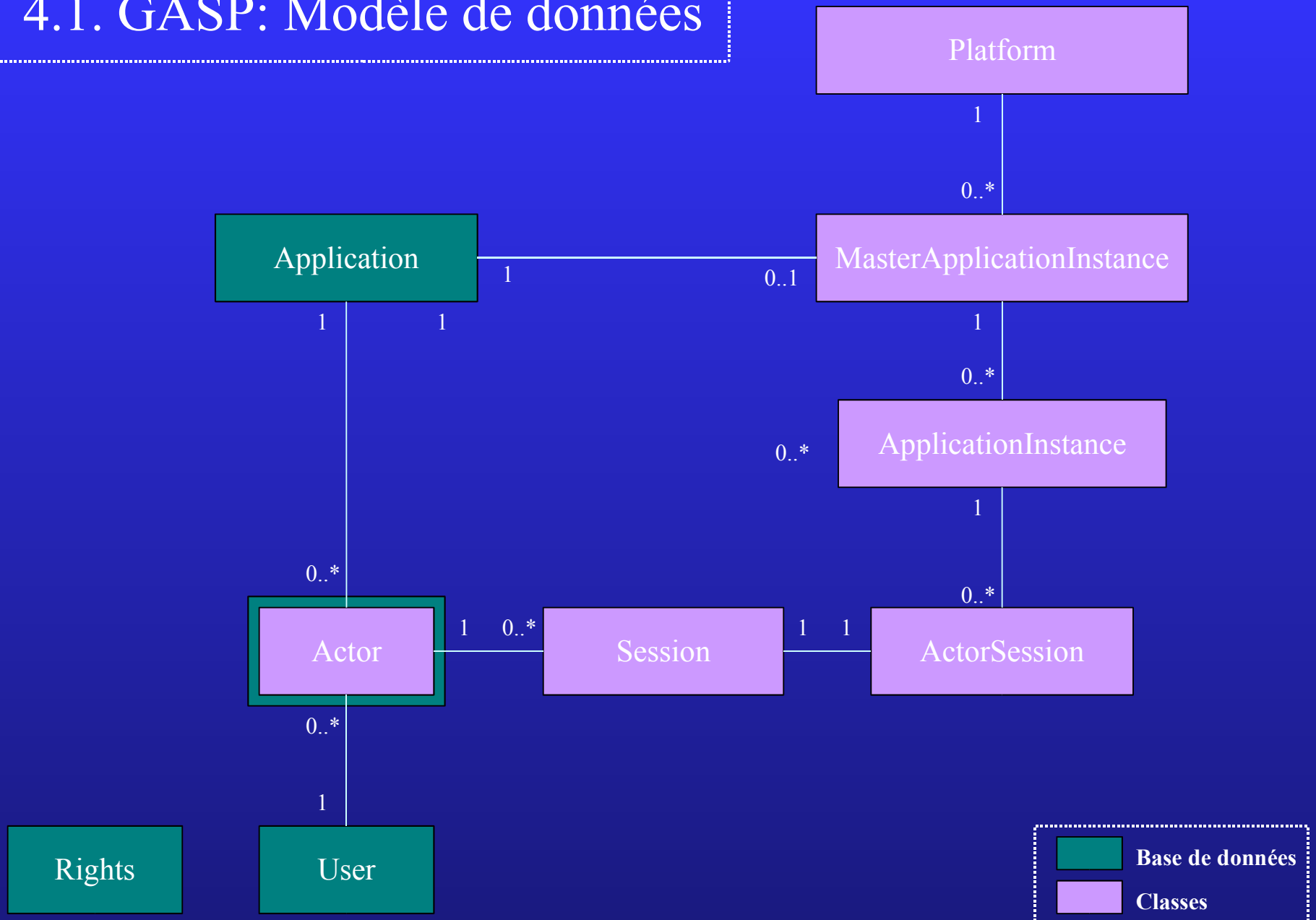


4. GASP « GAming Services Platform »

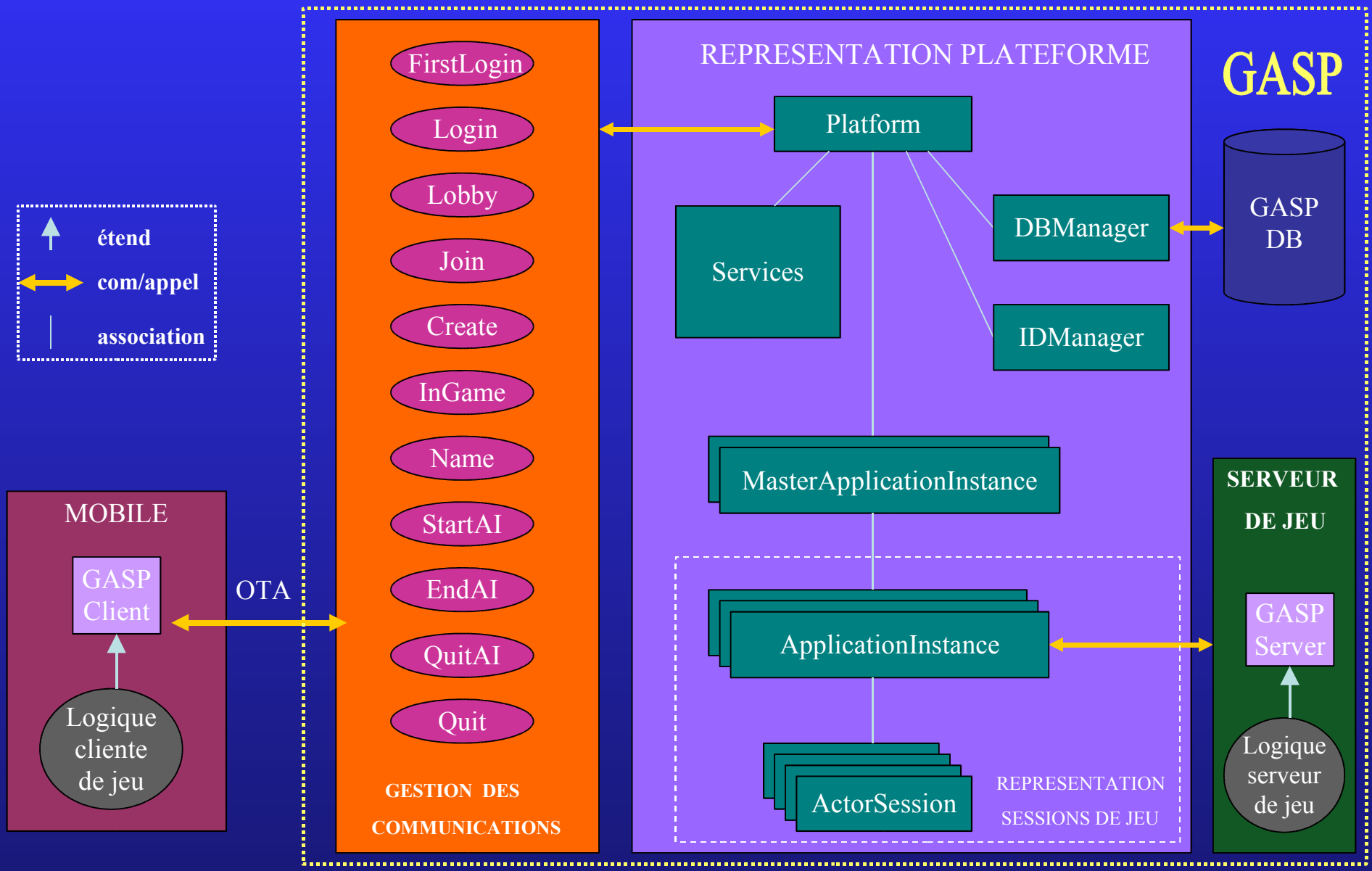
Objectifs

- Implanter les spécifications des services pour les jeux de l'OMA GS
- Open source (implémentation Java)
- Cibles
  - Mobiles J2ME, profils MIDP et Doja
  - Jeux « pseudo » temps réel (protocole HTTP sur 2/2.5/3G)
  - Petits éditeurs
- Prise en compte des contraintes de développement des jeux sur mobile (taille du code, taille des messages échangés)

