A Lightweight Currency Paradigm for the P2P Resource Market

David A. Turner
Dept. of Computer Science
CSU San Bernadino
San Bernardino, CA
dturner@csusb.edu

Keith W. Ross
Dept. of Computer and Information Science
Polytechnic University
Brooklyn, NY
ross@poly.edu

February 3, 2004

Abstract

A P2P resource market is a market in which peers trade resources (including storage, bandwidth and CPU cycles) and services with each other. We propose a specific paradigm for a P2P resource market. This paradigm has four key components: (*i*) pairwise trading market, with peers setting their own prices for offered resources; (*ii*) multiple currency economy, in which any peer can issue its own currency; (*iii*) a simple, secure application-layer protocol; and (*iv*) entity identification based on the entity's unique public key. We argue that the paradigm can lead to a flourishing P2P resource market, allowing applications to tap into the huge pool of surplus peer resources. We illustrate the paradigm and its corresponding Lightweight Currency Protocol (LCP) with an example in content distribution.

1 Introduction

Today many peers (that is, Internet-connected computer systems) possess surplus bandwidth, storage and CPU resources; for example, a peer with a broadband DSL connection typically utilizes only a small fraction of its transmission, storage and computation capacity. When aggregated together across all peers worldwide, these unused resources constitute a huge, untapped resource pool. This resource pool can potentially be used by many applications, including peer-driven content distribution, globally distributed archival storage, massively parallel computation and P2P file sharing. Of course, applications can only harness the resource pool if peers make available their surplus resources to them. Unfortunately, peers are typically "rational," and are thus reluctant to volunteer their resources without sufficient incentives [30, 1].

Now suppose the existence of an online market place where entities - such as peers, companies, users and so on - buy and sell surplus resources. In this market place, a given peer might purchase storage and bandwidth from a dozen other peers for the purpose of remotely backing up its files; a content publisher might purchase storage and bandwith from thousands of peers to create a peer-driven content distribution network; a biotechnology company might purchase CPU cycles from thousands of peers for distributed computation. If such a flourishing resource market existed, individual peers would be incited to contribute their resources to the market place, thereby unleashing the untapped resource pool.

In this paper we describe a resource market paradigm that has been expressly designed to incite peers to make their surplus resources available to P2P applications. The paradigm embraces a free market economy in which entities directly trade with each other for resources or services. At the heart of the

paradigm are lightweight currencies, which are not linked to real-world currencies and which, in fact, are not legal tender. We refer to the paradigm described in this paper as the **lightweight currency paradigm**.

Because lightweight currencies are not tied to real money, we conjecture that users would feel comfortable delegating trading decisions to local trading agents. Rather than directly making trading decisions, a user would only occasionally set high-level trading policies with its local trading agent. This delegation is important since micropayments impose mental decision costs that users prefer to avoid [23].

One might argue that the P2P resource market could use bartering in lieu of currency. In a bartering deal, a peer might trade, for example, a certain number of CPU cycles for a certain amount of remote storage. On the the other hand, in a P2P resource market using currency, a peer might trade CPU cycles for currency, and later trade, with a different peer, its earned currency for remote storage. As in real-world economies, currency can dramatically increase trading volumes, thereby enhancing the exploitation of the untapped peer resource pool. Furthermore, historical evidence suggests that currency markets can emerge without precursor barter markets; one such example is frequent flyer miles, which can be earned from and spent on airline tickets, hotel stays and rental cars. Another example is the implicit currency in the Kazaa P2P file-sharing system (which we discuss in Section 2).

In this resource market paradigm, any entity that sells resources/services sets its own prices for the resources/services. Furthermore, the paradigm does not impose a single currency - in this ultra-free market economy, any entity can issue its own currency. Instead of imposing a monolithic currency controled by some pre-determined authority, there will be multiple, competing currencies – with some currencies having more value than others.

As a simple example, a content publisher (for example, CNN, Disney, PBS, or BBC) can issue its own currency. The content publisher could then use its currency to purchase unused storage and bandwidth resources from Internet-connected peers. For example, Disney might offer Bob five Disney units per month if he agrees to store a Disney video clip and stream it (on Disney's behalf) to other peers 100 times per month. Bob can then use his earned Disney currency to receive other video clips from Disney for his family's viewing pleasure. Furthermore, if other content publishers accept Disney currency for payment, Bob can use his Disney currency to purchase content from these other publishers. Using a real-world currency for such micropayments is not feasible in this scenario due its high transaction costs. Other application scenarios include the sale and purchase of raw computing resources for grid systems, storage and bandwidth for redundant distributed backup systems, payment-based control of spam, gaming rewards, and the sale and purchase of basic services such as name resolution and proxy services. Importantly – as with modern, real-world currencies – a lightweight currency can be transferred among applications - that is, an entity can earn currency in the context of one application and spend the same currency in the context of another.

In this paradigm, each currency issuer maintains an account for each entity that holds some of its currency. When an entity buys a resource from another entity with a specific currency, the corresponding currency issuer simply debits the buying entities account and credits the selling entities account. Although resources are traded among peers, currency issuers and other infrastructure components would typically run on dedicated servers. Thus, although trades are peer-to-peer, we are not necessarily advocating that the trading infrastructure be peer-to-peer as in [36]. Entites that own and maintain infrastructure components would also be compensated, most likely through commissions.

In this we also describe the Lightweight Currency Protocol (LCP), whose messaging and security layers define how an entities interact with currency issuers. As part of the protocol, each entity is identified by a globally unique public key. *Importantly, the system does not use certificates to bind an identifier to a public key.* When an entity wants to transfer a specific currency to another entity, it establishes a secure channel with the currency issuer and then sends a message requesting that funds be transferred to a specific public key. We explain in this paper many of

the decisons that were taken in defining the LCP.

Although we have expressly designed the paradigm and protocol for enabling a P2P resource market, the lightweight protocol can potentially enhance or enable other applications - such as spam reduction and lookup services - for which transactions with real-world currencies are impractical.

