# Distributed Commerce Transactions with Timing Deadlines and Direct Trust

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

**Abstract**

In a multi-party transaction such as fulfilling an information request from multiple sources (also called a distributed commerce transaction), agents face risks from dealing with untrusted agents. These risks are compounded in the face of deadlines, e.g., an agent may fail to deliver purchased goods by the time the goods are needed. We present a distributed algorithm that mitigates these risks, showing that it is sound (produces only safe multi-agent action sequences) and complete (finds a safe sequence whenever one exists). We also show how the algorithm may be extended so that agents may interact directly with other participants rather than through a trusted intermediary.

## Introduction

The explosion of networked information sources leads to the sense that the answer to any question is out there, if only one has access to the right combination of sources. Search engines and information brokers help navigate the space, yet adding information services and sources which require payment exposes a number of hazards. A transaction with a customer, broker(s), and source(s) involves the following risks:

1. A customer spends money without receiving the promised document in time.
2. A broker or source sends a document to a customer without receiving payment.
3. A customer with a conjunctive request pays for one document without being able to obtain all of the others in time. (This risk is known as "buying half a conjunction.")
4. A broker buys a document but has its customer back out, leaving no one to re-purchase the document.

In (Authors 1996a; 1996b; 1997), we define a *distributed commerce transaction* where agents playing the roles of customer, broker, or source interact with each other through trusted intermediaries. The goal is to move documents from the sources to the customer. This problem is complicated by two factors. First, agents have a limited set of actions (sending a document or money, requesting a document or money, or returning a document or money, notifying pending status of exchange). Second, the distrust between agents results in added constraints on the orderings of actions (also called *execution sequences*) to prevent any of the four risks described above.

In addition, no single agent has a global view of all the constraints. Yet, in many cases, a careful ordering of the agents' actions can produce a *safe* sequence which avoids the four risks for all of the agents. If no safe sequence exists, we say the instance is *infeasible*.

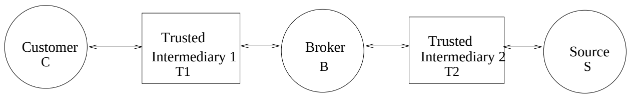


Figure 1: A simple example

### Examples

Figures 1 and 2 are two examples of distributed commerce transactions. In the first, a broker (B) is acting as the middleman between the customer (C) and source (S). Each step (S to B and B to C) uses a trusted intermediary (T2 and T1, respectively). The safe execution sequence is achieved by following the steps in the indicated order:

1. C requests document and sends money to T1.
2. T1 notifies B of request and presence of C's money.
3. B requests document and sends money to T2. (We assume that B has its own funds.)
4. T2 notifies S of request and presence of B's money.
5. S sends document to T2.
6. T2 sends document to B, B's money to S.
7. B sends document to T1.
8. T1 sends document to C, C's money to B.

The key feature of this example is that all exchanges take place using trusted intermediaries, thus eliminating risks 1 and 2. Moreover, B doesn't send money (Step 3) until C has guaranteed payment by sending money to T1 (Step 1), thereby eliminating risk 4.

The second example is a conjunctive request. The customer (C) wishes to get one document each from sources 1 and 2 (S1 and S2) and is using two brokers

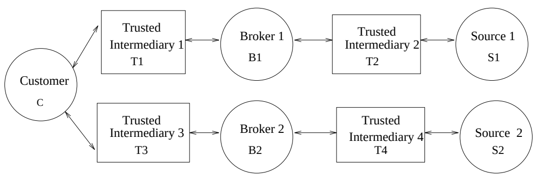


Figure 2: A conjunctive example

(B1 and B2) to get them. Trusted intermediaries T1 through T4 intervene in the process as shown in the figure. The transaction proceeds as described below, but (like all other sequences subject to these constraints) fails to achieve the global exchange:

1. C requests documents from B1 and B2.
2. B1 and B2 request documents from S1 and S2.
3. S1 and S2 send document to T2 and T4.
4. T2 and T4 notify B1 and B2 that the exchanges will complete when the brokers send money.
5. B1 and B2 request money from C.
6. C does nothing. The transaction fails.

