FIPA-Based Semi-centralized Protocol for Negotiation

Amjad Hudaib, Mais Haj Qasem(✉), and Nadim Obeid

King Abdullah II School for Information Technology,
The University of Jordan, Amman, Jordan
{ahudaib,obein}@ju.edu.jo, mais_hajqasem@hotmail.com

Abstract. An important application of multi-agent systems is task negotiation. The existing protocols for controlling negotiation in multi-agent systems are either centralized or decentralized. The centralized protocols suffer from dependency on the central agent. If any problem occurs at the central agent, such as shutting down or becoming slow, the whole system will be blocked. By contrast, decentralized protocol has insufficient information about the agents being used, which might result in a high possibility of agent failure due to internal or external factors, such as losing connections. In this paper, a semi-centralized protocol is proposed to overcome the limitation of the existing protocols and enable robust and adaptable solution in a highly dynamic environment. The outcome of Java Agent DEvelopment Framework (JADE's) implementation proves the capabilities of the proposed protocol. In comparison with the Contract Net Protocol, the proposed protocol shows significant improvement in time and communication overhead under various conditions.

Keywords: FIPA standards · Multi-agent system · Negotiation protocol

1 Introduction

Multi-agent systems are a computerized environment that facilitates intelligent agent communication and interactions to achieve certain tasks by incorporating the involved agents. Multi-agent systems are created mostly by the so-called agent-based model, which is an intelligent software characterized with states and behaviors. The interaction in such multi-agent systems is managed by protocol(s) that control the communication and interaction among the involved agents [1]. An important application of multi-agent systems is task negotiation. In the task negotiation, the client needs to find some servers to implement a specific task within reasonable time and performance. Subsequently, the client (initiator) calls for proposals for the intended task and the servers (participants) respond with their proposals. The initiator then decides whether to accept or reject any of these proposals [2].

The Foundation for Intelligent Physical Agents (FIPA) aims at developing standard protocols for agent communication by determining the external agent behavior side by side with the inside agent structure without focusing on the internal behavior of the agent. These specifications, together with the task implemented by the underlying multi-agent system can determine the specification of the internal agent behavior.

R. Silhavy et al. (eds.), *Cybernetics Approaches in Intelligent Systems*,
Advances in Intelligent Systems and Computing 661,
DOI 10.1007/978-3-319-67618-0_13

FIPA facilitates the interaction between agents in the standardized protocols by using the so-called Agent Communication Language (ACL) messages [3]. A FIPA ACL message consists of a set of fields that standardized the communication between agents. These fields are mandatory in some protocols and optional in other protocols, which facilitate efficient and standard communication between agents. In addition, the fields provide space for user-defined parameters that could be semantically identified by a related ontology that is shared among the communicated agents. The full set of FIPA ACL message parameters is given in Appendix A regardless of their specific protocol utilization [4].

This paper is organized as follows. Section 2 reviews the works of negotiation protocol. Section 3 presents our proposed semi-structured approach. Section 4 presents experimental results. The conclusion is given in Sect. 5.

2 Related Work

Various multi-agent protocols were proposed in the literature. Dang and Huhns [5] proposed a decentralized negotiation protocol to solve the problem of concurrent negotiation. The proposed protocol is composed of two phases, the proposal exchange phase and the proposal formalization phase. In the proposal exchange phase, all the agents exchange proposals until pre-accepted/pre-rejected message is issued by the initiator. In the proposal formalization phase, the agent with pre-accepted proposal sends its formal proposal and reaches an agreement with the initiator. This protocol facilitates concurrent negotiation that speeds up the negotiation process to some extent. However, in some cases, this pre-accepted/pre-rejected phase might slow the process if the involved agents could not reach an agreement. Moreover, this process requires exchange of enormous number of messages. Finally, bad reputation agent might decrease the performance of the system significantly. This protocol has been used by Williams [14] for negotiation in complex environment and by Haim et al. [15] for human–computer negotiation. Williams et al. [6] proposed a Concurrent Negotiation Protocol (CNP), which sets the policies for many-to-many negotiation process. The proposed protocol is centralized-based that depends on a coordinator to direct the negotiation process and adjust the parameters. These policies were then used to implement various negotiation protocols as have been proposed by Panagidi et al. [17], Aydogan et al. [18], and Niu et al. [19]. A review over the existing negotiation protocols is given in [20].

