An External Short Message Entity for Gambling Services

Enrique Marcote
Departamento de Electrónica y Sistemas
Universidad de A Coruña
Spain
mpquique@udc.es

Daniel I. Iglesia
Departamento de Electrónica y Sistemas
Universidad de A Coruña
Spain
dani@udc.es

Carlos J. Escudero
Departamento de Electrónica y Sistemas
Universidad de A Coruña
Spain
escudero@udc.es

Abstract

This paper introduces a new platform designed for mobile gambling services. The special characteristics of these services lead us to developed a multiservice platform that easily brought mobility to applications and services not designed for that.

This system was designed to work over SMS, allowing automation of the tedious work of programming reliable SMS-based interfaces. The new system automatically generates user front-ends for a wide variety of services based on forms. As we will see in the paper, the interfaces are defined by means of the novel W3C XForms standard.

The resulting platform was designed to be highly efficient, reliable and fault tolerant. Choosing Erlang/OTP as the development environment was a key factor to acquire these goals.

As part of the project an open source Erlang SMPP implementation was developed, key features of this library, named OSERL, are also described in this paper.

Categories and Subject Descriptors

D.m [**Software**]: Miscellaneous

General Terms

Design

Keywords

Mobile gambling, SMS, SMPP, XForms

1 Introduction

The tremendous success of cellular systems shows the demand for and acceptance of mobile communication. While today's systems mainly cover voice communication, market studies demonstrate an increasing demand for mobile data services.

The developments for covering these demands are accompanied by advances in terminals that bring graphical displays and allow downloads of new applications. In the next years, mobile services will represent the 30% of the telecommunication companies total earnings.

Recent European research studies [1, 2] reveal that mobile gambling services market, as a whole, is likely to be worth in excess of $16billion by 2008. Online gambling in the fixed Internet environment has already experienced a huge level of growth. What's more, in many countries, as Spain, lottery is a very popular game[1] with more than 1.000 million[2] tickets sold per year [3].

More than 92 per cent of the population owns a mobile phone in Spain, surprisingly high if we compare it to the percentage of people with Internet access, barely reaching a 25 per cent. Coupling the enormous market of mobile phones with the most popular lottery games seems to be a good deal.

Mobile lotteries have the advantage that they do not require complex graphic solutions or polyphonic sounds to be developed. Simple SMS (Short Message Service) capable phones are perfectly adequate. This compatibility opens up mobile lotteries to a vast user population across all geographic regions. Furthermore, SMS is still the most popular service, used by a 95 per cent of the subscribers [4].

Due to the great amount of actual and potential players, the new European electronic lottery seems to be especially suitable to go on mobile. 11.21 per cent of those polled in our market study [5] considered *very useful* a SMS service for playing the lottery, 42.46 per cent found it *useful*, and almost 20 per cent of the total would use this service *daily* or *weekly*.

This paper introduces AMUSE, a platform designed to support mobile gambling services (lotteries). Since lottery tickets may be seen as a input forms, AMUSE was designed support any kind of service using forms to interact with users. The platform needed to be reliable, robust, scalable and highly concurrent.

The paper is organized as follows: section 2 describes the system requirements for gambling services. Section 3 shows the overall structure of the platform. Sections 4 and 5 describe some addi-

[1] In its traditional paper form.
[2] American billion

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Erlang'04, September 22, 2004, Snowbird, Utah, USA.
Copyright 2004 ACM 1-58113-918-7/04/0009 ...$5.00

tional features of the system. Section 6 introduces OSERL, an open source Erlang SMPP implementation, developed for this project. Finally section 7 is devoted to conclusions.

2 System Requirements

The European lottery is a distribution of prizes by chance. A player makes a payment in return for the chance of winning a prize. A player chooses his or her own numbers from a selected range: five numbers out of a possible fifty, plus two stars (two additional numbers out of nine). The lottery operator then draws the numbers and stars at a selected date and time (every Friday at 10 PM) to determine the winners.

