A Distributed Multi-Agent Meeting Scheduler System

Ali Durmus, Nadia Erdogan

Electrical-Electronics Faculty, Department of Computer Engineering

Istanbul Technical University

Ayazaga, 34469, Istanbul, Turkey.

agent meeting scheduler system [2]. The system uses a multi-agent paradigm, where independent agents are responsible for autonomously deciding how the task is to be achieved and actually performing the necessary set of actions, including handling interactions with other agents. Each agent knows its user's preferences and calendar availability in order to act on behalf of its user. Agents negotiate by having one agent propose a meeting, which the other agents accept or reject, based on whether or not it fits their own schedules. In Section 2, we describe the distributed multi-agent meeting scheduler system, with the internal architectures of the agents involved. Section 3 focuses on the meeting scheduling protocol in detail. In Section 4, we discuss the results of our basic experiments. In Section 5, we discuss system features and Section 6 gives conclusions.

2. A Distributed Multi-Agent Meeting Scheduler System

In the system we present, each individual is associated with a different agent. The agents manage the scheduling process on behalf of the individuals they represent, with no human interaction. Each agent has access to calendar and preference information of its user that are kept at different sites, in accordance with the distributed nature of the problem.

A meeting has a date, a start time, and duration and it is

scheduled when all agents reach an agreement on values for these attributes. As, in real life scheduling process, finding a suitable location is another problem; locations where meetings can be arranged are also taken into account. Physical state information, such as capacity, calendar, and equipment possessed, of all locations are held in a central site.

The system consists of two types of agents: *scheduler agents*, which negotiate with one another on behalf of their users, and a *location agent* that holds location information and negotiates with scheduler agents. Our system does not have a fixed central control. This means that there is not a specialized control agent and each agent is able to try to schedule a meeting via negotiation. Thus, a scheduler agent can be in the *organizer role* when its user requests a meeting and coordinates the meeting scheduling process, or it can be in the *invitee role*, as a participant of a meeting. Several meetings can be undergoing scheduling. Therefore, an agent can simultaneously be involved in scheduling any number of meetings, acting as an organizer for some and an invitee for others.

We have classified invitees into two groups: *important invitees* and *regular invitees*, to meet real world requirements. Important invitees have to attend a meeting; therefore a meeting can only be scheduled only if all of the important invitees agree on a time slot. Attendance of regular invitees is not a necessity, but a certain number of them may be required to attend in certain cases.

2.1. Internal Architecture of Scheduler Agent

The architecture of a scheduler agent, as it is seen in Figure 1 , consists of the following components:

Preference Module: This module receives requests from the negotiation module and from the user interface to query user preferences about calendars and meetings. It also receives data from the user interface to update the preference database.

Preference Database: It holds user's meeting preferences. It provides information on calendar days or time intervals when the user wants no meetings. The database also holds additional information on the degree of privacy, showing the extent to which preference information can be exposed to other agents.

Calendar Module: This module receives requests from the negotiation module to query and update calendar information. It also receives requests from the user interface that query calendar information.

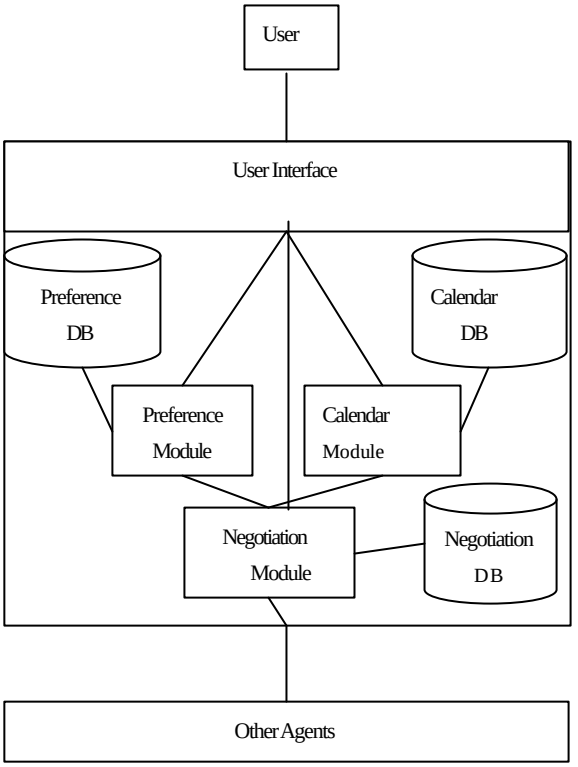

Figure 1: Internal Architecture of Scheduler Agent

Calendar Database: It contains personal calendar information.