In the second example, C is faced the risk of buying half a conjunction. If C responds to the requests of B1 and B2, B1 might provide the document while B2 returns the money. In that case, C has spent half of its money but has only a worthless subset of its desired documents. Thus, it can be shown that this is an infeasible instance.

## Timing and deadlines

In the previous examples, when a customer sends payment to a trusted intermediary, the customer cannot control how long the transaction might take. In a realistic solution, there must be a mechanism by which the customer can establish a deadline for receiving the ordered goods, with the ability to get a timely refund if the deadline is not met. In this section, we present a distributed algorithm which allows the generation of safe sequences that respect the deadline established by the customer. The algorithm is the significant contribution of this paper. We first describe the model of time that we use, then show two examples of deadlines in use, one where the deadline is attainable, the second where it is not. Next we describe the algorithm in greater detail, finally turning to the formal properties of the algorithm, showing its soundness and completeness under this formalization. Owing to space limitations, we omit certain details of the algorithm and proofs. Full versions may be found in (Authors 1996b).

We will model time using Lamport's representation of physical clocks for distributed systems (Lamport 1978). Each clock value is a number, and larger numbers represent later times than smaller ones. We ignore the problem of clock synchronization, assuming that all of the agents' clocks show the same value at one instant in time. Agents also have knowledge of how long delivery may take between any two agents,

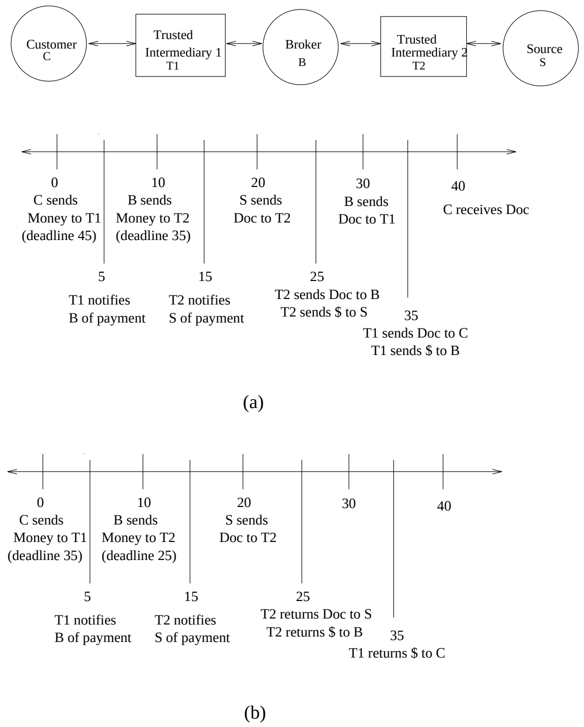


Figure 3: Exchanges with deadlines

and delivery times are symmetric between pairs of agents. Each agent has access to a local function called `DELIVERYTIMETO`($x$), which returns the upper bound for the amount of time that a delivery sent to neighboring agent $x$ will take, including any operations by trusted intermediaries in the exchange. Finally, in this idealization, the time requirement of local computation is negligible compared to communication time. These assumptions could be relaxed without significantly affecting the logic behind the algorithm.

### Examples using Time

We will repeat the first example under varying time conditions, showing one example where the transaction is still feasible, and a second where it cannot be completed in time. A deadline is added to each document request. In order for the request to be satisfied, the client must receive the document at or before the deadline. For the examples, assume that all of the delivery times are at most 5 seconds between an agent and a trusted intermediary, and at most 10 seconds between two principals. (These values are recorded in `DELIVERYTIMETO`.)

