Department of Computer Science
The University of Auckland
Te Waananga O Waipapa
Auckland, New Zealand

Towards An Efficiency Electronic Micro-payment System

Xiaoling Dai

This thesis is submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science at The University of Auckland.

September 2003

© 2003 Xiaoling Dai

Abstract

Current macro-payment systems used by most E-commerce sites are not suitable for high-volume, low-cost transactions, such as charging per-page for web site browsing. These payment technologies suffer from use of heavy-weight encryption technologies and reliance on always on-line authorisation servers. Micro-payment systems offer an alternative strategy of pay-as-you-go charging, even for very low cost, very high-volume charging. However, several different micro-payment schemes exist, not all suitable for all E-commerce uses.

In this thesis, we develop a new protocol called NetPay which is a lightweight, flexible, off-line and secure protocol for electronic commerce over the Internet. NetPay is designed to support purchases ranging in value from a few cents to several dollars under large numbers of micro-payments. It is based on decentralized verification of electronic currency at a vendor's server with off-line payment capture. This is performed with a touchstone and index of e-coins that are passed from vendor to vendor.

We describe the NetPay protocol and its properties. We identify a set of requirements for micro-payment systems that use NetPay and customer/NetPay component interactions. We then focus on the key issues of designing three kinds of CORBA-based NetPay systems with an on-line newspaper application and prototyping two of them which include server-side e-wallet and client-side e-wallet NetPay systems. We also describe design and implementation of a set of reusable NetPay components, which enable NetPay components to be seamlessly added to an existing example web application. Various technologies are used to build these systems including J2EE Enterprise Java Bean, Java Server pages, CORBA and database. We have carried out three kinds of evaluations of micro-payment and macro-payment purchasing models for an on-line newspaper application to assess their relative strengths and weaknesses.

Acknowledgments

It is a pleasure to thank the many people who made this thesis possible.

I would like to thank my Ph.D. supervisor, Professor John Grundy for him remarkable advice and guidance during my research. Without his enthusiasm, inspiration, ideas and knowledge, my thesis would not have reached so far. Throughout my thesis-writing period, he provided encouragement, sound advice, good teaching, and lots of good ideas. I would have been lost without him.

I would like to thank my previous Ph.D. supervisor, Professor Bruce Lo, who was always willing to give advice and help guide me to work out the research proposal throughout my first year study.

I would like to express my gratitude to all my student colleagues who helped and advised me during my work on this project.

A number of people were used throughout the evaluation of NetPay systems. I would like to thank those people for giving up their time to participate, and providing useful feedback about the systems.

Heartfelt thanks go to Leanne and Aaron for correcting the spelling and grammar of this thesis.

I wish to thank my parents, Tingxuan Dai and Jingzhi Liu. They bore me, raised me, supported me, taught me, and loved me.

Lastly, and most importantly, I would like to thank my husband – Zhenquan Li for his love, encouragement and support while I was studying and doing the research. He was first reader of my thesis and gave me many good advices. I would like to dedicate this thesis to my husband and my two lovely daughters Leanne and Andrea.

Table of Contents

Chapter 1 - Introduction

Chapter 2 - Background and Prior Work

Chapter 3 - The NetPay Protocol

Chapter 4 – NetPay System Requirements

Chapter 5 – CORBA-based NetPay

Chapter 6 – Component-based NetPay

Chapter 7 - Evaluations

Chapter 8 – Conclusions and Future Work

Appendix A: Questionnaires

Appendix B: Usability Testing Forms

Appendix C: MD5 Implementation 213

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The rapid growth of the Internet has led to the appearance of thousands of different web sites, which are created to provide information to millions of people around the world. There is a trend towards charging for site content on the Internet [8] in order for companies to make direct profit from information they provide, rather than relying on fickle or insufficient on-line advertising revenue [42, 47]. Macro-payment systems are used by most E-commerce systems today to allow customer to pay for content. These typically use credit card debiting, digital cash or real-time bank transfers, where a customer pays for products or services before or at the time of delivery. Such systems typically use complex encryption technologies and require communications with an authorisation server to request and confirm payment. Usually confirmation must be provided before the content or goods are supplied to the customer. This model suits low-to-medium volume transactions of medium-to-high value e.g. books, food, office stationary, home appliances, toys and so on.

For example, many sites have become subscription-only access e.g. on-line newspapers, academic and trade periodicals, help and advice columns, and so on. Subscription has the disadvantage of locking customers to one site and a "one size fits all" scenario where even if the customer wants a few items from the site, they have to pay for them all. An alternative model is where a customer pays as they go from a previously acquired (by macro-payment) E-wallet with E-coins he/she is charged per-page or per-group or per-download for material, often very low cost per item [29, 42, 47]. Ideally they can move to other sites and use the same E-money. This is the micro-payment model of on-line information, product and service purchase.

There are a number of micro-payment systems in various stages of development from proposals in the academic literature to systems in commercial use [54, 34, 35, 40, 72]. Though micro-payment protocols have received a lot of attention from researchers and cryptographers, only one micro-payment system which is Millicent [95] exists in general public use in Japan. All existing protocols for micro-payments have their strengths and

weaknesses in practical applications [23]. In Millicent [54], the third party must be online whenever the user wishes to interact with a new vendor, i.e., the system places a heavy real-time burden on the third party. In Mpay [40], customers can pay nothing to access services for a full day and also the customer's anonymity is not protected. In PayWord [72], the payword chain is customer and vendor specific, i.e., the system locks customers to some sites that they have the payword chains.

There are various ways to build micro-payment systems and embed them in existing web applications. Using a "hard-coded" CORBA-based style the micro-payment systems are easy to implement, but there are obvious drawbacks. For example, if we want to add some micro-payment functions to an existing system, the source code of the existed system must be changed to suit the new needs. A component-based software architecture is one solution to plug-in micro-payment components to an existing web application.

Software architecture is concerned with the description of elements from which systems are built and the interaction among those elements. It is a vehicle for communication among the elements and captures early design decisions for systems so that software architecture plays an important role in the system development. NetPay software architecture design should be scalable, reliable, secure and flexible. A thin-client multi-tier web-based e-commerce system will be used in NetPay system development.

1.2 Our Research

To overcome the weaknesses of existing micro-payment protocols, we propose a novel payment protocol, named NetPay. NetPay is a debit-based off-line system. It also addresses the problem of double spending and overspending by customers, and anonymity of the customer [22]. The key innovation of NetPay is that it uses Touchstones and Indexes passed from broker to vendors. The touchstone is used by a vendor to verify the electronic currency, and an Index is used to prevent double spending from customers. The NetPay protocol shifts the communication traffic bottleneck from a broker and distributes it among the vendors, placing some of the micro-payment processing burden on vendors when a customer wishes to purchase from a new vendor.

We have developed a software architecture for implementing NetPay-based micro-payment systems for thin-client multi-tier web applications. We focused on designing three kinds of "E-wallets" for NetPay and "hard-coded" vendor prototypes using a CORBA-based approach. The three kinds of e-wallets include: (1) A server-side e-wallet held on the vendor server the customer is currently buying content from; (2) a client-side e-wallet hosted on the customer PC (an application that stores e-coin information for debit by vendor servers and credit by the broker server); and (3) a client-side cookie-based e-wallet is stored in a temporary cookie e-wallet for debiting instead of the e-wallet database. Each approach has advantages and disadvantages. The first requires uses of e-coin ID and password to login to a vendor system. The later two require that the customers download an e-wallet application to their PC and the e-coin debiting time is slower than a server-side e-wallet system. The two kinds of NetPay broker and "hard-coded" NetPay vendor which are server-side e-wallet and client-side e-wallet were prototyped in order to carry out an evaluation of NetPay.

We have designed a new component-based NetPay vendor system for encapsulating micro-payment support for existing web-based applications. We use an example web-based application, an E-journal portal system that we have developed independently for a teaching project. The reusable NetPay components are plugged into the E-journal site to enhance it with micro-payment support with minimal or no code changes. There are three main ways to integrate the NetPay user interface facilities: (1) modify the existing system web pages to incorporate NetPay information; (2) generate web pages that display the existing system pages in frames and make appropriate interactions with NetPay EJB components; and (3) generate proxy web pages that interact with NetPay session beans and redirect access to the original web pages. In this thesis we show examples of the E-journal extended using the first approach. The NetPay system components are designed, implemented, plugged into the E-journal example system, and deployed to a J2EE server.

For the NetPay prototypes we designed three types of evaluation: a usability evaluation, to assess the users perception of NetPay-provided features, and to compare these against conventional subscription-based payment; a performance impact evaluation, to determine if adding NetPay micro-payment to typical vendor web servers would be viable for large transaction loading; a qualitative assessment, to determine how well our model and prototypes compare using some common assessment criteria. We analyse the results from

these three evaluation approaches to determine if (1) NetPay is usable as far as target users are concerned; (2) the performance overhead of NetPay micro-payment would be acceptable to information vendors; and (3) that NetPay meets the requirements for a micro-payment system for E-tailing applications as outlined above.

1.3 Thesis Outline

This thesis focuses on the details of micro-payment systems and their implementations. The structure is as follows.

- Chapter 2 presents the requirements for a general Electronic Payment System. It also presents previously proposed payment protocols in detail and discusses their advantages and disadvantages.

- Chapter 3 introduces a micro-payment protocol we propose called NetPay. This chapter contains a detailed description of the protocol, and explanation of the protocol with illustrations.

- Chapter 4 shows an example scenario of using a micro-payment – an on-line newspaper. It compares using NetPay approach for e-commerce and several other electronic payment models.

- Chapter 5 describes three kinds of NetPay-based system designs and implementations. These include the NetPay broker and "hard-coded" NetPay vendor that are completed in order to carry out the evaluation of NetPay.

- Chapter 6 describes a component-based NetPay vendor prototype's architecture and design and illustrates an E-journal system after plugging in these components.

- The evaluation of usability, performance, and a qualitative comparison of NetPay prototypes are discussed in Chapter 7.

- Chapter 8 gives a general discussion about micro-payments, describes the contributions of this research and provides an outline of future work.

Chapter 2

Background and Prior Work

Electronic commerce has been a very active field of study in recent years. World-wide proliferation of the Internet led to the birth of electronic commerce [6], a business environment that allows the transfer of electronic payments as well as transactional information via the Internet. There are many different protocols for electronic payment systems that can work across the Internet and most likely many thousands of sites using or trying to use these methods. Despite the differences in implementation, they are all based on a few important technologies. This chapter will briefly classify electronic payment systems and outline some cryptographic primitives required for the understanding of current payment protocols. The typical macro-payment architectures and models and several different micro-payment models are introduced to discuss their various advantages and disadvantages. Of course it is impossible to illustrate all the web application implementation technologies; here we just introduce the main component infrastructure and development technologies used in our work.

2.1 Overview

Electronic fund transfers have been around before the coming of the World Wide Web (WWW) [15]. In the early stage, financial institutions used private communication channels to transfer money orders between one another. This quickly expanded to enable large commercial partners to transfer fund with a bank as an intermediary among them. The term "Electronic Data Interexchange (EDI)" emerged and was formalized into a set of clearly defined standards. EDI defines typical business transactions such as purchase orders, invoices, account balances etc. [36, 15]. It is suited for business-to-business (B2B) communication by exchanging standard business data according to agreed formats between computers. However there were many standards developed for document formats and protocols for EDI.

With the advent of the WWW, the Internet has become a platform that an increasing number of companies turn to sell product innovations. Companies could establish virtual

shops to sell products directly to the customer over the web in order to increase their income. But the supporting electronic payment technologies were not available. The first approach to address this problem was to send unencrypted credit card numbers to the vendor across the network via web forms [18]. This is not secure for high-value transactions. For low-value transactions such as paying for web pages or delivering electronic content (news, stock quotes, etc.) these methods were not applicable either, because the value of the transmitted information would be too small to pay by credit card due to processing fees [82]. So different electronic payment protocols are needed for different business cases.

2.2 Main Properties for Electronic Payment Systems

Electronic payment solutions can be assessed by the key properties of security, anonymity, and scalability [49, 63, 77]. The following sections briefly describe these properties.

2.2.1 Security

Security remains one of the important obstacles to the general acceptance of electronic payment. Since payments involve actual money, payment systems will be a prime target for criminals. Because Internet services are provided today on networks that relatively open, the infrastructure supporting electronic commerce must be usable and resistant to attacks in an environment where eavesdropping and modification of messages is easy [49, 77].

2.2.2 Anonymity

A fundamental property of physical cash is that the relationship between customers and their purchases is untraceable. This means that cash payment systems do not allow payments to be traced without compromising the system's security.

In conventional purchases we have different levels of anonymity based on the purchase and method of payment. Most of money is spent in ways where there is no anonymity whatsoever. A lot of the modern day to day purchases are done with charge cards, credit cards or cheque based payment systems that have usually compromised the security by giving full anonymity only to the vendors, while the customer has been offered only partial

anonymity [62]. It is still possible to pay for most things with real cash which is very hard to trace and fully anonymous. However receipts reduce the level of anonymity associated with cash purchases. Kleiner [48] presents several good arguments both for and against anonymity of the buyer in electronic commerce. Therefore we can conclude that it is very likely that payment systems with different levels of anonymity will be wanted [77].

2.2.3 Scalability

As commercial use of the Internet grows, the demands placed on payment servers will grow too. The payment infrastructure as a whole must be able to handle the addition of users and vendors without suffering a noticeable loss of performance. The existence of central servers through which all transactions must be processed will limit the scale of the system [49]. The payment infrastructure must support multiple servers, distributed across the network.

Payment systems must be able to handle the addition of users and load in a certain range without negative impact on performance [5]. Also the cryptographic mechanism used to detect double spending directly affects scalability. For example a payment system using public key algorithms there may prove too expensive or too slow for the application.

2.3 Classification of Payment Systems

Currently there are a number of different payment systems at various stages of development [66]. These range from proposals in the research literature to systems currently on commercial trial.

2.3.1 On-line vs. Off-line Operation

Payment systems are often classified as either on-line or off-line [77]. Examples of on-line payment system include DigiCash [16], NetBill [20]. In an on-line payment system, every payment needs to be authorized by the bank that issued the coin in order to prevent double spending by the customers. This introduces a central bottleneck. A single point of transaction processing will raise the risk of system failure, reducing system reliability and will also cause transaction traffic congestion, increasing payment latency [49]. It also raises

the cost of transactions, and imposes a minimum cost per transaction, as the bank is faced with the real cost of authorizing each transaction.

Protocols that do not rely on a third party to guard against double spending are called offline payment protocols, e.g. Agora [34], MPTP [35], Mini-pay [40], PayWord and MicroMint [72]. Typically, these protocols are credit based, since the purchase is made available to the customer before the customer is debited with the payment. In a credit-based offline payment system, there is often no protection mechanism to prevent a customer from double spending, and spending more than the balance in their account (overspending). Double spending is usually detected at the time of the clearing process, when the vendor turns in the received coins to their respective banks. Once double spending is detected, typical ways to resolve are that the customer is penalized and/or expelled.

2.3.2 Software vs. Hardware Solution

Electronic payment systems prevent double spending and fraud in basically two different ways: hardware and software. The software approach is to structure the electronic money and cryptographic protocols to reveal the identity of the double spender by the time the piece of e-money makes it back to the bank. Most software implementations for payment systems are developed in the form of digital wallets which allow access to a user's money by being a digital repository for credit card numbers, ATM card and digital certificates [72] [40, 54]. These wallets are typically embedded in web browsers and store private information, as would a physical wallet. Users can decide what information will be stored in the wallet and to what parties the information can be released. Millicent has been used in Japan from 1999 [95].

Some electronic payment systems rely on tamper-proof hardware to enhance software solution. For example smart card is embedded microchips that record transaction values and available funds [42, 58]. One key advantage of smart cards is that they allow direct transfer of electronic cash between private users. The disadvantage is the cost of the card reader which is $20-$30 plus the cost of installation per customer.

2.3.3 Macro-payment vs. Micro-payment

Macro-payment systems are used by most E-commerce systems today. These typically use credit card debiting, digital cash or real-time bank transfers, where a customer pays for products or services before or at the time of delivery [39, 93, 16, 20]. Such systems typically use complex encryption technologies and require communications with an authorisation server to request and confirm payment. This model suits low-to-medium volume transactions of medium-to-high value e.g. books, food, office stationary, home appliances, toys and so on. The standard macro-payment methods cannot be effectively or efficiently applied for buying inexpensive information goods, like single articles of an on-line newspaper, because transaction costs are too high [34, 72].

An Internet micro-payment system would allow small amounts of money to be spent at web sites in exchange for content or services [3, 22, 40, 72]. Micro-payment system's designs are usually quite different from the existing "macro-payment" systems, since micro-payment systems must be very simple, secure, and efficient, with a very low overhead cost per transaction. This must also be taken into consideration for transaction security: high security leads to high costs and computation time. For micro-payments lower security may need to be provided [49]. This means lower security than on-line, “real time” authorisation of payment i.e. the payments may be off-line, credit-based purchase where the security level is not compromised by encryption technology. For credit-based, vendors have to accept on faith that they will be paid by broker, there is no double-spending of the electronic money used.

2.4 Cryptography Technologies

Cryptography refers to the technologies used for ensuring privacy and security on the Internet. Both the host server and the user’s PC can use the cryptographical computational methods to generate the codes used to encrypt data so that it can only be read by a recipient who has received the correct code or key to decrypt it back into its original form. By using long numbers and complex mathematical formulae to generate the keys used to secure data in cryptography technologies, a high level of security can be achieved. We briefly outline some cryptographic techniques required for the understanding of current payment protocol.

The three common technologies are symmetric cryptography, public key cryptography, and hash functions [43, 56, 78].

2.4.1 Symmetric Cryptography

In a symmetric cryptography the two parties use a common secret key that is used to encrypt and decrypt data between them. The information sent could only be read by the parties that know the key. The most widely used algorithm in this category is the Data Encryption Standard (DES) which uses a key of 64 bits [59]. The problem with the use of symmetric cryptography is communication between two parties in a secure network [56]. The real world problem is much more severe in an open network, where parties never had any kind relationship before they wish to enter into a communication. The public key approach was proposed in order to solve the problem highlighted above.

2.4.2 Public Key Cryptography

In public key cryptography messages are encrypted using one key, and they can be decrypted using another [56]. Usually one key, which called the “public key”, is open over the network and the other which called the “private key” is kept secret. Using these mechanisms two important facilities can be created that depend on whether the public key is as the encryption key or decryption key. Let us suppose that a party is Bob. If any party encrypt the message using Bob’s public key, only Bob can read it by using his private key, so any party can send a confidential message to Bob. On the other hand, if Bob encrypts the message using his private key; any party can read it and is assured that it was sent by Bob.

The examples above show how public key systems can be used for two purposes: encrypting a message with the recipient’s public key to achieve confidentiality or encrypting a message with the sender’s secret key to achieve message authentication. The standard algorithm for implementing public key cryptography can be used for both encryption and authentication and is called RSA algorithm. It is named after its inventors, Rirest, Shamir, and Adlemen who developed it in 1978 [70]. Its security is based on the difficulty of factoring very large numbers. Both of these involve applying the public key algorithm to the whole message. The public key algorithms in use today are

computationally intensive they may be too expensive or too slow for micro-payment applications [72].

2.4.3 Cryptographic Hash Function

One of the fundamental technologies in cryptography is the cryptographic hash function, also called a one-way hash function. A one way hash function is easy to compute but difficult to reverse. In mathematical terms, given a value x, it is easy to get $y=h(x)$ where h is a one-way hash function, but, given y, it is not feasible to compute the corresponding $x=h^{-1}(y)$. The MD5 (Message Digest) algorithm [71] is one of a series (including MD2 and MD4) of message hash algorithms developed by Ron Rivest. It involves appending a length field to a message and padding it up to a multiple of 512 bit blocks.

One key factor that must be supported in a micro-payment system is to allow a high number of low value transactions. Obviously, the transaction mechanism cost needs to be significantly lower than the value of the transaction if a transaction is to make economical sense. It depends on which cryptographic technologies and payment protocol are used in the system. Table 2-1 gives some estimated figures comparing the number of public-key and hash operations, and network connections that can be performed per second on a typical workstation [72].

Table 2-1 Comparison of computational speed of cryptographic and network operations on a typical workstation - From [72]

Operation	No. Per Second
Public-key signature generation (1,024-bit RSA)	2
Public-key signature verification (1,024-bit RSA)	200
One-way hash values compute (MD5/SHA)	20,000

Fast hash functions are more suitable for micro-payment systems, where speed is also important, than the much slower public key cryptography used in most macro-payment systems. However, some micro-payment schemes require using public key cryptography in

order to minimize communication costs such as PayWord and our protocol. It is therefore important to greatly reduce the number of these operations. A “payword chain” is generated by using a one way hash function and is going to be used to represent a set of E-coins in the NetPay system.

2.5 Macro-payment Systems

Within macro-payment systems there are three distinct payment methods: credit card based digital cash, and electronic checks (account based).

2.5.1 Credit-card Based

Credit based payment systems, such as SET [93] and CyberCash [21], are both online and post paid payment by credit card. **CyberCash** (CyberCash Inc., Reston, VA) provides customer software, vendor software and a gateway to support the secure communication of credit card transaction over the Internet. Customers download *CyberCash Wallets.* Vendors use the *Secure Merchant Payment System (SMPS).* The system relies on the use of existing financial networks that are totally independent of the Internet for communicating between the CyberCash Gateway Server and the banks or credit institutions.

Figure 2-1 represents a model how credit cards can be used in a secure way across the Internet. The protocol is called CyberCash that provides customer software, vendor software and a gateway to support the secure communication of credit card transaction over the Internet.

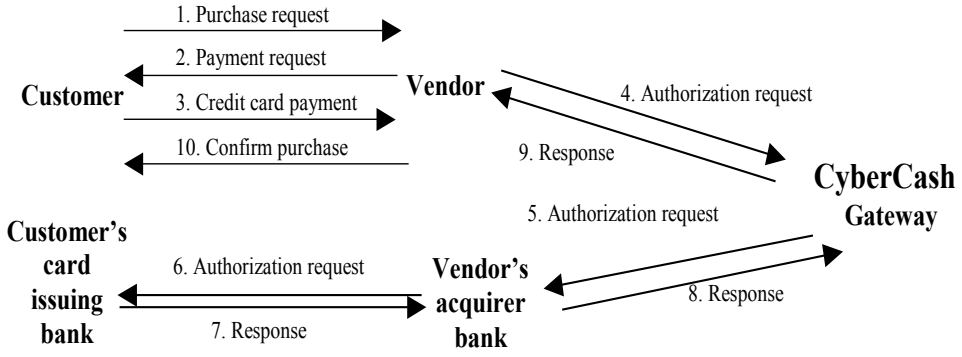


Figure 2-1 The CyberCash system model

The following is an outline of how CyberCash works:

1. The customer browses the home page of the vendor's online site and adds desired items to the cart.

2. When the customer wishes to pay, the vendor's server sends the wallet software a payment request message that describes the purchase and indicates which credit cards the vendor accepts.

3. A credit card payment message, including a signed and encrypted description of the transaction, along with the customer's credit card number is sent back to the vendor.

4. The vendor forwards the payment message with his signature and encrypted description of the transaction to the CyberCash gateway.

5. CyberCash decrypts and compares the two messages and their signatures.

6. If they match, it sends an authorization request to the vendor's bank that forwards the message to the customer's bank.

7, 8, and 9. The customer's bank sends an approval or denial to CyberCash pass through the vendor's bank and the CyberCash gateway.

10. The vendor's software confirms the purchase to the customer's wallet software when the vendor receives an approval message.

2.5.2 Electronic Cash-based

There are many payment systems based on e-cash payment, such as DigiCash which is an online and prepaid payment system [16], and CAFÉ which is an offline and prepaid payment system [9]. DigiCash is a fully anonymous electronic cash system using blind signature techniques which invented by David Chaum. DigiCash is implemented using RSA public-key cryptography. Every customer in the system has their own public/private key pair. Special client and vendor software is required to use the DigiCash system. The

client software is called a "cyberwallet" and is responsible for withdrawing coins from and depositing coins into a bank, and paying coins to or receiving coins from a vendor.

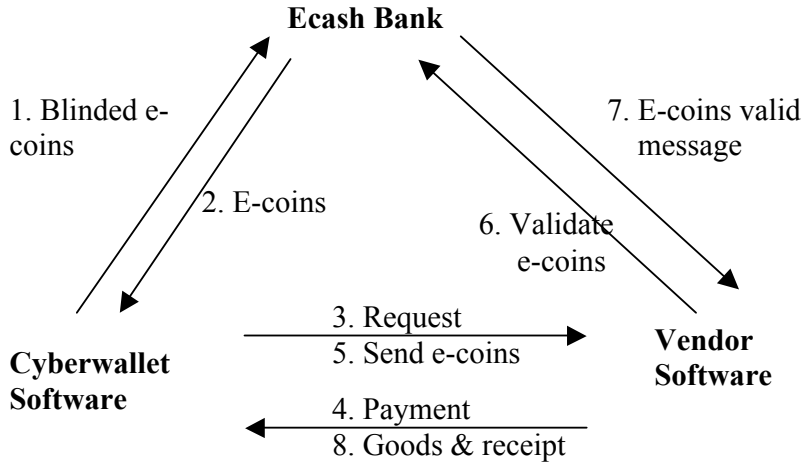


Figure 2-2 The DigiCash system model

The following outline how DigiCash works:

1. To make a withdrawal from the bank, the customer's cyberwallet software generates random serial numbers which are used in the blind signature technique [17] for the e-coins. The blinded e-coins are then packaged into a message, digitally signed with the customer's private key, encrypted with the bank's public key, and sent to the bank.

2. The bank checks the signature, debits the signature owner's account, validates the e-coins, and returns the e-coins to the customer. The customer e-wallet un-blind the e-coins by dividing out the blinding factor. Since the bank couldn't see the serial numbers on the e-coins, the cash is fully anonymous.

3. The customer sends a purchase request to the vendor.

4. The vendor software contacts the customer's cyberwallet asking for payments.

5. The cyberwallet software encrypts the e-coins with the vendor's public key and sends them to the vendor.

6. The vendor must verify the e-coins which are valid and have not been double spent with the bank.

7. The bank validates the e-coins by checking the serial numbers with the large on-line database of all the serial numbers ever spent and returned to the bank. If the e-coins are valid, the value of the e-coins is credited to the vendor's account. The e-coins are destroyed and the serial numbers added to the database of spent coins.

8. The vendor sends the purchased goods and receipt to the customer.

2.5.3 Account-based

The NetCheque payment system developed at the Information Sciences Institute of the University of Southern California is a distributed accounting service consisting of a hierarchy of NetCheque servers (banks) that are used to clear cheques and settle interbank accounts [62]. Figure 2-3 describes such a hierarchy.

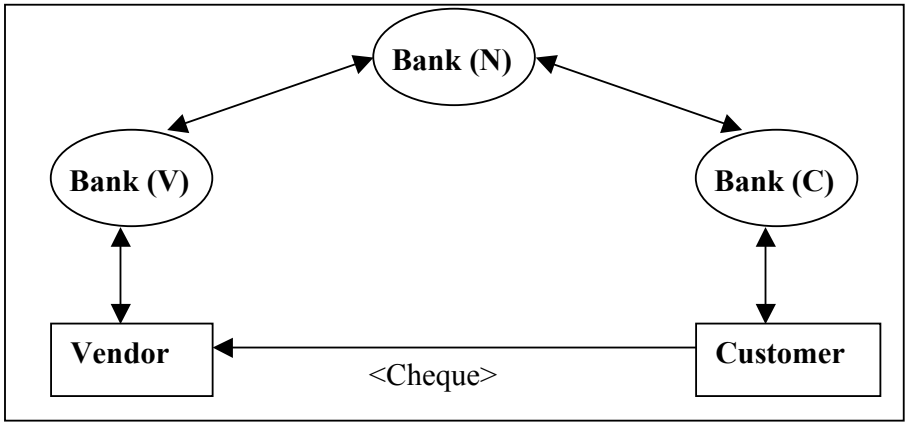


Figure 2-3 Hierarchy of NetCheque servers

A NetCheque account is similar to a conventional bank account against which account holders can write electronic cheques. An electronic cheque is like a conventional paper cheque in that it contains the customer's signature. Unlike a paper cheque, it has to also be endorsed by the vendor before the cheque is paid. Kerberos tickets [17] were used in the NetCheque system to create electronic signature and endorse cheques. Kerberos is based on symmetric key cryptography and is thus more computationally efficient than schemes based

on public key algorithms. A NetCheque contains seven parts which are amount, unit of currency, date, account number, payee(s), customer signature and endorsement(s) by the vendor and bank(s). To write a cheque, a customer needs to do the following things:

1. The customer generates the cleartext portion of the cheque, specifying an account against which the cheque is to be drawn, the amount, the currency unit, and the payee. The account number and date are read from the customer's chequebook file.
2. The customer obtains a ticket from a Kerberos server that is used to authenticate the customer to the customer's bank and allows the customer to share a session key with the bank.
3. The customer generates a chequesum on the contents of the cheque and places it in an authenticator.
4. The customer encrypts the authenticator with the session key and appends the ticket and the authenticator to the cheque.

The cheque can be send to the vendor through electronic mail over an unsecured network. To deposit the cheque, a vendor and bank(s) need to do following things:

1. On receiving a payment the vendor reads the cleartextparts of the cheque, obtains a Kerberos ticket to be used with the payer's bank, generates an authenticator endorsing the cheque in the name of payee for deposit only into the payee's account, and appends the endorsement and the ticket to the cheque. An encrypted connection is opened to the payee's bank and the cheque is deposited.
2. If the payee and the payer both use the same bank, the response will indicate whether the cheque is cleared.
3. If different banks are used, the vendor's bank sends an indication to the vendor that the cheque has been for collection.
4. If the cheque has to be cleared through multiple banks, each bank attached its own endorsement to the cheque.
5. Once the cheque has been cleared by the customer's bank, the attached endorsements can be used to trace back the path to the vendor's account and eventually credit vendor's account for the same.

2.5.4 Discussion

Macro-payment systems provide high level security, because they are generally on-line systems. They can ensure the money is there and has been transferred to vendor before transaction performed, but all on-line payment schemes introduce a central bottleneck, a single point of failure that increases payment latency. It also raises the cost of the transaction, and imposes a minimum cost per transaction, as the bank is faced with the real cost of authorizing each transaction. As a result, the macro-payment systems are generally not suitable for high-volume, low value payment transactions.

2.6 Micro-payment Systems

Unlike macro-payment systems, micro-payment systems focus on supporting low-value high volume e-payment transactions [88]. We review the key concepts of several micro-payment systems below, identifying their key strengths and weaknesses.

2.6.1 Millicent

Millicent [54], a micro-payment system implemented by Digital Equipment Corp, now owned by Compaq, went live in June 1999 in Japan, with wallets starting at 1000 yen and payments as small 5 yen. Millicent does not fall into either the online or the offline category, but rather is a distributed allocation of funds to vendors, who locally authorize payments. Thus it is nearly an automated account based system.

Millicent introduces new kind of currency – a “scrip”, which is digital money that is issued by a single vendor. Scrip has a value, just as cash does, but it has value only when spent with specific vendor. Scrip consists of a signed message attesting that a particular serial number holds a particular value. In addition to the necessary contents of electronic cash the scrip will also hold an expiration date and information on the particular vendor with whom the scrip can be redeemed.

The system includes customers (C), vendors (V), and a broker (B) and assumes that the broker is honest and is trusted by both the customers and the vendors. The following presents the steps for a complete Millicent session.

A.

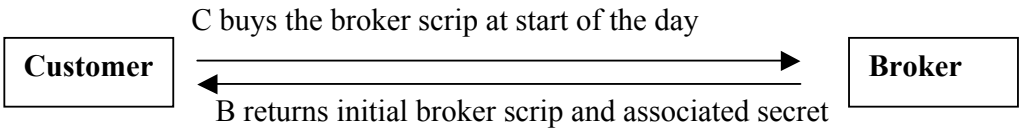


Figure 2-4 Customer buys broker scrip

If customers could buy scrip directly from vendor, Millicent would commit large accounts with vendors and tying customers down to large information providers. To avoid such a disadvantage, Millicent allows the scrip brokers to buy contracts from vendors to produce scrip for customers. This means that the customer must in nearly always be able to connect to the broker in order to be sure of the ability to make payment.

B1.

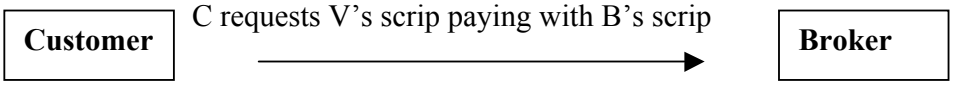


Figure 2-5 Customer buys vendor scrip

B2. There are three models in which the broker gets the vender scrip.

- *Scrip warehouse model* assumes a casual relationship between the broker and vendor.

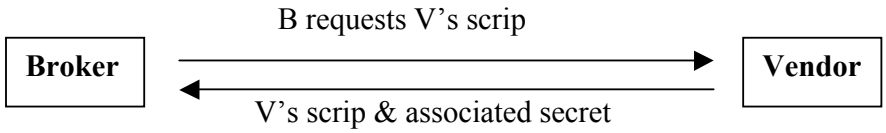


Figure 2-6 Broker requests vendor scrip

- *Licensed scrip production model* assumes a substantial and long-lasting relationship between the broker and vendor. If a broker buys a lot of scrip for a specific vendor, the vendor sells the right to the broker to generate vendor scrip that the vendor can validate and accept. This model is more efficient for the vendor and broker than the scrip warehouse model. There is less communication because the broker does not need to contact the vendor every time when the customer asks for vendor scrip.

- *Multiple brokers' model* assumes a relation between brokers, but requires no relationship between the vendor and customer's broker.

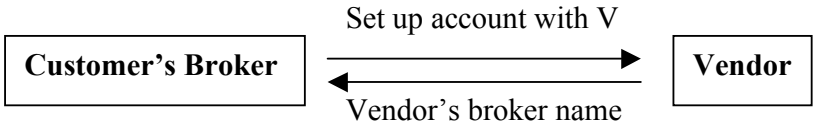


Figure 2-7 Customer's broker creates account with vendor

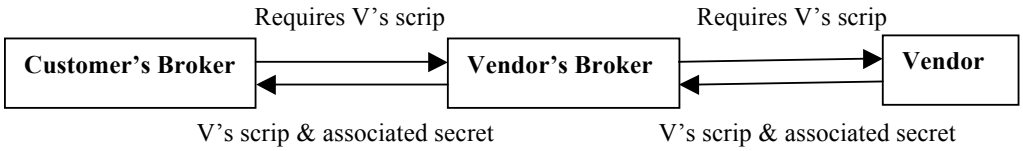


Figure 2-8 Customer's broker requires vendor scrip with vendor

B3.

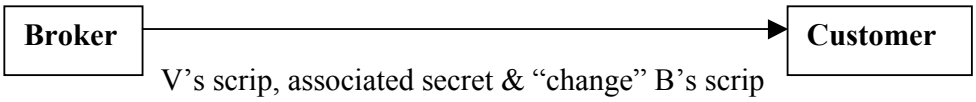


Figure 2-9 Broker sends scrip to customer

C.

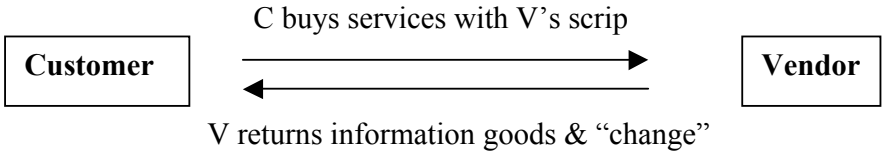


Figure 2-10 Customer buys services

The customer continues using the change to make more purchases with this particular vendor. When the customer wants to purchase with another new vendor, he needs to repeat step B and step C.

Millicent uses no public-key cryptography and is optimized for repeated micro-payments to the same vendor. Its distributed approach allows a payment to be validated, and double spending prevented without the overhead of contacting the broker on-line during purchase. Its main drawbacks include:

- The broker must be on-line whenever the customer wishes to interact with the new vendor.
- The customer must nearly always be able to connect to the broker in order to be sure of the ability to make payments, so that the system places a heavy real-time burden on the broker.
- The vendor scrip is vendor-specific and has no value to another vendor.
- The transaction is complex in this system when the customer and the vendor have different brokers.

2.6.2 Mpay

The Mpay micro-payment system proposal from IBM was previously named Minipay [40]**.** The system has been in internal tests and is also available to anyone in the Internet for testing purposes from 1998 [49]. Mpay is very similar to the billing mechanism of the third party value added services of the phone networks. The customer deals with his issuer, the vendor deals with his acquirer and the issuer and the acquirer settles the accounts. The system is suitable for selling inexpensive information and other similar services that are usually delivered on-line. This is reflected in the protocol.

There are five parties which are the customer, the vendor, the customer's billing system (Internet Access Provider--IAP), the vendor's billing system (ISP or bank), and an exchange connecting the IAP to the ISP or bank which would be a bank or similar financial institution in Mpay system.

1. *Daily certificate request*

In the Mpay system, the customer connects every day to his issuer to receive a *daily certificate.* This certificate signed by the issuer whose public key is known by all vendors. The certificate states that the customer has an account and tells the *recommended offline limit* for the daily purchases.

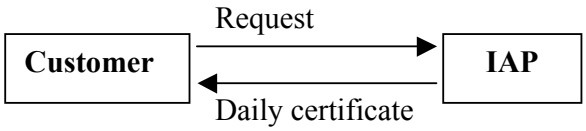


Figure 2-11 Customer requests daily certificate with IAP

2. *Transaction (online)*

The customer clicks on a specific type of a tag in his browser. The system encodes the cost, daily certificate and other necessary information and sends it to the server. This enables the customer to send the payment at the same time as the query for the vendor. The payment order is piggybacked on the request. The vendor verifies the signature of the IAP on the certificate. If the daily limit is not exceeded, the vendor immediately responds to the request. However if the daily limit is exceeded, the vendor connects to the IAP by sending an extra spending request. The IAP may or may not send an extra spending reply to the vendor according to the customer's record.

Figure 2-12 Customer buys services with vendor

3. *Daily deposit*

At a fixed period the vendor sends all the payment orders from all customers in a single, signed deposit message to the ISP. The ISP sorts the payment orders based on the IAP of each of the customers and sends the single signed deposit message to them.

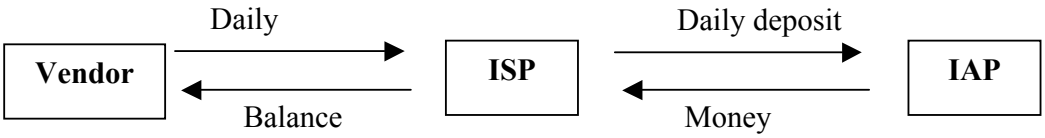


Figure 2-13 Vendor redeems payment orders

4. *Daily process*

At the end of the day or at the first purchase the next day, the customer contacts the IAP for their daily process. In this process the customer and the IAP compare their records for the previous day, all matching records are erased and replace them with a summarized and signed document of purchases and the balance.

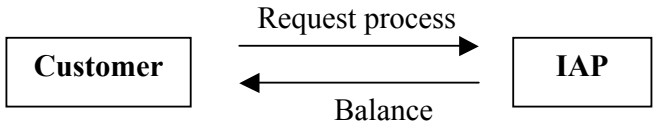


Figure 2-14 Customer confirms transactions with IAP

Mpay is based on a notational model and has off-line capability in its daily certificate. Mpay only uses one or no public key operation per purchase, so the transaction cost is low. Mpay system supports multi-currencies from the outset. An item to be bought can have its price described as a list of currencies and prices in these currencies. There is no extra communication required in the system, because of the payment order piggybacked on the information request. It is a real 'pay per click' system; the customer can ease to use it purchasing information goods.

Although Mpay is flexible, convenient and low-overhead payment system, the major shortcoming is that the customer can pay nothing to the IAP who still needs to pay the ISP after purchasing goods. The IAP can protect itself by requiring a deposit from the customer, and by terminating Mpay and other services (e.g. Internet access), but the customer is still free to spend for a full day. If a worldwide there are millions of online customers to use the payment system and a lot of IAPs, it seems to be difficult to terminate Mpay and Internet access to such customers. Even though the termination of Mpay and Internet access is possible to some customers, this increases initialization cost, thus finally driving up the cost of the system. There is also the question of who pays in case of fraud. The vendor will not receive the payment and a high level of fraud will drive up the cost of the system. Furthermore, the protocol is not fully anonymous due to the after-the-fact policing requirements. Thus the IAP is able to collect a complete purchase profile of their customers.

2.6.3 PayWord

PayWord is a micro-payment protocol proposed by Ron Rivest (MIT Laboratory for Computer Science, MA, USA) in 1996 [72]. The protocol aims to reduce the number of public key operations required per payment by using hash functions, which are faster. In PayWord customers generate their own "coins," or paywords, which are sent to vendors and then verified by brokers.

The model involves three parties, which are customer, vendor, and broker and assumes that the broker is honest and is trusted by both the customers and the vendors. The following presents the steps for a complete PayWord protocol session.

1. *PayWord certificate request*

In the beginning of the transaction, the customer establishes an account with a broker, who issues a digitally signed PayWord certificate, which contains identity and public key of the customer and expiration date. The certificate authorizes the customer to make PayWord chair (e-coins) and ensures vendors that the customer's paywords are redeemable by the broker.

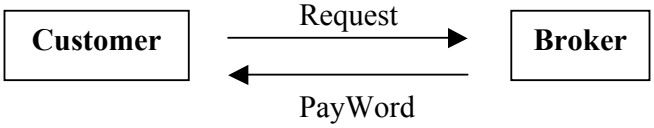


Figure 2-15 Customer buys Payword chain

2. *Transaction*

When a user wishes to make a purchase at a vendor for the first time in a day, he first randomly picks a payword seed w_n, where n is the number of e-coins that a customer would purchase at a typical vendor in a day. The customer then computes a payword chain by repeatedly hashing w_n: $w_{i-1} = h(w_i)$, where $i=1,\ldots, n$. The customer then sends the digitally signed commitment which includes w_0 the root of the payword chair and the certificate to the vendor. It is used to show the customer's intentions of spending paywords there.

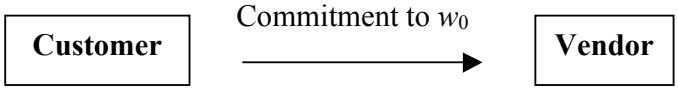


Figure 2-16 Customer sends commitment to vendor

To make *m* cents payment, the customer sends w_1 through w_m where m is the number of paywords the customer wishes to spend and the requirement of the information goods to the vendor. The vendor can easily verify this chain by hashing w_m *m* times until he reaches w_0. The vendor sends the information goods to the customer.

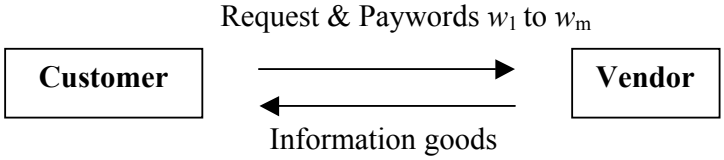


Figure 2-17 Customer buys information goods

3. *Redeeming*

At the end of each day, the vendor sends the customer's commitment and the highest payword spent to the broker. The broker verifies the paywords using the root w_0 and the customer's signature. If they are valid, the broker debits the spent amount from the customer's account and pays the vendor.

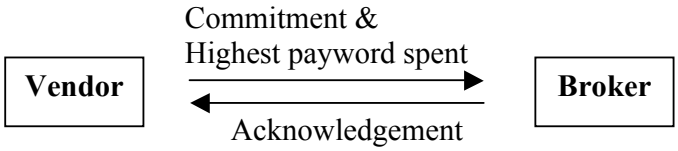


Figure 2-18 Vendor redeems paywords with broker

PayWord is an off-line system. The customer system only needs to contact the broker at the beginning of each certificate lifetime in order to obtain a new-signed certificate. The system aims to minimize the number of public key operations required per payment using hash operations instead whenever possible. It is credit-based scheme where a user's account is not debited until some time after purchases. This provides more opportunity for fraud since a large number of purchases can be made against an account with insufficient funds. The e-

coin (paywords) in the system is customer and vendor specific and the paywords in the chain have no value to another vendor.

2.6.4 PayWord-based Micro-payment Protocols

The area of micro-payment on the Internet has attracted much attention recently. There are several new micro-payment systems that are based on Payword micro-payment protocol. These systems can be classified as credit-based and debit-based.

PayFair [96] is a debit-based micro-payment system that employs some parts of the Payword scheme. A payword chain purchased from the broker will be bound to a specific vendor. Many payword chains can be purchased in advance from the broker and stored in the customer's machine. This improves the performance and also the complexity of sending another new payword chain to any specific vendor. There is no digital-signature required for witness of the payment promise in the system.

NMP [46] is a credit-based protocol that improves the fairness for customers from the Payword protocol. In the protocol, the customer generates a payword pair chain (Wi, Wi') and sends the first payword Wi to the vendor first. The i-th information goods is sent to the customer when the payword is valid, then the Wi' is sent to the vendor.

The Payword-based micro-payment systems described above share a common disadvantage which is that they are vendor specific. The e-coin (paywords) in the systems is vendor specific and the paywords in the chain have no value to another vendor.

2.6.5 Payment Systems Comparison

Ten system characteristics have been selected to evaluate micro-payment systems from different points of views [49]. The characteristics concern with independence, design, trust, transferability, security, privacy, divisibility, multi currency, ease of use, and on-line/off-line. We emphasize the last six, since the first four are closely related to the general criteria and features of the systems.

We discussed several micro-payment protocols in the previous sections, which mainly formed on the strength and weakness of the micro-payment systems. The comparison below is considered for micro-payment systems from the perspectives of reading the on-line newspaper (Customers), newspaper vendors, and micro-payment brokers or macro-payment authorisers. Table 2-2 lists the results of evaluating these various E-commerce payment system models. The evaluation criteria we used include:

Ease of use – A protocol provides ability for customers to use a certain system smoothly and easily. It is an important issue in micro-payments world. For maximum score there should be no login name and password required in a system. Customers only need to click and buy a web page. Customers should see pricing for articles clearly and seamlessly.

Security - The protocol prevents customers from double spending and any internal and external adversaries from forging coins. Since high security leads to high costs and computation time, security is also a key issue for keeping the transaction cost low.

Anonymity - The customer anonymity should be protected from vendors. Ideally it should not be possible to trace a customer's preferences and spending patterns just like with "real" coins and paper money.

Multi currency – A payment system should be able to operate with multiple currencies, by converting the currencies either inside or outside the system, since customers purchase services and information via the WWW to all over the world by using a payment system.

Robustness – the protocol is tolerant of network bottlenecks and broker/authoriser down-time. An on-line payment protocol typically leads to increase communication requirements. This in turn leads to slower processing and higher transaction cost.

Table 2-2 Comparison of E-commerce payment methods

System/ property	CyberCash	Millicent	Mpay	PayWord
Ease of use	**Low,** Customer contacts Broker every transaction.	**Medium,** Customer nearly always contacts Broker.	**High,** Customer only needs to click and see what he pays.	**Medium,** Customer generates and manages e-coins for every Vendor.
Security	**High,** the system performs checking, clearing and recording of transactions.	**Medium+,** the system prevents double spending by using Vendor-specific scrip.	**Medium,** the criminal is free to spend money to buy content for a full day.	**Low,** the system is credit-based scheme to provide more opportunity for fraud.
Anonymity	**Low,** the system records identities, exchanged amount and time of a transaction.	**Medium,** Broker knows who and where but not what. Vendors know what but not who.	**Low,** Customer's anonymity is not supported.	**Low,** Broker knows who and where but not what vendors know what and who.
Multi currency	**No,** only one currency $.	**No,** must be match with scrip.	**Yes,** converts	**Yes,** converts
Robustness	**Low**, on-line payments	**Low**, on-line payments	**High**, off-line payments	**High**, off-line payments

2.7 Internet Component Architecture

Building large, multi-tier, web-based applications requires a large number of software services, from dynamic HTML generation to business domain object persistence [11]. Internet component architecture provides a specification and set of commonly required system-level services to handle all of the complex details of components coordination. This in turn lets application developers to concentrate on business logic without concerning the complex services. Three examples of internet component architectures are: Object Management Group (OMG)'s CORBA, Sun's J2EE and Microsoft's .NET.

2.7.1 OMG's CORBA Architecture

The common Object Request Broker Architecture is a standard developed by the OMG to provide interaction among applications of different systems in heterogeneous environment [64]. CORBA's core is Object Request Broker (ORB) which locates on both the server and

the client. It acts as a central Object Bus over which the CORBA objects can communicate with each other. ORB creates and manages client/server communications between objects. ORB is in charge of the following tasks:

- searching object implementations for client requirements;
- Making these implementations prepare to accept requires;
- Transmitting all components of the requiring and answering data.

The CORBA Interface Definition Language (IDL) provides the operating system and the programming-independent interfaces to all services and components that are linked to the ORB. IDL defines the modules, interfaces and operations for the application and is not considered a programming language. Various programming languages, such as Java, C++, and Ada, supply the implementation of the interface via IDL mapping. CORBA objects can communicate over many popular networking protocols such as TCP/IP. ORBs communicate among different vendors over TCP/IP using the Internet Inter-ORB Protocol (IIOP).

To request a service, a CORBA client creates a remote object and registers it with the ORB. The generated stub converts the request which is written in the language of implementation of the client (such as Java or C++) into a remote request encoding using CORBA's IIOP Protocol. This encoded request is then sent to an ORB running on the machine where the remote object reference was obtained from. The ORB then transfers this request to a generated IDL skeleton which gives the IIOP-encoded remote request. The skeleton converts the IIOP protocol back into the target language of the remote object implementation. Finally the skeleton invokes the implementation of the remote object. When this returns, any return value is converted into IIOP returned via the ORB to the stub, which converts the value back into the target language of a remote object and return it to the client. Figure 2-19 shows the main components of the ORB architecture.

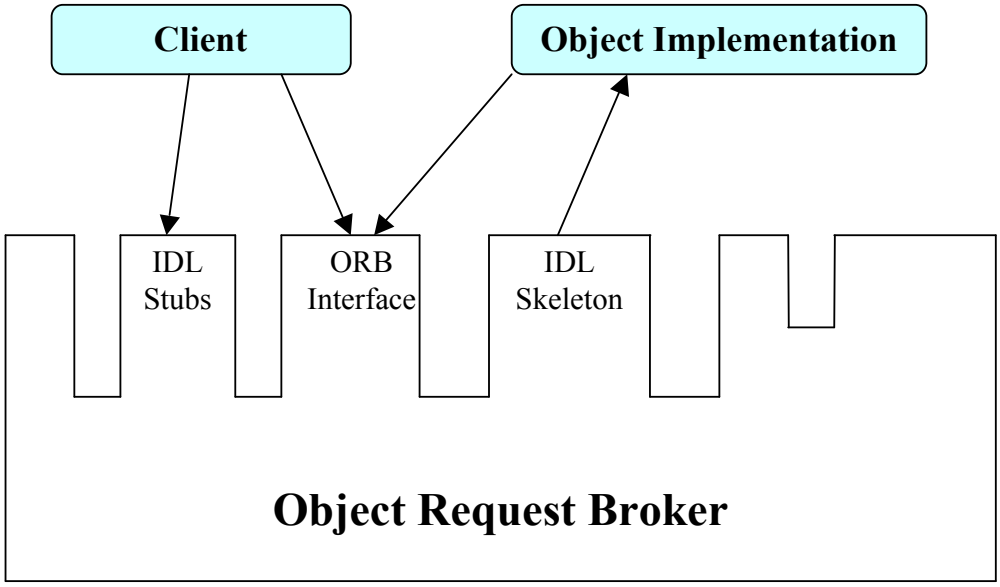


Figure 2-19 The ORB architecture

2.7.2 Sun's J2EE Environment

There are a number of Java technologies supporting the development of component-based, distributed systems [86]. These include:

- JavaBeans, which is the component model for Java. It is a set of standards for packaging Java-implemented services as components. By following this standard, tools can be built to inspect and control various properties of the component.

- Remote Method Invocation (RMI), which allows Java classes on one machine to access the services of classes on another machine.

- Java Naming and Directory Interface (JNDI), which manages the unique identification of Java classes in a distributed environment.

- Enterprise JavaBeans (EJB) is a distributed component based framework. It provides a standard means of defining server-side components, and specifies a run-time environment for hosting the components on the server side. The services include: process and thread dispatching, and scheduling; resource management; naming and directory services; network transport services; security services; and transaction

management services. There are three main types of EJBs: session beans, entity beans and message driven beans [85, 91].

At the end of 1999, Sun Microsystems announced their new enterprise solution architecture named Java 2 Enterprise Edition (J2EE) [44] which provides many new features and expand it scalability in enterprise computing to help enterprises to handle the growing fast Internet populations. J2EE platform is an architecture for developing, deploying, and executing applications in a distributed environment. It provides many services such as transaction management, security, client connectivity, and database access so developer can concentrate more on business logic rather than architecture design. The J2EE platform is a 3 or 4-tier client/server system which depends on the client being used in the system as shown in Figure 2-20.

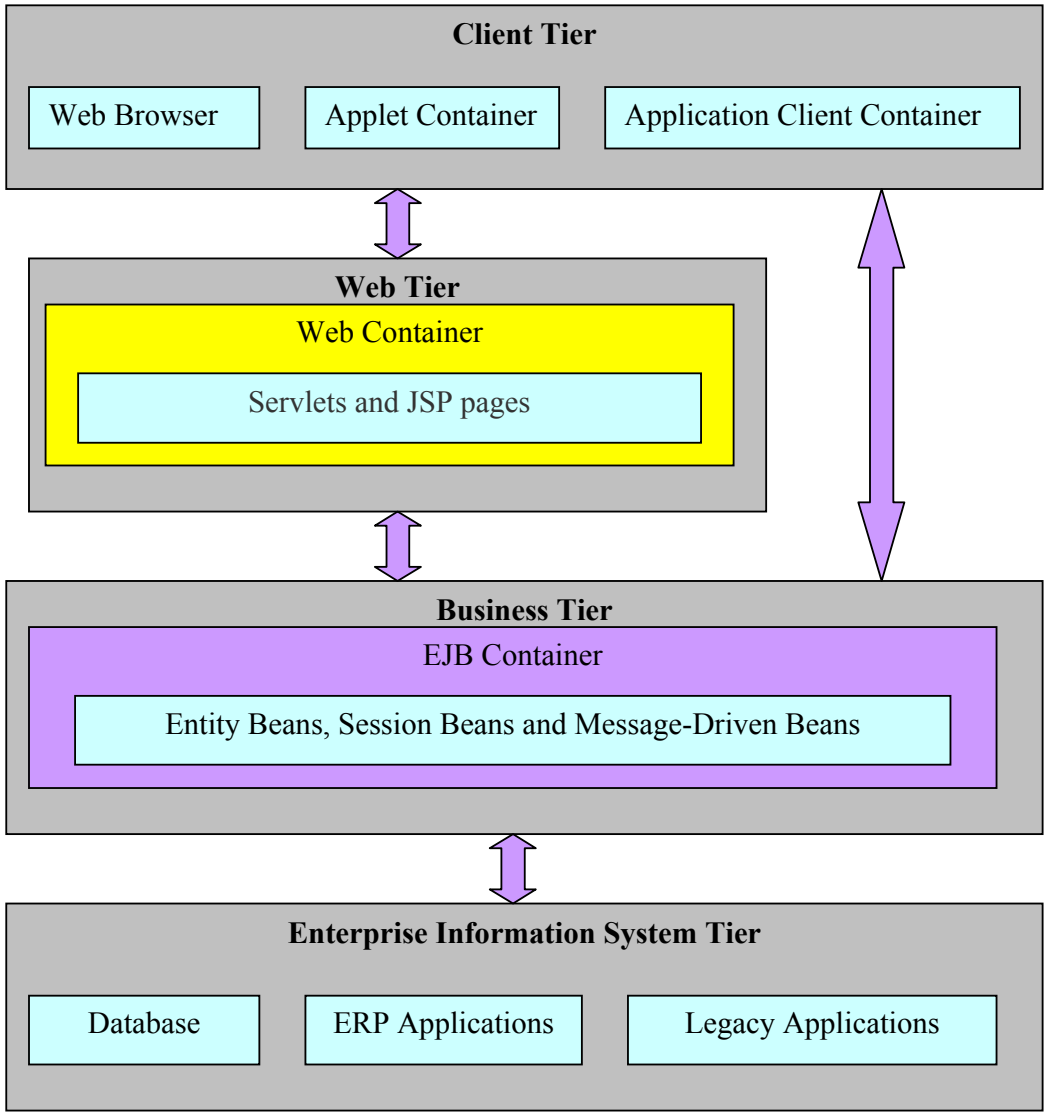


Figure 2-20 J2EE Distributed Multi-tiered applications architecture

The distributed architecture of the J2EE platform consists of the following four parts:

- **Client Tier** - The client tier is responsible for displaying the user interface. Depending on the actual program, it is a program, written in Java or any other language or a web browser. The Client tier contains the web clients, the applet container and the application clients that are executed by the users. The thin client with the web browse typically is a 4-tier system. The thick client with the standalone Java program running in an application client container typically is a 3-tier system. The intermediate client with the applet container (which in fact is most of the time a web browser) can be both a 3 and a 4-tier system, although most implementations will be a 4-tier system. In case of the application client container 3-tier system, the client tier connects directly to the business logic tier. In this case the communication between the client tier and the business tier will be RMI-IIOP based. In case of a web browser based 4-tier system, the client tier connects to the web tier using HTTP or HTTPS.

