Automatic Event Scheduling in Mobile Social Network Communities

Vaskar Raychoudhury, *Member, IEEE*, Ajay D. Kshemkalyani, *Senior Member, IEEE*, Daqing Zhang, and Jiannong Cao, *Senior Member, IEEE*

Abstract—Mobile social network (MoSoN) signifies an emerging area in the social computing research built on top of the mobile communications and wireless networking. It allows virtual community formation among like minded users to share data and to organize collaborative social activities at commonly agreed upon places and times. Such an activity scheduling in real-time is non-trivial as it requires tracing multiple users' profiles, preferences, and other spatio-temporal contexts, like location, and availability. Inherent conflicts among users regarding choices of places and time slots further complicates unanimous decision making. In this paper, we propose an autonomic system for activity scheduling in MoSoN communities. Our system allows flexible activity proposition while efficiently handling the user conflicts. As evident from our simulation and testbed results and analysis, our system can schedule multiple simultaneous activities in real-time while incurring low message and time cost.

Index Terms—Mobile social networks, collaborative event scheduling, virtual communities, conflict resolution

1 Introduction

Mobile social network (MoSoN) [1] is a hybrid of mobile communication and social networking technologies and provides various mobility and context-aware services. In a way, MoSoN aims to improve traditional social networking experience with the use of pervasive computing [2]. MoSoN users form virtual communities [3], [4], [5] for several purposes, such as, collaborative student environment, mobile music sharing, travel and tourism [6], and mobile business [7]. The virtual communities are groups of mobile users sharing the same social behaviour, interests, or goals. User communities may organize activities or events (we shall use these terms interchangeably) suited to their interests. One or more proposers start event scheduling at the same or different times. Examples of such community activities are hiking, dining out, partying, shopping, and going to a movie. But due to the dynamic and loosely-coupled nature of such communities, organizing a commonly agreed upon event is non-trivial. Below we discuss more of the challenges.

First, event scheduling requires collecting information regarding people's availabilities, time schedules, and preferences to join such an event. Collecting all the information manually and looking for common available time slots to schedule an activity with interested users while satisfying all user preferences is a daunting task. *Second*, since, there can be multiple simultaneous event proposals in a MoSoN community, a user may prefer an event proposal received later to an earlier one, and hence, may want to alter his earlier preferences. This may lead to cancellation of an event due to the want of minimum number of participants. Handling the complexity arising out of changing preferences later renders manual event scheduling more difficult. *Third*, event scheduling in MoSoN commonly faces the problem of collision of interests. Preferences of users vary from each other and sometimes it becomes hard to reconcile without proper means of conflict resolution. *Finally*, manual event scheduling, if possible, is time consuming. So, rescheduling of an event as a last moment measure (due to some urgency in the part of one or more prospective participants) or scheduling an altogether different event is extremely difficult.

To address the aforementioned challenges, and to facilitate event scheduling, we propose an intelligent agent based autonomic event scheduling system for MoSoN communities. Every user maintains an intelligent *User Agent (UA)*, which resides in the portable smart device (laptop, smart phone, tab) of the user and operates with minimal user intervention. It is knowledgeable about the user's profile and preferences and keeps track of the user's real-time context information, such as, his activity schedule (calendar) and current location. Armed with all these information, UAs of multiple users interact with each other to schedule a particular event against a proposal and inform other users about the final decision. In case a UA is unsure about the user's preferences, it can query the user and store his responses. However, user preferences may vary with time and situation.

Consider the following example scenario which illustrates the use of UAs in event scheduling in a MoSoN community. Raj is a first year undergraduate student in CSE

- *V. Raychoudhury is with the Department of Computer Science & Engineering, IIT Roorkee, Roorkee 247667, Uttarakhand, India. E-mail: vaskar@ieee.org.*
- *A.D. Kshemkalyani is with the Department of Computer Science, Univ. of Illinois at Chicago, IL 60607. E-mail: ajay@uic.edu.*
- *D. Zhang is with the RST Department, Institut Télécom Sudparis, Évry 91011, France. E-mail: Daqing.Zhang@it-sudparis.eu.*
- *J. Cao is with the Department of Computing, The Hong Kong Polytechnic University, Kowloon, HK. E-mail: csjcao@comp.polyu.edu.hk.*

Manuscript received 17 Aug. 2013; revised 21 Nov. 2013; accepted 15 Dec. 2013. Date of publication 15 Jan. 2014; date of current version 15 Oct. 2014. Recommended for acceptance by J. Lloret.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2013.2297111

department, IIT Roorkee. He joined the *IITR-CSE-2013* MoSoN community. Since he is new to north India, one Friday evening he thought of arranging a weekend sightseeing tour and launched an event scheduling through his UA within his community. His UA then communicates with the UAs of peer users to check their interest and availability for the proposed activity, place, and time. The UA finally returns a list of members willing to go for sightseeing with Raj at the proposed date, time, and place. However, another user Rahul of the same community prefers watching a movie to sightseeing in the weekend. So, after receiving Raj's proposal, he also launched a different event scheduling process in the same community for weekend movie going. Final objective for both the proposers is to maximize the number of the participants for their proposed activity.

Researchers have studied related problems like, group decision making, collaborative decision making, and group consensus, where a group of users are faced with the problem of deriving a decision that is in accord with the group's intention. Decision support for small groups has been studied by Fjermestad and Hiltz [8]. Later, decision-support systems for larger societal-scale groups have also received enough attention from researchers [9], [10], [11]. However, all such works have focused on offline learning over collected data using AI, fuzzy logic [12], [13], argumentation [14], machine learning based techniques, or optimization approaches and seem unfit to be directly applied to our problem. To the best of our knowledge, there is still no research focusing on event scheduling in MoSoN communities except our own [15]. In brief, this paper makes the following contributions:

- We propose the first (to the best of our knowledge) autonomic event scheduling system for MoSoN which works by collecting and analyzing profiles, preferences, and other spatio-temporal context information of social peers, without any user intervention.
- We propose four different parameter specification models with varied flexibility for the event proposer and they become instrumental in resolving conflicts between multiple users.
- We propose two distributed event scheduling algorithms and provide thorough analysis of time and message costs required. Our simulation and testbed results corroborate with our theoretical analysis and show that our message and time costs are significantly low.

2 Related Works

Applications developed over MoSoN are finding plenty of uses in our daily lives. Location based services, such as, Foursquare [16], Jiepang [17], and iPhone BreadCrumbs [6] are allowing people to socialize while on the move. Waze [18] allows social sharing between drivers on the highways. There are many media sharing services and other related MoSoN applications to carry out plenty of tasks. In short, MoSoN connects users through sharing of location, communication, proximity, activity, status, calendar, and many other contexts.

MoSoN can also be used to build up several dedicated social communities with different goals or objectives. Recently, the SOCIETIES project [19] identifies the needs of various MoSoN communities with several pervasive computing features, like context-awareness; data and resource sharing; service provisioning; learning, reasoning and predicting; decision-making and pro-activity; security and privacy. Examples of such mobile and pervasive environments where MoSoN communities will be useful consists of scenarios [19] like, researchers in a conference, students in a university, search and rescue workers during a disaster, social car-pooling between drivers and commuters of the same route, and patients and care-givers in a healthcare facility. Many of the aforementioned scenarios require organizing events among community members. However, there are no such research works to facilitate autonomic event scheduling for MoSoN users.

Researchers in MoSoN have hitherto focused mainly on building applications to cater to different user needs. Another prominent research area is building middleware platforms to facilitate mobile social application development. Some of the famous MoSoN middleware are SAMOA [20], MobiSoC [21], and MobiClique [22]. MobiSoC introduces a community concept which is an abstraction for aggregating users and physical places. They can maintain a complete social state of a community and can identify emergent community patterns by analyzing geo-social relationships between people and places. SAMOA, on the other hand, followed a user-centric approach. MobiClique is also mainly a user-centric MoSoN built over ad hoc social settings. However, MobiClique helps users to grow their social links opportunistically by connecting two previously unknown proximate users with pre-specified common preferences or interests. We are also leveraging the idea of MoSoN community of users with common interests and preferences.

