Agent-Based Meeting Scheduling Support using Mobile Clients

Markus Niederer, Alexander Schatten
Institute of Software Technology and Interactive Systems
Information and Software Engineering Group
Vienna University of Technology
Vienna, Austria
{niederer,schatten}@ifs.tuwien.ac.at

Abstract

In this paper, we present a distributed multi-agent meeting scheduling system for mobile devices. This mobile agent system supports people on the move to locate feasible time slots to meet with each other, using their mobile devices. It negotiates the best suitable times integrating timetables and time preferences from the participants. Unlike to current approaches, the negotiation of the meeting time is fully executed on mobile devices. As mobile devices are limited in resources, the introduced protocol additionally allows, to outsource the main part of the negotiation to a self-contained scheduling service.

1 Introduction

In the last years the number of people owning and using mobile phones or personal digital assistants (PDAs) increased significantly. The prices for mobile communication including mobile internet services seem to decrease constantly. The results are obvious: People are potentially reachable anywhere and at every time. SmartPhones and PDAs have become our permanent companions. On the other hand, time has become an invaluable property. Not only in business life the dictum "time is money" faces us every day. But scheduling meetings has become an increasing difficult activity for busy people. Several applications support people find time slots to meet each other. The techniques to schedule meetings range from simple "free-busy" calculations as in Microsoft Outlook [11] and Lotus Notes [9] to extended agent- and preference-based approaches [2, 3, 6, 7].

It seems natural to bring scheduling applications to mobile devices, as they are permanently available and additionally often have calendar and addressbook functionality already in place. But mobile devices do still have limitations in computing resources, most of all in visualisation and usability. However, with every generation mobile devices become more powerful and hence open possibilities for new areas of application.

We analysed several academic and commercial meeting scheduling applications and approaches. All analysed systems require at least one basis station, most of the time the user's personal computer; either to perform the negotiation of meetings, or to present the negotiation results [14]. In our opinion none of the investigated concepts are primarily designed to be used on mobile devices. In this paper we will introduce a meeting scheduling concept and prototypical implementation for mobile devices called *MeetMe*. The MeetMe agents are designed to support meeting scheduling using mobile devices. Unlike the current approaches, the negotiation of the meeting can be fully executed on mobile devices. The agents are running in an automated and distributed setting.

This paper is structured as follows: In section 2 we will describe current approaches of scheduling systems, take a look at their negotiation algorithms and evaluate their usefulness for mobile devices. In section 3 the scheduling concept of MeetMe and the negotiation protocol are introduced. Section 4 describes the development of the MeetMe prototype and gives a system overview. Finally in section 5 we give conclusions and an outlook for future research work.

2 Related Work

Probably one of the most spread procedures in the business world to schedule meetings is by using Microsoft's Outlook [11]. To schedule a meeting efficiently, the initiating user needs to have access to all the participants' (public) calendars.

A number of limitations and problems with Outlook have been shown: Although the proposed time is marked free in a user's calendar, there could be an earlier request for the same time by another meeting. Furthermore the shared cal-

endar approach does not allow users to give detailed information about there preferences when they would like meetings to take place. It is quite possible that an unfavourable time will be finally chosen [5].

Furthermore Outlook is limited to propose only one time slot in every scheduling iteration. Outlook offers the opportunity to give counter-proposals, but every new suggestion again has to be sent to each attendee. Besides the users can only accept or deny a proposal, Outlook does not implicate a "soft" weighting of a proposed time slot, like "This time slot is ok, but the other one is better".

To decrease the human communication overload while scheduling meetings, the main part of the indispensable work could be delegated to personal scheduling agents.

In the protocol of Sen and Durfee every meeting has a particular agent (called host), responsible for the scheduling process. The host contacts the other attendees' agents (called invitees) to announce the meeting using a bidding process [13].

As an extension of the shared personal calendar style approach of Microsoft's Outlook, Crawford and Veloso designed an "Open-Negotiator", to schedule and re-schedule meetings in a fully automated and distributed setting [5].

