Optimising context data dissemination and storage in distributed pervasive computing systems

Ioanna Roussaki\textsuperscript{a,}\textsuperscript{*}, Maria Strimpakou\textsuperscript{a}, Carsten Pils\textsuperscript{b}, Nikos Kalatzis\textsuperscript{a}, Nicolas Liampotis\textsuperscript{a}

\textsuperscript{a} National Technical University of Athens, School of Electrical and Computer Engineering, 9 Heroon Polytechniou Street, 15773 Zographou, Athens, Greece
\textsuperscript{b} Waterford Institute of Technology, Telecommunications Software \& Systems Group, Waterford, Ireland

ABSTRACT

Context management systems are expected to administrate large volumes of spatial and non-spatial information in geographical disperse domains. In particular, when these systems cover wide areas such as cities, countries or even the entire planet, the design of scalable storage, retrieval and propagation mechanisms is paramount. This paper elaborates on mechanisms that address advanced requirements, including support for distributed context databases management; efficient query handling; innovative management of mobile physical objects and optimization strategies for distributed context data dissemination. These mechanisms establish a robust spatially-enhanced distributed context management framework that has already been designed and carefully implemented and thoroughly evaluated.

© 2009 Elsevier B.V. All rights reserved.

doi:10.1016/j.pmcj.2009.07.015

1. Introduction

Network operators, sensor networks or even web resources capture valuable information such as device location and status, user profiles and movement patterns, network performance, etc., in order to provide enhanced telecommunication services to their clients. This data is considered to be vital context information \[1\] that can be exploited to customize services, to anticipate user intentions and to ultimately reduce human-to-machine interactions \[2\]. However, even though context information holds out the prospect of enhancing user experience and increasing revenues, disseminating it across distributed nodes is not straightforward. On the one hand, the various infrastructures that store and manage context data are heterogeneous, while there is no standardized interface that supports context information exchange. On the other hand, information, which cannot be retrieved when necessary, is valueless. In an open context marketplace, where a wide variety of information types is traded, context consumers are challenged by the discovery of the required context data. In this perspective, timely delivery of context information is crucial, due to the fact that most data sources provide real-time information. Thus, efficient approaches for distributed storage, retrieval and timely delivery of context data are essential for the success of context-aware computing systems. In particular, when context aware systems are not confined to a single building, but extend their functionality to cities, countries or even the entire planet, scalable and efficient data distribution mechanisms are paramount.

In order to support the provision of personalized and context aware services, irrespective of the user device and the underlying network, information originating from multiple context providers is usually required \[3\]. These context providers do not necessarily belong to the same legal entity or administrative domain. In such a federated context management
environment, user context information is a commodity that is protected by the administrative domains that possess it and therefore, stakeholders will hardly agree on a single Database Management System (DBMS). Though the stakeholders do not share context information, users have access to such information in case they have subscribed for it. This is, however, not a hard constraint. On the one hand, most DBMSs are relational databases that implement the SQL standard. On the other hand, standardizing the basic database structure is inevitable for the support of universal and efficient information exchange. Thus, a single virtual database that extends across all CM nodes must be in place. In this perspective, a Context Distributed Database Management System\(^1\) (CDDBMS) should be established that would be responsible for processing context information as if this is maintained by a single database, while in fact the context information is stored and controlled by multiple administrative domains. Thus, the CDDBMS should provide the same functionality as a centralized database management system, making transparent to the context consumer that distributed query handling and interaction with remote context nodes are required.

The heterogeneity of context information imposes heavy requirements on the CDDBMS supporting pervasive systems. Undoubtedly, the most important and, at the same time, most critical context information that needs special treatment is location information. As users move across the globe, context-aware systems need to establish scalable mechanisms for ubiquitously supporting users irrespectively of their mobility pattern and current location. The introduced CDDBMS system should support effective mechanisms for dealing with mobile physical objects with regards to handling frequent updates of location information and is inspired by the mobile IP protocol. This paper initially focuses on presenting the details and advantages of the implemented CDDBMS.

Even though location is very critical in context-aware systems, there is a very wide variety of context information that is also necessary. In a pervasive world with millions of moving users and billions of interacting devices, an immense amount of heterogeneous context data is likely to be requested for delivery to remote nodes every second. In such an environment, apart from a quite small part of context data that will be strictly available to its owners (private), most context information will be a commodity to be traded among either a restricted consumers’ group (semi-private) or frequently, among a large consumers’ group (public). Even a well designed, distributed and scalable context management infrastructure would eventually fail in answering such continuously expanding demands for context information across remote context nodes. Especially, in cases where the requested context information constitutes a frequently updated piece of context data (e.g. room temperature) that should be conveyed to the client each time a new context value is available, the context management system, as well as the underlying network will be eventually flooded with the continuous context data updates to remote consumers. Each context consumer will have different requirements with regards to the context updates he/she is willing to receive, e.g. he/she might desire to be informed once every day for the room temperature and not every second, thus making things even more complicated. Obviously, it is insufficient or even impossible to address each user’s needs in full, anytime and anywhere. But if a group of users is interested in the same dynamic context information, sophisticated context dissemination mechanisms should be employed that will aim to simultaneously satisfy the entire group, although taking into consideration the individual characteristics and requirements of each user. The design and evaluation of such advanced context dissemination mechanisms is the main focus of this paper.

The rest of the paper is structured as follows. Section 2 elaborates on the Context Distributed Database Management System, briefly introducing the functional architecture of the established distributed context databases including the database schema, the supported query languages, as well as details about the mobile physical objects management. Section 3 presents an optimized context data dissemination mechanism caters for the cost-minimizing selective update of context replicas distributed in a multitude of context nodes throughout the network. More specifically, it focuses on establishing the general problem description, while also providing the formal problem statement and a brief review of the most relevant research work. Section 4 proposes heuristics algorithms for solving the context data dissemination problem distinguishing between four cases that are thoroughly described. Section 5 analyses in detail the conducted experiments and presents the obtained evaluation results. Section 6 performs a comparison of the proposed approach with the relevant state of the art work. Finally, in Section 7 conclusions are drawn, while an outline of the current status and future plans is provided.

2. The Context Distributed Database Management System

The CDDBMS is a peer-to-peer database comprising of node servers, each of which stores information about at least one predefined information domain. Such domains include for instance: user profiles, accounting information of a telecommunication operator, service advertisements or information related to a specific geographical area. Context producers store their data with respect to domains and consumers query information accordingly. Two domains are distinguished: Logical Domains (Logic-D) and Geographic Domains (Geo-D). Logic-Ds contain non-spatial information such as user profiles and are rather independent from other domains. Geo-Ds, however, carry strong spatial inclusion relationships. For example, the suburb is a geographic specialization of the city domain and thus, there is an inclusion relation between the City-Node and the Suburb-Node. Inclusion relations span a directed tree on the Geo-D space, where

\(^1\) A DDMS is a software system that permits the management of the distributed database and makes the distribution transparent to the users. A distributed database is a collection of multiple, logically interrelated databases distributed over a computer network [4]. Distributed database management systems are typically implemented as client-server systems or peer-to-peer systems [5].
Fig. 1. CDDBMS high-level deployment and architecture.

each CDDBMS is a node and each inclusion relation is an edge. This graph structure perfectly matches an R-Tree [6], an indexing structure of spatial databases. As a result, the geographic inclusion dependencies imply a structure which perfectly caters for spatial queries [7].

The infrastructure exploited by the stakeholders of the context marketplace is heterogeneous. Different database management systems, which store context information, are deployed. Though, most databases implement the SQL standard, they provide different extensions or do not fully comply with it. Moreover, these systems do not offer federation facilities and the lack of a standardized database schema makes it almost impossible to discover the entries queried. To solve this problem, the CDDBMS nodes implement both a standardized database schema and a standardized query interface.

2.1. The CDDBMS architecture

Fig. 1 illustrates the CDDBMS architecture. It comprises multiple independent CDDBMS nodes that communicate via the internet to exchange query statements and result sets. Each CDDBMS node comprises a Node-Manager and an off-the-shelf database management system (DBMS). The Node-Manager implements the management logic and the interfaces required for exchanging information with other nodes. Synchronous and asynchronous communication of context data is supported by the Query and the Event interfaces. When a CDDBMS node receives a request for manipulating information referring to data items stored in the local repository the procedure is straightforward. When a data item is not stored in the local DBMS it has to be retrieved from a remote node. An efficient look-up mechanism for finding the item is essential for the scalability of the CDDBMS. To this end the DBMS implements a simple look-up mechanism: Each data item is associated with a Home Node which acts as a master repository of the item. The Home Node actually guarantees that it stores the item and that client Nodes can retrieve the item from it. The (communication) address of the Home Node is encapsulated in the identifier of the data item. This distribution concept is inspired by the Home Location Register (HLR) and Visitor Location Register (VLR) approach of the GSM user profile database. Likewise, the Home Node is also used for consistent data updates. All updates must be processed by the Home Node. As long as a data item is not updated at the Home Node, the update is not valid.

The DBMS is the actual repository of the node’s context data. Basically any off-the-shelf relational database engine meets the requirements of the CDDBMS. However, as spatial queries are a major feature of the CDDBMS, all CDDBMS nodes that cover a Geo-D must implement the Simple Feature Standard SQL (SFS) of the Open Geospatial Consortium (OGC) [8]. This standard specifies SQL-like spatial query statements and 3-dimensional shapes that are used to describe geographic areas. All major database management systems, such as Oracle (www.oracle.com), DB2 (www-306.ibm.com/software/data/db2), PostGis extension of PostgreSQL (www.postgresql.org) or MySQL (www.mysql.com), are shipped with an SFS extension.

