Trustworthy Agent-Based Recommender System in a Mobile P2P Environment

Nabil Sahli[1], Gabriele Lenzini[2], and Henk Eertink[2]

[1] Dhofar University, Salalah, Sultanate of Oman
[2] Novay, P.O. Box 589, 7500 AN Enschede, The Netherlands
{gabriele.lenzini,henk.eertink}@novay.nl

Abstract. Current major P2P systems focus on PCs and do not provide services for the mobile environment. Compared to traditional P2P, characteristics of Mobile P2P include unreliable connections, limited bandwidth and constraints of mobile devices. In addition, nomadic users demand applications and services that are context-aware, personalised, secure, and trustworthy. Recommender systems are one of these applications. In this paper, we aim at building a mobile P2P recommender system which dramatically reduces wireless traffic between peers, brings trustworthiness (each peer can choose to rely on opinions of peers whom he trusts), and ffers unobtrusiveness (the target system is mainly autonomous and requires a minimum user intervention). Our solution is based on multi-agent systems and is illustrated on a slow-food restaurant recommender system.

1 Introduction

There is no doubt that mobile computing is increasingly important especially after the explosion of smartphones (networked PDA). While many distributed applications and services (e.g., file sharing) are still not appropriate for mobile devices because they require large hardware capabilities, others are more suitable. For instance, sharing opinions (e.g., recommendations) is an application which does not need a lot of resources (storage) and thus is a more appropriate mobile service. A mobile recommender system would certainly have many advantages over those which already exist on PC. Indeed, besides the ubiquitous access (members have an anytime-anyplace connection) advantage, the service can also be location based. For example, a mobile recommender system for restaurants would suggest good restaurants according to the current location of the user. The user could also rate a restaurant by geotagging[1], which frees him from entering the location information about the restaurant. Most successful recommender systems implemented so far are based on a client/server architecture where there is one central service provider and many users. In this architecture, the system collects the users' ratings (or opinions) and then applies an algorithm on the stored data to derive recommendations. However, when the trustworthiness

[1] *"The process of adding geographical identification metadata to media"* (Wikipedia).

D. Beneventano et al. (Eds.): AP2PC 2008/2009, LNAI 6573, pp. 59–70, 2012.

and the personalisation of these recommendations is an issue of great concern, a decentralised approach might be more appropriate. Unlike a centralised recommender (based on collaborative filtering) which relies on all peers' ratings, we suggest a decentralised recommender system which also considers ratings of peers on which a given user trusts.

Compared to traditional server/client technology on the Internet, the peer-to-peer (P2P) technology has capabilities to realise highly scalable, extensible and efficient decentralised recommender systems. However, current major P2P systems such as Gnutella and Napster, focus on PCs and do not provide services for the mobile environment. In fact, compared to traditional P2P, characteristics of Mobile P2P (MP2P) include unreliable connection, limited bandwidth and constraints of mobile devices (smaller screen, limited interface capabilities, restricted computational power). In addition, nomadic users demand applications and services that are context-aware, personalised, secure, and trustworthy. Most of current MP2P architectures are still suggesting a direct wireless communication between peers, which is demanding in terms of bandwidth and then communication reliability and cost.

The main idea of our work is to assign two agents to each mobile user: an *Embedded-Agent* which resides in the mobile device and captures the user's context and ratings, and a *Delegate-Agent* which resides in an open multi-agents architecture where it meets other peers. Using the context, the profile, and the experience (transmitted by the *Embedded-Agent*) of the user, *Delegate-Agent* maintains a personal community of trustworthy recommenders and a list of rated items. Consequently, we ensure more unobtrusiveness, trustworthiness, and context-awareness. Since all *Delegate-Agents* interact in a meeting space, wireless traffic between peers is dramatically reduced.

The rest of the paper is organised as follows. Section 2 presents a brief background on recommender systems (centralised vs. decentralised and content-based vs. collaborative filtering approaches). Section 3 describes our proposed architecture and its fundamental concepts. Section 4 is dedicated to the scenario we have implemented to recommend slow-food restaurants to a mobile community. Section 5 compares our approach to related works. Finally, Section 6 summarises our contributions and presents our future works.