The famous CNP proposed by Smith [7] is based on a decentralized approach. Aknine et al. [8] extends CNP to solve the problem of the time-out mechanism used in the CNP. In the original CNP, the initiator waits for a specific time for proposals and responds to its call for proposal. According to Smith [7], the problem arises when the contractor loses the chance to anticipate this call with short waiting time. In addition, the contractor will lose other contracts in waiting for response of its proposal in long waiting time. The proposed protocol allows all the involved agents to exchange messages about the calls and to make decisions, thereby eliminating the possibility of late response between the initiator and the participant. Weyns et al. [9] modified the CNP by allowing dynamic task to change in the dynamic environment. The proposed

protocol is identical to the original protocol, but it allows the initiator and the participant to deny its proposal/acceptance of the proposal if there are better opportunities, as long as the participant has not start executing the job. Cao [10] proposed an agent communication model for automating the negotiation process. The goal of this model is to formalize the way in which agents in multi-agent system can establish an understandable interaction among each other. This common understanding is facilitated by using an ontology that could semantically identify the meaning of objects being negotiated. The utilization of ontology was also addressed in FIPA protocols [11]. The FIPA protocols are discussed in the following subsections.

2.1 Contract Net Protocol

Contract Net Protocol [7] is one of the FIPA formal protocols that were developed for task negotiation and allocation in multi-agent systems. In Contract Net Protocol, two nodes are involved, the initiator, which is looking for an agent to implement a specific task and the participant(s), which is the expected server node. At any time, the initiator can be the participant and the participant can be the initiator, depending on the situation. The communicated specification of the Contract Net Protocol is illustrated in Fig. 1 and is described as follows:

- **Call for Proposals:** The initiator starts the interaction with the available participants (e.g., n-participants) by sending call for proposals, which includes the specification for the task.

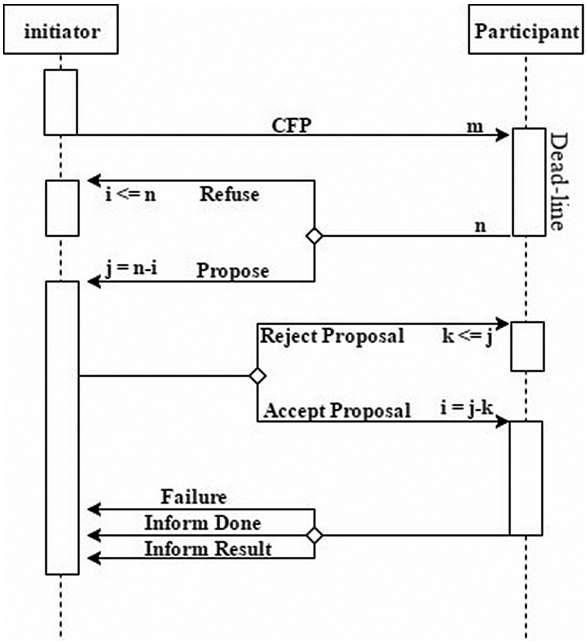

Fig. 1. Contract net FIPA protocol [11]

- **Participants' Proposals:** Active participants (e.g., m-agents, subsets of n, m <= n), reply with their proposals or with their refusal to propose within a deadline that is determined in the call for proposals or determined by the working environment.
- **Response to Proposals:** The initiator responds to the received proposals by sending an acceptance message to the proposal that has been selected and responds to others by sending rejection messages, which informs the end of the interaction: The selected participant implements what has been agreed in the proposal and informs the initiator about the results preceded by informing the end of the interaction, thereby preparing the initiator to receive the results. In some cases, if the participant fails to implement what has been agreed in the proposal, then this participant sends a failure notification to the initiator [11].

The advantages of this protocol are allowing the initiator to select the most suitable proposal among several submitted proposals and allowing all participants to compete among each other. Meanwhile, the disadvantages are: the initiator might select a participant with poor performance or repudiation due to the lack of history exchange among the initiator and the participants. Moreover, the outside factors of some participant might make it worse than others even if that participant has the best proposal, which leads to wrongly selecting that participant by the initiator. Subsequently, recording, managing, and exchanging the history of the participant are required in order to avoid wrong selection. The history should not be limited to the interaction between the initiator and the participant and shall be extended to all transactions in the system. Finally, this extra information should not increase the communication overhead in the multi-agent system.

