Making Trust Explicit in Distributed Commerce Transactions *

Steven P. Ketchpel Hector Garcia-Molina
Stanford University
Computer Science Department
Stanford, CA 94305
{ketchpel, hector}@cs.stanford.edu

Abstract

In a distributed environment where nodes are independently motivated, many transactions or commercial exchanges may be stymied due to a lack of trust between the participants. The addition of trusted intermediaries may facilitate some exchanges, but others are still problematic. We introduce a language for specifying these commercial exchange problems, and sequencing graphs*, a formalism for determining whether a given exchange may occur. We also present an algorithm for generating a feasible execution sequence of pairwise exchanges between parties (when it exists). Indemnities may be offered to facilitate previously infeasible transactions. We show when and how they enable commercial transactions.*

1 Introduction

Electronic commerce is a rapidly expanding field, with vast computer networks bringing together many potential customers and producers, increasing the number and range of contacts, and increasing the likelihood of matching producers and consumers. For example, consumers may find a number of information providers able to jointly fill an information request. A second example is the sale of computational resources, where processors offer to sell their idle time to others with computationally intensive work that can be parallelized. However, the ability to make matches around the globe also increases the likelihood that consumers and producers will have had no prior interactions, and hence will not know how trustworthy each party is. Therefore, additional protection may be required to guarantee the exchange of goods or services between mutually distrustful parties.

In this paper we develop mechanisms for completing commercial exchanges between parties that may or may not carry through on their commitments. The following example illustrates the problem we are considering (as well as some of the traditional authentication and security issues we are *not* covering). Consider a customer who wishes to purchase a particular document. Through a mechanism not covered here, the customer finds a publisher who offers to sell a digital document that matches the customer's specification, and they negotiate a price. Since the customer and publisher do not have an established relationship, and the publisher might be an untrusted individual, how can the exchange of funds and document take place? If the customer first sends the funds, the publisher might keep them and not provide the document, or might provide an incorrect document. If the publisher gives the document first, the customer might refuse to pay later.

While cryptographic techniques have been suggested for solving this problem [14, 15], we believe they are very expensive. Here we look at a more practical type of solution, that involves a trusted intermediary or third party. If a trusted party exists, then the solution is relatively simple [9]. The customer sends his payment and document description to the trusted party; similarly, the publisher sends the document to the trusted party. The third party verifies that the document matches the specification, and then sends the document to the customer and the payment to the publisher. Both the customer and the publisher trust the third party to honor its commitment to complete the exchange correctly.

Notice that this concept of a *trusted agent* (whose only role is to facilitate an exchange) is an abstraction of what occurs in practice. In particular, a trusted third party often has additional functions. For instance, the trusted party may also be the "electronic mall" where shoppers and producers "meet." Thus, the trusted agent not only makes guarantees about payments and deliveries, but also helps match customers and producers. Another common scenario is when one party is directly trusted by the other. In this case, there is

*This material is based upon work supported by the National Science Foundation under Cooperative Agreement IRI-9411306. Funding for this cooperative agreement is also provided by DARPA, NASA, and the industrial partners of the Stanford Digital Libraries Project. The first author was partially supported by a National Defense Science and Engineering Graduate Fellowship.

still a trusted agent role, but it is taken on by one of the main parties. Our abstraction separates the trust role from other roles (even that of principal party in the exchange) to help us understand how trust works in more complex interactions. Also notice that the mechanics of trusted agents may differ. For example, when a credit card company plays the role of a trusted agent, it often guarantees payment *before* the customer pays, thus making it possible for the producer to send the good directly to the customer. The model we will present in Section 2 can model these different mechanics, although for simplicity in the rest of the paper we only cover trusted agents that wait for goods and payments before proceeding.

Having discussed how a trusted agent facilitates an exchange between *two* parties, let us now consider a commercial exchange involving three or more entities. For example, the customer may be performing a search through an information broker that indexes documents stored at several servers. The customer wants to pay only if he receives *all* of the required documents, not if he receives only some of them. For example, one source may provide annotations or reviews of documents, while a second source provides the raw documents. The annotations by themselves may not be interesting; similarly, the raw documents are not as useful. Each source wants to release its document only if payment is guaranteed. The broker does not want to pay the sources until it has secured payment from the customer, but the customer does not want to pay anything unless complete delivery of all documents is guaranteed. If all the entities involved trust a third party, the problem can be solved easily, as a distributed transaction coordinated by the trusted agent. However, what happens if the broker and the customer share a common trusted agent that is not trusted by the sources, and the broker and sources have common trusted agents not trusted by the customer? We believe that this will be an ever more common situation in worldwide networks: there may be locally trusted agents, but no universally trusted agent that can be counted on by entities everywhere. Can the exchange we have illustrated be completed successfully in this scenario? If so, what are the steps? What happens if the broker in the example directly trusts one of the sources (i.e., the source takes on the role of trusted agent)? Is this the same as the source trusting the broker? As we will see, not all types of exchanges are feasible, so it is important to understand which are feasible and what role trust plays in them. Note that traditional agreement protocols do not solve this type of problem since commit protocols rely on trust among all parties. (We discuss the relationship to Byzantine agreement and other related protocols in more detail in Section 7.)

In this paper we will present a general framework for this type of problem, and show how trusted intermediaries may be represented. We develop an algorithm to generate an *execution sequence* (a total order of the pairwise exchanges needed to carry out the distributed transaction) for those exchanges recognized as *feasible*. A feasible exchange can be carried out in such a way that no participant ever risks losing money or goods without receiving everything promised in exchange. If the exchanges are carried out in the specified execution sequence, the interests of all parties will be protected. We also explore the notion of *indemnity* to facilitate commercial exchanges. For instance, a broker that is trying to sell a document it does not yet have, can offer some cash as collateral to a trusted agent. If the document is later provided by the broker, the trusted agent refunds the indemnity payment. If the broker fails to deliver the document, the customer expecting it is offered the cash in its place.

There are a number of issues that will not be covered in this paper, not because they are unimportant, but because solutions exist or are emerging. For example, as an entity communicates with a trusted agent, each needs to verify the other's identity [10, 8, 7]. For the actual exchange of funds between two parties, there are multiple mechanisms [17, 3, 11], each with its own verification procedure. When a customer receives a digital document, he could make and distribute additional copies illegally. This can be impeded (although there is no foolproof way to stop it) by watermarking schemes [1] or copy detection mechanisms [16, 2] for example. Verifying that the document the customer receives is the one that he ordered is more challenging. Sometimes the goods can be verified by the trusted agent before the exchange, but in other cases it is necessary to rely on "catch and punish" models of policing the quality of promised goods [9]. We also ignore privacy and protection concerns which might be addressed by using encryption.

