Reasoning about Trust Groups
to Coordinate Mobile Ad-Hoc Systems

Licia Capra
Department of Computer Science
University College London
Gower Street, London WC1E 6BT, UK
L.Capra@cs.ucl.ac.uk

Abstract

The increasing popularity of mobile computing devices, coupled with rapid advances in wireless networking technologies, have created the infrastructure needed to support the anywhere-anytime computing paradigm. Middleware systems have started to appear that aim at facilitating coordination among these devices, without the user even thinking about it, thus receding technology into the background. However, faced with overwhelming choice, additional support is required for applications to decide who can be trusted *among this plethora of interacting peers. In this paper we propose a coordination model that exploits trust groups in order to promote safe interactions in the ubiquitous environment. Trust groups are* asymmetric*, that is, each device has its own view of the groups it belongs to, and* long-lived*, that is, their lifetime spans an extended period of time, despite group membership being dynamically handled. The dynamics of trust group creation, evolution and termination are described, based on the history of interactions of the device and on the ontology used to encode the context of trust. The programmer efforts required to reason about trust groups when coordinating mobile ad-hoc systems are discussed.*

1 Introduction

Mobile computing devices, such as mobile phones, personal digital assistants, digital cameras and the like, are getting increasingly ubiquitous. Their computing capabilities are growing quickly, while their size is shrinking, so that we rely on an everyday larger number of devices to accomplish any task at hand. Wireless networks of broader bandwidth allow these mobile units to aggregate and form complex distributed system structures, thus providing users anytime-anywhere access to their personal information, as well as public resources and services.

Coordinating these devices has been recognised as a major challenge. Mobile ad-hoc networks form opportunistically, with nodes entering and leaving the communication range dynamically. The execution context changes quickly as well, and devices are required to react to variations in both local and remote resource availability. In order to simplify application programming, new coordination paradigms that take into consideration the characteristics of the mobile computing environment have emerged, and middleware systems that support these paradigms have been developed. For example, JEDI [11] offers a publish-subscribe coordination paradigm that enables nodes to interact, despite network disconnections, by exchanging events; Lime [19] exploits tuple-space based primitives, together with an abstraction of context as data, to facilitate coordination both among mobile devices and between devices and their execution context.

However, one of the most difficult challenges of the mobile ad-hoc environment has not received much attention yet, that is, how to decide *who to trust* in this plethora of opportunistically connected peers. Each time an interaction takes place, we face an inherent risk as we can never be certain of the trustworthiness of the entities we interact with, or that mediate the interaction. The perceived risk is much higher in mobile ad-hoc settings than in traditional distributed systems, because of the lack of administrative boundaries, the anonymity of the entities we interact with, the speed at which new entities come into reach while others disappear, and so on. In these circumstances, collaboration may seriously be hindered and mobile devices may prefer to shut down connectivity, unless they are provided with a means to reduce the exposure to risky transactions.

A trust management framework (TMF) offers a solution to the problem. It aims at reducing the uncertainty that characterises mobile ad-hoc interactions by enabling devices to form, exchange and evolve trust opinions about other agents in the system. Such a trust management model should not be artificially imposed on the user, but rather perceived as natural, thus making it ultimately invisible. In order to achieve this goal, trust dynamics in the human society have to be observed and captured into computer models. The following two aspects of human trust are of particular relevance:

We reason as individuals - Intuitively speaking, trust can be defined as the degree of belief about the behaviour of another entity, or agent, upon which we depend (for example, to have a service delivered). These beliefs are usually based on our direct experiences with the agent (i.e., our history of interactions), and on recommendations (i.e., advice) that other agents provide us. Individuality of reasoning implies that the same history of interactions and the same set of recommendations may lead different people to form different beliefs about the trustworthiness of the same agent, because their natural disposition to trust (i.e., the way they reason about past experiences and recommendations) may differ.

We behave in groups - The entities we most frequently interact with form fairly stable groups; for example, we often eat in the same restaurants, we buy books from the same booksellers, and so on. Also, before interacting with an agent for the first time, we seek the advice of entities we know and trust: our family, our friends, our colleagues at work, and, in general, fairly stable communities of recommenders that share our interests and opinions.

In order to support subjective reasoning, we have developed hTrust [9], a trust management model and framework that relieves the application programmer from tedious tasks, such as collecting and processing recommendations from other agents in the network, while exporting a simple interface that, given in input the pseudonym of an entity, returns a trust prediction of the entity's behaviour. To capture the natural disposition to trust of the user of the device, and thus support subjectivity of reasoning that is typical of human trust, users can customise a set of functions that hTrust internally uses to compute trust.

We argue that supporting subjective reasoning is not enough. The information each device has to manage to reason about trust can be massive; for example, when asking for recommendations about a particular agent, hundreds or thousands of replies may be returned to the device for processing. However, only a very small fraction of these recommendations are actually taken into consideration (those coming from our acquaintances, that is, our *trust groups*), as most of them will typically come from unknown, or untrustworthy, recommenders. We argue that, besides enabling subjective reasoning, trust group modeling must be supported, in order to achieve more efficient and effective processing of trust information, as well as to foster a more natural coordination paradigm.

In this paper, we present a novel coordination model for mobile ad-hoc networks that reasons about trust groups to decide who to interact with. In Section 2, we provide an overview of hTrust, pointing out its limitations as far as coordination of mobile ad-hoc systems is concerned[1]. Section 3 characterises the trust groups we are interested in (i.e., long-lived groups that are asymmetrically defined by each agent), and it formalises the dynamics of trust group creation, evolution and dissolution, based on an agent's history of interactions and on the ontology used to encode the context of trust. In Section 4 we discuss a coordination middleware that realises the trust group model previously defined. Section 5 compares our work with others in the field and, finally, Section 6 concludes the paper and examines future directions of research.

