Credit transfer within market-based resource allocation infrastructure

Tyler Close
HP Laboratories Palo Alto
HPL-2006-5
January 9, 2006*

market-based,
resource allocation,
credit transfer,
agoric,
strategyproof, IOU,
smart-contract

Credit transfer protocols are being designed and implemented in market-based resource allocation infrastructures. The design details of these protocols determine the access-control policies that can be expressed and the trading patterns that can be supported. A protocol that enables fine grained manipulation of ownership authorities can support a wider range of access policies and trading patterns than proposed designs, while reducing implementation complexity. Meeting the challenges of a market-based resource allocation infrastructure may yield a credit transfer protocol that is applicable in other domains.

* Internal Accession Date Only
An abridged form of this paper will be published in the Financial Cryptography Conference 2006, 27 February to 2 March, 2006, Anguilla, British West Indies

Approved for External Publication

Credit transfer within market-based resource allocation infrastructure

Tyler Close
Hewlett-Packard Laboratories, Palo Alto
tyler.close@hp.com

Abstract

Credit transfer protocols are being designed and implemented in market-based resource allocation infrastructures. The design details of these protocols determine the access-control policies that can be expressed and the trading patterns that can be supported. A protocol that enables fine grained manipulation of ownership authorities can support a wider range of access policies and trading patterns than proposed designs, while reducing implementation complexity. Meeting the challenges of a market-based resource allocation infrastructure may yield a credit transfer protocol that is applicable in other domains.

Introduction

A rich body of research explores the use of market-based mechanisms for resource allocation within computing infrastructure. Aiming to reproduce the scalability and efficiency of market-based resource allocation within the general economy, this research proposes analogous institutions for allocating CPU cycles, memory, bandwidth, etc. within a computer, a data center, an intranet, or even the Internet. Participants in these infrastructures interact through the exchange of ownership claims, varyingly referred to as: money, tokens, tickets, claims, rights and here referred to as credits. Some designs propose a new credit transfer protocol, whereas others point to existing e-cash cryptographic protocols. In each case, the choice of protocol influences the features and construction of the proposed resource allocation mechanism.

This paper studies the impact of the credit transfer protocol on the design and resulting features of proposed market-based resource allocation infrastructures. This analysis focuses on features identified in the research literature as crucial to the success of such infrastructure. The common theme among these features is facilitating beneficial trade. Description of the features follows.

Strategyproof

From seminal works [1] to current implementations [2][3][4][6][16], the vision for market-based resource allocation is one for managing an ecosystem of self-interested parties. Such parties will exercise infrastructure protocols to their advantage, even if this strategy is to the detriment of others. The challenge is therefore to design *strategyproof* [5] protocols that enable cooperation without vulnerability.

Universal

Just as in the general economy, different kinds of goods in a computational economy are more efficiently allocated by different kinds of auctions. Moreover, there is continuing research into the optimum kind of auction for any particular good and so competing services are to be expected and must be accomodated. An *open* infrastructure [1][5], requires a credit transfer protocol free from higher-level policy assumptions [13]. Such a universal protocol enables the deployment and composition of unforeseen services through credit transfers [8].

Transparent

Simulation of market-based resource allocation mechanisms has revealed that widespread access to comprehensive price information is key to market efficiency [2], much as it is in the general economy. A credit transfer protocol that ensures collection of price information helps create market *transparency*.

Liquid

Participants in a market-based infrastructure react to changing requirements or market conditions by trading. Timely trading requires market liquidity. Creating market liquidity requires fungible assets [5] and support for arbitrageurs [8].

Controlled

As network resource allocation infrastructures, such has PlanetLab [21], have grown, demand has evolved for segregation within the network. Some providers do not wish to service any client, but also do not wish to be isolated from the rest of the network. Participants have a need to control their potential counter-parties in transactions. In another form of control, data center resource allocation infrastructures have developed with defined consumer versus merchant roles that underlie the business plan. Providing protocols that preserve the autonomy of resource providers is crucial to securing their participation [5]. As demonstrated by industry experience with SMTP [9], it is often not feasible to later add control features that a protocol was not designed to support.

Simple

Simplicity is a subjective requirement when compared to the previously listed ones; however, its importance to a successful infrastructure cannot be overlooked. Reducing coordination costs between participants is crucial to creating a broadly inclusive infrastructure [5].

