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. 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.

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 an equilibrium, or *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 negotiation rules. 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 the section on "Deterring Deviation", 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.

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

We thank Jeffrey MacKie-Mason, Terence Kelly, and anonymous referees for helpful comments. This work was supported by DARPA/ITO grant F30602-97-1-0228, and a NASA/JPL Graduate Student Researchers Fellowship. For an extended and updated version of this report, please contact the authors.

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.

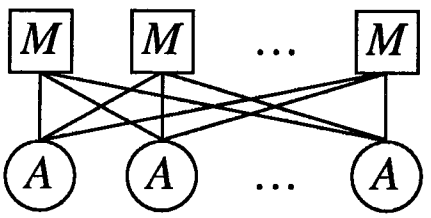

Figure 1: Complete-graph negotiation network.

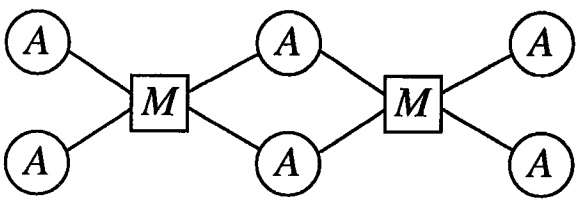

Figure 2: Supply-chain negotiation network.

The negotiation protocol comprises two general types of messages. Agents send `OFFER` messages to 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 $\mathcal{M}_a$.

Communication is reliable but asynchronous. 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 $\text{send}_i(j, msg)$ the action by entity i sending message msg to entity j. Similarly, i's action receiving msg from j is denoted by $\text{receive}_i(j, msg)$.

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

1. *All sent messages in P have reached their intended recipients. That is, if $\text{send}_i(j, msg)$ occurs in P, then so does $\text{receive}_j(i, msg)$.*
2. *No entity has any P messages to send, based on P messages received to this point.*

For the specific instance of a negotiation protocol, when the process is quiescent:

1. No agent wishes to send another `OFFER` message, based on its received set of `NOTIFY` messages.
2. No mediator needs to issue further `NOTIFY` messages, based on its received set of `OFFER` messages.

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.

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.

The Dijkstra-Scholten Algorithm

Dijkstra and Scholten (1980) 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 (1996). 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.

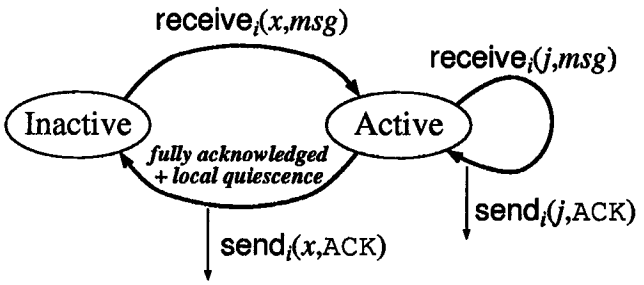

Figure 3: Schematic state diagram for non-source entity i in the Dijkstra-Scholten algorithm.

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 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:

- all of its messages have been acknowledged, *and*
- 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.

Proposition 1 (Dijkstra & Scholten (1980)) *The quiescence detection process terminates iff the underlying protocol is quiescent.*

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.

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 equilibrium 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.

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.

Deviations

There are four 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.[1]
3. Premature acknowledgment of activator, that is sending ACK while not in a fully acknowledged state.
4. Spurious acknowledgments not corresponding to received messages.

Deviation #4—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.

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.

[1]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.

Similarly, agents have discretion to send OFFER messages without having been activated by prior received messages. Such actions effectively amount to initiating a distinct negotiation process, as discussed above.

Incentives to Deviate

Consider first the failure-to-acknowledge deviations (#1 and #2). By failing to send an ACK when appropriate, 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. Since deals are typically voluntary, such a situation can often be avoided by design of the negotiation mechanism. Mechanisms that guarantee outcomes such that no participants are worse off than initially are termed *individually rational.*[2]

Proposition 2 *If the underlying negotiation mechanism is individually rational, then no agent can ensure a better outcome for itself by failing to acknowledge messages or activations as dictated by the quiescence detection protocol.*

This proposition 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 negotiations. This second indirect effect can occur only if actually dependent negotiations are being treated as independent for purposes of quiescence detection. Because linking of negotiations is generally within the agents' discretion, this situation cannot be prevented. Viewed from the perspective of

[2]Technically, if the negotiation game is one of incomplete information (the typical case), we require *ex post* individual rationality.

negotiation P, external factors (including other negotiations) can induce arbitrary time preferences for completion of P. For example, the resolution of P may provide information crucial to the agent's strategy for ongoing negotiation P'. To the extent that acknowledgment strategy provides new opportunities (on top of those already present in underlying protocol P) for influencing completion time, they provide incentives to deviate from the quiescence detection protocol.