In Section 2 of this paper we outline a number of approaches to tapping the surplus resource pool. We also provide an overview of our lightweight currency paradigm. In Section 3 we describe the Lightweight Currency Protocol (LCP), which defines four message types and a security sublayer based on SSL. In Section 4 we describe how several P2P applications can integrate the LCP and how currency can be transferred among applications. These applications include P2P content distribution, P2P distributed archival storage, spam reduction and derivative services. In Section 5 we examine possible LCP application interfaces. In Section 6 we describe related work. In Section 6 we describe the scalability of the Lightweight Currency System. In Section 7 we describe related work. In Section 8 we describe future directions of research.

2 Lightweight Currency Paradigm

2.1 Why Not Resource Cooperatives

One approach to tapping the surplus peer resource pool is to design a **resource cooperative**. A resource cooperative consists of (*i*) a large number of entities that contribute resources to a resource pool, and (*ii*) a central authority which dynamically reallocates the contributed resources. To clarify the notion of a resource cooperative, consider the special case of a storage cooperative, whereby each member contributes disk storage resources to the cooperative and receives as payment from the cooperative free globally distributed file backups. In such an economic system, members do not directly interact with each other but instead interact with the cooperative's resource manager. For example, when a member wants to backup a file, it makes a request to the resource manager; the resource manager then determines the peers in the cooperative over which the file will be replicated (and perhaps fragmented). The resource manager could run on a centralized node or it could be distributed across many (or all) of the peers in the cooperative. PAST [29], CFS [6] and Oceanstore [16] are examples of distributed resource management protocols for P2P archival storage cooperatives.

In a more general resource cooperative, members contribute transmission bandwidth and CPU cycles in addition to disk storage. Anderson and Kubiatowicz outline an architecture for a general resource cooperative[2]. Their architecture includes a central group of servers that manage the resources in the participating peers. Anderson and Kubiatowicz suggest using currency for incentives, whereby the centralized resource manager credits members for contributing resources and debits members for utilizing others' resources[2].

Although resource cooperatives are potentially viable long-term paradigms for P2P resource sharing, we believe they are impractical for near-term adoption (3-5 years). As acknowledged by Anderson and Kubiatowicz, centrally managed systems have a chicken-and-an-egg problem, requiring initial membership levels to be high for applications to be useful[2]. Thus, due to the absence of short-term incentives, there is no evolutionary path for adoption. We instead advocate a more grass-roots, market-driven approach for jump-starting a global P2P resource market in the short term. In the long-term, our approach may become the dominant paradigm; or it may become a fundamental building block in resource cooperatives with centralized management.

2.2 Incentives in P2P File Sharing

P2P file sharing systems - such as Napster, Gnutella, and Kazaa – have had limited success at tapping the surplus peer resource pool[27, 15, 10]. In these systems, users volunteer their storage and bandwidth resources to store and transmit files for the benefit of a large community. However, it is widely documented that the popular P2P file sharing systems are havens for "free riders": a significant fraction of users do not contribute any resources, and a minute fraction of

users contribute the majority of the resources[30, 1].

Recently, Kazaa, which is as of this writing the most popular P2P file sharing system, introduced an incentive scheme to encourage users to upload files. In this scheme, each Kazaa client keeps track of the difference between the number of bytes it has uploaded and the number of bytes it has downloaded. If this difference is above a specific threshold, then the user is provided priorities in download queues. Although this incentive scheme is easy to cheat (for example, by generating artificial uploads in a high-speed LAN), we conjecture that a relatively small fraction of users are actually cheating. One can view the difference of uploaded and downloaded Kazaa bytes as a kind of currency that a client earns and spends. We also conjecture that this scheme incites many (perhaps a large fraction of) users to be productive uploaders, thereby abating the free-rider problem.

Our lightweight currency paradigm is akin to the Kazaa scheme in the sense it uses currency as a means to incite peers to offer resources to other peers. However, in our paradigm, the lightweight currency protocol can be adopted by any P2P application (for example, P2P content distribution, P2P backup storage, P2P file sharing). Furthermore, the paradigm allows a peer to earn currency with one application and then spend the currency on another application, which should give significantly higher trading volumes than would the totality of disjoint resource markets.

There are interesting analogies between P2P resource economies and real-world economies. The resource cooperatives are analogous to communist economies, in which there is a central authority that manages prices, currencies and resource allocation. Our lightweight currency paradigm is somewhat analogous to a free-market, capitalist economies, in which individuals trade directly with each other. However, as we'll see in the next subsection, the lightweight-currency paradigm has some components that are not present in prominent real-world economic systems - most notably, the currencies are not expected to be used as legal tender.

2.3 Paradigm Components

We use the term **entity** as a general term to describe the user of lightweight currency. An entity can be an individual, an organization, a computer system, or a software agent. An entity typically has **resources** (for example, storage, bandwidth and CPU cycles) or **services** (such as a lookup or banking service) that it is prepared to share with other entities.

We now list the cornerstones of our paradigm. It consists of five entwined economic and technological components:

1. **Public-key identification:** The system does not use certificates to bind an identifier to a public key; rather, each entity uses its public key directly for identification purposes. This convention provides two benefits: entities can establish globally unique identities without the need of a centralized naming service, and entities can authenticate and establish secure communication channels. We will sometimes use A1A1, B2B2, and C3C3 to represent in examples the public keys that are used as both names of entities and the basis of establishing secure connections.

2. **Free-market trading:** An entity that wishes to sell a resource/service sets its own price (or takes bids) for the resource/service. Entities transact directly with each other in a pairwise fashion.

3. **Multiple currencies:** The paradigm does not specify a central authority that issues and manages currency. Instead, in this ultra-free-market economy, any entity may issue a currency, which can then be held by other entities. By letting any entity issue a currency, these currencies will compete with each other, providing economic efficiencies. In the long-run, there may be only a small number of highly coveted currencies, as there are a small number of coveted real-world currencies today. This multiple-currency P2P environment is analogous to the environment during the U.S. westward expansion in the nineteenth century, when wildcat currencies were issued by local banks, companies, and even individuals [20].

4. **Not legal tender:** The lightweight currencies are intended for micro payments, typically made by automated agents acting on the behalf of entities such as users and companies. In order to minimize the transaction costs associated with these payments, the paradigm does not provide legal recourse for resolving trading disputes. The paradigm is instead based on trust, reputation, and replicated transactions.