The preceding feature list is an amalgam of requirements set forth in a number of papers [1][2][3][4][5][6][8], including all those presented in [5]. The review of proposed credit transfer protocols does not yield one that satisfies all these constraints. The second half of this paper presents a protocol that does satisfy all these constraints.

Market-based resource allocation infrastructures operate in a rich scope encompassing mutually suspicious participants and the creation and trade of new goods. Many infrastructures describe themselves as a 'computational economy' and are worthy of the term. The richness of these infrastructures makes them especially interesting to study, since it may be hoped that the credit transfer design will be applicable in other domains facing similar challenges.

Proposed credit transfer mechanisms

SHARP

SHARP [3] is a framework for secure distributed resource management in an Internet-scale computing infrastructure. A SHARP prototype manages access to virtual machines in PlanetLab [21], a federation of servers across the Internet.

The medium of exchange in SHARP is the *ticket*. A ticket is a certificate chain, where each certificate in the chain is referred to as a *claim*. A claim is a signed assertion that a principal, identified by a public key, may be granted access to a *resource set* during a specified time interval. A resource set is a number of units of a specified type; for example, a number of virtual machines.

The principal identified by a claim may delegate access to another principal by producing a new ticket containing an additional claim asserting this delegation. Double spending, or generating multiple tickets delegating the same access claim, is explicitly allowed. This policy, called *oversubscription*, improves resource utilization.

SHARP principals interact by requesting and trading claims for resource sets. In theory, a principal, Bob, with access to a virtual server during a future timeslice could trade this asset with another principal, Carol, with access to a virtual server during a present timeslice. Unfortunately, the details of the SHARP protocol complicate this exchange.

Appraisal complexity

In judging the value of the trade, Bob must consider not only the value of the two timeslices, but also the degree to which Carol is oversubscribing her timeslice. This information is not available to Bob at the time of the exchange and so must be a guess based on information provided by Carol, and Carol's reputation, as seen by Bob. The requirement to dynamically track the reputation of potential trading partners is an impediment to trade.

Loss of fungibility

Since Carol may oversubscribe her claim, the timeslice offered by Carol is not interchangeable with one offered by David. The value of Carol's timeslice incorporates Carol's reputation and the value of David's timeslice incorporates David's reputation. Essentially, each claim is a new brand of currency with its own value proposition. Under the SHARP protocol, claims are not fungible. Fungible assets are crucial to market liquidity.

Value dilution

Completing the Bob and Carol trade results in the creation of two new tickets: one delegating Carol's claim to Bob and one delegating Bob's claim to Carol. These new claims again represent two new non-fungible brands of currency, but with more complicated value propositions. Since Bob may oversubscribe the claim delegated to him by Carol, its value to a prospective buyer is now a function of the value of the underlying timeslice, Carol's reputation and Bob's reputation. By taking possession of the asset, Bob has devalued it. Each trade of the asset further devalues it, as prospective buyers must take into account the possibility of oversubscription by an ever larger pool of principals. Devaluation of traded assets discourages participants from trading in response to changing market conditions.

Reputational burden

A new participant without an established reputation faces the daunting reality that the value of any acquired asset will immediately drop to zero, since prospective buyers have no means by which to appraise the trustworthiness of the claim. As a result, a new participant is unable to trade in response to changing requirements or market conditions.

No counter-party restriction

Since the holder of a claim may delegate it to any other principal, the resource provider has no control over the pool of principals that may redeem the claim. This shortcoming is noted in the SHARP paper:

"That is, the owner of the resource (the site authority) cannot prevent an agent from carelessly certifying some malicious entity to access the resource."

The SHARP paper claims that this risk is mitigated by the nature of the asset, a virtual machine isolated from other virtual machines. However, if the legal responsibility for a denial of service attack falls to the site authority, this design deficiency may be a showstopper. For some asset classes, counter-party restriction is an absolute requirement for deployment.

Although not discussed in the SHARP paper, one possibility is for the site authority to hold a delegator responsible for the misbehaviour of a delegatee. Such a policy would restrict trade, as a seller could only trade with buyers he is willing to vouch for. A participant with no reputation would now have difficulty buying, in addition to the previously noted selling difficulty.

Opaque markets

Using the SHARP protocol, two participants could agree on a trade and complete it as a purely bilateral operation. In this scenario, the pricing information generated by the trade is known only to the two participants. Other participants are unable to react to the lost price signal and so cannot adjust their trading activity to changing market conditions. This loss of transparency results in market inefficiency.

