bypass the charge for printing. Merchants may still choose to distribute special software in the belief that tampering will be infrequent. Similarly, there is no technical means to prevent consumers from violating copyright by redistributing information.[1]

NetBill Design

There are a number of challenges to making electronic commerce systems feasible: **High Transaction Volumes at Low Cost —** If information is sold for a few pennies a page, then an electronic commerce system must handle very large transaction volumes at a marginal cost of a penny or less per transaction.

Authentication, Privacy, and Security — The Internet today provides no universally accepted means for authenticating users, protecting privacy, or providing security.

Account Management and Administration — Consumers and merchants must be able to establish and monitor their accounts.

This article describes the architecture of NetBill, a system designed to meet these goals. Our students and we have implemented three generations of NetBill prototypes. In partnership with Visa International and Mellon Bank, we will mount a trial in early 1996 in which various forms of information are sold to consumers using NetBill.

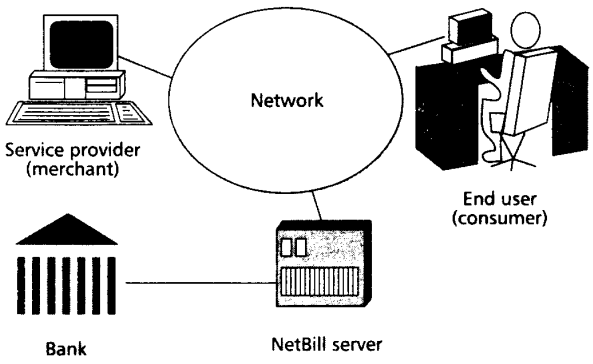
NetBill Architecture

NetBill uses a single protocol that supports charging in a wide range of service interactions. NetBill provides transaction support through libraries integrated with different client-server pairs. These libraries use a single transaction-oriented protocol for communication between client and server and NetBill; the normal communications model between client and server is unchanged. Clients and servers can continue to communicate using protocols optimized for the application — for example, video delivery or data base queries — while the financial-related information is transmitted over protocols optimized for that purpose. This approach allows NetBill to work with information delivery mechanisms ranging from the WWW to FTP and MPEG-2 streams.

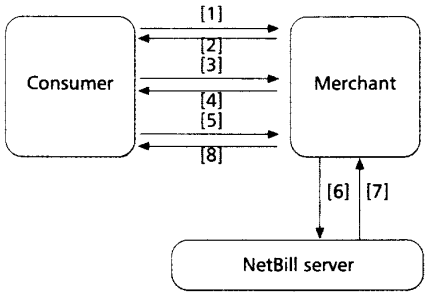
The client library — which we call the Checkbook — and the server library — the *Till* — have a well defined API allowing easy integration with a range of applications. (Below we describe how we integrate these libraries with web clients and http servers.) The libraries incorporate all security and payment protocols, relieving the client/server application developer from having to worry about these issues. All network communications between the Checkbook and Till are encrypted in order to protect against adversaries who eavesdrop or inject messages.

The NetBill Transaction Protocol

Before a consumer begins a typical NetBill transaction, she will usually contact a server to locate information or a service of interest. For example, the consumer may request a Table of Contents of a journal showing available articles, and a



■ **Figure 1.** *NetBill scenario.*



■ **Figure 2.** *Transaction protocol.*

list price associated with each article. NetBill provides support for three phases in the purchase process: price negotiation, goods delivery, and payment. The process begins when the consumer requests a formal price quote for a product. This price may be different than the standard list price because, for example, the consumer may be part of a site license group, and thus be entitled to a marginal price of zero.[2] Alternatively, the consumer may be entitled to some form of volume discount, or perhaps there is a surcharge during the peak hour.

Requesting the price quote is easy. As we discuss below, in a WWW browser application we have built, a consumer requests a price quote by simply clicking on a displayed article reference.

The consumer's client application then indicates to the Checkbook library that it would like a price quote from a particular merchant for a specified product. The Checkbook library sends an authenticated request for a quote to the Till library, which forwards it to the merchant's application (Fig. 2, Step 1).

The merchant then must invoke an algorithm to determine a price for the authenticated consumer. He returns the price quote through the Till, to the Checkbook (Step 2), and on to the consumer's application.