2 hTrust Overview

hTrust promotes trust-aware collaborations in mobile ad-hoc networks by enabling each *trustor* agent a to collect and process trust information about a *trustee* agent b, so to form a trust opinion before interaction takes place. Sources of trust information are: *direct experiences* and *recommendations*.

Direct Experiences. The trustor's history of interactions with b is processed and kept locally in the form of a single *aggregated trust information* tuple:

$$[a,\ b,\ l,\ s,\ k,\ t].$$

The meaning of the tuple is as follows: agent a trusts agent b at level $l \in [-1, 1]$ (-1 meaning total distrust, and 1 meaning blind trust) to carry out service s. For example, we may specify that Alice (a) trusts Bob's eBookshop (b) at level 0.8 (l) to sell travel books (s). Because in mobile ad-hoc settings agents can have only a partial knowledge of their surroundings, their trust opinions contain a level of uncertainty. In order to distinguish between 'don't trust' (i.e., trust-based decision) from 'don't know' (i.e., lack of evidence), we explicitly model the degree of knowledge $k \in [0, 1]$ in the trust opinion expressed, with 0 meaning unknown, and 1 meaning perfect knowledge. The higher the number of direct experiences happened between the trustor and the trustee, the higher the degree of knowledge. The trustor's knowledge k decays with time; we thus associate, to each tuple, a timestamp t indicating at which time the knowledge k refers to. A service s of particular importance, provided by virtually

[1] Although we describe trust group reasoning on top of the trust management model we have previously developed, the concepts illustrated in this paper can be applied to different TMFs.

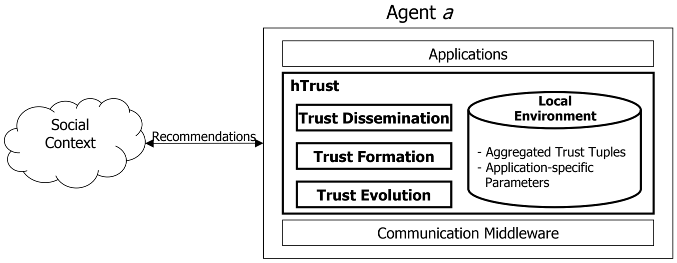

Figure 1. hTrust Overview.

every agent in the system, is the service of supplying the recommendations themselves. In human interactions, we tend to value more recommendations coming from people who have given us good recommendations in the past (i.e., people with whom we shared opinions), while discarding recommendations coming from unknown recommenders, or from recommenders with whom we had divergence of opinions. Agents are thus judged based on the quality of the recommendations they give, in the same way they are assessed for any other service they provide.

Recommendations. When direct experiences are not available (e.g., because no interaction has ever happened in the past between the trustor and the trustee), the trustor may ask other agents in the environment (what we call the *social context*) for recommendations. For example, Alice may be willing to buy books from Bob's eBookshop provided that it has been recommended by Clare (agent x). A recommendation tuple sent by x about agent b looks like:

$$[x,\ b,\ l,\ s,\ k,\ t]_{SK_x}.$$

A recommendation is thus computed by signing the local aggregated tuple; a signature is necessary to prove the recommendation's authenticity. We refer to x as to the agent's pseudonym; it is the piece of information the agent is known for in the system (e.g., its public key[2]).

Figure 1 shows hTrust model overview. The *trust formation* component is used whenever agent a is willing to make a prediction about the trustworthiness of another agent b; both a's past opinion about b and recommendations are processed (using customisable functions) to compute a range of possible trust values. Intuitively, the lower the confidence in the trust data (parameters k and t), the wider the predicted range, and viceversa. Recommendations are collected by the *trust dissemination* component. Upon completion of an interaction between a and b, the *trust evolution* component of agent a updates a's local environment: the aggregated trust information tuple that refers to b is updated based on b's just perceived trustworthiness; moreover, the tuples referring to agents that have sent a new recommendations about b are updated based on the difference between the recommended and the perceived trust. Details about the algorithms used to realise trust formation, dissemination and evolution can be found in [9].

Although enabling subjective reasoning, the social network model hTrust is based on, that is, a flat collection of individual agents, is too simplistic and far from reality. In human interactions, we view the social network as a set of (possibly overlapping) communities, and we most frequently coordinate with the communities we belong to. By modeling the social network as a collection of communities (or groups), rather than a collection of individuals, more effective and efficient trust reasoning can be achieved. For example, when seeking for recommendations about a specific service provider, rather than querying the social network at large, we may query only the community of people that we know can provide us with useful information about it, thus increasing the quality of the information received (effectiveness), and reducing the number of recommendations that have to be processed (efficiency). In the following section, we illustrate how to model groups on top of a flat social network and how to exploit them to promote trust-aware coordination.

3 Trust Group Coordination Model

Groups can be characterised using two orthogonal dimensions: *symmetric* versus *asymmetric*, and *volatile* versus *long-lived*. In a *symmetric* group, each member has the same view of who the other members of the group are; for example, the Mobile System Group at UCL is a symmetric group. Symmetric groups often have wide visibility, and their existence is acknowledged even by non-members; new members are usually accepted after explicit join requests have been processed. In an *asymmetric* group instead, each member has its own vision of who the members of the group are; for example, the group of Ann's friends may include Bob, but Bob's group of friends may not include Ann. Asymmetric groups have local visibility, and their dynamics are entirely defined by the member whose point of view is under consideration. *Volatile* groups have a very short lifetime, that is, they are created, managed and then destroyed rather quickly; for example, the group of people on a train from London to Cambridge. *Long-lived* groups have a longer lifetime instead, although membership can be very dynamic; for example, the group of our colleagues at work.