Tycoon

Tycoon [4] is a market-based resource allocation system targeted primarily at the intranet or data center. In a Tycoon data center, client jobs bid for resources on provider servers. Auctions on the provider servers aim to optimize resource allocation among potential clients.

Tycoon uses an account based design, where credits are transferred between accounts via signed messages exchanged with a central bank. The bank protocol assumes the existence of a PKI and roughly synchronized clocks.

A payer initiates a payment to a payee by sending a signed message to the bank consisting of: the payer identifier; the payee identifier; the amount to transfer; and a timestamp. The bank verifies the signature, checks that the message is new and that the funds are available. If so, the bank transfers the specified amount to the payee and returns a signed receipt consisting of: the payer identifier; the payee identifier; the amount transferred; and the timestamp specified by the payer. The payer forwards the receipt to the payee, who verifies: the bank's signature; the receipt is new; and the payer identifier, payee identifier and amount are as expected. If so, the payee records that payment has been made.

Although the bank protocol is specified in terms of timestamps, the protocol seems to only require a counter at the payer and a record at the payee of the last counter value presented by the payer. As such, the requirement for synchronized clocks may be ignored in evaluating the protocol.

No counter-party restriction

Since a client can request a payment to any provider, any client with sufficient funds can present a valid receipt to any provider. This shortcoming may be a problem if not all provider servers should be available to all clients.

No role division

The bank protocol does not discriminate between clients and providers, so any client may receive payments from other clients. In a utility data center deployment, it may be undesirable for a client to purchase a large block of provider resources and enter into competition with the provider. The resource provider has no ability to control the set of merchants.

Requires a PKI

The Tycoon paper claims that simplicity is a key goal and advantage of the bank protocol. Deploying and managing a PKI for a changing user population is itself a complex task that can be avoided in the design of a credit transfer mechanism. Reducing coordination costs is a key component of facilitating beneficial trade.

Payee resident double spending logic

The bank does not itself provide double spending checks, and so requires that the payee maintain the applicable state and do the applicable checks. Such a protocol is more susceptible to implementation errors when there are many different payee implementations. Eliminating this programming burden on payees further facilitates beneficial trade.

Opaque markets

Though not relevant to the current Tycoon design, the bank protocol cannot ensure market transparency. Since any participant can directly pay any other participant, it is not possible to enforce use of an intermediary that records price information.

ERTP

ERTP [6][7] is a credit transfer protocol derived from the Space Bank memory allocation protocol in the KeyKOS operating system [10]. ERTP was itself used for resource allocation in WebMart [12] deployments.

ERTP is a capability-based [11] protocol with an object-oriented API. A `Purse` object maintains a count of rights of a specified brand. For example, 10 pages of memory. Possession of a reference to a `Purse` object implies ownership of the specified number of rights. A brand of right is represented by an `Issuer` object. An `Assay` object is a type-safe description of a number of rights. For example, a query for the current balance of a `Purse` object returns an `Assay` object. Figure 1 provides the ERTP Java interface declarations.

```
interface Issuer {
    Purse makeEmptyPurse();
    Assay vouchForAssay(Assay candidate);
    Purse vouchForPurse(Purse candidate);
}
interface Purse {                    interface Assay {
    Assay depositAll(Purse src);         double compareTo(Assay other);
    Assay getAssay();                    Issuer getIssuer();
    Issuer getIssuer();                  void transfer(Purse src, Purse dest);
}                                    }
```

Figure 1: ERTP interface

A payer transfers rights to a payee by first creating a new `Purse` via `Issuer.makeEmptyPurse()`. Using `Assay.transfer()`, the payer transfers rights from his privately held `Purse` to the newly created `Purse`. The payer then passes a reference to the newly created `Purse` to the payee. The payee takes exclusive ownership of the received rights by calling `Purse.depositAll()` on his privately held `Purse`, passing the received `Purse` as the argument.

No counter-party restriction

In ERTP, the `Issuer` object embodies the authority to create a new `Purse` and thus transact in a particular brand of right. The ERTP does not support keeping the `Issuer` object private. A reference to the `Issuer` can be gotten from any `Purse` or `Assay` object; thus, any holder of rights can transfer rights to any party. Even if all holders of rights were willing to help control the pool of potential counter-parties, the ERTP does not enable them to do so, since the basic payment mechanism involves creating a new `Purse` and passing it to another participant. Under the ERTP, a resource provider has no ability to control its potential counter-parties.