- **Web Tier** – The web tier is responsible for the presentation and application logic. The web container contains Servlets and Java Server Pages (JSP) pages that generate the appropriate HTML that can be displayed by the browser based clients. Because of the nature of web tier the components will be highly application dependent, although it is possible that presentation or application logic is shared between applications.

- **Business Tier** - The business tier is responsible for the business logic. It is possible to reuse business logic between applications in the J2EE platform. Enterprise JavaBeans (EJB) provides a standard server-side distributed component model. These enterprise beans solve or meet the needs of a particular business domain such as banking, retail etc., The EJB container of the J2EE server hosts the enterprise beans.

- **Enterprise Information System (EIS) Tier** – The EIS tier contains enterprise resources, such as database systems, legacy applications and Enterprise Resource Planning (ERP) applications.

J2EE architecture is composed by many software components. It provides many new enterprise APIs that are built in the Java 2 Platform Standard Edition. It uses Java Database connectivity (JDBC) APIs to connect database and retrieve information. All business logic

encapsulate in EJBs, reusable components that can be accessed by client program. Java Naming and Directory Interface (JNDI) provides a unified naming and directory interface for looking up service components. It also can use RMI-IIOP protocol to connect with CORBA architecture.

2.7.3 Microsoft's .NET Architecture

To enable sharing of functionality across desktop applications, Microsoft developed the Component Object Model (COM) as the basis for inter-application communication. The .NET Platform as a programming model represents the next stage in the evolution of COM, Microsoft's component platform.
.NET is a platform which uses the component as its basic building block to create a component infrastructure for web middleware [1]. The .NET Framework takes the best aspects of COM and combines them with the web computing. The developer model for building web services with Microsoft.NET is shown in Figure 2-21.

Briefly, Figure 2-21 can be explained as follows:

- **Client tier –** The client tier is either a traditional 'thick' client or a web browser connect to Active Server Pages (ASP.NET) which renders user interfaces in HTML or XHTML. Heavyweight user interfaces are built using Windows Forms.

- **Web tier -** The web tier contains ASP.NET technology that sits upon the Common Language Runtime (CLR), allowing web page development to any language that supports the CLR within the enhanced ASP environment.

- **Business tier -** The business tier of the .NET application is built using .NET managed components. This tier performs business processing and data logic. It connects to databases using Active Data Objects (ADO.NET) and existing systems using services provided by Microsoft Host Integration Server 2000, such as the COM Transaction Integrator (COM TI).

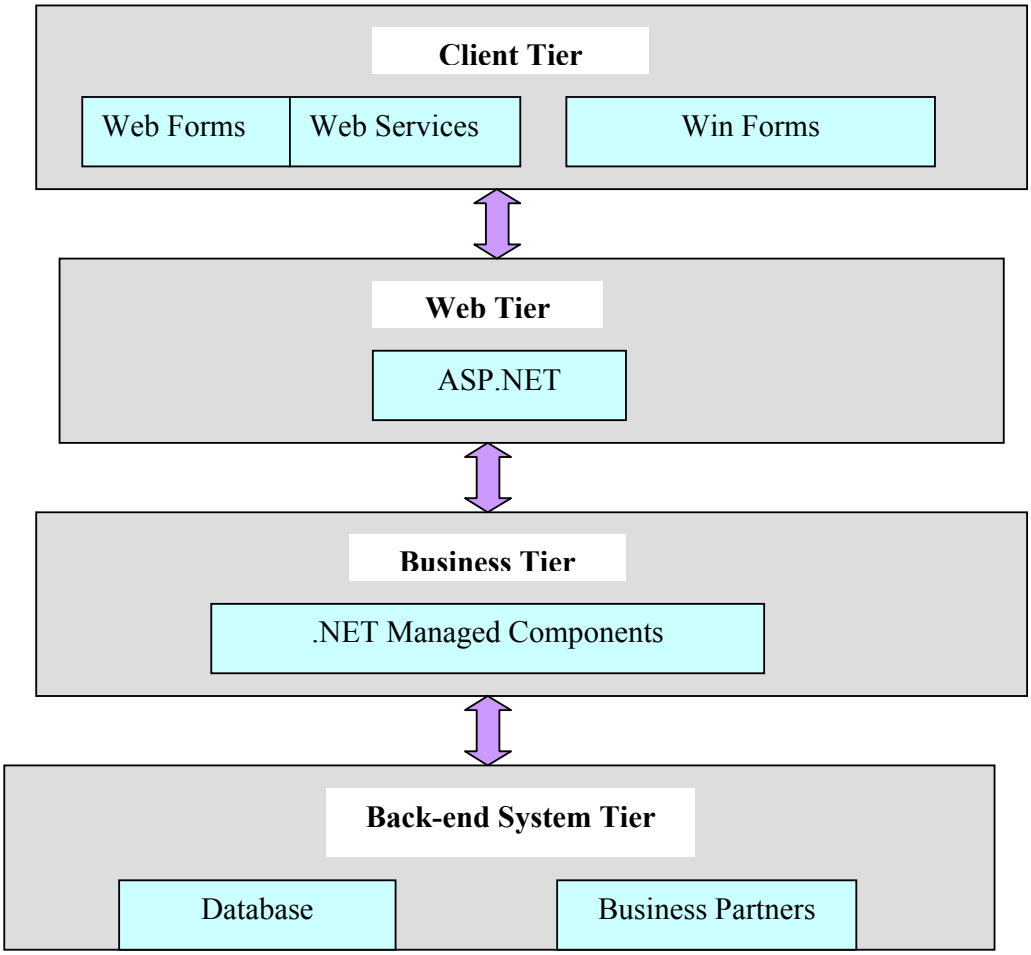


Figure 2-21 Developing web services with Microsoft.NET

- **Back-end System Tier** - The back-end system tier contains enterprise resources, such as database systems and business partner which can connect with the .NET application through web services technologies e.g. Simple Object Access Protocol (SOAP). .NET uses SQL server 2000 to connect database and retrieve information.

There is a platform limitation for .NET technology on platforms. .NET only runs on Windows, its supported hardware, and the .NET environment.

The CORBA standard has been widespread in the area of objected-oriented and distributed systems. It supports independence of the computer architectures and programming languages to be used. It can be used on different kinds of operating system platforms from mainframes to UNIX boxes to Windows machines.

J2EE platform provides a simplified approach to developing scalable and high-availability Internet/Intranet applications. It extends J2SE with many enterprise-related APIs, the Web

and the EJB component model, and runtime containers to host Web and EJB components. One of J2EE's major advantages is that most of the J2EE vendors do offer operating system portability. One of J2EE's major disadvantages is that the choice of the platform dictates (demand) the use of a single programming language [79, 80].

CORBA and J2EE are open specifications and are not products. Microsoft's .NET platform vision is a family of products. The major disadvantage of this approach is that it is limited to the Windows platform, so applications written for the .NET platform can only be run on .NET platforms. The major advantage of this approach is that the cost of developing applications is much lower, since standard business languages can be used and device independent presentation tier logic can be written [30, 92].

2.8 Web Component Technologies

Many software industries expand their business market towards the Internet/Internet environment. They all belong to web-based business applications that require database connection, user authentication, session management and dynamic HTML generation. Several methods are used to implement web application. Traditionally systems were developed using Common Gateway Interface (CGI). CGI creates a separate process for each request. When request increase, servers will be loaded with too many simultaneous process. So that CGI suffers from performance limitations [45].

A few years later, server-side Java technologies, such as Java Servlet and Java Server Pages (JSP), were proposed. Web applications using these technologies take Java's natural advantages of portability, reusability, and flexibly.

2.8.1 Java Servlets

Java Servlet [45] is a Java class that extends a J2EE-compatible Web server. Each servlet class produces dynamic content in response to service requests to one or more URLs. To execute a servlet it is necessary to connect the web server to a servlet engine and to configure the web server in such a way it can recognize which clients' requests are directed to a servlet. Servlet engine runs a Java Virtual Machine (JVM), manages servlets, and maintains HTTP session state using cookie and local data stored on the server. Generally a servlet is invoked by a post operation, executed through an HTML form. Then the servlet

executes the requested computation according to the received parameters and sends back to the client an HTML page containing computation's results.

2.8.2 Java Server Pages

Java Server Pages (JSP) technology is built on top of the Java Servlet API, which can provide dynamic user interface in platform-independent way [87]. JSP separates presentation from dynamic content. This makes it possible for developers to change page layout without being concerned about any underlying dynamic content.

A JSP page is a text-based document that contains two types of text: static template data and JSP elements. Static template data can be expressed in any text-based format such as HTML, XML, and WML. JSP elements contain JSP tags and Java code to construct dynamic content. During the process, these static data are unchanged and sent to the client to be rendered as user interfaces in the browser. The dynamic data logic is processed by the Java code.

The native scripting language for JSP pages is based on the Java programming language and all JSP pages are compiled into Java Servlets. When receiving a request from the browser, the JSP engine checks whether the requested JSP is the first request or has been changed. If yes, the JSP pages are complied into a Servlet by the JSP engine. The JSP performs Servlet tasks: request processing, forwarding, and communicating with business objects which may be a remote CORBA object or an Enterprise Java Bean.

As we mentioned in Section 2.7.2, the JSP page can interact with back end resources via an EJB component in J2EE environment. The EJB component manages access to the back end resources, which provides scalable performance for high numbers of concurrent users [44].

2.9 Summary

In this chapter, we have introduced background information on electronic payment systems and several macro-payment and micro-payment systems. Some possible Internet technologies are also described for their implementation. There is a growing need for an effective, efficient micro-payment technology for high-volume, low-value E-commerce products and services. Current

macro-payment approaches do not scale to such a domain. Most existing micro-payment technologies proposed or prototyped to date suffer from problems with security, lack of anonymity and performance. In the next chapter we propose a new micro-payment protocol called NetPay in order to overcome the problems.

Chapter 3

The NetPay Protocol

This chapter proposes a new offline micro-payment protocol – NetPay. The main aim of this protocol is that it shifts the communication traffic bottleneck from a broker and distributes it among the vendors, places some of the micro-payment processing burden on vendors when a customer wishes to purchase from a new vendor. We give a motivation for the NetPay protocol; describe the protocol and some of its new characteristics.

3.1 Motivation

Though offline protocols have received a lot of attention from researchers and cryptographers, no offline payment systems are currently in general public use. The experimental system, Millicent [54] uses no public-key cryptography and is optimized for repeated micro-payments to the same vendor. Its distributed approach allows a payment to be validated and double spending be prevented without the overhead of contacting a third party (often called broker) online during purchase from same vendor. However the third party must be online whenever the user wishes to interact with a new vendor. The system places a heavy real-time burden on the third party in such circumstances. Mpay [40] is based on a notational model and has off-line capability in its daily certificate. Although Mpay is flexible, convenient and a low-overhead payment system, customers can pay nothing to access services for a full day. Due to the after-the-fact policing requirements, vendors know customers' information from daily certificates. Customer's anonymity is not supported in Mpay. PayWord [72] is a credit-based off-line protocol. The system aims to minimize the number of public key operations required per payment using hash operations. However the payword chain is customer and vendor specific, a customer can generate e-coins (payword chain) that provides more opportunities for fraud and then spend them to a specific vendor.

We developed a new protocol called NetPay that allows customers to purchase information from vendors on the WWW without having to involve a third party in every transaction [22, 23, 24]. At the same time, NetPay protocol minimises the number of expensive public-key

operations [72] required per payment. Hash function operations [71] are used, in order to reduce the transaction overhead. In addition, the NetPay system is capable of preventing customers from double spending, vendors from double depositing and both from forgery.

The NetPay micro-payment protocol is based on the main approach used on the PayWord [72] system. However, NetPay is a debit-based system and it also addresses the problem of double spending and overspending by customers, and protects anonymity of the customer.

3.2 NetPay Scenario

In this section, we describe the transactions and related issues in the NetPay protocol. The system includes a customer (C), vendor (V), and broker (B). We assume that the broker is honest and is trusted by both the customers and the vendors. The customers and the vendors may be dishonest. The vendors and the customers open accounts and deposit funds with the broker. The payment only involves C and V and B is responsible for the registration of customers and for crediting the vendor's account and debiting the customer's account.

- **Broker:** The role of the broker is to manage accounts of customers and vendors, and to store e-coin information. It serves as accounting intermediary between customers and vendors. A broker produces an e-coin chain which can be bought by the customers. E-coin chains are used by the customers to pay for on-line content with vendors. A broker provides a redeem service which can be used to redeem spent e-coins by vendors. A broker handles real-money transactions which are outside the scope of NetPay to sell e-coin chain and to redeem spent e-coins for real money.

- **Customer:** The customer registers and buys a payword chain with the broker. The e-coin chain stores in his/her e-wallet for further purchases with vendors. The e-wallet of the customer is responsible for sending e-coins as payment to vendors for buying information goods. When the e-wallet receives a purchase request from a vendor, it checks whether it has the appropriate amount of e-coins to pay for the requested information goods. If not, the customer directly buys more with the broker. Then the e-coins are sent to the vendor to pay.

- **Vendor:** The vendor offers information goods and sells them to the customers. When it receives e-coins from a customer's e-wallet as payment for a purchase, the e-coins are verified whether it was tampered with, forged or spent doubly. If all checks succeed the e-coins are stored for later redemption.

Figure 3-1 shows the NetPay payment model. This model works as follows:

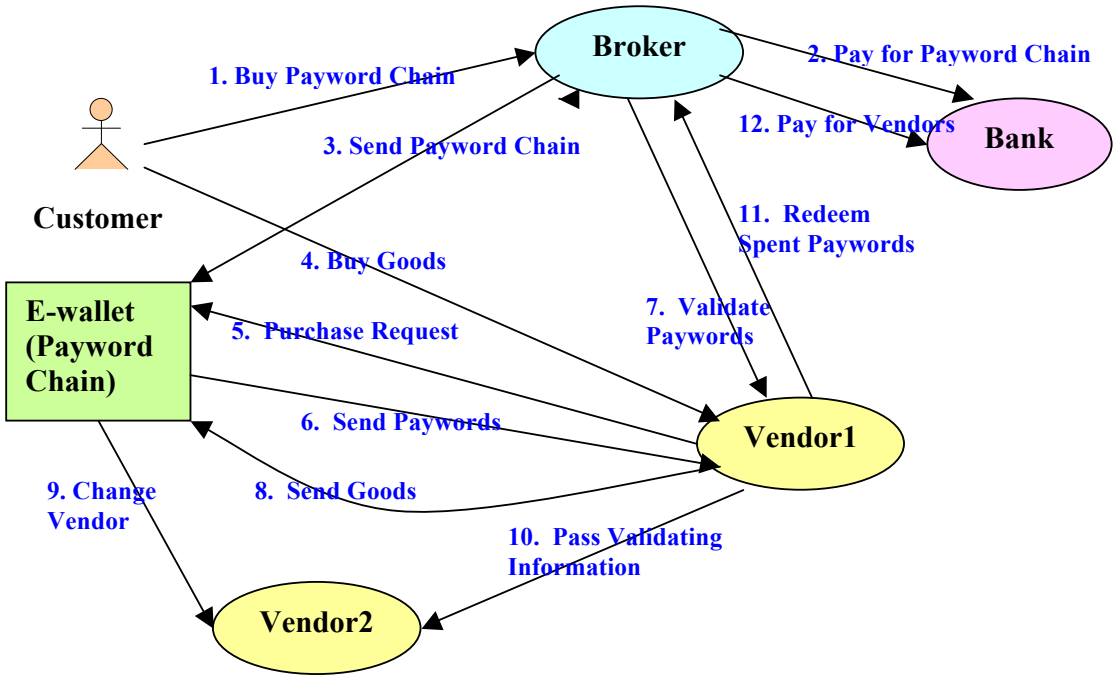


Figure 3-1 NetPay basic interaction between the parties

Initially a customer accesses the broker's web site to register and buy a number of e-coins from the broker (1) using a single macro-payment (1). The broker sends an "e-wallet" that includes the e-coin chain to the customer (3).

When the customer wishes to purchase information goods from Vendor1 site (4), the Vendor1 sends a purchase request to the customer's e-wallet (5) and the e-wallet sends e-coins to the vendor1 (6). Then Vendor1 gets validating information from the broker and verifies the e-coins (7). If the payment is valid, the information goods is sent to the customer (8). The customer may purchase other information goods, their coins being debited. If coins run out, the customer is directed to the broker's site to buy more.

When the customer changes to Vendor2 (9), Vendor2 first requests the current e-coin validating information from the Vendor1. Vendor2 contacts Vendor1 to get the e-validating information and then debits e-coins for further information goods (10).

At the end of each day, the vendors send all the spent e-coins to the broker redeeming them (11) for real money (12).

3.3 Preliminaries

There are a number of cryptography and micro-payment terminologies used in the NetPay micro-payment protocol. The details of these terminologies are given as follows.

1. ***Hash function*** - In mathematical terms, a one way function *h* means that given a value (real number, or complex number) x, there is one and only one $y=h(x)$ (y is also a real number or complex number), but, given y, there are more than one x such that $y=h(x)$. A one way hash function *h* is a data value, which is easy to compute the data value $y=h(x)$ but difficult to reverse, i.e., it is difficult to find data value x such that $y=h(x)$ for a given data value y.

   The one way hash function MD5 used in NetPay implementation is an algorithm that has the two properties. It seems impossible to give an example of hash function used in hash chain in a form of normal functions in mathematics. The difficulties include:

   - The value of a mathematical function is a real or complex number (a data value for hash function);

   - It is always possible to compute the set $X = \{x | x = h^{-1}(y)\}$ for a given y for a mathematical function *h* (not satisfying the two properties of the hash function).

2. ***Payword Chain*** – A "payword chain" is generated by using a one way hash function. Suppose we want to generate a payword chain which contains ten "paywords". We need randomly pick a payword seed W_{11} and then compute a payword chain by repeatedly hashing

$W_{10} = h(W_{11})$, $W_9 = h(W_{10})$,
……,

$W_1 = h(W_2)$, $W_0 = h(W_1)$

where h(.) is a hash function such as MD5 and W_0 is called the root for the chain. The MD5 (Message Digest) algorithm [71] is one of the series of messages in hash algorithms and involves appending a length field to a message and padding it up to a multiple of 512 bit blocks. This means that every payword W_i is stored as a 32 length string in a database. A payword chain is going to be used to represent a set of E-coins in the NetPay system.

3. ***E-coin*** – An "e-coin" is a payword element such as W_1 or W_{10}. The value of a payword e-coin might be one-cent but could be some other value.

4. ***E-wallet*** – An "e-wallet" is used to store e-coins and send e-coins to a vendor paying for information goods, i.e. it shows one or more payword chains

5. ***Touchstone*** – A "touchstone" is a root W_0 and is used to verify the paywords $W_1, W_2, ... W_{10}$ by taking the hash of the paywords in order W_1 first [$h(W_1) = W_0$], then W_2 [$h(h(W_1)) = W_0$], and so on. This is used to verify the e-coins are "valid" i.e. have not been forged.

6. ***Index*** – An "index" is used to indicate the current spent amount of each e-coin (payword) chain. For example if you have spent 2cs (W_1, W_2) to buy an information goods, the current index value is 3 in the previous example of a chain $W_1 ... W_{10}$.

The management of the security of e-coins is one of the key issues in micro-payment systems. NetPay uses a low-cost per transaction but high security method between customers and vendors to secure the use of e-coins [22, 23]. This method adopts the passing of "touchstones" used to verify the validity of an e-coin passed to a vendor from a customer's e-wallet. When a customer first tries to spend an e-coin the vendor communicates with the broker to obtain a validating touchstone for the e-coin. Each e-coin encodes a "payword chain" where a fast hashing function gives the next valid coin in the chain each time a coin is spent. An index associated with each e-coin indicates the amount spent so far. When a customer moves to another vendor site, the new vendor obtains the

touchstone value and index from the previous vendor. The initial transfer of e-coins from broker to customer is secured by public key encryption. The index value associated with the e-coin is used to prevent customer from double spending and ensures no conflicts between vendors [22]. The vendor does not know the identification of the customer at any stage, preserving their anonymity. In NetPay, the customer needs to contact the broker to buy e-coins only when his e-coins run out and thus it is a full off-line system during purchase of information goods.

3.4 NetPay Transactions

Our micro-payment protocol NetPay [22] aims to provide a secure, cheap to implement, widely available, debit-based protocol for micro-payments. NetPay differs from previous protocols in the following aspects: NetPay uses touchstones signed by the broker and Index's signed by vendors passed from vendor to vendor. The signed touchstone is used by a vendor to verify the electronic currency – paywords, and a signed index is used to prevent double spending from customers and to resolve dispute between vendors.

In a NetPay system, there are four trasactions which are customer-broker, customer-vendor, vendor-vendor, and vendor-broker transactions. How the NetPay protocol works in each transaction will now be described in more detail. We adopt the following notations:

IDa --- pseudonymous identity of any party A in the trade community issued by the broker.
PK-a --- A's public key.
SK-a --- A's digital signature.
{x}SK-a --- x signed by A.
{x}PK-a --- x is encrypted by A's public key.

3.4.1 Transaction 1: Customer – Broker

Before a customer asks for service from the first vendor (V_1), she has to register and send an integer n (M1), the number of paywords in a payword chain the customer applied for, to the broker (Figure 3-2). The broker completes two actions:

- Debits money from the account of C and creates a payword chain W_0, W_1, W_2,…,W_n, W_{n+1} which satisfy $W_i = h(W_{i+1})$, where i = n, …, 0. Root W_0 is used to verify the validity of the paywords W_1, W_2, …, W_n by vendors and the broker. Seed W_{n+1} is kept by the broker to be used to prevent the customer from overspending and forging paywords in that chain. The customer only receives ID_e (e-coin ID) and paywords W_1, W_2,…,W_n that are encrypted by customer's public key from the broker (M2) as shown in Figure 3-2.

$$M2 = \{ ID_e, W_1, W_2, \dots, W_n, \text{B's host and port}\}_{PK\text{-customer}}$$

- Save IDe, W_0, W_{n+1}, and amount to the broker database.

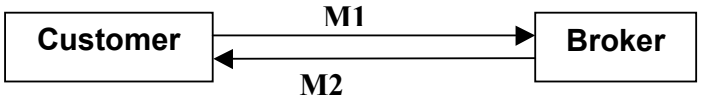


Figure 3-2 Customer buys e-coins transaction

For example, the customer sends n=50 to the broker who generates the IDe=1 and payword chain {W_0, W_1, W_2, … ,W_{50}, W_{51}}. The e-wallet is thus {IDe, W_1, W_2, … ,W_{50}, "www.sharlene.com", "1051" (the broker's port)} as shown in Figure 3-3 and the broker saves IDe, W_0, W_{n+1}, and 50 to its database.

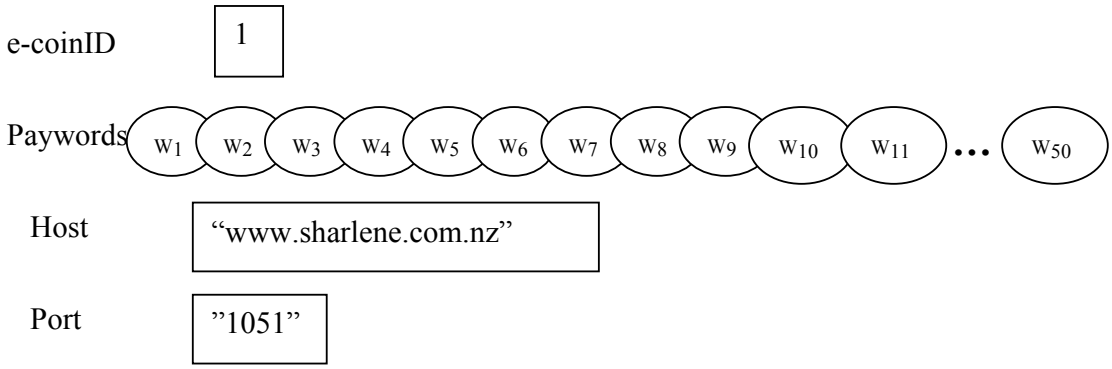


Figure 3-3 Customer e-wallet database in the beginning of the transaction

The customer-broker transaction guarantees no overspending and forging. The broker selects the seed W_{n+1} to create the payword chain which satisfy $W_n = h(W_{n+1})$, $W_{n-1} = h(W_n)$, …, $W_1 = h(W_2)$, $W_0 = h(W_1)$ and keep the seed W_{n+1} secretly. It is impossible to forge the paywords in that chain by customers, vendors and attackers, since they do not

have the seed W_{n+1}, i.e. it is impossible to generate other paywords in a chain by knowing some of them in the chain since h() is a truly one-way hash function [71].

3.4.2 Transaction 2: Customer – Vendor

The following sequence of messages describes a transaction between a customer and a vendor in the course of a purchase. The customer and vendor need to agree on the amount that C pays. In our scheme, the price of a web page might be one-cent, but could be some other amount.

When a customer purchases information from V_1, V_1 sends a price (e.g. m), host and port (M3) to the customer's e-wallet. The e-wallet compares the host and port in M3 with the previous host and port. If different, the e-wallet sends a message M4 back to V_1;

M4 = { IDe, paywords, B's host and port};

where paywords = $\{W_1, W_2, \ldots, W_m\}$. For example , to make a 2cs (m=2) payment, the customer sends the paywords W_1, W_2: Paywords = $\{W_1, W_2\}$ to the V_1.

If the customer fist time makes a purchase with a vendor using the e-coin, V_1 sends the IDe (M5) to the broker for requesting the touchstone. The broker computes the touchstone for the payword chain:

$$M6 = T = \{IDe, W_0\}_{SK\text{-}broker}$$

and sends it to V1. T is signed by B. The touchstone authorises V_1 to verify the paywords using root W_0 and redeems the paywords with the broker as shown in Figure 3-4.

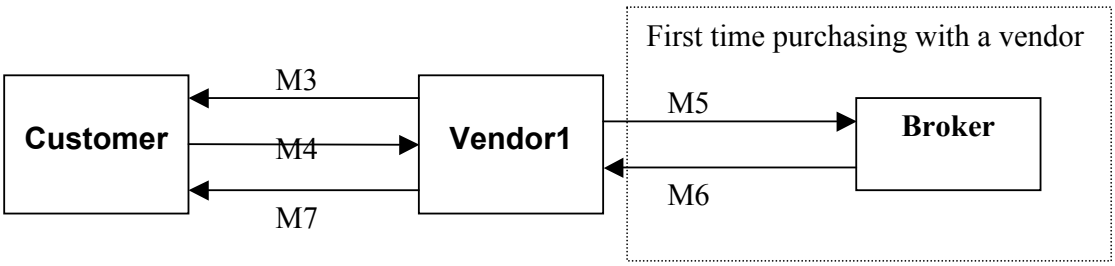


Figure 3-4 Customer buys information goods transaction

The paywords are verified by taking the hash of the paywords in the order W_1 first, then W_2, and so on. The paywords W_1 and W_2 are valid if the hash matches the root of the chain (W_0) in the touchstone ($h(W_1)=W_0$, $h(h(W_2))=W_0$). This works because the hash function with the property $W_{i-1}= h(W_i)$ ($i = 1, 2, \ldots, n$) and V_1 gets W_0 from the broker.

On the other hand, it is hard for V_1 to create W_1 even though he knows W_0 since the generation of a value that would hash to W_0 is computationally infeasible due to the nature of the one-way hash function [71]. For the same reason, it is also hard for an attacker to generate valid paywords in the chain even if he knows W_0 or some paywords except for the seed W_{n+1} [71, 56].

If the paywords are valid, they will be stored for a later offline transaction with the broker. The customer is supplied with the information goods (M7). Multiple payments can be charged against the length of the payword chain, until the payword chain is fully spent or the customer no longer requires information goods on WWW [71].

For example the V_1 sends M3 = {3cs (the price of the information goods), "www.enewspaper1.com.nz", "1053" (V_1's port)} to the customer's e-wallet, when the customer requests to buy an information goods which costs 3cs. The e-wallet compares "www.enewspaper1.com.nz", and "1053" in M3 with "www.sharlene.com.nz" and "1051" in the current record of the e-wallet and then sends M4 = {IDe, $W_1W_2W_3$, "www.sharlene.com.nz", "1051"} to the V_1. The current state of the customer e-wallet database is shown in Figure 3-5.

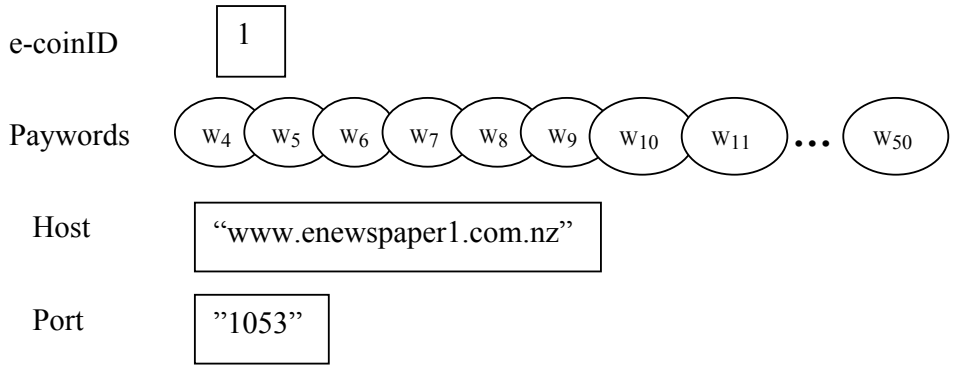


Figure 3-5 Example of customer e-wallet database after first transaction

The V_1 gets T (M6) from the broker by sending IDe (M5) and then verifies W_1, W_2, W_3 by using W_0 such as $h(W_1)=W_0$, $h(h(W_2))=W_0$, $h(h(h(W_3)))=W_0$. If the paywords are valid, V_1 sends the information goods to the customer (M7) and saves IDe=1, index=4, price=3, W_0, paywords= $W_1W_2W_3$ in a redeem database as shown in Figure 3-6.

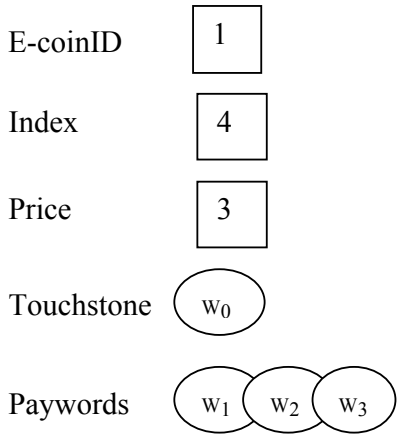


Figure 3-6 Example of redeem database after first transaction

The customer continues to buy other information goods which costs 2cs, the V_1 sends M3 = {2, "www.enewspaper1.com.nz", "1053"} to the customer's e-wallet. The e-wallet compares "www.enewspaper1.com.nz", and "1053" in M3 with "www.enewspaper1.com.nz", and "1053" in the current record of the e-wallet and then sends M4 = {IDe, W_4W_5, "", ""} to the V_1. The current state of the e-wallet database is shown in Figure 3-7.

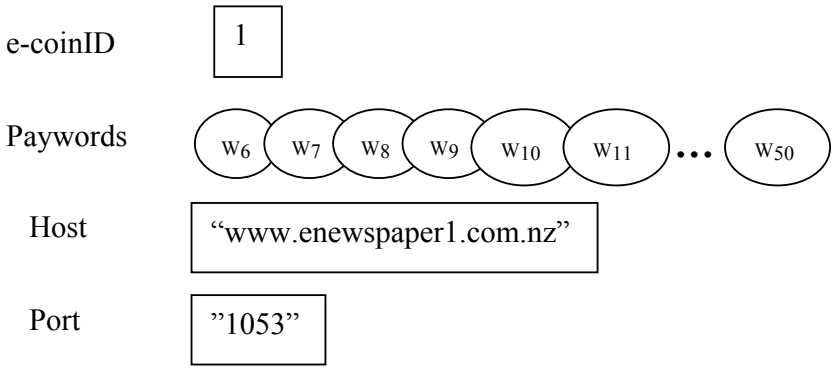


Figure 3-7 Example of the e-wallet database after second transaction

The V_1 verifies W_4, W_5 by using W_0 obtained before. If the paywords are valid, V_1 sends the information goods to the customer (M7) and saves IDe, index=6, price=2, W_0, paywords= W_4W_5 to the redeem database as shown in Figure 3-8.

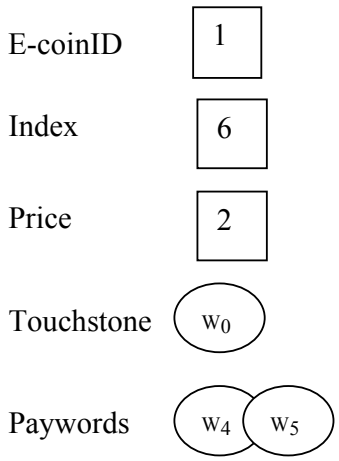


Figure 3-8 Example of redeem database after second transaction

3.4.3 Transaction 3: Vendor – Vendor Payword Relocation

As we described the NetPay scenario in Section 3.2, the vendor2 requests the current e-coin index and the touchstone from vendor1 to verify the e-coins when the customer changes purchasing from the vendor1 to a vendor2 as shown in figure 3-9.

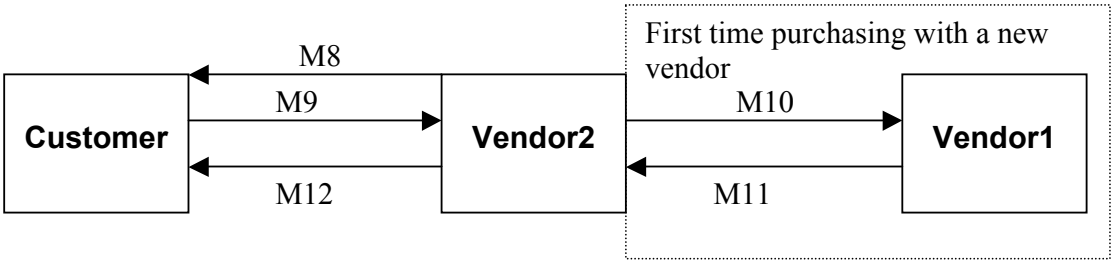


Figure 3-9 Paywords relocation transaction

When the customer wishes to purchase information goods at V_2, V_2 sends a price, host and port (M8) to the e-wallet. The e-wallet compares the host and port in M8 with the previous host and port. If different, the e-wallet sends a message

$$M9 = \{IDe, paywords, V_1\text{'s host and port}\}$$

to V_2. V_2 transmits the IDe (M10) to V_1 requiring touchstone and index. The V_1 signs the following transmission message:

$$Index = \{IDv_1, i\}_{SK\text{-}v1}$$

along with the payword chain touchstone, and transmits them to V2 (M11), where i is the index of the last payword V_1 received. The Index may be used for disputes between the vendors, and the touchstone is used to make future transactions with C and to redeem the paywords from the broker. After V_2 verifies the paywords using the touchstone and the index, the customer receives the information goods (M12).

For example, when the customer requests to buy an information goods which costs 4cs with V_2, V_2 sends M8 = {4, "www.enewspaper2.com.nz", "1054" (V_2's port)} to the customer's e-wallet. The e-wallet compares the "www.enewspaper2.com.nz" and "1054" in M8 with the "www.enewspaper1.com.nz" and "1053" in the current record of the e-wallet and then sends M9 = {IDe, $W_6W_7W_8W_9$, "www.enewspaper1.com.nz", 1053} to the V_2. The current state of the e-wallet database is shown in Figure 3-10.

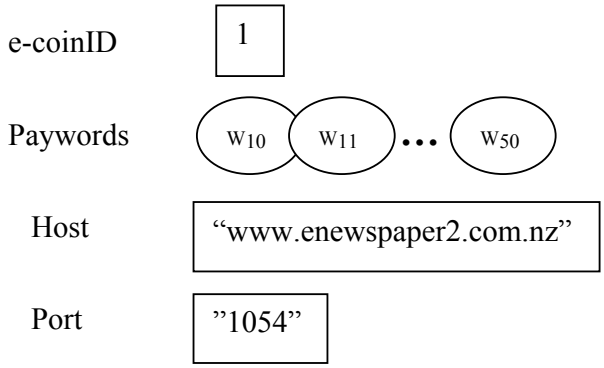


Figure 3-10 Example of the e-wallet database after sending e-coins to the V2

The V_2 gets the touchstone and index (=6) (M11) from the V_1 by sending IDe (M10) and then verifies W_6, W_7, W_8, W_9 by using W_0. If the paywords are valid, V_2 sends the information goods to the customer (M12) and saves the IDe, index=10, price=4, W_0, paywords= $W_6W_7W_8W_9$ to a V_2's redeem database as shown in Figure 3-11.

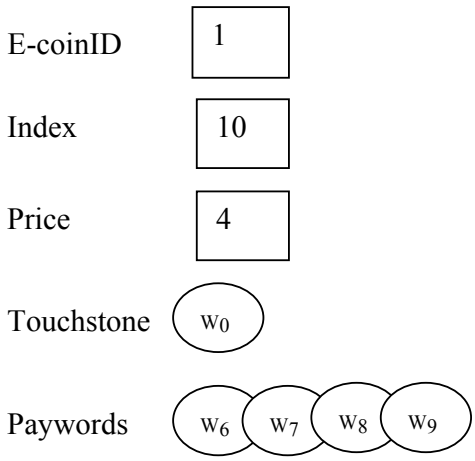


Figure 3-11 Example of redeem database after first transaction with V2

The customer can continue to buy other information goods with the V_2. This transaction has two advantages: firstly, the transfer of the message M11 from V1 to V2 does not involve the broker, it reduces the communication burden of the broker; secondly, the message M11 includes the index of the paywords, it prevents the customer from double spending when the customer purchases from another vendor.

3.4.4 Transaction 4: Vendor – Broker Offline Redeem Processing

At the end of each day (or other suitable period), for each payword chain, all vendors need to send all paywords that they received from customers to the broker and redeem them for real money. To do this a vendor must aggregate the paywords by each e-coinID and send the following message to the broker

M13 = {IDv, IDe, Payments}

The broker needs to verify each payword received from the vendor by performing hashes on it and counting the amount of paywords. If all the paywords are valid, the broker deposits the amount to the vendor's account, and then sends an acknowledgement

M14 = {Balance Statement of the vendor's account}

to the vendor as shown in Figure 3-12.

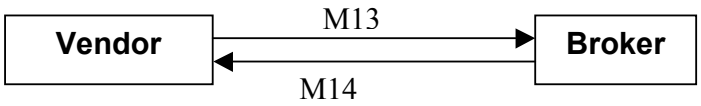


Figure 3-12 Vendor redeem transaction

For example, at the end of each day, V_1 aggregates two payments as shown in Figure3-6 and Figure3-8 for IDe=1 and sends IDv_1 and IDe along with 6 (index), 5 (price), W_1W_2 … W_5 (paywords) (M13) shown as Figure 3-13 to the broker. The broker verifies the paywords (W_1W_2 … W_5) by using W_0, index (6) and price (5). If they are valid, the broker deposits 5cs to the vendor1's account and send the balance to the vendor1 (M14).

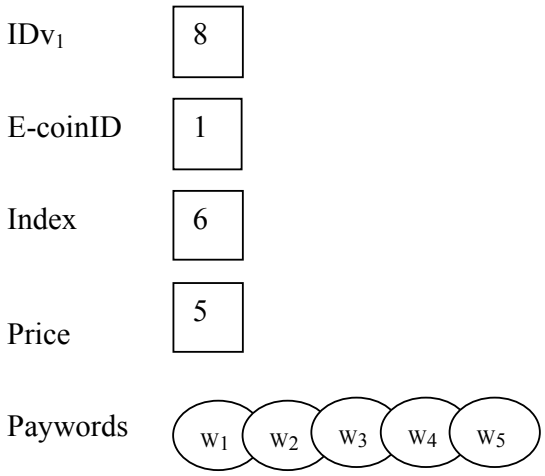


Figure 3-13 V_1 aggregates two payments

3.4.5 Possible Extension - Divisibility

The NetPay system introduced so far has only one payword value e.g. 1c per coin. In this subsection, we describe how to extend the system to paywords with any values. This means users can extend the system to any number of values as required by simply following the changes illustrated below.

The broker could generate payword chains that have different values for customer, e.g, assume 6 values of these chains, \$0.01, \$0.02, \$0.05, \$0.10, \$0.20, and \$0.50 are required for every customer. Let these payword chains be

$W_{1\,1}, W_{1\,2}, \ldots, W_{1\,n}$

$W_{2\,1}, W_{2\,2}, \ldots, W_{2\,n}$

……

$W_{6\,1}, W_{6\,2}, \ldots, W_{6\,n}$

Where $W_{i\,j}$ satisfy $W_{i\,j} = h(W_{i\,j+1})$ (j = n, … ,0 and i = 1, … ,6) and h() is the hash function introduced before. Roots $W_{1\,0}, W_{2\,0}, \ldots, W_{6\,0}$ are used to verify the validity of the paywords by the vendors and broker. Seeds $W_{1\,n+1}, W_{2\,n+1}, \ldots, W_{6\,n+1}$ are kept by the broker to prevent the customer from forging and overspending. The messages in NetPay should be changed as follow.

M2 = {IDe, $W_{1\,1}, W_{1\,2}, \ldots, W_{1\,n}, \ldots, W_{6\,1}, W_{6\,2}, \ldots, W_{6\,n,}$ B's host and port } $_{PK\text{-}customer}$

M5 = T = {IDe, $W_{1\,0}, W_{2\,0}, \ldots, W_{6\,0}$ } $_{SK\text{-}broker}$

Paywords = { $(W_{i\,j})$ } where i = 1, …, 6 and j = 1, …, n.

Index = {$ID_{v1,}\, i_1, i_2, \ldots, i_6$ } $_{SK\text{-}v1}$

All other procedures in the process remain unchanged.

3.5 Customer Anonymity

A fundamental property of physical cash is that the relationship between customers and their purchases is untraceable. This means that the payment systems do not allow payments to be traced without compromising the system's security.

The payword chain in the NetPay system is sold by a broker and redeemed by different vendors. The broker cannot determine where the customer is going to spend the paywords except for the first purchase by the customer, because the payword chain is not vendor-specific. But the broker knows the vendors that a customer traded with when the vendors redeem paywords. In other words, NetPay offers partial anonymity.

3.6 Differences between NetPay and PayWord

The basic parts of the NetPay protocol are based on the PayWord protocol [72]. Like PayWord, NetPay transfers money to vendors in the form of a payword chain. The difference between PayWord and NetPay is the allocation and distribution of the payword

chain. In PayWord, a payword certificate authorises a customer to generate payword chains and guarantees that a specific broker will honour them. The payword chain is vendor-specific and customer-specific. Hence the paywords in the chain have no value to another vendor.

In contrast, NetPay is a debit based scheme, in which the payword chain is generated by the broker for every customer. The customer spends paywords from one vendor to another without involving of the broker. The payword chain remains active at only one vendor at a time, thus preventing customers from double spending. The payword chain in NetPay system is customer-specific but not vendor-specific. The paywords can be spent with any vendor. Customers in the NetPay do not need to generate and manage the vendor-specific payword chains for every vendor, thus the NetPay is easier to use than PayWord for customers.

3.7 Summary

Vendor specific e-coins and no anonymity for customers are two common disadvantages in many existing micro-payment systems. Since only the broker knows the mapping between the pseudonyms (IDc) and the true identity of a customer in NetPay, the protocol protects the customer's privacy from vendors. The protocol also prevents customers from double spending and makes it very difficult and expensive to forge e-coins, so it satisfies the requirements of security as a micro-payment system should have. The protocol is "cheap" since it typically involves no public-key operations per purchase. The e-coin (payword chain) is not vendor specific and can be used to buy information goods from any vendor using the same broker. We now describe some prototype NetPay-based thin-client web applications that enable customers to click and buy information goods on the Internet by using the NetPay protocol.

Chapter 4

NetPay System Requirements

In the previous chapter we have described a new micro-payment protocol – NetPay. This chapter describes the specification of a NetPay-based micro-payment e-commerce system by capturing the essential user requirements with Object Oriented Analysis (OOA). It starts with a brief introduction of the problem domains by introducing scenarios where vendors want to charge on a usage basis and compare the use of several micro-payment models. This is followed by discussing the NetPay system's requirements for an on-line newspaper application. A detailed OOA specification of NetPay is presented, which consists of three parts: system requirements, use case modelling, and OOA modelling.

A software process model simply includes five phases which are requirements engineering, design, implementation, testing, and maintenance [83]. The requirements engineering phase which includes user requirements and use case, non-functional requirements and object oriented analysis will be introduced in this chapter [14]. The design and implementation phases which contains software architecture design, object oriented design, database design and implementation for a CORBA-based and a component-based NetPay will be introduced in Chapter 5 and 6.

4.1 Problem Domain

Before identifying the user requirements of NetPay, some practical issues are discussed in detail. These focus on the reasons for developing a NetPay micro-payment system, which mainly concerns the selling of goods on the Internet. In addition, NetPay's requirements are also compared with other micro-payment systems like Millicent, Mpay or PayWord which are alterative micro-payment models for this purpose.

4.1.1 Practical Issues

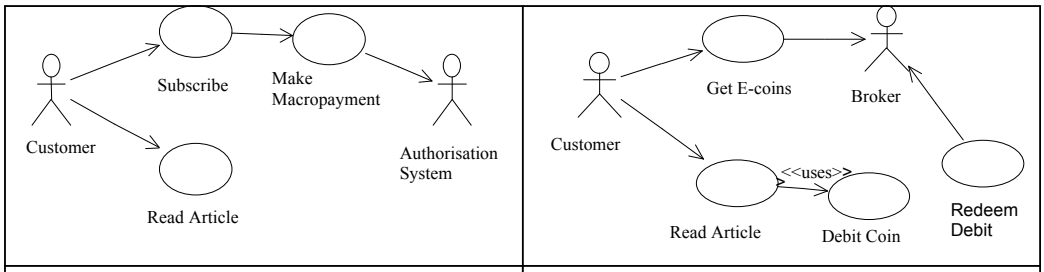
With the development of Internet businesses, more and more content providers are switching once free content or services to a paid subscription model or pay-per-click model, eliminating the revenue relying on only an advertisement market [68, 54]. Today there are already many newspapers and journals in electronic form. Some of newspapers and journals allow their regular subscribers to read the articles on the net for free while they also get a normal paper copy of them. Such a procedure seems to waste resources since the subscribers can print the articles in which they are interested on net and thus there is no need to read the paper copy. Micro-payment systems could be used to make things different for online contents or services [41, 55, 63]. You can read and download an article and only pay a small amount of money e.g. 10c or 20c.

There are many possible applications of micro-payment systems. They can be used to buy not only just text-based information but also other goods. One of the non-text-based goods is online-music [57], where one can download a single song from on-line music sites by paying a small amount money. There is also a multitude of game sites [12] on which micro-payment systems could be used. Another candidate for micro-payment is clip-media services where one can purchase graphics, audio, and video online [53, 51].

4.1.2 An E-newspaper with Two Payment Methods

Assuming a reader wants to read an on-line newspaper. Using subscription-based payment, they would first have to subscribe to the newspaper by supplying payment details (credit card etc) and the newspaper system would make an electronic debit to pay for their subscription, by communicating with an authorisation server. The user would then normally go to the newspaper's site where they login with an assigned user name and password. The newspaper looks up their details and provides them access to the current edition if their subscription is still current. If the user's subscription has run out, they must renew this by authorising a payment from their credit card. Figure 4-1 (a) outlines the key interaction use cases for this scenario. Problems with this approach are that there is no anonymity for the user (the newspaper system knows exactly who they are and when and what they read), they can not browse other newspapers without first subscribing to them too, and they must pay for the whole newspaper, even if they want just one or two sections or articles.

An alternative approach is a micro-payment model [52]. The user first goes to a broker and purchases "E-coins" using a single macro-payment. These are stored in an E-wallet on the user's machine. The user can then visit any newspaper site they wish, with their wallet giving the site an E-coin. Each time they view an article (or section or page, depending on the item charged for) their E-coin is debited. The vendor redeems debits with the broker (for "real" money") periodically e.g. each night/week. The user can move to another site and unspent money associated with their E-coin is transferred from the first vendor to the second. If coins run out, the user communicates with the broker and authorises another macro-payment debit. Figure 4-1 (b) outlines the key interaction use cases for this scenario.



(a) Typical macro-payment interaction model.

(b) Possible micro-payment interaction model.

Figure 4-1 Two on-line newspaper interaction scenarios

Currently many customers rely on credit cards for e-commerce purchases. However the credit card method cannot be effectively or efficiently applied for buying a large number of inexpensive information goods with different vendors' sites, such as single articles of an on-line newspaper, because transaction costs are too high. Encryption mechanisms used are slow and each transaction typically "costs" a few cents. An Internet micro-payment system would allow spending large numbers of small amounts of money at web sites in exchange for various content or services, as in the E-newspaper scenario above [23]. As noted by Blankenhorn [8] there is a real need for micro-payment systems.

4.1.3 Comparing NetPay with Payment Systems

We evaluated four of the payment protocols in Section 2.6.5 using the five evaluation criteria. Table 4-1 lists the results of comparison for NetPay protocol with the four payment system models.

Table 4-1 Comparison of payment methods with NetPay

System/ property	CyberCash	Millicent	Mpay	PayWord	NetPay
Ease of use	**Low,** Customer contacts Broker every transaction.	**Medium,** Customer nearly always contacts Broker.	**High,** Customer only needs to click and see what he pays for.	**Medium,** Customer generates and manages different e-coins for every Vendor.	**High,** Customer clicks and gets the content. No login for client-side e-wallet.
Security	**High,** the system performs checking, clearing and recording for every transaction.	**Medium+,** the system prevents double spending by using Vendor-specific scrip.	**Medium,** but the criminal is free to spend money to buy content for a full day.	**Low,** the system is credit-based scheme to provide more opportunity for fraud.	**Medium+,** the system prevents double spending by transferring touchstones and indexes between vendors.
Anonymity	**Low,** the system records identities, exchanged amount and time of a transaction.	**Medium,** Broker knows who and where but not what. Vendors know what but not who.	**Low,** Customer's anonymity is not supported.	**Low,** Broker knows who and where but not what vendors know what and who.	**Medium+,** Customer's anonymity is protected from vendor.
Multi Currency	**No,** only one currency $.	**No,** must be match with scrip.	**Yes,** converts	**Yes,** converts	**Yes,** converts
Robustness	**Low**, on-line payments	**Low**, on-line payments	**High**, off-line payments	**High**, off-line payments	**High**, off-line payments

The NetPay system, which is an off-line protocol, allows customers to purchase high-volume, low-cost per item information from vendors on the web without involving the broker in every transaction. The number of expensive public-key operations required per payment are minimised by using fast hash function operations to get the next payword chain coin, in order to minimise the transaction overhead [22, 23]. Customers are prevented from double spending as the index of the payword chain indicates the balance of the customer's e-wallet**,** and the touchstone can be used to verify the payword chain. NetPay allows customers to move transparently from one vendor site to another, with a single e-coin touchstone and index transfer between vendors. Since only the broker knows the mapping between the pseudonyms (IDc) and the true identity of a customer, the protocol protects the customer's privacy.

4.2 User Requirements and Use Cases

The requirements analysis process defines the principles, standards, techniques, activities, and steps that must be applied to develop requirements for a problem domain. Requirements define what the system needs to perform rather than how the system is

designed, programmed or tested [83]. The basic process we go through to determine the requirements of NetPay system is to identify and describe users in the system, elicit user requirements of the system, analyse and document the user perspectives on the system. To do this, UML Use Case diagrams and use case descriptions are the common ways.

Suppose the e-newspaper sites want to use the NetPay micro-payment system to sell articles on a per usage basis. The system involves four parties which are NetPay broker site, e-newspaper sites, customers and a bank macro-payment system. The customers can be classified as registered customers and unregistered customers. Only registered customers can buy e-coins from a broker's site and click-buy-article with any newspaper sites. Both types of customers can search and view article titles on line.

Initially a customer accesses the broker's web site to register and acquire a number of e-coins from the broker (bought using a single macro-payment). The broker sends an "e-wallet" that includes the e-coin ID, touchstone, and e-coins to the customer.

The customer browses the home page of the newspaper web site and finds a desired news article to read. Each article will typically have a small cost e.g. 5-10c, and the customer would typically read a number of these. When wishing to read the details of an article, the customer clicks on the article heading and the vendor system debits the customer's e-coins by e.g. 10c (by taking 1, 2 or more e-coins from their payword chain, up to 10c value).

The newspaper system verifies that the e-coin provided by the customer's e-wallet is valid by use of a "touchstone" obtained once only from the broker. If the payment is valid (coin is verified and sufficient credit remains), the article is displayed on the screen. The customer may browse other articles, their coins being debited (the index of spent coins incremented) each time an article is read. If coins run out, the customer is directed to the broker's site to buy more.

When the customer changes to another online newspaper (or other kind of vendor using the same e-coin broker currency), the new vendor site first requests the current e-coin touchstone information from previous vendor's site. The new vendor contacts the previous vendor to get the e-coin touchstone and "spent coin" index and then debits coins for further news articles.

When the previous vendor system is "down", a backup server in the system sends the e-coin ID, the touchstone, and the index to the broker. The new vendor could contacts with the broker to get the e-coin touchstone and the "spent e-coin" index.

At the end of each day, the vendors all send the e-coins to the broker redeeming them for real money (done by macro-payment bank transfer from the broker to vendor accounts).

From above, several stakeholders of the required system can be identified and the main scenarios of use of the system have been presented. The following sections illustrate functional and non-functional requirements for the system.

4.2.1 Use Case Diagram

Use Case analysis is one of the first and primary means of gathering requirements in the behavioural methodology. Use cases are a standard technique for gathering requirements in many modern software development methodologies. Use cases are included in the Unified Modeling Language (UML) which is very common software modelling approach [30, 10]. This section focuses on use case modeling to specify the NetPay micro-payment system's requirements, so as to get a more comprehensive appreciation of the functional requirements on the e-newspaper with NetPay systems.

Use case is a requirements capture technique used in this project to identify, clarify, and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal. The actors interact with the system via each specific use case. Actors are typically the people who use the system, or the external entities that need to interact with the system. In the NetPay micro-payment system, actors are customers (e.g. newspaper customers), macro-payment system (e.g. bank), on-line e-newspapers, and a broker which is responsible for the registration of customers and for crediting the e-newspaper's account and debiting the cystometer's account via a macro-payment system. We assume that the broker is honest and is trusted by both the customers and the e-newspapers.

Figure 4-2 illustrates the system's main use case view. It shows the customer can register and purchase some e-coins (using macro-payment such as credit card) with the broker site.

When a customer wants to buy articles he/she can directly login or go to e-newspaper1 site, then click desired article heading. The vendor system debits and verifies the e-coins. If e-coins are valid, the customer can view or print or download the content of the article. E-newspaper2 requires touchstone and index from e-newspaper1 when the customer changes to e-newspaper2 purchasing articles. At the end of each day, all e-newspaper systems send redeem messages to the broker system for real money.