2.2. Database schema and query language

As the information traded at the context marketplace is heterogeneous, only a very simple database schema, which is yet perfectly aligned to the major query uses cases, suits the CDDBMS. Key-value pairs, hereafter called Attributes, are the simplest form of storing context information and constitute the core of CDDBMS schema. Attributes are grouped by Entities that are interconnected by Associations [9]. The CDDBMS is thus quite similar to the World Wide Web: Attribute sets correspond to the content of a web page, Entities to web pages and Associations to sets of hyperlinks. This design perfectly caters for two major query use cases.

- **Entity-based queries/Browsing:** Starting from a known context data item, applications (or context consumers in general) can follow Associations to discover new information, i.e. Entities and Attributes. For example, when a user’s entity is known, an application can follow the “myPreferences” Association to discover the user’s service preferences or the “deviceUsed” association to identify the device currently utilized by the user. Entity-based queries require previous knowledge about the content of the database (the context data item entry) and its structure (Associations). For more details on the manner navigational queries are handled by the CDDBMS you may refer to [3].

- **Spatial queries:** As spatial inclusion relations can be directly mapped to Associations, navigational queries can also be used to look-up spatial data. However, the burden of evaluating spatial information is put on the query client. A query client must just provide a spatial description. The databases analysis this description and returns the data accordingly. To this
end, the distributed spatial query processing algorithm utilizes inclusion relations. As already mentioned, spatial queries are supported by the SFS standard of the OGC [8] that is used for describing 3-dimensional shapes and geographic areas.

Like the database schema, the query language for executing both query types is also inspired by the web technology. Each context Model Object is identified by a URL structured as follows:

\[
\text{NDQL://hostname:port/database/ModelType/type/number}
\]

The language is designed to provide a view on the database which supports the integrity of the schema. An NDQL projection/selection query is expressed as follows:

\[
\text{GET Entity FROM } [\text{assoc-id}] \text{ WHERE Association type=}[\text{entity-type}] \\
\text{GET Entity FROM LOCATION=}[\text{coordinates}] \text{ WHERE sem=}[\text{OWL-desc}]
\]

The first query is an example of an entity-based query and looks-up all Entities of type [entity-type] that belong in the association-set of Association [assoc-id]. The second query is an example of a spatial query and looks-up all Entities that match the specified OWL description [OWL-desc] [10] and are located in the spatial area [coordinates]. Note, that for the first query the reference to Home-Node is encapsulated in the Association identifier, while for the second it is described by the Location constraint (actually this might refer to multiple Nodes). Modification queries, i.e. inserts, updates and deletes, have the following format:

\[
(\text{UPDATE} | \text{ADD} | \text{DELETE}):[\text{reference}]
\]

where [reference] is a reference to an instantiated context Model Object [10].

2.3. Dealing with mobile physical objects

The CCDBMS is designed to manage spatial models of the physical world that cover large areas or even the entire planet. To this end, the nodes, Geo-Ds respectively, are ordered in an R-Tree structure. Fig. 2 illustrates such a structure. Each Geo-D, depicted on the right side, is associated with a node (corresponding domain node) of the distributed spatial database. A domain context data object (or simply context entity according to the context database schema presented briefly in the previous section) stored at each node specifies the geographical coverage. When a node receives a query, it first checks whether one of the queried context entities is within the coverage domain, Geo-D respectively. If the query is relevant to the domain, either the corresponding context data items are instantly retrieved, or the query is passed on to the appropriate child Geo-D. If the query is not relevant to the Geo-D, it is passed on to the parent node (given it did not originate there). Finally, it traverses the spatial CDDBMS node hierarchy down to the responsible Geo-D Nodes. As more than one Node might be responsible for the target Geo-D, the query might be duplicated and forwarded to all responsible Node-Managers. Result sets are directly forwarded to the requesting Node-Server, which is also responsible of orchestrating join operations.

This hierarchy is not without consequence to the storage approach. Objects (or Entities, as referred to in the remainder of the paper) must be maintained by the node that manages the Geo-D where the entity is currently located, or at least in one of the top level Geo-D nodes (yet, the latter case has an impact on the query process). Moving data items between domain nodes does no harm as most changes are performed within local network domains. Yet, any changes to mobile objects must be synchronized with the master copy at the Home Node. Frequent synchronizations of master copies might increase the network traffic considerably and constitute a serious problem to the scalability of the system.

At present, location sensors (i.e. context sources capturing location information) do not provide the location of a user. They rather provide coordinates of a device a user is using. Without knowing which device is being used, it is impossible
Fig. 3. Chain of pointers for mobile physical object handling.

to determine the user location. Devices run a CDDBMS Node Server that must be attached to the hierarchy of the spatial database. A mobile device’s Node Server will be hereafter referred to as a Mobile Node Server or just Mobile Node. The Geo-D of a Mobile Node is the mobile device itself. For small devices this is just a logical domain, but for large devices such as cars, aircrafts or even ships, Geo-Ds cover geographic areas, which are however mobile with regards to the outside world. The current context of a mobile device is stored in its Mobile Node Server. Thus, finding a mobile device is equivalent to finding its Mobile Node Server. The approach is depicted in Fig. 3. Each Mobile Node is represented by a Home Entity in its Home Node and by a Visited Entity in every Geo-D it visits. The Home Node is a Node Server in the fixed network; thus the route to this server never changes. When a Mobile Node moves into a different Geo-D, a Visited Entity is created at the Node that manages the domain. The Visited Entity inter-links the Mobile Node and the Home Node. Processing of non-spatial statements is thus similar to mobile IP. The Home Node re-directs query statement to the Visited Parent, which finally dispatches it to the Mobile Node.

As the Home Node stores the Master Copy of the Entity that models the Mobile Node, all statements related to the Mobile Node are sent to the Home Node first. The latter looks-up the Entity and extracts the URL of the Visited Node or one of the Visited Nodes (pointer chain). Since statements are directed to the actual Mobile Node, data synchronization with the Home Node is no longer necessary. To minimize the number of Home Node updates even further, Parent Nodes of Visited Nodes might carry Pointer Chain Entities. These context entities establish a pointer chain from the Home Node to the Visited Node. Thus, when a Mobile Node changes its Visited Node only the joint Parent of the new and the old Node must be updated (rather than the Home Node). The pointer chain is sound and the Home Node still maintains a valid (indirect) pointer to the actual Mobile Node.

When a Mobile Node Server changes the Geo-D, the routing entries, the pointer chain respectively, must be updated. To this end, Node Managers (A and B) implement a simple handover procedure. Consider for example, two CDDBMS Nodes that manage adjacent Geo-Ds, where each Geo-D has its own location sensor. Initially, user X has just a location entry at Node A, the location value is frequently updated by the location sensor of Geo-D A. When the user moves into Geo-D B and thus gets into the sensor range of this domain’s location sensor, a new location entry is created in Geo-D B. Both location entries (at Node A and B) might virtually contain the same sample data, but since different location sensors have captured the samples an aggregation of the two is not possible (at least not to the CDDBMS). When user X leaves Geo-D A and thus gets out of its sensors’ range, the location sensor might signal that the location entry is invalid. The CDDBMS Node removes it, and the handover is then completed. If the location sensor fails to signal that the entry is invalid, the location entry is removed, when no updates are received within a predefined timeframe; again the handover is completed.

The handover sets new routing paths for query statements. Since neither handover nor statement processing are executed instantly, the handover protocol must take delayed query statements into account. For example a statement is issued at time $t$ while at time $t + h$ a handover is performed, yet the completion of the re-routing is finished at time $t + h + k$. Thus, when the statement is routed to the destination Visited Parent in the time interval $[t + h, t + h + k]$, it ends up at the old Visited Parent rather than at the new one. To process the statement, the old Visited Parent must re-route it to the Mobile Node. Thus, in order to support smooth handovers, Geo-Ds must slightly overlap. In particular, each domain is subdivided in an inner domain and an outer domain. Mobile Nodes that are located in the outer domain are registered with two Node Servers. This guarantees that query statements can be routed to the Mobile Node even if it frequently changes the Geo-D.

The CDDBMS also implements another handover procedure that supports a graceful handshake between two Node Managers. It is applicable to mobile devices that are equipped with a location sensor. Thus with respect to the attached
sensor the location of the device is clear when Geo-Ds are crossed and thus the data object is exchanged in a handshake between the two databases. Such soft handovers are initiated whenever a mobile device crosses the border of a Geo-D. Range-monitoring queries allow keeping track of these events. Cai et al. propose efficient mechanisms for range-queries based on safe regions [4]. With the help of these queries, location sensors are only required to send updates when they leave a save region; thus saving power consumption of the mobile device. However, it is up to the underlying DBMS to implement advanced features like this. The CDDBMS just glues the Node Servers together and exploits their capabilities.

Details on the performance evaluation of the mechanisms offered by the CDDBMS, and in particular the mobile object handling approach, are provided in [3].

3. An optimized context data dissemination mechanism

Undoubtedly, location awareness at anytime and anyplace is a fundamental requirement in a pervasive computing environment. At the same time however, the location-awareness imposes an additional degree of complexity, as it is the most dynamic context information usually involved and requires special handling & management to overcome the inevitable scalability problems. The introduced context distributed database management system alleviates such restrictions establishing a flexible mechanism for maintaining in the home node a chain of pointers to the visited (local) node where the dynamic information of the context mobile entity is collected, including the entity’s location information. In this perspective, the location-awareness problem is efficiently solved, while the invocation of multiple updates in the home node each time a mobile user changed his/her location is reduced to minimal. However, the effectiveness of this mechanism is questionable for other types of context information, as it is not always to the interest of the consumer to communicate remotely in order to acquire context data from a different node; or for an application to be obliged to submit even the simplest context queries through a maze of pointers from the home node to the visited node of the mobile user. Furthermore, it is anticipated that in most situations many consumers will be interested in the same context information. Additionally, several such context consumers may be residing at the same node (e.g. different applications used by the same user on the same device or multiple users utilizing terminals of the same public server). In this perspective, it is much more convenient and efficient to establish context data replicas on some context nodes that will receive updates from the home node depending on how often the original data changes and how popular it is for the various interested consumers residing at the specific node. With regards to situations involving mobile users, additional restrictions may arise (e.g. concerning limited connectivity and bandwidth, unknown network conditions, safety, agreement between administrators, etc.), thus rendering imperative the need to establish mechanisms in support of optimized context information replica dissemination and selective updates.

