An RFID-based platform supporting context-aware computing in complex spaces

Ayis Ziotopoulos, Margarida F. Jacome and Gustavo de Veciana
Department of Electrical and Computer Engineering
University of Texas at Austin Austin, TX 78712
ziotopou,jacome,gustavo@ece.utexas.edu

Abstract

Ubiquitous computing promises to both assist us in everyday tasks and enhance our capabilities. Key elements towards fulfilling this goal are exploiting the physical and logical context in which computation occurs, in order to scope the interaction between users and applications. In this paper we describe an RFID-based platform allowing mobile entities to transparently associate with ubiquitous applications running within complex physical spaces. Entity-application associations occur only for applications within a physical space whose services are within predefined sets specified by an entity for that space type. Mobile entities in our platform are uniquely identified by a temporary ID, and further characterized by a set of attributes describing the above-mentioned set of services. Computation is mediated through the exchange of protocol messages guarded by such attributes. Furthermore, relevant application state is distributed on each mobile entity through a set of messaging boards, enabling a targeted form of communication and cooperation among ubiquitous applications. In this paper we report on our experience experimenting with this platform. Our initial results indicate that this platform is suitable for current RFID technology and exhibits low-cost, scalability and privacy.

1 Introduction

Fast paced advances in semiconductor technology, low-power design, and wireless and sensing technology, provide a rich infrastructure to support the next-generation of information technologies, adhering to the so called *ubiquitous computing* paradigm [6]. Perhaps the most important characteristic of ubiquitous computing is that "it disappears into the background by weaving itself into the fabric of everyday life" [6], creating what is usually referred to as 'ambient intelligence'. In such 'smart' physical environments, a myriad of 'invisible' computers will constantly assist us in everyday tasks.

The realization of the scenario just described requires overcoming significant obstacles, many of which have at their root the employment of abstractions more appropriate for the PC world than the new field of ubiquitous computing. To name just a few, the weight, energy consumption, monetary cost and interface of future mobile computers will significantly influence the adoption of the ubiquitous computing paradigm. Additionally, issues like poor wireless connectivity, effective sharing of limited bandwidth and privacy threats call for more 'localized' user-application interactions as opposed to 'centralized' interactions where service is provided from a few master points only.

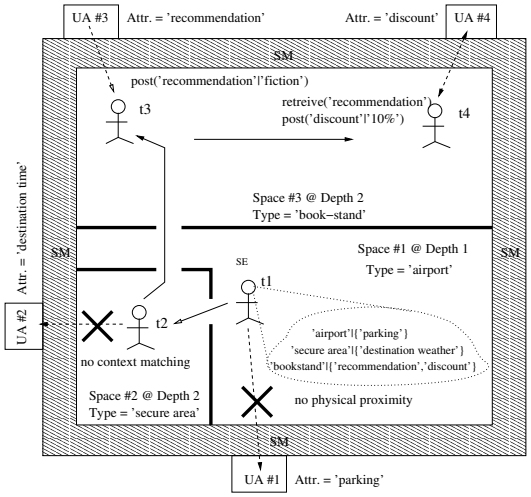

Figure 1. Main elements of a complex space realizing our platform

In this paper, we advocate for a ubiquitous computing platform that employs the use of context in order to tackle the above issues. We focus on two types of context: *physical* and *application* context. For the scope of this paper, the physical context expresses the type of the space hosting an application e.g. airport, book-stand, etc. while the application context expresses the type of service an application offers e.g. parking, book recommendation, etc. In this paper we assume that the spaces where entities reside

are *complex* i.e spaces where one space is 'nested' inside another, e.g. in Fig. 1, we see that the book-stand (defined to be at depth 2) lies inside the airport (defined to be at depth 1). We chose to focus on these types of spaces because they can easily model the vast majority of spaces humans spend most of their time e.g. homes, business buildings, shopping malls etc. Our work is motivated by passive RFID technology being a candidate fabric for our platform. Passive RFID technology offers distinctive advantages in terms of cost, weight and energy consumption giving our platform some of the desired properties alluded to before almost 'by construction'. Extensive use of RFID technology as a means to store context and data is assumed. Specifically we propose a platform in which mobile entities e.g. humans, robots, pets carry with them a minimalistic artifact to hold information describing the current context and application data only. All the rest e.g. energy, computational power and logic are provided by applications embedded in the environment.

