THE PERVASIVE ADVERTISING SYSTEM FOR MOBILE DEVICES

Mohammed Mahmudur Rahman
Zinnia Sultana
Lecturer, Dept. of Computer Science & Engineering,
International Islamic University Chittagong, Bangladesh.

ABSTRACT
In pervasive advertising distribution systems, the relevance of advertisements sent to users is an aspect of paramount importance, because it benefits both the advertiser, who increases the likelihood of a purchase by the user, and the user itself, who uses the advertising channel with greater satisfaction. In order to send relevant advertisements, an advertiser must know the situation, the interests, the activities and the intentions of users: in one word, their context. However, the disposal of this personal information to advertisers has clear repercussions on users’ privacy. The purpose of this work is the design and implementation of a pervasive advertisement distribution platform for mobile devices which maximizes the relevance of advertisements sent to users while preserving their privacy. The relevance is computed by the means of ontological matching between the users and the advertisements’ contexts, while privacy is guaranteed by the fact that matching happens directly on the user’s mobile device, thus his context data are never sent to the advertisers.

Keywords: Pervasive, Context dimension, Ontology, Reasoning, Module

I. INTRODUCTION
The advent of the internet has revolutionized the field of advertising by opening a brand new path for reaching potential customers. The availability of new channels, capillarity and coverage of advertising distribution have seen and keep seeing a constant growth, mainly due to the swift technological evolution of the web and traditional media. However, this new pervasive advertising scenario has introduced the two fundamental issues of relevance and privacy. A key instrument for advertisement’s relevance maximization is user profiling: owning information about the user’s needs, situation, interests and intentions allows creating offers for products he really needs, or could potentially be interested about. In other words, in order to send relevant advertisements, the user’s context must be known.

Web pages content can be analyzed in real time both syntactically and semantically, generating a relevance-sorted summary of the page topics, and putting together a context-coherent advertising offer; or, in web search engines, the queried keywords can be analyzed together with the user’s search history, building a profile of his interests and consequently refining the criteria for choosing the most relevant advertisements in a range of available offers. Obviously, the more detailed the user’s context representation is, the more efficient these approaches become, and this basic principle has crossed the frontier of web advertising, making the ownership of personal information about users one of the most urgent priorities for the communication industry. This brings us to the second issue of pervasive advertising: privacy.

II. FRAMEWORK FOR PERVERSIVE ADVERTISING
The Pervasive Advertising project (Figure 1) consists in the development of a framework for pervasive advertising which aims to maximize the relevance of commercial offers while preserving users’ privacy. It is a client-server system, which involves just two entities: the advertiser and the user. The former is equipped with a Pervasive Advertising Hotspot, a WiFi router (the server) which he can use to create advertisements (Pervasive Advertising) in the form of multimedia coupons; each Pervasive
Advertising is bundled with a description of the correlated context, i.e., of the advertised product or service and of the target of users which the offer is addressed to. On the other hand, the user is equipped with an application installed on his mobile device (the Pervasive Advertising client), which he can use in order to specify a description of his context, in terms of current situation, position, interests, needs and so forth. The Pervasive Advertising client scans the surrounding networks in search of Pervasive Advertising hotspots; when it finds one, it downloads the contexts associated to the advertisements and, once it has compared them with the user’s context, it displays the offers whose relevance goes beyond a predefined threshold, discarding the others. Both user’s and advertisement’s contexts are expressed semantically in form of instances of the same shared ontology, which defines the general context vocabulary, and the comparison consists in a matching between these two instances: therefore, contexts are defined and compared in a semantic fashion, partially solving the ambiguity issue that affects purely syntactic matching, which is the most widely adopted matching method. The advertisements are sorted by their matching score with user’s context; in case of a tie, the can be sorted at a finer-grained level, by the means of a syntactic comparison between two sets of keywords, the first being user-defined and the other being attached to the advertisement’s context. The entire process is completely anonymous: a Pervasive Advertising hotspot is open to any platform client, and sends its advertisements and the associated contexts without the need of any user authentication; besides, the client never sends information about the user’s context to the advertisers, thus protecting his privacy.