2.2 Recruiting Interaction Protocol

Recruiting Interaction Protocol [12] is another formal protocol of the FIPA that is developed for recruiting-based task allocation in multi-agent systems. In Recruiting Interaction Protocol, three nodes are involved, namely, the initiator, which is looking to implement a specific task, the recruiter, which is a middle agent that interacts with other agents and knows their capabilities and the sub-protocols agent(s) that perform the actual job. At any time, the initiator can be the sub-protocol and the sub-protocol can be the initiator depending on the situation. The communicated specification of the Recruiting Interaction Protocol is illustrated in Fig. 2 and is described as follows:

- **Proxy:** The initiator starts the interaction with the recruiter by sending a proxy message, which includes the specification for the intended task, the specification of the task, and the number of sub-protocols to be communicated.
- **Recruiter Response:** The recruiter replies by accepting or refusing the agreement, or it might happen that the recruiter fails to find a match and responds with a failure message.
- **Recruiter Proxy:** The recruiter communicates with the available agents to implement the initiator's task.
- **Inform the end of the interaction:** The recruiter informs the initiator about the results of the communication with the sub-protocols (done/failure).
- **Inform the end of the interaction:** The selected sub-protocol implements what has been agreed and informs the initiator about the results [13].

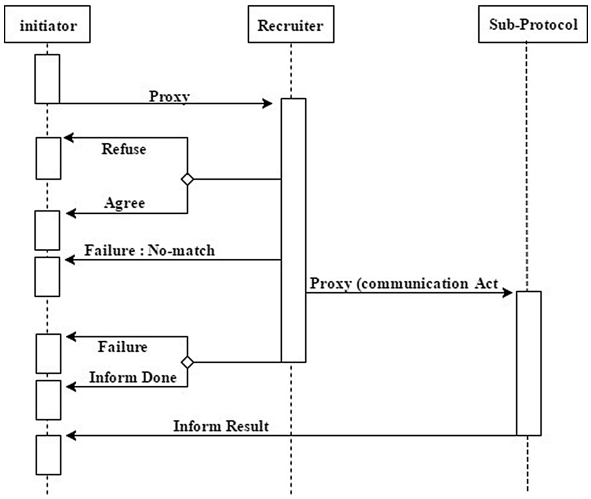

Fig. 2. Recruiting interaction FIPA protocol [13]

The advantages of this protocol are being robust and adaptable in dynamic situations. The ability to reduce the communication cost and the probability of failure having a reference, which is the recruiter, knows the capabilities of the servers. In addition, as the sub-protocol forwards the results directly to the initiator, that reduces the load to some extent on the recruiter. Meanwhile, the disadvantage of this protocol is the load created at the recruiter in a large system is responsible for negotiation and communication when it is used in a negotiation task. Moreover, if the manager dies, all systems will be shut down. Moreover, this protocol does not perform well in some systems that depend more on competency. Subsequently, for a negotiation task, integrating Contract Net and Recruiting Interaction Protocols is required to reduce the load on the central agent while maintaining low failure probability and low communication to implement a good negotiation process.

3 Proposed Work: Semi-centralized Negotiation Protocol

The proposed protocol is of semi-centralized nature, which involves using a dedicated central agent with specific roles. The proposed protocol extends the well-known Contract Net Protocol to be used for efficient and trusted negotiation task in multi-agent systems with low communication overhead. The enhancement is embodied in introducing a new agent that plays the role of the system manager. The manager tasks are as follows:

- Keep track of the status of all agents in the system. Each agent informs the manager when he activates and deactivates. This process minimizes the cost of informing all agents.

- Keep track of all the tasks history in the system. Thus, the manager has a record about the achievements and the performance of the agents.
- Run an application-specific function to select the best agent(s) to perform a given task.

3.1 The External Agent Behavior

The proposed protocol is illustrated in Fig. 3 and the associated communications are discussed as follows:

- **Call for Participant:** The initiator initiates a call for participant to the manager who knows about the potential participant based on the specification included in the proposal.
- **Manager List:** The manager determines the potential participant(s) and replies to the initiator with a list.
- **Call for Proposal:** The initiator initiates a call for proposal to the potential participants.
- **Participants Proposals:** Participants (m′ agents, subsets of n′, m′ <= n′), reply with their proposals or with their refusal to propose.
- **Initiator Response to Proposal:** The initiator responds to the received proposals by sending an acceptance message or a rejection message.

