Concurrent Multiple-Issue Negotiation for Internet-Based Services

Jiangbo Dang and Michael N. Huhns

University of South Carolina
Columbia, SC 29208 USA
{dangj, huhns}@sc.edu

Abstract. Negotiation is a technique for reaching mutually beneficial agreement among autonomous entities. A concurrent negotiation problem occurs when one entity is negotiating simultaneously with several other entities to reach agreement. In the context of Internet-based services, we present a protocol to support concurrent negotiation over multiple issues. By extending existing negotiation protocols, our described protocol enables both service requestors and service providers to manage several negotiation processes in parallel. The protocol allows the negotiation participants to make durable commitments to reduce the occurrence of a decommitment situation. Since colored Petri nets have greater expressive power than finite state machine and support for concurrency, we use them to represent our negotiation protocol and facilitate our analysis of desirable properties. More importantly, our protocol can be analyzed for deadlock, liveness and other faults by using the existing colored Petri nets tools.

1. Introduction

Negotiation is a process by which autonomous entities communicate and compromise to reach agreement on matters of mutual interest, while maximizing their individual utilities. In e-commerce, Web services, and on-line supply chains, the participants use negotiation about the properties of the services they request and provide to enter into binding agreements and contracts with each other. To meet the requirements of the participants, multiple issues including both functionality issues and quality issues, e.g., response time, cost, price, need to be taken into account.

In an Internet-based service environment, it is very likely there are multiple service requestors and providers negotiating simultaneously. Researchers are interested in concurrent negotiation because (1) it is both time efficient and robust when an entity (a software agent herein) negotiate with multiple other entities to choose the best offer, and (2) it is essential when an agent requests a service involving multiple agents, as in a supply-chain problem.

In a many-to-many negotiation, each agent is involved in possibly many negotiations with different participants at the same time, so the overall negotiation is many-to-many among the agents. However, each individual negotiation involves just two agents, so it is bilateral. We herein refer to the entire negotiation process as many-to-many bilateral negotiation. In this paper, we consider a competitive environment and assume the agents are self-interested and know only their own

Formatted: Font: Bold, Complex Script Font: Bold, Do not check spelling or grammar

Deleted: Concurrent Multiple-Issue Negotiation for Internet-Based Services

negotiation preferences. Based on the two-phase commit protocol [1] from database transaction theory and the extended contract-net protocol [2], we present a negotiation protocol to support many-to-many negotiation where many agents negotiate with many other agents simultaneously [3].

Petri nets are a graphical and mathematical modeling tool applicable to many systems. They are a promising tool for describing and studying information processing systems that are characterized as being concurrent, asynchronous, distributed, and parallel [4]. Since colored Petri nets (CPNs) support for concurrency and have greater expressive power than finite state machines (FSMs), we propose the use of colored Petri nets as a modeling language for our concurrent negotiation protocol.

This paper advances the state of the art in the following ways. First, most existing protocols for concurrent negotiation do not deal with issues such as negotiation consistency and decommitment situation where agents tend to drop their commitments in a contract when they find a better offer later. Second, most CPN-based conversation models do not deal with concurrent negotiation processes in a competitive service-oriented environment. In contrast, this paper (1) introduces a concurrent negotiation protocol that enables both service requestors and providers to engage in several negotiation processes simultaneously; (2) describes a negotiation process that reduces the bias and the number of decommitment situations; (3) formulates and analyzes the protocol using colored Petri net technologies; and (4) explores the use of colored Petri nets in modeling and analyzing communication interaction.

Presented in the context of Internet-based services, the remainder of the paper is organized as follows. Section 2 discusses related work. Section 3 describes the negotiation protocol and Section 4 details the algorithms. Section 5 analyzes the property of the proposed protocol theoretically. Section 6 discusses further issues related to concurrent multiple-issue negotiation and Section 7 concludes

2 Related Work

Negotiation for services involves a sequence of information exchanges among parties to establish a formal agreement among them, whereby one or more parties will provide services to one or more other parties. Therefore, concurrent negotiation is necessary for an Internet-based service domain.