2 Background on Recommender Systems

Two common ways of determining trust among peers are through using policies or reputation. Policies frequently involve the exchange or verification of *credentials*, which are information issued by one peer, and may describe qualities or features of another peer. Reputation (addressed here) is an assessment of a peer based on the history of interactions or observations, either directly (personal experience) or as reported by others (recommendations or third party verification). Various reputation-based systems have been proposed in different P2P systems. Two main approaches are used: centralised or decentralised.

In a centralised approach, observations about peers are reported and then stored in a central database. The reputation system (usually the central database

itself) uses these data to calculate the reputation of each peer. This approach is used in the reputation systems of eBay and Amazon. This centralised approach is not compatible with the design philosophy of P2P systems. We thus address a decentralised approach which can offer the three following desirable features: (i) no central authority (individuals are not dependent on a global rating system), (ii) personalised reputation, which means that the reputation of a particular peer is not built upon the opinions of the whole community but rather of a group of selected peers, and (iii) peers' preferences and profiles are taken into account.

There has been research about moving recommender systems toward a distributed architecture while using agents (cf. [1–7]). In a decentralised approach, members store their own observations locally. If A wants to find out about the reputation of B, it looks for other peers that interacted with B (called witnesses) and asks them for their observations about B. In this approach, reputation is calculated in a distributed manner, which provides a level of freedom to peers in choosing the method of calculating reputation according to their beliefs and preferences. Besides and since each peer can choose its own witnesses, it provides him more confidence on the resulting reputation value compared to the centralised approach. Consequently, the decentralised approach is more convenient for open communities such as P2P systems.Our work is related to these aforementioned systems. Differences are discussed in Section 5.

3 Proposed Approach

In this section, we briefly introduce the main concepts which are used in our approach: the Virtual Agora, the Register of (Un)Trusted Recommender and the Register of Trusted Items.

Virtual Agora (VA). It is a virtual open space (e.g. web site, server) where peers meet, interact, share experiences about items of interest. Items are advertised in the VA (i.e., only their names and characteristics are known within the VA). For instance an item can represent a restaurant and its description. The VA concept suggests three main characteristics: (i) Openness which means that peers from various sources can freely join or leave the VA at any time, (ii) no central authority that controls peers, and (iii) persistence of these peers (if desired) within the VA, which also suggests the persistence of the VA and its continuous availability.

Register of (Un)Trusted Recommender (TRec). From the subjective viewpoint of a peer A, TRec is the set of peers (met in the VA) whose (positive or negative) trustworthiness has been evaluated by A. A will consult them when she/he needs an opinion about a certain item. The term "subjective" means that each peer has a personalised TRec. For example, let us suppose that peer A is interested in Bikes. It joins a VA whose members are interested in the subject "bikes". A mets {B;C;D;E;F;G;L;..} and after some interactions forms her initial TRec (see [8] for more details) composed of {B;C;D;E} as illustrated in Fig. 1. B,

instead, may have a different community (e.g., {A;C;F;G;L}). From this point of view, our TRec concept is similar to wide-spread communities such as Msn Messenger (http://www.messenger.com) but as we will see, the trustworthiness of each is explicitly maintained.

Register of Trusted Items (TRat). It describes how much a peer trusts a set of items. Each relationship is weighted with the current rating that A gives to the item. The ratings are subjective to the member's profile and way of judging. We assume that the members of the community share some algebra of rating. Moreover, aside to the rating itself, the relationship keeps log of the information used to compose the rating (i.e. recommendations and recommenders). For example, in Fig. 1 b., *item2* is rated 3 by C (following a direct experience, i.e. C is the rater) and 4 by B (here B is not the initial rater but it is rather L).

