Context-based Grouping and Recommendation in MANETs

Yves Vanrompay, Manuele Kirsch Pinheiro, Nesrine Ben Mustapha, Marie-Aude Aufaure

Context-based Grouping and Recommendation in MANETs

Yves Vanrompay[1], Manuele Kirsch Pinheiro[2], Nesrine Ben Mustapha[1], Marie-Aude Aufaure[1]
[1] *MAS Laboratory / Ecole Centrale Paris, France*
{Yves.Vanrompay, Nesrine.BenMustapha, Marie-Aude.Aufaure}@ecp.fr
[2] *Centre de Recherche en Informatique / Université Paris1 – Panthéon Sorbonne, France*
Manuele.Kirsch-Pinheiro@univ-paris1.fr

ABSTRACT

We propose in this chapter a context grouping mechanism for context distribution over MANETs. Context distribution is becoming a key aspect for successful context-aware applications in mobile and ubiquitous computing environments. Such applications need, for adaptation purposes, context information that is acquired by multiple context sensors distributed over the environment. Nevertheless, applications are not interested in all available context information. Context distribution mechanisms have to cope with the dynamicity that characterizes MANETs and also prevent context information to be delivered to nodes (and applications) that are not interested in it. Our grouping mechanism organizes the distribution of context information in groups whose definition is context based: each context group is defined based on a criteria set (e.g. the shared location and interest) and has a dissemination set, which controls the information that can be shared in the group. We propose a personalized and dynamic way of defining and joining groups by providing a lattice-based classification and recommendation mechanism that analyzes the interrelations between groups and users, and recommend new groups to users, based on the interests and preferences of the user.

INTRODUCTION

Context distribution is a key aspect for successful applications in mobile and ubiquitous computing environments. Such applications typically need context information that is acquired by multiple context sensors distributed over the environment. Context-aware applications collect and react to this information, exploiting it through predefined adaptation mechanisms. Such mechanisms may vary from content adaptation to resource-driven adaptation and application deployment (Preuveneers et al. 2009). Context information on which these adaptation mechanisms rely is also extremely varied, being potentially any information that can be used to characterize the situation of an entity (a person, place, or object) considered as relevant to the interaction between a user and an application (Dey et al. 2001). Context-aware applications adapt their operations to such context information in order to increase usability and effectiveness by taking environmental context into account (Baldauf et al. 2007).

The success of such adaption mechanisms depends then on the availability of context information, which is disseminated in Mobile Ad Hoc Networks (MANETs). However, in practice, only a fraction of all the observable context information is of interest for the user or application. For instance, in a metro station, a wide variety of information can be available: temperature and humidity, available computing infrastructure, network status, etc. Each context-aware application running in such an environment will use a subset of all available information. A travel guide application requires available computing infrastructure, network status and user profile information, but does not need temperature and humidity information, which is context information exploited by maintenance and control applications. Context

distribution, which is defined as the capability to gather and deliver context information to interested entities (Bellavista et al., 2013), has to cope with this reality and adapt the distribution process to application needs.

In previous work (Kirsch Pinheiro et al., 2008), we designed a context-based grouping mechanism in which groups of peers were defined based on a criteria set (e.g. the shared location and interest) and a dissemination set (i.e. which information can be shared in the group). This approach relied on a rather static perception of groups, being defined explicitly by the developer at design time. Inference or recommendation of new groups taking into account the interests and preferences of the user was not considered. We supposed that the initiative to search for a specific group and the decision to join it was the responsibility of the user. In this chapter, we propose a more personalized and dynamic way of joining groups by providing a mechanism that makes it possible for the system to analyze the interrelations between groups and users, and recommend new groups to users, based on the situation and profile of the user, and representing concepts and groups by Galois lattices. Galois lattices are well-defined and exhaustive representations of concepts embedded in data sets (Ventos & Soldano, 2005), which can be used to represent conceptual hierarchies that are inherent in data (Stummer et al 2001).

In this chapter we present a context-based grouping mechanism, which allows the definition of groups based on contextual characteristics shared among members of the group. Each group is defined by these characteristics and specifies which context information can be distributed among group members. New groups can be inferred from existing ones by the incorporation of conceptual clustering with Galois lattices. This allows us to discover and analyze relations between different groups and the nodes contained in them. Based on a user profiles' similarity with the elements specifying the group and the current situation of the user, the system can decide to give a recommendation to the user to join a group. This leads to a dynamic, proactive and personalized user experience.

This chapter is organized as follows: Section 2 presents an overview of related work; Section 3 introduces a motivating scenario that illustrates the application of the proposed context grouping mechanism; Section 4 introduces the basic context grouping definition, after which the use of Galois lattices to automatically infer these groups is explained in section 5; Section 6 presents the context grouping mechanism's applicability to MANETs, including a lattice-based approach for joining suitable groups. Finally we discuss future work directions and conclusions.

RELATED WORK

Context information is a central element of context-aware applications. Such information, often acquired by multiple context sensors distributed over the environment, is used by context-aware applications for multiple adaptation purposes (Preuveneers et al. 2009). Thus, the success of such mechanisms depends on the availability of context information, which is disseminated in Mobile Ad Hoc Networks (MANETs).

Information dissemination is a common topic in many distributed system approaches, including Peer-to-Peer ones. Architectures such as (Harjula et al., 2006; Khambatti et al. 2003) propose different ways to manage and distribute information over a set of peers. Harjula et al. (2006) proposes a topic-based publish-subscribe system with SIP/SIMPLE. This proposal uses P2P SIP for the structure and maintenance of a P2P overlay. Khambatti et al. (2003) propose a grouping mechanism based on the notion of community. Communities are groups formed based on users (or peers) common interests, declared as peer attributes. Thus, communities are automatically defined over common claimed attributes, which correspond to published attributes from the set of all observable/available attributes: a community C is a non-empty set N of peers sharing a common signature, a signature S is the intersection set of claimed attributes claim(k) for all k in N. Nevertheless, these proposals usually handle stable information sets. For instance, communities proposed by Khambatti et al. (2003) are based on claimed attributes that are not supposed to evolve quickly. This is not the case for context-aware applications. Context information on ubiquitous environments can be characterized by its dynamicity, evolving with environment changes and user's movements. In other words, context-aware applications need proper

context distribution mechanisms that are able to cope with context data and with mobile and ubiquitous environments.

Context distribution can be defined as the function in charge of distributing relevant context data to interested entities (Bellavista et al. 2013). Context distribution is typically handled by middleware systems on top of which these applications are built. Different approaches have been proposed in the literature, starting from centralized solutions in which a central server concentrates and manages context information collected from the environment (Baladauf et al., 2007). Examples of such centralized architecture for context distribution include Henricksen et al. (2005) and Paganelli et al. (2006), which propose centralized entities (context repositories for the former and a domain server for the latter), managing the context information and handling the client requests for it. These centralized approaches are prone to scalability and single point of failures.