By considering negotiation as a distributed constraint satisfaction problem, [5] presents a framework to support one-to-many negotiation by coordinating a number of concurrent one-to-one negotiations and allowing several possible negotiation strategies for a coordinator. However, issues arising in many-to-many negotiation, such as consistency, coordination, and decommitment risk, are too difficult to be handled by this framework.

Nguyen et al. [6] present a heuristic model for coordinating concurrent negotiation and an integrated commitment model, which enable agents to reason about when to commit or decommit. However, their model is obviously biased in favor of the buyer. To mitigate the problem, they allow the seller to decommit, stray from its commitment, by paying a pre-computed decommitment penalty, and then both buyer

Formatted: Font: Bold, Complex Script Font: Bold, Do not check spelling or grammar

Deleted: Concurrent Multiple-Issue Negotiation for Internet-Based Services

and seller can renege on the previous deal. Incorporating seller decommitment into their model has no advantage, because in one-to-many negotiation there are no other buyers for a seller, so it has no incentive to decommit and will never do so rationally. On the other hand, breaking a commitment is always a hard decision to make, because more issues beyond a decommitment penalty usually need to be considered, such as reputation and user feedback.

In [2], an extended version of the contract-net protocol is presented to support concurrent negotiation processes for a task contractor. It is efficient and failure tolerant compared to the basic contract-net protocol. However, it does not allow counterproposals, which are very important in negotiations involving time constraints, especially when multiple issues are involved.

The value of a conversation-based approach is largely determined by the conversational model it uses [7]. Petri nets are a well known graphical and mathematical modeling tool and have been applied in modeling agent communication interactions [4, 7]. In [8], researchers apply message sequence charts to specify the interaction between organizations by using Petri nets to model the workflow inside the organization.

3 Negotiation Protocol

3.1 A Motivating Example

To illustrate our protocol, we present a motivating *GetStockQuote* scenario, where a service requestor a_1 wants to obtain a stock quote from a service provider. Figure 1 shows that a_1 locates two service providers, b_1 and b_2, that meet its functionality requirements, so a_1 starts two negotiation processes, one for each provider, to find the one that provides better service with lower cost. If b_1 is already negotiating with another potential service requestor a_2, b_1 will negotiate with a_1 and a_2 at the same time to determine which one will make a better offer. Most existing protocols cannot handle this situation properly. In some protocols, for example, if b_1 is one of the providers negotiating with a_1 concurrently, b_1 must wait until all a_1's negotiation threads end. Even if b_1 has reached an agreement with a_1 earlier, a_1 can still reject it; it is not finalized until a_1 has finished all its negotiation threads. These protocols bind b_1 to a one-sided commitment and cause b_1 to lose time and reduce its chances of reaching a contract with other agents. The protocols are biased in favor of a_1, but still cause a_1 the trouble of a likely decommitment to the previously agreed proposals, and lead to the loss of utility (decommitment penalty) and reputation that would harm an agent interested in long-term cooperation and gains.

By considering the two-phase commit protocol and the extended contract-net protocol, we introduce two phases of accept and reject into the alternating offers protocol [9] to support concurrent multi-issue negotiation. During a negotiation session, an agent can use a number of messages when communicating with its opponent. The negotiation performatives are defined in Table 1 and illustrated by our *GetStockQuote* scenario.

Messages	Descriptions
propose	A requestor initiates the negotiation by proposing

Formatted: Font: Bold, Complex Script Font: Bold, Do not check spelling or grammar

Deleted: Concurrent Multiple-Issue Negotiation for Internet-Based Services

	an offer for a service.
counter-propose	An agent offers a new proposal in response to the previous proposal.
formally-propose	An agent formalizes its pre-accepted proposal.
pre-accept	An agent temporarily accepts a proposal.
pre-reject	An agent temporarily rejects a proposal.
accept	An agent accepts a proposal.
reject	An agent rejects a proposal.

Table 1. Negotiation Performatives

In multi-issue negotiation, different agents have different preferences over the negotiation issues. Their preferences are usually represented in the form of their utility functions with issues as variables. For the alternating offer protocol, an agent makes an offer that gives it the highest utility at the beginning of the negotiation, and then incrementally concedes by offering its opponent a proposal that gives it lower utility as the negotiation progresses.