All combinations of these characteristics are plausible. In this paper, we focus on *asymmetric, long-lived* groups, as we believe they best suit our target scenario. To begin with, because of the dynamicity of mobile ad-hoc settings, symmetric groups cannot be efficiently maintained; even simple issues, such as using a voting procedure to accept

[2] We assume each agent has got a pair of public/private keys (perhaps more than one), that is managed via an independent public-key management system specifically developed for ad-hoc networks (e.g., [8, 16]).

a new member in a group, could become very difficult to deal with, if current group members cannot be reached for long periods of time. Second, as trust is subjective, it is unlikely that all members of a group would reach a consensus, further supporting our focus on asymmetric groups. Finally, as trust opinions are strongly based on past experiences, reasoning about volatile groups would do no better than performing a random choice about who to trust and who not to, as there would be no historical information to rely on.

In the remainder of the paper, we illustrate how to reason about asymmetric, long-lived trust groups to coordinate mobile ad-hoc systems.

3.1 Groups of Trust

The first question we need to answer is: who do we want to include in a trust group (i.e., who are we willing to coordinate with)? Usually, we want to include only those agents that have been trustworthy in the past, and that we have known long enough to give us confidence they will be trustworthy also in the future. Both information, that is, trustworthiness and confidence, can be found in the aggregated trust information tuples.

Given a set $\mathcal{I}$ of aggregated trust tuples, the set of agents we are willing to include in a trust group at a particular point in time is defined as follows:

$$\mathcal{A}_\tau(t_{now}) = \{x \mid x \in \pi_{trustee}(at \in \mathcal{I} | f_\tau(at, t_{now}) \geq \tau_{min})\},$$

where $\pi_{trustee}$ projects a tuple onto the trustee field; f_τ is an agent's specific function that computes the expected minimum trust of agent x based on the aggregated trust tuple $at = [a, x, l, s, k, t]$ at a specific point in time; and $\tau_{min} \in [-1, 1]$ represents an agent specific threshold so that, if the expected minimum trust is below τ_{min}, the agent is not included in the trust group. Different choices of f_τ are plausible, because of the subjective nature of trust. An example of f_τ is the following:

$$\begin{aligned}
&f_\tau : \mathcal{I} \times \ \mathcal{T} \rightarrow [-1, 1] \\
&f_\tau([a, x, l, s, k, t], t_{now}) = l - \Delta l, \\
&\quad \Delta l = l - l * k * \max\left(0, \frac{T - (t_{now} - t)}{T}\right),
\end{aligned}$$

where Δl represents the potential loss of trust due to time and knowledge uncertainty, and T is the time interval during which interactions are observed. This function takes into consideration the fact that trust information is time-sensitive: the older the information, the higher the uncertainty it embeds. For example, given the aggregated trust tuple $at = [a, b, 0.75, \texttt{bookseller}, 0.92, 1 * t]$, parameter $\tau_{min} = 0.45$, and $T = 10 * t$ for some time unit t, then b would be included in a's trust group at time $t_{now} = 3 * t$ as the following holds: $f_\tau(at, 3 * t) = 0.75 - (0.75 - 0.75 * 0.92 * \frac{10*t-(3*t-1*t)}{10*t}) = 0.75 - 0.19 = 0.56 \geq \tau_{min}$.

The above definition causes a projection of all the aggregated trust tuples an agent has collected over an arbitrary long history of interactions and recommendation processing, onto the group of tuples whose trustees currently have an expected minimum trustworthiness that is above an agent's defined threshold. By promoting an agent's coordination with the entities belonging to this group, the risks inherent to interactions in uncertain environments are reduced. However, this first level of grouping is not enough to support effective and efficient trust reasoning. For example, when we are looking for a trustworthy electronic bookseller, we would like to focus only the (trustworthy) agents that provide such service, without processing the aggregated trust information that refers to different services. We argue that, as trust information is context-dependent (i.e., it refers to a specific service), context must be taken into consideration when modeling trust groups. In the following section, we illustrate how to identify groups of agents that share the same interests (i.e., the same context).

3.2 Groups of Interest

As shown in Section 2, each aggregated trust tuple $[a, b, l, s, k, t]$ contains information about the context s to which the trust information refers to. This context can be simply defined as the type (or category) of service that the trustee b provides; for example, the above tuple may express a's trust in b as a $s =$'book seller', $s =$'flight booker', $s =$'restaurant advisor', and so on. In this work, we assume that context is represented by means of a shared ontology. In its simplest instance, an ontology [12] includes a vocabulary (i.e., the syntax), a taxonomy (i.e., the semantics - a tree structure imposed over a vocabulary), and a set of relationships among the elements of the taxonomy. Various ontologies (e.g., [2, 1, 3]) have been proposed to describe classes of services, each with different expressive power and targeted to different domains. In this paper, we are not interested in what particular ontology is used; we simply exploit its tree structure, where each node represents a service category, this being a sub-category of its parent node. We also make the assumption that each mobile device has access to (the portion of) the tree that defines services of interest to him.

Given the group $\mathcal{A}_\tau(t_{now})$ of the agent's most trusted interacting peers at time t_{now}, we operate a partition based on different service categories, and define a set of *groups of interest*, that is, groups whose members are providers of the same type of service s_i (or its sub-categories) , as follows:

$$\begin{aligned}
\mathcal{A}_{(\tau,s_i)}(t_{now}) = \{x \mid x \in \pi_{trustee}(at \in \mathcal{I} | f_\tau(at, t_{now}) \geq \tau_{min} \\
\wedge\, \pi_{service}(at) = instanceOf(s_i))\},
\end{aligned} \qquad (1)$$

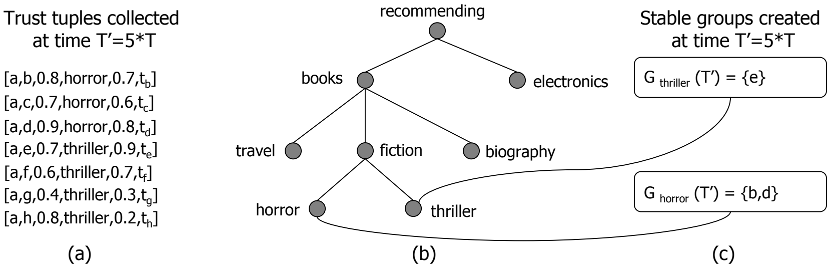

Figure 2. Group Dynamics - Group Creation Example.