A different approach is then required in which the generation and the management of context information is distributed over the network. Such a distributed approach has to match requirements such as heterogeneity, scalability and robustness (Baladauf et al. 2007). In addition to these, an effective context distribution mechanism has to fulfill some extra requirements, among them (Bellavista et al., 2013):

a) *decoupling among context producers and consumers*: context producers and consumers do not need to know each other, context distribution has to transparently route produced context data from producers to all interested consumers;
b) *adaptation to mobile and heterogeneous environments*: context distribution has to promptly adapt to mobile nodes moving in and out, sometimes randomly;
c) *appropriate enforcement to context visibility scope*: context data has typically a limited visibility scope that depends on the physical or logical neighborhood Context distribution must enforce such scope to avoid useless management overload.

Despite the importance of these requirements for context distribution over mobile and ubiquitous environments, many of the proposed solutions in the literature do not reach all these requirements. For instance, Romero et al. (2010) propose a distribution mechanism that handles context data as a RESTFul resource. Even if this approach successfully handles context data interoperability, it fails in decoupling context producers and consumers. Indeed, context producers and consumers declare themselves using a component-based definition language. Communication among these elements is guaranteed by REST principles, which obliges consumers to know the multiple producers. Similarly, Devlic and Klintskog (2007) have proposed a context engine providing mechanisms to retrieve, synthesize and distribute context information in distributed mobile environments. This engine uses a query principle in which a node can query context information from another node. Again, context consumers and producers have to know each other in order to allow context distribution.

In order to cope with the previously mentioned requirements, some context distribution mechanisms propose a P2P approach. For instance, Ye et al. (2007) proposes a context sharing mechanism in which context information remains locally stored on the peers and only an access reference is registered on remote peers. The discovery of new peers is performed by broadcasting messages and context queries to remote peers are sent through unicast messages. Even if this mechanism matches the first two requirements, it fails on fulfilling the third one, since broadcasting registration messages is performed regardless of peer interests on context data. Peers have to manage such context discovery process by themselves. Besides, such a flat structure may raise scalability issues in ubiquitous environments, since every peer potentially registers references to available context data on every discovered peer.

A grouping mechanism can be used in order to prevent such flat structure. For instance, Paridel et al. (2011) propose a context-based grouping mechanism particularly designed for vehicular networks. Such mechanism intends to reduce network traffic by controlling context propagation between groups. However, such groups are defined based on a strict criterion defined by vehicle location and direction. No logical neighborhood (other than location) can be defined, which may comprise the visibility scope of some context data.

Context grouping mechanisms, such as Paridel et al. (2011), can be considered as an *informed context data distribution*, since it exchanges context data according to a dynamic adapted distribution process, in opposition to an *uninformed context data distribution*, which blindly routes context data without inspecting their content (Bellavista et al., 2013). In a previous work (Kirsch-Pinheiro et al., 2008), we have proposed an informed context distribution mechanism, which is inspired by P2P dynamic group definition proposals. However, different from traditional P2P mechanisms, context groups can be seen as communities defined using common context information. We have designed a context-based grouping mechanism in which groups of peers were defined based on a criteria set (e.g. the shared location and interest) and a dissemination set (i.e. which information can be shared in the group). This approach relied on a rather static perception of groups, which had to be defined explicitly by the developer at design time. Inference or recommendation of new groups taking into account the interests and preferences of the user was not taken into account. We supposed that the initiative to search for a specific group and the decision to join it was the responsibility of the user. This lack of dynamicity on group definition can be considered as a limitation, since groups should be previously defined and cannot adapt themselves to interests of nodes that are currently available in the network.

We propose a more personalized and dynamic way of joining groups by providing a mechanism that makes it possible for the system to analyze the interrelations between groups and users, and recommend new groups to users. Recommendation systems have moved from being used by websites like Amazon.com to key elements for realizing personalization and user-centric computing. Recent research combines context-aware computing with recommendation systems and user profiles in order to flexibly and dynamically meet user needs on mobile devices in changing environments (Nauer & Toussaint, 2007). A promising way to achieve even further personalization is the representation of the available information by concept lattices (Mao et al., 2010; Li et al., 2007; Kwon & Kim, 2009). Our goal is to dynamically recommend context-based groups to the user in a personalized way leveraging concept clustering by these lattices. Inspired by these works, we propose in this chapter a new context grouping mechanism, capable of adapting the grouping definition by recommending new groups to users, based on the analysis of Galois lattices representing concepts and groups.

MOTIVATING SCENARIO

In order to demonstrate the importance of the context grouping mechanism, we describe a motivating scenario in a metro station. Let's consider Jim, a tourist visiting Paris. Jim is looking for a restaurant and he is using a travel assistant application in order to better organize (and enjoy) his stay on Paris. The travel assistant application is a context-aware application that observes and distributes context information in the network. Indeed, this application will gather information about location, network status, available infrastructure (e.g. available devices and display screens), personal details, visited locations, supplier booking references (for restaurants, hotels, or travel agencies), and user preferences around Jim in order to adapt its content and behavior to the available resources (using, for instance, a big screen available in the metro station to show Jim how to reach the restaurant). Some of these data will be locally gathered on Jim's device (e.g. Jim's location), but others will be made available by other devices (nodes) in the neighborhood.

As stated earlier in this chapter, in a metro station, several context-aware applications coexist. Among the different nodes executing these applications, different context information is gathered and can be exchanged among these nodes. Groups of nodes can then be logically formed based on the nodes interests in context information.

Nevertheless, the travel assistant application is not the only context-aware application running in the metro station. We have also a traffic information application and metro maintenance applications. Each of these applications will gather and distribute different context information, according to their needs. A traffic information application will be interested on context information such as location, network status and direction, while a maintenance application will be more interested in user role, humidity, temperature

and available infrastructure. If every node connected to this network starts distributing all collected context information, devices connected to it will quickly overload, and Jim's phone will probably waste too much time filtering context data it receives.
In order to prevent disagreements that can be caused by such unnecessary processing, applications may constitute groups of nodes sharing common interests on context information. Such groups can be defined based considering node neighborhood, not only network or physical neighborhood, but also logical neighborhood, defined using common context information. Group templates can be manually defined, based on a static perception of user interests on lodging, attraction sites, travel location, transport issues. Since each interest category contains a lot of dimensions, this leads to a combinatorial explosion of the number of templates.

For instance, Jim is using his mobile device and asks for the restaurant the most visited in Paris providing Italian food. The travel assistant should search in group templates repositories to find an exact matching with *Jim's interest*.
Two possible scenarios can happen. The first one concerns the fact that no exact matching is found so that the system will rely only on shared information about restaurant suppliers and will provide the user with a long list of restaurant. In a second case, the system finds many general templates that are not relevant for Jim's request.
In both cases, the travel assistant cannot find dynamically the appropriate group template according to the user interest and Jim will abandon the use of his travel assistant since he can by himself search over the web for the most visited Italian restaurants. This is due mainly to two facts:

- Manually defined group templates do not reflect the actually shared information about user interests
- A static context grouping mechanism cannot carry out deep personalization as it is based on static matching with predefined criteria in templates.

