UNENFORCED E-COMMERCE TRANSACTIONS

Electronic commerce currently relies on traditional third-party enforcement mechanisms that don't necessarily translate well to cyberspace. Chunking algorithms offer one possibility for unenforced transactions between self-motivated agents.

TUOMAS W. SANDHOLM
Washington University

In conventional commerce, deals are usually enforced by law. For example, if a car dealership fails to deliver an automobile after a customer has paid for it, the customer can resort to litigation. Such enforced protocols, however, are problematic in electronic commerce. First, adequate laws for e-commerce may be lacking, or the transacting agents (human or computational) may be governed by different laws—for example, they may be sited in different countries. Further, the laws that do exist might not be strictly enforced, or resorting to enforcement might be impractically expensive. Finally, an e-commerce party may vanish at any point in time—for example, by logging out—thus making laws unenforceable unless the vanished agent represents some real-world party, and the connection between the agent and the accountable real-world party can be traced.

Current e-commerce technology is based on enforced transactions. The problems of traceability and trust are being tackled, for example, by establishing trusted third parties like banks and credit-card companies, as well as by attempting to build cybercommunities of trust. The infrastructure for e-commerce among computational agents is also being developed along this line. For example, Telescript technology strives to strictly and accountably tie each computational agent to the real-world party that it represents.

This article describes a method of carrying out e-commerce transactions that doesn't require enforcement. The method enables trans-

1089-7801/97/$10.00 ©1997 IEEE

actions in settings where the parties cannot identify each other, or where litigation is not viable. It also allows computational agents to be more autonomous by not requiring them to be strictly tied to the real-world parties they represent. In cases where this type of unenforced exchange is possible, it is preferable to the strictly enforced mode of exchange because it saves enforcement costs (for example, litigation costs or operations costs of trusted third-party intermediaries) and it is insensitive to enforcement uncertainties.

The description here is a partial overview of a method presented in more detail elsewhere.[1] The method is based on managing the exchange between two agents—a supplier and a demander—so that the gains from completing the exchange (*cooperating* according to a contract) at any point are larger for both agents than the gains from terminating it (*defecting* the exchange prematurely by vanishing).

For example, defection might benefit a demander agent if the supplier agent has delivered much more than what the demander has yet paid for. By intelligently splitting the exchange into smaller chunks, the agents can avoid situations that motivate either of them to defect. We will call a sequence of deliveries and payments *safe* if neither agent is motivated to defect at any point in the exchange.

The method requires an exchange manager module to be added to each agent's architecture, as shown in Figure 1. This module is potentially different for each agent. Its role is to schedule an agent's deliveries (or payments) in a safe sequence. This is in the agent's self-interest. The exchange manager also provides the agent's negotiator module with information on whether a proposed contract can be carried out safely. Unless enforcement is guaranteed, the negotiator should only agree to contracts that can be executed without motivating the opponent to defect.

We will begin here by discussing one exchange in isolation, deriving the conditions for safe exchange and identifying an inherent restriction concerning the completion of the exchange. We can then look at exchanges in the context of future transactions in order to solve the completion problem.

SAFE EXCHANGE OF GOODS AND PAYMENTS

We can view the fulfillment of a mutual contract as one agent delivering goods—information, computation services, or other types—and another agent paying for them. The payments may be in money or in some other arbitrarily divisible commodity. The exchange proceeds on two axes:

- the portion of goods of the contract delivered by exchange step n is called $x_n \in [0, 1]$, and
- the cumulative payment so far is $p_n \in [0, p^{\text{contr}}]$,

where p^{contr} is the total payment specified in the contract.

The agents can make simultaneous moves and can observe the moves made so far by the other agent . They