$\pi_{service}$ being a function that projects a tuple onto the service field, and s_i being a service category in the ontology in use. As each $\mathcal{A}_{(\tau,s_i)}(t)$ contains a smaller number of members than $\mathcal{A}_{\tau}(t)$, the processing of trust information gains in terms of efficiency, as fewer trust tuples have to be processed to decide with whom to interact. However, to promote effectiveness of reasoning as well, our definition of groups must be further refined; in particular, we aim at capturing in our trust groups those agents that are fairly *stable* over time, as these are more likely to provide us with useful information in the future, as opposed to agents with whom we share only very short and/or unstable periods of interactions. In the next section, we illustrate how stable groups of trustworthy agents that provide the same type of service can be identified and maintained over time.

3.3 Group Dynamics

The lifecycle of agent a's trust groups can be described as follows.

Phase 0 - Knowledge Gathering. Let T be the time interval during which a observes interactions in the environment. During the first $T' = p * T$ time intervals, the basic mechanisms of hTrust are used to allow agent a to gather trust information about other agents in the system (in the form of direct experiences and recommendations). This information is maintained in the agent's local environment as aggregated trust tuples associated to the nodes of the ontology tree in use, depending on the service category they refer to.

Phase 1 - Group Initialisation. After p time intervals have passed, and the local environment has been populated with aggregated trust tuples, a process starts that aims at grouping these tuples into stable trust communities of interest. In particular, the ontology tree is traversed bottom-up, and for each node a stable group is created as follows:

$$\mathcal{G}_{(\tau,s_i)}(p*T) = \{\ x \mid \#\{y \in \biguplus_{j=1}^{p} \mathcal{A}_{(\tau,s_i)}(j*T) | y = x\} \geq \mu\ \} \tag{2}$$

where $\mu \leq p$ is an agent-defined parameter indicating the minimum number of times an agent x must have been considered trustworthy (according to definition 1) over the last p observations to be included in the current group of stable interacting peers. The closer the value of μ to the number p of time intervals that have been observed, the more stable the defined group. Let us consider, for example, the ontology tree shown in Figure 2(b); for the first $p = 5$ time intervals, a uses hTrust to maintain local trust tuples, arriving at time $5 * T$ with the set of aggregated tuples shown in Figure 2(a). Every T time units, formula 1 is used to compute a snapshot of trusted groups of interests. For example, assuming $\mathcal{A}_{(\tau,\text{horror})}$ to be $\emptyset$ at time T, $\{d\}$ at time $2T$, $\{b, d\}$ at time $3T$, $\{b, d\}$ at time $4T$, and $\{b, c, d\}$ at time $5T$, and assuming $\mu = 3$, then a stable trust group $\mathcal{G}_{(\tau,\text{horror})}(5T) = \{b, d\}$ is created, as shown in Figure 2(c).

Phase 3 - Group Maintenance. Instead of querying the social context at large to decide which agents to coordinate with, the group associated to node s in the ontological tree is considered whenever agents operating in context s are needed. After initial stable trust groups have been created using formula 2, they are maintained over time using the following formula:

$$\mathcal{G}_{(\tau,s_i)}((p+q)*T) = \{x| \#\{y \in \biguplus_{j=q+1}^{p+q} \mathcal{A}_{(\tau,s_i)}(j*T)|\ y = x\} \geq \mu\}\ , \tag{3}$$

that is, a stable trust group of interest $\mathcal{G}_{(\tau,s_i)}$ at time $p + q$ is computed as the set of agents that have proven to be trustworthy at least μ times in the last p observations (the first q observations are discarded instead, as they are now considered outdated). Note that each group $\mathcal{G}_{(\tau,s_i)}$ is computationally cheap to maintain, as calculations performed at the previous maintenance step can now be reused. In particular, $\mathcal{G}_{(\tau,s_i)}((p + q) * T)$ is simply computed as:

$$\left[\biguplus_{j=q}^{p+q-1} \mathcal{A}_{(\tau,s_i)}(j*T)\right] \setminus \mathcal{A}_{(\tau,s_i)}(q*T) \uplus \mathcal{A}_{(\tau,s_i)}((p+q)*T)\ ,$$

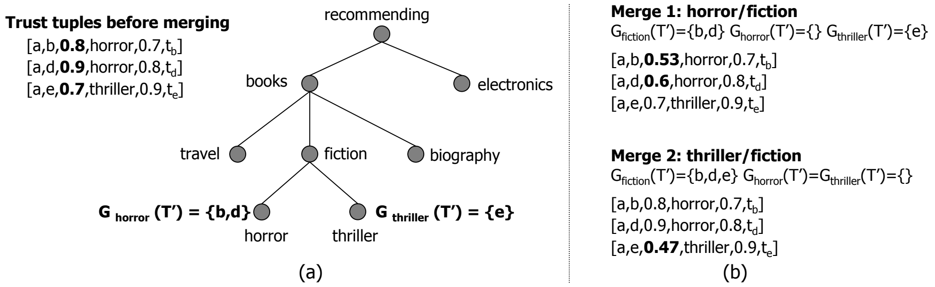
Trust tuples before merging
[a,b,0.8,horror,0.7,t_b]
[a,d,0.9,horror,0.8,t_d]
[a,e,0.7,thriller,0.9,t_e]
recommending
books
electronics
travel
fiction
biography
$G_{horror}(T') = \{b,d\}$
$G_{thriller}(T') = \{e\}$
horror
thriller
(a)
Merge 1: horror/fiction
$G_{fiction}(T')=\{b,d\}$ $G_{horror}(T')=\{\}$ $G_{thriller}(T')=\{e\}$
[a,b,0.53,horror,0.7,t_b]
[a,d,0.6,horror,0.8,t_d]
[a,e,0.7,thriller,0.9,t_e]
Merge 2: thriller/fiction
$G_{fiction}(T')=\{b,d,e\}$ $G_{horror}(T')=G_{thriller}(T')=\{\}$
[a,b,0.8,horror,0.7,t_b]
[a,d,0.9,horror,0.8,t_d]
[a,e,0.47,thriller,0.9,t_e]
(b)

Figure 3. Group Dynamics - Group Merging Example.

where the multi-union $\biguplus_{j=q}^{p+q-1} \mathcal{A}_{(\tau,s_i)}(j*T)$ is ready available from the previous maintenance step. New direct interactions of a with other agents in the system, as well as newly received recommendations, are fed into the system and accounted for in the computation of the last set $\mathcal{A}_{(\tau,s_i)}((p+q)*T)$ (see formula 1).