Fig. 1, shows the three key elements of our platform: (1) ubiquitous applications (UAs); (2) serviced entities (SEs); and (3) a space manager (SM). Such a physical space will be denoted from now on as a *managed* space. The logic underlying the operation of a managed space is as follows. Local ubiquitous applications, running on the myriad of (smart) objects populating the particular physical space, provide the actual computing services. E.g. in Fig. 1, UA #1 represents a ubiquitous application offering parking services (e.g. directions to parking space and prices) inside the airport. The 'serviced entities' represent mobile entities e.g. people, robots, pets, etc., that temporarily enter the managed space, and potentially benefit from its services. E.g. in Fig. 1, the little human figure that moves around is modeled as a SE. Each SE carries with it information describing the current context as well as application data. Only when one such SE comes to physical proximity with a given smart object, the set of ubiquitous applications running (locally) on that specific object, will transparently react depending on whether there is a context match (as specified by the SE and the application). The reaction can be the provision of the object's service or total lack of it. *No* involvement from the SE is required. E.g. at time $t1$ and $t2$ the SE in Fig. 1, does not receive any service since it is not close to UA #1 and does not share any interest in UA #2's services(UA #2 lets SEs find out about the time at their destination, attribute 'destination time', while the SE only cares about the weather at its destination inside this space, attribute 'destination weather'). Part of an application's reaction can be writing its data to the SE, data that will be available for other applications to inspect and update when the SE comes to physical proximity to them. This mechanism realizes a form of application communication. E.g. at time $t3$ UA #3 that runs inside the book-stand makes a recommendation about a fiction book to the SE, attribute 'recommendation',

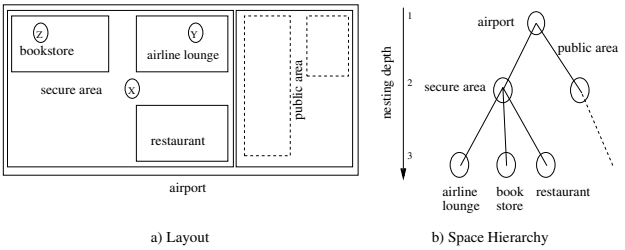

Figure 2. Airport Example

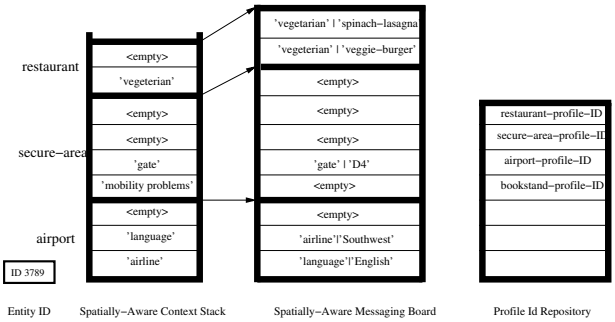

Figure 3. Key SE's Abstractions

by posting an appropriate message *on* it. At $t4$ this information is exploited by UA #4 to give a 10% discount to the SE, attribute 'discount'. (This would not have happened unless the SE had specified that inside book-stands it cares about the attributes 'recommendation' and 'discount'). Finally the space manager, depicted as a layer that *logically* surrounds all spaces in Fig. 1, provides an OS-like (distributed) layer, i.e., offers a set of basic resource management and operational services/primitives to be used by, both, local UAs and SEs. Throughout the paper, we illustrate our approach using a running example of an airport structured as a managed space. To make our scenario more realistic we assume that the SE models a human passenger and all the SE abstractions are stored in an RFID chip in the passenger's ticket.

The remaining of this paper is as follows. In Section 2, we discuss the key abstractions underlying the proposed platform. Section 3 presents the actual protocol used by our abstractions. Finally, in Section 4 we discuss a low cost implementation platform on top of RFID technology – this 'low end' implementation demonstrates the immediate benefits of our approach's inherent simplicity.

2 Key Abstractions Supporting Context-Aware Computing

In this section, we expand more on some elements of the key abstractions: the *space type* and the *SE*.

2.1 The Space Type and Its Attributes

As alluded to in the introduction, a key objective of our approach is to enhance entities immersed in a given physical

space with information to be used by ubiquitous computing applications to seamlessly establish the potential relevance of their services to such entities. Towards that, our approach consists of abstracting the very services meaningful inside a particular space in terms of attributes. A *space type* can thus be defined as a collection of attributes.

