In *Proceedings of the Fourth International Conference on Multiagent Systems,*
July 2000

Distributed Quiescence Detection in Multiagent Negotiation*

Michael P. Wellman   William E. Walsh
University of Michigan Artificial Intelligence Laboratory
1101 Beal Avenue, Ann Arbor, MI 48109-2110 USA
{wellman, wew}@umich.edu

Abstract

In a distributed multiagent negotiation involving multiple issues, it is often desirable to finalize deals only when all related issues are resolved. However, detecting that a multiagent negotiation has reached a globally quiescent state can be a nontrivial task in a distributed, asynchronous system. We present a quiescence detection protocol based on the Dijkstra-Scholten algorithm for distributed termination detection. The protocol operates as a layer on top of an underlying mediated negotiation protocol. If agents conform to the detection protocol, the detection process terminates iff the negotiation is quiescent. We discuss agent incentives to deviate from the protocol, and describe extensions that enforce adherence with respect to the most significant potential deviations.

1. Introduction

Agents with distinct interests or knowledge can benefit by engaging in *negotiation* whenever their activities potentially affect each other. Through negotiation, agents make joint decisions, involving allocation of resources, adoption of policies, or any issue of mutual concern. Multiple related issues are typically negotiated at once, with each negotiation issue involving multiple agents.

As an example, consider a building contractor, who negotiates deals to perform various construction projects, and negotiates with skilled trades agents and suppliers for labor and materials. These negotiations are highly interdependent, as the labor and material should match that required for contracted projects, and the profitable prices in one realm depend on the prices obtained in others. Moreover, given fixed capacities, the desirability of obtaining one (building/employment/procurement) contract depends on whether it can obtain others, and at what terms.

Given that these negotiations proceed simultaneously, the agent faces a difficult strategic problem in managing its commitments. It would not want to finalize contracts with its labor before it can establish profitable building contracts, nor would it want to commit to projects for which it cannot acquire the necessary resources. Engaging in commitments with respect to one negotiation before others are resolved inherently entails substantial risk, but hesitation to propose such commitments can preclude participation in complex activities.

A common approach to mitigate this problem is to organize the negotiation process so that it iteratively admits tentative (or progressively strengthening) commitments, and reveals intermediate information about the status of individual issues being negotiated. Deals are not finalized until the overall negotiation reaches a *quiescent* state.

However, detecting quiescence can be a nontrivial task in a distributed, asynchronous system. Quiescence is a global property, yet each participant has only a local view of the negotiations it is directly engaged in. Information about the state of the system can be disseminated only via messages, which may be arbitrarily delayed. Additionally, we prefer that messages signalling quiescence follow the natural channels of the negotiation protocol, rather than some special-purpose paths to a global information aggregator.

In this paper, we present a protocol for quiescence detection based on a well-established method for distributed termination detection: the Dijkstra-Scholten algorithm. The protocol operates as a layer on top of an underlying negotiation protocol. This approach is highly general, as we require only that the underlying negotiation be *mediated*. Mediators facilitate negotiation by managing the flow of information and enforcing

*A previous version of this paper was presented at the *AAAI-99 Workshop on Negotiation: Settling Conflicts and Identifying Opportunities.*

negotiation rules.[1] As we show, mediators can also help to enforce the quiescence detection protocol.

In the next section, we present the underlying abstract model of mediated negotiation. We then describe the Dijkstra-Scholten algorithm and its application to negotiation protocols. In Section 4, we consider the incentives for agents to conform to the protocol, and extensions that can inhibit them from deviating when it is in their interest.

2. Negotiation Model

Let $\mathcal{A}$ be a set of agents, and $\mathcal{M}$ a set of mediators, each managing a specific negotiation issue involving some subset of the agents. For example, a mediator might control the exchange of a particular resource type, or settlement of some designated concern. We denote the agents interacting with mediator m by $\mathcal{A}_m$. Inversely, $\mathcal{M}_a$ comprises the mediators agent a interacts with.