If a group of agents associated to service category s is not found in the tree, the tree is traversed bottom-up until a group is found; note that, the further up we move, the lower the chances to find such a group, because loss of precision in context definition implies lower trust values for the agents (i.e., the chances for an agent's trust to pass the threshold τ_{min} defined in formula 1 decrease). If a group is not found either on node s or on the nodes on the path from s to the root, hTrust basic mechanisms are used to gather fresh information about available providers of service s; a group creation may follow. It may also happen that, although a group exists, a is not satisfied with it, as it contains too few members (this number $size_{min}$ being agent specific). Consider, for example, the case where group cardinality is exactly 1: in this situation, agent a would restrict its interactions to a single other agent in the system, thus severely restricting its coordination within the social context. However, many other agents may exist that a trusts in a very similar, although not identical, context. For example, according to Figure 2, agent a would consider recommendations of thriller books only from agent e; however, there exists another group of agents (i.e., b and d) that are trusted in a similar (although not identical) context (i.e., recommendations of horror books). In these situations, it would be better to merge similar, very small groups into a bigger one, thus increasing the variety of trust information processed and, consequently, the potential for coordination. Before defining the dynamics of group merging, a definition of *similarity of context* is required.

Given two nodes s_1 and s_2 in the ontology tree, we define the *distance* of the two nodes $d(s_1, s_2)$ as the *least* number of nodes for s_1 to traverse to s_2[3]. For example, the distance between 'horror' and 'books' in Figure 2(b) is 2. Based on this definition, we can describe *group merging* as follows. For simplicity, we focus on the merge of the stable group g_1 of a child node s_1, with the stable group g_2 of a father node s_2; the merging among n groups arbitrarily placed in the ontology tree can be defined as a sequence of this basic merge operation.

$$\begin{aligned}
&merge(g_1, g_2) = g_2 \cup \Big\{ x \ \in g_1 \mid f_\tau(at', t_{now}) \geq \tau_{min} \\
&\text{with } at = [a, x, l, s_1, k, t] \mapsto at' = [a, x, l', s_1, k, t], \\
&l' = l * \Big(1 - \frac{h(s_2 \setminus s_1)}{D_{max}}\Big) \Big\} \qquad (4)
\end{aligned}$$

After merging, s_1 is left without group, while s_2 has an associated group that contains the union of g_2 and g_1. When moving one level up in the ontology tree, the aggregated trust tuples of the agents in g_1 have to be adjusted, as they now refer to a different (broader) context. In particular, given the aggregated tuple $at = [a, x, l, s, k, t]$ for an agent x in g_1, the trust value l in at is replaced by $l' = l * (1 - \frac{h(s_2 \setminus s_1)}{D_{max}})$, with D_{max} being the maximum distance between two arbitrary nodes in the ontology tree (for a balanced ontology tree, D_{max} would be $2 * height$, $height$ being the depth of the tree), and $h(s_2 \setminus s_1)$ representing the maximum distance between s_1 and any newly acquired context (that is, the maximum distance between the parent node s_2 and any of its descendants not belonging to the subtree rooted in s_1). Intuitively, the broader the new context to which agents in g_1 refer to (that is, the deeper the ontological subtree to which they are associated after merging), the higher the fading of the trust values to which they are associated (because of lower precision in the definition of the context of trust). Note that, when adjusting the trust tuple values during merging, so to take into account the additional uncertainty due to the loss of precision in the specification of context, an agent's trust level $f_\tau(at', t_{now})$ may not reach the minimum value τ_{min} required for that agent

[3] The definition of distance we have provided may be enriched using weights associated to the various nodes of the given ontology. For example, assuming the existence of another leaf 'romance' being subnode of 'fiction', we may want our definition of distance to return a higher value for the distance between 'romance' and 'horror', than the value returned for the distance between 'horror' and 'thriller'. In this paper, we do not investigate this possibility.

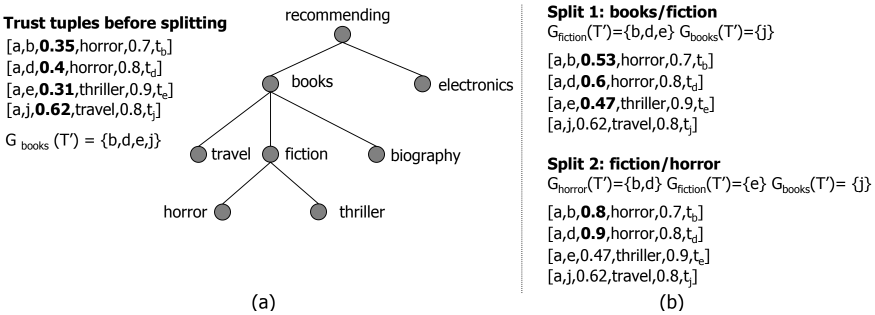
Trust tuples before splitting
[a,b,0.35,horror,0.7,t_b]
[a,d,0.4,horror,0.8,t_d]
[a,e,0.31,thriller,0.9,t_e]
[a,j,0.62,travel,0.8,t_j]
G books (T′) = {b,d,e,j}
recommending
books
electronics
travel
fiction
biography
horror
thriller
(a)
Split 1: books/fiction
G_fiction(T′)={b,d,e} G_books(T′)={j}
[a,b,0.53,horror,0.7,t_b]
[a,d,0.6,horror,0.8,t_d]
[a,e,0.47,thriller,0.9,t_e]
[a,j,0.62,travel,0.8,t_j]
Split 2: fiction/horror
G_horror(T′)={b,d} G_fiction(T′)={e} G_books(T′)= {j}
[a,b,0.8,horror,0.7,t_b]
[a,d,0.9,horror,0.8,t_d]
[a,e,0.47,thriller,0.9,t_e]
[a,j,0.62,travel,0.8,t_j]
(b)