Globally, the heterogeneous devices, the amount of dynamically generated context data, the limited connectivity and the need for autonomous management raise questions that require special management for the effective implementation and provision of context-aware services and applications. This section elaborates on sophisticated mechanisms for the dissemination and selective updates of context objects and treats their management respectively to their popularity and the explicit requirements of consumers on precision constraints. The proposed mechanism of selective context information dissemination constitutes an additional facility that is incorporated in the existing architecture of the context management system. The proposed mechanism does not encompass methods for the selection of the optimal placement of replicas on central nodes of the network. On the contrary, it assumes that each consumer or group of consumers is capable of explicitly declaring its wish for the establishment of a replica on a specific context node. If this is not possible, in case a central visited node perceives increased demand for a specific context object, it can also require maintaining a context replica for the enhancement and more efficient support of interested consumers. In a nutshell, the designed and implemented framework aims at optimizing and controlling the amount of exchanged context data so that (i) the context sources are relieved from the burden of disseminating frequent updates to the home node, while (ii) the context consumers are not overloaded with context information that does not interest them for the time being.

3.1. Problem framework

As it has already been presented, the home node is the main coordinator for handling any changes that take place in the entities that fall in the node’s administrative domain. Therefore, it is the only responsible node for securely disseminating the context updates to the respective context replicas. The home node role is twofold: (i) to instruct the relevant remote context sources about the forwarding intervals/thresholds of the local context updates by evaluating the real update patterns in conjunction with the consumers’ access patterns and requirements and (ii) to decide when to update the replicas of the context entities of its administrative domain that reside on remote nodes. The basic motivation for this study originates from the observation that there is a fundamental trade-off between the communication cost introduced by the maintenance of fully synchronized context replicas and the degree of synchronization that is eventually necessary. This critical attribute is reported as balancing the parameters of precision and system performance, where the precision is a metric of the degree of synchronization between the master copy and a remote copy, and the system performance represents how sparingly the communication resources are used. For example, when the value of a context object changes rather fast, optimal performance can only be achieved by sacrificing the precision of the copy and on the contrary, the requirement on high precision tends to degrade the performance.
In general, the communication cost is critical in distributed environments, either because the available network bandwidth is limited, or because the network resources can only be used at a premium. As the communication resources are precious consuming products, the complete and instant synchronization of distributed context replicas with the master copy cannot be achieved when the volume of data or the rate of change is high, greatly increasing the introduced communication cost. In the studied framework, the context information consumers play a decisive role in the selective context replica updates process, as they can explicitly declare their requirements on context data precision. Of course, all these parameters are not enough when considered individually to determine a context dissemination and update policy, as there are also other issues that have a great impact on this, such as the replica update rate.

As it has already been presented, each mobile context object constitutes by definition a visited node for the entity’s owner, e.g. the PDA for the user, and it allocates a limited version of the context management system for the administration of the mobile node and its communication with the central context servers of the domains where it moves. In this perspective, a mobile node can host replicas of context information similarly to static nodes and receive updates or even dispatch updates if it operates as context source, while the mobile node is moving. For example, let us consider a diabetic user named Alex, the blood sugar level of whom is dynamically monitored and recorded periodically in his portable device via a sensor located on him. His blood sugar values are obviously essential context information for the hospital he is being treated at, as well as his family. Especially when the blood sugar level exceeds a particular threshold and Alex is in danger unless he receives adequate medication immediately. By establishing context replicas in the hospital context node, as well as the PC at Alex’s home, where at least one of his family members is always located, and enforcing selective context updates dissemination when the blood sugar exceeds the safe limits, it is ensured that the user will receive the medical care that might be necessary, without being bothered with continuous updates for informing the hospital about each new recorded blood sugar sample. Considering the requirements of the interested consumers, the home node should be in place to instruct Alex’s mobile device on selective blood sugar values dissemination, thus suppressing some of the frequent local updates.

As already stated, it is assumed that the context consumers are capable of identifying and declaring their need for the establishment of context replicas. In this framework, the interested consumers are suggested to announce an approximate precision constraint regarding the desirable context value threshold above which they need to be aware about the updated context information value. Each visited context node, which hosts a context information replica can function as context source and/or context consumer and is responsible for monitoring this particular context data and record statistics with regards to the frequency of reading or writing requests, respectively. These parameters are periodically transmitted to the home node, which processes them along with the various consumers’ precision constraints and determines the context update policy. In Fig. 4, a high-level deployment view of the context sources and replicas is illustrated, which applies to the aforementioned scenario with the diabetic user Alex. In this example, a central context home node is examined that maintains a multitude of master context entities, including entity X that represents user Alex. As Alex is diabetic and his blood sugar is continuously monitored and recorded in his PDA, various copies of user X are distributed in various visited context nodes that are interested in Alex. For example, the nodes of the hospital and user’s house, where the doctors and Alex’s family are located respectively, are presented in Fig. 4. Both will obviously require frequent updates for Alex blood sugar levels, at least when this exceeds certain and well-known thresholds. In this use case, the hospital and the user’s house context nodes both act as context consumers, while Alex’s mobile node (namely, his PDA) acts as context source. In general, requirements for strict consistency maintenance between the distributed context copies and the master copy would impose that whenever one node received an update request (in this case the user’s PDA), initially it would be handled locally while consequently, the new context value would be propagated to the home node. The latter would afterwards selectively inform the context replicas residing at remote context nodes about the updated context value. The introduced mechanism for selective context data dissemination supports the decision making process of the various context sources on whether/when to send the updated context values depending on the precision requirements and access behavior of all consumers of this context data item, irrespective of the node they reside at. Furthermore, exactly the same mechanism is exploited by the home node to support its decision concerning whether/when to send the updated context values it received from the context sources to the various replicas distributed over the network. It is important to mention that the preceding scenario constitutes a special use case since it involves medical emergency that calls for individual treatment in order to ensure that the relevant user will acquire the necessary context update at any cost, in case of emergency, e.g. the blood sugar reaches critical levels. Nevertheless, the scenario presentation serves two major objectives: On the one hand, it elucidates the necessity of establishing advanced context dissemination mechanisms, while on the other hand, it underlines important future research tracks that need further clarification (e.g. incorporation of priority constraints), as mentioned in Section 7, were future plans are drawn.

The proposed heuristic algorithm, that will be presented thoroughly subsequently, is applied on both problems and thus provides an end-end solution in the problem of selective context update of distributed context replicas (End-to-End Selective Context Replicas Update Problem). In order to further explain the implementation of the algorithm two cases are distinguished: (i) the context sources are established close to the home node and (ii) the context sources are located distantly. When the context sources are distributed in nodes near to the home node, they are in the place to propagate the updated information values to the home node each time a context update value takes place in the respective source, without imposing unbearable communication cost. In this case the proposed algorithm is used by the home node in order to support its decision with regards to whether/when to dispatch the update context values that received from the respective sources in the various distributed replicas. The decision is taken separately for each context replica, since each node holding such a copy serves a...
update and monitor many consumers with different precision constraints at the same node

Visited Context Node (Context Server Residing in the user's Home)

update and monitor master context objects

replicas context objects

replicas context objects

replicas context objects

Visited Context Node (Context Server Residing in the Hospital)

visited context node (user's PDA)

update and monitor update and monitor

Context Source

Context Consumer

Context Consumer

update blood sugar

Fig. 4. Abstract representation of the proposed context dissemination scheme.

multitude of consumers that are interested for the same context information. The algorithm is executed initially once and separately for each copy, as soon as the home node assembles all the essential information. Thus, when multiple copies of the same context information are being installed in different network nodes, i.e. \( r = 1, 2, \ldots, R \), the proposed optimisation algorithm should be executed \( R \) times, for each node. Subsequently, the algorithm is executed again each time a change takes place with regards to a distributed replica and its consumers (e.g. consumer addition or withdrawal, consumer’s access rate modification, precision requirements’ revision, …). Nevertheless, the algorithm’s repetition concerns exclusively the node that holds the specific copy. Additionally, the algorithm needs to be re-executed for all nodes holding a context replica in case the context update rate in the home node alters. In this case, the home node is responsible to record the context update rate, since all updated values from the various sources eventually reach it. In the second case, where the context sources are distributed in distant nodes and therefore the dissemination of all updated values in the home node does not constitute a cost-effective solution, the proposed algorithm is executed as follows: The home node selects the min \( \Delta V_r \), and it guides the sources to send the updated context information values considering this threshold min \( \Delta V_r \). Furthermore it asks the sources to inform it about the local context update rate. Eventually, the home node uses the cumulative context update rate that corresponds to the whole context sources set. The remainder situations that require re-execution of the algorithm remain the same with the previous case. Thus, the proposed approach is suitable for determining the overall selective context update policy both from the context sources to the home node, as well as from the home node to the nodes holding context replicas.

