Architecture of a Micro-payment System for Thin-client Web Applications

Xiaoling Dai
Department of Computer Science
University of Auckland
Private Bag 92019, Auckland
New Zealand
xiaolingd@hotmail.com

John Grundy
Department of Electrical and Electronic Engineering
and Department of Computer Science
University of Auckland
Private Bag 92019, Auckland, New Zealand
john-g@cs.auckland.ac.nz

Abstract

Some web-based services need to be charged for on a per-use basis, where each usage may be many thousands or even millions per day. Micro-payment is an approach to charging for web content (typically) for situations with a small cost-per-use/high use-frequency. We describe a prototype architecture for a new micro-payment model, called NetPay. We present an object-oriented design and describe a prototype implementation of NetPay for an on-line newspaper. We report on initial evaluation results deploying our NetPay prototype and outline directions for future research in micro-payment implementations.

Keywords: electronic-commerce micro-payment system, software architecture, electronic wallet

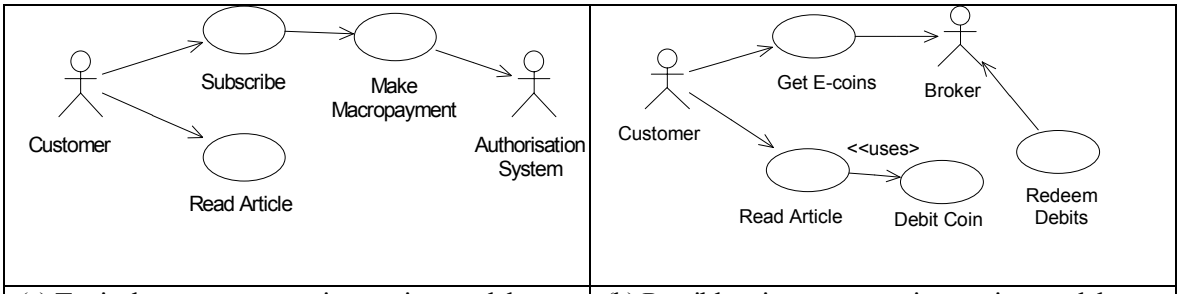
1. Introduction

Most current e-commerce systems adopt a macro-payment model and architecture. A user makes a small number of purchases which have a reasonably high cost per purchase. In order to pay for purchases, a "heavy weight" interaction between the vendor of the product or service and an authorisation agent (bank, credit-card company etc) system is carried out. This typically involves the user supplying credit card details or "digital money" certificates, which are communicated to the authorisation system using complex encryption algorithms. Business processing logic and database updates are performed by the authoriser before the purchase is approved. The vendor system waits for approval before providing the customer with goods or services. This approach works well for relatively small numbers of transactions and relatively high purchase price (to offset the cost of authorisation) [3]. In some e-commerce scenarios this approach has a number of fundamental flaws. It requires the authorisation system to always be on-line. High numbers of transactions or low-price purchase items are infeasible, due to bottle-necking or prohibitive cost per-transaction. In addition, with some approaches the customer's identity can not generally be hidden from the vendor.

We describe the NetPay micro-payment model and architecture we have been developing. NetPay provides an off-line micro-payment model using light-weight hashing-based encryption. A customer buys a collection of "e-coins" using a macro-payment from a broker. These coins are cached in an "e-wallet" on the customer's machine. The customer, when buying many small-cost items from a vendor, pays for these transparently by the passing of e-coin information to the vendor. Periodically the vendor redeems the e-coins with the broker for "real" money. E-coins can be transparently exchanged between vendors when the customer moves to another site. We describe the software architecture and design we have developed for NetPay for deployment with thin-client vendor interfaces i.e. HTML and WML-based interfaces for customers. We describe a prototype implementation of NetPay using Java, Java Server Pages, CORBA and Enterprise JavaBeans. We comment on the performance of this prototype and outline our further plans for research and development.

2. Motivation

Assume a reader wants to browse an on-line newspaper [3] 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



(a) Typical macro-payment interaction model. (b) Possible micro-payment interaction model.

Figure 1. Two on-line newspaper interaction scenarios.

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 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. These issues apply to many other information sources on the internet where vendors want to charge for content [1,6].

