Wi-Fi Proximity as a Service

A new approach for distributing hyper-local data

Dmitry Namiot
Lomonosov Moscow State University
Faculty of Computational Mathematics and Cybernetics
Moscow, Russia
dnamiot@gmail.com

Manfred Sneps-Sneppe
Ventspils University College
Ventspils International Radio Astronomy Centre
Ventspils, Latvia
manfreds.sneps@gmail.com

Abstract—**This paper describes a new model for delivering hyper-local data for mobile subscribers. Our model uses Wi-Fi proximity as a service. In our concept, any exiting or even a specially created Wi-Fi hot spot could be used as presence sensor that can trigger access for some user-generated information snippets. In this approach we can discover hyper local data as data snippets that are valid (relevant) for mobile subscribers being at this moment nearby some Wi-Fi access point. And an especially developed mobile application (context-aware browser) can present that information to mobile subscribers. As the possible use-cases, we can mention for example news and deals delivery in malls, news feeds for office centers and campuses, Smart City projects, personal classifieds, etc.**

Keywords-Wi-Fi; proximity; collaborative location; indoor positioning; context-aware browsing.

I. INTRODUCTION

Technically, proximity sensor is the sensor able to detect the presence of nearby objects without any physical contact [1]. In this article, we will describe a model where the exiting or even especially created wireless networks could be used as proximity sensors. The detected proximity will trigger hyper-local news data delivery to mobile subscribers.

Many mobile applications can be characterized as collaborative in the sense that mobile nodes use the wireless network to interact with other mobile nodes that have come together at some common location. Collaborative nodes typically come together in some area, establish associations with other collaborative nodes dynamically, and make use of common services already available in that locality or provided by members of the group. The members of such a group may migrate together, like the visitors in the mall, some group of pedestrian, etc. [2].

Although these applications may use infrastructure networks, they will often use ad hoc networks since they are immediately deployable in arbitrary environments and support communication without the need for a separate infrastructure. This collaborative style of application may be useful in the ubiquitous computing [3]. Context awareness is defined as complementary element to location awareness. Whereas location may serve as a determinant for resident processes, context may be applied more flexibly with mobile computing with any moving entities, especially with bearers of smart communicators. Context awareness originated as a term from ubiquitous computing or as so-called pervasive computing which sought to deal with linking changes in the environment with computer systems, which are otherwise static [4].

Modern applications adopt a context-aware perspective to manage:

a) Communication among users and among systems, or between the system and the user,

b) Situation-awareness, like modeling location and environment aspects (physical situation) or the current user activity (personal situation)

c) Knowledge chunks: determining the set of situation-relevant information, services or behaviors [5].

In our approach, we are dealing with context-aware knowledge chunks. Let us start with the base element – location.

There are many different approaches for getting location info for mobile subscribes. In general, it could be pretty standard nowadays (GPS, cell-id, assisted GPS [6]), but everything is getting more complicated as soon as we need indoor positioning. Due to the signal attenuation caused by construction materials, the Global Positioning System (GPS) loses significant accuracy indoors. Instead of satellites, an indoor positioning system (IPS) relies on nearby anchors (nodes with a known position), which either actively locate tags or provide environmental context for devices to sense. The localized nature of an IPS has resulted in design fragmentation, with systems making use of various optical, radio, or even acoustic technologies [7].

Nowadays, a great number of technologies are being used for indoor localization, such as Wi-Fi, RFID etc. [7]. However, all of them require the utilization of their own API with their own protocols. This can be a big challenge for developing heterogeneous scenarios where different localization systems have to be used for a location service.

One of the most used approaches to indoor location is Wi-Fi based positioning. A standard Wi-Fi based positioning system, such as the one offered by Cisco is completely software-based and utilizes existing Wi-Fi access points installed in a facility and radio cards already present in the user devices. Companies could deploy also Wi-Fi based radio tags that use industry standard components that adhere to the 802.11 standards. This approach allows for the use of commercial off-the-shelf hardware and drivers to produce a standards-based radio tag that can communicate bi-directionally over the 802.11 networks.