Definition 1 *A* negotiation network *is an undirected bipartite graph with vertices $V = \mathcal{A} \cup \mathcal{M}$ and edges E linking agents and mediators. For $m \in \mathcal{M}$ and $a \in \mathcal{A}$, there exists an edge $\langle m, a\rangle \in E$ iff $m \in \mathcal{M}_a$. (We could equivalently define edges in terms of the $\mathcal{A}_m$ sets.)*

Particular negotiation mechanisms may require that the network structure be fixed throughout the negotiation process. Typically, each mediator m knows $\mathcal{A}_m$, and each agent a knows $\mathcal{M}_a$.

For example, Figure 1 depicts a negotiation network where all agents interact with all mediators. In the negotiation network of Figure 2, the agents are organized in a three-level supply chain, with two agents (perhaps competing) at each level.

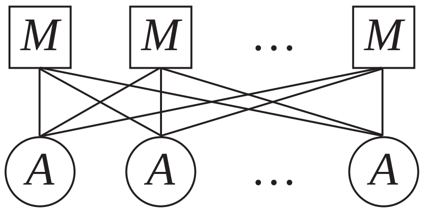

Figure 1. Complete-graph negotiation network.

The negotiation protocol comprises two general types of messages. Agents send `OFFER` messages to

[1] Tidhar and Rosenschein [13] argue, in the context of an augmented version of the CONTRACT NET, that mediators can also help to reduce agent communication and search costs.

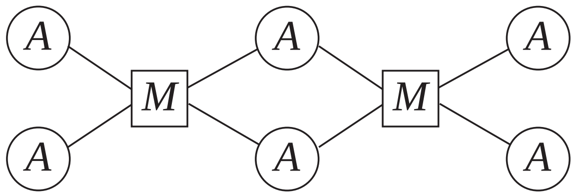

Figure 2. Supply-chain negotiation network.

mediators, and mediators send `NOTIFY` messages to agents. The particular form and content of these messages varies according to the domain-specific rules enforced by the mediators and the negotiation policies of the agents. For instance, in an auction-mediated negotiation protocol, an `OFFER` is typically a bid to buy or sell a good, and a `NOTIFY` may be an indication of the current going price (e.g., the highest bid so far). More generally, we allow both types to range over arbitrarily defined message spaces. We assume that a mediator m implements fixed rules for issuing a `NOTIFY` message to agent $a \in \mathcal{A}_m$, as a function of `OFFER` messages it has received from all agents in $\mathcal{A}_m$. Agent a executes its negotiation strategy, which dictates what `OFFER` message (if any) it transmits to mediator $m \in \mathcal{M}_a$ as a function of the history of `NOTIFY` messages received from mediators in $\mathcal{M}_a$.

Implicit in the preceding description is that once negotiation starts, behavior is completely determined by the messages passed within the negotiation network. That is, we assume that no external events intervene to cause changes in mediator rules or agent strategies.

Communication is reliable but asynchronous.[2] That is, all messages sent eventually reach their recipients, although we impose no bound on the delays. Note that even if all mediators and agents have deterministic behaviors, an overall negotiation may be nondeterministic due to this asynchrony.

We designate by $\mathsf{send}_i(j, msg)$ the action by entity i sending message msg to entity j. Similarly, i's action receiving msg from j is denoted by $\mathsf{receive}_i(j, msg)$.

Definition 2 *A negotiation protocol P is* quiescent *when:*

1. *All sent messages in P have reached their intended recipients. That is, if $\mathsf{send}_i(j, msg)$ occurs in P, then so does $\mathsf{receive}_j(i, msg)$.*

2. *No agent wishes to send another `OFFER` message, based on its received set of `NOTIFY` messages in P.*

[2] See Fagin et al. [5], for example, for a formal definition of *asynchronous reliable message passing* (a.r.m.p.) systems.