Berry et al. have introduced a personalised agent for time management and meeting scheduling called PTIME [1]. The PTIME agent is a single-calendar scheduler dependent on other agents to coordinate shared calendar entities.

Chun et al. introduced an agent-based negotiation algorithm within an environment they called "Mobile Agents for Office Automation" (MAFOA) [4] using two types of agents: a secretary and a meeting agent.

Hasine and Ho introduced a model to solve dynamic meeting scheduling problems [8]. They designed a distributed approach based on the DARC[1] model using a peer-to-peer approach.

An application designed to be partly used on mobile devices was introduced by Mynatt and Tullio [12]. They implemented a system to visualise the attendance likelihood and events co-scheduled by the users colleagues. The user's calendar data is hosted on the user's mobile Personal Digital Assistant (PDA) using synchronisation and a matching module on a central instance.

Except for the concept of Tullio et al. [14] the current approaches have fully disregarded the increasing request of meeting attendees to be unbound to fixed desktop systems. None of the approaches discussed above are designed to accomplish the vital need to be used on mobile devices. Furthermore, it is necessary for such a system to be accessible independent from the organisational structure the attendees belong to, i.e., a meeting should be negotiable although the participants are working for different companies. The evaluation of the current approaches has indicated, that shared calendar systems are inappropriate for organisation overlapping meeting scheduling.

Agent-based systems have shown much better performance in scheduling meetings in a distributed setting. Furthermore small independent agents seem to be the best approach to support meeting scheduling for mobile devices. The agent-based system should communicate through a wide spread and easy adaptable protocol. To use a well known protocol has big advantages: Most firewalls have standardised roles for this protocols, which securely allow the required data to pass. Furthermore it is easy for IT administrators to configure their firewalls to only let pass needed information.

In 2004 the Internet Engineering Task Force formalised the communication protocol for messaging and presence called XMPP (eXtensible Messaging and Presence Protocol) as proposed standard [10]. Initially design for instant messaging between humans, the protocol is also applicable to exchange other data. It allows near-real-time transfer of nearly every type of data and in the meantime, it has become a established protocol as well.

3 MeetMe Scheduling Concept

In our opinion a modern scheduling-support system should be capable in supporting a wide variety of different clients including desktop clients and mobile clients. For better flexibility a proprietary centralised architecture should be avoided and open standards for communication and negotiations are preferable to allow a broad range of services to connect to this system. In this section we introduce the MeetMe scheduling concept. First we identify three types of agents. In the second part of this section, we illustrate how the agents are orchestrated to negotiate meetings. Finally we introduce our negotiation algorithm, adopted to the requirements of mobile devices.

3.1 Types of Agents

We define three types of actors, taking part while a meeting is scheduled:

Initiator The person that hosts the meeting and invites the required persons. The initiator is responsible for the meeting and the negotiation of the best feasible time. He or she decides where the meeting is held and who to invite. Moreover the initiator acts as a kind of supervisor in cases of error during the scheduling process, e.g., he or she decides how to proceed, when no time slot can be found, or when one or more participants reject the final result. The initiator can also be the coordinator and/or a participant of the meeting. Only the initiator can cancel the meeting scheduling process.

[1] distributed reinforcement of arc consistency

Coordinator The host that technically coordinates the scheduling of the meeting. This is the point where all messages and responses intersect during the scheduling process. The coordinating agent requests the timetables from the participants, calculates the best fitting time slots, distributes the calculation result, matches the preferences and instructs the participants' agents to add the final result to their users calendars. Furthermore the coordinating agent keeps the initiator up-to-date, by sending status messages and answering status requests.

Participant One or more invitee(s), that are supposed to attend a meeting. The coordinator and the initiator can be participants as well. Participants are classified as *Very important*, *Mandatory*, *Minor* or *Info*. Attendees characterised as *Very important* or *Mandatory* have to be available for the meeting to be confirmed. When no time slot can be found, where all of the mandatory participants are free, the initiator receives an error message. The meeting can also not be scheduled, if one or more of the *Very important* or *Mandatory* participants reject the final result. The difference between these two classifications is that timetables of the *Very important* attendees are weighted higher and therefore do have more influence on the negotiation of the best time slots. *Minor* participants can attend a meeting, where users classified as *Info* will only receive an information about a meeting after the meeting is accepted by every participant. This classification is extending the classification of [4] with the superordinate instance of *Very important* attendees and the *Info* classification for users not participating the meeting, but want to be informed about when the meeting will occur.