Negotiation Module: It is the most important module in the agent structure. It receives requests and data as input from the user interface and coordinates the scheduling process by communicating proposals to other agents in the system. It also negotiates with other agents on their further meeting requests as an invitee. It keeps the information that it receives from other agents for a certain meeting scheduling session at the negotiation database and uses them to produce new meeting time proposals and counter meeting time proposals. If acting as an invitee, it produces acceptance and rejection responses to other agents' request messages.

Negotiation Database: This database holds information gathered from the messages of other agents for each meeting scheduling session. The negotiation module uses these data to reason about other agents' calendar information.

User Interface: This is the human interface to the system as it is shown Figure 2. It receives data and requests from the

user to schedule a meeting and it forwards them to the negotiation module. It also receives queries and update requests from the user about preference and calendar information. It forwards the requests to the target modules and presents the resulting information to the user.

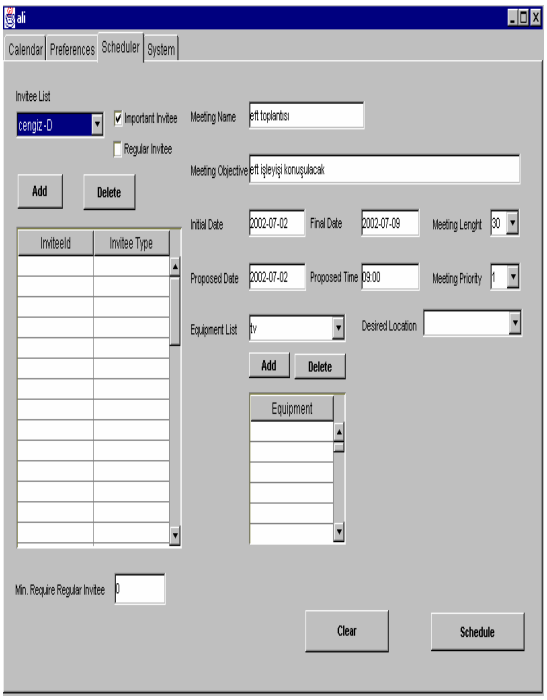

Figure 2: User Interface of Scheduler Agent

physical information. It produces a counter proposal in some cases.

Negotiation Database: This database holds information, gathered from other agents' messages, for each meeting scheduling session.

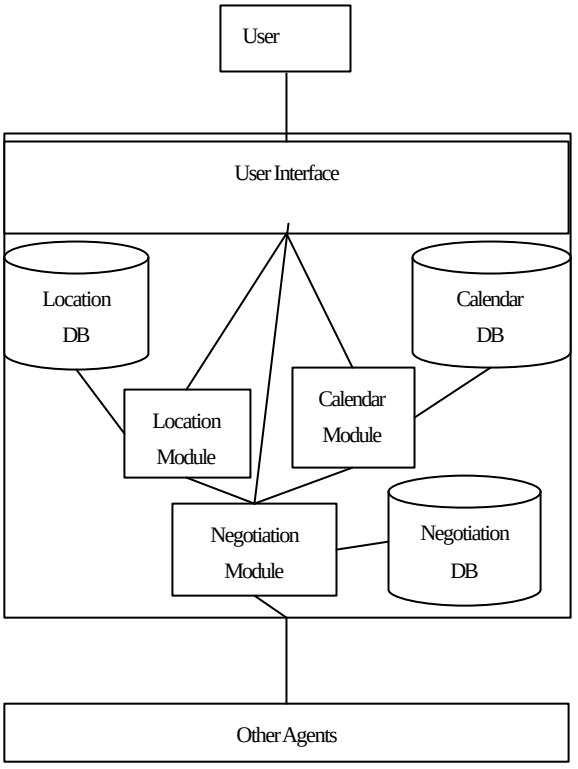

Figure 3: Internal Architecture of Location Agent

2.2. Internal Architecture of Location Agent

The architecture of the location agent, as it is seen in Figure 3, consists of the following components:

Location Module: This module receives queries and update requests to query and update physical information of locations where meetings can be held

Location Database: It holds physical information of locations where meetings can be held. Physical information includes capacity and equipment present.