Pervasive Advertising enables small businesses to reach customers who need their services in a context-aware, low-cost and opportunistic fashion. By means of the Pervasive Advertising infrastructure, different shops may federate in a community, thus sharing the common infrastructure and reducing implementation costs. The infrastructure may be kept independent of telecommunication providers, since it can be implemented using wireless-LAN (Local Area Network) technologies, for example by means of WiFi and WiMAX hotspots; this also implies that customers only need a WiFi-enabled device and our mobile application (the Pervasive Advertising client) to receive the advertisements, without paying for network traffic. Also the merchants easily benefit from this very cheap solution: in principle, not even an internet connection is strictly needed to send their advertisements. The same very light infrastructure may be used also for conveying more information, not strictly business-related, like museums, cultural events, some mobility info and the like. Another advantage consists of the possibility for business participants to suggest to their customers other related businesses belonging to the same community, in order to increase the community volume of trades. We believe this to be a novel approach w.r.t. the state of the art of pervasive advertising, regarding both the relevance maximization problem and, particularly, privacy and advertising ethics.
Pervasive Advertising is a mobile advertisement distribution system with client-server architecture and two main actors: the user and the advertiser. The client and server components consist of two distinct applications running on the user’s mobile device and the advertiser’s WiFi hotspot respectively; the two actors interact with these applications independently and, most importantly, anonymously.

Advertisements are distributed as Pervasive Advertising. A Pervasive Advertising is a digital document describing a certain number of commercial offers about goods or services, and encapsulates context data defining which users the advertisement is targeting. Pervasive Advertising are created by advertisers and reside on their hotspot devices.

A. Use cases

Each actor participates in distinct use cases: advertisers create and distribute Pervasive Advertising using the tools provided by their Pervasive Advertising hotspots, while users interact with their mobile devices in order to define what they are looking for, and receive relevant advertisements.

a. User use cases

In a Pervasive Advertising-based system, user’s interaction is both active and passive. In order to be presented with relevant offers, he must actively provide the application with information about himself, his situation and what he’s looking for; then, he will be passively notified about advertisements that match his interests. The main tasks involved in a typical usage scenario, and summarized in Figure 2, are:

- Profile and queries creation and management
- Hotspot scan activation
- Visualization of relevant advertisements

b. Advertiser use cases

On the server-side of the system, the advertiser interacts with the administration application installed on his Pervasive Advertising Hotspot. The device is in fact a specialized WiFi router, which controls a Pervasive Advertising local network. The installed application provides a web server, which the clients can download Pervasive Advertising definitions from, and a backend UI, where the Advertiser can define Pervasive Advertising and their associated contexts. The advertiser’s tasks (shown in Figure 2) are:

- Pervasive Advertising creation and management
- Advertisement context
- Advertisement description
- Advertisements performance measurement

B. Shared models

User’s and advertiser’s applications share two models: the context model, which is used to represent contexts of users and advertisements, and the advertisement description model, which defines an ontology for semantic representation of commercial offers, goods and services.

C. Modules and interaction

Figure 3 summarizes the system architecture, showing different modules and their interaction; progressive numbers are used in order to highlight the typical interaction workflow, and will be used as a reference in the text.

The server application comes as an extension to OpenWRT, an operating system for embedded devices.
OpenWRT features a Linux kernel, a package management system for installing third-party applications, and a build system for creating binary distributions with a preinstalled set of packages. A Pervasive Advertising server is an OpenWRT distribution featuring Pervasive Advertising and other necessary extensions, installed on a standard WiFi router. The Pervasive Advertising extension has two purposes: it augments the router’s administration interface with Pervasive Advertising management functions, and provides a web server that allows clients to download Pervasive Advertising definitions.

a. Server modules
As shown in Figure 3, the advertiser interacts with two modules: the Hotspot Administration Backend (1) and the Pervasive Advertising Editor (2). The former is used for standard networking management: as any other router, this device manages a WiFi network that can be configured in order to, for example, allow internet access to its clients. This network will be used by clients in order to download Pervasive Advertising definitions.

