DEVELOPMENT OF A CONTENT BILLING ARCHITECTURE FOR A CELL PHONE-BASED INTERNET SYSTEM

EDUAN BASSON

Dissertation submitted in fulfilment of the requirements for the degree **Master in Electronic Engineering** at Northwest University, Potchefstroom Campus.

Promoter: Dr. J.F. van Rensburg

November 2006

Pretoria

Abstract

The convergence of the cell phone industry and the Internet brings with it a myriad of services, to be accessed when on the move. It also brings a need for a paradigm shift in its billing mechanism. The current billing operation does not cater for different services on the same channel with different ratings. It also often makes it impossible to bill in real time.

These two limitations may cause revenue loss for cell phone operators, especially since the South African market focuses on prepaid users requesting services that may range from as little as 5c to R10 or R20 per service. Additional functionality is needed to eliminate discouraging, non-configurable error messages generated by the network when services are down, or when real-time billing systems refuse delivery of a service due to insufficient funds on the user's account.

Motivated by these restrictions, it seemed not only sensible but very important to assess the state of the industry and available technologies, in order to define a new content billing architecture. Specifications and requirements for such a solution were researched and formulated, in order to complete the design and implementation.

The choice for a cell phone Internet technology to support the new system fell on the Wireless Internet Gateway (WIG), as it caters to 99% of the South African market by delivering text-based Internet content to 2G cell phones. The content-based billing application (CBA) was developed and tested in the laboratory to verify its functionality and performance. It implements all the specifications and requirements.

The CBA was then installed on the network of a South African cell phone operator. End-to-end performance tests on the complete system, including its latency and throughput levels, confirmed that it successfully implemented all requirements and business logic.

Samevatting

Die sameloping van Internet- en selfoontegnologieë bring 'n groot verskeidenheid dienste mee wat op feitlik enige plek toeganklik is. Hierdie gevarieerde dienste noodsaak 'n nuwe meganisme vir kosteverrekening. Die huidige rekeningstelsel neem nie in ag dat verskillende dienste, gelewer op dieselfde kanaal, verskillende kostes kan hê nie. Dit is dan ook byna onmoontlik om kliënterekeninge oombliklik te debiteer.

Hierdie leemte kan uitloop op 'n inkomsteverlies vir selfoonoperateurs, veral omdat die mark grootliks uit voorafbetaal-gebruikers bestaan en die koste van dienste enigiets tussen tien sent en R20 kan beloop. Ontmoedigende foutboodskappe aan gebruikers wat dienste aanvra wat nie beskikbaar is nie of waarvoor die fondse in die rekening te min is, moet ook uitgeskakel word om 'n vriendeliker gebruiksondervinding te skep.

Hierdie motiverings het aanleiding gegee tot 'n ondersoek na die huidige stand van die industrie en beskikbare tegnologie. Die behoefte aan 'n inhoudgebaseerde kosteverrekeningstelsel was duidelik. Die vereistes en spesifikasies vir so 'n oplossing is dus nagevors en neergelê vir die ontwerp en implementering van die nuwe stelsel.

Die keuse vir 'n selfoon-Internettegnologie om die nuwe produk te ondersteun het op die "Wireless Internet Gateway" (WIG) geval, aangesien dit feitlik die totale Suid-Afrikaanse mark vir teksgebaseerde Internet-inhoud na 2G-selfone bedien. Laboratoriumtoetse het getoon dat die werkverrigting en funksionaliteit van die nuwe meganisme vir inhoudgebaseerde kosteverrekening, wat gedoop is as CBA, aan al die spesifikasies en verwagtings voldoen.

Die CBA is toe in die netwerk van 'n Suid-Afrikaanse selfoonoperateur geïnstalleer. Vergelykende toetse op die totale stelsel het bewys dat dit goeie sakelogika akkommodeer en 'n funksionele oplossing bied vir die probleme van inhoudgebaseerde kosteverrekening.

Acknowledgements

I am grateful to everybody who supported me throughout the course of my studies.

Firstly, I would like to thank my previous employer, Atos Origin, for the opportunity to do the research, and for providing the laboratory environment for testing and implementing this project.

I would also like to thank everyone at CRCED Pretoria, and especially Dr Johann van Rensburg who acted as my mentor. They made the process running smoother than I ever thought possible and turned my studies into a uniquely educational experience.

Then I would like to express my gratitude to all my friends who stayed by me all through this sometimes lonesome affair.

I often take the ongoing support of my parents for granted. So I would like to use this opportunity to put it in black and white: thanks, mom, you're the reason for almost all of my good qualities; you have always been a great help in everything I've ever asked from you. All my other strong points I owe to my father; thanks, dad for your wisdom and guidance that has been with me all my life and throughout the course of this project.

Abbreviations

3GPP	3G Partnership Project
3GSM	3rd Generation GSM
ATP	Acceptance Test Procedure
CBA	Content Billing Application
CDMA	Code Division Multiple Access
DNS	Domain Name Server
CP	Content Provider
CRE	Content Rating Engine
DP5	Delivery Platform 5
EDGE	Enhanced Data-rate for GSM Evolution
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IP	Internet Protocol
JRE	Java Runtime Environment
LAN	Local Area Network
MMS	Multimedia Messaging Service
MSIE	Microsoft Internet Explorer
OS	Operating System
RAM	Random Access Memory
RFC	Request for Comment
RPM	Red Hat Package Manager

SDLC	Software Development Life Cycle
SIM	Subscriber Identity Module
SM	Short Message
SMS	Short Message Service
SMSC	Short Message Service Centre
SOAP	Simple Object Access Protocol
STK	SIM ToolKit
TAR	Tape Archive
TCP	Transmission Control Protocol
TDMA	Time Division Multiple Access
URL	Uniform Resource Locator
USSD	Unstructured Supplementary Service Data
VAS	Value Added Services
W3C	World Wide Web Consortium
WAN	Wide Area Network
WAP	Wireless Application Protocol
WIG	Wireless Internet Gateway
WML	Wireless Markup Language
WWW	World Wide Web
XML	eXtensible Markup Language

Table of contents

ABSTRACT II

SAMEVATTING III

ACKNOWLEDGEMENTS IV

ABBREVIATIONS V

TABLE OF CONTENTS VII

LIST OF FIGURES IX

LIST OF TABLES X

1 INTRODUCTION 1

1.1 Preamble 1

1.2 Evolution of the cell phone industry 1

1.3 Overview of the cell phone industry in South Africa 4

1.4 Value added services in the cell phone industry 6

1.5 Internet technologies 9

1.6 Convergence of cell phone and Internet technologies 11

1.7 Overview of existing content billing applications 14

1.8 Need for new content billing architecture 18

1.9 Overview of the dissertation 20

2 REQUIREMENTS FOR A NEW CONTENT BILLING ARCHITECTURE 21

2.1 Introduction 21

2.2 Peripheral systems and technologies 21

2.3 Communications interfaces between peripheral systems 23

2.4 Hardware and software requirements for content billing architecture 25

2.5 Overview of requirements for new content billing architecture 28

2.6 Conclusion 30

3 DESIGN AND IMPLEMENTATION OF THE CONTENT BILLING ARCHITECTURE 32

3.1 Introduction 32

3.2 High-level specifications 33

3.3 Interfacing with peripheral systems 34
3.4 Functional overview of programme logic 37
3.5 Sequence of actions 37
3.6 Design and implementation 39
3.7 Functional testing of architecture requirements 42
3.8 Conclusion 53

4 APPLICATION AND TESTING OF NEW CONTENT BILLING ARCHITECTURE 54

4.1 Introduction 54
4.2 Application and testing environment 55
4.3 Testing and verification procedure 57
4.4 Test results and analysis 62
4.5 Example of the functionality of the new content billing architecture 65
4.6 Conclusion 67

5 SUMMARY AND RECOMMENDATIONS 68

5.1 Conclusion 68
5.2 Recommendations for future development 69

REFERENCES 71

APPENDICES 76

APPENDIX A DESIGN FLOW DIAGRAMS 76

APPENDIX B INSTALLATION AND CONFIGURATION 80

APPENDIX C LABORATORY TEST RESULTS 86

APPENDIX D ACCEPTANCE TEST RESULTS 98

APPENDIX E END-TO-END TEST DATA 100

List of figures

Figure 1-1: Timeline of the three generations in cell phone history 4
Figure 1-2: Convergence between Internet and cell phone technologies 11
Figure 1-3: SmartTust DP5 WIG 13
Figure 1-4: A standard billing process 16
Figure 2-1: SmartTust DP5 WIG 21
Figure 2-2: Basic connectivity between WIG and Content Provider 22
Figure 2-3: Functional diagram of the CRE 23
Figure 2-4: Connectivity of the peripherals in the content billing system 24
Figure 2-5: The relationship between software technologies on a server 26
Figure 2-6: Proposed solution and its connections with peripheral systems 28
Figure 3-1: The role of the CBA in connectivity 32
Figure 3-2: Interfaces in the content billing system 34
Figure 3-3: Sequence of Actions 38
Figure 3-4: Software modules of CBA 40
Figure 3-5: Software Development Life Cycle 43
Figure 3-6: Test configuration for WIG Interface test 44
Figure 3-7: Test configuration for CRE Interface test 45
Figure 3-8: Test configuration for CP Interface test 46
Figure 3-9: Test configuration for performance tests 52
Figure 4-1: Position of CBA in Content Delivery System 54
Figure 4-2: Environment used for end-to-end tests 62
Figure 4-3: Throughput 63
Figure 4-4: Latency 64

List of tables

Table 1-1: State of the GSM market 2004 5
Table 2-2: Comparison of existing billing applications based on functionality 18
Table 3-1: Requests on the CRE Interface 36
Table 4-1: Test cases 59
Table 4-2: Summary of the results of the ATP 61
Table 4-3: Visual representation of the content billing functionality. 66

1 Introduction

1.1 Preamble

The cell phone industry is in its fourth decade, and during this time it brought forth a rich world of useful services, each with its own technologies and needs. This dissertation concerns one of these needs, namely content billing for Internet services hosted on cell phone networks. This service relates to the convergence of two different technologies: Cell phones and the Internet.

Cell phones and the Internet are briefly discussed in this section, both as technologies and as industries, and the convergence of the two is outlined. Thereafter, the dissertation addresses a specific need that is created by this convergence, namely that of content billing, mentioning existing systems and outlining the reasons for the development of a new solution.

1.2 Evolution of the cell phone industry

A historical overview of the evolution of the cell phone industry can be done based on the industry's own classification of first, second and third generations. First generation refers to the analogue cellular phones of the 80s. The second generation is marked by the move to digital technologies, while the third generation involves the addition of data functionalities like cell-phone based Internet access.

On April 3, 1973, Motorola employee Dr. Martin Cooper placed a call to rival Joel Engel, head of research at AT&T's Bell Laboratories, while walking the streets of New York City talking on the first Motorola DynaTAC prototype [1]. Motorola has a long history of making automotive radios, especially two-way radios for taxicabs and police cruisers. In 1978 Bell Laboratories launched a trial of the first commercial cellular network in Chicago using the now dated Advanced Mobile Phone System [2].

1.2.1 First Generation Cellular

The entry into the commercial world of the first generation cellular phone was made by Motorola with the DynaTAC 8000X [1]. This generation of analogue cell phone was never commercially introduced in South Africa, but began to proliferate in some of the countries through the 1980s with the introduction of cellular networks with multiple base stations located relatively close to each other. Protocols were developed to handle the automated handover between two cells. As soon as the signal strength of a particular cell overrides the signal strength of another cell, the handset will initiate this handover, making true mobile communication possible.

In Switzerland, the name for the big car-based phone model was "Nationales Autotelefon", and the abbreviation of it (Natel) persists as the common designation for mobile phones. Soon, some of these bulky units were converted for use as transportable phones the size of a briefcase. Motorola introduced the first truly portable, hand-held phone. These systems later became known as first generation (1G) mobile phones [3].

In September 1981 the first cell phone network with automatic roaming was started in Saudi Arabia, with the Nordic countries following one month later with automatic roaming between countries [4].

1.2.2 Second Generation Cellular

In the 1990s, second generation (2G) mobile phone systems began to be introduced. The first digital cellular phone call was made in the United States in 1990, and in 1991 the first Global System for Mobile Communications (GSM) network opened in Europe. 2G phone systems were characterised by digital circuit switched transmission and the introduction of advanced and fast phone to network signalling [5].

The scaling down of the technology started, with tiny 100g to 200g handsets becoming the norm. This change was due to technological improvements such as more advanced

batteries and more energy-efficient electronics, but was also largely the result of the higher number of cellular base stations caused by increasing usage levels [6].

1.2.3 Third Generation Cellular

The third generation followed the second generation closely, and while the popular (and commercial) definition might have been a little fuzzy of what set of functions the third generation encompassed, the IMT-2000 definition standardised its meaning. The protocols and services contained in this function set consisted of many different standards, with different contenders pushing their own technologies. So, this process did not standardise on a technology, but rather on a set of requirements (2 million bits per second maximum data rate indoors, 384 kilobits per second outdoors, for example [6]).

During the development of these third generation systems, a half-breed arose in the marketplace, dubbed 2.5G, including functionality for General Packet Radio Service (GPRS). This system provided some of the features of 3G without fulfilling the promised high data rates or the full range of multimedia services. For example, CDMA2000-1X (CDMA: Code Division Multiple Access) delivered theoretical maximum data speeds of up to 307 kilobits per second [7]. This was purely a marketing term, and there was no standard for it.

Presently, live streaming of radio and television to third generation handsets is developing internationally, but is still a future plan for South Africa.

To put the chronological view of these technologies in perspective, see Figure 1-1.

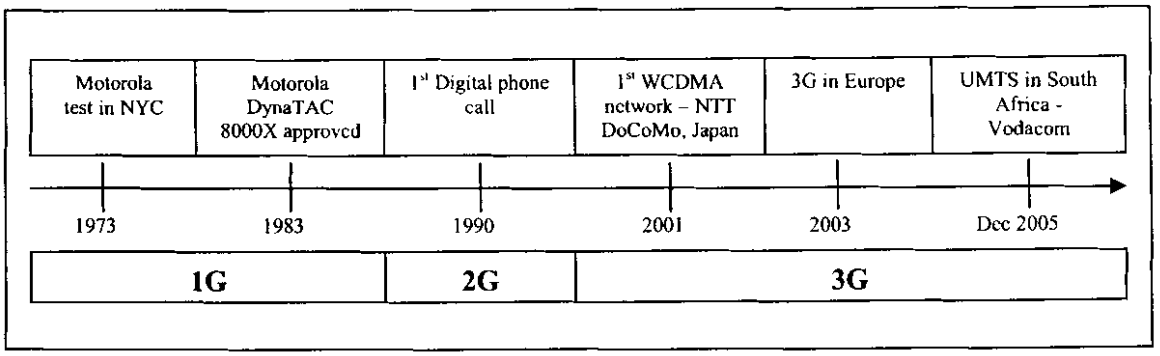


Figure 1-1: Timeline of the three generations in cell phone history

1.3 Overview of the cell phone industry in South Africa

The market in South Africa currently consists of three cell phone operators, with the fourth, Virgin Mobile, presently being launched.

MTN and Vodacom were issued mobile licenses in 1993, with the third license awarded to Cell C in February 2001 [8]. The third generation offerings of data and multimedia brought forth a growth of 39% in revenues to R3.5 billion in 1999/2000 [8].

The singular fixed line operator, Telkom, currently focuses on bandwidth availability, which is one of the drivers behind the current convergence of the telecommunications, information technology and broadcasting industries. The total number of 2 megabyte circuits grew by 53% in 1999/2000, with even higher growth of 78% in Integrated Services Digital Network primary rate services [8]. This has an impact on the cell phone industry, as the monopoly by Telkom forces customers to look toward the cell phone operators for this convergence.

The following usage statistics were for 2004, the year in which this development was implemented and, therefore, the relevant year for this dissertation. Current usage statistics show a total of 32.4 million cell-phone users in South Africa, according to the Active 3 method of only counting customers who have used the service in the last concurrent 90

[19]. This figure could be misleading, as the anonymity of prepaid packages makes it hard to extrapolate a meaningful headcount of users.

The following figures show the state of the market in 2004, when this study was conducted.

Global mobile users	2 billion
SA Mobile users	18.7 million
SA Prepaid users	16 million
African users	83 million
Global 3G users	130 million
Global monthly SMS	36 per user

Table 1-1: State of the GSM market 2004 [10]

People in previously under-serviced areas in South Africa are making over 35 million calls (1.5 million minutes) per month from Vodacom's 2 135 community phone shops [19]. This highlights the necessity to cater for the low end of the technological market, as these users will typically not use the third generation services due to their high costs. Most of the urban areas and national roads in South Africa have GSM 900 coverage.

If one calculates the ratio of South Africa's mobile technology users against the global number, we get

$$\frac{18.7 million}{2 billion} = 0.935\%$$

This shows that South Africa has a fair share of the global market in this industry.

According to MyADSL, the current total of 3G users locally is 104 000 [11]. Vodacom reported 179 576 3G active handsets in their 2006 business report [12]. If one uses the higher figure, the present ratio of South Africa's 3G users to the total of mobile users calculates as follows:

$$\frac{179576}{18.7million} = 0.96\%$$

This means that about 99% of South Africa's market does not possess 3G capable equipment. This shows the greater need for 2G compatible value added services, as will be discussed in the next section. Vodacom also reported that in 2005 the number of 3G users was less than 10% of the value in 2006, at 10 878 handsets. This means indicates a substantive growth in this area, and that the gap between the number of 2G and 3G users is shrinking. It is clear, though, that it will take some years for the great need for 2G services to dissipate due to the size of this gap.

South Africa's market is dominated by prepaid customers (85%, of which Cell C caters to 76% [13], Vodacom 88% [12], and MTN 84% [14]). This underlines the need for real-time billing, as a request for a service or content must be considered on the basis of the customer's current credit state.

1.4 Value added services in the cell phone industry

The third generation of GSM saw an immense growth in the offerings of Value Added Services (VAS), using Wireless Application Protocol (WAP) services and Multimedia Messaging Service (MMS). As the market in South Africa favours the low-end of the technology, VAS services are focused on utilising second generation technologies like Short Message Service (SMS) and Unstructured Supplementary Service Data (USSD).

These VAS services address users' needs for entertainment, information and connectivity. The limited and focused functionality of these services tend to emulate the experience obtained from Internet connectivity, thus fuelling the convergence between

the Internet and cell phone systems. In order to rapidly increase the Average Revenue per User, operators must go beyond the traditional methods of increasing usage of Value Added Services [15].

1.4.1 SMS

As a second generation technology, all handsets in the South African market are equipped to handle SMS messages to be sent and received. This service is restricted to alphanumeric data that allows messaging between mobile phones and other equipment such as voice mail systems and email.

SMS is a store-and-forward system [16]. Messages are sent to a Short Message Service Centre (SMSC) from various devices such as another mobile phone or via email. The SMSC interacts with the mobile network to determine the availability of a user and the user's location to receive a Short Message (SM).

Because SMS uses the control channel (rather than the voice channel), a unique feature of SMS is that the user can receive an SMS whether or not a call is in progress. If the handset is not turned on, or not ready to receive a message due to processor restrictions, the SMSC will wait for a signal from the handset or the message will go into a retry queue [17]. A delivery receipt is sent to the SMSC upon delivery to the handset, allowing the SMSC to provide confirmation of receipt to the sender upon request.

Because of this store-and-forward method, the immediate delivery of any SMS cannot be guaranteed, and systems that rely heavily on delivery of multiple SMs may have unpredictable response times.

1.4.2 Multimedia Messaging Service (MMS)

Another store-and-forward system, introduced in the third generation, but with more complex data formats than SMS, is the MMS. MMS allows mobile subscribers to exchange multimedia messages. It may be seen as an evolution of SMS, with MMS supporting the transmission of additional media types:

- Text
- Picture
- Audio
- Video
- Combinations of the above.

MMS is an important emerging service, which allows the sending of multiple media in a single message, and the ability to send a message to multiple recipients. The originator can easily create a Multimedia Message, either by using a built-in or accessory camera, or by using images and sounds stored previously in the phone (and possibly downloaded from a web site) [18].

1.4.3 Unstructured Supplementary Service Data (USSD)

The GSM standard defines a separate channel for transmitting information in an unstructured format, called USSD. USSD provides session-based communication, enabling a variety of applications.

In operation, USSD is used to send text between the user and an application. USSD should be thought of as a trigger rather than an application itself. However, it enables other applications. In operation, it is not possible to bill for USSD directly, only for the

application associated with the use of USSD such as circuit switch data, SMS, or prepaid recharge [19].

The limitations in the functionality of USSD stems from its configuration procedure: For every request code that is sent from a handset to the operator, a similar code must be configured on the USSD gateway. This means variables inside request codes cannot be used. USSD can be very effectively used to request data, and this functionality is currently available.

1.4.4 Wireless Application Protocol (WAP)

WAP is an enabling technology based on the Internet client server architecture model, for transmission and presentation of information from the World Wide Web (WWW) and other applications utilising the Internet Protocol (IP) to a cell phone or other wireless terminal. Ericsson, Motorola, Nokia, and Phone.com founded the WAP Forum in June 1997 to create license-free standards for the entire industry to use in order to develop products based on WAP [20].

While the early implementation of WAP have been relatively unimpressive from a user experience, WAP is poised to leverage packet data networks, push-based services, colour and animation on the handset, and value-added services such as location-based services.