Calendar Module: This module receives requests form the negotiation module and from the user to query and update calendar information.

Calendar Database: It holds calendar information of locations.

Negotiation Module: It receives messages from other agents to schedule a meeting and replies with acceptance or rejection messages, according to the location calendar and its

User Interface: It enables the user to query and update the location and calendar database by means of the location module and calendar module.

3. The Meeting Scheduling Protocol

Before presenting the meeting scheduling algorithms, we want to focus on certain decision criteria that influence the efficiency and speed of the scheduling process.

Search Bias: Distributed meeting scheduling can be thought of as a distributed search process because agents search for a suitable time interval in their calendar by using their calendar and preference information to make a proposal or a counter proposal for a meeting time. There are three types of search bias: linear early, linear least dense, and hierarchical, that can be used in searching a calendar [3].

In the system we present, we have used linear early

search bias, as it is the simplest and the easiest to implement. We have implemented it making an agent start searching its calendar at the earliest possible scheduling opportunity, skipping over any intervals overlapping with already scheduled meetings, and negotiating with the earliest free interval on the calendar long enough to accommodate a meeting.

Counter Proposal: An invitee agent can be a passive attendant who replies questions with simply yes or no, or can be an active attendant who makes a counter proposal for the meeting time when it rejects an agents' proposal. The ability to make a counter proposal by taking agents' calendar and preference information into account shortens the distributed meeting scheduling process [4]. In the system we present, an invitee who rejects a proposal produces a counter proposal, giving the nearest later time when it can meet, in accordance with the linear early search bias.

3.1. The Meeting Scheduling Algorithm

A user who requests a meeting supplies the following information to the organizer agent initially through the user interface:

- Meeting name and objectives
- Meeting length in hours
- Initial and final date of the time interval. For example, between 11.03.2002 and 15.03.2002.
- Initial proposed time interval for the meeting (optional)
- Attendants of the meeting: important and regular invitees separately
- Equipment needed at the meeting (for example, projector, television, computer...)
- Desired location that user prefers the meeting to take place in (optional)

Location of attendants, that is the traveling distance between two appointments, is ignored in our algorithms.

1. The organizer agent receives data and meeting scheduling request from the user. If the user specifies initial time slot, the organizer agent takes it as an initial proposal. If not, by using its calendar and preference information it tries to find the earliest time slot where the meeting will fit. If it can't find one, the scheduling process ends with failure. If it finds one, it announces the meeting to all invitee agents and the location agent through a proposal message. Messages that are sent to invitee agents and the location agent have a different format. The organizer agent blocks that particular time slot in its calendar.
2. An invitee agent that receives the announcement proposal evaluates the bid by using its calendar and preference information. After the evaluation, it either accepts or rejects the proposal.
   - If the proposed time slot is free in its calendar, it accepts the proposal. It blocks the proposed time slot in its calendar and unblocks time slots that were previously blocked for this meeting. Next, it sends an acceptance message to the organizer agent.
   - If the proposed time slot is not suitable for the invitee agent, it tries to find a counter proposal time slot by using the features of the meeting, its calendar and preference information. It sends the organizer agent a message that contains the counter proposal if it can find a free time slot. The message also contains conflicting time intervals that caused the rejection of the organizer agents' proposal if the agent's degree of privacy permits to send reasons. The invitee agent unblocks time slots that were previously blocked for this meeting and blocks the counter proposal time slot in its calendar if it has created one.
3. The location agent that takes the proposal evaluates the bid by using its calendar, desired property of the meeting place and the features of available locations. If it can find a suitable location for the meeting at the proposed time slot, it sends the organizer agent an acceptance message. If it can't find a suitable location, it tries to find a counter proposal time slot in its calendar. Then, it sends the organizer agent a message that contains rejection response and a counter proposed time slot, if found. The location agent unblocks time slots that were previously blocked for this meeting and it blocks the counter proposal time slot in its calendar if it has sent one.
4. The organizer agent evaluates the response messages:
   - If it receives acceptance messages from all invitee agents and the location agent, it sends them a confirmation, meeting successfully scheduled message and commits blocked time slots in its calendar for this meeting.
   - If it receives acceptance messages from all important invitee agents, a predefined ratio of regular invitee agents, and the location agent, it sends them successfully a meeting successfully