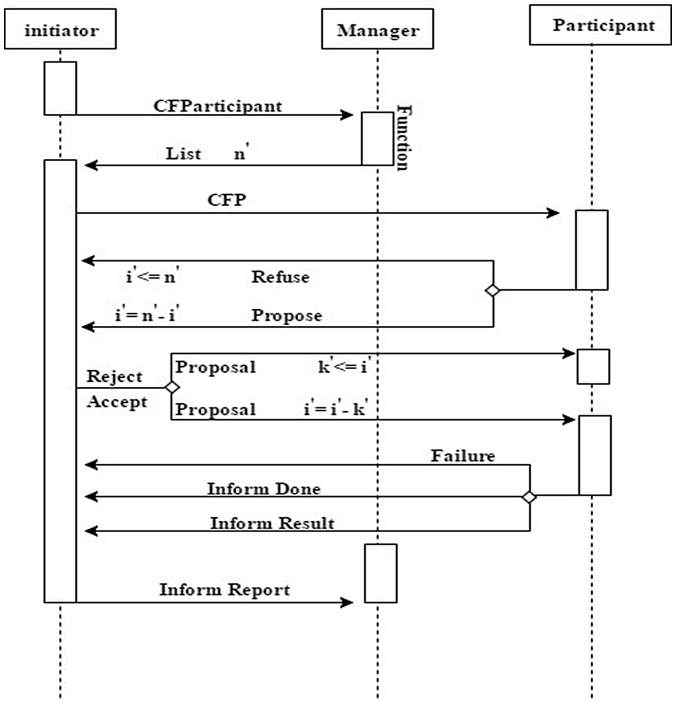

Fig. 3. Proposed protocol

- **Inform the end of the interaction:** The selected participant implements what has been agreed in the proposal and informs the initiator about the results preceded by informing the end of the interaction, thereby preparing the initiator to receive the results. In some cases, if the participant fails to implement what has been agreed in the proposal, then this participant sends a failure notification to the initiator who has to start a new call for proposals. Moreover, the initiator sends the result to the manager to save the history about the performance of the participant agent.

As can be noted, the advantages of the enhanced version over the original protocol are as follows:

- Enhances the performance of the system as a whole by distributing the tasks according to some application-specific function that considers each agent individually and the system as a whole.
- Reduces the possibility of failing while implementing the assigned task, as the manager will select highly successful agents according to the saved history.
- Although the number of exchange messages seems to increase, the benefits of having the manager will reduce the number of calling for proposals as only selected agents will be communicated. Thus, the number of messages will be reduced accordingly in considerably large systems.

3.2 The Proposed Internal Agent Structures

The internal structure of an agent is composed of behavior and states, which transfer the agent from one behavior to another. The states and behavior of the agents in the proposed protocol are discussed in the following. The initiator agent has seven behaviors, and these are:

- **Call for Participants**, in which the initiator sends a call to the manager.
- **Waiting for Participants List**, in which the initiator waits for a response from the manager.
- **Call for Proposals**, in which the initiator sends a call for proposal to the participants.
- **Waiting for Proposals**, in which the initiator waits for a response from the participant.
- **Response to Proposals**, in which the initiator sends acceptance and rejection messages to the participants.
- **Waiting for Results**, in which the initiator waits for the result from a participant.
- **Inform Results**, in which the agent sends a report about the performance of the participant (done/failure/time/cost/etc.) to the manager.

Nine state transitions of the agent are found, which transfer the initiator from one behavior to another as listed in Table 1. Figure 4 illustrates the states and behavior of the initiator agent.

The participant agent has four behaviors, and these are:

- **Waiting for a Call**, in which the participant waits a call for proposal from another agent.

Table 1. Initiator agent state-transition list

State ID	From behavior	To behavior	Condition
S1	Call for participants	Waiting for participants list	Sending out the message (call for participants)
S2	Waiting for participants list	Call for proposals	Receiving a message with non-empty list of agents or time-out
S3	Call for proposals	Waiting for proposals	Sending out the messages (call for proposal)
S4	Waiting for proposals	Responses for proposals	Receiving at least one message with a proposal
S5	Responses for proposals	Waiting for results	Sending out the messages (accept proposal)
S6	Waiting for results	Inform results	Receiving a message from the participant agents (results/failure)
S7	Inform results	Call for participants	Sending out the messages (inform results to the manager)
S8	Responses for proposals	Call for participants	Sending out the messages (rejected proposal). All proposals are rejected
S9	Waiting for proposals	Call for participants	Time-out

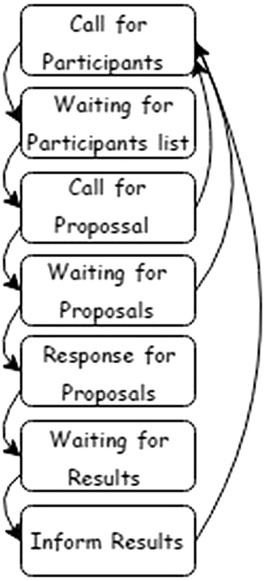

Fig. 4. Initiator agent internal structure

- **Proposing**, in which the participant sends a proposal to the initiator.
- **Waiting for Response**, in which the participant waits for a response from the initiator.
- **Inform Results**, in which the participant sends out the results of the implemented task to the initiator.