Community detection [23], [24] is an important and crucial research problem in social networks which has been extended to community detection in MoSoN [25]. This ensures the existence of community behavior in mobile social environment. However, so far little attention has been paid to schedule group activity in mobile social communities. Earlier there were research carried out in the area of collaborative decision making and group consensus. As the name suggests, they focused on making decision favorable to the group members, in general, considering both small [8] and large [9], [10], [11] social groups. Usually, they work by developing an ideal solution based on pre-collected user's knowledge and by modeling user objective functions. Then they check existing solutions to find out the one closest to the ideal solution. Our approach, however, is more application-oriented and light-weight. Users use their mobile handheld devices to schedule an activity by collecting information and making a final decision in real-time and on-the-fly.

As already stated in Section 1, during autonomic decision making, conflicts can often arise between several information sources (e.g., profiles and preferences of multiple users) on which the decisions are based. To the best of our knowledge, no significant research work has been carried out to resolve such conflicts. This is mainly because, projects that

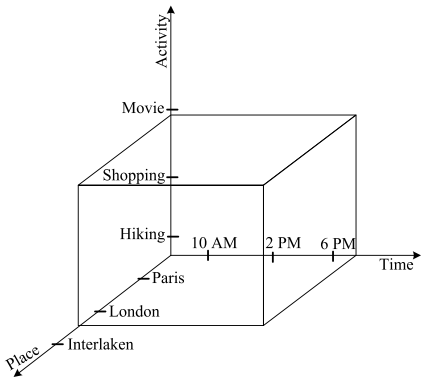

Fig. 1. Model M_1.

support adaptive behaviors, often make decisions based on a single information source. Projects such as Ubisec [26], [27] and Spice [28] make decisions based on individual user's profile (specifically the user's preference set) alone and hence multi-user conflicts do not occur. We have, however, handled user conflicts in group activity scheduling and provided viable solutions for conflict resolution.

3 System Model and Event Detection Techniques

In this section we describe our data structure and system models and provide a classification of different decision making techniques.

3.1 Data Structure

We assume that a pervasive MoSoN environment is composed of multiple mobile users equipped with smart devices (e.g., smartphones, tabs, etc.) connected wirelessly and they communicate through asynchronous message passing. Mobile users form a social community of like-minded users and schedule different activities and events through community-wide group decision making.

As mentioned in Section 1, a proposed event is actually a group activity like, shopping and hiking. We define an event (ξ) using a sextuple $(U_{id}, A, T, P, N, \Theta)$, where

- U_{id}: identifier of the user proposing the event (ξ),
- A: set {1 to α} of activities,
- T: set {1 to τ} of available time slots,
- P: set {1 to ρ} of places preferred by the user,
- N: minimum number of participants required to hold the event, and
- Θ: set {0 to 9} of integers representing user preference for a particular place and/or time slot.

We assume that the duration of an activity is implicitly defined by the activity. For example, Time to watch a movie is 2 hours, or hiking requires 6 hours, and so on. We consider that a single user can propose one or more events. Multiple users can also propose different events at the same time and all those events will be circulated across the community members in order to gather consensus in favor of them. Finally, the event with maximum number of participants (N) will be selected as the generally agreed upon one and the information will be circulated to the prospective participants. Our proposed system models use different

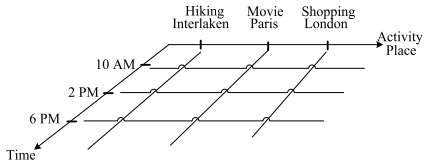

Fig. 2. Model M_2.

combinations of A, P, and T elements to account for different levels of flexibility in event proposing.

3.2 System Models

During launching an event, a user mainly considers the three major elements, Activity (A), Place (P), and Time (T) from the event sextuple along with their corresponding values (Θ). Considering inter-dependence of the A, P, and T elements, we propose the following four different system models for intelligent agent based event scheduling.

3.2.1 *Model* M_1

In this model, we consider A, P, and T elements of an event ξ are independent of each other. Thus, the same activity with different place-time combinations can receive different preference value (Θ) from the user. For the sake of convenience, we define M_1 as (A, P, T, Θ). Fig. 1 aptly illustrates M_1, where any point in the XYZ plane uniquely identifies an activity using different A, P, and T values. Preference values for A-P-T combinations are entered by users depending on different practical considerations. For example, movies in the weekdays can be watched only in the evening, or shopping in either Paris or London is better than that at Interlaken, and so on. M_1 provides the highest level of flexibility in selecting elements and that leads to the largest local space for M_1 as this: $|A| \times |P| \times |T| = \alpha \times \rho \times \tau$.

3.2.2 *Model* M_2

This model (Fig. 2) is a slightly more rigid compared to M_1 and here Place (P) is implicitly defined by activity (A), as AP. M_2 takes a single input for a combined Activity-Place (AP) and Time (T) element, where AP and T values are independent of each other. So, users can prefer (Movie-Paris, 6 PM) over (Movie-Paris, 2 PM) by entering suitable values. From Fig. 2, we can observe that, due to inherent linking up of A and P, the local space in M_2 is a subset of points on the plane and is bounded by: $|A \times P| \times |T| \cong |P| \times |T| = \rho \times \tau$, where $|A| \ll |P|$.

3.2.3 *Model* M_3

M_3 extends M_2 by further constraining the latter. It takes two input preferences from user for two independent elements, combined Activity-Place (AP) and Time (T), and they are represented with the tuples (AP, Θ_{AP}) and (T, Θ_T), respectively. Here, a user can specify a fixed value for a time slot, independent of activity. For example, a user can specify a fixed Θ_T value for some/any relaxing activity after working hours. Fig. 3 shows the data points from which a user needs to choose while using M_3. The local space for M_3 is bounded by: $|A \times P| + |T| \cong |P| + |T| = \rho + \tau$, where

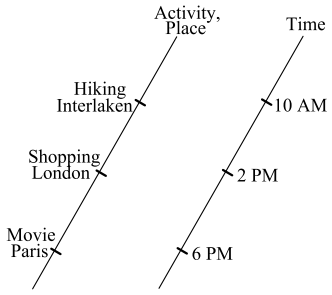

Fig. 3. Model M_3.

$|A| \ll |P|$. In M_2 and M_3, the distinction between places and activities has been removed:

3.2.4 Model M_4

M_4 in Fig. 4 allows users to input preferences separately and independently for Activities (A), Places (P), and Time slots (T) using the following three tuples: (A, Θ_A), (P, Θ_P), and (T, Θ_T). M_4 is restrictive (therefore weakened) as compared to M_1, as the user cannot input preferences over all combinations of A, P, and T variables. So, comparing Figs. 1 and 4, a user can specify only three different values instead of 27 as in M_1. So, the local space of M_4 is also less than that of M_1 and is represented as: $|A| + |P| + |T| = \alpha + \rho + \tau$.

4 Our Proposed Algorithms

In this section, we present our algorithms for autonomic event scheduling in MoSoN. As stated earlier, MoSoN operates in a distributed dynamic setting, in which a user does not know the preference values of A, T, and P of other users. The objective of our algorithms is to facilitate distributed coordination among socially connected peers to schedule a commonly agreed social activity, at an agreed place and time while trying to maximize the number of participants.

4.1 Assumptions and Variables

When a user (U) wants to schedule an activity, he can do so by launching an event (ξ). A user can even propose an event which is not in his preference list, as long as it is popular and social. When event (ξ) is launched, it invokes our social event scheduling algorithm for the MoSoN community of U.

