GASP: an Open Source gaming service middleware dedicated to multiplayer games for J2ME based mobile phones

PELLERIN Romain

RIAM Project (1 year): JIMM (Bouygues Telecom, CNAM-Cedric, Filao/Infraworlds)

Research project: CNAM-Cedric, INT, Filao/Infraworlds

Contents

1. Mobile phones multiplayer games constraints
2. Game middleware functionalities
3. Gaming Services Platform (GASP)
4. KouizMarket: A game using GASP
5. Conclusion and Future works
6. Kouizmarket demonstration

1. Mobile phones multiplayer games constraints

- Mobile phone networks constraints 2/2.5/3G

	GPRS	EDGE	UMTS
Latency (ms)	> 800	> 800	> 350
Bandwidth (Kbps)	20	59	384

  - high latency => round-robin, RTS but no real time games as First Person Shooter!
  - low bandwith => control the volume of data transmited to the network!
  - no IP address => server to client direct communication impossible!

- J2ME Mobile phone specificities
  - Network protocols available
    - HTTP (MIDP 1.0/2.0 and DoJa 1.5/2.5)
    - HTTPS (MIDP 2.0 and DoJa 1.5/2.5)
    - Sockets (MIDP but restricted by operator)
    - => only one « universal » protocol: HTTP!
  - Application level constraints
    - Size code : MIDP 100-200Ko / DoJa 30Ko => need light middleware client interface!
    - Java APIs reduced, for example: No object serialization!

2. Game middleware functionalities

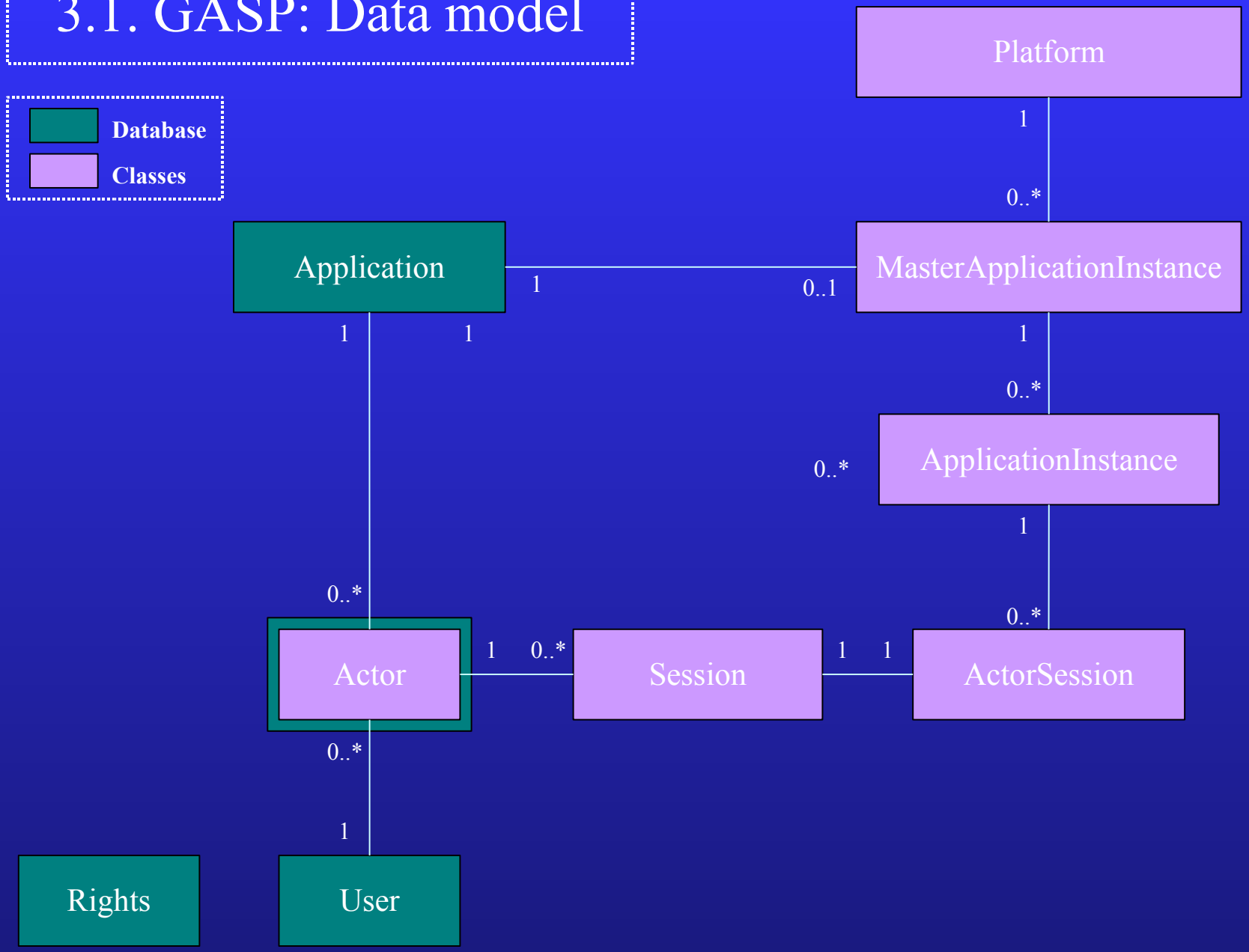
- Game services
  - **Players**: Account management, lobby room handling, buddy management, high scores handling...
  - **Editors**: Game publishing (includes updates), players management, competition management...
  - **System administrators**: Monitoring, logging, load sharing...
- System services
  - Security, data exchange optimisation, load balancing, fault tolerance, persistency, monitoring, logging, data mining…