Fig. 2a, presents a simplified view of an airport. This figure shows several space types e.g. a generic airport space and other more specialized ones e.g. a secure area. Attributes abstracting the relevant services in a e.g. restaurant include vegetarian food, kosher restrictions, low fat diet etc. while attributes relevant to a e.g. secure area used for departures include gate information/navigation (denoted from now on as 'gate'). While above we give suggestive names to space types and attributes, truly those are just identifiers. Thus, the combination e.g. $< secure\ area, gate >$ will unambiguously abstract gate related services inside the secure area of the airport. Naturally, the definition of such service taxonomies will ultimately require a flexible standardization effort, enabling new space types and attributes to be added to the standard, as their need becomes apparent.

2.2 The Serviced Entity: SE

A serviced entity or SE is defined by the following set of abstractions which are physically stored on it:

Temporary ID. Requiring entities to provide some form of 'permanent identification' upon entering a managed space could potentially infringe on one's privacy and is actually unnecessary – rather, in our approach each SE entering a managed space is simply assigned a temporary ID – see Section 3 for details on the protocol used for the purpose. (Note also that privacy considerations may further dictate that such temporary IDs are to be periodically dropped and reassigned.)

Spatially-Aware Context Stack. The spatial organization assumed in this paper supports complex spaces, the one nested inside each other. We use a tree to encode this structure, see Fig. 2b. The actual location of any entity can thus be defined as a path from the root of the tree to the node corresponding to the 'innermost' space where the entity is currently located. For example, entities X and Y in Fig. 2a, are currently located in 'airport→secure area', and 'airport→secure area→airline lounge', respectively.

Services in our platform are related to the spaces in which they are offered. A stack is used for representing that, since it naturally captures the containment relationships alluded above. The *Spatially-Aware Context Stack* is responsible for holding the attributes characterizing the classes of services the SE is interested in, in the context of its current physical location. Stack frame (or entry) *i*+1 represents a physical space enclosed within the physical space associated to stack frame *i*. Thus, for example, an entity positioned at Y in Fig. 2a, will have its stack filled as symbolically depicted in Fig. 3, i.e., with frames 1, 2 and 3 of the stack representing the airport, the secure area, and the airline lounge, respectively. Fig. 3, shows a simple context stack for an airline passenger – attributes characterizing services for this passenger in the broad context of the airport may include his/her airline, language and so forth.

Naturally, while at a given position, the entity is characterized not only by its attributes for the innermost space (stored in the frame currently at the the top of the stack), but also by its attributes associated to all of the enclosing spaces – that is, it continues to be sensitive and respond to applications and messages addressing it in the context established by all such enclosing spaces (established by the remaining stack frames). As the entity moves around across different spaces, frames that are no longer relevant are popped from the stack and new relevant ones are pushed into it – Section 3 gives details on the adopted stack update protocol.

Location models have been used in the literature e.g. [1, 2] to provide applications with a way to describe information about the current location of an entity. In our approach, the underlying location model for a space is a tree but location information is *distributed* in the spatially-aware context stack of each SE. Moreover, the stack responds dynamically to the movement of the SE in contrast to statically mirroring the structure of a space.

The Profile Repository of an Entity. A *profile* is a subset of a space's attributes, defining a compatible/coherent set of services within that space. In practice, we use standardized sets of attributes for each space. Moreover we expect that, for many space types, it will be possible to define a number of typical profiles, that can be then directly adopted by entities, when they enter a space of that particular type for the very first time. Each such profile will be encoded using an ID, the *profile ID*. Entities should choose the profile that best describes the services they are interested in, for a given space. The fact that the profile used will typically be a proper superset of the actual services of interest to the user adds to the privacy of the platform, since a potential attacker will be provided with information only a subset of which, is true. The profile is a novel anonymizing mechanism that nicely fits in a ubiquitous environment. Each SE holds a repository of its profile IDs for relevant space types – see Fig. 3.

The Spatially-Aware Messaging Board. Communication and cooperation between ubiquitous applications in our platform is scoped by the SEs it is intended to service. SEs contain messaging boards for that purpose. For each space held in the context stack of an SE, a corresponding messaging board is defined. As depicted in Fig. 3, messages posted by ubiquitous applications in these boards are indexed by attributes. In the simple example shown in Fig. 3, there is a message posted on the 'secure area' messaging board of the particular entity reffering to the passenger's airline. (In the figure, for clarity, the attribute used to index each message

is separated from the body of the message by a vertical bar.) Only messages guarded by attributes contained in the SE's profile can be posted to a messaging board. Thus, attributes in our platform have a *filtering* functionality. This highly focused form of communication adds to the scalability of our design in contrast to other centralized solutions found in the literature using blackboards e.g. [4].