With the aim of these two problems, the main motivation resides on providing a spontaneous way to build group templates by formally analyzing the large amount of data about user interests. Group templates have to be built based on the most common criteria associated with a high number of similar user interests.

For instance, the travel assistant application can enter in a group defined based on user preferences and location information, including travelers located in the same metro station that have similar interests, while a maintenance application can get in a group defined based on the user role. In each group, nodes can share different context information: network status, available infrastructure for the former, humidity and temperature for the latter. By separating nodes in groups, based on shared context information, we can isolate Jim's phone from context information coming from nodes interested only in the maintenance application. However, let's consider that there is no other traveler in the metro station that shares Jim's preferences. Based on the group definition used by travel assistant application, Jim will not receive any context information, since no other node shares with him context values used as criteria for defining the group. Even if the context information needed by Jim's travel assistant is available on other nodes, Jim's node will be outside groups sharing such information. For instance, let's consider that the traffic information application defines a group based only on location information and shares context information related to network status, available infrastructure and direction. In this case, Jim's phone may detect that there is another group in the network and decide to enter this group in order to receive the context information it needs.

In this chapter, we handle context distribution by proposing a context grouping mechanism that allows the definition of groups based on common shared context information.
Additionally, we propose using Galois lattices in order to improve such grouping mechanism. Lattices are used for recommendation purposes, adapting the groups to the current situation of the node neighborhood. Indeed, lattices provide a natural and formal setting to discover and represent groups of interests

hierarchies. “Formal concept analysis” is mainly used for data analysis, i.e. for investigating and processing explicitly given information (Szathmary & al., 2008) (Gao and Lai, 2010). Lattice-based analysis can bring out context-based groups that were not predefined by the applications or the middleware.

CONTEXT-BASED GROUPING WITH GROUP TEMPLATES

Context dissemination is organized in groups, which are defined using context information. The main principle consists in forming groups with nodes that share common observable context items (for example, the same location, the same network connection, etc.). Within these groups, it is specified which type of context information will be disseminated. The disseminated information typically corresponds to interests shared amongst the members of the group like traffic alerts or weather information. Grouping can be seen as corresponding to the node neighborhood. The notion of neighborhood is semantically extended to include not only the notion of network neighborhood (nodes in the same network), but also geographical neighborhood (nodes in the same location) or other application-defined criteria (nodes executing over similar devices, nodes acting on behalf of users playing a given role in an organization, etc.). Thus, applications can determine the criteria for the group formation. These criteria for forming context groups depend on the context characteristics, which can be described by the context model used. All context elements defined in the context model can be used and combined by the application or the middleware to form groups (for example, groups based on the location, on the network connection, etc.). Once the criteria are defined, the groups are formed and used as context addresses for distribution purposes. Figure 1 gives an illustration of the grouping definition. This figure shows a set of nodes forming two groups: (i) a first group is formed based on the node location (group formation criterion) and disseminates information about device available memory, screen size and battery; (ii) a second group is defined based on the user role and the group disseminates location information. The criteria defining the groups are at runtime translated to real values of observable context information (e.g. location = room x, user role = expert), which are used to disseminate the allowed context information, i.e. device resource information for the first group, user location for the second one. In this sense, this context grouping mechanism can be seen as a decentralized publish-subscribe mechanism since applications can decide to join only groups disseminating context information, which the application is interested in.

Figure 1. Two context group definition composed by group criteria (“Location”, “UserRole”) and corresponding dissemination set (“Memory, ScreenSize, Battery”, “Location”).

Only information related to the group is disseminated to group members. We consider two types of context requests: pull requests in which nodes query context information from other nodes in the neighborhood, and push requests in which nodes proactively send context information to other nodes in the neighborhood. The group definition becomes then central for context distribution over the network. A context group G_D is defined as follows:

$$G_D =< C_D, I_D >,$$

where C_D is the criteria set, i.e. set of context elements that determine the context group. I_D is the dissemination set, also called the working set. It is the context information that can be disseminated in this group, i.e. the context elements that are allowed to be disseminated (by push requests) or requested (by pull requests) in this group.

Initially the group definition is established at design time. Each application defines its groups based on its context information needs. Thus, for group management, the application has to determine the criteria for forming a group and the context information, which is allowed to be distributed in this group. An

application defines a *group template* in which it stipulates, using queries in a Context Query Language (CQL) (Reichle et al. 2008), the criteria and dissemination sets that define the group. The CQL (Reichle et al. 2008) allows the expression of fine-grained groups through complex queries, exploiting logical constraints and semantic references to entities represented in the context model. As an example, Figure 2 shows two CQL queries defining, respectively, the criteria set (Figure 2a) and the dissemination set (Figure 2b), which act as a template for a context group. This template defines a context group based on the user's location information (Figure 2a). It expresses that the shared context item by which the group is defined is equal to the room the user is in. The group allows the dissemination of context information related to the user's device memory status (Figure 2b). For readability purposes, we express this group template using a literal representation as illustrated by Figure 2c.

Figure 2. Context definition in XML.

For the application, the group definition remains stable over time, since its template remains the same. However, since this definition relies on context information, it is naturally dynamic. The values corresponding to the group criteria and the dissemination set are updated by the reasoning layer according to the runtime context changes. The context group template is used to instantiate an actual context group: the query representing the group criteria is processed and the current values of the corresponding context elements are used to form the concrete context group. For example, given a group G1 defined by <[position], [memory]>, as was illustrated in Figure 2, its definition criterion is translated to [position = room x], corresponding to the current value of this context element. Once the value of the context elements used as criteria change (e.g. the user goes into another room), the group definition is updated to the new value ([position = room y]), and accordingly to this new situation, nodes may leave and join (or create) the corresponding group at the network level.

Based on the group definitions, the context group mechanism organizes the context dissemination and requests, by handling accordingly the context query messages and dissemination messages coming from other nodes over the network, as well as context request messages coming from the application. The context group mechanism hides from the application dynamic changes related to the context groups. Nevertheless, the grouping mechanism depends on the query that determines the grouping criteria. If this query is not well-formed or is ambiguous, the context grouping mechanism is unable to interpret it and to form or join the context groups. For instance, given a location-based context group, an application can consider that two users are in the same location if they are in the same room (such as in the query in Figure 2a), or in the same city, or in the same coordinate interval. Even if the CQL allows a query just indicating the context concept location, this is not enough to indicate without ambiguity what the application treats as being in the same location. Applications using the proposed context grouping mechanism should avoid queries with any ambiguity when defining context grouping criteria.