3. GASP « GAming Services Platform »

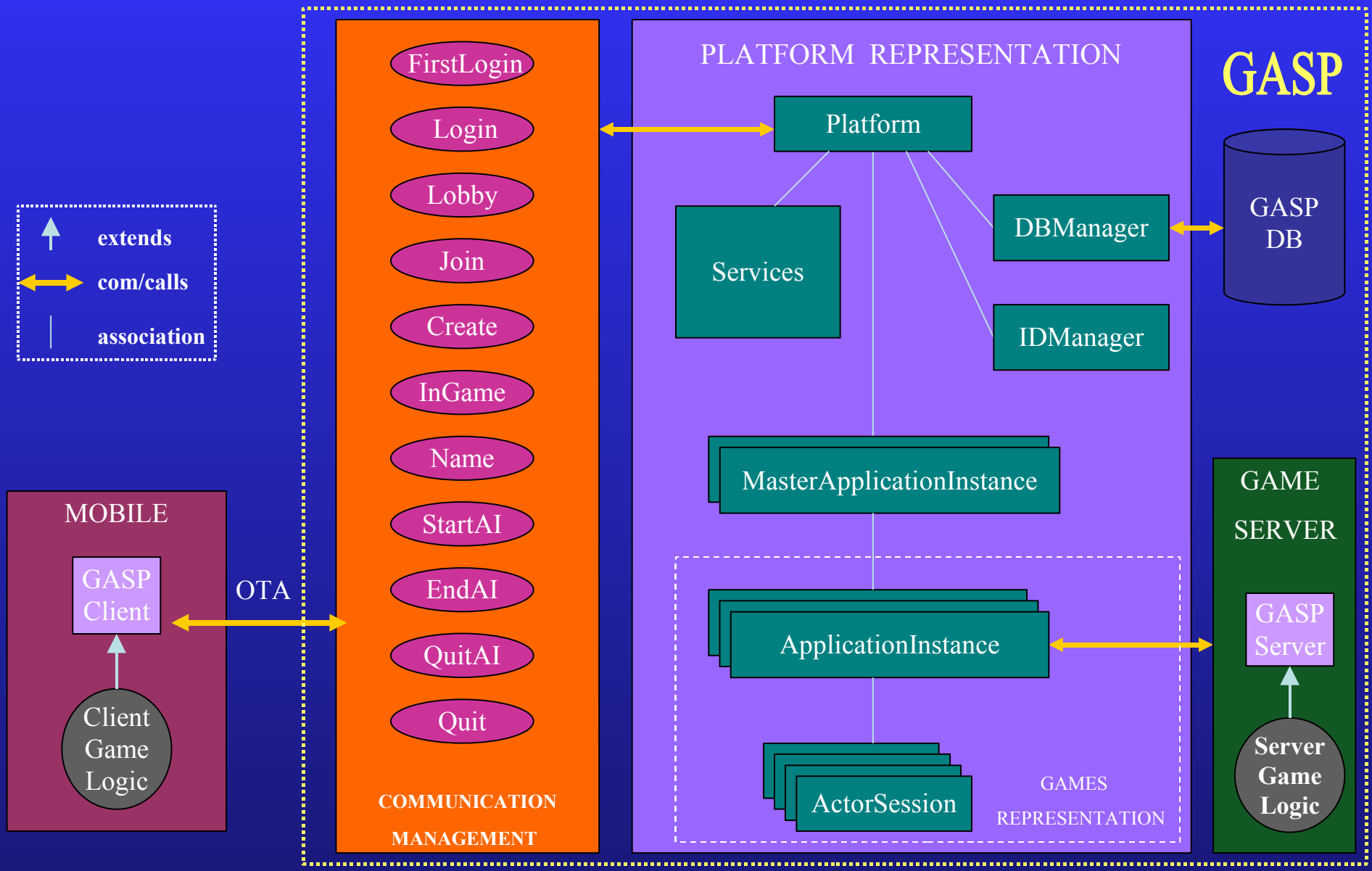
Objectives

- Implement OMA Gaming Services Specifications
- Open source (Java)
- Target
  - J2ME based mobile phones, MIDP and Doja profiles
  - *Near real-time* games (HTTP over 2/2.5/3G)
  - Little editors
- Consider mobile phone games development constraints (code size, communication data volume)