Thus, a standard Wi-Fi based positioning system can realize any type of location-aware application that involves PDAs, laptops, bar code scanners, voice-over-IP phones and other 802.11 enabled devices. For embedded solutions, there is no need for the client to include a specialized tag, transmitter, or receiver.

Because of the entire use of standards-based hardware, such as 802.11b, 802.11g, and 802.11a, a standard Wi-Fi based solution rides the installed based and economies of scale of the networks and end user devices that are proliferating today. Without the need for additional hardware, a company can install the system much faster and significantly reduce initial and long-term support costs. A common infrastructure supports both the data network and the positioning system, something companies strive for. The positioning system works wherever there is Wi-Fi coverage.

Wi-Fi location positioning is based on a grid of Wi-Fi hotspots providing, in general, 20–30 meters location accuracy. For more accuracy, there needs to be more access points. There are many articles devoted to Wi-Fi positioning. For example, we can combine a reference point-based approach with a trilateration-based one etc. Several layers of refinement are offered based on the knowledge of the topology and devices deployed. The more data are known, the better adapted to its area the positioning system can be [8].

Lets us mention also one more interesting approach: collaborative location (CL) [9]. The most interesting approach for our future development is Collaborative Location-sensing. Cooperative Location-sensing system (CLS) is an adaptive location-sensing system that enables devices to estimate their position in a self-organizing manner without the need for an extensive infrastructure or training.

Simply saying, hosts cooperate and share positioning information. CLS uses a grid representation that allows an easy incorporation of external information to improve the accuracy of the position estimation [9].

The motivation for CL and CLS is very transparent. In many situations, due to environmental, cost, maintenance, and other obstacles, the deployment of a dense infrastructure for location sensing is not feasible. It is exactly what we wrote about infrastructure-less system. In CLS, hosts estimate their distance from their neighboring peers. This can take place with any distance estimation method available (e.g., using signal strength). They can refine their estimations iteratively as they incorporate new positioning information.

Another interesting aspect for our approach is dynamic location based services. For example, AROUND [10] architecture is proposed as an approach for supporting location-based services in the Internet environment. AROUND provides a service location infrastructure that allows applications to select services that are specifically associated with their current location. The architecture includes a flexible scope model that defines the association between services and location, and a service location infrastructure organized by spatial criteria and optimized for location-based queries.

And at this point we are ready to make the last proposition before switching to our SpotEx model. Of course, the acronym LBS (Location Based Systems) contains the word "location". But do we really need the location for the most of the services? As seems to us the final goal (at least for the majority of services) is to get data related to the location, rather than location itself. Location in the classical form (latitude, longitude) here is just an intermediate result we can use as key in our requests for obtaining data (our final goal). So, why do not request data directly if we can estimate location? We need to explain, what does it mean "estimate" here. It is exactly the proximity. But here are the obvious obstacles. The location services have got a wide support on mobile world for example. We should make proximity as easy to use for developers as location has now become. Then we will unleash a similar explosion of innovative services.

II. RELATED WORK

Lets us overview some existing projects. AllJoyn [11] is a peer-to-peer technology that enables ad hoc, proximity-based, device-to-device communication without the use of an intermediary server. Technically, it is a set of APIs for developers.

Nokia Instant Community is a new, instant way for communities to socially interact when in close proximity, without the need for WLAN infrastructure or Bluetooth and cellular connections [12]. The clever thing about Nokia Instant Community is that it doesn't need the Internet. It means no searching for a Wi-Fi hotspot to get you in touch, and you don not need infrared or bluetooth either. It works completely by using the device's adhoc Wi-Fi. But we should mention here that the whole idea looks very similar to Wi-Fi direct spec.