An alternative approach is a micro-payment model. There are several approaches to micro-payment [4,5,7,8,9] - we outline the basic interactions of the NetPay model we have developed. *Figure 1* (b) outlines the key interaction use cases for this scenario. 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, 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. The standard macropayment 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 [4,7]. Encryption mechanisms used are slow and each transaction typically "costs" a few cents. Macro-payment suits spending small numbers of large amounts. An Internet micropayment 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. The design of micro-payment systems are usually quite different from existing macro-payment systems, since micropayment systems must be very simple, secure, and efficient, with a very low cost per transaction. This must also be taken into consideration for transaction security: high security leads to high costs and computation time. For micropayments low security can be applied.

3. Overview of NetPay

A Netpay micro-payment system includes customers (e.g. newspaper readers), vendors (e.g. on-line e-newspapers) and a broker. We assume that the broker is honest and is trusted by both the readers and the e-newspapers. The micro-payments only involve readers and e-newspapers, and the broker is responsible for the registration of readers and for crediting the e-newspaper's account and debiting the reader's account. *Figure 2* illustrates key NetPay component interactions.

Initially a reader accesses the broker's web site to open an account 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 and e-coins to the reader and the reader's host caches this information. The reader browses the home page of the newspaper web site and finds a

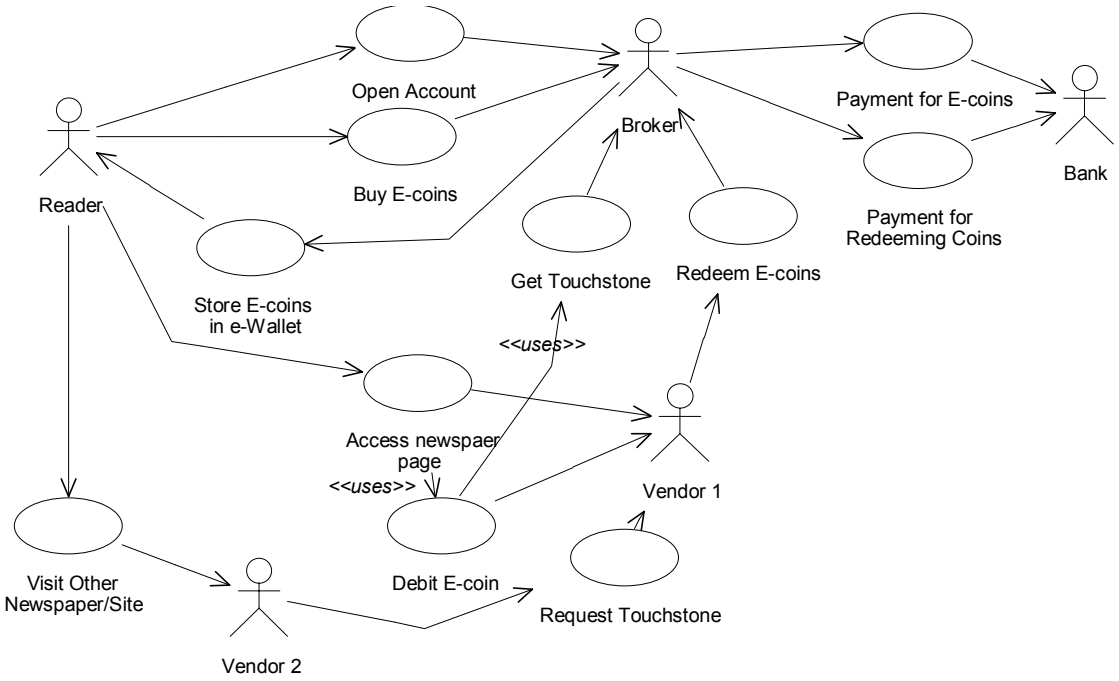


Figure 2. Basic NetPay component interactions.

desired news article to read. Each article will have a small cost e.g. 5-10c, and the user would typically read a number of these. When wishing to read the details of an article, the reader clicks on the article heading and the vendor system debits the reader's e-coins by e.g.. 10c. The vendor verifies that the e-coin provided by the reader'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 reader 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 reader is directed to the broker's site to buy more. When the reader 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 to further news articles. 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).

The management of the security of e-coins is one of key issues in micropayment systems. Netpay uses a low-cost per transaction yet high security method between customers and vendors to secure the use of e-coins [2,3]. 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 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, vendor from over-debiting and ensures no conflicts between vendors [2]. The vendor does not know the identify 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 it is a full off-line system.