Let a and b represent the negotiating agents and I a set of n negotiation issues, where $I=\{I_1,I_2,\ldots I_n\}$. Given $O_{b->a,k}$ representing an offer from b to a at negotiation round k, we define a's utility as $U_a(O_{b->a,k})$. Agent b's utility is defined similarly.

Definition 1: In a negotiation where agent a negotiates with a set of agents $B=\{b_1, b_2,\ldots,b_n\}$ concurrently, agent b_i's offer $O_{bi->a,k}$ is better than agent b_j's offer $O_{bj->a,k}$ iff $U_a(O_{bi->a,k})>U_a(O_{bj->a,k})$.

Definition 2: In a negotiation where agent a negotiates with a set of agents $B=\{b_1, b_2, \ldots, b_n\}$ concurrently, agent b_i's offer is *acceptable* to agent a at round k if (1) $U_a(O_{bi->a,k}) >= U_a(O_{a->bi,k+1})$ and (2) $U_a(O_{bi->a,k}) =$ **argmax** $U_a(O_{bj->a,k})$ for $b_j \in B$,where **argmax** function return the max value from a set.

As shown in Figure 1, a requestor agent a_1 locates two provider agents, b_1 and b_2, and then initiates two negotiation threads simultaneously by sending them its proposal. After evaluating the received proposal, b_2 sends its counter-proposal to a_1. Although b_1 is negotiating with another requestor agent a_2 when it receives a_1's proposal, b_1 sends its counterproposal to a_1, since b_1 has not yet reached any agreement with a_2. After evaluating it, a_1 finds that b_2's counterproposal is acceptable and pre-accepts it. a_1 will pre-reject other counterproposals at the same time. b_1 receives the pre-reject from a_1 and the pre-accept message from a_2, so b_1 sends the formal-proposal to a_2 and pre-rejects all other requestors. While pre-accepted b_2 sends a_1 its formal-proposal, other pre-rejected agents send their counter-proposals to a_1. a_1 accepts b_2 and rejects all other providers, if b_2's formal-proposal is still acceptable. Similarly, a_2 sends the accept message to b_1 and the reject message to other providers.

Formatted: Font: Bold, Complex Script Font: Bold, Do not check spelling or grammar

Deleted: Concurrent Multiple-Issue Negotiation for Internet-Based Services

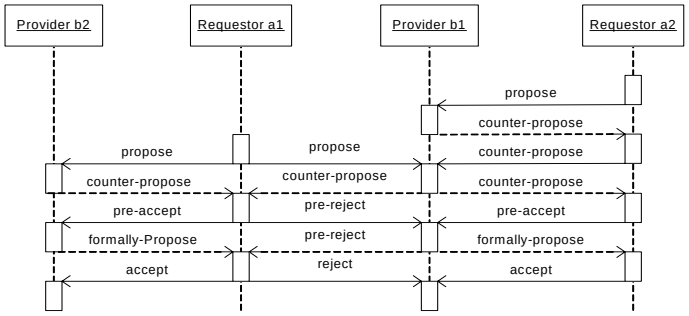

Fig. 1. A Concurrent Negotiation Sequence Diagram

3.2 Colored Petri Nets

In an Internet-based environment, service requestor and provider agents need to comply with an interaction protocol in order to negotiate successfully. It is important for the protocol itself to be correct, unambiguous, complete, and verifiable. The ability to express correct protocols depends on the specification language or tool used to model the protocol. FSMs can express a variety of protocols in a conceptually simple and intuitive way. However, they are not adequately expressive to model more complex interactions, especially those with some degree of concurrency. Petri nets (PNs) were originally defined in answer to the limited modeling power of finite-state models. They provide the benefits of FSMs while allowing greater expressivity and concurrency. This is our motivation for adopting PNs as an alternative mechanism.

A Petri net is a directed, bipartite graph whose nodes represent the possible states and actions of a process and whose arcs represent possible transitions among states and actions. Concurrency is provided by the presence of multiple tokens that move through the graph and indicate its current status. In a colored Petri net, each token is extended with a value referred to as *color*, which is a schema or type specification.

CPNs have great value for modeling a concurrent protocol since they provide: (1) a relatively simple and precise formal model, (2) an intuitive graphical representation, (3) full expressiveness with explicitly represented states, (4) support for concurrency, and (5) a firm mathematical foundation for investigating and verifying properties [8].