Each agent is able to act as Initiator, Coordinator or Participant. The main part of the negotiation work is done by the coordinating agent. The more persons are invited to participate an event, the more difficult for the coordinating agent to calculate the best feasible time. As mobile devices are limited in resources, the negotiation could be outsourced to a detached scheduling service.

3.2 Orchestration of the Scheduling Process

This section describes the orchestration of the scheduling process between the initiating agent (also acting as participant), the coordinating agent and one participant agent. The process is only prototypical and can be enlarged as desired. The initiator wants to schedule a meeting with the second participant, while the third agent is coordinating the negotiation. As mentioned above, the coordinating role can be held either by the initiating agent, one participant, or any other agent (a specialized coordination service agent for example).

To set off the negotiation, the initiator records all the required data like time horizon (e.g. *this week*, *next week*, *in the next two weeks*, etc.), *duration*, *location* of the meeting and the user names of the participants. The initiator starts the scheduling process by sending an coordination request together with the entered data to the coordinating agent. When the coordinating agent accepts the coordination of the meeting, it sends a timetable query request to each participants' agents.

The participants' agents now calculate the times of availability of their users. Therefore the agents contact their calendar services to retrieve the necessary availability data for the requested time horizon. With this data, the agents calculate an weighted timetable, depending on the preferences of their user.

After receiving the times form the participant agents, the coordinating agent matches the timetables and calculates possible time slots for the meeting. If no time slot can be found, the process stops with an error and the initiator can decide to expand the time horizon for the meeting, or cancel the negotiation process.

When the previous step has generated one or more possible time slots to meet, the times are presented to the participants and they can estimate these time slots. The preferences are matched again by the coordinator. When there are no time slots left, because the users did not like one of the time slots or they prefer different time slots, the initiator gets an error message and he can decide to cancel the process or start all over by choosing an other planing horizon.

When one or more time slots are left, it is the coordinators decision to determine the time of the meeting. The participants again do have the possibility to decide whether they accept or deny the final result. When one of the participants deny the final meeting time, the initiator gets an error message and he can decide to cancel the process, or start all over choosing an other planing horizon. After all participants have accepted the time, the meeting is fixed and the users can add the meeting to their calendars.

3.3 Negotiation Algorithm

The coordinating agent first requests the timetables from each participants' agent. Each time slot from the attendees timetable is augmented with individual weights, representing the participants' preference to schedule a meeting. When a user initiates a new meeting to negotiate, he or she assigns each participant with an individual priority for this meeting (*Info*, *Minor*, *Mandatory* or *Very important*). The weighted timetables are matched by the coordinating agent, considering the priority level of each participant. Therefore the time slots are divided into predefined intervals. The weights for these intervals then are normalised and aggregated, taking account of each participants priority. Finally the intervals are combined again and the algorithm selects the time slot with the highest overall weight, suiting the es-

timated duration of the meeting. If there are more than one time slots with the same weight, the earliest time slot is selected.

Similar to Wainer et al. [15] a meeting m is defined as a 6-tuple $\langle A, \bar{a}, c, d, H, O \rangle$, where

- $A = \{a_1, a_2, \ldots, a_n\}$ are the participants agents,
- $\bar{a}$ is the initiator of the meeting. Commonly $\bar{a} \in A$, but the protocol allows, that a initiator is not participant of a meeting. In this case $\bar{a} \notin A$,
- c is the coordinator of the meeting. This is the agent, that supervises the orchestration between the agents A and the initiator $\bar{a}$, and performs the negotiation task of the meeting scheduling. In most cases $c \in A$ and $c = \bar{a}$. The coordinating task can also be outsourced to an scheduling service, in this case $c \notin A$,
- d is the planed duration of the meeting,
- $H = \{h_s, h_e\}$ the planing horizon defined as the start and the end of the time horizon to schedule the meeting,
- $O = \{o_1, o_2, \ldots, o_n\}$ set of the participants priority for the meeting to schedule.