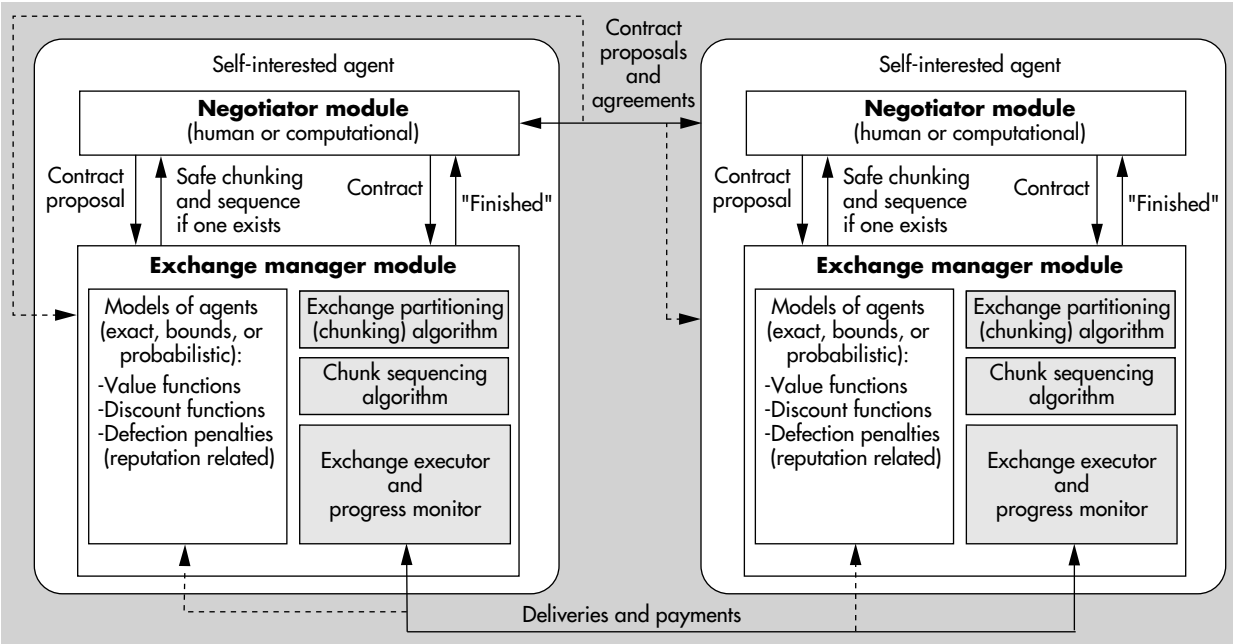


Figure 1. The architecture of two self-interested agents with special emphasis on the exchange manager modules. The negotiation system may include more than two agents. The exchange manager module gives the negotiation module feedback on whether a proposed contract can be carried out safely. The exchange manager also executes the exchange in each actual contract. The dashed lines show what information the exchange manager can use to update its models of other agents.

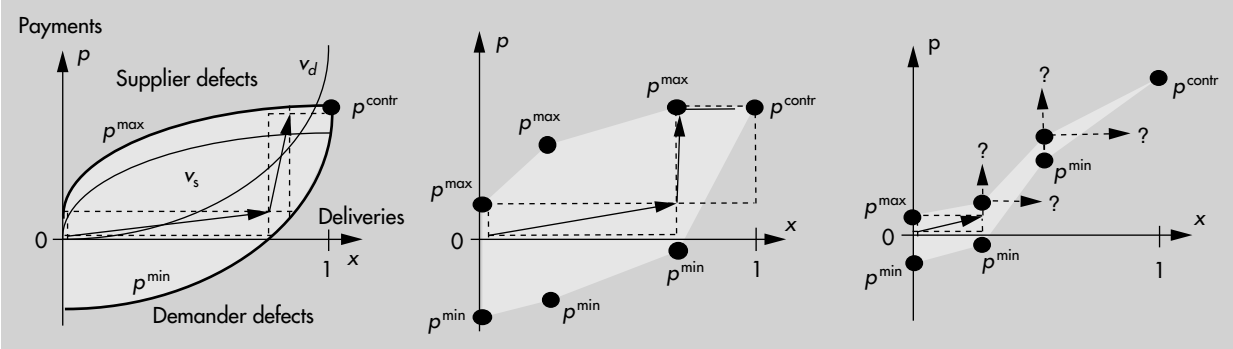


Figure 2. Examples of unenforced exchange: (left) safe exchange with uncountable goods, (middle) safe exchange of countable goods, (right) safe exchange of countable goods impossible. In each of the three cases, $p^{min}(1) = p^{max}(1) = p^{contr}$. The agents make simultaneous moves and try to stay inside the safe region at each step (that is, the supplier does not want to deliver too much at once, while the demander does not want to pay too much). Each agent wants to make sure that safety is maintained even if the opponent does not deliver or pay at all on the current move. That is why the exchange on the right cannot be safe.

value the goods x according to nondecreasing functions that are in equivalent units of payment p. The supplier's value function $v_s(x)$ describes how much cost the supplier incurs by generating and delivering x. The demander's value function $v_d(x)$ describes what the goods x are worth to the demander. Trivially, $x_0 = 0$, $p_0 = 0$, and $v_s(0) = v_d(0) = 0$.