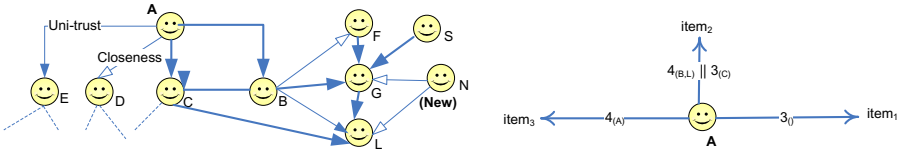

Fig. 1. a) Trusted Recommender and b) Trusted Ratings concepts

Both TRec and TRat have the following characteristics: (i) trust-based: the respective elements are linked by trust relationships, (ii) dynamic: trust relationships are continuously updated, and (iii) self-organised: members autonomously organise their trust relationships.

3.1 General Architecture

An appropriate architecture for a system which will support our approach should fulfill the requirements mentioned in Section 1. To this end, we propose the architecture illustrated in Fig. 2. The idea is to assign two agents to each mobile user. One agent (*Embedded-Agent*) is embedded in the mobile device and one agent (*Delegate-Agent*) is representing the mobile user in the VA. We thus decouple the "classical" *peer* into two pieces. In what follows we describe the main components of the architecture and demonstrate how the requirements mentioned above are achieved.

Virtual Agora. It is a meeting place for peers' *Delegate-Agents*. It includes a *Bulletin Board* and a *Policy Manager*. The former is required while the latter (gray in the Figure) is optional (depending on the application).

Bulletin Board. It is in charge of searching peers. In P2P networks, searching can be centralised (e.g. Napster has a resource directory server), distributed (in pure P2P networks like Gnutella), or intermediate (e.g. KaZaA) based

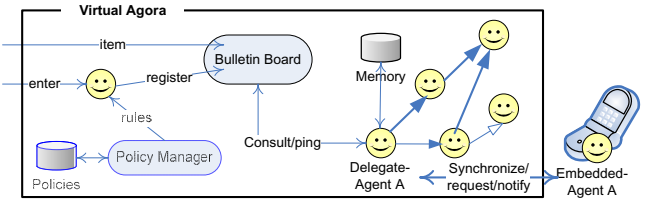

Fig. 2. Simplified Global Architecture (rounded rectangles refer to components)

on *nodes* and *supernodes*. In our architecture we opt for a centralised solution, named Bulletin Board (BB). This component is part of the VA and is managing the list of agents within the VA. A new agent in the VA can consult this list to discover other similar peers. Later, the agent relies more on its own network of peers (TRec), which considerably frees the BB. To keep its list up-to-date, the BB has to register new agents joining the agora and periodically ping agents in its list in order to check their availability. Moreover, BB keeps the list of items to be evaluated. We suppose that each item is uniquely identified among all peers.

Policy Manager. It is optional and used to grant rights and dictate rules and policies to different types of agents if the application requires such features. In this paper we do not focus on this component.

Delegate-Agent. This agent is hosted in the VA to represent the user vis-a-vis other peers. By interacting with other peers, it builds the TRec/TRat of its end-user and answers requests of recommendations (coming from its user or other peers). It is thus aware of its user's profile and preferences.

Embedded-Agent. This light agent (i.e. has few data and functionalities) is a proxy between the user and the *Delegate-Agent*. It mainly (i) notifies *Delegate-Agent* about user's ratings and tags, changes of interests or preferences, (ii) sends the updated user context to *Delegate-Agent*, and (iii) requests recommendations on behalf of the user.

Memory. Each *Delegate-Agent* has its own memory where it stores data which are useful when building the TRec/TRat or interacting with pairs.

Let us show how this architecture can fulfill the requirements presented in the introduction. By using the *Delegate* and *Embedded* agents, we allow mobile users to take advantage of the VA concept (to meet other peers). Besides, since a minimum interaction (to synchronise their data, to send and receive requests, and to notify each other about important changes) is needed between the two agents, the use of the wireless network is dramatically reduced. All interactions between the *Delegate-Agent* and the members of its TRec are indeed executed locally (in the VA). Regarding, managing the registration and discovery of the VA's members is ensured by the *BB* (and potentially by the *Policy Manager*).