4.1. GASP: Modèle de données



4.2. GASP: Architecture



4.3. GASP: Interfaces

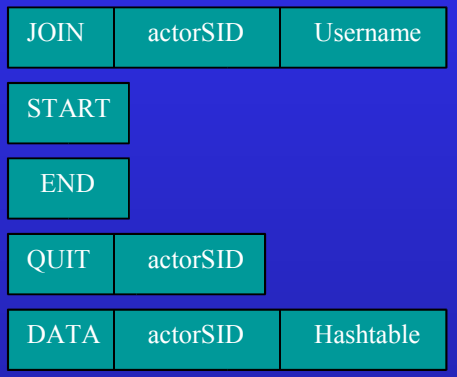
org.mega.gasp.client.GASPClient	Rôle
• int firstLogin(int userID, int username, String pwd)	Première connexion
• int login(int actorID, String username, String pwd)	Connexion à la plateforme
• Vector getApplicationInstances (int sessionID)	Récupération des sessions de jeu
• int joinAI (int sessionID, int appInstanceID)	Rejoindre une session
• int joinAIRnd (int sessionID)	Rejoindre aléatoirement une session
• int createAI (int sessionID, int min, int max)	Créer une session
• int createAIPriv (int sessionID, int min, int max, Vector actors)	Créer une session privée
• String name(int actorSessionID, String pseudo)	Changer de pseudonyme
• void getEvents(int actorSessionID)	Récupération des évènements
• void startAI(int actorSessionID)	Démarrer la partie
• void sendData(int actorSessionID, Hashtable data)	Envoyer des données
• void endAI(int actorSessionID)	Arrêter la partie
• void quitAI(int actorSessionID)	Quitter la partie
• void quit (int sessionID)	Déconnexion de la plateforme

4.3. GASP: Interfaces (2)

org.mega.gasp.server.GASPServer	Rôle
• void onJoinEvent(JoinEvent je)	JoinEvent listenner
• void onQuitEvent(QuitEvent qe)	QuitEvent listenner
• void onStartEvent(StartEvent se)	StartEvent listenner
• void onEndEvent(EndEvent ee)	EndEvent listenner
• void sendDataTo (int actorSessionID, DataEvent de)	Envoyer des données à un joueur
• void onDataEvent(DataEvent de)	DataEvent listenner