If a merchant must perform a database lookup on every purchase request to determine if the consumer is a subscriber, the price quotation step becomes expensive to implement. Accordingly, we have developed the notion of a *credentials server* that can issue a temporary credential to the consumer indicating that she is a subscriber. This

[1] *Separately, we are researching means of embedding a unique watermark in each copy sold which would allow illegal copies to be traced to the source.*

[2] *In the special case of free information, we can optimize our protocol still further.*

credential, similar in concept to a Kerberos 5 proxy [3], can be presented with each price quote request to enable rapid identification of subscribers. Operation of credential servers can be separated from the merchant service. For example, a digital library running at the University of California at Berkeley could provide site licensed material to all nine UC campuses based on credentials issued by separate servers maintained at each campus.

Upon receipt of the price quote, the consumer's application then must make a purchase decision. The application can present the price quote to the consumer or it can approve the purchase without prompting the consumer. For example, the consumer may specify that her client software accept any price quote below some threshold amount; this relieves her of the burden of assenting to every low-value price quote via a dialog box.

Assume the consumer's application accepts the price quote. The Checkbook then sends (Step 3) a purchase request to the merchant's Till. The Till then requests the information goods from the merchant's application and sends them to the consumer's Checkbook encrypted in a one-time key (Step 4), and computes a cryptographic checksum (such as MD5 [4]) on the encrypted message. As the Checkbook receives the bits, it writes them to stable storage. When the transfer is complete, the Checkbook computes its own cryptographic checksum on the encrypted goods and returns to the Till a digitally signed message specifying the product identifier, the accepted price, the cryptographic checksum, and a time-out stamp; we refer to this information as the *electronic payment order* (EPO) (Step 5). Note that, at this point, the consumer can not decrypt the goods; neither has the consumer been charged.

The NetBill server is highly reliable and highly available. All transactions at the NetBill server are atomic: they either finish completely or not at all. NetBill is never in doubt about the status of a purchase.

Upon receipt of the EPO, the Till checks its checksum against the one computed by the Checkbook. If they do not match, then the goods can either be retransmitted, or the transaction aborted at this point. This step provides very high assurance that the encrypted goods were received without error.

If the checksums match, and the merchant agrees with the item description and price in the EPO, the merchant's application appends the decryption key for the goods to the EPO and endorses — digitally signs — the entire message. The application sends the endorsed EPO to the NetBill server (Step 6).

The NetBill server verifies that the consumer and merchant signatures are valid, indicating consumer and merchant acceptance of the terms of the EPO. If the consumer has the necessary funds or credit in her account, the NetBill server debits the consumer's account and credits the merchant's account, logs the transaction, and saves a copy of the decryption key. The NetBill server then returns to the merchant a digitally signed receipt containing the decryption key, or an error code indicating why the transaction failed (Step 7). The merchant's application forwards the NetBill server's receipt (which includes, if appropriate, the decryption key) to the Checkbook (Step 8).

Protocol Failure Analysis – The above description assumed that no failures occurred during the execution of the protocol. In reality, the protocol must gracefully cope with network and host failures. One of our goals is to tightly link two events: charging the consumer and delivering the goods. The consumer should pay exactly when she receives the information goods.

The NetBill server is highly reliable and highly available. All transactions at the NetBill server are atomic: they either finish completely or not at all. NetBill is never in doubt about the status of a purchase. We cannot make similar assumptions about the reliability of the merchant's and consumer's software; they must maintain a state consistent with the NetBill server.

First, consider the protocol from the perspective of the consumer's application. Up to Step 5, when the consumer application acknowledges receipt of the information goods, the consumer application knows that no transaction has occurred. That is, the consumer does not have access to the product and the merchant does not have the consumer's money. Once the application sends the EPO, the consumer is *committed* to the transaction and must be prepared to accept the purchase. If the consumer's application does not receive a response from the merchant's application, then it is the responsibility of the consumer's application to determine what happened; the consumer's application can poll either the merchant application or the NetBill server to determine the status of the purchase request. If the merchant's application did not successfully forward the EPO to the NetBill server, then the EPO will have expired and the NetBill server will respond to the consumer's application that the purchase has failed. Of course, the consumer still does not have the one time key, so while the consumer still has her money, she also does not have the goods. If, on the other hand, the transaction succeeded before communication failed, then the consumer's application can find the status of the purchase and, if appropriate, the decryption key from either the merchant's application or the NetBill server (which has registered the key). If both are unreachable, the consumer's application must continue to poll.