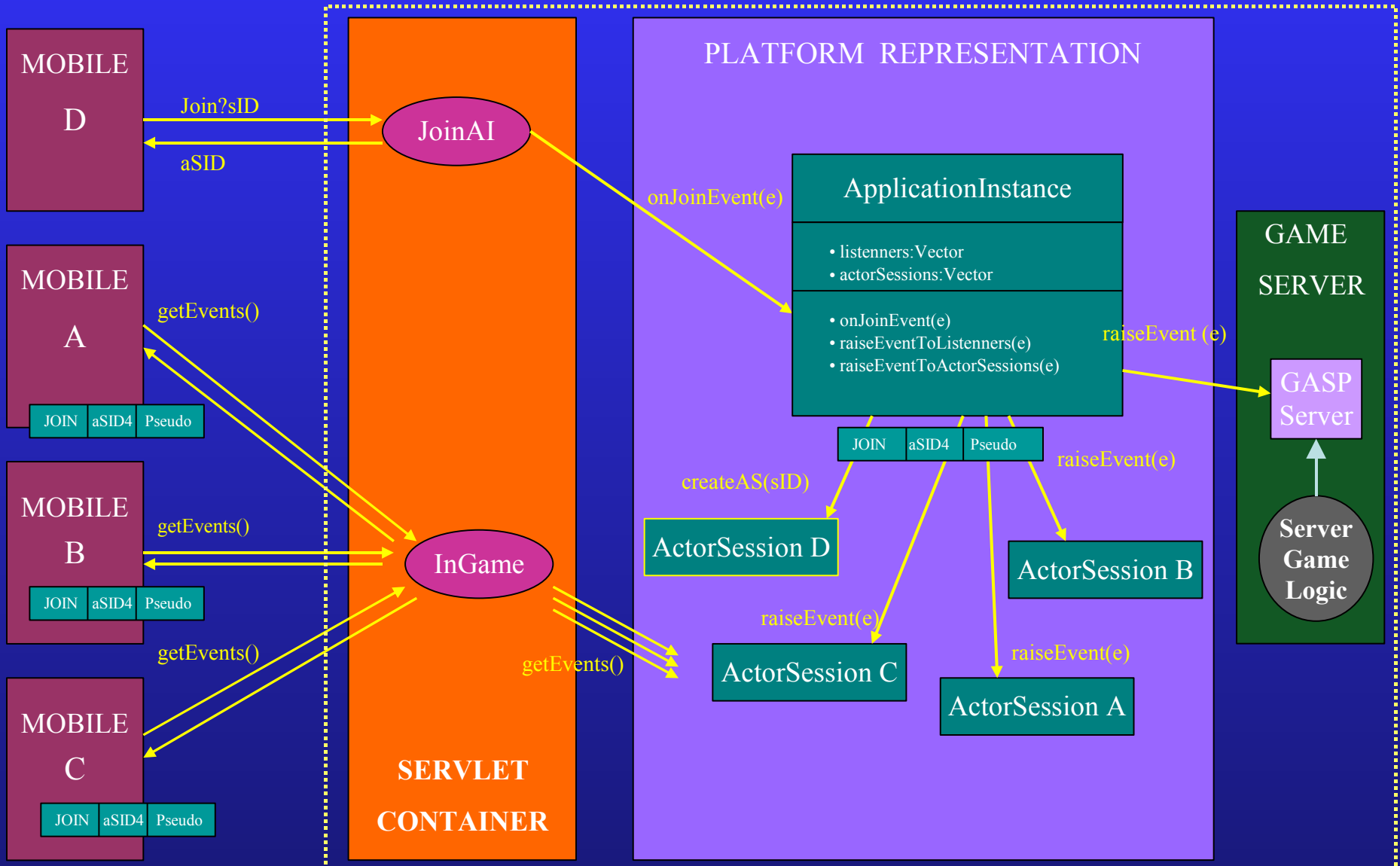
3.1. GASP: Data model



3.2. GASP: Overview



3.3. GASP: Events model



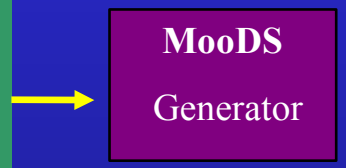
3.4. GASP : MooDS protocol

- **M**obile **O**ptimized **O**bjects **D**escription & **S**erialization (MooDS)
  - Goal: Increase communication speed by reducing length of in-game data messages
  - Algorithm: Send the game objects by values (primitive types)

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

types.xml

Objects description file



gamePackage.CustomTypes

void encodeUpdate(DataOutputStream dos)

Update decodeUpdate(DataInputStream dis)

void encodeData(Hashtable h, DataOutputStream dos)

Hashtable decodeData(DataInputStream dis)

CustomTypes.java

Contains Encoding/Decoding methods included in game client & server packages

4. KouizMarket: A game using GASP

- Game design (multiplayer)
  - *Near real-time* multiplayer game
  - 4 farmers and a maximum of 4 creatures, the « Kouiz »
  - Market interaction: one to one negociation
  - Fight interaction: 3 rounds, 4 actions (attack, jump, slide, catch)

Fight interaction

- Kouizmarket's use of GASP
  - Development simplified thanks to:
    - Doja GASP client interface for i-Mode™
    - Lobby functionalities
    - Object oriented message communication
    - Automatic game session management
  - Persistency delegated to server logic (specific DB access)
  - 3 seconds periodic requests
  - Average data volume: 219 bytes per second

Market interaction

5. Conclusion & Perspectives