In the first example (Figure 3(a)), C places an order at time 0 for document D, and wants it before time 45. The sequence of steps is described in Table 1. In step 3, B calculates the deadline of 35 by knowing that C must have the document by time 45, and `DELIVERYTIMETO`($C$) is 10, so B must send the document by time 35 in order to guarantee its arrival by

Table 1: Steps for first deadline example

| | Time | Action |
|---|---|---|
| 1 | 0 | C requests document (deadline 45) and sends money to T1. |
| 2 | 5 | T1 notifies B of request (deadline 45) and presence of C's money. |
| 3 | 10 | B requests document (deadline 35) and sends money to T2. |
| 4 | 15 | T2 notifies S of request (deadline 35) and presence of B's money. |
| 5 | 20 | S sends document to T2. |
| 6 | 25 | T2 sends document to B, B's money to S. |
| 7 | 30 | B sends document to T1. |
| 8 | 35 | T1 sends document to C, C's money to B. |
| 9 | 40 | C receives document, B receives money. |

Table 2: Steps for second deadline example

| | Time | Action |
|---|---|---|
| 1 | 0 | C requests document (deadline 35) and sends money to T1. |
| 2 | 5 | T1 notifies B of request (deadline 35) and presence of C's money. |
| 3 | 10 | B requests document (deadline 25) and sends money to T2. |
| 4 | 15 | T2 notifies S of request (deadline 25) and presence of B's money. |
| 5 | 20 | S sends document to T2. |
| 6 | 25 | T2 returns document to S, returns money to B. |
| 7 | 35 | T1 returns money to C. |

time 45.

In the second variant (Figure 3(b)), C demands having D at time 35, a full 5 seconds before the agents were able to guarantee the arrival of the document in the previous variant. Consequently, this second variant is infeasible, if transfers do take their worst case upper bound. The sequence of steps is described in Table 2.

## Algorithm Description

Our fully distributed algorithm generates a safe sequence of actions which results in the customer receiving the requested documents before the stated deadline, if such a sequence exists. If the algorithm fails to find a sequence, the problem instance is infeasible. Each agent is treated as a separate process in the system, though all principals run the same algorithm. (Trusted intermediaries are special. They simply moderate exchanges between untrusting principals.) The algorithm follows an *event-based* framework. As the ultimate customer tries to fulfill its information request, it generates subsidiary requests which are sent to other agents. The agents responding to these events consider whether they can safely satisfy the request. In some cases this can be done without appealing to other agents, but frequently the agent will have to send out one or more requests and make decisions based on the outcome of the requests. The sequence of these actions is guaranteed to be safe because of the tests that each agent performs before taking an action.

Each agent makes a local decision what to do next based on the current state of requests with neighboring agents, the current time, the deadline given by its direct customer, and the "ultimate deadline", given by the original customer. An agent has a request from its customer, or from the trusted intermediary shared with that customer. (In the latter case, the trusted intermediary is holding the customer's money to pay for the request, guaranteeing that the customer cannot back out.) In order to fulfill this request, the agent may be acting as a broker and decomposing the request into multiple requests where it acts as the customer. The agent chooses an action from the following five cases:

1. *The ultimate deadline for this exchange has expired.* In this case, the agent should end the exchange as expediently as possible.
2. *The agent has all of the pieces to fulfill the information request.* In this case, the agent is either done (if it is the ultimate customer) or can send the answer to the trusted intermediary shared with the requester. Even if the document is not guaranteed to get to the customer or the trusted intermediary in time, there is a chance that it will, and sending to the trusted intermediary does not cost the agent anything.
3. *The agent has promises from trusted intermediaries that they have the documents that will fulfill the request.* In this case, the agent evaluates whether all of the documents are guaranteed to arrive before its deadline. If so, it can safely send money to the trusted intermediaries to obtain the documents (if its own customer has sent money to a trusted intermediary). The documents will arrive before the agent's deadline, which was set to give the agent enough time to re-send it to the requester. If there is only a single document, the agent can safely send money even if the document is not guaranteed to arrive in time. The trusted intermediary will send the document only if it will arrive in time, and there is no risk of buying half a conjunction. (See the following section for more discussion on this point.)
4. *The agent and trusted intermediaries are still missing one piece for the request.* In this case, the agent determines the deadline by which it must receive the document. To factor in the time required to travel to the customer $y$, the agent subtracts `DELIVERYTIMETO`($y$) from the deadline that $y$ had established. The ultimate deadline is whatever value $y$ passed as its ultimate deadline. If $y$ has given money to the trusted intermediary for this exchange, the current agent can do likewise. This may expedite the acquisition of the missing piece.
5. *The agents and trusted intermediaries are missing*

*two or more pieces for the request, or the customer has not guaranteed payment.* In this case, the agent is reduced to merely requesting sources to provide the necessary pieces without guarantee of payment. The agent determines the deadlines as specified in the previous case. The sources may be willing to send the document to a trusted intermediary if they do not need to spend any resources. If they have to spend money to acquire these documents from other sources, however, they will wait until the customer makes a binding promise of payment.