Our VA-based solution is not a pure decentralised approach and therefore it may have a single point of failure (if the VA becomes temporarily unavailable).

We thus propose that the *Embedded-Agent* periodically (e.g., daily or weekly depending on the application) sends a copy of the TRat to the *Delegate-Agent*. If the VA is unavailable (for any reason) and that *Delegate-Agent* cannot be reached, the *Embedded-Agent* may use its TRat copy (certainly not optimised but still useful) to advice the end-user.

3.2 Agent Models

The *Belief-Desire-Intension* (BDI) model [9] has evolved over time and has been applied in several of the most significant multiagent applications developed up to now. It actually offers an interesting framework to design deliberative agents which are able to act and interact autonomously and according to their mental states. This model is the most appropriate for Delegate-Agents. We propose a BDI-based architecture which main components are illustrated in Fig. 3.

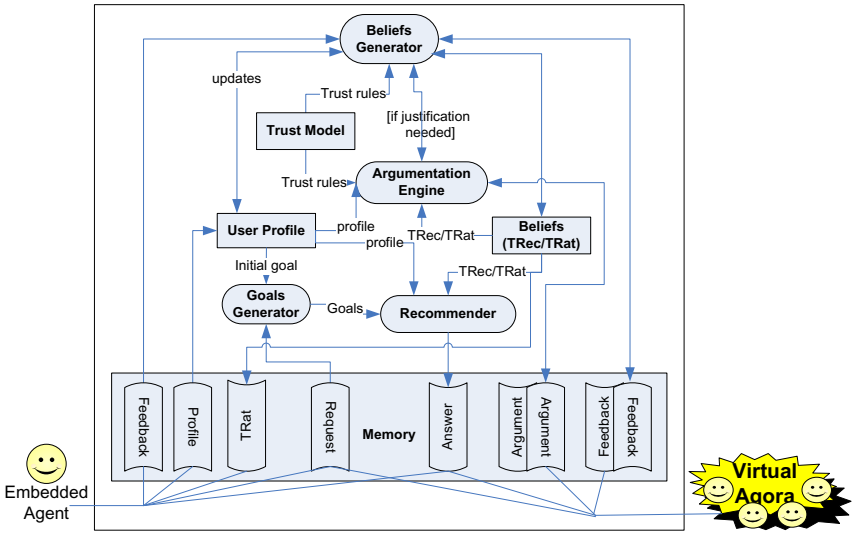

Fig. 3. Functional view of the Delegate Agent's BDI architecture

In what follows we describe the processes (rounded rectangles) and how they are related to the different data (rectangles). In the "Memory" component, two different shapes are used to show whether the data is an input (e.g. *Profile*) or an output (e.g. *Answer*).

– ***Goals Generator.*** It generates the goal(s) that the agent has to follow. When it first joins the VA, an initial goal is to build a TRec/TRat according to its user's profile. Later, its goals are dictated by specific requests coming from its user or members of the VA. This component generates then goals (which match the *Desire* concept in the BDI model) for the *Recommender*.

- ***Beliefs Generator.*** It generates the agent's beliefs received from the user (as feedback sent by the *Embedded-Agent*) or from other agents (as feedback or argument). While the former type of information is directly added as a belief, the latter is filtered according to a *trust model* and an *argumentation* process if necessary (see below). Final beliefs represent what the agent trusts and this corresponds to *Beliefs* in the BDI model.
- ***Argumentation Engine.*** This process is in charge of arguing with other agents in order to help the *Beliefs Generator* building its beliefs. For example, when the agent receives conflicting recommendations from different "trustful" peers, it can ask them to justify their opinions and argues with them in order to make a better choice regarding these recommendations. We describe the argumentation model in [10].
- ***Recommender.*** It uses the generated beliefs in order to answer incoming requests (from *Embedded* or other peers of the VA). The *Argumentation Engine* and the *Recommender* correspond to the BDI's *Intensions*.
- ***Memory.*** All inputs and outputs are stored in the *Memory* (see Fig. 3). Depending on the end-user's preference and the type of application, certain data are stored as long-term memory while others as short-term memory.
- ***Beliefs.*** Following Castelfranchi's postulate "Believe only if you have reasons to believe" [11], we make a distinction between memory and knowledge, between storing an information and accepting/believing it. Data stored in *Memory* are thus filtered to obtain *Beliefs* (represented by the TRec/TRat).
- ***User's Profile.*** It is composed of the user's *Preferences*, *Expertise*, and *Context*. The *Preferences* can be used for instance to discover how the user prefers the agent to present the information to him. *Expertise* is the skill or knowledge of a person who knows a great deal about a specific thing. Another important piece of information considered in the user's profile is the *Context* (location, time, social context, etc., see Section 6).
- ***Trust Model.*** This is the set of trust-based rules that the agent has to follow to infer trustworthy relationships (see Sub-section 3.3 and [8]).