First the participants' agents A are requested to generate a set of weighted time slots $U(a_i, m)$ according to their users preferences, where

- $U(a_i, m) = \{u_1, u_2, \ldots, u_k\}$ set of weighted participant time slots for the meeting.

After each participants' agent A has executed the timetable generation, the coordinating agent c transforms the weighted timetables U of the participants A into a set of time slots T of the length v, where T is between $h_s \in H$ and $h_e \in H$, and where

- $T(a_i, m, U, v) = \{t_1, t_2, \ldots, t_g\}$ set standardized time slots of the length,
- v length of the interval of T,

Finally, to receive the most feasible time slot for the meeting m, the coordinating agent c executes for the number of participants N the function

$$W_j(m, T_j) = \sum_{i=1}^{n} \frac{1}{N} \cdot |T_j(a_i, m, U_i, v)| \cdot o_i$$

The higher the value of $W_j(m)$, the more preferred the meeting m to schedule during the time slot t_j. The time slots with the highest overall weights $\max_{W(m,T)}$ are chosen as most feasible times for all participants Q, where

- $Q = \{q_1, q_2, \ldots, q_z\}$ set of z times with the highest overall weight $\max_{W_j(m,T)}$.

If some time slots have the same weight, the earliest time slot will be chosen. Q is sent to the participants' agents A as result of the calculation.

After the possible time slots for the meeting are calculated, the coordinating agent c sends the time slots with the highest overall weight Q to the participants' agents A and requests them to estimate their preferences $R(m, Q)$. When the coordinating agent has received the rankings $R(m, Q)$ from the participants A, it executes the preference matching function $S(m, R)$, to find the most preferred time slot for the number of participants N:

$$S_j(m, R_j) = \sum_{i=1}^{n} \frac{1}{N} \cdot |R_j(m, Q_j)| \cdot o_i$$

The higher the value of $S_j(m, R)$, the more the participants A prefer to schedule the meeting m at the time q_j. The time with the highest overall ranking $x = \max_{S(m,R)}$ will be distributed as final result.

The algorithm is extending the approach of Wainer et al. with the participants priorities for the meeting. Depending on the importance of the invitee, the preferences have greater or less influence on the final result. In the approach of Wainer et al. the preferences of each attendee are treated equal.

4 Prototypical Implementation

From the results of our study, we implemented a prototype of MeetMe, based on the new open platform for mobile devices Android[2]. Figure 1 shows the general architecture of the system. The agents communicate over XMPP [10] the Extensible Messaging and Presence Protocol using the Open Source Smack API[3]. The agent implementation is based on four layers:

- *MeetMe Client* - The user interface of the agent to interact with the user.
- *MeetMe Service* - The background service coordinating the scheduling of the meetings and listening for incoming requests (scheduling and coordinating requests).
- *MeetMe API* - The platform independent implementation of the scheduling protocol and algorithm. The API follows a straight forward design and can be easily integrated on every Java based system.

[2] http://code.google.com/android/
[3] http://www.igniterealtime.org/projects/smack