3.2. Formal problem statement

Consider a single home CM node that holds the master copy of a context data item \( x \). Replicas of \( x \) are distributed across numerous visited CM nodes. Let \( V \) represent the current value of object \( x \) in the home CM node that undergoes updates over time, while \( V_n \) represents the value of object \( x \) residing at node \( n \). Additionally, each context consumer request is accompanied with a precision constraint specifying the maximum acceptable divergence of the \( x \) replica’s value with regards to the actual current value of \( x \). At this point, it is assumed that all nodes are always connected to the network and that the infrastructure is robust, i.e. there is enough bandwidth available, node failures are infrequent, etc. To evaluate the proposed context replica update strategy a stochastic study is provided that considers two discrete random variables: \( A \) that represents the number of outdated retrievals of \( x \) and \( B \) that represents the number of redundant updates of the context replicas of \( x \).
The proposed strategy aims to minimize that values of both these variables. In this framework, three probabilities have been measured over various experiment settings: \( P(A) \) and \( P(B) \), as well as the probability \( P(A \cup B) \). In the subsequent paragraphs, the scope of the three measured probabilities and the purpose they serve are thoroughly explained.

\( P(A) \) is the probability that a consumer retrieves outdated context data. In essence, it quantifies the degree of consistency between a context master copy and its remote replica. The home CM node server propagates a selected subset of context updates to a specific replica of context data item \( x \). Thus, only the context updates that introduce divergence between the replica’s value and the current context value, which is above a predefined threshold, are propagated to the CM node of the replica. Hereafter, we will refer to the case where the context value the consumer retrieves from the visited CM node differs from the context value at the home CM node, more than the precision constraint specified by the consumer, as outdated context retrieval. Using the notation defined above, a context access request submitted at time \( t_b \) by consumer \( C \) with precision constraint \( p_C \), results in outdated context retrieval if \( \| V(t_b) - V^C(t_b) \| > p_C \), where \( V^C(t_b) \) is the context value retrieved from the consumer \( C \) at time \( t_b \), and \( V(t_b) \) is the context value updated in the context server at the time \( t_b \), which indicates the last context update at the server before time \( t_b \).

\( P(B) \) is the probability of redundant replica updates (i.e. propagated replica updates the values of which are never retrieved by context consumers). It indicates how sparingly communication resources are used and is a measure of the communication cost. Once a context value divergence threshold is applied, the context server disseminates updated context values that result in exceeding the predefined threshold. From these context replicas updates, users benefit only from those that modify the required context value outside the bounds of their precision constraint. Also given the fact that consumers are not informed of the updated replicas values every time but only access the current replica value when necessary, for a consumer \( C \) with precision constraint \( p_C \), the replica update that occurs at time \( t_b \) is redundant when: (i) either \( t_b < t_R \) and no access requests occur between replica update times \( t_b \) and \( t_{R,1} \), or (ii) \( \| V^C(t_{A,1}) - V^C(t_b) \| \leq p_C \), where \( t_{A,1} \) is the time of the previous access request of \( C \), \( t_A \) is the time of the last access request of \( C \), \( t_{R,1} \) is a replica update time that occurs between the last two access request of \( C \) (i.e. it stands that \( t_{A,1} < t_R < t_{A,1} \)), \( V^C(t_{A,1}) \) is the context value retrieved by \( C \) at \( t_{A,1} \), and \( V^C(t_{A,1}) \) is the context value retrieved by \( C \) at \( t_{A,1} \) in case a replica update has occurred at time \( t_{R,1} \).

When context data changes rapidly, optimized network resources utilization can be achieved by sacrificing context replicas consistency. On the other hand, achieving high probability of consistency tends to degrade performance. Based on the aforementioned definitions of the two adopted probability metrics, we should expect \( P(A) \) to continuously increase as the applied value threshold increases, while the exact opposite should take place with regards to \( P(B) \). The rationale behind this assumption is straightforward: As the upper bound of the threshold increases, it is likely that more context outdated retrievals will occur and fewer replica redundant updates. In an effort to substantiate this claim, we performed some initial simulations. If the replica’s context current value and the updated context value at the home CM node differ by more than \( \Delta V \), then and only then the updated value is propagated to the CM node of the replica. We define \( \Delta V^+ \) as the value of quantity \( \Delta V \) that minimizes probability \( P(A \cup B) \). Subsequently, estimation process for the value of \( \Delta V^+ \) is thoroughly described.

As one may easily conclude, the probability \( P(A) \) for a consumer to receive outdated information increases as \( \Delta V \) increases. However, as \( \Delta V \) increases, the context server reduces the number of context replica updates and therefore, the probability \( P(B) \) of redundant updates decreases. Obviously, these two introduced probabilities represent opposing interests. Indeed, \( P(A) \) expresses the context consumer’s interest, while \( P(B) \) reflects the proper utilization of the system’s resources. In the established context management domain, we are interesting in satisfying both criteria simultaneously, while trying to address the context consumers’ bounded precision requirements. This approach allows us to balance the benefits of both parties in order to maximize the user benefit, while avoiding redundant utilization of the system’s resources. In this respect, we conducted series of experiments to measure probability \( P(A \cup B) \). The objective of these experiments is to observe and record the context value difference threshold that minimizes the probability \( P(A \cup B) \). Since the two probabilities \( P(A) \) and \( P(B) \) have been carefully calculated taking into consideration the precision requirements of the two consumers, the point where \( P(A \cup B) \) is minimum represents the situation where the following two requirements are addressed in the best possible manner: (i) reduced communication cost that corresponds to minimum redundant replica updates and (ii) consumers’ precision constraints satisfaction that results from minimizing the possibility of outdated context data access. Therefore, the \( \Delta V \) value for which \( P(A \cup B) \) curve exhibits its minimum, corresponds to the optimal threshold \( \Delta V^+ \) above which the home CM node should propagate the context updates to the respective replicas. This selected \( \Delta V^+ \) value will be used to establish the most efficient context consistency control policy for the specified access/update rates, and precision constraints.

4. Heuristics for problem solution

In order to study the problem of selective context replica update, several parameters need to be introduced. Let \( u \) represent the update rate based on which the context data item \( x \) is updated at the home CM. Hereafter, the problem of optimal update value threshold will be studied for an arbitrary CM node \( R \) that hosts a replica of context data item \( x \). Let \( C_i, i = 1, 2, \ldots, N \) be a consumer of context data item \( x \) residing at node \( R \). Let \( a_i \) be the access rate based on which the consumer \( C_i \) attempts to access the value of context data item \( x \), and \( p_i \) be the precision requirement of this consumer. This means that if \( V_R \) is the value of the replicated data item \( x \) at node \( R \) when \( C_i \) accesses \( x \), and \( V_H \) is the value of the master data item \( x \) at the home CM node at that time then: \( C_i \) is satisfied with the retrieved value \( V_R \), if and only if \( V_R \in [V_H - p_i, V_H + p_i] \).
The selective context replica update problem that is studied hereafter, concerns the determination of the optimal context value difference $\Delta V^*$ above which the server (home CM node) will propagate the updated values to the remote context replica residing at the arbitrary CM node $R$, given the context update rate $u$, the access frequency $a_i$ for all consumers $C_i$, $i = 1, 2, \ldots, N$ and the consumers’ precision constraints $p_i$. As already stated, $\Delta V^*$ is such that the probability $P(A \cup B)$ is minimised. To produce the formulas estimating quantity $\Delta V^*$, the methodology described hereafter has been adopted. First, the objective is to minimise the root mean square error of the estimated $\Delta V^*$ with regards to its actual value. Second, four parameters that affect $\Delta V^*$ have been identified: (i) the context update rate, (ii) the number of the consumers, (iii) the access rates of the consumers and (iv) the precision constraints of the consumers. Third, over 150,000 experiments have been conducted with different settings (i.e. combinations of values of these four parameters), where the actual value of $\Delta V^*$ has been measured. Fourth, an attempt has been made to collect general observations concerning the impact of these four parameters on $\Delta V^*$ (e.g. the precision constraints of fast consumers prevail when their access rates are higher than the context update rate). Fifth, based on these observations, it has been concluded that the four parameters aforementioned have a different impact on quantity $\Delta V^*$ depending on the ratio of the consumers’ access rates to the context update rate. Sixth, thus, four Cases have been distinguished to solve the problem, based on the ratio of the access rates $a_i$ to the context update rate. Case 1, concerns the situation where all consumers are considerably slower with regards to the context update rate, i.e. their access rates are lower than $\frac{u}{2}$. In Case 2, the access rates of all context consumers are comparable to the context update rate, thus lying in the interval: $[\frac{u}{4}, 2u)$. In Case 3, all consumers are significantly faster than the context sources, i.e. their access rates are higher than $2u$. Finally, all other possible situations are classified in Case 4. These four Cases have been distinguished due to the fact that the experiments conducted indicated that quantity $\Delta V^*$ varies in a similar fashion within each Case, over the access and update rates, as well as the number of context consumers and their precision constraints. Seventh, each of the over 150,000 experiments conducted has been assigned to one of the Cases distinguished. Eighth, for the experiments of each Case various functions of the aforementioned parameters or of combinations of these parameters (i.e. average precision constraints and average precision constraints weighted by the respective access rates) have been tried to approximate the actual value of $\Delta V^*$. These functions were: Exponential, Fourier, Gaussian, Interpolant (linear, nearest neighbor, cubic spline, shape-preserving), Polynomial (linear, quadratic, cubic, 4th degree polynomial,...), Power and Rational. These have been tested using Version 7.0.4 of Matlab (www.mathworks.com/products/matlab/). Of course, there are also other methods and models in the literature that are used for function approximation and curve fitting, such as neural networks, fuzzy logic, support vector machines, genetic algorithms, etc. In the current version of this study, these models have not been considered, as we chose to base the design of heuristic algorithms on the more lightweight approximation techniques. The resulting root mean square errors of the evaluation experiments that are presented in Section 5 justify this choice. Ninth, the formulas estimating quantity $\Delta V^*$ have been selected among these functions, aiming to minimize the root mean square error. These heuristic formulas are presented in the remainder of this section.