Now consider the protocol from the perspective of the merchant's application. Before it forwards the EPO and invoice to the NetBill server, the merchant's application knows that the transaction has not occurred. After it forwards the EPO and invoice, however, the merchant's application is *committed* to the transaction and must obtain the result from NetBill. If the merchant's application does not receive a response from the NetBill server, the merchant's application must poll the NetBill server.

The protocol is much simpler for the NetBill server than for the other parties. The NetBill server is never in a state in which it depends on a response from another entity to determine the status of a transaction. Until the NetBill server receives the EPO and invoice from the merchant's application, it knows nothing about the purchase. Once it receives the EPO and invoice it

has all the information necessary to approve or reject the purchase.

We use the term *certified delivery* to describe the mechanism of delivering encrypted information goods and then charging against the consumer's NetBill account, with decryption key registration both at the merchant's application and the NetBill server. The NetBill transaction protocol also exhibits a number of other desirable features:

- Support for flexible pricing: by including the steps of offer and acceptance, we provide an opportunity for the merchant to calculate a customized quote for an individual consumer.
- Scalability: the bottleneck in the NetBill model is the NetBill server which supports many different merchants. Our transaction protocol minimizes the load on the NetBill server and distributes the burden over the many consumer and merchant machines. Note that a single interaction with the NetBill server both verifies the availability of funds and records the transaction. It is not possible to have less than one interaction with the NetBill server.[3]
- Protection of consumer accounts against unscrupulous merchants: in a conventional credit card transaction, the merchant learns the consumer's credit card number and can submit fraudulent invoices in the consumer's name. In a NetBill transaction, the consumer digitally signs the EPO using a key that is never revealed to the merchant, thus eliminating this threat. Moreover, the consumer has proof of the exact nature of the information goods received, providing evidence in case a dishonest merchant attempts to deliver faulty information goods.[4]

NetBill Account Management

In this section, we discuss how consumers and merchants can manage their NetBill accounts.

NetBill supports a many-to-many relationship between *consumers* and *accounts*. A project account at a corporation can have many users authorized to charge against it. Conversely, an individual consumer can maintain multiple personal accounts. Every account has a single user who is the account *owner*; and the account owner can grant various forms of access rights on the account to other users.

Consumer account administration is provided through WWW forms. Using a standard WWW browser, an authorized consumer can view and change a NetBill account profile, authorize funds transfer into that account, or view a current statement of transactions on that account. Authentication and security are provided by treating account information as billable items. NetBill provides account information to consumers using the NetBill protocol. NetBill can be configured to provide this information for free or for a service charge, as desired.

Automating account establishment for both consumers and merchants is important for limiting costs. (Account creation is one of the largest costs associated with traditional credit card and bank accounts.) To begin the process, a consumer retrieves, perhaps by anonymous ftp, a digitally signed NetBill security module that will work with the consumer's WWW browser. Once the consumer checks the validity of the security module, she puts the module in place. She then fills out a WWW form, including appropriate credit card or bank account information to fund the account, and submits it for processing. The security module encrypts this information to protect it from being observed in transit. The NetBill server must verify that this credit card or banking account number is valid and that the consumer has the right to access it. There are a variety of techniques for this verification: for example, consumers may telephone an automated attendant system and provide a PIN associated with the credit card or bank account to obtain a password.

NetBill Costs and Interaction with Financial Institutions

In a modern market economy, there are many forms of money, but two distinct poles typify the range of alternatives: tokens and notational money. Currency consists of unforgeable tokens that are widely accepted by both buyers and sellers as a store of value. In a cash transaction, the seller delivers goods to the consumer while the consumer delivers currency to the seller. Other projects are developing forms of electronic currency for network commerce based on unique digital bit strings [5].