Roughly speaking, a self-interested supplier agent will cooperate through the rest of the exchange from an arbitrary point (x, p) if its future compensation is at least as great as its future cost—that is, if $p^{contr} - p \geq v_s(1) - v_s(x)$. This assumes that the demander will finally increase cumulative payment to p^{contr}. It will shortly be shown that this is indeed the case. To facilitate presentation here, let's define $p^{max}(x)$ intuitively as the maximum cumulative payment that the demander can pay for a given cumulative delivery x without inducing the supplier to defect (see Figure 2):

$$p^{max}(x) \stackrel{def}{=} p^{contr} - v_s(1) + v_s(x).$$

Similarly, a self-interested demander agent will cooperate through the rest of the exchange from an arbitrary point (x, p) if the compensation it has to pay is smaller than or equal to its added value—that is, if $p^{contr} - p \leq v_d(1) - v_d(x)$. This assumes that the supplier will finally increase total delivery to 1, which we will shortly see to be the case. Now, $p^{min}(x)$ is defined as the minimum cumulative payment that has to be made for a given cumulative delivery x so that the demander is not induced to defect:

$$p^{min}(x) \stackrel{def}{=} p^{contr} - v_d(1) + v_d(x).$$

If the agents do not know each other's value functions, they can proceed using bounds they do know. The supplier is safe using an upper bound for $p^{min}(x)$, which equates to a lower bound for $v_d(1)$ and an upper bound for $v_d(x)$. The demander is safe using a lower bound for $p^{max}(x)$. Although the agents are safe using these bounds, the use of bounds disables even possible exchanges if the bounds are too far off from the actual values $p^{min}(x)$ and $p^{max}(x)$.

Now, two exchange strategies are formulated: one for the supplier and one for the demander. Relative to Figure 2, we can define the strategies as follows:

Definition 1 (Supplier's strategy): At any step of the exchange, if the exchange is within the safe region (gray region in Figure 2), then supply as much as possible while keeping the exchange in the safe region—assuming that the demander does not pay at all on this step. If the exchange is not in the safe region, exit.

Definition 2 (Demander's strategy): At any step of the exchange, if the exchange is within the safe region, then pay as much as possible while keeping the exchange in the safe region—assuming that the supplier does not supply at all on this step. If the exchange is not in the safe region, exit.

The results regarding safe exchange are slightly different for countable and uncountable goods.

Countable Goods

Countable goods are goods that are inherently split into atomic chunks. The chunks cannot be further split—that is, they are not arbitrarily divisible. In this section, let's assume that the delivery order of the chunks is fixed.

Consider an example implementation called the Traconet (TRAnsportation COoperation Network) multiagent system.[2] Traconet used agents to represent dispatch centers negotiating whose vehicles should transport which parcels. Taking care of one parcel is an atomic chunk because the task cannot be split. A contract between two agents might involve multiple tasks (to avoid local optima in distributed

task allocation), so the total exchange could be split into smaller parts. For example, the demander could first pay part of the contract price and the supplier would then deliver one parcel. After that the demander would pay some more and the supplier would deliver another parcel, and so on.

Some transactions can be carried out without enforcement while others cannot. The following condition precisely characterizes what transactions can be executed safely:

Condition 1 (Safe transaction possible): For every two consecutive amounts x and x' of cumulative delivery, $p^{\max}(x) \geq p^{\min}(x')$.

Note that consecutive amounts are amounts that differ by just one chunk. This consideration is independent of the agents' strategies. At any step of the exchange, the supplier can deliver the next chunk, a number of next chunks, or nothing.

I have shown elsewhere that if Condition 1 holds, the supplier's strategy and the demander's strategy are best responses to each other—that is, neither agent is motivated to deviate from its strategy given that the other party does not deviate.1 In the same work, I also show that this best-response (stability) property not only holds for the beginning of the exchange, but also prevails throughout it. (In terms of game theory, this means that the two strategies are in subgame perfect Nash equilibrium.[3,4]) Furthermore, the proposed strategies lead to completion of the exchange (see Figure 2, middle). Finally, there is no other pair of strategies that satisfies the stability property and completes the exchange in fewer steps.