Such context grouping definition performed at design time has an important drawback: group criteria remain stable and it does not adapt to networks variations and opportunities. For instance, a node can find itself alone in a group whose criteria are not shared by any other node. We supposed that the initiative to search for a specific group and the decision to join it was the responsibility of the user. In this chapter, we propose a more personalized and dynamic way of joining groups by providing a mechanism that makes it possible for the system to analyze the interrelations between groups and users, and recommend new groups to users, based on the situation and profile of the user, and representing concepts and groups by Galois lattices. In next section, we present how lattices can contribute to the group formation and management, realizing an important inference method for group templates.

LATTICE-BASED GROUP FORMATION AND MANAGEMENT

Although group profiles can be defined explicitly by the application developer, this is not optimal because it restricts the kind of groups that can be formed. In general, it is not possible to know at design time what groups will be relevant in a given context. In order to provide a personalized experience to the user, in

which groups can be dynamically recommended in function of the situation and interests of the user, we propose the use of concept lattices. Contextual elements that (potentially can) specify groups are represented as concepts in a Galois lattice, which allows conceptual clustering and the discovery of novel groups that match user's needs. Lattice-based formal concept analysis takes as input a matrix specifying the set of nodes in the MANET (the *objects*) and the corresponding context elements (the *properties* or *attributes*) specifying the groups each node belongs to. This representation allows us to:

- find "natural" node clusters: the set of all nodes that share a common subset of context elements
- find "natural" context element clusters: the set of all context elements shared by one of the natural node clusters.

Natural context element clusters correspond with natural node clusters, and a concept is a pair containing both a natural property cluster and its corresponding natural object cluster.

By analyzing and visualizing this emerging and evolving concept lattice, we gain the following three features. 1) group membership and dependencies between groups can be inferred to dynamically *form new groups* based on group templates suitable to the overall user interests. 2) we can efficiently *recommend possibly interesting groups* to users. The detection of potentially suitable groups uses similarity metrics between the elements specifying a group and the users' interests, which are represented as scores associated with concepts and attributes in an ontology. Alternatively, similarity between different users can be calculated in order to recommend groups based on collaborative filtering. 3) By showing in real time (using existing visualization tools) the evolution of the (relative) size of and relations between groups, "*hot topics*" can be detected and used for commercial purposes.

In the next section we explain how group templates can be inferred using Galois lattices. Then, we present how this approach can help in recommending interesting groups to users.

Inference of Group Templates

We propose techniques based on formal concept analysis (FCA) to identify a) sets of users that share common interests and b) relations and dependencies between interests. Analyzing this information allows us to automatically derive appropriate group templates.

In FCA, a concept is identified both by an extent, i.e. a set of objects that share the concept's attributes, and an intent, the set of attributes common to the concept. The starting point of FCA is a formal context, which is defined as a triple (G, M, I), where G is a set of objects, M a set of attributes and I a binary relation over G x M. A formal context corresponds to a binary matrix with the rows being the objects and the columns the attributes. Given $O \subseteq G$, O' is defined as the set of attributes of M which are shared by all elements of O. Given $A \subseteq M$, A' is defined as the set of objects that share all attributes in A. A formal concept of a context is a pair (O, A), O being the objects that share all attributes of A. O is the extent and A the intent, i.e. O = A' and A = O'. The set of all concepts derived from the formal context can be represented as a lattice, based on the subconcept relation $\leq$ for partial ordering: $(O_1, A_1) \leq (O_2, A_2)$ if $O_1 \subseteq O_2$ or equivalently that $A_2 \subseteq A_1$. The maximum element in this ordering is the concept with an empty extent, and the minimum is the concept with an empty intent.

Contrary to the design time and rather static definition of groups by application developers, our goal is to infer suitable candidates for group templates by analyzing a set of users with their interests, and the relations between these interests. The user's interests and preferences are expressed in ontology modules as entities and attributes that are annotated with scores $-1 \leq w_i \leq 1$ indicating a user's (dis-)interest in a particular item. Each ontology module is domain-specific, representing for example interests in houses for sale or restaurants. Figure 3 shows a hierarchical modular ontology for the tourism domain. Each leaf

node corresponds to a specific ontology sub-module representing an entity with attributes according to the specified schema. For example, the restaurant sub-module contains attributes like location, food type and dress code. Scores associated with these attributes indicate whether the user prefers e.g. Indian or Italian restaurants, or whether the dress code should be formal or casual.

Figure 3 Hierarchical modular ontology for tourism

Each scored ontology module serves as input for the construction of the formal context (G, M, I). The set G holds all users, while M contains relevant attributes (interests). I is defined as follows:

$I(g_i, m_j)$: G x M $\rightarrow$ {0, 1} = 1 if $w_j(u_i) \geq T$, otherwise 0

$w_j(u_i)$ is the score assigned for user i to attribute j, and T is a threshold between 0 and 1. An example formal context, the crosses indicating a value of 1 for I, is shown in Figure 4.

Figure 4 Example formal context and corresponding Galois lattice for restaurant module

Based on this formal context, we build a concept lattice (illustrated in Figure 4) allowing us to identify the sets of attributes which are best suited to be taken into account for the definition of group templates. The lattice assigns users to (sets of) interests, which are the building blocks for the group template definition. A user U_i is interested in all attributes that are connected to and hierarchically higher in the lattice than the node the user is assigned to.

From the Galois lattice we can automatically infer association rules like the ones shown in the box below. For example, rule 1 states that 7 users are interested in restaurants with formal dress code, and 6 users out of these 7 prefer french food. From the association rules, we infer suitable group templates, which balance generality and specificity. Group templates should not be too general because contextual information must be disseminated as much as possible only to users that are interested in that information. Users do not have to be flooded with all kinds of information. On the other hand, templates should not be too specific, because this leads to a large number of sparsely populated groups. Intuitively, this corresponds to concepts in the lattice on a not too high and not too low level, i.e. somewhere in the middle.