A MoSoN community can be represented as an undirected graph $G = (V, E)$, where V is the set of users in the community and E is the set of social links between those users. Here, we assume that G is completely connected, so that, each user can directly communicate with every other user through asynchronous message passing. In order to save message and time costs, we propose to execute our algorithm using an overlay constructed over G.

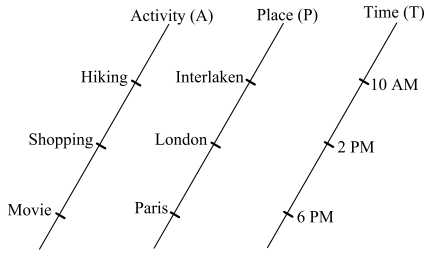

Fig. 4. Model M_4.

TABLE 1
Variables Used in Event Scheduling Algorithms

Variable	*Significance*
C_m	Mobile social community with m members
U	Id of user $u \in C_m$
$P_{list}[i]$	List of i places corresponding to the set P in the event (ξ) sextuple
$T_u[24]$	Time array of user u with size 24 X 1 (24-slots of 1-hour each). Each element is Boolean with values: 0 (user available) and 1 (user not available)
$LDM_u[i,j]$	Local data matrix at node u having i rows ($P_{list}.size$) and j columns ($T_u[24].size$)
P_A	Proposer of activity A, root of the binary tree overlay
$BH_A[m]$	Array representation of a binary tree overlay rooted at P_A
PAR_{uA}	Parent of node u in the overlay (for activity A), *null* for P_A
$Succ_{uA}$	Set of successors (two child nodes) of node u in the overlay (for activity A), *null* for the leaf nodes
RES_A	List of $P_i \in P_{list}$ and $T_j \in T_u$ for which consensus occurs
$FinRes_A$	Final result pair from RES_A

The overlay is basically a binary tree which considers the node ids as the key values. The overlay is a dynamic structure created on-the-fly when user, U launches event ξ, and it sets U as the root. Rest of the tree nodes are chosen randomly from the set V. The overlay formation precedes the event scheduling algorithm.

We further assume that there can be multiple event scheduling going on in the MoSoN community at the same time. For example, if a user U_2 receives the event ξ_1 proposed by U_1 and does not feel like participating, he can initiate a new event scheduling by launching a separate event ξ_2. In order to facilitate multiple parallel event scheduling, we allow co-existence of several overlays rooted at individual nodes in V. Every root node, i.e., an event proposer maintains an array representation of their individual binary tree overlay and shares it with the nodes in their overlay in the initial stages of our algorithm.

Connectivity among nodes in MoSoN can be through Internet. Thus, it hardly matters whether the users are within the same subnet or scattered over a larger area. Since, the Internet connectivity is considered as stable and continuous (through Wi-Fi or 3G), mobility patterns of the users are not of foremost importance as long as they are connected. Moreover, as the MoSoN users are forming interest-based groups, it is assumed that they are voluntarily available during an event scheduling. However, the users can suddenly leave the network due to unavailability of Internet connection or some other personal issues. In order to check whether the user is still connected in the overlay during high message transmission delays, we use a probe-reply mechanism.

In Table 1, we have listed the variables used for describing the pseudocode of our proposed algorithms. LDM is a data matrix created with columns and rows corresponding to sizes of time array, T_u [24] (or proposed time slots, for M_2), and place list, P_{list}, respectively.

4.2 Algorithm Description

In this section, we introduce our proposed algorithms for automatic event scheduling in MoSoN communities. We have developed one algorithm each based on M_2 and M_3

(Section 3.2). Our proposed algorithms operate in the following three phases:

- Phase 1: Form an overlay and disseminate ξ. Collect global knowledge about A, T, P of all users.
- Phase 2: evaluate most popular activity.
- Phase 3: distribute the result.

We shall initially describe the algorithm based on M_3 followed by the algorithm based on M_2:

4.2.1 Social Event Scheduling Algorithm (M_3)

Phase 1: A user, U, launches event ξ and communicates it through the overlay down to the leaf nodes along with the array representation of the binary tree overlay. The leaf nodes enter their preference values for places and send it to their parents in the tree. Thus, when the phase 1 concludes, every node knows their respective position in the overlay and the root node has the list (P_{list}) containing place preferences of all other members of his community.

Phase 2: After phase 1 ends, U sends the P_{list}, to the leaf nodes through the overlay. Each leaf node creates a GDM and enters their preferences (0-9) for respective places, where 0 and 9 represents the user's absolute disapproval and approval for a particular place. If the user's preference rating for the ith place in the P_{list} is '$\geq$' 5, then for all free time slots $(T[j] = 1)$ he increases the value of LDM $[i, j]$ by 1. After processing for each place in the P_{list} he then sends the processed LDM to his parent node in the overlay. The parent node waits to receive LDM matrix from both the children and merges them using matrix addition to generate the updated LDM. Then it adds its own preference rating to that one and forwards it to its parent until it reaches the root. When the root receives LDM from its children, it merges them and calls the new updated LDM as Global Data Matrix (GDM) (see Fig. 7). GDM $[i, j]$ physically signifies the numbers of people who rated the corresponding place $P_{list}[i]$, greater than or equal to 5 and are available for the activity at the time *T[j]*. To find the most popular place-time combination for the proposed activity, the root node searches for the MAX(GDM $[i, j]$) which might occur for multiple i, j values. In case of multiple MAX GDM values, the proposer will have the authority to choose a particular place and a particular time.

Phase 3: This phase involves distributing the results of the processing done in Phase 2 by the proposer to the users along with a commit deadline for the activity. The user after receiving the result has to commit to one proposer within the specified time, in case, multiple decision making algorithms are being executed in parallel. Intermediate nodes of the overlay help to pass any message originating either at the root or at a leaf node.

4.2.2 Social Event Scheduling Algorithm (M_2)

The algorithm for M_2 (Section 2, supplemental file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2013.2297111) is almost similar to M_3 except for a few changes. M_2 algorithm does not take the available time slot matrix (T_u) as input (see Table 5). Instead, the event launcher suggests some time slots and forwards it to the peer users. Other users may add other favourable time slots if they wish. Otherwise they just enter their preferences for the proposed time slots. For LDM processing while M_3 uses the time slot matrix, M_2 uses the preference ratings of a user against the proposed time slot(s) of the event launcher.

Phase I @ P_A

1. Propose activity A AND create $BH_A[m]$
2. Suggest a place P , add it to P_{list}
3. $\forall s \in Succ_{uA}$ Send (A, P_{list}, $BH_A[m]$) tuple to s
4. Wait to receive (A, P_{list}) from both child
5. Start *Phase II @ PA*

Phase I @ any non-root node u

6. receives (A, P_{list}, $BH_A[m]$) from P_A AND sets $PAR_{uA} \leftarrow P_A$
7. Find out the child nodes from $BH_A[m]$ and set to $Succ_{uA}$
8. **if** $|Succ_{uA}| \neq$ NULL,
9. $\forall s \in Succ_{uA}$ send (A, P_{list}, $BH_A[m]$) to s
10. node u waits to receive (A, P_{list}) from both child and merges the P_{list} (s)
11. Suggest alternate places (P) (optional) AND $P_{list} = P_{list} \cup P$
12. send (A, P_{list}) to PAR_{uA}

Phase II @ P_A

13. $\forall s \in Succ_{uA}$ Send (A, P_{list}) to s
14. Wait to receive (A, LDM_u) from both child
15. Invoke *LDM_processing ( )* AND rename LDM_{P_A} as *GDM*
16. Start *Phase III @ P_A*

Phase II @ any non-root node u

17. receives (A, P_{list}) from P_A
18. **if** $|Succ_{uA}| \neq$ NULL,
19. $\forall s \in Succ_{uA}$ send (A, P_{list}) to s
20. node u waits to receive (A, LDM_u) from both child
21. invoke *LDM_processing ( )*
22. send (A, LDM_u) *to* PAR_{uA}

LDM_processing ( )

23. **if** $|Succ_{uA}| =$ NULL Generate LDM_u of size ($|P_{list}| \times |T_u|$)
24. **else** for both the children m and n of u
25. $LDM_u[i, j] = LDM_m[i, j] + LDM_n[i, j]$, $\forall i \in [1, |P_{list}|]$ and $\forall j \in [1, |T_u|]$
26. $\forall p \in P_{list}$ provide preference rating (R_p) between *(0-9)*
27. $\forall p_i \in P_{list}$, *if* $R_{p_i} \geq 5$
28. $\forall t_j \in T_u$ if $t_j = 1$
29. LDM [i, j] = LDM [i, j]+1

Phase III @ P_A

30. MAX_{GDM} = MAX(GDM[i, j])
31. Add to RES_A all (i, j) pairs for which GDM[i, j] = MAX_{GDM}
32. Choose one (i, j) pair from RES_A randomly and store in $FinRes_A$
33. Disseminate the final result to all the nodes in the overlay

Fig. 5. Algorithm for the Model M_3.