No role division

Both `Purse` and `Assay` objects include the authority to receive a payment. Since the authority to hold rights includes the authority to receive payments, the ERTP does not support a role division between consumer and merchant. The resource provider has no ability to control the set of merchants.

Opaque markets

An `Assay` object embodies the authority to transfer rights between participants. Since the ERTP does not support keeping the `Assay` object private, the resource provider is in a poor position to demand that all price information from trades be reported. As a result, it is impossible to ensure market transparency.

Ecash

Design of ecash protocols that emulate the properties of physical cash is a historically active area of research. Since participants in a market-based resource allocation infrastructure engage in online transfer of ownership claims, it is tempting to reuse this work. However, a closer analysis reveals the requirements for a credit transfer protocol differ significantly from the goals for ecash.

Opaque markets

Untraceable payment [22] is the motivating feature for ecash protocols. This feature is not one of the requirements set forth for a credit transfer protocol. Moreover, implementation of this feature may be at odds with market transparency, which is a requirement. Guaranteeing market transparency specifically requires correlation of payments.

No counter-party restriction

Anonymity is also a central feature of many ecash protocols. Again, this feature is not one of the requirements set forth for a credit transfer protocol. Anonymity may also be at odds with counter-party restriction, which is a requirement.

No role division

Redeeming a received coin for a newly minted coin is a basic operation in many ecash proposals; however, this design may not permit enforcement of a consumer versus merchant distinction, as the authority to take payment is widespread.

Summary

A credit transfer protocol grants resource providers greater autonomy by restricting the authority of credit owners. In general, an ecash protocol seeks to provide cash owners with greater autonomy by enabling anonymous and untraceable transactions. Ownership in a market-based resource allocation infrastructure may be very unlike cash in the distribution of authority between participants. Also, privacy, which is of paramount importance in an ecash application, is not relevant in the studied market-based resource allocation infrastructures. Cash is simply a different application. It is also therefore true that the credit transfer protocols studied in this paper may not be appropriate for use in cash applications.

IOU protocol

The IOU protocol [14] is defined in terms of the web-calculus [15], a programming language independent interface language using capability-based security semantics. For familiarity, the design is presented here using its mapping to the Java language. The corresponding interfaces are shown in Figure 2.

```
interface Account {                     interface Hold {
    GUID getBrand();                        GUID getBrand();
    int getBalance();                   }
    Hold accept();
    Hold offer(int amount);
    int reclaim(Hold child);
}
interface Terms {                       interface Restrictions {
    GUID getBrand();                        GUID getBrand();
    int transfer(Hold src, Hold dst);       Account approve();
}                                       }
```

Figure 2: IOU interface

Notice the major authority divisions represented by these four interfaces. The authority to transfer credits between owners is distinct from the authority to be an owner and both are separate from the authority to approve new owners. These authority divisions are crucial to satisfying the requirements set forth for a credit transfer protocol.

Account versus Hold

An `Account` embodies the authorities needed to play a consumer role using a particular brand of credit. An `Account` maintains a count of credits of a specified brand. A brand of credit is represented by a GUID. An `Account` holder can spend credits by invoking `Account.offer()` and passing the returned `Hold` to the payee. If the purchase does not complete, the credits in the `Hold` can be reclaimed by using it as an argument to an `Account.reclaim()` invocation. Once this invocation completes, the argument `Hold` is destroyed and is no longer eligible to contain credits. Only a `Hold` produced by either this `Account`'s `offer()` or `accept()` method is a valid argument to this `Account`'s `reclaim()` method. The `Account.accept()` method produces an empty `Hold`, into which a holder of a `Terms` object can transfer credits.

Terms

A `Terms` embodies the authority to transfer credits between participants. A `Terms`, together with an `Account`, provides the authorities needed to play a merchant role for a particular brand of credit. `Terms.transfer()` transfers all credits in a source `Hold` to a destination `Hold`, returning the number of credits removed from the source `Hold`.

Restrictions

A `Restrictions` embodies the authority needed to play a conformance officer role for a particular brand of credit. The `Restrictions.approve()` method produces a new `Account`. The holder of the `Restrictions` can require that participants meet certain requirements before invoking the `approve()` method on their behalf.

Meeting the requirements

Explanation of how the IOU protocol meets all of the requirements set forth for a credit transfer protocol follows.