The Netpay system 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 [2,3].

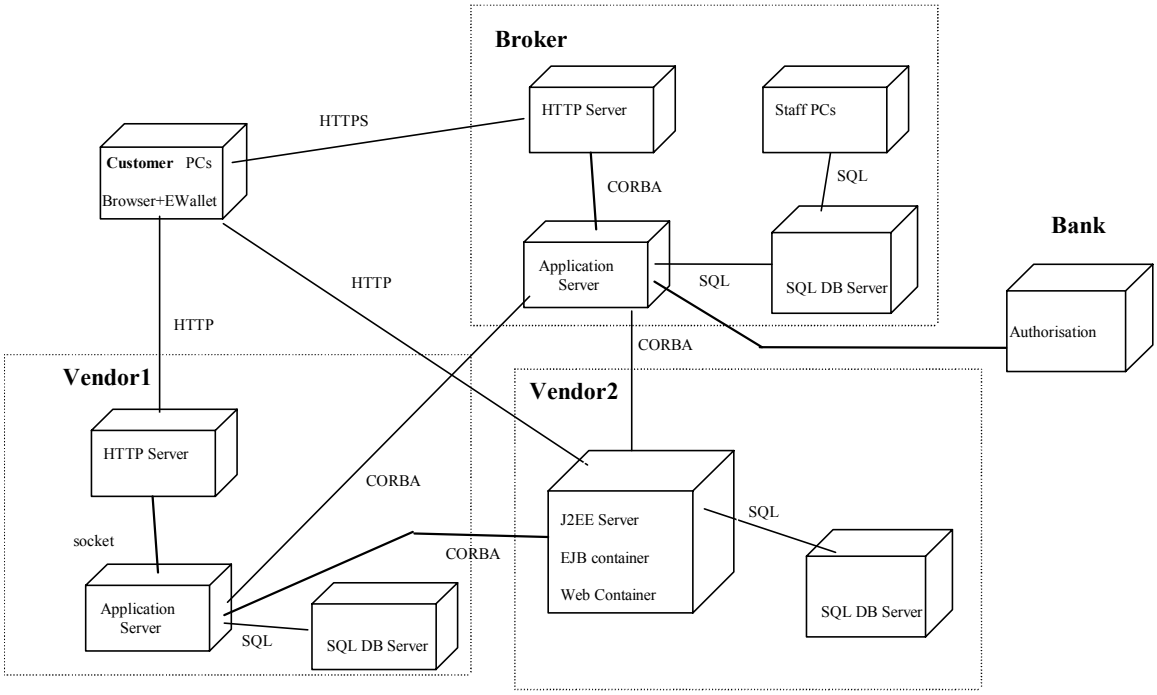


Figure 3. Basic NeyPay software architecture.

Customers are prevented from double spending as the index of the payword chain indicates the balance of the customer's e-wallet**,** and the hashing function can be used to verify the index from the touchstone. NetPay allows customers to move transparently from one vendor site to another, with a single e-coin touchstone and index transfer between vendors.

4. Netpay Architecture

We have developed a software architecture for implementing NetPay-based micro-payment systems for thin-client web applications. Netpay micropayment transactions involve three key parties: the Broker Server, the Vendor Server, and the Customer browser. This architecture is illustrated in *Figure 3*.

The Broker provides a database holding all customer and vendor account information, generated coins and payments, redeemed coins and macro-payments made (buying coins and redeeming money to vendors). The Broker application server provides a set of CORBA interfaces vendor application servers communicate with to request touchstones and redeem e-coins. This server also communicates with one or more bank servers to authorise macro-payments (customer buying coins or broker paying vendors when redeeming spent coins). The Broker web server provides a point of access for customers to buy e-coins and check their e-wallet balances and transaction history.

The customer runs a web browser that accesses the broker and vendor servers, and may also contain an e-wallet implemented by the use of an Applet or Active-X object. In our current NetPay prototype we use two other kinds of e-wallet - one held server-side and one held client-side. The client-side e-wallet is an application running on the client PC holding e-coin information. The server-side e-wallet resides on the vendor server and is transferred from the broker to each vendor in turn the customer is buying content from. When buying e-coins the Broker's application server updates the customer's e-wallet (cached e-coin information). When purchasing information using micro-payment, the vendor's web server accesses e-coin information using the customer's e-wallet.

