Distributed Context Models in Support of Ubiquitous Mobile Awareness Services

Jamie Walters, Theo Kanter, and Roger Norling

Department of Information Technology and Media
Mid Sweden University, Sundsvall Sweden
{jamie.walters,theo.kanter,roger.norling}@miun.se

Abstract. Real-time context aware applications require dynamic support reflecting the continual changes in context. Architectures that distribute and utilize the supporting sensor information within the constraints of publish-subscribe systems provide sensor information in primitive forms requiring extensive application-level transformations limiting the dynamic addition and removal of sources. Elevating sensors to first class objects in a meta-model addresses these issues by applying ontological dimensions in direct support of context. This paper proposes an extension of such a model into a distributed architecture co-located with context user agents. This arrangement provides clients with a model schema which is continually evolving over sensor domains. In addition, the evolving model schema represents an accurate temporal view of a userâĂŹs context with respect to the available sensors and actuators.

Keywords: Object-Oriented, Context Awareness, Peer-to-Peer, Context Agents.

1 Introduction

The dynamic nature of context information creates support for the deployment of real-time dependent applications and services across stationary as well as mobile networks. These applications, while becoming increasingly ubiquitous, provide users with information and services outside the scope of current stationary computing. Enabled devices such as mobile phones, television sets and IPTV boxes are all aiming towards this service provisioning paradigm of "everywhere computing" [18]. Of importance is their ability to apply computing to social situations; "mediating" by provisioning, delivering or acting upon user-based information [19], [11]. Such information is regarded as user context information [22] and drives the adoption of ubiquitous computing services and applications.

Ubiquitous Services will enable the realization of seamless media. Service providers need to be able to deliver media solutions across a variety of platforms and scenarios based on user preferences. This would create globally accessible media distribution and provide seamless access and transfer as users navigate around the Internet of "Media" Things [21]. Social media and social networking are enhanced by the ability to manipulate presence profiles and information in real-time in response to changes in context, creating true real-time online presence and interaction [12].

As these applications become increasingly mobile, the need to derive models capable of representing and deriving context within real-time limits, increases. Architectures such as the IP Multimedia Subsystem (IMS) [3] and SenseWeb [13] project present opportunities for deriving and persisting information in support of context. However such centralized approaches are network-centric and require clients to retrieve context information from service portals on the Internet. Mobile broadband access services via 3G mobile systems can deliver access to context information from service portals in the Internet, but the centralized architectures incur severe limitations for real-time context aware applications. Mobile access via 3G mobile systems to context information centralized in service portals on the Internet also suffers from unreliable connectivity. In response to the challenges, we seek a data-centric approach which allows for real-time context-aware applications to leverage distributed access to context information, by relocating context information closer to end-points. Improvements in offering seamless connectivity via heterogeneous mobile broadband systems would further strengthen the approach.

One solution is to distribute the sensor information in a P2P overlay as achieved with DCXP [14]. However, such approaches either assume that sensor information is represented as context information or that applications and services gain access via support which associates sensor information with context. Consequently, access to raw sensor information does not negate the need to access metadata required for applying this information to some presence or context model in support of applications and services.

Architectures such as DCXP [15] resolve this problem by offering interworking with a centralized platform where sensor information required can be accessed in real-time. However, in order for applications and services to create some intelligent behavior through the use of agents leveraging context information, the support is required to access prescence information which is organized in a centralized model. Subsequently, this points to issues around the ability of the support to scale well. Therefore, there exists a need to substitute the centralized element of the DCXP hybrid while capitalizing on its distributed properties. This, along with the application of an intelligent context representation model, can create a solution for enabling distributed context in support of the proliferation of ubiquitous computing applications and services.

2 Motivation

There exists a need for real-time distributed access to sensor information. Such information, organized into models representing users and their interactions with the real world, motivates research into supporting systems and technologies. With the *Internet of Things* moving from a concept to a reality; dynamic yet robust context-centric architectures must be created in direct response to the need to fulfill the demands for information created by the integral user-services relationships.