1.5 Internet technologies

The Internet as we know it consists of protocols and specifications agreed upon by different associations. Its core technology may be explained by looking at the transfer protocol and the language of the web pages. The mechanics of the underlying hardware layer, or networks, will be regarded as outside of the scope.

The main technologies responsible for the mechanics of the Internet are:

- **HyperText Transfer Protocol (HTTP).** This is the protocol used to transfer web-page data on the Internet. This is a stateless transaction-based protocol, and may be seen at the beginning of all Uniform Resource Locators (URLs) – or Internet addresses in the form of http://... .
- **HyperText Markup Language (HTML).** This is the language used to encode all internet pages, to be interpreted by web browsers to display text, images and other objects, all stored on a web server.
- **Internet applications.** Web pages have evolved from static information pages to become more and more dynamic. Web server logic and client side scripting like Javascript made more complex, interactive and useful Internet services possible, from search engines to photo albums to online bookings and reservations – the possibilities are endless.
- **Email.** The Internet is also responsible for passing electronic messages from senders to recipients, called emails. This technology is not relevant to this dissertation, and will not be discussed further.

methods are defined to request data, and thus create a transaction on the HTTP protocol. These two methods are called HTTP "get" and "post" requests. In the case of a "get" request, a URL (starting with http://) is followed by attribute names and value pairs, separated from the URL by a question mark, and separated from each other by ampersands. An example of such a request is:

`http://www.example.com/examplepage?attribute1=value1&attribute2=value2`

A "post" request has these attributes and values contained in the body of the request forming part of the HTTP packet. Thus such a request will not contain the attributes and values in the address itself. The details of this type of request fall outside the scope of this dissertation.

1.6 Convergence of cell phone and Internet technologies

The enormous growth of the Internet as an extensive source of information has caused a need to access data even when on the move. For this purpose, cell phone connectivity to the Internet has grown as a new product that cell phone operators needed to focus on. Different protocols have been developed to enable this interconnectivity, with the WAP protocol acting as the gateway between the GSM technology and the fixed network of the Internet.

The development of different technologies that enable this interconnectivity is causing a convergence between the Internet and cell phone worlds, as depicted in Figure 1-2.

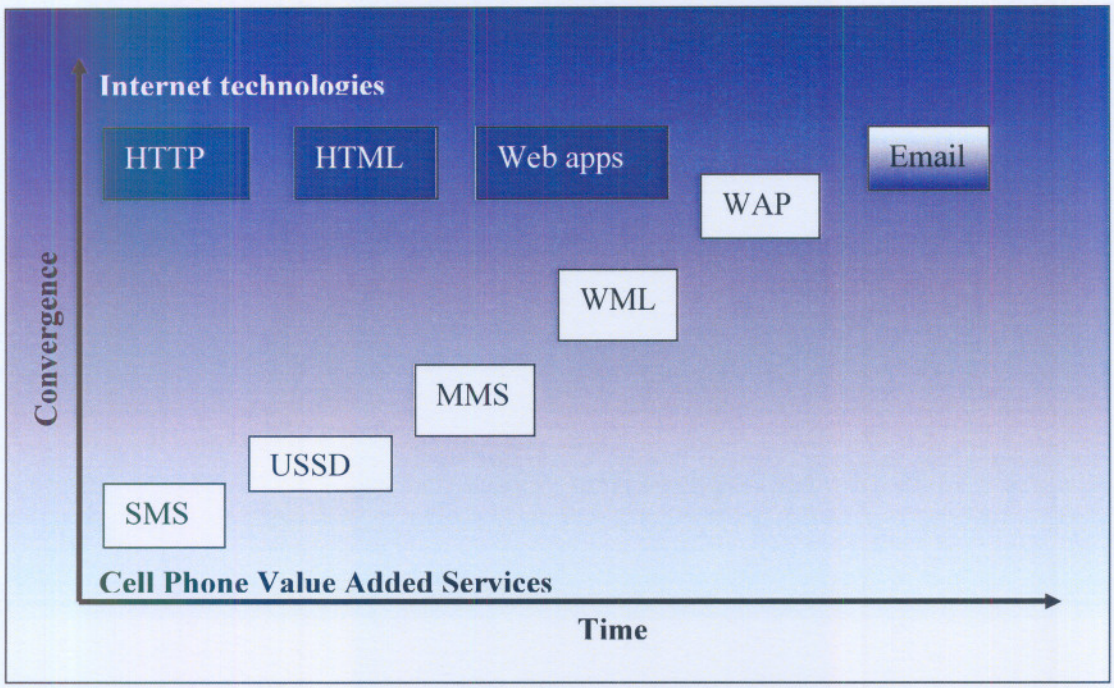


Figure 1-2: Convergence between Internet and cell phone technologies

This section delves into these convergent technologies, namely technologies that enable the transmission of Internet data over cell phone networks and into handsets.

Key players in the delivery of Internet content to cell phones, currently, are:

- **General Packet Radio System (GPRS).** GPRS is a connectivity solution based on IPs that supports a wide range of enterprise and consumer applications. With throughput rates of up to 40 kilobits per second, users have a similar access speed to a dial-up modem, but with the convenience of being able to connect from anywhere. For operators, the adoption of GPRS is a fast and cost-effective strategy that not only supports the real first wave of mobile Internet services, but also represents a big step towards 3GSM (or wideband-CDMA) networks and services [21].
- **Enhanced Data-rate for GSM Evolution (EDGE)** provides up to three times the data capacity of GPRS. Using EDGE, operators can handle three times more subscribers than GPRS. EDGE uses the same Time Division Multiple Access (TDMA) frame structure, logic channel and 200 kHz carrier bandwidth as today's GSM networks, allowing it to be overlaid directly onto an existing GSM network. For many existing GSM/GPRS networks, EDGE is a simple software-upgrade [22]. EDGE allows the delivery of advanced mobile services such as the downloading of video and music clips, full multimedia messaging, high-speed colour Internet access and email on the move.
- **3rd Generation GSM (3GSM)** enables the provisioning of mobile multimedia services such as music, TV and video, rich entertainment content and Internet access. The technology on which 3GSM services are delivered is based on a GSM network enhanced with a Wideband-CDMA (W-CDMA) interface. Global operators, in conjunction with the 3G Partnership Project (3GPP) standards organisation, have developed 3GSM as an open standard [23].
- **The Wireless Internet gateway (WIG)** gives terminals (handsets with WIG-enabled browsers in this case) access to simple Internet applications. It brings connectivity to legacy handsets via SMS and supports end-to-end security, "push" and location-based services [24].

The South African user base has a large percentage of users who prefer prepaid contracts and use previous generation handsets. Keeping this in mind, a system has been adopted that makes simple Internet connectivity possible using SMS as carrier. This system is

Smarttrust's Delivery Platform 5 (DP5) WIG, indicated in Figure 1-3. This will be the technology focused on in this dissertation.

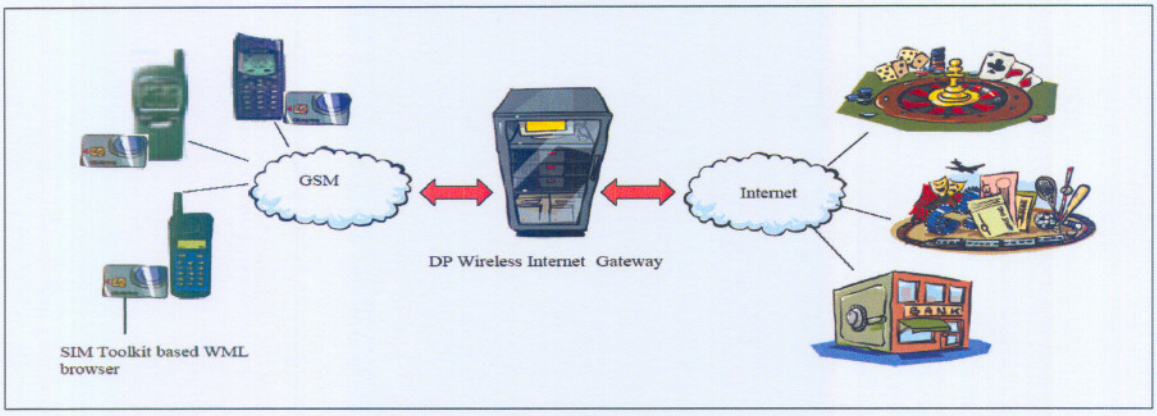


Figure 1-3: SmartTust DP5 WIG [25]

The system uses a special Subscriber Identity Module (SIM) card, the Wireless Internet Browser or WIB-enabled 32K SIM. It is compatible with most handsets on the market. SMS is used as carrier, but the Internet content is displayed with text and links as a pop-up message on the screen of the handset. These SMs are not stored on the SIM, but will only reside in memory until the current page is exited.

The DP5 is multi-functional, but for the sake of this dissertation it may be regarded merely as a WIG that converts 8-bit data SMs into HTTP. Wireless Markup Language (WML) traffic passes this request to the specified Content Provider (CP) on the Internet and converts received responses back to 8-bit data SMs. All HTTP requests are sent in the form of "get" requests, as the WML specification is limited to this form of request [26].

The data in these SMs is in a compiled byte-code format, and contains WML code that can be interpreted by the Wireless Internet Browser located on the SIM, which in turn converts the WML code into handset menu data using the SIM Toolkit (STK).

1.7 Overview of existing content billing applications

The usual motive for commercial service provisioning is profit. For that reason charging and billing are at the core of the telecommunications business [27].

As mentioned in section 1.4, VAS is instrumental in the significant boosting of the Average Revenue per User of the cell phone industry; therefore, the accurate billing of these services is crucial. Legacy systems allow billing for communication by channel and time, but as yet the content of these messages could not be analysed.

As VAS utilises a single channel for a myriad of services, a content billing system that distinguishes between the different services needs to be incorporated. This system will not only base the billing on the amount of data, the channel used and the time, but also on the nature of the content [28].

For carriers to take full advantage of emerging industry trends [29], real-time convergent billing platforms are required for subscribers to pay 'real' rates for content and services they access, such as news, games, video broadcasts, or whatever service the cell phone operator may offer.

1.7.1 Existing billing applications

The leading players in the telecommunications charging and billing environment are named in this section, with short explanations of the services their products offer.

Cisco

Cisco Systems provides a billing solution called the Content Services Gateway, which was used by more than 25 wireless carriers at the time of publication of the product documentation. It provides three capabilities to cell phone operators in order to offer

content-based billing: dynamically examining content, controlling subscriber access to services in real time, and enforcing subscriber account balances [30].

Portal Infranet

Portal's Infranet billing solution (acquired by Oracle) is an open-architecture platform that lets a company manage every aspect of customer management and billing. The server platforms that Infranet supports are HP-UX, Windows NT or 2000, and Sun Solaris, with IBM AIX support planned for the near future. The system also requires a relational database, either Oracle or SQL Server [31].

Amdocs

Amdocs Billing supports convergence for multiple technologies, including wireless, fixed line, cable and satellite, customer configurations such as prepaid or post-paid, and channels such as voice, data and broadband. Its flexible rating and billing functionality makes it easy to bundle different services and resources into sellable offers, allowing the operator to deliver the right offer to the right customer, at the right time. This allows operators to maximize the value of their customer relationships, while reducing system costs through widespread system consolidation [32].

MaxBill MaxGen

MaxGen is designed to support the needs of complex, modern billing scenarios, while being able to support the volumes of traditional fixed-line and utility billing, due to its modern architecture. It boasts a multi-threaded parallel architecture and should be run on multiple processors to take advantage of this fact. MaxGen is convergent, able to calculate charges for a variety of products and services on the same bill [33].

1.7.2 Standard billing process operation

The operation of the standard billing process will be explained, as well as the reasons why these content billing products do not fulfil the requirements of the system.

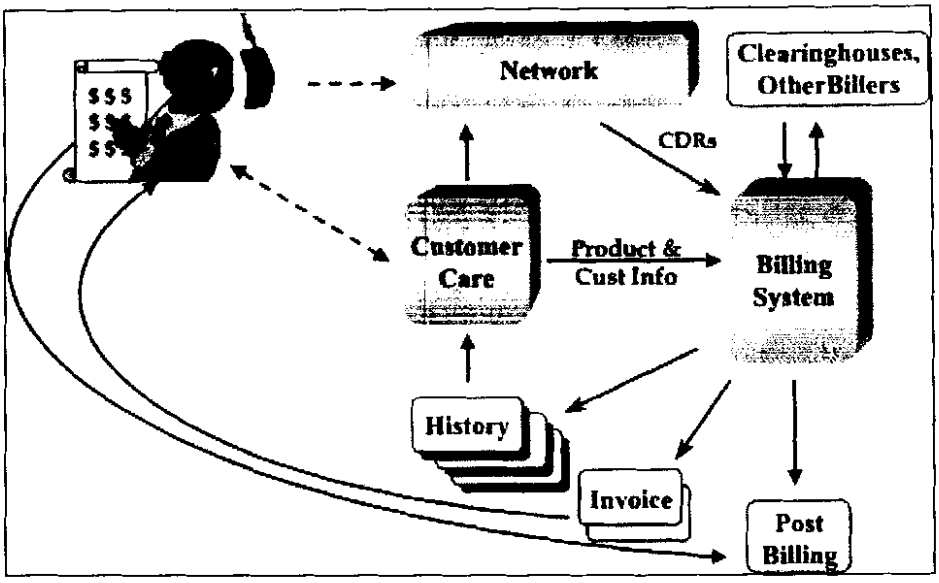


Figure 1-4: A standard billing process [34]

Figure 1-4 describes a standard billing process. In this diagram, a customer is first created on the system when the customer calls customer care or works with an activation agent. The agent (customer care) enters the customer's service preferences into the system, checks for credit worthiness, and provides the customer with a phone number so that the customer may make and receive calls through the network.

As the customer uses the operator's services, the connections made by the network and VAS systems create records of their activities. The billing system now guides and updates these Charging Data Records to their correct customer and rating information. The updated billing records are placed in a billing pool so that they may be combined into a single invoice that is sent to the customer.

The customer then sends his payment to the telecom service provider, which is recorded in the billing system. History files are then updated for the use of customer service representatives and auditing managers [34].

1.7.3 Limitations of existing billing applications

The billing process has certain limitations of which the following has an impact on this system:

- Billing records are recorded in Charging Data Records in real time, but cannot be used in real time to decide delivery of a service.
- This lack of real-time operation also causes a delay in charge reversal due to unsuccessful delivery of services.
- Flat rate charging has to be used, due to the lack of information on the content and nature of the service offered.
- The billing system cannot decide the path of service delivery. The service will always be delivered by the same CPs, as the routing of information is considered as external to the billing process.

The existing billing products named in section 1.7.1 do not all share the same limitations, but not one is totally free of these issues, as shown in Table 1-2.

Existing application	Real-time billing	Real-time charge reversal	Content-based charging	Routing
Cisco	✓	✓	✓	×
Portal Infranet	✓	✓	×	×
Amdocs	✓	✓	✓	×
MaxBill MaxGen	×	×	×	×

Table 1-2: Comparison of existing billing applications based on functionality

1.8 Need for new content billing architecture

Designing a billing system for the typical South African market, with its high percentage of prepaid users of second-generation cell phones, needs a special approach in order to maximise revenue and customer satisfaction. As shown, the existing billing solutions have not been designed with this specific situation in mind. Therefore, this dissertation proposes a new solution.

This proposed content billing architecture is designed to solve these limitations by defining the following needs:

- **Absolute real-time billing**. At the moment of request for a service, the subscribers' billing history and balance should be taken into account. The response or action should be based on this information, in order to decide whether to deliver or not.
- **Absolute real-time charge reversal**. In case a requested service cannot be delivered, due to network failure or for any other reason, the reversed charge and the original balance should immediately be made available to the subscriber for further service requests.

- **Content-based charging**. Each and every request for information must be configurable for a specific charge.
- **Billing-based routing**. The system must decide where to send requests for information, based on the subscribers' current billing history and balance. In this way, instead of the requested content, the operator may rather deliver instructions or advertisements to a subscriber with depleted funds, or when the requested service is temporarily out of order, or for any other reason.

The effects of implementing a billing service that addresses the above needs are:

- No services will be delivered to subscribers who do not have sufficient funds in their accounts at the time of their request. This means that the operator will not lose revenue or be forced to incorporate a credit plan for subscribers.
- Accurate billing for services will mean that subscribers will be charged immediately for all services received, and any error in the delivery of a service will immediately reflect on the subscriber's account as a reverse charge.
- Services can be configured to have a wide variety of prices. As service charges will no longer be based on flat rates or volumes of data, but on content, the operator's costs can be configured to 10c, or R10, or any other value.
- The current mechanics of requests sent over the WIG make it difficult to change the destination of the HTTP request. With the new content billing architecture, the address of every service can be configured just as easily as the charging for that service. Alternative destinations can also be configured to host content to be displayed, in case access to the service cannot be granted.

The new content billing architecture is designed to solve the limitations of products existing at the date of the investigation, and caters specifically as a solution for the unique needs of the South African cell phone market.

1.9 Overview of the dissertation

This introduction has shown that:

- Cell phone usage and the resultant technological advances are on a convergent path with Internet technologies.
- Revenue generated by these convergent services is of major importance to South Africa's cell phone operators, as in the rest of the world.
- South Africa's market consists of a large number of low-end users, whose equipment is not quite state of the art. Thus, operators in South Africa need to adopt a specific approach to convergence, taking into account the possibility of providing Internet services using 2G technologies.
- The current billing model is insufficient for the delivery of Internet content, and a new model based on real-time content-based billing must be developed.

This study concerns itself with implementing middle-ware into the current service delivery system at a dedicated cell phone operator, in order to enable content billing and routing as specified by the operator, and described in section **Error! Reference source not found.**. The solution will be researched, designed, implemented, installed and tested.

Section 1 will describe the requirements of the new solution. Section 1 will explain the specification and design choices made for this solution, as well as the process of developing, installing and testing this solution. Verification and analyses of the finished solution will follow in section 1, and conclusions given in section 1.

2 Requirements for a new content billing architecture

2.1 Introduction

A content billing server must be designed for a cell phone Internet system. This server will connect to the SmartTrust DP5 WIG, which currently acts as the gateway between the operator's Short Message System and the Internet, and which is introduced in section 1.6.

The WIG gives SMS-enabled terminals (in this case handsets with STK-based WIG-enabled browsers) access to WML-based applications [24], making simple Internet connectivity possible by using SMS as carrier. The basic architecture of the WIG system is shown in Figure 2-1.

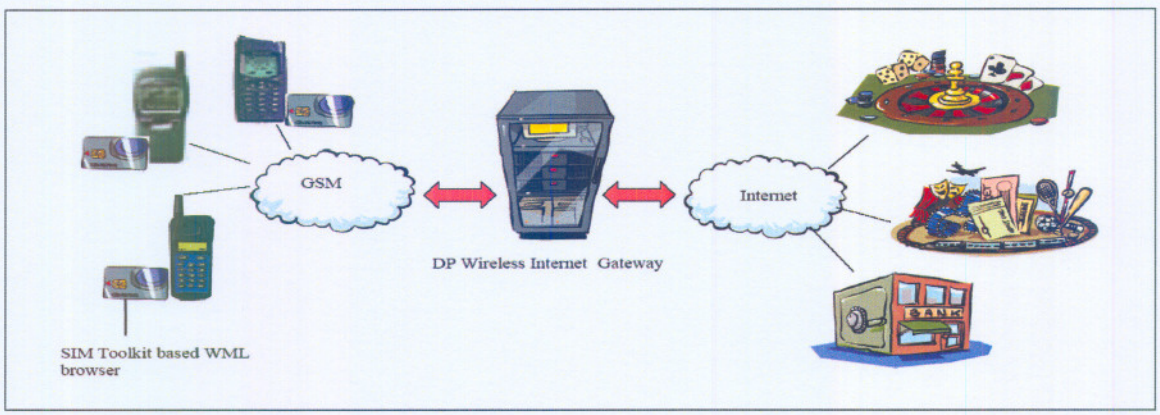


Figure 2-1: SmartTust DP5 WIG [35]

2.2 Peripheral systems and technologies

As already discussed, operators need the new billing strategy to apply to the Internet data that is currently carried through the WIG. This will then be the source of data to be analysed, and will form the client-side peripheral system to the content billing server.

A second peripheral system will be the Content Rating Engine (CRE), a server that is available for supplying rating and routing information, and will be briefly explained in section 2.2.2 below.

2.2.1 Wireless Internet Gateway

The WIG server generates Charging Data Records based on the transactions on the system, but these records only provide batch billing functionality. If an operator wants to bill in real time and thus be able to refuse services in case of lack of funds, a different implementation of billing must be introduced. Therefore, the CBA with which this dissertation concerns itself, should be designed to interface with this Internet Gateway. For more detail on the DP5 WIG system, refer to section 1.6.

The basic connectivity of the WIG platform to an arbitrary content provider on the Internet is shown in Figure 2-2 below, ignoring any network hardware in the connection that can be considered as transparent (like firewalls and routers).