scheduled message. It sends cancellation messages to the regular invitee agents that have rejected the meeting and commits blocked time slots in its calendar for this meeting.

- If it doesn't receive an acceptance message either from the location agent or at least from one important invitee agent or from a predefined ratio of regular invitee agents, it evaluates the counter proposals of invitee agents and the location agent, using the criteria we've mentioned above. The counter proposals that are common are sent to invitee agents and the location agent. The agents that receive these new proposals send responses in yes or no format, according to their evaluations. If a common counter proposal is not present, the organizer agent stores reasons of rejection from invitee agents at the negotiation database. Next, it tries to find a new free time slot by using the information resulting from rejection arguments, its calendar and preference information, and the meeting information supplied by the user. It chooses the farthest time from the proposed time returned by invitees if it fits its calendar or tries to find another time slot beyond that time. As it is known that invitees respond with the closest time to the proposed time slot in their counter proposal according to the linear early search bias, the earliest time the organizer should propose in the next iteration is the farthest time from the proposed time slot. This saves a number of extra iterations because if the yes/no strategy was chosen, with no counter proposals, the organizer would have to step through its calendar for each available time slot and get a negative response until it reached the nearest available times of the invitees. If it cannot find a suitable time slot, it sends cancellation messages to all agents involved and unblocks time slots that were previously blocked for this meeting in its calendar. Otherwise, it continues at step 2 with the new proposal.

5. The agents that receive a successfully scheduled message commit blocked time slots in their calendars for that meeting. The agents that receive a cancellation unblock time slots that were previously blocked for this meeting in their calendar.

The algorithm does not allow for cancellation of prescheduled meetings.

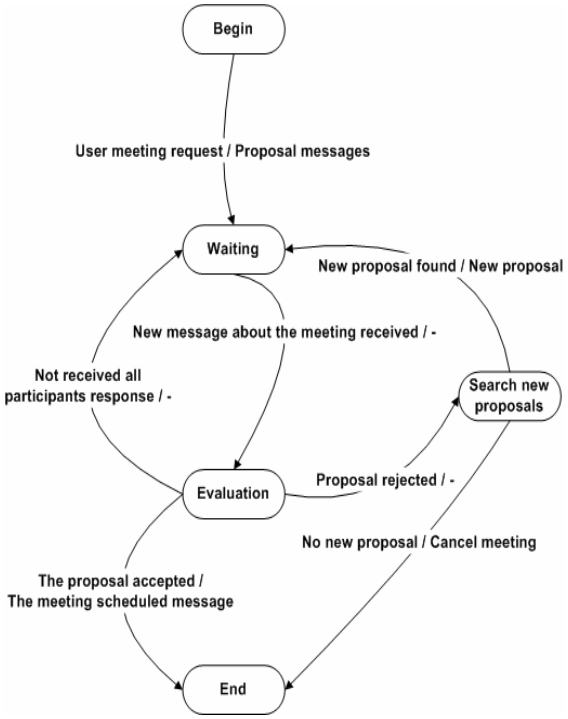

Figure 4: State Transition Diagram of an Organizer Agent

3.2 State Transition of Agents During Scheduling

We will use state transition diagrams to describe state evolution of agents during the negotiation protocol. Figure 4 and Figure 5 show the state transitions of an organizer and an invitee agent respectively. For each transition there is an input and output message, in the a/b format, where a represents a condition that exists or an input message that is received and causes an output message b to be sent. The symbol "-" indicates the absence of an input or an output message.