It was also shown that the required conditions are minimal in the sense that if Condition 1 does not hold, no strategies lead to completion of the exchange while maintaining safety.

RELATED RESEARCH

The basic idea of enhancing cooperation by making the present less important than the future has been suggested, for example, in Axelrod.[1]

Telser[2] has analyzed safe exchanges of goods and payments in a setting similar to the one presented here. However, his work differs in many respects. For example,

- it analyzes a sequence of exchanges as a repeated game where the end of the sequence is not known, except probabilistically;
- his agents do not evaluate the option of breaching within an exchange and can therefore only defect between complete exchanges in the sequence; and
- he does not consider how to split the exchange into chunks, or how to sequence the chunks.

Computerized negotiation has mostly been studied with respect to *ex ante* rationality: what contracts seem desirable to the agents and are stable (in the sense that agents do not want to deviate from abiding by the contract given that other agents do not deviate) *before* the contract is carried out (for example, see Rosenschein and Zlotkin,[3] Sandholm and Lesser,[4,5] and Tennenholtz and Moses[6]). The work presented here suggests that contracts should also fulfill the condition of *interim* rationality: abiding by the contract should be desirable to the agents at each step of the contract.

Interim stability conditions were studied in multiagent planning without side payments in Brainov.[7]

Splitting an exchange into chunks in order to avoid defection has analogs in nature. Even very simple animal species have evolved to use such splitting. For example, the mating process of the black hamlet, a hermaphrodite fish, follows a similar protocol. The black hamlet carries both sperm and eggs simultaneously, but cannot fertilize its own eggs. The goal of one fish is to let another fertilize its eggs in exchange for allowing it to fertilize the eggs of the other fish. However, sperm is abundant while eggs are scarce; so if one fish lays all its eggs, the partner would fertilize them and swim away while saving its own eggs for transactions with others. To avoid such defection, one fish lays a few eggs, and the other fertilizes them. Then the other lays a few eggs, and the former fertilizes them, and the transaction continues in such small chunks.[8]

REFERENCES

1. R. Axelrod, *The Evolution of Cooperation*, Basic Books, 1984.
2. L.G. Telser, "A Theory of Self-Enforcing Agreements, *J. Business*, Vol. 53, No. 1, 1980, pp. 27-44.
3. J.S. Rosenschein and G. Zlotkin, *Rules of Encounter*, MIT Press, Cambridge, Mass., 1994.
4. T.W. Sandholm and V.R. Lesser, "Coalitions Among Computationally Bounded Agents," *Artificial Intelligence*, Vol. 94, No. 1, 1997, pp. 99-137; special issue on Economic Principles of Multiagent Systems.
5. T.W. Sandholm and V.R. Lesser, "Advantages of a Level Commitment Contracting Protocol," *Proc. Nat'l Conf. Artificial Intelligence*, MIT Press, Cambridge, Mass., Aug. 1996, pp. 126-133.
6. M. Tennenholtz and Y. Moses, "On Cooperation in a Multi-Entity Model," *Proc. 11th Int'l Joint Conf. Artificial Intelligence*, Morgan Kaufmann, San Francisco, Calif., 1989, pp. 918-923.
7. S. Brainov, "Deviation-Proof Plans in Open Multiagent Environments, *Proc. 11th European Conf. Artificial Intelligence*, John Wiley & Sons, New York, N.Y., 1994, pp. 274-278.
8. J. W. Friedman and P. Hammerstein, "To Trade, or Not To Trade; That Is the Question," in *Game Equilibrium Models I: Evolution and Game Dynamics*, R. Selten, ed., Springer-Verlag, Berlin, 1991, pp. 257-275.

What are some practical implications of Condition 1? First, safe exchange requires the supplier to value the cumulative delivery less than the demander does, and the contract price is somewhere in between these two valuations. Second, safe (isolated) exchange is impossible if the agents would be better off by making the contract for a smaller amount of goods. Third, fulfillment of Condition 1 is facilitated by $p^{\max}(x)$ being high and $p^{\min}(x')$ being low. In other words, safe exchange is enhanced if the supplier incurs most of its cost from the early portion of the exchange; it is hindered if the demander acquires most of its value from this portion. Luckily, in practice, the supplier usually does have setup costs and the demander often acquires most of the value from the completion of the exchange.