4.2.3 Popularity-Based and Preference-Based Methods

The pseudo-codes presented in Fig. 5 show only the *popularity-based* place-time selection approach where we consider preferences '$\geq$5' as valid preferences (*Line 27, Fig. 5*) and any lower preference value as the user's unwillingness to take part in the activity.

We shall show in the following section using examples that there is also a *preference-sum based* approach where all input preferences for places and time slots are considered. For a particular place and time slot, preferences entered by all the users are summed up to find the highest total preference, and the corresponding place time combination is finally chosen for hosting the activity.

TABLE 2
User Place-Time Preferences

		Activity 1					Activity 2				
		P_1	P_2	P_3	P_4	P_5	P_1	P_2	P_3	P_4	P_5
User1	T_1	**8**	9	1	9	6	1	5	**0**	8	7
	T_2	2	6	6	1	1	1	8	6	7	6
	T_3	1	7	3	5	1	9	5	9	1	1
User2	T_1	**0**	2	5	9	9	9	4	**6**	8	5
	T_2	4	9	3	5	2	5	3	6	1	1
	T_3	6	2	6	6	7	2	3	4	3	6
User3	T_1	**1**	9	9	4	8	2	9	**0**	3	9
	T_2	7	2	5	6	8	0	9	9	1	4
	T_3	4	0	2	9	1	0	4	2	3	1
User4	T_1	**1**	4	9	7	9	3	3	**1**	9	1
	T_2	9	5	1	1	2	6	2	7	5	4
	T_3	8	5	9	0	4	5	2	1	0	2
User5	T_1	**6**	0	8	9	6	3	8	**4**	6	5
	T_2	8	2	8	2	9	2	7	8	7	4
	T_3	1	9	0	7	8	5	8	0	9	1

5 Example Applications

In this section, we show two example application execution walkthroughs, one for each M_2 and M_3. Each example application considers the case of two parallel activity scheduling. The two-activity case can be easily extended and applied for multi-activity scenarios. For each case, there is a popularity-based and a preference-sum based solution.

5.1 M_2 : Five Users and Two Activities

Consider a MoSoN with five users with node ids 1 to 5. We assume that node 1 and 3 launch event scheduling in this MoSoN at the same or different times. The details of the proposed event are represented with the following parameters. The broadcast tree below gives the organization of the nodes in the binary tree overlay with the event proposer (initiator) as the root.

Initiator: 1 0 1 0 0 [initiating user ids (randomly selected) are identified with 1, rest are 0].
Name of the activity: Movie (Activity1) & Sightseeing (Activity2).
Time Length of the activity: 3 and 6 hours.
Name of the places proposed for Movie: PVR (P_1), Adlabs (P_2), Inox (P_3), Fun Cinemas (P_4), Cinemax (P_5).
Name of the places proposed for Sightseeing: Delhi (P_1), Agra (P_2), Jaipur (P_3), Roorkee (P_4), Rishikesh (P_5).
Time slots proposed (for movies): 12 (T_1), 15 (T_2), 18 (T_3).
Time slots proposed (for Sightseeing): 11 (T_1), 14 (T_2), 17 (T_3).
Broadcast tree 1: 1 2 3 5 4.
Broadcast tree 2: 3 4 5 1 2
User Place-time preference: shown in the Table 2 (all values have been randomly generated).

5.1.1 Preference-Based GDM

For activity 1 in Table 2, if we add up the preference values entered by each user for the (P_1, T_1) tuple, it will give us $(8+0+1+1+6) = 16$. However, the addition is done from the leaf nodes towards the root following the overlay tree structure. The final GDM will look like the one in Table 3, where, the maximum value is 38 corresponding to tuples (P_4, T_1) and (P_5, T_1).

TABLE 3
GDM for Preference-Based M_2

	Activity 1					Activity 2				
	P_1	P_2	P_3	P_4	P_5	P_1	P_2	P_3	P_4	P_5
T_1	16	24	32	**38**	**38**	18	29	11	34	27
T_2	30	24	23	15	22	14	29	**36**	21	19
T_3	20	23	20	23	21	21	22	16	16	11

The root node can choose either tuple and declare it as the final result, which says that, the users can go for a movie at 12 noon either at the Fun Cinemas or at the Cinemax. Similarly, for activity 2, the maximum value is 36 corresponding to tuple (P_3, T_2), which says that, the users can go for sightseeing at 3 PM in Jaipur.

5.1.2 Popularity-Based GDMs

Searching for place popularity for activity 2 in Table 2, we can see that for the tuple (P_3, T_1) (highlighted with *bold* face) there is only one value (among all users) entered by User $2 (= 6)$ which is '≥ 5'. So, the final value in the GDM for this place will be $(0+1+0+0+0) = 1$. This value signifies that only one user prefers this place for the activity. The final GDM at the root node looks like the one in Table 4.

From Table 4, the final result for activity 1 is (P_5, T_1) and for activity 2, the final result is (P_3, T_2) and in both the cases, all five users are ready to join the activity. Presenting the results in more formal way, all the five users are ready to go for a movie at 12 noon in Cinemax, and/or for sightseeing in Jaipur at 3 PM.

Each of the preference and popularity-based methods for the current example uses 40 messages to finish decision making and 13 logical time units.

5.2 M_3: Six Users and Two Activities

Consider the same MoSoN as before, only with six members instead of five. The details of the proposed activity are represented with the following parameters:

Initiator: 1 0 1 0 0 0.
Name of the activity: Movie (*Activity1*) & Sightseeing (*Activity2*).
Time Length of the activity: 3 and 6 hours.
Name of the places (Movie): PVR (P_1), Adlabs (P_2), Inox (P_3), Fun Cinemas (P_4), Cinemax (P_5).
Name of the places (Sightseeing): Delhi (P_1), Agra (P_2), Jaipur (P_3), Roorkee (P_4), Rishikesh (P_5).
Broadcast tree 1 (for Movie): 1 3 6 2 4 5.
Broadcast tree 2 (for Sightseeing): 3 2 4 6 1 5.
Available Time slots Matrix for users ($T_u[24]$): Table 5.
Place preference matrix for users: Table 6.
(All values in Table 5 and Table 6 have been randomly generated).