While *Delegate-Agent* is deliberative, *Embedded-Agent* is more a reactive agent [12]. Indeed, the latter does not support any reasoning, it is only making the bridge between the user and the *Delegate-Agent*. It thus mainly reacts to incoming events. The main components of this reactive architecture are:

- ***Interface.*** This component interfaces with the user to get his requests or profile data and answer his requests.
- ***Synchroniser.*** It is in charge of synchronising data with *Delegate-Agent*. It mainly sends it requests and receives its answers, notifies it about changes on the user's profile, periodically requests a copy of the TRat, and forwards user's feedback to it.
- ***Beliefs.*** The *Embedded-Agent* only needs a copy of the TRat to be used in case of a VA failure. This is thus the only content of the *Beliefs*.
- ***User's Profile.*** Same as for the *Delegate-Agent*. Indeed, it is the *Embedded-Agent* which sends this profile to its *Delegate-Agent*.

3.3 Trust Evolution in TRec/TRat

According to to the literature, agent A's trust in item i is built from direct experiences [7] or from indirect experiences [4] (coming from other peers) or from a combination of both. Instead of using an unknown network of recommenders, we propose that A uses its own TRec to obtain focused recommendations (represented by TRat) about i. After being initialised, TRat and TRec are continuously updated and consulted. Due to the space constraint, we only present the general idea of these phases and do not talk about other important parameters for trust evaluation such as *Time* and *Context*. Our formal model of trust is described in [8] while solutions to integrate *Context* and to calculate similarities can be found in [13].

Bootstrapping. If A is new in the VA, it first has to build its initial TRec. To this end, it forms *Agent Closeness* (a weak trust relationship) relationships (Fig. 1 a.) with agents showing similar profiles and preferences as its user. If no agent similarities are found, it may establish *Closeness* relationships with few agents which are judged by the VA as trustful (these are agents who are the most trusted by other peers, i.e. present in many peers' TRecs). A has also to build its initial TRat by adding items that are potentially interesting for this user (for example in Fig. 1 b., A rates *item1* 3 using *item closeness*), or by asking members (recommenders) of its TRec (for example in Fig. 1 b., A receives two ratings about *item2*: 4 from B and 3 from C). These recommendations are not necessarily the result of the recommenders' direct experience (e.g., in Fig. 1 b., while C recommends *item2* using its direct experiences, B suggests a recommendation (4) of one of its TRec's members, L).

Consulting. When its end-user (or another peer of its TRec) requests a recommendation, A consults its TRat to reply to the request. If the information is not available in its TRat, it directly asks (by sending messages) its trustful recommenders (those in its TRec). When A has several recommendations about i, it weights them according to the trust relationships it has with these recommenders. For example, in Fig. 1 a. and from the perspective of A, a recommendation coming from E (*closeness*) has more weight that one coming from D (*uni-trust*, which reflects trust after direct experience).

Updating. Even if the end-user is not requesting any recommendations, A continually remains in the VA and whenever it receives a new information from another trusted peer, it updates its TRat (and eventually TRec if a new *close* peer has joined the VA) consequently. When A now receives a feedback from its user (following a direct experience) about i, it does not only update TRat, but also adjusts its trust relationships with its TRec's members based on what they had previously suggested about i. For example, if B turns out to be good recommender, A changes its trust relationship on B from *closeness* to *uni-trust*.