LocalSocial by Rococosoft [13] offers a proximity platform for sharing social data. LocalSocial combines information about people and things close to a user, with information from one or more of their social networks, and makes that information available to a mobile application in a simple way.

LocalSocial has 2 key components:

1. The LocalSocial Mobile Library, which is deployed on the handset. This gives developers APIs that allow them to:

a) Use proximity to detect devices nearby using Bluetooth, WiFi, and NFC

b) Connect securely to the LocalSocial Service

c) Discover information about the devices nearby - including opt-in social tags and information

e) Send messages to devices nearby, store tags about those devices for later

2. The LocalSocial Cloud Service is a web-hosted service, which mediates between the mobile application and the cloud, and stores and retrieves information about the devices (as tags), tracks analytics, and provides social connections to key social networks.

What are the common points to all projects? It is device based API. At the first hand, they are oriented to the device – to-device proximity. It is probably more convenient to the social networks. In our project, we will target device-to-infrastructure objects proximity. It is more convenient to the information providing. For example, deliver some news data for people in Smart City applications; provide dynamic information systems in campuses and office buildings, etc.

III. OUR APPROACH - SPOTEX

What if we stop our traditional indoor positioning schema on the first stage: detection of Wi-Fi networks? This detection actually already provides some information about the location – just due to local nature of Wi-Fi network. And as the second step we add the ability to describe some rules (if-then operators, or productions) related to the Wi-Fi access points. Our rules will simply use the fact that the particularly Wi-Fi network is detected. Based on this conclusion we will open (read – make them visible) some user-defined messages to mobile terminals. Actually, it is a typical example for the context aware computing. The visibility for user-defined text (content) depends on the network context.

The first time this service SpotEx (Spot Expert [14]) was described by the authors in article published in NGMAST-2011 proceedings [15]. It is a working application for Android platform.

Obviously, our SpotEx model is based on the ideas of Wi-Fi proximity. Any Wi-Fi hot spot works here just as presence sensor. But we are not going to connect mobile users to the detected networks and our suggestion does not touch security issues. It is not about connectivity at all. We need only SSID for networks and any other public information.

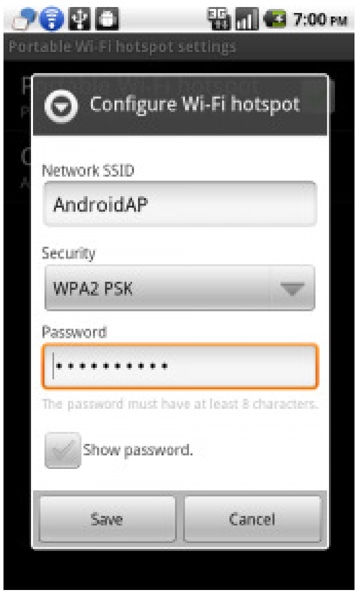
So, our service contains the following components:

- database (store) with productions (rules) associated with Wi-Fi networks. It is web-based data store
- rule editor. Web application (including mobile web) that lets users add (edit) rule-set, associated with some Wi-Fi network
- mobile applications, that can detect Wi-Fi networks, check the current conditions against the database and execute productions

How does it work? We can take any exiting Wi-Fi network (or networks especially created for this service – the most interesting case, see below) and add some rules (messages) to that network. Message here is just some text that should be delivered to the end-user's mobile terminal as soon as the above-mentioned network is getting detected via our mobile application. The word "delivered" here is a synonym for "available for reading/downloading".

The possible use cases, including commercial deployment are obvious. Some shop can deliver deals/discount/coupons right to mobile terminals as soon as the user is near some predefined point of sale. We can describe this feature as "automatic check-in" for example. Rather than directly (manually or via some API) set own presence at some place (e.g., similar to Foursquare, Facebook Places, etc.) and get deals info, with SpotEx mobile subscriber can pickup deals automatically. Campus admin can deliver news and special announces, hyper local news in Smart City projects could be tight (linked) to the public available networks and delivered information via that channel etc.

