Distributed Management of Contextual Affinities in Context-Aware Systems

Robert Schmohl and Uwe Baumgarten
Technische Univerität München, Department of Informatics
Garching bei München, Germany
schmohl@in.tum.de, baumgaru@in.tum.de

Abstract

In context-aware computing, distributed information about entities, such as people, places and objects, is captured and made available to applications, which utilize this context. With contexts from multiple entities available, the degree of alikeness of those contexts poses an interesting piece of information. We have tackled the issue of contextual proximity by employing mechanisms known from location-based services on conventional context-aware computing. This paper discusses the idea of the Contextual Map, a context model inspired by conventional map-models in location-aware computing. The Contextual Map represents contextual information in a multi-dimensional map model, where proximate representations of contexts imply affinity between those contexts. We consider entities, which compose such contexts, are potentially distributed and mobile. This highly dynamic environment puts our focus on the issue how those context carriers are distributed and which architectural conclusions are to be drawn.

1. Introduction

Context-aware computing focuses on utilizing any information describing the situation of an entity, which may be a place, a person, an object, etc. [5]. This information, forming the *context* of such an entity, is usually derived from its current surrounding, which is captured by sensors of devices associated with the entity, or the infrastructure hosting such devices [6, 7]. In this paper, we present a context model, which enables the detection of similar contexts, i.e. entities that share a similar situation.

Our approach to illuminate similarities among individual entities' contexts is inspired by the principles of managing proximity in the domain of location-awareness. For this purpose, we borrow proximity detection-mechanisms employed in Cartesian map models by leveraging the concept of *geographical proximity* to the context-aware computing domain. Geographical proximity in the location domain expresses that entities are close to each other. By projecting this setting on more general contexts, *contextual proximity* may express that contexts are alike or affine, hence "close" to each other in terms of their individual situations.

To enable contextual proximity management we have proposed the Contextual Map model [17]. By abstracting an entity's contextual attributes into a multi-dimensional map, we assign each entity's context a unique *position* in this map. Since the Contextual Map is settled in Euclidean space, we apply the Euclidean distance metric to identify contextual proximity among multi-dimensional contexts. Hence, we consider "close" contexts in the Contextual Map to be similar as we would consider close objects being geographically proximate in a location-based map model. The distance between contexts is equivalent to their similarity degree. Figure 1 sketches the idea behind the Contextual Map.

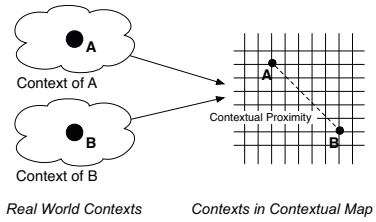

Figure 1. Contexts in the Contextual Map

The environment in which a context-aware system is applied may be populated by many mobile entities. For this reason, we explore the architectural requirements to employ the Contextual Map in distributed systems. We analyze the applicability of our context model in client-server systems, peer-to-peer and wireless sensor networks.

This paper explores the applicability of contextual similarity management in distributed environments. Section 2 enumerates the work related to our project. Subsequently, we introduce our Contextual Map model in section 3. Section 4 explains, how the Contextual Map is employed to

Digital Object Identifier: 10.4108/ICST.WICON2010.8517
http://dx.doi.org/10.4108/ICST.WICON2010.8517

monitor affinity between contexts and how to identify contextual proximity. Section 5 tackles the challenges on how to efficiently employ the Contextual Map model in a distributed environment. The paper is concluded in section 6.

2 Related Work

Lots of research is available in the domain of context-aware computing. Reviews about that domain include our generalized architecture derived from context-aware computing principles [15], a survey on context awareness in mobile environments, context management as a process [6], as well as a survey on context models [18].

An approach, which is similar to our work, is depicted by the theory of context spaces introduced by Padovitz et al. [10]. As the Contextual Map, context spaces aim at the multi-dimensional representation of contextual attributes. Multi-dimensional context spaces are partitioned into regions denoting bounds of specific contextual situations. Concrete context states mapped from real world entities are then associated to such context regions according to their proximity in the context space. The context space model is employed to maintain stability of contexts in context-aware systems by evaluating and refining uncertain contextual data, which is represented in proximity of each other in the context space [11, 12, 13]. Mapping contexts to the nearest regions removes uncertainty from erroneous contextual sensor data.

Roman et. al [14] have conducted similar work in bounding similarities in contextual information topologically. They define thresholds so that exceeding of those thresholds equals crossing a certain similarity degree.

Proximity and separation detection requires frequent location updates of the involved mobile hosts. Küpper and Treu [9] present update semantics how to keep the update count minimal and knowledge about object locations as current as possible. This is done by employing sophisticated algorithms with circular and strip-like update zones in 2-dimensional space.