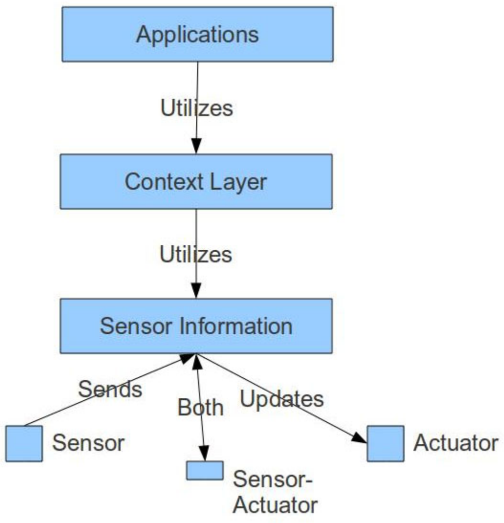

Fig. 1. Context-Awareness Support Model

Our previous work with the DCXP protocol [14] explored distributed approaches to gathering and sharing sensor information in support of context. This provided the sensor information needed to create simple applications based on user context. However, more advanced applications and services such as agent-based computing require context information to be presented and described in an intelligent manner. This adds meaning to the sensor data, the entities they describe and the sensors themselves. This mandated the type of general architecture shown in figure 1.

At its base, such a support requires real-time access to sensor information generated by users or their environment. We deem solutions such as IMS [3] or Senseweb [13] to be inadequate owing to their centralization. They are regarded as being non-scalable with respect to the real-time properties mandated by the dynamic behavior of users and their environment. The DCXP approach produced the ranges in response times adequate enough to support real-time context dependent services. Furthermore, it proved that distributed systems were more capable of achieving this than centralized approaches building on mobile or web services.

The current DCXP architecture is centered around the dissemination of context information as raw sensor values. While this provided the sensor information needed to create simple applications based on user context, more advanced applications and services such as agent-based computing require context information to be presented and described in an intelligent manner; adding meaning to the sensor data, the entities they describe and the sensors themselves. To this end we proposed the use of the Context Information Integration model (CII) [8], an object oriented model supporting context aware applications.

There remains however, the question of scalability within this model as the model in its present implementation, is a centralized solution. It creates a repository over a relational database with context user agent nodes, interacting with context information as required. Simply distributing the current model across the DCXP framework would create ubiquitous islands of context information and subsequently obstacles to seamless reasoning over the global data or an applicable subset. The simple approach of using distributing a relational database such as in [26] and making use of the advanced research in database distribution is not applicable. This, as such a distribution assumes communication reliability in order to maintain database integrity across wide area networks which cannot be guaranteed in heterogeneous mobile scenarios [2], [27]. This is also undermined by the fact that relational databases are highly inefficient for supporting real-time data manipulation, evolution and querying.

An approach would be to use cloud data store such as Big Table [4] capable of enabling broad access to context information. This, however is not represented in an intelligent manner and therefore the resulting support would rather simply replace the persistence layer within the architecture while offering no additional knowledge to the agents or applications residing on top.

A solution should also consider that dynamic characteristic of context is expressed in the continual changing of sensor information to which a user has access. As an entity changes state in real-time, it will encounter new sensors or sensor information, such as entering a building or vehicle. This requires that a schema of available context information about a user must be maintained and kept current; an evolving meta model as suggested in [17]. In solutions such as IMS, such user information is centralized with scalablity issues. With solutions such as [9], [24], [10], citeKawakami2008 this information could be stored within the ontologies. This however would then be subjected to the performance issues raised in [28]. Supporting such dynamic scenarios also requires means of quickly and efficiently locating relevant sensors within the framework as well as a standardized means of addressing and naming these sensors within disparate infrastructures.

Our research in this direction is further motivated by the benefits that arise when meta-data with some added value is given to sensors in a distributed environment. The concept, that while sensors give rise to context information, they themselves must also possess some context; determining relations with other sensors and entities within the framework.

In order to meet the requirements of providing real-time distributed context information, we must leverage the benefits of distribution coupled with the intelligent representation of context within the CII model. Our approach, would require that context be represented in a intelligent manner capable of supporting the range of services dependent on context information, but be made available in a manner that is readily accessible across distributed architecture devices and with indifference to network heterogeneity.