1 < 7 > DC:formal =[86 %]=> < 6 > FT:French;
2 < 7 > FT:Indian =[86 %]=> < 6 > loc:Paris;
3 < 5 > loc:Paris DC:formal =[100 %]=> < 5 > FT:French;
4 < 6 > FT:French DC:formal =[83 %]=> < 5 > loc:Paris;
5 < 6 > loc:Paris FT:French =[83 %]=> < 5 > DC:formal;
6 < 6 > FT:Italian =[83 %]=> < 5 > FT:French;
7 < 6 > FT:Italian =[83 %]=> < 5 > loc:Paris;
8 < 4 > loc:Paris FT:Italian FT:French =[100 %]=> < 4 > DC:formal;
9 < 4 > FT:Italian DC:formal =[100 %]=> < 4 > loc:Paris FT:French;
10 < 4 > FT:Indian FT:French =[100 %]=> < 4 > loc:Paris;
11 < 4 > FT:French withTerrace =[100 %]=> < 4 > DC:formal;
12 < 4 > DC:formal withTerrace =[100 %]=> < 4 > FT:French;
13 < 5 > conn:wifi =[80 %]=> < 4 > FT:Indian;
14 < 5 > loc:Paris FT:French DC:formal =[80 %]=> < 4 > FT:Italian;
15 < 5 > FT:Italian FT:French =[80 %]=> < 4 > loc:Paris DC:formal;
16 < 3 > loc:Paris FT:Indian FT:French DC:formal =[100 %]=> < 3 > FT:Italian;

The association rules allow us to identify the group templates that realize a balanced number of groups with users who share common interests. Two basic metrics are used to find the sets of attributes defining the group templates:

- Support s: This metric denotes the proportion of users that expressed their interest in a set of attributes. For example, association rule 4 indicates that 6 users (out of the total of 15) have an interest in restaurants serving French food with formal dress code, meaning support = 0.4.
- Confidence c: This metric expresses the proportion of users having an interest in the rules' consequent, given they have an interest in the rules' antecedent. In rule 4 for example, 5 out of 6 users that are interested in French restaurants with formal dress code are interested in Parisian restaurants. So confidence in this case is 0.83.

The group templates' dissemination set is defined as the attributes of those association rules which have a minimal support s_{min} and minimal confidence c_{min}. These minimal values can be tuned according to the application needs and priorities. For example, if it is important that not much information is disseminated in groups that is irrelevant for some users, c_{min} should be put rather high. For our purposes, we set s_{min} to 0.4 and c_{min} to 0.8. Minimal support ensures that only concepts with extents having at least $100.s_{min}$ % of all users are considered candidates for the group templates. Algorithms for computing frequent item sets are used to build such a so called Iceberg lattice (Stumme et al. 2002).

As soon as the appropriate candidate attribute sets for the group templates are identified, a second filtering step takes place using the stability index metric φ (Kuznetsov et al., 2007).

Let K = (G, M, I) be a formal context and (A, B) be a formal concept of K. Then φ is as follows:

$$\varphi(A, B) = \left| \{C \subseteq A \mid C' = B\} \right| / 2^{|A|}$$

Informally, a concept (i.e. group in our case) is considered stable if its intent (set of interests) does not depend much on each particular object of the extent (set of users). Stability expresses how much the set of attributes forming a group template depends on the interests of its individual users. If a group is persistent, it does not depend on a few users: some users leaving the group should not change the set of interests the group represents. It should be noted that we already minimize this risk by incorporating the minimal confidence metric in step 1. A stable group also does not merge with a different group or split into several independent subgroups when some users leave the group.

Apart from the "intensional" stability, we also use the extensional stability, which measures how the users of a particular group depend on the particular attributes. If we remove some attributes from the group template definition, does the set of users remains more or less the same. Extensional stability indicates how much the group of users depends on a particular interest.

Group Recommendation Using Local Views

The previous section explained how group templates can be derived given the interests of a set of users by building a global lattice. This lattice gives a global view of the group templates and serves as a bootstrap to start users joining and leaving groups according to their needs. When a user found a group that matches his interests and joins it, the members of the group can recommend other possibly interesting groups to the newly joined that they themselves are member of. This collaborative filtering process allows to exploit the local knowledge of each user about groups he is a member of to efficiently recommend other groups.

Indeed, a group can be seen as a community, and groups that are popular in that community can be of interest for the newly joint user.
The local information is represented by local lattices, where the objects are the users and the attributes are groups that the users are member of. The formal context defines a membership function of users to groups for one specific group A, as is exemplified for a group with 4 users in the table below.

A	B	C	D	F
U1	X	X		
U2	X		X	
U3				X
U4	X	X		

Figure 5 Local lattice for group A

When a user U5 joins group A, it will publish its group membership information and the local lattice is updated. First, the local lattice, shown in Figure 5, is used to check which groups are popular in group A, and groups are accordingly recommended to U5. In our example, the support for group B is 0.75 and this group is recommended to U5. In case U5 is already a member of one or more of the groups in the local lattice, a recommendation will be made containing the direct children of these groups.

LATTICE-BASED GROUPING IN MANETS

The context grouping definition, as presented in the previous section, can be applied in many different settings. Indeed, it is independent from the architecture adopted by the hosting system. Nevertheless, a context grouping mechanism like this one is particularly interesting for highly dynamic environments, in which entities may appear and disappear at any moment, such as MANETs. When considering MANETs, one should consider its dynamic nature. As a consequence of this dynamic nature, assuming any centralized approach in such environment is impractical. We cannot suppose neither any previous knowledge about the nodes availability on the network. In such conditions, a peer-to-peer (P2P) approach imposes itself as an interesting solution for handling such dynamic environments. In this section, we present a P2P-based architecture for nodes adopting our context grouping mechanism and we discuss its applicability to MANETs.

Architecture

Several works (Hu et al., 2008; Ye et al., 2007; Paridel et al., 2011) have proposed P2P-based approaches for context distribution, some of them dividing peers according to their roles or capabilities. Hu et al. (2008) considers three kind of peers, mainly according to device capabilities and associated roles for context distribution purposes: (i) sensor peers, which correspond to devices providing only raw context data (typically, context sensors); (ii) consumer peers, resource-constrained devices that need polished (and synthesized) context information; (iii) disseminator peers, which represent peers providing both provider and consumer functionalities, they provide context distribution services and perform the context reasoning. We agree with this distinction, since, for us, the heterogeneous nature of nodes involved in the context distribution process cannot be ignored and should be considered. Such distinction on three kinds of nodes allows a better applicability of the grouping mechanism to different kinds of mobile devices. Nodes with limited device capabilities can be used as sensor or consumer peers, while more powerful devices, typically last generation smartphones, will be used as disseminator peers. Indeed, current smartphones are more and more powerful compared to older mobile phones. Dual-core processors are now equipping several models, and quad-core devices are expected for the next generation of smartphones and tablets (Olivarez-Giles, 2012; Petric, 2012). This kind of devices, which is becoming extremely popular nowadays, can be used as disseminator peers, since they have enough computational power to support both context providing and reasoning. Moreover, information distribution becomes

more and more feasible since recent approaches optimize the efficiency of peer-to-peer content distribution techniques close to a theoretical lower bound (Kangasharju et al., 2006), even under the assumption of heterogeneous bandwidth and under churn (i.e. peers joining and leaving). Also, techniques have been developed (Wolfson et al., 2007) that explicitly take into account resource constraints like limited energy, communication bandwidth and storage of devices in a mobile ad-hoc environment.
For our purpose, we are particular interested in disseminator peers, which take full responsibility for context distribution in this architecture. Besides, we consider that some of these disseminator nodes play also a relay role, connecting different groups in the network. These relay nodes, acting as super-peers, can keep a partial knowledge of the network state that can be applied for lattice construction (cf. previous section).
Each disseminator peer has a multilayer architecture represented in Figure 6: a "reasoning" layer, which manages the context grouping (together with other context reasoning functionalities, which is out of the scope of this chapter), and a "distribution" layer, which handles the context dissemination according to the grouping definitions. In this way, the grouping mechanism can be defined independently of the actually used distribution technologies. Each layer is represented by a service that is in charge of layer responsibilities. Thus, each disseminator peer offers, as illustrated by Figure 6, a "Context Service", which provides context clients with a way to query context or to subscribe to context updates, and a "Distribution Service", which disseminates context information in a certain scope.