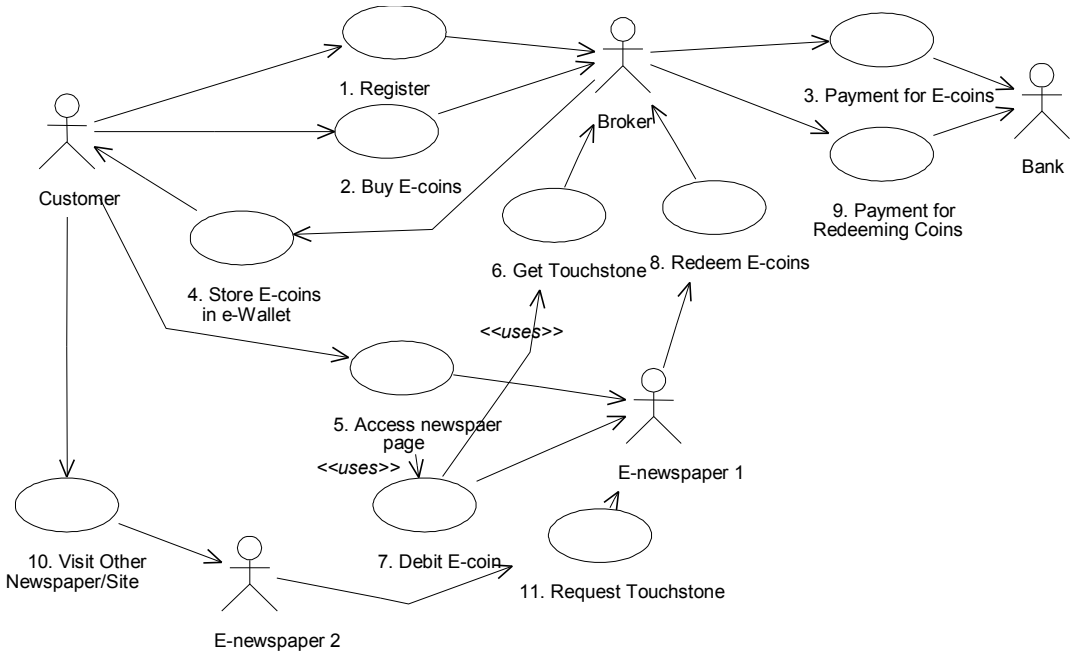


Figure 4-2 E-newspaper with NetPay system main use case diagram

When a customer first tries to spend an e-coin the vendor communicates with the broker to obtain a validating touchstone for the coin. Each e-coin encodes a "payword chain" where a fast hashing function gives the next valid coin in the chain each time a coin is spent. An index associated with each e-coin indicates the amount spent so far. When a customer moves to another vendor site, the new vendor obtains the touchstone value and index from the previous vendor. The transfer of e-coins from broker to customer is secured by public key encryption. The index value associated with the coin is used to prevent customer from double spending and ensures no conflicts between vendors [20].

4.2.2 Use Case Descriptions

Brief descriptions for register, buy e-coins, debit e-coins and redeem e-coins use cases are show in this subsection. The tables 4-2 to 4-5 describe the event flow of the use cases and the screen dumps (Figure 4-3 to 4-5) show the corresponding interfaces of the use cases.

Table 4-2 Register Use Case

Use Case Name:	Register
Description:	Used by customers to register via broker web page.
Event Flows:	1. Customer enters broker's home page and selects "register". Customer fills in registration information which includes customer name, email address, password, and credit card details to open an account as shown in Figure 4-3. Then the customer clicks register button. If the bank information is not correct, go to step 3. 2. Broker generates a customer ID and creates an account for customer. 3. Incorrect credit card information – error message displayed. Go to 1.
Related Actors/Use Cases:	Used by customer actor
Special Conditions:	

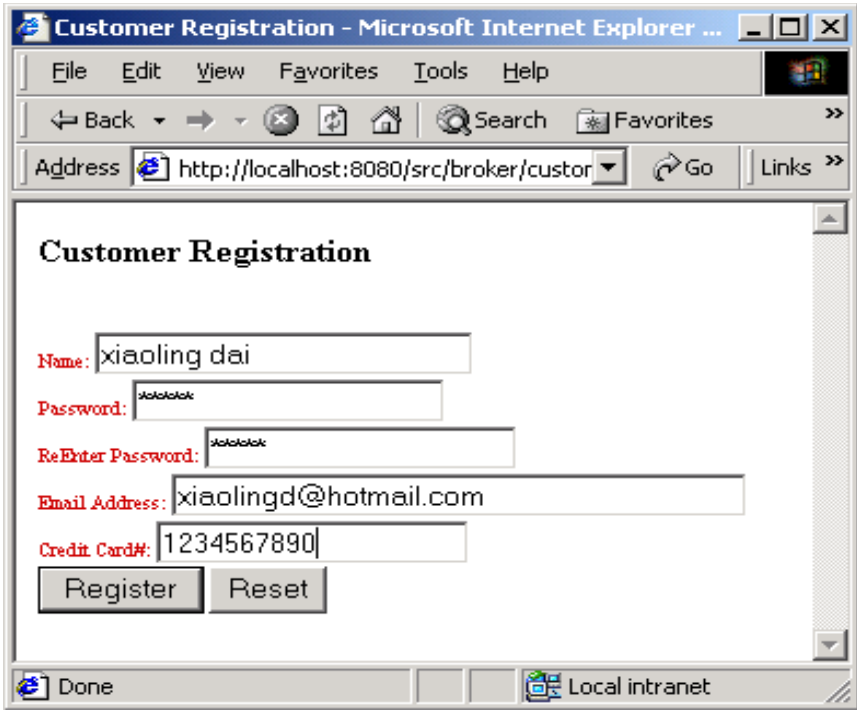


Figure 4-3 Example customer registration

Table 4-3 Buy E-coins Use Case

Use Case Name:	Buy E-coins
Description:	Used by customers to buy E-coins with broker system
Event Flows:	1. Customer enters broker's home page and selects "Buy E-coins". Customer types in customer ID, password, and a e-coin amount which is number of e-coins required and clicks "Login&Buy" button. 2. The system displays the customer ID and amount and customer click "OK" to confirm the transaction as shown in Figure 4-4. 3. Broker debits money from the credit card or the bank account customer supplied via a macro-payment transaction. If the account does not exist, go to 5 4. Broker generates e-coin payword chain which includes e-coin ID, root, seed, paywords, amount and sends to customer's PC. 5. Incorrect credit card information – error message displayed. Go to 1.
Related Actors/Use Cases:	Used by customer actor
Special Conditions:	

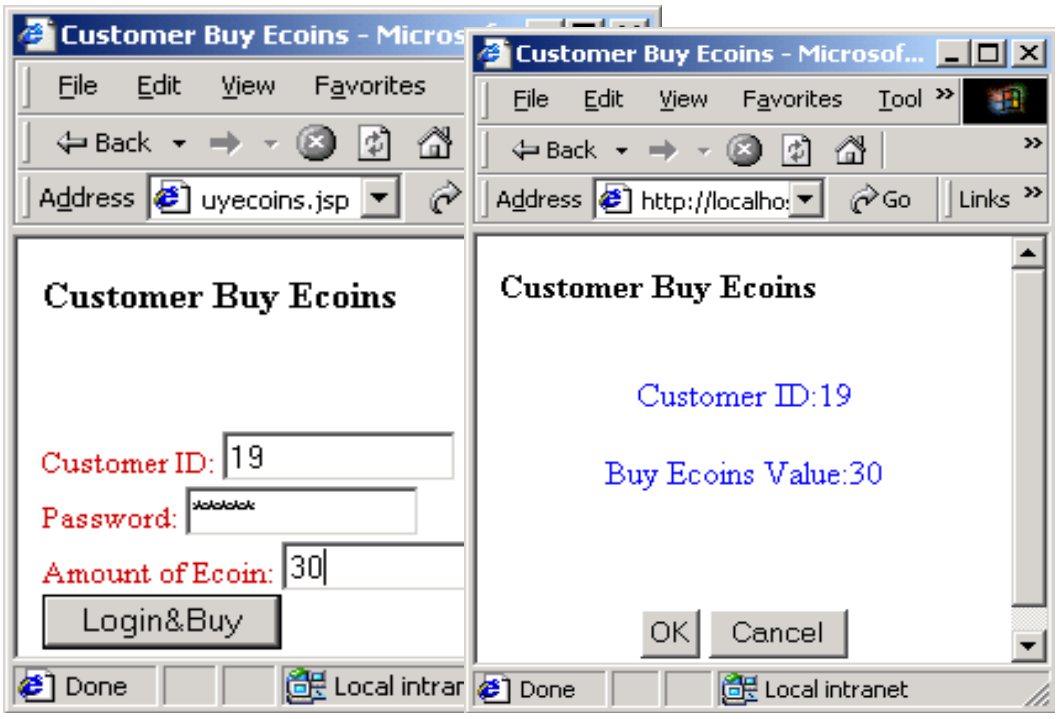


Figure 4-4 Example customer buys e-coins

Table 4-4 Debit E-coins Use Case

Use Case Name:	Debit E-coin
Description:	Used by customers to buy news articles via e-newspaper site.
Event Flows:	1. Customer browses through the homepage of the online newspaper site and clicks the title of the desired news article to buy as shown in Figure 4-5(1). 2. E-newspaper system debits the e-coins from e-wallet and verifies the e-coins by using touchstone. If the e-coins are invalid or not enough e-coins left in the e-wallet, go to 4. 3. Article content and remain e-coins are left in the e-wallet displayed on the screen as shown in Figure 4-5(2). 4. If not enough e-coins left, direct to broker site.
Related Actors/Use Cases:	Used by customer actor
Special Conditions:	

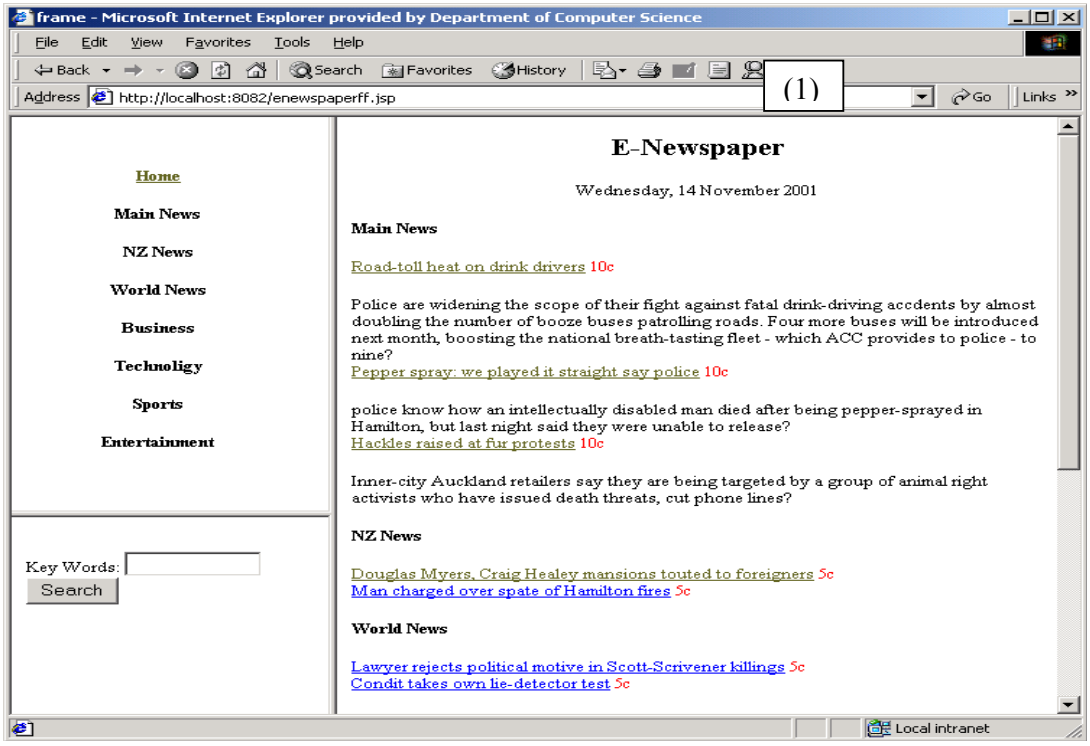


Figure 4-5(1) Example e-newspaper web site

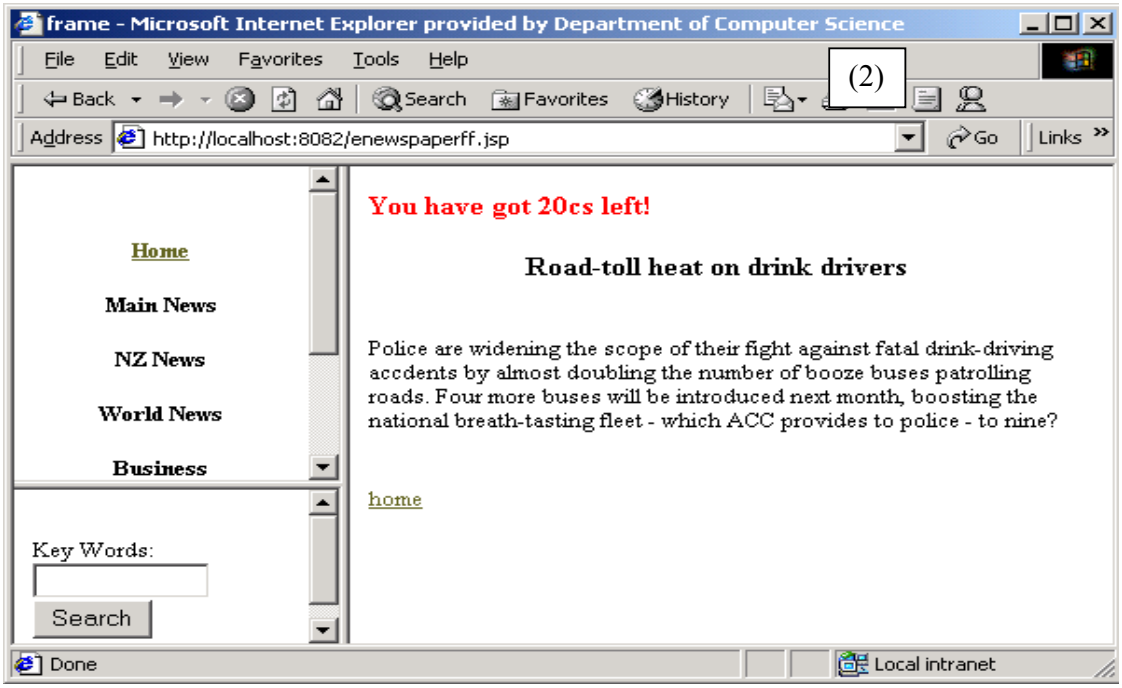


Figure 4-5(2) Example of customer spending E-coins at an E-newspaper site

Table 4-5 Redeem E-coins Use Case

Use Case Name:	Debit E-coin
Description:	Used by e-newspaper sites to redeem e-coins from broker.
Event Flows:	1. At fixed period, the e-newspaper system automatically sorts all the payments received from all customers based on the e-coin ID and sends them to the broker. 2. The total redeemed coin value is received from broker.
Related Actors/Use Cases:	Used by e-newspaper actor
Special Conditions:	

4.3 Non-functional Requirements

Non-functional requirements refer to the restrictions that do not relate directly to the functions or operations to be performed by the system [79]. They are the constraints on system operations. In this system, non-functional requirements include:

Performance Characteristics

1. Response time should be less than 5 seconds to get information goods under heavy loading of up to 100 simultaneous customer connections, for article search regardless of WAN transmission speed.

Quality Issues

2. Online access to either the broker system or the newspaper systems is not very critical, but downtime should be limited to 1 hour.

3. Selling content is critical – downtime must be less than 10 minutes.

4. In case of delays or error, the system should indicate this by some sort of visual display.
5. Power failure needs to be handled by UPS for the broker's and vendors' main server machine and database.

Security and Integrity Issues

6. Customer requests especially for payment information should be encrypted using PGP which is standard for Pretty Good Privacy. It can encrypt messages so that an unintended recipient person cannot read it. When encrypted, the message looks like a meaningless jumble of random characters.

7. Only broker can generate e-coins and change account data of customer and vendors.

8. The web server should be secured very carefully covering any known and popular security holes and exploitation points.

9. The local unit running the web site is responsible for the security, and should take care to ensure that it doesn't become a potential point of entry into the entire system. Otherwise adversaries can use the e-wallet to click and buy articles.

10. Backups should be done every night.

4.4 NetPay OOA Modelling

OOA modelling mainly focuses on class modelling which describes static system structure and sequence modelling that simulates a system from dynamic behaviours. Developers can gain a very good understanding of a system's functional properties [83]. In this section, the static NetPay system structure will be described in a class diagram and sequence diagrams will be used to describe dynamic behaviours among the objects of NetPay system.

4.4.1 Class Modelling

The class modelling focuses on static system structure in terms of classes and association. A class diagram describes the classes of the system, their inter-relationships, and the operations and attributes of the classes [83]. Class diagrams are typically used to:

- Explore domain concepts in the form of a domain model
- Analyze requirements in the form of a conceptual/analysis model
- Depict the detailed design of object-oriented or object-based software

To do this the basic sets of objects that are needed to build these distributed systems should be found and their relationships are determined. Based on the use cases and scenarios discussed in Section 4.2 and 4.3, the OOA-level class diagram for an E-newspaper with NetPay micro-payment system was built, which is shown in Figure 4-6.

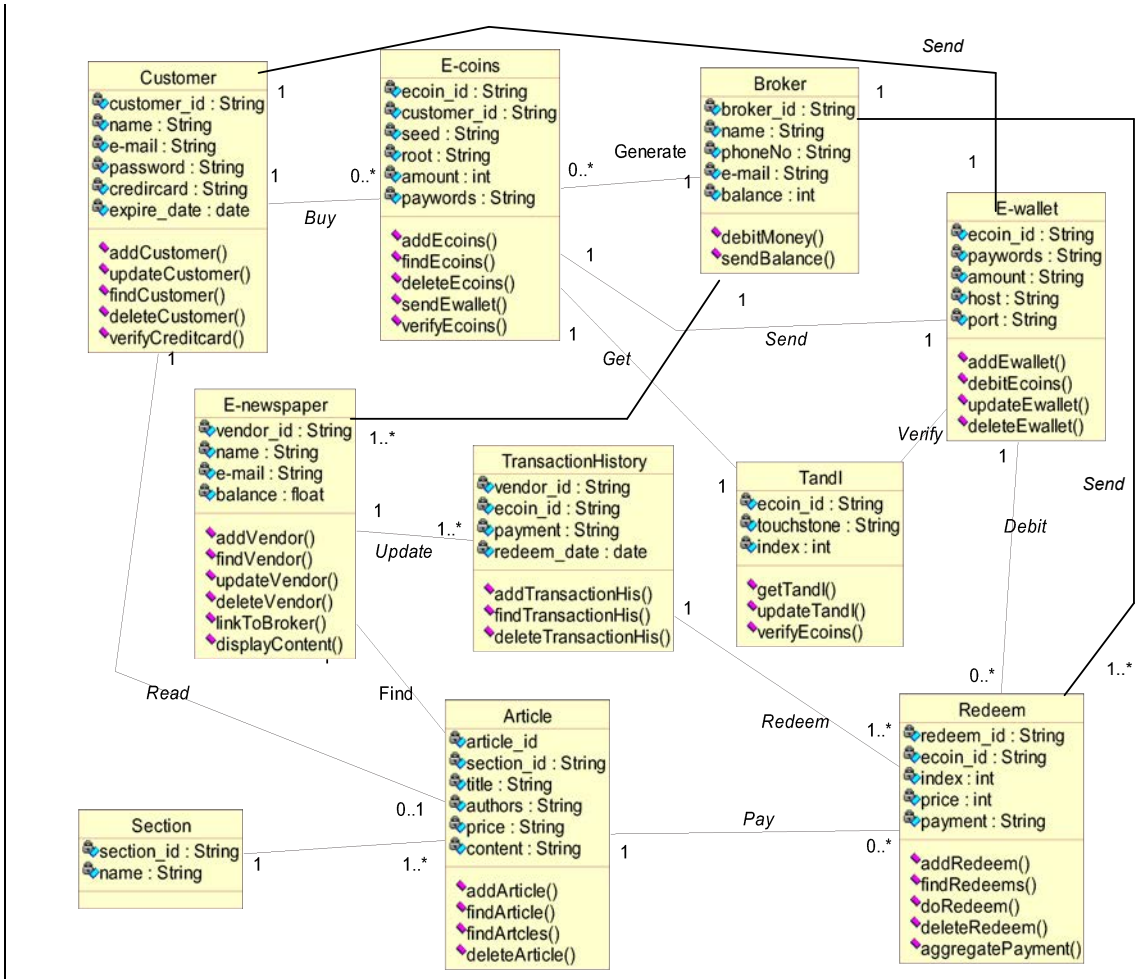


Figure 4-6 E-newspaper with NetPay micro-payment system class diagram

- **Customer:** This object encapsulates all the information of customer's personal and account information. Customer purchasing news articles need to register an account with the broker. This object includes customer_id, name, e-mail address, and password. It also provides a set of business methods to access the information.

- **E-coins:** This object records bought e-coins information which includes ecoin_id, root, seed, paywords, amount (the number of e-coins the customer bought for) and the information access methods.

- **Broker:** This object encapsulates all the information of broker account information. Broker needs to debit money from customers and deal with redeeming e-coins with vendors. This object includes broker_id, name, phone number, e-mail address, and balance.

- **E-wallet:** This object is generated from e-coins object to record customer's current e-wallet information which contains ecoin_id, touchstone, index, and paywords. It has functions for inserting, updating and deleting e-wallet.

- **TandI:** This object records the touchstone and index information and is used to verify e-coins by vendors.

- **Article:** This object contains all the properties of news articles which are an article ID, title, price, and content. It also provides the functions for searching, displaying, adding and deleting article information.

- **E-newspaper:** This object describes E-newspaper information which includes vendor_id, name, e-mail address, and balance. It also specifies a set of business methods to access its information.

- **Redeem:** This object contains all the information of e-coin payments by customers including redeem_id, ecoin_id, index, price, payment. It also provides the functions for adding, finding, and deleting payment information.

- **TransactionHistory:** This object describes all the redeem information including history_id, vendor_id, ecoin_id, payment and redeem date. It also provides the functions for adding, selecting, and deleting redeem information.

4.4.2 Sequence Modelling

A class modelling captures only the static structure of a system. In most cases, a description in terms of dynamic behaviours is needed. This section describes the interaction among the objects of NetPay system with four UML sequence diagrams.

1. *Buy E-coins*

   Figure 4-7 shows the sequence diagram for a registered customer actor who wants to buy e-coins using a credit card.

- Customer logins by entering customer ID, password, and amount of e-coins before purchasing e-coin. New customers need to register first.

- Customer clicks on Login&Buy button, the broker system displays customer ID and amount in order to confirm the purchasing.

- Customer clicks OK button, the broker system debits money from the customer's credit card through a macro-payment system.

- Broker system then generates a unique e-coin ID and an e-coin chain, and records the e-coin data.

- When all done, the broker system sends an e-wallet to the customer.

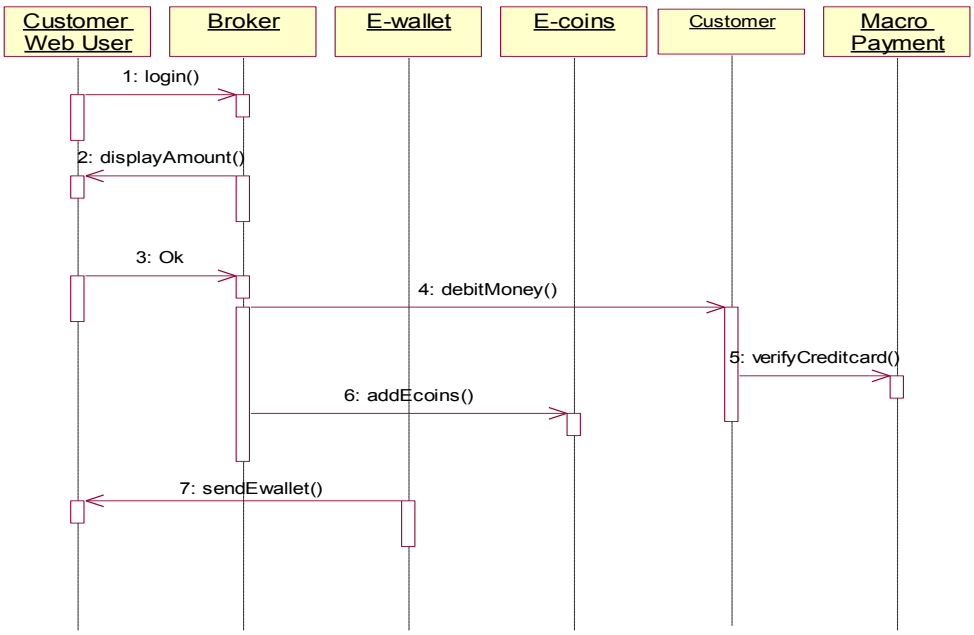


Figure 4-7 Buy e-coin sequence diagram

2. *Buy Content*

Figure 4-8 shows the sequence diagram for a customer actor who wants to buy article content with E-newspaper1 using an e-wallet.

- Customer clicks on the title of a desired article, the E-newspaper1 system debits e-coins and gets Broker's host and port from customer's e-wallet.

- If Enewspaper1 has not obtained TandI, the E-newspaper1 system gets the touchstone and index from an e-coin object in the broker system and verifies the e-coins.

- If there are no e-coins left, the customer is directed to buy e-coins with the broker.

- If the e-coins are valid, the E-newspaper1 system generates a unique redeem ID and records the redeem data.

- E-newspaper1 system then displays article content to the customer.

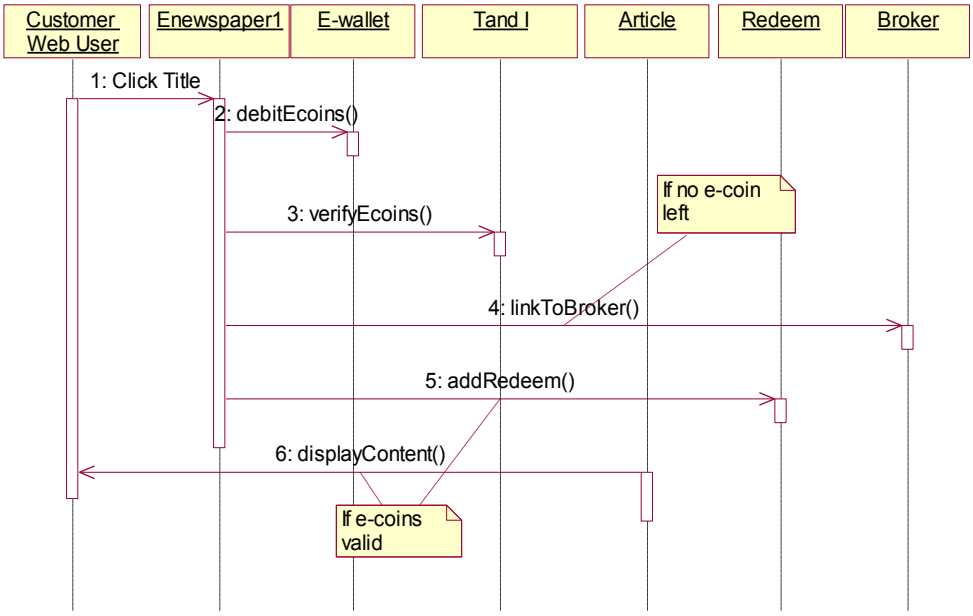


Figure 4-8 Buy article content with E-newspaper1 sequence diagram

3. *Different Vendor*

Figure 4-9 shows the sequence diagram for the customer actor who wants to change to an E-newspaper2 site to buy an article content using the same e-wallet.

- Customer clicks on the title of a desired article, the E-newspaper2 system debits e-coins and gets the E-newspaper1's host and port from the customer's e-wallet.

- If there is no e-coin left, the customer buys e-coins with the broker.

- If TandI does not exist, the E-newspaper2 system gets the touchstone and the index from TandI object in the E-newspaper1 system and verifies the e-coins.

- If the e-coins are valid, the E-newspaper2 system generates a unique redeem ID and records the redeem data.

- E-newspaper2 system then displays article content to the customer.

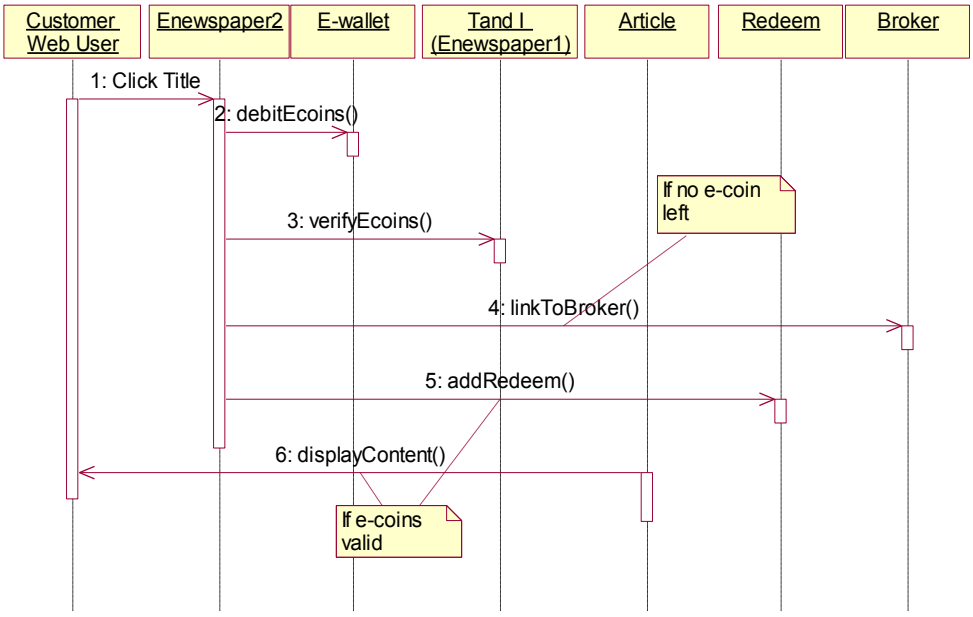


Figure 4-9 Buy article content with E-newspaper2 sequence diagram

4. *Redeem E-coins*

Figure 4-10 shows the sequence diagram for an E-newspaper actor who wants to redeem e-coins with the broker.

- E-newspaper staff clicks Redeem item, the system aggregates all redeem data by using ecoin_id and displays it on the screen.

- E-newspaper staff selects an ecoin_id and the system sends all redeem data related to the ecoin_id to the broker.

- Broker system gets a touchstone of the e-coin from the e-coins object and verifies the redeem data.

- If all redeem data are valid, the broker system generates a unique history_id and records the transaction history data.

- Broker system then sends updated balance to the E-newspaper.

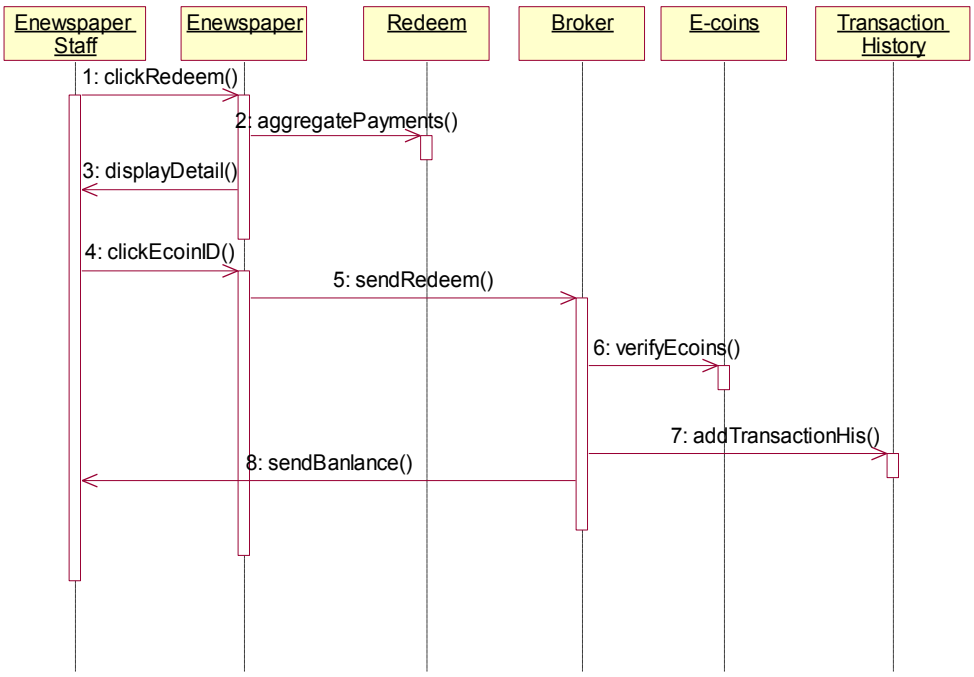


Figure 4-10 Redeem e-coins with the broker sequence diagram

4.5 Summary

NetPay, as a secure, cheap, widely available, and debit-based protocol of a micro-payment system, is a novel research work. Compared with the conventional micro-payment systems, the development requirements of NetPay are rather different. Therefore, a specific description on these requirements is essential to determine the viability of NetPay system.

System requirements define the goal of a system and provide the description of this system's behaviours. We applied UML use cases which include use case diagrams and description to capture functional requirements for the e-newspaper with a NetPay micro-payment system. UML class diagrams were used to describe the type of the objects and static relationship among them in the system. Interactions among the objects of NetPay system are described with UML sequence diagrams. These system requirements are defined and gathered for the following design phase, and two different concrete designs of the NetPay system were created based on the results of the analysis. These are described in detail in the following two chapters.

Chapter 5

CORBA-based NetPay

Based on the NetPay system requirements and functional specification from the previous chapter, we introduce a system design for NetPay to describe how this specification is to be implemented. This task involves several aspects. Firstly, we describe the software architecture that we have developed for NetPay for deployment with thin-client vendor interfaces, i.e. HTML interfaces for customers. Secondly, the design of a NetPay broker and "hard-Coded" E-newspaper click-pay system are described in detail. Thirdly, the implementations of two kinds of NetPay prototypes client-side and server-side e-wallets are introduced and user interactions with these prototypes are explained with examples.

5.1 NetPay Architecture

Software architecture is concerned with how to identify software components, on which machines these components are going to run and how they are connected. There are two main distributed system architectures which include thin-client and thick-client for three- or multi-tier distributed client/server architecture.

5.1.1 Thin-client vs. Thick-client Architecture

Thin-client (server-based) architecture allows a client to exchange data with a server with minimal processing at the client level. It eliminates any download and installation of user interface software on the user's machine. Because the client user interface directly runs from any browser, it avoids the potential problem when any new software is run on a user's machine. The server side programs are easier to update and revise with a thin-client architecture. The main disadvantages of thin client architecture are that it limits user interface capability and the response speed of the system may be slower since the client has to be always connected to the server.

Thick-client architecture is where the client has local GUI and some business logic on the client device. Since the client is a program installed on the device, it provides rich screen

functionality. The client does not always require being connecting with the server. However users are required to download and install software on client side. This is the biggest disadvantage of all thick clients. Client software requires redistribution to user's PC when a user interface is updated or introduced new features by a server. Even with the extreme care used in constructing installation scripts, the variation in users' machines is enough to cause some problems.

The main purpose of the NetPay micro-payment system is that any registered customer clicks to buy on-line information. As we identified in the previous chapter, ease of use is an important criteria for micro-payment systems. We chose to use a thin-client technology to implement our customer clients – HTML browsers. This allows for a very wide range of customers using standard web browser software, without the need for separate installation of browsing and micro-payment clients. Thin-client architecture is faulted for having a single point of failure that affects the entire system. Modern systems however employ controller redundancy to reduce the risk of failure. NetPay is an off-line micro-payment system that allows customers to purchase information from vendors on the WWW without having to involve a broker in every transaction. So vendor systems can in fact be scaled to meet system size requirements.

5.1.2 NetPay Architecture

In order to develop NetPay broker and NetPay-based vendor systems, software architecture should be considered carefully. The architecture should be scalable, reliable, secure and flexible. The most common software architecture is the client-server architecture [11]. Generally a client-server architecture is divided into 3 tiers which are an interface tier, a processing tier and a data storage tier. Each tier is responsible for a separate task. The interface tier processes user interaction. The processing tier handles requests, responses and other services between client and server. The data storage tier is used to manage shared data. We can split these tiers into a multi-tier architecture. In the multi-tier architecture, the processing tier can be split into a presentation tier and a business logic tier.

A thin-client multi-tier web-based e-commerce system usually consists of a client which is a browser providing the user interface, a Web server which provides the presentation logic,

an application server which handles the business logic and a database server which provides data management as shown in Figure 5-1.

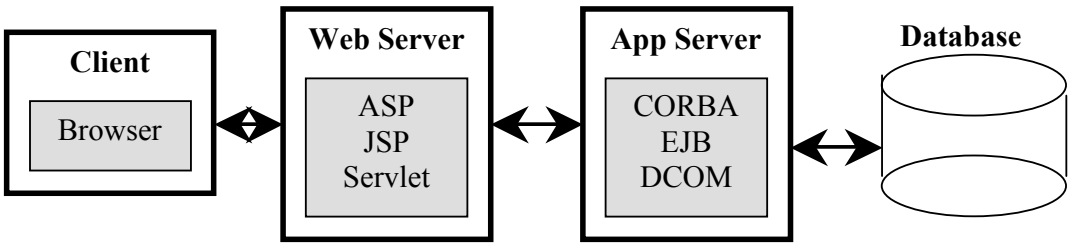


Figure 5-1 Multi-tier Web-based with thin-client architecture

A multi-tier architecture design has many advantages [11]. Partitioning a design into tiers allows designers to choose the appropriate technology for a given situation. It allows development and modifications within one tier without affecting or requiring changes to the other tiers. The key multi-tier benefit is improved scalability since the application servers can be deployed on many machines and the database only requires connections from a smaller number of application servers.

A peer-to-peer network makes it possible for two or more personal computers (PCs) to pool their resources together. Every PC within the network is able to use shared items such as cycles and/or storage space or support collaborative environment. The main difference between peer-to-peer and client-server architectures is that network information is decentralized in a peer-to-peer architecture and centralized in a client-server architecture. Peer-to-peer and client-server are both beneficial architectures for the right type of organization. Peer-to-peer computing has become popular e.g. file-sharing and decentralized processing [50, 95]. Client-server is the best for large-scale organizations that need to be able to grow without worrying about whether the network can handle new users, who need to track tasks in the network for security or other reasons.

From the above discussion, the web-based multi-tier architecture is the most suitable architecture model that meets both NetPay broker and NetPay-based vendor systems requirements and constrains.

The UML provides a simple kind of architecture diagram called a deployment diagram that is made up of machines, machine connections and process names. NetPay micro-payment

transactions involve three key parties: the Broker Server, the Vendor Server, and the Customer browser. This architecture is illustrated in Figure 5-2.

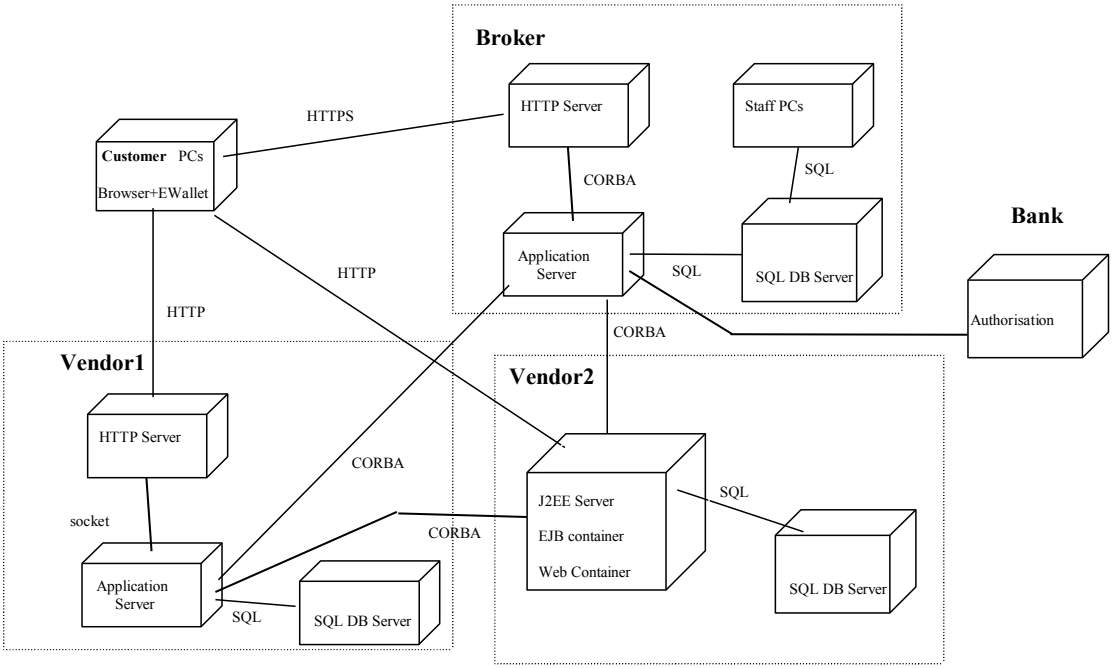


Figure 5-2 NetPay system deployment diagram

In the Broker system, a web browser provides customers with user interfaces and enables customers to access Register and Buy e-coins facilities. The browser connects to a HTTP server, which runs JSPs to handle requests and provides presentation logic. The application server provides business logic and communicating with one or more bank servers to authorise macro-payments. CORBA is used to connect the web tier and the application tier via WAN (Wide Area Network) /LAN (Local Area Network). 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. A LAN is a high-speed data network that covers a relatively small geographic area.

The Broker database stores all customer and vendor account information, generated coins and payments, redeemed coins and macro-payments made which is used to buy coins and redeem money to vendors**.** There are staff PCs running customer and vendor data maintenance applications which connect directly to the database.

A customer runs a web browser that accesses the broker and vendor services, and may also run a client-side e-wallet implemented by the use of a Java application.

Vendors may use quite different architectures. In the Vendor #1 in Figure 5-2, a web browser enables customers to access Search and Buy content services. The web server runs JSPs to provide content that needs to be paid through accessing the customers' e-wallets to obtain e-coins. The application server provides business logic and communicates with the broker and other vendors. CORBA is used to connect the web tier and the application tier via WAN/LAN. A vendor relational database stores vendor data and redeem data.

CORBA is also used to communicate among the broker application server and the vendors' application servers via WAN/LAN. The broker application server provides a set of CORBA interfaces with which vendor application servers communicate to request touchstones and redeem e-coins. A vendor application server communicates with the broker application server to obtain touchstone information to verify the e-coins being spent and to redeem spent e-coins and other vendor application servers to pass on e-coin indexes and touchstones.

Vendor #2 uses a J2EE-based architecture with J2EE server providing Java Server Pages (web services) and Enterprise Java Beans (application server services), along with a relational database to hold vendor data [94]. We will describe the more details about the design and implementation of this vendor application in Chapter 6.

5.2 NetPay E-wallets

Three kinds of e-wallets have been designed in the NetPay systems.

a) Server-side e-wallet: some people prefer to access Internet not only on an individual computer (e.g. businessmen who often travel around). A Server-side hosted e-wallet is suitable for these people. The server-side e-wallet is stored on the vendor server and is transferred from the broker to each vendor when required.

Figure 5-3 shows how a vendor application server debits e-coins from the server-side e-wallet. When a customer clicks title of an article on his/her browser, the web server sends the request to the vendor application server which debits e-coins from the customer's e-wallet paying for the content.

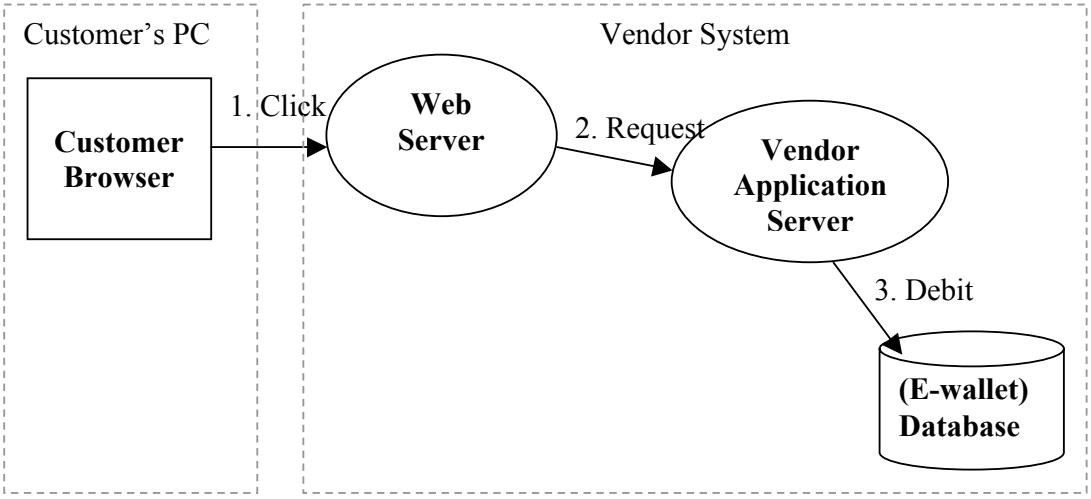


Figure 5-3 Server-side e-wallet

Customers can buy articles using the server-side e-wallet anywhere in the world and the e-coin debiting time is fast on the server-side e-wallet system. However customers are required to remember e-coin IDs and password to log into a newspaper site when they change vendor.

b) Client-side e-wallet: some people prefer to access the Internet using one machine (e.g. housewives who stay home most of the time). A Client-side e-wallet is more suitable for these kinds of people. The client-side e-wallet is an application running on the client PC which holds e-coin information.

Figure 5-4 shows how a vendor application server debits e-coins from the client-side e-wallet. When buying an article content a customer clicks the title of the article on the web browser and then the web server sends the request to the vendor application server. The vendor application server sends the price of the article to the e-wallet application and then the e-wallet application returns the e-coins paying for the content to the vendor application server. The client-side e-wallet avoids need to login. However every click-buy transaction requires communication from vendor web server to customer e-wallet application across the internet.

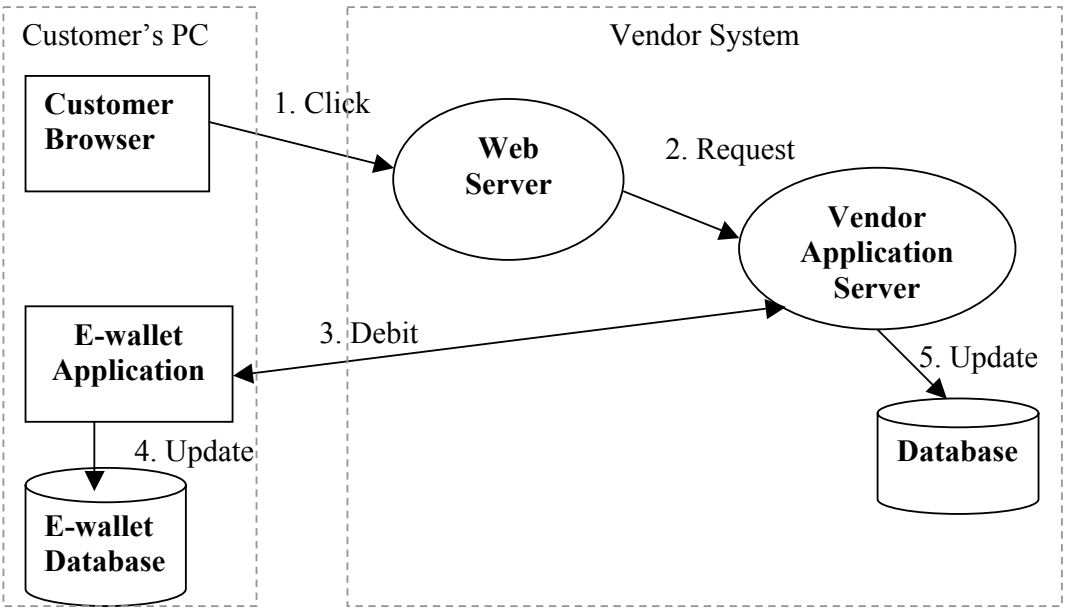


Figure 5-4 Client-side e-wallet

Customers can buy article content using the client-side e-wallet at different newspaper sites without the need to log in after the e-wallet application is downloaded to their PC. The e-coin debiting time is slower for a client-side e-wallet than the server-side e-wallet due to the extra communication between vendor application server and customer PC's e-wallet application.

b) Client-side cookie-based e-wallet: To reduce the e-coin debiting time, we created a temporary cookie e-wallet caching the e-wallet data for debiting instead of the e-wallet database.

Figure 5-5 shows how a vendor application server debits e-coins from such a client-side cookie-based e-wallet. When a customer finds a desired article, he/she clicks the article heading on the web browser. The web server sends the request to the vendor application server. Only for the first time when the customer buys content from the vendor web site does the vendor application server need to get the e-coins from the e-wallet application. It then creates a "cookie" to store the remaining e-coins. Once the cookie is created, the vendor application server debits e-coins from the cookie directly. The e-wallet application could read the cookie to know how many e-coins left when the customer wants to check the balance of the e-wallet. This reduces the need for the vendor application server to

communicate with client PC-based e-wallet, caches the e-coins in HTTP request which holds cookies.

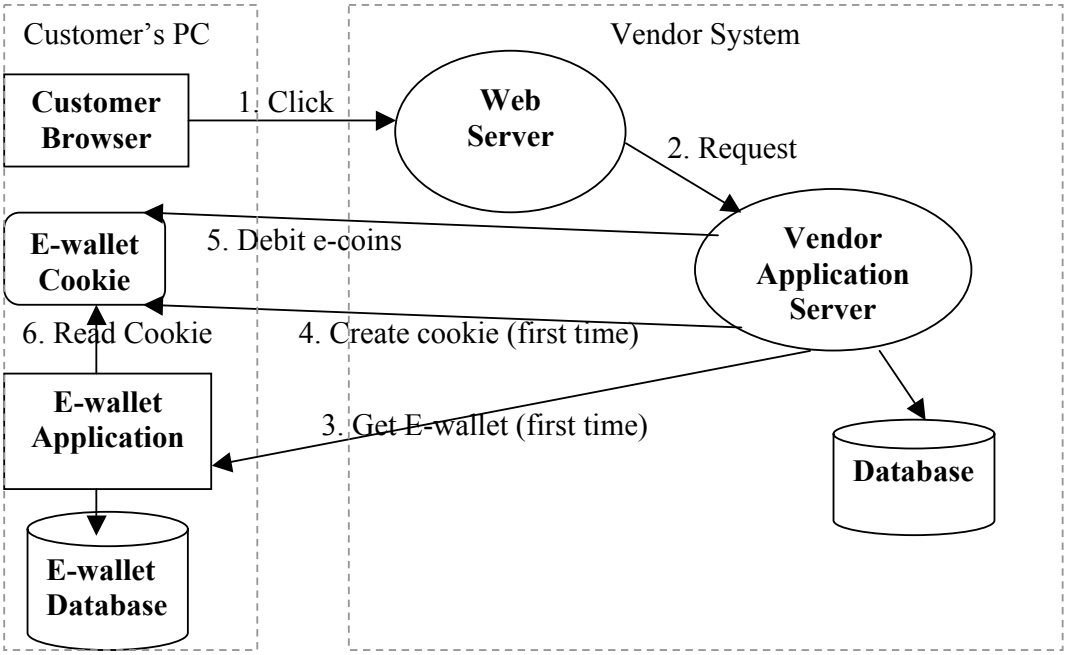


Figure 5-5 Client-side cookie-based e-wallet

When the customer changes to another vendor, the new vendor contacts the previous vendor to request the touchstone and the index of the e-wallet, and the previous vendor application server gets the remaining e-coins from the cookie, stores them back into the e-wallet database and then deletes the cookie.

The e-coin debiting time is medium on client-side cookie-based e-wallet. It is suitable for a customer performing many purchases from a single vendor, then changing to another vendor.

5.3 NetPay Object-oriented Design

Extending the OOA objects and software architecture of the NetPay system, a more detailed OO design will be described in the following sections. Object-oriented design (OOD) is concerned with developing an object-oriented model of a software system to implement the identified requirements [83]. There are two parts in OOD which include static and dynamic OOD.

5.3.1 Static System Design

OOD builds on the products developed during OOA by refining OOA candidate objects into appropriate OOD classes, determining service classes which are based on the software architecture, and implementing message protocols for all objects presented in the OOA [83]. In our current NetPay prototype we use two kinds of e-wallet which are a server-side e-wallet and a client-side e-wallet. The broker application sets the e-wallet which stores the e-coins in the server-side or client-side.

5.3.1.1 Server-side E-wallet NetPay

The server-side e-wallet should be transferred from the broker to each vendor in turn the customer is buying content from. Vendor systems need to know the location of the customer's e-wallet and to get the e-wallet contents. To do this we designed the broker application server so that it provides a set of CORBA interfaces with which the vendor application servers communicate to request an e-wallet location or to get an e-wallet. The vendor application servers also provide a CORBA interface in order for other vendor application servers to get the e-wallet if it has been passed to one of them. The e-wallet is thus passed from vendor to vendor as needed. The problem is that the new vendor can not get the e-wallet when previous vendor crashes.

When a customer first clicks the Login&Buy button to purchase e-coins on the browser, the HTTP server runs JSPs handling the request. The Broker application server communicates with a macro-payment system to debit money from the customer bank account and stores the e-coins information in the database.

When the customer goes to a vendor site, he/she needs to login by entering the e-coin ID and the password. A JSP page handles the login request. If the e-wallet does not exist, the vendor's application server communicates with broker application server via CORBA to get the e-wallet location which includes host and port of the broker or previous vendor. Then it communicates with the broker/previous vendor via CORBA to get the customer's refreshed e-wallet which includes ecoinID, touchstone, index, paywords, and amount. After the customer clicks the article handle, a JSP page deals with a display content request. The

vendor application server debits e-coins from the server-side e-wallet paying for the content. The server-side e-wallet NetPay design feature as illustrated in Figure 5-6.

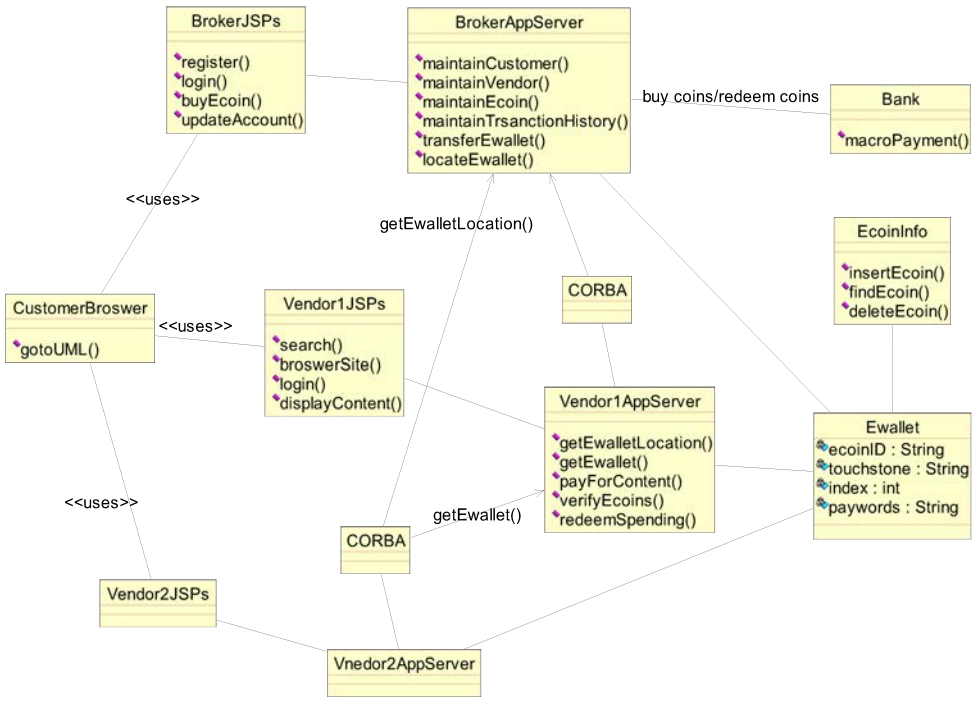


Figure 5-6 Server-side e-wallet NetPay design feature

5.3.1.2 Client-side E-wallet NetPay

The client-side e-wallet is implemented as a Java application runs on the client PC. According to our protocol in Chapter 3, a touchstone and an index (T&I) of a customer's e-coin should be passed from the broker to each vendor. To do this we design that the broker application server provides a CORBA interface vendor application servers communicate with to get the T&I to verify e-coins. The vendor application servers also provide a CORBA interface in order for another vendor application server to communication with it to pass the T&I, avoiding use of the broker where possible.

When a customer first clicks the Login&Buy button to purchase e-coins on the browser, JSPs running on the web server handle the request. The Broker application server communicates with macro-payment system to debit money from the customer bank account and then sends the e-coins to the customer's e-wallet on the customer machine.

A JSP page deals with a display content request when the customer clicks a title of an article. The vendor application server connects with the e-wallet application and sends the price of the content. The customer's e-wallet returns with the e-coins and the location of the T&I to the vendor application server. The vendor application server communicates with the broker or previous vendor via CORBA to obtain the T&I which are used to verify the e-coins. The client-side e-wallet NetPay design feature as illustrated in Figure 5-7.