The trusted intermediaries also have a central role in enforcing deadlines. An exchange which might not complete by the deadline will not be executed by the trusted intermediary, even if both the document and money are present.

## Discussion of Algorithm

In order to keep track of the progress toward a transaction, each request contains two separate deadlines, one for the direct customer and one for the ultimate customer. So in the first deadline example, for the exchange between S and B, 35 was the deadline, but the ultimate deadline was 45, the one established by C. These two deadlines provide different information. If the earlier deadline is met, payment is guaranteed. If the ultimate deadline has passed, then there is no chance the exchange will succeed, and an agent need not bother taking any further action, other than closing out the transaction. If the earlier deadline has expired, but the later one has not, then the agent should take costless actions which may allow the exchange to proceed. In this "gray area", the exchange may succeed if transit times are less than their worst case `DELIVERYTIMETO` values, but it is not guaranteed.

For example, in case three, the agent is considering whether to send money to a trusted intermediary who has a desired document. If it is 8 time units before the agent must have the document, and each transfer can take up to 5 units, the agent should send payment if there is only one document. Even though there is no guarantee that the exchange will be successfully completed, there is no risk to the agent. If the link from agent to trusted intermediary takes only 2 time units, then the trusted intermediary can complete the exchange with a guarantee that the document will arrive in time. On the other hand, if the payment takes 5 time units to get to the trusted intermediary, the exchange will not be completed, since there is the risk of the document arriving after the agent's deadline. Therefore, the trusted intermediary will simply return the agent's money. If there are two documents, the agent cannot safely send payment to both, since one agent may get it quickly and send the document, but the other gets it slowly and returns the money. In that outcome, the agent has bought half a conjunction.

Before any "expensive" action is taken (such as acquiring a document on behalf of another or sending a document to its requester), a check is made to ensure that the customer will receive the document in time. If not, the expensive action is not taken, but (if possible) a riskless action is substituted, such as making a request. Before making a payment, a customer determines the maximum amount of time required for it to acquire all of the source documents. This calculation allows for multiple documents to be sent in parallel, so the total time for receiving all of the documents is just the longest individual time, not the total of all the times. It also takes into account the document's point of origin, knowing whether the document is at the trusted intermediary or must travel "two hops" from the source. If this customer is a broker who will be sending the documents to another client, the broker will consider not only the time that required to obtain all the needed documents, but also the time to send the combination on to the client.

## Soundness

In this section, we outline the proof showing that the algorithm is sound, that is, it never subjects an agent who abides by the algorithm to any of the four risks (see Intro.), even if other principals (not trusted intermediaries) deviate from the algorithm. The full proofs of soundness and completeness may be found in (Authors 1996b).

**Theorem 1:** *Any sequence of actions produced by the algorithm is riskless with respect to the temporal safety conditions.*

We show that each of the four risks cannot arise in a sequence of actions generated by the algorithm.

1. *Whenever a customer sends money, it always receives the document before the specified deadline, or it receives a full refund:* A customer will only send money to a trusted intermediary. The intermediary will return the money unless it has received the desired document from the provider, and is certain that it will reach the customer in time.
2. *Whenever a provider sends a document, it always receives payment, or the document is returned:* As in the previous case, providers only give documents to trusted intermediaries. The trusted intermediaries send a document to the customers only if the payment is sent to the provider at the same time.
3. *A customer never buys half a conjunction:* A customer never sends money for one conjunct unless it is sure that all of the other conjuncts are readily available before the deadline. Payments are made only when at most one document is not held by the customer or a trusted intermediary (case three or four of the algorithm).

   After payment for the last missing conjunct is forwarded to the shared trusted intermediary, there are two possible outcomes:

   (a) The provider can deliver the document, allowing the customer to obtain the full conjunction by sending payment to each trusted intermediary that holds a conjunct, or

(b) The provider fails to deliver the document, so the trusted intermediary returns the customer's payment, and the customer has lost nothing, since it has not paid for any of the other conjuncts.