3. No mediator needs to issue further NOTIFY messages, based on its received set of OFFER messages in P.

Once a process has reached quiescence, the negotiation is complete, and the mediators can finalize the deals, settlements, or other outcomes negotiated, and commence the execution phase. The difficulty, of course, is that no individual entity (agent or mediator) can determine based on the negotiation messages it sends and receives whether the process is quiescent. At best, all it can tell is whether it is *locally quiescent*, that is, whether it has no more negotiation messages to send based on those received to that point. Under the asynchronous communication model, no finite "timeout" threshold on local quiescence is sufficient to ensure global quiescence.

3. Quiescence Detection

The quiescence detection protocol operates as a layer on top of the underlying negotiation protocol. The overall protocol augments the negotiation protocol (i.e., `OFFER` and `NOTIFY` messages as described in the previous section) with additional messages to manage the quiescence detection process.

3.1. The Dijkstra-Scholten Algorithm

Dijkstra and Scholten [4] proposed a termination detection algorithm which directly serves our purpose. The authors consider "diffusing computations": distributed processes where one entity, (the *source*) sends a message to another, thus activating it. Once activated, an entity may send messages to others, thus activating them, and may continue to send and receive further messages until it has no more to send. The diffusing computation terminates when no entities send any more messages.

We describe the essentials of the algorithm next; for a more complete description, analysis, and discussion, consult the original, or Lynch [7]. Starting from an initially quiescent state, a source becomes active, and sends a message to another entity. The newly activated entity makes note of which entity activated it, and proceeds with the protocol.

The behavior of generic (non-source) entity i in the protocol is presented schematically in Figure 3. In the initial quiescent state, i is inactive. It becomes active on receipt of a message from its *activator*, x. It then transitions into its active mode, and thereafter responds to every message received from any agent with an acknowledgment message, `ACK`. It also keeps track

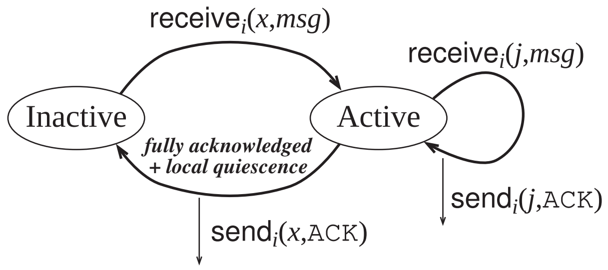

Figure 3. Schematic state diagram for non-source entity i in the Dijkstra-Scholten algorithm.

of how many of its own sent messages have been unacknowledged. This can be implemented by a *deficit* counter, which is incremented on sending a message (other than `ACK`), and decremented on receiving an `ACK`. Whenever it reaches a state where (1) all of its messages have been acknowledged, *and* (2) it is locally quiescent, it transitions back to the inactive mode, and sends an `ACK` message to its activator.

The outstanding activation relationships in this protocol form a tree, with the source at its root. The tree is extended whenever a new activation occurs, and it is resected whenever a leaf entity becomes locally quiescent. Once this resection process reaches the source (i.e., the source has received acknowledgments for all messages), the "diffusion" is complete, and the quiescence detection process terminates.

Observation 1 (Dijkstra & Scholten [4]) *The quiescence detection process terminates iff the underlying protocol is quiescent.*

3.2. Application to Negotiation

Since it is defined as a layer on top of another protocol, applying the Dijkstra-Scholten algorithm to our negotiation model is straightforward. Agents and mediators augment their behaviors by passing and tracking `ACK` messages according to the quiescence detection protocol. Their resulting behavior is essentially a composition of their basic negotiation behavior with the transition diagram of Figure 3.

For the case where negotiation is initiated by a single entity (agent or mediator), that entity plays the role of source, and is the detector of global quiescence. This case applies when the negotiation network can be considered initially quiescent, with all subsequent activity triggered by a state change in one entity.

Perhaps more commonly, negotiation begins in a state with multiple entities in locally non-quiescent states. For example, a negotiation situation might be triggered by some external event (e.g., a scheduled start, or announcement of a new opportunity), upon which several agents wish to send OFFER messages. To handle this case, we augment the negotiation network by introducing a special mediator, called the *quiescence detector*, q. $\mathcal{A}_q$ consists of all agents who are potentially active at initiation. Mediator q plays the role of source, and activates all of the relevant agents immediately on commencement of the negotiation.