The key requirements a mobile gambling platform must satisfy are:

- Support for a great amount of concurrent users. Even players can buy[3] their lottery tickets from Monday to Friday, most of the users concentrate on Friday, next to the draw, and on Monday, they buy new tickets after checking their prizes on the previous draw.
- The service must be extremely reliable. Since lottery plays with money, users are very sensitive to the reliability of the system. Note that prizes reach more than 30.000.000 euros every week. System failures could be dramatically determinant in the success of the service, preventing users to trust the system and therefore using it. Not to mention the liability cost due to service malfunction or failures.
- 24x7 service availability. Potential users of this service greatly value the advantage of sending their bets[4] any day at any time, in contrast to traditional mechanisms, subject to restricted timetables.
- A quick response must be send back to the user. This circumstance gets of an special importance when time to draw is close, which in turn is expected to be the moment with higher loads on the system.
- Scalable. The growth on the number of users is expected to be exponential. The underlying system must be designed with this principle in mind.
- Payment. Since the number of chances can be variable in a lottery slip, the amount to pay cannot be fixed, therefore, a mechanism for mobile payment must be enabled.

Although our key target service was the European electronic lottery, there exist other electronic lotteries and sports pools in Spain that we didn't want to neglect. From the early stages we aimed to design a system ready to add new services whenever desired, with a minimum development cost.

3 AMUSE: A Multi-Service ESME

Our proposed solution copes with all the requirements expounded on Section 2. The resulting system is a multi-service platform conceived to run on a distributed environment. Multiple services may run within a single application, simplifying their implementation and maintenance. SMS interfaces for new services are declared in a simple XML [6] document, according to the XForms [7] standard. In addition, YAWS was integrated into AMUSE to provide a web interface to services backend.

[3] Validate and pay.

[4] With the selected numbers.

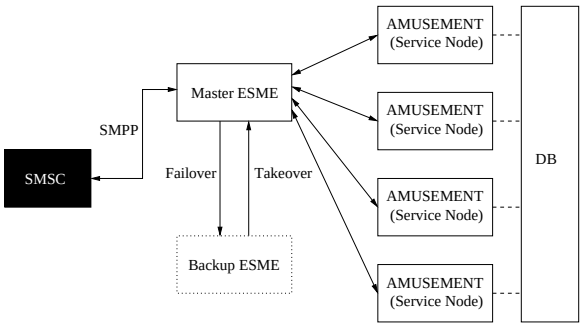

Figure 1. Multi Service ESME.

The entire development has been made in Erlang/OTP [8], ideal for the implementation of fault tolerant, concurrent applications. Having Erlang/OTP as the development environment permitted to easily integrate the different components of the system: SMSC, web server, DBMS and gambling applications. Providing common supervision mechanisms and the ability to run them as a whole.

3.1 System Structure

Figure 1 depicts the overall structure of AMUSE.

A master ESME (External Short Message Entity), bound as a transceiver to the SMSC (Short Message Service Center), conforming to SMPP (Short Message Peer to Peer) protocol [9], is responsible for receiving SMs (Short Messages) with the requests and sending response messages back to the SMSC. Requests are delivered by the ESME among available service nodes, called AMUSEMENTs, depending on their availability and particular load.

If ever the active ESME goes down, a backup ESME is started, getting the control, binding AMUSE back to the SMSC, and registering itself as the master ESME. The AMUSEMENTs won't be affected by this failure. This recovering strategy was implemented by simply exploiting Erlang's builtin failover and takeover mechanisms.

Service nodes store their data in a distributed database, implemented using Mnesia [10]. Every installed service runs distributively on each available AMUSEMENT.

3.2 AMUSEMENT: AMUSE Main Entity

Multiple services may run within a single AMUSEMENT (figure 2). All the requests are received by the service manager and forwarded to the destination service. Requests are identified by a keyword. Keywords are unique across all services. Will see how service requests are defined in greater detail later in this section.

YAWS [11] is started in every node to offer secure web access to service backends. Lottery services require human interaction for validating betting slips. A web based interface, using YAWS, has been developed for lottery dealers to do administrative tasks.