Figure 4. Group Dynamics - Group Splitting Example.

to belong to a trust group (see formula 4). If this is the case, the agent is not included in the newly formed group, thus remaining an orphan: its trust tuple goes back to the local environment knowledge database, with the trust value restored to the context it originally referred to. Figure 3 shows an example of this process. On the left hand side (a), the state before merging is shown: two stable groups have been defined and associated to service categories 'horror' and 'thriller' respectively; the aggregated trust tuples that refer to agents in these groups are also shown. Because these groups contain very few members, they are merged in two steps, as shown on the right-hand side (b): a merge between horror and fiction occurs first, leaving the group for horror empty, while creating a new group for fiction and adjusting the trust values for agents b and d using the formula $l' = l * (1 - \frac{1}{3})$ ($D_{max} = d(recommending, horror) = d(recommending, thriller) = 3$, and $h(fiction \setminus horror) = d(fiction, thriller) = 1$); a second merge between thriller and fiction leaves the group for thriller empty and enlarges group fiction, while adjusting the trust value for agent e.

Group merging proceeds until either a group is created with sufficient members, or the group is dissolved (because the loss in context precision causes the agents' trustworthiness not to pass the minimum threshold). Viceversa, when a accesses an intermediate node whose group is too highly populated (this number $size_{max} > size_{min}$ being agent's specific), a split procedure is started, to refine the originally oversized group (oversizing may happen as a result of maintaining groups originated from merge procedures, thus covering broad contexts). Group splitting requires symmetric operations to be performed: given a stable group g_1 of a parent node s_1, the agents in g_1 whose context s belongs to the subtree rooted in the node s_2 (child of s_1) are removed from g_1 to form an independent group g_2 associated to s_2[4]. Also, trust values are re-adjusted (in particular, increased), as a consequence of precision gain in context definition.

$$\begin{aligned} & split(g_1) = (g_1', g_2) \mid g_1' = g_1 \setminus g_2 \ \wedge \\ & g_2 = \Big\{ \ x \in g_1 \mid \pi_{context}(at) = instanceOf(s_2) \\ & \text{with } at = [a, x, l, s_2, k, t] \mapsto at' = [a, x, l', s_2, k, t], \\ & l' = l / \big(1 - \frac{h(s_2 \setminus s_1)}{D_{max}}\big) \ \Big\} \end{aligned} \tag{5}$$

Figure 4 shows an example of this procedure. Before splitting (a), a single group containing agents b, d, e, j is associated to node 'books'. After a first split (b), a new group is created and associated to 'fiction', and the trust values of interested agents (i.e., b, d, e) are adjusted using the formula $l' = l/(1 - \frac{1}{3})$ ($D_{max} = d(recommending, horror) = d(recommending, thriller) = 3$, and $h(books \setminus fiction) = d(books, travel) = d(books, biography) = 1$). A second split follows that creates a group associated to service category 'horror', while adjusting the trust values of agents b, d.

Groups are implicitly dissolved in three circumstances: during a group merge (the child node group is dissolved), during a group split (the parent node group may be dissolved if it does not contain further members), and during group maintenance if no agents survive the pruning defined by formula 3 (e.g., because of inactivity or loss of trust).

As shown, groups are dynamically managed by means of both reactive and proactive operations: merging and splitting are reactively performed when anomalous situations are detected by a upon accessing a trust group in the ontological tree, while maintenance is proactively repeated at regular intervals of time on the currently defined trust groups. Also, hTrust basic mechanisms of querying the network at large are used from time to time, to provide a with fresh information, thus avoiding the risk of over-restricting a's knowledge of its surroundings.

4 Trust Group Coordination Middleware

The model described in the previous sections has been realised by means of the coordination middleware depicted

[4] Note that, during merging, the context specified in the trust information tuples was not changed, so that the original context the tuples refer to is not lost.

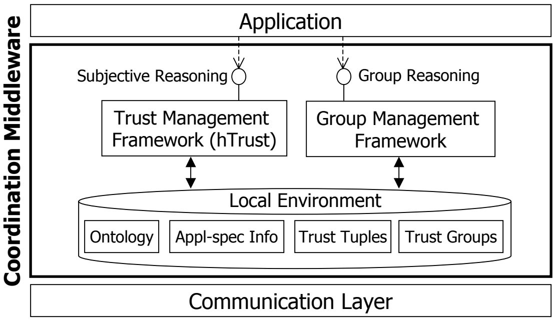

Figure 5. Coordination Middleware.

in Figure 5 (components that are not the focus of the paper, such as discovery, are not shown). The middleware provides the device (and its applications) with an image of the mobile ad-hoc system as a collection of communities, each focused on a specific service category, and populated by the most trusted agents delivering that service (according to the agent's perspective); low-level and tedious tasks, such as handling device connectivity, developing recommendation exchange protocols, implementing group dynamics etc., are not exposed to the applications. Application developers engineer trust-based collaborations by means of two simple interfaces: an interface that enables subjective reasoning about individual agents, and an interface that enables group reasoning. The former is provided by hTrust [9]: given an agent's pseudonym in input, the agent's expected trust is returned, based on the agent's past behaviour and the collected recommendations; maintenance of the agent's history of interactions in the form of aggregated trust tuples is performed by hTrust, without the application having to care about it. However, as argued before, subjective reasoning is not enough, as mobile ad-hoc settings are populated by large numbers of devices providing the same services, and it would be highly inefficient to make a prediction about all of them before deciding with whom to interact. The latter interface, realised using the model described in this paper, overcomes this limitation: given in input a service category, the group of stable, most trusted agents delivering that service is returned. In particular, our group management model processes the information contained in the aggregated tuples and, together with an ontology of service categories of interest to the agent, dynamically maintains stable trusted group of agents delivering the same service. As a result, an agent can more efficiently and effectively decide who to coordinate with.