Strategyproof

The IOU protocol enables the construction of strategyproof market mechanisms by providing effective property rights. A participant can gain exclusive ownership of credits and has autonomy in the choice to redeem or sell them.

Universal

Similar to the reviewed protocols, the IOU protocol uses units of a specified brand as the unit of account. A common protocol provides for the exclusive transfer of these credits from one owner to another. More specialized kinds of transfer are implemented through the creation of a derivative currency and an associated smart contract [6] for redeeming the derivative brand credits for the base brand credits. A smart contract is simply a software agent that performs credit transfers according to predetermined rules. An example is described in a later discussion of oversubscription in SHARP.

Transparent

In the IOU protocol, the authority to transfer credits between distinct owners is reified in the `Terms` object. The creator of a brand of credit can ensure market transparency by only granting the `Terms` capability to market mechanisms that publish price information. In this case, owners of these credits are unable to trade with each other, except through the authorized market mechanisms.

Liquid

Since credit transfers using the IOU protocol are exclusive, owners and potential owners can freely trade amongst themselves without need to consider the trustworthiness of their counter-party. All credits of a particular brand are fungible and are only dependent upon the reputation of the party that issued the credits. Credits can be traded at high velocity, without negatively impacting their value, thus encouraging trade and the presence of arbitrageurs.

Controlled

The IOU protocol most clearly distinguishes itself from the other reviewed protocols by the fine level of control available to resource providers.

A resource provider can restrict its pool of counter-parties by keeping the `Restrictions` capability private and only granting an `Account` to an approved participant. The approved participant can fully utilize its `Account` without exposing the represented authority to other parties. Exclusive ownership of credits is only achievable by a participant with an `Account`, thus preserving the binding between credit owner and approved counter-party.

The IOU protocol also supports a consumer versus merchant distinction by separating the authority to offer payment from the authority to take payment. A resource provider can restrict the pool of authorized merchants, by restricting access to the `Terms` capability. Without the `Terms` authority, a participant is unable to take exclusive possession of offered credits.

The IOU protocol disaggregates general ownership authority into the authority to: hold credits, offer credits and transfer credits. This decomposition enables implementation of a wide range of access-control policies through the selective granting or withholding of capabilities.

Simple

The IOU protocol is highly configurable; however, this configuration is expressed through the composition of a small set of primitives. The entire protocol consists of four interfaces and a total of six methods (ignoring the brand property of each interface which is provided for optional type checking). Restriction of a participant’s possible actions is expressed by the absence of a capability. In other words, access-control policy is expressed through the reduction of coordination costs. Further, fundamental features, such as double spending prevention, are not expressed through additional checks, but through the innate workings of the protocol, such as determining the amount of a received payment. In the IOU protocol, security is a side-effect of the way in which credit transfers are expressed.

Using the IOU protocol

To show the IOU protocol is at least as expressive as the protocols it proposes to replace, this section examines how the IOU protocol could be used in place of the previously reviewed protocols.

IOU in SHARP

Oversubscription

Oversubscription is a key feature in SHARP for improving resource utilization. Using the IOU protocol, a participant wishing to oversubscribe held resource credits does so by issuing a new, derivative brand of credit. The participant sells credits issued from this new derivative currency. Buyers of the new derivative currency redeem it by spending it back to the issuing participant who in trade returns an equivalent number of the held base brand credits. If the issuing participant's supply of base brand credits is exhausted, it cannot satisfy the request. The shortchanged buyer can thus easily identify the offending party, just as it can in the SHARP protocol.

Creation of new currencies is now an explicit operation, instead of an implicit part of every transfer as it was before. By only creating new currencies when required, credits remain fungible and thus market liquidity is maintained.

Non-repudiation

The SHARP paper claims non-repudiation as a significant feature of their mechanism; however, it is unclear what value this feature has when SHARP resource claims are only ever probabilistic resource claims. A participant who would otherwise repudiate a resource claim delegation can instead simply state that the resource claim was oversubscribed and another participant redeemed it first. This other participant need not be distinct from the delegating participant. As such, non-repudiation isn't actually a feature of the SHARP design taken as a whole.

Non-repudiation of transfers is a feature subsumed by the exclusive transfers provided by the IOU protocol. Once a payee has taken exclusive ownership of received credits, the payer lacks the authority to take back the spent credits.

IOU in Tycoon

Tycoon currently has modest requirements for the bank protocol. A simple transfer from a client to a provider is the only operation specified. This transfer remains simple with the IOU protocol.