4 Implemented Scenario

Let us suppose that Bob is interested in slow-food. He wants to (i) find good slow-food restaurants in his city (or cities he is visiting), (ii) asks for reliable recommendations about specific restaurants from people having the same preferences as him, and (iii) share his own experience with other fans of slow-food. He signs up in a slow-food Virtual Agora (interfaced by a Web site to facilitate access for users) and sends a *Delegate-Agent* (called MyDelegate) to this Agora. Regarding the Embedded-Agent (EA-Bob), it has already been installed in Bob's mobile phone. In this scenario we do not use argumentation (see Sub-section 3.2) since the mechanism is not implemented yet.

By signing up, Bob has to fill in a form about his preferences concerning slow-food restaurants. For instance, he has to indicate which criteria are the most important according to him to rate a restaurant (price, quality of food, or service, etc.) or how he rates certain specific restaurants (in Fig. 4, he has one personal rating about restaurant R3). This information would be used by MyDelegate to argue when conflicting ratings arise (during the argumentation process) but mainly to build its initial TRec/TRat. Let us suppose that 3 agents (among others) Member1, Member2, and Member4 are now part of the TRec of MyDelegate and that MyDelegate has a *Closeness* relationship with all of them (value of trust in each member is "1", see Fig. 4). While building its initial TRat, and given the description (in the *BB*) of restaurant R4, MyDelegate decides to collect recommendations about R4. It then asks its TRec's members for advice and receives two conflicting recommendations about R4: Member4 recommends the restaurant (value="5", see Fig. 4) while Member1 and Member2 do not (value="1"). MyDelegate adds these different ratings to its TRat (concerning item R4). We suppose now that Bob is intending to have diner in restaurant R4 and that in the meaning while TRat has remained unchanged (i.e., MyDelegate did not receive any new information about R4 from its recommenders). Bob interacts then with his mobile to ask for recommendation. EA-Bob forwards the request to MyDelegate. This latter, noticing that R4 is in its TRat, tries to derive a personal rating about R4. Since Member4 is not more trustworthy than the two others, and without an argumentation mechanism (in this case), MyDelegate process an average rating (here 7/3=2.33) and communicates the chosen rating (2.33 over 5) to EA-Bob. Given the low rating, Bob decides not to go to the restaurant.

Let us suppose now that, few days later, Bob was invited in R4 restaurant by his friend and that contrarily to his expectations he appreciated his meal. He thus used his mobile to positively rate the restaurant before he left.

EA-Bob forwards this rating (for example "4.5" to MyDelegate which uses Bob's feedback to assess its TRec/TRat. In this case, the feedback confirms that Member4 was right and is likely to be trustful whereas Member1 and Member2 are maybe not. As a consequence, MyDelegate strengthens its relationship with Member4 by giving a higher weight to the corresponding link that becomes a *Uni-trust* and weakens its relationship with Member1 and Member2 by giving a lower weight to the corresponding *closeness* links. MyDelegate also update the

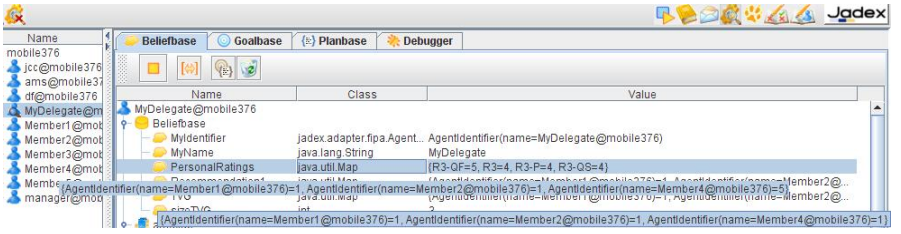

Fig. 4. Snapshot of the MyDelegate Agent in Jadex environment (due to the space constraint, we highlight data with tooltips)

TRat by assigning "4.5" (as a direct experience of Bob) to R4. An argumentation mechanism would have allowed MyDelegate to make a better choice about R4 since MyDelegate could have had justifications about these contradictory recommendations and could have decided according to Bob's profile and to how much arguments and proofs (of recommenders) are strong and acceptable.