![Figure 3: A schematic view of Pervasive Advertising architecture, with progressive numbers indicating a typical interaction workflow.](image)
Pervasive Advertising are created with the Pervasive Advertising Editor. This module provides an user interface for defining the advertisement’s context. During context definition, available dimensions and relative values are taken from the shared Context Model, while the Advertisement Description Model provides the semantic properties that are used in order to describe the advertisement contents.

The Pervasive Advertising Service module is a transparent service process which provides an external interface for clients. Currently, the only interaction between clients and server is the downloading of Pervasive Advertising definitions, so, in our prototype, we implemented this module as a minimal web server that listens for client requests, and sends stored Pervasive Advertising through HTTP protocol.

b. Client modules
In order to receive relevant advertisements, the user has to configure the Pervasive Advertising application entering his profile and search queries with the Profile and Queries Editor. Since both profile and queries are sets of dimension assignments, they are defined using the same interface, only at different times: profile is defined when the user initially configures the application, while queries are created, deleted, activated and deactivated during the whole application lifecycle, based on the evolution of user’s needs. The user interface for profile and queries allows the creation of dimension assignments; depending on whether the user is defining his profile or search queries. The product of the previous activities is the User’s Context. It is a collection of documents containing the dimension assignments coming from profile and queries, in a serialized format: one document for the profile, and one for each query. These documents are stored on device’s memory, and will be used later, during the matching process.

Whenever WiFi networks find a Pervasive Advertising network, it connects to the server’s Pervasive Advertising Service and downloads the available Pervasive Advertising.

The Pervasive Advertising Matcher module contains the core functionality of the whole system, as its purpose is deciding which Pervasive Advertising are relevant and which are not. The matching engine takes as input a couple of contexts, and outputs the matching degree between these two contexts. Each couple is composed of a source context and a target context. The source context consists in the combination of the assignments coming from user’s profile and one of the queries, while the target context comes from one of the Pervasive Advertising.
When the user wants to see relevant advertisements, he interacts with the Advertisement Viewer. This module reads the associations produced by the Pervasive Advertising Matcher during the last update process, and shows lists of the most relevant advertisements grouped by query. For each query, relevant advertisements are ordered based on their matching degree, higher degrees first. Due to screen size limits of mobile devices, these lists show, for each advertisement, just a small coupon, which is an image attached to the Pervasive Advertising or, if it is not available, an automatically generated coupon based on the advertisement contents. If the user selects a specific advertisement, a new interface is shown, which shows details extracted from the advertisement description, and allows navigation of other attached media and possible hyperlinks.

IV. ADVERTISEMENT AND CONTEXT MODELING

A Pervasive Advertising is an electronic advertisement consisting of an ontology containing the description of one or more offers related to the advertisement, and links to human-friendly resources such as graphic coupons. A Pervasive Advertising carries context data representing the target of the advertisements. Inside the ontology, these information are expressed semantically as assertions about advertisements, context and other involved entities. The semantic vocabulary used to make these assertions is defined in the following shared data structures. Figure 5.1 summarizes the structure of a Pervasive Advertising ontology, and relationships with the shared models.

A. Advertisement description

In Pervasive Advertising, the ontology describing the advertisements consists of assertions that use the advertisement description model as a vocabulary. A semantic classification of advertisements has a number of advantages: the model is easily extensible, the client application can use these information in order to choose the appropriate interface for presenting an advertisement based on the offer or offered item kind, or search the available advertisements for a specific item, exploiting the semantics in order to provide coherent results.

B. Context modeling

Context is a powerful instrument for relevance maximization in information retrieval and online advertising, and different modeling approaches have been widely studied in research. Knowing user’s context means being able to tailor content based on his current situation or activity, thus providing the information he’s looking for with greater accuracy, or offers for products and services he will likely be interested into.