The experiment framework that led to these formulas has been built as follows: the context server update arrivals and access request arrivals have been modelled as Poisson processes, the initial values for the context information are randomly selected among these functions, aiming to minimize the root means square error. The resulting root mean square errors of the evaluation experiments that are presented in Section 5 justify this choice. Ninth, the formulas estimating quantity $\Delta V^*$ have been selected among these functions, aiming to minimize the root mean square error. These heuristic formulas are presented in the remainder of this section.

The experiment framework that led to these formulas has been built as follows: the context server update arrivals and access request arrivals have been modelled as Poisson processes, the initial values for the context information are randomly selected in the range $[0, 10000]$ and the maximum context value difference between two consecutive server updates is uniformly distributed in the interval $[−0.1, 0.1]$, the sample values for the precision constraint for each consumer are $0:0:1:1$ $(0, 0.1, 0.2, 0.3 \ldots 1)$, the sample values for threshold $\Delta V$ are $0:0:1:1$, and the simulation time for each experiment has been set equal to 50 s.

4.1. Case 1: All access rates lower than update rate

As already stated, in Case 1 it stands that $a_i < \frac{u}{2}$, $\forall i = 1, \ldots, N$. Herewith, two sub-cases are distinguished:

Case 1.1. If $\frac{1}{N} \sum_{i=1}^{N} p_i \equiv \bar{p} \leq 0.23$, then the selected context value update threshold is provided by the following expression:

$$\Delta V^* = \min \left\{ \max \left\{ p_i \right\}, \max \left\{ \bar{p}, \left( 0.006176 \cdot \frac{u}{a} - 0.0932 \right) \cdot N + 0.01642 \cdot \frac{u}{a} + 0.3408 \right\} \right\}.$$  

This formula reflects the experimental observation obtained that slow consumers having very strict precision constraints result in $\Delta V^*$ that decreases as the number of the consumers increase in an almost linear fashion. The instances of the experiments conducted to support the design of this $\Delta V^*$ expression are approximately 120,000.

Case 1.2. If $\bar{p} > 0.23$, then the following steps need to take place in order to calculate $\Delta V^*$:

- Sort the consumers $C_i$, $i = 1, 2, \ldots, N$, by descending order of their precision constraint $p_i$. Thus, the sorted list of consumers $C_i$, $i = 1, 2, \ldots, N$, is such that $p_{i'} \geq p_{i'+1}$, $\forall i' = 1, 2, \ldots, N$ and $\forall i = 1, 2, \ldots, N - i'$.
- Select parameter $k$ as follows: $k = \min \left\{ k' \mid \sum_{i=1}^{k'} p_i \geq 0.75 \cdot \sum_{i=1}^{N} p_i \right\}$.
- Then, the selected context value update threshold is provided by the following expression: $\Delta V^* = \frac{\sum_{i=1}^{k} p_i}{k}$.

This formula represents the fact that when there are only slow consumers in a context node requesting the same piece of context data, then the consumers of looser precision constraints prevail in the determination of $\Delta V^*$. The instances of the experiments conducted to support the design of this $\Delta V^*$ expression are approximately 170,000.
4.2. Case 2: All access rates comparable to update rate

As aforementioned, in Case 2 it stands that \( \frac{2}{5} \leq a_i < 2u, \forall i = 1, \ldots, N \). For this case, the selected context value update threshold is provided by the following expression:

\[
\Delta V^* = \min \left\{ \max \{p_i\}, \max \left\{ \frac{\min \{p_i\} \cdot 0.3201 \cdot \left( \sum_{i=1}^{N} (p_i \cdot a_i) \right)^{2} + 0.7221 \cdot \sum_{i=1}^{N} (p_i \cdot a_i) - 0.03294 \right\} \right\}.
\]

This formula reflects the experimental observation obtained that when consumers attempt to access context data almost as often as these are updated, the resulting \( \Delta V^* \) increases as the access rates and the precision constraints increase. Specifically when all consumers have the same access rates, \( \Delta V^* \) depends on the average consumer precision constraints based on a quadratic form. The instances of the experiments conducted to support the design of this \( \Delta V^* \) expression are approximately 280 000.

4.3. Case 3: All access rates higher than update rate

As already stated, in Case 3 it stands that \( a_i \geq 2u, \forall i = 1, \ldots, N \). Then, the following steps need to take place in order to calculate \( \Delta V^* \):

- Select all the consumers that have access rates \( a_i \geq \bar{a} \equiv \frac{1}{N} \cdot \sum_{i=1}^{N} a_i \). Let these consumers be represented by \( C_j, j = 1, 2, \ldots, M \), while the rest of the consumers are represented by \( C_j', j' = M + 1, \ldots, N \).
- Sort the consumers \( C_j, j = 1, 2, \ldots, M \), by descending order of quantity \( a_j \cdot (1 - p_j) \). Thus, the sorted list of consumers \( C_j, j = 1, 2, \ldots, M \), is such that \( a_j \cdot (1 - p_j) \geq a_{j+1} \cdot (1 - p_{j+1}), \forall i = 1, 2, \ldots, M \) and \( \forall i = 1, 2, \ldots, M - i' \).
- Select parameter \( k \) as follows: \( k = \min \left\{ k \mid k' \geq \frac{a}{V} \right\} \).
- Let \( a_{\text{max}} \) be the maximum access rate for the first \( k \) consumers in the sorted list, i.e., \( a_{\text{max}} \equiv \max_{i'=1,\ldots,k} \{a_{i'}\} \).
- Then, the selected context value update threshold is provided by the following expression:

\[
\Delta V^* = \begin{cases} 
\frac{\sum_{i=1}^{k} \left\{ p_i \cdot \frac{a_i}{a_{\text{max}}} \right\}}{k}, & \text{if } p_1 \cdot \frac{a_1}{a_{\text{max}}} < 0.5 \text{ and } \sum_{i=1}^{k} \left\{ p_i \cdot \frac{a_i}{a_{\text{max}}} \right\} \leq 1 \\
0.4 \cdot \sum_{i=1}^{k} \left\{ p_i \cdot \frac{a_i}{a_{\text{max}}} \right\}, & \text{if } p_1 \cdot \frac{a_1}{a_{\text{max}}} < 0.25 \text{ and } \sum_{i=1}^{k} \left\{ p_i \cdot \frac{a_i}{a_{\text{max}}} \right\} > 1 \\
1.4 \cdot p_1 \cdot \frac{a_1}{a_{\text{max}}}, & \text{if } 0.25 \leq p_1 \cdot \frac{a_1}{a_{\text{max}}} < 0.5 \text{ and } \sum_{i=1}^{k} \left\{ p_i \cdot \frac{a_i}{a_{\text{max}}} \right\} > 1 \\
p_1 \cdot \frac{a_1}{a_{\text{max}}}, & \text{if } p_1 \cdot \frac{a_1}{a_{\text{max}}} \geq 0.5.
\end{cases}
\]

This formula represents the fact that the faster consumers that also have the strictest precision constraints are dominant in the determination of \( \Delta V^* \). Twenty percent of these consumers are sufficient to decide on the value of \( \Delta V^* \). The instances of the experiments conducted to support the design of this \( \Delta V^* \) expression are approximately 360 000.

4.4. Case 4: Highly diverging access rates

As aforementioned, Case 4 addresses all situations that cannot be classified under Case 1, Case 2 or Case 3. To study this case, the set of the context consumers need to be clustered. These clusters are defined as follows:

- **Cluster I**: \( C_i \subseteq \text{Cluster}_I \iff a_i < \frac{2}{5} \), \( \text{Cluster}_I = \{C_j\}, j = 1, 2, \ldots, N_i, \frac{1}{N_i} \cdot \sum_{j=1}^{N_i} p_j' = \bar{p}', \frac{1}{N_i} \cdot \sum_{j=1}^{N_i} a_j' = \bar{a}' \).

- **Cluster II**: \( C_i \subseteq \text{Cluster}_II \iff \frac{2}{5} \leq a_i \leq 2u \), \( \text{Cluster}_II = \{C_j\}, k = 1, 2, \ldots, N_{II}, \frac{1}{N_{II}} \cdot \sum_{j=1}^{N_{II}} p_j'' = \bar{p}'', \frac{1}{N_{II}} \cdot \sum_{j=1}^{N_{II}} a_j'' = \bar{a}'' \).

- **Cluster III**: \( C_i \subseteq \text{Cluster}_III \iff a_i > 2u \), \( \text{Cluster}_III = \{C_j\}, l = 1, 2, \ldots, N_{III}, \frac{1}{N_{III}} \cdot \sum_{j=1}^{N_{III}} p_j''' = \bar{p}''', \frac{1}{N_{III}} \cdot \sum_{j=1}^{N_{III}} a_j''' = \bar{a}''' \).

Hereafter, four sub-cases are distinguished:

**Case 4.1.** If (i) \( p''' \leq \bar{p}'' \) and \( p'' \leq \bar{p}' \) or (ii) \( \bar{a}''' \cdot N_{III} \geq 10 \cdot \bar{a}'' \cdot N_{II} \) and \( \bar{a}'' \cdot N_{II} \geq 40 \cdot \bar{a} \cdot N_1 \), then the problem is reduced to Case 3 where only the customers in Cluster III participate, i.e., \( \Delta V^* = \Delta V_{\text{Case 3}}^{a'} \cdot \{c_j\}_{a'} = \Delta V_3 \).

**Case 4.2.** Else, if (i) \( 2.5 \cdot \bar{a}'' \cdot N_{II} \geq \bar{a}''' \cdot N_{III}, p''' \leq \bar{p}'' \), and (ii) \( p'' \leq \bar{p}' \) or \( \bar{a}'' \cdot N_{II} > 10 \cdot \bar{a} \cdot N_1 \), then the problem is reduced to Case 2 where only the customers in Cluster II participate, i.e., \( \Delta V^* = \Delta V_{\text{Case 2}}^{a''} \cdot \{c_j\}_{a''} = \Delta V_2 \).