Especially, we would like to point attention to the most interesting (by our opinion, of course) use case: Wi-Fi hot spot being opened right on the mobile phone. Most of the modern smart phones let you open Wi-Fi hot spots. We can associate our rules to such hot spot (hot spots) and so our messages (data snippets) become linked to the phones. Practically, we are getting dynamic LBS here: phone itself could be moved and so, the available data will be de-facto moved too.


Portable Wi-Fi hotspot settings

Configure Wi-Fi hotspot

Network SSID: AndroidAP

Security: WPA2 PSK

Password: ••••••••••

The password must have at least 8 characters.

Show password.

Figure 1. Wi-Fi hot spot on Android

This use case is probably the most transparent demonstration of SpotEx model. We can open "base" network right on the mobile phone, attach ("stick") rules for the content to that network and it is all do we need for creating a new information channel. There is no infrastructure except the smart phone and we do not need a grid of devices as per CLS models. By the way, it is the main difference from the centralized location frameworks from Goggle, Nokia or Ericsson, too. All the frameworks for Wi-Fi positioning are using static databases.

And note again that this approach does not touch security and connectivity issues. You do not need to connect mobile subscribers to your hot spot. SpotEx is all about using hot spot attributes for triggers that can discover the content. The term Wi-Fi proximity is used sometimes in connection with Wi-Fi marketing and mean on practice just setting a special splash screen for hot spot that can show some advertising/branded messages for users during the connection to that hot-spot. Unlike this, SpotEx threats Wi-Fi hot spots just as sensors.

How our productions data store (base of rules) looks like? Each rule looks like a production (if-then operator). The conditional part includes the following objects:

Wi-Fi network identity,
signal strength (optionally),
time of the day (optionally),
client ID (see below).

In other words, it is a set of operators like:

IF *network_SSID* IS 'mycafe' AND *time is 1pm – 2pm* THEN { present the coupon for lunch }

Because our rules form the standard production rule based system, we can use old and well know algorithm like Rete [16] for the processing. A Rete-based expert system builds a network of nodes, where each node (except the root) corresponds to a pattern occurring in the left-hand-side (the condition part) of a rule. The path from the root node to a leaf node defines a complete rule left-hand-side. Each node has a memory of facts, which satisfy that pattern. This structure presents essentially a generalized tree. As new facts are asserted or modified, they propagate along the network, causing nodes to be annotated when that fact matches that pattern. When a fact or combination of facts causes all of the patterns for a given rule to be satisfied, a leaf node is reached, and the corresponding rule is triggered [17].

The current implementation for mobile client based on Android OS. This application uses *WiFiManager* from Android SDK - the primary API for managing all aspects of Wi-Fi connectivity. This API let us pickup the following information about nearby networks:

SSID - the network name.
BSSID - the address of the access point.
capabilities - describes the authentication, key management, and encryption schemes supported by the access point.
frequency - the frequency in MHz of the channel over which the client is communicating with the access point.
level - the detected signal level in dBm.

So, actually all the above-mentioned elements could be used in our productions. And now we can prepare rules like this:

IF *network_SSID* IS 'mycafe' AND level > -60db AND time is 1pm – 2pm AND network_SSID 'myStore' is not visible THEN {present the deals for dinner}

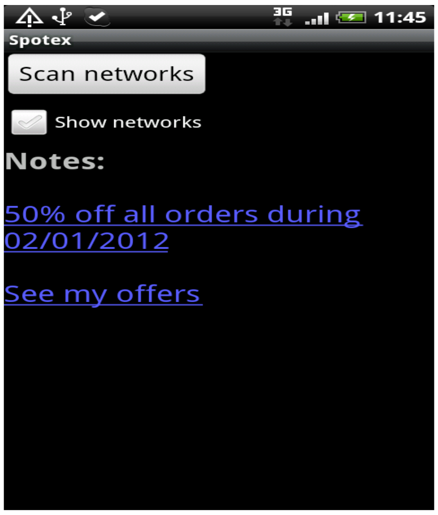
Block {*present the deals for dinner*} is some data (information) snippet presented in the rule. Each snippet has got a title (text) and some HTML content (it could be simply a link to external site for example). Snippets are presenting coupons/discounts info for malls, news data for campuses, etc.