We believe that understanding the notions of trust and indemnity for distributed commercial exchanges is important, since they will play an ever increasing role as networks bring together large numbers of customers and providers of electronic goods and services. Furthermore, trust (or the lack thereof) significantly impacts the types of algorithms used and their cost. For example, if all parties trust each other, exchange algorithms can be very efficient. However, as we will see, as fewer parties trust each other and more intermediaries are involved, the overhead will significantly increase. Feasibility and efficiency are two important implications of distributed trust.

Section 2 details the framework for describing distributed transactions. Two examples appear in Section 3 and are developed in Sections 4 and 5, which show how to create and reduce the *sequencing graph* and find the *execution sequence*, respectively. In Section 6 we will see that indemnity payments can make possible exchanges that were not previously feasible, and we show how to compute the overall indemnity amounts needed. Section 7 relates the formalism we propose here to existing solutions for associated problems. Section 8 discusses the cost incurred by mistrust. Finally, conclusions and directions for future research appear in Section 9.

2 Problem specification

In this section, we describe the classes of participants in a distributed transaction, show schemas for the actions that they can perform, and describe a method for representing the essentials of an exchange while leaving the details implicit.

2.1 Principals

There are three classes of principals which might be involved in a distributed transaction: producers, consumers, and brokers. In the context of information sales, the producers are information retrieval sources or libraries. A consumer is a computer user with an information request. A broker is an intermediary that has information about which sources are relevant for a particular query. (See, for example, [5].) In the context of computation subcontracting, a producer is a processor with idle resources; a consumer needs additional computation power; and a broker might be a network manager capable of matching them.

2.2 Actions

In order to model and analyze the types of exchanges described here, we consider only those actions which result in transfers between parties of a transaction. The $give_{a \to b}(d)$ action indicates that a gives b document d. Although a payment is only a special case of a $give$ action, for ease of exposition, it will be denoted $pay_{b \to a}(m)$, signifying that b has paid a the dollar amount m. Moreover, when one object is returned to its sender (e.g., if a broker returns a document to the source because the buyer has not sent payment), the action will be signified with a mathematical inverse, $give^{-1}_{a \to b}(d)$, which corresponds to the first $give$ action being compensated for.

It is also necessary to explicitly model time in the actions. When a source sends a document to a trusted intermediary, it should specify a deadline by which it expects to receive payment or have the document returned. Similarly, a customer needs to specify an amount of time which the intermediary may hold its payment before it is to be returned or the document provided. In the discussion that follows, we will assume that the deadlines allotted are always sufficiently generous to allow the transaction to be completed. (We believe that the results of Sections 4 and 5 can be extended to handle tighter deadlines, but this will not be covered here.)

2.3 States

We represent the state of an exchange as the unordered set of actions that were executed by any party during the exchange. Each party has a set of partial state descriptions such that any state which contains a superset of the actions of any element in that set (but does not contain another action by that party) is *acceptable*, that is, an outcome in which the agent receives all of the goods it paid for, and payment for all goods delivered. Alternative representations of the state (such as a sequence of actions [4]) would be more expressive, but the notation here is sufficient for the types of problems described. For instance, in the simple case where customer c wants to purchase document d from producer p for m dollars, there are four acceptable states for the customer. In the first, he gives the money to the producer and receives the document in exchange. Therefore, $\{give_{p \to c}(d),\, pay_{c \to p}(m)\}$ is an acceptable final state of the transaction. A second acceptable execution is where the customer pays the merchant, but the merchant is unable to provide the goods, and therefore refunds the money. This outcome is represented by the set $\{pay_{c \to p}(m),\, pay^{-1}_{c \to p}(m)\}$. A third acceptable state is the status quo, simply $\{\}$. The fourth desirable outcome is perhaps less realistic, but the customer can hope that he is given the goods without paying for them, $\{give_{p \to c}(d)\}$.

The producer's view of the acceptable transactions overlaps with the customer's, in that any of the first three states are acceptable. The fourth state is different, however, with the producer hoping to receive money from the customer while providing nothing, $\{pay_{c \to p}(m)\}$.

In addition to specifying acceptable executions, each party also specifies one that is "preferred." This device prevents a seller from always refunding the buyer's money, even when the goods could be provided. While that would always end up in an execution that is acceptable to both parties, the one where the exchange is completed is preferred by both. An *execution sequence*, which is a total order of the actions making up the distributed transaction, is acceptable only if the resulting final state is acceptable to all participants. A *protocol* is a set of instructions for each participant that governs its actions. A protocol is acceptable only if all of the possible exchange execution sequences sanctioned by that protocol are acceptable.

2.4 Constraints

In addition to the conditions placed on the exchange execution due to the acceptable final states, there may be additional constraints placed on the ordering of actions for practical reasons. For example, a party cannot send a document that it does not have. Therefore, if a producer is to send a document to a broker and the broker is to send that same document to the consumer, then clearly the producer to broker transfer must precede the broker to consumer transfer. The notation for constraints separates the two related events with an arrow, with the earlier one at the point of the arrow. The example above would be denoted:

$give_{b \to c}(d) \to give_{p \to b}(d)$.

2.5 Trusted components

Since the principals may have no previous working relationship, nor any expectation for continued interaction in the future, they may be distrustful of one another. Therefore, a transaction which demands that one side pay for services not yet received is likely to be viewed with skepticism by the buyer. On the other hand, the seller does not want to give its goods or services away without some guarantee of future payment.

Trusted third parties enable transactions under these conditions. In terms of the action/state formalism described above, trusted intermediaries have a limited set of actions available. They may perform either $give$ or pay actions, but only if the inputs have been provided by another party. They may also reverse actions in which they were the recipient, returning either money or goods to sender. These are the $give^{-1}$ and pay^{-1} actions described above. Typically, a trusted component acts merely as a conduit, sending goods or payments from their source to destination. However, when an exchange does not proceed as planned (e.g., if one party does not provide its promised goods) the trusted component reverses actions as necessary and terminates the exchange, returning the parties to status quo.

Another action that trusted components may perform is *notify(x)*. A *notify(x)* action informs one principal, x, that the other principals have fulfilled their parts of the exchange, and the exchange will be completed as soon as the notified principal complies. A notification allows a principal to take action based on this knowledge. Notifications also require temporal information. The broker should determine the time until the notification expires by finding the earliest expiration of the other pieces held for the exchange. If an agent provides the missing component from a notification before that notification expires, the agent is assured that the exchange will occur. After the notification expires, the intermediary is no longer bound to complete the exchange, since it may have returned some or all of the elements needed for the exchange.