When an organizer agent is in the initial state, it receives a meeting-scheduling request from its user. It sends a proposal message to the invitee agents and the location agent and enters a waiting state. On each newly received message, the agent passes into an evaluation state where message content is evaluated and the presence of certain conditions are checked. If all reply messages have not been received yet, the agent goes back to the waiting state.

If messages from all participants for that meeting are received and the proposal is accepted, it sends meeting successfully scheduled message to all participants and goes to the end state. However, if the proposal is rejected, it enters the search new proposal state. In this state, if a new proposal

can be produced, it sends this proposal to all participants and goes back to the waiting state. Otherwise, it sends a cancel meeting message to all participants and enters the end state. A similar analysis is relevant for the state transition diagram of an invitee agent that is shown in Figure 5.

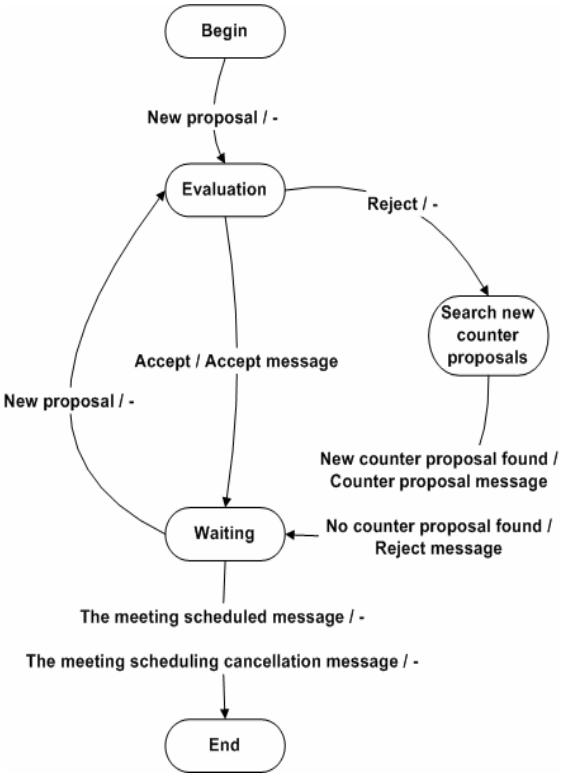

Figure 5: State Transition Diagram of an Invitee Agent

4. Experimental Results

We have implemented the above-proposed meeting scheduling protocol using the JATLite agent system [5]. In the system, independent autonomous agents, each located on different computers, communicate and negotiate with each other over the network in order to schedule meetings in a distributed way. We have identified some experimental variables, such as the duration and the number of participants of a meeting, the time interval when a meeting should be scheduled, number or locations where a meeting can be held, that we think effect the scheduling process and have carried out a number of experiments with those variables as parameters. We consider the scheduling process between 8 agents that act on behalf of their users and a location agent. Each agent owns a calendar of 1,2,3, or 5 days, depending on the experiment, each with time slots being initially free. Possible durations of meetings are 30, 120, or 180 minutes and meetings may be requested to be scheduled in the next 1, 2, 3, or 5 days (time intervals) of the calendar. If not otherwise stated, experiments involve 8 agents, each trying to schedule meetings of the same duration and in the same time interval concurrently, with the remaining 7 agents expected to take part as invitees.

We have observed that, in situations where concurrent scheduling of several meetings is taking place, some meetings block time slots that cause other meetings to be abandoned due to lack of available times within the meeting's time interval. However, those blocked slots might be released later. Therefore, we have slightly modified the scheduling algorithm by starting a new iteration in case of a scheduling failure. As blocked time slots that cause a proposal to be rejected might later become free, a second attempt to schedule a meeting may lead to success. In real life, however, one cannot judge if further iterations will result in more meetings scheduled successfully as all information is distributed and there is no central controller agent that has access to all the information that guides the scheduling process. As concurrent scheduling of several meetings would be a rare situation, we think real life scheduling need not be carried out with iterations. In the following experiments, the actual time to schedule a meeting is directly proportional to the number of iterations required to schedule it. The following discussion is on the results we have obtained from those experiments.