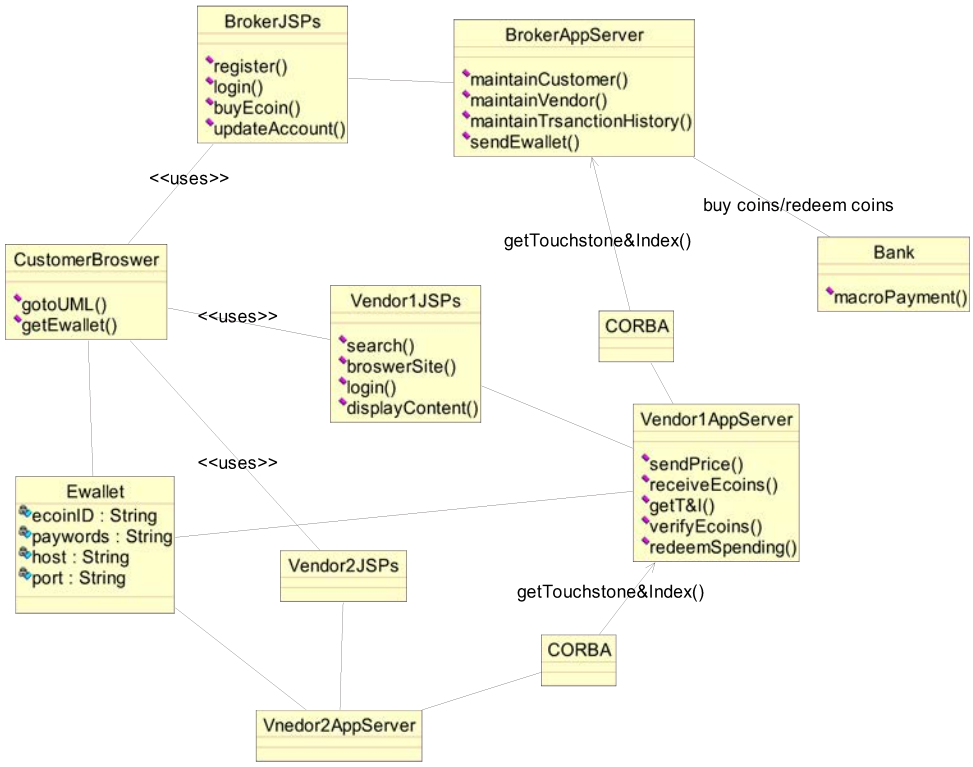


Figure 5-7 Client-side e-wallet NetPay design feature

5.3.1.3 Multi-tier NetPay Architecture

The NetPay broker and the NetPay-based vendor systems are built on the top of the multi-tier web-based architecture shown as the following:

- Client tier (HTML Browser): On the client tier, the web browsers provide consistent and good rendering supports. Nowadays the browsers are almost available on all types of platforms. The browser communicates with the Web server which runs the JSPs.

- Web tier (Web Server and JSPs): In the web tier in the systems, Java Server Pages (JSPs) and JavaBeans are used to service the web browser clients, process request from the clients and generate dynamic content from them. After receiving the client request, the JSPs request information from a JavaBean. The JavaBean can in turn request information from an application server (CORBA). Once the JavaBean generates content the JSPs can query and display the Bean's content.

- Application Server tier (CORBA): In the NetPay broker and hard-coded vendor systems, CORBA is used as the middleware for the application server, which is implemented in the Java language that has a CORBA IDL mapping. We will choose EJBs in the component-based NetPay vendor system that will be described in the next chapter.

- Database Server tier: On the back-end of the systems we use Ms SQL to implement the databases accessed with SQL server via a Java Database Connectivity (JDBC) interface. JDBC, which is a multi-database application programming interface, provides Java applications with a way to connect to and use relational databases. When a Java application interacts with a database, JDBC can be used to open a connection to the database and SQL code is sent to the database.

5.3.1.4 Broker OOD Class Diagrams

The broker system is divided to four parts which are register, buy e-coins, redeem spending and transfer e-wallet. The following are the example system's objects detailed design structure.

- ***Register Objects*** is a four-tier structure and responsible for new customers and vendors register with the broker system as shown in Figure 5-8.

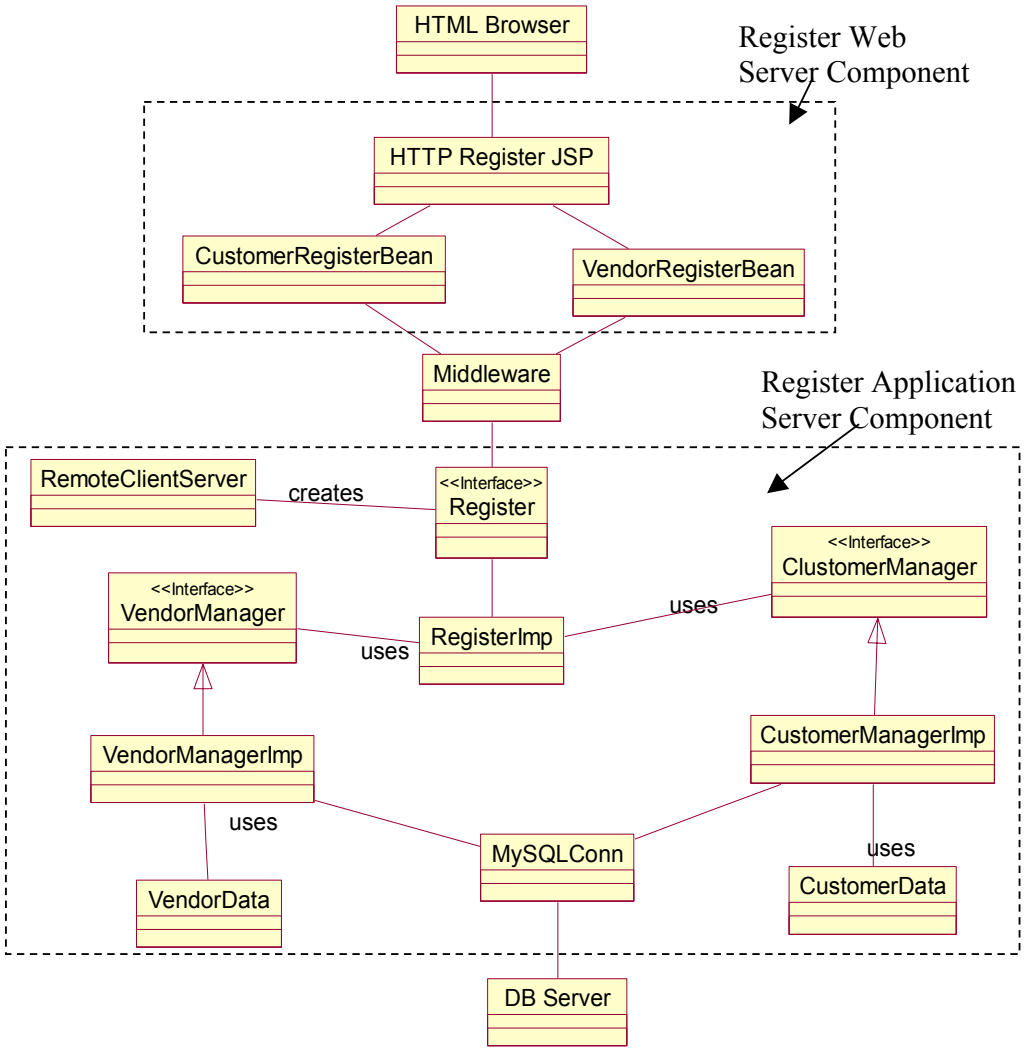


Figure 5-8 Register class diagram

HTTP register JSP deals with requests from HTML client. The requests incorporate receiving customer or vendor information; sending reset/cancel/register requests to bean objects which generate customerID or vendorID when the register action is received; forwarding customerID or vendorID to HTML client.

Customer/VendorRegisterBean provides the functions to register, reset, and cancel customer or vendor information.

Register is an interface which provides functions for new customer and vendor to register or reset.

RegisterImp implements the functions of Register interface.

RemoteRegisterServer creates a remote register object running in application server.

CustomerManager and VendorManager are interfaces which provide insert, select, update and delete functions to manage customer and vendor data.

CustomerManagerImp and VendorManagerImp implement the functions of CustomerManager and VendorManager interfaces.

CustomerData or VendorData holds the account details for a customer or a vendor.

MySQLConn is used by manager objects to connect to DB.

Buy E-coins Objects is a four-tier structure and is used to deal with registered customers purchasing e-coins with the broker system as shown in Figure 5-9.

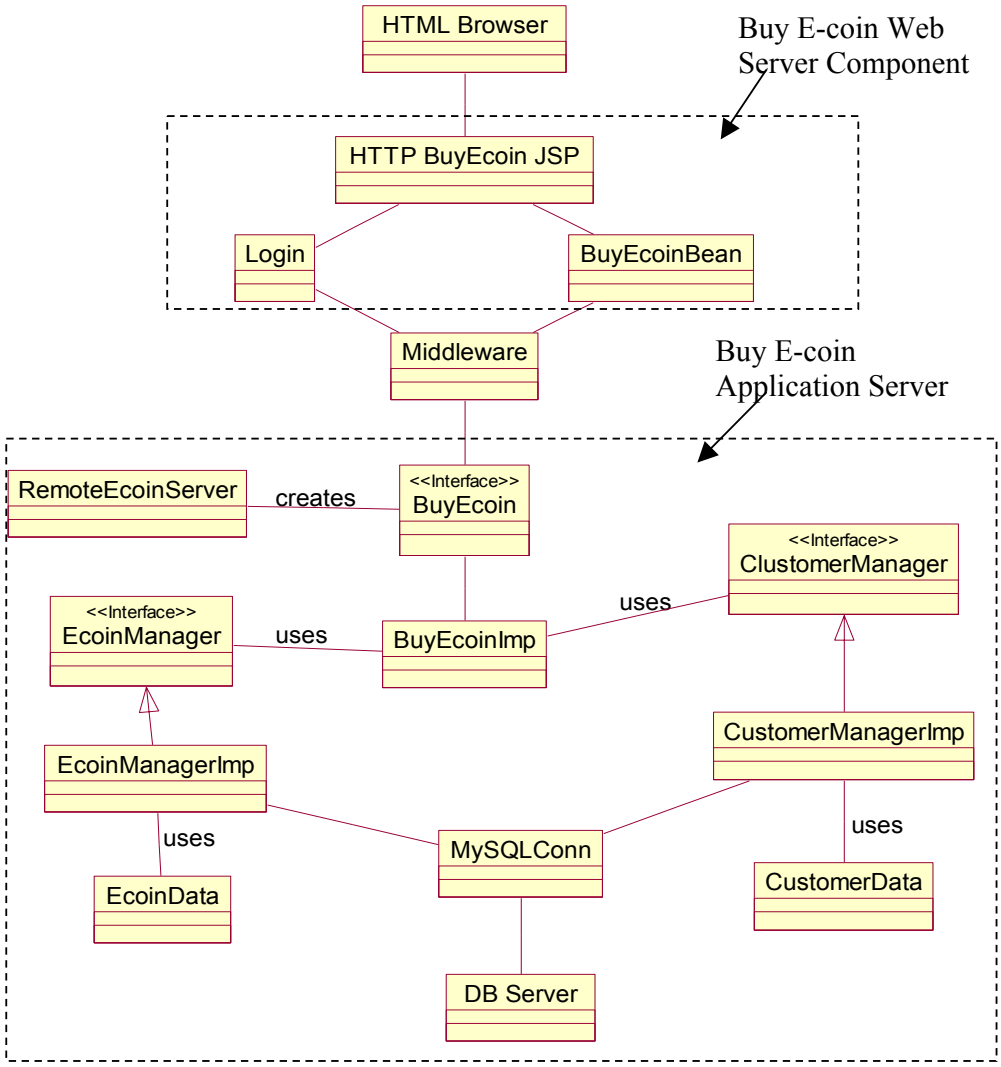


Figure 5-9 Buy e-coin class diagram

HTTP BuyEcoins JSP process requests from HTML client. The requests contain login, logout, sending the customerID, e-coin amount and buy request to bean object, forwarding the ecoinID to HTML client (only for server-side NetPay).

Login provides the functions to process login, logout requests.

BuyEcoinBean provides the functions to deal with buy e-coins request.

BuyEcoins is an interface which provides functions for customer to purchase ecoins using credit card. The functions include login to broker system, debiting amount from customer's

credit card**,** generating ecoin, and sending ewallet to customer PC (only for client-side ewallet).

EcoinManager is an interface which provide insert, select and delete functions to manage customer and vendor data.

EcoinManagerImp implements the functions of EcoinManager interface.

EcoinData contains all attributes of an e-coin.

- ***Redeem Objects*** provide a CORBA interface for vendors' application server communicating with to redeem e-coins and record transaction histories as shown in Figure 5-10.

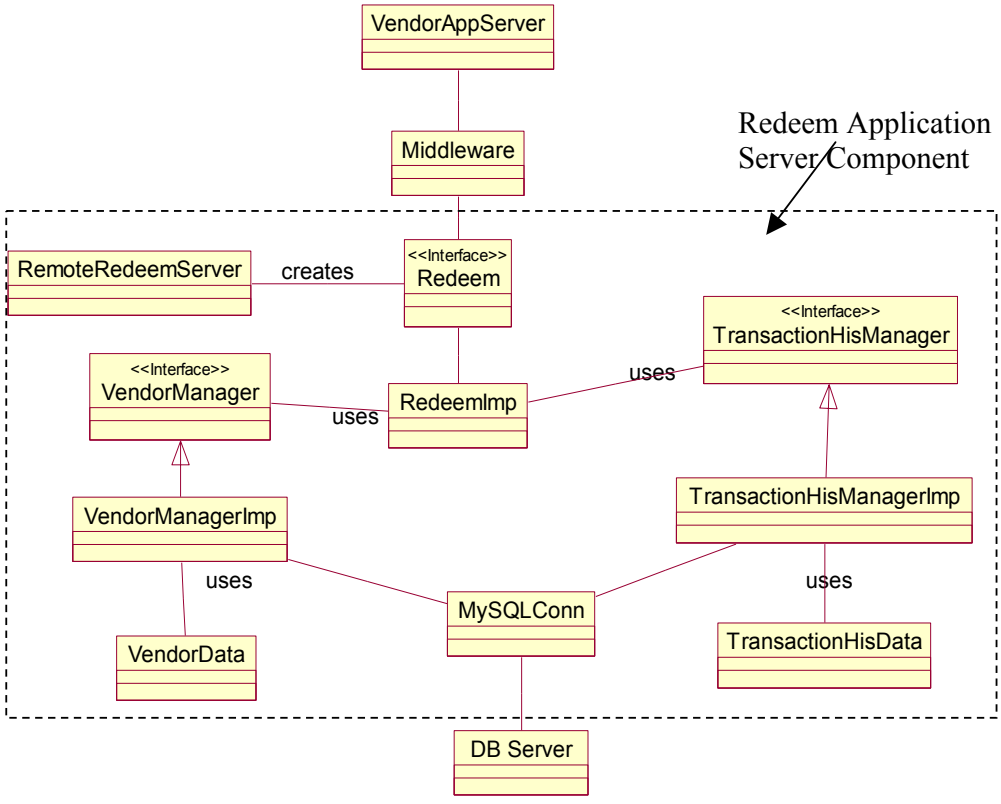


Figure 5-10 Redeem e-coin class diagram

Redeem is an interface which provides functions for a vendor to redeem spent e-coins. The functions include receiving redeem information, verifying spent e-coins, updating vendor account balance, sending balance to vendor and storing transaction history.

TransactionHisManager is an interface which provides insert, select, and delete functions to manager transaction history data.

TransactionHisManagerImp implements the functions of TransactionHisManager interface.
TransactionHisData contains all redeem messages.

- ***Transfer E-wallet Objects*** provide a CORBA interface with which the vendors' application server communicates to get e-wallet locations or e-wallets as shown in Figure 5-11. The objects are used only for server-side e-wallet NetPay broker system.

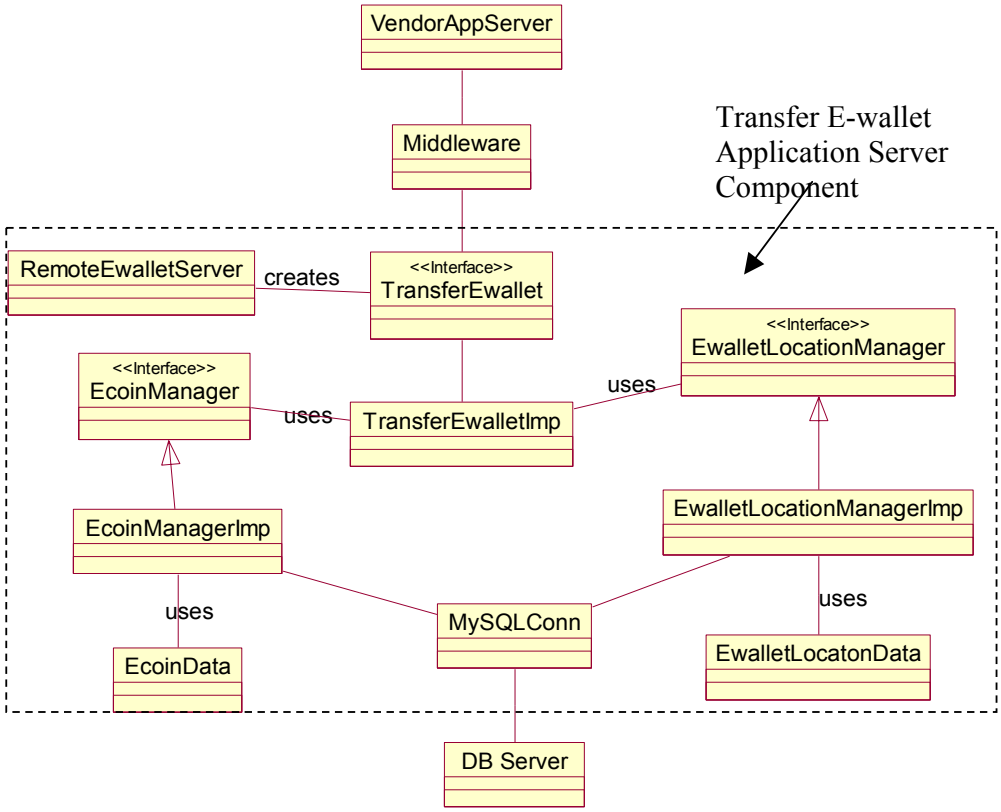


Figure 5-11 Transfer e-wallet class diagram

TransferEwallet is an interface which provides functions for a vendor to request e-wallet information and the e-wallet location including host and port of the previous vendor. The functions contain transferring e-wallet with ecoinID, receiving host and port of current vendor and transferring host and port of the previous vendor.

TransferEwalletImp implements the functions of TransferEwallet interface.

EwalletLocationManager is an interface which provides insert, select, update and delete functions to manager ewallet location data.

EwalletLocationManagerImp implements the functions of EwalletLocationManager interface.
EwalletLocationData holds all attributes for describing an e-wallet location.

- ***Transfer TandI Objects*** provide a CORBA interface with which the vendors' application server communicates to get T&I as shown in Figure 5-12. The objects are used only for client-side e-wallet NetPay broker system.

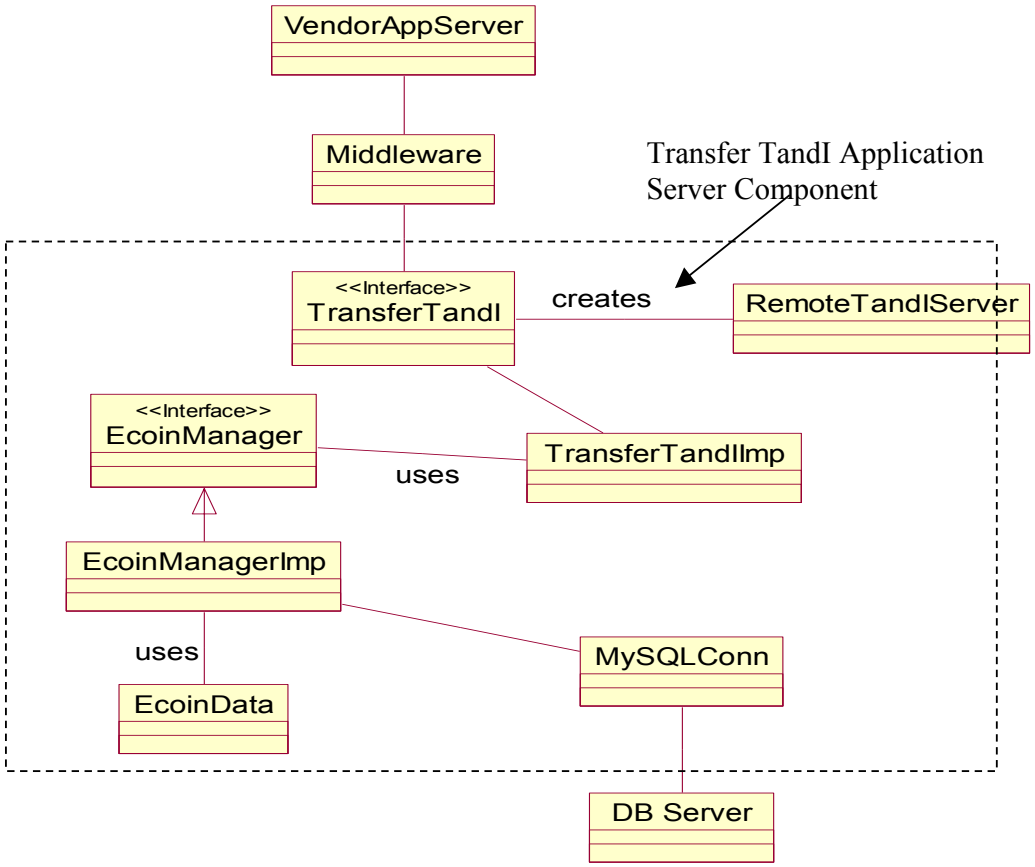


Figure 5-12 Transfer Touchstone and Index class diagram

TransferTandI is an interface which provides functions for a vendor to request TandI information including e-coinID, touchstone, and index of an e-coin. The functions contain transferring e-coinID, touchstone, and index to the vendor.

TransferEwalletImp implements the functions of TransferEwallet interface.

5.3.1.5 Vendor OOD Class Diagrams

Vendors provide on-line services and contents e.g. in our scenario, an e-newspaper which includes newspaper home, search articles and buy content objects. The followings are the objects detailed design structure.

- ***Newspaper Home Object*** is a four-tier structure and allows customers to search and browser articles on newspaper sites as shown in Figure 5-13.

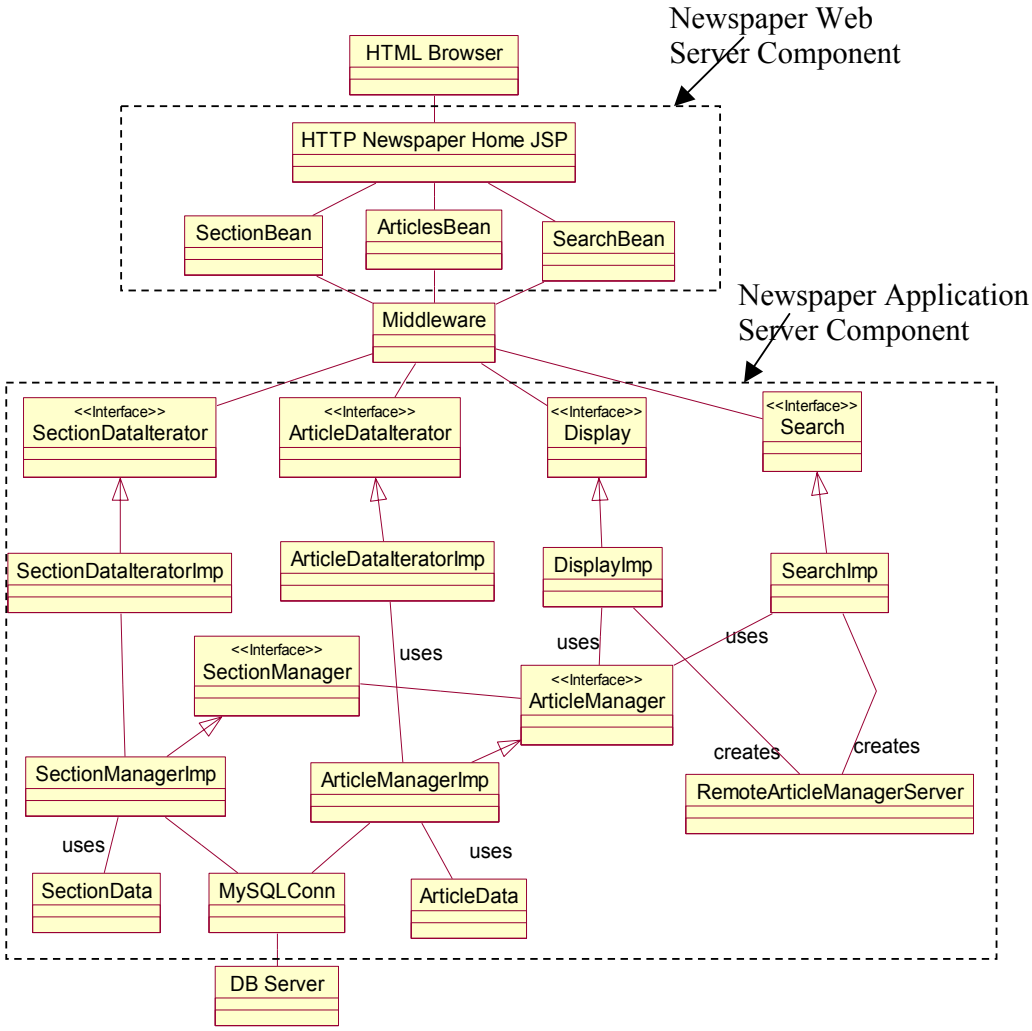


Figure 5-13 Newspaper homepage class diagram

HTTP Newspaper Home JSP handles requests from HTML client which include displaying section name and the corresponding articles; dispatching these requests to bean object; deals with and forwarding responses to client.

SectionBean/ArticleBean contains all functions that enable customer to browser e-newspaper site.

Display is an interface which provides displaying sections and articles functions.

DisplayImp implements the functions of Display interface.

RemoteArticleServer creates display and search objects running in application server.

SectionManager/ArticleManager provides all functions to display, insert, update, and delete sections/articles.

SectionManagerImp/ArticleManagerImp implements the functions of SectionManager/ArticleManager interface.

SectionData/ArticleData holds all attributes for sections and articles.

SectionDataIterator/ArticleDataIterator processes section/article data to client.

SectionDataIteratorImp/ArticleDataIteratorImp implements the functions of SectionDataIterator/ArticleDataIterator interface.

ArticleSearchBean provides all functions that enable customer to search articles by using key words, authors, and prices.

Search contains all functions of the search articles application server which include searching articles by key words, searching articles by authors, and searching articles by prices.

SearchImp implements the functions of Search interface.

Buy Content Object is a four-tier structure and is used to debit e-coins from customers' e-wallet, to store redeem data and to display article contents on the browser as shown in Figure 5-14. The buy content application server communicates with the broker application server to get an e-wallet location and an e-wallet (or a T&I using client-side e-wallet) and other vendor application server to get an e-wallet (or a T&I). The application server provides a CORBA interface with which other vendors' application server communicates. The e-wallet application is used for the application server debiting e-coins when customers use a client-side e-wallet NetPay system.

HTTP Article Content JSP deals with article content display request; deliveries the request to buy content bean object; produces and forwards article content to client.

BuyContentBean provides functions that enable customer to buy article content.

RemoteEwalletServer creates buy content object running in the application server.

BuyContent is an application server-side interface which provides selecting e-wallet, requesting e-wallet from broker or another vendor, debiting e-coins, verifying e-coins, and displaying content functions.

BuyContentImp implements the functions of BuyContent interface.

RedeemManager provides functions to insert, select, and delete redeem data.

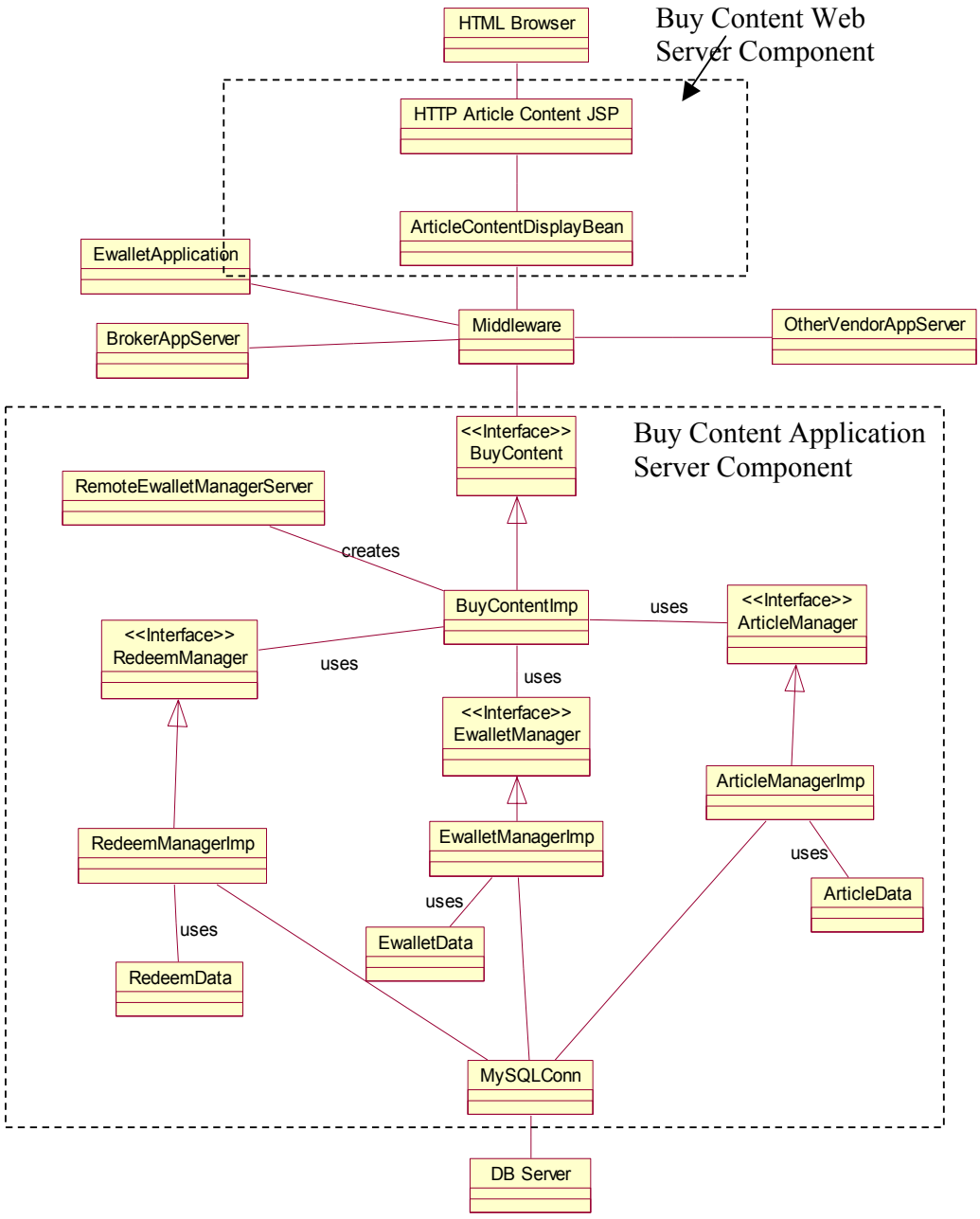


Figure 5-14 Buy content class diagram

RedeemManagerImp implements the functions of RedeemManager.

RedeemData holds all attributes for describing spending e-coins.

- ***Redeem Spending Object*** is a four-tier structure and is used to redeem payments with the broker system as shown in Figure 5-15. The application server sends all payment data to the broker application server.

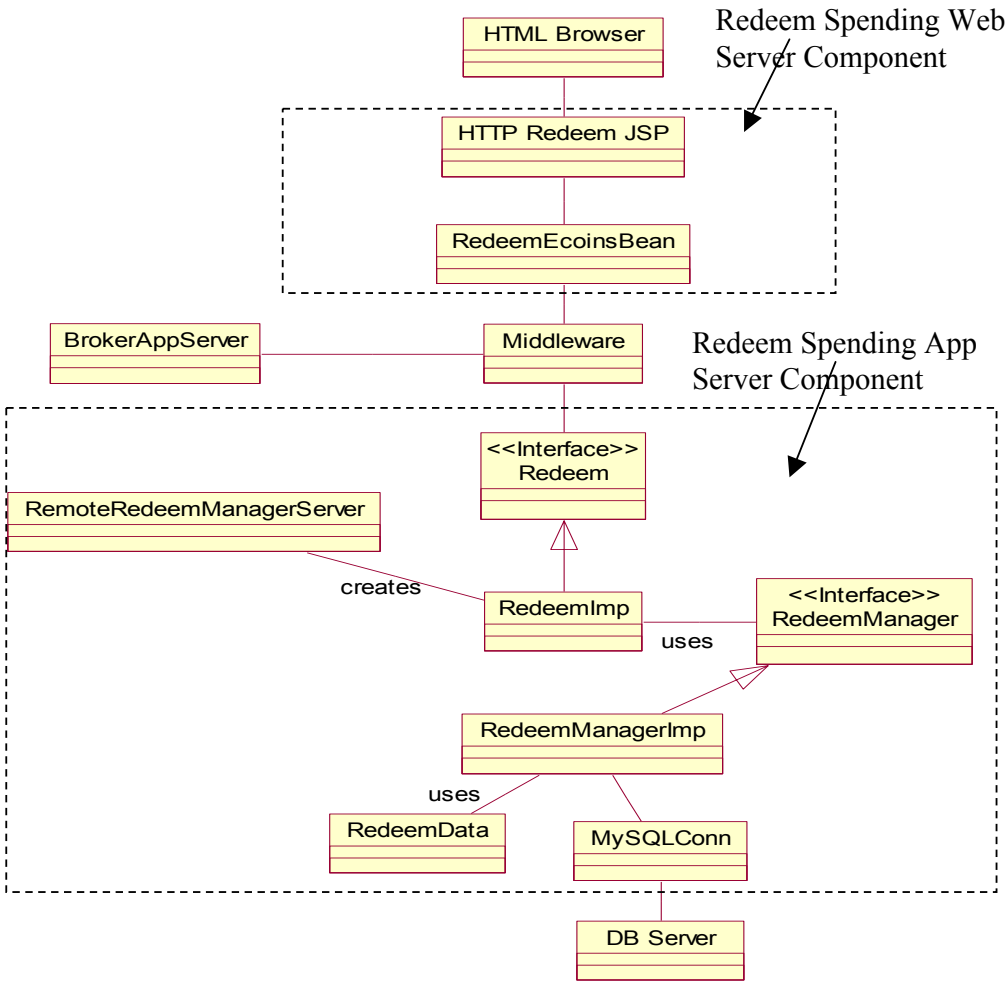


Figure 5-15 Redeem spending class diagram

HTTP Redeem JSP deals with redeem spending request; deliveries the request to a redeem bean object; produces and forwards transaction list and balance to client.

RedeemBean provides functions that enable vendor to redeem spending e-coins.

RemoteRedeemServer creates redeem object running in the application server.

Redeem is an application server-side interface providing functions which contain aggregate payments from all of its e-coins; display them to the vendor; and send payments to the broker for redeeming.

RedeemImp implements the functions of Redeem interface.

5.3.2 Dynamic System Design

The dynamic behaviour of NetPay systems are described using UML sequence diagrams in this section. One of the advantages of this diagram is displaying the order of the events. It gives a sense of how interactions occur between various components in the system and helps developers to determine the system designs that are actually going to work. There are three kinds of NetPay system that are server-side e-wallet, client-side e-wallet, and client-side cookie-based e-wallet. These are described in the following.

5.3.2.1 Server-side E-wallet NetPay

- *Buy E-coins*

The Buy E-coins sequence diagram Figure 5-16, shows how a customer buys e-coins with broker using server-side NetPay. When buying e-coins the customer enters the amount of e-coins, and then clicks buy button. The web browser requests the e-coins from the broker JSP which sends the request to the Java Bean. Java Bean requests the e-coins from Broker Application Server (BAS) who communicates with the macro-payment to debit the customer's credit card. The BAS stores the e-coins in the database for transferring to a vendor later and sending e-coin ID to the customer.

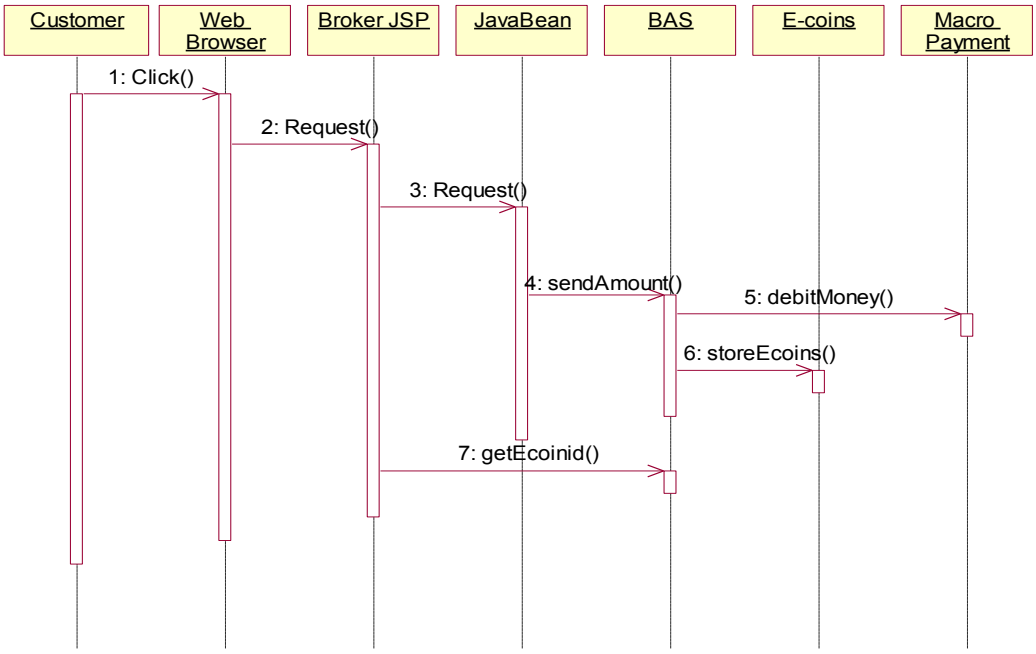


Figure 5-16 Buy e-coins with server-side NetPay sequence diagram

- *Buy Content*

  The buy content sequence diagram Figure 5-17, shows how a customer click-buys article content with E-newspaper site using server-side NetPay. When buying an article the customer selects the article for reading e.g. clicking on the URL in a returned article search page. The web browser requests the article content from the JSP that requests payment for the content of the article from the e-wallet bean. The e-wallet bean communicates with Vendor Application Server (VAS) to debit the customer's e-wallet to pay for the article. If the e-wallet does not exist, VAS requests e-wallet location from the BAS and e-wallet from BAS or VAS. VAS then verifies the e-coins by using the touchstone and the index. If e-coins are valid, VAS stores them in the redeem database and the news article content is displayed to the customer.

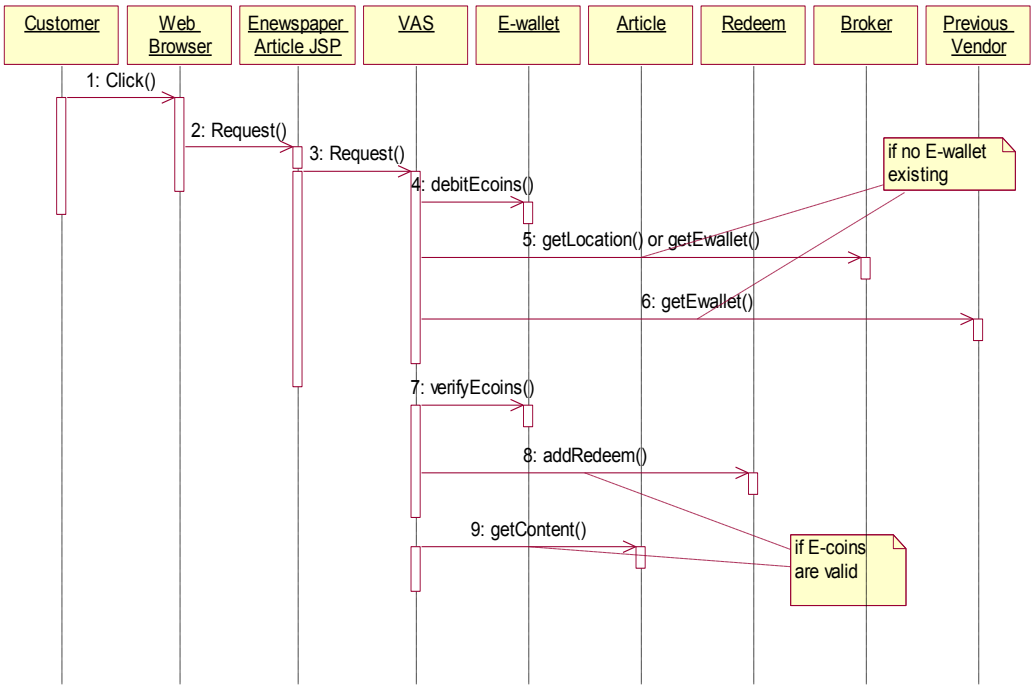


Figure 5-17 Click-buy article content with server-side NetPay sequence diagram

- *Redeem spending*

  The redeem spending sequence diagram Figure 5-18, shows how a vendor redeem spent e-coins with broker using server-side NetPay. When redeeming spent e-coins the vendor clicks an e-coinID item button. VAS selects the payments with the e-coinID end sends all involved information to BAS. The balance for the e-coinID is displayed to the vendor.

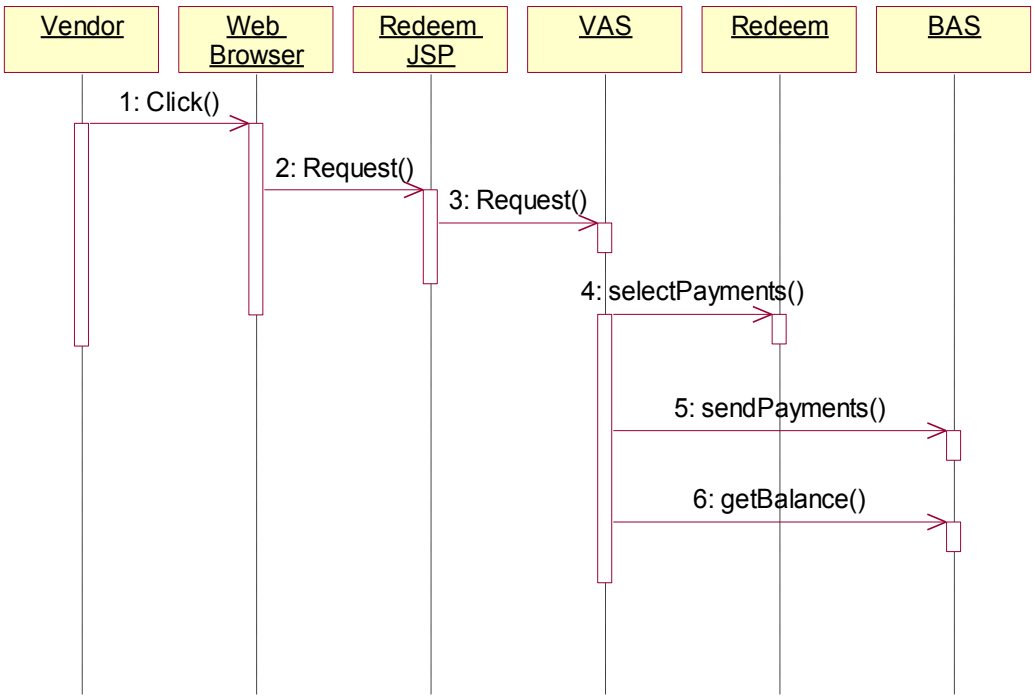


Figure 5-18 Redeem spending with server-side NetPay sequence diagram

5.3.2.2 Client-side E-wallet NetPay

- *Buy E-coins*

Figure 5-19 shows how a customer buys e-coins using client-side NetPay. When buying e-coins a customer logins to the system by entering customer ID and password; then enters the amount of e-coins; and clicks buy button. The web browser requests the e-coins from the JSP which sends the request to the JavaBean. JavaBean requests the macro-payment for the e-coins from BAS to debit the customer's credit card. The BAS stores e-coins in the database, and sends e-coins to e-wallet java application in customer's PC.

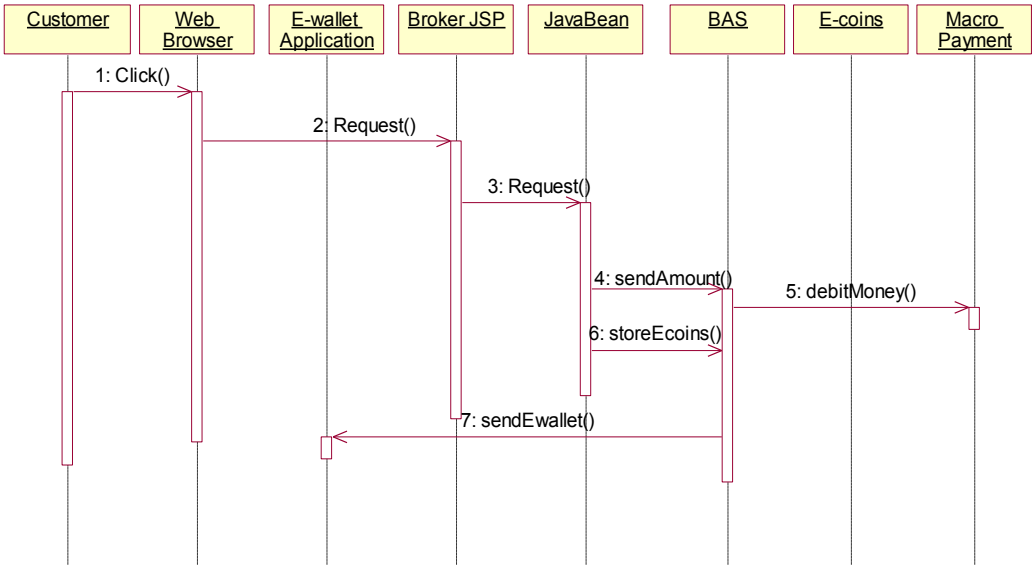


Figure 5-19 Buy e-coins with client-side NetPay sequence diagram

- *Buy Content*

  Figure 5-20 shows how a customer click-buys article content using client-side NetPay. After reading the abstract of an article the customer wants to read its details. The customer clicks the title of the article. The web browser requests the article content from the JSP that requests payment for the content of the article from VAS through e-wallet bean. VAS debits the customer's e-wallet to pay for the article. VAS then sends article price to the e-wallet application and receives corresponding e-coins from it. If the touchstone and the index of the e-wallet are not existed, the VAS requests them from pervious VAS or BAS. VAS verifies the e-coins. If the e-coins are valid, VAS stores them in the redeem database and the news article content is displayed to the customer.

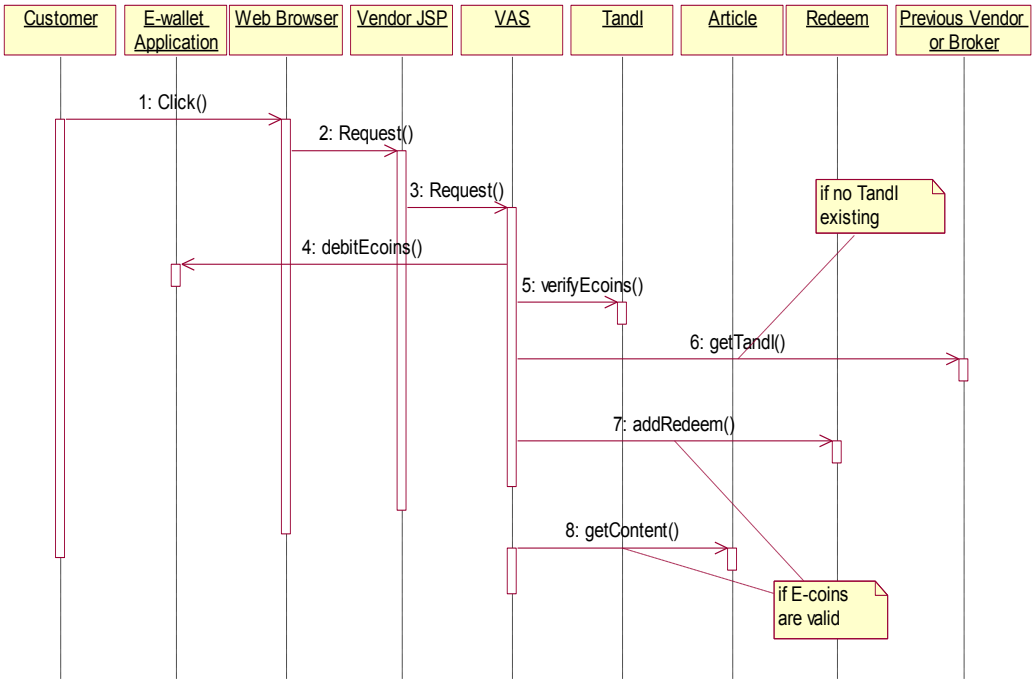


Figure 5-20 Click-buy article content with client-side NetPay sequence diagram

- *Redeem spending* is the same as that in server-side e-wallet NetPay.

5.3.2.3 Client-side Cookie-based E-wallet NetPay

- *Buy E-coins* is the same as that in client-side e-wallet NetPay.

- *Buy Content*

  When a customer finds a desired article, he/she clicks the article heading. The web browser requests the article content from the JSP that requests payment for the content of the article from VAS through e-wallet bean. When the customer first time buy content with the vendor, VAS requests e-wallet information with e-wallet application and stores the e-wallet to the cookie in customer's PC in order to decrease the delay of debiting e-coins. VAS debits the customer's e-wallet to pay for the article. If the touchstone and the index of the e-wallet are not existed, the VAS requests them from pervious VAS or BAS. VAS then debits the e-coins from the cookie e-wallet and verifies the e-coins. If the e-coins are valid, the news article content is displayed to the customer. The sequence diagram Figure 5-21, shows how a customer click-buys article content using client-side cookie-based e-wallet NetPay.

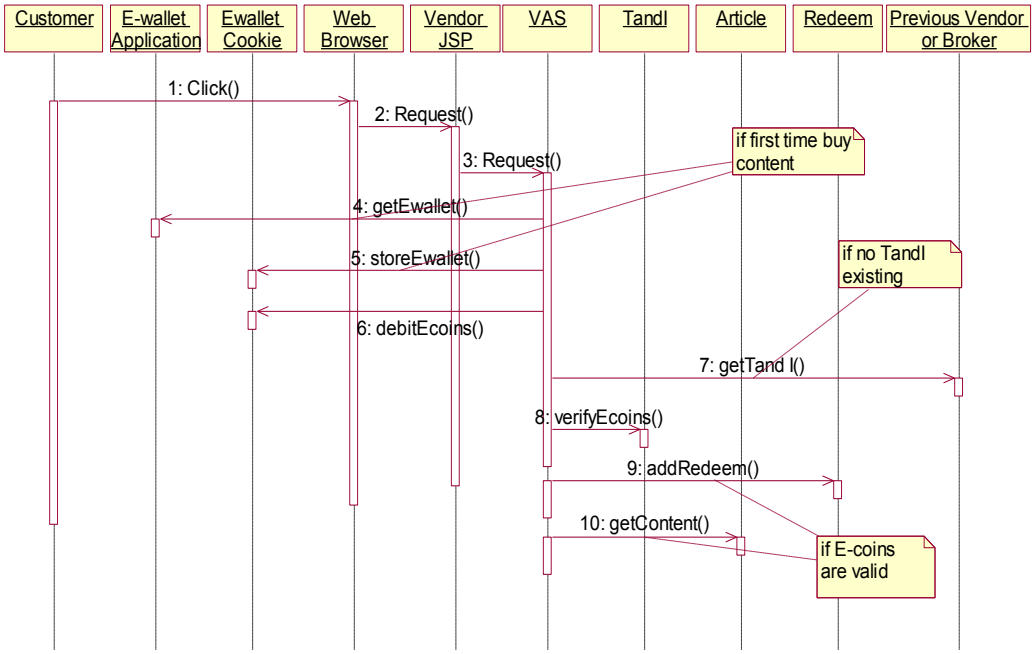


Figure 5-21 Click-buy article content with client-side cookie-based e-wallet NetPay sequence diagram

- *Redeem spending* is the same as that in server-side e-wallet NetPay.

5.4 Database Design

A database is an organized collection of related information. A Database Management System (DBMS) allows a user to store, update and retrieve data in abstract terms and thus make it easy to access and maintain information from a database. Two relational databases which include the broker database and the vendor database are designed in NetPay system in order to persist for the system information storage.

5.4.1 Broker Database

In the broker database, there are eight database tables that can be classified by their purpose which are describing relationships between business entities and recording the next ID.

Describing relationships between business entities: The customer and e-coins tables have a one-to-many relationship: a customer can buy many e-coins, but each e-coin refers to a single customer. The e-coin and transaction history tables have a one-to-many relationship: an e-coin can be involved in many transactions and each transaction only can use one e-coin. The vendor and the transaction history tables have a one-to-many relationship: a customer's e-wallet can make many purchases with a vendor and each transaction only can involve one vendor. Only server-side e-wallet NetPay broker needs to record e-wallet locations for vendors to relocating e-wallets in the e-wallet location table. The Figure 5-22 shows broker database Entity Relationship Diagram (ERD). The types of data in the broker database have been described in Figure 4-6.

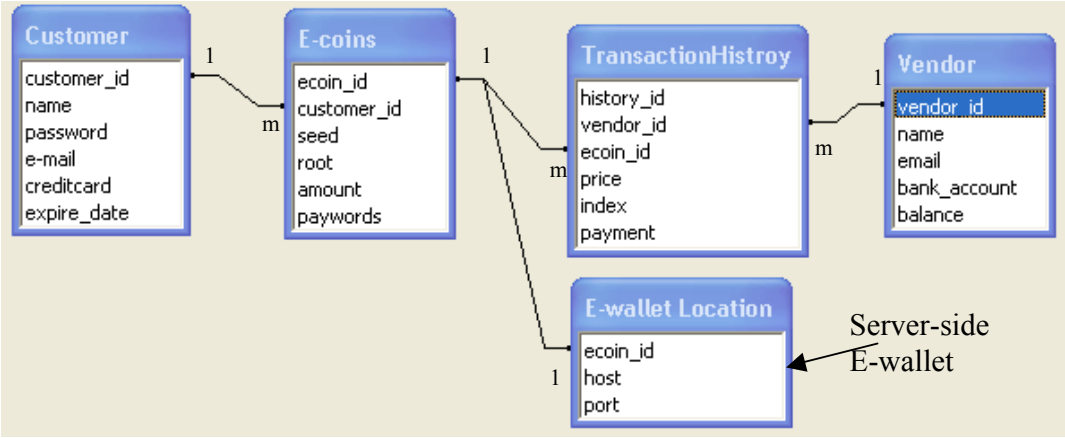


Figure 5-22 Broker system database ERD

Recording the next ID: There are three tables which are next_customer_id, next_vendor_id, and next_ecoin_id. Each of these tables has single column named id. The value of ID is the next primary key of the entities that is used to increment the ID value in the tables.

5.4.2 Vendor Database

In the vendor database, there are five database tables which are section, article, e-wallet, redeem and next_redeem_id. The section and article tables have a one-to-many relationship: a section may have many articles, but each article refers to a single section. The TandI (or e-wallet only for a server-side e-wallet) and redeem tables have a one-to-many relationship: an e-coin may be debited many times for article content payment and

every payment only involves single e-coin. The article and redeem tables have a one-to-many relationship: an article can be paid many times and every payment only can involve one article. The next_redeem_id table has single column named ID. The value of ID is the next primary key of the redeem table that is used to increment the ID value in the table. The client-side e-wallet table is stored at the client PC. The Figure 5-23 shows the relationship between business entities in vendor system. The types of data in the vendor database have been described in Figure 4-6.