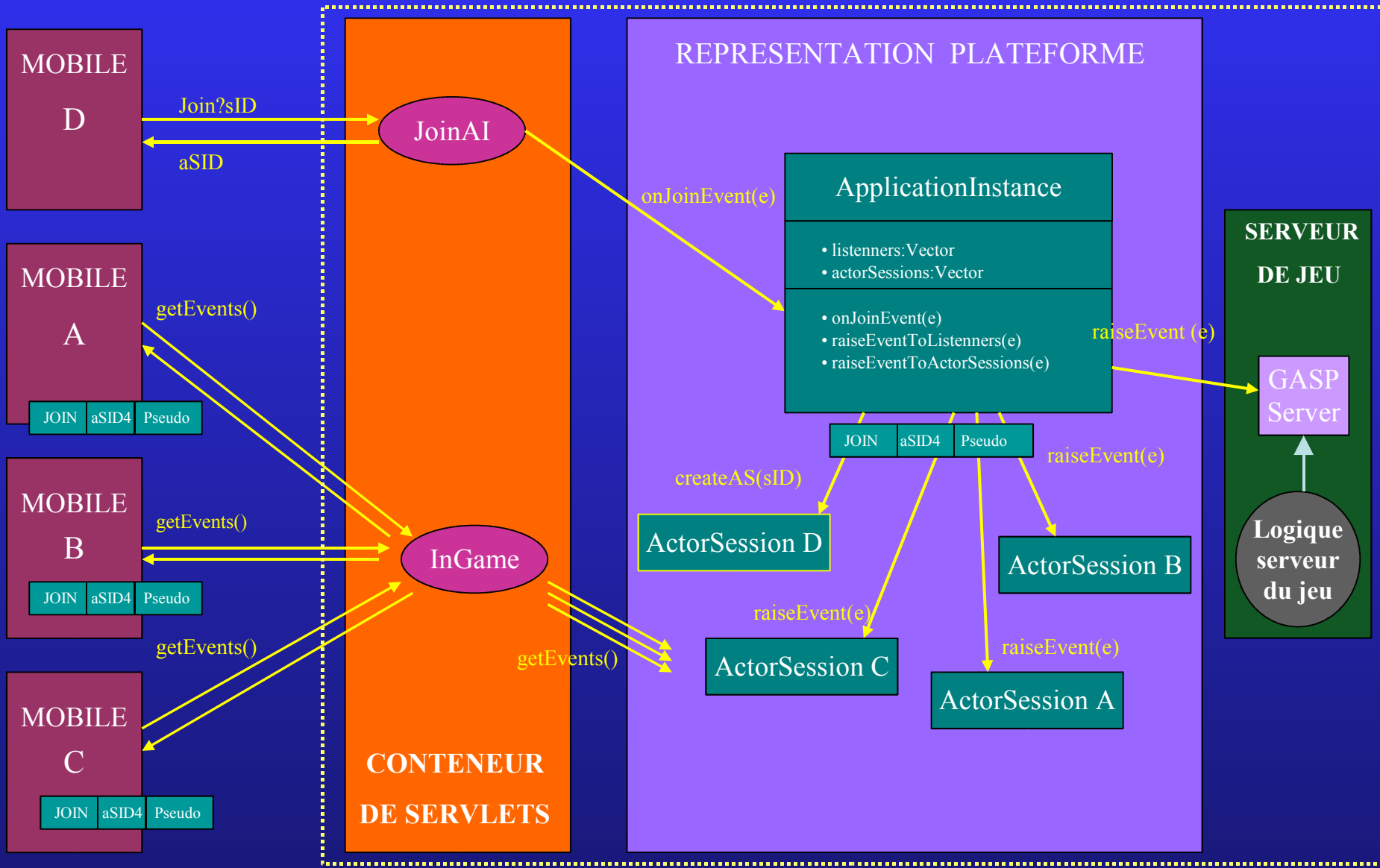
4.4 GASP: Modèle Evénementiel

- 5 types d'événements:
  - JoinEvent => un nouveau joueur rejoint la partie
  - StartEvent => début de la partie
  - EndEvent => fin de la partie
  - QuitEvent => un joueur quitte la partie
  - DataEvent => données envoyées en cours de jeu par un joueur ou par la logique serveur du jeu



- 3 types d'écouteurs d'événements:
  - ApplicationInstance (session de jeu)
  - ActorSessions (les joueurs)
  - GASPServer (l'interface étendue par la logique serveur du jeu)

4.4. GASP: Modèle Evènementiel (2)



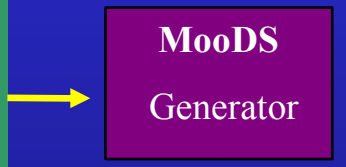
4.5. GASP : Protocole MooDS

- **M**obile **O**ptimized **O**bjects **D**escription & **S**erialization (MooDS)
  - But: améliorer la vitesse de communication en réduisant la taille des messages de données transférées en cours de jeu
  - Algorithme: transférer les objets messages du jeu par valeur (types primitifs)

```
<types>
  <type name='Update'>
    <element name='aSID' type='short'/>
    <element name='x ' type='int'/>
    <element name='y' type='int'/>
    <table name='t' type='double'>
      <row/> …
    </table>
  </type>
  …
</type>
```

types.xml

Fichier de description de types d'objets transférés en cours de jeu



gamePackage.CustomTypes

void encodeUpdate(DataOutputStream dos)

Update decodeUpdate(DataInputStream dis)