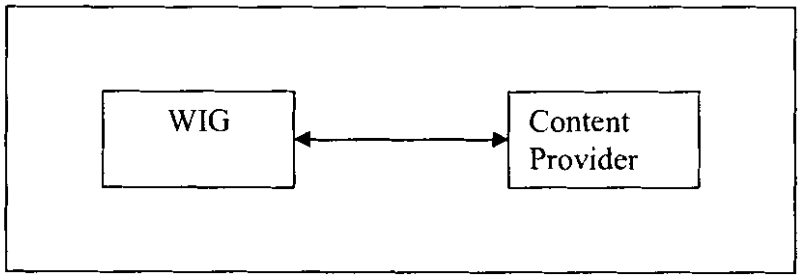


Figure 2-2: Basic connectivity between WIG and Content Provider

A system should be developed and installed between the WIG platform and the CP, shown above, to intercept requests for content coming from a user, and apply business rules to this request before forwarding it to the CP.

2.2.2 Content Rating Engine

The Content Rating Engine is responsible for linking content request strings (HTTP "get" requests in this case) with ratings configured on the CRE, and for returning routing information (a URL, in this case). Any request for rating is forwarded by the Content Rating Engine (CRE) to the billing platforms on the network, as shown in Figure 2-3.

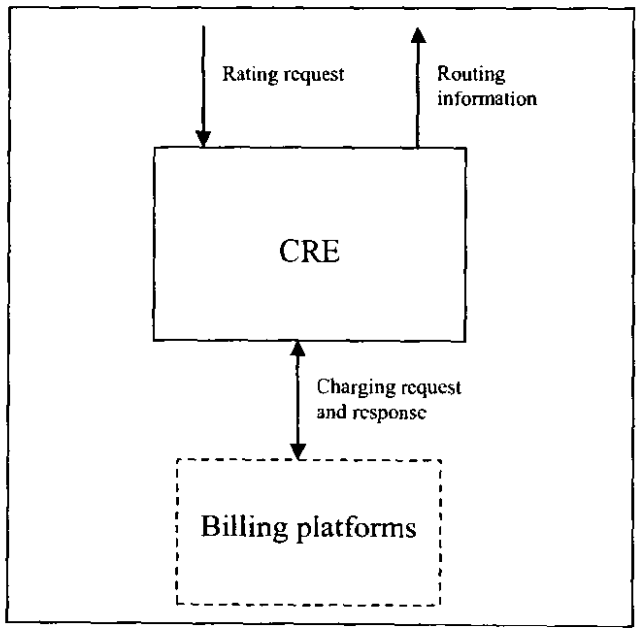


Figure 2-3: Functional diagram of the CRE

2.3 Communications interfaces between peripheral systems

Figure 2-4 gives and overview of the proposed content billing architecture.

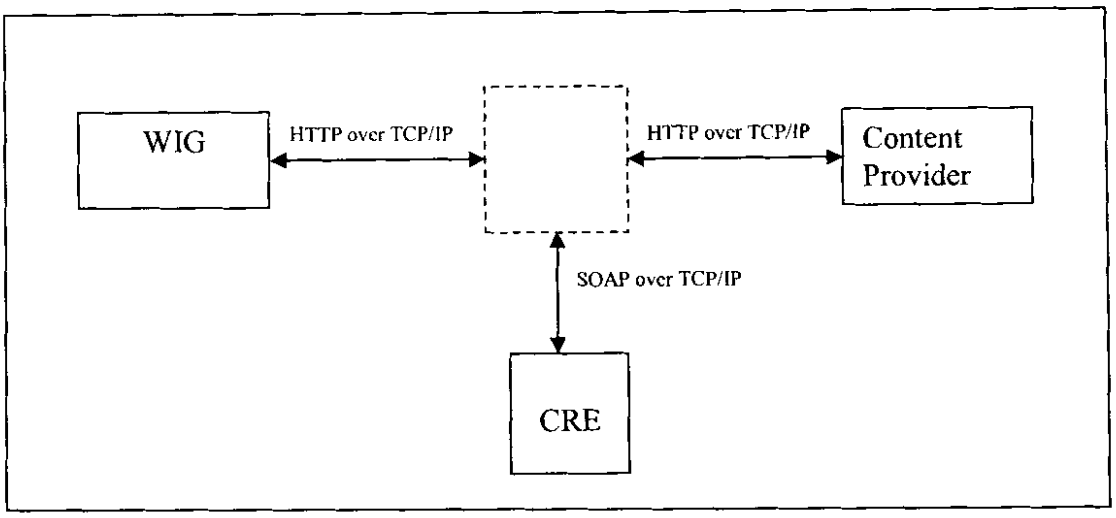


Figure 2-4: Connectivity of the peripherals in the content billing system

The interfaces between the different components have already been determined, and are:

- **Transmission Control Protocol/Internet Protocol (TCP/IP).** The connection between all components in this system will be over the TCP/IP protocol (defined in RFC 793 [36]). This protocol will be transported across both a Local Area Network (LAN) and Wide Area Network (WAN) for access to the Internet. The LAN will exist on the cell phone operator's premises, and is described as "a group of computers and associated devices that share a common communications line or wireless link" [37]. According to Cisco, a major networking equipment company, "a WAN is a data communications network that covers a relatively broad geographic area and that often uses transmission facilities provided by common carriers, such as telephone companies" [38]. In terms of this architecture, this refers to the Internet.

- **HyperText Transfer Protocol (HTTP).** The communication between the WIG platform and any CPs will be in HTTP/1.1 (defined in RFC 2616 [39]). This protocol is defined to fit in the application level of the Open Systems Integration (OSI) model, and is used for distributed hypertext information systems. The protocol does not carry information regarding the state or form of the data, and is used for hypertext transport. It has defined methods for request ("get" and "post"), error codes and header formats. The protocol abstracts the data transferred in distributed systems to such a degree that the systems themselves can be built without consideration for the type and content of the data.

- **Simple Object Access Protocol (SOAP).** The interface with the CRE server is specified to use the SOAP protocol (v.1.2) [40], as defined by the WWW Consortium (W3C) [41]. This is described as a lightweight protocol intended for exchanging structured information in a decentralised, distributed environment. It uses eXtensible Markup Language (XML) technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols.

2.4 Hardware and software requirements for content billing architecture

The choice of hardware was based on the following major factors:

- **Available hardware.** If already purchased hardware can be used, the project will save money.
- **Uniformity of hardware.** If hardware used in the server room is uniform as far as possible, maintenance will be easier.
- **Minimum processing requirements.** The available hardware must be capable of at least the minimum of processing speed.

The technology available for use in this solution includes the Operating System (OS) to be run on the server, the programming language to be used for the core application, and a method to control the application from the OS. The relationship between these software technologies is shown in Figure 2-5.

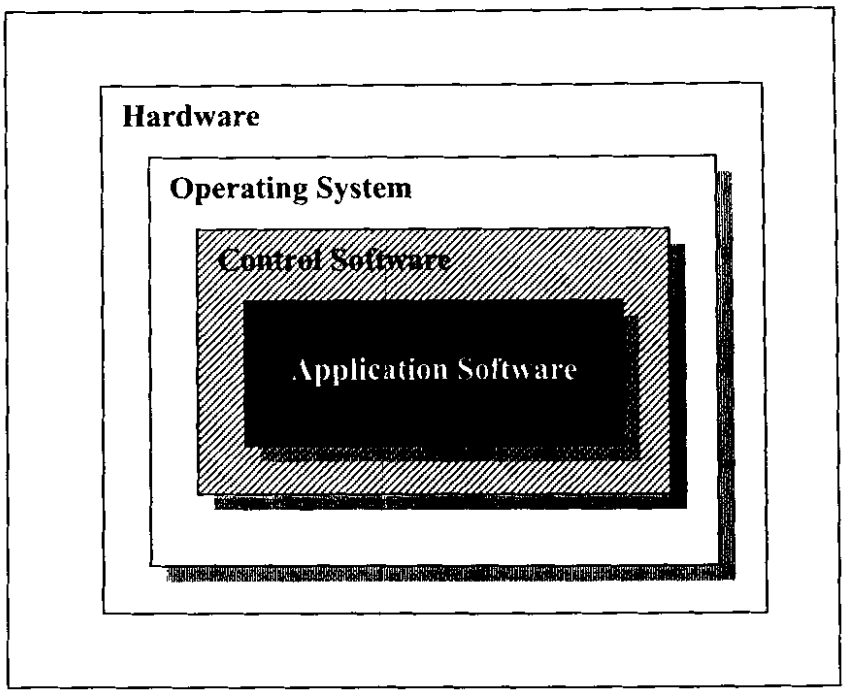


Figure 2-5: The relationship between software technologies on a server

The choices made for these three areas are described in this section.

2.4.1 Operating System – Linux

Linux is a free Unix-type OS originally created by Linus Torvalds with the assistance of developers around the world. Developed under a public license, the source code for Linux is freely available to everyone.

Along with the fact that it is freely distributed, Linux's functionality, adaptability and robustness have made it the main alternative for proprietary Unix and Microsoft OSs. IBM, Hewlett-Packard and other giants of the computing world have embraced Linux and support its ongoing development. More than a decade after its initial release, Linux is being adopted world wide primarily as a server platform [42].

These factors make Linux a good choice for this implementation, as this OS can offer the reliability a server needs at almost no cost.

2.4.2 Application Software – Java

Java is an object-oriented programming language that was developed by James Gosling and colleagues at Sun Microsystems in the early 1990s. Unlike conventional languages that are generally designed to be compiled to native code, Java is compiled to a byte code that is then run by a Java Virtual Machine (JVM).

The language itself borrows much syntax from C and C++ but has a much simpler object model and does away with low-level tools like programmer-manipulated pointers [43].

The speed and ease of development using the Java language, combined with its proven reliability and ability to handle complex business and architecture logic, makes it a good choice for the development environment of this product.

Java is often criticised, especially when compared to the older and more complex language C++, as being slow due to the extra overhead necessary for its execution environment (the JVM). It will be shown that the requirements for the throughput of this server are rather low (section 3.2), and Java is capable of meeting these requirements.

2.4.3 Control Software – Bash scripting

The Linux OS, as indeed all Unix-like OSs, implements a command layer, called a shell. More than just the insulating layer between the OS kernel and the user, it is also a fairly powerful programming language. A shell programme, called a script, is an easy-to-use tool for building applications by connecting system calls, tools, utilities, and compiled binaries. Shell scripts lend themselves exceptionally well to administrative system tasks and other routine repetitive jobs that do not require the bells and whistles of a full-blown, tightly structured programming language [44].

development will be using Bash, an acronym for "Bourne-Again shell" and a pun on Bourne's now classic Bourne shell [45]. Bash has become a de facto standard for scripting on all flavours of Unix and Linux.

2.5 Overview of requirements for new content billing architecture

content billing system needs a free-standing server installed on the operator's running proprietary software. The network must be configured so that this server se to the WIG server and the firewall. This server will run the new CBA. It will all HTTP requests coming from the WIG server.

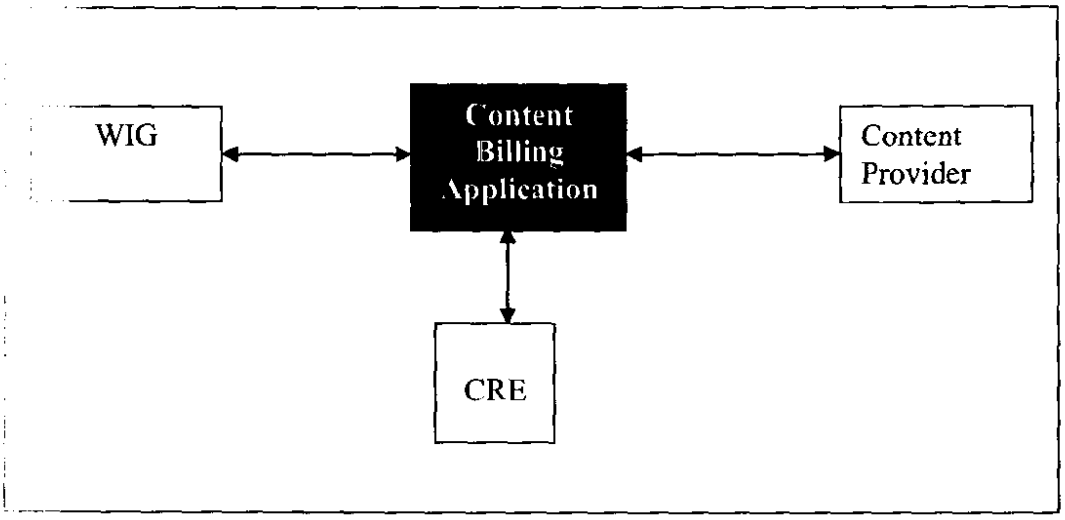


Figure 2-6: Proposed solution and its connections with peripheral systems

These requests must be halted, and information specific to the request must be extracted and passed on to the CRE. The CRE implements business logic to determine whether the request will be allowed, billing the user in real time.

If the CRE approves the request, it sends back a response containing a new URL, which will let the CBA rebuild the request and pass it on to the CP via the firewall and Internet.

It must be possible to change the routing configuration on the operator's systems, removing the dependence on the CP's URL configurations.

The response from the CP is checked for validity by the CBA. If the response is deemed valid (no error message from CP, not timed out on WIG), the response is forwarded to WIG to be delivered to the handset. If these validity conditions fail, another request is sent to the CRE to do a reversal of charging, and an error message is generated by the application to send as a response to WIG and subsequently to the user.

2.5.1 System requirements

The CBA will be running on a free OS, namely Linux. The application must support HTTP, and specifically WML content. Requests must not be passed straight through the application, but must be intercepted by a CRE system for validation and routing before it can be passed on. This must all be done without closing the session on the WIG server.

A summary of the requirements of this system is as follows:

- Absolute real-time billing.
- Absolute real-time charge reversal.
- Content-based charging.
- Billing-based routing.

2.5.2 CRE interface requirements

A normal WIG "get" request consists of a host name and (optionally) a port, followed by a list of parameters specific to the application, followed by the user's phone number.

For use in this system, the host name will define the type of application that is sending the request. This will be followed by parameters describing the choice of CP and the type of action to be performed at that CP.

This will be translated into a proper URL at the CRE, using the correct host name, port, Common Gateway Interface (CGI) service and parameters.

2.5.3 Testing requirements

Based on the basic functional or "system" requirements mentioned in section 2.5.1, the following testing requirements can be defined:

- Current WIG services must be able to retrieve content from their respective CPs, assuming that the CRE validates these transactions.
- Proper denial messages must be sent as a response if the CRE cannot validate the request.
- Error messages must be produced if any response from a CP is invalid (including non-connectivity and time-outs).
- Assuming the CRE functions as specified, it should be able to log billing and reversal of billing due to WIG transactions and failures.

2.6 Conclusion

A new content billing server is proposed as a solution for the inability of existing software to bill for wireless Internet traffic based on content. The peripheral systems for such a solution were defined, as well as the interfaces to be used between all the systems. Hardware and software options were discussed, and based on all of these. The system, interface and testing requirements were defined.

In the next section these requirements will be translated into specifications for the new solution, so that the new content billing architecture can be designed, implemented and tested.

3 Design and implementation of the content billing architecture

3.1 Introduction

Network connectivity is needed between the WIG platform and all web servers that are to act as CPs. The connectivity to these CPs must also be dependent on the ability of the customer to pay for the content. In addition, routing of requests to CPs must be configurable. For this three-pronged goal of connectivity, real-time billing and routing, a CBA server needs to be developed and installed on the intranet. It will connect to the Content Rating Engine (CRE), and all requests and responses will flow through it. This architecture is shown in Figure 3-1.

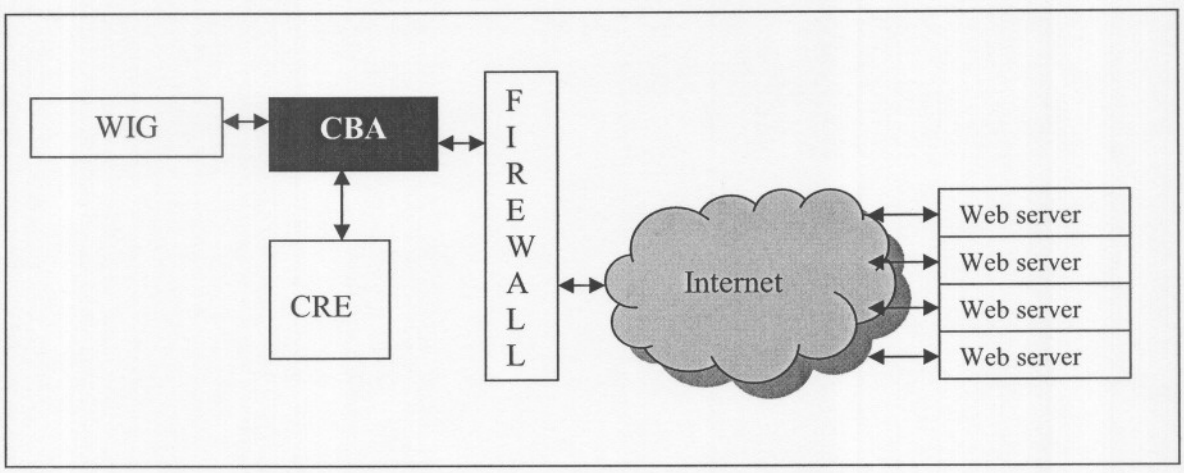


Figure 3-1: The role of the CBA in connectivity

The CBA needs to be able to handle the flow of HTTP requests. It will connect to a CRE server, which will have to give permission for requests to be passed to the CP before this is done, and which will provide the CBA with the URL for the CP, as this is not specified in the request to the CBA. If this permission cannot be granted, a denial message must be returned. If a valid response cannot be returned from the CP, the billing on the CRE must

be reversed, and a replacement message is passed to the WIG system to deliver to the handset.

Certain CPs can be configured in order not to use the CBA server. This is useful in the case of locally hosted web servers (servers within the customer's intranet with free content) where charging and routing services would be unnecessary.

3.2 High-level specifications

This section deals with the specifications for the CBA, focusing on performance related specifications (throughput and latency, or response time), as well as with architectural choices (multi-threading) and operational specifications (logging).

These specifications are:

- **Throughput.** It is assumed that the CBA will not have to handle more than 20 requests per second [46]. The licensing on the WIG server further limits flow of messages to 10 SMs per second, which at worst case (1 SM per request) will limit the throughput of the system to 10 requests per second.
- **Latency.** Delays in delivery of responses are expected to be within reasonable limits, and not detrimental to the user's experience [46].
- **Multi-threading.** To achieve a throughput as high as possible, the CBA will make use of multiprocessing techniques to utilise processing power as efficiently as possible.
- **Logging.** A standard logging class must be used so that the format of the logs is recognisable and external log-analyser tools can be used. A Java class that conforms to these requirements is **Log4J**, and it is thus specified that this class must be used to handle logging.

To calculate the maximum latency (response time) permissible, one needs to look at the mechanism for request and response flow through the WIG system. Requests and responses will flow via SMS, and the response can only be shown as soon as the last SM is received. The platform limits requests to be at least one and at most three SMs long, and responses to be at least one and at most seven SMs long [47]. Thus the total SMs sent and received per request can range from 2 to 10.

With limitations on the network and the handset's processor, SMs cannot be expected to be received much faster than one per second, which means that without the content billing server, delays of 2 to 10 seconds and more may be expected. (In practice, around 15 to 20 seconds were experienced due to other delays on the network; it was thus agreed that further delays of up to 10 seconds at high loads would be acceptable.)

3.3 Interfacing with peripheral systems

All modules of this system will reside on an interconnected TCP/IP network. The WIG, CBA and CRE will reside on the customer's internal LAN, while the CPs can be anywhere on the Internet, or also within the internal LAN. This is shown in Figure 3-2.

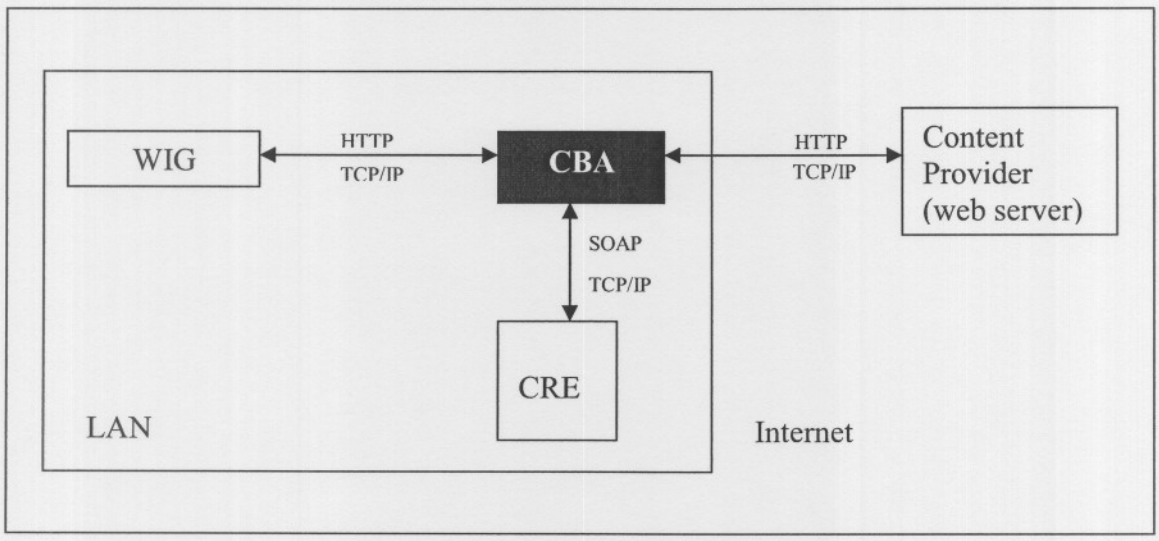


Figure 3-2: Interfaces in the content billing system

The CBA thus communicates with 3 other modules, namely the WIG, the CRE and a CP (in the form of a web server). The following protocols will be used on these connections:

1. HyperText Transfer Protocol (HTTP) between the WIG platform and the CBA.
2. HTTP between the CBA and the CP.
3. SOAP between the CBA and the CRE.