3 The Contextual Map Model

In this section we introduce the Contextual Map model. After presenting the model's structural characteristics we focus on how contextual information is mapped into this model. To improve the reader's understanding, we are going to illustrate the working principle of the Contextual Map by employing an example, which exploits the context of a weather station.

3.1 Composition

The key idea of the Contextual Map model is to represent the context of entities in an n-dimensional realm. The context of a single entity corresponds to a single entry in the Contextual Map. Such an entry is composed of n coordinates and hence can be interpreted as a *position* in the map. Referring to our example, the context of our weather station denotes the current weather conditions at a specific location. Its context represents a single position in the Contextual Map, whereas the context of another weather station denotes another position.

An entity's context, captured by various context sensors, is mapped into the n dimensions of the Contextual Map by employing a particular *mapping function*. Basically speaking, this function represents an abstract converter translating heterogeneous contextual attributes to vectors represented as points in the Contextual Map. We are going to sketch such a function in section 3.2.

After mapping the information into the Contextual Map, the entity's current context is represented by n coordinates inside the map. Analogously, one can imagine a 3-dimensional map of space, which is employed to represent the *location* of entities in space. We extend this 3-dimensional model by adding additional dimensions, in order to include further contextual information other than location. Finally, each dimension corresponds to an elementary contextual attribute of a complete piece of context. In the Contextual Map, a single dimension is the smallest unit of context representation. E.g, a weather station's temperature readings can be mapped to one dimension whereas its humidity measurement can be mapped to another.

To further augment our model, we allow the Contextual Map to be partitioned into *ranges*. A range groups the map's dimensions, which share typological similarities. For instance, our weather station's location is a piece of context, decomposing into three subordinate building blocks corresponding to the location's Cartesian coordinates. In the Contextual Map, three dimensions can be dedicated for coordinate representation, grouping them to the range R_{loc} dedicated to the representation of locations. Since the weather station's context includes more than merely its location, additional ranges must be included. E.g., an additional range R_{env} may include the station's environmental readings with a single dimension being dedicated to each sensor (e.g. temperature, humidity, etc.). Practically speaking, ranges represent "submaps" of the Contextual Map. Since they are grouping contextual attributes type-specifically, positions of contexts in the Contextual Map are especially expressive on range level. E.g., the representation of R_{env} shows the contextual position of the current weather. In the rest of this paper, we refer to those range-specific contextual positions as *range contexts*.

Digital Object Identifier: 10.4108/ICST.WICON2010.8517
http://dx.doi.org/10.4108/ICST.WICON2010.8517

Ranges spawn a d-dimensional map with the dimensions as their axes. The quantification unit of a range must be uniform on all of its axes fitting the contextual attributes represented by that particular range (e.g. kilometers for R_{loc}). In the rest of this paper we use d to refer to a range's number of dimension, whereas n denotes the number of dimensions of the entire Context Map.

3.2 Context Mapping

Based on the context model introduced in the previous section 3.1, it remains to be clarified how contextual information is mapped into the Contextual Map. The *mapping function* inputs quantified contextual data from according context sources and maps them to Cartesian coordinates of the diverse ranges in the Contextual Map:

$$f_{mapping} : s \rightarrow \vec{v} = \begin{pmatrix} v_1 \\ v_2 \\ ... \\ v_n \end{pmatrix} | s \in S, \vec{v} \in V_n \qquad (1)$$

where S is the set of quantified context source data and V_n the correspondent n-dimensional realm in the Contextual Map. $\vec{v}$ represents a single context in the Contextual Map corresponding to the entity's real-world context. Due to the fragmentation of V_n into ranges, the definition of $f_{mapping}$ is basically comprised of mapping individual contextual data to the correspondent ranges in V_n. It is to be noted, that S may cover a very heterogeneous spectrum of contextual sources [16] implying heterogeneous definitions of $f_{mapping}$ as well. $f_{mapping}$ inputs the quantified context data and transforms it to its correspondent Cartesian n-dimensional representation to be stored in the Context Map.

Given our weather station example, we first define a representation for its context in the Contextual Map model. Subsequently, we define a mapping function, which defines how contextual data from context sensors is mapped into that map. We proceed as followed. First, we define two context types: location and environmental data. We assign each type a range with an according amount of dimensions in the Contextual Map, as shown in table 1. Consequently, we assign suitable quantification units to the dimensions of each range.