Technically, any snipped could be presented as a link to some external web site/mobile portal or as a mobile web page created automatically by the rule editor included into SpotEx. Rule editor works in both desktop and mobile web. So, once again just having ordinary smart phone is enough for creating (opening) information channel for delivering hyper-local news data.

In case of presenting our data as links to some existing mobile sites (portals), SpotEx works as some universal discovery tool. De facto it lets mobile subscribers to be aware about context-relevant web resources. And owners for the mobile resources can describe own sites via rules rather then present individual QR-codes or NFC-tags for example.

In case of describing some content right in the SpotEx the whole system works in this part as content management system (CMS). SpotEx rule editor creates mobile web page for the each provided data snippet and hosts that page on the own server. It means by the way, that for presenting our data we can use any resources that could be presented on HTML pages. In particularly, any multimedia content is also supported.

SpotEx mobile application, being executed, creates dynamic HTML page from titles (according to rules that are relevant in the given context) and presents that mobile web page to the user. It works just as a classical rule based expert system: matches exiting rules against the exiting context and makes the conclusions. Existing content here is a description for "Wi-Fi environment": list of hot spots with attributes. And conclusion here is a list of titles that can be presented as a dynamically created mobile web page. On that page each discovered title could be presented as a hyperlink that points to the appropriate data snipped. Any click on the interested title opens the snippet (shows or discovers data to mobile user).



Notes:

50% off all orders during 02/01/2012

See my offers

Figure 2. SpotEx client

So, for the mobile users the whole process looks like browsing, where their browser becomes aware about hyper-local content. It is a typical example of context-aware retrieval. Context-aware retrieval can be described as an extension of classical information retrieval that incorporates the contextual information into the retrieval process, with the aim of delivering information relevant to the users within their current context [18].

The context-aware retrieval model includes the following elements:

- a collection of discrete documents;
- a set of user's retrieval needs, captured in a query;
- a retrieval task, to deliver the documents that best match the current query, rated on the basis of a relevance measure;
- the user's context, used both in the query formulation and associated with the documents that are candidates for retrieval.

It is obviously, that all the above-mentioned tasks are components of SpotEx.

As per other functionality of our context-aware browser, we can highlight the following notes. At the first hand, we can note that it is the “pull model”, versus the “push model” that proposed by Bluetooth marketing for example. It could be more convenient (more safe) for the users – there are no automatically downloaded files/messages, etc. But in the same time nothing prevents us from updating that dynamic web page automatically (e.g., by the timer) and simulating “pull model” in the user-safety mode.

At the second hand, we can note that because it is browsing, the whole process is anonymous. Indeed, there is no sign-in in the SpotEx. Of course, any data snippet may lead to some business web site/portal, where that site may ask about login etc., but the SpotEx itself is anonymous. Unlike social networks like Foursquare you do not need to disclose your identity just for looking mall’s deals for example.

But, in the same time, we still can collect some meaningful statistics in SpotEx. Because the model requires Wi-Fi to be switched on, we have automatically unique ID for the each client. It is MAC-address. And it is actually global UUID. So, where we have not login info for our clients, we still can distinguish them. It let us detect for example, the same person, who did that already twice during the last week, opens that the particular data snipped.

Because mobile users in SpotEx model actually work with web pages, we can use pretty standard methods for web server log analysis for discovering user’s activities.

A statistical analysis of the server log may be used to examine traffic patterns by time of day, day of week etc. So, we can detect frequent visitors, usage patterns etc. And even more – we can use that information in our rules. E.g., some mall may offer special things for frequent visitors, etc. Data from real time analytics for our info snippets could be used in conditional parts of our rules.