The interfaces used between these modules are:

- **WIG interface.** The CBA will interface with the DP5 WIG server using the transaction-based HTTP protocol. It acts as a web server to the WIG server.
- **CP interface.** The CBA shall interface with the CPs over HTTP. It acts as a client to the CPs.
- **CRE interface.** The Content Rating Engine is developed as a proprietary product, and its details fall outside the scope of this dissertation. It is to be interfaced with by the CBA, and the specific SOAP requests to be used on this interface are described in Table 3-1.

SOAP request	Request attributes	Response attributes	Description
WIG routing request	Transaction ID	Error code	The CBA sends this request to the CRE via SOAP. In the case of an error being returned from the CRE, the modified URL will point to a static WML file containing the error text, hosted on a web server (to be defined). If the request is valid, a modified and configurable URL is returned.
	Event_ID	Modified URL	
	URL	CRETxID	
	Get parameters		
	User's phone number		
Delivery failure update	Transaction ID	Error Code	This request is used to facilitate reversal of transactions already charged.
	Event_ID		

Table 3-1: Requests on the CRE Interface [48]

If the CRE module becomes unavailable, the requests to it will "time out". In this case the CBA will respond to all requests with an error message. It is not possible to just pass the requests on to the CPs, as the CRE is not available to do the necessary routing.

The CBA should function in a way that is transparent to the Delivery Platform and CP applications. CPs should not need to make any modifications to their existing web server software. It is also impossible to change DP5's WIG interface.

3.4 Functional overview of programme logic

A system was thus developed to run on a single piece of hardware, to interface between the WIG server, the CRE server, and any number of CPs on the Internet, via the firewall on the LAN.

This system accepts HTTP requests from the WIG platform, as passed on from the browsers on the customer's handset. These requests are intercepted and parsed, and information of these requests is passed on to the CRE server.

The CRE server responds either with routing information in case permission is granted to provide the information, or with error codes if not. This routing information can be used to rebuild an HTTP request, and this request is sent to the proper CP.

The response from this CP is passed back to the WIG platform, to be forwarded to the customer. If the CRE server returns a denial code or any other error state, or if the CP returns an error, this error must be dealt with by the system to formulate a proper message to be returned to the customer via the WIG platform.

3.5 Sequence of actions

The flow of data through the system is shown in Figure 3-3, and the correlation to the sequence of actions in the system is explained below.

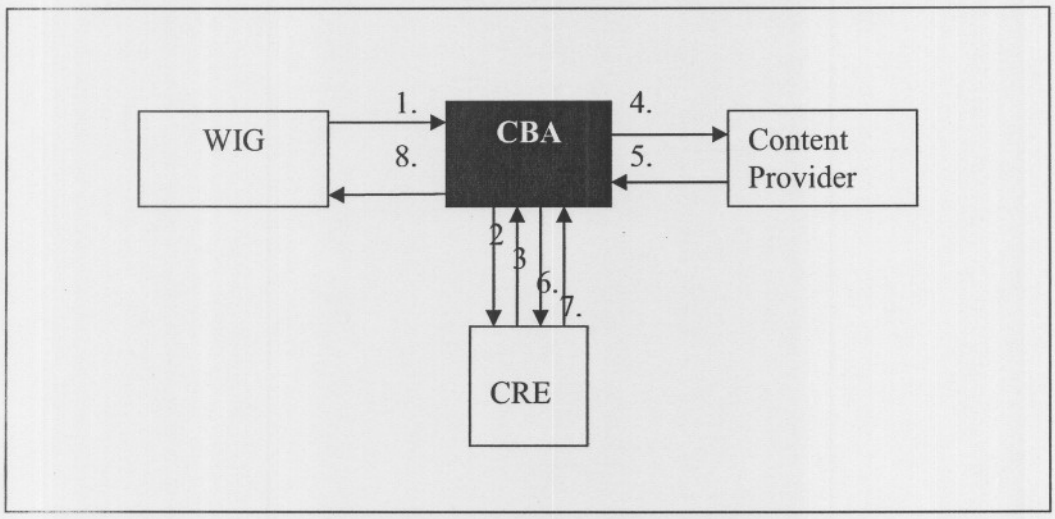


Figure 3-3: Sequence of Actions

The system must perform the following actions (see Appendix A for flow diagrams depicting these steps):

1. HTTP request from WIG.
2. Request is intercepted, parsed and passed via SOAP to CRE for validation. The full URL, including the phone number of the user, is passed to the CRE.
3. CRE responds via SOAP to the application with validation. If the request is not validated a denial response is sent straight back to WIG in WML.
    a. A transaction ID will be associated with the request, indicating the beginning of the external transaction. This transaction ID could be used for multiple purposes (internal audit, statistics, process monitoring). The response from the CRE is received synchronously, thus it is a direct reply to a request.
    b. If the CRE decides not to route the request to the CP, it will return a URL pointing to a static WML page on a web server that contains a fitting error message. This "message server" thus replaces the role of the CP. The error is logged in an error log file in an arbitrary format.
4. The successfully validated request is rebuilt and passed on to the CP specified in the CRE response, in the original HTTP protocol. Thus, the CBA can control routing of the messages.
5. The CP responds with WML data.

6. The response is tested on the CBA for validity. A reversal request is sent to the CRE if the response is deemed invalid, otherwise no request is sent and step 8 follows.
7. CRE reverses the charge incurred in step 3, and acknowledges successful reversal back to the CBA.
8. If step 6 has deemed the response valid, this response is passed on to WIG to be delivered on the handset. Otherwise, a configurable error message is passed on to WIG to be delivered.

3.6 Design and implementation

The functional design of the software is made in terms of the specifications and requirements. This is discussed in broad terms in this section, without going into detail regarding the actual code.

3.6.1 Programme flow

The programme flow is depicted in flow diagrams in Appendix A. This corresponds to the required sequence of actions as described in section 3.5. The process on the upward leg of a request is depicted in the first flow diagram and the downward leg in the second flow diagram. In two cases a time-out error may occur, the process of which is shown in the third flow diagram. The fourth flow diagram shows the process followed in case an HTTP error is experienced on the data returned from the CP.

The development of the CBA commences in the environment specified, incorporating the modular structure of section 3.6.2. Some more details are given on the modules in the paragraphs that follow.

3.6.2 High level design

The functionality of the system may be broken up into modules, namely:

1. WIG Interface Engine.
2. CRE Interface Engine.
3. CP Interface Engine.
4. Logging Module.
5. CBA Engine.
6. Configuration Module.
7. Shell Control Module.

These modules are depicted in Figure 3-4.

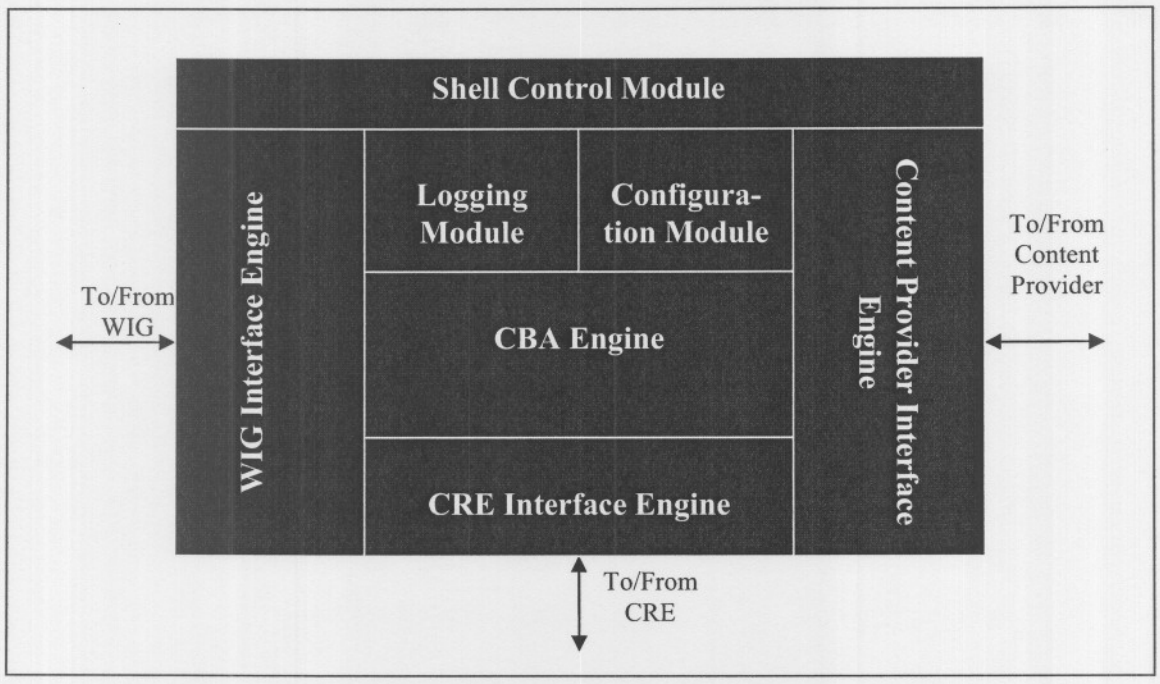


Figure 3-4: Software modules of CBA

These modules may be described as follows:

- **The WIG Interface Engine** is a listener, acting as an HTTP server to the WIG platform. All requests coming in from the WIG server must be intercepted and

parsed to be sent to the CRE interface. Every session created by an incoming request must be kept open so that the response received from the CP Interface can be used as the response to this session, after which the particular session must be closed.

- **The CRE Interface Engine** receives parsed data from the WIG interface, and forms a WIG Routing Request to be sent to the CRE in the format specified in section 3.3. If a connection cannot be made to the CRE or if time-out occurs with the response, a default error message must be returned to the WIG interface to be sent to the user. The response from the CRE is sent to the CP Interface Engine.
- **CP Interface Engine.** If routing information is received from the CRE Interface Engine, it builds an HTTP-formatted request and sends it off to the CP. A successful response from the CP will be passed to the WIG Interface Engine to be delivered to the WIG as a response to the original request. If no connection can be made to the CP, or if time-out occurs with the response, a default error message must be created and passed to the WIG Interface Engine.
- **The Logging Module** is called at start-up and exists as a single instance (singleton) throughout the operation of the system. It interfaces with all other modules to create log files describing the operation of the system. As no requirements were defined with regards to logging, an industry standard logging method is implemented using the Java library **Log4J**. This allows for 5 levels of log messages to be created, based on the nature of the message, namely:

  1. Debug
  2. Information
  3. Warning
  4. Severe
  5. Fatal

  The format and destination of log messages can be configured in the configuration module, based on their level.
- **CBA Engine.** The central core of the CBA engine will control the flow of requests and responses, handle error conditions, and listen to shell control and

configuration parameters. It will also manage the multi-threading of the system, by making sure that singleton classes like the main, logging and configuration classes are created only once, and all other classes are throttled so that a limited amount of threads are created, as set in the configuration module.

- **The configuration module** is also created only once, and will read a flat file containing all the parameters needed to make the operational choices necessary for this system. An example of these parameters appears in Appendix B.
- **Shell Control Module.** This module is not part of the Java application, but is coded in Bash shell scripts. It is responsible for house-keeping tasks like starting the Java application, stopping it in case of failure, and archiving or deleting log files after a pre-defined number of days. To realise these tasks, the following three commands are created on the Linux platform:
  - start
  - stop
  - archive

  The commands will be post-fixed by the `.sh` extension, so that the OS knows to interpret them as shell scripts.

3.7 Functional testing of architecture requirements

The computer industry standard known as the Waterfall SDLC or Software Development Life Cycle [49], and other development practices, specify that during the development cycle every unit that can function independently must be tested as a unit. The methodology and results of this unit testing is given in the next section. An SDLC task flow applicable to this kind of development is shown in Figure 3-5.

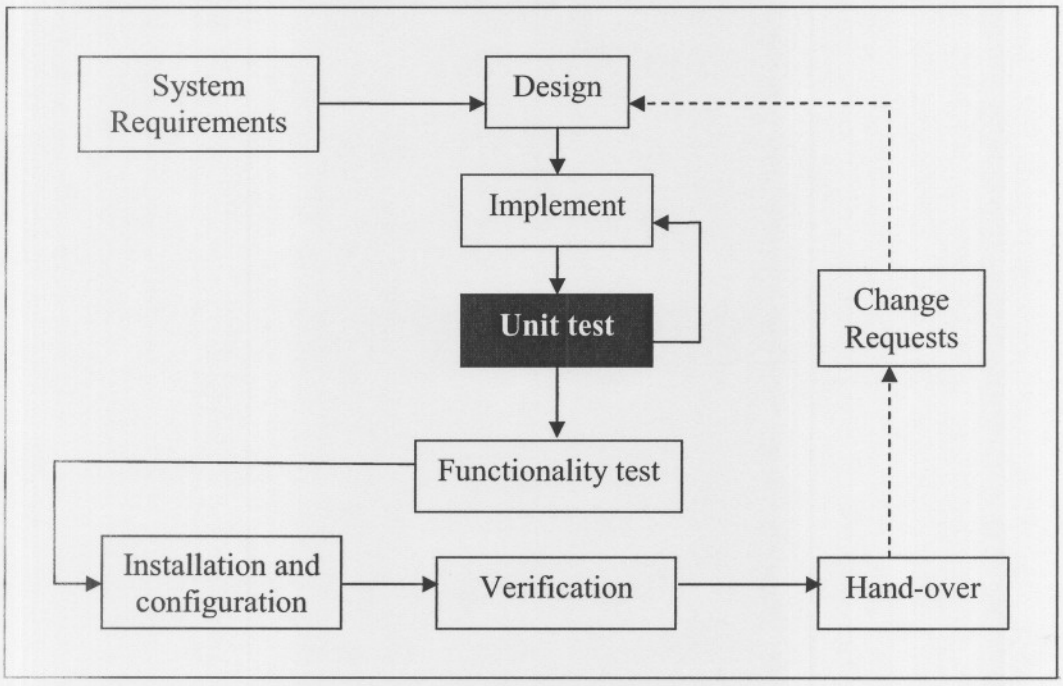


Figure 3-5: Software Development Life Cycle

Sections 3.7.2 and 3.7.3 will prove that the functionality as defined by the requirements is delivered by the product and that the performance constraints are adhered to. Both these tests represent the functionality test in the SDLC.

The verification in the SDLC will be handled in section 1 as a case-study.

3.7.1 Unit testing

Unit testing was done at the completion of every testable unit in the development process. As far as possible these tests relate to the modules defined in section 3.6.2, with some exceptions. This is because these modules represent logical modules, and are not necessarily independent.

1. WIG Interface Engine Test

The upstream side of the WIG platform, which is to connect to the WIG Interface Engine module of the CBA, is specified to comply with the same HTTP standard with which any compliant web browser should comply [50]. This means that the WIG Interface Module can be tested using such a web browser to send an HTTP "get" request, and to display the response received from the CBA. In order to simulate the request coming from the WIG platform, Microsoft Internet Explorer (MSIE) was used, configured to route all requests to a "proxy server" with the details of the CBA.

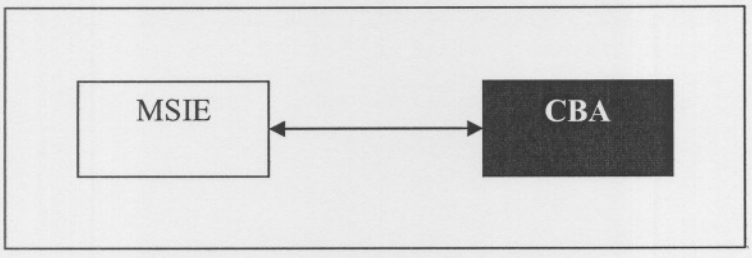


Figure 3-6: Test configuration for WIG Interface test

A "get" request was made to the CBA using MSIE, and a satisfactory response was received and displayed on the browser. The results of this test are shown in Appendix C.2.

2. CRE Interface Engine Test

As no CRE server was available in the development environment, a simulator was created based on the specifications of the CRE [40], and connected as shown in Figure 3-7. This test needs to make a valid SOAP-formatted request from the CBA to the CRE simulator, and the response must be successfully interpreted.

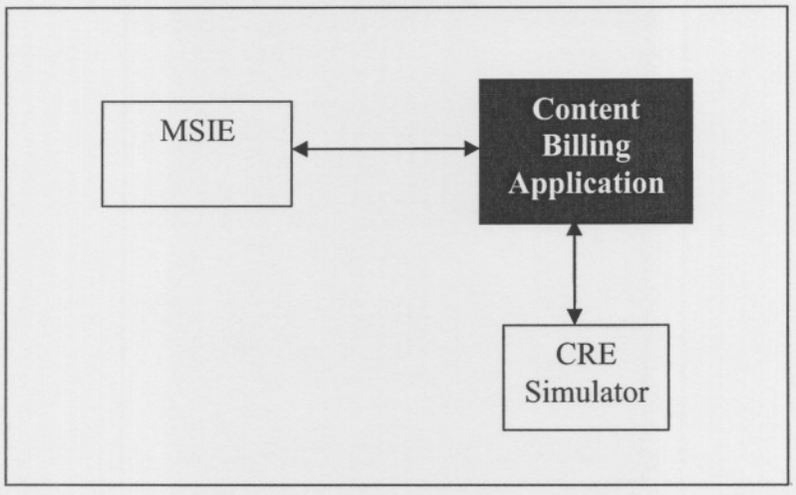


Figure 3-7: Test configuration for CRE Interface test

With the CRE simulator and the CBA running, MSIE was again used to send an HTTP "get" request to the CBA, and the parsed response from the CRE interface was logged and inspected. The results of this test are shown in Appendix C.3.

3. CP Interface Engine Test

The CPs in this system represent any form of web server that might be made available to the system via the Internet, and must thus also comply with the HTTP over TCP/IP standard that the Internet uses. A local web server was thus installed (Apache – a popular free web server) and configured to serve a test WML page on a specific URL. After configuring the CRE simulator to route to this URL given a test HTTP "get" request from the WIG Interface, MSIE was again used to generate this request.

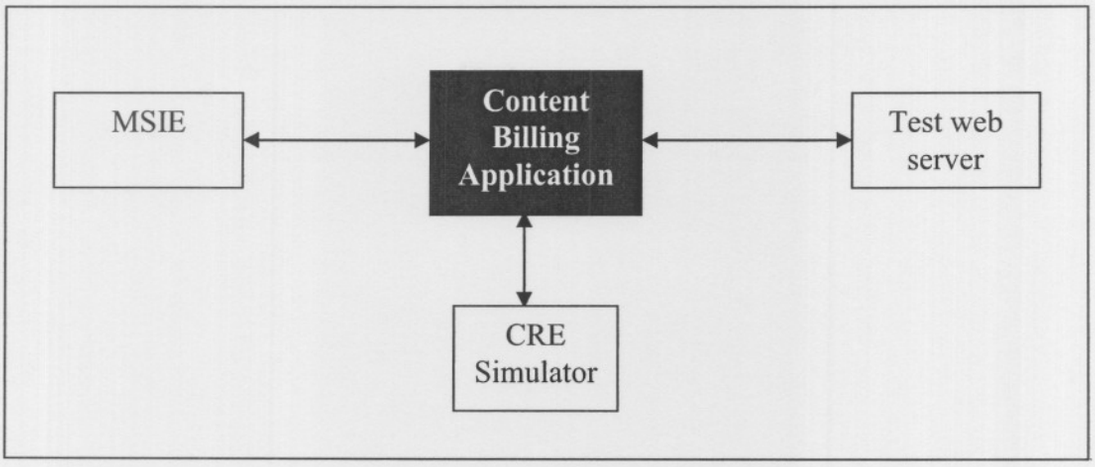


Figure 3-8: Test configuration for CP Interface test

The results of this test are shown in Appendix C.4.

4. Logging Module Test

According to the design (section 3.2), no requirements were made for the format of logging. The logging module was thus successfully tested as a by-product of all the other tests, as any operation of the application was creating log messages in the log files in the format decided upon.

5. Shell Control Module Test

Two of the shell scripts were also used throughout the testing process, as the CBA had to be started and stopped with the `start.sh` and `stop.sh` shell scripts. These performed satisfactorily.

The third shell script, `archive.sh`, was developed to delete files older than the number of days given as an input parameter. Thus after two days of testing and subsequently creating log files, a running of the following command would delete the log files more than one day old, which proves the operation of this script:

```
> archive.sh 1
```

3.7.2 Functionality testing

In order to test the functionality of the system against the requirements as well as the design on which the decision fell, a series of tests was devised and performed. As this testing was part of the implementation stage of the SDLC, the environment was not available for these tests, and simulators were incorporated to simulate different conditions in the environment.

More details on these tests appear in Appendix C, but the tests are described below.

1. URL and Error code retrieval

The first functionality to be tested is part of the interface with the Content Rating Engine, and covers the CBA's ability to extract the new URL that the CRE might return given a routable (chargeable) request, or the error message it might return in the event of a request that cannot be routed.