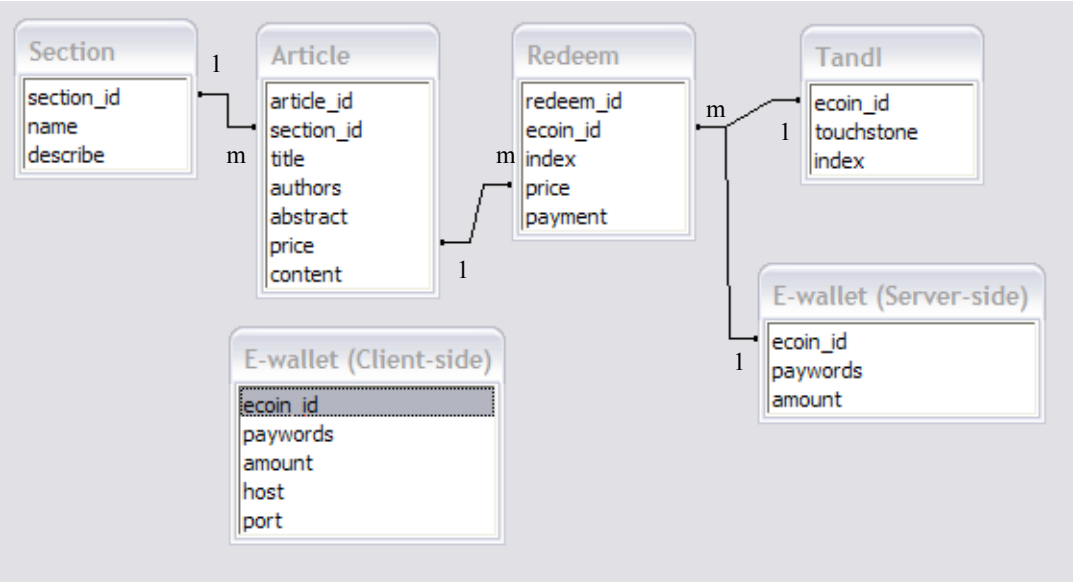


Figure 5-23 Vendor system database ERD

In modified versions of the NetPay prototypes, we choose a transaction temporary file to record the data for redeeming instead of the redeeming database in order to reduce the e-coin debiting time. At a suitable period, the system moves the data in the temporary file to a vender database. The number of transaction records in the redeem table could be large but the records can be deleted after redeeming. This temporary redeem transaction table approach improves vendor server efficiency and is explained in more detail in Chapter 7.

5.5 NetPay Implementation

In this section we focus on the issues of how to build prototypes of NetPay broker and vendor systems based on the design mentioned before. The basic user interfaces of NetPay systems are presented and user interactions are explained with examples.

5.5.1 NetPay Broker

The broker manages customer and vendor accounts, e-coin creation and spent redemption, touchstone supply for e-coin verification, and macro-payment handling for e-coin purchase by customers and payment to vendors for spent e-coins. Our current broker implementation provides a database holding this information, an application server providing these business functions, a CORBA interface for vendor application servers and a JSP-implemented HTML interface for customers.

The CORBA interface allows vendor systems to request e-coin touchstone information (allowing vendors to verify a customer's e-coins) and redeem coins spent at the vendor by customers. We chose to use CORBA to provide a platform and a language-independent interface supporting a wide range of possible vendor implementation technologies.

Two kinds of the NetPay prototype have been implemented. In the Client-side NetPay, e-wallet is hosted on the customer PC and is an application that stores e-coin information for debit by vendor servers and credit by the broker server. In the server-side NetPay, e-wallet is held on the vendor server that the customer is currently buying content from. E-coin information is passed onto a new vendor when the customer moves to that vendor site and makes their first micro-payment purchase.

5.5.1.1 Server-side NetPay Broker

Initially a new customer accesses the broker's web site as shown in Figure 5-24 and clicks registration item to register for opening an account.

Figure 5-24 Server-side NetPay Broker Home

The customer registers by entering the required information and clicks Register button as shown in Figure 5-25. The system verifies any input data and an error message is displayed on the screen if the all requirements are not satisfied. For example the customer enters two different passwords; the customer enters an invalid credit card number; and so on.

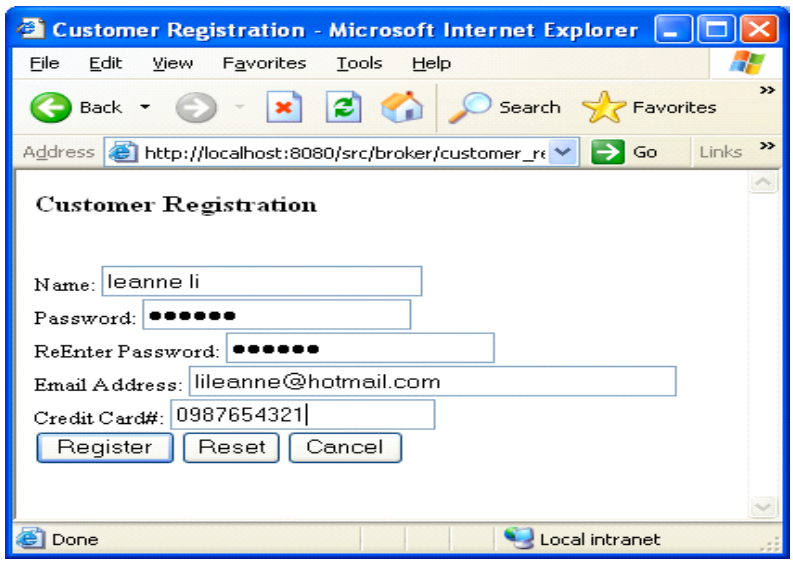


Figure 5-25 Example of HTML customer registration

After the customer successfully register with the system, the broker system generates the customer ID automatically and reminds the customer to remember it for later login to the system as shown in Figure 5-26. The customer then can clicks Buy Ecoin to purchase some e-coins.

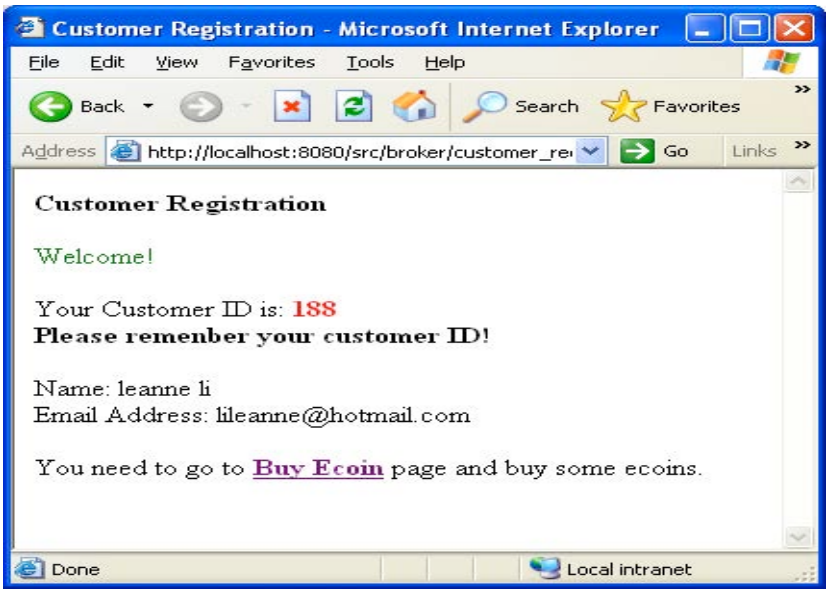


Figure 5-26 Example of HTML registration information

When needing to buy some e-coins, the customer logs into the system and enters amount of the e-coins. The HTML interface is used by customer to purchase e-coins as shown in Figure 5-27. The broker system debits the customer's supplied credit card to pay for the coins by communicating with macro-payment system and then generates e-coins which are stored in the database. The customer needs to remember the e-coinID e.g. 267 for accessing a vendor site.

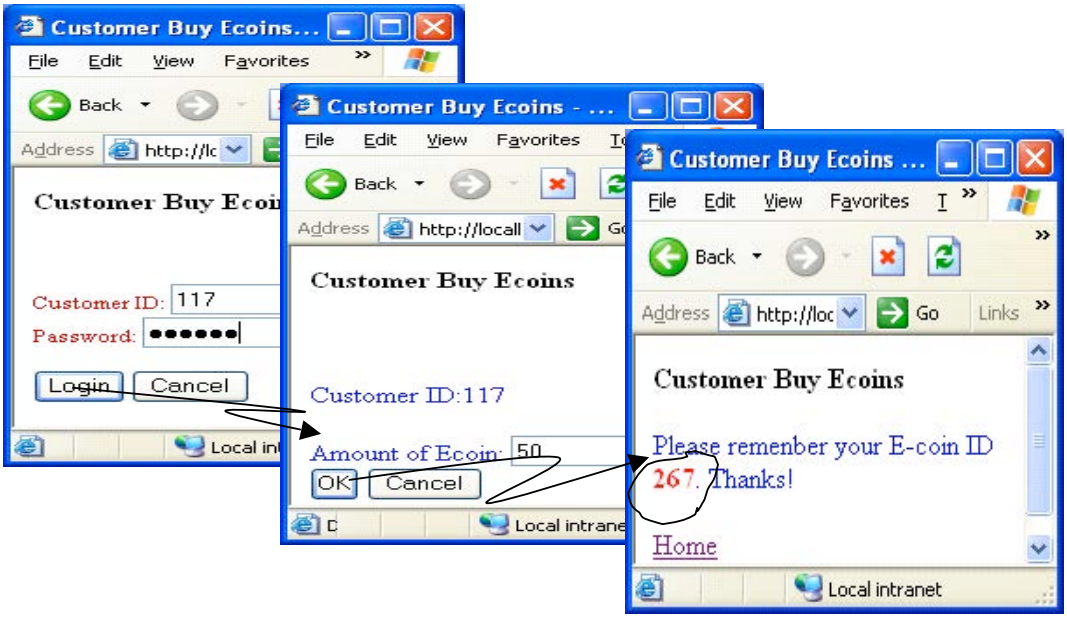


Figure 5-27 Example of HTML customer buy e-coins

5.5.1.2 Client-side NetPay Broker

The HTML interface for client-side NetPay used by customers to register with the broker is the same as that for server-side NetPay, the difference is that the customer using client-side NetPay needs to download and run e-wallet software. The HTML interface for client-side NetPay used by customers to register is shown in Figure 5-28. The customer can register with the broker and then click E-wallet to download e-wallet application software.

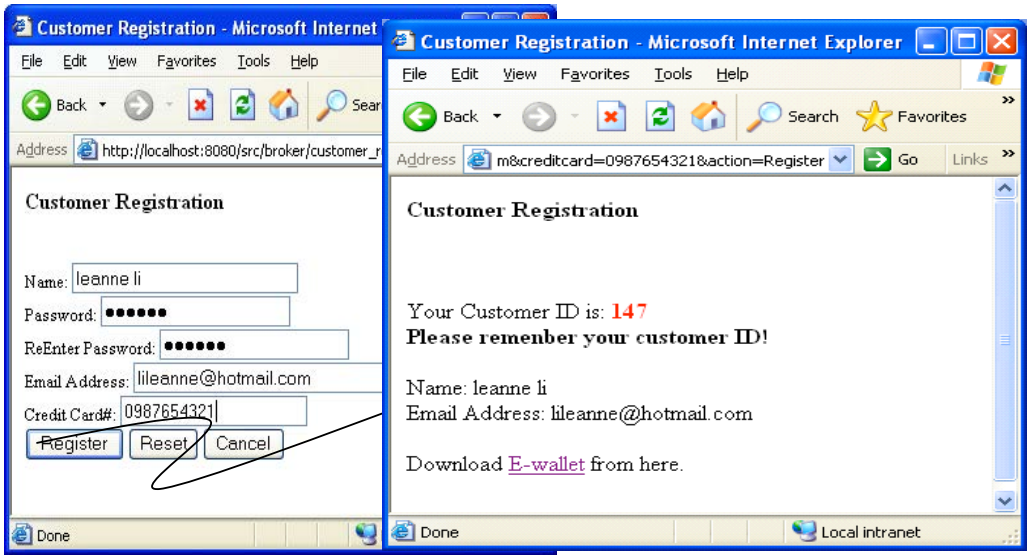


Figure 5-28 Example of HTML customer registration and download e-wallet

The customer then clicks the icon of the e-wallet application to run the software and login to it as shown in Figure 5-29(1). The customer can check the balance of the e-wallet from account menu. As shown in Figure 5-29(2), there is no e-coin in the e-wallet.

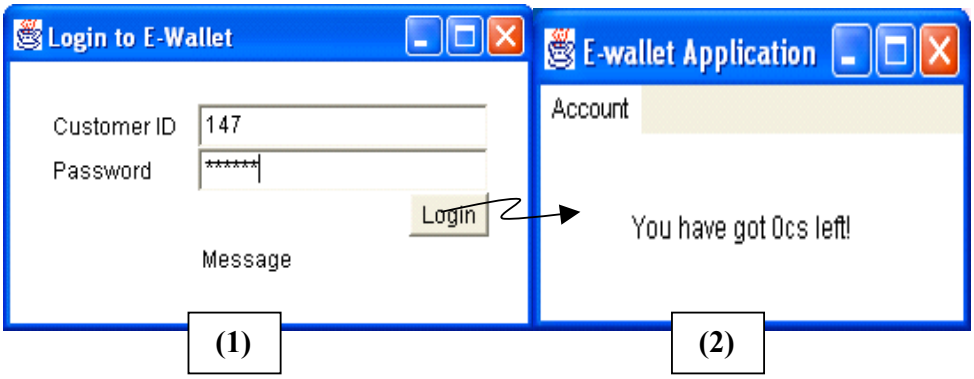


Figure 5-29 Example of e-wallet application

The HTML interface for client-side NetPay used by customers to purchase e-coins with the broker is the same as that for server-side NetPay, but the customers do not need to remember e-coin ID. After a customer login to the system and enters the amount of the e-coins, the e-coins are generated by the system and sent to the customer's e-wallet application on the customer PC as shown in Figure 5-30.

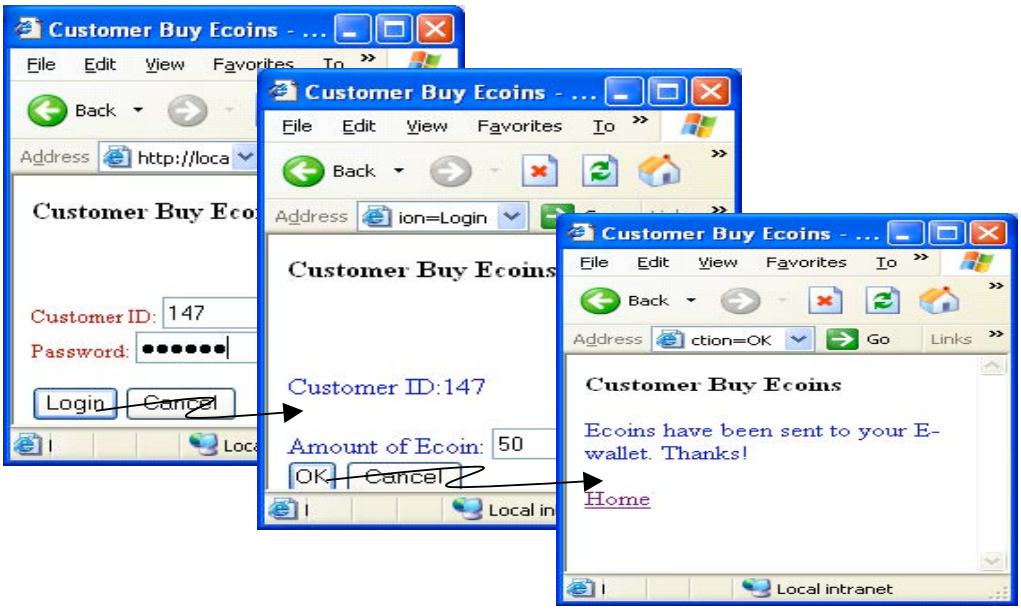


Figure 5-30 Example of HTML customer buy e-coins

Now the customer can check the e-wallet balance from account menu, e.g. there is 50cs in the e-wallet as shown in figure 5-31.

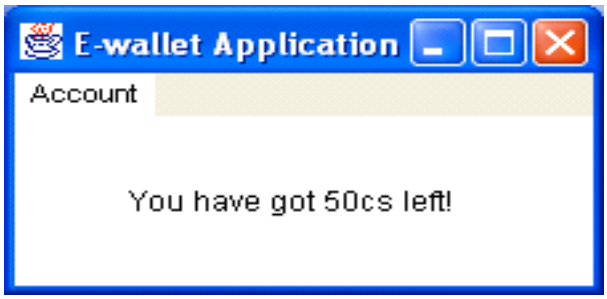


Figure 5-31 Example of checking balance in e-wallet application

The differences between client-side and server-side NetPay systems are that there is no need for customer to download the e-wallet but the customer needs to remember the e-coinID and the customer needs to login and input their e-coinID and password when accessing a newspaper site for server-side NetPay E-wallets.

5.5.2 NetPay Vendor

Vendors provide a set of pages implementing on on-line service provider e.g. in our scenario, an e-newspaper.

5.5.2.1 Server-side NetPay Vendor

Customers are required to login to the newspaper site by entering e-coin ID e.g. 276 and password before buy the articles as shown in Figure 5-32a. After the customer logs into the newspaper site, the site obtains the customer's e-wallet from the broker or another vendor in order to debit and verify the e-coins which are used to pay for the content. The vendor Java Server Pages not only provide searching, browsing and newspaper content for the customer, but also indicate article cost and the amount of e-coins in the e-wallet as shown in Figure 5-32b. After buying an article, the vendor Java Server Pages indicate the amount of e-coins are left in the customer's e-wallet as shown in Figure 5-32c.

The differences to non-NetPay-based newspaper site user interface are logging into the site by inputting e-coin ID (e.g. 276) and password in order to obtain the customer's e-wallet, displaying article cost (e.g. 5cs or 10cs) and the balance of the e-wallet (e.g. You have got 50cs left!).

Leanne as a customer goes to Enewspaper1 site and enters 276 and the password and then she clicks Start Buy button (Figure 5-32a). The system gets her e-wallet from the broker and display "You have got 50cs left!" information. She wants to read the first news which costs 5cs and clicks the title of the news (Figure 5-32b). The system debits e-coin from her e-wallet and displays "You have got 45cs left!" information and the content of the news (Figure 5-32c).

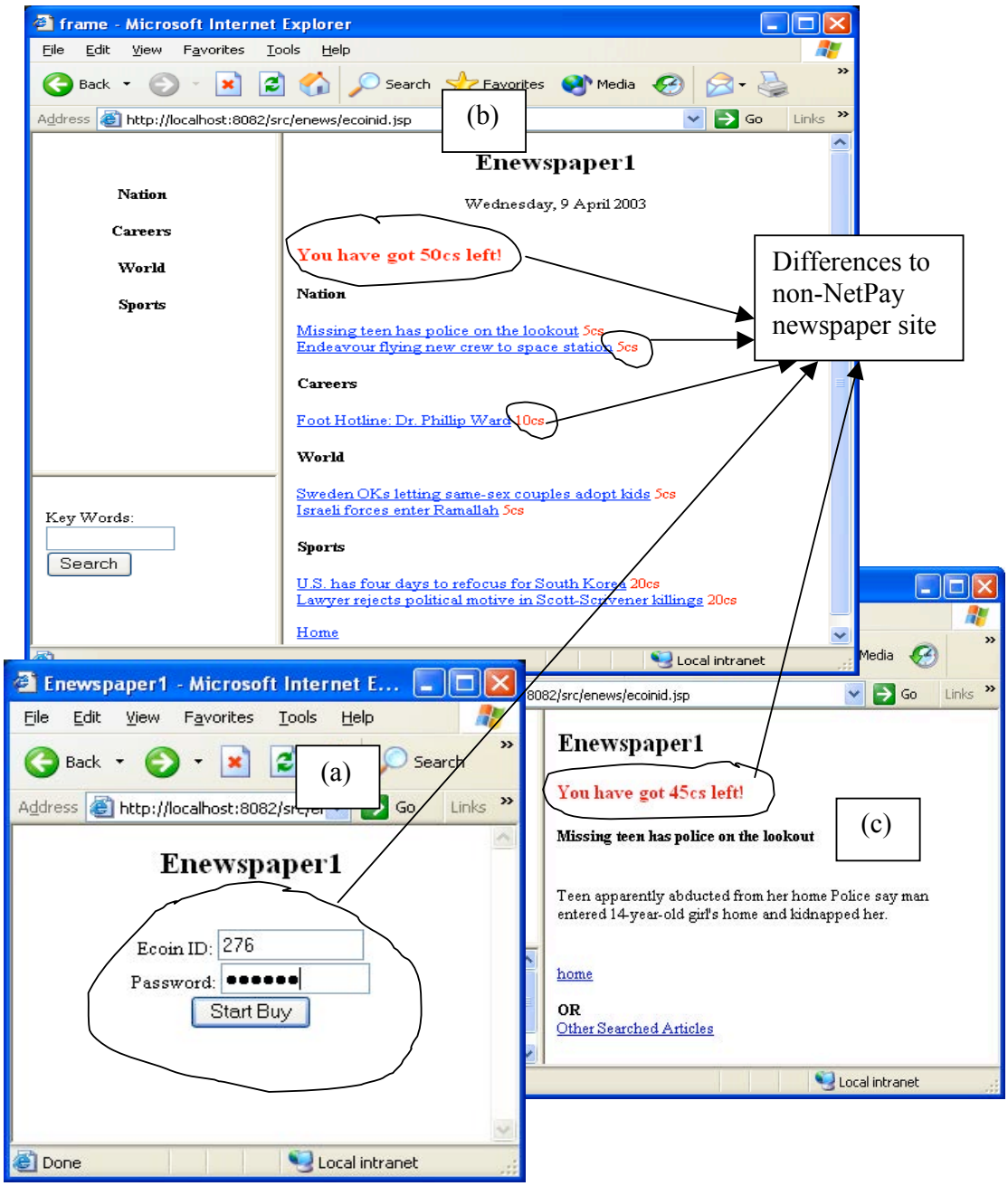


Figure 5-32 Example of HTML customer login and buy article content with Enewspaper1

When the customer moves to another vendor (Enewspaper2), he/she needs to login with newspaer2 site first. The touchstone and current index value of the e-coins are obtained from the previous vendor via the vendor's CORBA interface. After the customer logins to the e-newspaper2 site, the amount of left e-coins is displayed on the screen. The customer can continue to purchase article content with the e-newspaper2 as shown in Figure 5-33.

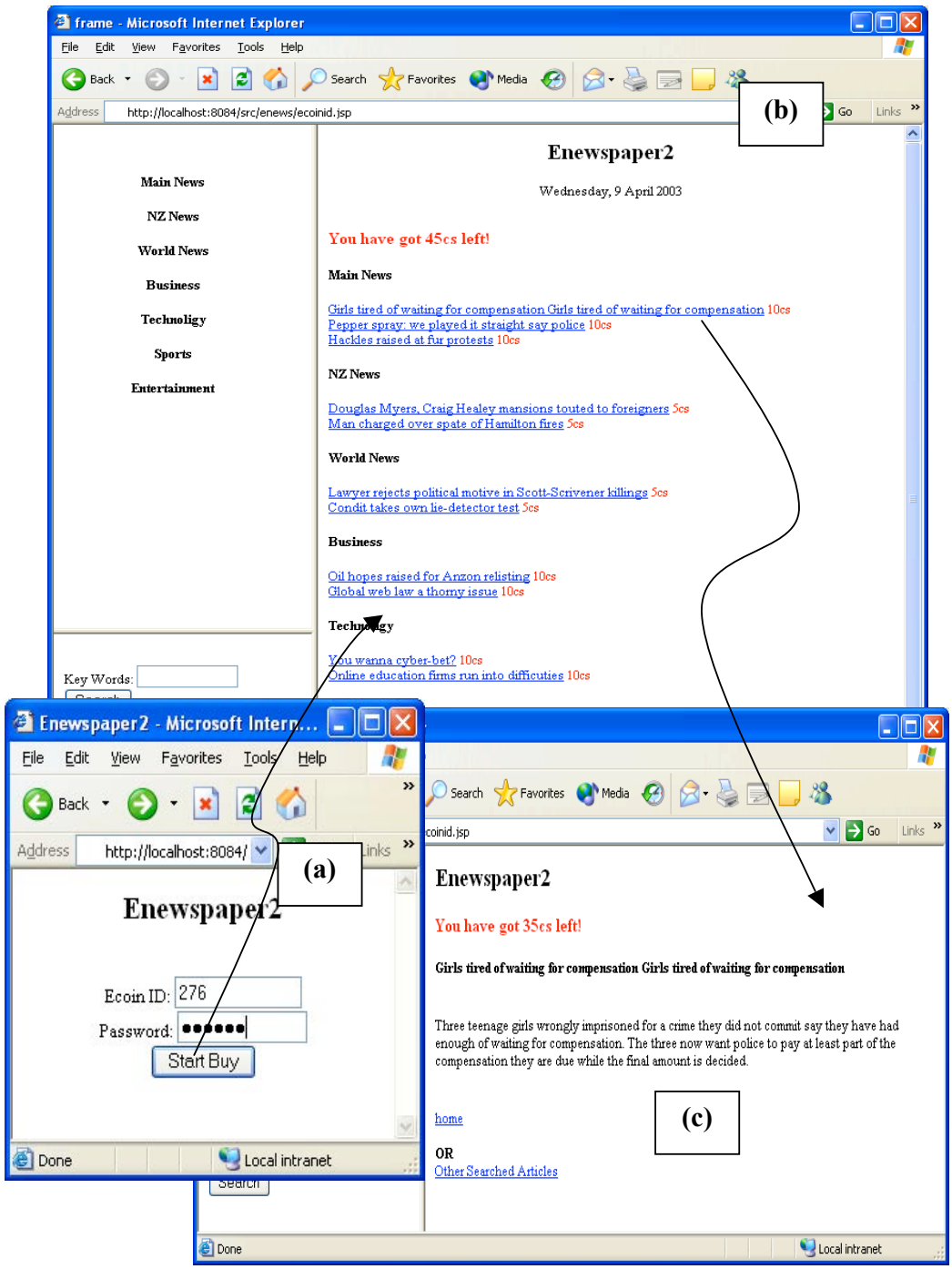


Figure 5-33 Example of HTML customer login and buy content with Enewspaper2

Leanne changes to Enewspaper2 site and login again (Figure 5-33a). Her e-wallet is obtained from Enewspaper1 system and "You have got 45cs left!" information is displayed on the screen (Figure 5-33b). She clicks the title of the first news which costs 10cs in main news section and "You have got 35cs left!" information and the content of the news are displayed on the screen (Figure 5-33c).

5.5.2.2 Client-side NetPay Vendor

In the client-side NetPay vendor, customers need to run e-wallet application first as shown in Figure5-34a and then access Enewspaper1 as shown in Figure5-34b. When the customer clicks the title of the article, Enewspaper1 system requests e-coins with e-wallet application. If the e-coins valid, the content of the article is displayed on the screen as shown in Figure5-34c. The customer can check the balance from e-wallet application window as shown in Figure5-34d.

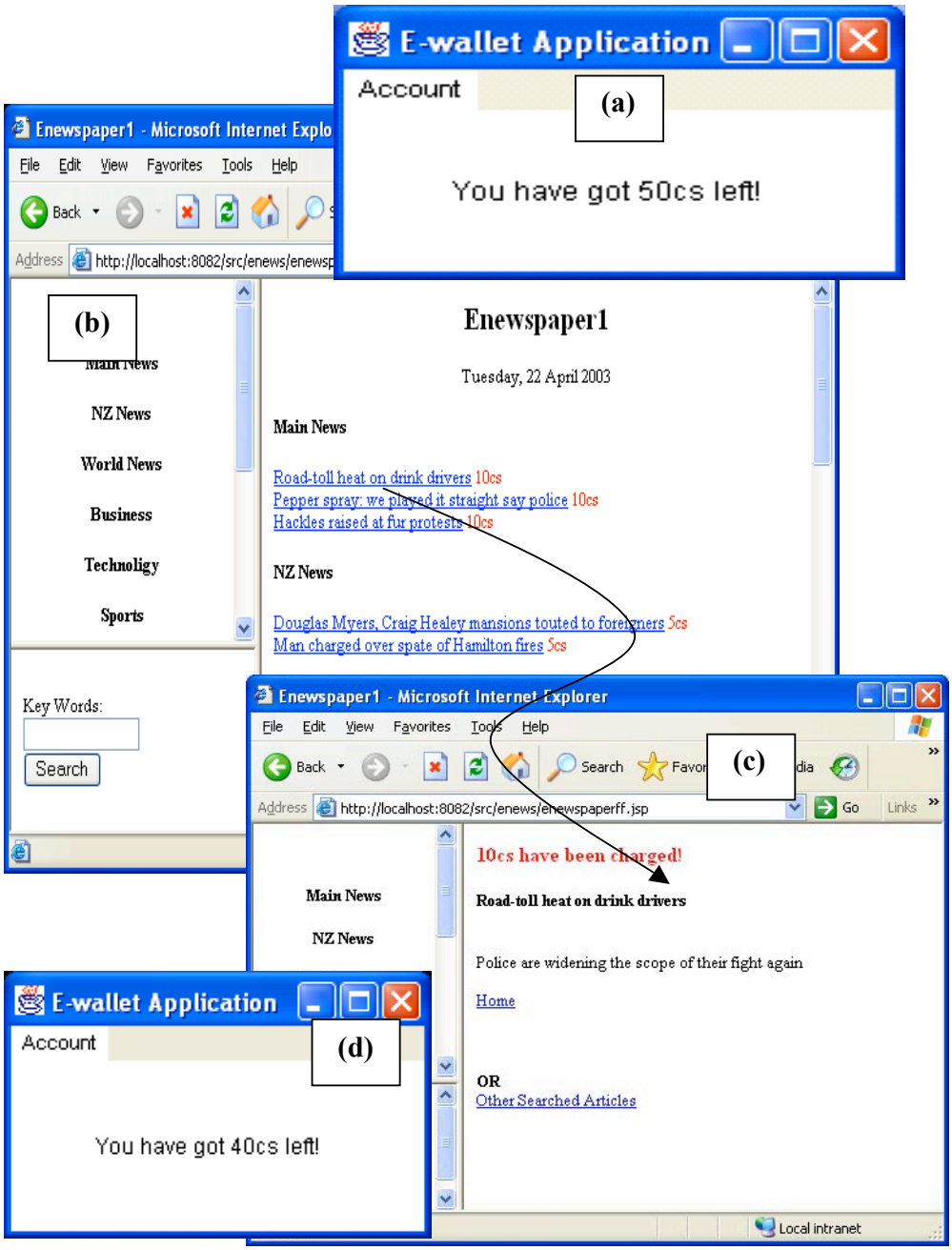


Figure 5-34 HTML customer buy content and e-wallet application example

Leanne as a customer wants to read news article online using client-side NetPay. She runs the e-wallet application on her machine and clicks the account menu to check the balance of her e-wallet (50cs) (Figure 5-34a). She then goes to Enewspaper1 site and browsers the site. She clicks the title of the first main news which costs 10cs (Figure 5-34b). The content of the article and "10cs have been charged!" information are displayed on the screen (Figure 5-34c). She goes to the e-wallet application window again and checks the balance from account menu (Figure 5-34d).

It is easy to change to another vendor site for customers by using client-side NetPay vendor system. When the customer wants to move Enewspaper2 site, he/she only need to enter the site and click-buy articles.

5.5.3 Subscription-based Vendor

In the subscription-based Enewspaper1 system, a customer should subscribe with Enewspaper1 site with subscription fee e.g. $6 per week or $20 per month (Figure 5-35a). Then the customer enters the required information (Figure 5-35b). The system debits money from the customer's credit card (Figure 5-35b).

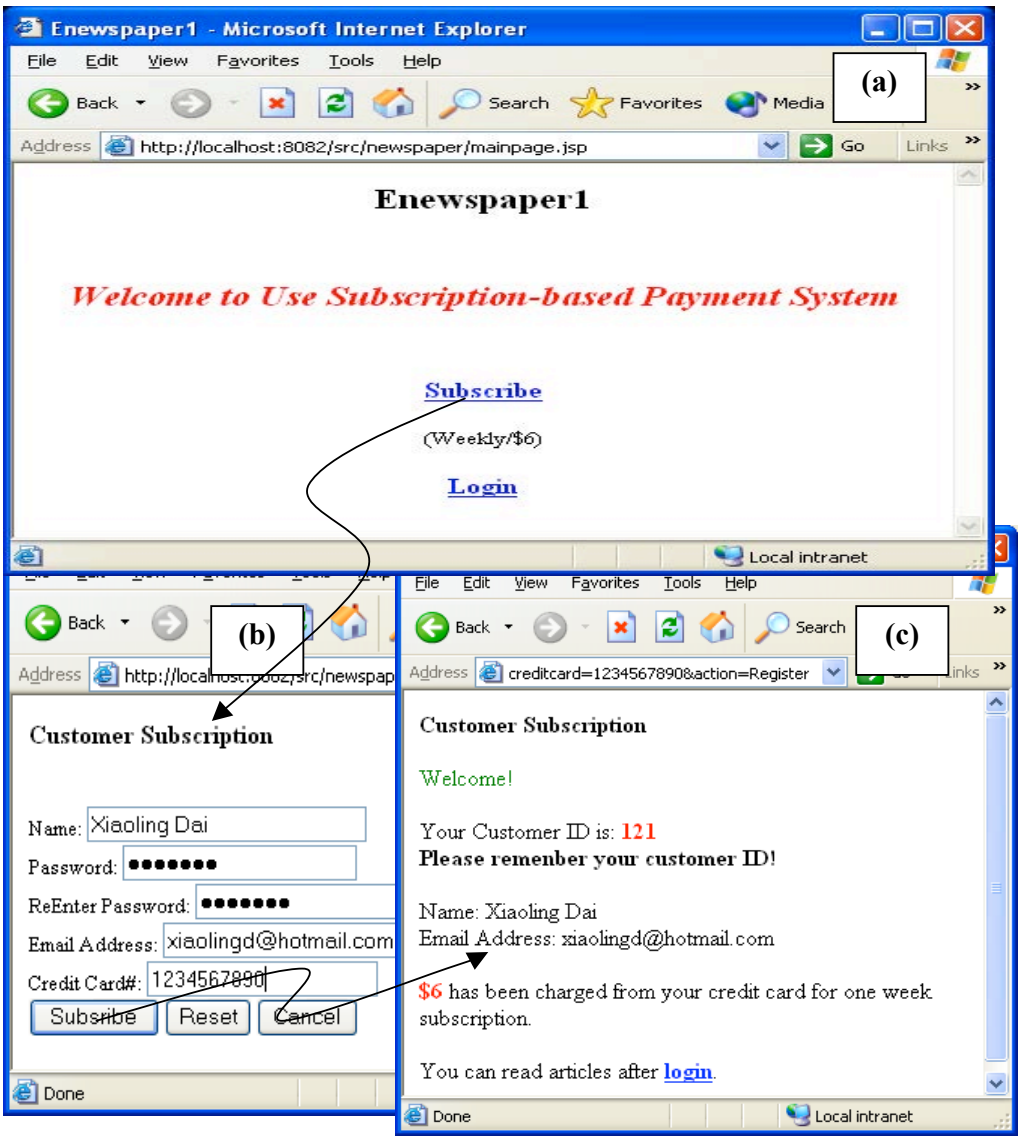


Figure 5-35 Example of HTML customer subscription with Enewspaper1

Sharlene as a customer goes to Enewspaper1 site and enters customer ID e.g. 121 and password and then clicks login button (Figure 5-36a). She browsers the site and clicks the title of the third main news (Figure 5-36b). The content of the news is displayed on the screen (Figure 5-36c).

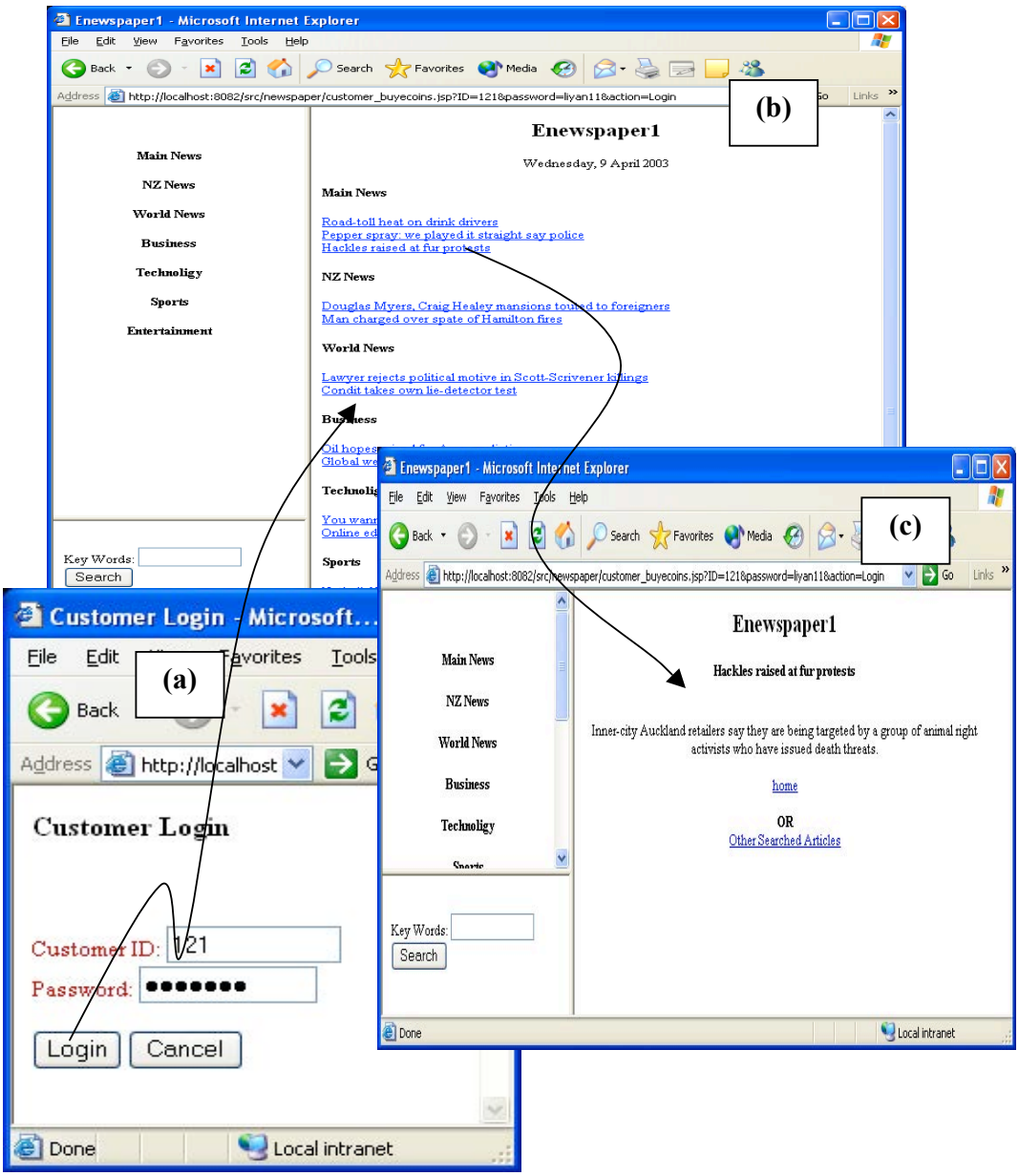


Figure 5-36 Example of HTML customer login and read article content with Enewspaper1

The customer pays another subscription fee to subscribe with Enewspaper2 site when she/he wants to read the article on Enewspaper2 site.

5.6 Implementation and Experience

We have used Java Server Pages (JSPs) to implement NetPay web services, JavaBeans to implement the web service components, CORBA to implement our remote application server objects, and JDBC to implement data management.

It is useful to split the task of generating dynamic content into two parts: the presentation logic which determines how information is presented to the user and business logic which control the relationship between input, process, and output. JSPs are used to handle the presentation logic and JavaBeans are used to handle the business logic. JSPs technology is a simple but powerful way to dynamically generate HTML on the server side [83]. With JSPs, Web pages can be created quickly and easily with dynamically generated content. A JavaBean is Java class that can be easily reused and composed together into application. One of the main goals of the JavaBeans architecture is to provide a platform neutral component architecture.

As we mentioned in section 2.7, there are three popular middleware component technologies which are OMG's CORBA, Java's EJB and Microsoft's .NET. The CORBA standard has been widespread in the area of objected-oriented and distributed systems. It supports independence of the computer architectures and programming languages to be used. It allows users a vendor-independent choice of ORB products and can be used on different kinds of operating system platforms from mainframes to UNIX boxes to Windows machines. .NET is primarily implemented on Windows operating system and is not platform independent. EJBs provide a portable server-side component model and the development of remote object is more complex than CORBA. EJBs and CORBA provide basically similar approaches to realize distributed systems.

5.7 Summary

This chapter has described the thin-client, multi-tier architecture for NetPay system for the requirements specification described in Chapter 4. The static and dynamic designs of three kinds of NetPay systems have been discussed. Each of NetPay systems has its own advantages and disadvantages – server-side NetPay requiring customers to remember e-coinID, the client-side and cookie-based NetPays requiring customers to download and run

e-wallet application. The broker and the “hard-coded” vendor prototypes are implemented for both server-side and client-side NetPay systems. Using a “hard-coded” style the NetPay vendor systems are easy to implement, but there are obvious drawbacks. For example, if a vendor system does exist and we want to add some NetPay functions, the source code of the existed system must be changed to suit the new needs. The next chapter will present a component-based NetPay E-journal system in order to plug-in NetPay components to an existed E-journal system.

Chapter 6

Component-based NetPay

In the previous chapter we described the hard-coding of NetPay support into web-based applications. We now describe a component-based approach for encapsulating micro-payment support for web-based applications - "vendors" of products, services or information to be purchased by micro-payments. In this chapter we use an E-journal example system and then enhance this by adding NetPay components via J2EE Enterprise JavaBean components and Java Server Page proxies. These reusable NetPay components are plugged into an existing journal site to enhance it with micro-payment support with minimal or no code changes. We will describe this new NetPay prototype's architecture and design and illustrate the E-journal interface after plugging in these components.

6.1 Motivation

In Chapter 5 an approach of hard-coding NetPay support into web-based applications was introduced. There are some disadvantages to this approach. The major disadvantages are:

- Difficulty and time consumption to add NetPay support to existing applications. For example, the source code must be changed to suit the new needs when a vendor wants to add NetPay micro-payment support to an existing web application. So the developers need to spend more time to do this and test it.

- Reusability level is lower. The "hard-coded" system from Chapter 5 was developed to solve a special case. The system is only developed for selling articles on on-line newspaper site using NetPay support. Some NetPay functions and interfaces are fitted into E-newspaper system. Therefore, one can't reuse the NetPay objects into another existing web application as they are currently implemented.

To overcome these disadvantages, a component-based NetPay vendor system is considered. One of the characteristics of such a system is that its components can be plugged into an existing web system. Such a micro-payment system would ideally be easily reusable i.e.

- No requirement for extensive redevelopment to integrate it with the web application's architecture;

- No requirement to modify the web application itself providing effective and efficient debiting of customer "E-coins" and redemption of these coins via a broker for "real" money;

- Integrating seamlessly with both the architecture and user interfaces of the web application.

The use of component-based approaches to building and extending enterprise and web-based systems has become popular [2, 4, 7, 31, 19, 75]. Many approaches focus on enterprise business logic extension [4, 31, 90], rather than a combination of user interface and logic extensions [19, 38]. We will try to build a component-based NetPay vendor system based on the "hard-coded" system. The new system to add NetPay to an existing system uses a similar technique used for adding collaborative work components to an existing travel-planning application [38]. The new system should match the needs of existing systems and have NetPay micro-payment support integrated seamlessly with minimum effort with their existing web application architecture. We need to find out the general components of a NetPay vendor system, using plug-and-play in the component-based system to add NetPay components to existing web applications. In the E-journal example, the journal provider would want to charge small amounts on a per-article basis (perhaps varying amounts). The main purposes of the component-based NetPay vendor system are:

- To separate the NetPay EJBs from the particular domain knowledge of the web application, enabling each enterprise bean to be reused in different EJB-based vendor systems via plug-and-play with the existing vendor components;

- To plug the NetPay EJBs into the E-journal's existing application server;

- To annotate the E-journal's JSPs making the appropriate E-coin balance, article cost, credit checks and coin debits to the NetPay EJBs.

6.2 A Component-based NetPay Architecture

We describe the component-based NetPay architecture in this section and aim to develop component-based NetPay vendor services, supporting much more easily and seamlessly reused vendor server-side NetPay functionality. NetPay micro-payment transactions involve three key parties: the Broker Server, the Vendor Server, and the Customer browser.

The architecture is illustrated in Figure 6-1. The Broker server and the Customer browser are the same as those described in Chapter 5.2. The Vendor web sites provide a web server and possibly a separate application server, depending on the web-based system architecture they use. Vendors may use quite different architectures and implementation technologies. In Figure 6-1, Vendor #1 uses a web server with Perl-implemented CGI scripts, C++-implemented application server and relational database. Vendor #2 uses a J2EE-based architecture with J2EE server providing Java Server Pages (web user interface services) and Enterprise Java Beans (application server services), along with a relational database to hold vendor data.

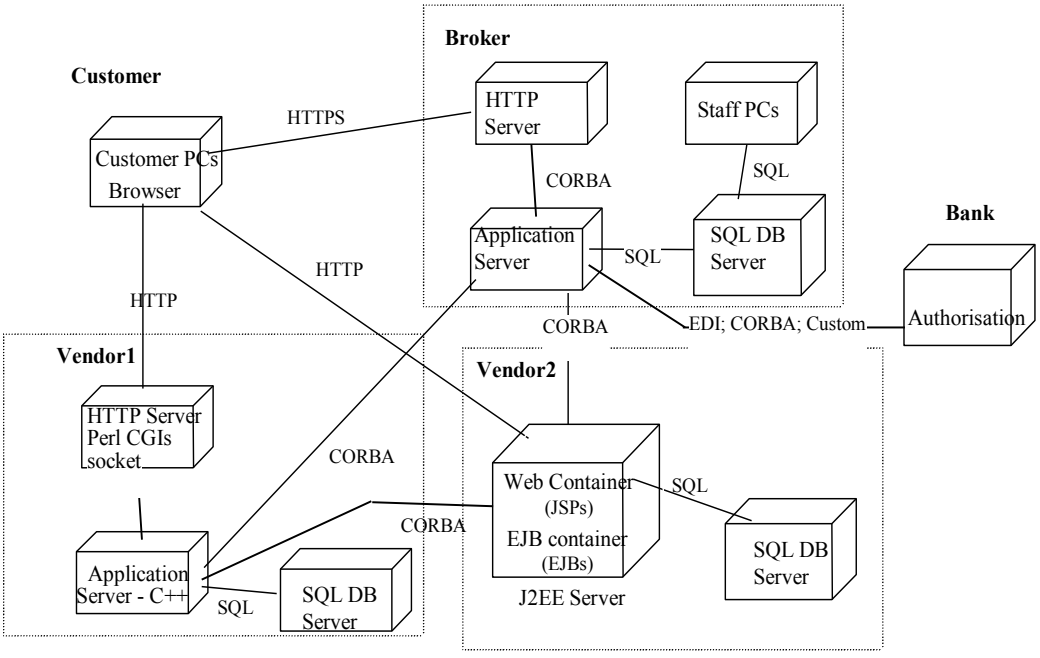


Figure 6-1 Component-based NetPay software architecture

As a platform-independent and language-independent distributed object protocol, CORBA is an ideal choice when integrating systems developed in multiple programming languages

[81]. The vendor #1 C++ and the vendor #2 EJB application servers access the broker application server via CORBA to obtain touchstone information to verify the e-coins being spent and to redeem spent e-coins. They communicate with other vendor application servers which may be a C++ vendor #1 or a J2EE vendor #2 or CORBA vendor to pass on e-coin indexes and touchstones via a CORBA interface.

As described in Chapter 2.10, J2EE platform provides a simplified approach to developing scalable and high-availability Internet/Intranet applications. It extends J2SE with many enterprise-related APIs, the Web and the EJB component model, and runtime containers to host Web and EJB components. The component-based NetPay vendor system is a J2EE-based application that is partitioned into four layers. Each layer has a different responsibility in the overall deployment and is described as follows:

- **Client layer** – Web Browser.

- **Web layer** contains web components dealing with user interface and user interaction. In this layer the system uses JSPs to handle the presentation logic and Servlets to process all requests from URLs. The JSP pages in the system rely on Java-Beans components for interaction with the Enterprise JavaBeans.

- **Business logic layer** contains EJB components that work together to solve business problems. In this project, these components are Enterprise JavaBeans that include session beans and entity beans.

- **Data layer** is used by the business logic layer that stores data in a database. Cloudscape relational database as the storage mechanism for the entity bean will be used in the system.

6.3 Enterprise JavaBeans (EJB)

As mentioned in Section 6.2, EJB components are used to solve business problems in the business logic layer. The EJB component model is designed to enable enterprises to build scalable, secure, multi-platform, business-critical applications as reusable, server-side components. The EJB model defines components which are called enterprise beans that

allow the developer to write business objects using the services provided by the J2EE platform [44, 73]. There are two main kinds of enterprise beans which are Session Beans and Entity Beans. The most recent EJB specification added a third kind which is Message-Driven Beans to help integrate JMS (Java Message Service) message queues with enterprise beans [85, 91].

6.3.1 Session Beans

A session bean is an object that presents a transient conversation with a client. The session bean's methods are invoked when the client needs to access the application that is deployed on the server. Session beans provide a client's view of the application's business logic. They implement business logic, business rules and workflow. For example, a session bean performs debit e-coins, request e-wallet, etc. There are two subtypes of session beans which are stateful and stateless session beans.

- A stateful session bean contains conversational state on business process with its client. For example, a client is doing on-line shopping and adding items in their shopping cart, the contents of the shopping cart are specific to a particular customer session and need not be saved unless the customer is ready to place an order. So that the system should keep track the state of shopping cart. In this situation, shopping cart can be a stateful session bean.

- Stateless session beans are designed to provide server-side business processes and do not maintain any state information for a specific client. For example, a client calls an article session bean to get the articles information. Article bean can be designed as a stateless session bean.

Stateless session beans minimize the resources needed to support a large number of clients, applications using this approach may scale better than those using stateful session beans.

6.3.2 Entity Beans

An entity bean represents a business data such as a customer or a piece of e-wallet information stored in a persistent storage. The bean provides the methods to access and

manipulate on that data. An entity bean allows shared access from multiple clients. If the state of the entity bean is being updated by a transaction at the time of a server crash, the entity bean‘s state is automatically reset to the state of the last committed transaction. There are two subtypes of entity beans which are Container Managed Persistence (CMP) and Bean Managed Persistence (BMP).

- CMP contains all data persistence managed by EJB Container. A bean provider should use some mechanism usually an XML configuration file to define the database table to object instance variable mappings. Then the EJB container handles saving, loading, finding database operations for the bean provider.

- BMP means that a bean provider directly implements persistence in the enterprise bean class. The provider should write the full persistence operations such as saving, loading, and finding data.

The entity bean is used to represent a business entity, not a procedure. For example, the e-wallet enterprise bean would be an entity bean, but the debit e-coins enterprise bean would be a session bean.

6.4 Component-based NetPay Design

The E-Journal example system has a number of customer web browser clients used by customers to access the journal site and read article contents. Another web client is used by staff to manage the redemption of spent E-coins with the NetPay broker server. The vendor J2EE server has a number of web pages e.g. JSPs or Servlets and EJBs providing an implementation of the E-journal web system. We add to this a number of NetPay components: EJBs to provide E-wallet management (tracking spending of E-coins by customers; E-coin exchanges with the client-side E-wallet application or server-side E-wallet management; and touchstone exchanges with the NetPay broker or other vendors). We also provide redemption support for the vendor to communicate with the NetPay broker and redeem customer-spent E-coins for real money [25].

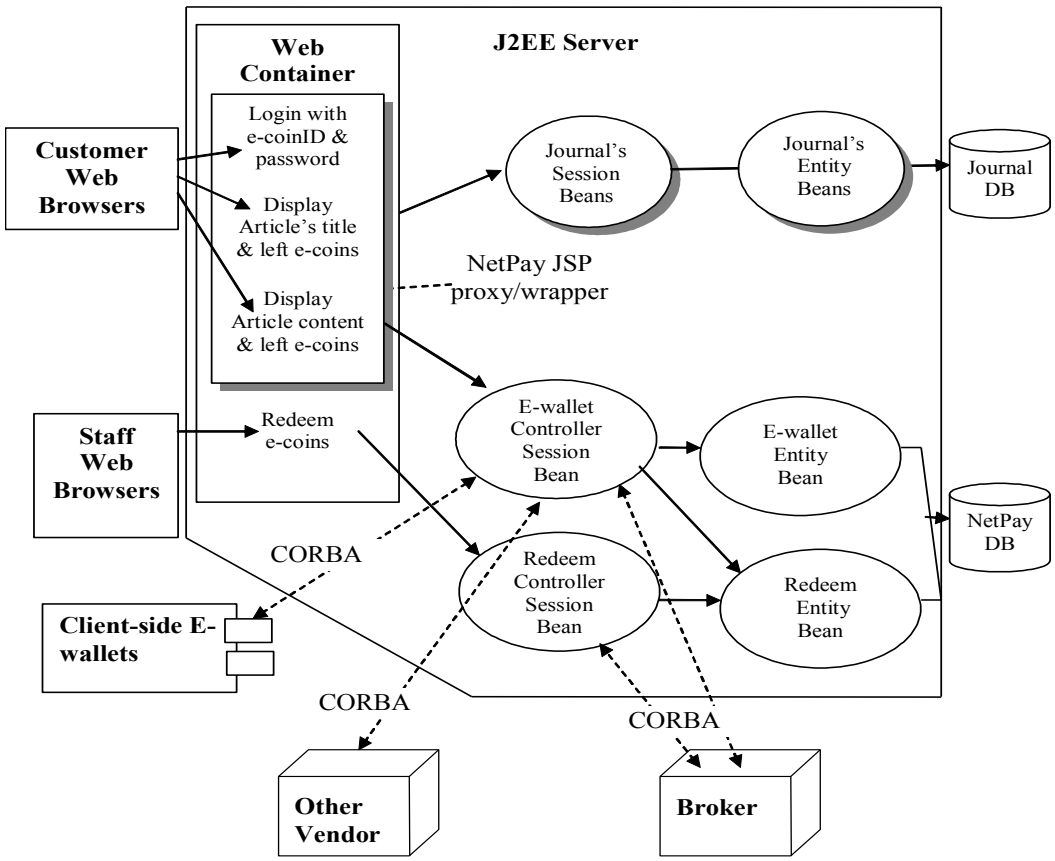


Figure 6-2 E-journal system with NetPay components

Figure 6-2 shows a high-level view of how these various components interact in the E-Journal example system. The end-user clients access only the session beans. Within the enterprise bean tier, the session beans are clients of the entity beans. On the back end of the system, the entity beans access the database tables that store the entity states. The Session beans access the client-side e-wallet application, broker server and other NetPay-implementing vendor servers via CORBA remote object interfaces.

6.4.1 Web Components Design

The one main part of the E-journal system consists of some B2C web components, which were designed using JSP, Servlet, and JavaBean. The web components manage interactions between web clients and application business logic, generate contents dynamically, present data and collect input, and control screen flow. Generally there are two types of architecture which are JavaBean based architecture and Model-View-Controller (MVC) architecture for design web components.

The JavaBean based architecture includes a web browser directly accessing JSP pages. The JSP pages access JavaBeans that represent the application model, and the next view to display is determined by request parameters. This architecture was used in Chapter 5 to design and implement CORBA-based NetPay-enabled E-newspaper prototypes. It is easy to implement, debug and quickly achieve some specified goals. However the architecture is not very flexible, scalable and customisable.

The MVC architecture was originally developed to map the traditional input, processing, or output tasks to the user interaction model. Each part of an MVC design has an independent role. The "model" is the business logic and data representation implemented by the application. The "view" is the component that provides data presentation and user inputs (JSP). Finally the controller dispatches requests and controls screen flows. Figure 6-3 shows the architecture of using JSP pages, Enterprise Beans, and controller components. The MVC architecture separates design concerns, decreases code duplication, centralizes control, and makes the application more easily modifiable. It provides more flexibility, scalability, and manageability.

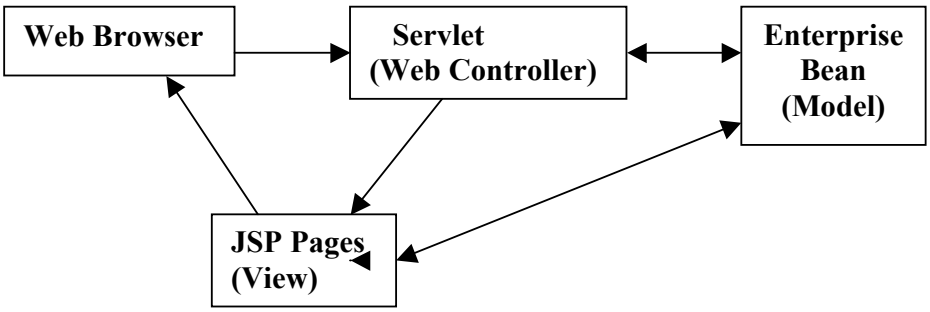


Figure 6-3 Model-View-Controller architecture

The MVC architecture is used in the E-Journal example system. In the application a controller was introduced between the browser and the JSP pages content being delivered. The controller consists of two components:

- Template.jsp determines the structure of each screen in order to maintain a common look across all the JSP pages and defines subcomponents used by each screen. All screens have the same banner, but different title and body content.