    - **Trust:** When first doing business with each other, the two trading entities will likely make small deals with each other. The size and/or frequency of the trades may increase as the entities do more and more business.
    - **Reputation:** We expect the emergence of third-party reputation systems, in which entities rate each other on the basis of previous transactions[7, 14, 3, 19, 22, 8]. If an entity A1A1 is considering purchasing resources/services from entity B2B2, A1A1 can consult a reputation system before proceeding with the transaction.
    - **Replicated Transactions:** Suppose A1A1 wants to backup its files in other entities' storage. Because transaction costs are expected to be minute, A1A1 can copy its files with a large number of entities. In this way, if one or more entities turn out to be unreliable - either because they are dishonest or because they have unreliable storage systems or network connections - with high probability A1A1 will still be able to retrieve its files from some entity with which it has an agreement.

    This not-legal-tender component is perhaps what distinguishes more than anything else lightweight currency from real-world currency. We draw an analogy with the Internet service model. In the Internet, applications receive no end-to-end guarantees on transmission rate or delay; nevertheless, the Internet's protocols and resource provisioning allow it to perform satisfactorily, and at low cost, for many compelling applications. In the same manner, lightweight currency transactions do not give buyers any hard guarantees; nevertheless, we expect lightweight currency to enable many new important applications. Because lightweight currencies are not tied to real-world currencies, we conjecture that users will feel comfortable delegating trading decisions to local trading agents, which would be broadly directed by user-specificed policies.

5. **Simple protocol:** In addition to implementation simplicity, one of the goals of the lightweight currency paradigm is ease of integration with new and existing services. For this reason, we avoid defining a complex protocol that provides strong anonymity. Consequently, currency issuers will be able to observe money flows between entities (which are identified by public keys), but will not know what has been purchased. The issuer will also be able to ascertain the IP address of these entities. If IP anonymity is desired, entities can hide their IP addresses to issuers by using go-between entities to perform message forwarding for them.

2.4 Operational Overview

A currency issuer maintains an account for each entity that holds its currency. The set of accounts maintained by a currency issuer can be thought of as a table that maps public keys to units held. An entity holding units of a given currency can send a message to the currency issuer to have funds transferred from its account to another entity's account. Typically, a transfer occurs during a sale of resources or services; the entity that requests transfer of funds is the buyer and the recipient of the funds is the seller. When a buying entity transfers units to a selling entity, the currency issuer debits the buyer's account by the transfer amount and credits the seller's account by the same amount. If the seller does not have an account with the issuer, the issuer creates an account for it.

The Lightweight Currency Protocol (LCP) uses four messages to transfer funds in a given currency from one entity to another.

- **transferFundsRequest message:** Sent by the buying entity to the currency issuer; specifies amount to be transferred and to which entity (identified by a public key).
- **transferFundsResponse message:** Sent by the currency issuer to the buying entity; indicates success or failure of the transfer.
- **getDepositsRequest message:** Sent by the selling entity to the currency issuer. When the selling entity expects funds to be transferred to it from another entity, it sends this message to the currency issuer.
- **getDepositsResponse message:** Sent by currency issuer to selling entity; informs seller of the amounts that have been transferred into its account since the issuer last replied to a getDepositsRequest message.

We will describe these four message types in greater detail in Section 3.

Figure 1 illustrates the basic components of the lightweight currency. In this figure, we use the names Alice, Bob and Claire as synonyms for three public keys. The scenario is that Alice buys something from Bob with currency issued by Claire, which we refer to as Claire dollars.

Suppose that Alice initially holds 100 Claire dollars. This means that Claire maintains an account for Alice in which 100 units are recorded. Also suppose that Alice wants to transfer 10 Claire dollars to Bob. The following steps are taken:

1. Alice establishes a secure connection with Claire, and sends a transfer funds request message that instructs Claire to transfer 10 units to Bob.
2. Claire debits Alice's account by 10 and credits Bob's account by 10. Claire returns a transfer funds response message to Alice.
3. Bob then establishes a secure, authenticated connection with Claire, and sends a get deposits request message.
4. In a get deposits response message, Claire returns a list of deposits made to Alice's account.

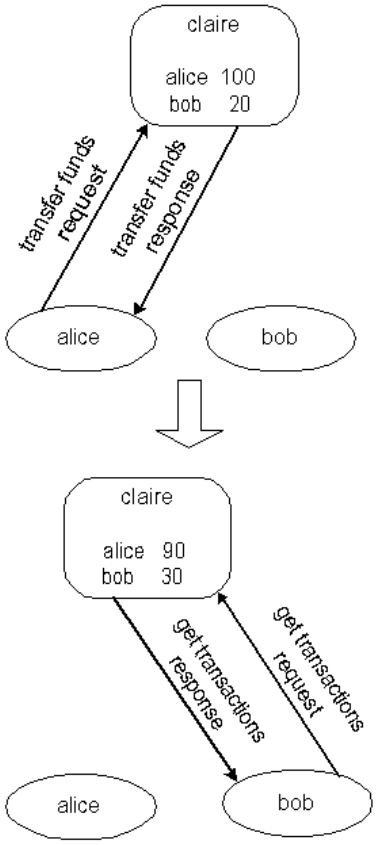

Figure 1: Alice transfers 100 Claire dollars to Bob

After transmitting the list, Claire may delete these deposit records in order to conserve space.

As previously discussed, lightweight currency is not intended to function as a general replacement for real-world currencies, but as an enabler of applications in which real-world currencies are impractical. In particular, it supports micro-payment based systems. In these systems, entities may fail to uphold contracts. In many cases, these failures are related to factors outside their control, such as network reliability. In this case, failure is not translated into legal recourse, but rather translated into diminished reputation and corresponding lowering of prices for service contracts (or complete loss of business). As an example, consider a market in which storage contracts are sold. These contracts provide the right to the buyer to write or read bytes at a rate of x dollars per megabyte over a period of 1 month. Suppose that it is believed that A1A1 fails to provide contracted read and write services with probability 0.25, and that B2B2 and C3C3 each fail with probability 0.5. In this case, purchasing service from B2B2 and C3C3 will provide a combined failure rate equal to purchasing service solely from A1A1. Thus, a fair pricing scheme would set the combined price for contracts from B2B2 or C3C3 to be equal to the price for a contract from A1A1. The lightweight currency encourages this type of probabilistic pricing rather than dispute resolution accomplished through legal recourse.