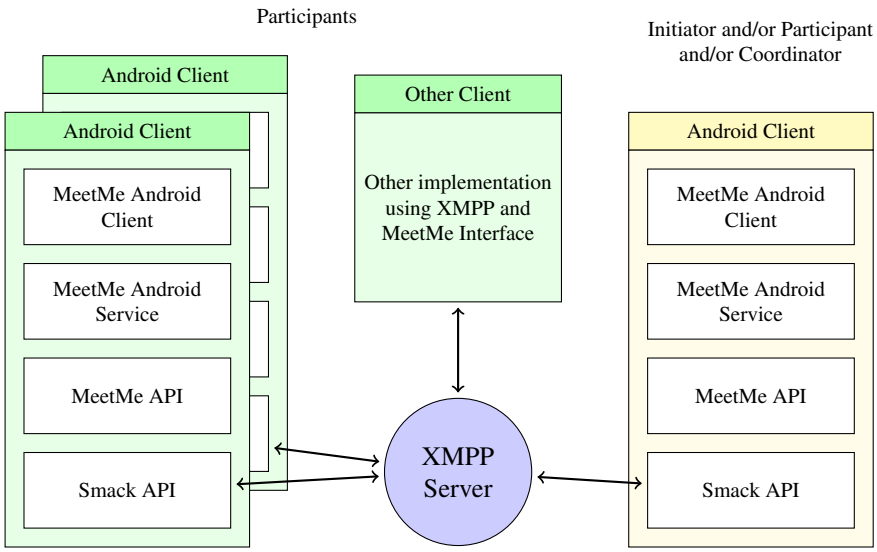

Figure 1. General Architecture

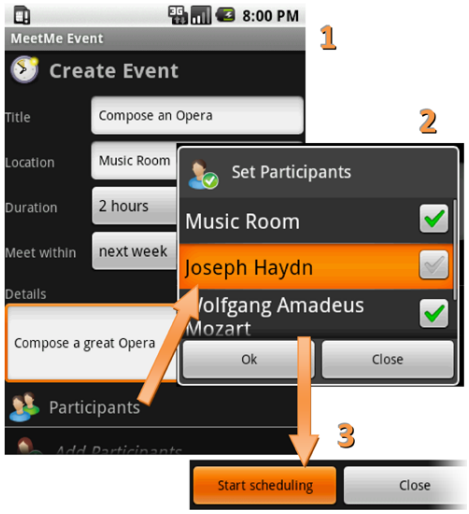

Figure 2. Adding a meeting to negotiate with the prototype implementation

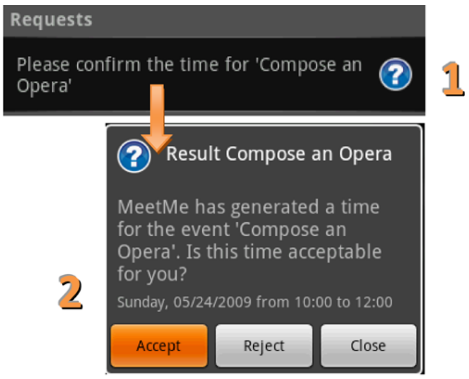

Figure 3. Result of Negotiation

- *Smack API* - Open Source XMPP client library for instant messaging and presence. To meet the requirements of Android, the Smack API had to be adapted.

The prototype of MeetMe was submitted to the Android Developer Challenge (ADC) in April 2008 and reached the top 25 percent of the 1,788 submissions. Figure 2 and 3 show screenshots from the prototype, adding a new meeting to negotiate and the result of the negotiation.

5 Conclusions and Further Work

In this work, we introduced the design and implementation of MeetMe, a distributed multi agent meeting scheduling system for mobile devices. The MeetMe agents support mobile people finding feasible times to meet, using their mobile devices. We investigated and described several current approaches and showed, that none of the analysed approaches are mainly designed to negotiate meetings on mobile devices.

As discussed in this paper, the current implementation of the MeetMe meeting scheduling system is matching the timetables of the participants with a basic transparency/weighted time slot algorithm. For now, only the weighted times from the participants calendars are integrated into the calculation and negotiation of the meeting

times. In the next step, more user specific data, available on the mobile devices, should augment the computation of the time slots.

Android, the platform the MeetMe prototype was developed for, includes localisation services, providing the current position of the device and its user. In one of the next steps, the position data will be integrated into the negotiation process to improve the result. If, by example, a participant is currently in India, it would be unlikely for him or her to follow a meeting in New York within the next 6 hours. Or if two agents, negotiating a meeting, know, that their users habitually play golf at the same court every Tuesday afternoon, it would be obvious to schedule the meeting at that time.