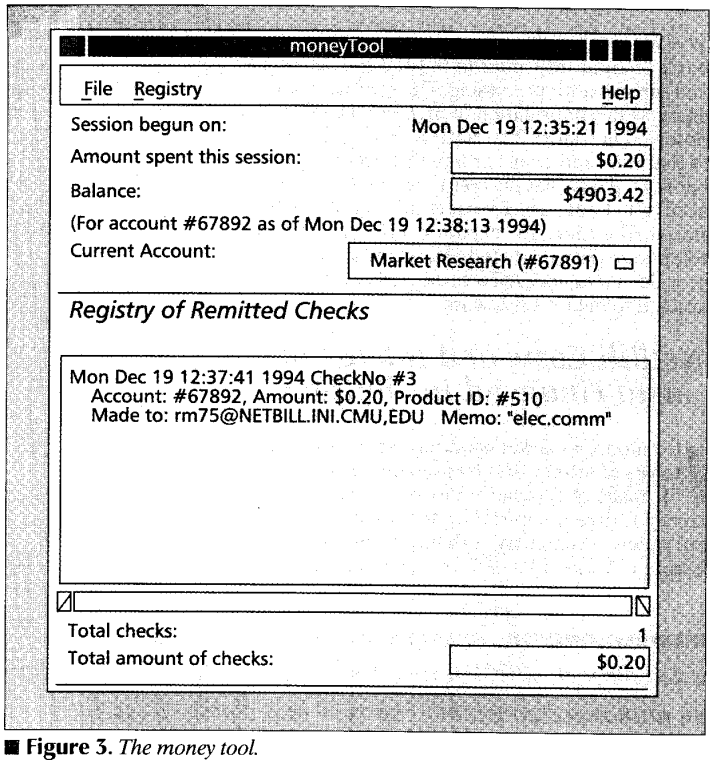
Consumer account administration is provided through WWW forms. Using a standard WWW browser, an authorized consumer can view and change a NetBill account profile, authorize funds transfer into that account, or view a current statement of transactions on that account.

Demand deposit accounts at a bank are an example of notational money: on instruction (a check) by a consumer, funds move from one ledger to another. A complex system involving intermediaries such as the Federal Reserve supports check clearing and settlements when the accounts are held at different banking institutions. Settlements can involve significant delays during which funds are not available to either party in a transaction. Notational accounts can have either a positive or negative balance, depending upon whether a bank is willing to extend credit to a buyer. For example, a credit card account runs a negative balance as the issuing bank executes instructions to transfer funds to a merchant's bank account.

Orders to transfer notational money are increasingly sent using electronic mechanisms: FedWire, automated clearinghouses (ACH), credit card authorization and settlement networks, and automated teller machine networks are all examples. NetBill also uses notational money. Because both consumers and merchants maintain NetBill accounts, interinstitutional clearing costs are not incurred for every transaction. NetBill accounts provide a low cost mechanism to aggregate small-value transactions before invoking a relatively high fixed cost conventional transaction mechanism. Consumers move money into their NetBill account in large chunks (for example, $50 to $100) by charging a credit card or through an ACH transaction. Similarly, money moves from a merchant's NetBill account to the

[3] In theory, one might bundle several transactions together and have them all processed as part of one interaction with NetBill. However usage data collected from Carnegie Mellon's Library Information System indicates that in the majority of cases, users contacting the library are looking for a single item, suggesting that bundling would not be appropriate.

[4] For more details on the security and transaction protocol see, B. Cox, D. Tygar, and M. Sirbu, "NetBill Security and Transaction Protocol," Proceedings of the 1995 Usenix Workshop on Electronic Commerce, New York, July, 1995.



■ **Figure 3.** *The money tool.*

merchant's bank through an ACH deposit transaction.

NetBill accounts can be either pre-paid (debit model) or post-paid (credit model). In the pre-paid model, funds would be transferred to NetBill in advance to cover future purchases. If the consumer does not have sufficient funds to cover a particular transaction, that transaction would be declined. The amount of any prepayment is set by the consumer, subject to minimums and maximums established by the NetBill operator. On pre-paid accounts, the system allows consumers to designate the balance at which she is prompted to transfer additional funds to NetBill. Because ACH transactions take several days to clear, a consumer prepaying her NetBill account through the ACH may not have immediate access to the funds. Funding through a credit card, while incurring larger transaction fees, allows immediate access to a prepayment.