Once q or another single-source entity detects global quiescence, it can disseminate the news through the negotiation network, informing the mediators and agents that they may proceed to execute negotiated deals.

3.3. Simultaneous Independent Negotiations

It is not always feasible or desirable to wait until an entire negotiation network has achieved global quiescence in order to begin executing deals. Indeed, in a comprehensive model, it may be that all of the agents of the world are ultimately connected by some path of mediator-agent relationships. Despite these links, it is likely that some areas of negotiation are completely or virtually irrelevant to others.

We can distinguish these independent negotiations for purposes of quiescence detection simply by maintaining separate identities for separately initiated negotiation protocols (perhaps involving distinct quiescence-detector mediators). Agents and mediators would maintain distinct quiescence-detection state (i.e., activation links and acknowledgment deficit counters) for each distinct protocol. All negotiation messages would need to specify which negotiation they are part of, and ACK messages would reflect back the negotiation identity of the messages they acknowledge. As usual, the source entity detects global quiescence, but limited to the particular negotiation initiated. It then informs involved entities that the negotiation is complete.

Even though separately-sourced negotiations correspond to independent quiescence-detection processes, interactions at the negotiation level could well cause dependence in their actual quiescence. Entities participating in multiple negotiations may choose to activate others based on some or all of these, and might predicate local quiescence on activity in all of the negotiations.

An interesting tradeoff arises for the case of loosely interdependent issues being negotiated. Treating them as part of the same negotiation process ensures that an overall quiescent state exists before executing some of the issues. On the other hand, separating them prevents one from necessarily being delayed by the other. Which concern predominates depends on such situation-specific factors as degree of interdependence, value of time, and likelihood and severity of delay. Agents must weigh these criteria in choosing which protocol to associate with each of their outgoing negotiation messages.

4. Deterring Deviation

The preceding discussion presumes that both agents and mediators will follow the specified protocol. Whereas the assumption is typically justified for mediators—which may be implemented or certified by some globally trusted authority—it is more questionable in the case of agents. Just as design of negotiation mechanisms must account for incentives faced by agents, we must consider these incentives in constructing any augmentation for quiescence detection [12].

4.1. Deviations

There are five distinct ways an agent could deviate from the quiescence detection protocol.

1. Failure to acknowledge a received message while in active mode.
2. Failure to acknowledge the activator upon reaching a fully acknowledged state.[3]
3. Premature acknowledgment of activator, that is, sending ACK while not in a fully acknowledged state.
4. Counterfeit activation, where the activating agent is not itself active in the specified negotiation.
5. Spurious acknowledgments not corresponding to received messages.

Deviation #5—spurious acknowledgment—is relatively easy to eliminate by technical means. By requiring specific references to the acknowledged message in ACK messages, entities can recognize whether an acknowledgment is meaningful. Doing so imposes some additional implementation overhead, requiring unique message identifiers and an acknowledgment state more complicated than deficit counters. This will typically be worthwhile in negotiation applications, and so we ignore this particular category of deviation henceforth.

[3] We need not consider the case that an agent is fully acknowledged but not locally quiescent, since by taking action (i.e., sending an OFFER) it will immediately enter a state of incomplete acknowledgment.

Note that we do not consider variation in behavior in the underlying negotiation protocol to be a form of *deviation*. Rather, we assume that the negotiation protocol itself does not restrict agents beyond what can be enforced by the mediators based on their available information. Agents typically have much discretion in their negotiation strategy, and this may well be influenced by the presence or absence of reliable quiescence detection. For example, an agent may choose not to activate a mediator if it is concerned about a potential delay in reaching global quiescence. This is perfectly compliant, and does not affect the correctness of the quiescence determination process.