The vendor sites provide a web server and possibly a separate application server, depending on the web system architecture they use. The Vendor web server pages providing content that needs to be paid for access the customers' e-wallets to obtain e-coins to decrement.

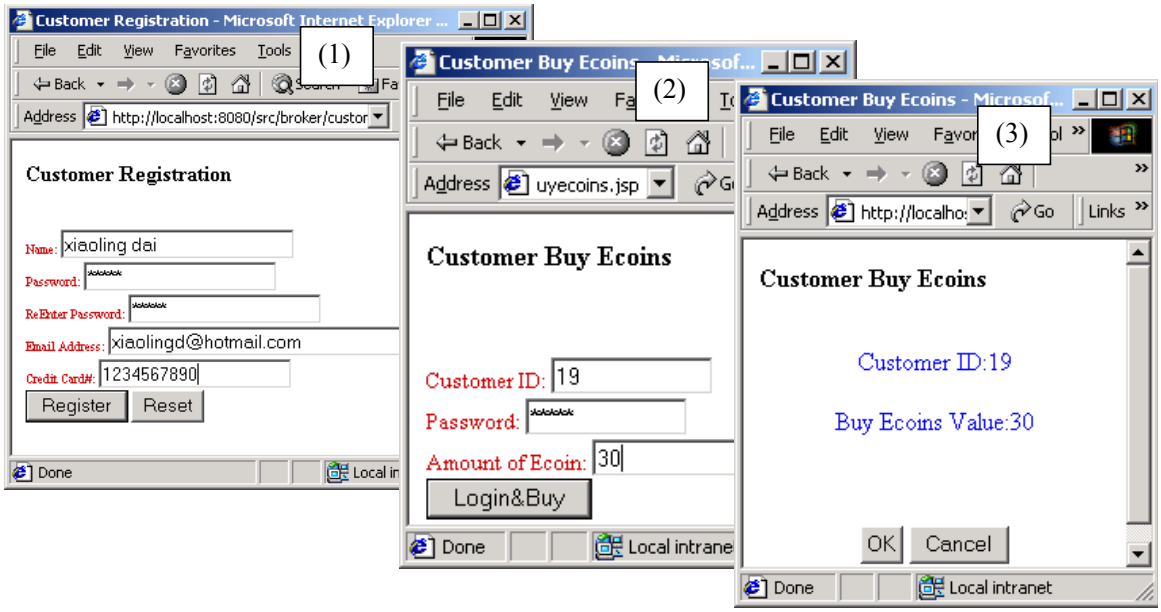


Figure 4. Customer purchasing E-coins from Broker.

The Vendor application server accesses 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 to pass on e-coin indexes and touchstones via a CORBA interface. Vendors may use quite different architectures. In the example above, Vendor #1 uses a web server, custom application server and relational database. 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.

5. NetPay Implementation

5.1 Broker

The broker manages customer and vendor accounts, e-coin creation and spend 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 redeeming of coins spent at the vendor by customers. We chose to use CORBA to provide a platform and language-independent interface supporting a wide range of possible vendor implementation technologies. The HTML interface used by customers to purchase e-coins is shown in *Figure 4*. The customer can register and create or maintain their account information (1). When needing to buy some e-coins, the customer authorises macro-payment by the broker (2) debiting the customer's supplied credit card to pay for the coins (3).

5.2 Customer

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. When the customer first goes to the broker and purchases e-coins, the broker's Java Server Pages providing the customers' account management web pages set the e-wallet in server-side or client-side storing the bought coins. This forms the customer's "e-wallet" as illustrated in *Figure 5*.