A trusted component is able to specify guarantees by listing the states (sets of actions) which may result from its actions. For example, the typical guarantee would be: $\{\}$ (nothing happens) or $\{give_{a \to t}(d)$, $notify(b)$, $pay_{b \to t}(m)$, $give_{t \to b}(d)$, $pay_{t \to a}(m)\}$ (the exchange works) or $\{give_{a \to t}(d)$, $notify(b)$, $give^{-1}_{a \to t}(d)\}$ (the notification expires, and a gets the document back when b doesn't pay) or $\{pay_{b \to t}(m)$, $notify(a)$, $pay^{-1}_{b \to t}(m)\}$ (the notification expires and b gets money back when a doesn't provide the document). An agent may act as a trusted intermediary between two principals if they both believe that the intermediary will fulfill its guarantees (restrict itself to the promised states) in all circumstances.

3 Interaction graphs and examples

In this section, two problems are specified according to the method described in the previous section. Each of the parties in the transaction is enumerated, along with the actions, acceptable final states, and the ordering constraints imposed by necessity. To describe the examples, we introduce a simple graphical notation.

An *interaction graph* describes in graphical form the parties involved and the interactions between them. It is similar to the communication graph described in Section 6.6.3 of [4], without the temporal component. The interaction graph does not specify the ordering of the actions, nor does it guarantee that a feasible execution sequence exists. (See Figures 1 and 2 for examples.) Formally, an interaction graph I is a triple (P, T, E) consisting of:

- P: The set of principals—customers, brokers, and sources, represented as circles in the graph
- T: The set of trusted components, represented as squares
- E: The set of edges (p, t), with $p \in P$ and $t \in T$, such that principal p uses trusted intermediary t to carry out one side of an exchange with an untrusted principal

As discussed in Section 1, we will always model the principals interacting only through trusted intermediaries, although this is merely an abstraction—an intermediary might just be a "persona" of one of the principals, if it is directly trusted by the other principal. Thus, the interaction graph is bipartite, with P and T forming a partition. In Section 4.2.3 we study the implications of having one principal directly trust another.

3.1 Example #1: consumer seeks document from producer via broker

In the interaction graph of Figure 1, a consumer desires a particular document, but does not directly know any source capable of producing it. The consumer (c) seeks the help of a broker (b), who has the requisite connection to a capable producer (p). All the parties are distrustful of one another, and consequently c and b interact through an intermediary they both trust (t_1), while b and p share trusted intermediary t_2. The three principals are the consumer (c), the producer (p), and the broker (b). The trusted intermediaries are t_1 and t_2. (Keep in mind that, as discussed in Section 1, t_1 and t_2 are abstract roles that could be taken on by various real services. For instance, t_1 could be the provider of the

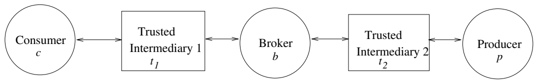

Figure 1. The interacting parties of a simple exchange

electronic catalog where c found out about b. Perhaps b trusts p directly, in which case the t_2 role is played by p. However, for now assume the trusted agents are separate.)

Since the two pairwise exchanges (c to b and b to p) are essentially identical, we focus only on the exchange between c and b. The relevant action for c is $\{pay_{c \rightarrow t_1}(m)\}$. For b, it is $\{give_{b \rightarrow t_1}(d)\}$. In addition to its notification responsibilities, the trusted intermediary can either forward the payment and document to their destinations, or cancel the transaction and return whichever pieces it has received. Formally, these actions are $\{$*notify(c)*, *notify(b)*, $pay_{t_1 \rightarrow b}(m)$, $give_{t_1 \rightarrow c}(d)$, $pay^{-1}_{c \rightarrow t_1}(m)$, $give^{-1}_{b \rightarrow t_1}(d)\}$. The acceptable states for the customer are: the status quo $\{\}$, receiving the goods and paying for them, $\{give_{X \rightarrow c}(d), pay_{c \rightarrow t_1}(m)\}$, with X ranging over $\{p, t_1, b, t_2\}$; receiving the goods without paying $\{give_{X \rightarrow c}(d)\}$; and paying the trusted intermediary, but receiving a refund instead of the goods $\{pay_{c \rightarrow t_1}(m)$, $pay^{-1}_{c \rightarrow t_1}(m)\}$. The acceptable states for the broker are: the status quo $\{\}$; receiving the money and sending the goods $\{give_{b \rightarrow t_1}(d)$, $pay_{X \rightarrow b}(m)\}$, with X ranging over $\{c, t_1, p, t_2\}$; receiving payment without sending the goods $\{pay_{X \rightarrow b}(m)\}$; and sending the goods to the intermediary, but getting them back $\{give_{b \rightarrow t_1}(d), give^{-1}_{b \rightarrow t_1}(d)\}$. For the trusted component, the acceptable states are: the status quo $\{\}$, performing the exchange $\{give_{b \rightarrow t_1}(d)$, $pay_{c \rightarrow t_1}(m)$, $give_{t_1 \rightarrow c}(d)$, $pay_{t_1 \rightarrow b}(m)\}$; backing out after receiving money from the consumer, $\{pay_{c \rightarrow t_1}(m)$, $pay^{-1}_{c \rightarrow t_1}(m)\}$; and backing out after receiving the goods from the broker $\{give_{b \rightarrow t_1}(d), give^{-1}_{b \rightarrow t_1}(d)\}$. A constraint requires the broker to have assurance from t_1 before sending his own money to t_2: $pay_{b \rightarrow X} \rightarrow$ *notify(b)* with X ranging over $\{p, c, t_1, t_2\}$.

The full exchange may be accomplished by c first sending payment to t_1, p providing the document to t_2, t_1 notifying b.[1] At this point, b knows that the money is available for it at t_1, so it goes ahead and purchases the document, i.e., b sends the money to t_2. Intermediary t_2 returns the document to b, and forwards the payment on to p. The document is sent from b to t_1, who forwards it to c, and sends the payment on to b. If some step is omitted, the trusted intermediaries return the pieces to their respective providers.

[1] As discussed in Section 2, we assume that the payment and document stay long enough at t_1 and t_2 for the rest of the operations to complete.

3.2 Example #2: buyer wants both documents from 2 brokers

Figure 2 is an interaction graph for a more complex exchange. As in the previous example, the consumer is interacting with brokers who have access to the right producers. In this example, however, the consumer wants information only available as a combination of documents provided by sources 1 and 2. Each of the two brokers is able to provide access to one of the sources by purchasing the desired document itself and reselling it to the consumer. The brokers each share a trusted intermediary with the consumer, and additional trusted intermediaries exist between the broker and the source of the document. The two documents are of value only if both are received, so the consumer is unwilling to buy one without some assurance that he will be able to obtain the second. As mentioned in Section 1, this may be because one document is a review or annotation of the other. As another example, one document may be the text of a patent and the other the diagrams. (In practice, patent text and diagrams are sold by different providers.)