TABLE 4
GDM for Popularity-Based M_2

	Activity 1					Activity 2				
	P_1	P_2	P_3	P_4	P_5	P_1	P_2	P_3	P_4	P_5
T_1	2	2	4	4	**5**	1	3	1	4	4
T_2	3	3	3	2	2	2	3	**5**	3	1
T_3	2	3	2	4	2	3	2	1	1	1

TABLE 5
Available Time Slots Matrix

User_Id	Time Slots (1-24)
U1	1 1 0 1 1 1 0 1 0 0 1 0 1 1 0 1 1 1 0 1 **1** 1 1 1
U2	0 0 1 0 0 0 1 0 1 1 0 1 0 1 0 0 0 1 0 1 **1** 0 0 0
U3	0 0 1 1 0 1 1 1 0 1 1 0 0 0 0 1 0 0 1 0 **1** 1 1 1
U4	0 1 1 1 1 0 0 0 1 0 0 1 0 1 0 1 0 0 1 1 **0** 0 0 0
U5	1 1 1 0 1 0 0 1 0 1 1 1 0 0 0 0 1 1 0 0 **1** 1 1 1
U6	0 0 0 1 1 1 0 0 0 0 1 0 0 1 1 0 1 1 1 0 **1** 1 1 0

TABLE 6
Place Preference Matrix

	Activity 1					*Activity 2*				
	P_1	P_2	P_3	P_4	P_5	P_1	P_2	P_3	P_4	P_5
U1	8	2	1	**7**	1	0	6	3	1	1
U2	0	0	7	**8**	2	4	6	5	8	3
U3	5	2	1	**3**	1	4	8	9	8	4
U4	4	7	9	**0**	5	8	7	4	5	7
U5	0	6	7	**9**	5	5	0	8	6	5
U6	4	5	9	**0**	7	6	0	1	3	0

5.2.1 Preference-Based GDM

LDM of a specific node (say node i) is prepared by taking the cross product of a column vector (input preferences of a user for all places) and a row vector (available time slots of the same user):

$$[\mathbf{LDM_{u_i}}] = [\mathbf{P_{u_i}}]^{\mathbf{T}} \times [\mathbf{T_{u_i}}]$$

$$\Rightarrow [82171]^T \times [1\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 1]$$

$\Rightarrow$ LDM for user 1 for activity 1 is as below in Fig. 6.

The final preference-based (or, preference-sum based) GDM is prepared following the overlay tree structure for the activity 1 is given in Fig. 7.

The final GDM in Fig. 9 shows that the highest value is 27 which is for the $(\mathrm{P}_4, \mathrm{T}_{21})$ element. This value is obtained by executing the following formula (1):

$$\sum_{i=1}^{6} U_i \cdot P_4 \times U_i \cdot T_{21} \tag{1}$$

where, $U_i \cdot P_4$ means preference of user U_i for place P_4, and $U_i \cdot T_{21}$means the availability of user U_i at time slot T_{21}. We have considered 6 users for this example.

Replacing respective values in formula (1), gives us the following: $(7^*1 + 8^*1 + 3^*1 + 0^*0 + 9^*1 + 0^*0) = 27$. So, the final result says that, the users can go for a movie at 9 PM (21:00 hrs) at the Fun Cinemas. Here, we assume that all users will agree to participate and to stay for the entire duration of the movie.

5.2.2 Popularity-Based GDM

In case of popularity-based GDM creation for M_3 user preference for a place is considered '1' only if the input preference value is '$\geq$ 5'. So, the LDM of user 1 (for activity 1) using the popularity-based model will look like the one in Fig. 10.

The highest value in the GDM for the popularity-based method (Fig. 11) is 3 which appears in many places. This signifies that at least three users are ready to participate in activity 1 at those particular places and time slots. One of the places where '3' appears in the GDM in Fig. 11 is the element at $(\mathrm{P}_4, \mathrm{T}_{21})$ and we got that using the following formula (2):

$$\sum_{i=1}^{6} (U_i \cdot P_4 == 1;\ if\ U_i \cdot P_4 \geq 5) \times U_i \cdot T_{21} \tag{2}$$

Similarly, we can calculate the GDM for activity 2 and find the final result(s) following the overlay tree at Fig. 8. The preference and popularity-based methods for the current example both use 50 messages to finish decision making and 16 logical time units.

1-1-0-1-1-1-0-1-0-0-1-0-1-1-0-1-1-1-0-1-1-1-1-1

8	8-8-0-8-8-8-0-8-0-0-8-0-8-8-0-8-8-8-8-0-8-8-8-8
2	2-2-0-2-2-2-0-2-0-0-2-0-2-2-0-2-2-0-2-2-2-0-2-2
1	1-1-0-1-1-1-0-1-0-0-1-0-1-1-0-1-1-1-0-1-1-1-1-1
7	7-7-0-7-7-7-0-7-0-0-7-0-7-7-0-7-7-7-0-7-7-7-7-7
1	1-1-0-1-1-1-0-1-0-0-1-0-1-1-0-1-1-1-0-1-1-1-1-1

Fig. 6. LDM for (Preference-based) Activity 1 (M_3).

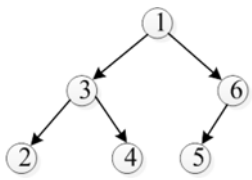

Fig. 7. Overlay Tree for Activity 1.

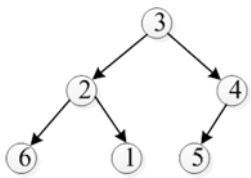

Fig. 8. Overlay Tree for Activity 2.

5.3 Analysis and Discussion

We can observe following facts from the above examples:

- The popularity-based method considers that if a user gives above average ($\geq$5) preference rating to a (Place, Time) combined metric (for M_2) or just Place metric (M_3), the user actually prefers the place and/or the time slot and will be serious in participating if the final event takes place in that place and/or time. This assumption though practical, leads to plenty of conflicts, as evident from the Table 4 and Fig. 11. In that case, one solution will be to allow users to participate in the same type of activity taking place at different place and/or time slots as chosen by different sub-groups. Another possible way to do away with the conflict is to use the preference-sum based method as discussed below.
- The preference-sum based method has a lower possibility of conflicts as evident in Table 3 and Fig. 9. The highest preference-sum is less likely to appear for many place and/or time elements. However, the drawback of the preference-sum based technique is

	T_1 to T_{24} →
P_1	08-12-09-21-16-17-05-13-04-05-17-04-08-16-04-17-12-12-13-12-17-17-17-13
P_2	08-15-15-16-20-09-02-10-07-08-15-13-02-14-05-11-13-13-14-09-15-15-15-10
P_3	08-17-24-20-26-11-08-09-16-15-18-23-01-26-09-11-17-24-19-17-25-18-18-09
P_4	16-16-20-10-16-10-11-19-08-20-19-17-07-15-00-10-16-24-03-15-**27**-19-19-19
P_5	06-11-13-14-18-09-03-07-07-08-14-12-01-15-07-07-13-15-13-08-16-14-14-07

Fig. 9. GDM for (Preference-based) Activity 1 (M_3).

that the highest preference-sum may be just contributed by very high preference ratings of only a handful of users and the voices of other users who have given very low rating to the same place and/or time element will get suppressed. But, for all practical purposes we have to accept the fact, that *for any community activity, there will always be some unsatisfied users*. Preference-sum based technique may reduce the number of unsatisfied users and can thus reach closer to resolving all conflicts.

- The preference and popularity-based methods for both M_2 and M_3 use same number of messages for the same number of events to schedule. This is mainly because of using the overlay structure. The overlay helps to reduce costs that could have occurred by simple message broadcasts.

6 Complexity Analysis

In this section we analyze the message and time complexity of our proposed algorithms. We assume that there are n users in a mobile social community. We further assume that the message transmission delay is negligible compared to the processing time at the nodes. We calculate the message and time complexity of our algorithms considering one phase at a time (except the last phase which is trivial). The binary tree overlay has height h with i levels, where i varies from 1 to h.

6.1 Phase I

6.1.1 Message Complexity (Bottom to Top)

Assuming each user (leaf node) proposes equal number of places on average, Places proposed per user $= K/n$, where K is the total number of places proposed.

So, the average size of a message packet generated at the leaf node is $c^*(K/n)$, where c is some constant.