Six state transitions of the agent are found, which transfer the participant from one behavior to another as listed in Table 2. Figure 5 illustrates the states and behavior of the participant agent.

Table 2. Participant agent state-transition list

State ID	From behavior	To behavior	Condition
S1	Waiting for a call	Proposing	Receiving the message (call for proposals)
S2	Proposing	Waiting for response	Propose
S3	Waiting for response	Inform results	Receiving the message (accept proposal)
S4	Inform results	Waiting for a call	Sending inform results (results/failure)
S5	Waiting for response	Waiting for a call	Receiving the message (reject proposal)
S6	Proposing	Waiting for a call	Refuse to propose

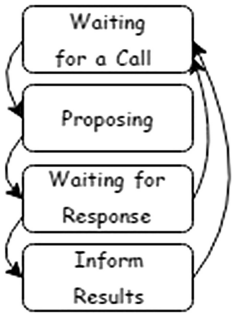

Fig. 5. Participant agent internal structure

The manager agent has two behaviors, and these are:

- **Waiting**, in which the manager waits a call for participants or receives the inform results from another agent.
- **Response**, in which the manager sends an agent list to the initiator.

Three states of the agent are found, which transfer the manager from one behavior to another as listed in Table 3. Figure 6 illustrates the states and behavior of the manager agent.

Table 3. Manager agent state list

State ID	From behavior	To behavior	Condition
S1	Waiting	Response	Receiving the message (call for participant)
S2	Waiting	Waiting	Receiving the message (inform results)
S3	Response	Waiting	Sending out the message (participant list)

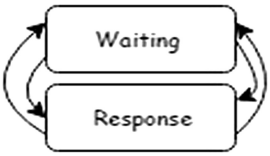

Fig. 6. Manager agent internal structure

However, the structure of these agents has to be merged because the agent can be the initiator and participant at the same time. The integration of the two agents as one is given in Fig. 7 and summarized in Table 4.