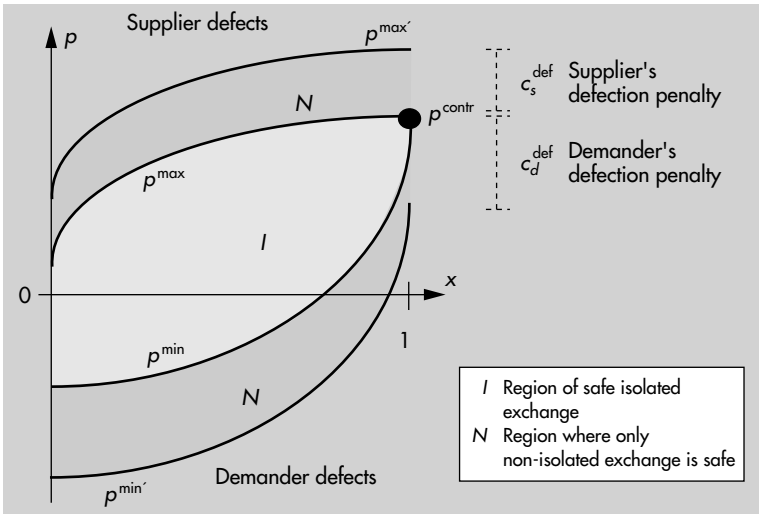


Figure 3. Defection penalties of nonisolated exchange give leeway to safe moves.

One problem with the method is that safe exchange requires at least one agent to be able to generate and deliver its last chunk without any expense. This is sometimes possible—for example, when the supplier has had to acquire a number of the last deliverables atomically. In this case, the cost of acquiring the deliverables can be entirely attributed to the first one—assuming negligible costs of physical delivery.

This might not occur very often in practice (see Figure 2, right). Intuitively, when no future benefit can be gained from exchanging, agents are better off defecting on the current move. If this problem occurs in an isolated exchange of a finite number of countable goods, it spoils the entire exchange. On the last move the supplier does not want to increase cumulative delivery to completion, because the demander would defect. Similarly, the demander does not want to increase cumulative payment because the supplier would defect.

Since both agents know that the last part of the exchange will not take place, they can analyze the exchange as if it did not have the last part, but then the second-to-last part has the same problem. Unless at least one agent can deliver that part without cost, neither agent wants to initiate it. This backward induction can be carried out up to the first exchange. In this case, neither agent will make any move, and the exchange will not take place.

Theoretically, there can be an infinite number of countable goods. In such cases, the backward induction argument does not apply because there is no point where an agent can say that the next move is the last.

Nor does backward induction apply to uncountable goods.

Uncountable Goods

Uncountable goods—such as gasoline, coffee, or monetary currency (without rounding, for example, to cents)—are goods that are not split into given fixed chunks but can be split arbitrarily. The results regarding safe exchange are almost analogous to the case of countable goods. The difference is that for uncountable goods, the conditions that allow safe exchange (Condition 1) are less strict.

Specifically, the condition does not have to hold for points of the exchange that are predetermined by the sizes of the indivisible chunks. Instead, the agents can alter the step size continuously. The strategies discussed earlier can be used to complete the exchange as long as there is some possible safe move of finite size that either agent can make from any safe point of the exchange.[1] If this relaxed condition is satisfied, the strategies are stable and the exchange is completed in the minimal number of steps.

Isolated safe exchange can also be problematic in the case of uncountable goods because it requires at least one agent to be able to generate and deliver its last chunk without cost. This problem is less severe than in the case of finitely many countable goods because the exchange can be made to *approach* completion. For example, using the strategies defined earlier, the size of the deliveries on the path of partial exchanges can decrease to approach zero—thus making the number of deliveries infinite (Figure 2, left). The agents can reach a cumulative delivery that is arbitrarily close to completion because the backward induction argument does not hold: There is no particular exchange step at which the agents could reason that neither will make a move.

NONISOLATED EXCHANGE

With both countable and uncountable goods, the problem of requiring at least one agent to be able to deliver the last part of an exchange without costs can be overcome by considering possible future interactions. An agent often interacts with other agents more than once, and current interactions may affect future options. For example, if an agent defects in the current exchange, its counterpart may not want to take on

future contracts with that agent. Moreover, the counterpart can notify other agents that the agent defected. Thus, the agent's interactions with third parties may also be hindered by defecting in the current exchange. There may also be a chance that the defector will be caught and penalized by law.