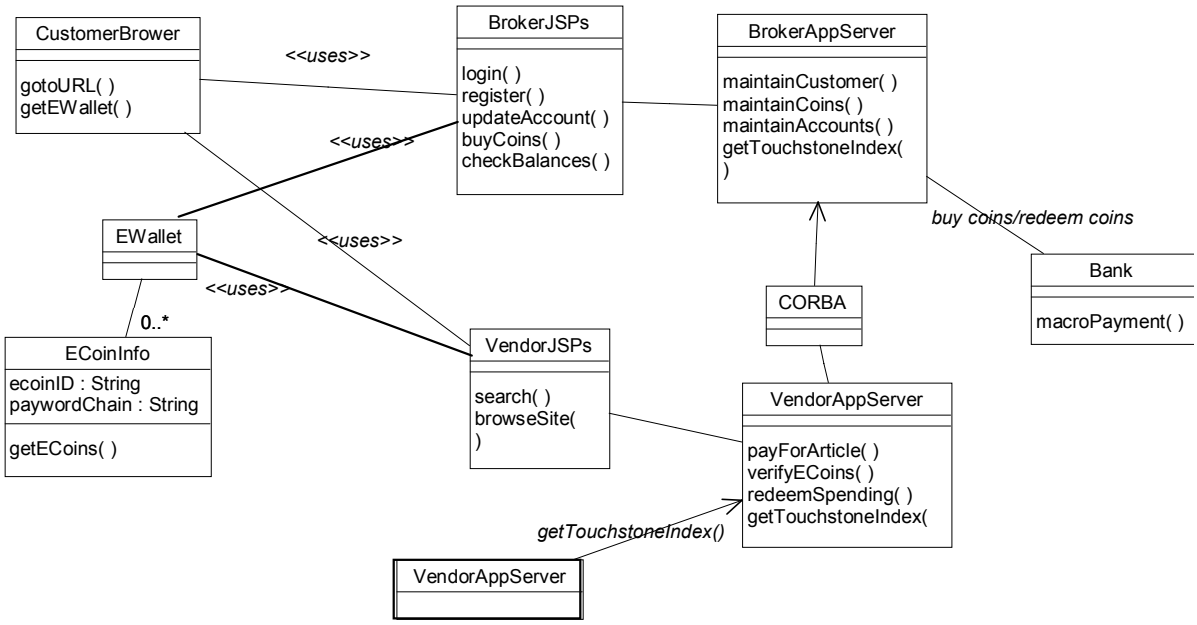


Figure 5. Customer, broker and vendor key design features.