3.2.1 Service Implementation

As mentioned before, the ability to define new lottery games (services) was a desired feature in our system. To attain such goal, a `gen_service` behavior was defined to help implement new services for AMUSE. The callback module must take care of the logic

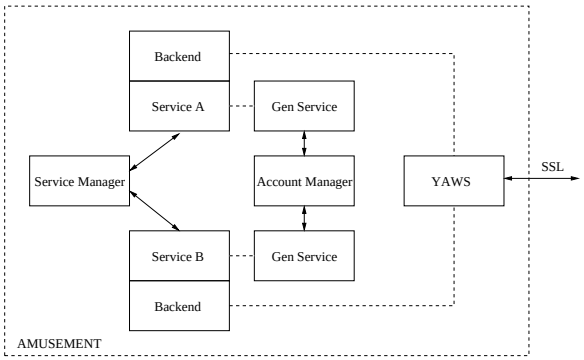

Figure 2. AMUSE main entity.

associated to the particular service, although interaction with the account manager and request validation are automatically handled by the behavior[5].

Service implementors may assume requests are properly validated, and in the exact same format they've specified in the XML service declaration.

In the XML document described next, a simplified declaration for the requests associated to the European lottery service is shown.

```
<?xml version="1.0"?>
<service>
  <model id="euro">
    <instance>
      <euro><count/><chance/></euro>
    </instance>

    <bind nodeset="count"
          type="xsd:integer"
          constraint=". &gt; 0 and . &lt; 7"/>

    <bind nodeset="chance"
          required="false()"/>

    <bind nodeset="chance/numbers/number"
          type="xsd:integer"
          constraint=". &gt; 0 and . &lt; 51"/>

    <bind nodeset="chance/stars/star"
          type="xsd:integer"
          constraint=". &gt; 0 and . &lt; 10"/>
  </model>

  <input model="euro" ref="count">
    <label>Number of chances (1-6)</label>
    <help>The second field must be a number
      between 1 and 6 (both included).</help>
  </input>

  <repeat model="euro"
          number="count"
          nodeset="chance">
    <repeat number="5" nodeset="numbers">
      <input ref="number">
        <label>Enter a number from 1 to 50.</label>
      </input>
    </repeat>
    <repeat number="2" nodeset="stars">
      <input ref="star">
        <label>Enter a number from 1 to 9.</label>
      </input>
    </repeat>
  </repeat>
</service>
```

This sample document declares a single request. Requests are identified by the model *id* keyword, which must be unique across all services. The specific format of a request is defined using the XForms standard. A service may declare as many requests as needed, one XForm per request.

XForms is W3C's specification for a generation of web forms that can be used with a wide variety of platforms including desktop computers, hand helds, information appliances, and even paper.

XForms standard defines a set of controls that are directly usable inside XHTML and other XML documents. An important concept in XForms is that forms collect data, which is expressed as XML instance data. Among other duties, the XForms Model describes the structure and the validation logic associated to the instance data. This is important, since the validation process may take place in the client application.

AMUSE supports both client and server side form validation, depending on the access method used by the client. A graphical Java front-end can be installed in the client handset to help her/him interact with AMUSE in friendly manner.

Besides this generic XForm interpreter. An specific Java front-end has been designed for the lotteries service. This client application displays graphically the lottery slips to users, where they can choose their numbers in a similar way as they do in the traditional paper form.

Those who don't have a Java based phone can directly use SMS to send requests to the service. For plain SMS access the form validation takes place in the server, forms are never sent to the user. The user sends input data in a SM having fields separated by blanks. Is on the server side where input is matched against the request XForm and validated. The request XForm is identified by a keyword. Messages shown next are examples of this type of SMS based requests.

Recalling the European lottery example, a typical SMS request to this service could contain the text[6]:

```
euro 2   5 12 28 29 30  2 8   7 9 17 34 50  6 7
```

chance nodesets are optional[7]. An user omitting them would tell the service to choose the numbers at random. Thus

```
euro 2
```

is also a valid request.

If an user sends an erroneous request, the system responds with a help message associated to the offending control.

Request: `euro two`

Response: `The second field must be a number between 1 and 6 (both included).`

[5] Validation may take place in the client front-end, as we'll see on section 4.

[6] `euro` is the keyword identifying the request

[7] required="false()"

Notice that the sample form declaration may lead to an incorrect selection of numbers, since constraints bound to *chance* nodesets are not enough to guard us from repeated numbers.