3 Lightweight Currency Protocol

In this section we provide an overview of LCP, which is at the core of P2P resource market paradigm. The complete specification of the protocol can be found at [35].

Although lightweight currency is not real-world money, it nevertheless has value, and so it is appropriate for lightweight currency protocol messages to be transported over a secure channel that protects from known communication threats. Our analysis reveals that the use of SSL/TLS can be used to accomplish this in an efficient manner.

In SSL, the client first establishes a TCP connection with the server, and then the two ends execute a handshake sequence in which an SSL session is established. The session is comprised of various values needed for encryption and decryption and other values needed for validating message integrity. Because this handshake operation is computationally expensive (involving public-key cryptography), SSL provides a mechanism whereby the client may resume the session across TCP connections. Because a holder of a currency may frequently exchange messages with the currency issuer, it is therefore advantageous for entities to maintain records of their past sessions, and for the currency holder to resume sessions with the currency issuer rather than establish new ones. In fact, currency issuers can encourage this behavior by charging higher transaction fees for transactions that do not resume sessions.

We choose to use a SOAP message format, because of its broad support across the major development environments. Thus, lightweight currency can be easily integrated into new and existing applications with the aid of web service development tools. Additionally, the protocol can evolve more easily, because an XML message format is very flexible.

Another fundamental design choice was whether to use a push or pull method for entities to be informed of the arrival of funds in their accounts.

- Under the **push approach**, currency issuers function as clients to interfaces exposed by their currency holders. When a currency issuer receives a request to transfer funds from a buying entity to a selling entity, it connects to the selling entity and sends a notification that a deposit was made to its account.
- Alternatively, under the **pull approach**, the issuer queues deposit notifications for its currency holders, and waits for these entities to connect to the issuer and request them.

The push approach to deposit notifications has a serious disadvantage when compared to the pull approach – it requires currency holders to expose an interface that is locatable by currency issuers. In the P2P resource market, many entities are not behind

static IP addresses, and therefore a dynamic lookup service would be needed to map public key identifiers to IP addresses. With the pull approach to retrieveal of deposit notifications, currency issuers always play the role of server in messaging operations, eliminating the need for a specialized look up service, and simplifying the design requirements for client software.

Under the pull approach, clients need to locate the interfaces of currency issuers who are identified by public key identifiers. Rather than implement a specialized lookup service to do this, it is possible to rely on the existing DNS lookup service. Therefore, currency issuers would be identified by domain names that are bound to their public keys by trusted certificate authorities. Simple currency holders, on the other hand, only function as clients in communication channels, and thus would not need to publish a domain name for lookup purposes. Thus, currency holders can operate without acquiring a certificate, and simply use unsigned or self-signed public keys.

Currency issuers that use a certificate to bind their public key to a domain name publish a Web Services Description Language (WSDL) document called lcp.wsdl that describes their Web Service interface. Clients retrieve the WSDL document by requesting lcp.wsdl through port 80 of the issuer's domain name using unencrypted HTTP. The WSDL document will conform to generally accepted standards for whatever version of the LCP the issuer is using, however, individual currency issuers have the freedom to bind the service to a URL of their choosing. In other words, all parts of the WSDL document will be identical across all issuers except for the service element, which specifies a concrete binding of the interface to a specific URL.

In the following, we explain the elements of messages used by currency holders to transfer funds to other entities by way of an example. In these protocol examples, we exclude the containing SOAP envalope and related details.

Suppose that Alice and Bob are two entities that have negotiated some type of purchase, and it is time for Alice to make a payment to Bob. In their negotiation, both sides agreed to a payment of 100 money.com dollars, that is, for 100 units of the currency issued by the entity whose Web service interface is defined in http://money.com/lcp.wsdl. Suppose that Alice has connected with money.com previously, but Bob has not. In this case, Alice resumes her SSL session with money.com, and submits the following request to transfer funds to Bob.

```
<transferFundsRequest>
  <recipient>B0CF...7E24</recipient>
  <amount>100</amount>
  <transactionId>12345678</transactionId>
  <messageCounter>3398</messageCounter>
</transferFundsRequest>
```

The recipient element contains Bob's public key identifier. If money.com does not have an account for this public key, money.com will create the account after receiving the message, and will set its balance to 100. The transactionId field contains an identifier that Alice and Bob agreed to during their negotiations; it provides a way for Bob to associate the transaction with Alice when he retieves deposit notifications from money.com.

The purpose of the message counter in the transferFundsRequest message is to allow retransmission of messages in the case that network failure results in the sender's inability to determine whether a transfer message has arrived or not. The sender can retransmit transferFundsRequest messages without fear that the currency issuer will execute a given transfer request more than one time. In this case, currency issuers must record the state of the message counter for each currency holder, and discard redundant messages.

Money.com returns the following LCP response to the previous transferFundsRequest message:

```
<transferFundsResponse>
  <result>success</result>
  <fee>1.2</fee>
</transferFundsResponse>
```

Money.com will debit Alice's account by 101.2 units, and credit Bob's account by 100 units. After this operation is performed, money.com queues a deposit notification, which is delivered to Bob when he requests it. In the case that Bob has never connected with money.com before, he will initiate an new SSL

session with money.com, and send the following getDepositsRequest message to obtain a list of deposits made to his account:

```
<getDepositsRequest>
  <millisecondsWait>3000</millisecondsWait>
</getDepositsRequest>
```

Because Alice and Bob may send their messages at the same time, it is possible that Bob's request for deposit notifications arrive before Alice's transfer funds request. For this reason, Bob specifies a non-zero millisecondsWait value that tells money.com to wait at least that many milliseconds before returning an empty list of deposits. Money.com returns the following response to Bob:

```
<getDepositsResponse>
  <deposits>
    <deposit>
      <transactionId>1234</transactionId>
      <amount>100</amount>
    </deposit>
  </deposits>
  <fee>0</fee>
  <balance>9200</balance>
</getDepositsResponse>
```

The getDespoitsResponse contains three child elements: an array of deposit elements, a fee element, and a balance element. In this case, there is only a single deposit, which contains the amount deposited and the transactionId that Bob uses to associate the funds arrival event with Alice. The fee element contains any fee imposed by the currency issuer for processing a getDespositsRequest; a non-zero fee can be used to recoup the cost of processing these messages. Finally, the balance element reports the balance of funds in Bob's account.