With this setting, it is possible to individually define an example mapping function $f_{mapping}$, which transfers the weather station's context into the Contextual Map. First of all, we require $f_{mapping}$ to handle each context type and its corresponding range separately. $f_{mapping}$ takes the quantified contextual data captured and derived from the weather station's context and calculates its position in the Contextual Map adhering both range and axis definitions. Mapping location into R_{loc} is quite straight forward: The station's coordinates are translated to Cartesian coordinates in the map. The coordinates for R_{env} are calculated by determining the relation of the current value to the pre-defined extremes. Let temperature to be allowed to adopt a value between -50°C and 50°C resulting in 100 adoptable units. With the axes of R_{env}'s dimensions ranging from 0 to 100, a currently measured temperature of 25°C corresponds to a coordinate of 75 for the temperature dimension. The mapping of the remaining environmental measurements works analogously.

Range	# of dimensions	Axis units
R_{loc} (location)	3 (x, y, z)	Cartesian coordinates (kilometers)
R_{env} (environmental readings)	4 (temperature, barometric pressure, humidity, wind speed)	0 (min) to 100 (max)

Table 1. Example ranges

Finally, we can derive $f_{mapping}$ for our example. Assume that s_{lat} and s_{long} depict the latitude and longitude, and s_{alt} the altitude from the location sensor, where s_{lat} and s_{long} are given in DMS-coordinates and s_{alt} is given in meters. Further, assume that s_{temp} depicts the temperature in degrees Celsius and s_{hum} the humidity in percent. The correspondent Cartesian coordinates in the Contextual Map are given by x,y and z for R_{loc} and $temp$ and hum for R_{env}. Assuming that $lat2cart$ and $long2cart$ transform longitude and latitude to Cartesian coordinates, respectively, we can finally define $f_{mapping}$:

$$f : \begin{cases} \begin{pmatrix} s_{lat} \\ s_{long} \\ s_{alt} \end{pmatrix} & \rightarrow \begin{pmatrix} x \\ y \\ z \end{pmatrix} \Big| \begin{array}{l} x = lat2cart(s_{lat}) \\ y = long2cart(s_{long}) \\ z = s_{alt}/1000 \end{array} \\ s_{temp} & \rightarrow temp \,|\, temp = (s_{temp} + 50°C)\frac{1}{°C} \\ s_{hum} & \rightarrow hum \,|\, hum = s_{hum} * 100 \\ \dots \end{cases} \qquad (2)$$

For reasons of simplicity we have omitted the remaining two dimensions of R_{env} (as stated in table 1).

4 Determination of contextual Affinities

So far, we have solely depicted the Contextual Map model in the previous section 3. This section focuses on the basic working principles of the Contextual Map handling contextual updates and identifying contextual proximity.

4.1 Affinity Detection with Boundaries

In this section a metric for monitoring contextual proximity is introduced. We present the concept of contextual

Digital Object Identifier: 10.4108/ICST.WICON2010.8517
http://dx.doi.org/10.4108/ICST.WICON2010.8517

boundaries, which represent the base for further applications of the Contextual Map.

Contextual boundaries represent the degree of alikeness between contextual information. More precisely, contextual boundaries may be defined between different entities to determine the affinity of their individual contexts. This affinity expresses how interesting or - contrarily - uninteresting another entity's context is. Concluding, context getting *closer* also raises its relevancy for the concerned entity [14].

For example, a car driver may be interested in proper weather conditions on his route, hence taking action if a proximate weather station reports strong weather. We define the contextual boundaries between the entities *car driver* and *weather station* on two ranges: R_{loc} and R_{env} (compare table 1). The weather station's physical position is represented in R_{loc}, its currently measured environmental readings are iteratively mapped to R_{env}. The car driver's context is represented by mapping his current location to R_{loc} and his personal conception about not-agreeable weather to R_{env}. We now aim at defining a context boundary, which delimits the driver's ability to drive from the situation of getting into extreme weather that keeps him from driving. The boundary concerning R_{loc} expresses a weather station's distance to the driver, so that it actually becomes of his interest. The boundary on R_{env} focuses on how close the current weather at a station may come to the weather conditions regarded as critical by the driver. As it can be observed, the boundary on both R_{loc} and R_{env} is represented by a "proximity threshold" defining proximity between two contexts and setting the boundary on the specific range. The contextual boundary between the entities *driver* and *weather station* gets crossed if the driver gets close to the weather station, which reports conditions exceeding the driver's critical criteria. Figure 2 illustrates the definition and crossing of this example boundary.

In summary, a context boundary defines a specific *degree of affinity* between different contexts. The boundary is composed of proximity thresholds defined individually on relevant ranges. Each threshold defines range-specfic degree of affinity concerning the range-specific context. Given this example, we can formally define contextual boundaries as followed:

- A context boundary B may affect several ranges $R_1, ..., R_m$. A *proximity threshold* t_i is defined on each of those ranges R_i. Each t_i is a single value dependent on all of the R_i's dimensions, hence, defining the degree of alikeness of R_i's contexts. Basically speaking, the threshold represents the *distance*, which delimits the range's contextual information to be "proximate" or not.