3 Approach

3.1 Distributed Context eXchange Protocol

DCXP is an application level XML based context exchange protocol which offers reliable communication among nodes that have joined a peer-to-peer network as illustrated in [15]. It utilizes a distributed hash table based on Chord [25] for message forwarding and naming resolutions. Any device on the Internet may implement the DCXP protocol and register with the architecture in order to share or digest context information. The original DCXP protocol was limited to exchanging sensor information within a publish and subscribe pattern. While DCXP works as expected for exchanging simple context information, we need to make several changes in order to enable support for creating context models.

We replace the XML format for exchanging DCXP messages with a JSON [6] based format without making any significant changes to the message structures. We attempt to gain speed [20] with regards to constructing, disseminating and digesting messages with an emphasis on the resource-limited devices that would participate in the overlay. Further to this, we introduce a *TRANSFER* and a *SET* primitive to complement the existing protocol set. We reason that in a distributed environment, a sensor will not always be managed closest to the end points requiring its values. As such, a time critical application could be adversely affected by the overheads required even within a minimalistic P2P implementation. Where this occurs, rather than ascribing to the established publish-subscribe protocol, the node, using the *TRANSFER* primitive, could request that the sensor responsibility be relocated locally. One primary difference however, is that the *TRANSFER* primitive, unlike the remaining protocol set, can be accessed by the applications and services above the CII model. This allows applications and services to influence such a decision within the DCXP protocol.

The *SET* primitive, extends DCXP to accommodate actuators, such that we can now permit applications and services to modify environment based on context information. This more accurately represents the sensor-actuator end points of underlying *Wireless Sensor Actuator Networks* and accommodates scenarios such as intelligent homes where the climate control is a combination of both sensing the temperature and adjusting it to suit user preferences.The URI based Universal Context Identifier (UCI) detailed in [16] remains in use but is extended to accommodate distributed addressing within the object model overlay. This is addressed later in the paper.

3.2 The Meta Model

The CII meta-model as detailed in [8] describes an *entity-predicate-entity* triple implemented in an object-oriented framework. Such a model, as illustrated in figure 2 is similar in concepts to the semantic web approaches, however it remains advantageous with regards to the time taken to traverse and reason over an object-based model. Such a model provides a way to represent the realtionships and interactions among the *connected things* within an *Internet of Things*. Where

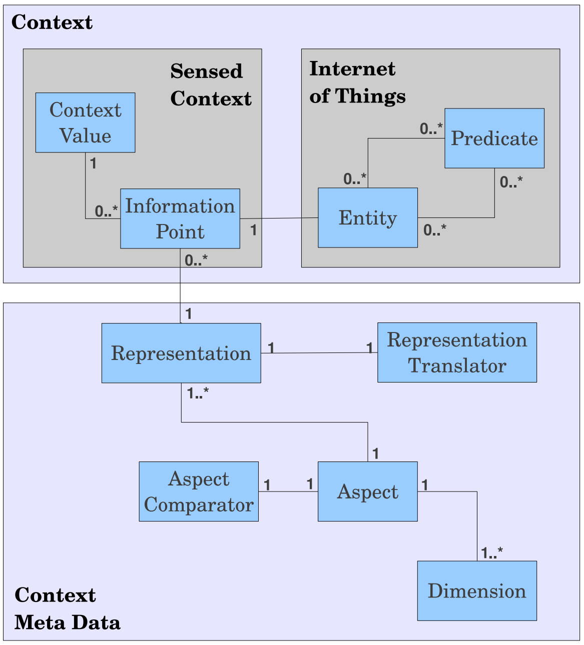

Fig. 2. The Distributed Context Information Integration Model (DCII)

things can range from sensors and actuators to virtual information sources such as social networks, media, people and infrastructure.

The CII model can be extended with new sub-concepts of *Entity* and *Information-Source*. These concepts would be presented as classes following a standard interface. This integration would be made possible by adaptive software techniques such as a combination of computational reflection, automated compilation and dynamic class loading. Agents, applications and services reside above and use the meta-model as a source of data and deriving context information.