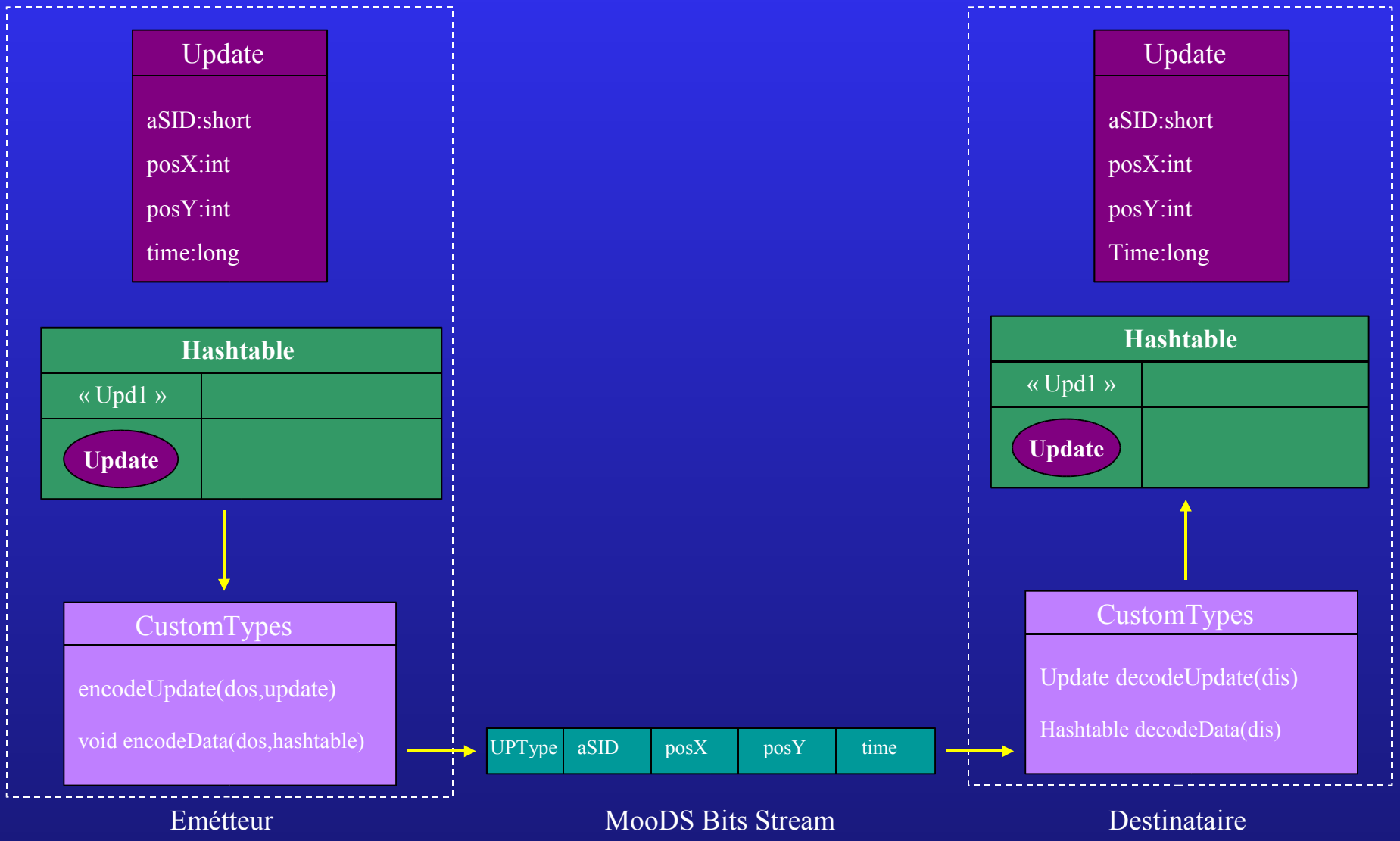
void encodeData(Hashtable h, DataOutputStream dos)

Hashtable decodeData(DataInputStream dis)