We chose Jadex [14] as a development environment for the VA. Besides the fact that Jadex extends a reliable environment (Jade), it handles the BDI concept which is very useful in our case to easier implement the VA's members. We are currently working on integrating context and argumentation aspects.

5 Related Work

Most of the decentralised agent-based recommender systems [1–7] present four main differences with our approach. First, and except [6], they are all not suitable for mobile environments (require huge traffic between remote peers). Second, most of them use witness reputation mechanisms (e.g. [4, 5, 7]). In our approach we only use members' reputation in the bootstrapping phase, if the agent has no past relevant experience. Our list of trusted pairs is rather more subjective (first built according to closeness and then updated based on direct experience). Third, to our knowledge, none of the existing systems (including [6]) has proposed a decentralised recommender system which supports a high level of personalisation. In fact, a peer gets a much smaller set of opinions (in number) when asking for a recommendation. Nevertheless, these opinions are of high quality since they are more personalised (trusted source, justified opinions, similar context and profile). We actually focus on quality (of recommendations) rather than on quantity. Finally, in most of the existing distributed recommender systems, when a peer leaves the community the referral pointers become obsolete and the knowledge of the quitter is lost. In our system and since each peer shares its knowledge with the community, we claim that a peer's knowledge will remain available within peers which have accepted it (after argumentation). We thus enhance the persistence of knowledge in P2P systems.

With respect to the proposed architecture (Fig. 2), similar concepts to VA can be found in the literature. For example, the *Advertisement Infrastructure* [15] is a space where agents can build collaborative plans, while the ToothAgent system [16] offers a centralised service that agents use to interact and to meet servers on behalf of their users. The VA novelty is to allow entities to build their own subjective network of trust (TRec and TRat). Recommendation are not processed equally for all the members and items are not rated independently form the tastes of a subject or his/her past experiences.

6 Conclusion and Future Work

In this paper we presented an agent-based architecture for mobile P2P recommending systems. By decoupling a peer into two agents (*Delegate* and *Embedded*), our approach allows to dramatically reduce wireless traffic between peers (a mobile peer has a limited interaction with its representative in the VA). By adopting a BDI model, we also give agents more autonomy, which considerably frees users from managing their trust on other peers. From the recommender system perspective, it also goes beyond the traditional collaborative filtering approach by allowing each peer (a BDI agent) to manage its own trust on other peers, which ensures more reliable and trustworthy recommendations. By doing so, we go a step further than similar works such as [6] or TAKEUP [17].

As future work, we intend to integrate our argumentation model [10] in the prototype, which will allow peers to make better choices when evaluating recommendations. Besides, and given that the current prototype has only focus on the *Delegate-Agent* side (and the Virtual Agora), we are also working on the *Embedded-Agent* side and especially on integrating the context to our architecture. One way of capturing context is via *ContextWatcher* [18]; it is a mobile application developed in our research lab, and which aims at making it easy for an end-user to automatically record, store, and use context information. This context information will be used as an input parameter for the *Embedded-Agent* and forwarded to the *Delegate-Agent*. This later will use this information to evaluate its trust on received recommendations. For instance, let us suppose that each user's tagging (or rating) of a certain restaurant R4 will be associated with the context captured by ContextWatcher (e.g., time and location). A *Delegate-Agent* would then probably trust a rating which was recently made in the same city as R4 more than an out-dated rating which was made far from the R4's neighborhood. The *Delegate-Agent* can also use this contextual data during the argumentation process in order to evaluate proofs.