The next stage of development targets the simplicity of preparing data for SpotEx model. What if instead of the separate database with rules (as it is described above) we add the ability to provide a special markup for existing HTML files?

So, rather than writing separate if-then rules we can describe our rules right in HTML code. Technically, we can add for example HTML div blocks with attributes that describe our rules (their conditions). Now, using some JavaScript code we can loop over such div blocks and simply hide non-relevant from them. For doing that we need to make sure that our JavaScript code is aware about the current context. We can achieve that via a special light implementation of local web server. This web server, being hosted right on the mobile phone (on the Android in our case) responds actually only to one type of requests. It returns the current context (Wi-Fi networks) in JSON (JSONP) format.

Why do we need a web server? It lets us stay in the web development domain only. There is a simple and clear instruction for web masters:

- add SpotEx script to your page

<script type = ”text/javascript” src = http://localhost:8080/spotex.js> </script>

- describe your info snippets as div blocks:

```
<div rel=”spotex” net=”WiFi_SSID” levelMin=”” levelMax=””>
        Your HTML code
</div>
```

Our “old” rules could be presented via collection of attributes for HTML tags.

In this case, JavaScript code loaded from local server will be able to proceed all the div blocks related to SpotEx, and set visibility attributes depending on the context.

Such simple trick let us convert any existing HTML page into “Wi-Fi context aware” version. Note that if our script is not available, the page will work as a “standard” HTML page.

There is also a “side” effect for SpotEx application – WiFiChat service [19]. This mobile application uses the principles described in this article and offers communication tools (web chat and discussions groups) for mobile users nearby the same Wi-Fi access point. Think about it as “SpotEx with predefined content”. The typical use case – we have Wi-Fi network in the train and this application automatically provides the discussions forum for the passengers. Or, keeping in mind that the “base” Wi-Fi network for this service could be opened right on the phone, this application can present personal forum (classified for example) as well as web chat for phone owner. This Android application is actually a wrapper for web mashup that combines HTML5 web chat engine and cloud based forums from Disqus:

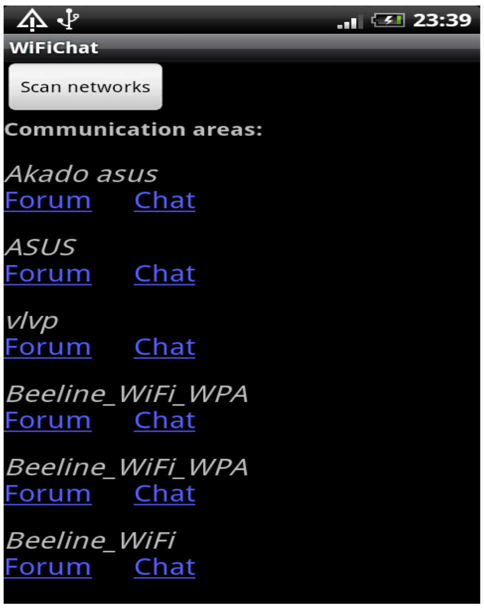


Figure 3. Wi-Fi chat application

It is the typical tool for the communications on the go. It is used right now in the public transport where some WiFi service is present.

IV. THE FUTURE DEVELOPMENT

Here, we see several almost obvious steps. At the first hand it is open API. In the current implementation SpotEx front-end actually obtains data in JSON (JSONP) format from our server-side database.

As soon as API is going live, the next step is almost mandatory. It should be the stuff that will simplify the development. The good candidates here are web intents [20] Web Intents is a framework for client-side service discovery and inter-application communication. Services register their intention to be able to handle an action on the user's behalf. Applications request to start an action of a certain verb (for example share, edit, view, pick, etc.) and the system will find the appropriate services for the user to use based on the user's preference. It is the basic.