The brokers are also concerned that they will purchase the document for resale, then find the consumer has changed his mind (perhaps because the other broker cannot obtain the document), leaving the broker holding an unwanted document. We omit the formal action/state description of the interaction for space considerations, hoping that the following informal description will suffice.

The consumer is willing to pay for both documents only if both are received. Each broker is willing to buy its respective document only if fully convinced that the customer will subsequently purchase it from the broker. The sources are willing to sell their goods to the brokers only if they receive payment. The trusted components simply enforce the exchanges. This exchange is not feasible, and fails on one sticking point: the customer is unwilling to commit to purchasing the first broker's document until he is sure that he can obtain the document from the second broker. Therefore, the first broker will not risk spending its own resources to obtain the document from its source, and is unable to guarantee delivery of the document because it doesn't yet own it. But the second broker goes through a symmetric analysis, and is also unwilling to acquire the second document from the second source, lest the first broker not keep its end of the exchange with the customer. If that happened, the second broker would have obtained the document with its own money, but have no customer to whom it may be resold.

In practice, many commercial exchanges may not be as complex as that of Figure 2. However, as information resellers and mediators become more common, information and documents will be combined and enhanced, leading to complex royalties and payment arrangements. Similarly, there will be complex contracts set up among service

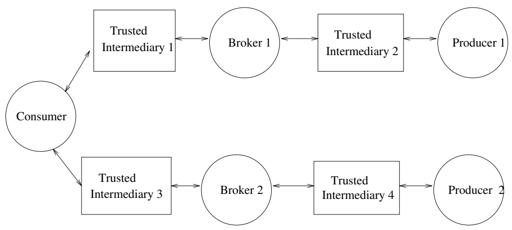

Figure 2. The interacting parties of a more complex exchange

providers that may lead to sequences of agreements similar to those of Figures 1 and 2. In such an environment it is important to understand what exchanges are possible at all, and which are inherently impossible. In Section 4, we introduce sequencing graphs, a general mechanism which determines when an exchange is feasible. Section 6 introduces a mechanism called an indemnity, which allows problematic exchanges such as Figure 2 to occur.

4 Sequencing graphs

The sequencing graph is a tool which finds a sequence of execution steps that will lead to a feasible exchange if a feasiblity test is met. The graph may be created mechanically from information about the interactions among the parties in the transaction, supplemented with the temporal constraints (described in Section 2.4) which prevent parties from transferring money or information that they do not have yet. A pair of reduction rules is then applied repeatedly to the sequencing graph. After the sequencing graph has been reduced as much as possible, an objective test determines whether the exchange is feasible. The introduction of the sequencing graph formalism, the manipulation rule, and the test to determine exchange feasibility are the major contributions of this paper.

The framework of actions and states described in Section 2 is very expressive. We are interested in applying it to a subclass of problems which conform to the instances of exchange problems. In these problems, the each party desires an item that the other has, and is willing to give up its own asset to acquire the item. The examples in the previous section are two instances of exchange problems, as is any multi-party interaction which can be reduced to the pair-wise exchanges through trusted intermediaries. These problems have the additional property that there is a single "commit" point at one of the pair-wise interactions which determines whether the whole interaction will proceed. Therefore, in the following sections, we focus on this type of exchange problem.

4.1 Creating a sequencing graph

The sequencing graph consists of two types of nodes: *commitment* nodes and *conjunction* nodes. A commitment node represents a decision to commit to a particular direct, pair-wise exchange between two parties, corresponding to an edge in the interaction graphs like those in Figure 1 and 2. Conjunction nodes represent a constraint between two or more commitments such that one will be done only if they all are. There are three typical uses of conjunction nodes. First, a trusted component commits to exchange goods from two distrusting parties. Second, a customer (or other principal) wants to obtain a set of documents, which are useful only if all are received. Third, a broker will commit to obtain a document only if it has a committed buyer. This third type of conjunction is the only one with an ordering component—the commitments controlled by a conjunction of the first two types can be undertaken in any order. This difference is reflected in the edges connecting the commitments to the conjunction node. In the case of conjunctions which have no ordering constraints, the edges are undirected edges represented in black. If there is a temporal ordering, the edge for the commitment which must be achieved first is a red edge, represented in bold in the figures.

The sequencing graph representations of the exchanges of Figure 1 and 2 appear as Figure 3 and 4, respectively. In these figures, hexagons are commitment nodes, representing the commitment between two parties, and are labeled with the agents involved in that commitment. Squares are conjunction nodes, linking all of the commitments entered into by one agent. Conjunction nodes are labeled with $\bigwedge_x$, where x is the agent who is common to all of the commitments.

Definition A *sequencing graph* SG of interaction graph $I = (P, T, E)$ is a 4-tuple (C, J, R, B):

- C: Commitment nodes, one node for each element of E in I. Let $c \in C$ correspond to edge (a, b) in I, if $\bigwedge_a \in J$ then $(c, \bigwedge_a) \in R \cup B$, and if $\bigwedge_b \in J$ then $(c, \bigwedge_b) \in R \cup B$.
- J: Conjunction nodes, one node for each internal node (a node with more than one edge) of I.
- R: Red high-priority edges, (c, j), $c \in C$, $j \in J$ and $\forall b \in C, b \neq c$, where commitments b and c share in common the agent involved in conjunction j, the commitment represented by c must precede the commitment b.
- B: Black regular priority edges, (c, j), $c \in C$, $j \in J$, where commitments b and c share in common the agent involved in conjunction j, and $(c, j) \notin R$.

A sequencing graph is always bipartite, with the commitment and conjunction nodes forming the partition. Therefore, an edge will always connect one conjunction node and

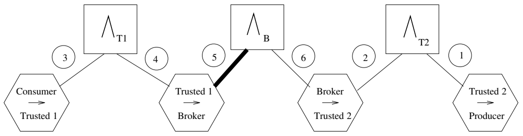

Figure 3. The sequencing graph for Figure 1. The circled numbers refer to the order of edge elimination described in Section 4.2.2

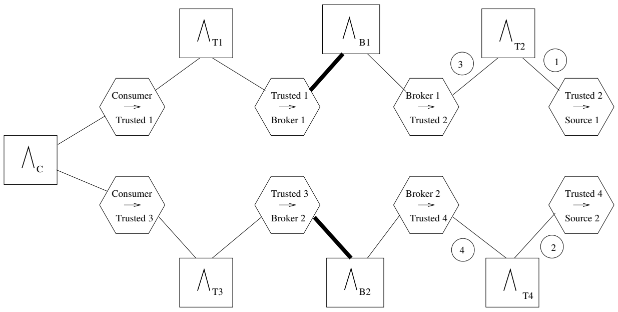

Figure 4. The sequencing graph for Figure 2.

one commitment node. Note that the red/black device limits the expressiveness of the allowable sequencing constraints, but suffices for the exchange problems considered here.