- B is defined by the set of thresholds $t_1, ..., t_i, ..., t_m$ of all ranges R_i relevant to B: $B = (t_1, ..., t_m)$.

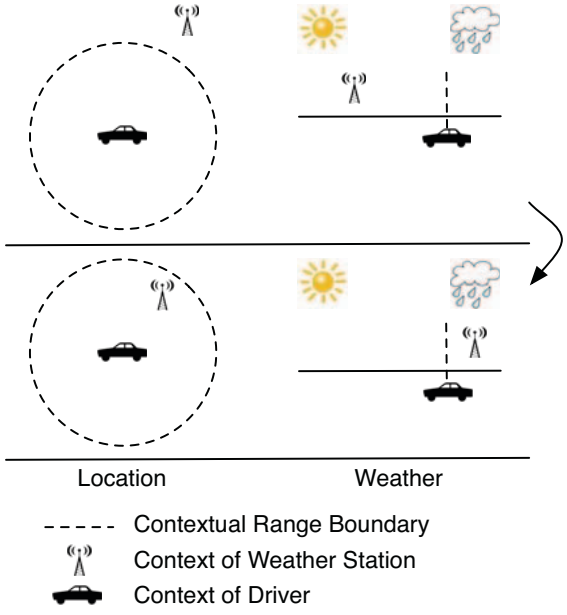

Figure 2. Crossing Context Boundary Example

With the principle of contextual boundaries defined, we are now about to discuss how contextual boundaries can be exploited using the Contextual Map model. As we have argued earlier, range contexts that are close to each other on range level are contextually affine in regard to the context attributes represented by that range. A context boundary may erect concrete delimiters for that context affinity by the definition of its range-specific thresholds. Hence crossing context boundaries corresponds to different entities' contexts getting either more ore less affine (or rather contextually "closer/farther" to/from each other), thus crossing the affinity degree specified by the boundary.

With context boundaries defined in a Contextual Map, the next step is to determine crossings of those boundaries indicating changes of contextual affinities. Since a context boundary's thresholds are each defined on a single range, the check for two contexts having crossed the boundary by converging or separating from each other is performed range-specifically. It is to be reminded that each range represents a d-dimensional "submap" of the Contextual Map. Thus, we proceed by calculating the range-level *distances* between the two contexts, hence determining the Euclidean distances between their range contexts on each range affected by the boundary. We proceed range by range, and check, whether the Euclidean distance between the two correspondent range contexts has fallen below or exceeded the threshold of the current range: Given d dimensions on a range R, and two contexts, P and Q, the Euclidean distance

Digital Object Identifier: 10.4108/ICST.WICON2010.8517
http://dx.doi.org/10.4108/ICST.WICON2010.8517

between the range contexts of P and Q on R is denoted by:

$$d_{\overline{PQ}}(R) = \sqrt{\sum_{i=1}^{d}(p_i - q_i)^2} \quad (3)$$

with p_i and q_i being the coordinates of dimension i of the range contexts P and Q, respectively. With the distance $d_{\overline{PQ}}(R)$ calculated, we employ proximity and separation detection [9] to determine, if the contexts of the two entities are "closing" or "separating" on range level. Consequently, the thresholds of a contextual boundary serve as proximity/separation thresholds to enable proximity or separation alerts. Such an alert is triggered when all of the boundary's thresholds have been breached, i.e. when the boundary has been crossed on each relevant range. The triggering of such proximity or separation alerts corresponds to the contexts P and Q getting either more or less affine, hence having crossed the boundary B.

It is to be noted that the detection of a context boundary crossing requires knowledge about distances between P and Q at two different times (before and after the crossing). Such determinations are made upon receiving an update from an affected entity. The corresponding update semantics, which have been extended to work in d-dimensional space, are discussed in section 4.2.

Returning to our weather station example, we assume that each weather station and the car driver get a constantly updated Contextual Map position associated with their respective context. The car driver has defined a contextual boundary defining dangerous weather at which he is not willing to drive (i.e. weather coming close to his critical weather definition). Hence this boundary is decomposed into two ranges R_{loc} and R_{env} as depicted earlier in figure 2: on R_{loc} the driver's tolerated distance to a weather station reporting bad weather, and on R_{env} the tolerance interval to weather conditions, at which the driver refuses driving. Let the boundary be further defined as 10 kilometers on R_{loc} and 15 units on R_{env} (see table 1). The Euclidean distances between the contexts of driver D and weather station W on each range are computed upon receiving a contextual update. In this example, such an update may yield changing weather conditions and/or changing driver's location. As soon as the driver gets closer than 10 kilometers to a weather station, which reports weather less than 15 units distant to the driver's critical weather definition, the defined context boundary is crossed and a proximity alert is triggered. Hence, the driver may take action changing his route or stop driving. This corresponds to the Euclidean distances $d_{\overline{DW}}(R_{loc}) = 10$ and $d_{\overline{DW}}(R_{env}) = 15$, which implies a boundary definition of $B = (t_{loc} = 10, t_{env} = 15)$.