Similarly, agents have discretion to send `OFFER` messages without having been activated by prior received messages. As long as the `OFFER` does not falsely claim to be part of an existing negotiation (in which case it constitutes a counterfeit activation deviation), such actions effectively amount to initiating a distinct negotiation process, as discussed in Section 3.3.

4.2. Incentives to Deviate

4.2.1 Failure to Acknowledge

By failing to send an `ACK` when appropriate (deviations #1 and #2), an agent effectively holds up the quiescence detection process. But since no negotiation messages are contingent on prior `ACK` messages, this does not at all affect the set of actions available to any other agent, or the information disseminated by mediators. Although it is conceivable that by delaying acknowledgments, the agent could affect the *timing* of quiescence detection in a systematic way, the underlying asynchronous communication model limits the predictability of these effects. Moreover, delaying acknowledgments is indistinguishable from accounted-for delays in communication. Thus, the set of possible runs of the negotiation protocol is unaffected by failure to acknowledge.

The only certain effect is that if the agent fails to acknowledge indefinitely, it can prevent quiescence from ever being detected, and thus the negotiated deals from being executed. This can be of direct benefit to the agent only if it is better off in the status quo than in its negotiated deals.[4]

This still leaves open the possibility that agents withhold acknowledgments in an attempt to influence the outcome of negotiations. Because the underlying negotiation protocol is unaffected by quiescence detection, this can occur only indirectly, due to sensitivity of agents' preferences on the time it takes to complete a negotiation. In particular, the *threat* that an agent may delay or block execution of negotiated deals causes other agents to modify their behavior—perhaps making concessions to the threatening agent. Similarly, delays in one negotiation may affect behavior in other related (but independent with respect to quiescence detection) negotiations.

4.2.2 Premature Acknowledgment

In a premature acknowledgment (#3), an agent informs its activator that it is quiescent, even though negotiation via mediators that it in turn activated may still be ongoing. Such a deviation could clearly cause the protocol to incorrectly detect global quiescence. Unfortunately, an agent may well have a strong incentive to deviate in this way.

For example, consider the negotiation network of Figure 4, an instance of the network of Figure 2. In this example, agent $\hat{A}$ was activated by mediator M_1, and in turn activated M_2, which activated three other agents. Although $\hat{A}$ considered its negotiations at M_1 and M_2 to be dependent at the outset, the negotiation could reach a state where $\hat{A}$ considers the result from mediator M_1 to be sufficiently favorable to outweigh the uncertainty in the outcome at M_2. By sending `ACK` to its activator, M_1, it may cause the source to detect quiescence, and thus finalize its favorable outcome. Waiting until the rest of the network is quiescent runs the risk that A_0 sends an `OFFER` to M_1 that diminishes the value of $\hat{A}$'s outcome.

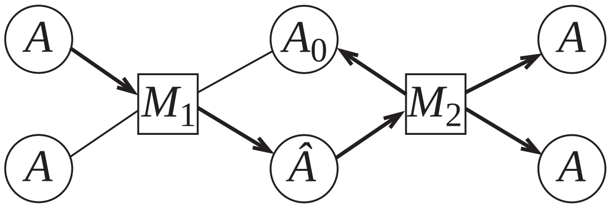

Figure 4. Agent $\hat{A}$ may have an incentive to deviate by prematurely acknowledging mediator M_1.

Although $\hat{A}$ has no way to know for sure that its improper acknowledgment will in fact result in a premature quiescence detection and improved outcome, the point is that by deviating from the protocol, it can in fact prevent another agent (A_0) from taking an allowed negotiation action. For this reason, we consider premature acknowledgment a significant potential deviation, and explore extensions of the protocol to counteract it.

[4] Mechanisms that guarantee outcomes such that no participants are worse off than initially are termed (ex post) *individually rational*.