3.2.2 Payment Mechanisms

Lotteries can not be sold on credit, nor use disownable payment systems. Prepaid accounts are the most appropriate choice on this kind of service. AMUSE integrates an account manager to handle prepaid accounts. Subscribers are shared among all available services.

During the analysis stage, four payment mechanisms were considered in first place: Mobipay [12], ePagado [13], simpay [14] and premium services.

As we started to dig into the real state of each platform, we found that mobile payment in Spain is not yet completely adopted and still pending are some legal issues. The announcement of simpay was in first place very promising, but soon we turned out to think that this system might not ever get to market. For simple services, the Premium payment system could be suitable[8], but this type of payment is beyond the domain of AMUSE.

AMUSE provides utilities to work with both payment systems: Mobipay and ePagado. In addition, our current SMPP implementation supports the entire specification of this protocol, version 5.0, which includes several PDU attributes for billing information, so this kind of payment procedure through the SMSC should also be possible.

Currently our customers may use Mobipay or ePagado to add funds to their service's prepaid account. Lottery slips are charge directly to this account. We decided not to use Mobipay, neither ePagado, on every individual transaction, because these systems would increase user's request cost in around 30 euro cents.

4 Client Front-Ends

Plain SMS interface makes AMUSE compatible with just about every cell phone in the market[9]. New terminal capabilities are also exploited by providing a better and more user friendly access via a Java front-end that must be installed on the client's phone.

Client front-ends receive and interpret locally the XForm associated to a request. XForms are compressed, encapsulated inside a SM[10], and sent to the client. The XForm is sent to the user only if the XForm associated to the request is not already locally available, or whenever a newer version of the XForm exists in the server.

The front-end displays the form to the user, gathers input data and sends the submission information to the service.

```
<?xml version="1.0"?>
<euro>
  <count>1</count>
  <chance>
    <number>4</number>
    <number>10</number>
    <number>14</number>
    <number>23</number>
    <number>45</number>
    <star>2</star>
    <star>7</star>
  </chance>
</euro>
```

Validated XML submission data is compressed, encapsulated inside a SM and sent to the server.

5 Accessing HTTP Services

A HTTP proxy service has been implemented to easily communicate AMUSE with preexisting web like services. AMUSE provides a HTTP proxy that automatically translates SMS requests into HTTP posts.

We could easily create a web service for giving tidal information using the open source XTide [15] software. Implementing a SMS service to send this information to subscribers would be a matter of minutes.

```
<?xml version="1.0"?>
<service>
  <model id="tide">
    <instance>
      <tide><location/><day>0</day></tide>
    </instance>

    <bind nodeset="day" required="false()"
          type="xsd:integer"/>
  </model>

  <input model="tide" ref="location">
    <label>Location:</label>
    <help>Enter the name of a location</label>
  </input>

  <input model="tide" ref="day">
    <label>Day:</label>
    <help>For when you want the predicion?
      Please input a number, 0 for today,
  1 for tomorrow...</help>
  </input>
</service>
```

This XML document declares the requests associated to a simple SMS service that gives us information about the state of the ocean tides in the location we enter.

To consult tides for tomorrow in the city of La Coruña, we would send a SM such us

```
tide Coruna
```

In response, a SM with the tidal information is sent back to our phone.

```
 3:13 (3.31 m) High Tide.
 6:56 Sunrise.
 9:20 (0.53 m) Low Tide.
15:39 (3.74 m) High Tide.
21:58 (0.49 m) Low Tide.
22:08 Sunset.
```

It'll be also possible to open an Internet session from our phone browser and visit a bookmarked site giving tidal information, but let us mention that in Spain, today, such an Internet access could be

[8] Fixed price per SM, with a maximum amount of 2 Euros in Spain

[9] Samples on this type of access were given in section 3.2

[10] one or more

up to three times more expensive to customers than the corresponding SMS based service. Anyway, accessing HTTP services is just an add-in we thought could be useful for some special occasions: for implementing WAP-Push based services, or to provide a simple way for letting final users define their own alerts or SMS based services.

6 OSERL: Open SMPP Erlang Library

The SMPP protocol defines a set of operations, each one taking the form of a request and response PDU (Protocol Data Unit).