Figure 6. Disseminator peer architecture.

In the reasoning layer, the *context service* is in charge of interpreting group criteria for forming groups depending on the context characteristics. Context data is represented using a particular context model, which defines and describes context information. The context model delimits what context elements can be potentially used as grouping criteria: all context elements defined in the context model can be used and combined to form groups (for example, groups based on the location, on the network connection, etc.). In our case, we adopt the MUSIC Context Model (Reichle et al. 2008). This model enables the representation of context information in terms of *context elements*, which provide context information about *context entities* (the concrete subjects the context data refers to: a user, a device, etc.) belonging to specific *context scopes*. The Context Query Language (Reichle et al. 2008), used for describing the group profile, adopts the same model. It allows applications to submit to the context service queries about an entity and referred to a context scope, specifying a set of constraints that represent the filters of the query.
The context service interprets group criteria, represented using a CQL query, and uses this information to manage the group membership according to context changes. It provides the group management mechanisms to create, update and leave context groups. In addition to managing peer groups, the context service also handles the application requests for context information, in the form of pull or push requests. Based on the group definitions, the context service organizes the context dissemination and requests, by handling accordingly the context query messages and dissemination messages coming from other peers over the network, as well as context request messages coming from the application.

Besides, the reasoning layer offers a recommendation service, which is responsible for analyzing network information and eventually recommend new groups to the context service. Periodically, this recommendation service gathers context information from the different peers belonging to the same groups the peer belongs to. Disseminator peers acting as relay peers are particularly well-placed for gathering context information from groups and build a more complete lattice representing peers interests, sharing this information on each "context gather" request performed by neighborhood peers. Based on this information, the recommendation service builds a lattice and may then recommend new groups, as explained previously in this chapter. It may recommend these new groups to the applications, on the application layer, requiring then an explicit user validation, or directly to the context service (corresponding to an implicit validation), according to user's preferences.

Concerning the *distribution layer*, it acts as a mediator between the reasoning layer and the mechanisms offered by the P2P core service at the network layer. The *distribution service* interprets the groups defined by the context service and controls the distribution of the context information over the network according to these definitions. It guarantees that only information related to the group is disseminated to the group members. Besides, the distribution service handles the context group formation at the network level. As shown in Figure 6, the distribution service is responsible for isolating the context and recommendation services from real P2P technology used in the network level. This means it has to translate the context groups to traditional P2P groups that can be understood by the technologies used on the P2P core level of the architecture. Actually, different P2P technologies are available and the distribution service can be potentially confronted to different ones. We may cite, for instance, JXTA (2012) and Pastry (2012) as two examples of P2P technologies that can be used in the network level for our context grouping mechanism. JXTA (2012) is an open network platform designed for P2P computing, which provides a common set of open protocols for peer-to-peer communication. Basically, it standardizes the manner in which peers perform common operations such as discovering each other, self-organizing into peer groups, communicating, etc. Developers can leverage open source implementations of JXTA protocols for creating point-to-point applications. Pastry (2012) is a generic, scalable and efficient substrate for P2P applications. Pastry nodes form a decentralized, self-organizing and fault-tolerant overlay network within the Internet. It provides efficient request routing, deterministic object location, and load balancing in an application-independent manner. Furthermore, Pastry provides mechanisms that support and facilitate application-specific object replication, caching, and fault recovery. FreePastry is an open-source implementation of Pastry intended for deployment in the Internet. Such multiplicity of available P2P technologies is the main reason behind the separation between reasoning and distribution layers in the proposed architecture. On the one hand, the context grouping mechanism can be defined independently of the technology used at the network level. On the other hand, the distribution service decouples the operations performed on the groups from the P2P technology used to implement these operations.

In order to keep this independence between the reasoning and distribution layer, the context and recommendation services interact with the distribution service only through a well defined API. This interface defines the functionalities supplied by the distribution layer, hiding technology details to the context service and to recommendation service. Through this API, it is possible to consider different implementations for the distribution service, without compromising the reasoning layer implementation.
For instance, the recommendation service relies on the distribution service API for gathering peers interests (notably context information in which neighborhood peers are interested on), group information and even neighborhood local lattices.

Applicability to MANETS

This section describes how we can efficiently search across the P2P network exploiting local and global view lattices. A query for a specific group is not flooded over the network, but routed over a super-peer structure. A hybrid P2P network includes a set of peers P and super-peers SP. Links exist exclusively as pairs $\{SP_i, SP_j\}$ and $\{SP_i, P_j\}$. Compared to pure P2P systems, an architecture with super-peers has several advantages. The search is much faster since it is broken into a search for information from a smaller set of super-peers, each having their own indexed information from their set of peers. The problem of network flooding associated with pure P2P systems is hereby resolved. Also, each super-peer cluster corresponds to an autonomous unit not depending on a central server for information exchange. Lastly, super-peers introduce load balancing by only allowing nodes with sufficient computing power to act as super-peer.

In the following we explain the lattice-based group finding approach by a running example. The tables in Figure 7 show three super-peer clusters with their associated peers. For each peer it is indicated whether

there is an interest in high-level concepts like hotels, restaurants, museums or transport. The super-peers with their clusters are depicted in Figure 8. The global view lattice for each super-peer cluster, derived from the tables in Figure 7, is also shown. The virtual groups formed by the common interests of the peers are represented by the ovals. For example, P15, P16 and P17 are interested in hotel information and form a virtual group within cluster SP1.

Suppose a user P18 is searching for museum information (C4) having a set of attribute values (AT1_V2, AT2_V2 and AT3_V1) and is connected physically to SP1. The super-peer SP1 searches in its lattice for groups that are associated to museum interests (i.e. concept C4). As can be seen in Figure 8, there are no groups in cluster SP1 that can satisfy the user request. So SP1 will transfer the request to the super-peers in its neighborhood, being SP2.
In cluster SP2, the same search process is performed and it found a possibly relevant group associated to C4, consisting of 6 peers. Next, a deep search is carried out inside these groups to find the suitable subgroup that matches with the set of attribute values (ATi_Vi), which is the specific part of the user request.