The hindering effect a defection may have on future contracts can be thought of as an extra cost for the defector. This cost may motivate agents to cooperate in the current exchange even if defection would be rational when the exchange is considered in isolation. The methods for calculating defection impacts on the future are beyond the scope of this article. We assume that both agents know their own and their opponent's defection costs. (Note that if an agent does not know the opponent's defection cost, the agent can use a lower bound for that cost. This way the agent is safe; but again, if the bound is too far off, even possible exchanges may be disabled.)

We can denote the supplier's defection cost by c_s^{def} and the demander's by c_d^{def}. The defection costs can be incorporated into the analysis of safe exchange by redefining $p^{\max}(x)$ and $p^{\min}(x)$ (see Figure 3):

$$p^{\max'}(x) \stackrel{\text{def}}{=} p^{\max}(x) + c_s^{\text{def}}$$

$$p^{\min'}(x) \stackrel{\text{def}}{=} p^{\min}(x) - c_d^{\text{def}}$$

The stability results, exchange-completion results, and step-minimality results from above apply to this case. The defection costs give leeway to the exchange, thus possibly enabling safe exchange to be completed even if neither agent can generate and deliver its last chunk for free. So the safe-exchange method is most useful when there is some cost to defection, but not enough to allow the exchange to occur safely in one step.

With uncountable goods, even the smallest reputation-related defection penalties suffice to allow safe completion of the exchange. With countable goods, if the smallest possible atomic chunks are large, the defection penalties must also be large for the supplier to safely deliver the last chunk and the demander to safely make its last payment.

THE ROLE OF TIME IN SAFE EXCHANGE

To summarize a detailed study of the role of time in safe exchange,[1] agents are motivated to carry out isolated unenforced exchanges immediately if the supplier discounts payments more sharply (or equally) than production costs and the demander discounts the value of goods more sharply (or equally) than payment. This condition on the supplier is rather natural. For example, in a stable environment, the supplier's current value of producing an item should remain constant, but payment is discounted. The condition on the demander is more stringent. It is realistic in the case where the demander's need for the goods is urgent.

If these conditions on discounting do not hold, the outcomes vary. The exact forms of the discount functions define whether the exchange can be carried out safely immediately, later (different moves in the exchange may be postponed different amounts), or not at all.

Time discounts reduce the viability of taking advantage of nonisolated exchange because one of the parties might be motivated to postpone indefinitely. In such cases, immediate moves can be stimulated by deadlines or lateness penalty schedules, which themselves need not be enforced.

DELIVERY SEQUENCING

So far we have discussed exchanges in which the delivering order of the goods is fixed beforehand. (An analogous situation occurs when the order of delivery does not matter because all goods are equivalent.) In this section we analyze an exchange where partial deliveries—individual countable goods, or atomic chunks of countable or uncountable goods—can be delivered in any order, as long as all of them get delivered.

The goal is to sequence the chunks in a way that allows safe exchange: Some sequences do while others do not. The problem differs significantly based on whether the chunks to sequence are independent or not.

Sequencing Independent Deliveries

A *chunk-sequencing algorithm* is an integral part of a self-interested agent's exchange manager module (see Figure 1). Here it is assumed that the chunks are independent, that is, that

- the demander's added value from one chunk does not depend on the other chunks delivered so far, and
- the supplier's cost for delivering a chunk does not depend on other chunks delivered earlier.

This lets us associate each chunk c with two values, $\Delta p_c^{\max}$ and $\Delta p_c^{\min}$, that fully characterize how much the maximum and the minimum cumulative payments change as c is delivered.

For example, an agent could make a contract to carry out a number of matrix multiplications. Multiplying two matrices neither facilitates nor hinders multiplying some other two, so the chunks are independent with respect to the supplier. The chunks are truly independent if they are also independent with respect to the demander—based on the uses of the multiplication results.

Call a delivery sequence safe if it satisfies Condition 1. I have developed a fast, greedy algorithm that finds a safe ordering if one exists. The algorithm takes the following inputs:

- a set of chunks C,
- a vector of $\Delta p_c^{\max}$ values,
- a vector of $\Delta p_c^{\min}$ values,
- the contract price p^{contr}, and

- the defection penalties ($c_s^{def} = c_d^{def} = 0$ in the case of isolated exchange).