4.2 Reducing a sequencing graph

This section describes a process by which sequencing graphs are reduced. Edges are removed from the graph, corresponding to incremental steps toward a problem solution. Graph reductions can find the existence of a feasible execution and order the execution of steps if one exists. The two reduction rules correspond to intuitively meaningful steps in the exchange process, and there is a simple test for determining if the reduced graph corresponds to a transaction which can be carried out protecting all of the participants. We will give an intuitive justification for the two reduction rules, then show how they are applied in the examples from Section 3. Figures 5 and 6 show the final reduced version for these examples. The first example has an execution sequence which provides the needed assurances, while the second one does not. These results match the informal arguments of the outcomes given above in Section 3. After the intuitive descriptions and the running examples, we will give formal definitions of the reduction rules and feasibility test.

4.2.1 The reduction rules

The first reduction rule applies when a commitment node is on the fringe of a sequencing graph. In that case, if there are no conjoined commitments which take precedence (are connected with red edges), the commitment can be made provisionally, and "control" passed back to the corresponding conjunction node. There is nothing preventing the commitment from being made, so if the other commitments conjoined with that commitment are also possible, the whole commitment will be made. For example, in Figure 3, the commitment nodes (hexagons) at either end of the sequencing graph are on the fringe, and the edges connecting them to their conjunction nodes may be eliminated.

The second reduction rule applies when a conjunction node is on the fringe of the sequencing graph. Here, the other commitments required for the full conjunction have been provisionally made. Therefore, if this last commitment can be achieved, the whole conjunction can be executed. In Figure 3, once the outermost commitment nodes are disconnected, the two conjunction nodes corresponding to the trusted intermediaries are fringe nodes. The second reduction rule is applicable, removing the edges between the conjunction nodes and their remaining commitments.

4.2.2 Reducing examples #1 and #2

With these intuitive descriptions in mind, we will walk through the full reduction process with the two examples from Section 3. In Example #1, from Figure 3, we start by applying Rule #1 to the edge between "Trusted2 → Producer" and $\bigwedge_{T2}$. Since the commitment node is a fringe node (has only one connecting edge), the rule is applicable. As a result, the connecting edge is deleted. Intuitively, this means that Producer can give the document to Trusted2, letting Trusted2 later make the final commitment for this sale. (As we will see in Section 5, each deletion in the sequencing graph generates an action in the protocol for the exchange.)

After this deletion, $\bigwedge_{T2}$ is a fringe node. Rule #2 applies, and the edge between $\bigwedge_{T2}$ and "Broker → Trusted2" is deleted. Intuitively, Trusted2 can notify the broker that it has the document, so that the commitment of the sale is now in the hands of the broker. Now the commitment node for "Broker → Trusted2" is a fringe node. However, Rule #1 is *not* applicable, because the red edge connecting the commitment "Trusted1 → Broker" with $\bigwedge_B$ takes precedence over the black edge between $\bigwedge_B$ and "Broker → Trusted2". (Intuitively, the broker cannot commit to purchase the document yet—first, it has to secure payment.) We are not at an impasse, however, because Rule #1 applies to the edge between commitment node "Consumer → Trusted1" and $\bigwedge_{T1}$. After removing this edge, $\bigwedge_{T1}$ is a fringe node. Rule #2 sanctions the removal of the edge connecting $\bigwedge_{T1}$ with "Trusted1 → Broker". Now, only two edges remain—both connected to $\bigwedge_B$, one red and one black. The black edge is still blocked, but the red edge may be removed by Rule #1. The last edge between $\bigwedge_B$ and "Broker → Trusted2" may be removed by an application of either Rule #1 or Rule #2. Intuitively, the broker can be given the decision to purchase the document by Trusted2 and can be given the decision to complete the sale by Trusted1. So it can safely purchase the document, knowing that the customer's funds will be available at Trusted1. With all of the nodes disconnected (Figure 5), we see that this is a feasible transaction. In Section 5, we will produce the execution sequence that results from this reduction sequence.

We now turn to Example #2, from Figure 4. The reductions begin in the same way as in the previous example, with Rule #1 applying first between "Trusted2 $\rightarrow$ Source1" and $\bigwedge_{T2}$, and between "Trusted4 $\rightarrow$ Source2" and $\bigwedge_{T4}$. After these edges are removed, Rule #2 applies between $\bigwedge_{T2}$ and "Broker1 $\rightarrow$ Trusted2". Rule #2 also applies between $\bigwedge_{T4}$ and "Broker2 $\rightarrow$ Trusted4". After these four edges have been removed, however, we find that no others can be removed (see Figure 6). The only two fringe nodes, "Broker1 $\rightarrow$ Trusted2" and "Broker2 $\rightarrow$ Trusted4" are connected to their respective conjunction nodes by black edges that are subjugated to the red edges of those nodes. Therefore, the black edges may not be removed and we have reached an impasse, and we cannot show the transaction to be feasible.

4.2.3 Direct trust between principals

Example #2 offers an interesting study of the effect of principals trusting other principals directly. Here, we consider two variants on Example #2, one where Source1 trusts Broker1, and a second where Broker1 trusts Source1. In the first variant, the exchange becomes feasible; but in the second, it remains unfeasible. This difference underscores the fact that trust need not be symmetric (one party can trust another without being trusted by it), and the asymmetry can directly affect the ultimate feasibility of transactions.

In the first variant, Source1 trusts Broker1. Therefore, Broker1 is also playing the role of Trusted2. When the point of the previous impasse is reached, (Figure 6) Rule #1 is applicable to the edge between $\bigwedge_{B1}$ and "Broker1 $\rightarrow$ Trusted2". It is permissible to remove this edge, even though it is pre-empted by the red edge connecting $\bigwedge_{B1}$ and "Broker1 $\rightarrow$ Trusted1", because Broker1 is playing the role of Trusted2. Intuitively, Broker #1 has risk-free access to document #1, so it is unnecessary to secure the commitment from the customer before sending the document to the intermediary between the broker and the customer. Removing the edge between $\bigwedge_{B1}$ and "Broker1 $\rightarrow$ Trusted2" triggers a domino effect. The red edge from $\bigwedge_{B1}$ to "Trusted1 $\rightarrow$ Broker1" falls next, and the remainder of edges are removed starting from that point and proceeding counter-clockwise around Figure 4. Once the red edge between "Trusted3 $\rightarrow$ Broker2" and $\bigwedge_{B2}$ is removed, the black edge from $\bigwedge_{B2}$ to "Broker2 $\rightarrow$ Trusted4" is vulnerable. By eliminating all of the edges from the graph, the transaction is shown to be feasible.