Intents play the very important role in Android Architecture. Three of the four basic OS component types - activities, services, and broadcast receivers - are activated by an asynchronous message called as intent.

Intents bind individual components to each other at runtime (you can think of them as the messengers that request an action from other components), whether the component belongs to your application or another.

Created intent defines a message to activate either a specific component or a specific type of component - an intent can be either explicit or implicit, respectively.

For activities and services, an intent defines the action to perform (for example, to "view" or "send" something) and may specify the URI of the data to act on (among other things that the component being started might need to know). For example, our intent might convey a request for an activity to show an image or to open a web page. In some cases, you can start an activity to receive a result, in which case, the activity also returns the result in an Intent (for example, we can issue an intent to let the user pick a list of nearby images and have it returned to us - the return intent includes data in some format)

Going to our context aware browsing it means that our mobile devices will be able to present local data without low-level programming.

Web Intents puts the user in control of service integrations and makes the developers life simple.

Here, is the modified example for web intents integration for the hypothetical web intents example:

1. Register some intent upon loading our HTML document

```
    document.addEventListener("DOMContentLoaded",
function() {
        var regBtn = document.getElementById("register");
        regBtn.addEventListener("click", function() {
window.navigator.register(http://webintents.org/spotex,
undefined);
        }, false);
```

2. Start intent's activity and pass it extra data (context info)

```
        var startButton =
document.getElementById("startActivity");
        startButton.addEventListener("click", function() {
          var intent = new Intent();
          intent.action = "http://webintents.org/spotex";
          intent.putExtra("WiFi_List", List_Of_Networks);
            window.navigator.startActivity(intent);
        }, false);
```

3. Get local info snippets (note – in JSON rather than XML) and display them in our application

```
        window.navigator.onActivity = function(data) {
          var output = document.getElementById("output");
          output.textContent = JSON.stringify(data);
        };
      }, false);
```

Obviously, that it is much shorter than the long sequence of individual calls as per any Open API.

The key point here is *onActivity* callback that returns JSON formatted data. Additionally, web intents based approach is asynchronous by its nature, so, we don need to organize asynchronous calls by our own.

Also, we are planning to add Bluetooth measurements too. But, by our vision, we should avoid the typical Bluetooth usage cases and do not use push proxy as per classical Bluetooth marketing. We think that the end users do not welcome at least push approach and it is the source of problems with Bluetooth proximity. Vice versa, in SpotEx Bluetooth nodes will be used the same manner we are using Wi-Fi access points – as presence triggers. In other words, we will add the ability to describe rules for Bluetooth nodes too.

SpotEx approach could be extended also towards accumulating some ideas from the collaborative locations. We can add trilateration terms (conditions) to our rules, but present them in terms of fuzzy logic (close than, relatively close, etc.) It helps us incorporate grid data in case of many devices without any infrastructure preparation.

The next area we are going to pay attention to is Wi-Fi Direct specification. Wi-Fi Direct devices can connect directly to one another without access to a traditional network, so mobile phones, cameras, printers, PCs, and gaming devices can connect to each other directly to transfer content and share applications anytime and anywhere. Devices can make a one-to-one connection, or a group of several devices can connect simultaneously. They can connect for a single exchange, or they can retain the memory of the connection and link together each time they are in proximity [21].

As per Wi-Fi Direct spec, a single Wi-Fi Direct device could be in charge of the Group, including controlling which devices are allowed to join and when the Group is started. All Wi-Fi Direct devices must be capable of being in charge of a Group, and must be able to negotiate which device adopts this role when forming a Group with another Wi-Fi Direct device. The device that forms the Group will provide the above described dynamically assembled web page with discovered services. It is how SpotEx could be extended to Wi-Fi Direct.

V. CONCLUSION

This paper described a new context-aware browsing model for mobile users based on the ideas of Wi-Fi proximity. Developed service can use any existing as well as the especially created (described) Wi-Fi network as presence trigger for discovering user-defined content right to mobile subscribers.