4. *A broker never buys a document without being able to re-sell it:* This condition is dependent on two aspects. First, a customer may back out of a purchase. This threat is removed by ensuring that the customer has given money to the trusted intermediary before the provider spends money to acquire a document. Second, the customer's deadline may expire after the broker buys the document but before it is sent to the customer. To prevent this risk, the broker undertakes a second test, spending money to acquire a document for re-sale only if it is guaranteed that the full request can be completed and sent to the customer in time.

Therefore, if none of the four unsafe cases may arise from the sequences suggested by the algorithm, any sequence suggested by the algorithm is riskless. ■

### Completeness

Here we outline the proof technique to show that if there is an exchange which does not cause any agent to suffer a risk prohibited by the soundness conditions, then the algorithm will find that exchange or an equivalent one. In (Authors 1996b), we show via an inductive argument that the customer will use the most expedient method to notify the provider of the request and the status of payment for that request. Money will be advanced to a trusted intermediary whenever it may be done so without risking violation of the soundness conditions. The provider will acquire the documents as quickly as possible, and dispatch them to the customer as quickly as possible. Moreover, the delivery always involves a trusted intermediary who moderates the exchange, protecting the document from disclosure without payment, or disclosure after the deadline.

## Direct Trust

In the preceding discussion, all exchanges used trusted intermediaries because the principals did not trust each other. It may be the case, however, that one principal does trust another, because of an established reputation (e.g., a government institution or a well-known commercial venture) or due to a history of interactions. The risk of reputation damage may outweigh the gain from a dishonest completion of the current transaction. Direct trust need not be symmetric. A customer may trust a merchant without the merchant trusting the customer. Direct trust manifests itself in two ways:

1. If a provider trusts a customer, the provider will send a document directly to the customer even before receiving payment, trusting that the customer will subsequently pay for or return the document.
2. If a customer trusts a provider, it may advance payment to the provider, trusting that the money will be returned if the document cannot be obtained.

At first glance, the latter point seems trivial. The difference between a direct payment and the guaranteed notification from a trusted intermediary is minor, and only matters if the agent does not have enough money to acquire the document without the customer's payment. Direct trust allows some of the customer's money to be spent by the broker for the desired document. In contrast, if the broker receives only a promise of payment, it will not have enough to purchase the document and the exchange would fail.

### Changes to Algorithm

The changes to the algorithm to enable direct trust between principals are relatively minor. The difference in behavior, however, may be significant. In each instance where an item was formerly sent to the trusted intermediary, a check is made to see if the recipient is trusted directly. If so, the intermediary is bypassed.

Another addition is required to ensure the fulfillment of obligations undertaken as a result of being trusted. If a customer has received a document from a source that trusts it, it must pay for the document if the rest of the exchange is completed, or return the document if the exchange is aborted. After a successful completion, the agents examine the set of documents they handled to be sure that they have paid for each one, paying the source immediately for any received on direct trust. For each unsuccessful completion, the documents received on trust must be returned to the source (or just deleted). Likewise, any payment that the customer advanced to a provider must be returned.

### Successful Exchange in Example 2 with Direct Trust

Certain exchanges that are infeasible without direct trust become feasible once it is added. The conjunctive example (Figure 2) was previously infeasible. Here, we consider the same example with S1 directly trusting B1. Moreover, for the sake of simplicity, we assume that the deadlines are not constraining, and, therefore, may be ignored. Under these conditions, the exchange is feasible. The initial steps are the same, up until S1 has discovered that it has D1, the document required to fulfill B1's request. Because S1 trusts B1, the document is sent directly to B1 rather than to T2. Since the exchange is ongoing (B1 has not yet received payment for D1), B1 is not obligated to pay S1 for D1, but will pay for it if and when C buys the document. When B1 gets D1, it can send it to the requesting customer. Since there is no trust relationship between B1 and C, the document is sent to trusted intermediary T1.