3.3 Negotiation Protocol

Figure 2 describes the concurrent negotiation protocol for services. For simplicity, some constraints such as time-out and exception handling have been omitted from the figures. The service requestor (state 1) initiates a negotiation process from place 1 by sending an initial proposal to the service provider. The provider evaluates it (place 2) and if this proposal is acceptable, the provider pre-accepts it (place 4); otherwise, it counter-proposes (place 3). Two entities send counter-proposals back and forth before they find an acceptable offer (places 2 and 3). A provider may pre-reject a proposal if it has pre-accepted another requestor or has been pre-accepted by another requestor (place 5). The pre-accepted requestor sends its formal proposal to the provider (place 6). If this formal proposal is acceptable, it is accepted by the provider (place 12).

Formatted: Font: Bold, Complex Script Font: Bold, Do not check spelling or grammar

Deleted: Concurrent Multiple-Issue Negotiation for Internet-Based Services

Otherwise, this proposal is pre-rejected (place 5) and the requestor can send an improved counterproposal (place 7), which could be pre-accepted by the provider (place 4) or rejected finally (place 13).

Two-phase commitment of *(pre-accept* and *accept*) and the corresponding two-phase rejection (*pre-reject* and *reject*) are necessary to deal with concurrent encounters. With this protocol, the concurrent negotiation process has two stages. In stage one, service entities exchange counter-proposals after service requestors initiate the negotiations. Once an entity receives an acceptable proposal, the entity announces that the negotiation stage two begins by sending *pre-accept* to the entity who sent the acceptable proposal and sending *pre-reject* to the rest of the negotiating opponents. In stage two, the negotiation enters a process similar to a last-round first-price auction. The pre-accepted entity sends back its formal proposal while other pre-rejected entities send their counterproposals for their final tries. If the former proposal is still acceptable, it will be accepted formally and other offers will be rejected to end the negotiation. Otherwise, it will be prejected (detail is discussed in Section 5). In Figure 2, state 1, 2, 3 belong to negotiation stage one, and the rest states belong to negotiation stage two.

In this section we use colored Petri nets as the modeling tool and present our negotiation protocol with the negotiation performatives in Table 1. In this model, a colored token has an attached data value, which may be of arbitrary complex type. Color sets determine the possible values of tokens. Usually a language called CPN ML is used to make CPN declarations. The CPN color set is defined in Table 2.

```
color PERFORMATIVE = with propose | counter-propose | formally-propose | pre-reject |
                                    pre-accept | reject | accept;
color AGENTNAME = string;
color AGENTTYPE = with requestor | provider
color STATUS = with f1 | f2- | f2+
color ID = product AGENTTYPE * AGENTNAME * STATUS;
color VALUE = with float | int | string ;
color CONTENT = list VALUE;
color MESSAGE = product PERFORMATIVE * ID * ID * CONTENT * ROUND;
color ROUND = int

var m : Message;
var p, q : PERFORMATIVE;
var c, d : CONTENT;
var s, r : ID;
var k : ROUND;

fun acceptable(c: CONTENT);
```

Table 2. Color Set Declaration

The CPN-ML descriptions in Table 2 give a formal way to express the negation process in the system. The messages used are based on our proposed negotiation performatives. The type AGENTNAME is defined as a string, and AGENTTYPE is defined as an enumeration type containing two possible values: *requestor* and *provider*. STATUS indicates different negotiation status, i.e., f1 denotes the first negotiation stage in which none of the agents is pre-rejected or pre-accepted yet, f2-

Formatted: Font: Bold, Complex Script Font: Bold, Do not check spelling or grammar

Deleted: Concurrent Multiple-Issue Negotiation for Internet-Based Services

denotes the status in which one agent has pre-rejected other agents, while f2+ denotes the status in which one agent has pre-accepted another agent. The type ID is defined as the product of the types AGENTNAME and AGENTTYPE, and STATUS. The CONTENT type represents the possible negotiation offer in a multi-issue negotiation and is defined as a list of values of integer, real, or string types. We also define several variables with the declared types and a Boolean function *acceptable*( ), which takes a CONTENT variable as the argument and produces a true/false value as explained in Definition 2.