4.2 Update Semantics

An obvious precondition for detecting proximity/separation of contexts in the Contextual Map is the knowledge about its most current positions in the map. Context actuality is achieved by contextual updates, i.e. updates containing an entity's most current context information. The challenge is to define efficient update semantics, so that a minimal number of updates delivers the most current context possible. E.g., frequent updates issued in short intervals deliver very actual context information, but are highly inefficient, since most of them are redundant due to missing context changes.

Besides polling, periodic and distance-based update semantics, zone-based strategies are of particular interest. In [9] a zone-based technique is presented allowing the acquisition of the most current context with a minimal number of location updates. With little effort, we can transfer those principles on the Contextual Map. The proximity and separation detection mechanisms in [9] focus on entities leaving or entering update zones based on circles and strips in 2-dimensional space. Extending those mechanisms on the Contextual Map model equals the extension of the update zone definitions to an arbitrary amount of n dimensions. This concludes the definition of n-dimensional hyperspheres and hyperplanes in the Contextual Map, both corresponding to the circular and strip-based update zones in 2-dimensional space, respectively. A thorough discussion about the definition of such update zone in n-space is presented in [17].

5 Aspects of Distribution and Mobility

Since the Contextual Map model handles dynamic contexts, it is clearly applicable to ubiquitous computing environments. Systems settled in this domain naturally exhibit a high degree of distribution and mobility. This section discusses the applicability of the Contextual Map in such distributed environments.

5.1 Context Model Distribution

In a distributed system, the Contextual Map itself can be deployed differently on the various distributed components. For this purpose, we have defined three specifications of the context model, which basically differ in the scope of an entity's contextual vicinity in the map.

- *Local Map:* A local Contextual Map solely focuses on a single entity's context. Since there is no information about other context in the map, it only serves as a data structure for determining the need of triggering contextual updates by monitoring the entity's context position in an update zone in the local map (see section 4.2).

Digital Object Identifier: 10.4108/ICST.WICON2010.8517
http://dx.doi.org/10.4108/ICST.WICON2010.8517

- *Vicinity Map:* This specification is centered around the entity's context and encompasses all of the other entities' contexts, which are currently known. This set of contexts literally defined the entity's known contextual vicinity.
- *Global Map:* The global Contextual Map is not centered around single entities, but encompasses all contexts of entities known by the entire system. Thus, a global map denotes a system's global context.

Figure 3 illustrates the three specifications with the local map missing any other contexts in contrast to the other map specifications. Comparing those three alternatives, one can observe a direct dependency between expressiveness of the map model and its required capabilities concerning calculation and storage. The more expressive a context model is the more capabilities it requires on its host, as expressed in table 2.

Map Model	Expressiveness / Contextual Compass	Capability Requirements
Global Map	high, encompasses current context of entire system	high (store and analyze entire context)
Vicinity Map	moderate, only neighboring context is visible and current	moderate (confined to neighboring context)
Local Map	low, none but own context available	low (monitor own context for update determination)

Table 2. Contextual Map Variants

It is crucial to keep in mind that our discussion about distribution of entities focuses on techniques related to communication, updating and deployment implied by distributed systems. In our discussion, the location data implied by distribution does *not* play any role for any contextual information. The emphasis of this entire section is the focus on transportability and deployment of context information rather than processing it context-awarely (including location). Context processing has already been discussed previously in section 4.

5.2 Distribution Aspects

From here on, we discuss the applicability of the Contextual Map model on diverse distributed architectures. In particular, we focus on the client-server paradigm, peer-to-peer networks, hybrid peer-to-peer systems and wireless sensor networks (figure 4). We have identified the following aspects being of importance:

- *Map distribution:* With the global map providing the richest amount of information, it is naturally not generally applicable to distributed systems. According to the restrictions implied by distribution, the less capable map specifications may have to be employed. E.g., in a resource-constrained system, a vicinity map or even a local map may have to be used.

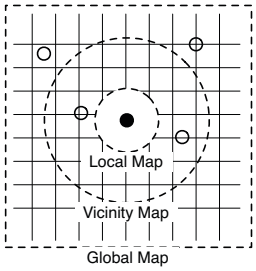

Figure 3. Distribution of the Contextual Map

- *Update Semantics:* The Contextual Map complies with the deployment in highly dynamic environments, requiring contextual updates to provide the overall system with current contextual information. The semantics for the consequently required contextual updates have already been discussed in section 4.2. We are about to enrich this discussion with the applicability to distributed system architectures.
- *Capabilities:* The presence of mobile hosts implies the existence of significant hardware restrictions [19]. Since the Contextual Map model requires a certain level of performance it may not be able to be deployed on any available host.