In the second variant, Broker1 trusts Source1. Therefore, Source1 is also playing the role of Trusted2. This allows the edge between "Trusted2 $\rightarrow$ Source1" and $\bigwedge_{T2}$ to be removed by either clause of Rule #1, but after the first four edges have been removed to reach the state of Figure 6, no further reductions are possible, and the transaction remains infeasible.

4.2.4 Formal details

Graph reductions may be done in a greedy fashion—any applicable reduction may be applied at any time, in any order, until no further graph reductions are possible. Once the graph reduction process is complete and no further reductions are possible the reduced graph may be tested for feasibility (lack of edges). If one series of reductions leads to removing all of the edges, any other series of reductions will eliminate all of them also. Examples of infeasible reduced graphs are those such as in the second example, where the commitment node of the red edge of some conjunction node cannot be satisfied. If the reduced graph does not pass the feasibility test, then no determination can be made by this process. There may still be some series of steps which will achieve the transaction while providing all parties the necessary protection. Proving its impossibility is still an open research question. If the reduced graph is feasible, there is a protocol such that any execution sequence sanctioned by the protocol allows only acceptable final states for all parties to the exchange. The following section describes how to find that execution sequence.

Definition: Reduction Rule #1: Applicable when a commitment node is on the fringe. (R and B are the red and black sets from the original graph, R' and B' are the corresponding sets used in the reduced version.)
IF [$(c, j) \in R \cup B$, AND c is a fringe node in the reduced graph ($\nexists k \in J, k \neq j,\ (c, k) \in R' \cup B'$),]
AND

1. (c, j) is not pre-empted by a red edge ($\nexists(b, j) \in R$), OR

2. the trusted agent role of commitment node c is played by the principal of c [2]

THEN (c, j) may be removed from the graph,
$R' = R' - (c, j)$ and $B' = B' - (c, j)$, though (c, j) will appear in only one of R' and B'.

[2] A commitment node is always between a trusted agent and a principal, due to the bipartite nature of the interaction graph. A principal "plays the role" of the trusted agent when the principal on the other side of the trusted agent trusts this principal directly. In this case, the "trusted agent" is an abstraction, a persona played by the principal.

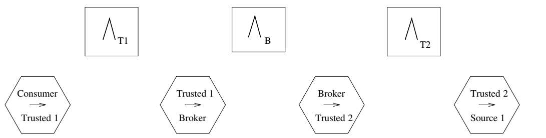

Figure 5. The reduced sequencing graph for Figure 1

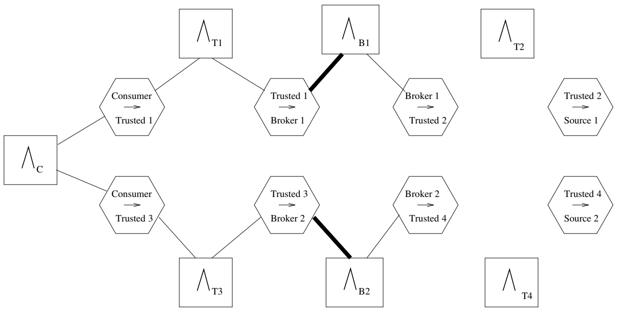

Figure 6. The reduced sequencing graph for Figure 2

Definition: Reduction Rule #2: Applicable when a conjunction node is on the fringe.
IF $(c, j) \in R \cup B$,
AND j is a fringe node of the reduced graph ($\nexists b \in C, b \neq c,\ (b, j) \in R' \cup B'$),
THEN (c, j) may be removed from the graph,
$R' = R' - (c, j)$ and $B' = B' - (c, j)$, though (c, j) will appear in only one of R' and B'.

Definition: Feasibility Test: Applicable when no further reductions are possible.
A reduced graph is feasible if all edges have been removed, $R' \cup B' = \emptyset$.

5 Recovering the execution sequence

If a sequencing graph is feasible, then there is an execution sequence that ensures that the distributed transaction will end in a state acceptable to all participants. The ordering corresponds to the order of deletion of edges in the sequencing graph (also the order in which commit points are reached) with one exception. In the case where a conjunction node has a red "priority" edge, it must be committed first, but executed last. This matches our intuition that a broker should have a buyer committed before he obtains goods, but must obtain the goods before he is able to give them to the customer. One feasible execution sequence (there may be others) is simply executing pairwise transactions in the order in which the commitment nodes became disconnected from the graph, deferring any commitment nodes connected to their conjunction nodes with a red edge until after all the black edge nodes have been executed. When a conjunction node is disconnected, a *notify* action is generated.

For instance, if we consider the graph reductions for Example #1, we find the following execution sequence:

1. Producer sends document to Trusted2.
2. Trusted2 notifies Broker.
3. Consumer sends money to Trusted1.
4. Trusted1 notifies Broker.
5. Broker sends money to Trusted2. (Red edge is delayed.)
6. Trusted2 sends document to Broker.
7. Trusted2 sends money to Producer.
8. Broker sends document to Trusted1.
9. Trusted1 sends document to Consumer.
10. Trusted1 sends money to Broker.

Note that the broker in Example #1 must have the funds to purchase the document *before* it receives the customer's money. If the broker were counting on the customer's funds to buy the document, then we would have an infeasible problem. Since the broker must obtain the money from the customer before providing payment to the source, the commitment between the broker and Trusted1 must be finalized before the broker can deal with Trusted2. This restriction adds the constraint $pay_{b \to p}(m) \to pay_{c \to b}(m)$. In Figure 3, this new scenario would change the black arc between $\bigwedge_B$ and the "Broker → Trusted2" node to a red arc. This means that there are two red edges emerging from $\bigwedge_B$, each of which must be done first. Since this is impossible, the whole exchange is infeasible. The reduction rules will not allow the removal of either red edge, and therefore, the graphic feasibility test will not show the exchange to be feasible.