The previous implementation [8] is focused on a largely centralized approach, where the model resides on a single end point with the CUAs reporting sensor information and sensor modifications. This introduced a single point of failure for the model. Distributing this model to create the DCII mandated two key departures from the CII model. Firstly, the concept of *Information Sources* has been superseded by *Information Points* and extends the model to support the changes reflected in section 3.1. Actuators now become Information Sinks with the following reverse properties of Information Sources:

– Comparing input values : a Fahrenheit value could be passed into the end point and compared with the threshold value of the actuator
– Representations and translations : exposing multiple representations for accepting and translating input, an actuator implemented in Centigrade could accept temperature settings in Fahrenheit

Secondly, we look at *Presentities* as regarded by [5] as: *An entity that posseses and is defined by its presence information.* We regard presentities as being separate in behavior from other entities such as sensors or actuators. We add a *Schema Entity* which is attached to a presentity and describes the current model of sensors and actuators that provide context information supporting the presentity. In this way the *watchers*, regarded by [5] as *the entities interested in a presentity's prescence*, may has access to a defined real-time picture of all the information points related to a presentity. It can then choose which sensors to use in order to deliver its services.

Schema entities, however have one additional property; they expose a *publish/subscribe* interface. We take this approach in order to avoid having to synchronize large datasets distributed around the architecture. Watchers can therefore subscribe to a schema and be notified as it changes. There is no need to issue queries to nodes or databases or for watchers to be concerned with checking for updated presence information.

3.3 The Overlay

The CII model details an implementation on top of the existing DCXP framework. DCXP was built as a context information exchange framework utilizing an underlying Chord [25] P2P overlay. Our requirements with respects to real-time constraints, scalablilty and peformance motivates a substitution of the Chord overlay with the more advanced PGrid architecture [1]. PGrid introduces several advantages over the existing Chord overlay including:improved search functionality, more efficient routing, complete decentralization and load balancing. From this we gain two key functionality required to obtain the real-time qualities needed for supporting an object based context model.

Firstly, a P2P overlay on which to implement the DCXP protocol; a key requirement for a the model. Based on the PGrid support, the overlay is responsible for routing DCXP messaging primitives around the overlay, sending messages between nodes, resolving nodes responsible for a UCI or a range of UCIs as well as building and maintaining the overlay. The DCXP layer implements the DCXP primitives described in: [16] along with the extensions described in section 3.1. This preserves the original architecture as well as the functionality of existing dependent solutions. It further maximizes interoperability by allowing endpoints incapable of supporting the complete interface to gain access to lower level user-based context information.

Secondly, we benefit from an advanced indexing layer native to PGrid, which provides for a distributed indexing service implemented on top of the overlay. We extend this, in order to define how different entities are described, indexed and queried within the index and refer to this as the Resource Index (RI). All resources such as entities, presentities and schema are identified by their UCI and stored within the RI. This provides for a quick searching mechanism for all entities within the architecture. A secondary result of using PGrid is that such an implementation would be more suitably adapted for highly dynamic and flexible

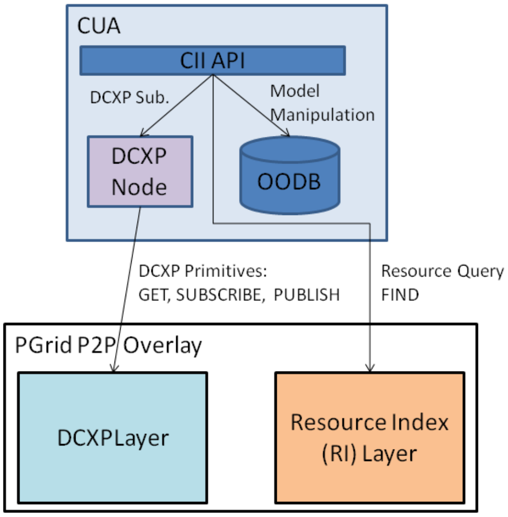

Fig. 3. Distributing the Meta Model

environments such as mobile networks which are underpinned by issues relating to heterogeneity and reliability.