This test is done in two steps:

1. A test URL is configured in the CRE simulator to be returned as a response to any request. One may confirm that this URL is extracted by looking at the log messages.
2. The CRE simulator is then configured to return no URL, but rather an error code, to any request. This error code must be shown in the log messages, and a new URL must be generated by the CBA, to serve a static error page hosted on the CBA itself.

Appendix C.5 shows that these two tests gave the expected results.

2. Client request to new URL

In order to test the routing functionality of the CBA and Content Rating Engine combination, it was necessary to configure the CRE Simulator to return a certain URL in case a test "get" request is received by the CBA and passed on to the CRE.

Both the "get" request URL and the new URL must be valid in the test environment, and so the URLs are chosen as follows:

- "Get" request URL: `http://localhost/cba/cp`
- New URL: `http://localhost:8080/cba/cp`

In order to test a URL on port 8080, another web server is installed, namely Jakarta Tomcat. This server was developed by the Apache Group, for serving Java J2EE pages, but it is also capable of displaying static WML pages for these purposes.

Even though these URLs are not very different, changing a single character would have meant that a URL transformation has taken place, and routing is being implemented.

By examining the log messages and noting that the response received on the web browser comes from the resource at the new (or rerouted) URL, this test proves successful. See Appendix C.6 for the CBA's log entries.

3. Time-out to host

Using the Jakarta Tomcat server as a CP simulator, it is possible to set a delay into its turn-around time greater than the CP time-out setting configured in the CBA. In this case the Tomcat server receives a request from the CBA, and waits a while before responding, thus simulating a slow Internet connection or server latency.

As soon as the CBA waits longer than the configured time-out value, it will close the session and respond with an error message. The response on the web browser and the log entries shown in Appendix C.7.1 indicates that the configured error message has been returned.

4. Time-out to CRE

In the case of internal network problems or operational problems on the CRE, which may cause an inability to do charging or routing in a timely fashion, routing should be considered as impossible, and no content can be delivered by the CBA.

This is tested by introducing a delay in the response-time of the CRE simulator. If this delay is longer than the CRE time-out configured on the CBA, the CBA should close the session and respond with an internally generated error message. This error message is also configurable.

The response on the web browser and the log messages of this test (shown in Appendix C.7.2) indicates that this error message has been returned.

5. GET error

The CBA must cater for the eventuality of the HTTP "get" request received from the WIG server being malformed or illegal. This is tested here, by making an illegal request in the web browser, and sending that request to the CBA. The CBA will recognise the request to be illegal, and will deflect the request to a static error page, which is returned to the web browser as response.

The error page was indeed displayed on the web browser. The log messages of this test can be found in Appendix C.7.3.

6. HTTP error

The response from the CP to a request from the CBA must also be distrusted, and the CBA must handle responses with error statuses in a user-friendly way. This is done by replacing the offending response with a configurable error message, and passing that to the user.

This is tested by routing to a non-existent page on the test web server, and making a request that will be routed to that page. The test web server will respond with an error page and a status code of 404 (the HTTP-specified error code for a "Page not found" condition). The CBA parses the response for any status codes other than 200 (HTTP-specified status code for "OK"), and recognises an illegal response. An extract of the HTTP Status Codes from the HTTP Specification RFC2616 [51] is shown in Appendix C.1. This response is intercepted by the CBA, and replaced with a configurable, user-friendly error message.

The user-friendly error message was indeed displayed on the web browser. The results for this test are shown in Appendix C.7.4.

7. Reversal

A further, and very important, functionality that is related to the previous test of errors conditions is that of Charge Reversal. In the case of a request that has already been routed and charged for on the CRE, that fails due to time-out to the CP or any HTTP error, the CBA must send a second request to the CRE to reverse the charges incurred in the previous request.

This may be tested with either of the previous two tests. The entries related to the DeliveryFailureUpdate and DeliveryFailureResponse SOAP messages sent to the CRE can be seen in the logs in Appendix C.8, which proves the success of this test.

3.7.3 Performance testing

The expected throughput of the system has previously been defined as 10 HTTP requests per second, and also a further requirement that the system should not produce delays that negatively affect the user experience (section 3.2).

With the Content Billing Engine functioning on its own, connected only to the CRE simulator and test web server, a performance high above the specifications for the full system is expected. Two tests will be done, the first to measure the response time (latency) of a single request, and the second to measure the maximum throughput of the CBA itself.

In order to make multiple requests and measure response times, the MSIE web browser used as a WIG simulator must be replaced by an HTTP "get" request simulator. This has been developed in Java. It can be configured to make any number of identical requests, separated by a configurable delay, and record the details and delay of each response.

Latency test

With a test configuration as shown in Figure 3-9, one makes a single request and measures the delay before a response is received by the "get" request simulator. This delay will be the result of delays in the CBA, CRE simulator and the test web server, and is thus a pessimistic approximation of the delay caused by the CBA itself.

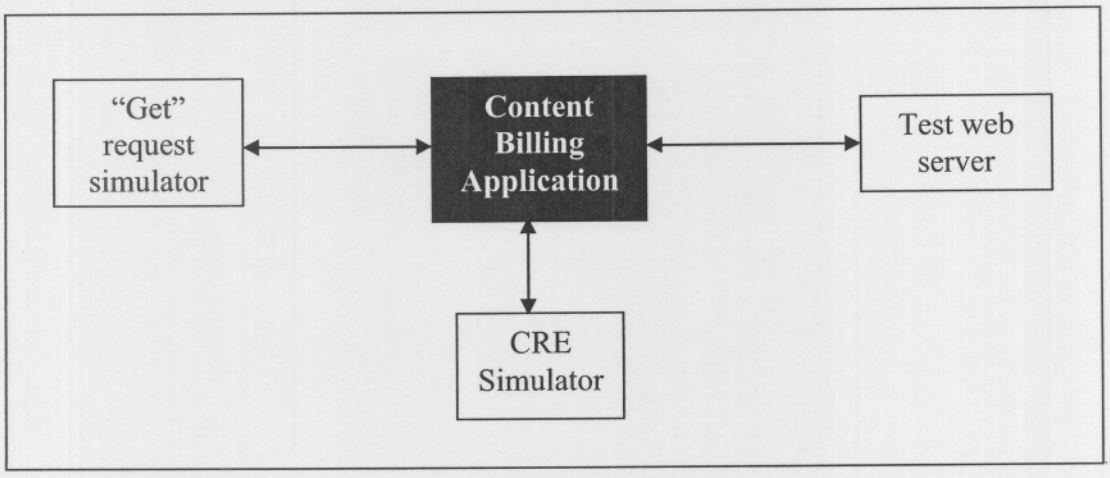


Figure 3-9: Test configuration for performance tests

Running this test 10 times and averaging the delays produced the following figure:

$$\Delta t_{avg} = 121ms$$

Based on the slightly subjective requirement of "no negative effect on user experience" (section 3.2), a value of roughly an eighth of a second seems reasonable.

For details of these 10 tests refer to Appendix C.9.

Throughput test

To measure the maximum throughput which the CBA can possibly handle, the configurable value of "maximum number of simultaneous threads" was raised to a very high number, like 99999, and multiple requests were created with the "get" request simulator described in the previous test, with no delays in between. The number entered for requests to be generated is 1001, which may at best prove that more than a thousand requests can be received in short succession – two orders of magnitude greater than the specification of 10 sessions per second (refer to section 3.2).

The results of this test can be seen in Appendix C.10, and the following throughput has been calculated for this test:

Throughput = 184 sessions/second

This is in fact one order of magnitude greater than the specification (section 3.2) and is thus accepted.

3.8 Conclusion

In this section a new CBA was designed, implemented and tested. Specifications were drawn up based on the requirements of the system and operating environment, as well as best practices in the industry. Interface requirements were defined so that the CBA would be able to connect and communicate with its neighbours on the cell phone operator's network. The functionality and behaviour of the system was defined, and the modular architecture was designed. Thereafter, all these specifications, requirements and functionalities were tested, and the values returned showed that the Application achieved all these goals.

The next section will describe the installation and verification of the CBA on a cell phone operator's network as a case study.

4 Application and testing of new content billing architecture

4.1 Introduction

The proposed CBA was completely implemented, and all the necessary testing done to prove that the specified functionality and performance had been met.

This next step is the installation, configuration and verification of the CBA on an operator's network. Verification includes the performance of the content delivery system as a whole. The position of the CBA in the system is shown in Figure 4-1.

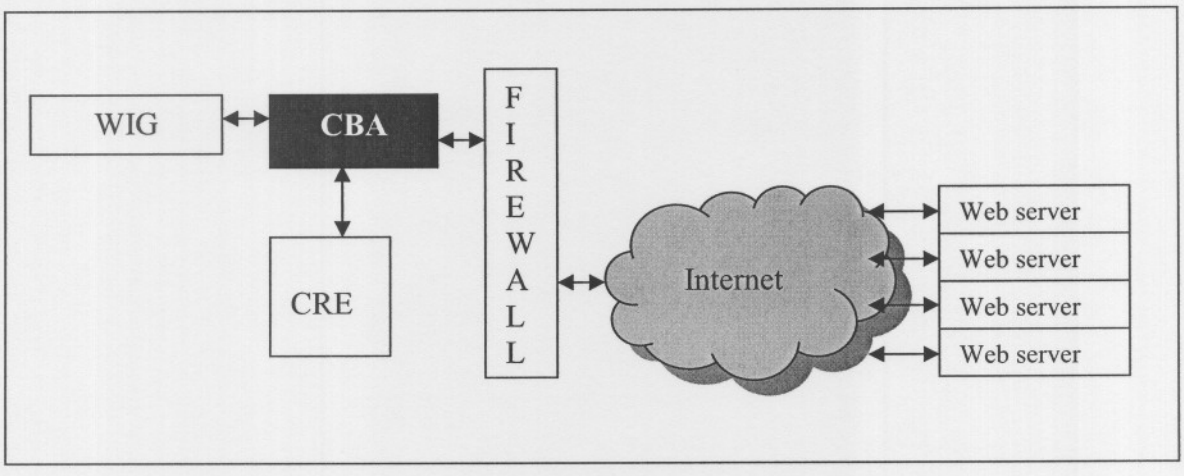


Figure 4-1: Position of CBA in Content Delivery System

The installation, configuration and verification of this solution will be done as a case study.

4.2 Application and testing environment

The installation of the CBA commences in the operator's live environment. This does not impact service delivery of the operator, as no content is live at the start of the installation, and the launch of live content is only to occur after full installation and verification have been finalised.

4.2.1 Resources and prerequisites

The server hardware used for the testing of the CBA is [52]:

- Dell PowerEdge 6600

It has the following specifications: [53]

- Up to 4 Intel® Xeon® processors with an internal operating frequency of at least 1.40 GHz
- Up to 16GB RAM
- Up to 3.6TB of data storage

This is a rack-mounted server in a server room, already installed by technical staff. It has a keyboard, trackball and monitor available, as well as a CD-ROM drive to be used for OS installation.

Access has to be given to this machine in order to reach the following servers:

1. Domain Name Service (DNS) server, for resolution of server names
2. CRE
3. CPs via firewall

Actions that had to be taken before commencing with installation:

1. Hardware described in Appendix B.1 connected to network and powered on.
2. Network configured as required, and described in Appendix B.2.
3. Linux Red Hat 7.3 installed and configured. See Appendix B.3 and B.4.
4. Java Runtime Environment (JRE) 1.4 installed and configured, as in Appendix B.5.

4.2.2 CBA installation and configuration

The CBA has been compiled from Java source code and packaged in a Tape Archive or TAR file. This file is simply copied into the root directory of the user created on the server for the CBA, and extracted (see Appendix B.6). This creates the folder structure shown in this appendix.

The configuration file, `cba.cfg`, is configured based on the specific details of the environment. The example in Appendix B.6.2 shows the values used for this installation. It can be noted that the value for the maximum number of simultaneous threads, max_threads, is set to 10 000. This is the highest number we can use without causing memory problems on this particular configuration of hardware, and is considered to be the optimum value for this installation. If the hardware configuration is changed, specifically with regards to the amount of Random Access Memory (RAM), this value will have to be reviewed.

The application is started using the shell script, `start.sh`. To verify that it is running, the output of the log files in the logs directory is viewed and interpreted.

4.3 Testing and verification procedure

Verification is done on-site, with the CBA connected to all the peripheral servers in the system, and configured as specified above. First, functionality is verified in the form of an Acceptance Test Procedure (ATP, Appendix D), which has been accepted and agreed to by the operator. The second step tests the performance of the system in the form of end-to-end tests.

4.3.1 Testing requirements

The ATP is drawn up based directly on the specifications of the system, as follows:

1. A **successful session** consists of:
    a. A valid HTTP "get" request received from the WIG server.
    b. A valid SOAP request sent to the CRE server.
    c. A valid and timely response received from the CRE server.
    d. A valid request to a CP made successfully.
    e. A valid and timely response received from the CP.
    f. A valid response sent back to the WIG server for delivery to the user.
2. A **WIG session time-out** occurs when the WIG server sends a request through, but does not receive any response within its configured time-out period. An error message will be created by the WIG server and delivered to the user.
3. The **CBA offline** condition occurs if the CBA server is offline, and thus the WIG server cannot successfully connect to the CBA to deliver a request. An error message will be created by the WIG server and delivered to the user.
4. The **CRE offline** condition occurs after this sequence of events:
    a. A valid request is received from the WIG server.
    b. A valid SOAP request is sent to the CRE server.
    c. The CBA receives no response from the CRE before its configured time-out period expires.

The CBA should generate a configurable error message and deliver it to the WIG server to be delivered to the user.

5. A **CP offline** condition occurs after this sequence of events:
    a. A valid HTTP "get" request received from the WIG server.
    b. A valid SOAP request sent to the CRE server.
    c. A valid and timely response received from the CRE server.
    d. The CP specified in the response from the CRE server cannot be reached or connected to.

    The CBA should generate a configurable error message and deliver it to the WIG server to be delivered to the user.

6. A **charge reversal** occurs if:
    a. a CP offline condition exists, or
    b. the CBA experiences a time-out while waiting for a response from the CP, or
    c. the CP returns a response to the CBA with an HTTP status other than 200 (which indicates a status of "OK", see Appendix C.1 Table C-1).

    The CBA should send a valid charge reversal request to the CRE, and a valid response must be received. The CBA must then generate a configurable message and deliver it to the WIG server to be delivered to the user. The user's account balance should be restored to the same value as before the request was made.

7. **Free content** is any request that is configured on the WIG server to bypass the whole content billing system, and go straight to a CP (which will usually be internal to the operator's network). Thus, a request must be made by the user to such free content, and the response must be received without any session created on the CBA or CRE. The user's account balance should be unaffected.

4.3.2 Functionality testing

Functionality testing was done at the operator's premises in the form of an Acceptance Test Protocol. Test Cases that were covered by this ATP are shown in Table 4-1, and are directly based on the testing requirements in section 4.3.1.

Nr	Test case
1.	Successful session
2.	WIG session time-out
3.	CBA offline
4.	CRE offline
5.	CP offline
6.	Charge reversal
7.	Free content

Table 4-1: Test cases

Before these tests could be performed, the following preconditions were needed to be true:

1. WIG Services software is downloaded on the handset.
2. WIG, CBA, CRE, Internet Connection and CPs are in full operational state.
3. SIM has sufficient funds.
4. Test SIM has been provisioned on the test WIG platform.
5. A network connection exists between WIG and application, application and CRE as well as application and test CP.

The following steps had to be taken during the execution of the ATP, and are indicated in the execution steps of the ATP (Appendix D.2).

1. Note the configured time-out of the Session on WIG.
2. Disconnect CBA.
3. Connect CBA.
4. Disconnect CRE.
5. Connect CRE.
6. Disconnect CP.
7. Note the subscriber's account balance.

The ATP test was successfully executed in the presence of the operator, and a summary of the results appear in Table 4-2.

Test case	Results	Pass
Successful session	Handset displayed response. Log files confirmed that Session ID was closed.	✓
WIG session time-out	After about 40 seconds a response was received indicating that the CP was not available.	✓
CBA offline	After about 2 minutes a response was received on the handset indicating that the CP was not available. This error message was different from the previous one, as it was configured on the WIG server.	✓
CRE offline	A response was received on the handset almost immediately, indicating that there was an internal problem. No functionality was possible without the CRE server.	✓
CP offline	After about 20 seconds a response was received indicating that the CP was not available.	✓
Charge reversal	The same response as in test case 5 was received at the handset, and the log files indicated that the charge was reversed.	✓
Free content	The response from the free content server was displayed on the handset. The log files indicated no interaction with the CRE server, so no charge could have been incurred.	✓

Table 4-2: Summary of the results of the ATP

4.3.3 End-to-end performance tests

In order to test the performance of the full system, only the servers upstream from the WIG server could be used, as the WIG server had no means of generating requests with a chosen frequency. The "get" request simulator developed for the performance tests

described in section 3.7.3 had been used for this purpose. The test environment that was used, is depicted in Figure 4-2.

Since the requirements specified that no more than 10 requests per second needed to be handled by the system (section 3.2), tests were done with different rates of requests generated by the "get" request simulator up to 12 requests per second (120% of the specification).

Detailed results of these tests appear in Appendix E.

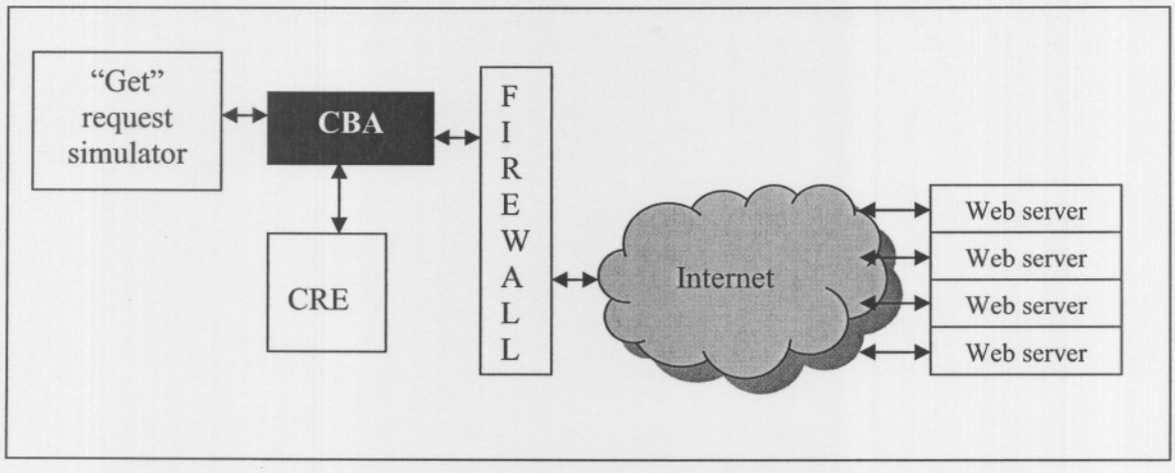


Figure 4-2: Environment used for end-to-end tests

4.4 Test results and analysis

A set of 24 tests was done, each incrementing the rate of requests with 0.5 sessions per second, and each test consisting of 100 requests made to the CBA by the WIG simulator. The CBA was configured for optimal operation, with the value for the maximum number of simultaneous threads set to 1000 (see section 4.2.2).

The following values were recorded or calculated:

1. Number of requests.
2. Delay between requests.
3. Rate of requests.
4. Average response time.
5. Average rate of responses.

These results are shown in Appendix A. A graph of the average rate of responses against the rate of requests is shown in Figure 4-3. A straight line at an angle of one response per one request shows the ideal throughput, flattening out at 10 sessions per second, as that is the specified maximum throughput.

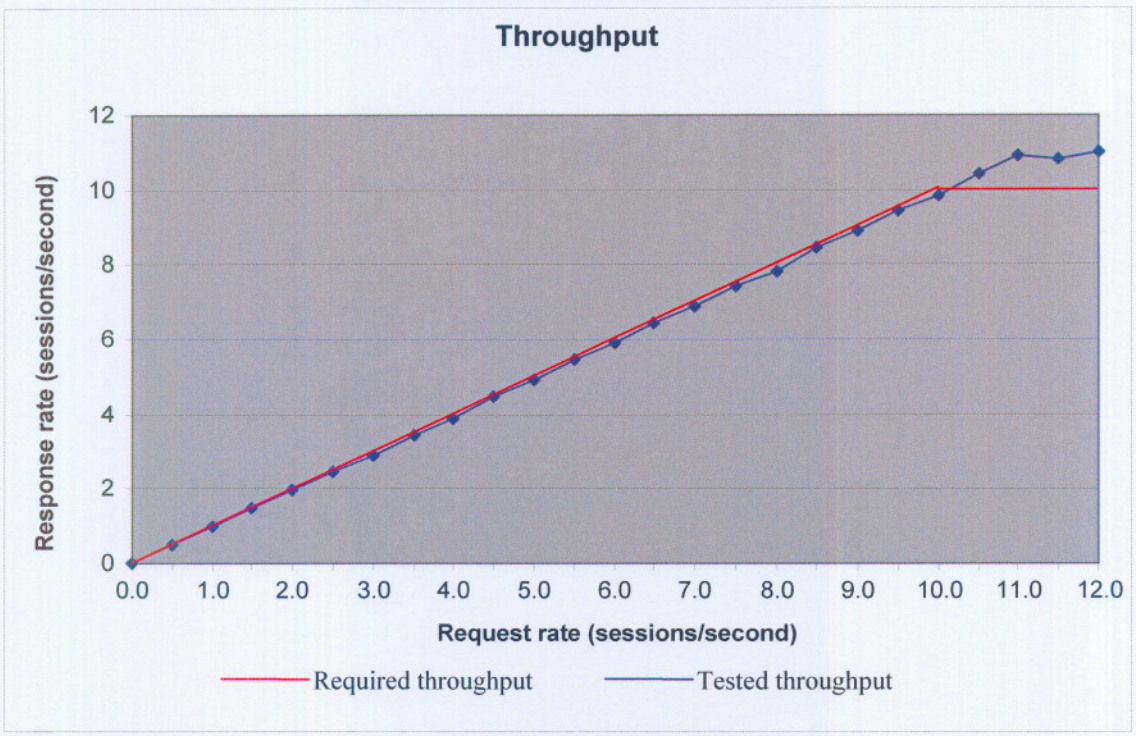


Figure 4-3: Throughput

This proves that the specified throughput of 10 sessions per second has been reached. The reason for the slight deviations in the response rate is due to the latency in the system,

which causes the total number of responses in a given time to be slightly less than the number of requests. Over a long period this will average out, and the response rate should be exactly equal to the request rate.