The algorithm outputs a safe sequence of the chunks if one exists or reports that the problem is not solvable if there is none. The actual algorithm is nontrivial and presented in detail elsewhere.[1]

I tried several greedy algorithms to solve this sequencing problem, starting with the intuitive ones. Most of them do not guarantee that a safe sequence is found even if one exists. For example, counterexamples having just two chunks can defeat algorithms that greedily pass only forward and maximize $\Delta p_c^{max} - \Delta p_c^{min}$ or minimize Δp_c^{min}. In other words, these algorithms do not find a safe sequence even for some very simple problem instances. On the other hand, my algorithm has been proven to find a safe sequence if one exists. Furthermore, it runs in quadratic time in the number of chunks.

Sometimes the division of the exchange into chunks is not externally fixed but can be decided by the agents—for example, at contract time. For this reason, an *exchange-partitioning (chunking) algorithm* should be added to the architecture of the exchange manager module (see Figure 2). Exchange chunking can be done top-down by generating a chunking and then running the chunk-sequencing algorithm to test its safety. If the chunking is not safe, it can be refined by splitting chunks further. Splitting is monotonic in the sense that no split can make a safe exchange unsafe. Therefore this chunking algorithm does not need to backtrack.

The top-down method can be used for uncountable goods also. The minus side of the approach is the need to guess the splits. If they are guessed badly, more chunks may be generated than are necessary to enable safe exchange.

One bottom-up approach for chunking is to sequence the smallest possible atomic chunks using the chunk-sequencing algorithm. Using the pair of exchange strategies defined earlier, the agents may then be able to deliver multiple atomic chunks at any step of the exchange without changing the order of the chunks. Bottom-up chunking requires no guessing of splits, but it can be computationally complex if the number of smallest possible chunks is large. It cannot be applied to uncountable goods because the number of smallest possible chunks is infinite.

Sequencing Interdependent Deliveries

Sometimes partial deliveries are interdependent. The value of a chunk may depend on which chunks have been delivered before it. In manufacturing, for example, the first products can be thought of as more costly than subsequent ones because the fixed costs (rent, acquired equipment, and so on) can be associated with the earlier products.

Similarly, a data-retrieval agent may incur large costs in searching for certain information. Once the information is found, subsequent searches of related information are less expensive. The demander may also value a chunk differently depending on the other chunks delivered so far. In the aforementioned Traconet system,[2] the chunks (transport tasks) were interdependent for both supplier and demander. Transporting a parcel often affects the marginal cost of transporting others; for example, a vehicle may be able to carry two parcels to adjacent locations, thus reducing the marginal cost of both tasks. Conversely, one parcel may fill up the vehicle so that another task must be handled by a more costly vehicle. Some contracts involve multiple tasks. So if our safe-exchange mechanisms had been used, sequencing of interdependent chunks would have been required.

In general, interdependent goods cannot be sequenced in polynomial time in the number of chunks if the algorithm is required to find a safe solution when one exists. Just representing the problem requires $\Theta(2^{|C|})$ space because for each set of chunks in the power set of all chunks, p^{max} and p^{min} must be represented; in the worst case, this information cannot be compressed.

Nevertheless, if the number of chunks per contract is small—as in Traconet—exponential search among sequences of chunks is viable. In such cases, the advantages of safe unenforced exchange often outweigh the extra computational load. Furthermore, there are some special cases of the interdependent chunks problem that may be solvable in polynomial time.

CONCLUSIONS

Exchanges between self-motivated agents can be transacted without third-party enforcement by splitting larger exchanges into smaller parts so that neither agent is motivated to defect at any point. The possibility of safe exchange depends on the demander's and the supplier's value functions for the goods. Safe exchange is enhanced if the supplier incurs most of its cost from the early portion of the exchange, while it is hindered if the demander acquires most of its value early. Luckily, in practice, the supplier usually does have setup costs and the demander often acquires most of the value from the completion of the exchange.

The chunking algorithms, chunk-sequencing algorithm, and exchange strategies presented here can be used even if the supplier must deliver multiple items of different goods. However, further work is required to develop algorithms for settings where the demander's order of payments (made in money or some good) matters as well. This problem may arise if both agents are to exchange multiple items of different goods, or even when the demander pays using a single good that is not arbitrarily divisible. A further complication is that an agent's utility may depend on which combinations of items it has received and delivered.