3.4 The Distributed Meta Model

Section 2 addressed some of the issues surrounding the distribution of context within fixed and mobile computing environments. The CII model as it stands, meets its initial requirements with regards to representing context information. However the current implementation does so as a centralized solution consisting of an object-oriented overlay connected to a relational database. Objects are described, initialized, utilized, destroyed and persisted within this same space, i.e. on a single node. All requests must be sent to the same server for execution and while such a model creates the intelligence required, the real-time properties are undermined by the limitations of its scalability.

Distributing this model, while achieving and sustaining real-time targets, requires a rethinking of the underlying architecture in support of real-time properties. There are three main problems to be addressed by the proposed solution: firstly the need for a scalable distributed architecture for routing context information among nodes in real-time; secondly real-time querying and indexing infrastructure for locating entities and information across a distributed architecture. Thirdly, a means of constructing and manipulating complex object-oriented context models representing the highly dynamic real world interaction of presentities. A support for this is illustrated in figure 3. The first two are resolved by the DCXP and Resource Index layers respectively.

The third, the CUA or the Context User Agent is an extension of the existing CUA from a sensor information dissemination point to an intelligent context node with the ability to initialize, store and manipulate context models required to support applications and services being executed at an endpoint. With the original CII implementation, an object-layer residing over a relational database, was co-located with the CUA. Citing the comparison of relational and object-oriented database sytems in [23], we replace the RDMS with an OODB component, negating the need for an object layer above persistence. In addition to this we gain speed by exploiting the advantages of OODB with respect to real-time performance [7]. The native implementation of the *entity-predicate-entity* triples further improves performance over the original CII implementation which is required to fetch and construct the CII models in the object layer.

Within a DCII, an object representation of a sensor or actuator is defined. The object is stored locally in an OODB and also indexed in RI. The same happens for all entities including presentities. The application or service responsible for the creating and maintaining the object further adds relationships to its schema, define translation rules and access permissions.

When a sensor or actuator is added to the DCXP layer it is given a UCI of the format:

dcxp://user[:password]@domain[/path[?options]]

The original DCXP interactions remain. The local CUA implements an object representation of the sensor as described by the CII model and stores it in the local OODB. Each object is assigned a UCI of the format:

dcii://user[:password]@domain[/path[?options]]

The object UCI is stored in a DCXP-like distributed hash table to facilitate real-time resolution of UCI to object queries. The object is then serialized and indexed by the RI layer for supporting searching and browsing.

An application requiring use of a sensor implements a reference to sensor object. The local object layer is then responsible for resolving the UCI, fetching the object description, initializing the object and translating this into the *GET* or *SUBSCRIBE* primitives of the underlying sensor. It further maintains this relationship until the sensor is no longer required. This interaction is local to the application and its CUA. The CUA responsbile for the sensor object is not required to particpate in this relationship.

Such object-application relations are straightforward, however when an application requires the context-model for a presentity in order to deliver some service or user experience, the model then introduces more complexity. Here, we utilize the schema objects described in section 3.2. The CUA local to a presentity maintains its schema, adding or removing sensors or other information sources that contribute to its prescence profile. Where an application requires the Information Points relating to a person, a subscription is made to its schema, the schema is initialized on the CUA local to the application. This liberates the CUA hosting the presentity from maintaining resources capable of supporting all the services connected to it. The schema is then resolved to a DCXP

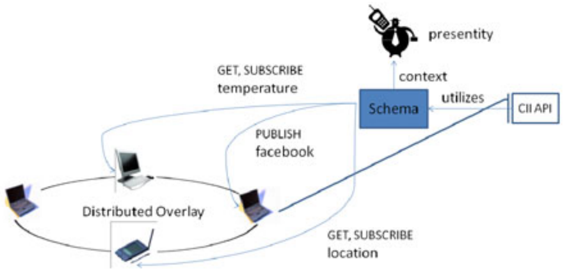

Fig. 4. Distributed Meta Model

PUBLISH/SUBSCRIBE with the concrete sensor/actuator sources. The application's local CUA is then responsible for maintaining and updating the local representation of the schema relative to the application's requirements. With this loose coupling, applications can ignore schema changes that do no impact on their performance, eg: an application that requires location information may ignore schema changes that adds new temperature sensors but would update the context-model if a new location sensor is added to a presentity. Such an approach is beneficial in resource-constrained devices where only a subset of the schema may be implemented.