Any internal node combines the place lists received from children nodes, and then adds its own place preferences before sending it further.

So, the average message sizes at an internal node at level i: $c^*(2^{h-i+1}-1)^*(K/n)$.

Summing up, total size of all messages sent from leaf to root $= \sum_{i=1}^{h} 2^i(2^{h-i+1}-1)^*c^*(K/n)$, where, $2^i =$ number of nodes at level i:

	1-1-0-1-1-1-0-1-0-0-1-0-1-1-0-1-1-1-0-1-1-1-1-1
8 (≥ 5)	1-1-0-1-1-1-0-1-0-0-1-0-1-1-0-1-1-1-0-1-1-1-1-1
2	0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0
1	0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0
7 (≥ 5)	1-1-0-1-1-1-0-1-0-0-1-0-1-1-0-1-1-1-0-1-1-1-1-1
1	0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0

Fig. 10. LDM for (Popularity-based) Activity 1 (M_3).

	T_1 to T_{24} →
P_1	1-1-1-2-1-2-1-2-0-1-2-0-1-1-0-2-1-1-1-1-2-2-2-2
P_2	1-2-2-2-3-1-0-1-1-1-2-2-0-2-1-1-2-2-2-1-2-2-2-1
P_3	1-2-3-2-3-1-1-1-2-2-2-3-0-3-1-1-2-3-2-2-3-2-2-1
P_4	2-2-2-1-2-1-1-2-1-2-2-2-1-2-0-1-2-3-0-2-**3**-2-2-2
P_5	1-2-2-2-3-1-0-1-1-1-2-2-0-2-1-1-2-2-2-1-2-2-2-1

Fig. 11. GDM for (Popularity-based) Activity 1 (M_3).

$\Rightarrow (2^{h+1} * h - (2^{h+1}-2))^* c^*(K/n).$

$\Rightarrow (2^{h+1}(h-1)+2)^* c^*(K/n).$

Therefore, the total message complexity:

$$\left[c^*(2^{h+1}(h-1)+2)^*\frac{K}{n}\right] + \left[c^*\left(2^{h+1}-2\right)^* n\right]$$
$$= \boldsymbol{O(K\ log_2\ n + n^2).}$$

6.1.2 Time Complexity (Top to Bottom)

When the message is coming from top to bottom, every node needs to record its parent and successor nodes from the array representing the binary tree, which takes constant time, c_1. So, the time complexity from top to bottom is h^*c_1.

6.1.3 Time Complexity (Bottom to Top)

Assuming T_p is the logical time unit to propose a place and it includes the constant time taken to combine places, the time taken to propose K/n places per leaf node is $(K/n)^*T_p$.

Since, internal nodes combine place list received from their children, the time taken will be proportional to the size of the place list received. So, for a node at level i, time taken to combine the place lists is $(2^{h-i+1}-2)^*(K/n)^*T_p$.

Adding the time taken to propose a place, time of processing at a node of level $i = (2^{h-i+1}-1)^*(K/n)^*T_p$.

Summing up time taken at all levels,

$$\sum_{i=1}^{h}\left(2^{h-i+1}-1\right)^*\frac{K^*}{n}\ T_p$$
$$\Rightarrow \frac{K}{n}{}^*T_p^*\left(2^{h+2}-h-3\right).$$

Summing up time complexity in both top to bottom and bottom to top cases, total time complexity is:

$$\frac{K}{n}{}^*T_{\mathrm{p}}{}^*\left(2^{h+2}-h-3\right)+h^*c_1 = \mathrm{O}\left(K^*T_p + log_2 n\right)$$
$$= \mathbf{O}(\boldsymbol{K + log_2\ n}),$$

assuming T_p is constant.

6.2 Phase II

6.2.1 Message Complexity (Top to Bottom)

Message in phase 2 primarily consists of exchanging the LDM matrix of size $= K^*T$ (place-time matrix).

The root node sends the place list to leaf nodes through every internal node in the overlay. So, the message size is K^*c_2.

Top-to-bottom message complexity $= n^*K^*c_2$ (one message is sent by each node, so total number of messages sent is $O(n)$).

TABLE 7
Simulation Parameters

Parameters	Values
Number of users, (n)	*10, 15, 20, 25*
Number of simultaneous activity proposals	*1, 2*
Number of places proposed for each activity	*5*

6.2.2 Message Complexity (Bottom to Top)

Each node sends a message of size K^*T to its parent. So, each node receives a message of size K^*T. Total message complexity $= n^*K^*T + n^*K^*c_2 = \boldsymbol{O(n^*K^*T)}$.

6.2.3 Time Complexity (Top to Bottom)

Since, we assume that there is no delay in message exchange, the time spent is just for processing at a node. The nodes just need to send the place list of size K to their children. Assuming constant time (say, c_3) for this job, the time complexity is h^*c_3, where h is height of the tree.

6.2.4 Time Complexity (Bottom to Top)

Each node rates the places. Assume time taken to rate a place is T_r, the time spent by each node in rating the places is K^*T_r. Time to process the LDM is $K^*T^*c_4$, where, c_4 is some constant.

Also each internal node combines the LDM received from both children. So, the time taken for this is $c_5^*K^*T$ where c_5 is some constant.

Total time taken at a node at level i is $K^*T_r + K^*T^*c_4 + K^*T^*c_5 = K^*T_r + K^*T^*c_6$. Summing up for all nodes,

$$\sum_{i=0}^{h}(K^*T_r + K^*T^*c_6) = h^*(K^*T_r + K^*T^*c_6).$$

Total time complexity is: $h^*K^*(T_r + T^*c_6) + h^*c_3$

$\Rightarrow \mathrm{O}(log_2 n^*K^*(T_r + T))$.
$\Rightarrow \boldsymbol{O(log_2 n^*K^*T)}$, *assuming* T_r *is constant*.

In the following section we shall present our simulation results and show that they corroborate with the complexity analysis just presented.

7 Performance Evaluation

We have carried out extensive simulations to evaluate the performance of our algorithms. Both of our algorithms (M_2 and M_3) have been simulated and compared based on proposed performance metrics. We also developed a prototype MoSoN system and evaluated the M_2 algorithm.

7.1 Simulation Setup and Metrics

The simulation system consists of two modules: the network overlay and the social decision making algorithm. The main parameters of the simulations are shown in Table 7. We consider a maximum of 25 users forming a MoSoN. Since, the MoSoNs are not very large conglomerations; the number of users considered can ideally model most of the usual cases.

We have simulated multiple parallel event scheduling where multiple decision making initiatives are ongoing parallely within the same MoSoN community. So, we have total eight combinations depending on M_2/M_3 algorithm for popularity-based/preference-based 1-activity/2-activity cases. User preference values in all experiments have been randomly generated.

We consider that the user devices in a MoSoN form a completely connected graph. As stated earlier, in order to keep a check on the number of message exchanged between users for decision making, we proposed to develop a tree-structured (binary tree) network overlay. Our algorithms are implemented as applications running on top of the network overlay. Each user device houses a software agent which accepts user input preferences (for activity, places, and time slots) and other inputs and communicates with peer agents through message exchange.

In the simulations, we measure the performance of all the algorithms using the following metrics:

- *Total decision-making time* (DMT_{tot}) or simply, *Time* is defined as the mean time elapsed between the instant at which a node launches an event scheduling process and the instant at which it knows the identity of the nodes participating in the proposed activity. Less time to reach a decision is a measure of how efficient the algorithm is.
- *Message overhead* (M_{tot}) or simply, *Message* is defined as the total number of messages exchanged among all the user nodes in order to make a final decision. The less is the message overhead, the more is the saving in resource for a mobile node.