There are only a few open source projects devoted to SMPP implementation, like *Open SMPP* by Logica [16] and an *Open Source WAP and SMS gateway* by the Kannel Software Foundation [17]. The latter goes well beyond an SMPP API, providing a full product to host a WAP gateway (however it does provide support for SMPP). Logica's SMPP implementation is an open source library programmed in Java, which made it unsuitable for our development. On the other hand, reusing SMPP related code of Kannel's WAP gateway is not straightforward. Anyway, none of these tools cover the entire SMPP protocol specification as OSERL does.

The first challenge to face SMPP protocol implementation was to develop packing and unpacking functions for every command PDU. The entire protocol specification was translated into Erlang terms using a predefined set of primitive types (read records). Then, the encoding and decoding functions for the primitive types were implemented, and the whole problem of the SMPP PDU packing/unpacking cleanly solved.

But this is not the only improvement offered by OSERL. Besides the PDU format, encoding (decoding) mechanisms and associated error codes, the SMPP protocol specification defines the way to implement an application with a proper SMPP *behavior*. Even though the behavior of every ESME is predefined by the protocol specification in many senses, nowadays every developer must program these dynamic aspects of the protocol on its own. OSERL provides a generic ESME and a generic SMSC behavior implementation that transparently handles just about every feature of the SMPP protocol that leaves room for automation, and there are many of them (as we will see on section 6.2).

OSERL is an open project and it is freely available at the Source Forge [18].

6.1 ESME and SMSC Behaviors

OSERL defines a couple of SMPP behaviors, generic ESME and SMSC behaviors. Behaviors are formalizations of design patters which can be used to program certain common problems, an ESME or SMSC in this particular case. The ESME and SMSC behaviors were defined as an extension of OTP's standard *gen_server* behavior.

gen_esme and *gen_smsc* behaviors export the same set of callbacks that *gen_server* does, having these callbacks the exact same meaning as their homonymous functions defined in OTP's standard behavior. In addition to these server callbacks, *gen_esme* defines:

handle_outbind/3 Handle outbind operations.

handle_alert_notification/3 Handle *alert_notification* operations.

handle_operation/3 Handle *data_sm* and *deliver_sm* operations.

handle_unbind/3 Handle unbind requests.

handle_listen_error/1 Handle listener socket failures.

While the *gen_smsc* behavior exports:

handle_bind/3 Handle *bind_receiver*, *bind_transmitter* and *bind_transceiver* operations.

handle_operation/3 Handle *broadcast_sm*, *cancel_sm*, *cancel_broadcast_sm*, *query_broadcast_sm*, *query_sm*, *replace_sm*, *submit_multi*, *submit_sm* and *data_sm* operations.

handle_unbind/3 Handle unbind requests.

handle_listen_error/1 Handle listener socket failures.

Never a malformed or unexpected PDU would trigger a callback. SMPP errors are detected and transparently handled by lower layers.

Implementing an ESME or a SMSC is greatly simplified by means of these behaviors. SMS developers only need to implement the logic of their applications by defining the set of callbacks exported by *gen_esme* or *gen_smsc*, forgetting about SMPP errors, malformed PDUs, congestion and so forth.

6.2 Features

Due to the behaviors described in previous section, OSERL handles transparently many aspects of the SMPP protocol, saving lots of work to developers by:

- Automatically handling PDU sequencing: assigns monotonically increasing sequence numbers and matches requests with their corresponding responses in an asynchronous manner.
- OSERL automatically handles all timers defined by the SMPP protocol specification version 5.0: session init timer, enquire link timer, inactivity timer and response timer. On a timer expiration, the action recommended in the protocol specification is triggered.
- Handles every operation failure, including when the PDU is unrecognized or malformed, invalid field length, PDU data is unexpected and deemed invalid and the PDU is not allowed in the current state.

  Upon failure the appropriate response is automatically issued. Even errors associated to individual parameters of the PDUs are detected and reported using the corresponding error code.
- Detects and reports connection failures.
- Controls congestion using the *congestion_state* field.
- All the forward and backwards compatibility guidelines were adopted.
- Supports every operation and every field of the SMPP protocol version 5.0.