In the Pervasive Advertising system, context is used in order to represent the current situation and needs of an user, or of all the users which an advertisement is targeting. Situation and needs are often bound to a specific location, and are constantly evolving; so, space and time are two key aspects that our context model should be able to describe. Pervasive Advertising’s application domain is real life; therefore, its context model must be flexible enough to represent a significant subset of all possible user’s circumstances, activities and kinds of products and services users could be looking for.

Formality is also important because, in Pervasive Advertising, context instances are exchanged between applications; the meaning of a context defined by an advertiser must be interpreted by the user application consistently w.r.t. the advertiser intentions. Definition and maintenance of the Pervasive Advertising context model are centralized: the model is predefined and distributed together with the client and server applications, which use it without modifying it. The applications will check for new context model versions and, when one is available, they will update their model accordingly; backward compatibility with older model versions will be enforced by model designers.

In Pervasive Advertising, user’s and advertisement’s contexts must be matched in order to determine relevant advertisements for the user; the ability to perform matching between context instances requires support for various features by the context model. In order for the
matching results to be reliable, the context model must support sound validation: this way, the client application can validate the contexts contained in downloaded Pervasive Advertising, and match only the valid ones against the user’s context. Besides performing formal validation, clients should be able to detect and ignore forged Pervasive Advertising; without this kind of validation, a dishonest advertiser could craft Pervasive Advertising with contexts that always results relevant during the matching process, thus nullifying the whole purpose of the system.

V. PERVERSIVE ADVERTISING CONTEXT DIMENSION TREE

In the Pervasive Advertising system the Context Dimension Tree (CDT) formalism will be only used as a foundation for the context model; for this reason, details about the generation of chunk configurations and view definition are out of scope.

Figure 5 shows an example CDT for the Pervasive Advertising domain. The CDT specification does not explain the semantics for the diagram root node; in our interpretation, it is a concept representing every possible situation in the application domain. For this reason, in the Pervasive Advertising CDT diagram we added a domains dimension, which is an implied meta-dimension for domain selection; this dimension becomes the new root node of the diagram, while the original root node becomes one of its concepts, representing the Pervasive Advertising domain. This modification doesn’t alter the CDT specification, as it only adds an interpretation for an aspect that was not considered in the original model.

The example in figure 5 contains five dimensions at the first level; besides the three aforementioned default dimensions, we have introduced two more dimensions: age and gender. These two dimensions model unchanging aspects of the user, so can be considered profile-related dimensions; on the other hand, the other dimensions (and their sub-dimensions) describe transient properties related to current user’s situation and needs, so they can be considered query-related. Figure 5.5 highlights this distinction between profile and query-related dimensions by grouping them in different colored boxes.

VI. ONTOLOGICAL REPRESENTATION

The Pervasive Advertising CDT is mapped to an ontological representation, thus enabling processing and reasoning over context data. In order to represent a CDT as an ontology, the first thing we had to do was choosing an appropriate formalism. In Pervasive Advertising, we represent ontologies by the means of the CA-DL description logic.

In our ontological representation of the CDT, context is expressed as a set of assignments of values to dimensions, and possible parameter assignments. The main differences from the original CDT specification are the organization of both dimensions and values in hierarchies, the enforcement of strong type-control over
assigned values, and the possibility of expressing n-ary constraints over the dimension values, while the CD is limited to binary constraints only.

Figure 6: The context model vocabulary ontology (OWL). Classes are represented as ellipses, properties as diamonds and data types as rectangles.

VII. PERVASIVE ADVERTISING MATCHING

Contextual matching is the core feature of Pervasive Advertising, as it allows to tailor relevant advertisements based on the current user situation and interests. One key difference between our approach and the others we have, is that context matching, in Pervasive Advertising, is user-centric: user’s context data never leaves his mobile device, thus ensuring his privacy and preventing advertisers from collecting valuable profile informations.

We will now give some formal definitions for the context model entities, in order to give an algorithm-oriented view of context instances as data structures. After each definition we explain how these formal entities are mapped to the corresponding ontology elements; then we will show how operations on these entities are mapped to queries against the ontological representation of context instances.