Figure 4-4 is a graph that shows the average response time, or latency, against the rate of requests. The darker area indicates the acceptable values of latency at the specified request rates.

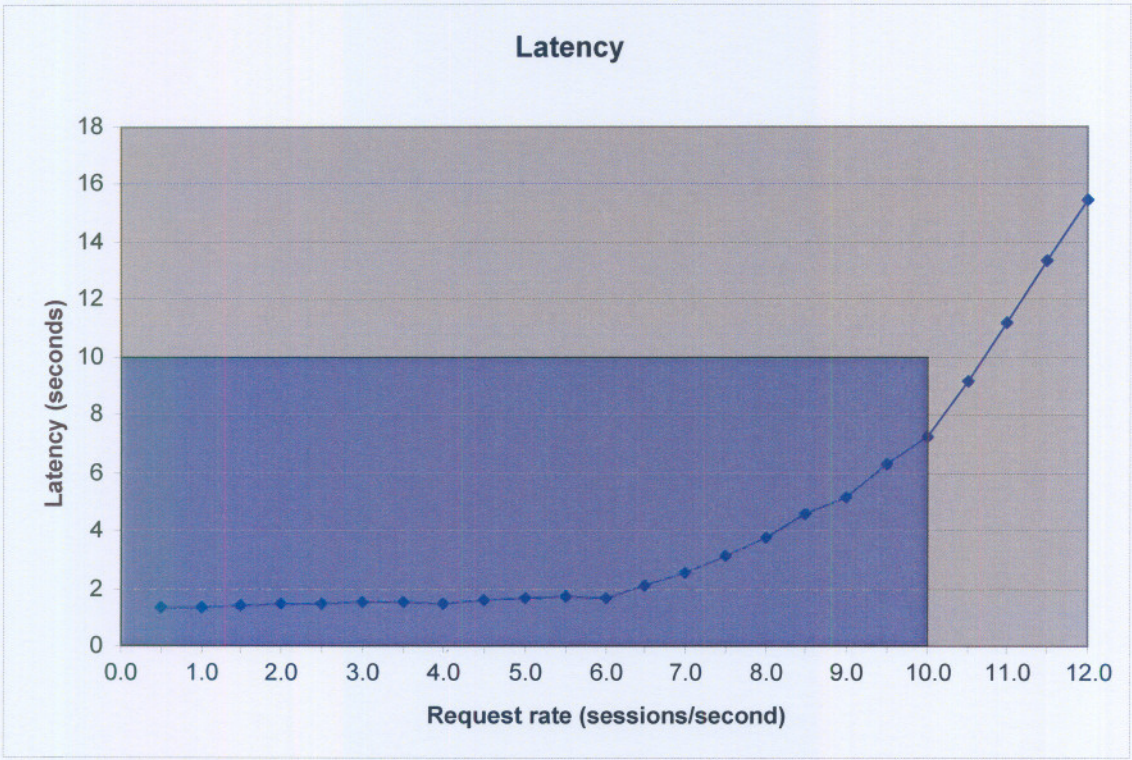


Figure 4-4: Latency

This shows that the latency of the system gradually increases with the increased request rate, due to the necessity of the software to spawn more simultaneous threads that have a certain overhead. This could be minimised by incorporating a thread pool in the code. While the performance of the system was adequate, this proved not to be necessary.

These specified values of latency are lower than the values obtained in the tests done for latency and throughput on the CBA alone, using simulators for the peripheral servers (section 3.7). This shows that the peripheral systems are responsible for most of the latency and throughput limitations, and that the CBA is not the first bottleneck with regards to performance. If the needs of the system change, the performance of the CRE and possibly the connection to the CP should first be improved.

4.5 Example of the functionality of the new content billing architecture

In order to illustrate the functionality of the new content billing architecture, the system will be viewed from a user's perspective. This will give a clear indication of the user experience created by this solution.

In this hypothetical test, the user has a 2G cell phone with a WIG dating service application installed for requesting Internet hosted content. The SIM card is on a prepaid contract with the cell phone operator, and has enough funds for requesting the service once only. For the sake of this test, assume the service costs R10, and the user has R15 in the prepaid account at the start of this demonstration. Table 4-3 shows a representation of the cell phone's display and the current state of the user's prepaid account.

Action	Cell phone display	Account representation
Choose age, gender and location of potentials.		Balance 15.00
Send request. The charge for the service is debited from the account.		Dating -10.00 Balance 5.00
Problem at CP's side. Error response received. Charge is reversed by crediting with the same amount.		Dating -10.00 Reversal 10.00 Balance 15.00
Send request again at a later time, when the issue at the CP has been resolved. Response received and charged for.		Dating -10.00 Reversal 10.00 Dating -10.00 Balance 5.00
Send request again. User does not have enough funds for the service this time. Insufficient funds response received, and no charge is incurred.		Dating -10.00 Reversal 10.00 Dating -10.00 Balance 5.00

Table 4-3: Visual representation of the content billing functionality.

It is clear that the actual charging and reversal of erroneous charges occur in real time, and that routing decisions are based on the current balance in the user's account. The application's routing functionality is more transparent and thus more difficult to show visually.

4.6 Conclusion

With the design and implementation of the CBA completed in section 1, this section focused on implementing the new solution. This was done as a case study, by installing the CBA into a content delivery system at a cell phone operator.

With the system configured, end-to-end tests were performed to ensure that the functionality of the system matched the specifications. It was found that absolute real-time content billing had been achieved, and that routing of requests to CPs was possible.

The performance of the complete system was also tested against the requirements, and although the results were a little down from when the CBA was tested on its own, the complete system still performed within the required limits.

5 Summary and recommendations

5.1 Conclusion

This dissertation considers the convergence of the Internet and cell phone industries and its implications in terms of the variety of evolving services and the rising need for absolute real-time billing. It also points out that the market in South Africa is almost 1% of the global GSM market, and that only 1% of the South African market has 3G-capable equipment. In addition, 85% of South Africa's cell phone users prefer prepaid contracts with their cell phone operators. It states that the need for real-time content billing directly affects the revenue creation of cell phone operators. These important facts underline the need for an effective real-time billing solution that caters for 2G Internet technologies and, at the same time, ensures a friendlier user environment for prepaid users.

The SmartTrust DP5 WIG server provides the technology for generating Internet traffic to and from 2G mobile equipment (cell phones) via the responsible cell phone operator's network. The WIG platform transforms SMS data into WML pages that are sent using the HTTP protocol over TCP/IP, which is compatible with the Internet. An external server is also used to return a new URL for requests, and to connect to the charging systems, called the CRE.

The real-time billing solution is, therefore, designed to connect to the WIG server, the CRE server and any number of Internet CPs. The design defines and incorporates proper interface specifications and requirements for functionality and sequence of actions. Design choices such as hardware, OS and software architecture and technologies, are explained, as well as testing requirements and methodology. The new solution has been dubbed the CBA.

The complete design, coding and installation of the CBA system is tested against the requirements and specifications. Load tests are done to measure latency and throughput, and the results are compared with the requirements.

The new system is capable of receiving a request for content from a user via the handset, the GSM network and the WIG platform, parsing this "get" request and determining the proper charge for the requested data or service. Such costs are deducted in real time from the user's account. The request is then redirected to the proper URL (which is configurable in case the CP's configuration needs to change), and the required content is returned. Should the content fail to be delivered to the user due to any "error" condition, the charge is successfully reversed. Thus, the balance on the user's account returns to the same value as before the request was made, in real time.

The required throughput rate of the system is set to a maximum of 10 sessions per second. The actual maximum throughput rate is about 12 sessions per second, but delays due to latency may result in a fall in the response rate versus the request rate.

The latency level of the system is required to be "not detrimental to user experience". As the system currently needs to send one to three SMs per request, and receives one to seven SMs per response, an additional delay of less than 10 seconds is acceptable. Performance tests show that the latency stays under 2 seconds for server loads of up to 6.5 sessions per second. At the maximum throughput of 10 sessions per second, the latency measures approximately 7 seconds, and climbs steeply. The approximate maximum of 10 seconds latency is reached at a throughput of about 11 sessions per second.

This design, successfully installed in the network of a South African cell phone operator as a proper solution to their requirements for a CBA, has been in use for about two years now. Although business information is not available, WIG content is still being provided and billed based on the requested content.

5.2 Recommendations for future development

Although the performance of the solution satisfies the current requirements of the operator, an increase in cell phone Internet traffic may consequently demand increased

performance. The expensive way to do this is by increasing the specifications of the hardware. The cheaper and more effective option is to optimise the performance of the software and reduce unnecessary overhead. One way of doing this is by incorporating thread-pooling in the software, so that new threads are not created, but threads are rather kept in a pool and used as required.

As the capabilities of the cell phone technology grow, more and more online functionality will become available to the cell phone user. Some cell phone Internet systems already provide a much richer user experience than the text-only Internet pages used by this system. The purpose and target market for this system have been explained (prepaid users with second generation handsets), but the third generation cell phone Internet systems may also need content billing systems, possibly of a much higher complexity than the system described here.

Therefore, a more feature-rich system may be designed, which is not so dependent on interface requirements of a specific type of cell phone Internet platform, but which should rather be able to interface with a wide variety of technologies, delivering and charging for a plethora of multimedia content. More cell phone oriented content, which is presently not available to Internet users – such as ring-tones and cell phone wallpapers – may also be handled by this system.

The convergence of the cell phone technology and the Internet, therefore, requires a convergence of solutions for the different Internet and cell phone systems, as well as all other data services that cell phone operators might offer. These convergent systems will mean one content billing system that handles every aspect of rating and charging on all content requested from and delivered to cell phone handsets.

References

[1] McKay, Niall, "The Portable Cell Phone Started Here", 15 July 2003, http://www.thefeaturearchives.com/topic/Announcements/Martin_Cooper__The_Portable_Cell_Phone_Started_Here.html, Accessed: 21 February 2007

[2] Farley, Tom, "Privateline.com: Mobile Telephone History", Telektronikk, v3, April, 2005, p27. Also online: http://www.privateline.com/PCS/history9.htm, Accessed: 21 February 2007

[3] Wikipedia Online Encyclopedia, "History of mobile phones", http://en.wikipedia.org/wiki/History_of_mobile_phones, Accessed: 21 February 2007

[4] UMTSWorld, "UMTS / 3G History and Future Milestones", http://www.umtsworld.com/umts/history.htm, Accessed: 21 February 2007

[5] Anon, "Cell Phone Systems – overview and history of cell phone from the first analogue systems through to the latest 3G systems", http://www.radio-electronics.com/info/cellulartelecomms/celldev/cellulardev.php, Accessed: 21 February 2007

[6] Bing, Benny, "Broadband Wireless Access", TechOnline, 29 May 2001, http://www.techonline.com/community/ed_resource/feature_article/14365, Accessed: 21 February 2007

[7] Anon, "Cell Phone Systems – overview and history of cell phone from the first analogue systems through to the latest 3G systems", http://www.radio-electronics.com/info/cellulartelecomms/celldev/cellulardev.php, Accessed: 21 February 2007

[8] Bridges Inc, "South Africa telecommunications overview, commentary, and statistics (policy brief)", 2 May 2001, http://www.bridges.org/publications/123, Accessed: 21 February 2007

[9] Mwanza, C, "Die geneuk met intekenaartalle", Fin Week, Media24 Ltd., 15 June 2006, p17

[10] Anon, "Statistics of cellular in South Africa", http://www.cellular.co.za/stats/statistics_south_africa.htm, Accessed: 21 February 2007

[11] MyADSL, "South African broadband users pass the quarter million mark", 3 July 2006, http://www.mybroadband.co.za/nephp/?m=show&id=3392, Accessed: 21 February 2007

[12] Vodacom, "Vodacom Group Annual Results", 2006, p3. Also online: www.telkom.co.za/pls/portal/docs/page/contents/minisites/ir/docs/vodacom_2006AR.pdf, Accessed: 21 February 2007

[13] Cell C, "Cell C Media Briefing", April 2006, p7. Also online: http://www.cellc.co.za/images/Media%20Briefing%2025%20April%202006%20vFINAL4.pdf, Accessed: 21 February 2007

[14] MTN, "MTN Integrated Business Report", 2005, p22. Also online: http://www.mtn.com/mtn.group.web/investor/financial/financial_reports.asp, Accessed: 21 February 2007

[15] MobileIN.com, "Mobile VAS", http://www.mobilein.com/mobile_VAS.htm, Accessed: 21 February 2007

[16] ISO/IEC, "Information technology – Telecommunications and information exchange between systems – Private Integrated Services Network – Specification, functional model and information flows – Short message service", Reference number ISO/IEC 21989:2002(E), 2002, p4

[17] Trosby, Finn, "SMS, the strange duckling of GSM", Telektronikk 3.2004, 2004, p192

[18] GSMWorld, "MMS - What is MMS?" http://www.gsmworld.com/technology/mms/whatis_mms.shtml, Accessed: 21 February 2007

[19] Mobile in a Minute, "Unstructured Supplementary Services Data", http://www.mobilein.com/ussd.htm, Accessed: 21 February 2007

[20] Mobile in a Minute, "Wireless Application Protocol", http://www.mobilein.com/wap.htm, Accessed: 21 February 2007

[21] GSMWorld, "GPRS Platform", http://www.gsmworld.com/technology/gprs/, Accessed: 21 February 2007

[22] GSMWorld, "EDGE Platform", http://www.gsmworld.com/technology/edge/index.shtml, Accessed: 21 February 2007

[23] GSMWorld, "3GSM Platform", http://www.gsmworld.com/technology/3g/index.shtml, Accessed: 21 February 2007

[24] Telkom, "Investor Relations: Glossary", http://www.telkom.co.za/minisites/ir/glossary.html, Accessed: 21 February 2007

[25] SmartTrust Group, "Technical Description, Delivery Platform Version 5.2", Document number: 17409077, Revision: M, 2001, p18

[26] SmartTrust Group, "WIG Browser Request Protocol Specification", Document number: 17455121 Revision: D, 2000, p5

[27] Ginzboorg, P, "Seven comments on charging and billing", Communications of the ACM, v 43, n 11, Nov 2000, p89-92

[28] Reynolds, P, "A new architecture for 3rd generation mobile networks", Proceedings of the 2nd International Network Conference, 2000, p89

[29] R McBride, "Can you bill for convergence?" Telecommunications, part 7, 2004, p29-31

[30] Cisco Systems Inc, "Content-Based Billing, Increasing Revenue Within the Confines of the Limited Wireless Spectrum", 2005. Also online: http://www.cisco.com/en/US/netsol/ns341/ns396/ns177/ns278/networking_solutions_white_paper0900aecd8021cd51.shtml, Accessed: 21 February 2007

[31] Goldman, Jeff, "Billing Systems & Services: Infranet by Portal", T-Online, 2001, http://www.isp-planet.com/services/billing/infranet.html, Accessed: 21 February 2007

[32] Amdocs, "Billing", http://www.amdocs.com/Site/Offerings/Offering+Framework/Revenue+Management/Billing/Next-Generation+Convergent+Billing.htm, Accessed: 21 February 2007

[33] Maxbill, "Products: MaxGen", http://www.maxbill.com/NN/maxgen.html, Accessed: 21 February 2007

[34] Ofrane, Avi, Harte, Lawrence, "Introduction to Telecom Billing, Usage Events, Call Detail Records, and Billing Cycles", ALTHOS Publishing, 2003

[35] SmartTrust Group, "Technical Description, Delivery Platform Version 5.2", Document number: 17409077, Revision: M, 2001

[36] DARPA Internet Program, "RFC 793 - Transmission Control Protocol", 1981, http://www.faqs.org/rfcs/rfc793.html, Accessed: 21 February 2007

[37] Webopedia Computer Dictionary, "What is local-area network?", 2003, www.webopedia.com/TERM/L/local_area_network_LAN.html, Accessed: 21 February 2007

[38] Cisco, "Introduction to WAN technologies", http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/introwan.htm, Accessed: 21 February 2007

[39] Network Working Group, "Hypertext Transfer Protocol - HTTP/1.1", 1999, http://www.w3.org/Protocols/rfc2616/rfc2616.html, Accessed: 21 February 2007

[40] Accenture, Content Routing Engine Web Services Detailed Design, 2003, p5

[41] W3C, "SOAP Version 1.2 Part 0: Primer", W3C Recommendation, 2003, http://www.w3.org/TR/2003/REC-soap12-part0-20030624/ , Accessed: 21 February 2007

[42] Linux Online! "What is Linux", http://www.linux.org/info/index.html, Accessed: 21 February 2007

[43] Wikipedia Online Encyclopedia, "Java programming language", http://en.wikipedia.org/wiki/Java_programming_language, Accessed: 21 February 2007

[44] The Linux Documentation Project, "Advanced Bash-Scripting Guide, Part 1: Introduction", http://www.tldp.org/LDP/abs/html/part1.html, Accessed: 21 February 2007

[45] The Linux Documentation Project, "Advanced Bash-Scripting Guide, Chapter 1: Why Shell Programming?", http://www.tldp.org/LDP/abs/html/why-shell.html, Accessed: 21 February 2007

[46] Accenture, Content Routing Engine Web Services Detailed Design, 2003, p17

[47] SmartTrust Group, "WIG Application Guidelines", Document number: 17455120, Revision: F, 2001, p6

[48] Accenture, Content Routing Engine Web Services Detailed Design, 2003, p14

[49] Royce, Dr Winston W, "Managing the development of large software systems", TRW Publishers, 1970

[50] SmartTrust Group, "WIG Browser Request Protocol Specification, Delivery Platform Version 5.2", Document number: 17455121, Revision: D, 2001, p5

[51] Fielding et al, "Hypertext Transfer Protocol - HTTP/1.1, RFC 2616", http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html, Accessed: 21 February 2007

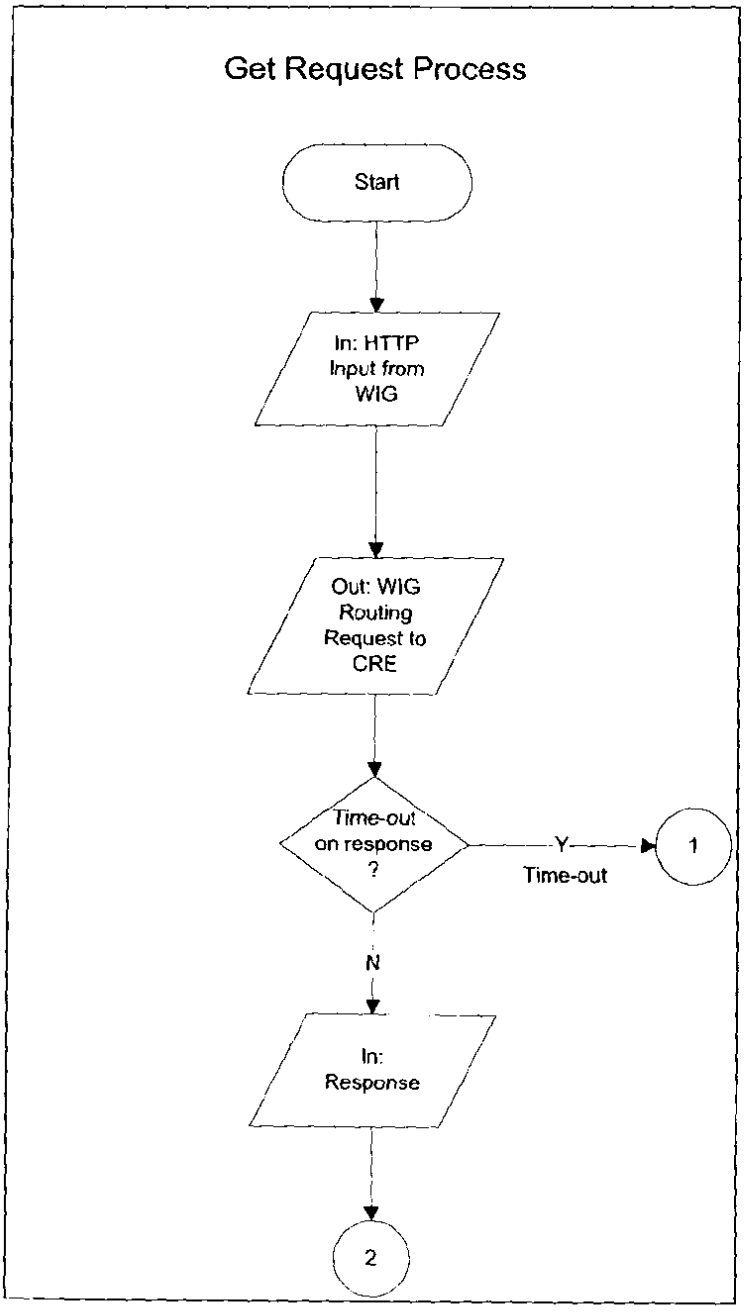
[52] SchlumbergerSema, "DP5 HTTP Proxy: User Requirement Specifications", 2003, p9