In the rest of this section, we focus on scenarios, where entities are attached to a device committing their contexts to the repository implementing the Contextual Map model. The entities are explicitly thought to be mobile implying the usage of mobile devices with restricted capabilities.

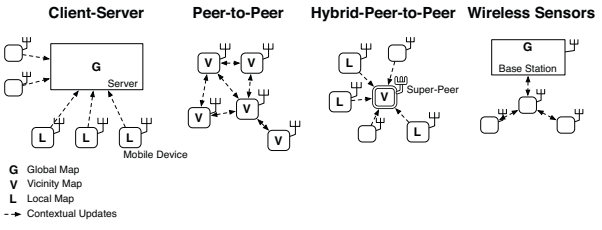

Figure 4. Distributed Architectures

5.3 Client-Server Paradigm

This scenario encompasses a server, where all of the contextual information from mobile clients converges.

Digital Object Identifier: 10.4108/ICST.WICON2010.8517
http://dx.doi.org/10.4108/ICST.WICON2010.8517

Map distribution Since the context management occurs on the server, it is suitable to employ a global map there. It provides a complete snapshot of the entire context situation of the covered area. Clients may be equipped with a local map to enable efficient updates. However, if required by application cases, any deployment on the clients can be avoided as well in order to render the clients as lightweight as possible (see figure 4).

Update Semantics Given a local map on the client device, an update zone in the Contextual Map can be defined as the metric for defining context being sufficiently different to justify an update (see section 4.2). Alternatively, given a lightweight client missing a local map, one must fall back on conventional methods, particularly polling updates and periodic updates.

Capabilities The omnipotent server allows any deployment thought necessary by corresponding use cases. Restrictions only apply to mobile clients, which may not be able to fully support the deployment of a local Contextual Map. However, those less potent clients devices may still be integrated into this architecture neglecting the zone-based update semantics in n-space.

5.4 Peer-to-Peer Networks (P2P)

Pure P2P-networks lack any central components for coordinating interoperability and context acquisition. Hence, in our discussion we focus on mobile peers roaming in wireless networks.

Map distribution The lack of a central instance in a P2P-based system makes it impossible to employ a global context representation, forcing the utilization of map specifications other than the global map. Further, the lack of a global context representation requires the peers to have information about other contexts. Hence, peers build up a Contextual Map centered around their own context by exchanging information about other contexts with neighboring peers (see figure 4). This process reflects the local maintenance of a map about contexts inside a peer's known vicinity. Accordingly, the vicinity context map is the map model of choice. As a consequence, the global context is represented by the sum of all peers' vicinity maps[1]. None of the peers is able to construct a global and current context representation in its map model. As a consequence, no implications on global context can be conducted in the entire network. Assuming that peers exchange context information not only about themselves but about their neighborhood as well, i.e. committing their entire vicinity in updates, each peer receives context information from other non-neighboring peers over time. This may occur, since context information from remote peers can "dig through" to a peer by being passed in multiple updates. Eventually, a peer may receive complete context information covering an area, which is much larger than its neighborhood. However, it takes time to receive remote context information due to the numerous update iterations. For this reason, remote context may be outdated once it arrives. It seems reasonable to let a peer decide when to prune outdated context, so that it is neither included in its vicinity map nor committed further to neighboring peers.

[1] more precisely, the global context is represented by the sum of local contexts, since other contexts in vicinity maps may not be current

Update Semantics The strategies of efficiently determining suitable times for issuing contextual updates are only applicable, if the update's recipient is always available. In P2P-networks, this is not the case. For this reason, the update semantics have to be restricted to exchanging updates between peers during their periods of interaction between each other. Since those connections between peers are usually transient, updates are committed upon establishing *contact* with another peer. Those "initial updates" may not only include the own context, but also a set of all relevant other contexts. This is reasonable, since the update's recipient may be unable to determine the other entity's contexts included in the update. Finally, the update strategies encompassing polling, periodic updates and the zone-based updates in n-space complement the update semantics for peers during their interaction periods.

Capabilities In this setting, we face a set of mobile peers with restrained capabilities, which comprise the entire system. However, the mobile devices need to provide sufficient computational power and storage. Both are necessary for updating and hosting a vicinity map. Devices, which do not excel the minimal requirements cannot participate in this system. Alternatively, they can be employed in a hybrid-P2P system (see next section 5.5).