**Case 4.3.** Else, if \( 2.5 \cdot \bar{a}'' \cdot N_{II} \geq \bar{a}''' \cdot N_{III}, p''' \leq \bar{p}'' \), then the problem is reduced to Case 1 where only the customers in Cluster I participate, i.e., \( \Delta V^* = \Delta V_{\text{Case 1}}^{a'} \cdot \{c_j\}_{a'} = \Delta V_1 \).
Case 4.4. Otherwise:

\[
\Delta V^* = \frac{\sum_{j=1}^{N_l} \left[ a_j^l \cdot (1 - p_j^l) \right] \cdot \Delta V_1 + \sum_{k=1}^{N_l} \left[ a_k^l \cdot (1 - p_k^l) \right] \cdot \Delta V_2 + \sum_{j=1}^{N_l} \left[ a_j^l \cdot (1 - p_j^l) \right] \cdot \Delta V_3}{\sum_{j=1}^{N_l} \left[ a_j^l \cdot (1 - p_j^l) \right] + \sum_{k=1}^{N_l} \left[ a_k^l \cdot (1 - p_k^l) \right] + \sum_{j=1}^{N_l} \left[ a_j^l \cdot (1 - p_j^l) \right]}
\]

In order to determine the value of \(\Delta V^*\) in Case IV, an attempt has been initially made to reduce it to one of the previous cases. If the cluster with the fastest consumers is either more numerous or is characterized by the lowest average precision, then the problem is reduced to Case III. Similar reductions can be made to the other two cases. However, when no cluster prevails over the other two, \(\Delta V^*\) is expressed by a linear combination of the three thresholds resulting when the three consumer clusters are treated independently as instances of Cases I, II and III. In this function, the weights of the three components are formulated based on the access rates and precision constraints of the respective consumer clusters. The instances of the experiments conducted to support the design of this \(\Delta V^*\) expression are approximately 320,000.

5. Experiments and evaluation

The results presented in this section attempt to evaluate the introduced formulas with regards to real data produced from a series of experiments. Our objective is to calculate parameter \(\Delta V^*\) for various values of the input variables and then, study how this actual optimal threshold fluctuates with regards to the estimated \(\Delta V^*\) value, as this is provided by the heuristic formulas presented in the former section. In the experiments conducted the situation studied is as follows: there is a single home CM node, which serves a multitude of context consumers that query for the same piece of replicated context information from a visited CM node. Both context server update arrivals and access request arrivals have been modelled as Poisson processes. The initial values for the context information are randomly selected in the range [0, 100,000] and the maximum context value difference between two consecutive server updates is uniformly distributed in the interval [1, 1].

The sample values for the precision constraint for each consumer are 0:0.05:1, while for threshold \(\Delta V\) the sample values are 0:0.01:1. It has been assumed that quantity \(\Delta V^*\) is equal to the sample value that minimizes probability \(P(A \cup B)\) (for details you may refer to Section 3.2). This is of course a near optimal context value threshold that is however very close to the actual optimal value since the sample values are numerous. For all experiment families, the simulation time for each experiment has been set equal to 10 s.

5.1. Case 1: All access rates lower than update rate

Aiming to evaluate the heuristics designed for Case 1, the estimated \(\Delta V^*\) value, as well as the actual \(\Delta V^*\) value are calculated for two families of experiments, each corresponding to one of the sub-cases distinguished in Case 1. The series of the experiments conducted for the specific case concern up to 20 consumers that all demonstrate access rates equal to 10 requests/s (i.e. \(a = 10\)).

For the experiments aiming to evaluate the heuristics of sub-case 1.1, the precision constraints of the consumers have been selected so that \(\frac{1}{\sum_{n=1}^{N} p_n} \equiv \bar{p} \leq 0.23\), while each precision constraint lies in the interval [0, 1]. To generate the specific precision constraints, a normal distribution has been used having mean value below or equal to 0.23 and standard deviation between 0.01 and 0.2. Thus, approximately 13,000 combinations of up to 20 precision constraints have been produced to test sub-case 1.1. The context server update arrival has been selected as follows: \(u = 20:10:100\), thus rising the overall number of the conducted experiments to 120,000. For the aforementioned experiment settings, the actual \(\Delta V^*\) value has been calculated, while subsequently, the estimated \(\Delta V^*\) values have been produced according to the heuristic formula of sub-case 1.1 presented in Section 4.1. In Fig. 5, the mean values for quantities \(\Delta V^*_{actual}\) and \(\Delta V^*_{estimated}\) are depicted for the various update rates, over the number of consumers. These values have been used to calculate the root mean square error (RMSE), which is also depicted in Fig. 5. As one may easily observe, the RMSEs never exceed value 0.02, which is considerably low, thus indicating the high effectiveness and suitability of the respective heuristic formula.

The mean values of the residuals \(\Delta V^*_{actual} - \Delta V^*_{estimated}\) over the number of consumers are illustrated in Fig. 6, along with an indication of the minimum and maximum residual values observed. The above findings concerning the evaluation of the respective heuristic formula are confirmed in full. An additional observation that can be made here is that, in principle, the highest the number of the consumers, the lowest the error of the heuristic formula is. This has been expected, as the more the context consumers are, the lowest the optimal \(\Delta V^*\) value is.

In conclusion, the heuristic approach designed can be safely exploited in the context management infrastructure for selectively disseminating context information in remote context nodes, when the various consumers’ context access rates are significantly lower than the update rate, while their precision constraints are considerably low.

For the experiments aiming to evaluate the heuristics of sub-case 1.2, the precision constraints of the consumers have been selected so that \(\frac{1}{\sum_{n=1}^{N} p_n} \equiv \bar{p} > 0.23\), while each precision constraint lies in the interval [0, 1]. To generate the specific precision constraints, a normal distribution has been used of mean value lying in the interval [0.25, 1] and standard deviation between 0.01 and 0.2. Thus, approximately 19,000 combinations of up to 20 precision constraints have been produced to test sub-case 1.1. The context server update arrival has been selected as follows: \(u = 20:10:100\), thus rising the overall
number of the conducted experiments to 170 000. For the aforementioned experiment settings, the actual $\Delta V^*$ value has been calculated, while subsequently, the estimated $\Delta V^*$ values have been produced according to the heuristic formula of sub-case 1.2 presented in Section 4.1. In Fig. 7, the mean values for quantities $\Delta V_{\text{actual}}^*$ and $\Delta V_{\text{estimated}}^*$ are depicted over the mean value of the precision constraints of all consumers. As one may easily observe, the two curves are quite close, thus indicating the suitability of the respective heuristic formula.

The diagram of Fig. 8 illustrates the mean values of the residuals $\Delta V_{\text{actual}}^* - \Delta V_{\text{estimated}}^*$ over the mean value of the precision constraints of all consumers. The RMSE calculated is equal to 0.0258, which is rather low, thus verifying the effectiveness of the respective heuristic formula. An additional observation that can be made here is that, in principle, the residuals are of negative value. This means that the estimated optimal $\Delta V^*$ value is in general higher than the actual optimal $\Delta V^*$ value. This indicates that there is a little room for improvement for the respective heuristic formula. In any case though, as the errors are quite low, it can be safely stated that the heuristic approach designed can be efficiently exploited in the dissemination
policies of the context management system, when the various consumers’ context access rates are significantly lower than the update rate, while their precision constraints are not low.

### 5.2. Case 2: All access rates comparable to update rate

A similar approach has been adopted to evaluate the heuristics designed for Case 2. Thus, the estimated and the actual $\Delta V^*$ values have been calculated for a wide family of experiments. All experiments conducted for the specific case concern up to 50 consumers that all demonstrate access rates that lie in the interval [50, 150]. The precision constraints of the consumers have been selected as follows: $p_i = 0.001:1$, while the context server update arrival is equal to 100 updates/s (i.e. $u = 100$). Thus, approximately 280 000 experiments with different settings have been conducted to verify the heuristic formula designed for Case 2. In Fig. 9, the mean values for quantities $\Delta V^*_{\text{actual}}$ and $\Delta V^*_{\text{estimated}}$ are depicted over quantity $\sum_{i=1}^{N} \left( p_i \cdot a_i \right) / \sum_{i=1}^{N} a_i$, along with an indication of the minimum and maximum $\Delta V^*_{\text{actual}}$ values demonstrated. As it can easily be observed, the two curves almost overlap, thus indicating that the respective heuristic formula (presented in Section 4.2) approximates very efficiently the actually observed optimal value for the minimum context value difference threshold.

In Fig. 10, the mean values of the residuals $\Delta V^*_{\text{actual}} - \Delta V^*_{\text{estimated}}$ are illustrated over quantity $\sum_{i=1}^{N} \left( p_i \cdot a_i \right) / \sum_{i=1}^{N} a_i$. The RMSE calculated is equal to 0.0169, which is significantly low, thus confirming the suitability and validity of the designed heuristic formula. Therefore, it is proved that the heuristic approach significantly optimizes the context management facilities, when the various consumers’ context access rates are comparable with the context update rate at the home node.