Figure 7 Physical super-peer clusters with peers expressing their concept interest

Figure 8 Super-peer networks with global view lattices
For this, we use the local view lattice of concept C4 in cluster SP2, shown in Figure 9. The local view lattice for each concept shows the detailed interests of the peers belonging to the virtual group of the concept museum. Since P18 is specifically interested in attributes AT1_V2, AT2_V2 and AT3_V1, we assign it to the context group shown in the box in Figure 9, and which already contains P24 and P25.

Figure 9 Local view lattice for museum concept in SP2

Finally, Figure 10 shows P18 in cluster SP1 having joined the appropriate museum group, after which the global lattice of cluster SP1 is updated to reflect this change.

Figure 10 P18 joined the appropriate museum group, after which SP1's global lattice is updated

FUTURE RESEARCH DIRECTIONS

The proposed context grouping mechanism represents a first step for the combination of context distribution and recommendation systems. Context-based recommendation systems are already an important tendency (Adomavicius et al. 2005 and Abbar et al. 2009). Nonetheless, this work considers a different point of view: the use of recommendation techniques for context distribution. We are strongly convinced that context information is more than a simple parameter for adaptation purposes, it is the key for providing a personalized experience to the user. More than a new context distribution mechanism, what we propose here is a fully context-based distribution mechanism that can be used to distribute not only necessary context information, but also to distribute any information corresponding to current user's needs in real-time. By considering concept lattices for analysing emerging groups of interests, we can efficiently recommend possibly interesting groups to users, which represents a new opportunistic

adaptation mechanism for context-aware computing. We can also detect and visualize "hot topics", which may represent an important source of information for application designers in the future.
Distribution mechanisms like this one, that explore both context information and recommendation techniques, represent new opportunities to be explored by context-aware application, notably the applicability to social computing. Such applications, which represent an important application domain for pervasive computing, can heavily benefit from initiatives like this one. Nevertheless, a more formal analysis of the complexity presented by lattice building is still missing. Even if such analysis is particular important considering lattice-based recommendation techniques, we have to consider that, contrarily to traditional recommendation systems, in this particular case of context-based grouping, not the number of users, nor the attribute set will be very large in real life cases. A practical evaluation of the proposed mechanism will be realized in order to validate this hypothesis and to supply necessary data to complete this analysis.

CONCLUSION

In this chapter, we have proposed a context grouping mechanism for context distribution over MANETs. This grouping mechanism allows the management of groups of peers based on shared context information. However, a static definition of group profiles at design time, is not enough since it is quite difficult to predefine groups that will be relevant for users in a given context. In order to tackle this problem, we have considered that groups could be dynamically recommended by the grouping mechanism. Thus, we proposed an improved context grouping mechanism that is capable of recommending to users new groups based on current available context elements and on user's interests (including interests on context information itself). The improved grouping mechanism uses Galois lattices in order to recommend groups that best match the interests. By using Galois lattices, we can automatically infer suitable candidates for group templates by analyzing a set of users with their interests and the relations between these interests.
The next step in this work includes its evaluation in a more practical scenario. A Java implementation is under development for this purpose. A practical evaluation of the proposed lattice-based context grouping mechanism will allow us to analyze performance of the proposed mechanism, and notably the lattice-based recommendation, and to demonstrate the feasibility of this approach, mainly through a formal analysis of the complexity of lattice building.

REFERENCES

Abbar, S., Bouzeghoub, M., & Lopez, S. (2009). Context-aware recommendation systems: a service-oriented approach. *35th International Conference on Very Large Data Bases (VLDB)*, France

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 17(6), 734-749

Baldauf, M., Dustdar, S. & Rosenberg, F. (2007). A survey on context-aware systems. *International Journal of Ad Hoc Ubiquitous Computing*., 2, (pp. 263–277).

Bellavista, P.; Corradi, A.; Fanelli, M.; Foschini, L. (2013). A survey of context data distribution for mobile ubiquitous systems, ACM Computing Survey (CSUR), vol. 45, n° 1, Mar. 2013, 1-49.

Devlic, A., & Klintskog, E. (2007). Context retrieval and distribution in a mobile distributed environment, *Third Workshop on Context Awareness for Proactive Systems* (CAPS 2007), Guildford, UK, June 2007

Gao, J. & Lai, W. (2010). Formal Concept Analysis Based Clustering for Blog Network Visualization. *ADMA* (1) 2010. (pp. 394-404).

Harjula, E., Ala-Kurikka, J., Howie, D. & Ylianttila, M. (2006). Analysis of peer-to-peer SIP in a distributed mobile middleware system. *IEEE Global Telecommunications Conference* (GlobeCom06), (pp. 1–6), 2006

Henricksen, K., Indulska, J., McFadden, T., & Balasubramaniam, S. (2005) Middleware for distributed context-aware systems. In Vol *3760 of Lecture Notes in Computer Science*, (pp. 846–863). Springer Berlin / Heidelberg, 2005

Hu, X., Ding, Y., Paspallis, N., Bratskas, P., Papadopoulos, G., Vanrompay, Y., Kirsch Pinheiro, M. & Berbers, Y. (2008). A Hybrid Peer-to-Peer Solution for Context Distribution in Mobile and Ubiquitous Environments, *17th International Conference on Information Systems Development* (ISD2008), (pp. 501-510).

JXTA: The Language and Platform Independent Protocol for P2P Networking, http://jxta.kenai.com/

Kangasharju, J. & Kangasharju, J. (2006). An Optimal Basis for Efficient Peer-to-peer Content Distribution Algorithms, In Proceedings of the 15th International Conference on Computer Communications and Networks, (pp. 481-486)

Khambatti, M., Dong Ryu, K. & Dasgupta, P. (2003). Structuring peer-to-peer networks using interest-based communities. In volume 2944 of Lecture Notes in Computer Science, (pp. 48–63). Springer, 2003.

Kirsch Pinheiro, M., Vanrompay, Y., Victor, K., Berbers, Y., Valla, M., Frà, C., Mamelli, A., Barone, P., Hu, X., Devlic, A. & Panagiotou, G. (2008). Context grouping mechanism for context distribution in ubiquitous environments, *OTM 2008 Conferences*, (pp. 571-588).

Kuznetsov, S., Obiedkov, S. & Roth, C. (2007). *Reducing the Representation Complexity of Lattice-Based Taxonomies* In: *Conceptual Structures: Knowledge Architectures for Smart Applications*. Springer, Berlin / Heidelberg , (pp. 241-254).

Kwon, O. & Kim, J. (2009). Concept lattices for visualizing and generating user profiles for context-aware service recommendations, Expert *Systems with Applications*, Volume 36, Issue 2, Part 1, March 2009, (pp. 1893–1902).

Li, K., Du, Y., & Xiang, D. (2007). Collaborative Recommending Based on Core-Concept Lattice, *Theoretical Advances and Applications of Fuzzy Logic and Soft Computing*, *Advances in Soft Computing*, 2007, Volume 42/2007, (pp. 583-592).