4.2.3 Counterfeit Activation

Counterfeit activation (#4) serves much the same purpose as premature acknowledgment. Suppose $\hat{A}$ has an interest in P, though it is not currently active.[5] Then it may have an incentive to activate another agent A in P (via some mediator M). The reason is that if A is subsequently truly activated in P, it will ACK immediately, even if it is not locally quiescent. Thus, P may end prematurely, perhaps to the benefit of $\hat{A}$. The reasoning is very similar to the premature-acknowledgment deviation.

Fortunately, our remedy for premature acknowledgment also effectively prevents counterfeit activation.

4.3. Mediator Backbone

In addressing the potential incentives for agents to deviate from the quiescence detection protocol, we exploit the fact that agents communicate directly only with mediators, which can be presumed to faithfully execute the protocol. By effectively "short circuiting" the mediator-to-mediator paths through agents, we can eliminate the ability of agents to cut off negotiation through premature acknowledgment or counterfeit activation.

However, since mediators do not necessarily know all the pathways of potential interaction, we depend on the agents to set up these mediator-to-mediator links. We operate the protocol as usual, with the following modification. All OFFER messages carry with them an indication of which mediator activated the sending agent. If this OFFER activates the receiving mediator, then rather than consider the agent its activator, it treats the agent's activator (another mediator) as its own. Thus, it immediately acknowledges the agent, and notifies the agent's activator that it has another unacknowledged message (i.e., an outstanding activation).

To prevent deception by the agent, this notification could extend into a full-blown authentication process, where the two mediators validate that a proper activation link is being formed. The newly activated mediator would not act on the agent's message until establishing this activation link from its predecessor.

The result is a modified activation tree, where the mediator-to-mediator links form a backbone, and agents appear only as leaves. The modification introduces $O(|\mathcal{M}|)$ new communication channels, but retains the distributed communication structure of the original network. For consistency, we require that even single-source agent initiations be represented by a quiescence-detector mediator. For example, Figure 5 depicts a version of the supply chain network with activation paths through agents short-circuited by direct mediator-to-mediator links (cf. the activation pattern of Figure 4).

[5] For example, it may have been previously active in P, or it may be interested in P' which it knows is linked to P, indirectly, by some other agents.

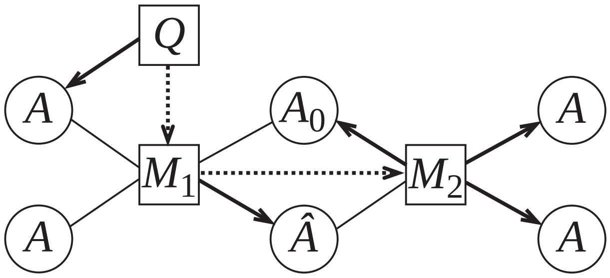

Figure 5. The supply chain network with a mediator-to-mediator activation backbone.

Observation 2 *In the augmented quiescence detection protocol, no agent can cause incorrect quiescence detection via premature acknowledgment or counterfeit activation.*

With the mediator backbone, the only possible deviations in the scheme are #1 and #2—failure to acknowledge. However, since agents appear only as leaves, they are never in a position of waiting for activation acknowledgments. Thus, the quiescence detection protocol calls for them to acknowledge all received, non-activating messages, as soon as they complete their local computation and message passing. To guard against failure or non-responsiveness of an agent, we might impose a timeout threshold beyond which the need for acknowledgment expires. This directly eliminates the possibility of deviation #1.

Although imposing such timeouts can lead to incorrect quiescence detection given asynchronous communication and uncertain computation time, the risk in doing so for message acknowledgments is much lower than for imposing timeouts on activation acknowledgments. It might therefore be worthwhile to accept this risk in order to deter anticipated delaying tactics.

To eliminate deviation #2, we require some further coordination within the mediator backbone. Specifically, mediators receiving OFFER messages acknowledge not only the sending agents, but also the activating mediators of those senders. As a result, the activating mediator can verify whether the agent is really in an unacknowledged state. After a designated activation-timeout period, it is up to the agent to either acknowledge or name the mediators it is waiting on. The reasonable activation-timeout period may be longer than the normal message acknowledgment timeout (to allow for the agent's substantive activity), but it still serves the purpose of bounding the possibility of delay.