It should be noted that the protocol is still in experimental stages, and that at this point it has been designed to ease implementation. For this reason, we rely on SSL/TLS rather than invent a special purpose security protocol. Also, rather than use the more flexible document-stlye SOAP message mechanism, we chose to use the RPC-style mechanism, because of greater support and programmer familiarity. The only optimization consideration that has been taken into account is to use SSL session resumption to reduce encryption/decryption overhead.

4 P2P Resource Market Examples

An important distinction is between raw resources and derived services. Raw resources include storage, bandwidth and compute power. These raw resources provide the building blocks for all other tradable services, such as data backup, multimedia streaming, proxy services, lookup services, instant messaging, software distribution, media distribution, etc. Therefore, a resource market in which raw resources are traded is possible. Derived services can thus be produced using the resources provided through the raw resource market.

Raw Resource Market

One possibility for building the raw resource market is to use a basic storage contract that grants to the buyer the right to read and write bytes at a specified price into a block of memory for an interval of time. As an example, suppose Bob sells to Alice a storage contract with the following specifics:

> In exchange for 10 Claire dollars, Bob grants to Alice 1 megabyte block of memory into which Alice can write bytes in exchange for 1 Claire dollar per kilobyte, and out of which she can read bytes at a rate of 4 Claire dollars per kilobyte. This agreement is valid for the period beginning at 12 noon GMT 12-jul-2003 and ending at 12 noon GMT 26-jul-2003.

Notice that this contract makes no guarantees on data transmission rates. We assume that Alice and Bob have an ongoing relationship, and that Alice maintains a history of read and write transmission rates from/to Bob. Alice uses her assessment of Bob's quality of service when deciding how much to pay for storage contracts with Bob.

Alice can use her contracted rights for various applications. For example, she can store data with Bob for backup purposes. As another example, Alice could be a content publisher, and she uses her storage contract with Bob to distribute content. Thus, Alice does not need to be a consumer of Bob's specific services, because the currency they use in their transactions is fully transferable and not tied to a specific application. This provides in peer-to-peer systems a natural, market-driven mechanism to allocate resources to applications.

P2P Content Distribution

When an entity sells something, it accepts payment in a currency that it deems worthwhile. For example, suppose that news.com sells news stories in video format with prices in a currency issued by money.com. Alice desires to view these videos, and so she configures her storage sales service to accept payments in money.com dollars. In turn, news.com may use money.com dollars it earns to purchase storage contracts from Alice, in order to deliver the video from Alice to other lightweight currency paying customers.

To bootstrap this type of system, money.com could construct a storage selling/video player system that users could download and install in their systems. Then, money.com sells video delivery software to news.com and other video content publishers. These relatively small-scale content publishers can then distribute their content over a peer-to-peer delivery system that dynamically scales to accommodate any number of systems with minimal impact on bandwidth resources of the content publishers. With valued money.com dollars in circulation, any number of other services can be sold in exchange for these dollars, or for other competing currency.

P2P Distributed Archival Storage

Lightweight currency can be used as the basis of a P2P distributed storage archival service. In such a system, entities purchase storage contracts in the raw resource market in order to replicate their archives across geographically distributed nodes. Such an approach is a means to preserve data from loss due to catastrophic events, such as fire, flooding, earthquake, war, etc. in which locally stored backups are destroyed along with the original data.

Note that improved efficiency is possible with the use of a transferable currency. For example, suppose that two P2P applications operate as closed systems. In one system, users stream video on demand from a news service; in the other system, users store large data sets for long-term archival purposes. In the content delivery application, the scarce resource is likely to be bandwidth, while the scare resource in the archival system is likely to be storage. If both systems were to buy and sell resource contracts in the raw resource market, greater economies of scope result.

Spam Reduction

Another possible application of lightweight currency is to reduce spam by requiring a payment for the delivery of email. The use of payments is an acknowledged solution to the spam problem [ref]. Because lightweight currencies are fully transferable between entities and not tied to any specific application context, they are a natural candidate to be used in payment-based control of spam.

Derivative Services

Because many computers are assigned IP addresses dynamically, peer-to-peer applications that require the user to expose an interface (for example, SIP applications [SIP]) require the user be locatable through a lookup service. Such a lookup service could charge fees in (lightweight dollars) to provide an economic incentive for providing the service.

Another likely derivative service is banking. With banking, a user relies on a trusted service to manage financial transactions on its behalf. This bank would provide a Web interface for users to check received payments and their balances, and also to transfer payment to other entities.

5 Application Interfaces with LCP

In general, in order for entities to participate in markets that rely on lightweight currency, entities will need applications that earn currency along with applications that spend currency. There are special cases when this may not be true, such as users that purchase lightweight currency with real-world money for the purpose of spending through a single application. Another special case is when an entity uses only currency it issues to purchase services, and does not provide any service for which the currency is redeemable. However, in this section we will focus on those entities that wish to earn currency through the sale of one or more services, and spend currency through one or more applications. As we explain below, LCP is sufficiently flexible to allow for a variety of application interfaces.

Suppose an entity has multiple applications (for example, a raw storage application and a raw CPU cycle application) that are engaged in selling and buying activities. The applications that spend money should be supplied by the applications that earn money. There are two broad approaches to solving this integration problem: one based on a *single identity*, and another based on *multiple identities*.

Under the multiple identity approach, each application creates a public key identity. An application that earns currency has that currency deposited into accounts that are associated with their identity. An application that spends money also generates a public key for the purpose of receiving money from the earning applications, and then spending this money for services provided by the applications of other entities. For this system to work, users must either manually instruct earning applications to transfer money to spending applications, or to specify polices for these systems to follow for automatic transference of funds from the accounts or earners to the accounts of spenders.

For the remainder of this section we focus on the single identity approach. Under this approach, a single software agent is used to manage all LCP interactions with other entities. There are two variations on the single identity approach: one that relies on a local LCP agent, and another that relies on an external LCP agent that we refer to as a bank service.