In the credit model, transactions would be accumulated with payment to NetBill triggered by either time (based on a pre-established billing period) or dollar amount (based on a pre-established limit). Because granting credit creates a risk of non-payment, higher transaction fees may be associated with credit, versus prepaid accounts.

The design space for electronic transaction systems has three crucial dimensions: risk, delay, and cost. For immediate transactions, risks of fraud or non-payment can be dealt with in two ways:

- Incorporating an insurance fee proportional to the transaction amount.
- Investing in sophisticated security systems with (high) fixed costs independent of transaction size.

Credit card systems are of the first type, typically charging 1 to 3 percent of the value of the transaction, while FedWire takes the second approach. Delay can reduce risk by allowing verification of fund availability before committing a transaction, and by allowing batching to achieve economies of scale, particularly in inter bank settlements. However, delay imposes opportunity costs when funds are not available until cleared.

NetBill is optimized for very low marginal transaction costs (on the order of 1 cent) on small value transactions (on the order of 10 cents). Fixed networking costs are reduced by using the Internet, with its substantial economies of scale, as opposed to a dedicated single function network. Because both consumers and merchants maintain accounts at NetBill, most transfers are internal to NetBill; this reduces both risk and processing cost. When fund transfers outside NetBill are necessary, they can take advantage of aggregation, which spreads fixed transaction costs over larger sums. Use of ACH transfers and prepaid accounts minimizes risk at the cost of some delay before incoming funds are available; where NetBill offers deposits through credit cards, or grants credit itself, the risk increases and must be passed on to consumers as higher fees.

NetBill keeps other costs of operation low by: automating all account administration functions; using techniques like certified delivery to reduce the incidence of complaints and consumer service costs; and using a modern distributed processing approach for the core NetBill processing system.

An Example of NetBill with the World Wide Web

Because WWW browsers and servers are a *de facto* standard for distributing information over the Internet, we have created a prototype implementation of NetBill that allows for billing of WWW transactions. Rather than link the NetBill libraries with a WWW browser and http server, we have enabled commerce with no modification to either the browser or the server. Our design introduces two entities in order to support the exchange of money for goods: the *MoneyTool* and the *Product Server*. The MoneyTool runs on the consumer's machine and works with a Web browser. It allows the consumer to authenticate, select accounts, approve/deny transactions, and monitor expenditures. The Product Server, which incorporates the Till libraries, works with the http server to sell information products.

When a consumer clicks on a product in a product server's catalog, the server returns a special file with a mime type containing information about the server's identity, the product to be ordered, and the port number of the product server. This mime type spawns a helper program in the same way that jpeg, sound, and mpeg files currently do. The spawned program communicates the contents of the file between the browser and the MoneyTool.

The MoneyTool acts as the consumer's application in the NetBill transaction protocol described above. After it receives and decrypts the goods, it uses the remote control function of Mosaic to cause the browser to display the received information. Besides implementing the steps in the protocol, the Money Tool provides a number of useful functions to help the consumer manage trans-

actions (Fig. 3). The MoneyTool:

- Provides an authentication dialog window.
- Provides a running total of expenditures in the current session and the current balance in the consumer's NetBill account.
- Provides a listing of all EPOs processed in the current session.
- Can be configured to automatically approve expenditures below a threshold.
- Can be used to retrieve the product encryption key from NetBill in the event of failure of a merchant host.

Spyglass has recently proposed a standard Software Developer's Interface (SDI) for interfacing with browsers [6] which has also won support from Netscape. In the future we expect to integrate the MoneyTool with the browser using this mechanism. Use of the SDI interface will allow purchase requests to be passed directly to the MoneyTool without the round trip currently required to send back a special mime type.

In the current implementation, the initial request for goods to the http server causes the server to run a script that writes information about the request to a temporary file at the server. When the Product Server receives a request for a price quote from the MoneyTool, it must access the server's database to determine the price quote based on the consumer identity. If the quotation is approved, the product server finds the goods using the information saved by the http server and completes the NetBill transaction protocol. Once we shift to using the SDI interface to the browser, the product request information will be forwarded directly to the product server in the price request, avoiding the need for a temporary file.