7.2 Simulation Results and Analysis

Below we present our simulation results with analysis. We have simulated different versions of our algorithms using C++ and labeled them as *M_1_Popu_MII* (message cost in popularity-based single activity M_2), *T_2_Pref_ MIII* (time cost in preference-based dual activity M_3), and others, like, *M_2_Popu_MII*, *M_1_Pref_MII*, *M_2_Pref_ MII*, *T_1_Popu_-MIII*, *T_2_Popu_MII*, and *T_1_Pref_ MIII*.

Each simulation is triggered by a single user (for '1Act') launching a new event and forwarding to neighbor nodes in the overlay. For a multi-activity scenario, any user not satisfied by the activity proposal s/he received can propose an alternate activity which will trigger a different simultaneous activity scheduling.

Each value in the simulation result is obtained by averaging over 10 different runs. We also tested for three, five, six, and 30 users, but since, the results are similar, we have just omitted them. Here we present the general performance comparison between the eight different cases of our algorithm in Figs. 12 and 13. More in-depth comparisons between popularity-based and preference-based M_2 and M_3 algorithms are included in the supplemental file, available online.

Message complexity: The total number of messages increases linearly with the number of users, i.e., $\mathrm{O}(n)$, consistent with the plot results (Fig. 12). Since, the size of each message is $\mathrm{O}(n)$, the total message complexity turns out to be $\mathrm{O}(n^2)(\tilde{}(Klog_2 n+N^2))$ as calculated in Section 6.

Time complexity: In phase 1, the time complexity as calculated theoretically comes out to be $\mathrm{O}\ (K + log_2 n)$. Now in our tests, K is almost constant, so the results of the testing

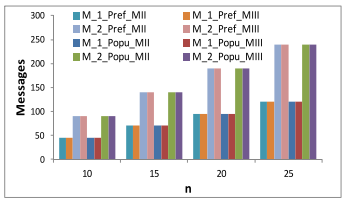

Fig. 12. Message Cost versus n.

must show a complexity of $O(log_2 n)$ which can be seen clearly in the plots (Fig. 13). In phase 2, the logical time complexity comes out to be $O(K^*T^*log_2 n)$. So, ultimately, the nature of graph is $O(log_2 n)$, assuming K^*T constant, i.e., sub-linear which is consistent with the plot results (Fig. 13).

7.3 Prototype MoSoN Structure and Description

We developed a prototype MoSoN system using seven students of CSE, IIT Roorkee and implemented the M_2 algorithm over it. We consider social communities of different sizes varying from three to seven users who are given seven Android smartphones (two Samsung Galaxy S3, three Samsung Galaxy S4 (mini), and two Samsung Galaxy Grand) which are interconnected in P2P manner using Samsung Chord API [29]. The API allows real-time content sharing between multiple devices in a dynamic P2P network. We assumed that all the users are connected through the campus Wi-Fi.

Before the actual event scheduling, a binary tree overlay is formed with the event proposer as the root of the tree. For the sake of simplicity, the prototype system assumes that only the root node proposes an event along with places and time slots. Participants can only accept or reject the event proposal. Participants can join or leave the community at any moment and the absence of a participant is detected through a probe-response mechanism.

We have measured the time taken for event scheduling in our prototype system. The time varies with the community size and the capacity of the smartphone acting as the root. We tried two different smartphones as root nodes— Galaxy S3 and Galaxy S4 (mini) and the performance using Galaxy S3 is superior to the other (Fig. 14). However, we must insist on the fact that the event scheduling delay also depends on the human factor, as the user needs to enter his response quickly to reduce the total delay.

8 Conclusion and Future Works

In this paper, we proposed a novel and flexible system model and two algorithms for real-time autonomic event scheduling in MoSoN communities. We considered mainly the place and time slots which are agreed upon by all the participants and the objective was to maximize the participants in a particular social activity. Our in-depth analysis aided by simulation and test results shows that our algorithms are useful and practical and incur low cost. In future, we want to investigate cross-community event scheduling and other challenging issues. We would like to make the system more intelligent, so that, it can proactively inform a member about other members' current activities.

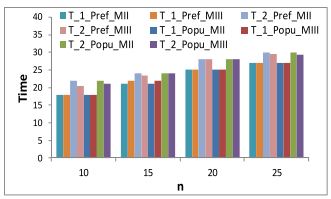

Fig. 13. Time Cost versus n.

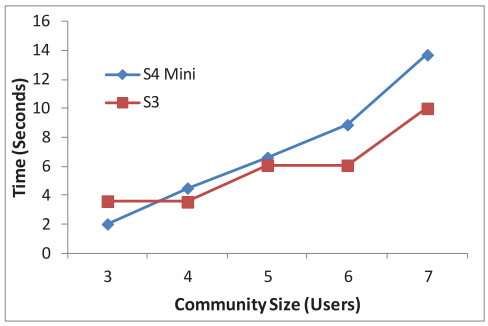

Fig. 14. Event scheduling time in prototype system.

Acknowledgments

This work was partially supported by the following grants: MHRD (GoI) FIG (A) 100579-ECD, DST (GoI) SB/FTP/ ETA-23/2013, and EU FP7 SOCIETIES. The authors thank the following students (CSE-IITR) for their support: Siddharth Maheshwari, Mohit Bakshi, Kanik Gupta, Vishal Mittal, Sandeep Singh Sandha, Smriti Vashisth, Sneha Padgalwar Rohan Prasad, Ramanpreet Singh, and Astuti Sharma.

References

[1] N. Kayastha, D. Niyato, P. Wang, and E. Hossain, "Applications, Architectures, and Protocol Design Issues for Mobile Social Networks: A Survey," *Proc. IEEE*, vol. 99, no. 12, pp. 2130-2158, Dec. 2011.

[2] P. Bellavista, R. Montanari, and S.K. Das, "Mobile Social Networking Middleware: A Survey," *Pervasive Mobile Comput.*, vol. 9, no. 4, pp. 437-453, Aug. 2013.

[3] T. Nguyen, S.W. Loke, T. Torabi, and H. Lu, "PlaceComm: A Framework for Context-Aware Applications in Place-Based Virtual Communities," *IOS J. Ambient Intell. Smart Environments*, vol. 3, no. 1, pp. 51-64, 2011.

[4] C. El Morr and J. Kawash, "Mobile Virtual Communities Research: A Synthesis of Current Trends and a Look at Future Perspectives," *Inderscience Int. J. Web Based Communities*, vol. 3, no. 4, pp. 386-403, 2007.

[5] D. Zhang, Z. Wang, B. Guo, X. Zhou, and V. Raychoudhury, "A Dynamic Community Creation Mechanism in Opportunistic Mobile Social Networks," *Proc. IEEE 3rd Int. Conf. Social Computing and IEEE 3rd Int. Conf. Privacy, Security, Risk and Trust (SocialCom/PASSAT)*, pp. 509-511, 2011.

[6] http://www.iphonebreadcrumbs.com/, 2014.

[7] J. Subercaze, P. Maret, J. Calmet, and P. Pawar, "A Service Oriented Framework for Mobile Business Virtual Communities," *Proc. Pervasive Collaborative Networks - IFIP TC 5 WG 5.5 Ninth Working Conf. Virtual Enterprises*, pp. 493-500, 2008.

[8] J. Fjemestad and S. Hiltz, "Experimental Studies of Group Decision Support Systems: An Assessment of Variables Studied and Methodology," *Proc. 30th Hawaii Int. Conf. Syst. Sci. Information Systems Track-Collaboration Systems and Technology*, 1997.

[9] M. Turoff, S. Hiltz, H. Cho, Z. Li, and Y. Wang, "Social Decision Support Systems (SDSS)," *Proc. 35th Hawaii Int'l Conf. Systems Science*, 2002.

[10] M.A. Rodriguez and D.J. Steinbock, "A Social Network for Societal-Scale Decision-Making Systems," *Proc. North Am. Assoc. for Computational Social and Organizational Science Conf.*, 2004.