- Dispatcher.java, a servlet, processes requests and forwards to template.jsp.

6.4.1.1 E-Journal Main Page Class Diagram

Figure 6-4 shows the example of E-journal site web page template. The body contains two parts which are category and search table and contents table. The contents tables provide journals, publish years, issues, articles and article content information.

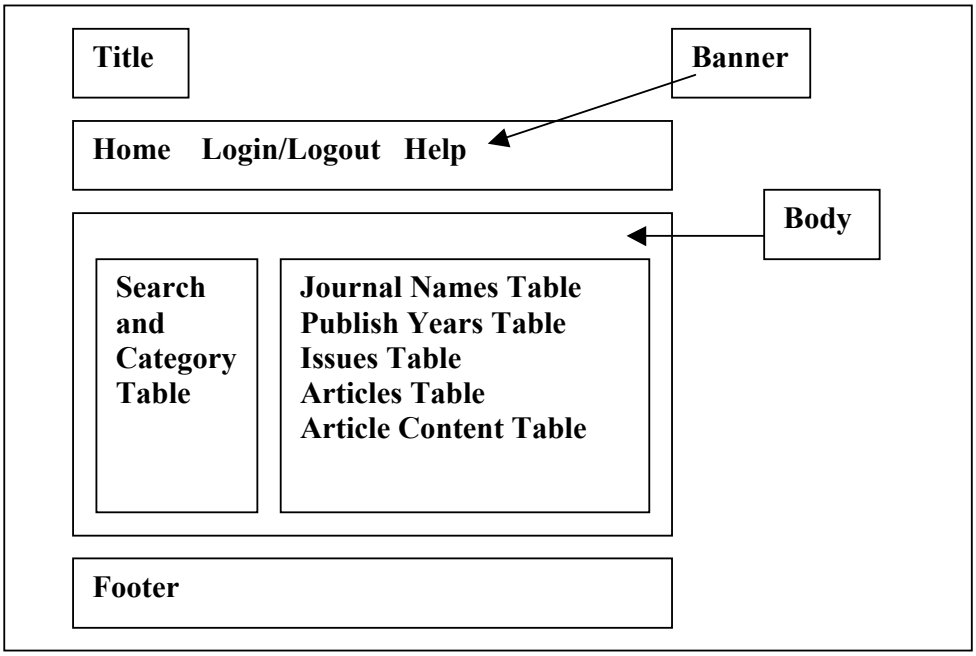


Figure 6-4 E-journal site web page template

Figure 6-5 shows journal web component main page class diagram. Main page is used as a home page in on-line journal scenario. A main page contains title and main template. Main template is composed by banner, body and footer sub-components.

- Banner contains some links such as home, terms, login (if required) and help. Each link will point to a page.
- Body can be sub-divided into search component, journals table, published years table, issues table, articles table and article content table. Customers can search articles by entering an author or a key word or any other related words in the text field to perform the search function. Journals table displays journals' name. Published years table displays the year published for a journal. Issues table displays the issues in a year for a journal such as published month and year, volume number,

and issue number. Articles table displays the details of articles published in a journal, such as article title, authors, page(s), price and abstract. Article content table displays an article contents such as title, authors, etc. The body component may be replaced by other contents such as e-wallet balance information or login success information.

- Footer component contains relevant links and other information such as copyright, web master email link, etc.

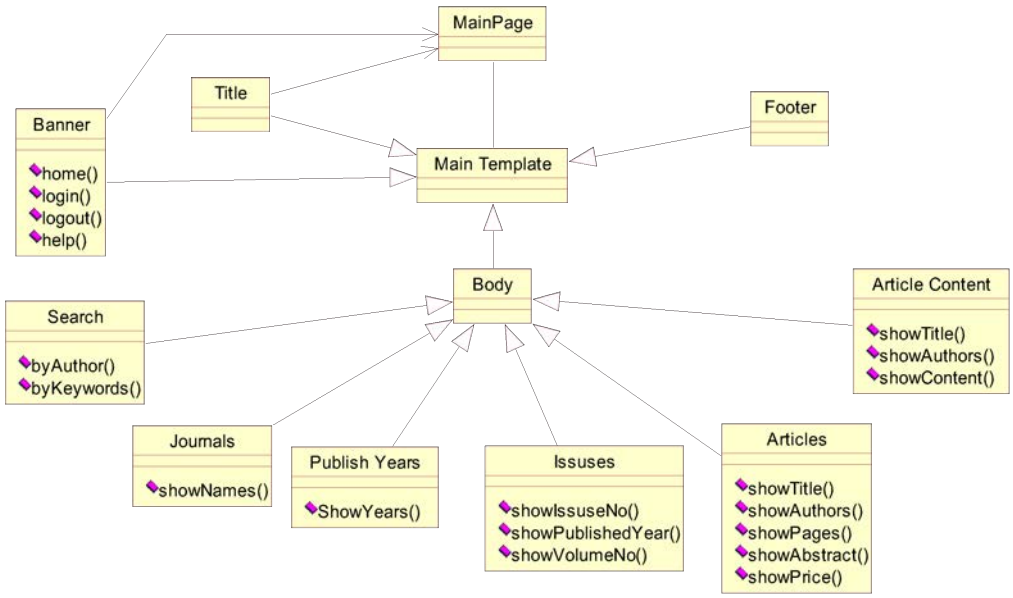


Figure 6-5 Journal main page class diagram

For simplicity, in our E-journal example system, the body of the main template can only have two parts which are articles table and article content table as shown in Figure 6-6.

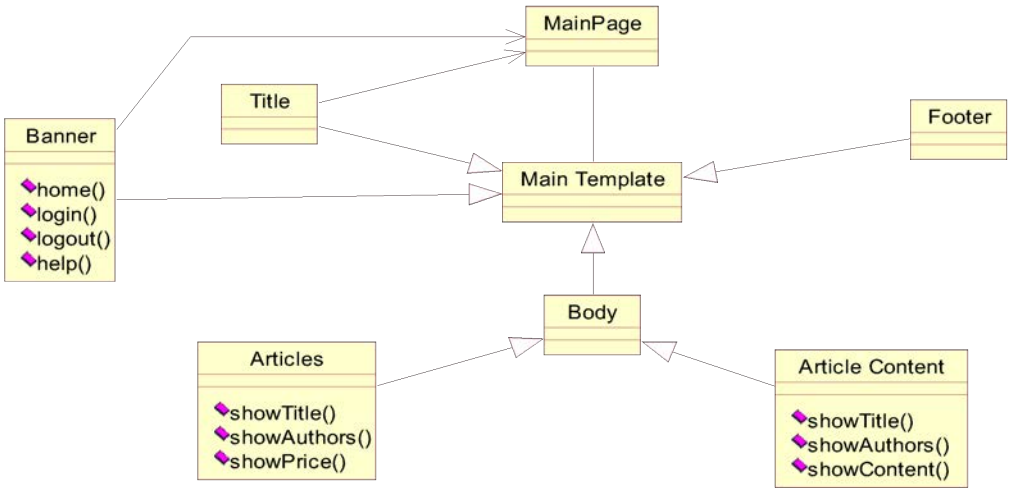


Figure 6-6 E-journal example system main page class diagram

6.4.1.2 Web Component Interaction

Figure 6-7 shows the flow of a request through E-Journal example Web components.

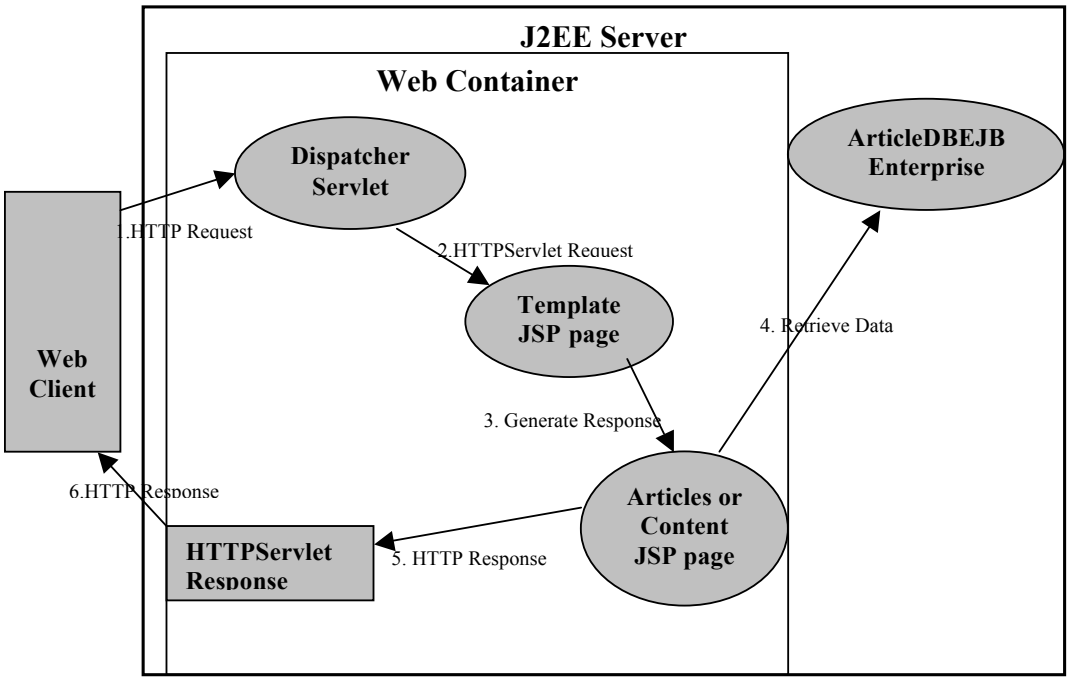


Figure 6-7 E-Journal Web component interaction

A HTTP request (1) is delivered to the dispatcher component which processes and then forwards the HTTPServlet request (2) to the template.jsp. The template.jsp generates the response (3) by including the responses from the body subcomponent (Articles or Content JSP page) which retrieves data (4) from enterprise bean and transmits it (5 and 6) to the client for presentation.

6.4.2 NetPay Integration with E-journal Web Page

In order to add our NetPay micro-payment facility to the E-journal, or to other 3rd party J2EE-based applications, we need to be able to add our EJBs to their J2EE server and to detect when pages are being accessed by customers that need to be paid for. We also need to ensure that if the customer attempting to access does not have enough e-coins they are directed to the NetPay broker site to buy some more. If the customer wants a server-side e-wallet managed by the vendor verses running a client-side e-wallet application, we need to

have the vendor to obtain the customer NetPay user-name/password and obtain the e-wallet from the NetPay broker or the previously-visited NetPay-enabled vendor. In addition the customer usually wants an idea about the cost of an article or other information/service before purchase, and accesses to his/her available credit in e-coins.

There are three main ways to integrate the NetPay user interface facilities: (1) modify the existing system web pages to incorporate NetPay information; (2) generate web pages that display the existing system pages in frames and make appropriate interactions with NetPay EJB components; and (3) generate proxy web pages that interact with NetPay session beans and redirect access to the original web pages. These approaches are illustrated in Figure 6-8.

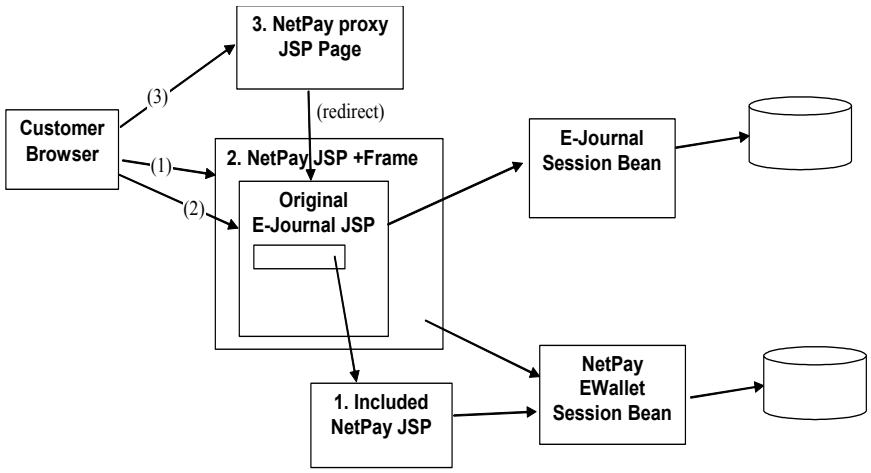


Figure 6-8 Ways of integrating NetPay functionality with E-journal web pages

1. Modifying the existing system web pages

The articles.jsp is modified to retrieve price data from ArticlePriceEJB enterprise bean for displaying article price information or retrieve e-wallet data (for server-side NetPay) from e-wallet enterprise bean for displaying e-wallet information. Figure 6-9 depicts the interaction between these Web components.

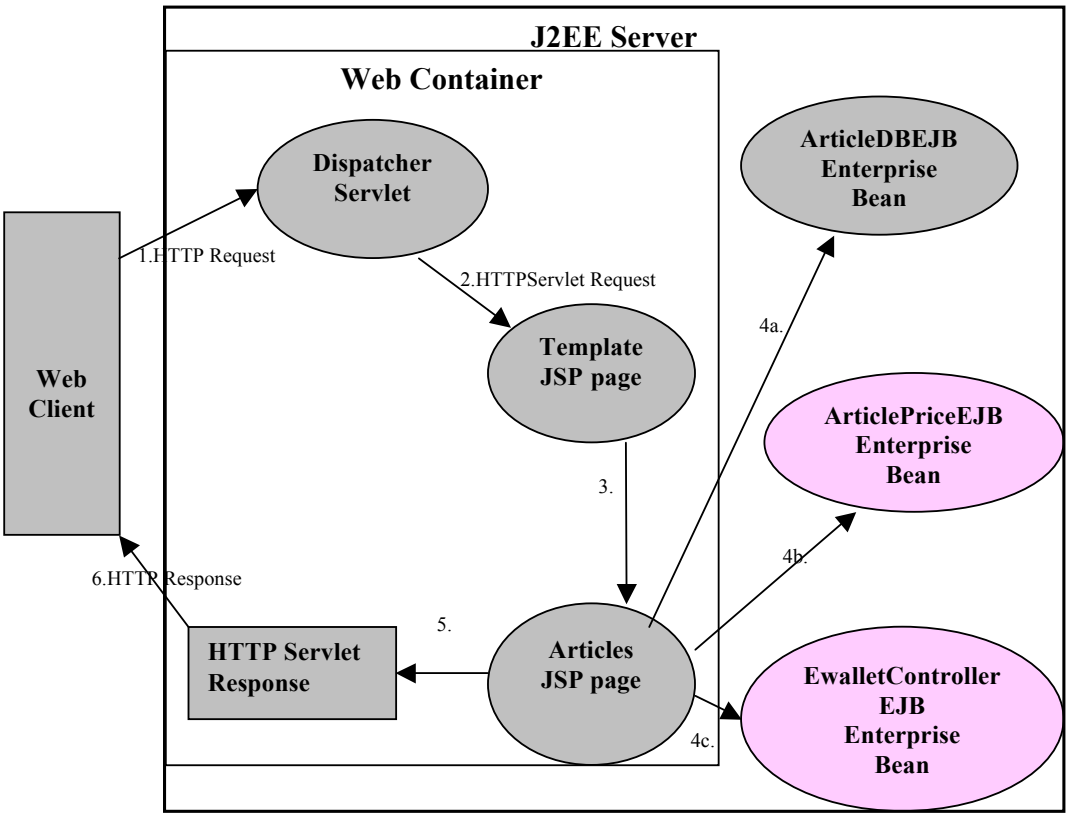


Figure 6-9 Web component interaction after modified article.jsp

A HTTP request (1) is delivered to the dispatcher component which processes and then forwards the HTTPServlet request (2) to the template.jsp. The template.jsp generates the response (3) by including the responses from Articles JSP page. Articles JSP page retrieves article contents from the article enterprise bean (4a) and article price from the article price enterprise bean (4b), and e-wallet data from e-wallet enterprise bean (4c). Articles JSP page transmits responses (5 and 6) to the client for presentation.

The content.jsp is modified to make payment from e-wallet enterprise bean in order to debit e-coins paying for article content and Login.jsp is implemented (for server-side NetPay) to retrieve e-wallet data from e-wallet enterprise bean. Figure 6-10 depicts the interaction between these components.

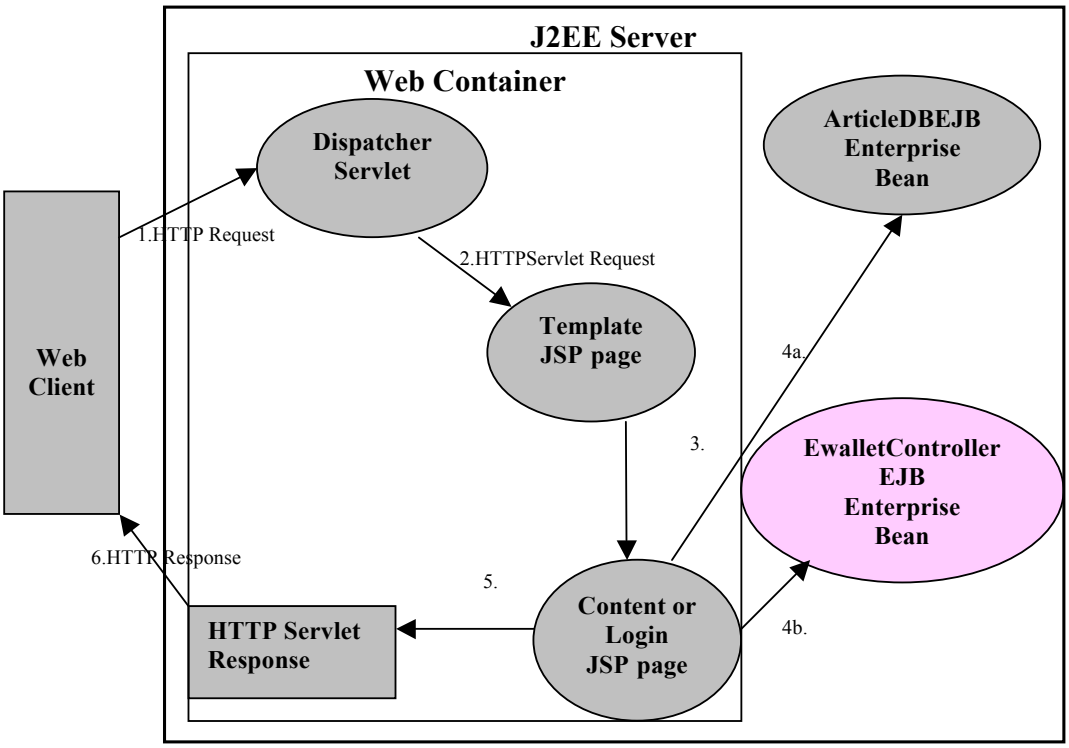


Figure 6-10 Web component interaction after modified and implemented JSP pages

A HTTP request (1) is delivered to the dispatcher component which processes and then forwards the HTTPServlet request (2) to the template.jsp. The template.jsp generates the response (3) by including the responses from Content or Login JSP page. Content JSP page retrieves article contents from the article enterprise bean (4a) and Login JSP page retrieves e-coin ID and password from the e-wallet enterprise bean (4b). Articles or Content JSP page transmits responses (5 and 6) to the client for presentation.

2. Generating NetPay JSP pages

A HTTP request (1) is delivered to NetPay JSP pages which are generated to display article.jsp in frames and retrieve article price data from the article price enterprise bean (3) and e-wallet data from the e-wallet enterprise bean (4). The article.jsp retrieves articles' title and author data from the article enterprise bean (2). NetPay JSP pages display the articles and e-wallet information to the client (5). Figure 6-11 depicts the interactions between these components.

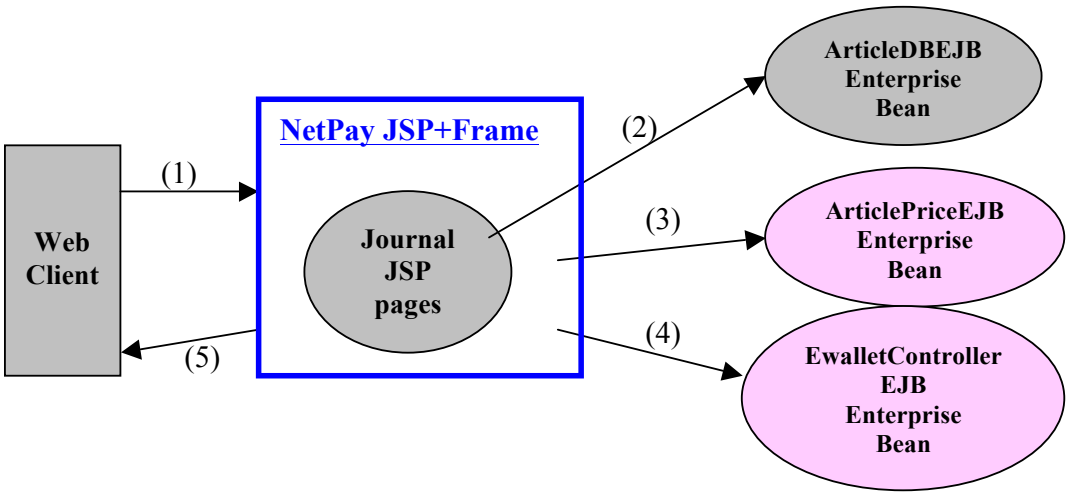


Figure 6-11 Generating NetPay JSP pages

NetPay JSP pages also display content.jsp in frames and interact with the e-wallet enterprise bean in order to debit e-coins.

3. Generating NetPay proxy JSP pages

A HTTP request (1) is delivered to NetPay proxy JSP pages which are generated to obtain article price data from the article price enterprise bean (2) and e-wallet data from the e-wallet enterprise bean (3) for displaying costs of the articles and e-wallet information. NetPay proxy JSP pages then redirect to article.jsp accessing the journal home page (4). When a customer wants to read an article content, NetPay proxy JSP pages interact with the e-wallet enterprise bean to debit e-coins from customer's e-wallet (3) and then redirect to content.jsp which retrieve article content from article enterprise bean (5). Finally NetPay proxy JSP pages display the article content to the client (6). Figure 6-12 illustrates the interaction between these components.

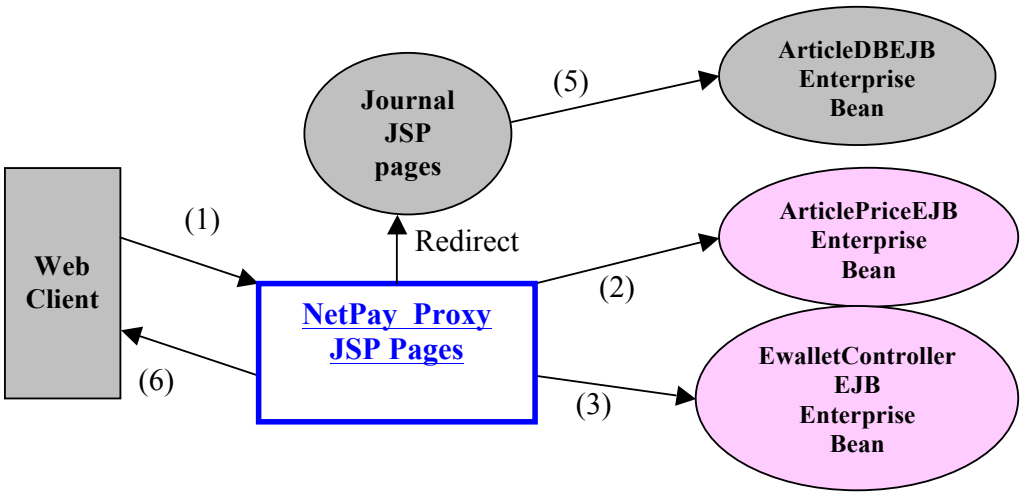


Figure 6-12 Generating NetPay proxy JSP pages

Each of the above three approaches has advantages and disadvantages:

- The first requires updates to the existing system web page implementations. For example, in the journal example system, article.jsp was modified to interact with the price and the e-wallet enterprise beans for displaying the costs of articles and e-wallet balance. content.jsp was modified to debit e-coin from the e-wallet by interacting with the e-wallet enterprise bean before displaying an article content. This can be done easily. However this requires some changes to the code of the existing system. The first JSP interception approach is used to develop the NetPay vendor prototype.

- The latter two requires renaming of these pages so the generated pages are passed to the control at appropriate times. There is no code impact to the existing system's infrastructure, but its design and implementation are more complex.

6.4.3 Enterprise Beans Design

All enterprise beans that permit remote access must have a home and a remote interface. To meet the needs of a specific application, an enterprise bean may also need some helper classes. All enterprise beans also require an enterprise bean class.

Remote Interface extends from EJBobject (RMI distributable object). Remote interface defines the set of business methods available to clients. The business methods are implemented in an enterprise bean class.

Home Interface extends from EJBHome. Home interface defines the methods that allow a client to create, find, or remove an enterprise bean instance. findByPrimaryKey() method uses primary key (such as: ID) to find a unique object instance, that return the remote interface to client who may perform business methods and data persistence.

Enterprise Bean Class provides the actual implementation of the business methods for an enterprise bean. A business method defined in the enterprise bean class is called by the EJB container when the client calls the corresponding method listed in the remote interface.

6.4.3.1 E-Journal Enterprise Bean Design

In our E-journal example system, the enterprise bean is only responsible to select article data from a journal database. Stateless session beans can offer better scalability for applications that require large numbers of clients. We choose stateless session bean architecture in which the session bean has code of business logic and database access. One session bean, ArticleDBEJB, represents an interface for the journal article database and is used to select article records as shown in Figure 6-13.

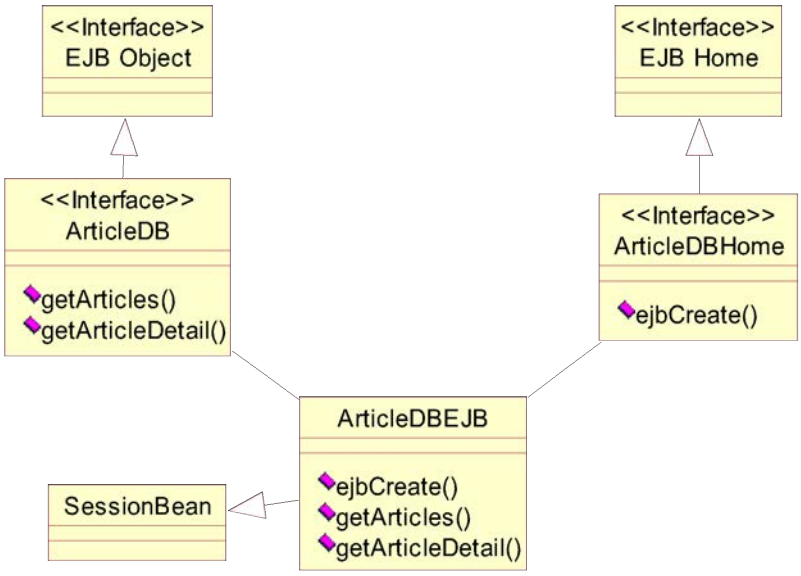


Figure 6-13 Article session bean class diagram

ArticleDBEJB is a session bean class and is used to get articles and the details of an article. It extends from the SessionBean and implements business methods defined in the remote interface. The ArticleDBEJB bean class has several methods that obtain articles and content of an article from journal database:

- *Get articles method* obtains articles' title and authors from the journal database.
- *Get Article detail method* selects an article title, authors and content by using article_id.

6.4.3.2 Article Pricing Enterprise Bean Design

As we mentioned in Section 6.4.3.1, the article price enterprise bean is also used to select article price data from the NetPay database. The stateless session bean architecture is used to represent an interface for the article price database table and to select article price records as shown in Figure 6-14.

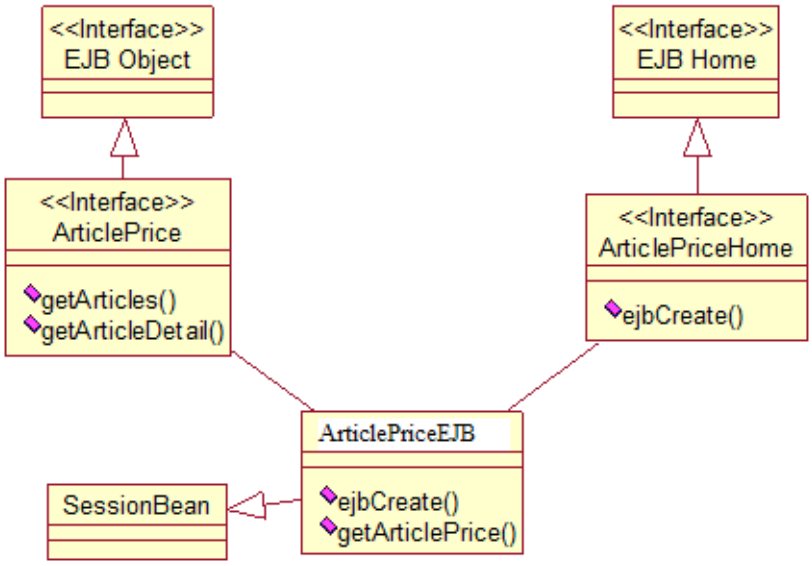


Figure 6-14 Article price session bean class diagram

ArticlePriceEJB is a session bean class and is used to obtain price of articles. It extends from the SessionBean and implements business methods defined in the remote interface. The ArticlePriceEJB bean class has one method to obtain the price of an article from NetPay database ArticlePrice table:

- *Get Article price method* selects an article price by using article_id.

6.4.3.3 NetPay Enterprise Beans Design

The session bean and entity bean architecture is used in NetPay vendor system. The session beans are used to implement business logic. The system uses the entity beans as a java object representation of the underlying relational database. There are four NetPay enterprise beans that have been added to the E-journal system: e-wallet controller session bean, redeem controller session bean, e-wallet entity bean, and redeem entity bean. To construct NetPay vendor enterprise beans, the following classes are needed:

Remote interfaces: EwalletController, Ewallet, RedeemController, Redeem

Home interfaces: EwalletControllerHome, EwalletHome, RedeemControllerHome, RedeemHome

Session bean classes: EwalletControllerEJB, RedeemController

Entity bean classes: EwalletEJB, RedeemEJB

The next two subsections explain these classes in detail and describe interactions between them.

1) E-wallet Enterprise Beans

E-wallet enterprise beans OOD class diagram is illustrated in Figure 6-15.

EwalletControllerEJB is a session bean class and is used to handle payment transaction. It extends from the SessionBean and implements business methods defined in the remote interface. The EwalletControllerEJB bean class has several methods that debit e-coins from the customer's e-wallet to pay for article content:

- *E-wallet (or TandI for client-side NetPay) request method* communicates with the broker or a vendor to request e-wallet.

- *Make payment method* gets e-coins from the e-wallet database (or the e-wallet application for client-side NetPay) and verifies e-coins.

- *Verify e-coins method* verifies the e-coins by using touchstone and index.

- *Update e-wallet method* updates the e-wallet (TandI for client-side NetPay) with new index, paywords, and amount and accesses a RedeemEJB entity bean to insert the payments to Redeem database.

- *Transfer e-wallet (or TandI for client-side NetPay) method* sends the e-wallet to a vendor who requests it and then deletes the e-wallet (or TandI).

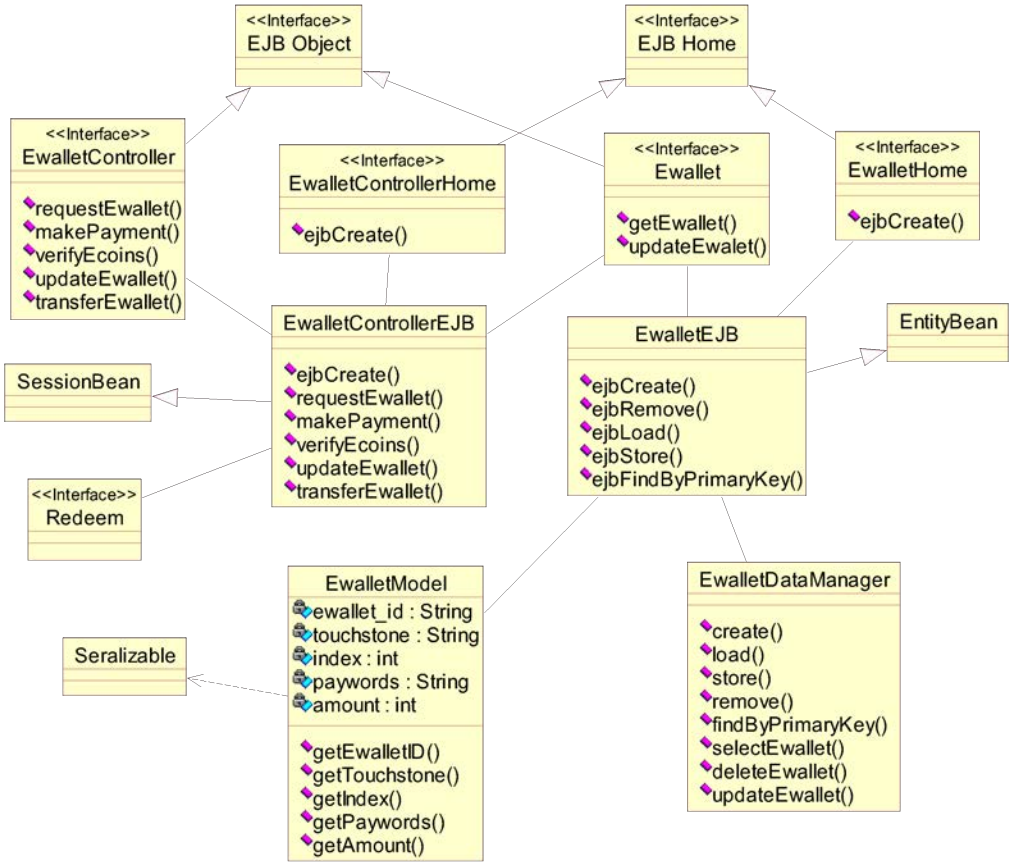


Figure 6-15 E-wallet enterprise beans (server-side) class diagram

EwalletEJB is an Entity bean and is used to represent e-wallet data view. It extends from EntityBean and implements business methods defined in the remote interface. It provides a component-based interface to Ewallet database table. For each column in a table, the corresponding entity bean has an instance variable. Because it uses bean-managed persistence, the entity beans contain the SQL statements that access the tables. For example, the Create method of the EwalletEJB entity bean calls the SQL INSERT command. The actual database access performs in helper class EwalletDataManager.

The NetPay database tables that are used by our entity beans may be added to the existing E-journal database if this is possible, or may be stored in a separate database of their own if required.

Helper classes:

EwalletModel is a data model helper class. E-wallet detail information is coded in this class. The attributes include ewallet_id, touchstone, index, paywords, and amount. Inspecting methods such as get() functions are provided.

EwalletDataManager is a data manager helper class that accesses the database. Database SQL issues such as insert, delete, update, select and business methods are implemented here. This class isolates the EJBs from particular database technology and SQL formats used.

Buy Content Using Server-side NetPay Sequence Diagram

Figure 6-16 shows how a customer buys content with our NetPay-enabled E-Journal application. For example, after a customer finds a desired article, he/she clicks the title of the article. The web browser requests the article content from the appropriate JSP, and this JSP or its generated proxy requests payment for the content of the article from the NetPay E-wallet session bean. The e-wallet session bean contacts the e-wallet entity bean to debit the customer's e-coins to pay for the article. If there is no existing e-wallet, the e-wallet session bean contacts either broker or previous vendor to get them to verify the e-coins. If insufficient coins are available, the customer is directed to the broker site to buy more. Otherwise, the journal article content is displayed to the customer.

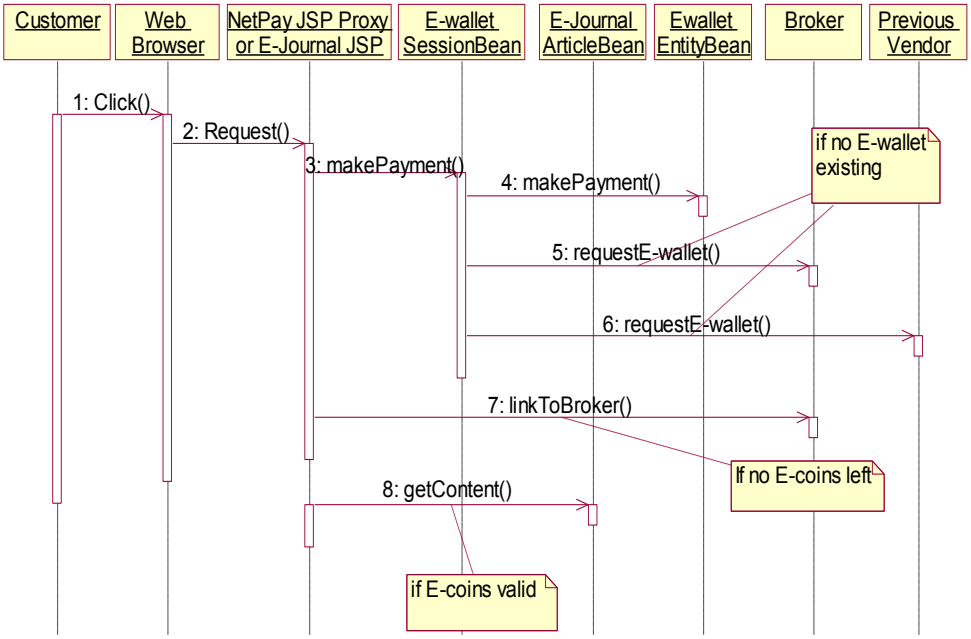


Figure 6-16 Click-buy article sequence diagram with server-side system

Buy Content Using Client-side NetPay Sequence Diagram

A sequence diagram in Figure 6-17 shows how interactions occur between various components in our NetPay-enabled E-Journal application. For example, when buying an article the customer selects the article for reading e.g. clicking on the URL in a returned article search page or in a journal content page. The web browser requests the article content from the appropriate JSP, and this JSP or its generated proxy requests payment for the content of the article from the NetPay E-wallet session bean. The e-wallet session bean communicates with the client-side E-wallet application to debit the customer's e-coins to pay for the article. If there is no touchstone and index existing, the e-wallet session bean contacts either broker or previous vendor to get them to verify the e-coins. If insufficient coins are available, the customer is directed to the broker site to buy more. Otherwise, the journal article content is displayed to the customer.

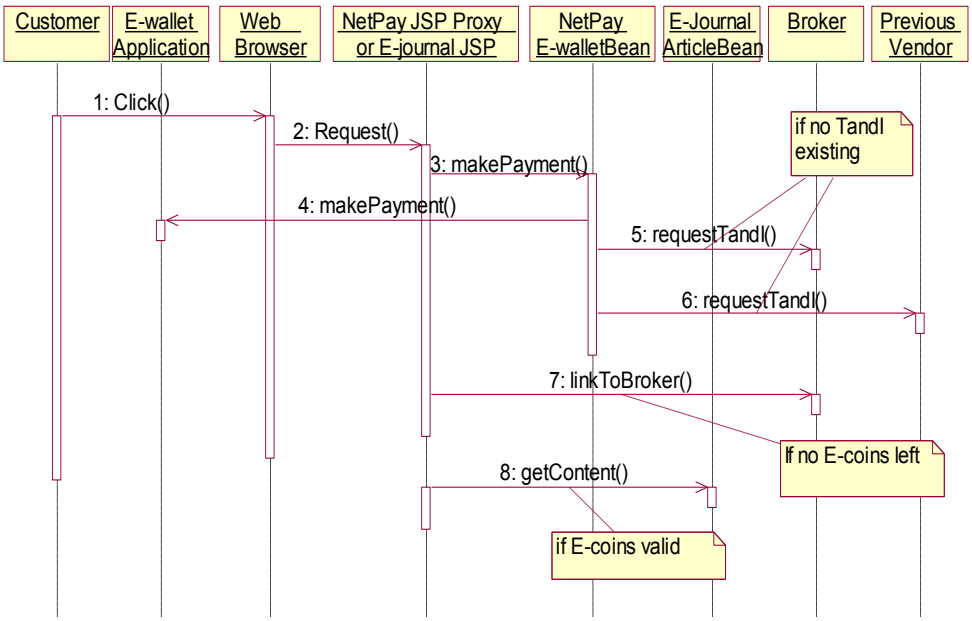


Figure 6-17 Click-buy article sequence diagram with client-side system

2) Redeem Enterprise Beans

E-wallet enterprise beans OOD class diagram is illustrated in Figure 6-18.

RedeemCotrollerEJB is a session bean class and represents a client's view of the redeem business logic. It extends from the SessionBean and implements business methods defined in the remote interface. The RedeemControllerEJB bean class has several methods that process spent e-coins and send them to a broker:

- *Select payments method* selects payments by e-coinID.

- *Send redeem method* communicates with the broker to send the selected redeem data.

RedeemEJB is an entity bean class and represents Redeem database view. It used to record and maintain redeem records and extends from EntityBean and implement business methods defined in the remove interface. The actual database access is performed in the helper class RedeemDataManager.

Helper classes:

RedeeModel is a data model helper class. Redeem detail information is coded in this class. The attributes include redeem_id, ewallet_id index, price, and paywords. Inspecting methods such as get() function are provided.

RedeemDataManager is a data manager helper class that accesses to database. Database SQL issues such as insert, delete, update, select and business methods are implemented here.

Figure 6-18 Redeem enterprise beans class diagram

Redeem Spending Sequence Diagram

The Figure 6-19 shows how a vendor redeem spent e-coins with broker using client-side NetPay. When redeeming spent e-coins the vendor clicks an e-coinID item button. The web browser sends the requests to the redeem JSP which requests redeem from the NetPay E-

wallet session bean. E-wallet session bean selects the payments for the ecoinID and sends all involved information to BAS. The balance for the ecoinID is displayed to the vendor.

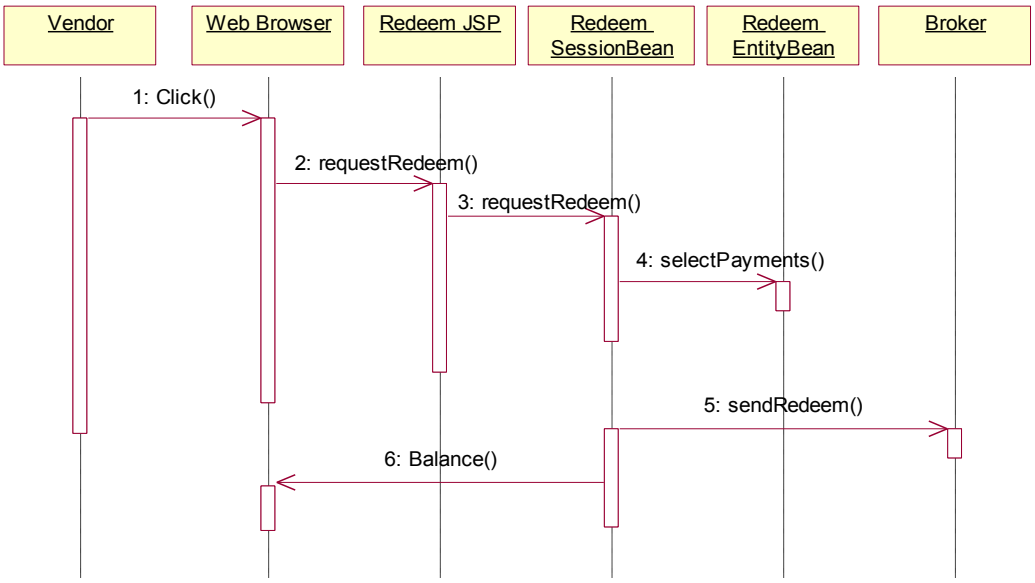


Figure 6-19 Redeem sequence diagram with client-side system

6.5 Component-based NetPay Vendor Implementation

In the previous chapter we implemented a CORBA-based vendor architecture [23]. However, these NetPay components are not optimally reusable and substantial modifications may need to be made to an existing web-based application to incorporate them. We implemented these new NetPay J2EE EJBs and JSP includes allowing for much easier plug-in of NetPay micro-payment facilities into existing J2EE applications.

6.5.1 Packaging J2EE NetPay Vendor

J2EE platform offers five types of components which are enterprise beans, servlets, JSP pages, applets, application clients, and connectors. Developers can assemble J2EE applications from these components involving a two-step process: assembling components into modules which are EJB, Web, Application client and Connector modules, packaging these modules together to create a J2EE application that is ready to deploy into an operational environment. Figure 6-20 illustrates the various types of J2EE modules and how they can be deployed.

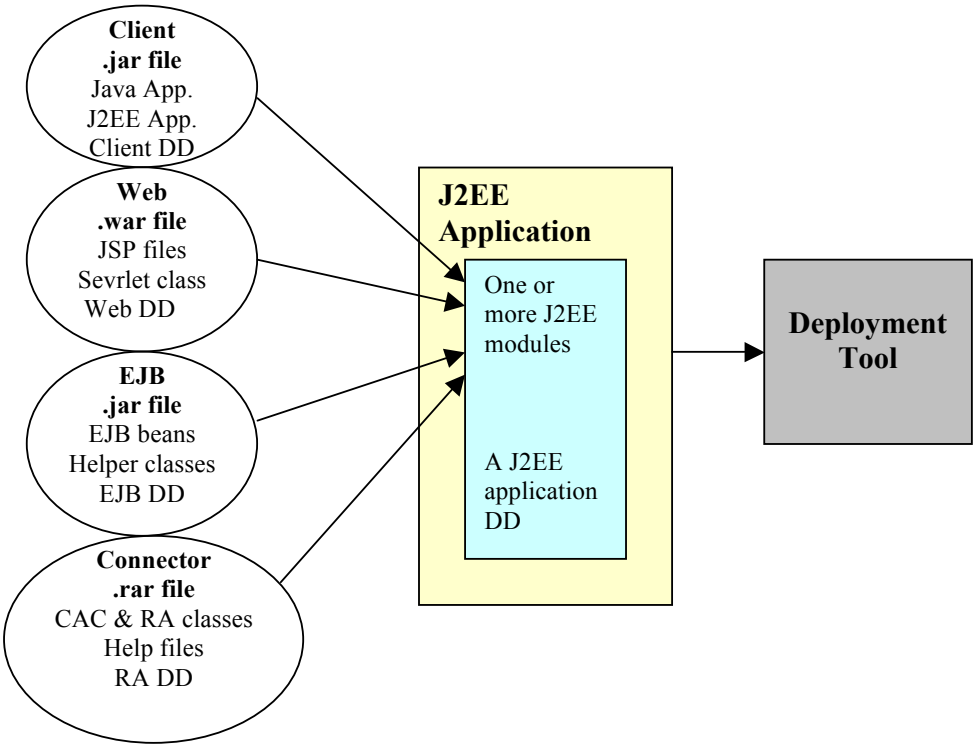


Figure 6-20 J2EE packages

Each module is formatted in a special type of file. A J2EE application is packaged as a portable deployment unit called an enterprise archive (EAR) file which is standard JAR file with an .ear extension. A deployment descriptor (DD) XML file describes the relationships between components and their external dependencies or assembly information. A client application jar file may contain java applications, J2EE applications and the deployment descriptor (DD) XML file. An EJB module formats into a jar file that contains java class files for the enterprise beans and their remote and home interfaces, helper java class files that the enterprise beans depend on and an EJB DD file. A Web module is formatted into a Web archive (WAR) file which is a JAR file with .war extension. It contains JSP files, Servlet class, static document (HTML, images and so on), applets files, and a Web DD file. A connector module formats into a resource adapter archive (RAR) file which contains java class files for implementing both Connector Architecture Contracts (CAC) and the resource adapter (RA), helper files and a resource adapter DD file. In J2EE NetPay-enabled vendor system, we will use Web and EJB modules.

6.5.2 NetPay Components Plug-in

The Ewallet and Redeem EJB components are plugged into the existing E-journal system by deploying them into the E-journal system's J2EE server. The Ewallet component is used to obtain an e-wallet from the broker or another vendor, make payments by using the client-side or server-side Ewallet managed e-coins, and generate payments data. The Redeem component is responsible for selecting payments and sending these to the NetPay broker. EJB deploytool provides an interface to define relationships between enterprise beans. This makes it easier to plug-and-play components. Figure 6-21 illustrates the interface and the relationship between Ewallet component and Redeem component. There is no relationship among existing journal component (ArticleDBJAR) and NetPay components. The NetPay components are plugged in the existing system very straightforwardly.

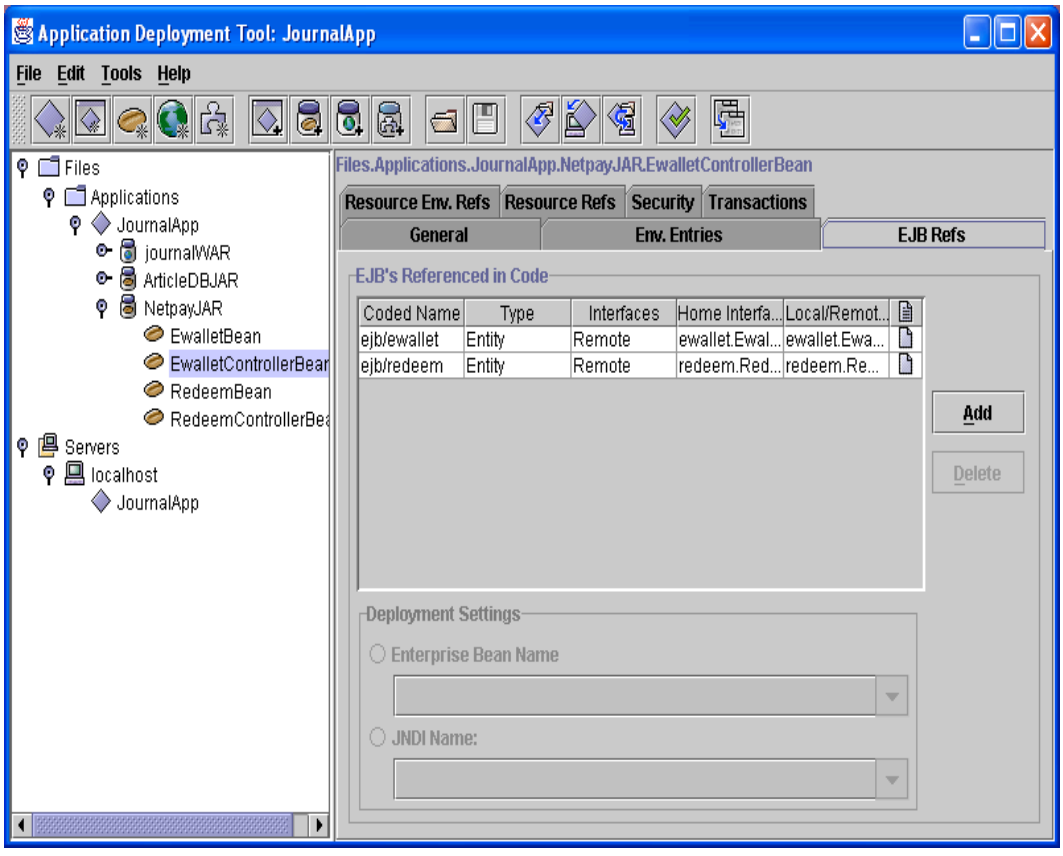


Figure 6-21 Plugging in the NetPay vendor-side components with a J2EE deployment tool

6.5.3 Implementation

In this section we will use screen dumps to depict the E-journal example component and a NetPay-enabled E-journal example component. Internet Explorer 5.5 is used to display the HTML content. We describe the basic user interfaces of the components and explain interactions with examples.

6.5.3.1 E-journal example system

When a customer uses the web browser to access the E-journal example system, the E-journal home window will appear. On this window (Figure 6-22a), each article title and the author are displayed on the screen. The customer could choose a desired article and click the title of the article. The article content is displayed on the content window (Figure 6-22b). Consider a case when a user, Leanne, wants to read articles on the E-journal example site. She accesses the site using her web browser and finds the second article to read as shown in Figure 6-22a. She then clicks the title of the second article and the content of the article is displayed on the screen as shown in Figure 6-22b.

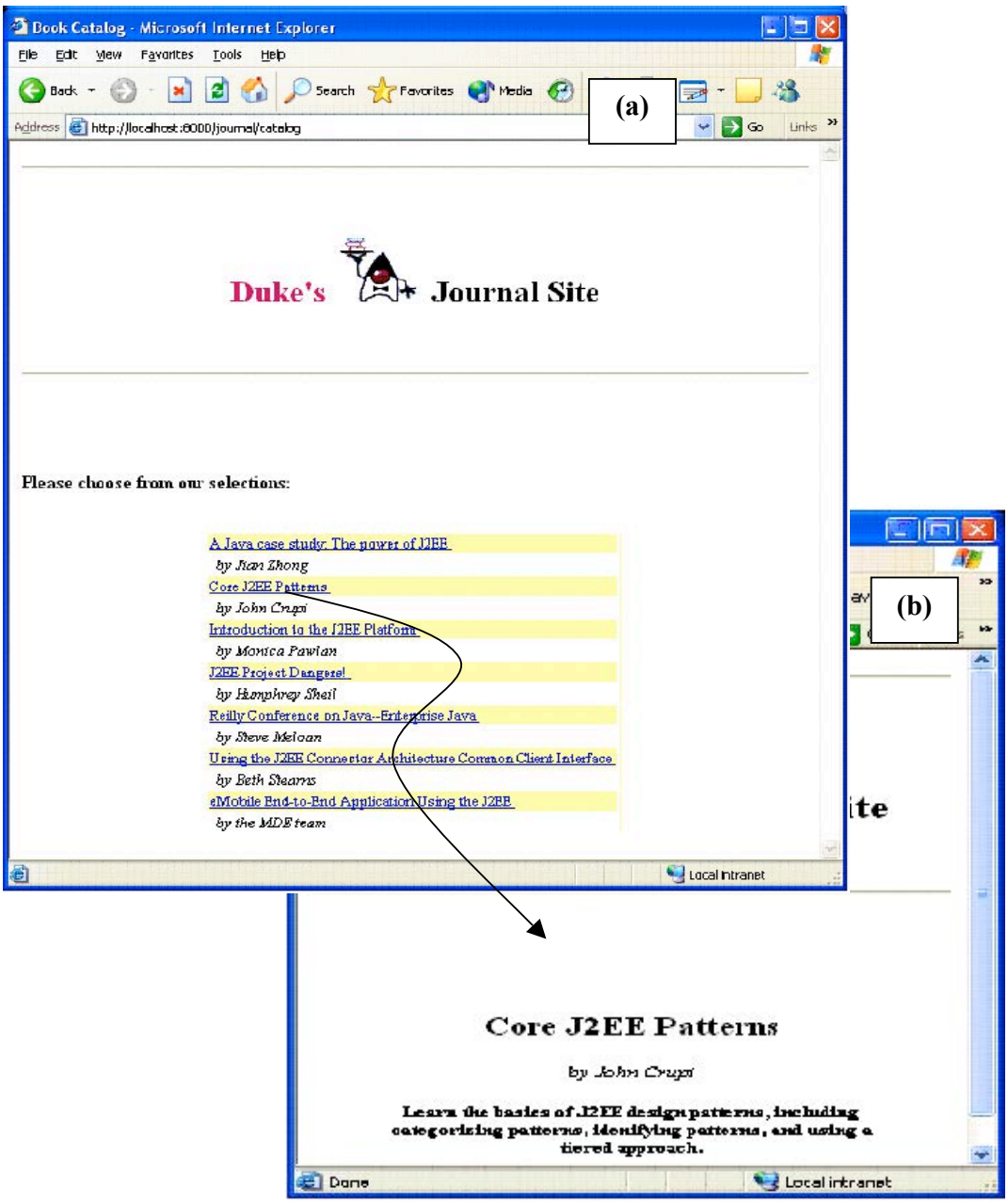


Figure 6-22 Example of non-NetPay E-journal system

6.5.3.2 NetPay-enabled E-journal example system

Figure 6-23 shows the click-buy articles interface for the E-journal after extending the system via the NetPay components. The Login JSP page is added to the system in order to get the e-coin ID from a customer when using server-side e-wallet NetPay (Figure 6-23a). Any input data verification is done locally. An error message will be displayed if there is a bad data. For example the customer presses the Login button without entering e-wallet ID

or password or the customer enters an invalid id. The article and content JSP pages were modified to include our NetPay micro-payment JSP includes so that e-coin credit and article prices are displayed on the screen (Figure 6-23b). Article pricing is stored in the NetPay database and the JSP includes are parameterised with article information (category, title, URL) to look up the appropriate pricing.

Consider a case when a user, Leanne, goes to the NetPay-enabled E-journal site and enters e-wallet ID e.g. 345 and password and then clicks Login button as shown in Figure 6-23a. After Leanne successfully accesses the system, the "You have got 70cs left!" information and each article price are displayed on the home window. She can start to browser the site and find an article that she wants to read as shown in Figure 6-23b. She clicks the title of the second article which costs 20cs and the remaining e-coin credit and the content of the article are displayed on the content window as shown in Figure 6-23c.