5.5 Hybrid Peer-to-peer Networks

H-P2P-networks [8] share the same layout as their pure P2P-counterparts. The distinguishable difference is the existence of *super-peers* in H-P2P architectures. Those are dedicated mobile nodes, which function as regional coordinating components within the network. Although exhibiting slight client-server techniques, super-peers are only accessible by peers in reach.

Map distribution The existence of super-peers allows to apply a light-weight client-server approach (discussed in section 5.3) to the H-P2P architecture. Super-peers are deployed with a vicinity map, as their regular counterparts

Digital Object Identifier: 10.4108/ICST.WICON2010.8517
http://dx.doi.org/10.4108/ICST.WICON2010.8517

in the pure P2P-architecture (section 5.4). With the super-peers equipped with a vicinity map, it suffices to use local maps on the remaining peers (see figure 4). The super-peers actually act as they were servers for all peers in reach. As with pure P2P-networks, a global context representation is not available to any peer or super peer. An approximation of larger areas beyond neighboring peers is only available to super peers under the restrictions of context actuality as discussed previously in regard to pure P2P-networks.

Update Semantics In H-P2P-networks, contextual updates are propagated analogously to P2P-networks, except that regular peers do not receive contextual updates. With a local map, they are unable to process other contexts but their own.

Capabilities Basically, the same restrictions impacting the design of P2P-networks are applicable to H-P2P as well. However, since local maps are less resource-consuming, capability requirements on regular peers are less demanding. As we have stated earlier already, it is even possible to employ peers without any Contextual Map deployment, hence minimizing the capability demands on those. However, the latter minimization efforts come with the cost of neglecting the zone-based update semantics in n-space.

5.6 Wireless Sensor Networks (WSN)

In mobile WSNs, highly mobile sensor nodes operate with minimal hardware capabilities. Those nodes are usually capable of processing simple tasks, storing several bytes of information and transmit them wirelessly to a server component, namely the base station. Usually, the base station needs to be in reach of sensor node for transmission, since the node's range is limited due to its hardware restrictions.

Map distribution With the base station attached to a capable machine, it can maintain a global map for the entire system. The sensor nodes, however, do not have the capability to host any context model (see figure 4).

Update Semantics and Capabilities Due to minimal hardware and transmission range of the sensor nodes, only updates upon contact with the base station can be issued. Given the case that sensor nodes are able to communicate among each other, updates can also be exchanged upon contact between nodes. This approach implies that nodes are capable of storing contextual information other than their own, which they forward to other nodes upon contact. Hence, those foreign updates are not processed, but stored and readied for transmission to the base station. Because of the limited storage capabilities, the buffer for foreign updates may be full before reaching the base station. In this case, it is prudent to erase the oldest updates, since they most likely contain the most obsolete context information. However, dropping aged context information may result in some nodes' context data not reaching the base station at all. As a consequence, the global map at the server may not have an encompassing representation of the global context. In addition, the nodes' context data may take a considerable time to get to the base station and this into the global map. The actuality of context is further degraded by this circumstance. For this reason, the global map's capabilities in WSNs have to be diluted from the observation in table 2. Although the base station's global map aims at constructing current and all-encompassing context, it cannot be guaranteed due to the discussed architectural restrictions.

Architecture	Map Distribution	Update Semantics	Capabilities
Client-Server	1 Global, ∞ Locals	update zones, periodic, polling	potent
P2P	Vicinity (∞),	on contact	limited
H-P2P	Vicinity (∞), Local (∞)	on contact	limited
WSN	Global (1), none (∞)	on contact	minimal

Table 3. Comparison of Architectures

5.7 Deployment

The issue of deployment concerns individual mobile devices, which employ one of the Contextual Map options, i.e. either the vicinity map or its local counterpart. To run such a component of the mobile device, the most suitable way is to deploy it into the device's middleware platforms (e.g. J2ME [1]) or as middleware itself. The Contextual Map then acts as a middleware between context-aware applications and the device's enabling operating system.

In order to function appropriately, the employment of such a middleware needs to concur with following aspects:

- The *hardware capabilities* of the carrier device needs to comply with the middleware's hardware requirements. This aspect applies to the case when the Contextual Map is employed as stand-alone middleware. If embedded into existing middleware, the device usually possesses sufficient capabilities of supporting the hosted middleware.
- The *access to context sources*, which are controlled by the operating system needs to enabled and maximized. This includes on-device sensors as well as context sources outside the device (most likely connected

Digital Object Identifier: 10.4108/ICST.WICON2010.8517
http://dx.doi.org/10.4108/ICST.WICON2010.8517

wirelessly). The more fine-grained access to context sources is available the more accurate the context monitoring is.