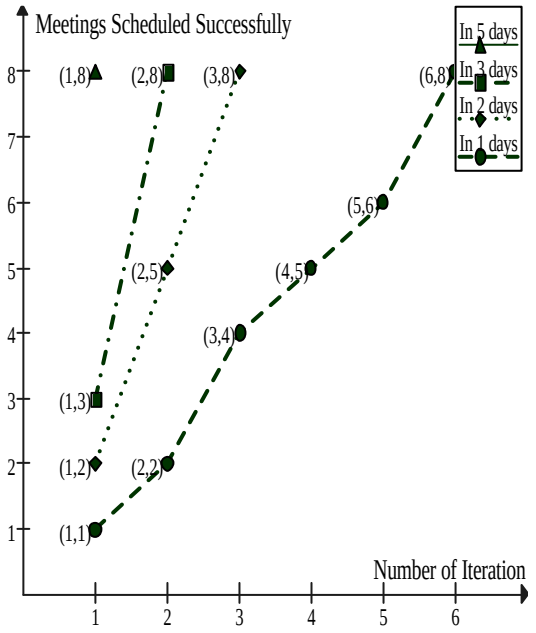

Figure 6: Results for Experiment 1

Experiment 1: Scenario: All 8 agents concurrently request meetings with the remaining 7 agents as invitees. The length of the meeting is kept constant at 30 minutes and there is a single meeting location. The basic parameter of the experiment is the time interval in which meetings can be scheduled and it is varied to 1, 2, 3, and 5 days. Figure 6. plots the number of scheduled meetings for each number of iterations determined experimentally. As expected, decrease in the time interval increases the number of iterations for successful scheduling. All of the eight meetings were scheduled in a single iteration when the time interval was the longest. However, as the time interval was shortened, more iteration was necessary. For example, for the time interval of 2 days, the first iteration was able to schedule two meetings only, and three more were added with each following iteration.

Experiment 2: Scenario: All 8 agents concurrently request meetings with the remaining 7 agents as invitees. The parameters of the experiment are the length of a meeting and the time interval in which it can be scheduled. The length of meetings is varied to 30, 120, and 180 minutes. The time interval is varied to 1, 2, and 3 days. No iteration is applied. Figure 7 plots the number of meetings of different lengths scheduled successfully for different time intervals. As the number of available time slots increases for decreasing lengths of meetings, we observe that a greater number of shorter meetings can be scheduled in a certain time interval when compared with the numb er of longer meetings in that time interval.

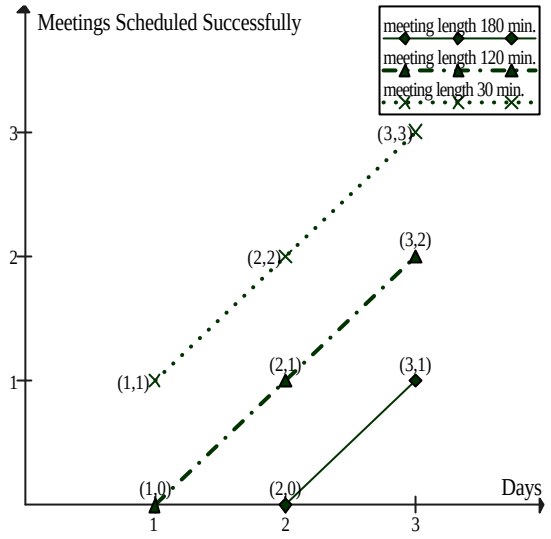

Figure 7: Results for Experiment 2

Experiment 3: Scenario: In this experiment, the length of a meeting is kept constant at 30 minutes. The parameters of the experiment are the number of agents involved in the scheduling process and the time interval in which meetings are to be scheduled. The number of agents that simultaneously request meetings to be scheduled are varied to 8, 6, and 4 while the time interval is varied to 1, 2, and 3 days. As observed in Figure 8, since the number of time slots for a particular meeting duration is constant, as the number of agents increases, the number of meeting successfully scheduled decreases for each time interval