Finally, the mediator backbone is also helpful with respect to disseminating the fact of quiescence, once detected. Mediator q sends a quiescence signal to mediators that it has communicated with (those it activated at some point in the negotiation), and these in turn do the same. Every relevant mediator m eventually receives the signal, and forwards to its known agents, $\mathcal{A}_m$, along with the result of the negotiation.

5. Applications

Auctions provide a direct example of mediators, and systems of related auctions an instance of the negotiation model described above. An agent sends *bids* (`OFFER` messages) to the auctions to specify its willingness to buy or sell, and an auction issues *price quotes* (`NOTIFY` messages) to indicate the current going price for a good. Auctions of related goods are sometimes run simultaneously in a coordinated fashion—as in the US FCC spectrum auctions [8]—to help agents manage their commitments. To ensure that any and all auctions close iff all agents have finished bidding in all auctions, operators typically resort to central control (e.g., synchronization); the techniques described here would achieve this property in distributed asynchronous environments as well.

In prior work [14], we apply a simultaneous ascending auction-based protocol to construct supply chains in task allocation networks with hierarchical dependencies. Consumer agents initiate the bidding, and subsequent `OFFER` and `NOTIFY` messages diffuse through the dependency network of agents and auctions. If some auctions were to close before global quiescence is reached, infeasible supply chains (producers do not acquire all inputs necessary to produce an output) could result. In an extension of this protocol to support quiescence detection, the activation and acknowledgments would follow the natural lines of mediator/agent communication, although not necessarily in the directions indicated by the buy/sell relationships.

In computational market approaches based on general-equilibrium systems (e.g., WALRAS [15]), the negotiation structure resembles that of Figure 1 (though complete connectivity is not required). Distributed price adjustment algorithms, such as *tatonnement*, serve as the underlying negotiation protocol, and quiescence detection determines when a *competitive equilibrium* has been reached. Andersson and Ygge [2], describe how equilibrium convergence can be accelerated by distributing auction computation for a good between multiple nodes in a tree structure. This approach extends our negotiation model in that messages pass between mediators in the tree, as well as between mediators and agents. The basic approach to quiescence detection applies in this more distributed context as well.

6. Distributivity, Feasibility, and Decommitment

One way to avoid all of the difficulties of distributed quiescence detection is to enlarge the scope of mediation, so that a single negotiation mechanism covers all related concerns. Combinatorial auctions, for example, have received much attention in this literature recently [1, 6, 11] as an approach for allocating multiple related goods. Whereas such mechanisms clearly play a role, we consider it equally clear that it is not plausible to couple under a single mechanism the negotiation of all potentially related issues. Choosing the scope of a negotiation mechanism presents tradeoffs between allocation quality, distributivity, and related factors [16]. Issues of latency and quiescence are particularly relevant to this choice.

Even in a completely distributed negotiation, quiescence detection may not be required to ensure feasibility. For example, in domains that do not include production (such as Sandholm's [10] generalization of Rosenschein and Zlotkin's [9] *task-oriented domains*), every locally feasible trade results in a globally feasible allocation. Trades can be executed immediately and independently of other trades and thus local quiescence is sufficient for feasibility.

We have emphasized the need for establishing global quiescence when issues are highly interdependent, as when the feasibility is contingent on their joint outcome. Even without strict feasibility constraints, however, hasty trades could preclude potentially better deals arising in the future, or could leave agents with goods that are not valuable, given their other holdings. Andersson and Sandholm [3] find that it is often possible to mitigate the effects of myopic behavior by allowing agents to decommit when better opportunities become available. They include penalties for decommitment to ensure that contracting and decommitment do not cycle indefinitely.

More generally, defining the commitment entailed by various negotiation actions is a central matter for designers of negotiation mechanisms. We would expect that varying levels of commitment will be widely useful in multiagent negotiation.