We argue that the coordination model described in this paper is well-suited for the mobile setting for the following reasons: first, it is completely decentralised (i.e., each agent is a self-contained unit of trust information, with groups defined in a completely asymmetric way). Moreover, the resource demands imposed by the implementation of the framework are customisable, so that devices with different computing capabilities can tune the amount of resources devoted to trust management. For example, application-specific parameters (e.g., minimum trust level τ_{min}) can be chosen so to cause different overheads. We acknowledge the fact that a unique, optimal choice of these parameters does not exist, as they are domain-dependent; it is our plan for the future to tailor the framework to a specific application domain, and to empirically evaluate the impact of different choices of these parameters both on resource usage, and on efficiency and effectiveness gains over non group-based trust coordination models.

5 Related Work

The need to coordinate a growing number of mobile devices in scenarios dominated by uncertainty and high dynamicity, such as the mobile ad-hoc setting, has resulted in a growing community of researchers investigating trust management issues.

To date, most of the proposed solutions focus on providing support for *subjective reasoning*. In [4], a trust management model is proposed to give autonomous entities the ability to reason about trust, without relying on a central authority. Based on direct experiences and recommendations, each entity is able to derive trust measures, thus being responsible for its own fate. The approach relies on the assumption that entities will behave socially, exchanging recommendations when requested to do so, although no incentives are provided for this to happen. Also, no mechanism to dynamically re-evaluate trust decisions is discussed. In [7], a mechanism to detect and isolate misbehaving nodes at the network (routing and forwarding) level is proposed. The main advantage of the mechanism is that it works even without assuming the cooperativeness of the nodes; however, decisions about what nodes to isolate are performed in a completely automatic and homogeneous way. While this approach may work well at the network level, its lack of subjectivity severely limits its applicability at the application level, where the user's disposition has to be accounted for. As part of the SECURE project [13, 22], a trust management model has been defined that uses local trust policies to form and dynamically re-evaluate trust, based on past personal observations and recommendations; the computed trust values are then exploited to assess risks involved in the transaction, and then determine what behaviour the entity must adhere to. The model makes explicit, for the first time, the distinction between trust and knowledge, although the uncertainty that time brings in is not taken into consideration; also, the issue of malicious behaviours has not been explored. In [17], a mechanism for the management of distributed reputation in mobile ad-hoc networks is presented, that is able to effectively detect malicious recommenders based on the idea of 'recommendation reputation', that is, agents are judged based on the recommendations they have given in the past (although trust and knowledge are still confused). Social control mechanisms have been proposed to automatically isolate malicious entities and exclude them from future interactions, without having to rely on a trusted third party (e.g., [21]). The underlying assumption is the view of a mobile system as an ecologic system [18], where the interaction of the participants determines the success of the individual participant. Although sharing the same basic assumption, approaches developed to date are fairly limited, in that they do not capture a variety of aspects peculiar to human trust; for example, ways to recover from a bad reputation, and natural disposition to trust unknown entities. In [20], a different approach to distributed reputation management is proposed, that prescribes the use of first-hand experiences only, to circumvent the limitations of recommendations. We believe it to be unfeasible to work with first hand experiences only, as this would quickly saturate the system.

While supporting subjective reasoning to different extents, none of the approaches outlined above attempts to model *trust group reasoning*. Research in the area of group management is mainly found in the multi-agent system community. However, most of these works are funded on completely different assumptions that limit their applicability to our setting. For example, approaches such as the one described in [23] focus on *action* coordination, that is, how to distribute tasks rather than who to interact with. Other approaches (e.g., [6]) tackle the issue of which other agents to deal with; however, they assume that agents (and their roles) are known within the boundaries of a certain organisation, and thus cannot be applied to the mobile ad-hoc setting. In [14], the formation of trusted coalitions of agents is discussed; however, the paper presents very early work and ideas, without details about how coalitions are actually formed and how they evolve.

The model we proposed in this paper is a first attempt to combine subjective trust reasoning with trust group management, in a framework where the context of trust is also explicitly taken into account.

In our work we are tackling the problem of trust management from an engineering point of view, providing an operational model that programmers can actually exploit to build trust-aware systems; it is worth mentioning that various formalisms of trust have been proposed too, in order to help reasoning about trust. In [15], an opinion model based on subjective logic is discussed that can assign trust values in the face of uncertainty; however, the approach does not describe how to compute these values. In [10], a formal model for trust formation/negotiation, evolution and propagation is presented; however, the protocols for exchanging recommendations and for dynamically re-evaluating trust are not provided. Similar considerations hold for the formal trust models described in [5] (based on probability theory) and [24] (based on lattice, denotational semantics and fixed point theory).

6 Conclusion and Future Work

Coordinating the large, always increasing number of devices that populate mobile ad-hoc networks, has been recognised as a major challenge. In order to simplify application programming, this paper has presented a coordination model that fosters the engineering of trust-based collaborations, by means of long-lived, asymmetric, trusted groups of interest. Trust information is gathered, in the form of aggregated trust tuples, via a trust management framework, such as hTrust. This flat, unorganised information is then processed to identity fairly stable communities of an agent's most trusted interacting peers, based on an ontology that describes the service categories an agent accesses; the dynamics of group creation, evolution and dissolution have been defined. Although group management requires some additional resource consumption over non group-based solutions, it later simplifies an agent's reasoning about which other agents to deal with, thus actually achieving more efficient and effective coordination.