3.5 Evolving Context Schemata

The schema entity detailed in section 3.2 introduces support for the dynamic behavior of context information. We need to maintain a model that continually desribes the current situation of a presentity. This we regard as schema evolution; the continual adding or removing of information sources that are available to a presentity and subsequently its watchers. Model schema evolution will take place progressively, continually deriving its new state from application level interaction. Applications such as user agents can negotiate the addition of new sensors or actuators to a presentity's schema.

The watchers (applications, services, etc) can *SUBSCRIBE* to the schema object and receive notifications of any schema changes.

The evolution of the schema provides for personal preferences with regards to this problem. Schemata are seen as being infinitely composable and reusable such that a new schema may be constructed over existing schemata. Such an example might be the need to express the collaborative context of all the occupants in a room in order to derive an accurate context-model for the room itself. This however can be limited by the fact that the schemata are time constrained and encapsulate the composition of a subset sensors attached to a presentity P over time t.

The process of evolving of a schema is triggered by the presentity itself establishing a recursive dependency where, a schema expresses a new context-model which is used by the applications. The applications through defined dynamic interactions allow for resource discovery which may in turn trigger the evolution of the schema. Applications dependent on a presentity's context do not need to discover and negotiate with sensors directly. Information sources will be seamlessly added or removed modifying the schema being used.

4 The Model as a Service

The CII model utilizes information sources from cloud based services such as social networking sites, messaging end points or any service interested in context information, thereby enabling the missing building block of cloud computing [12]. With a centralized model, such implementations are managed by a single end point or user is required to manage all the required interfaces. This reduces the "open" properties of any such framework and re-inforces the single point of failure issues described in section section2.

Interworking cloud services within the DCII model may however be done with relative ease while exploiting its distributed properties. Figure 5 shows a computer wishing participating within the architecture needs only to implement the basic services (P2P, RI, DCII) in order to interface with the model. With this extension, the model provides distributed access to its services to any external application or service. Services such as social networks, messaging platform or even existing prescence systems such as IMS can interopolate with and gain access to the information contained within the model. We can also expose the model in other formats which can enable additional data mining and knowledge discovery. At the core of this, we are proposing a model that permits us to exploit the benefits of distribution while sharing information with existing and legacy context centric systems.

4.1 Service Scenario

John Smith, is actively enagaged in social activities and is interested in using services that are capable of providing him a user experience based on his context. He uses Skype and Facebook and wants to be able to modify his status and profile in response to his context.

He subscribes to a profile updating service on the internet which is capable of modifying his Facebook profile. The company has a server which joins the DCXP overlay and implements a CUA. They now find John using his UCI and subscribes to his schema which is maintained by his local CUA, located in his house. His CUA maintains a track of his interaction with the connected things infrastructure and continually maintains and updates his profile as a list of connected objects referenced on a distributed architecture. The real-time qualities of such a solution allows for the implemented Facebook module to propagate this context information to his profile.

Simulataneously, his Facebook connection acts like an information source providing information about his social interactions, friends connected, messaging status, etc. these are translated by the translation component, attached to all

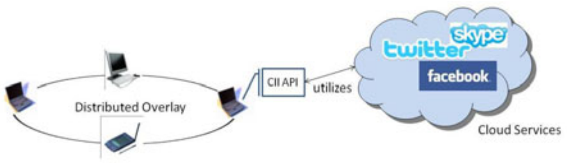

Fig. 5. Cloud Services Integration

information points in the model, into any representation required by another end point. As such his skype connector running on his local computer now displays his status as a combination of his current context and his Facebook profile , eg. *I am eating ice cream – 27°C, GPS[27.564, 65.4324], Humidity 78.5%, etc.* This adds more dimensions to his context profile and changes quickly as John moves around his daily routine. Applications looking at his context, knowing that John gets a fainting spell over 27°C can look at his current context profile to see if he has access to a temperature actuator in order to reduce his local temperature. Such applications are able to gain access to one as soon as John enters his house and adjust the temperature accordingly.