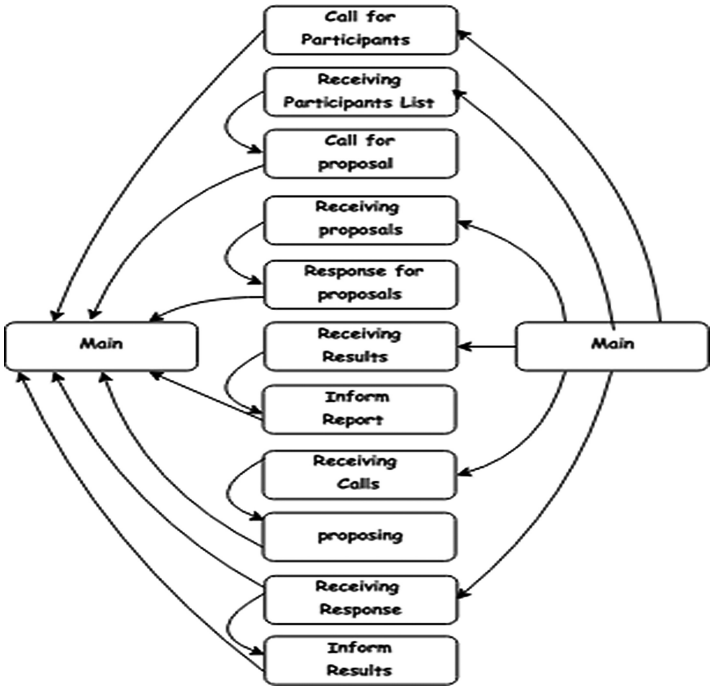

Fig. 7. Main agent internal structure

Table 4. Main agent state list

State ID	From behavior	To behavior	Condition
S1	Main	Call for participants	If there is a task to be negotiated
S2	Main	Receiving participants list	Receiving a message with non-empty list of agents or time-out
S3	Main	Receiving proposals	Receiving at least one proposal or time out
S4	Main	Receiving results	Receiving the results of the negotiated task
S5	Main	Receiving calls	Receiving CFP
S6	Main	Receiving response	Receiving response for a proposal (accept/reject)
S7	Call for participants	Main	Sending out the messages (call for participants)
S8	Call for proposals	Main	Sending out the messages (call for proposals)
S9	Response for proposals	Main	Sending out the messages (accept proposal/reject proposal)
S10	Inform report	Main	Sending out the messages (inform report to the manager)
S11	Proposing	Main	Sending out a proposal
S12	Receiving response	Main	The proposal has been rejected
S13	Inform results	Main	Sending out results of the implemented task
S14	Receiving participants list	Call for proposals	Processed the received list
S15	Receiving proposals	Responses for proposals	Processed the received proposals
S16	Receiving results	Inform report	Processed the received results
S17	Receiving calls	Proposing	Processed the CFP
S18	Receiving response	Inform results	The proposal has been accepted

4 Implementation and Results

The proposed protocol is simulated in Java using JADE, a Java API for implementing multi-agent systems. The advantages of JADE are having several graphical tools that are able to capture the process while debugging and running the implemented system, fully support FIPA specification, and capable to work in distributed systems. To prove the significance and the abilities of the proposed protocol as described earlier, several scenarios were tested under various conditions as summarized in Table 5.

Table 5. Testing cases

ID	#Agents	#Good	#Bad	#Tasks	Comments
1	2	2	0	2	Normal case, with two agents and a manger
2	4	3	1	4	Case with bad reputation agents
3	100	40	60	100	Compare with the contract net protocol
4	500	200	300	1000	Compare with the contract net protocol
5	1000	100	900	1000	Compare with the contract net protocol

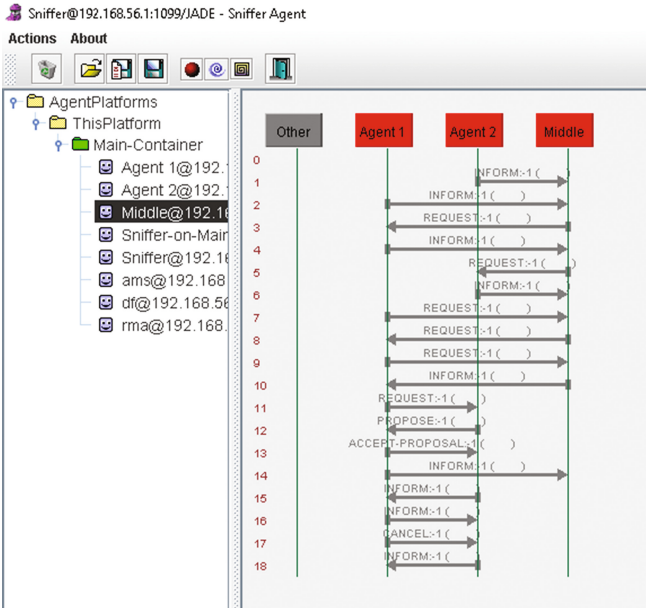

Fig. 8. First execution flow in JADE

The outcome of the first case is illustrated in Fig. 8. In this case, the proposed protocol performs normal task without any special interaction activities, as discussed previously. Meanwhile, agent four is considered as bad agent in the second case. Moreover, the proposed protocol does not allocate any job to the fourth agent, and the manager agent keeps asking for update from that agent (Fig. 9).

The time and number of messages for the rest of the cases are illustrated in Figs. 10 and 11, respectively. The proposed protocol and the well-known Contract Net Protocol reduce both time and communication overhead. This reduction is increased as the number of involved tasks is increased.