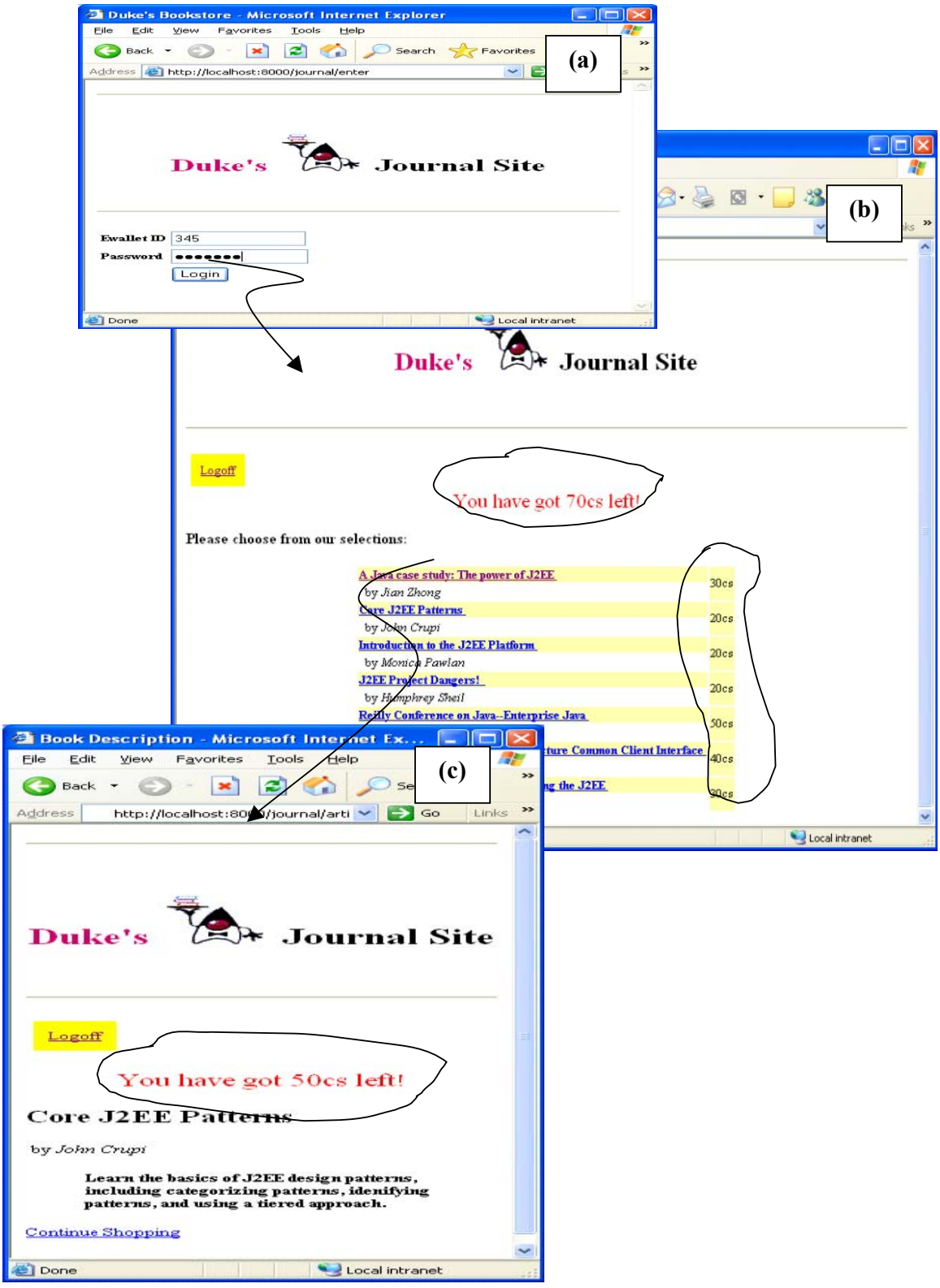


Figure 6-23 Example of NetPay-enabled E-journal system

When the customer first tries to read an article, the vendor obtains their e-wallet which includes the validating touchstone, index and e-coins from the broker, in order to verify that

the e-coins are valid [23]. When moving to another vendor, the touchstone and current index value of the e-coins are obtained from the previous CORBA vendor or EJB vendor.

6.6 Summary

We have built NetPay vendor Enterprise JavaBeans to provide plug-in vendor micro-payment support components and plugged in EJBs into the E-journal's existing application server and annotated the E-journal's JSPs to make appropriate function calls to the NetPay EJBs. This allows for minimal or no code impact (none if using proxy JSP pages) to the existing system's infrastructure. The NetPay EJBs are completely separated from the particular domain knowledge of the web application, enabling each enterprise bean to be reused in different EJB-based vendor systems via plug-and-play with the existing vendor components. The NetPay vendor system components have been designed, implemented, plugged into the E-journal example system, and successfully deployed to a J2EE server running the E-journal web site.

NetPay vendor functions could be integrated into an existing CORBA-based system in a similar manner and deployed with existing CORBA remote objects i.e. via standard ORB. However we would have to modify the existing objects or web server pages that call these objects to invoke the NetPay pay-per-click functions when needed.

Chapter 7

Evaluations

In this chapter, we describe three kinds of evaluations we have done on our NetPay prototypes, to assess micro-payment verses macro-payment usability, performance and overall qualitative characteristics for E-newspaper systems' payment methods. We compared two kinds of NetPay-based micro-payment systems (client-side wallet and server-side wallet) and a subscription-based macro-payment system.

7.1 Motivation

After having developed a new model of micro-payment for e-commerce applications, and building a prototype of this system - NetPay, we wanted to assess its worth compared to macro-payment systems. We had also developed two models for managing electronic coins ("e-coins") in our NetPay system – managing e-coins in a client-side electronic wallet ("e-wallet"), where the encoded coin information resides on a customer's computer, or in a server-side e-wallet, where the coin information resides on vendor servers and can be exchanged from vendor to vendor.

To carry out an evaluation of NetPay and compare its two e-wallet support approaches to traditional macro-payment based E-newspaper payment methods we assessed the characteristics of NetPay-based micro-payment systems from several perspectives. We wanted to assess and understand customer perceptions of using NetPay and the advantages and disadvantages they saw with the system, as well as gain an understanding of the usefulness of the approach for information vendors.

The impact on the performance of the vendor system of debiting coins and tracking payments must be light to make the system practical. Basically a micro-payment system should provide a lightweight mechanism for paying for on-line content where there are a large number of payments for quite small units of information [23, 42, 27]. Such a system needs to support both the buying of coins, and electronic money, by customers and per-click debiting of coins by vendors [33].

To evaluate our prototypes we determined that three types of evaluation would be required: a usability evaluation, to assess the user's perception of NetPay-provided features, and to compare these against conventional macro-payment subscription-based payment models [60, 69]; a performance analysis of NetPay added to a vendor system; and a qualitative analysis of NetPay's functional characteristics [26]. Each of these evaluations only examines some of the issues to do with usability, performance impact and qualitative assessment of the NetPay prototypes. In the following sections we report results of:

- Usability evaluation via a survey-based approach with representative target users of NetPay. We then analyse the results from this evaluation to determine if NetPay is usable as far as target users were concerned.

- Performance impact evaluation of our NetPay prototype was carried out to determine if adding it to typical vendor web servers would be viable for large transaction loading. The results are discussed to determine if the performance overhead of NetPay micro-payment would be acceptable to information vendors.

- Qualitative assessment of NetPay and conventional macro-payment approaches was done to determine how well our model and prototype compare using some common assessment criteria and whether NetPay meets the basic requirements for a micro-payment system for on-line Web applications.

7.2 Usability Evaluation

Usability evaluation is a process that measures the ease of learning, easy to use, and the effectiveness of a product [74]. There are a number of methods that may be used to capture the information to support usability evaluation such as focus group, user testing, observation, survey/questionnaire, interview, performance measure etc [61]. We choose to use a task list and post-survey questionnaire rather than other usability evaluation approaches. A usability evaluation in this section surveyed users of the prototype to assess their impressions of the approach when carrying out information purchasing tasks using a micro-payment verses a macro-payment protocol. We focus in this experience on the usability of the NetPay prototype user interface using several common usability measures.

Another measure would be interface suitability i.e. for each user task, does the prototype provide appropriate support for the user. We have not considered this except via open-answer comments from our surveyed users.

7.2.1 Procedure

We evaluated participants' user satisfaction, navigational efficiency, effectiveness and general preference for the three payment systems – subscription-based macro-payment, server-side and client-side NetPay micro-payment with two newspaper sites [69, 74]. These measures are the standard ones for determining how "usable" an interactive system is, and allow us to make judgements on the suitability of the interface for the tasks being carried out [60]. Efficiency was measured by the degree of ease to change different newspaper sites and the speed of article content loading. Effectiveness was measured by assessing operations needed by the customer to complete their purchases. Satisfaction was a subjective measure assigned by each participant in the experiment.

Ten participants were invited to participate in the usability tests. They are students from Department of Computer Science and Mechanical Engineering. They were selected randomly in student labs and were willing to attend the test. They did not know the researcher. Every participant was required to fill out a pre-test questionnaire (see Appendix A), which is used to find out his/her computer background and experience of using E-tailing web sites. Ten participants volunteered for our usability study of NetPay. They were an equal mix of non-IT specialists and graduate students from the departments of mechanical engineering and computer science at the University of Auckland, the later who were frequent users of on-line information portals. All participants were familiar with using E-tailing web sites, particularly for purchasing books, CDs and clothing.

The tests were done in the Computer Science Department offices at the University of Auckland. The application servers used are: newspaper1 and newspaper2 providing subscription-based macro-payment; broker, newspaper1, and newspaper2 providing server-side NetPay micro-payment; and broker, newspaper1, newspaper2 providing client-side micro-payment. These application servers and web server were deployed for this experiment on the same host on the Windows XP network. The participants in the

experiment used other PCs connected this network to carry out a set of tasks including registering, subscribing, buying coins, reading articles and reading articles over multiple sittings.

The participants carried out a set of information purchasing tasks from our three E-newspaper prototypes, one using subscription-based macro-payment; one using a client-side NetPay e-wallet and one using a server-side NetPay e-wallet. Participants were asked to complete five tasks with each system.

- Subscribe to the newspaper site or register and buy e-coins with a broker
- Read 3 articles on newspaper1 site
- Change to newspaper2 site and read 3 articles
- If subscription expired or e-coins run out, the user must renew it
- Read articles on the two vendor sites a second time, subsequently to the first use of the system

Firstly every participant undertook all these tasks according to the task lists. Then they could use the three systems across a number of sessions. After completing these information searching and access tasks with each of these payment systems, participants were asked to fill out a post-test questionnaire (see Appendix A) that contained a subjective rating assigned to each tested characteristic for each payment system.

7.2.2 Results

In the post-test questionnaire, we used a 5-point rating scale (1= Strongly Disagree, 5=Strongly Agree) to rate each tested characteristic. We also included open questions to gain user feedback to help in the qualitative analysis evaluation work. We presented the average ratings for the tested characteristics in a bar chart form as shown in Figure 7-1. The tested characteristics are:

a. *Ease of use*: Payment system is easy to use.
b. *Efficiency1*: It is easy to move around different newspaper sites.
c. *Efficiency2*: The speed of article content loading is fast enough.
d. *Efficiency3*: It is easy to deal with subscription expired or e-coin run out.
e. *Preference*: You preferred to use this system widely.

We choose to use some common usability criteria [74] so that we could assess the overall usability characteristics of the NetPay prototype. The usability evaluation involves only a small user base.

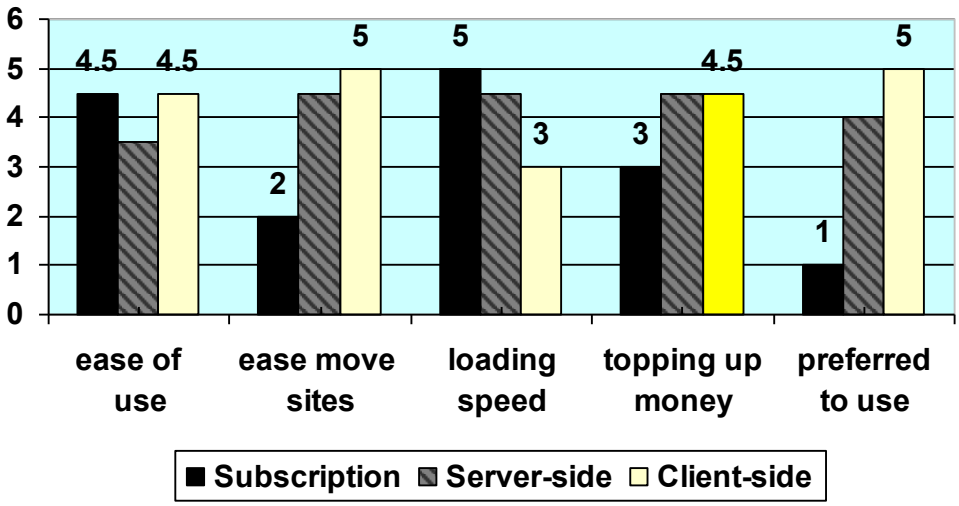


Figure 7-1 Three payment systems usability test results

In this study, ease of use, efficiency, and satisfaction/preference results mainly favoured the client-side e-wallet NetPay system. However, this approach incurred an extra delay in page display due to communication from the vendor to the customer PC's e-wallet application, which the other systems don't have. Participants stated the article contents at different newspaper sites were easy to read without log in and the balance can be checked any time. The server-side NetPay system allowed users to read articles on different computers, but customers needed to remember e-coin IDs and had to log into the new newspaper site when changing vendor. The article content loading was very fast on the subscription-based system, but the users found that it is not as convenient to change vendor. The users generally needed to spend more money in order to subscribe to the whole newspaper provided by each site. Open question results revealed that client-side NetPay was found to be significantly preferred over a subscription-based system. In addition, server-side NetPay was more preferred than subscription-based system for this E-newspaper application domain. The participants did the three systems in the same order, i.e., reading articles first and then changing sites. This is a weakness of the experiment.

7.3 Performance Impact Evaluation

A performance impact evaluation assessed the performance of NetPay-enabled web sites to determine the overhead of the micro-payment extensions made to the software, particularly in regard to user response time and additional overhead on server CPU time.

7.3.1 Design

Our three prototypes providing subscription-based payment, server-side and client-side NetPay micro-payment have been tested for application server performance and client response time. The key aim was to test how long a newspaper site takes to serve client requests when extended to use each of the three payment systems, from the time the customer clicks the title of an article to the time the article is fully displayed on screen. In order to do this we developed a pseudo-web browser to perform large numbers of requests to the web server and to time the response time of the web server. The macro-payment subscription-based approach makes one expensive macro-payment debit for pay for the initial subscription and then simply checks the whether a customer, after login, has a valid subscription. The micro-payment systems need to carry out an e-coin debit of the customer's e-wallet with each purchase of an article. We measured the CPU time taken by the vendor's web server and the overall time taken to action the page display. The longer the delay to display a page, the more problematic for the customer in terms of vendor information response time. Users probably want to response in a few seconds. Anything shorter doesn't matter e.g. ½ second to 2 seconds, but longer e.g. greater than 5 seconds would not be good. The more CPU consumed by the server-side, the less overall client requests and lower response time overall can be supported. This evaluation is not considering e.g. amount (total size) of data that the vendor server must store to redeem coins. It is looking at response time and vendor server loading, but other measures could be data size impact, cost (e.g. broker fees, macro-payment fees) and so on. We did the tests on machines on local, high-speed LAN which was unloaded by others at the time. However the results would be very different if across WAN or heavily loaded network.

7.3.2 Results

We set up the performance tests in an office in the Computer Science Department at the University of Auckland. The application servers are subscription-based macro-payment newspaper, server-side NetPay micro-payment (broker and newspaper), and client-side NetPay micro-payment (broker and newspaper). These application servers and web server were deployed for this experiment on the same host on the Windows XP network. In these tests, we were concerned about the overhead of the NetPay extensions made to the vendor system only. The network delay is not considered here but it may be a factor. Customers in the tests used the same PC to read articles over the three systems.

The diagram Figure 7-2, shows how a customer click-buys article content with E-newspaper site using server-side NetPay. When buying an article the customer selects the article for reading e.g. clicking on the URL in a returned article search page. The web browser requests the article content from the JSP that requests payment for the content of the article from the e-wallet bean. The e-wallet bean communicates with Vendor Application Server (VAS) to debit the customer's e-wallet to pay for the article. If the e-wallet does not exist, VAS requests e-wallet location from the BAS and e-wallet from BAS or VAS. VAS then verifies the e-coins by using the touchstone and the index. If e-coins are valid, VAS stores them in the redeem database and the news article content is displayed to the customer.

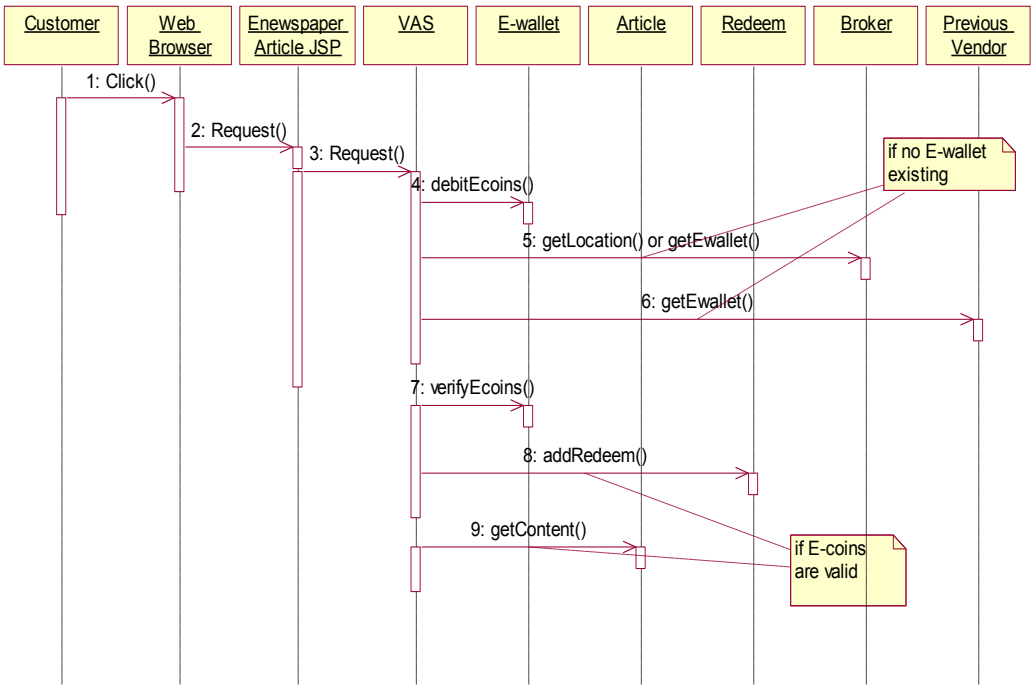


Figure 7-2 Click-buy article content with server-side NetPay sequence diagram

We ran two sets of performance impact tests, one on our original NetPay prototypes and a comparable macro-payment subscription-based system and one on modified versions of the NetPay prototypes, optimized to provide lower e-coin management overhead. This was done due to the large database overhead the original prototypes incurred for debiting e-coins.

The results of the first set of performance impact tests are shown in Table 7-1. The response time measures how long it takes for a page to be returned from the vendor site. The server CPU time measures the time spent in the vendor's server debiting NetPay e-coins. We didn't consider caching vendor-to-vendor impact.

Table 7-1 Initial prototype performance

System	Response Delay Time (average)	Server NetPay CPU Time Usage (average)
Subscription-based	16ms	N/A
Server-side NetPay	80ms	64ms
Client-side NetPay	950ms	60ms

From Table 7-1, the server-side NetPay takes 80ms-16ms=64ms for e-coin debiting per article and Client-side takes 950ms-16ms=934ms total time, though the time to debit coins is taken by the client's e-wallet application, not the vendor's application server. The large overhead in the server for the server-side NetPay prototype is due to the database transactions it carries out to record coin updates and debits to redeem to the broker. Note that multi-threading in the server allows the vendor to serve other clients during NetPay debits but the server-side e-wallet incurs a heavy duties overhead.

To reduce the e-coin debiting time, we created a transaction temporary file recording the data for redeeming instead of the redeeming database. Because of the optimized efficiency of such a temporary file, the e-coin debiting time decreases dramatically especially for server-side NetPay system. The results are shown in Table 7-2. At the end of each day, the system redeems the coins or updates the database, and then deletes the records in the transaction temporary file. From Table 7-2, Server-side NetPay takes 30ms-16ms=14ms for e-coin debiting per article and Client-side takes 900ms-16ms=884ms after the application of the temporary file. The impact of the NetPay micro-payments on the vendor application server are greatly reduced, but the client-side e-wallet still incurs considerable response time delay due to the need for the vendor to communicate back to the customer's PC hosted e-wallet application for each e-coin debit.

Table 7-2 Prototype performance after using a temporary file

System	Response Delay Time (average)	Server NetPay CPU Time Usage (average)
Subscription-based	16ms	N/A
Server-side NetPay	**30ms**	14ms
Client-side NetPay	**900ms**	12ms

The e-coins debiting time contains two parts which are debiting e-coin from an e-wallet and storing spent e-coins to a database for redeeming. Using a temporary file instead a redeem database to store redeem data decreases the time of storing spent e-coins due to the database on the server to record redeem data. The server NetPay CPU time for a client-side and a server-side NetPay are almost the same, because the servers need to update the e-wallet in server-side NetPay or update the touchstone&index for client-side NetPay, and stores redeem records in both systems. A client-side cookie-based e-wallet NetPay micro-payment system described in Section 5.2 could be used to reduce the time of debiting e-coins from the customer's e-wallet in the customer's PC because the vendor application server debits the e-coins from cookie e-wallet directly.

7.3.3 NetPay Case Study Using Argo/MTE Tool

Argo/MTE is an architecture modeling tool [37]. It provides a methodology to automate the process of software architecture analysis, design, and evaluation. Argo/MTE can generate fully functional applications called a performance test bed for any intended system architecture designs. It also manages a large distributed environment for carrying out the architecture performance testing [13].

A performance impact evaluation for the NetPay vendor systems was described in previous sections. However, the evaluation only involved one customer simulated using the NetPay-enabled vendor systems and purchasing articles at a time, to determine the theoretical maximum response time of a NetPay-enabled web site. However, the article display time must be an acceptable time under much heavier, concurrent loading of customer requests for practical NetPay-enabled applications. In a case study of the NetPay architecture performance by Yuhong_Cai [13], ten customers were simulated as purchasing an article's contents simultaneously in a performance evaluation of the server-side NetPay-enabled

newspaper system. Note that while this has provided some useful results, we also need to add NetPay to a real existing web application and carry out a more thorough performance evaluation to assess if a NetPay-enabled vendor system would be viable for large transaction loading in the future.

The web-based multi-tier architecture for both NetPay broker and NetPay-based vendor systems was designed in Chapter 5. NetPay micro-payment transactions involve three key parties: the Broker Server, the Vendor Server, and the Customer browser. The architecture is illustrated in Figure 7-3.

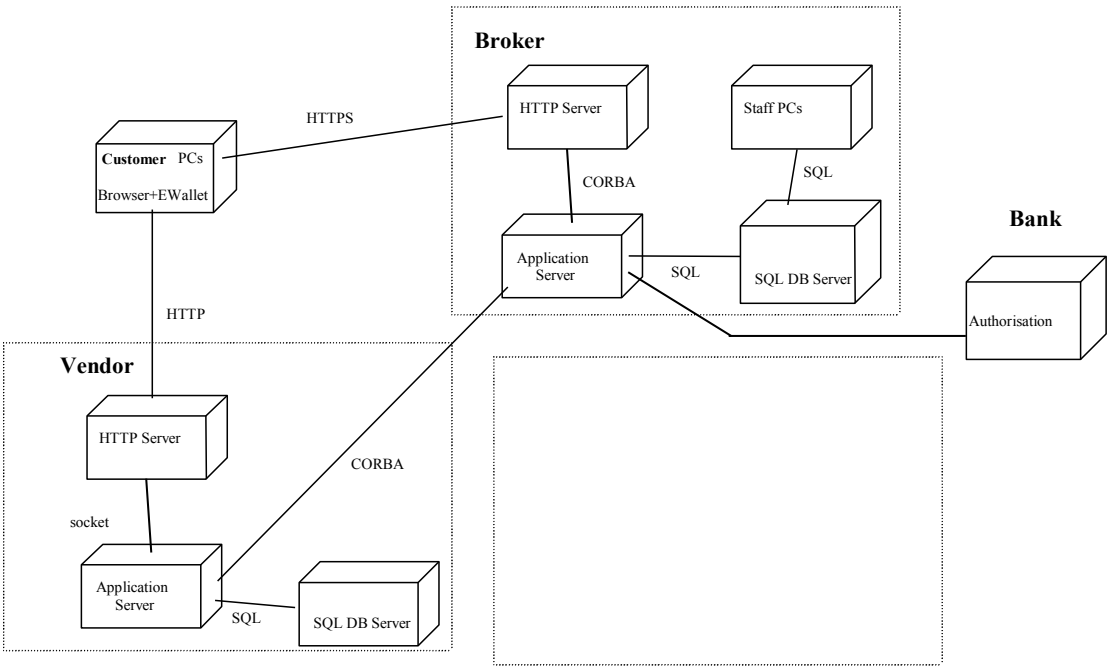


Figure 7-3 NetPay system deployment diagram

A customer runs a web browser that accesses the broker and vendor services, and may also contain an e-wallet implemented by the use of a Java application.

CORBA is used to communicate among the broker application server and the vendors' application servers via WAN/LAN. The broker application server provides a set of CORBA interfaces with which vendor application servers communicate to request touchstones and redeem e-coins. A vendor application server communicates with the broker application server to obtain touchstone information to verify the e-coins being spent and to redeem spent e-coins and other vendor application servers to pass on e-coin indexes and touchstones.

A NetPay architecture performance evaluation case study was carried out by using the Argo/MTE tool [13] to generate a test bed which represents an architecture of a NetPay real system, to do a large scale performance evaluation of the generated test bed and the NetPay real system, and to compare results of the performance evaluations. The main components were picked up from the NetPay system in order to generate test bed for NetPay system.

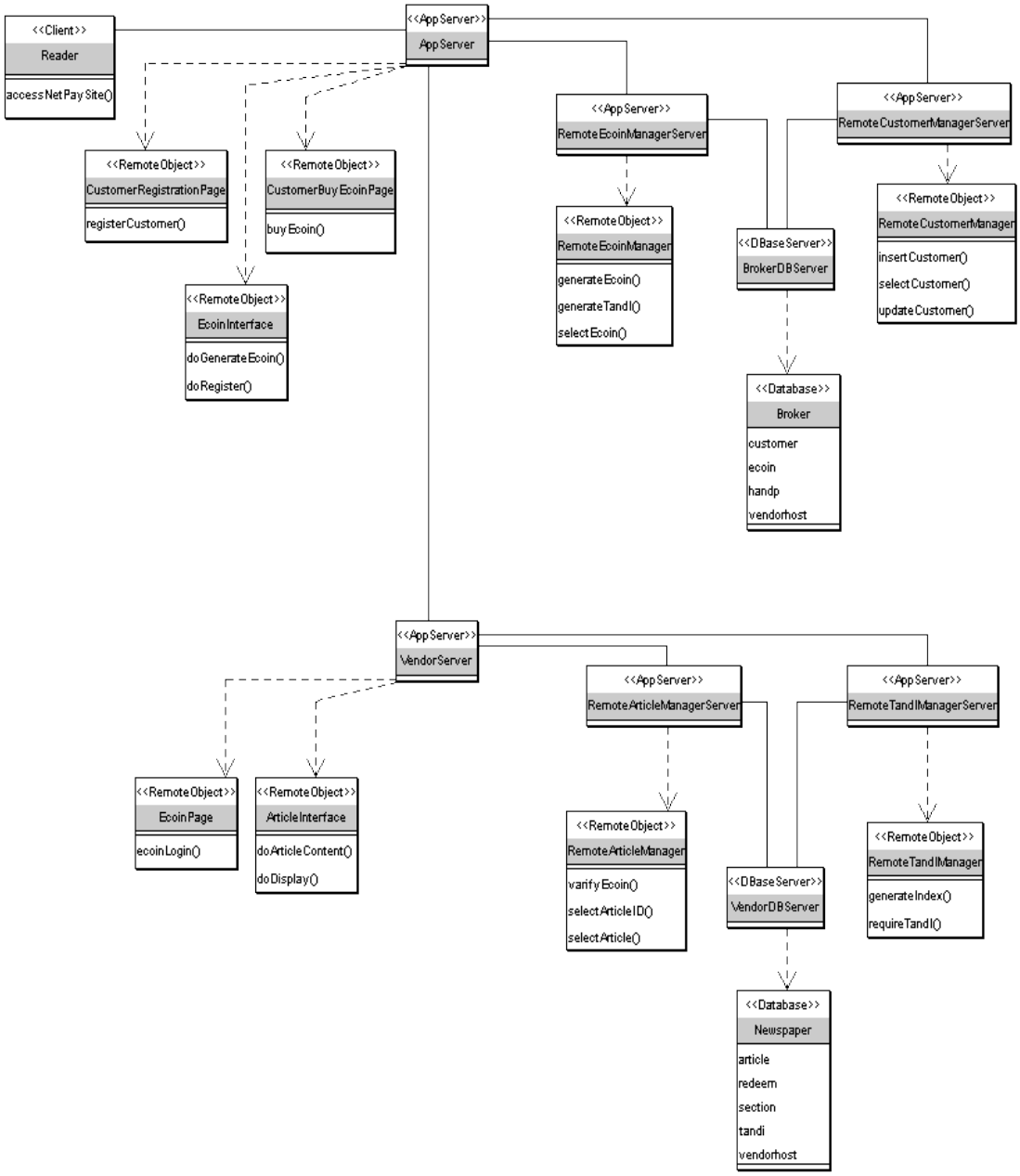


Figure 7-4 The complicated architecture of NetPay software system

Figure 7-4 shows the complicated architecture of NetPay software system. The diagram was drawn using the Argo/MTE tool developed by Yuhong Cai. There are 20

entities/components involved in the architecture. Each entity belongs to a type, and entities within the same type show similar characteristics and play similar roles in the system. For example, entities “EcoinPage”, “ArticleInterface”, and “CustomerRegistrationPage” are within the type of “RemoteObject”, and they all are hosted by application servers and provide services for remote clients. All components in architecture are connected, and different connection specifies different relationship between two connected entities. For example, the connection between entities “Reader” and “AppServer” represents the association relationship that “Reader” uses “AppServer”. The connection between entities “AppServer” and “CustomerRegistrationPage” represents the ownership relationship that “AppServer” owns “CustomerRegistrationPage”.

Entity “Reader” is within the type “Client”. This entity represents users of NetPay system. By assigning suitable values to the design parameters of this entity the generated test bed can contain many concurrent users. There are four application servers involved in the system. Each server hosts one or more remote objects that provide services for remote clients. The system also uses two database servers: “BrokerDBServer” which hosts “Broker” database and “VendorDBServer” which hosts “Newspaper” database.

The real NetPay system and its test bed are compared in the case study. The two systems have very similar structure and consist of same components and same relationships among components. Even when comparisons were made at method level, the structure of both systems is close enough. The differences between two systems are: (1) the NetPay real system contains some simple business logic while its test bed doesn’t contain any; (2) the NetPay real system has small-sized data passing around while its test bed does not pass around any data.

The tests were done in the Computer Science Department offices at the University of Auckland. The application servers used are: “RemoteEcoinManagerServer”, “RemoteCustomerManagerServer”, “RemoteArticleManagerServer”, and “RemoteTandIManagerServer” for both the NetPay real system and its test bed providing server-side micro-payment. These application servers and web server were deployed for this experiment on the same host on the Windows XP network. Ten threads were set up to simulate ten customers simultaneous accessing the system.

The generated NetPay test bed performance and the NetPay real system component implementations' performance were compared and it was found that both the test bed and prototype NetPay system performance are reasonably close. The systems were tested under a load of ten customer simultaneous accesses being simulated in the performance evaluation. In the case study, 15 tests were done in order to analyze the difference in performance between the performance of NetPay real system and that of its test bed

Table 7-3 gives the results of the performance for the generated test bed code and the prototype NetPay server-side E-wallet system code for an article content buy and process, and an average response delay time for both systems. The average display response delay time for an article content display with the server-side NetPay system is acceptable. These were "warm-up" values, and ideally could be discarded. These results show that when 10 simultaneous client requests for an article are made to our NetPay-enabled prototype web site, on average it takes 247ms for a response to be sent back to a client for display by their browser. The maximum delay encountered was 593ms, which is actually during the first test run and is an outlying result.

Table 7-3 The performance for an article content display process

Test	Response delay time with the generated test bed code (ms)	Response delay time with the prototype server-side NetPay system code (ms)
1	407	593
2	172	282
3	187	218
4	157	219
5	172	265
6	157	203
7	188	203
8	172	218
9	187	218
10	188	219
11	203	203
12	203	203
13	187	203
14	188	266
15	203	203
Average	**198**	**247**

Further performance testing is required of the NetPay prototype and any real deployment of NetPay to further assess its performance. This would include varying numbers of simultaneous client requests e.g. 5 to 50; heterogeneous client requests e.g. buying different articles from same vendor; and varying configurations of the web server e.g. different thread pool sizes, memory allocation to threads and database connection pools. Comparing performance of NetPay under high simultaneous loads to non-NetPay, subscription based E-newspaper would be interesting too.

7.4 Qualitative Comparison to Macro-payment

Our qualitative evaluation assessed factors such as customer effort in using a NetPay-enhanced web site from a customer's perspective, along with the cost/benefit of the system for customers, vendors and brokers.

7.4.1 Design

The qualitative assessment of our three prototype E-newspaper web sites assesses various factors associated with their costs and benefits for customers, vendors and the broker organization. The assessment criteria included:

- Number of customer interactions with the web site(s) needed to read articles
- Information retention needed by customers to use web site(s)
- Cost to customers depending on subscription and article pricing and article usage
- Cost to vendors of subscription authorization and e-coin redemption
- Cost to brokers of providing e-coins and redeeming coins with banks

The results for this analysis were obtained from analyzing the performance of each payment method in order to satisfy a payment scenario.

7.4.2 Qualitative Analysis

With our qualitative assessment we wanted to measure the different characteristics of macro-payment and micro-payment approaches, and to analyse the differences between our two NetPay e-wallet models. The results of this assessment are summarized in Table 7-4a

and Table 7-4b. In the table 7-4a, "+" means the advantages of the protocols and "-" means the disadvantages of the protocols.

Table 7-4a: Qualitative assessment summary

Criteria	Macro-payment	Server-side Wallet	Client-side Wallet
Customer interactions	1.Subscribe (customer and credit card details) 2.Login to web site 3.Article read 4.Subscribe if move to another vendor +after subscribe/login simply read articles -must supply personal details -must subscribe for each vendor	1.Purchase e-coins from broker 2.Login to web-site 3.Article read 4.Login to new vendor +after login simply read articles +only login to new vendor -must recall e-coin ID, password -must supply for each vendor	1.Download wallet software 2.Purchase e-coins 3.Article read +don't need login/password for any vendor +simply read articles -must download, install and have running client-side e-wallet software
Information retention	Need to remember username/password. May avoid if use same PC always (can store in browser cookies)	Broker username and password needed to purchase coins. E-coin ID and password needed to access server-side wallet information.	Broker username and password needed to purchase coins.
Subscription cost Low e.g. $10 High e.g. $50	Low – if use moderate number of articles, more cost-effective for customer. High – need to use substantial amount of articles for benefit. No cost savings for customers if use multiple vendor sites.	N/A	N/A
Article cost Low e.g. 2c High e.g. 10c	N/A	Low – customer likely to read more. Vendor needs more reads to cover costs. High – customer likely to read less. Cost savings to customers. Can price articles differently.	Low – customer likely to read more. Vendor needs more read to cover costs. High – customer likely to read less. Cost savings to customers. Can price articles differently.

Table 7-4b: Continue of qualitative assessment summary

Criteria	Macro-payment	Server-side Wallet	Client-side Wallet
Article requests by customers Low e.g. <10 High e.g. >20	Low - No cost benefit for vendor High - Large number by many customers effects system performance	Low – if low cost, vendor makes little profit. High – if high cost to customer, may be more costly than macro-payment approach High numbers impact overall vendor server performance.	Low – if low cost, vendor makes little profit. High – if high cost to customer, may be more costly than macro-payment approach. Has large response time impact (in current implementation). Very high numbers impact overall response time of vendor server.
Vendor Benefit	"Brand capture" of customers due to use of subscription to each vendor site.	If large enough vendor community can encourage movement, partnerships. Customers need to login to access wallets.	If large enough vendor community can encourage movement, partnerships. No login for wallet access needed but need wallet installed on customer PCs.
Vendor cost	Need to buy macro-payment supporting software and pay bank for facility. Need to price subscription to adequately cover costs.	Need to allow broker to take portion of overall customer payments OR broker takes costs from customer direct. Need to price articles so cost per article/ and number of articles used cover costs. The performance overhead on the vendor server is significant.	Need to allow broker to take portion of overall customer payments OR broker takes costs from customer direct. Need to price articles so cost per article/ and number of articles used cover costs. There is little performance overhead on vendor server but response time reduction for customer.
Broker cost	N/A	May charge vendors for each e-wallet request or portion of redeemed coin amount. May charge customer for each e-coin purchase. Possibility of high number of e-coin requests from vendors.	May charge vendors for each e-wallet request or portion of redeemed coin amount. May charge customer for each e-coin purchase. Low overall e-coin requests as client-side wallet brokers these.

7.4.3 Summary of Results

In summary, a macro-payment approach is more beneficial for the customer if they typically read a large portion of the on-line newspaper articles, or if a comparable micro-payment approach has a high-cost per article for the user. However, the micro-payment approach wins out when the customer typically users a small portion of the articles, articles

are low-priced and if the customer reads articles from multiple newspapers and can use their e-coins across any of these vendors. There is a performance cost for the vendor in providing a micro-payment approach in terms of time taken to track e-coin spends and redemption. However, there is also normally a high cost to the vendor of providing macro-payment support for subscription purchase.

One interesting issue is whether vendors would "buy in" to a micro-payment system approach. By using subscription-based macro-payments to access information, vendors can lock in customers i.e. achieve "brand capture" and discourage customers from moving to other information sources e.g. other E-newspapers as they have already made a significant financial commitment to one newspaper. Similarly, there needs to be sufficient vendors sharing the same micro-payment system and "currency" to allow useful movement by customers from vendor to vendor. In addition, the software maintenance overhead of installing a micro-payment system must be considered by vendors. One approach is to adopt a portal-based approach to accessing multiple vendors through a single micro-payment enabled portal which does the debiting and redeeming of spending on behalf of multiple vendors.

Many existing web sites provide services to users for free but use advertising embedded within pages or as pop-up windows to generate revenue. However, studies have suggested that vendors would prefer a payment mechanism which is on a per-usage basis, either subscription-based or micro-payment-based to provide greater reliability of income [6]. A key outstanding challenge with micro-payment systems is being able to spend currency (e-coins) at a wide range of vendors – if the customer must purchase different "currencies" for different groups of vendors then this will be both inefficient in terms of expenditure and incur overheads of the customer memorizing different usernames, e-coin Ids and passwords. One approach is to support inter-broker micro-payment e-coin exchange transparently when the customer visits a vendor that uses a different broker's currency.

7.5 Summary

We described three kinds of experiments we have done on our NetPay prototype, to assess micro-payment versus macro-payment usability, performance impact and overall qualitative characteristics for e-commerce systems payment. We compared two variants of our micro-

payment system deployed with an on-line newspaper web site and a macro-payment, subscription-based variant of the web site. These evaluations have indicated that for users the micro-payment approach has some appeal over traditional macro-payment approaches and that for some usage patterns the micro-payment approach is far more efficient in terms of cost to the customer.

Chapter 8

Conclusion and Future work

This chapter summarizes the work done in this thesis and presents some possible future work in this area.

8.1 Contributions

There are a number of micro-payment systems such as Millicent, Mpay, and PayWord. Most existing micro-payment technologies proposed or prototyped to date suffer from problems with communication, security, lack of anonymity and many are vendor-specific. The main contributions made by the research described in this thesis include:

- Proposing a new protocol called NetPay to address problems with communication, security, anonymity and many are vendor-specific. The NetPay protocol shifts the communication traffic bottleneck from a broker and distributes it among the vendors by using Touchstones and Indexes. Customers are prevented from double spending as the index of the payword chain indicates the balance of the customer's e-wallet**,** and the touchstone can be used to verify the payword chain has not been tampered with. The protocol protects the customer's privacy from vendors because vendors only know customers' ID. NetPay allows customers purchasing on-line content from multiple vendor sites by using an e-coin chain which is not vendor-specific.

- Developing a software architecture for implementing NetPay-based micro-payment systems for thin-client multi-tier web applications. There are three kinds of NetPay systems that have been designed, which include server-side e-wallet, client-side e-wallet, and client-side cookie-based e-wallet. Each of these NetPay systems has its own advantages and disadvantages.

  Using the server-side e-wallet, customers can buy on-line content anywhere in the world and content display time is faster than the client-side NetPay systems. However

customers need to remember e-coin IDs and password to log into a vendor site when he/she wants to buy contents.

Using the client-side e-wallet, customers can buy on-line contents at different vendor sites without log into the site at their PC. Customers need to download the e-wallet application to their PC when they register with the broker. The content display time is slower than the server-side e-wallet and cookie-based client-side e-wallet due to the communication between vendor application server and customer PC's e-wallet application.

A cookie-based client-side e-wallet system is suitable for customers performing many purchases from a single vendor, and then changing to another vendor. The content display time is faster than a client-side e-wallet but slower than a server-side e-wallet.

- Prototyping the broker and an example of "hard-coded" vendor sites - an on-line newspaper site - for both server-side and client-side e-wallet NetPay systems. The broker implementation provides a database holding customer and vendor accounts, e-coin information, an application server providing business functions, a CORBA interface for vendor application servers to request e-coin touchstone and redeem spent e-coins, and a JSP-implemented HTML interface for customers. The "hard-coded" vendor implementation provides a JSP-implemented HTML interface enabling customers to access Search and Buy content services, a web server runs JSPs to provide content that needs to be paid to access the customers' e-wallets, an application server providing business logic, a CORBA interface communicating with the broker and other vendor application servers and a relational database storing vendor data and redeem data.

- Implementing several NetPay EJB components. The component-based NetPay vendor system is developed to support much more easily and seamlessly reused vendor server-side NetPay functionality. The reusable NetPay EJB components are implemented and plugged into an existing journal site to enhance it with micro-payment support with minimal or no code changes.

- Designing and running three kinds of evaluation experiments which include usability, performance impact and qualitative assessment for NetPay-enabled e-newspaper prototypes. A usability evaluation surveyed users of the prototype to assess their impressions of the approach when carrying out information purchasing tasks using a micro-payment versus a macro-payment protocol. The client-side and server-side NetPay was more preferred over a subscription-based system for the E-newspaper application domain. A performance evaluation assessed the performance of NetPay-enabled web sites to determine the overhead of the micro-payment extensions made to the software, particularly in regard to user response time. Server-side NetPay only takes 14ms for e-coin debiting per article, but the client-side e-wallet still incurs considerable response time delay due to the additional overhead of the vendor connecting to a customer PC. A qualitative evaluation assessed factors such as customer effort in using a NetPay-enhanced web site from a customer's perspective, along with the cost/benefit of the system for customers, vendors and brokers.

8.2 Conclusions

In this thesis, some related existing works of macro-payment and micro-payment systems were studied and the advantages and disadvantages of these systems were investigated. A new protocol was proposed called NetPay which is suitable for micro-payments in distribute systems on the WWW. The protocol is "cheap" since it just involves one or no public-key operations per purchase. NetPay is very suitable for a customer performing many low-value purchases from a vendor, then changing to purchasing from another vendor.

The NetPay protocol is based on the PayWord protocol [72]. The PayWord protocol allows a customer to generate a payword chain and spending the paywords to a specific vendor. The payword chain is vendor and customer specific. The payword chain in NetPay protocol is generated by the broker for every customer who spends paywords from one vendor to another without involving the broker, so the payword chain is not vendor-specific. The paywords can be spent with any vendor. NetPay is a basic offline protocol. Millicent [54] is almost an on-line system and places a heavy real-time burden on the broker. A scrip as an electronic currency in Millicent is also vendor-specific.

The NetPay protocol protects the customer's anonymity from vendors and prevents customers from double spending and any internal and external adversaries from forging. In Mpay [40], customer's anonymity is not supported and customers can pay nothing to access services for a full day.

The system requirments for a NetPay-based micro-payment e-commerce system were identified with an on-line newspaper scenario, where the vendor wants to charge on a per-news article basis. Our NetPay prototype systems provide HTML-based thin-client user interfaces. Customers are allowed to access the NetPay facilities with HTML browsers. Micro-payment system can be used to buy not only just news articles but also other goods e.g. online-music where customers can download a single song by paying 50 cents or more and a multitude of game sites where customers can play an on-line game by spending small amounts of money. Our NetPay system should meet the requirements of these systems.

We chose to use a Web-based multi-tier thin-client architecture that meets both NetPay broker and NetPay-based vendor systems requirements and constrains to implement the NetPay system. This allows for a very wide range of customers using standard web browser software, without the need for separate installation of browsing and micro-payment clients. It also improves scalability of the system since the application servers can be deployed on many machines and the database only requires connections from a smaller number of application servers.

We designed server-side and client-side e-wallets for the NetPay system using a CORBA-based approch. In a server-side e-wallet NetPay system, we designed that the broker application server provides a set of CORBA interfaces with which the vendor application servers communicate to request an e-wallet location or to get an e-wallet. The vendor application servers also provide a CORBA interface in order for other vendor application servers to get the e-wallet. In a client-side e-wallet NetPay system, a touchstone and an index (T&I) of a customer's e-wallet are passed from the broker to each vendor. We designed the broker application server to provide a CORBA interface vendor application servers communicate with to get the T&I to verify e-coins. The vendor application servers also provide a CORBA interface in order for another vendor application server to communication with it to pass the T&I, without use of the broker. The main problem with

this approach is that a vendor system can not get the e-wallet or T&I if a previous vendor system crashed. Prototypes for a NetPay broker and "hard-coded" vendor were developed. This incorporates a broker used to generate, verify and redeem e-coins; a customer e-wallet stored either client or server-side, and vendor application server components. The NetPay architecture provides for both secure and high transaction volume per item by using fast hashing functions to validate e-coin unspent indexes. Using a "hard-coded" style the NetPay vendor systems are easy to implement, but it is hard to add some NetPay functions to an existing system. CORBA-based architecture was found to be difficult to add to existing systems whereas EJB one easier.

We designed and developed several NetPay EJB components that can be seamlessly added to existing J2EE-based web applications. NetPay functionality is embodied in Enterprise JavaBean software components and JSP includes or proxies, allowing the existing application to be easily micro-payment enabled. The NetPay EJBs use a CORBA infrastructure to communicate with customers' client-side e-wallet applications, with a broker server, and with other vendor application servers, whether J2EE-based or not. We have successfully added NetPay components to a separately-developed J2EE-implemented E-journal application to demonstrate our approach's feasibility. An article price enterprise bean is designed to select article price data from the NetPay database. We only generated a simple pricing database which included article ID and the price of the article in our prototype. However there are huge numbers of articles, songs, or games in a real web site. The generation of pricing database for NetPay may need some greater effort for a real web site.

The NetPay user interface facilities are integrated by modifying the existing E-journal example system web pages. An implementation of a component-based NetPay vendor system should consider the possible use of other two integration methods which generate NetPay JSP pages and NetPay proxy JSP pages in order for no code impact to the existing system.

We assessed the server-side and the client-side e-wallet NetPay systems deployed with an on-line newspaper web site and compared these with a macro-payment, subscription-based variant of the web site. We carried out some usability testing via a survey-based approach with representative target users of NetPay. A sample potential user group was surveyed.

Further analysis is needed on whether familiar or unfamiliar users of E-commerce systems will prefer it and whether the overall approach is valuable in terms of customer effort and economic trade-off. A performance impact evaluation was curried out to assess the overhead of the micro-payment extensions made to the software. We also carried out an assessment of customer effort and economic trade-off when using these services and compare the results of this assessment to a survey of customers using each system.

A usability evaluation is good for assessing and understanding customer perceptions of using two kinds of NetPay systems and the advantages and disadvantages they saw with the systems, and to gain an understanding of the usefulness of the approach for information vendors. The article contents at different newspaper sites were easy to read without log-in in client-side NetPay system, but this approach incurred an extra delay in page display. The server-side NetPay system allowed users to read articles anywhere in the world, but customers needed to remember e-coin IDs. The usability evaluation involves only a small user base.

The idea of the impact performance evaluation in the vendor system was to compare non-NetPay system with NetPay-enabled systems for a response delay time. The results of modified NetPay prototypes are that the subscription-based system takes 16ms and the server-side NetPay takes 30ms for a response delay per article. However these tests run with only one customer connection to the vendor server at a time. The case study using Argo/MTE [13] showed that the response delay time is reasonable in the performance of the server-side NetPay system when simulated 10 concurrent customers buying articles at the same time. Performance of NetPay prototype under heavy loading was found to be adequate.

In the qualitative evaluation, a macro-payment approach is more beneficial for the customer who reads a large portion of the on-line newspaper articles. However, the micro-payment approach wins out when the customer reads a small portion of the articles, articles are low-priced and if the customer reads articles from multiple newspapers and can use their e-coins across any of these vendors. A key outstanding challenge with micro-payment systems is being able to spend e-coins at a wide range of vendors.

8.3 Future Work

This thesis has systemically described the protocol, architecture, design and implementation of NetPay micro-payment systems. NetPay is superior to other existing micro-payment protocols on several key issues, making it cheap, safe and practical for many low-cost, high-volume transactions. However, there is still a lot of research needed in the future.

We have only prototyped the protocol and investigated its feasibilities for some cases. We need to find the ways to overcome some technical problems occurred in the applications of the NetPay systems such as if vendor servers crash.

In the component-based NetPay vendor system, we can set pricing of contents with existing applications in different ways. For example, the pricing of an article content in a newspaper site can be set by sections e.g. local and world news articles cost 10cs each, sport articles cost 20cs each and so on. In a music site, we can charge songs by classification of songs e.g. popular top ten songs cost $1 for each download, other songs for adults cost 50cs, and the songs for children cost 20cs, and so on. Besides the works improving and implementations of the current NetPay systems, we have identified the following possible future research.

Some currently available micro-payment protocols are not only specifically designed for selling information goods on the Internet, but they can also be used for wireless communications [76, 28, 90]. The SVP-based micro-payment scheme that uses tamper-resistant devices was proposed by DongGook Park for wireless communications [28]. The scheme aims to avoid customers and vendors executing the three-way challenge-response protocol for every micro-payment. This can be an important issue for mobile communications where call charges are still large in comparison with Internet based communications. It also reduces delay and removes the possibility of incomplete payment protocols due to communications failures. Our NetPay server-side e-wallet could be used to provide a similar capability.

With the growth of mobile computing technologies, the popularity of mobile devices (e.g. mobile phone, PDAs) has increased over past a few years [89], different software applications can be deployed on these mobile terminals and can communicate with other

applications or information systems (e.g. Internet) through a wireless network. In general, WAP (Wireless Application Protocol)/WML (Wireless Markup Language) or similar technologies are adopted to provide thin-client solutions. We need to investigate approaches to using NetPay for mobile information content micro-payment applications with a server-side e-wallet storage by the mobile device. A NetPay-enabled application needs to provide HTML (web browser) and WML (mobile) user interfaces and support multiple input devices. Customers are allowed to access the same NetPay facilities by using different platforms.

As we mentioned in Chapter 6, NetPay EJB components were designed, implemented and plugged into existing J2EE-based web applications, but the systems still need some manual work to develop a NetPay-enabled application. We need to focus on the development of tools to allow existing component-based applications to be NetPay-enabled without any manual component programming, deployment and configuration. We also need to apply these to experimenting with adding NetPay to other 3rd party J2EE web applications and carry out a performance evaluation to assess a NetPay-enabled vendor system would be viable for large transaction loading. The results will be analysed to determine if the performance overhead of NetPay micro-payment would be acceptable to information vendors.

There is not currently a way for some vendors who only want to use NetPay facilities temporarily. A portal infrastructure [67] could be designed using web services that will allow a NetPay-enabled vendor to act as a purchasing portal to non-NetPay supporting vendors by redirecting page accesses to these vendors and charging the customers e-coins in the process. A portal infrastructure provides the members of a portal a single place to access NetPay facilities that are used to charge for contents. It also represents the merger of Internet technology with a vendor's internal systems and content in such a way as to give the customers a single point of access to all of the information they need. This approach will allow for dynamic registration of vendors and support cross-vendor product searching. A customer registers and buys e-coins with a NetPay broker, than spends the e-coins through a NetPay portal system to a non-NetPay vendor site for purchasing information goods. The NetPay portal system redeems the e-coins with the broker and may charge 10% or more for the NetPay services, to pay for these. Customers could purchase information

goods with different vendors in the portal system without transferring his/her e-wallet or T&I from a broker to vendors.

Publications during PhD study

Book Chapter

1. Xiaoling Dai and John Grundy, *Customer Perception of a Thin-client Micro-payment System Issues and experiences,* Advanced Topics in End User Computing, Volume III, 2003. M. Adam Mahmood et al eds, to appear.

Refereed Articles:

2. Xiaoling Dai and John Grundy, *Architecture for a Component-based, Plug-in Micro-payment System,* In Proceedings of APWEB 2003, Lecture Notes in Computer Science, Springer-Verlag, pp. 251 – 262.
   http://link.springer.de/link/service/series/0558/tocs/t2642.htm

3. Xiaoling Dai and John Grundy, *Customer Perception of a Thin-client Micro-payment System Issues and experiences*, Journal of End User Computing, 15(4), pp 62-77, (2003).

4. Xiaoling Dai and John Grundy, *Architecture of a Micro-Payment System for Thin-Client Web Applications*, In Proceedings of the 2002 International Conference on Internet Computing, Las Vegas, CSREA Press, June 24-27, pp. 444-450.

5. Xiaoling Dai, John Grundy and Bruce Lo*, Comparing and contrasting micro-payment models for E-commerce systems*, In Proceedings of the International Conferences of Info-tech and Info-net (ICII), China, 2001.

6. Xiaoling Dai and Bruce Lo, *Netpay – An Efficient Protocol for Micropayments on the WWW,* In Proceedings of the Fifth Australian World Wide Web Conference, Ballina, (1999).
   http://ausweb.scu.edu.au/papers/#technical

Reference

1. .NET, The Microsoft .NET homepage: http://www.microsoft.com/net/basics/

2. Aleksy, M., Schader, M. and Tapper, C.: “Interoperability and interchangeability of middleware components in a three-tier CORBA-environment-state of the art”. Proceedings Third International Enterprise Distributed Object Computing, 1999, IEEE CS Press, pp. 204—221.

3. Anderson, R., Manifavas, C. and Sutherland, C.: "NetCard - a Practical Electronic Cash System". In Fourth Cambridge Workshop on Security Protocols. April, 1996.

4. Allen, P.: “Realizing E-Business with Components”. Addison-Wesley, October 2000.

5. Aukia, P. and Lehmann, J-B: “Mechanisms in Electronic Commerce Using Micropayments”. White paper. http://studwww.eurecom.fr/~lehmann/study/

6. Aukia, P. and Oy, N.: “Models of Electronic Commerce”, White Paper, 1995. http://www.tcm.hut.fi/Opinnot/Tik-110.501/1995/commerce.html

7. Bichler, M., Segev, A., Zhao, J.L.: “Component-based E-Commerce: Assessment of Current Practices and Future Directions”. SIGMOD Record 27(4)(1998), pp.7—14.

8. Blankenhorn, D.: “Charging for Content”. E-commerce times. 2001. http://www.ecommercetimes.com/perl/story/306.html.

9. Boly, J. P., Bosselaers, A., Cramer, R., et al.: “The ESPRIT Project CAFÉ, High Security Digital Payment Systems”, Third European Symposium on Research in Computer Security, LNCS 875, Springer-Verlag, Berlin 1994, pp. 217-230.

10. Booch, b., Rumbaugh, J. and Jacobson, I.: “The Unified Modeling Language User Guide”. Addison-Wesley, 1998.

11. Brown, Alan W.:"Large-scale, Component-based Development", Upper Saddle River, NJ: Prentice Hall PTR, 2000.

12. Brown, E.: "Micro-payment schemes promise to make the Web profitable – one penny at a time" 1999. http://newmedia.com/newmedia/97/08/fea/micropayments_small_change.html

13. Cai, Y., Grundy, J.C., Hosking, J.G., Dai, X. Software Architecture Modelling and Performance Analysis with Argo/MTE, In Proceedings of the 2004 Conference on Software Engineering and Knowledge Engineering, Baniff, Canada, June 20-24 2004.