Under the local LCP agent approach, the user installs agent software to communicate with currency issuers. In this case, the local LCP agent must expose a local interface through which local application software can communicate.

5.1 Bank Services

Finally, the user can use a **bank service** to coordinate earning and spending of currency. In this approach, the user relies on a trusted service to manage its accounts and financial transactions on its behalf. This bank would provide a Web interface for users to manually control their accounts, or configure the service so that user agents (such as media players, raw resource sales, archival systems, online games, etc.) would be given appropriate rights to control funds in the account. Such a service is useful for users that wish to consolidate their lightweight finances under a single system, rather than letting applications generate their own lightweight currency identities. It is also useful for users that desire to access lightweight currency dependent application functionality from more than one computer. Also, the user can safely manage his identity and currency holdings without fear that a local system crash will result in the permanent loss of his private key, and thus his established identity.

There are two basic types of banking service: transparent and opaque banking. In **transparent banking**, the user's identity is a unique public key; however, now the bank service has the responsibility of storing and securing the user's associated private key. In **opaque banking**, the user's identity is comprised of the bank's public key identifier plus a name unique among the set of bank customers. For opaque banking, the user does not need a public/private key pair for LCP. For opaque banking, the bank can use a trusted certificate authority to bind its public key to its URL. Thus, for example, if Bob has an account with bank.com, and his identifier with bank.com is bob, then Bob's trading partners can input the user-friendly name bob@bank.com into their application

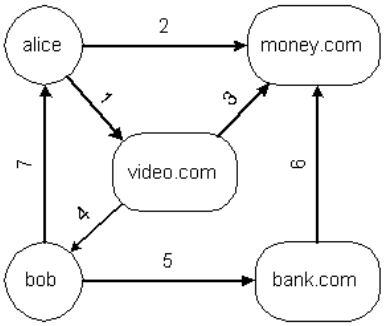

Figure 2: Example with bank

interfaces to identify Bob and locate the interface through which funds can be delivered to Bob. Additionally, this simplifies the method by which Bob configures his application software, because rather than inputting a long public key, he simply inputs the bank's URL, and Bob's logon id and password with bank.com.

5.2 Comprehensive Example

Now, we wish to illustrate with a concrete example how the aforementioned systems might interact for a typical scenario. The scenario is that user Alice wants to view a video available through video.com. Alice has a media player that communicates with an LCP agent that runs on her own system. Alice earns currency by selling storage and bandwidth in the raw resource market.

Bob also sells storage in the raw resource market. However, unlike Alice, Bob uses the bank service located at bank.com. In our example, video.com has a basic storage contract with Bob, and that under this contract, Bob is currently storing the video that Alice is requesting.

We assume for this example that Alice, Bob and video.com are using money issued by money.com.

Figure 2 illustrates the interactions between entities. Alice begins by connecting to video.com (1), and negotiating for the purchase of her desired video. After both ends come to agreement regarding price, Alice next connects to money.com and requests that funds be transferred to video.com (2). At the same time, video.com connects to money.com and requests a statement of account activity (3). Incorporated into this request message is an element labeled max-wait-time, which is an instruction to money.com to not return an empty list until waiting the specified period of time. This max-wait-time element is used in case the seller's request reaches the issuer before the issuer has processed the transfer funds request from the buyer.

After video.com obtains notification of the deposit of funds from Alice, it requests that money.com transfer funds to Bob. The amount of funds requested in this case will equal the read price specified in the storage contract that video.com purchased from Bob. The message that video.com constructs for this transfer operation will include the public key for bank.com and a userid element that contains the identifier bob, which identifies Bob's bank account.

Now, video.com is prepared to execute its read rights with Bob. To do this, video.com connects to Bob, and requests retrieval instructions (4). In response to this request, Bob connects to his bank to check that the payment for the read operation has arrived (5). In turn, bank.com connects to money.com to request recent account activity (6). After money.com informs bank.com that the expected funds have arrived, bank.com passes on the confirmation to Bob. Now, Bob responds to video.com's read request by sending retrieval instructions.

After obtaining the retrieval instructions, video.com returns them to Alice. Finally, Alice connects to Bob, and uses the retrieval instructions to retrieve the video (7).

If Bob were using a transparent, instead of an opaque, banking service at bank.com, then the example would have the same number of steps. The only change would be Step 4, in which the message from video.com to money.com would omit the userid element (and the bank's public key), and simply provide a public key that represents Bob.

6 Scalability

Perhaps the principle advantage of designing an application as a peer-to-peer system is its potential to scale to any number of nodes. If the lightweight currency paradigm is to be used in support of peer-to-peer applications, then its scalability ought to be evaluated. We identify several possible solutions to the scaling problem.

If systems are designed to accept an increasing number of currencies, then it is possible that currency policies can be arranged so that the number of currencies in circulation increases in proportion to increases in the number of new nodes. In this case, currency issuers will need to handle approximately the same average number of transactions. However, if pair-wise relationships occur between randomly selected nodes, then negotiations between nodes will require settling on a currency from a growing list of currency providers. However, in the case that trading relationships between nodes are relatively stable, then this complexity will not be an issue.

If systems (or their users) do not accept an increasing number currencies as new nodes are added to the system, then one or a few large currency issuers will dominate. In this case, there must be economic incentives for these issuers to exist. If users are willing to pay for lightweight currency, then large-scale issuers can recoup their expenses (or make a profit) for creating and operating the physical infrastructure to support a large-scale operation. If on the other hand users are unwilling to purchase lightweight currency from large issuers, then an issuer would be motivated to operate a currency only to the extent that it provides it with services it needs. If the demand for the issuers currency exceeds levels needed by the issuers to fulfill its needs, then there will be a shortage of this currency in the market place, and peers will be forced to use other available currencies to complete transactions. In this way, the market is forced to accept multiple currencies. Because issuing a currency is straightforward, a general shortage of currency will incite peers to issue currencies in place of providing other services as a means to establish purchasing power in resource markets.