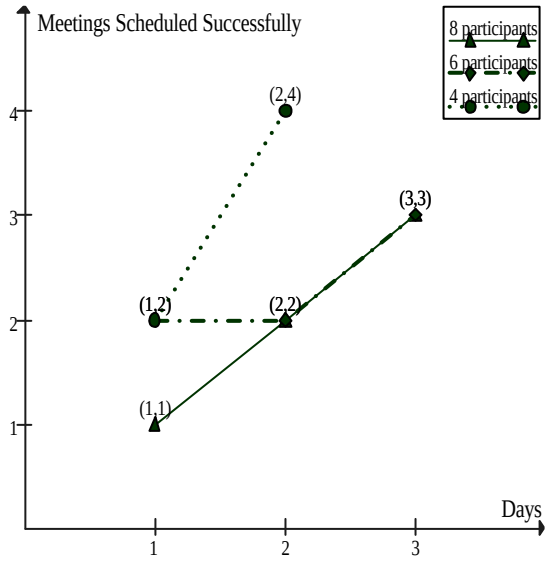

Figure 8: Results for Experiment 3

5. System Features

The distributed multi-agent system that we have presented in this paper has the following features:

Consistency with the characteristics of the problem: The system is in consistence with the characteristics of real life meeting scheduling process, as meeting scheduling in real life is distributed in nature, with autonomous participants.

Autonomous: Agents make their decision autonomously when they receive a request from their environment. When they make their decision they use both their knowledge and the users' knowledge that the agents act on behalf of. The agents act autonomously on behalf of users and don't need their intrusion.

Intelligent: Agents in the systems make decision by using users' data, data gathered from other agents during the meeting scheduling process and rules defined in the meeting scheduling algorithms. This causes agents in the system to be intelligent.

Learning: Agents in the systems try to make inference from information gathered from other agents supplied as rejection reasons. They try to learn parts of other agents' calendar in order to use that new knowledge in search of new proposals.

Collaborative: Distributed meeting scheduling process requires participation of all attendees. Agents in our system work collaboratively to solve the meeting-scheduling problem successfully.

Flexibility: The system presents two kinds of flexibility. One of them is supplied by JATLite message router that enables agents to connect/disconnect to system any time. Their message is not lost while they are in disconnected state. Agents can also connect to the system from any place on the network, which means that their IP can change between connections. The second flexibility is that new agents can easily be added to system. They only need to register to the system in order to be part of it.

6. Conclusion

In this paper, we have presented a new distributed multi-agent meeting scheduler system, with its design and implementation details, and a new meeting scheduling protocol. We have used some techniques in our protocol to shorten the meeting scheduling process time. One of the techniques is the use of counter proposals that give the organizer agent clues on nearest available time slots of invitees and eliminates several unproductive attempts. Also, agents in our system expose some private information, explaining why when they reject a meeting proposal. The organizer agent that acquires other agents' calendar information uses this information to produce better proposals, which, in turn, shortens scheduling time. We believe that our approach of distributed scheduling in a dynamic domain can successfully be applied to a wide variety of scheduling problems.

References

[1] Eaton, P. S., E. C. Freuder, R. J. Wallace, "Constraints and Agents: Confronting Ignorance", AI Magazine, Summer 1998, 51-65.

[2] Ali Durmus, " A Distributed Multi-Agent Meeting Scheduler System", Msc. Thesis, Istanbul Technical University, Institute of Science and Technology, 2002.

[3] Sandip Sen and Edmund H. Durfee : Unsupervised Surrogate Agents and Search Bias Change in Flexible Distributed Scheduling. Proceeding, First International Conference on Multi-Agent Systems p.336-343 San Franscisco, CA June 1995.

[4] Sandip Sen : An automated distribut. meeting scheduler. IEEE Expert 12(4) p.41-45.

[5] JATLite (Java Agent Template, Lite) – http://cdr.stanford.edu/ ProcessLink/ Papers/