6 Indemnifying to enable more transactions

This section introduces another mechanism for use between distrusting parties with a mutually trusted intermediary. A principal can make a credible promise by setting up an indemnity account. One principal gives the trusted intermediary a sum of money to be held in trust, along with a set of conditions under which the money is forfeit to another principal. For example, in the exchange detailed in Figure 2, each broker might offer the price of the other document as an indemnity, once it has obtained a promise from the seller to deliver its own document. Broker #1 would give Trusted Intermediary #1 the price of document #2, to be forfeited to the customer if the customer provides payment for document #1 to the trusted intermediary, but Broker #1 doesn't provide the requested document within a reasonable time. If the goods are provided when payment is furnished, then the indemnity amount is refunded to the broker. The exchange is feasible even if Broker #2 does not offer a similar indemnity, because the customer is satisfied with either of the two possible outcomes. With document #2 from Broker #2, he will either get document #1 and obtain the full set of goods desired in the exchange, or he will get enough money from Broker #1's penalty to offset the cost of document #2.

In terms of the graphic representation, an indemnity allows a conjunction node to be split. For instance, in Figure 4, if Broker #1 offers an indemnity, then the customer is willing to consider the transfers as two separate transactions (one for document #2, the other for either document #1 or the value of document #2). This means that the conjunction node at the left edge of the figure is split, and the edge connecting $\bigwedge_c$ and "Customer → Trusted1" is removed. With the removal, the reduction rules may be applied repeatedly, removing all the remaining edges and generating a feasible execution sequence.

In general, an indemnity may be given to remove a conjunctive edge of the second type (a customer demanding multiple documents in order to agree to purchase any of them). The principal providing the indemnity must share a trusted intermediary with the one requesting the indemnification, and the amount of the indemnity must be high enough to compensate for the worst case outcome. Usually the broker or source involved in providing a document will give an indemnity for that document, but this is not necessary—an indemnity may be offered by any party interested in moving the transaction forward.

Figure 7 shows an example involving a customer seeking three documents from three different brokers and sources. Again, the customer only wants the full set of documents, and will not pay for any unless all are obtained. In this example, document #1 has a cost to the customer of $10, document #2 is $20, and the third document is $30. Without indemnities, this transaction is infeasible, failing for the same reason the two broker/source case fails. This example shows that the order in which indemnities are offered is significant. In one ordering, Broker #1 offers an indemnity first, and must offer $50, to protect the customer from obtaining documents #2 and #3, at a cost of $50, and failing to get #1. Even after Broker #1 offers the indemnity, the transaction is not feasible, because the problem is essentially still a two broker problem between #2 and #3. If Broker #2 offers an indemnity next, he must set aside $40, the amount that the customer puts at jeopardy by acquiring documents #1 and #3 without having #2. When the customer has indemnities from the two brokers, he is willing to go ahead and offer the money to the trusted intermediary shared with the third broker, and the transaction proceeds. If Broker #3 offered an indemnity first (which would require $30), and was followed by Broker #2 offering the same $40, the

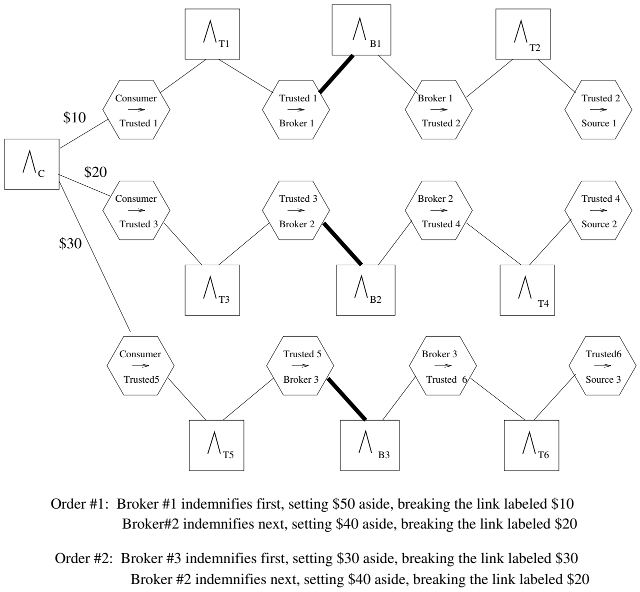

Figure 7. The effects of selecting different indemnification orders

customer could continue the transaction offering the money for document #1 to Trusted Intermediary #1. Therefore, the transaction could proceed with total indemnities of $70, versus the $90 required by the first ordering described.

A simple greedy algorithm allows selection of the ordering that minimizes indemnities. The commitment node which connects to the subtree with the highest cost should be indemnified first. The commitment node with the second highest cost should be indemnified next, and so on, by decreasing cost. The last commitment node in the conjunction does not need an indemnity, and by this ordering, the lowest cost commitment node is last. Since the indemnity required for any node in a conjunction is the total of the costs of all other pieces, the piece with the lowest cost has the highest indemnity. With that piece last, it need not be indemnified, so the total indemnity cost in the conjunction is minimized.

7 Relation to previous work

This section compares the problem that we are addressing to related work in the area of distributed databases and related fields. First, we show how the distributed transaction is different than a multiple party transaction that might be implemented with a two-phase commit protocol. Second, we compare the result to the problems solved by sagas. Third, we show how the distributed transaction among distrusting parties is different than Byzantine agreement. Finally, we show how the graph formalism is related to Petri nets.

7.1 Two phase commit protocols

In the traditional view of a transaction, all of the components of the system share the goal of maintaining a consistent state. Two phase commit protocols [6] are a way of ensuring that all the components in a system either perform the update or all of them reject the update and maintain the status quo. Our distributed transactions differ in that the system components have no such "global" view. If a customer can get a document without paying for it, that is a perfectly acceptable outcome. The merchant may disagree. Each principal has its own view of what acceptable outcomes are. Secondly, in transaction processing, it is traditionally assumed that a single designer has control over the programs that each process is running—that is, each node can assume the others will be following the protocol. No such assumption is made in the distributed transactions described here.

7.2 Sagas

Our representation of system state was motivated by the saga[6]. A saga is a sequence of actions that result in an acceptable final system state when they are executed. Essentially, what we propose here is for each agent (node in the interaction graph) to have its own set of acceptable sagas. Our graph representation and reduction rules can establish that there is an execution satisfying the sagas for all of the involved parties.

7.3 Byzantine agreement

Explicit distrust is a feature shared by the Byzantine agreement problem [12]. To reach a Byzantine agreement, the nodes following the protocol must reach agreement about a value in the face of arbitrary failures and deceptions on the part of the "traitorous" nodes. A Byzantine protocol is successful if the non-failing nodes can agree even if some large fraction (typically up to 1/3) of the nodes fail to follow the protocol. Distributed transactions have some of the same flavor of trying to protect those following the protocol from the actions of the deviants. Here again, however, the nodes have different opinions about the acceptable outcomes, and are, in essence, trying to force "agreement" on their own desired outcome. Moreover, the presence of some trusted nodes allows agreement without replicating the actions and communication among several equivalent agents and determining the outcome by guaranteeing a non-traitorous majority.