### 5.3. Case 3: All access rates higher than update rate

The heuristics designed for Case 3 have been empirically evaluated following a similar approach to Cases 1 and 2. Thus, the estimated and the actual $\Delta V^*$ values have been calculated via a wide range of experiments involving up to 30 consumers that all demonstrate access rates that lie in the interval [200, 1000]. The precision constraints of the consumers have been selected as follows: $p_i = 0.001:1$, while the context server update arrival is equal to 100 updates/s (i.e. $u = 100$). Thus, approximately 360 000 experiments with different settings have been conducted to verify the heuristic formula designed
for Case 3. However, as the range of both the access rates and the precision constraints is very wide, the optimal context value difference thresholds greatly varied. Thus, putting down a diagram illustrating the mean values for quantities $\Delta V_{\text{actual}}^*$ and $\Delta V_{\text{estimated}}^*$ over some function $f(p_i, a_i)$ was rather misleading, as there have been several different inputs for $p_i$ and $a_i$ that result in different $\Delta V_{\text{actual}}^*$ and $\Delta V_{\text{estimated}}^*$, but the same value of $f(p_i, a_i)$. Therefore, it has been decided to depict only a narrow set of experiments in Fig. 11, which presents the mean values for quantities $\Delta V_{\text{actual}}^*$ and $\Delta V_{\text{estimated}}^*$ for 10 consumers with access rates within the interval [200, 1000]. These quantities are illustrated over the two minimum precision constraints $p_1$ and $p_2$. As it can easily be observed, the two surfaces are quite close, thus indicating that the respective heuristic formula (presented in Section 4.3) is quite suitable to approximate the actually observed optimal value for the minimum context value difference threshold for this specific set of experiments.

In Fig. 12, the mean values of the residuals $\Delta V_{\text{actual}}^* - \Delta V_{\text{estimated}}^*$ are illustrated over quantity $\frac{\sum_{i=1}^{N}(p_i \cdot a_i)}{N \cdot a_{\text{max}}}$ for the entire set of the 360,000 experiments conducted. The RMSE calculated is equal to 0.0142, which is even lower than the one observed in Case 2, thus indicating that the designed heuristic formula is very efficient in approximating the actual optimal context value difference threshold. Therefore, it is proved that the heuristic approach designed is highly effective, when the various consumers’ context access rates are significantly higher than the context update rate at the home node. An additional observation that can be made in Fig. 12 is that the residuals are quite lower than the RMSE for $\frac{\sum_{i=1}^{N}(p_i \cdot a_i)}{N \cdot a_{\text{max}}} < 0.8$. This indicates that the proposed heuristics are more accurate in case the “faster” consumers do not present very high precision constraints.
This result has been expected, as the consumers that clearly “prevail” over the others are the ones that are both the fastest and have the lowest precision constraints. As long as this is the case, the proposed heuristic formula is almost 100% accurate.

5.4. Case 4: Highly diverging access rates

In order to evaluate the performance of the heuristics designed for Case 4, the estimated $\Delta V^*$ value, as well as the actual $\Delta V^*$ value are calculated for four families of experiments, each corresponding to one of the clusters distinguished in Case 4. The series of the experiments conducted for the specific case concern up to 50 consumers with varying access rates within the interval [50, 1000]. The precision constraints for each consumer lies in the interval [0, 1], while the mean context server update arrival has been selected equal to 100 updates/s (i.e. $\mu = 100$). For the four experiment families, approximately 320000 experiments have been performed to calculate the actual $\Delta V^*$ value for various combinations of precision constraints and access rates. Subsequently, the estimated $\Delta V^*$ values have been produced according to the heuristic formulas that have been presented in Section 4.4. The resulting $\Delta V_{\text{actual}}$ and $\Delta V_{\text{estimated}}$ are used for calculating the RMSE for each sub-case, which is depicted in Fig. 13.

The RMSE values illustrated in Fig. 13 clearly indicate that the introduced formulas reproduce quite successfully the actual $\Delta V^*$ value, as the RMSE does not exceed the threshold of 0.025 in any of the four clusters. Therefore, the designed heuristic approach significantly enhances the context dissemination processes, when the various consumers’ context access rates vary significantly and thus, can not be treated uniformly. No additional figure is presented for this case, as the $\Delta V_{\text{actual}}$ and $\Delta V_{\text{estimated}}$ values are too complex to depict (very high degree of multidimensional space) given the high number of consumers and the wide interval of access rates values considered.

6. Related work

The research track in the field of context-awareness goes back in the ‘90’s and includes the Active Badge System developed at Olivetti Research Lab [5] and the ParcTab system developed at the Xerox Palo Alto Research Center [11]. A few years later, Cyberdesk [12] built an generic architecture to handle limited types of context. At the same time, the Cyberguide application [13] enhanced a guidebook by adding location awareness and a simple form of orientation information. The Ektara architecture [14] reviewed a wide range of context-aware computing systems, identified their critical features and proposed a functional architecture for the development of real-world applications. Later on, Mediacup [15] and TEA (http://www.teco.edu/tea/) projects tried to explore the possibility of hiding context sensors in everyday objects.

A quite promising approach was introduced by the Context Toolkit [16], the first one that isolated the application from context sensing. Later on, a research team in the Georgia Institute of Technology built the Context Fabric [17] aiming in enhancing the functionality of Context Toolkit. The Aura Project [18] at Carnegie Mellon University investigated how applications could proactively adapt to the surrounding environment. While the Context Toolkit focused on developing an object oriented framework and allowed use of multiple wire protocols, Aura focused on developing a standard interface for accessing services and forced all services and clients to use the same wire protocol. This sacrificed flexibility, but increased interoperability.

HotTown [19] project developed an open context service architecture. All entities were represented by mobile agents. In HotTown, entities could exchange, merge and interpret context knowledge in the end devices. The Cooltown project by HP labs introduced a uniform Web presence model for people, places and things [20]. Rather than focusing on creating the best solution for a particular application, Cooltown built a general-purpose mechanisms for providing Web presence for people, places and things, but its use was limited to tourist guide applications.
Other interesting research activities include the CoBrA, SOCAM, CASS and CORTEX projects. CoBrA (Context Broker Architecture) [21] is an agent based architecture, which has adopted an OWL-based ontology approach and offers a context inference engine. The CoBrA architecture lacks the necessary structure for establishing a large-scale system extending beyond a single place. Subsequently, the SOCAM (Service-oriented Context-Aware Middleware) project [22] is based on a central server that retrieves context data from distributed context providers. SOCAM also uses ontologies to model context and implements a context reasoning engine. The major disadvantage of SOCAM architecture is its centralized implementation. Another server-based middleware for context-aware mobile applications on hand-held and small mobile computers is designed within the CASS (Context-awareness sub-structure) project [23]. CASS opens the way for context-aware applications configurable by users, but its use is limited to small mobile terminals. The CORTEX project is based on the Sentient Object Model [24] and is applied in ad-hoc mobile environments.

The Context Management Framework (CMF) [25], designed from the VTT Technical Research Centre of Finland, presents a uniform mobile terminal software framework for acquiring and processing context information and uses ontologies to model context. The proposed approach falls short in its applicability since it concerns collection of information related to mobile terminals. The use of agents and ontologies for collecting context information is also proposed in the Agent-based Context-Aware Infrastructure (ACAI) project [26] from the University of Ottawa. However, the approach is described quite generally without presenting implementation details and mainly without justifying the use of many agents. Project Nexus [27] of the University of Stuttgart provides an open global platform for supporting spatially aware applications, just like WWW servers support web-based applications. The same project is also studied in [28], where the established platform is extended in order to focus on data management aspects in large-scale pervasive computing systems that use different server implementations tailored to specific classes of data. The Nexus initiative has performed quite interesting research work and its indisputable asset is that it represents the first integrated context platform that adopts a spatially-aware world model for distributing and managing context information. Their pioneer work influenced our context management system in many ways, but we chose to proceed in different research areas, like the ones discussed in this paper, not yet addressed in Nexus project. Finally, the Pervasive Autonomic Context–aware Environments (Pace) project [29] from the University of Queensland, developed a complicated layered architecture in support of a proposed graphical context model and preferences’ model. The established infrastructure permits the user to control his context but does not address efficiently various critical management issues, e.g. distribution.

Throughout the research literature, considerable effort has been spent on managing replicas in distributed computing systems. The vast majority of these studies focus on systems that require careful cooperation and standard protocols between the participating parties in order to ensure perfectly synchronized replicas. Nevertheless, because of the autonomy of sites as well as the heavy performance requirements, many networked applications cannot support this level of cooperation [30]. In these cases, the various copies are not fully synchronized, but there are several updates policies that periodically enforce replica synchronization to specific requirements. An important design issue in systems that maintain replicated information is how to ensure the consistency of these replicas without limiting their availability or using more resources than necessary. This need becomes imperative in an open context marketplace, where a wide variety of information types is traded, timely delivery of context information is crucial. To the best of the authors’ knowledge, no prior work has been conducted to empirically measure and evaluate various selective context replica update policies over time. Thus, the short summary of related work provided subsequently, originates in traditional computing areas, such as distributed systems and the World Wide Web (WWW).

In the distributed World Wide Web (WWW) environment, the need for abandoning stringent transactional replication protocols that eventually guarantee one-copy serializability and performing asynchronous propagation of updates in a non-transactional fashion is most obvious, as the minimization of the response time and the maximization of service availability are mandatory. This situation is further strengthened by the fact that exact consistency is virtually impossible in the presence of the high degree of autonomy featured on the Web makes things even clearer, as well as the fact that the volume of data is vast and aggregated data change rates are astronomical [31]. On the Web, two forms of approximate replication are currently in heavy use: Web caching [32–35] and Web crawling [36–40]. In general, the research work performed in the web domain mainly focuses on maintaining a local copy of the web as up to date as possible, which requires maximizing the fraction of remote pages the local copy of which is up to date. A common limitation in all these approaches is that the refreshing policies are built based solely on predictions concerning which source data objects have changed and how much [41]. The goal of our research is different: we aim to maximize the freshness of the various context replicas distributed throughout the network, so that the precision constraints of the various context consumers are addressed as much as possible, also considering the context update rate and the consumers’ access patterns. In particular, assuming that the above parameters are known, we aim to determine the optimal context value threshold, above which the context server will propagate the updates to the respective replicas. By estimating this threshold based on the update and access rates, as well as the consumer’s precision constraints, the proposed technique achieves substantially reduced utilization of the system’s resources, while addressing the consumers’ requirements.