[11] K. Nikos, D. Papadias, and C. Pappis, "Computer-Mediated Collaborative Decision-Making: Theoretical and Implementation Issues,," *Proc. 32 nd Hawaii Int. Conf. Systems Science*, 1999.
[12] F. Herrera, E. Herrera-Viedma, and J.L. verdegay, "A Model of Consensus in Group Decision Making Under Linguistic Assessments Fuzzy Sets and Systems," vol. 78, no. 1, pp. 73-87, Feb. 1996.
[13] N. Karacapilidis and C. Pappis, "Computer-Supported Collaborative Argumentation and Fuzzy Similarity Measures in Multiple Criteria Decision Making," *Computers Operations Res.*, vol. 27, nos. 7/8, pp. 653-671, June 2000.
[14] N. Karacapilidisa and D. Papadias, "Computer Supported Argumentation and Collaborative Decision Making: The HERMES System," *Inf. Syst.*, vol. 26, no. 4, pp. 259-277, June 2001.
[15] V. Raychoudhury et al., "Automatic Event Scheduling in Mobile Social Network Communities," *Proc. IEEE Int'l Conf. Social Computing (Socialcom)*, 2013.
[16] [Online] Available: https://foursquare.com/, 2013.
[17] http://hk.jiepang.com/, 2014.
[18] http://www.waze.com/, 2014.
[19] [Online] Available: http://www.ict-societies.eu/files/2011/11/SOCIETIES_D22.pdf, 2013.
[20] D. Bottazzi, R. Montanari, and A. Toninelli, "Context-Aware Middleware for Anytime, Anywhere Social Networks," *IEEE Intell. Syst.*, vol. 22, no. 5, pp. 23-32, Sept./Oct. 2007.
[21] A. Gupta, A. Kalra, D. Boston, and C. Borcea, "MobiSoC: A Middleware for Mobile Social Computing Applications," *Mobile Netw. Appl. J.*, vol. 14, no. 1, pp. 35-52, 2009.
[22] A. Pietilainen, E. Oliver, J. LeBrun, G. Varghese, and C. Diot, "MobiClique: Middleware for Mobile Social Networking," *Proc. Second ACM Workshop Online Social Netw. (WOSN)*, pp. 49-54, 2009.
[23] S. Fortunato, "Community Detection in Graphs," *Phys. Rep.*, vol. 486, no. 3, pp. 75-174, 2010.
[24] M. Plantié and M. Crampes, "Survey on Social Community Detection," *Social Media Retrieval*, pp. 65-85, Springer, 2013.
[25] P. Hui, Pan, J. Crowcroft, and E. Yoneki, "Bubble Rap: Social-Based Forwarding in Delay Tolerant Networks," *Proc. ACM MobiHoc*, pp. 241-250, 2008.
[26] Ubisec STREP Project, 6th Framework Programme, Homepage URL: http://www.clab.de/en/researchprojects/completed-research-projects/ubisec/index.html, Jan. 2004-Feb. 2006.
[27] J. Groppe and W. Mueller, "Profile Management Technology for Smart Customizations in Private Home Applications," *Proc. 16th Int. Workshop Database Expert Syst. Appl. (DEXA)*, pp. 226-230, 2005.
[28] C. Cordier, F. Carrez, H. Van Kranenburg, C. Licciardi, J. Van der Meer, A. Spedalieri, and J.P.L. Rouzic, "Addressing the Challenges of Beyond 3G Service Delivery: The SPICE Service Platform," *Proc. Workshop Appl. Services in Wireless Netw. (ASWN '06)*, 2006.
[29] http://developer.samsung.com/chord, 2014.

Vaskar Raychoudhury received the BTech degree in information technology from the B.P. Poddar Institute of Management & Technology, Kolkata (*affiliated to The University of Kalyani, West Bengal*), in 2003, and the MS degree in information technology from the School of Information Technology, Indian Institute of Technology, Kharagpur in 2006. He received the PhD degree in computer science from The Hong Kong Polytechnic University in 2010. Later he continued his postdoctoral research in the same university for a year before moving to the Institut Telecom SudParis, in France where he worked for another year as a postdoctoral research fellow. He is currently an assistant professor in the Department of Computer Science and Engineering, Indian Institute of Technology Roorkee. His research interests include mobile and pervasive computing, context-awareness, and (mobile) social networks, and he keeps publishing high-quality journals and conference papers in these areas. He has served as a program committee member in ASE/IEEE Socialcom, ICDCN, and many others. He serves in the capacity of the reviewer for many top IEEE and Elsevier journals. He is a member of the ACM, the IEEE, the IEEE Computer Society, and the IEEE Communication Society.

Ajay D. Kshemkalyani received the BTech degree in computer science and engineering from the Indian Institute of Technology, Bombay, in 1987, and the MS and PhD degrees in computer and information science from the Ohio State University in 1988 and 1991, respectively. He spent six years at IBM Research Triangle Park working on various aspects of computer networks, before joining academia. He is currently a professor in the Department of Computer Science at the University of Illinois at Chicago. His research interests include distributed computing, distributed algorithms, computer networks, and concurrent systems, and he has published extensively in top-quality journals and conferences in these areas. In 1999, he received the National Science Foundation Career Award. He has served in various positions (such as a general chair, a program co-chair, a steering committee member, or a program committee member) for international conferences such as IEEE ICDCS, IEEE SRDS, ACM PODC, and ICDCN. He served on the editorial board of the Elsevier journal, computer networks and is currently on the editorial board of the *IEEE Transactions on Parallel and Distributed Systems*. He has coauthored a book entitled *Distributed Computing: Principles, Algorithms, and Systems* (Cambridge University Press, 2008). He is a distinguished scientist of the ACM and a senior member of the IEEE.

Daqing Zhang received the PhD degree from the University of Rome La Sapienza and the University of L'Aquila, Italy in 1996. His research interests include mobile social networking, large-scale data mining, urban computing, context-aware computing, and ambient assistive living. He is a professor at Institut Mines-Télécom/Télécom SudParis, France. He has published more than 180 referred journal and conference papers, all his research has been motivated by practical applications in digital cities, mobile social networks, and elderly care. He is the associate editor for four journals including *ACM Transactions on Intelligent Systems and Technology*. He has been a frequent invited speaker in various international events on ubiquitous computing. He is the winner of the Ten Years CoMoRea Impact Paper Award at IEEE PerCom 2013, the Best Paper Award at IEEE UIC 2012, and the Best Paper Runner Up Award at Mobiquitous 2011.

Jiannong Cao received the BSc degree in computer science from Nanjing University, Nanjing, China, and the MSc and the PhD degrees in computer science from Washington State University, Pullman, WA. He is currently a chair professor and the head of the Department of Computing at Hong Kong Polytechnic University, Hung Hom, Hong Kong. He is also the director of the Internet and Mobile Computing Lab in the department. His research interests include parallel and distributed computing, computer networks, mobile and pervasive computing, fault tolerance, and middleware. He has coauthored a book in mobile computing, coedited nine books, and published over 300 papers in major international journals and conference proceedings. He has directed and participated in numerous research and development projects and, as a principal investigator, obtained over HK$25 million grants from government funding agencies and industries. He has won numerous prizes and awards, and is very active in professional activities locally and internationally. He was the coordinator in Asia and now the chair of the Technical Committee on Distributed Computing of IEEE Computer Society. He has served as an associate editor and a member of the editorial boards of many international journals, including the *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, *IEEE Networks, Pervasive and Mobile Computing Peer-to-Peer Networking and Applications*, and *Journal of Computer Science and Technology*. He has also served as a chair and a member of organizing / program committees for many international conferences, including PERCOM, INFOCOM, ICDCS, IPDPS, ICPP, RTSS, DSN, ICNP, SRDS, MASS, PRDC, ICC, GLOBECOM, and WCNC. He is a senior member of China Computer Federation, a senior member of the IEEE, and a member of the ACM.