While this is achievable on current systems such as IMS, they do not store context profile information explicitly, rather a current set of values. There does not exist a means of idenitfying sensors within proximity. Secondly, centralization points to their inablility to scale well since all services depending on John's profile would be attempting to connect to the same WebService portal and potentially offloading computational tasks. Within our service capable model, this can be distributed and managed accordingly. John never has to think about adding computation resources at home to meet the demands of all the applications and services depending on his context. Such services can exist on nodes within the architecture subscribing to his profile and creating an experience in response.

5 Concluding Remarks and Future Work

In this paper, we presented a further extension to the CII called the DCII and an architecture for supporting the distributed dissemination, agregation and reasoning over such a model. This solution creates the support required to implement the real-time dependent context services conducive with an Internet of Things. Within this model, we address the issues surrounding the scalablity of centralized sensor information access by extending and using DCXP. We added new primitives to complement the existing set and enable support for actuators and resource relocation. This extends our current research to supporting the more accurate Wireless Sensor Actuator Networks as opposed to only sensing scenarios. We however, maintain the proven real-time information dissemination unachievable with previous approaches such as Senseweb.

In response to the need for an intelligent layer above DCXP, the DCII approach builds on the previous CII research by now including distributive support.

This, by extending the CUA to include an OODB and a DCII API; creating relations within a native database and removing the added layer of abstraction of the CII model. With this distribution we premise that we gain the ability to build and modify complex sensor relationships in real-time to reflect the dynamic realities of the Internet of Things.

We introduced the Resource Index to provide a means of indexing, searching and browsing context objects in real-time. This will benefit applications involved in data mining and resource discovery across the architecture. We also introduced the use of the sensor schema to model the evolving context information related to presentities. This we maintain through a publish/subscribe interface. We also addressed the interoperability of our solution with cloud services through service extensions allowing connectors and translators to both input and utilize information within the architecture. This creates a model capable of integration with social network services, messaging platforms or scheduling applications.

Future work includes implementing the architecture on top on the existing DXCP and subsequently conducting case studies comparing its performance with existing prescence based systems. We envision further research in the areas of linked sensors and subsequently the ability to dynamically index and rank sensors in terms of relativity to other sensors, entities and search queries across the system. By being able to usefully connect sensors and entities such as is with web content, we see the possibility of enabling ranking and searching much akin to modern search engines.

Acknowledgement. This research is a part of the MediaSense project, which is partially financed by the EU Regional Fund and the County Administrative Board of VÃďsternorrland.

References

1. Aberer, K., Cudré-Mauroux, P., Datta, A., Despotovic, Z., Hauswirth, M., Punceva, M., Schmidt, R.: P-Grid: a self-organizing structured P2P system. ACM SIGMOD Record 32(3), 29 (2003)
2. Barbara, D.: Mobile computing and databases-a survey. IEEE Transactions on Knowledge and Data Engineering 11(1), 108–117 (1999)
3. Camarillo, G., García-Martín, M.A.: The 3G IP multimedia subsystem (IMS): merging the internet and the cellular... David López (2004)
4. Chang, F., Dean, J., Ghemawat, S., Hsieh, W.C., Wallach, D.A., Burrows, M., Chandra, T., Fikes, A., Gruber, R.E.: Bigtable: A Distributed Storage System for Structured Data. ACM Transactions on Computer Systems (TOCS) 26(2) (2008)
5. Christein, H., Schulthess, P.: A general purpose model for presence awareness. In: Distributed Communities on the Web, pp. 24–34 (2002)
6. Crockford, D.: Introducing JSON (2010)
7. DiPippo, L.C., Wolfe, V.F.: Object-based Semantic Real-time Concurrency Control (1995)
8. Dobslaw, F., Larsson, A., Kanter, T., Walters, J.: An Object-Oriented Model in Support of Context-Aware Mobile Applications. In: Source, Chicago, pp. 1–16. SpringerLink, Heidelberg (2010)