7 Related Work

Many digital cash schemes were proposed in the mid 1990s, including Digicash, CyberCash and First Virtual [11] [37]. Because of their complexity – including public-key certificates, serial numbers, and blinded signatures – these protocols are inappropriate for the P2P resource market. Moreover, because of poor timing (mid 1990s) and system complexity, the startups behind most of these digital cash schemes have failed. One digital cash startup, Paypal [24], is successful. Paypal's currency is frequently used by eBay buyers and sellers. However, Paypal is not suitable for the P2P resource market. It collects a hefty commission from sellers, and users need to link their paypal accounts into their checking accounts or credit cards, which gets too close to the user's real money. Instead, we are proposing a scheme that is distinct from real money, allowing users to more comfortably engage in automated resource sharing markets.

There are dozens of P2P file sharing projects to date (for example, see [27, 15, 10, 21]). Most of these projects focus exclusively on file sharing and do not extend to P2P content distribution, backup storage, and grid computing. The now defunct MojoNation attempted to integrate currency into P2P file sharing[21]. We believe that MojoNation failed because it was a profit-driven enterprise rather than an open-source driven project, it focused on file swapping rather than a generic P2P resource market, and it employed a relatively complex currency model. As discussed earlier, Kazaa has implicitly adopted a currency scheme to incite users to upload files. Kazaa's currency shares some characteristics with our own - for example, neither are tied to real-world money or are legal tender. However, the Kazaa currency is limited to the Kazaa application; the currencies proposed in this paper are issued by arbitrary entities and transferred among applications.

Cooper and Garcia-Molina [5] have recently studied bidding and bartering for storage space in a P2P data archival system. Although their approach uses a pairwise transaction model, users transact directly in storage units rather than in currency units. The paper [5] does not explore broader resource cooperatives that include transmission bandwidth and com-

putation as resources, and the paper does not develop a currency model (lightweight or heavyweight). Furthermore, the focus of the work is on simulating different bidding mechanisms rather than developing currency protocols and prototypes.

Many researchers are currently exploring distributed hash tables (DHTs) for building novel P2P systems and applications [26, 32, 28, 39]. We briefly remark that DHTs and LCP can be coupled. For example, a DHT-based archival storage system can be built in which the data behind the keys does not contain the files themselves but instead pointers to the files. When a user wants to replicate a file (for archival purposes), the user transacts in the P2P resource market to establish a contract with a partner; once the file is replicated in the partner's peer, the corresponding pointer file in the DHT location service is updated to point to the new copy.

Recently, there has been a burst of activity in P2P economics. A number of research groups are studying reputation systems for P2P systems [7, 14, 3, 19, 22, 8, 13]. This work in P2P reputation systems complements our own: because the currencies provide no legal recourse, assurances need to be provided through other mechanisms, such as reputation, trust, and transaction replication.

There has also been a flurry of activity concentrated on incentives in P2P systems. Some of this work argues for using currency to incite peers to contribute resources. The KARMA framework uses a single currency and has been designed to integrate with a DHT infrastructure; in particular, the accounting and banking are distributed over the P2P nodes within the DHT [36]. Part of our vision is that the currency infrastructure (that is, currency issuers and banks) is to be implemented on dedicated servers, whereas the resources are to be traded directly between pairs of peers. Such a hybrid P2P/client-server architecture has also been used in instant messaging (where buddies are located with dedicated servers) and Napster (which maintained a centralized directory of users' MP3s) [18]. As with Web-based e-mail (e.g., Hotmail and YahooMail), we expect major currency issuers to scale by adding more servers to server farms. A coin-based micropayment system that has been designed for P2P systems is presented in [38]. The scheme uses legal recourse, and its target application scenario is the sale of content. Other recent papers investigating incentives in P2P applications include [25, 9, 4, 17, 31]. Game-theoretic models of micro-payment schemes are presented in [12, 19].

8 Future Work

In this paper we introduced a lightweight currency paradigm and protocol for P2P resource and service trading. Using this paradigm and protocol as a foundation, we are currently exploring several avenues of research. We are implementing the LCP in several applications, including a peer-driven content streaming system and an archival storage system. We are investigating the implementation of scalable currency issuers and LCP banks. We are examining alternative applications of lightweight currency, including spam reduction [34, 33]. We are developing a theory for optimal selection of service providers based on price and estimated reliability. We are also constructing a Web site for developers interested in participating in the lightweight currency project.

References

[1] E. Adar and B. Huberman. Free Riding on Gnutella, October 2000. First Monday, http://www.firstmonday.dk/issues/issue5_10.

[2] D. Anderson and J. Kubiatowicz. The Worldwide Computer. *Scientific American*, 286:28–35, March 2003.

[3] A. Asvanund, S. Bagla, M.H. Kapadia, R. Krishnan, M.D. Smith, and R. Telang. Intelligent Club Management in Peer-to-Peer Networks. In *Workshop on Economics of Peer-to-Peer Systems*, jun 2003. Papers published on Web site: http://www.sims.berkeley.edu/research/conferences/p2pecon/index.html.

[4] B. Chun, Y. Fu, and A. Vahdat. Bootstrapping a Distributed Computational Economy. In *Workshop on Economics of Peer-to-Peer Systems*, jun 2003. Papers published on Web

site: http://www.sims.berkeley.edu/research/conferences/p2pecon/index.html.

[5] B.F. Cooper and H. Garcia-Molina. Bidding for storage space in peer-to-peer data preservation systems. In *Proceedings of Multimedia Computing and Networking 2002 (MMCN'02)*, January 2002.

[6] F. Dabek, , F. Kaashoek, R. Morris, D. Karger, and I. Stoica. Wide-Area Cooperative Storage with CFS. In *Proceedings of ACM SOSP'01*, Banff, Canada, October 2001.

[7] C. Dellarocas and P. Resnick. Online Reputation Mechanisms: A Roadmap for Future Research. In *Workshop on Economics of Peer-to-Peer Systems*, jun 2003. Papers published on Web site: http://www.sims.berkeley.edu/research/conferences/p2pecon/index.html.

[8] D. Dutta, A. Goel, R. Govindan, and H. Zhang. The Design of a Distributed Rating Scheme for Peer-to-Peer Systems. In *Workshop on Economics of Peer-to-Peer Systems*, jun 2003. Papers published on Web site: http://www.sims.berkeley.edu/research/conferences/p2pecon/index.html.