- GASP is a Java Open Source middleware dedicated to multiplayer games for J2ME based mobile phones

- Status
  - Open Source under GNU L-GPL throught ObjectWeb consortium
  - Development according to a spiral model
  - First release = prototype
  - Doja multiplayer game prototype using GASP: « KouizMarket »

- Perspectives
  - Development of an Open Source GASP demonstration game
  - Network load and machine load tests
  - Study consistency for multiplayer games dealing with high latency (interpolation & extrapolation, i.e dead-reckoning)
  - New functionalities:
    - ✔ Improved players communities management
    - ✔ Competition management, reflection on a generic competition framework
    - ✔ Multiple game execution model (automatic, proprietary…), reflection on a generic game execution framework

6. KouizMarket demonstration

Demonstration

…

Bibliography

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

[2] GASP ObjectWeb website, http://gasp.objectweb.org

[3] Pellerin, Delpiano, Gressier-Soudan, Simatic. *GASP: un intergiciel pour les jeux en réseaux multijoueurs sur téléphones mobiles,* UbiMob'05, Grenoble, France. June 2005

[4] Pellerin. *GAming Services Platform, Plateforme pour les jeux multijoueurs sur mobiles*. Research Master report (DEA). CNAM-Cedric/INT/Paris 6. September 2004

[5] Pellerin. *Mobile Gaming Services, Services pour les jeux multijoueurs sur mobiles*.Bibliography report. CNAM-Cedric/INT/Paris 6. June 2004

3. Game middleware for mobile market overview

• Lot of commercial platforms (Nokia SNAP, Terraplay Move, mFormat…)