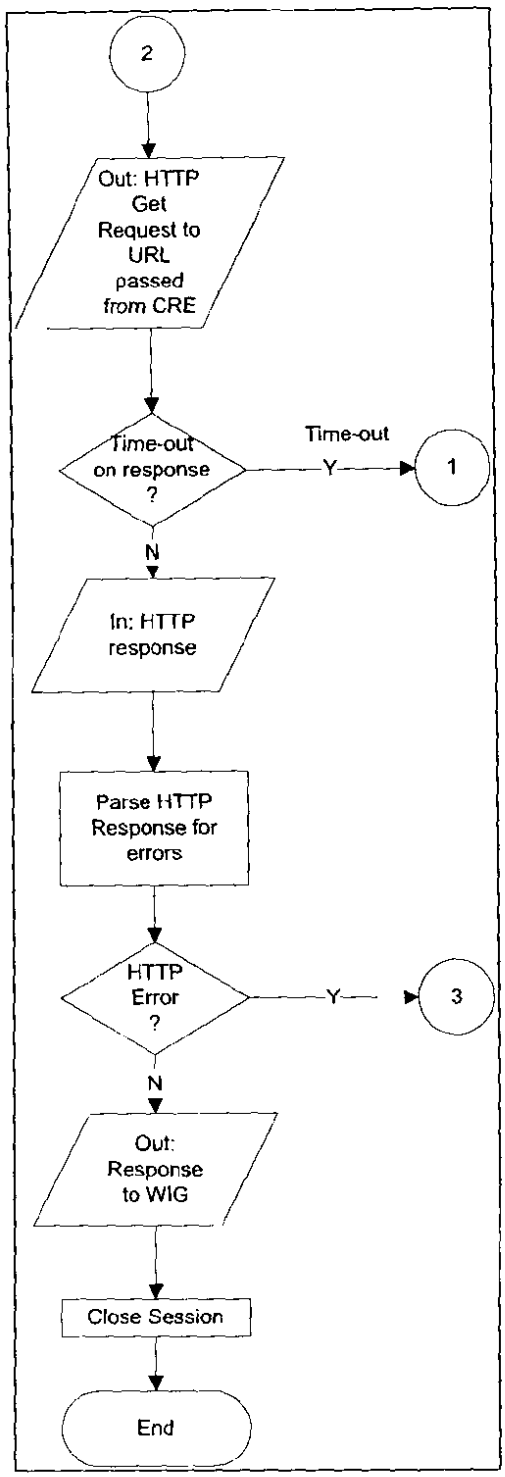
[53] Dell, "Dell PowerEdge 6600 Systems User's Guide", Also online: http://support.dell.com/support/edocs/systems/pe6600/en/ug/028uuaa0.htm#1039239, Accessed: 21 February 2007

APPENDICES

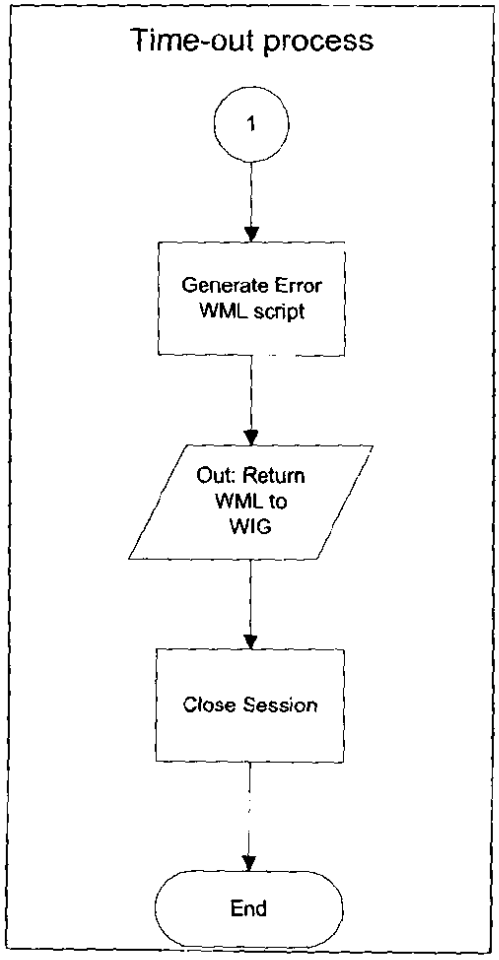
Appendix A Design flow diagrams

A.1 Request process

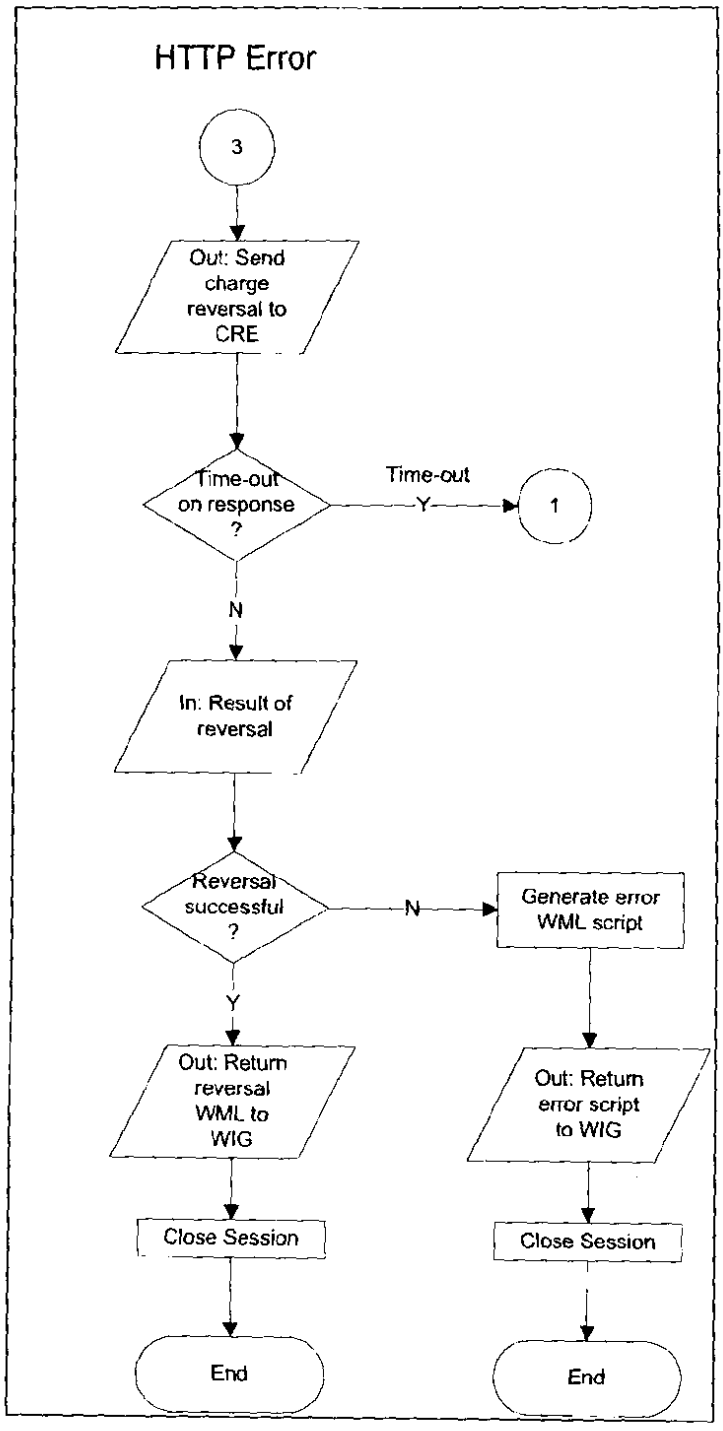



2
Out: HTTP Get Request to URL passed from CRE
Time-out on response ?
Time-out
Y
1
N
In: HTTP response
Parse HTTP Response for errors
HTTP Error ?
Y
3
N
Out: Response to WIG
Close Session
End

A.2 Time-out process



A.3 HTTP Error Process



Appendix B Installation and configuration

B.1 Receivables

The following is necessary from the operator:

- Network connectivity between application and WIG on ports 80, 8080, 8081 and 8082.
- Network connectivity between application and CRE on SOAP port.
- Network connectivity between application and all CPs on all required ports.
- CRE functionality, i.e. the CRE must provide validation of requests and routing to CPs, as well as validation of responses.

B.1.1 Hardware

The hardware was chosen by the operator, as shown in section 2.4, as:

- Dell PowerEdge 6600

B.1.2 Software

The software was specified by the operator as:

- OS: Red Hat Linux 7.3
- Java Run-Time Environment – newest stable version

B.2 Network prerequisites

The hardware must be on the internal network, close to the WIG platform. Ports must be open for HTTP (80, 8080, 8081 and 8082) traffic to flow between the application and the

Internet, via the firewall. All HTTP data must also be allowed to flow between the application ports and the WIG platform. SOAP connectivity between the CBA and CRE on CRE ports must be possible.

Connection	Ports
WIG to CBA	Port configured on CBA
CBA to all CPs	80, 8080, 8081, 8082
Internet to CBA	Response on all ports.
CBA and CRE	80

Table B-1: Connections

B.3 Linux installation

Insert Disk 1 of the Red Hat 7.3 suite in the CD-ROM and boot, making sure the CD-ROM is configured in BIOS as the primary boot device. This will present you with the Red Hat graphical installation utility. Use the trackball to make all selections and follow on-screen instructions.

Answer the questions posed by the installation utility, noting the following:

1. Install Type: Server.
2. Partition: Use default or any other preferred configuration.
3. Use LILO or GRUB as boot manager (author used LILO).
4. Network configuration. This is site specific and can be requested from the operator.
5. Firewall: Select "None" as the system is connected to a trusted network.
6. User accounts: Configure a root user and one additional user to be used to run the application.

B.4 Linux configuration

After installation is finished and the machine is rebooted, log in as root and configure the parameters described in the following paragraphs (in no particular order). After finishing with this section, reboot the machine to make sure all changes are introduced and start-up scripts are operating as they should.

B.4.1 Domain Name Service (DNS) server

The IP address of the DNS server must be entered into the file `/etc/resolv.conf`. To tell Linux to use DNS as the primary source of host name resolution, edit `/etc/nsswitch.conf` in the following way: The line containing "hosts:" should read:

```
hosts:      dns files nisplus
```

B.4.2 Hosts file

Entries in the hosts file are only necessary if DNS lookups fail. In this case added entries for the following are necessary:

1. The CBA's address and host name.
2. The CRE's address and host name.
3. All CPs' addresses and host names.

B.4.3 Host name

To set the machine's hostname and domain, edit the following file: `/etc/sysconfig/network`. The first line should be:

```
NETWORKING=yes
```

In the second line, enter the host name followed by the domain name as assigned by the network administrator:

```
HOSTNAME=<host name>.<domainname>
```

B.4.4 Daemons

Some of the server daemons (tasks running in the background to enable us to connect to the server) are disabled by default and need to be enabled. In the directory `/etc/xinetd.d`, edit two files, `wu-ftpd` and `telnet`, so that the line containing "disable" reads:

```
disable=no
```

B.4.5 Full duplex mode for Network Interface Cards

To switch on the Network Interface Cards full duplex mode, run the following command:

```
> ethtool -s eth0 duplex full speed 100 autoneg off
```

Store this command in one of the start-up scripts so that it is run every time the machine is rebooted.

B.5 JRE installation

Download the latest installation of the JRE 1.4 from Sun Microsystems (http://java.sun.com). This installation will either be in the Red Hat Package Manager

(RPM) or TAR format (look at the file's extension). Copy the file to the server's hard drive by using the floppy drive, a USB memory key or the network, and install.

B.5.1 Installing a JRE RPM

If the JRE is downloaded as a RPM (with extension .rpm), with the name <package name>, install it using the following command:

```
> rpm -i <package name>
```

B.5.2 Installing a JRE TAR file

If the JRE is downloaded as a TAR file (with extension .tar), with the name <tar file>, move the tar file to any other path, as long as it's a root-owned path, and run the following command:

```
> tar -xvf <tar file>
```

B.5.3 Configuring JRE

Create a symbolic link to the package directory, with this command:

```
> ln -s <jre path> /usr/local/java
```

B.6 CBA installation

B.6.1 Installation

To extract the cba.tar file, run the following command:

```
> tar -xvf cba.tar
```

This will create the following directory structure and files:

```
+-- ----start.sh
        stop.sh
      | archive.sh
        cba.cfg
        ---logs
        ----classes
```

B.6 Configuration parameters

An example of the configuration file follows.

```
#-   Configuration---
      05 09:21:14 SAST 2003
      time-out=Internal connection timed out. Please try again later or contact
      are.
      ryFailureUpdate=DeliveryFailureUpdate
      me=cre_user
      out=60
      perror=Host returned an error. Please try again later or contact customer care.
      HTTP/1.1
      rload=Server too busy. Please try again later.
      h=passwd.sav
      0
      store=/usr/java/jre/lib/security/cacerts
      shold=1000
      ere=Error routing request. Please contact customer care.
      and_filename=sid.sav
      BAID
      d=CBAREVID
      tingRequest=WIGRoutingRequest
      s=10000
      t=120
      knownhost=Host could not be reached. Please try again later or contact customer

      http\://CRE-SERVER\:80/CRE_WS
      ALL
      size=1024
      funds=Sorry, you appear to have insufficient funds for this transaction. Please
      ur account.
      uget=Error in request. Please redownload application and try again.
      space=http\://www.operator.net/
      unds=7210
      -out=Connection timed out. Please try again later or contact customer care.
```

Appendix C Laboratory test results

C.1 HTTP status codes

The status codes returned from the CPs indicate whether an HTTP response is valid, and if not, what the reason is for the error. Some of these codes are given in Table C-1 below.

HTTP status code	Description
200	OK
400	Bad request
401	Unauthorized
402	Payment required
403	Forbidden
404	Page not found
405	Method not allowed
406	Not acceptable

Table C-1: Some HTTP Status codes [51]

C.2 WIG Interface Engine test

Using Internet Explorer, send request to the CRE simulator (Apache, Jakarta), via CBA (localhost port 8088).

Extract of the CBA logs:

```
Apr 15, 2003 1:53:24 PM : INFO    : proxConfServlet setupLogging : Logger created.
Logfile='2003-04-15-135324.log'
Apr 15, 2003 1:53:24 PM : INFO    : proxConfServlet setupLogging : File Log level set to
ALL
Apr 15, 2003 1:53:24 PM : INFO    : proxConfServlet setupLogging : Logger created.
Logfile='2003-04-15-135324.log'
```

```
Apr 15, 2003 1:53:24 PM : INFO    : proxConfServlet setupLogging : File Log level set to
ALL
Apr 15, 2003 1:53:24 PM : INFO    : CBA main : HTTP Proxy started...
Apr 15, 2003 1:53:27 PM : FINE    : CBA go : Request received from
Socket[addr=/127.0.0.1,port=1227,localport=8088]
Apr 15, 2003 1:53:27 PM : FINEST  : CBA go : Deflecting from:
Socket[addr=/127.0.0.1,port=1227,localport=8088]
Apr 15, 2003 1:53:27 PM : FINEST  : CBA go : Deflecting to:
Socket[addr=/127.0.0.1,port=80,localport=1228]
Apr 15, 2003 1:53:27 PM : FINEST  : CBA go : Deflecting from:
Socket[addr=/127.0.0.1,port=80,localport=1228]
Apr 15, 2003 1:53:27 PM : FINEST  : CBA go : Deflecting to:
Socket[addr=/127.0.0.1,port=1227,localport=8088]
Apr 15, 2003 1:53:27 PM : FINE    : SocketDeflector run : GET http://localhost/CBA/cre
HTTP/1.0
Accept: */*
Accept-Language: en-us
Pragma: no-cache
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
Host: localhost
Proxy-Connection: Keep-Alive


Apr 15, 2003 1:53:27 PM : FINE    : SocketDeflector run : HTTP/1.1 200 OK
Date: Tue, 15 Apr 2003 11:53:27 GMT
Server: Apache/1.3.27 (Win32) mod_jk
Servlet-Engine: Tomcat Web Server/3.2.4 (JSP 1.1; Servlet 2.2; Java 1.4.1_01; Windows
2000 5.0 x86; java.vendor=Sun Microsystems Inc.)
Connection: close
Content-Type: text/html

<html>
<head><title>creservlet</title></head>
<body>
<p>The CRE simulator servlet has received a request (GET or POST). This is the reply.</p>
</body></html>
```

The response at the end of the log shows that the request reached the CRE simulator, proving the functionality of the WIG Interface Engine.

C.3 CRE Interface Engine test

This test is the same as the previous one, except for the retrieving and logging of the URL and HTTP response status code afterwards.

Extract of the CBA logs:

```
Apr 16, 2003 1:32:11 PM : INFO    : proxConfServlet setupLogging : Logger created.
Logfile='2003-04-16-133211.log'
Apr 16, 2003 1:32:11 PM : INFO    : proxConfServlet setupLogging : File Log level set to
ALL
Apr 16, 2003 1:32:11 PM : INFO    : proxConfServlet setupLogging : Logger created.
Logfile='2003-04-16-133211.log'
Apr 16, 2003 1:32:11 PM : INFO    : proxConfServlet setupLogging : File Log level set to
ALL
Apr 16, 2003 1:32:11 PM : INFO    : CBA main : HTTP Proxy started...
Apr 16, 2003 1:32:15 PM : FINE    : CBA go : Request received from
Socket[addr=/127.0.0.1,port=1414,localport=8088]
Apr 16, 2003 1:32:15 PM : FINE    : RequestThread run : URL=http://localhost/CBA/cre
```

```
Apr 16, 2003 1:32:15 PM : FINEST  : RequestThread run : GET http://localhost/CBA/cre
HTTP/1.0
Accept: */*
Accept-Language: en-us
Pragma: no-cache
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
Host: localhost
Proxy-Connection: Keep-Alive




Apr 16, 2003 1:32:15 PM : FINE    : ResponseThread run : STATUS=200
Apr 16, 2003 1:32:15 PM : FINEST  : ResponseThread run : HTTP/1.1 200 OK
Date: Wed, 16 Apr 2003 11:32:15 GMT
Server: Apache/1.3.27 (Win32) mod_jk
Servlet-Engine: Tomcat Web Server/3.2.4 (JSP 1.1; Servlet 2.2; Java 1.4.1_01; Windows
2000 5.0 x86; java.vendor=Sun Microsystems Inc.)
Connection: close
Content-Type: text/html

<html>
<head><title>CRE Simulator</title></head>
<body>
<p>The CRE simulator servlet has received a request (GET or POST). This is the reply.</p>
</body></html>
```

The log shows that a value for URL and STATUS has been extracted, proving the functionality of the CRE Interface Engine.

C.4 CP Interface Engine test

Using the CreClient code, a sample SOAP-XML message is sent to see if anything can be received.

SOAP-XML Request:

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
       <m:WIGRoutingRequest xmlns:m="Some-URI">
           <TxID>1</TxID>
           <URL>http://localhost/CBA/cp</URL>
           <EventID>999</EventID>
       </m:WIGRoutingRequest>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP-XML Response:

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
   <SOAP-ENV:Body>
       <m:WIGRoutingRequestResponse xmlns:m="Some-URI">
           <Error>0</Error>
           <mURL>http://localhost:8080/CBA/cp</mURL>
       </m:WIGRoutingRequestResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Extract of the CBA log:

```
Apr 17, 2003 3:14:02 PM : INFO    : proxConfServlet setupLogging : Logger created.
Logfile='2003-04-17-151402.log'
Apr 17, 2003 3:14:03 PM : INFO    : proxConfServlet setupLogging : File Log level set to
ALL
Apr 17, 2003 3:14:03 PM : INFO    : proxConfServlet setupLogging : Logger created.
Logfile='2003-04-17-151403.log'
Apr 17, 2003 3:14:03 PM : INFO    : proxConfServlet setupLogging : File Log level set to
ALL
Apr 17, 2003 3:14:03 PM : INFO    : CBA main : HTTP Proxy started...
Apr 17, 2003 3:14:04 PM : FINE    : CBA go : Request received from
Socket[addr=/127.0.0.1,port=1774,localport=8088]
Apr 17, 2003 3:14:04 PM : FINE    : RequestThread run : URL=http://localhost/CBA/cp
Apr 17, 2003 3:14:04 PM : FINEST  : CreClient <init> : __SOAP packet sent to CRE__
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
       <m:WIGRoutingRequest xmlns:m="Some-URI">
           <TxID>1</TxID>
           <URL>http://localhost/CBA/cp</URL>
           <EventID>999</EventID>
       </m:WIGRoutingRequest>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

Apr 17, 2003 3:14:08 PM : FINEST  : CreClient <init> : __SOAP packet received from CRE___
Apr 17, 2003 3:14:08 PM : FINEST  : CreClient <init> : <SOAP-ENV:Envelope
Apr 17, 2003 3:14:08 PM : FINEST  : CreClient <init> :   xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
Apr 17, 2003 3:14:08 PM : FINEST  : CreClient <init> :   SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
Apr 17, 2003 3:14:08 PM : FINEST  : CreClient <init> :    <SOAP-ENV:Body>
Apr 17, 2003 3:14:08 PM : FINEST  : CreClient <init> :
<m:WIGRoutingRequestResponse xmlns:m="Some-URI">
Apr 17, 2003 3:14:08 PM : FINEST  : CreClient <init> :           <Error>0</Error>
Apr 17, 2003 3:14:08 PM : FINEST  : CreClient <init> :
<mURL>http://localhost:8080/cp</mURL>
Apr 17, 2003 3:14:08 PM : FINEST  : CreClient <init> :
</m:WIGRoutingRequestResponse>
Apr 17, 2003 3:14:08 PM : FINEST  : CreClient <init> :    </SOAP-ENV:Body>
Apr 17, 2003 3:14:08 PM : FINEST  : CreClient <init> : </SOAP-ENV:Envelope>
Apr 17, 2003 3:14:08 PM : FINEST  : RequestThread run : GET http://localhost/CBA/cp
HTTP/1.0
Accept: */*
Accept-Language: en-us
Pragma: no-cache
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
Host: localhost
Proxy-Connection: Keep-Alive
```

These logs show that the CP URL has been specified by the CRE, and the specified URL has been reached.