The states of a CPN are represented by *places*, drawn as ellipses or circles. Each place has an associated type (color set) determining the kind of data that the place may contain. The colors of a CPN can be arbitrarily complex, e.g., a message with fields of integer, real, string, and list as we defined in Table 2. The type of a place is written to the lower left or right of the place. The type definitions for places are omitted since all places have the same type: MESSAGE.

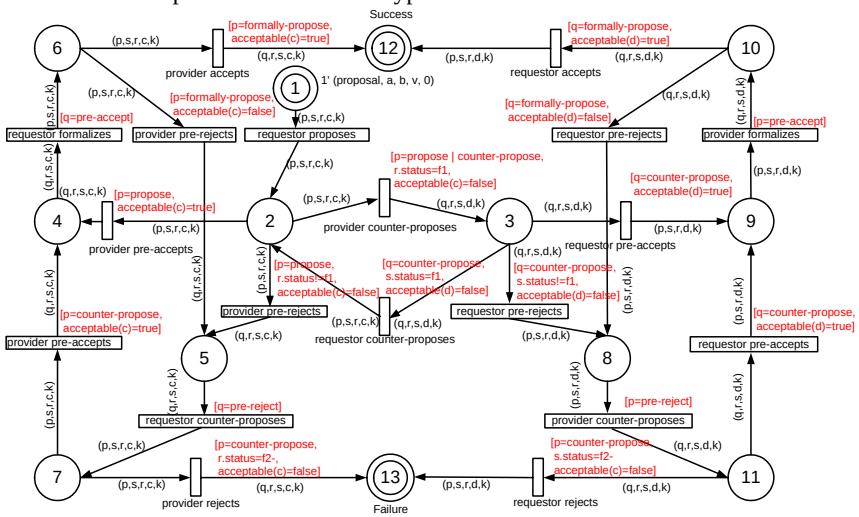

Fig. 2. Colored Petri Net for Concurrent Negotiation

A state of a CPN is called a *marking*. It consists of a number of tokens distributed on the places in the CPN. The tokens that are present on a particular place are called the marking of that place. Each token carries a value (color), which belongs to the type of the place on which the token resides. As an example, a possible marking of the place 1 is 1'(propose, a, b, v, 0). This marking contains one token with value MESSAGE=(propose, a, b, v, 0), where the performative is *propose*, sender agent is *a*, receiver agent is *b*, and offer is a list named v at round *k=0*. By convention, the symbol prime (`) denotes the number of appearances of the token. An initial marking is used to describe the initial state of the system and is written on the upper left or right of the place by convention. In our CPN, the place 1 has an initial marking consisting of a single token with the value [propose, a, b, v, 0]. Initially, the remaining places contain no tokens.

Formatted: Font: Bold, Complex Script Font: Bold, Do not check spelling or grammar

Deleted: Concurrent Multiple-Issue Negotiation for Internet-Based Services

Transitions represent the actions of a CPN and are drawn as rectangles. In order for a transition to be enabled in a marking, it must be possible to bind data values to the variables appearing on the surrounding arc expressions and in the guard of the transition such that: (1) each of the arc expressions evaluate to tokens that are present on the corresponding input place; and (2) the guard (if any) is satisfied [10]. The transition (requestor propose) has five variables: p of type PERFORMATIVE, s of type ID, r of type ID, c of type CONTENT, and k of type ROUND as shown in Table 2. We assign data values to the variables of the transition (requestor propose) by creating the following binding: $p \leftarrow proposal$, $s \leftarrow a$, $r \leftarrow b$, $c \leftarrow v$, $k \leftarrow 0$. In addition to the arc expressions, it is possible to attach a list of Boolean expressions (with variables) to each transition. The list of Boolean expressions is called a *guard*. It specifies that we only accept bindings for which all Boolean expressions evaluate to true. As an example, the transition (provider counter-propose) uses a guard with three Boolean expressions: [p=proposal|counter-proposal, r.STATUS=f1, acceptable(c)=false]. The result is true only if all expressions are true. The meanings of the expressions are strightforward, except the operator '.' is used to extract STATUS information from a variable of type ID in the arc expression r.STATUS.