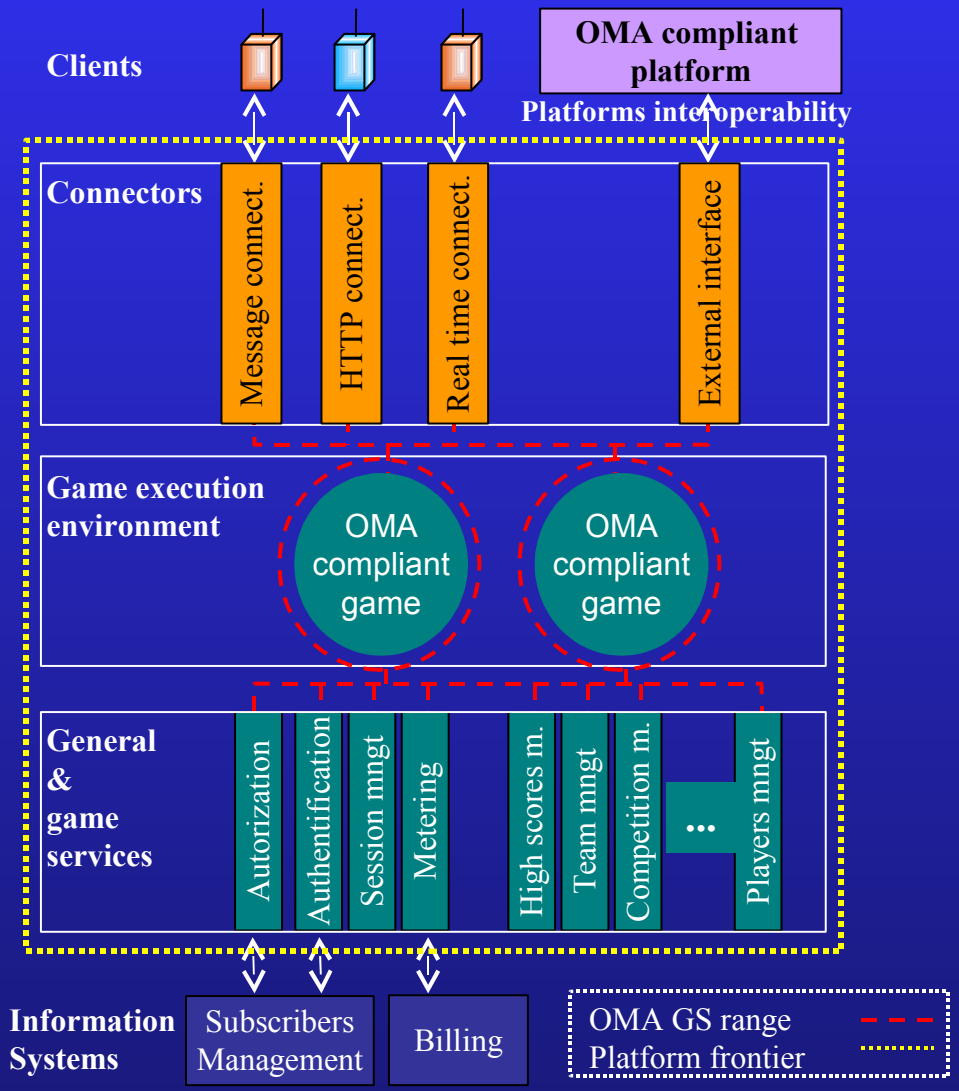
• No open source platform

➢ Problems dealing with proprietary architecture specifities

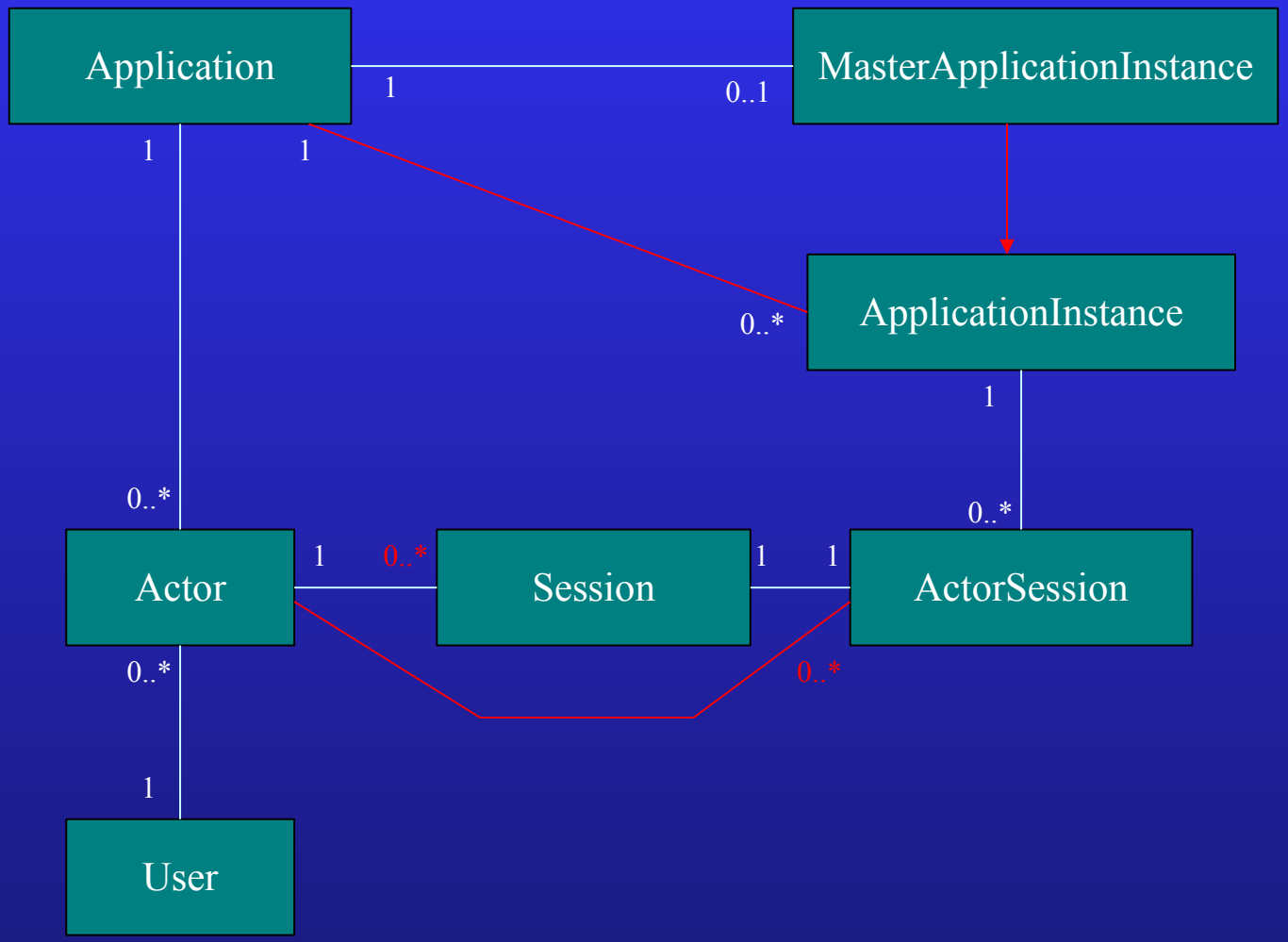
- heavy game contraints
- no interoperability
- domain normalization

➢ Normalization: **Open Mobile Alliance** (2002) , « **Games Services** » working group [OMA GS]

- architecture (picture)
- functional specifications



OMA
Gaming Services Specifications



GASP Client API

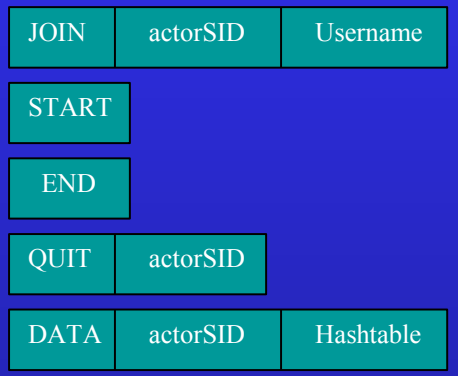
org.mega.gasp.client.GASPClient	Role
• int firstLogin(int userID, int username, String pwd)	First connection
• int login(int actorID, String username, String pwd)	Connect to the platform
• Vector getApplicationInstances (int sessionID)	Get the game sessions list
• int joinAI (int sessionID, int appInstanceID)	Join a game session
• int joinAIRnd (int sessionID)	Join a random game session
• int createAI (int sessionID, int min, int max)	Create a game session
• int createAIPriv (int sessionID, int min, int max, Vector actors)	Create a private game session
• String name(int actorSessionID, String pseudo)	Change pseudoname
• void getEvents(int actorSessionID)	Get the platform events
• void startAI(int actorSessionID)	Start the game
• void sendData(int actorSessionID, Hashtable data)	Send data
• void endAI(int actorSessionID)	Stop the game
• void quitAI(int actorSessionID)	Quit the game session
• void quit (int sessionID)	Logout from platform

GASP Server API

org.mega.gasp.server.GASPServer	Role
• 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)	Send data to a specific player
• void onDataEvent(DataEvent de)	DataEvent listenner

3.2 GASP: Events model

- 5 event types
  - JoinEvent => A new player joins
  - StartEvent => Start of the game
  - EndEvent => End of the game
  - QuitEvent => A player quits
  - DataEvent => In-game data received from a player or the server game logic



- 3 types of event listeners
  - ActorSessions (the players)
  - ApplicationInstance
  - GASPServer (the server game logic)

GASP
Events Model: MooDS protocol – Example