A context model \( CM \) is defined as:
\[
CM= <D,V,P,PV,r,p(parent(\.))> \quad \text{where:}
\begin{align*}
D & \text{ is the set of dimensions defined in the context model;} \\
V & = \{V_d, d \in D \} \text{ the set of allowed values } V_d \text{ for each dimension } d \in D;
\end{align*}
\]

\[
\begin{align*}
\mathbf{P} & = \{P_v, v \in V \} \text{ is the set of value parameters } P_v \text{ for each dimension value } v \in V; \\
\mathbf{P} & = \{PV_p, p \in P \} \text{ is the set of allowed values } PV_p \text{ for each parameter } p \in P; \\
r & \in D \text{ is the root dimension of the context model} \\
parent(\.)) & \text{ is a function}
\end{align*}
\]

A dimension assignment \( DA_d \) for dimension \( d \in D \) w.r.t. a context model \( CM \) defined as
\[
DA_d = (v, PA(\.)), \quad v \in V_d, \quad PA: P_v \rightarrow PV_p
\]

where:
\begin{itemize}
\item \( v \) is the assigned value;
\item \( PA(\.)) \) is a function that maps each parameter \( p \in P \) to a value \( pv \in PV_p \)
\end{itemize}

A context instance \( C \) w.r.t. a context model \( CM \) is defined as
\[
C=(D',DA(\.)), \quad D' \subseteq D, \quad DA: D' \rightarrow DA_d
\]

where
\begin{itemize}
\item \( D' \) is the set of assigned dimensions, such that, \( d \in D', \ parent( d) \in D' \)
\item \( DA(\.)) \) is a function that maps each dimension \( d \in D' \) to a dimension assignment \( DA_d \)
\end{itemize}

We will now discuss the actual algorithms; we give a bottom-up explanation, starting with more specific routines and finishing with the global algorithm that will
use them. Algorithm 1 is used in order to compute a scalar value called dimension distance. The algorithm takes as input a dimension \( d \) and a set of assigned dimensions \( D' \), and recursively computes the dimension distance as the number of dimension assignments in \( D' \) for dimensions that have \( d \) as an ancestor.

**Algorithm 1 dimensionDistance\( (d, D') \)**

1: \( \text{dist} \leftarrow 0 \)
2: for all \( d_{\text{child}} \in D' \mid \text{parent}(d_{\text{child}}) = d \) do
3: \( \text{dist} \leftarrow \text{dist} + 1 + \text{dimensionDistance}(d_{\text{child}}, D') \)
4: end for
5: return \( \text{dist} \)

Thus, the output value \( \text{dist} \) is in the range \([0,n]\), where \( n \) is the number of dimensions in the subtree rooted at \( d \).

**Algorithm 2 matchDimensionAssignment\( (d, C^{\text{source}}, C^{\text{target}}) \)**

Require: \( d \in D'_{\text{source}} \)

1: \( \text{score} \leftarrow 0 \)
2: if \( d \in D'_{\text{target}} \) then
3: \( \text{score} \leftarrow -1 \)
4: \( D_{\text{source}} = (v^{\text{source}}, P^{\text{source}}(\cdot)) \leftarrow D_{\text{source}}(d) \)
5: \( D_{\text{target}} = (v^{\text{target}}, P^{\text{target}}(\cdot)) \leftarrow D_{\text{target}}(d) \)
6: if \( \exists \text{matcher} = \text{getSpecializedMatcher}(d, v^{\text{source}}, v^{\text{target}}) \) then
7: \( \text{score} \leftarrow \text{matcher}(d, D_{\text{source}}^{\text{source}}, D_{\text{target}}^{\text{target}}) \)
8: else if \( \forall v_{\text{user}} = v^{\text{target}} = v \text{ and } P^{\text{source}}(p) = P^{\text{target}}(p) \forall p \in P_1 \) then
9: \( \text{score} \leftarrow 1 + \text{dimensionDistance}(D_{\text{target}}^{\text{target}}) \)
10: end if
11: end if
12: return \( \text{score} \)

Context matching, in Pervasive Advertising, is the result of the aggregation of the local matching scores for each dimension assignment in the user’s context, against the dimension assignments contained in an advertisement’s context. Algorithm 2 is used to compute the partial matching score for a dimension assignment. It takes as input a dimension \( d \), which is the assigned dimension, a source context instance \( C^{\text{source}} \), which is already known to define an assignment for dimension \( d \), and a target context instance \( C^{\text{target}} \), which is the context instance we want to test, and outputs a matching score, which is a real number in the interval \([0,1]\).