6.3 Support for other SMS Protocols

Currently EMI UCP protocol [19] is also being implemented. Other protocols such us CIMD [20] or OIS [21] are planned to be added in a recent future to OSERL. The implementation of these protocols is subject to the requirements of the operators we hire to launch the lottery service.

7 Conclusions

In this paper we have described AMUSE, an ESME developed to host multiple gambling services. Resulting ESME turned out to be a versatile ESME that automates the process of creating SMS interfaces for a wide variety of form based services. AMUSE offers an unique tool for multiple services, enhanced with supervision, fault tolerance mechanisms, and the capability of running on a distributed production environment. Our requirements have been fulfilled in part by choosing Erlang/OTP as the development environment.

OTP (Open Telecom Platform) is a development system platform for building telecommunications applications, and a control system platform form running them. Is not just that Erlang/OTP became the perfect development environment for our project. Adopting this technology permitted us to move away from our initial centralized conception of the lotteries service, leading us to a completely new design, a straightforward implementation, and and integrated, compact solution.

The paper also introduced OSERL, an open source SMPP library developed for AMUSE. OSERL currently gives support to SMPP. EMI UCP, CIMD and OIS are also planned to be supported in a near future.

8 Acknowledgments

Article supported by Xunta de Galicia, project PGIDT02TIC10501PR.

A Abbreviations

AMUSE A Multi-service ESME

AMUSEMENT AMUSE Main Entity

DB Data Base

EMI External Machine Interface

ESME External Short Message Entity

GUI Graphical User Interface

MC Message Centre

RE Routing Entity

SM Short Message

SMS Short Message Service

SMPP Short Message Peer-to-Peer Protocol

OIS Open Interface Specification

OSERL Open SMS Erlang Library

OTP Open Telecom Platform

PDU Protocol Data Unit

UCP Universal Computer Protocol

WAP Wireless Access Protocol

XML Extensible Markup Language

B Additional Authors

Additional authors: José M. Domínguez, email: doming@udc.es.

C References

[1] J. research. Gambling on mobile: key sectors. http://www.juniperresearch.com.

[2] A. GROUP. Mobile entertainment applications & markets. http://www.arcgroup.com.

[3] L.A.E. Informe de loterías y apuestas del estado. http://onlae.terra.es.

[4] I. B. C. Services. El negocio de los servicios y aplicaciones de telefonía móvil en el mercado residencial. Technical Report estudio de mercado, IBM, 2003.

[5] D. de Electrónica y Sistemas. Universidad de A Coruña. Estudio de mercado sobre nuevos servicios de loterías a través del móvil. http://www.moviapuestas.com.

[6] Extensible markup language (xml). http://www.w3.org/XML/.

[7] Xforms - the next generation of web forms. http://www.w3.org/MarkUp/Forms/.

[8] C. W. J. Armstrong, R. Virding and M. Williams. *Concurrent Programming in ERLANG*. Prentice Hall, 1995.

[9] S. Forum. Short message peer-to-peer protocol specification. Technical Report Version 5.0, SMS Forum, Feb. 2003.

[10] H. N. H. Mattsson and C. Wirkström. Mnesia a distributed robust dbms for telecommunications applications. Technical report, Ericsson Telecom AB.

[11] Y. development team. Yet another web server. http://yaws.hyber.org.

[12] Mobipay, take your mobile and pay. http://www.mobipay.com.

[13] epagado, un servicio de bankinter. http://www.epagado.com.

[14] Simpay. Pay for stuff with your mobile. http://www.simpay.com.

[15] D. Flater. Xtide: Harmonic tide clock and tide predictor. http://www.flaterco.com/xtide/.

[16] Logica. Open smpp. http://opensmpp.logica.com.

[17] Kannel. Wap gateway. http://www.kannel.org.

[18] E. Marcote. Open smpp erlang library. http://oserl.sourceforge.net/.

[19] Vodafone. External machine interface description. Technical Report Version 3.2, Vodafone, Feb. 2002.

[20] N. A. S. Center. Cimd interface specification. Technical Report Release SC5B CD2, Nokia, 2002.

[21] Sema. Sema smsc version g8.1. open interface specification. Technical Report Version 5.8, Sema, Jan. 2001.