So far I have assumed that the cost of splitting the exchange into chunks (which may be inherent to the splitting or caused by the increased transaction costs from having to pay and deliver in more than one step) is negligible compared to the

cost of producing the goods and the value of the payments. The cost of splitting the exchange is therefore not incorporated in the model. It follows that the agents' payoffs do not depend on the number of chunks that the exchange is broken into. Such costs would motivate the development of safe chunking algorithms that provably minimize the number of chunks. In addition, the increased transaction costs would directly factor into the exchange strategies: Defection is more favorable if future transaction costs are large.

In general, there is a trade-off between using enforced and unenforced exchange mechanisms. Enforced protocols save the cost of splitting the exchanges into parts and delivering the parts separately, but they incur enforcement expenses such as litigation costs and the operation costs of trusted third parties. ■

ACKNOWLEDGMENTS

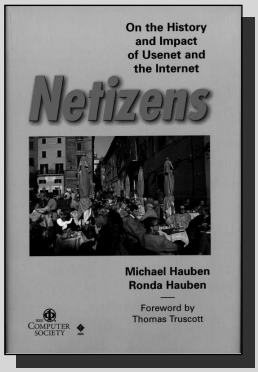
This work is supported by National Science Foundation Faculty Early Career Development (CAREER) Program award IRI-9703122 and NSF grant IRI-9610122. The author would like to thank Neil Immerman and Victor Lesser for their help.

REFERENCES

1. T.W. Sandholm, *Negotiation among Self-Interested Computationally Limited Agents*, doctoral dissertation, Univ. of Mass., Amherst, 1996; available at http://www.cs.wustl.edu/~sandholm/dissertation.ps.
2. T.W. Sandholm, "An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations," *Proc. Nat'l Conf. Artificial Intelligence*, MIT Press, Cambridge, Mass., 1993, pp. 256-262.
3. D. Fudenberg and J. Tirole, *Game Theory*, MIT Press, 1991.
4. R. Selten, "Spieltheoretische behandlung eines Oligopolmodells mit Nachfrageträgheit," [in German] *Zeitschrift für die gesamte Staatswissenschaft*, Vol. 12, 1965, pp. 301-324.
5. D.A. Lax and J.K. Sebenius, "Insecure Contracts and Resource Development," *Public Policy*, Vol. 29, No. 4, 1981, pp. 417-436.

Tuomas Sandholm is assistant professor of computer science at Washington University, where he heads a six-person research group on multiagent systems. His research interests include artificial intelligence, multiagent systems, game theory, machine learning, resource-bounded reasoning, electronic commerce, and network pricing. He has worked on two fielded AI systems: a pension law expert system and a large-scale transportation optimization application. Sandholm received an MS with distinction in industrial engineering and management science from the Helsinki University of Technology, Finland, in 1991. He received an MS and a PhD in computer science from the University of Massachusetts at Amherst in 1994 and 1996, respectively.

Readers may contact Sandholm at the Dept. of Computer Science, Campus Box 1045, Washington University, One Brookings Drive, St. Louis, MO 63130-4899; e-mail sandholm@cs.wustl.edu. The URL for his home page is http://www.cs.wustl.edu/~sandholm.



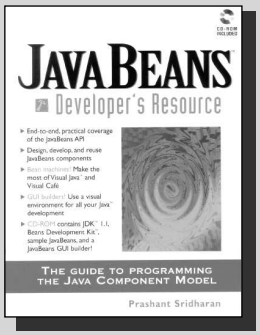
A fascinating account of the past, present, and future of the Internet

Netizens

On the History and Impact of Usenet and the Internet

by Michael Hauben and Ronda Hauben

Netizens looks at the creation and development of the global computer network. The book is a detailed description of the Net's construction and a step-by-step view of the past, present, and future of Usenet



JavaBeans

Developer's Resource

by Prashant Sridharan

The Java Beans API will be pivotal in helping companies transition from monolithic applications to secure collections of distributed, networked mini-applications. Learn all you need to know to design, develop, and use Java components—straight from a JavaBeans developer.

Call toll-free:

Online Bookstore:

IEEE COMPUTER SOCIETY