Additional Issues

As described above, NetBill is well suited for supporting commerce in information goods. However, the NetBill model can also be extended in a variety of ways to support other types of purchases. For example, NetBill could be used equally well for conventional bill paying. A consumer could view a bill presented as a Web page; instead of buying information goods, we can think of the consumer as buying a receipt for having paid the bill.

If the product to be bought is a one-hour movie, it is likely that the consumer will want to stream the data directly to a viewer, which conflicts with NetBill's model of certified delivery. We are exploring alternative approaches, such as using the standard NetBill protocol to periodically buy a key for the next N minutes of an encrypted video stream.

We are also exploring the software rental application. A software vendor could incorporate the Checkbook library in any arbitrary application software. Periodically, the software would ask the consumer to approve the purchase of a key for the next month's operation. (This requires mechanisms to prevent the software vendor from including a Trojan Horse designed to capture a renter's password.)

NetBill is well suited for supporting commerce in information goods. However, the NetBill model can also be extended in a variety of ways to support other types of purchases.

Acknowledgments

Much of the development of NetBill has been done by students in project courses taken as part of Carnegie Mellon's graduate program in Information Networking. We thank all of those students for their help and ideas.

We gratefully acknowledge the support of the National Science Foundation (under cooperative agreement IRI-9411299), Visa International, and an internal grant from the Information Networking Institute.

Notes

For more information on NetBill, including a fuller version of this paper, please look at our WWW page at `http://www.ini.cmu.edu:80/netbill/`.

NetBill is a registered trademark of Carnegie Melon University.

References

[1] M. Porat, The Information Economy [U.S. Office of Telecommunications, 1977].

[2] R. Seitz, "Phone Companies Join Their Rivals in the Facts Business," *The New York Times*, June 7, 1992.

[3] B. C. Neuman, "Proxy-Based Authorization and Accounting for Distributed Systems," *Proc. of the 13th Int'l Conf. on Distributed Computing Systems*, May, 1993, pp. 283-291.

[4] Rivest, The MD5 Message Digest Algorithm, April 1992.

[5] D. Chaum, "Achieving Electronic Privacy," *Scientific American*, 267, vol. 2, 1992, pp. 76-81.

[6] See: http://www.spyglass. com:4040/newtechnology/integration/iapi.htm.

Biographies

J. D. TYGAR [M '82] received an A.B. from the University of California, Berkeley, and a Ph.D. from Harvard University. He is an associate professor of computer science at Carnegie Mellon University, Pittsburgh, Pennsylvania. His interests include electronic commerce, computer security, cryptography, and distributed systems. In addition to his previous work on the ITOSS Security Toolkit, the StrongBox distributed security project, and the Miro security specification language, he directs several projects including the NetBill microtransaction system (with M. Sirbu), the Dyad secure coprocessor system, and a project with the U.S. Postal Service to establish cryptographic postage indicia standards. He has consulted widely for industry and government. He was named a Presidential Young Investigator by the National Science Foundation. He is a member of the American Association for the Advancement of Science, the Association of Computing Machinery, the American Mathematical Society, and the Mathematical Association of America.

MARVIN SIRBU holds a joint appointment as professor in Engineering and Public Policy, the Graduate School of Industrial Administration, and Electrical and Computer Engineering at Carnegie Mellon University, Pittsburgh, Pennsylvania. His interests are in telecommunications and information technology, policy, and economics. In 1989 he founded the Information Networking Institute at CMU which is concerned with interdisciplinary research and education at the intersection of telecommunications, computing, business and policy studies. Recent research activities have focused on electronic commerce, the economics of electronic publishing, compatibility standards in communications and computers, pricing of new telecommunications services and new technologies for the local loop. He received S.B., S.M. and Sc.D. degrees in electrical engineering from the Massachusetts Institute of Technology. Prior to coming to Carnegie Mellon in 1985, he taught at the Sloan School of Management at M.I.T. where he also directed its Research Program on Communications Policy.

[5] `http://www.spyglass.com:4040/newtechnology/integration/iapi.htm`.