Mao, Q., Feng,B., Pan, S., Zheng, Q. & Liu, J. (2010). New collaborative recommendation approach based on concept lattice, *Fuzzy Systems and Knowledge Discovery (FSKD)*, 2010 Seventh International Conference on , vol.4, no., (pp.1803-1807).

Nauer, E. & Toussaint, Y. (2007). Dynamical Modification of Context for an Iterative and Interactive Information Retrieval Process on the Web. CLA 2007

Olivarez-Giles, N. (2012). Does Samsung's Galaxy S III Smartphone Even Need Quad-Core Power?. Gadget Lab, Wired.com, May 3, 2012. http://www.wired.com/gadgetlab/2012/05/quad-core-vs-dual-core-phones-tablets-nvidia-samsung-galaxy/

Paganelli, F., Bianchi, G. & Giuli, D. (2007). A context model for context-aware system design towards the ambient intelligence vision: experiences in the etourism domain. In *Proceedings of the 9th conference on User interfaces for all*, ERCIM'06, pages 173–191, Berlin, Heidelberg, 2007. Springer-Verlag

Paridel, K., Yasar, A., Vanrompay, Y., Preuveneers, D. & Berbers, Y. (2011). Teamwork on the road: Efficient collaboration in VANETs with context-based grouping, The International Conference on Ambient Systems, Networks and Technologies (ANT-2011), *Procedia Computer Science*, volume 5, (pp. 48-57). Elsevier

Pastry : A substrate for peer-to-peer applications, www.freepastry.org

Petric, D. (2012). Quad-Core Smartphones Dominated Mobile World Congress 2012. Brighthand: Smartphone News & Reviews, March 01, 2012. http://www.brighthand.com/default.asp?newsID=18664&news=Mobile+World+Congress+MWC2012

Preuveneers, D., Victor, K., Vanrompay, Y., Rigole, P., Kirsch Pinheiro, M. & Berbers, Y. (2009). Context-Aware Adaptation in an Ecology of Applications, In: Dragan Stojanovic (Ed.), *Context-Aware Mobile and Ubiquitous Computing for Enhanced Usability: Adaptive Technologies and Applications*, IGI Global, 2009

Reichle, R., Wagner, M., Khan, M.U., Geihs, K., Lorenzo, L., Valla, M., Fra, C., Paspallis, N., Papadopoulos, G.A.. (2008). A Comprehensive Context Modeling Framework for Pervasive Computing Systems. In: Meier, R., Terzis, S. (eds.) DAIS 2008. *LNCS, vol. 5053*. Springer, Heidelberg (2008)

Reichle, R., Wagner, M., Khan, M.U., Geihs, K., Valla, M., Fra, C., Paspallis, N., Papadopoulos, G.A. (2008). A Context Query Language for Pervasive Computing Environments. In: 5th IEEE Workshop on Context Modeling and Reasoning (CoMoRea), *6th IEEE Int. Conf. on Pervasive Computing and Communication (PerCom)*, pp. 434–440 (2008)

Romero, D., Rouvoy, R., Seinturier, L. & Loiret, F. (2010). Integration of Heterogeneous Context Resources in Ubiquitous Environments, *36th EUROMICRO International Conference on Software Engineering and Advanced Applications* (pp. 123-126), ACM Press,

Szathmary, L., Valtchev, P., Napoli, A. & Godin, R. (2008). Constructing Iceberg Lattices from Frequent Closures Using Generators. In *Proc. of the 11th Intl. Conf. on Discovery Science* (DS '08), pages 136-147, Budapest, Hungary, Oct 2008.

Stumme, G., Taouil, R., Bastide, Y., Pasquier, N. & Lakhal, L. (2002). Computing iceberg concept lattices with TITANIC. *Data and Knowledge Engineering 42* (pp. 189-222).

Ventos, V. & Soldano, H. (2005). Alpha Galois Lattices: An Overview. ICFCA 2005 (pp. 299-314).

Wolfson, O., Xu, B. & Tanner, R.M. (2007). Mobile Peer-to-peer Data Dissemination with Resource Constraints, In *MDM '07 Proceedings of the 2007 International Conference on Mobile Data Management* (pp. 16-23)

Ye, J., Li, J., Zhu, Z., Gu, X., & Shi, H. (2007). PCSM: A Context Sharing Model in Peer-to-Peer Ubiquitous Computing Environment. In: *International Conference on Convergence Information Technology*, (pp. 1868–1873).

ADDITIONAL READINGS

Bartlang, U. (2010). Architecture and Methods for Flexible Content Management in Peer-to-Peer Systems. Vieweg+Teubner, Wiesbaden.

Ben Yahia, S., Mephu Nguifo, E., & Belohlavek, R. (Eds.). (2006). Concept lattices and their applications, LNCS vol. 4923, Springer.

Bolchini, C., Curino, C., Quintarelli, E., Schreiber, F. & Tanca, L. (2007). A data-oriented survey of context models. In: *SIGMOD Records* 36, 4, (pp. 19-26).

Buchholz, T., Hamann, T. & Hubsch, G. (2004). Comprehensive Structured Context Profiles: Design and Experiences. In: *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*. (pp. 43).

Lukasiewics, T. (1995). Uncertain reasoning in concept lattices. In: *Proceedings of the 3rd European Conference on Symbolic and Quantitative Approaches,* LNCS vol. 496 (pp. 293-300).

Paspallis, N., Rouvoy, R., Barone, P., Papadopoulos, G., Eliassen, F. & Mamelli, A. (2008). A pluggable and reconfigurable architecture for a context-aware enabling middleware system. In: *LNCS vol. 5331*, Springer Berlin Heidelberg, (pp. 553-570).

Schilit, B., Adams, N. & Want, R. (1994). Context-Aware Computing Applications. In: *IEEE Proceedings of the Workshop on Mobile Computing Systems and Applications* (pp. 85-90).

Szathmary, L. & Napoli, A. (2004). Knowledge Organization and Information Retrieval using Galois Lattices. In: *LNCS vol. 3257* (pp. 511-512).

KEY TERMS AND DEFINITIONS

Context: Any information that is considered to be relevant for the user and the application in a given situation.

Context distribution: The dissemination of context information in a network of nodes amongst context-aware applications.

Concept lattice: A formal representation of concepts in a domain consisting of objects having attributes.

MANET: Mobile Ad-hoc Network. A network of mobile nodes that organizes and manages itself.

Recommendation: The ability to recommend things of potential interest to users, mainly based on the interests of the user himself and on the interest of similar users.

Disseminator peer: Resource-rich device in a P2P network, responsible for managing, reasoning with, and distributing context information.

Ontology: A formal explicit specification of a shared conceptualization.

Adaptive system: A system that is aware of its context, and is able to react on changing context in order to express optimal behavior with respect to a goal.

Personalization: The customization of a system taking into account the characteristics, preferences and interests of a particular user.