Each client is issued an `Account`. Each provider is issued a `Hold` and a reference to the `Terms`. A client initiates a payment by sending an `Account.offer()` message. The reference for the message return is sent to the provider. The provider sends the received reference as the first argument in a `Terms.transfer()` message. The second argument in the message is the provider's `Hold`. Upon successful completion of the `Terms.transfer()` message, the provider records that payment has been made.

The number of network trips before payment is recorded is the same in either implementation. Assuming the underlying messaging protocol supports reference pipelining [18], the client's `Account.offer()` message and message to the provider are sent simultaneously. The critical path thus includes one network trip from the client to the provider. The provider subsequently sends out the `Terms.transfer()` message. One network round trip between the provider and the bank must complete before the result of the `Terms.transfer()` message is known. The Tycoon bank protocol requires a network round trip between the client and the bank, followed by a network trip between the client and the provider. Essentially, switching to the IOU protocol simply inverts the critical path message sequence, putting the round trip to the bank after the message between the client and the provider, instead of before.

Double spending logic is now implemented at the bank, instead of the provider. The provider is freed from implementing this check and maintaining the required state.

The Tycoon bank protocol also suffers from an error condition that the IOU protocol does not. Using the Tycoon bank protocol, a client can unilaterally place funds in the name of a specified provider. If the provider server is down, the funds become inaccessible, since the provider is not available to either accept the funds or refund them. Using the IOU protocol, the client could use an `Account.reclaim()` message to reclaim the funds when the provider fails to respond to the payment message.

IOU in ERTP

Specific market mechanisms implemented in terms of the ERTP are better implemented using more restrictive authority grants when using the IOU protocol; however, to show that any ERTP mechanism can be implemented using the IOU protocol, the ERTP itself is here implemented in terms of the IOU protocol.

The `Issuer` is a tuple containing: the `Terms` and `Restrictions`. A `Purse` is a tuple containing: a `Hold` and the `Issuer`. An `Assay` is a tuple containing: an amount and the `Issuer`.

`Issuer.makeEmptyPurse()` produces a `Purse` containing: `Restrictions.approve().accept()` and the `Issuer`. `Issuer.vouchforAssay()` produces an `Assay` containing: the amount from the candidate `Assay` and the `Issuer`. `Issuer.vouchForPurse()`does a `Terms.transfer()` from the candidate's `Hold` to a newly created `Hold`. The returned Purse contains: the newly created `Hold` and the `Issuer`.

`Purse.depositAll()` does a `Terms.transfer()` from the provided `Purse`'s `Hold` to the private `Hold`. The returned amount is used to construct the return `Assay`. `Purse.getAssay()` does a `Terms.transfer()`, using the `Hold` for both arguments. The returned amount is used to construct the return `Assay`.

`Assay.transfer()` first creates a new `Account` and deposits the credits from the source `Purse`'s `Hold`. `Account.offer()` is used to construct a new `Hold` containing the transferred amount. This `Hold` is transferred to the destination `Purse`'s `Hold`. Another `Hold` containing the remaining amount is produced and then transferred to the source `Purse`'s `Hold`.

Conclusion

This paper summarizes requirements, drawn from the research literature on market-based resource allocation, for a credit transfer protocol. An analysis of proposed protocols reveals which constraints have yet to be satisfied. The IOU protocol is described and found to satisfy all requirements set forth. The IOU protocol is then examined as a replacement credit transfer protocol within the previously analyzed resource allocation infrastructures. This examination shows the IOU protocol can meet all the functionality requirements of the existing protocols; while simultaneously providing better conformance to the principle of least authority [20] and reducing implementation complexity.

The IOU protocol is used in DonutLab [16], a decentralized implementation of PlanetLab [21]. The designers of DonutLab found the IOU protocol provided the flexibility required to meet their security goals, while being simple and productive to work with [17].

Market-based resource allocation infrastructures were conceived with ambitious goals and have been measured against the needs of large user populations. There is reason to hope a credit transfer protocol that meets the needs of this challenging application will also find applicability in similar domains. The IOU protocol may also be assisted in finding cross-domain use by a design that satisfies a larger set of requirements [14] than just those set forth for a resource allocation infrastructure.

Acknowledgments

The IOU protocol is the culmination of over 20 years of effort [19][7][12] by the capability-based security community to build ever more flexible credit transfer protocols. Design of the IOU protocol proceeded from experience with, and analysis of, ERTP [6].