6 Conclusion and Outlook

With the Contextual Map, we have proposed a context model concept facilitating the work with contextual affinity in distributed environments. The definition of context boundaries allows to determining contexts getting more or less affine, respectively, i.e. changing their degree of "similarity". In this paper, we have evaluated the deployment options of the Contextual Map in distributed environments. This evaluation allows further steps to be taken to employ the Contextual Map model in pervasive computing environments.

Currently, the Contextual Map is undergoing prototypic implementation in a J2EE-based client-server environment. In our mid-term roadmap we plan to port the functionality to mobile APIs, such as Symbian [3], Android [4] or Windows Mobile [2]. Deploying the Contextual Map on mobile devices will allow us to use it in a mobile P2P-network. In the end, this course of action will allow us to analyze issues on how to integrate the context model in proven context-aware systems.

References

[1] The java me platform. http://java.sun.com/javame/.

[2] Microsoft windows mobile. http://www.microsoft.com/windowsmobile/, June 2008.

[3] Symbian developer network. http://developer.symbian.com/, June 2008.

[4] Google android. http://developer.android.com/, June 2009.

[5] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles. Towards a better understanding of context and context-awareness. In *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pages 304–307, London, UK, 1999. Springer-Verlag.

[6] E. Christopoulou, C. Goumopoulos, and A. Kameas. An ontology-based context management and reasoning process for ubicomp applications. In *sOc-EUSAI '05: Proceedings of the 2005 joint conference on Smart objects and ambient intelligence*, pages 265–270, New York, NY, USA, 2005. ACM Press.

[7] H. W. Gellersen, A. Schmidt, and M. Beigl. Multi-sensor context-awareness in mobile devices and smart artifacts. *Mob. Netw. Appl.*, 7(5):341–351, 2002.

[8] E. Harjula, M. Ylianttila, J. Ala-Kurikka, J. Riekki, and J. Sauvola. Plug-and-play application platform: towards mobile peer-to-peer. In *MUM '04: Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, pages 63–69, New York, NY, USA, 2004. ACM Press.

[9] A. Küpper and G. Treu. Efficient proximity and separation detection among mobile targets for supporting location-based community services. *SIGMOBILE Mob. Comput. Commun. Rev.*, 10(3):1–12, 2006.

[10] A. Padovitz, S. Loke, and A. Zaslavsky. Towards a theory of context spaces. *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*, 1:38–42, March 2004.

[11] A. Padovitz, S. W. Loke, A. Zaslavsky, and B. Burg. Stability in context-aware pervasive systems: A state-space modeling approch. In *Proceedings of the 1st International Workshop on Ubiquitous Computing*, 2004.

[12] A. Padovitz, S. W. Loke, A. Zaslavsky, and B. Burg. Towards a general approach for reasoning about context, situations and uncertainty in ubiquitous sensing: Putting geometrical intuitions to work. In *2nd International Symposium on Ubiquitous Computing Systems (UCS2004)*, 2004.

[13] A. Padovitz, A. Zaslavsky, S. W. Loke, and B. Burg. Maintaining continuous dependability in sensor-based context-aware pervasive computing systems. *System Sciences, 2005. HICSS '05. Proceedings of the 38th Annual Hawaii International Conference on*, 1:290a–290a, Jan. 2005.

[14] G.-C. Roman, C. Julien, and Q. Huang. Network abstractions for context-aware mobile computing. In *ICSE '02: Proceedings of the 24th International Conference on Software Engineering*, pages 363–373, New York, NY, USA, 2002. ACM Press.

[15] R. Schmohl and U. Baumgarten. Context-aware computing: a survey preparing a generalized approach. In *IMECS 2008: Proceedings of the International MultiConference of Engineers and Computer Scientists 2008*. International Association of Engineers, 2008.

[16] R. Schmohl and U. Baumgarten. A generalized context-aware architecture in heterogeneous mobile computing environments. *Wireless and Mobile Communications, 2008. ICWMC '08. The Fourth International Conference on*, 1:118–124, 27 2008-Aug. 1 2008.

[17] R. Schmohl and U. Baumgarten. The contextual map - a context model for detecting affinity between contexts. *MobileWireless Middleware, Operating Systems, and Applications*, 1:171–184, 2009.

[18] T. Strang and C. Linnhoff-Popien. A context modeling survey. In *Proceedings of the Workshop on Advanced Context Modelling, Reasoning and Management associated with the 6th International Conference on Ubiquitous Computing (UbiComp), Nottingham.*, 2004.

[19] R. Want and T. Pering. System challenges for ubiquitous & pervasive computing. In *ICSE '05: Proceedings of the 27th international conference on Software engineering*, pages 9–14, 2005.

Digital Object Identifier: 10.4108/ICST.WICON2010.8517
http://dx.doi.org/10.4108/ICST.WICON2010.8517