[9] M. Feldman, K. Lai, J. Chuang, and I. Stoica. Quantifying Disincentives in Peer-to-Peer Networks. In *Workshop on Economics of Peer-to-Peer Systems*, jun 2003. Papers published on Web site: http://www.sims.berkeley.edu/research/conferences/p2pecon/index.html.

[10] The Gnutella Protocol Specification. Available from http://gnutella.wego.com.

[11] S. Godin. *Presenting Digital Cash*. Sams, Indianapolis, 1996.

[12] P. Golle, K. Leyton-Brown, I. Mironov, and M. Lillibridge. Incentives for Sharing in Peer-to-Peer Networks. In *Proc. of the 2001 ACM Conference on Electronic Commerce*, 2001.

[13] B. Gross and A. Acquisti. Balances of Powery on eBay: Peers or Unequals. In *Workshop on Economics of Peer-to-Peer Systems*, jun 2003. Papers published on Web site: http://www.sims.berkeley.edu/research/conferences/p2pecon/index.html.

[14] S.D. Kamvar, M.T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the Twelfth International World Wide Web Conference*, 2003.

[15] KaZaA Web Site. http://www.kazaa.com.

[16] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, , and B. Zhao. OceanStore: An Architecture for Global-Scale Persistent Storage. In *Proceedings of the Ninth international Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000)*, November 2000.

[17] S. Kumvar, B. Yang, and H. Garcia-Molina. Addressing the Non-Cooperation Problem in Competitive P2P Systems. In *Workshop on Economics of Peer-to-Peer Systems*, jun 2003. Papers published on Web site: http://www.sims.berkeley.edu/research/conferences/p2pecon/index.html.

[18] J.F. Kurose and K.W. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet*. Addison-Wesley, Boston, MA, 2002.

[19] K. Lai, M. Feldman, I. Stoica, and J. Chuang. Incentives for Cooperation in Peer-to-Peer Networks. In *Workshop on Economics of Peer-to-Peer Systems*, jun 2003. Papers published on Web site: http://www.sims.berkeley.edu/research/conferences/p2pecon/index.html.

[20] Cedar rapids website. http://www.cedar-rapids.org/ushers.

[21] Mojonation web site. http://www.mojonation.com.

[22] T. Moreton and A. Twigg. Trading in Trust, Tokens and Stamps. In *Workshop on Economics of Peer-to-Peer Systems*, jun 2003. Papers published on Web site: http://www.sims.berkeley.edu/research/conferences/p2pecon/index.html.

[23] A.M. Odlyzko. The history of communications and its implications for the Internet. Available online at http://www.research.att.com/ãmo, 2000.

[24] Paypal web site. http://www.paypal.com.

[25] K. Ranganathan, M. Ripeanu, A. Sarin, and I. Foster. To Share or Not to Share. In *Workshop on Economics of Peer-to-Peer Systems*, jun 2003. Papers published on Web site: http://www.sims.berkeley.edu/research/conferences/p2pecon/index.html.

[26] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proceedings of the 2001 ACM SIGCOMM Conference*, 2001.

[27] K.W. Ross and D. Rubenstein. P2P Systems: Infocom 2003 Tutorial. http://cis.poly.edu/ ross/papers.

[28] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, November 2001.

[29] A. Rowstron and P. Druschel. Storage Management and Caching in PAST, A Large-scale, Persistent Peer-to-peer Storage Utility. In *Proceedings of ACM SOSP'01*, Banff, October 2001.

[30] S. Saroiu, P. Gummadi, and S.D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Proceedings of International Conference on Distributed Computing Systems*, 2002.

[31] J. Shneidman and D. Parkes. Rationality and Self-Interest in Peer-to-Peer Networks. In *Proceedings of 2nd Int. Workshop on Peer-to-Peer Systems*, feb 2003.

[32] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, 2001.

[33] D. Turner and Ni Deng. Payment-based email. submitted.

[34] D. Turner and D. Havey. Controlling spam through lightweight currency. In *Proceedings of the Hawaii International Conference on Computer Sciences*, January 2004.

[35] D.A. Turner and K.W. Ross. Lightweight Currency Protocol. Internet Draft, September 2003. Expires 2004.

[36] V. Vishnumurthy, S. Chandrakumar, and E. Gun Sirer. KARMA: A Secure Economic Framework for Peer-to-Peer Resource Sharing. In *Workshop on Economics of Peer-to-Peer Systems*, jun 2003. Papers published on Web site: http://www.sims.berkeley.edu/research/conferences/p2pecon/index.html.

[37] P. Wayner. *Digital Cash: Commerce on the Net*. Moran Kaufmann, San Francisco, 1995.

[38] B. Yang and H. Garcia-Molina. PPay: Micropayments for Peer-to-Peer Systems. In *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS)*, oct 2003.

[39] B. Y. Zhao, J. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, UC Berkeley, 2001.

David A. Turner

David Turner received a Ph.D. in Computer Science from the Eurecom Institute, Sophia-Antipolis, France in June, 2001. In August, 2001, he joined the Computer Science Department at California State University San Bernardino (CSUSB), where he holds the position of assistant professor. Prior to his Ph.D. research, he worked as a transportation planner and software developer. At the Eurecom Institute, he focused on the area of multimedia messaging. Since joining CSUSB, his focus has shifted to micro-payment-based ecommerce and the software engineering of Web applications. Additionally, he lead the creation of a new degree program at CSUSB leading to a Bachelor of Arts in Computer Systems, which emphasizes software development in an interdisciplinary setting.

Keith W. Ross

Professor Ross joined Polytechnic University as the Leonard J. Shustek Professor in Computer Science in January 2003. Before joining Polytechnic University, he was a professor for five years in the Multimedia Communications Department at Eurecom Institute in Sophia Antipolis, France. From 1985 through 1997, he was a professor in the Department of Systems Engineering at the University of Pennsylvania.
Professor Ross's research interests are in computer networking, content distribution, peer-to-peer networking, video streaming and performance modeling. Professor Ross is co-author (with James F. Kurose) of the best-selling textbook, Computer Networking: A Top-Down Approach Featuring the Internet, published by Addison-Wesley He is also the author of the research monograph, Multiservice Loss Models for Broadband Communication Networks, published by Springer in 1995.