The resulting score is maximal (\( \text{score} = 1 \)) if the target context instance doesn’t define any assignment for sub dimensions of \( d \), while it is scaled otherwise. The special return value -1 represents an invalidating partial matching: it is returned when the target context contains an assignment for \( d \) to a different value w.r.t. the source context, and no specialized matcher is available that can compare those values. In this case, the standard matching behavior is to invalidate the whole target context, i.e., tell the main matching algorithm to discard the target context (by giving it a null global matching score) independently from the other dimension assignments: this is because the situation represented by the source context and the one represented by the target context belong to two distinct sets, so their matching is null regardless of the other dimensions.

Algorithm 3 is a helper subroutine that takes as input a set of assigned dimensions \( D' \) and returns a subset of its input that only contains leaf dimensions. A leaf assigned dimension is a dimension from \( D' \) that is a leaf in the hierarchical tree of dimensions.

**Algorithm 3 leafAssignments\( (D') \)**

1: \( D'_{\text{leaf}} \leftarrow \emptyset \)
2: for all \( d \in D' \) such that \( \exists d_{\text{child}} \in D' \mid \text{parent}(d_{\text{child}}) = d \) do
3: \( D'_{\text{leaf}} \leftarrow D'_{\text{leaf}} \cup \{d\} \)
4: end for
5: return \( D'_{\text{leaf}} \)

This algorithm is used in the main Pervasive Advertising matching algorithm in order to reduce the set of source dimension assignments to be matched against the target context instances: leaf dimensions are sufficient because their presence in the target context instance implies the presence of all the assignments of their ancestor dimensions to the same values they are
assigned to in the source context instance.

Algorithm 4 matchPervAds($C^{prf}, C^{qry}, C^{adv}$)

1: for $i = 1$ to $N$ do
2: \[ C^{user} = (D^{user}, D^{user}(\cdot)) \leftarrow C^{prf}_i \cup C^{qry} \]
3: \[ D^{leaf} \leftarrow leafAssignments(D^{user}) \]
4: for $j = 1$ to $M$ do
5: \[ m_{i,j} \leftarrow 0 \]
6: for all $d$ in $D^{leaf}$ do
7: \[ score \leftarrow matchDimensionAssignment(d, C^{user}, C^{adv}_j) \]
8: if $score = -1$ then
9: \[ m_{i,j} \leftarrow 0 \]
10: break
11: else
12: \[ m_{i,j} \leftarrow m_{i,j} + score \]
13: end if
14: end for
15: \[ m_{i,j} \leftarrow \left| m_{i,j} \right| \]
16: end for
17: end for
18: return $m$

We now present the matching algorithm used for computing the relevance of a set of advertisements for a given user. The Pervasive Advertising Matcher module takes as input a set of context instances representing user’s context (profile and queries) and advertisements’ contexts, and produces a mapping between each user-defined query and a (non-strict) subset of the advertisements whose elements are relevant for that query. Algorithm 4 is used for computing these mappings. More formally, for $N \geq 1$ queries and $M \geq 1$ Pervasive Advertising, the algorithm takes as input a set of context instances $<C^{prf}, C^{qry}, C^{adv}>$ where:

- $C^{prf}$ is the context instance representing the user’s profile;
- $C^{qry}$ is a set of $N$ context instances representing user’s queries;
- $C^{adv}$ is a set of $M$ context instances representing advertisements.

The returned value is a $N \times M$ matrix $m$, where $m_{i,j}$ is the matching score of the $j$-th advertisement against the $i$-th query. These scores are used by the Advertisement Viewer module to sort the advertisements by relevance for each query; the user will be able to see them, grouped by query, in decreasing relevance order, so that he will likely consider the most relevant ones first.