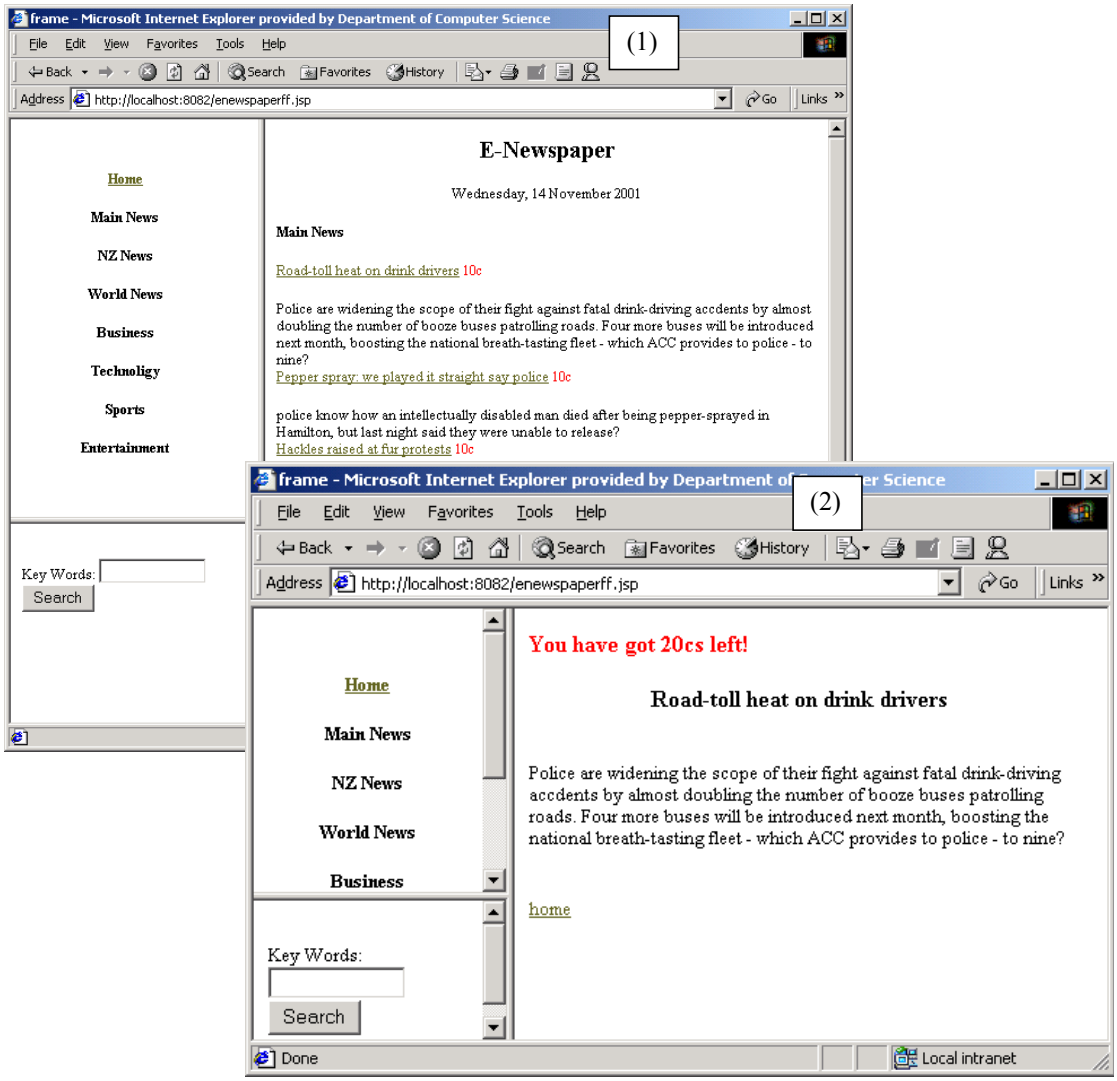


Figure 6. Customer spending E-coins at an E-newspaper site.

Client-side e-wallets are hosted on the customer PC and are an application that stores e-coin information for debit by vendor servers and credit by the broker server. A server-side e-wallet is held on the vendor server 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.

5.3 Vendor

Vendors provide a set of pages implementing on on-line service provider e.g. in our scenario, an e-newspaper. The vendor Java Server Pages not only provide searching, browsing and newspaper content for the customer, but also indicate article cost, as shown in *Figure 6* (1). After reading an article, the vendor Java Server Pages indicate the amount of e-coins redeemable at this vendor are left in the customer's e-wallet (2). When the customer first tries to read an article, the vendor obtains the validating touchstone and index from the broker, in order to verify that the e-coins are valid [2]. When moving to another vendor, the touchstone and current index value of the e-coins are obtained from the previous vendor via the vendor's CORBA interface.

The vendor application server can be implemented using various technologies. We have built vendors using Java Server Pages and are currently building NetPay vendor Enterprise JavaBeans to provide plug-in vendor micro-payment support components.

6. Summary

We have developed a prototype architecture to support an efficient, secure and anonymous micro-payment system. 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. Our NetPay architecture provides for both secure and high transaction volume per item by using fast hashing functions to validate e-coin unspent indexes. NetPay is an off-line protocol allowing the vendors to interact only with customers after initial coin validation. We are currently building EJB-based components for vendors to provide plug-and-play micro-payment support. We are investigating XML-based interaction between vendors and the broker using web services to provide a further abstracted communication mechanism and to support multiple brokers. We hope to explore further generalisation of our architecture for a wider range of E-commerce components.

References

[1] D. Blankenhorn, "Charging for Content, E-commerce Times", http://www.ecommercetimes.com/perl/story/306.html.

[2] X. Dai, and B. Lo, "NetPay – An Efficient Protocol for Micropayments on the WWW", Fifth Australian World Wide Web Conference, Australia, 1999.

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

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

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

[6] A. Herzberg, "Safeguarding Digital Library Contents - Charging for Online Content", D-Lib Magazine, January 1998, ISSN 1082-9873.

[7] M-S. Hwang, I-C. Lin, L-H. Li, "A simple micro-payment scheme", Journal of Systems & Software, vol.55, no.3, Jan. 2001, Elsevier, pp.221-9.

[8] M. Manasse, "The Millicent Protocols for Electronic Commerce", First USENIX Workshop on Electronic Commerce, New York, 1995.

[9] R. Rivest, and A. Shamir, "PayWord and MicroMint: Two Simple Micropayment Schemes", Proceedings of 1996 International Workshop on Security Protocols, Lecture Notes in Computer Science v. 1189, page 69-87. Springer, 1997.