Replication is also a key enabling technology in distributed data sharing systems for improving both availability and performance. In such environments, Olston [42] studies adaptive refresh policies on replicated data in order to gain fine-grained control over the trade-off between precision and performance. His approach guarantees a “divergence bound” on the difference between the values of the replicated data and the source data through the cooperation of sources [31]. The definition of “divergence” or “change” in this work is quite generic and can be also applied to our framework in case
of non-numerical data. However, there are many environments, including the WWW, where this push-based approach cannot be applied, as data sources do not inform clients of any changes, while in our context management system the home node domain is always informed about the status on the context data under its administration. In general, the approach proposed in this paper is partially inspired by the work presented in [42], but the special requirements of a context management infrastructure radically differentiate the designed solution. Firstly, in our system the replica nodes are not allowed to control the update procedure as in [31,42]. Instead, the client nodes are responsible only for informing the adequate context server about possible context updates, the access frequency rates, as well as the level of precision the various consumers require. Secondly, we are using different metrics for quantifying the trade-off between precision and performance. Finally, a major differentiating factor is that while we aim to address the precision requirements of various context consumers simultaneously, the research in [31,42] focuses on serving the precision requirements of a single consumer with multiple queries. In our approach, the context server is the single entity that decides on whether/when to update the nodes holding replicated context information, aiming to satisfy the various, and probably contradicting, precision constraints and considering the consumers’ access rates and context data update rate.

Finally, even though in the current version of this study no learning and prediction features have been considered, these facilities could potentially enhance the proposed framework enabling the prediction of the user behavior. In the domain of the presented context data dissemination problem, the development of predictive models for estimating varying parameters, such as the context request rate, update rate, the context precision constraints, node failures, network problems, connectivity failures, etc. would greatly enhance the proposed context data dissemination framework, resulting in optimized users’ service time, less network traffic and improved application run time, since the system would know in advanced the various nodes’ needs regarding context data. Hereafter, a brief summary of related research work in the area of context management systems enabled with learning facilities is provided; in an attempt to evaluate existing work and set the fundamentals for pursuing this issue in the near future work. Various research initiatives have focused on predicting the user’s behavior in pervasive environments. In [43], a system is presented that supports proactive, modelling-based, adaptations in a user’s office. The system learns the patterns of the user’s behavior in an office environment based on the context history. In [44], Bayesian networks are used for determining the user activities in a smart home environment, mainly based on sound sensor data. The Specter project [45] maintains a comprehensive log of the user’s behavior together with corresponding context descriptions, thus enabling adaptive systems to learn about users, to identify their habits and to improve the quality of user support. The AGAPE framework [46] exploits the visibility of context information (e.g., user location, user attributes/preferences, access device properties) to create and discover groups of interest for resource sharing, to monitor the availability of groups members and to dynamically arrange/requalify group members bindings to shared resources as changes in context operating conditions occur. The Synapse project [47] employs hidden Markov models to correlate information and learn different user’s habits by exploiting the recorded histories of context and services. A general shortcoming of current approaches is that little work has been directed towards the best use of newly learnt personal patterns, beyond the configuration of application appearance and some aspects of their behavior (e.g. automated call forwarding). In this perspective, it lies among the authors’ future plans to design suitable learning mechanisms in order to enhance the proposed selective context data dissemination framework.

7. Conclusions and future work

Pervasive computing environments impose several major challenges on context data management issues. The existing systems architectures and services are either not satisfactory or suitable for pervasive infrastructures viable in world scaling beyond a strictly constrained laboratory environment [10]. In order to address these new challenges, it is essential to establish innovative data storage and dissemination mechanisms that are applicable to any distributed context environment. The architecture of the Context Distributed DataBase Management System (CDDBMS) presented in this paper can be applied to a wide variety of devices ranging from resource-limited PDAs to central context servers. Moreover, the CDDBMS hides the increasing complexity of context management from external actors and incorporates advanced mechanisms for the support of mobile users, so that the various applications can operate smoothly, independent of the degree of the users’ mobility. In this framework, a location-based view in context data management has been adopted and therefore, the classification and storage of the context information is performed in collaborating and distributed databases, the hierarchy of which reflects the geographical structure of the physical world. A focal point in the introduced CDDBMS is that, for each piece of context information monitored, a master copy is maintained, residing at a central point of access, the home node, while replicas of the specific data can roam and operate in a remote geographic region. The implemented mobile object handling mechanism is based on a chain of pointers from the home node to the remote mobile node and succeeds in minimizing the location updates that reach the core network infrastructure.

The proposed CDDBMS has been designed to meet the requirements of telecommunication operators and context marketplaces. As such, it implements the features which have been presented in this paper. Other more advanced search features, such as context inference [48], query extension mechanisms and free-text based query handling [44] are also available in case the advanced Context Management layer [49] is running on top of the CDDBMS. Such sophisticated features are necessary to enable applications and application developers to discover the context information they need, irrespective of the data location, to prevent them from drowning in the information glut produced by large context source infrastructures and to obtain optimal context value estimations even if the necessary context sources are not available. Further research
plans involve the extension of the context query processing described to incorporate facilities for identifying the result set demonstrating the highest possible Quality of Context [50]. Additionally, in order to further increase the retrieval performance of the CDDBSMS, the Node-Manager is being extended with OWL-based semantic matching facilities. Finally, aiming to establish a global privacy protection scheme, the establishment of an infrastructure is being studied that exploits recommender systems [51] for controlled sharing of information concerning the trustworthiness of the various stakeholders in the context marketplace and for evaluating the potential privacy threats that may be introduced by specific parties.

The distributed context management systems are expected to collect, store and process context information originating from raw data generated at geographically disperse areas. Thus, due to reliability, performance, and cost issues, context databases need to be distributed and allow the creation of closely monitored context replicas in a number of remote context nodes. While in general, the storage of context data in a location oriented distributed database system simplifies the administrative schema alleviating a critical scalability burden, an important challenge still needs to be addressed: since access to context data is performed in a distributed fashion, the creation of context objects replicas is of strategic importance for optimizing the performance and the reliability of the context system. Ideally, the various context objects replicas in remote databases are completely synchronized with the master copy residing at the home node. Nevertheless, the distribution of all context updates to the master copy and subsequently, to the rest of the distributed replicas is usually not practical or prohibitively expensive. The collections of context data can be extensive or particularly dynamic and the network or the computing resources can be limited or usable at a premium, while at the same time users may not be interested in frequent context values readjustments and instead be willing to have access in bounded out-of-date information. In this perspective, the problem of optimally using the underlying communication resources in pervasive environments exploiting selective context update dissemination strategies is being studied in this paper and heuristics solutions are provided and evaluated. The designed formulas suggest a context value difference threshold over which the context update should be propagated to the remote nodes. The proposed context update policies aim to address multiple consumers' precision requirements and relieve all parties involved (including context sources, consumers and home nodes) from the burden of continuously propagating remote context updates or requests. The proposed approach has been evaluated via extensive experiments conducted under 1.25 millions of different settings. The results obtained concerning the performance evaluation of this approach indicate that the designed policies approximate very accurately the optimal value difference threshold, demonstrating root mean square error lower than 0.03 in all circumstances. Furthermore, the redundant context updates are suppressed in 82.6% of the cases, while the outdated context retrievals are reduced by 91.7%. Therefore, the proposed schema is considered of strategic importance for the viability of the context marketplace in a world-wide scale, since it successfully addresses the critical issue of efficient context data dissemination in distributed network nodes.

The authors are currently working on extending the existing approach addressing further requirements on behalf of the consumers (e.g. priority constraints), while network resources vary unexpectedly and dynamically (e.g. network failures occur). Priority constraints will be incorporated in the presented mechanism as an additional metric of the consumer's needs/desires (e.g. medical emergency) to acquire updates of a specific context data at any cost, irrespectively of the surrounding conditions. Further research plans include the extension of the introduced context dissemination strategy to integrate methods of evaluating Quality of Context parameters and study how such metrics can influence the decision about a context update policy, minimizing the effect of less reliable updates and strengthening the impact of trustworthy context sources. Additionally, the authors aim to examine more advanced and less lightweight approximation mechanisms to support the design of heuristics for distributed and selective context data dissemination, such as neural networks, fuzzy logic, support vector machines and genetic algorithms. Furthermore, it lies among the authors’ imminent plans to build support for automated decision making processes concerning the selection of the most appropriate network nodes [48] to host context replicas. Finally, the authors have already started to experiment with various learning techniques to enhance the facilities of pervasive computing environments [52,53], which they plan to couple with the presented framework in order to proactively adjust online the context value update threshold based on predictions of the context access behavior of new and existing context consumers, as well as on predicted network parameter values or failures.

References

Maria Strimpakou received her Diploma in Electrical and Computer Engineering from the National Technical University of Athens (NTUA) in 2000 and her Ph.D. in 2008 from the same university. Her research interests lie in the areas of context-aware systems, ontology design, distributed environments, etc.

Carsten Pils received a Dipl.-Inform. degree in 1997 and a Ph.D. in 2005 from the Technical University RWTH Aachen, Germany. Since 2005 he is working for the Telecommunications Software and Systems Group at the Waterford Institute of Technology, Ireland. His current research interests include context-aware computing and information retrieval.

Nikos Kalatzis received his Diploma in Physics in 2000 from the University of Ioannina and a MSc on Information Security from the University of London in 2002. He is currently pursuing a Ph.D. degree in NTUA and performs research work on distributed context management, history-of-context handling, user intent prediction, etc.

Nicolas Liampotis received his Diploma in Electrical and Computer Engineering from the National Technical University of Athens (NTUA) in 2005 and is currently pursuing a Ph.D. degree working in the NTUA Computer Networks Laboratory. His research interests include context-awareness, privacy and trust management in pervasive computing, distributed pervasive environments, etc.