7.4 Petri nets

The graph formalism we have described here is similar to, though weaker than, a Petri Net[13]. The exchanges can be captured in a Petri net formalism, with the added advantage that consumable resources (such as money) are modeled very naturally in the tokens of a Petri net. However, the specific problem of subset coverability (which might be used to establish the feasibility of an exchange) is still an open problem—there is no known algorithm. Perhaps exchanges can be modeled using a restricted class of Petri nets with better computational properties; this is an area for future research.

8 Cost of mistrust

We have seen in Section 4.2.3 that when the principals trust each other directly, some transactions that were infeasible in a distrusting environment become feasible. Even cases that are feasible in the distrusting environment may be more efficient if the principals are willing to interact directly. Two parties that trust each other can perform an exchange with two messages—each sending what the other wants. In contrast, when the two nodes distrust each other, four messages are required—two to the trusted intermediary, and two from the trusted intermediary.

If a single trusted intermediary may be used for the entire system in any exchange between two principals, then any exchange becomes feasible, without indemnities. A customer is willing to send full payment to the trusted intermediary, convinced that he will always receive the requested goods or his money back, regardless of the actions of the other principals. Similarly, the brokers and sources are willing to send their money or goods to the trusted intermediary with a set of constraints (marking the other exchanges that must occur if theirs is to occur). The trusted intermediary checks to see that the constraints are consistent with a completed exchange, and if so, executes it. For instance, in the second example, a broker is willing to send his money to the trusted intermediary with the restriction that it purchases the document from its source only when: 1) the customer agrees to buy the documents, 2) the other broker puts up his money for the document from the other source, and 3) both sources provide the documents.

9 Conclusion

In a large space of distributed information and resources, a customer is likely to be assisted by a number of brokers that monitor a large number of sources. Furthermore, complex queries may well require information from different sources, leading to an exchange with many parties. These parties might all have different interests and different trusted organizations. Therefore, a protocol which protects the sum of all their interests is required to enable a transaction. The process of specifying a distributed exchange, capturing the details in a sequencing graph, and then reducing the resulting graph allows such a protocol to be synthesized.

The examples presented here did not make full use of the expressiveness of the model introduced in Section 2. In future work, we intend to exercise the full power of the formalism. Moreover, we intend to formally prove the equivalence between the graph manipulation algorithms and a subclass represented by the formal model. Within this framework, we will establish the soundness and completeness of the algorithms. Future work will also extend the algorithms proposed here to allow a fully distributed approach, with each participant locally making decisions about the feasibility and sequencing of its own parts of the transaction. In this paper, we only considered scenarios where pairs of parties interact through a trusted agent. When an agent is trusted by more than two parties, additional distributed exchanges may become feasible, and our results should be extended to cover this case. Another interesting extension is trust relationships among the trusted intermediaries. A "hierarchy of trust" may allow more completed transactions, and model more closely the use of trust in the real world. Future work will also provide a more complete treatment of the temporal issues, and the complexities arising from the expiration of partial exchanges and notifications.

Acknowledgments The authors thank Daphne Koller, Joe Halpern, Martin Röscheisen, Andreas Paepcke, and Tim Stanley for helpful suggestions and useful discussions.

References

[1] Jack Brassil, Steven H. Low, N. F. Maxemchuk, and Larry O'Gorman. Electronic marking and identification techniques to discourage document copying. In *Proceedings of the Thirteenth Annual Joint Conference of the IEEE Computer and Communication Societies*, pages 1278–1287, Toronto, Canada, June 1994.

[2] Sergey Brin, James Davis, and Hector Garcia-Molina. Copy detection mechanisms for digital documents. In *Proceedings of SIGMOD'95*, 1995. Available at http://www-db.stanford.edu/people/serge/copy.ps.

[3] DigiCash. DigiCash brouchure. Available at http://www.digicash.com/publish/digibro.html, 1994.

[4] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.

[5] Luis Gravano and Hector Garcia-Molina. Generalizing GlOSS to vector-space databases and broker hierarchies. Technical Report STAN-CS-TN-95-21, Stanford University, May 1995. Available at ftp://db.stanford.edu/pub/gravano/1995/stan.cs.tn.95.21.ps.

[6] J. Gray and A. Reuter. *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann, 1993.

[7] N. Haller and R. Atkinson. On internet authentication, October 1994. Available at http://ds.internic.net/rfc/rfc1704.txt.

[8] C. Kaufman. Distributed authentication security service, September 1993. Available at http://ds.internic.net/rfc/rfc1507.txt.

[9] Steven Ketchpel. Transaction protection for information buyers and sellers. In *Proceedings of the Dartmouth Institute for Advanced Graduate Studies '95: Electronic Publishing and the Information Superhighway*, 1995. Available at http://robotics.stanford.edu/users/ketchpel/dags4.html.

[10] J. Kohl and C. Neuman. The Kerberos network authentication service (V5), September 1993. Available at http://ds.internic.net/rfc/rfc1510.txt.

[11] B. C. Neuman and G. Medvinsky. Requirements for network payment: The NetCheque perspective. In *Proceedings of IEEE COMPCON'95*, March 1995.

[12] M. Pease, R. Shostak, and L. Lamport. Reaching agreements in the presence of faults. *Journal of the ACM*, 27(2):228–234, April 1980.

[13] J. Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, 1981.

[14] M. Rabin. Transcation protection by beacons. *Journal of Computer and System Sciences*, 27:256–267, 1983.

[15] Tuomas Sandholm and Victor Lesser. Equilibrium analysis of the possibilities of unenforced exchange in multiagent systems. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 694–701, 1995. Available at ftp://ftp.cs.umass.edu/pub/lesser/sandholm-ijcai95-equilibrium.ps.

[16] N. Shivakumar and H. Garcia-Molina. SCAM: A copy detection mechanism for digital documents. In *Proceedings of the 2nd International Conference in Theory and Practice of Digital Libraries (DL'95)*, June 1995. Available at http://www-db.stanford.edu/pub/shivakumar/1995/scam.ps.

[17] L.H. Stein, E.A. Stefferud, N.S. Borenstein, and M.T. Rose. The Green commerce model. Technical report, First Virtual Holdings Incorporated, October 1994. Available at http://www.fv.com/tech/green-model.html.