When T1 receives the document, it recognizes that no payment for this document has yet been received, so it notifies C that when payment comes the document will be sent. This notification is stronger on two accounts than the payment request that B1 made in the conjunctive example above. First, it comes from a trusted intermediary rather than an untrusted agent. Second, it promises that the desired document is already held by the agent, rather than merely that money is required to attempt to obtain the document.

Given this stronger promise, C is willing to assume that it can obtain D1 later at its convenience (which it will do only if it can also get D2). Under this condition, C is willing to advance payment to T3 for D2. As such, C faces the same circumstances in obtaining D2 which it faced in the simple first example. The algorithm generates the same sequence of steps which allowed C to obtain the document in that example, and it results in C obtaining D2. With D2 in hand and D1 at T1, C is almost finished. C pays T1 in order to obtain D1.

T1 completes the exchange, sending D1 to C, and sending the payment to B1. When B1 gets the payment, it checks to see if it owes money for any of the source documents. Since D1 was received on trust, now that B1 has been paid for it, the obligation to re-pay S1 is active. B1 sends the money to settle up with S1 for the document that S1 sent due to direct trust.

### Soundness

Direct trust weakens the restrictions that agents place upon the exchanges. They no longer insist that they receive complete protection against deviant behavior from all the principals in the exchange. In particular, there are certain agents who are directly trusted, who are permitted greater latitude than other untrusted agents. A source will send a document to a directly trusted principal, even before payment has been guaranteed, believing that the recipient will eventually pay for the document or return it in good faith. A customer who directly trusts a source is willing to send payment before receiving the goods.

**Theorem 1:** *Any sequence of actions produced by the algorithm modified for direct trust is riskless with respect to the direct trust safety conditions.*

We show in turn that each of the four unsafe conditions cannot arise in a sequence of actions generated by the algorithm.

1. *Whenever a customer sends money, it always receives the document or a refund:* In the algorithm modified to permit direct trust, the customer will occasionally send payment directly to a trusted provider (in addition to trusted intermediaries). If the provider is able to acquire the document, the provider will enter a state where the document is sent directly to the customer. If the provider fails to obtain the desired document, the money is returned to the customer. Each failure condition such as an unknown document, deadline expiration, or a deadlock due to conjunctions causes the provider to return any funds it has been advanced on trust. Therefore, any sequence the algorithm generates will ensure that whenever a customer spends money without having it refunded, the customer will get the requested document.
2. *Whenever a provider sends a document, it always receives payment or the document is returned:* If the document in sent via a trusted intermediary, the document is available to the customer only when the payment is also sent to the provider. If the document is sent directly to a trusted customer, the provider receives payment when the customer obtains all of the desired documents if the customer is the ultimate customer, or is paid for them, if the customer is a broker. If the customer is unable to get the rest of the documents in the conjunction, the document sent on trust is returned when the failure is detected. Therefore, any sequence the algorithm generates will ensure that whenever a provider sends a document without having it returned, the provider will get receive payment.
3. *A customer never buys half a conjunction:* and
4. *A broker never buys a document without being able to re-sell it:* Direct trust does not impact the decisions of agents about purchasing documents. Therefore, the conservative policy followed in the original algorithm works for these cases as well.

Therefore, if none of the four unsafe cases may arise from the sequences suggested by the algorithm, any sequence suggested by the algorithm is riskless. ■

## Conclusion

If all of the participants in a multi-party transaction trust a single universal agent, simple commit protocols can ensure the atomicity of transactions, and guarantee that they complete by a deadline. Real world systems however, such as the Internet, include parties from many different jurisdictions, and may not have a single trusted authority. The fully distributed algorithm that we have presented allows the completion of many exchanges preserving guarantees of safety for the participants. It also identifies other transactions which are infeasible, allowing no safe sequence.

## References

Authors. 1996a. Making trust explicit in distributed commerce transactions. In *Proceedings of the 16th International Conference on Distributed Computing Systems*, 270–281.

Authors. 1996b. A sound and complete distriubted algorithm for distributed commerce transactions. Technical Report SIDL-WP-1996-0040, Institution.

Authors. 1997. Distributed commerce transactions. Submitted to *AAAI-97*.

Lamport, L. 1978. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM* 21(7):558–565.