VIII. ONTOLOGICAL REASONING

The Pervasive Advertising matching process, reasoning is performed on context instances in order to (a) validate them and (b) infer implicit knowledge. Validation is the process of determining whether a given context instance is consistent w.r.t. the axioms and assertions defined in the Pervasive Advertising context model and in the context model vocabulary ontologies. Inference of implicit knowledge is used in general for answering queries in a consistent fashion, and in particular in order to "materialize" dimension assignments that are not defined explicitly in the context instance, but are made implicit by the interaction of other defined assertions and the axioms of the context model. Figure 7 shows a simplified example of inference of implicit dimension assignments.

In Pervasive Advertising we use a full backward-chained rule-based reasoning algorithm. Since backward-chaining reasoners just apply those rules that are needed in order to answer the current query, they have a lower memory consumption w.r.t. other reasoning algorithms. The only memory consuming activity is tabling, which is needed in order to ensure termination in presence of recursive rules such as transitive closures. Since queries against a CA-DL context model can be rewritten as a
non-recursive Datalog program, we can write a custom set of production rules that only expresses the constructors allowed in CA-DL, such that backward-chaining can be safely performed without tabling with the guarantee that it will terminate, thus achieving a very good reasoning efficiency while containing memory usage. This is what has been done in the Pervasive Advertising client application.

IX. PLATFORMS AND TECHNOLOGIES

A. Client application
The Pervasive Advertising client has been developed as an application for the Google Android platform. Android is an open-source software stack for mobile devices that includes an operating system, middleware and key applications. We chose Android because it was the platform that best satisfied the requirements: WiFi API, Availability of Semantic Web libraries.

B. Server application
A Pervasive Advertising server is a Wi-Fi router which (a) provides an administration interface that advertisers use in order to create Pervasive Advertising and (b) features a web server from which clients can download Pervasive Advertising definitions. Our prototype server is a Linksys WRT54GL router; this model features a Linux kernel, and the source code for the kernel and all the drivers is publicly available, which makes it an optimal platform for embedded application development.

We installed on the router the OpenWRT operating system. OpenWrt is described as a Linux distribution for embedded devices. OpenWrt provides a fully writable file system with package management. This allows users to customize the device through the use of packages to suit any application, and developers to build an application without having to build a complete firmware around it. One of the packages we installed on the router is LuCI, a framework of libraries and applications for the OpenWRT platform. LuCI provides a web-based administration interface which allows a fine-grained management of every device feature, from simple networking configuration to the setup and management of a RADIUS server or PBX station.

Moreover, the LuCI web administration has a modular architecture, and it can be easily extended with plugins that add entirely new sections to the interface. The Pervasive Advertising server application is a LuCI extension written in the Lua language. The MVC architecture of the LuCI administration makes the application highly maintainable, and further extensible; web server support is provided by the uHTTPd package, a tiny single threaded HTTP server with TLS, CGI and Lua support.

X. CONCLUSIONS
We did not perform a complete experimental field evaluation due to the cost of infrastructures for a realistic deployment scenario: therefore, this is surely one of the most urgent priorities. However, our testing
activity allowed us to highlight some directions of improvement:

The implementation of an infrastructure for measuring the performance of advertisements, while preserving user’s privacy and anonymity. The server application could automatically replace hyperlinks to external websites contained in the advertisement descriptions with hyperlinks to a local proxy service (implemented in the server application itself) that tracks those requests and then redirects to the linked website. The advertiser could then measure the advertisement’s performance by analyzing the number of requests for the websites linked in each Pervasive Advertising, as with classic web analytics software.

The implementation of a peer-to-peer advertisement distribution mechanism between client, could ease the diffusion of Pervasive Advertising outside the boundaries of a Pervasive Advertising hotspot’s local network: this is particularly useful in case of WiFi routers, which have a quite limited range. The implementation of a system for active context creation; the user’s context can be partially inferred by analyzing his activities, such as visited websites, text messages and so forth.

REFERENCES