9. Elettronica, I.: Semantic-based Middleware Solutions to Support Context-Aware Service Provisioning in Pervasive Environments. Informatica (2008)
10. Gu, T., Pung, H.K., Zhang, D.Q.: A service-oriented middleware for building context-aware services. Journal of Network and Computer Applications 28(1), 1–18 (2005)
11. Iqbal, R., Sturm, J., Kulyk, O., Wang, J., Terken, J.: User-centred design and evaluation of ubiquitous services. ACM Special Interest Group for Design of Communication, 138 (2005)
12. Joly, A., Maret, P., Daigremont, J.: Context-Awareness, The Missing Block of Social Networking. ijwus.net 6(2), 50–65 (2009)
13. Kansal, A., Nath, S., Liu, J., Zhao, F.: SenseWeb: An Infrastructure for Shared Sensing. IEEE Multimedia 14(4), 8–13 (2007)
14. Kanter, T., Österberg, P., Walters, J., Kardeby, V., Forsström, S., Pettersson, S.: The MediaSense Framework. In: 2009 Fourth International Conference on Digital Telecommunications, pp. 144–147 (July 2009)
15. Kanter, T., Pettersson, S., Forsstrom, S., Kardeby, V., Norling, R., Walters, J., Osterberg, P.: Distributed context support for ubiquitous mobile awareness services. In: 2009 Fourth International Conference on Communications and Networking in China, pp. 1–5. IEEE, Los Alamitos (2009)
16. Kanter, T., Pettersson, S., Forsström, S., Kardeby, V., Norling, R., Walters, J., Österberg, P.: Distributed Context Support for Ubiquitous Mobile Awareness Services. In: Context (2009)
17. Kanter, T.G.: Going wireless, enabling an adaptive and extensible environment. Mobile Networks and Applications 8(1), 37 (2003)
18. Lee, J., Song, J., Kim, H., Choi, J., Yun, M.: A user-centered approach for ubiquitous service evaluation: An evaluation metrics focused on human-system interaction capability. In: Lee, S., Choo, H., Ha, S., Shin, I.C. (eds.) APCHI 2008. LNCS, vol. 5068, pp. 21–29. Springer, Heidelberg (2008)
19. Lyytinen, K., Yoo, Y.: Introduction. Communications of the ACM 45(12), 62 (2002)
20. Nurseitov, N., Paulson, M., Reynolds, R., Izurieta, C.: Comparison of JSON and XML Data Interchange Formats: A Case Study. cs.montana.edu (2009)
21. Oliveira, J., Carrapatoso, E.: Using context information for tailoring multimedia services to user's resources. In: Krishnaswamy, D., Pfeifer, T., Raz, D. (eds.) MMNS 2007. LNCS, vol. 4787, pp. 138–148. Springer, Heidelberg (2007)
22. Schmidt, A., Beigl, M., Gellersen, H.-w.: There is more to Context than Location (1998)
23. Smith, K.E., Zdonik, S.B.: Intermedia: A case study of the differences between relational and object-oriented database systems. ACM SIGPLAN Notices 22(12), 452 (1987)
24. Sousa, J.P., Carrpatoso, E., Fonseca, B.: A Service-Oriented Middleware for Composing Context Aware Mobile Services. IEEE, Los Alamitos (2009)
25. Stoica, I., Morris, R., Karger, D., Frans Kaashock, M., Balakrishnan, H.: ACM SIGCOMM Computer Communication Review 31(4), 149 (2001)
26. Stonebraker, M., Aoki, P.M., Litwin, W., Pfeffer, A., Sah, A., Sidell, J., Staelin, C., Yu, A.: Mariposa: a wide-area distributed database system. The VLDB Journal âĂŤ The International Journal on Very Large Data Bases 5(1), 048 (1996)
27. Ulusoy, O.: Transaction processing in distributed active real-time database systems. Journal of Systems and Software 42(3), 247–262 (1998)
28. Wang, X.H., Zhang, D.Q., Gu, T., Pung, H.K.: Ontology based context modeling and reasoning using OWL. IEEE, Los Alamitos