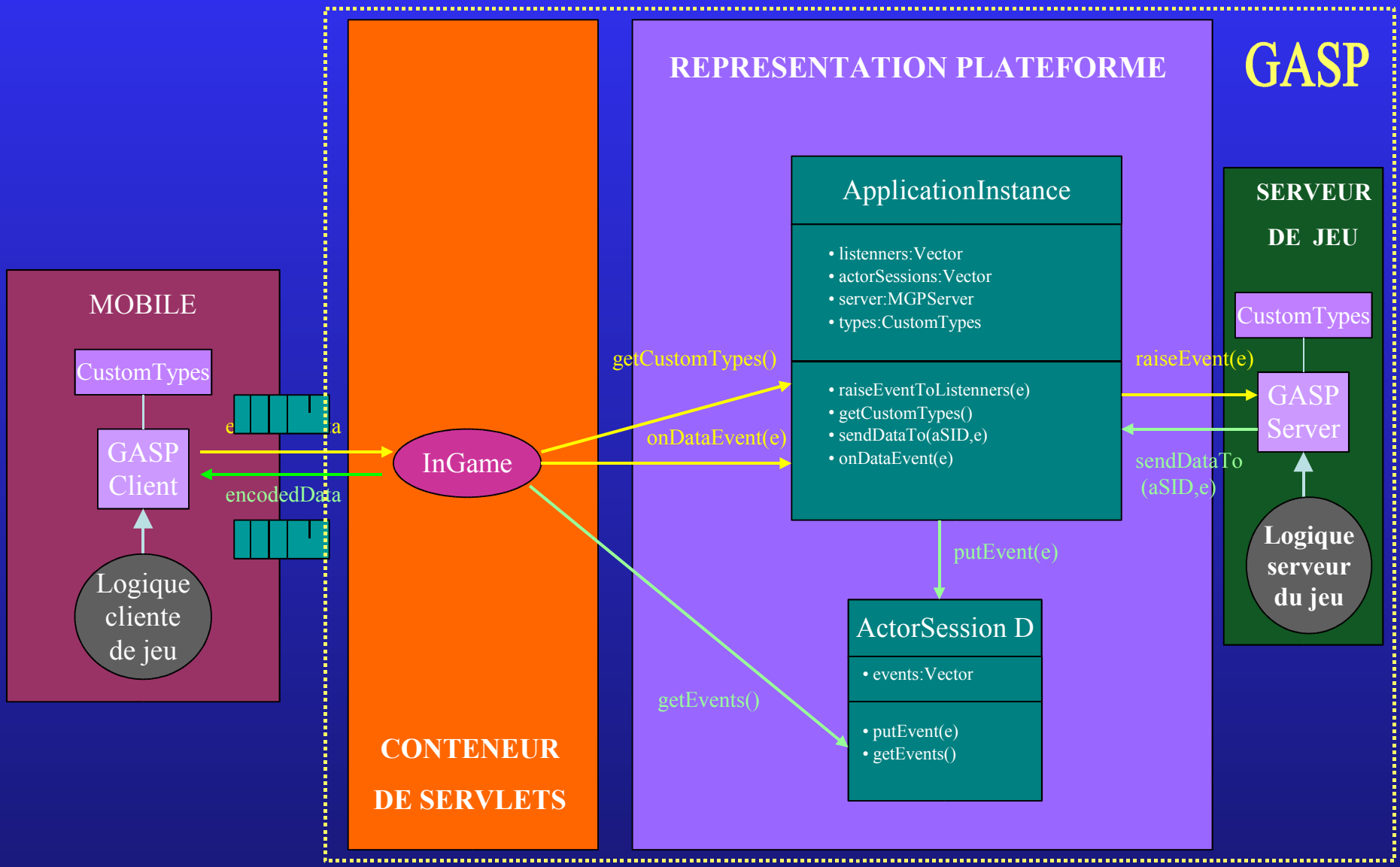
CustomTypes.java

Classe incluant les méthodes d'encodage/décodage des objets décris

4.5. GASP : Protocole MooDS (2)



4.5. GASP : Protocole MooDS (3)



5. Conclusion & Perspectives

- GASP est une plateforme Java open source dédiée aux jeux multijoueurs en ligne sur mobile

- Statut
  - Développement selon le modèle en spirale
  - Première version = Prototype
  - Prototype d'un jeu Doja reposant sur GASP dans le cadre du projet JIMM
  - Publication open source via le consortium ObjectWeb depuis Juin 2005

- Perspectives
  - Nouvelles fonctionnalités:
    - ✓ Gestion accrue des communautés de joueurs
    - ✓ Gestion des compétitions, réflexion sur un canevas générique de création de compétition
    - ✓ Réflexion sur les modèles d'exécution des jeux (automatique, propriétaire,…), canevas générique de modèles d'exécution
    - ✓ Persistance de données (pour le moment déléguée à la logique serveur)
    - ✓ Interfaçage avec les services du fournisseur de contenu (compte utilisateur, …)

Bibliographie

[1] http://www.openmobilealliance.org/tech/wg_committees/gs.html

[2] R. Pellerin. *GAming Services Platform, Plateforme pour les jeux multijoueurs sur mobiles*. Rapport Master Recherche (DEA). CNAM-Cedric/INT/Paris 6. Septembre 2004 (http://bscw.enst-bretagne.fr/pub/bscw.cgi/0/2731016)

[3] R. Pellerin. *Mobile Gaming Services, Services pour les jeux multijoueurs sur mobiles*. Rapport Bibliograhique. CNAM-Cedric/INT/Paris 6. Juin 2004 (http://bscw.enst-bretagne.fr/pub/bscw.cgi/0/2731016)