14. Champeaux, D. D., Lea, D. Object-oriented System Development, Chapter 2: Introduction to Analysis, Addison-Wesley, 1993.

15. Chan, S.: "Introduction to Electronic Data Interchange (EDI)", 1997. http://home.hkstar.com/~alanchan/papers/edi/#introduction

16. Chaum, D.: "DigiCash". 1995. http://www.digicash.com/

17. Chaum, D.: "Blind Signatures for Untraceable Payments". In Advances in Cryptology: Proceedings of Crypto'82, Plenum Press, 1983, pp. 199 – 203.

18. Cholesterol Control "Email Credit Card Order Form". http://www.cholesterolcheck.org/emailcreditcardform.htm

19. Chong, N.S.T. and Sakauchi, M.: "e-CoBrowse: co-navigating the Web with chat-pointers and add-ins - problems and promises". Parallel and Distributed Computing and Systems 2(2000), pp. 803—808.

20. Cox, B., Tygar, J. D. and Sirbu, M.: "NetBill Security and Transaction Protocol", The First USENIX Workshop on Electronic Commerce, New York, 1995.

21. Cybercash: http://www.cybercash.com/ and http://www.verisign.com/products/payment.html

22. Dai, X. and Lo, B.: "NetPay – An Efficient Protocol for Micro-payments on the WWW". Fifth Australian World Wide Web Conference, Australia, 1999.

23. Dai, X., Grundy, J. and Lo, B.: "Comparing and contrasting micro-payment models for E-commerce systems", Proceedings of International Conferences of Info-tech and Info-net (ICII), China, 2001.

24. Dai, X. and Grundy, J.: "Architecture of a Micro-Payment System for Thin-Client Web Applications". In Proceedings of the 2002 International Conference on Internet Computing, Las Vegas, CSREA Press, June 24-27, pp. 444—450

25. Dai X. and Grundy J.: "Architecture for a Component-based, Plug-in Micro-payment System", In Proceedings of the Fifth Asia Pacific Web Conference, Lecture Notes in Computer Science (LNCS 2642), Springer-Verlag, April 2003, pp. 251 – 262. http://link.springer.de/link/service/series/0558/tocs/t2642.htm

26. Dai X. and Grundy J.: "Customer Perception of a Thin-client Micro-payment System Issues and experiences", Journal of End User Computing, 15(4)(2003), to appear.

27. Domingo-Ferrer, J. and Herrera-Joancomarti, J.: "Spending programs: a tool for flexible micro-payments". Information Security. Second International Workshop, ISW'99. Lecture Notes in Computer Science, Vol.1729. Springer-Verlag, Berlin, Germany, 1999, pp. 1—13.

28. DongGook, P., Boyd, C. and Dawson, E.: "Micro-payments for wireless communications". Information Security and Cryptology - ICISC 2000. Third International Conference. Proceedings (Lecture Notes in Computer Science Vol.2015). Springer-Verlag, Berlin, Germany, 2001, pp. 192—205.

29. Ediberidze, A., Nikolashvili, M. and Abuashvili, N.: "Design of electronic payment systems for using into Internet". EUROMEDIA '99, pp.247-249.

30. Farley, J.: “Microsoft .NET vs. J2EE: How Do They Stack Up?”, 2000. http://java.oreilly.com/news/farley_0800.html

31. Fingar, P.: “Component-Based Frameworks for E-Commerce”. Communications of the ACM, 2000.

32. Fowler, M. and Scott, K.:”UML Distilled”. Addison-Wesley, 1996.

33. Furche, A. and Wrightson, G.: “SubScrip – An efficient protocol for pay-per-view payments on the Internet”. The 5th Annual International Conference on Computer Communications and Networks, USA, 1996.

34. Gabber, E. and Silberschatz, A.: "Agora: A Minimal Distributed Protocol for Electronic Commerce", Proceedings of the Second USENIX Workshop on Electronic Commerce, *O*akland, California, November 18-21, 1996, pp. 223-232.

35. Gabber, E. and Silberschatz, A.: "Micro Payment Transfer Protocol (MPTP) Version 0.1". W3C Working Draft, 1995. http://www.w3.org/pub/WWW/TR/WD-mptp.

36. GE Information System, Online EDI Service. 1997. http://www.getradeweb.com

37. Grundy, J.C., Cai, Y. and Liu, A. Generation of Distributed System Test-beds from High-level Software Architecture Descriptions, In Proceedings of the 16th International Conference on Automated Software EngineeringSan Diego, 26-29 Nov 2001, IEEE CS Press, pp. 193—200

38. Grundy, J.C., Wang, X. and Hosking, J.G.: “Building Multi-device, Component-based, Thin-client Groupware: Issues and Experiences”. In Proceedings of the 3rd Australasian User Interface Conference, Melbourne, Australia, 2002, pp. 28—30.

39. Hauser, R., Steiner, M. and Waidner, M.: "Micro-Payments based on iKp". 1996. http://www.zurich.ibm.com/iKP_references.html.

40. Herzberg, A. and Yochai, H.: “Mini-pay: Charging per Click on the Web”. 1996. http://www.ibm.net.il/ibm_il/int-lab/mpay

41. Herzberg, A.: “Safeguarding Digital Library Contents - Charging for Online Content”. D-Lib Magazine (1998), ISSN 1082-9873.

42. Hwang, M-S., Lin, I-C. and Li, L-H.: “A simple micro-payment scheme”. Journal of Systems & Software, 55(3)(2001), pp. 221—229.

43. ISO 7498-2: “Information Processing Systems – Part2: Security Architecture”. International Organization for Standardization, 1989.

44. Java 2 Platform, Enterprise Edition (J2EE). The J2EE homepage: http://java.sun.com/j2ee

45. Java Servlet Technology “The Power Behind the Server”, http://java.sun.com/products/servlet/index.html

46. Ji, D-Y. and Wang, Y-M.: “A micro-payment protocol based on PayWord”. Acta Electronica Sinica, 30(2)(2002), pp. 301—303.

47. Kirkby, P.: “Business models and system architectures for future QoS guaranteed Internet services”. IEE Colloquium on Charging for ATM, IEE 1997.

48. Kleiner, K. “Banking on electronic money”. New Scientist. 146, 1972, pp 26, 1995.

49. Kytöjoki, J. and Karpijoki, K.: “Micropayments – Requirements and Solutions”. http://www.tml.hut.fi/Opinnot/Tik-110.5...paper/micropayments/micropayments.html

50. Kern EM, Hahn A, Benger A. **Peer**-to-**peer** process integration in virtual engineering organizations. Processes and Foundations for Virtual Organizations. IFIP TC5/ WG5.5 Fourth Working Conference on Virtual Enterprises (PRO-VE'03). Kluwer Academic Publishers. 2003, pp.433-40. Norwell, MA, USA.

51. Library ClipArt Site: http://www.libraryclipart.com/news.html

52. Lipton, R. J. and Ostrovsky, R.: "Micro-Payment via Efficient Coin-Flipping". Proceedings of Financial Cryptography'98, LNCS Series 1465.

53. Lowen Color Graphics Site: http://www.lowencg.com/products.html

54. Manasse, M.: "The Millicent Protocols for Electronic Commerce". First USENIX Workshop on Electronic Commerce. New York, 1995.

55. McGarvey, R.: "Micro-payments enable teensy content purchases". Econtent, 24(1) (2001), pp. 18—21.

56. Menezes, A. J., Oorschot , P. C. and Vanstone, S. A.: "Handbook of Applied Cryptography". New York, 1997.

57. MP3 Web Site: http://www.mp3.com

58. Mondex International Ltd. Mondex electronic cash. http://www.mondex.com/.

59. National Institute of Standards and Technology (NIST). Federal Information Processing standard (FIPS) Publication 46-3: Data Encryption Standard (DES), October 1999. http://csrc.nist.gov/fips/fips46-3.pdf

60. Neilsen, J.: "The Usability Engineering Lifecycle". COMPUTER 25 (3)(1992), pp. 12—24.

61. Neilsen, J.: "Usability Engineering". Boston: Academic Press, 1993.

62. Neumann, B. and Medvinsky, G.: "Requirements for Network Payment: the NetCheque Perspective", Proceedings of IEEE Compcon'95, San Fransisco, March 1995.

63. O'Mahony, D., Peirce, M. & Tewari, H., “Electronic Payment Systems for E-Commerce”, 2nd Edition, Artech House, Norwood, MA., August 2001, pp. 342.

64. OMG’s CORBA, http://www.corba.org/

65. Peirce, M. and O'Mahony, D., Micropayments for Mobile Networks, in Proceedings of European Wireless '99, Munich, October 6th-8th, 1999, pp 199-204.

66. Pilioura, T.: “Electronic Payment Systems on Open Computer Networks: A Survey” Working paper. 1996.

67. Portals Community Homepage http://www.portalscommunity.com/library/

68. Posman, A.: “Would You Pay for Google?”, 2002. http://www.clickz.com/media/agency_start/article.php/1013901

69. Preece, J.: “A Guide to Usability: human factors in computing”. Addison-Wesley, 1993.

70. Rivest, R., Shamir, A. and Adleman, L.: “A Method for Ontaining Digital Signatures and Public-Key Cryptosystems.” Communication of ACM 21(2) (1978), pp. 120—126.

71. Rivest, R.: "The MD5 Message-Digest Algorithm". RFC 1321, Internet Activities Board, 1992.

72. Rivest, R. and Shamir, A.: “PayWord and MicroMint: Two Simple Micro-payment Schemes”. Proceedings of 1996 International Workshop on Security Protocols, Lecture Notes in Computer Science, Vol. 1189. Springer, 1997, pp. 69—87.

73. Roman, E.: “Mastering Enterprise Java Beans”, New York: John Wiley & Sons, Inc., 1999.

74. Rubin, J.: “Handbook of Usability Testing: how to plan, design, and conduct effective tests”, New York: Wiley, 1994.

75. Ryley, S.: “Corporate portal development: a practical approach ensures real business benefits”. Business Information Review 18(2)(2001), pp. 28—34.

76. Sangjin, K., Heekuck, O.: “An atomic micro-payment system for a mobile computing environment”. IEICE Transactions on Information & Systems. E84-D(6) (2001), pp. 709—716.

77. Schmidt, C. and Muller, R.: “A Framework for Micropayment Evaluation”. 1997. http://www.wiwi.hu-berlin.de/IMI/micropayments.html

78. Schneier, B.: "Applied Cryptography”. Second Edition, New York, 1996

79. Sessions, R.: “J2EE Versus .NET; The Latest Benchmark”, 2002. http://www.objectwatch.com/issue_42.htm

80. Sessions, R.: “Java 2 Enterprise Edition (J2EE) versus .NET Two Visions for eBusiness”, ObjectWatch, Inc. 2001. http://www.objectwatch.com/_Toc511347196

81. Siegel, J. : “Corba 3 Fundamentals and Programming”, New York: John Wiley & Sons, Inc., 2000.

82. Smalley, e. and Patch, K. : “Drop a dime on online Micro-payments were declared DOA a few years ago, but advances in smart cards and software may revive the virtual penny”, InfoWord, Novemenber, 1998, pp.71—75.

83. Sommerville, I.: “Software Engineering”, 5th Edition, Addison-Wesley, 1996.

84. Stern, J., Vaudenay, S.: “SVP: a Flexible Micro-payment Scheme”. Financial Crypto ’97, Springer-Verlag (1997), 161—171.

85. Sun Enterprise JavaBeans, http://java.sun.com/products/ejb/

86. Sun home, http://java.sun.com/products/

87. Sun Java Server Pages, http://java.sun.com/products/jsp/

88. Tang, L.: “A Set of Protocols for Micro-payment in Distributed Systems”. First USENIX Workshop on Electronic Commerce. 1995.

89. Tewari, H. & O'Mahony, D., Real-Time Payments for Mobile IP, IEEE Communications, 41(2)(2003), pp 126-136.

90. Tewari, H. & O'Mahony, D., Multiparty Micropayments for Ad Hoc Networks, in Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC) 2003, New Orleans, Louisiana, 16-20 March 2003.

91. Valesky, T.: “Enterprise JavaBeans: developing Component-based Distributed Applications”, Addison-Wesley, 1999.

92. Vawter, C. and Roman, E.: “J2EE vs. Microsoft.NET” http://www.theserverside.com/resources/article.jsp?l=J2EE-vs-DOTNET

93. Visa&MasterCard: "Secure Electronic Transactions in Visa and MasterCard", 1996. www.bofa.com/spare_change/

94. Vogal, A.: “CORBA and Enterprise Java Beans-based Electronic Commerce”. In International Workshop on Component-based Electronic Commerce, UC Berkeley, 1998.

95. Wang, J.: “CTFS: a new lightweight, cooperative temporary file system for cluster-based Web servers” Proceedings. IEEE International Conference on Cluster Computing. IEEE Comput. Soc. 2003, pp.316-23. Los Alamitos, CA, USA

96. Yen, S-M.: “PayFair: a prepaid internet ensuring customer fairness micro-payment scheme”. IEE Proceedings-E Computers & Digital Techniques, 148(6)(2001), pp.207—213.

Appendix A: Usability Testing Information

A.1 Usability Testing Questionnaire

The purpose of this questionnaire is to help us to gain feedback about using these payment systems to purchase on-line information goods. This feedback will help us to improve our systems and ensure these systems provide the people who will be using them with sufficient and efficient supports for their tasks.

All the information you provide is confidential. Your name is not stored with this questionnaire, and the information you provide will not be used for any other purpose.

A.1.1 Pre-test Questionnaire

1. **Personal information**

   Gender □ Male □ Female

   Age □ Under 18 □ 18-35 □ 36-45 □ 46-55 □ over55

   Education Secondary □ Tertiary □ Postgraduate □

   Major □ Computer Science □ Engineering
   □ Business & Economics □ Other _____

   Job Title_____

2. **How often do you use Internet?**

   Daily □ Weekly □ Monthly □ Occasionally □

3. **Have you ever spent money on the Internet? If yes, please specify what kinds of goods you purchase from Internet. (e.g. information goods or others).**

   No □
   Yes □ _____

4. **Have you read any on-line newspaper? If yes, specify which newspapers you read.**

   No □
   Yes □ _____

   Where do you prefer to read? Home □ Anywhere □

A.1.2 Post-test Questionnaire

- **Subscription Macro-payment System**

1. Please rate how ease or difficulty to use the system with each of the following statements by ticking the item that best expresses your opinion.

Statement	Very Difficulty				Very Ease
a. Subscribe with a site	☐5	☐4	☐3	☐2	☐1
b. Read articles	☐5	☐4	☐3	☐2	☐1
c. Change newspaper site	☐5	☐4	☐3	☐2	☐1
d. Subscription expired	☐1	☐5	☐4	☐3	☐2

2. This system had all the facilities you needed to effectively support your task(s).

Strongly Disagree Strongly Agree
4 ☐3 ☐2 ☐1 ☐5 ☐

3. The payment system worked very well with the on-line Newspaper system.

Strongly Disagree Strongly Agree
☐5 ☐4 ☐3 ☐2 ☐1

4. Was it easy to move around different newspapers using the system?

Very difficult Very easy
☐5 ☐4 ☐3 ☐2 ☐1

5. Do you satisfy the speed of article content loading in this system?

Very unsatisfied Very satisfied
☐5 ☐4 ☐3 ☐2 ☐1

- **<u>Client-NetPay Micro-payment System</u>**

1. **Please rate how ease or difficulty to use the system with each of the following statements by ticking the item that best expresses your opinion.**

Statement	**Very Difficulty**				**Very Ease**
a. Register with broker	˚4	˚3	˚2	˚1	˚5
b. Buy e-coins	˚5	˚4	˚3	˚2	˚1
c. Read articles	˚5	˚4	˚3	˚2	˚1
d. Change newspaper site	˚5	˚4	˚3	˚2	˚1
e. E-coin run out	˚5	˚4	˚3	˚2	˚1

2. **This system had all the facilities you needed to effectively support your task(s).**

Strongly Disagree Strongly Agree
˚5 ˚4 ˚3 ˚2 ˚1

3. **The payment system worked very well with the on-line Newspaper system.**

Strongly Disagree Strongly Agree
˚3 ˚2 ˚1 ˚5 ˚4

4. **Was it easy to move around different newspapers using the system?**

Very difficult Very easy
˚5 ˚4 ˚3 ˚2 ˚1

5. **Do you satisfy the speed of article content loading in this system?**

Very unsatisfied Very satisfied
˚5 ˚4 ˚3 ˚2 ˚1

- **Server-NetPay Micro-payment System**

1. **Please rate how ease or difficulty to use the system with each of the following statements by ticking the item that best expresses your opinion.**

Statement	Very Difficulty				Very Ease
a. Register with broker	ˆ5	ˆ4	ˆ3	ˆ2	ˆ1
b. Buy e-coins	ˆ5	ˆ4	ˆ3	ˆ2	ˆ1
c. Read articles	ˆ5	ˆ4	ˆ3	ˆ2	ˆ1
d. Change newspaper site	ˆ4	ˆ3	ˆ2	ˆ1	ˆ5
e. E-coin run out	ˆ5	ˆ4	ˆ3	ˆ2	ˆ1

6. **This system had all the facilities you needed to effectively support your task(s).**

Strongly Disagree Strongly Agree
ˆ5 ˆ4 ˆ3 ˆ2 ˆ1

7. **The payment system worked very well with the on-line Newspaper system.**

Strongly Disagree Strongly Agree
3 ˆ2 ˆ1 ˆ5 ˆ4 ˆ

8. **Was it easy to move around different newspapers using the system?**

Very difficult Very easy
ˆ5 ˆ4 ˆ3 ˆ2 ˆ1

9. **Do you satisfy the speed of article content loading in this system?**

Very unsatisfied Very satisfied
ˆ5 ˆ4 ˆ3 ˆ2 ˆ1

- **Comparison Questionnaires**

After using three payment systems to read articles, please answer the following questions by ticking the item that best expresses your opinion.

1. **Do you consider the systems suitable for wide use?**

   No ☐ Yes ☐

   If yes, which one do you prefer to use.

   Subscription-based ☐ Client-NetPay ☐ Server-NetPay ☐

2. **If you have 20 dollars, you want to purchase useful information on the Internet. Which payment system would you prefer to spend with?**

   Subscription-based ☐ Client-NetPay ☐ Server-NetPay ☐

3. **If you could, would you spend 6 dollars to subscribe the whole newspaper for one week?**

   Yes ☐ No ☐

   If No, would you prefer to spend a few cents to buy an article that you are very interested in?

   Yes ☐ No ☐

   If yes, which one do you prefer to use?

   Client-NetPay ☐ Server-NetPay ☐

4. **Do you have any other comments about the systems?**

   _____

   _____

   _____

A.2 Usability Testing Tasks

The tasks are designed to simulate using three different payment methods to purchase on line newspaper articles. Also, the tasks are developed to allow each user to use the same functions for each payment system including:

- Subscribe with the newspaper site or register and buy e-coins with a broker
- Read 3 articles on newspaper1 site

- Change to newspaper2 site and read 3 articles
- If subscription or e-coins run out, the user must renew it

A.2.1 Tasks for Subscription-based Payment

Assume that you are a reader. You want to purchase some articles using a subscription-based payment system on on-line newspaper site. The things you need to do are:

1. Subscribe to the newspaper1 site by supplying your information, which includes name, password, email address and payment details (credit card etc). The newspaper system makes an electronic debit ($6 weekly) to pay for your subscription and display customer ID1 that you should remember.

2. Login to the newspaper1 site with a customer ID1 and password. The newspaper1 system provides you access to the current edition. You can select to read 3 articles.

3. Change to the newspaper2 site. You need to subscribe to the newspaper2 site and remember customer ID2.

4. Login to the newspaper2 site with customer ID2 and password. You can select to read 3 articles on this site.

5. If your subscription has expired, you must renew it.

A.2.2 Tasks for Client-side e-wallet

NetPay micro-payment system (Client-NetPay)

Assume that you are a reader. You want to read some articles using a Client-NetPay-based micro-payment system on on-line newspaper site. The things you need to do are:

1. Open broker's web site and register by supplying your details that include name, password, email address and payment details (credit card etc).

2. The system displays your customer ID that you need to remember, then download e-wallet software.

3. Run e-wallet server and client software by clicks the icons.

4. Click Buy E-coins, login to broker site by enter customer ID and password; purchase "E-coins" by entering the amount of e-coins you required e.g. 50c. The broker sends e-coins to your e-wallet.

5. Open newspaper1 site and select to read 3 articles (5-15c) and run e-wallet client program to check the balance.

6. Change to newspaper2 site and select to read 3 articles (5-15c) and check the balance from e-wallet window.

7. If e-coins run out, go to Broker site to buy more.

A.2.3 Tasks for Server-side e-wallet

NetPay micro-payment system (Server-NetPay)

Assume that you are a reader. You want to read some articles using a Server-NetPay-based micro-payment system on on-line newspaper site. The things you need to do are:

1. Open broker's web site and register by supplying your details that include name, password, email address and payment details (credit card etc). The system displays your customer ID that you need to remember.

2. Login to broker site by enter customer ID and password; purchase “E-coins” by entering the amount of e-coins you required e.g. 50c. The system displays your e-coin ID that you have to remember.

3. Login to newspaper1 site by entering e-coin ID and password and select to read 2-3 articles (5-15c), then log out.

4. Login to newspaper2 site by entering e-coin ID and password and select to read 2-3 articles (5-15c), and then log out.

5. If e-coins run out, go to Step 2.

The following screen dumps Fig. A-1 to Fig. A-5 are for both Server-NetPay and Client-NetPay for Customer Registration, buy e-coins and login.

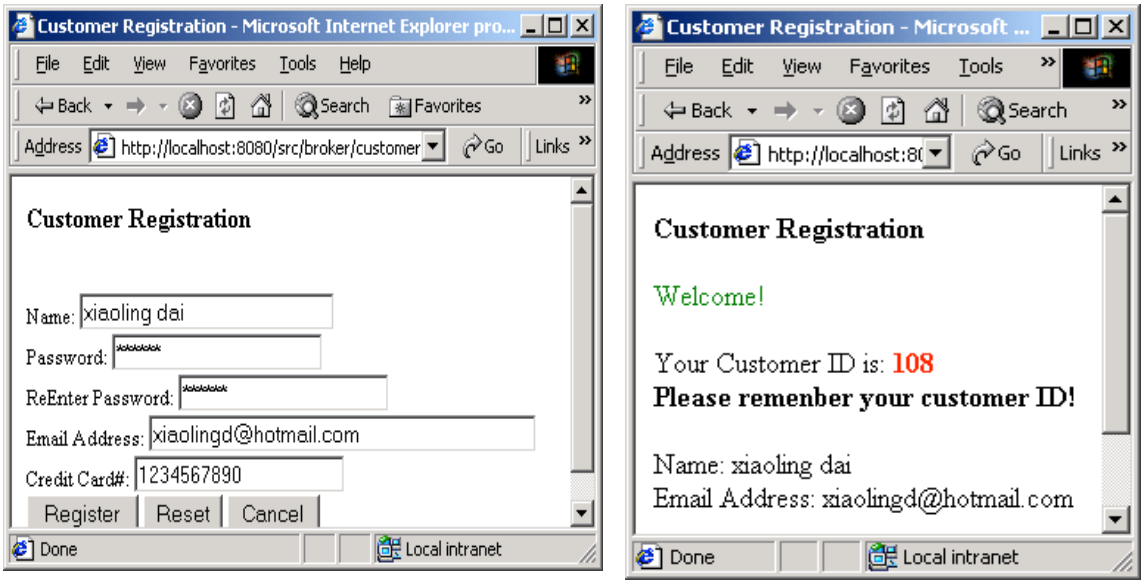


Figure A-1 Xiaoling registers with the broker

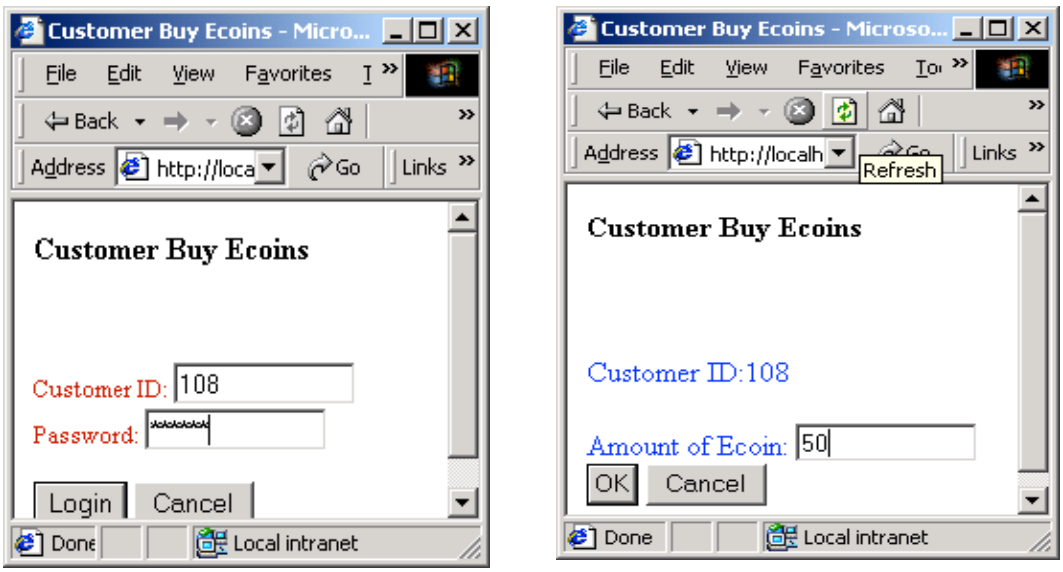


Figure A-2 Buy E-coins' screens are used in Server-NetPay and Client-NetPay

The following screens are used in Server-NetPay system only.

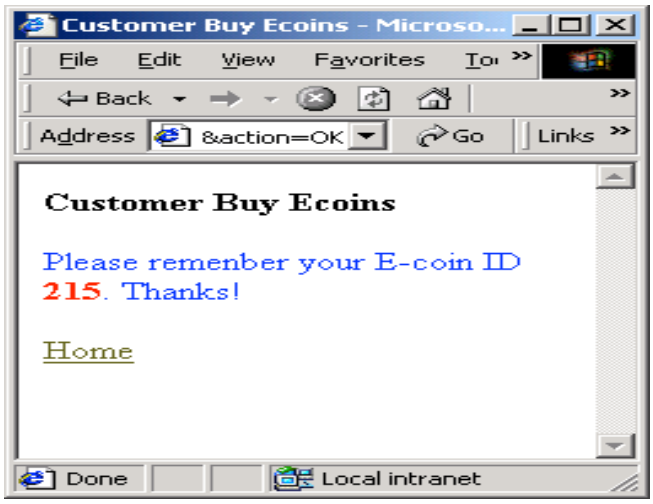


Figure A-3 Example customer buy e-coin with server-side e-wallet NetPay

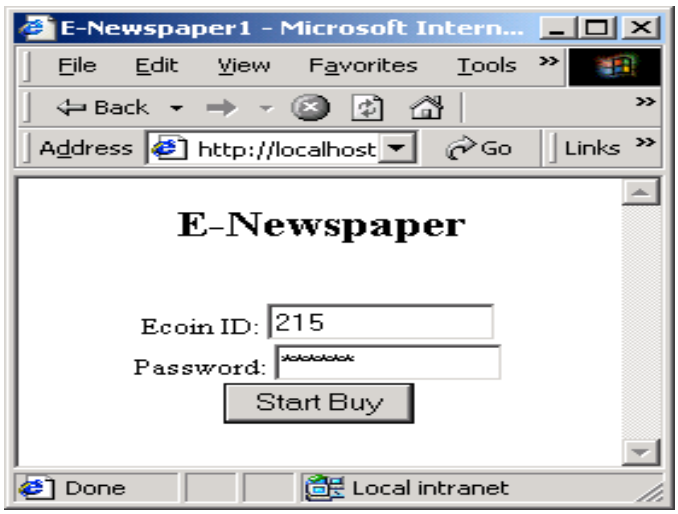


Figure A-4 Example customer login to the e-newspaper site

The following screen is a newspaper site. You can just click to read the articles.

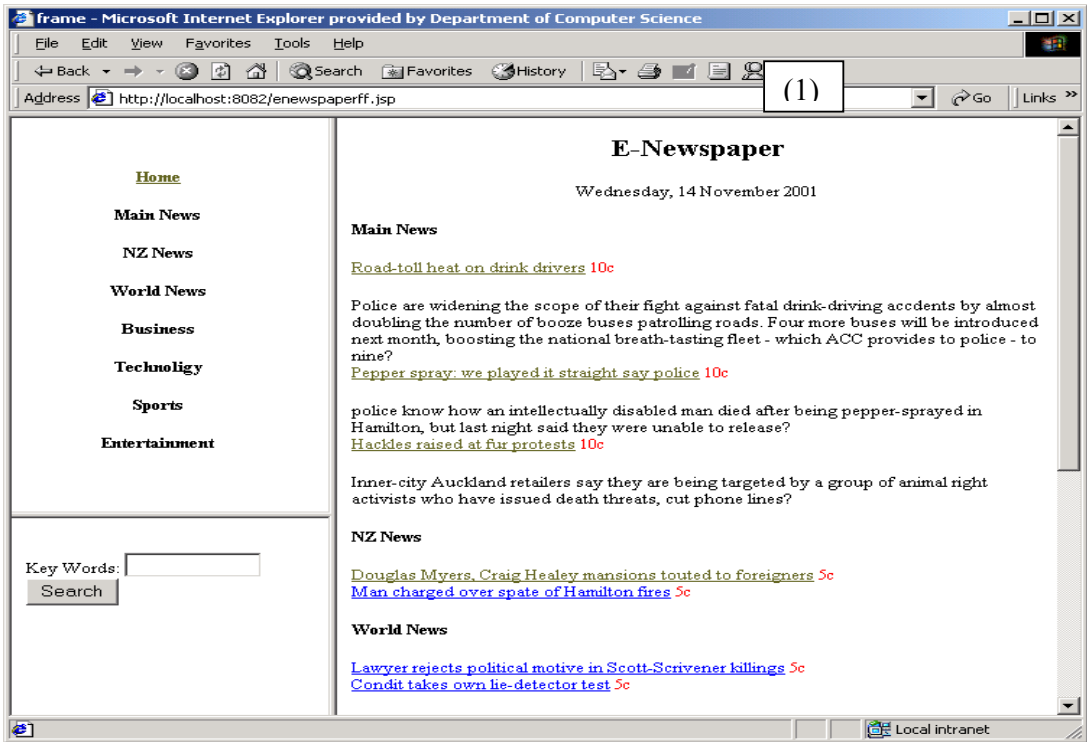


Figure A-5(1) Example e-newspaper web site

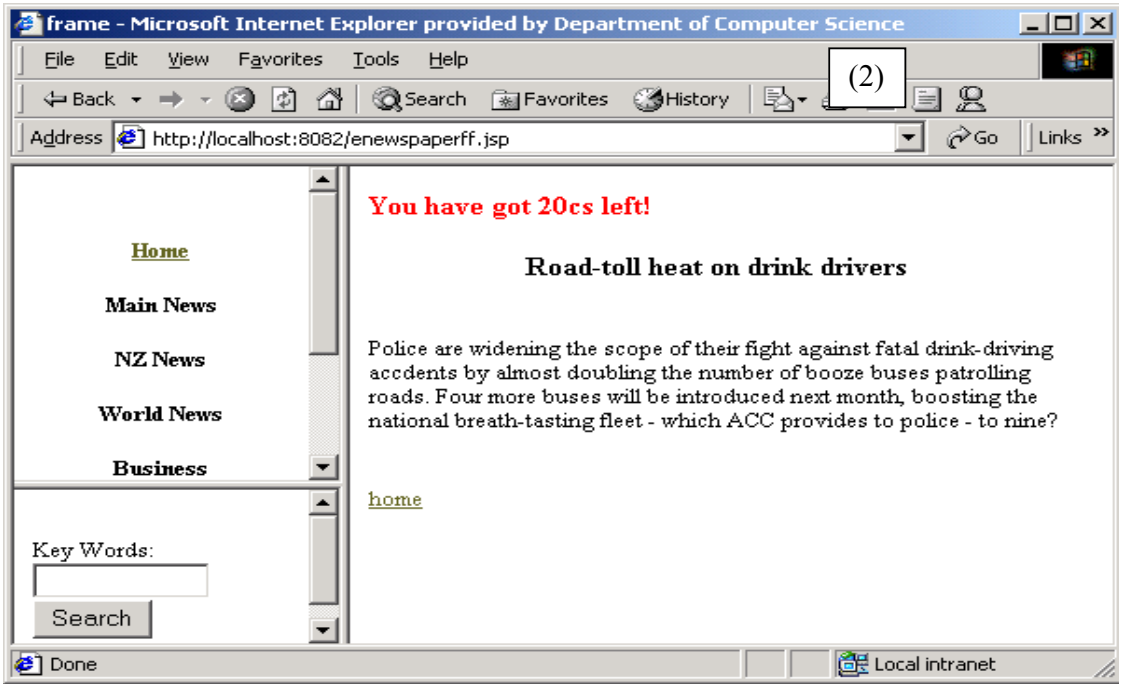


Figure A-5(2) Example of customer spending E-coins at an E-newspaper site

A.3 Summary

We summarize the results received from the participants in this section. Table A-1 to A-3 show the marks of the items in the questionnaires and Table 4 shows the preference.

A.3.1 Three Systems Data

A participant should run three systems which are subscription-based, server-side NetPay and client-side NetPay in the usability testing. In following tables, the marks for ease of use item for each system come from question 1 in the post-test questionnaire, the marks for change site item come from question 4, and marks for speed of a content display item come from question 5.

Table A-1 Subscription Macro-payment System

	Ease of Use	Change Site	Deal With Subscription Expired	Speed of Content Display
Participant1	**5**	**1**	**3**	**5**
Participant2	**5**	**2**	**4**	**5**
Participant3	**4**	**1**	**2**	**5**
Participant4	**5**	**3**	**3**	**4**
Participant5	**4**	**2**	**3**	**5**
Participant6	**4**	**2**	**4**	**5**
Participant7	**5**	**2**	**3**	**4**
Participant8	**4**	**2**	**2**	**5**
Participant9	**4**	**1**	**2**	**5**
Participant10	**5**	**3**	**3**	**5**

Table A-2 Client-side NetPay micro-payment system

	Ease of Use	Change Site	Deal With E-coin Run Out	Speed of Content Display
Participant1	**5**	**5**	**5**	**2**
Participant2	**5**	**5**	**4**	**3**
Participant3	**4**	**5**	**5**	**3**
Participant4	**5**	**5**	**5**	**4**
Participant5	**4**	**5**	**4**	**3**
Participant6	**5**	**5**	**5**	**2**
Participant7	**4**	**4**	**4**	**3**
Participant8	**5**	**5**	**5**	**3**
Participant9	**5**	**5**	**4**	**4**
Participant10	**5**	**4**	**5**	**2**

Table A-3 Server-side NetPay micro-payment system

	Ease of Use	Change Site	Deal With E-coin Run Out	Speed of Content Display
Participant1	**3**	**4**	**5**	**4**
Participant2	**4**	**5**	**4**	**5**
Participant3	**4**	**3**	**4**	**5**
Participant4	**3**	**5**	**5**	**4**
Participant5	**3**	**5**	**5**	**5**
Participant6	**2**	**4**	**5**	**5**
Participant7	**5**	**4**	**4**	**5**
Participant8	**3**	**5**	**5**	**4**
Participant9	**4**	**5**	**4**	**4**
Participant10	**4**	**4**	**4**	**5**

A.3.2 Comparison of Three Systems

Table A-4 gives preference of three systems according to the comparison questionnaire in Section A.1.2. The numbers in Table A-4 indicates that how many participants prefer to use a system.

Table A-4 Preference of three systems

	Widely Use	Prefer to Spend Money with
Subscription-based	1	1
Client-side NetPay	4	5
Server-side NetPay	5	4

Appendix B: Usability Testing Forms

B.1 Consent Form

THIS CONSENT FORM WILL BE HELD FOR A PERIOD OF SIX YEARS

Title: Netpay Electronic Micro-payment System
Researcher: Xiaoling (Sharlene) Dai

I have been given and have understood an explanation of this research project. I have had an opportunity to ask questions and have them answered.

I understand that the usability testing session will not be audio/video taped

I understand that I may withdraw myself or any information traceable to me at any time up to 01/06/2003 (date or stage of study specified by the researcher) without giving a reason.

I, the undersigned, hereby give my permission for the usability testing session.

Signed:
Name:
(please print clearly)
Date:

APPROVED BY THE UNIVERSITY OF AUCKLAND HUMAN SUBJECTS ETHICS COMMITTEE

on ..... for a period of ..... years, from ...../...../..... Reference ...../.....

(This section is to be completed **after** advice of approval has been received from the UAHSEC, and **before** the sheet is given to prospective subjects)

B.2 Application Form

University of Auckland Human Subjects Ethics Committee

RESEARCH PROJECT APPLICATION FORM (2002)

Ref No. .....
(This number will be assigned when the application is receipted)

Submit one unstapled copy of the form and all accompanying documentation to the Research Office. No handwritten forms please.

Note: Applicants may omit Section G and Section H as appropriate.

Please complete this form in reference to the UAHSEC Guidelines (Revised 1999) available on the University of Auckland website under Research at the University.

SECTION A: GENERAL

1. PROJECT TITLE
   Netpay Electronic Micro-payment System

2. APPLICANT/PRINCIPAL INVESTIGATOR
   (Use name of Supervisor if a Masters student, PhD students submit in their own name)

   Name: Prof. John Grundy

   Address: Department of Computer Science, Room 245, Level 2, 70 Symonds St, Auckland

   Email address: john-g@cs.auckland.ac.nz

   Phone number: 3737599-8761

3. NAME OF STUDENT: (*If applicable*)

   Address: Department of Computer Science, Room 211, Level 2, 70 Symonds St, Auckland

   Email address: xdai001@ec.auckland.ac.nz

   Phone number: 3737599-2089

   Name of degree: PhD

4. OTHER INVESTIGATORS:

   Names:

   Organisation:

5. AUTHORISING SIGNATURES:

Principal Investigator/Supervisor: .....Date: ……..

HEAD OF DEPARTMENT: ……..Date: ……..

HOD name printed: Professor John Hosking

SECTION B: PROJECT

Use language which is free from jargon and comprehensible to lay people.

1. AIM OF PROJECT:
State concisely the aim/s of the project. State the specific hypothesis, if any, to be tested.

In this project a prototype of Netpay micro-payment system is developed by using component-based technology. Customers are allowed to use the Netpay system to purchase inexpensive information goods through convention web browser on desktop/laptop, or WML browser on WAP mobile device. There are two kinds of e-wallets which are server-side and client-side in the system.

2. BACKGROUND:
Provide sufficient information to place the project in perspective and to allow the project's significance to be assessed.

With the rapid growth of the Internet into mainstream use has led to the creation of thousands of different web sites providing information service to millions of people around the world. Producing and maintaining a quality web site takes a great deal of time, dedication and money. To help offset the monetary costs, web site producers turn to using their site as a source of revenue through their information service (advertising and subscription). Therefore, a system to handle with such a trade is needed. Due to the trade involves small amount of money, we call the system a micropayment system. We have developed Netpay prototype for an on-line newspaper to support an efficient, secure and anonymous micro-payment system.

3. PROCEDURE:

PROJECT DURATION (***approximate dates***):

From 01/02/2001 to 01/06/2003

(i) State the approach taken to obtaining information and/or testing the hypothesis.

Two prototypes of Netpay micro-payment protocol built in this system will be used for usability testing.

(ii) State **in practical terms** what research procedures will be used, and how information will be gathered and analysed.

The information will be gathered through interviews, questionnaires and observation.

NB: Please state if the research involves potentially hazardous substances, e.g. radioactive materials.

4. Does this research involve the use of human tissue, body fluids, or remains? **Y** ☐ **N** ☐ ✓

If Y, please complete Section G and Section H as appropriate.

5. Does this research include the use of a questionnaire? **Y** ☐ **N** ☐ ✓
If Y, please attach a copy to this application form.

SECTION C: SUBJECTS

(The term 'subjects' is taken to mean participants, clients, informants and patients as well as persons subjected to experimental procedures.)

1. TYPES OF PERSON PARTICIPATING AS SUBJECTS:

Normal Adults Y ☐ N ☐ ✓

Applicant's students Y ☐ N ☐ ✓

Persons 15 and under Y ☐ N ☐ ✓

Persons whose capacity to consent is compromised Y ☐ N ☐ ✓

Prisoners Y ☐ N ☐ ✓

Other Y ☐ N ☐ ✓

(If YES, please explain)

2. a) Who are the subjects? What criteria are to be used for selecting them? State if the subjects perceive themselves to be in any dependant relationship to the researcher (for example, students of the researcher).

Auckland University Students

b) How are the subjects to be identified and recruited?
(If by advertisement attach a copy to this Application Form)

The potential subjects will be those students who have basic knowledge of WWW and will be recruited through verbal invitation.

c) Are there any potential subjects who will be excluded?

No.

d) If so, what are the criteria for exclusion?

e) How many subjects will be selected?

5-15

3. How will information about the project be given to subjects?

(*e.g. in writing, verbally - A copy of information to be given to prospective subjects should be attached to this application.)*

The information will be provided in writing, via Participant Information Form.

4. a) Will the subjects have difficulty giving informed consent on their own behalf?

(Consider physical or mental condition, age, language, legal status, or other barriers.)

Y ☐ N ☑

b) If subjects are not competent to give fully informed consent, who will consent on their behalf?

N/A

5. Will consent be gained in writing? Y ☑ N ☐

(If Y, attach a copy of the Consent Form which will be used.
If N, please explain.)

6. Will the subjects be audio-taped or video-taped ? Y ☐ N ☐ ✓

(If Y, make sure that this is clearly stated in the Participant Information Sheet and in the Consent Form)

7. Will confidentiality of information be preserved? Y ☑ N ☐

(In either case your Participant Information Sheet needs to fully inform your subjects of this)

8. In the final report will there be any possibility that individuals or groups could be identified? Y ☐ N ☑

(*If Y, please explain)*
*(M*ake sure this is clearly explained in the Participant Information Sheet)

SECTION D: OTHER PROJECT DETAILS - (Refer to Guidelines)

1. Where will the project be conducted?

University of Auckland

2. a) Who will actually conduct the study?

The student, Xiaoling (Sharlene) Dai

b) Who will interact with the subjects?

The student, Xiaoling (Sharlene) Dai

3. How much time will subjects have to give to the project?
(Indicate this in the Participant Information Sheet)

30 minutes

4. State the risks and benefits of the proposed research.

The built Netpay micro-payment system can be used to purchase inexpensive information goods on the Internet in the future

There is no risk.

5. Is deception involved at any stage of the research? Y ☐ N ☑

(If Y, please explain)

6. a) Are the subjects likely to experience discomfort (*physical, psychological, social)* or incapacity as a result of the procedures? Y ☐ N ☑
(If Y, please explain)
(Make sure this is clearly explained in the Participant Information Sheet)

b) What qualified personnel will be available to deal with adverse consequences or physical or psychological risks?

7. Will information on the subjects be obtained from third parties? Y ☐ N ☑
(For example – from subject's employer, teacher, doctor etc.
Indicate this in the Participant Information Sheet)

8. Will any identifiable information on the subjects be given to third parties? Y ☐ N ☑

(If Y, please explain)
(Make sure this is clearly explained in the Participant Information Sheet)

9. Provide details of any compensation and where applicable, level of payment to be made to subjects.

SECTION E: RETENTION OF CONSENT FORMS AND DATA

1. The Committee recommends that access to Consent Forms be restricted to the researcher and/or the Principal Investigator. If you wish to do otherwise please explain:

2. *The Committee normally requires that the Consent Forms be stored by the principal investigator, in a locked cabinet preferably on University premises.* If you wish to do otherwise please explain:

3. *The Committee recommends that the Consent Forms be stored separately from the data, and that they be retained for six years*. If you wish to do otherwise please explain:

4. Will data be retained for possible future research use beyond this project? Y ☐ N ☑
(Please explain)
If Yes, ensure this is clearly stated in the Participant Information and Consent Forms.

5. How and when will the data be destroyed?

All hard copy (e.g. questionnaires) will be shredded and all electronic data will be deleted before 01/06/2003.

SECTION F: FUNDING

Are funds being applied for or provided for this project? Y ☐ N ☑

(If Yes complete this section, otherwise proceed to Section G*)*

1. Is this project a UniServices Ltd. project? Y ☐ N ☐ √

(*If* Y, *state the Contract reference number* .....)

2. Will/Has an application for funds to support this project be/ been made to a source external to the University?

Y ☐ N ☑

If Y, state the name of the organisation/s.

3. Explain investigator's financial interest, if any, in the outcome of the project.

Nil.

SECTION G: (omit as appropriate)

IS THIS PROJECT A CLINICAL TRIAL? Y ☐ N ☐

(If Y please attach ACC Form A or B (see Guidelines)

1. Is this project initiated by a Pharmaceutical Company? Y ☐ N ☐

2. Are there other NZ or International Centres involved? Y ☐ N ☐

3. Is there a clear statement about indemnity? Y ☐ N ☐

4. Is Standing Committee on Therapeutic Trials (SCOTT) approval required? Y ☐ N ☐

5. Is National Radiation Laboratory approval required? Y☐ N☐

6. Is Gene Therapy Advisory Committee consultation required? Y☐ N☐

7. Is National Advisory Committee on Assisted Human Reproduction (NACHDSE) approval required? Y☐ N☐

SECTION H: HUMAN REMAINS, TISSUE & BODY FLUIDS (omit as appropriate)

1. How will the material be taken? (*eg, operation*)

Will specimens be taken for possible future use? Y☐ N☐
If yes, please explain.
(Make sure this is clearly stated in Participant Information and Consent Forms.)

2. Is material being recovered at archaeological excavation Y☐ N☐
(If Y - Have the wishes of Iwi and Hapu (*descent groups*), or similar interested persons, or groups, been respected with regard to human remains?) Y☐ N☐

3. a. Where will the material be stored?

b. How long will the material be stored?

4. a. How will the material be disposed of? **(I*f applicable*)**

b. Will the material be disposed of in consultation with the relevant cultural group? Y☐ N☐

5. Is the material being taken at autopsy? Y☐ N☐
(If Y –
a. Provide a copy of the information that will be given to the Transplant Co-ordinator. **(*Attach a separate sheet if necessary)***
b. State the information that the Transplant Co-ordinator will provide to those giving consent. **(*Attach a separate sheet if necessary)***
c. Where will the material be stored?
d. How will the material be disposed of **(*if applicable)*?**
e. Have the wishes of the whanau **(*extended family*)** or similar interested persons, or groups, been respected with regard to the disposal of human remains? Y☐ N☐

6. If blood is being collected -
a. What volume at each collection?
b. How frequent are the collections?
c. Who will collect it?

SECTION I: OTHER INFORMATION

1. The committee treats all applications independently. If you think there is relevant information from past applications or interaction with the Committee, please indicate and append.

2. Have you ever made any other related applications? Y ☐ N ☐ √

(*If yes, give relevant approval reference number/s*)

Declaration: The information supplied above is to the best of my knowledge and belief accurate. I have read the current Guidelines of the University of Auckland Human Subjects Ethics Committee and clearly understand my obligations and the rights of the subjects, particularly in regard to obtaining freely-given informed consent.

Signature of Applicant: .....
(*In the case of student applications the signature should be that of the Supervisor)*

Signature of Student: ..... ***If a student project both the signature of the Supervisor, as the applicant, and the student are required)***

Date: .....

HAVE YOU: - CHECKED ALL DOCUMENTATION FOR SPELLING AND

GRAMMAR AND ATTACHED THE FOLLOWING, (where applicable)

⇒ Copies of surveys and questionnaires
⇒ Copy of advertisement
⇒ List of interview topics
⇒ **PARTICIPANT INFORMATION SHEET (PIS)**
- ☐ On University letterhead
- ☐ Title at the top
- ☐ Indicated for which subjects it is intended
- ☐ Stated the name of the degree in which you are enrolled
- ☐ Included a timeframe for withdrawing data. (The Committee recommends to indicate a specified date if appropriate)
- ☐ Included contact names and addresses, for example, Supervisor, Head of Department and address only of Chair of Ethics Committee e.g. (The Chair, University of Auckland Human Subjects Ethics Committee, University of Auckland, Private Bag, 92019, Auckland, tel. 373-7599, extn. 7830).
- ☐ Ensured that the wording is clear and free from jargon.
- ☐ Included the approval wording at the end of the PIS

For projects in schools
- ☐ PIS for school, for example, Principal and/or Board of Trustees
- ☐ PIS for teachers
- ☐ PIS for student subjects
- ☐ Is the PIS in language suitable to the age of the students?
- ☐ PIS for parents (if students are younger than 16)

For projects in organisations
- ☐ PIS for Management, for example, Chief Executive Officer (where necessary)
- ☐ PIS for Line Managers
- ☐ PIS for employee subjects

⇒ **CONSENT FORM (CF)**
- ☐ Provided on University letterhead
- ☐ Included the title

- Included explicit consent for audio or video taping
- Worded the CF according to the research being conducted, for example, state if it is for an interview, observation, performance measures, test scores etc.
- Included the approval wording at the end of the CF
- Provided the written version of oral consent statement given to subjects (if appropriate)

For projects in schools

- CF/ permission for school, e.g., Principal and/or Board of Trustees
- CF/ permission for teachers
- CF for student subjects
- CF for parents/guardians
- Assent Form if students younger than 16

For projects in organisations

- CF/permission for CEO
- CF/permission for line management
- CF for employee subjects

⇒ **FOR PROJECTS INVOLVING SUBJECTS WHO SPEAK A LANGUAGE OTHER THAN ENGLISH**

- If involving Maori subjects, the PIS and CF to be provided in English and Maori
- If English is not the mother tongue for the subjects it may be appropriate to provide the PIS and CF in the language of the subjects

B.3 Participant Information Sheet

PARTICIPANT INFORMATION SHEET

Title: Netpay Electronic Micro-payment System

To: Subjects

My name is Xiaoling Dai. I am a student at The University of Auckland conducting enrolled in a PhD Degree in the Department of Computer Science. I am conducting this research for the purpose of my thesis on benefits of Netpay Electronic Micro-payment System.

You are invited to participate in my research and I would appreciate any assistance you can offer me. As part of my thesis I am conducting a case study on utilizing the prototypes of Netpay Electronic Micro-payment System I have developed.

I would like to observe you as you interact with these prototypes to achieve a set of pre-defined tasks, as well as ask you to fill out a questionnaire about you experience with these payment systems, but you are under no obligation at all to be interviewed. The assigned

tasks take about half an hour to three quarters of an hour and would be carried out at any suitable time. You can withdraw information any time up to 01 June 2003.

If you do wish to be interviewed please let me know by filling in a Consent Form and sending it to me or phoning me on 3737599-2089 during work hours. All information you provide in an interview is confidential and your name will not be used.

Thank you very much for your time and help in making this study possible. If you have any queries or wish to know more please phone me at the either of the numbers given above or write to me at:

Department of Computer Science
The University of Auckland
Private Bag 92019
Auckland. Tel: 373-7599 extn 2089

My supervisor is:

Professor John Grundy
Department of Computer Science
The University of Auckland
Private Bag 92019
Auckland. Tel. 373-7599 extn 88761

The Head of Department is:

Professor John Hosking
Department of Computer Science
The University of Auckland
Private Bag 92019
Auckland. Tel. 373-7599 extn 88297

For any queries regarding ethical concerns please contact:

The Chair, The University of Auckland Human Subjects Ethics Committee,
The University of Auckland, Research Office - Office of the Vice Chancellor, Private Bag 92019, Auckland. Tel. 373-7999 extn 7830

APPROVED BY THE UNIVERSITY OF AUCKLAND HUMAN SUBJECTS ETHICS COMMITTEE on ..... for a period of ..... years, from ..../..../.... Reference ...../.....

Appendix C: MD5 Implementation

Java technology has several packages that provide classes useful for writing secure applications. In JDK 1.1, three packages, which are java.security, java.security.acl, and java.security interfaces, are included in the *Java Cryptography Architecture* (JAC). The JAC is a framework for providing cryptographic capabilities to Java programs. The java.security package consists mostly of abstract classes and interfaces that encapsulate security concepts such as certificates, keys, message digests (such as MD5, SHA), and signatures.

To implement MD5 algorithm using the security package, we need a multi-step process as follows.

1. Get a seed as a byte[]

```
String seed=String.valueOf(Math.random());
byte[] msg=seed.getBytes();
```

2. Get a *MessageDiagest* for the appropriate algorithm.

```
MessageDigest md=MessageDigest.getInstance("MD5");
or
MessageDigest md=MessageDigest.getInstance("SHA-1");
```

3. Ensure the digest's buffer is empty

```
md.reset();
```

4. Fill the digest's buffer with data to compute a message digest from.

```
md.update(msg);
```

5. Generate the digest.

```
byte[] aMessageDigest=md.digest();
```

6. Convert the digest bytes to a string.

```
Stringcc=convert(aMessageDigest);
```

The following method generates seed, root and e-coin (payword chain) and stores them to a database.

```
public void generatEcoin(int CID, int necoins) throws ServerError

   {   StringBuffer ecoins= new StringBuffer();
       String arrayOfStrings[]=new String[necoins+1];
       String seed=String.valueOf(Math.random());
       byte[] msg=seed.getBytes();
       MessageDigest md=null;
try {

      md=MessageDigest.getInstance("MD5");
       } catch (Exception ex) {
          throw new ServerError("Digest Error: "+ex.toString());
       }

       for(int h=0; h<=necoins;h++){
           md.reset();
           md.update(msg);
           byte[] aMessageDigest=md.digest();
           msg=aMessageDigest;
           arrayOfStrings[h]=convert(aMessageDigest);
           }

       String root=arrayOfStrings[necoins];
        for (int i=necoins-1; i>=0; i--){
            ecoins=ecoins.append(arrayOfStrings[i]);
            }

           String ecoins1=ecoins.toString();
 try {

    MySQLConn.getInstance().execute("INSERT INTO ecoin
    (cid,seed,root,amount,chain) VALUES ("+
      CID+",'"+seed+"','"+root+"','"+necoins+"','"+ecoins1+"')");
      } catch (Exception ex) {
         throw new ServerError("SQL Error: "+ex.toString());
      }
  }
```

The following method verifies e-coins by using MD5, and the integer variable *hash* indicates the number of times perform hash to an e-coin.

```
public String VerifyEcoin(String ecoin,int hash) throws ServerError
 {
   try{
    if (ecoin!=null){
          byte[] w1=convertHex(ecoin);
          MessageDigest md = MessageDigest.getInstance("MD5");
          for (int i=1; i<=hash; i++)
          {md.reset();
```

```
          md.update(w1);
          w1=md.digest();
          }
         return(convertByte(w1));
          }
       return null;
    } catch (Exception ex) {
          throw new ServerError("MessageDigest:MessageDigest "+ex.toString());
      }
    }
```

The following method converts a string of hexadecimal digits into the corresponding byte array by encoding every two hexadecimal digits as a byte.

```
public static byte[] convertHex(String hexstring)
  {
   ByteArrayOutputStream baos=new ByteArrayOutputStream();
   for (int i=0; i<hexstring.length(); i+=2)
   {
    char c1=hexstring.charAt(i);
    if((i+1)>=hexstring.length())
      System.out.println("bad");
    char c2=hexstring.charAt(i+1);
    byte b=0;
    if ((c1>='0') && (c1<='9'))
      b += ((c1 - '0') * 16);
    else if ((c1>='a') && (c1<='f'))
      b += ((c1 - 'a' + 10) * 16);
    else if ((c1>='A') && (c1<='F'))
      b += ((c1 - 'A' + 10) * 16);
    else
      System.out.println("bad c1");

    if ((c2>='0') && (c2<='9'))
      b += (c2 - '0');
    else if ((c2>='a') && (c2<='f'))
      b += (c2 - 'a' + 10);
    else if ((c2>='A') && (c2<='F'))
      b += (c1 - 'A' + 10);
    else
      System.out.println("bad c2");
    baos.write(b);
   }
   return (baos.toByteArray());
  }
```

The following method converts a type array into a printable format containing a String of hexadecimal digit characters (two per type).

```
public static String convertByte(byte bytes[])

  {
  StringBuffer sb = new StringBuffer(bytes.length*2);
  for (int i=0;i<bytes.length; i++){
    sb.append(convertDigit((int) (bytes[i]>>4)));
    sb.append(convertDigit((int) (bytes[i]&0x0f)));
    }
 return (sb.toString());
 }

private static char convertDigit(int value)
  {
   value&=0x0f;
   if(value>=10)
       return((char)(value-10+'a'));
   else
      return ((char)(value+'0'));
  }
}
```