Supporting Persistent State. In addition to the abstractions presented previously, a persistent version of the profile repository and the messaging board exists, independent of spatial context. E.g. a message describing the allergies the patient suffers from should be stored in a persistent way independent of the space the passenger is in. Due to space limitation reasons we do not discuss this feature any further.

3 Companion Protocol

In this section, we describe the protocol used with SE devices implementing our abstractions.

3.1 Protocol Messages

Table 1 summarizes the messages defined in our protocol. They are organized into two groups: messages initiated by ubiquitous applications and messages initiated by the space manager. The first two columns of the table describe the function of the message and provide an abbreviation name for it and the third column specifies the message's sender and receiver.

The general pattern after which a message is formed obeys to the following structure: $< space\ type >< service\ driven\ identification >< data >$. The space type field is self-explanatory. The service driven identification is a concatenation of an attribute and the ID of the destination entity – yet, depending on the circumstances, a message may specify only one, or none, of these, by using wild-cards in the corresponding fields. The use of such wild-cards allows sophisticated functionality, e.g., a wild-card in, both, the attribute and the ID fields, enables a message to be broad-casted to all nearby SEs – which will then acknowledge their presence, by replying with their IDs. Finally, the data field, if present, carries data, e.g., messages to be posted on a messaging board.

3.2 Overview of Protocol Capabilities

The relevance of ubiquitous applications' services to SEs is assessed by the former using the *qping* message. In that message the attribute of interest should be placed in the appropriate message field. SEs that have subscribed to the particular attribute, reply with their temporary ID, asserting their interest. Otherwise, the message is filtered/ignored by the SE. For example, a ubiquitous application wishing to query a SE for interest in book recommendation, will send a *qping* message with the space type 'book-stand' and attribute 'recommendation' and a wild-card entity ID. If the SE has subscribed to the attribute 'recommendation', it will reply with a *qping_rsp_t*, containing its ID.

	Message Description	Message Name	Direction
SM Messages	announce managed space	man_space	SM → SE
	announce space identification	space_id	SM → SE
	request managed space_id	req_id	SE → SM
	assign managed space_id	assign_id	SM → SE
	reset SE	reset	SM → SE
	request spatially-aware profile	req_profile	SE → SM
	send spatially-aware profile	send_profile	SM → SE
AM Messages	qualified ping	qping	UA → SE
	qualified ping response true	qping_rsp_t	SE → UA
	qualified ping response false	qping_rsp_f	SE → UA
	post message	post	UA → SE
	retrieve messages	retrieve	UA → SE
	retrieve messages response	retrieve_rsp	SE → UA
	retrieve messages response end	retrieve_end	SE → UA
	reinstate message	reinst	UA → SE
	reinstate message response true	reinst_rsp_t	SE → UA
	reinstate message response false	reinst_rsp_f	SE → UA

Table 1. Summary Description of Protocol Messages

The protocol supports spatial-adaptivity as follows. Each actual (sub)space inside a managed space advertises its presence through the *space_id* message, indicating its type and level of nesting (i.e., *depth*). Upon receiving such a message, the SE can determine if this is a new space, by contrasting its type to that currently residing in its context stack, and the same depth. [1] If it is, the SE informs the space manager (SM) of its profile ID for the new space type (using the *req_profile* message), and receives from it the corresponding set of attributes (via the *send_profile* message). This causes the SE to add a new frame to its context stack, at the appropriate level, holding the attributes just received, and also instantiate a new messaging board for that space.[2] After that point, the SM is not involved in any other interaction with that SE. All subsequent interactions for that SE happen with the ubiquitous applications. This 'bootstrapping' functionality of the SM adds to the scalability of the system.

Communication/cooperation between ubiquitous applications, in the context of SEs, is supported through posting and retrieving of messages in appropriate messaging

[1] Driven by our implementation which relies on RFID technology we assume that space identification messages are of limited range and high directionality, this feature of RFID technology has been exploited also in [3], therefore there is no 'crosstalk' between messages by different spaces. The rest of the messages do not require this assumption since the contained space id can be compared against the established space id's in the context stack.

[2] The addition of a new frame loosely respects the semantics of a stack, in that all existing frames above the one just added are discarded/popped adds to the overall scalability of the platform.