The current implementation is limited to receive timetables from and save new meetings to Google Calendar. Data from other calendar services could easily be integrated, MeetMe allows to integrate new connectors as plug-ins. We plan to develop interfaces for *CalDav* and *iCal* in the next step.

Messages for temporary unavailable agents are stored in a database on the sending agent until the receiver is available. For future development the MeetMe API could also be adopted to implement a permanently available coordination service, which could prevent deadlocks between alternately unavailable agents.

The more persons invited to participate an event, the more resource consuming for the coordinating agent to calculate and negotiate the meeting. As mobile devices are limited in resources the scheduling of events could be outsourced to a separate scheduling service, reachable as user in the XMPP network. The MeetMe protocol contains coordination messages to port the main task of the negotiation to an scheduling service. In a future research project this negotiation service will be implemented.

References

[1] P. M. Berry, C. Albright, E. Bowring, K. Conley, K. Nitz, J. P. Pearce, B. Peintner, S. Saadati, M. Tambe, T. Uribe, and N. Yorke-Smith. Conflict negotiation among personal calendar agents. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 1467–1468, New York, NY, USA, 2006. ACM.

[2] M. Brzozowski, K. Carattini, S. R. Klemmer, P. Mihelich, J. Hu, and A. Y. Ng. grouptime: preference based group scheduling. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 1047–1056, New York, NY, USA, 2006. ACM.

[3] F. Calefato, F. Lanubile, and M. Scalas. Porting a distributed meeting system to the eclipse communication framework. In *eclipse '07: Proceedings of the 2007 OOPSLA workshop on eclipse technology eXchange*, pages 46–49, New York, NY, USA, 2007. ACM.

[4] A. Chun, H. Wai, and R. Wong. Optimizing agent-based meeting scheduling through preference estimation. *Engineering Applications of Artificial Intelligence*, 16:727–743(17), October 2003.

[5] E. Crawford and M. Veloso. Opportunities for learning in multi-agent meeting scheduling. In *Proceedings of the Fall AAAI Symposium on Artificial Multiagent Learning*, Washington, DC, October 2004.

[6] E. Crawford and M. Veloso. Learning dynamic preferences in multi-agent meeting scheduling. In *IAT '05: Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 487–490, Washington, DC, USA, 2005. IEEE Computer Society.

[7] M. T. Gervasio, M. D. Moffitt, M. E. Pollack, J. M. Taylor, and T. E. Uribe. Active preference learning for personalized calendar scheduling assistance. In *IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces*, pages 90–97, New York, NY, USA, 2005. ACM Press.

[8] A. B. Hassine and T. B. Ho. An agent-based approach to solve dynamic meeting scheduling problems with preferences. *Eng. Appl. Artif. Intell.*, 20(6):857–873, 2007.

[9] IBM Corporation. Lotus notes. IBM Corporation, Armonk, NY, USA, 2005.

[10] Internet Engineering Task Force. Extensible messaging and presence protocol (xmpp): Core, October 2004.

[11] Microsoft Corporation. Outlook. Microsoft Corporation, One Microsoft Way, Redmond, WA 98052-7329, USA, 2007.

[12] E. Mynatt and J. Tullio. Inferring calendar event attendance. In *IUI '01: Proceedings of the 6th international conference on Intelligent user interfaces*, pages 121–128, New York, NY, USA, 2001. ACM Press.

[13] S. Sen and E. Durfee. On the design of an adaptive meeting scheduler. *Artificial Intelligence for Applications, 1994., Proceedings of the Tenth Conference on*, pages 40–46, Mar 1994.

[14] J. Tullio, J. Goecks, E. D. Mynatt, and D. H. Nguyen. Augmenting shared personal calendars. In *UIST '02: Proceedings of the 15th annual ACM symposium on User interface software and technology*, pages 11–20, New York, NY, USA, 2002. ACM.

[15] J. Wainer, J. Paulo Roberto Ferreira, and E. R. Constantino. Scheduling meetings through multi-agent negotiations. *Decis. Support Syst.*, 44(1):285–297, 2007.