An *occurrence* of the *transition* (provider counter-propose) removes a token with value (propose, a, b, v, 0) from the place 2 and adds a token to the output place 3. The values of the tokens removed from an input place are determined by evaluating the arc expression on the corresponding input arc. Similarly, the values of tokens added to an output place are determined by evaluating the arc expression on the corresponding output arc. After evaluating the guard, the transition (provider counter-propose) updates the token by changing the performative to counter-propose, switching the sender and receiver, updating its offer, and incrementing the round *k*.

An execution of a CPN is described by an *occurrence sequence*. It specifies the markings that are reached and the steps that occur. In the initial marking, the transition (requestor propose) is enabled in a binding. Hence, the occurrence sequence can be continued. This leads to a new marking in which one of the transitions (provider counter-propose, provider pre-accept, and provider pre-reject) is enabled by evaluating the binding and guard values. An infinite occurrence sequence is an occurrence sequence consisting of an infinite number of markings and steps. An infinite occurrence sequence corresponds to a non-terminating execution of the system. We analyze the termination property of our protocol below.

Note that Figure 2 models concurrent pairwise negotiation between a service requestor and a provider. With our proposed protocol, both service requestors and service providers can manage several negotiation processes in parallel. In addition, the negotiation protocol is encoded inside arc expressions and guards, but the agent's negotiation algorithm is not specified in the above figure, and is described in the next section.

4 Negotiation Algorithm

With the defined color set and convention, we illustrate the proposed negotiation mechanism for service requestors in Figure 3. A service requestor initiates a negotiation process by sending a proposal to the corresponding providers.

Formatted: Font: Bold, Complex Script Font: Bold, Do not check spelling or grammar

Deleted: Concurrent Multiple-Issue Negotiation for Internet-Based Services

EvaluatingCounterProposal deals with all counterproposals in phase one. It evaluates the counterproposals from the providers and sends pre-accept, pre-reject, or counter-propose regarding the evaluation result. *EvalutingCounterProposal2* evaluates all counterproposals in phase two. It sends pre-accept to the sender if its proposal is acceptable; otherwise it sends reject. *EvaluatingFormalProposal* evaluates the formal-proposal from the pre-accepted entity. It sends accept if the formal proposal is acceptable and pre-reject otherwise. The CPN diagram for a service provider is similar to Figure 3 except it starts with a *ReceiveProposal* transition that leads to an *Evaluating* place, which is equivalent to *EvaluatingCounterproposal* in Figure 3.