Our plans for the future span various directions. In Section 4, we have discussed our intention to tailor the model to a specific application domain, in order to analyse the impact of different choices of application-specific parameters onto, for example, resource usage. Other issues on our agenda include refinements of the ontological dimension of trust; in particular, we intend to provide a richer definition of semantic distance, based on weights assigned to different nodes in the tree. This would allow applications to influence the way merging and splitting operations are performed. Also, at present, a unique minimum trust threshold is defined for an entire ontological tree; in the future, we plan to assign different thresholds to different subtrees, thus enabling applications to differentiate the sensitivity of different services. Finally, although we assumed the existence of a single, universally accepted ontology, it is very unlikely there will ever be one; it is our intention to investigate techniques to perform probabilistic, on-the-fly translations between different ontologies.

References

[1] OWL-based Web Service Ontology. http://www.daml.org/services/owl-s/, 2004.

[2] Resource Description Framework (RDF). www.w3.org/RDF/, 2004.

[3] Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. http://www.w3.org/TR/wsdl20/, 2004.

[4] A. Abdul-Rahman and S. Hailes. Using Recommendations for Managing Trust in Distributed Systems. In *Proc. of IEEE Malaysia International Conference on Communication (MICC'97)*, Kuala Lumpur, Malaysia, Nov. 1997.

[5] T. Beth, M. Borcherding, and B. Klein. Valuation of Trust in Open Networks. In *Proc. of the 3rd European Symposium on Research in Computer Security (ESORICS '94)*, pages 3–18, Brighton, UK, Nov. 1994.

[6] C. Brooks and E. Durfee. Congregating and Market Formation. In *Proc. of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 96–103, Bologna, Italy, July 2002. ACM Press.

[7] S. Buchegger and I. L. Boudec. The Effect of Rumor Spreading in Reputation Systems for Mobile Ad-hoc Networks. In *Proc. of WiOpt 2003: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, Sophia-Antipolis, France, Mar. 2003.

[8] S. Capkun, L. Buttyán, and J. Hubaux. Self-Organized Public-Key Management for Mobile Ad Hoc Networks. *IEEE Trans. on Mobile Computing*, 2(1):52–64, 2003.

[9] L. Capra. Engineering Human Trust in Mobile System Collaborations. In *Proc. of the* 12^{th} *International Symposium on the Foundations of Software Engineering (SIGSOFT 2004/FSE-12)*, pages 107–116, Newport Beach, CA, USA, Nov. 2004. ACM Press.

[10] M. Carbone, M. Nielsen, and V. Sassone. A Formal Model for Trust in Dynamic Networks. In *Proc. of First International Conference on Software Engineering and Formal Methods (SEFM'03)*, pages 54–63, Brisbane, Australia, Sept. 2003.

[11] G. Cugola, E. D. Nitto, and A. Fuggetta. The JEDI event-based infrastructure and its application to the development of the OPSS WFMS. *IEEE Transactions on Software Engineering*, 27(9):827–850, 2001.

[12] T. Edgington, B. Choi, K. Henson, T. Raghu, and A. Vinze. Adopting ontology to facilitate knowledge sharing. *Communications of the ACM*, 47(11):85–90, 2004.

[13] V. C. et. al. Using Trust for Secure Collaboration in Uncertain Environments. *IEEE Pervasive Computing Mobile And Ubiquitous Computing*, 2(3):52–61, Aug. 2003.

[14] N. Griffiths and M. Luck. Coalition Formation through Motivation and Trust. In *Proc. of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 17–24, Melbourne, Australia, July 2003. ACM Press.

[15] A. Jøsang. A Logic for Uncertain Probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 9(3):279–311, June 2001.

[16] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang. Providing Robust and Ubiquitous Security Support for Mobile Ad-Hoc Networks. In *International Conference on Network Protocols (ICNP)*, pages 251–260, Riverside, California, Nov. 2001.

[17] J. Liu and V. Issarny. Enhanced Reputation Mechanism for Mobile Ad Hoc Networks. In *Proc. of the* 2^{nd} *International Conference on Trust Management (iTrust)*, volume 2995, pages 48–62, Oxford, UK, Mar. 2004. LNCS.

[18] M. S. Miller and K. E. Drexler. Markets and Computation: Agoric Open Systems. In B. A. Huberman, editor, *The Ecology of Computation*, pages 133–176. Elsevier Science Publishers, 1988.

[19] A. Murphy and G.-P. Picco. Using Coordination Middleware for Location-Aware Computing: A Lime Case Study. In *Proc. of the 6th International Conference on Coordination Models and Languages (Coordination 2004)*, volume 2949 of *Lecture Notes in Computer Science*, pages 263–278, Pisa, Italy, February 2004. Springer-Verlag.

[20] P. Obreiter. A Case for Evidence-Aware Distributed Reputation Systems - Overcoming the Limitations of Plausibility Considerations. In *Proc. of the* 2^{nd} *International Conference on Trust Management (iTrust)*, volume 2995, pages 33–47, Oxford, UK, Mar. 2004. LNCS.

[21] L. Rasmusson and S. Janson. Simulated Social Control for Secure Internet Commerce. In *New Security Paradigms Workshop*, pages 18–26, Lake Arrowhead, CA, Sept. 1996. ACM Press.

[22] B. Shand, N. Dimmock, and J. Bacon. Trust for Ubiquitous, Transparent Collaboration. In *First International Conference on Pervasive Computing*, pages 153–160, Dallas-Fort Worth, Texas, Mar. 2003.

[23] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101:165–200, 1998.

[24] S. Weeks. Understanding Trust Management Systems. In *Proc. IEEE Symposium on Security and Privacy*, pages 94–105, Oakland, CA, May 2001.