C.5 URL and Error code retrieval

The CRE simulator hard-codes a test URL into the SOAP response. This test will show if this URL, as well as the hostname and port, can be extracted.

Extract of the System.out of the JVM:

```
[illegible]quest to CRE
[illegible] URL=http://localhost/CBA/cp
[illegible] http://localhost:8080/dpprox/cp
Host=localhost
[illegible]8080
STATUS=200
```

It should also be possible to extract the error code and redirect to the error page server in case of a non-zero error. Let's change the CRE simulator to return a non-zero error, with no mURL tag, and see what happens.

Extract of the CBA log:

```
Apr 22, 2003 4:00:43 PM : FINE    : CBA go : Request received from
Socket[addr=/127.0.0.1,port=2116,localport=8088]
Apr [illegible]03 4:00:43 PM : FINE    : RequestThread run : URL=http://localhost/CBA/cp
[illegible]03 4:00:43 PM : FINEST  : CreClient <init> : __SOAP packet sent to CRE__
[illegible]Envelope
  [illegible]OAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
      <m:WIGRoutingRequest xmlns:m="Some-URI">
         <TxID>1</TxID>
         <URL>http://localhost/CBA/cp</URL>
         <EventID>999</EventID>
      </m:WIGRoutingRequest>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

Apr 22, 2003 4:00:44 PM : FINEST  : CreClient <init> : __SOAP packet received from CRE__
Apr 22, 2003 4:00:44 PM : FINEST  : CreClient <init> : <SOAP-ENV:Envelope
Apr 22, 2003 4:00:44 PM : FINEST  : CreClient <init> :   xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
Apr 22, 2003 4:00:44 PM : FINEST  : CreClient <init> :   SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
Apr 22, 2003 4:00:44 PM : FINEST  : CreClient <init> :    <SOAP-ENV:Body>
Apr 22, 2003 4:00:44 PM : FINEST  : CreClient <init> :
<m:WIGRoutingRequestResponse xmlns:m="Some-URI">
Apr 22, 2003 4:00:44 PM : FINEST  : CreClient <init> :          <Error>-99</Error>
Apr 22, 2003 4:00:44 PM : FINEST  : CreClient <init> :
</m:WIGRoutingRequestResponse>
Apr 22, 2003 4:00:44 PM : FINEST  : CreClient <init> :    </SOAP-ENV:Body>
Apr 22, 2003 4:00:44 PM : FINEST  : CreClient <init> : </SOAP-ENV:Envelope>
```

Extract of the JVM's System.out:

```
Sending request to CRE
Descriptive URL=http://localhost/CBA/cp
Real URL=http://127.0.0.1/badrurl.wml
STATUS=200
```

This shows that because the error code is non-zero, the URL is changed to a static WML page on an error server. The absent mURL tag does not cause problems either.

C.6 Client request to new URL

To do this, the client requests the CRE simulator to return a real URL to the CP simulator on port 8080. Then he/she stops the Apache server. If one makes a request to the CP simulator on port 80, and it does not get deflected to port 8080, one should not be able to reach Tomcat. But if the CBA does deflect it to the real URL specified by the CRE simulator, one would reach Tomcat with the request.

The CBA edits the original HTTP packet to replace the descriptive URL in the "get" request with the real URL returned from the CRE. Both the "get" request string in the packet and the Host field must be changed.

These results may be visited in the CBA logs:

```
Apr 29, 2003 2:00:57 PM : INFO    : proxConfServlet setupLogging : Logger created.
Logfile='2003-04-29-140057.log'
Apr 29, 2003 2:00:57 PM : INFO    : proxConfServlet setupLogging : File Log level set to
ALL
Apr 29, 2003 2:00:57 PM : INFO    : proxConfServlet setupLogging : Logger created.
Logfile='2003-04-29-140057.log'
Apr 29, 2003 2:00:57 PM : INFO    : proxConfServlet setupLogging : File Log level set to
ALL
Apr 29, 2003 2:00:57 PM : INFO    : CBA main : HTTP Proxy started...
Apr 29, 2003 2:01:01 PM : FINE    : CBA go : Request received from
Socket[addr=/127.0.0.1,port=1971,localport=8088]
Apr 29, 2003 2:01:01 PM : FINEST  : SessionThread run : GET
http://163.187.225.108/ncs32sql/SilverStream/Pages/ncTime_V2Frameset.html HTTP/1.0
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-
powerpoint, application/vnd.ms-excel, application/msword, */*
Accept-Language: en-us
If-Modified-Since: Fri, 25 Apr 2003 08:20:03 GMT
If-None-Match: "0-41-3ea8efb3"
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
Host: 163.187.225.108
Proxy-Connection: Keep-Alive

.....................................................
```

```
Apr 29, 2003 2:01:01 PM : FINE    : SessionThread run :
URL=http://163.187.225.108/ncs32sql/SilverStream/Pages/ncTime_V2Frameset.html
Apr 29, 2003 2:01:01 PM : FINEST  : CreClient <init> : __SOAP packet sent to CRE__
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
       <m:WIGRoutingRequest xmlns:m="Some-URI">
           <TxID>1</TxID>

<URL>http://163.187.225.108/ncs32sql/SilverStream/Pages/ncTime_V2Frameset.html</URL>
           <EventID>999</EventID>
       </m:WIGRoutingRequest>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

Apr 29, 2003 2:01:02 PM : FINEST  : CreClient <init> : __SOAP packet received from CRE___
Apr 29, 2003 2:01:02 PM : FINEST  : CreClient <init> : <SOAP-ENV:Envelope
Apr 29, 2003 2:01:02 PM : FINEST  : CreClient <init> :   xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
Apr 29, 2003 2:01:02 PM : FINEST  : CreClient <init> :   SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
Apr 29, 2003 2:01:02 PM : FINEST  : CreClient <init> :    <SOAP-ENV:Body>
Apr 29, 2003 2:01:02 PM : FINEST  : CreClient <init> :
<m:WIGRoutingRequestResponse xmlns:m="Some-URI">
Apr 29, 2003 2:01:02 PM : FINEST  : CreClient <init> :            <Error>0</Error>
Apr 29, 2003 2:01:02 PM : FINEST  : CreClient <init> :
<mURL>http://localhost/CBA/cp</mURL>
Apr 29, 2003 2:01:02 PM : FINEST  : CreClient <init> :
</m:WIGRoutingRequestResponse>
Apr 29, 2003 2:01:02 PM : FINEST  : CreClient <init> :    </SOAP-ENV:Body>
Apr 29, 2003 2:01:02 PM : FINEST  : CreClient <init> : </SOAP-ENV:Envelope>
Apr 29, 2003 2:01:02 PM : FINE    : SessionThread run : Request deflected to
Socket[addr=localhost/127.0.0.1,port=80,localport=1974]
Apr 29, 2003 2:01:03 PM : FINE    : ResponseThread run : STATUS=200
Apr 29, 2003 2:01:03 PM : FINEST  : ResponseThread run : HTTP/1.1 200 OK
Date: Tue, 29 Apr 2003 12:01:02 GMT
Server: Apache/1.3.27 (Win32) mod_jk
Servlet-Engine: Tomcat Web Server/3.2.4 (JSP 1.1; Servlet 2.2; Java 1.4.1_01; Windows
2000 5.0 x86; java.vendor=Sun Microsystems Inc.)
Connection: close
Content-Type: text/html

<wml>
<card id="a">
<p>The content provider simulator servlet has received a GET. This is the reply.</p>
</card></wml>


Apr 29, 2003 2:01:03 PM : FINEST  : RequestThread run : GET http://localhost/CBA/cp
HTTP/1.0
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.ms-
powerpoint, application/vnd.ms-excel, application/msword, */*
Accept-Language: en-us
If-Modified-Since: Fri, 25 Apr 2003 08:20:03 GMT
If-None-Match: "0-41-3ea8efb3"
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
Host: localhost
Proxy-Connection: Keep-Alive
```

The response was generated by the Tomcat server, which proves that the request made to the Apache server was rerouted, as expected to be.

C.7 Errors

By adjusting the CP and CRE simulators and the CBA's time-out settings, we can simulate the following error conditions, and the resulting CBA logs are shown.

C.7.1 Time-out to host

With the delay set in the Java code of the Tomcat server to a value that will cause the CBA to 'time-out' while waiting for a response from the host, we get the following extract of the CBA log:

```
May 6, 2003 3:33:42 PM : WARNING : ErrorHandler time-outOnHost : Time-out on TxID[1]
May 6, 2003 3:33:42 PM : FINE    : ErrorHandler time-outOnHost : Responded to TxID[1]
with error message.
```

The log entries show the time-out and the fact that it responded with an error message. This error message respond is defined in the configuration file.

C.7.2 Time-out to CRE

With a long delay set in the response time of the CRE simulator, this extract of the CBA log is obtained:

```
May 7, 2003 12:40:25 PM : FINE      : CBA go : Request with TxID[1] received from
Socket[addr=/127.0.0.1,port=1809,localport=8088]

May 7, 2003 12:40:25 PM : FINEST    : SessionThread run : GET
http://163.187.225.108/ncs32sql/SilverStream/Pages/ncTime_V2Frameset.html HTTP/1.0

Accept: */*
Accept-Language: en-us
Pragma: no-cache
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
Host: 163.187.225.108
Proxy-Connection: Keep-Alive

.....
.....
.....
May 7, 2003 12:40:25 PM : FINE    : SessionThread run : TxID[1]
URL=http://163.187.225.108/ncs32sql/SilverStream/Pages/ncTime_V2Frameset.html
May 7, 2003 12:40:26 PM : FINEST  : CreClient run : __SOAP packet sent to CRE__
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
       <m:WIGRoutingRequest xmlns:m="Some-URI">
           <TxID>1</TxID>
```

```
<URL>http://163.187.225.108/ncs32sql/SilverStream/Pages/ncTime_V2Frameset.html</URL>
          <EventID>999</EventID>
       </m:WIGRoutingRequest>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

May 7, 2003 12:40:31 PM : WARNING : SessionThread run : Time-out to CRE on TxID[1]
May 7, 2003 12:40:31 PM : FINE    : SessionThread run : TxID[1] Real
URL=http://127.0.0.1/badcre.wml
May 7, 2003 12:40:31 PM : FINE    : SessionThread run : TxID[1] Request deflected to
Socket[addr=/127.0.0.1,port=80,localport=1812]
May 7, 2003 12:40:31 PM : FINE    : ResponseThread run : STATUS of TxID[1]=200
May 7, 2003 12:40:31 PM : FINEST  : ResponseThread run : HTTP/1.1 200 OK
Date: Wed, 07 May 2003 10:40:31 GMT
Server: Apache/1.3.27 (Win32) mod_jk
Last-Modified: Tue, 06 May 2003 13:29:51 GMT
ETag: "0-4f-3eb7b8cf"
Accept-Ranges: bytes
Content-Length: 79
Connection: close
Content-Type: text/vnd.wap.wml

<wml>
ERROR: Routing info not available. Please contact customer care.
</wml>.....
.....
.....
May 7, 2003 12:40:31 PM : FINEST  : RequestThread run : GET http://127.0.0.1/badcre.wml
HTTP/1.0
Accept: */*
Accept-Language: en-us
Pragma: no-cache
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
Host: 127.0.0.1
Proxy-Connection: Keep-Alive
```

In the log the error message is shown as a WML page fetched from the error-server.

C.7.3 GET error

By specifying an illegal "get" request, this extract of the CBA log is obtained:

```
May 7, 2003 1:04:28 PM : FINE    : CBA go : Request with TxID[1] received from
Socket[addr=/127.0.0.1,port=1955,localport=8088]
May 7, 2003 1:04:28 PM : FINEST  : SessionThread run : CONNECT gateway.nam.slb.com:443
HTTP/1.0
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
Host: gateway.nam.slb.com
Content-Length: 0
Proxy-Connection: Keep-Alive
Pragma: no-cache

.....
.....
.....
.....
May 7, 2003 1:04:28 PM : FINE    : SessionThread run : TxID[1]
URL=http://127.0.0.1/badget.wml
May 7, 2003 1:04:28 PM : FINE    : SessionThread run : TxID[1] Real
URL=http://127.0.0.1/badget.wml
May 7, 2003 1:04:28 PM : FINE    : SessionThread run : TxID[1] Request deflected to
Socket[addr=/127.0.0.1,port=80,localport=1956]
```

Again, the error message is fetched from the error-server, namely: badget.wml

C.7.4 HTTP error

Attempting to fetch a non-existent page from the CP, we get this extract of the CBA log:

```
    3 2:41:09 PM : FINEST  : RequestThread run : GET http://163.187.225.80/CBA/cp

  */*
  nguage: en-us
  cache
  Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
63.187.225.80
nnection: Keep-Alive


003 2:41:09 PM : FINE    : ErrorHandler httpError : Responded to TxID[1] with
 message.
3 2:41:09 PM : FINE    : ResponseThread run : STATUS of TxID[1]=404
003 2:41:09 PM : FINEST  : ResponseThread run : HTTP/1.1 404
```

shows that the CBA recognised the error status of 404, and responded with an error message". The error message is again defined in the configuration file.

Reversal

In the case of HTTP error and Time-out-to-host errors, a reversal should be made to the MSC to negate the charging that was incurred on the upstream leg.

Extract of the CBA log:

```
8   2003  4:13:04  PM  :  FINE          :  SessionThread  run  :  TxID[1]  Real
tp://163.187.225.80/CBA/cp

8, 2003 4:13:05 PM : WARNING : SessionThread run : IO Error : Connection refused:
nnect : 163.187.225.80:80
May 8, 2003 4:13:05 PM : FINE    : ErrorHandler unknownHost : Responded to TxID[1] with
nknown host error message.
May 8, 2003 4:13:06 PM : FINEST  : CreReversalClient run : SOAP-REV-TX TxID[1]

SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Body>
     <m:DeliveryFailureUpdate xmlns:m="Some-URI">
         <TxID>1</TxID>
         <EventID>999</EventID>
     </m:DeliveryFailureUpdate>
 </SOAP-ENV:Body>
```

```
</SOAP-ENV:Envelope>

May 8, 2003 4:13:07 PM : FINE    : ErrorHandler unknownHost : TxID[1] Reversed
successfully
May 8, 2003 4:13:07 PM : FINEST  : CreReversalClient run : SOAP-REV-RX TxID[1]

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
   <SOAP-ENV:Body>
       <m:DeliveryFailureResponse xmlns:m="Some-URI">
           <Error>0</Error>
       </m:DeliveryFailureResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

These logs show the DeliveryFailureUpdate and DeliveryFailureResponse SOAP messages between the CBA and the CRE, proving that the charge was reversed from the CBA's side.

C.9 Latency test

In order to test the average latency of the application, a WIG simulator was made to send a WIG-like request, and log the delay experienced in the response. This test was run 10 times, and the following results were logged:

Extract of the WIG simulator log:

```
May 10, 2003 11:11:34 AM : INFO    : Request sent : http://localhost/CBA/cp
May 10, 2003 11:11:34 AM : INFO    : Response received : 200 OK : Delay=224ms
May 10, 2003 11:13:07 AM : INFO    : Request sent : http://localhost/CBA/cp
May 10, 2003 11:13:07 AM : INFO    : Response received : 200 OK : Delay=134ms
May 10, 2003 11:13:16 AM : INFO    : Request sent : http://localhost/CBA/cp
May 10, 2003 11:13:16 AM : INFO    : Response received : 200 OK : Delay=103ms
May 10, 2003 11:13:31 AM : INFO    : Request sent : http://localhost/CBA/cp
May 10, 2003 11:13:31 AM : INFO    : Response received : 200 OK : Delay=101ms
May 10, 2003 11:14:02 AM : INFO    : Request sent : http://localhost/CBA/cp
May 10, 2003 11:14:03 AM : INFO    : Response received : 200 OK : Delay=105ms
May 10, 2003 11:14:17 AM : INFO    : Request sent : http://localhost/CBA/cp
May 10, 2003 11:14:17 AM : INFO    : Response received : 200 OK : Delay=107ms
May 10, 2003 11:14:43 AM : INFO    : Request sent : http://localhost/CBA/cp
May 10, 2003 11:14:43 AM : INFO    : Response received : 200 OK : Delay=114ms
May 10, 2003 11:15:09 AM : INFO    : Request sent : http://localhost/CBA/cp
May 10, 2003 11:15:09 AM : INFO    : Response received : 200 OK : Delay=111ms
May 10, 2003 11:15:36 AM : INFO    : Request sent : http://localhost/CBA/cp
May 10, 2003 11:15:37 AM : INFO    : Response received : 200 OK : Delay=103ms
May 10, 2003 11:15:50 AM : INFO    : Request sent : http://localhost/CBA/cp
May 10, 2003 11:15:50 AM : INFO    : Response received : 200 OK : Delay=108ms
```

C.10 Throughput test

A copy of the HTML output of the benchmark software is shown below:

CBA Benchmark results:

Benchmark started at 12:59:14 on 21/5/2004

Benchmark finished at 12:59:19 on 21/5/2004

Duration of test	5421ms
Number of tests done	1001
Average response time	139.0ms
Minimum response time	2ms
Maximum response time	686ms
Maximum simultaneous threads	268
Number of responses	1001

Appendix D Acceptance test results

D.1 Setup steps

1. Note the configured time-out of the Session on DP5.
2. Disconnect CBA
3. Connect CBA
4. Disconnect CRE
5. Connect CRE
6. Disconnect CP
7. Note the subscriber account balance.

D.2 Execution steps

Test case no.	Action	Expected result
1.	1. Use the Handset to request web hosted content using a WIG service. 2. Monitor whether the Session ID stays open until a response is received by the handset.	The response is received on the handset, and the Session ID is closed immediately after receiving a response.
2.	1. Setup step D.1 must be executed. 2. Request content. 3. Monitor the request until it goes through to CP, and then disconnect HTTP CBA.	The Session experiences a time-out if it does not receive the response within configured time. After the time-out a response is received indicating that the CP is not available.
3.	1. Setup step 2 must be executed. 2. Use the Handset to request content using a WIG service.	The Session experiences a time-out if it does not receive the response within configured time. After the time-out a response is received indicating that the CP is not available, in a different format than the previous response.
4.	1. Setup steps 3 and 4 must be executed. 2. Use the Handset to request content using WIG Services software.	The request is discarded and the appropriate error message is returned on the handset, because CRE is offline.
5.	1. Setup steps 5 and 6 must be executed. 2. Use the Handset to request a bank account balance using WIG Services software.	The Session experiences a time-out if it does not receive the response within configured time. After the time-out a response is received indicating that the CP is not available.
6.	1. Setup step 7 must be executed. 2. Use the Handset to request content using WIG Services software, and verify whether subscriber account is debited.	• The request is discarded, because the CP is offline. • The subscriber account is debited, but on the return leg this same amount is reversed, so that the balance returns to the value it was before this test.
7.	1. Setup step 7 must be executed. 2. Use the handset to request free content, like the Information page.	• The response is received on the handset. • The subscriber account has the same value after the test as noted before the test.

Appendix E End-to-end test data

The results of the 24 end-to-end tests, each containing 100 requests, are summarised below.

Test nr	Request rate (requests/second)	Response rate (responses/second)	Average latency (seconds)
1.	0.5	0.487	1.312
2.	1.0	0.989	1.367
3.	1.5	1.453	1.407
4.	2.0	1.963	1.439
5.	2.5	2.461	1.473
6.	3.0	2.892	1.528
7.	3.5	3.458	1.521
8.	4.0	3.903	1.486
9.	4.5	4.462	1.563
10.	5.0	4.913	1.672
11.	5.5	5.459	1.687
12.	6.0	5.901	1.625
13.	6.5	6.437	2.122
14.	7.0	6.872	2.546
15.	7.5	7.448	3.092
16.	8.0	7.843	3.782
17.	8.5	8.462	4.557
18.	9.0	8.887	5.128
19.	9.5	9.446	6.293
20.	10.0	9.845	7.237
21.	10.5	10.420	9.180
22.	11.0	10.894	11.226
23.	11.5	10.813	13.373
24.	12.0	11.034	15.461