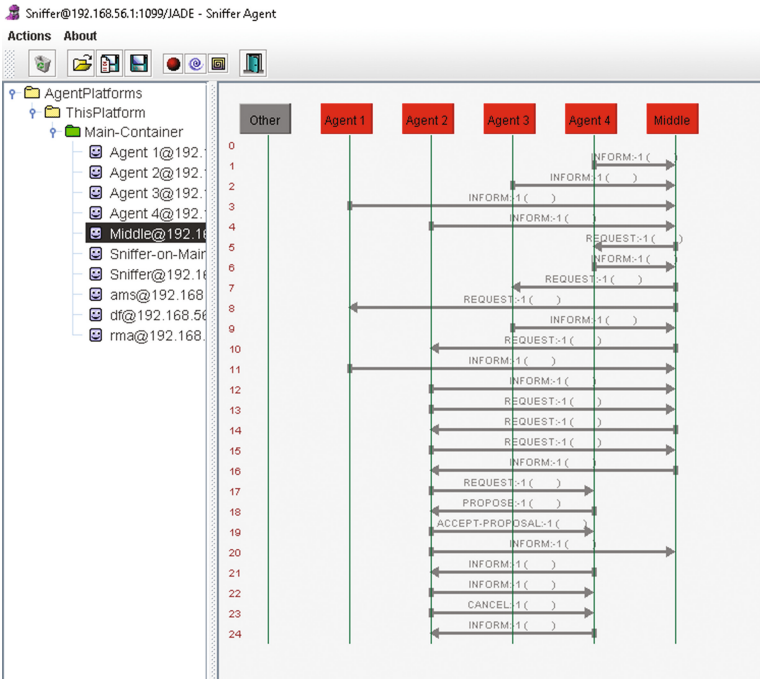

Fig. 9. Second execution flow in JADE

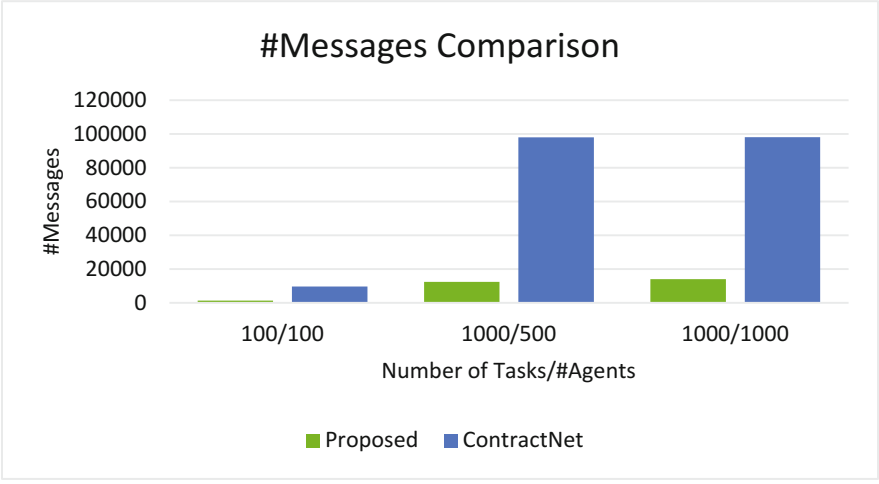

Fig. 10. Protocols evaluation based on message numbers

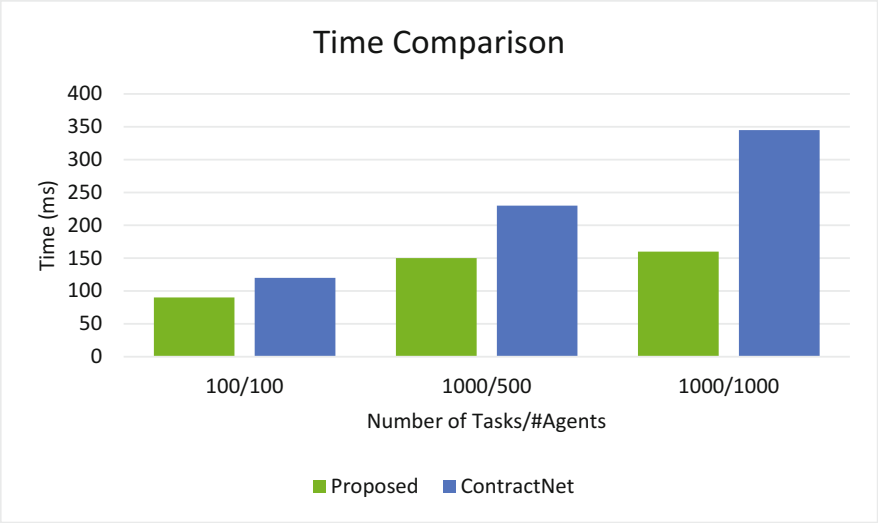

Fig. 11. Protocols evaluation based on execution time

5 Conclusion

This paper proposed a semi-centralized multi-agent communication protocol for negotiation tasks to overcome the limitation of the existing protocols, and enable robust and adaptable solution in a highly dynamic environment. The proposed protocol enhances the performance of the system as a whole by distributing the tasks to agents with good reputation to reduce the possibility of failing while implementing the task assigned, as the manager will select highly successful agents according to the saved history. The outcome of JADE's implementation proves the capabilities of the proposed protocol. In comparison with Contract Net Protocol, the proposed protocol shows significant improvement in time and communication overhead under various conditions.