Bill Frantz, Norm Hardy, Alan Karp, Mark Miller, Chip Morningstar and John Wilkes provided valuable feedback on early drafts of this paper.

References

[1] Mark. S. Miller and K. Eric Drexler; "Markets and computation: Agoric Open Systems"; *The Ecology of Computation*, B. A. Huberman, Ed.; pp. 133-176; North-Holland; 1988.
[2] Carl A. Waldspurger, Tad Hogg, Bernardo A. Huberman, Jeffrey O. Kephart, and W. Scott Stornetta; "Spawn: A Distributed Computational Economy"; *IEEE Transactions on Software Engineering*, 18(2):103-117, 1992.
[3] Yun Fu, Jeffrey Chase, Brent Chun, Stephen Schwab and Amin Vahdat; "SHARP: An Architecture for Secure Resource Peering", *ACM Symposium on Operating Systems Principles (SOSP)*, October 2003.

[4] Kevin Lai, Lars Rasmusson, Eytan Adar, Stephen Sorkin, Li Zhang, Bernardo A. Huberman; “Tycoon: an Implementation of a Distributed, Market-based Resource Allocation System”; Tech. Rep.; arXiv; 2005.
[5] Chaki Ng, David C. Parkes and Margo Seltzer; “Strategyproof Computing: Systems Infrastructures for Self-Interested Parties”; *Workshop on Economics of Peer-to-Peer Systems*, June 2003.
[6] Mark S. Miller, Chip Morningstar, Bill Frantz; “Capability-based Financial Instruments"; *Proceedings of Financial Cryptography 2000*.
[7] Mark S. Miller; “ERTP: The Electronic Rights Transfer Protocol”; http://www.erights.org/smart-contracts/ertp/; 1999.
[8] Michael P. Wellman; “Market-Oriented Programming: Some Early Lessons”; *Market-Based Control: A Paradigm for Distributed Resource Allocation*; World Scientific; River Edge, New Jersey; 1996.
[9] Jonathan B. Postel; “Simple Mail Transfer Protocol”; *RFC 821*; August 1982.
[10] Norm Hardy; "The KeyKOS Architecture"; *Operating Systems Review*; pp. 8-25; September 1985.
[11] J.B. Dennis, E.C. Van Horn; “Programming Semantics for Multiprogrammed Computations”; *Communications of the ACM*; 9(3):143-155; March 1966.
[12] Agorics Inc., “WebMart Overview”; http://www.agorics.com/Technologies/webmart.html; March 2002.
[13] R. Levin, E. Cohen, W. Corwin, F. Pollack, W. Wulf; “Policy / Mechanism Separation in Hydra”; *ACM Symposium on Operating System Principles*; 1975.
[14] Tyler Close; “Waterken IOU Design”; http://www.waterken.com/dev/IOU/Design/; 2004.
[15] Tyler Close; “web-calculus”; http://www.waterken.com/dev/Web/Calculus/; 2003.
[16] Marc Stiegler, Mark S. Miller, Terry Stanley; “72 Hours to DonutLab: A PlanetLab with No Center”; Tech Report; Hewlett-Packard Laboratories; 2004.
[17] Mark S. Miller; “An E IOU; always Y”; http://www.eros-os.org/pipermail/e-lang/2004-June/009850.html; June 2004.
[18] Barbara Liskov, Liuba Shrira; “Promises: Linguistic Support for Efficient Asynchronous Procedure Calls in Distributed Systems”; pp 260-267; *PLDI* 1998.
[19] Norm Hardy, et al…; “Space Banks {Getting New Pages and Nodes}; *Gnosis Manual*; Agorics 1981.
[20] Mark S. Miller, Jonathan S. Shapiro; “Paradigm Regained: Abstraction Mechanisms for Access Control”; *Proceedings of Eight Asian Computing Science Conference*; Tata Institute of Fundamental Research, Mumbai, India; Springer Verlag; 2003.
[21] Larry Peterson, Tom Anderson, David Culler, Timothy Roscoe; “A Blueprint for Introducing Disruptive Technology into the Internet”; *Proceedings of ACM HotNets-1 Workshop*; Princeton, New Jersey, USA; October 2002.
[22] David Chaum, Amos Fiat, Moni Naor; “Untraceable Electronic Cash”; *Advances in Cryptology, Crypto ’88*; pages 319-327; Springer-Verlag.