References

1. Foner, L.: Yenta: A multi-agent, referral based matchmaking system. In: Proc. of the 1st Int. Conf. on Autonomous Agents, pp. 301–307. ACM (1997)
2. Miller, B., Konstan, J., Riedl, J.: Toward a personal recommender system. ACM Transactions on Information Systems 22, 437–476 (2004)

3. Olsson, T.: Bootstrapping and Decentralizing Recommender Systems. PhD thesis, Uppsala University and SICS (2006)
4. Sabater, J., Sierra, C.: Social regret, a reputation model based on social relations. SIGecom Exchanges 3, 44–56 (2002)
5. Teacy, W., Patel, J., Jennings, N., Luck, M.: Coping with inaccurate reputation sources: Experimental analysis of a probabilistic trust model. In: Proc. of the 4th Int. Joint Conf. AAMAS, pp. 997–1004 (2005)
6. Tveit, A.: Peer-to-peer based recommendations for mobile commerce. In: Proc. of the 1st Int. Workshop on Mobile Commerce (WMC 2001), pp. 26–29. ACM, Rome (2001)
7. Yu, B., Singh, M.P.: An evidential model of distributed reputation management. In: Proc. of 1st Int. Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS 2002, Bologna, Italy, July 15-19, pp. 294–301 (2002)
8. Lenzini, G., Sahli, N., Eertink, H.: Agents Selecting Trustworthy Recommendations in Mobile Virtual Communities. In: Falcone, R., Barber, S.K., Sabater-Mir, J., Singh, M.P. (eds.) Trust 2008. LNCS (LNAI), vol. 5396, pp. 182–204. Springer, Heidelberg (2008)
9. Rao, A.S., Georgeff, M.: Bdi agents: from theory to practice. In: Proc. of the 1st Int. Conference on Multi-Agent Systems (ICMAS 1995), June 12-14, pp. 312–319. AAAI Press, San Francisco (1995)
10. Lenzini, G., Sahli, N.: Argumentation-based trust in recommender systems. In: Proc. of the Special session: Trust in Pervasive Systems and Networks, at SECRYPT part of the ICETE - The Int. Joint Conf. on e-Business and Telecommunications, Porto, Portugal (2008) (to appear)
11. Castelfranchi, C.: Reasons to believe: cognitive models of beliefs change. In: Proc. of the Int. Workshop on Cognitive, Computational and Logical Approaches to Belief Change, Amsterdam, The Netherlands (2004)
12. Brooks, R.A.: Intelligence without representation. Artificial Intelligence 47, 139–159 (1991)
13. Toivonen, S., Lenzini, G., Uusitalo, I.: Context-aware trustworthiness evaluation with indirect knowledge. In: Proc. of the 2nd Semantics Web Policy Workshop, ISWC 2006, Athens GA, USA. CEUR Workshop Proc. (2006), `http://CEUR-WS.org`
14. JADEX (2008), `http://vsis-www.informatik.uni-hamburg.de/projects/jadex/`
15. Maamar, Z., Sahli, N., Moulin, B., Labbe, P.: A software agent-based collaborative approach for humanitarian-assistance scenarios. Information and Security: An Int. Journal 8, 135–155 (2002)
16. Bryl, V., Giorgini, P., Fante, S.: ToothAgent: A Multi-agent System for Virtual Communities Support. In: Kolp, M., Henderson-Sellers, B., Mouratidis, H., Garcia, A., Ghose, A.K., Bresciani, P. (eds.) AOIS 2006. LNCS (LNAI), vol. 4898, pp. 212–230. Springer, Heidelberg (2008)
17. Schulz, S., Herrmann, K., Kalcklösch, R., Schwotzer, T.: TAKEUP: Trust-Based Agent-Mediated Knowledge Exchange for Ubiquitous Peer Networks. In: van Elst, L., Dignum, V., Abecker, A. (eds.) AMKM 2003. LNCS (LNAI), vol. 2926, pp. 89–106. Springer, Heidelberg (2004)
18. Koolwaaij, J., Tarlano, A., Luther, M., Nurmi, P., Mrohs, B., Battestini, A., Vaidya, R.: Contextwatcher - sharing context information in everyday life. In: Proc. of the IASTED International Conference on Web Technologies, Applications, and Services (WTAS 2006), pp. 39–60. ACTA Press, Calgary (2006)