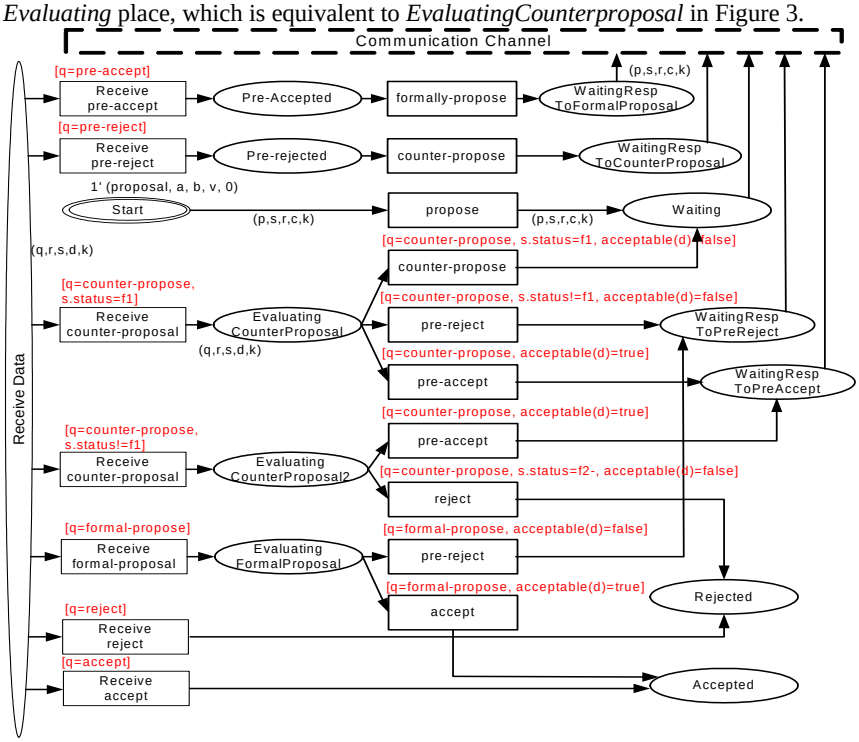

Fig. 3. Colored Petri Net for a Service Requestor

Since we assume that the agents are self-interested, it is possible for an agent to propose a very good offer in phase one in order to scare off its competition and then send a lower formal-proposal later. Although it likely is bested by other agents' counterproposals, we enforce a negotiation strategy that avoids this situation. Any formal-proposals worse than their pre-accepted proposals will be peremptorily rejected. The best offer from the received counterproposals will be pre-accepted as a replacement in this case.

Formatted: Font: Bold, Complex Script Font: Bold, Do not check spelling or grammar

Deleted: Concurrent Multiple-Issue Negotiation for Internet-Based Services

5 Theoretical Analysis

An important peroperty of a negotiation protocol is whether it can guarantee that any negotiation process following the protocol will eventually terminate.

Termination Property: Given a set of service requestors *A* and a set of service providers *B*, a negotiation process engaged by the entities from *A* and *B* using our concurrent protocol ends after a finite number of steps.

Proof: From Figure 2, we can see that three loops can occur during the negotiation process: (1) a loop on states 2 and 3; (2) a loop on states 5, 7, 4, 6, and (3) a loop on state 8, 11, 9, and 10. To prove that the protocol will end in a finite number of steps, we must prove that none of the three loops could have an infinite number of steps.

(1) In loop 2-3, service entities exchange counterproposals by the alternating offer protocol in which an entity has to concede to offer deals that are more likely to be accepted by its opponents, if it prefers an agreement to the conflict deal. These principles apply to all concurrent negotiation threads. Many existing mechanisms can guarantee agents will keep making progress during the negotiation. With a pre-defined minimum hop, one entity will eventually pre-accept the counter-proposal from its opponents and exit from the loop (1) or time out. Also, an agent can be pre-rejected out of the loop (1) when its opponent pre-accepts one of its peers from another concurrent negotiation thread.

(2) In loop 5-7-4-6, the pre-rejected requestor sends its new counterproposal to the provider and is pre-accepted; however, its formal proposal is then pre-rejected and the requestor has to send a new counterproposal. Let us assume a service provider *b* negotiates concurrently with a set of *n* service requestors: a pre-accepted requestor a_0 and a set $A' = \{A - a_0\}$ of *n-1* pre-rejected requestors. After the provider receives the formal proposal from a_0 and the counterproposals from A', let a_i be the one with the best offer from A'. By comparing the offers from a_0 and a_i, we have:

(a) If $U_b(O_{a_i \to b}) \le U_b(O_{a_0 \to b})$, requestor a_0 will be accepted and reach state 12; all requestors from A' will end with rejections and reach state 13. The negotiation ends

(b) If $U_b(O_{a_i \to b}) > U_b(O_{a_0 \to b})$, requestor a_0 will be pre-rejected and reach state 5, requestor a_i will be pre-accepted and enter state 4, and all other agents are rejected and out of the loop. There are only a_i and a_0 left. The negotiation ends if one of them can overbid the other in two consecutive rounds. An infinite loop occurs when a_i and a_0 keep overbidding each other alternately by a tiny amount. It can be prevented by defining a minimum increment ε or enforcing a time constraints on the protocol. Since both sides would be better off if they can reach a contract earlier, the provider should consider the time factor for proposals received in phase two. For example, before comparing two proposals, a time discount function $\delta(k) < 1$ (e.g., a normalized function whose value decreases exponentially with the time) can be applied to the counter-proposal that needs to evolve two more states to be a formal-proposal. Therefore, the counter-proposal needs to overbid the formal-proposal more to overrule it along the time and negotiation will end in a finite number of steps.