7. Discussion

The first contribution of this paper is a formulation of the distributed quiescence detection problem in multiagent, multi-issue negotiation, and a solution to that problem based on the Dijkstra-Scholten algorithm. The protocol works correctly given asynchronous reliable message passing, and follows the communication channels of the underlying negotiation.

The second contribution is an examination of the incentive effects of adding a protocol layer for quiescence detection. We show that agents will often have incentive to deviate from the basic protocol by prematurely acknowledging their activators, and propose an extension that forms a mediator backbone to prevent this behavior. We argue that the resulting augmented protocol potentially presents incentive for an agent to deviate by failing to acknowledge messages (the only deviation possibility left). Thus, it may sometimes be necessary to impose timeouts to avoid this behavior. This violates our respect for asynchrony, but in the version of the protocol with the mediator backbone, it imposes the deadline only for messages that are supposed to be acknowledged immediately anyway.

In the case of simultaneous independent negotiations, we generally leave it up to the agents to designate which negotiation process a sent message belongs to. However, in the augmented protocol with mediator backbone, the agent does need to establish that it has been activated by some mediator in negotiation protocol P. If it wants to start a new negotiation, it must invoke the permission of a quiescence detector mediator. Under what conditions the detectors should allow new negotiations to be initiated, or require that agents associate their `OFFER` messages with existing negotiations, is a policy matter worthy of study.

Acknowledgments

This project would not reliably quiesce unless we acknowledged our activator, Jeffrey MacKie-Mason, who prompted this investigation (which turned out to be a diffusing process indeed) with a question about termination. More reflexive thanks go to Terence Kelly and anonymous reviewers for helpful comments. This work was supported by a NASA/Jet Propulsion Laboratory Graduate Student Researcher fellowship and DARPA grant F30602-97-1-0228 from the Information Survivability program.

References

[1] A. Andersson, M. Tenhunen, and F. Ygge. Integer programming for combinatorial auction winner determination. In *Fourth International Conference on Multiagent Systems*, Boston, 2000.

[2] A. Andersson and F. Ygge. Managing large scale computational markets. In *31st International Conference on System Sciences*, pages 4–13, 1998.

[3] M. R. Andersson and T. W. Sandholm. Leveled commitment contracting among myopic individually rational agents. In *Third International Conference on Multiagent Systems*, pages 26–33, 1998.

[4] E. W. Dijkstra and C. S. Scholten. Termination detection for diffusing computations. *Information Processing Letters*, 11:1–4, 1980.

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

[6] Y. Fujishima, K. Leyton-Brown, and Y. Shoham. Taming the computational complexity of combinatorial auctions. In *Sixteenth International Joint Conference on Artificial Intelligence*, page 548, Stockholm, 1999.

[7] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[8] R. P. McAfee and J. McMillan. Analyzing the airwaves auction. *Journal of Economic Perspectives*, 10(1):159–175, 1996.

[9] J. S. Rosenschein and G. Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*. MIT Press, 1994.

[10] T. Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *Proceedings of the National Conference on Artificial Intelligence*, pages 256–262, Washington, DC, 1993. AAAI.

[11] T. Sandholm. An algorithm for optimal winner determination in combinatorial auctions. In *Sixteenth International Joint Conference on Artificial Intelligence*, pages 542–547, Stockholm, 1999.

[12] T. W. Sandholm. *Negotiation among Self-Interested Computationally Limited Agents*. PhD thesis, University of Massachusetts Amherst, 1996.

[13] G. Tidhar and J. S. Rosenschein. A contract net with consultants: An alternative architecture and experimental results. In *European Conference on Artificial Intelligence*, pages 219–223, Vienna, 1992.

[14] W. E. Walsh and M. P. Wellman. A market protocol for decentralized task allocation. In *Third International Conference on Multiagent Systems*, pages 325–332, Paris, 1998.

[15] M. P. Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research*, 1:1–23, 1993.

[16] M. P. Wellman, W. E. Walsh, P. R. Wurman, and J. K. MacKie-Mason. Auction protocols for decentralized scheduling. *Games and Economic Behavior*, to appear.