Finally, we consider premature-acknowledgment deviations (#3). In a premature acknowledgment, 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.

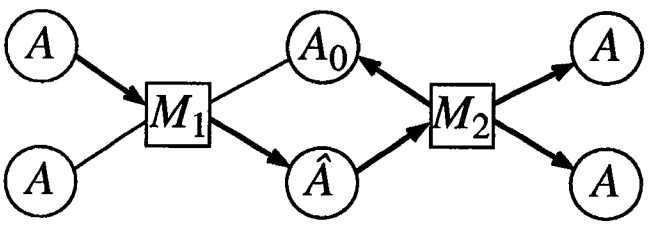

Figure 4: Agent $\hat{A}$ may have an incentive to deviate by prematurely acknowledging mediator M_1.

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.

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.

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).

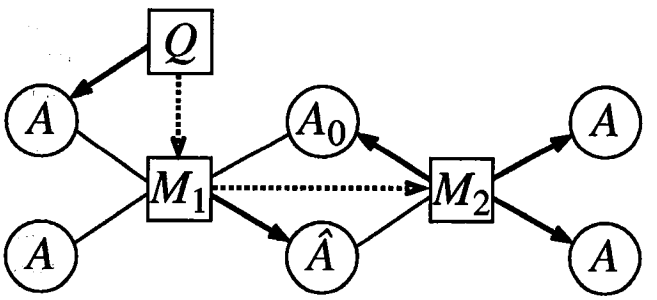

Figure 5: The supply chain network with a mediator-to-mediator activation backbone.

Proposition 3 *In the augmented quiescence detection protocol, no agent can cause incorrect quiescence detection via premature acknowledgment.*

Thus, the only deviations possible in the augmented scheme are #1 and #2—failure to acknowledge, which have the limited effect discussed above.

Moreover, 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 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. 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.

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.

Decommitment

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. Andersson and Sandholm (1998) 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.

Whereas allowing decommitment is in some respects an alternative to explicit quiescence detection, the approaches are often complementary. In particular, even with decommitment, it is still generally necessary to determine when it is safe to execute irreversible actions, such as actually performing a task or engaging in a production activity. Thus, we require a means to detect that the decommitment process has settled down. In this sense, decommitment acts are part of what we consider here the underlying negotiation protocol.

More generally, defining the commitment entailed by various negotiation actions is a central matter for designers of negotiation mechanisms. Whether to regard a particular sort of `OFFER` as binding subject to penalty for revocation, or as tentatively binding subject to completion of a broader negotiation (with rules for how the negotiation proceeds) has implications depending on many other aspects of the negotiation situation. We would expect that both types of policies will be widely useful in multiagent negotiation.

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 presents limited incentive for an agent to deviate by failing to acknowledge messages (the only deviation possibility left), as long as the underlying negotiation protocol is individually rational.

However, this precondition does not hold for all negotiation protocols. 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. Thus, there is no sense in which an agent can deviate by representing that a given message belongs to negotiation P. 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.

Our mediator backbone approach to enforcement could be applied to other protocols relevant to negotiation. For instance, we have examined a supply-chain formation mechanism whereby agents may decommit from contracts only if they do not achieve specified goals in other negotiations (Walsh & Wellman 1999). The mediators could enforce the rules to prevent spurious decommitment in this context.

References

Andersson, M. R., and Sandholm, T. W. 1998. Leveled commitment contracting among myopic individually rational agents. In *Third International Conference on Multi-Agent Systems*, 26–33.

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

Lynch, N. A. 1996. *Distributed Algorithms.* Morgan Kaufmann.

Walsh, W. E., and Wellman, M. P. 1999. Efficiency and equilibrium in task allocation economies with hierarchical dependencies. In *Sixteenth International Joint Conference on Artificial Intelligence.*