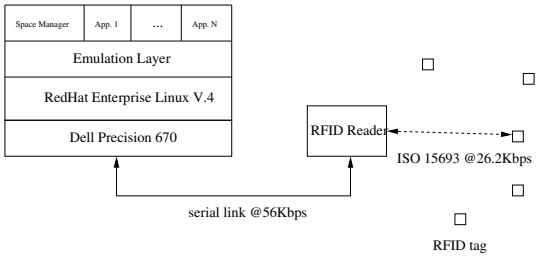

Figure 4. Software Emulation Stack

boards, using *post* and *retrieve*, respectively. All posted messages are guarded by attributes, making the messaging boards operate as associative memories. The management of the content of such boards is left entirely to the local ubiquitous applications responsible for creating and accessing the corresponding messages. Namely, when an application reads the set of messages indexed by a particular attribute, these messages are automatically removed from the messaging board – yet, the same application may post all or part of them again, exactly in their original form, or partially modified. We defined a special message (*reinst*) to allow such reposting in an efficient way. [3] The remaining protocol messages are mostly administrative, and will not be discussed due to space limitations. This protocol has been implemented, – see details in Section 4.

4 Prototyping Efforts

The abstractions comprising a SE embody most of the innovative aspects of our approach, and, most importantly, their cost/complexity define how popular can this type of technology eventually become. Details on our prototyping efforts are given below.

Initially, we implemented a hardware prototype of an SE in Verilog as a means of estimating the complexity of our abstractions. Our design takes 15000 gate equivalents, a number comparable to the gate count found in current passive RFID tags [5], demonstrating that this protocol could be targeted at that technology.

In order to assess the expressiveness of our protocol, we prototyped a platform for developing ubiquitous applications strictly relying on it. In order to overcome the lack of artifacts realizing an SE we developed a software middleware layer that emulates an SE operation on top of ISO 15693 RFID tags and exports a suitable API for implementing the space manager and applications alluded in Section 2. This implementation does not violate the principle of our platform since messages coming from an SE are always responses to a previous message from an UA or the SM. Therefore, the UA or SM expects a reply and can provide the power to an RFID-based SE to send it. Our middleware

[3] Work on the ***reinst*** message is still in progress, to efficiently support subsets of arbitrary size.

Message Sequence	Time(sec)
post	0.67
qping, qping_rsp	0.23
retrieve, retrieve_rsp	0.36
reinstate, reinstate_rsp	0.3

Table 2. Timing Results

allows us building and testing applications under realistic scenarios. Our experimental setting is shown in Fig. 4. Evaluating our platform we built synthetic applications that enabled us to experience the principle of operation of this new paradigm. Moreover we built a simplified version of the Airport scenario described in Section 2.

Additionally, we performed measurements on our platform to assert that our platform's time performance is suitable for adoption by current entities. The actual times per message exchange are shown in Table 2. The times mentioned are end-to-end measurements. Our emulation approach has a significantly high overhead since our protocol's messages are encapsulated in a series of RFID messages. E.g. a POST message in our implementation is implemented with 4 READ_MULTIPLE_BLOCKS messages and a WRITE_SINGLE_BLOCK message of the ISO 15693 RFID standard. Factors contributing to the times achieved include the time communicating through the RF medium, reading/writing to the tag, middleware execution overhead and the cost of context-switched to the OS for writing to the serial port. Nevertheless, the performance achieved, allows adoption of our technology even with today's limited support for our protocol/abstractions.

5 Acknowledgment

This work is supported in part by NSF Grant CSR-EHS 0509355.

References

[1] M. Bauer, C. Becker, and K. Rothermel. Location models from the perspective of context-aware applications and mobile ad hoc networks. *Personal Ubiquitous Comput.*, 6(5-6):322–328, 2002.

[2] C. Becker and F. Durr. On location models for ubiquitous computing. *Personal Ubiquitous Comput.*, 9(1):20–31, 2005.

[3] L. Ho, M. Moh, Z. Walker, T. Hamada, and C.-F. Su. A prototype on rfid and sensor networks for elder healthcare: progress report. In *E-WIND '05: Proceeding of the 2005 ACM SIGCOMM workshop on Experimental approaches to wireless network design and analysis*, pages 70–75, New York, NY, USA, 2005. ACM Press.

[4] B. Johanson, A. Fox, and T. Winograd. The interactive workspaces project: Experiences with ubiquitous computing rooms. *IEEE Pervasive Computing*, 1(2):67–74, 2002.

[5] S. E. Sarma, S. A. Weiss, and D. W. Engels. Radio-frequency identification: Security risks and challenges. *Cryptobytes*, 6(1), 2003.

[6] M. Weiser. The computer for the twenty-first century. *Scientific American*, pages 94–10, September 1991.