Formatted: Font: Bold, Complex Script Font: Bold, Do not check spelling or grammar

Deleted: Concurrent Multiple-Issue Negotiation for Internet-Based Services

As described in [7], CPNs can be checked for a variety of properties. Given a CPN, we might be interested in reachability: does the initial marking result in the correct negotiation result? We can perform a liveness test: does the negotiation process enter a "dead" state in which no further activity can occur? Colored Petri nets are accompanied by a number of techniques and tools for formal analysis and verification of such properties.

6 Concurrent Negotiation Issues

With this protocol, the concurrent negotiation process can be performed at two levels: The upper level (a coordinator) is responsible for coordinating all the threads and solving conflicts among them. The lower level (negotiation threads) deals directly with the various opponents and is responsible for deciding what counterproposals to send and what proposal to pre-accept. In each round, the threads report their status to the coordinator, and the coordinator updates the status of the other threads and uses the progress in one thread to alter the behavior of the agent in the other threads.

A commitment is a symmetric relationship that binds the participants in a negotiation. One-sided commitemen problems occurs when one party, e.g., service requestor, negotiate with several service provider candidates in parallel, after reaching an agreement with one provider, requestor continues negotiating with other providers and renege to the previous agreement if it finds a better offer later. Figure 2 shows that our protocol is neutral to both service requestors and service providers. Therefore, our protocol can eliminate the decommitment situations that arise in one-sided commitment.

In negotiation phase two, a service agent hosts a procedure that is similar to a last-round first-price auction. Since it is the final try for those pre-rejected agents to stay in the negotiation, it makes truth-telling about the reserve offer the dominant strategy for agents whose counterproposals are close to their reserve offers. At this time, they do not want to lose by providing an offer worse than the reserve offer and they do not want to win the negotiation with negative gains by providing an offer better than the reserve offer. Therefore, this protocol makes the negotiation more efficient.

7 .Conclusion

This paper investigates concurrent negotiation in a competitive environment and formulates a negotiation protocol with CPN technologies. There are several possible directions for future work. First, we will further investigate the effect of this protocol on an agent's negotiation strategy and develop a precise commitment model. Second, we can extend this protocol to support negotiation for a composed service with different service agents under constraints such as QoS and dependency issues among agents.

Formatted: Font: Bold, Complex Script Font: Bold, Do not check spelling or grammar

Deleted: Concurrent Multiple-Issue Negotiation for Internet-Based Services

References

[1] J. Gray, "Notes on Data Base Operating Systems," in *Operating Systems, An Advanced Course*, vol. 60. London, UK: Springer-Verlag, 1978, pp. 393--481.

[2] S. Aknine, S. Pinson, and M. F. Shakun, "An Extended Multi-agent Negotiation Protocol," presented at International Conference on Autonomous Agents and Multi-agent Systems, New York, USA, 2004.

[3] J. Dang and M. N. Huhns, "An extended protocol for multiple-issue concurrent negotiation," presented at AAAI 2005, Pittsburgh, PA, 2005.

[4] T. Murata, "Petri Nets: Properties, Analysis and Applications," *Proceedings of the IEEE*, pp. 541--580, 1989.

[5] I. Rahwan, R. Kowalczyk, and H. H. Pham, "Intelligent agents for automated one-to-many e-commerce negotiation," presented at the Twenty-Fifth Australasian Conference on Computer Science, Melbourne, Victoria, Australia, 2002.

[6] T. D. Nguyen and N. R. Jennings, "Reasoning about commitments in multiple concurrent negotiations," presented at the 6th International Conference on E-Commerce, Delft, The Netherlands, 2004.

[7] R. S. Cost, Y. Chen, T. Finin, Y. Labrou, and Y. Peng, "Using Colored Petri Nets for Conversation Modeling," in *Issues in Agent Communication*, F. D. a. M. Greaves, Ed.: Springer-Verlag: Heidelberg, Germany, 2000, pp. 178--192.

[8] W. M. P. v. d. Aalst, "Interorganizational workflows: An approach based on message sequence charts and petri nets," *Systems Analysis, Modeling, Simulation*, vol. 34, pp. 335--367, 1999.

[9] M. J. Osborne and A. Rubinstein, *A Course in Game Theory*: The MIT Press, 1994.

[10] K. Jensen, *Coloured Petri Nets*, vol. 1, Second ed: Springer, 1996.