Future work will focus on using one of the usual agents in the system as the manager agent. Subsequently, this agent can be replaced easily and dynamically at any time to increase the robustness of the proposed protocol.

References

1. Shen, W., et al.: Applications of agent-based systems in intelligent manufacturing: an updated review. Adv. Eng. Inform. **20**(4), 415–431 (2006)
2. Kersten, G.E., et al.: Shaman: software and human agents in multiattribute auctions and negotiations. In: Negotiation, Auctions, and Market Engineering, pp. 116–149. Springer, Berlin (2008)
3. O'Brien, P.D., Nicol, R.C.: FIPA: towards a standard for software agents. BT Technol. J. **16**(3), 51–59 (1998)

4. Labrou, Y., Finin, T., Peng, Y.: Agent communication languages: the current landscape. IEEE Intell. Syst. Appl. **14**(2), 45–52 (1999)
5. Dang, J., Huhns, M.N.: An extended protocol for multiple-issue concurrent negotiation. In: Proceedings of the National Conference on Artificial Intelligence, vol. 20, no. 1, p. 65, July 2005. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999
6. Williams, C.R., Robu, V., Gerding, E.H., Jennings, N.R.: Negotiating concurrently with unknown opponents in complex, real-time domains (2012)
7. Smith, R.: Communication and control in problem solver. IEEE Trans. Comput. **29**(12), 1104–1113 (1980)
8. Aknine, S., Pinson, S., Shakun, M.F.: An extended multi-agent negotiation protocol. Auton. Agents Multi Agent Syst. **8**(1), 5–45 (2004)
9. Weyns, D., Boucké, N., Holvoet, T., Demarsin, B.: DynCNET: a protocol for dynamic task assignment in multiagent systems. SASO **7**, 281–284 (2007)
10. Cao, M.: Agent communication language for automated negotiation online. J. Netw. **5**(6), 675–682 (2010)
11. Fip, A.: FIPA contract net interaction protocol specification (2001)
12. Hwang, P., et al.: Interaction protocols for a network of environmental problem solvers (2002)
13. Poslad, S.: Specifying protocols for multi-agent systems interaction. ACM Trans. Auton. Adapt. Syst. (TAAS) **2**(4), 15 (2007)
14. Williams, C.R.: Practical strategies for agent-based negotiation in complex environments. Diss. University of Southampton (2012)
15. Haim, G., An, B., Kraus, S.: Human–computer negotiation in a three-player market setting. Artif. Intell. **246**, 34–52 (2017)
16. Hao, J., et al.: An efficient and robust negotiating strategy in bilateral negotiations over multiple items. Eng. Appl. Artif. Intell. **34**, 45–57 (2014)
17. Panagidi, K., Kolomvatsos, K., Hadjiefthymiades, S.: An intelligent scheme for concurrent multi-issue negotiations. Int. J. Artif. Intell. **12**(1), 129–149 (2014)
18. Aydogan, R., et al.: Alternating offers protocols for multilateral negotiation. In: Modern Approaches to Agent-Based Complex Automated Negotiation. Springer, Heidelberg (2016)
19. Niu, L., et al.: A concurrent multiple negotiation protocol based on colored Petri nets. IEEE Trans. Cybern. (2016)
20. Adnan, M.H.M., et al.: Protocols for agent-based autonomous negotiations: a review. In: 2016 3rd International Conference on Computer and Information Sciences (ICCOINS). IEEE (2016)
21. Qasem, M.H., Al Assaf, M.M., Rodan, A.: Data mining approach for commercial data classification and migration in hybrid storage systems. World Acad. Sci. Eng. Technol. Int. J. Comput. Electr. Autom. Control Inf. Eng. **10**(3), 481–484 (2016)
22. Qasem, M.H., Faris, H., Rodan, A., Sheta, A.: Empirical evaluation of the cycle reservoir with regular jumps for time series forecasting: a comparison study. In: Computer Science On-Line Conference, pp. 115–124. Springer, Cham (2017)
23. Kadhum, M., Qasem, M.H., Sleit, A., Sharieh, A.: Efficient MapReduce matrix multiplication with optimized mapper set. In: Computer Science On-Line Conference, pp. 186–196. Springer, Cham (2017)