Proposed approach is completely software based and does not require additional hardware investments. For using SpotEx you need nothing except the smart phone. Also this approach supports ad-hoc solutions and despite existing Wi-Fi based indoor-location techniques does not require the upfront space preparations.

This service could be used for delivering commercial information (deals, discounts, coupons) in malls, hyper-local news data, data discovery in Smart City projects, personal news sites, etc.

REFERENCES

[1] C. Seeger, A. Buchmann, and K. Van Laerhoven: Wireless sensor networks in the wild: three practical issues after a middleware deployment Proceedings of the 6th International Workshop on Middleware Tools, Services and Run-time Support for Networked Embedded Systems ACM New York, NY, USA ©2011 ISBN: 978-1-4503-1069-7

[2] R. Meier, V. Cahill, A. Nedos, and S. Clarke: Proximity-Based Service Discovery in Mobile Ad Hoc Networks Distributed Applications and Interoperable Systems Lecture Notes in Computer Science, 2005, Volume 3543/2005, pp. 1147-1149, DOI: 10.1007/11498094_11

[3] M. Weiser: "Ubiquitous Computing," IEEE Hot Topics, vol. 26, pp. 71-72, 1993.

[4] S. Poslad Ubiquitous Computing: Smart Devices, Environments and Interactions DOI: 10.1002/9780470779446.fmatter Copyright © 2009 John Wiley & Sons, Ltd

[5] C. Bolchini1, G. Orsi, E. Quintarelli, F. A. Schreiber, and Letizia Tanca: Context modeling and context awareness: steps forward in the context-addict project. IEEE Data Eng. Bull., 34(2): pp. 47–54

[6] R. Ahas, S. Silm, O. Järv, E. Saluveer, and Margus Tiru: Using Mobile Positioning Data to Model Locations Meaningful to Users of Mobile Phones; Journal of Urban Technology Volume 17, Issue 1, 2010 pp. 3-27

[7] Comparison of Wireless Indoor Positioning Technologies http://www.productivet.com/docs-2/Wireless_Comparison.pdf Retrieved: Feb, 2012

[8] F. Lassabe, P. Canalda, P. Chatonnay, and F. Spies "Indoor Wi-Fi positioning: techniques and systems" Annals of Telecommunications Volume 64, Numbers 9-10, pp. 651-664, DOI= 10.1007/s12243-009-0122-1

[9] C. Fretzagias and M. Papadopouli, Cooperative Location-Sensing for Wireless Networks, Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04), p.121, March 14-17, 2004

[10] R. José, A. Moreira, H. Rodrigues, and N. Davies: The AROUND Architecture for Dynamic Location-Based Services in Mobile Networks and Applications Volume 8, Number 4, pp. 377-387, DOI: 10.1023/A:1024531629631

[11] AllJoyn https://www.alljoyn.org/ Retrieved: Mar, 2012

[12] Nokia Instant Community http://research.nokia.com/news/9391 Retrived: Feb, 2012

[13] LocalSocial http://www.rococosoft.com/products/localsocial Retrieved: Mar, 2012

[14] SpotEx http://spotex.linkstore.ru Retrived: Feb, 2012

[15] D. Namiot and M. Schneps-Schneppe: About location-aware mobile messages International Conference and Exhibition on. Next Generation Mobile Applications, Services and Technologies (NGMAST), 2011 5th International Conference on Issue Date : 14-16 Sept. 2011 pp.: 48 – 53 ISSN : 2161-2889 Print ISBN: 978-1-4577-1080-3 DOI : 10.1109/NGMAST.2011.19

[16] E. Friedman-Hill Jess in Action: Java Rule-based Systems, Manning Publications Company, June 2003, ISBN 193011089

[17] C. L. Forgy "RETE: A fast algorithm for the many pattern/many object pattern match problem", Artificial Intelligence 19(1): pp. 17-37, September 1982

[18] P. J. Brown and G. J. F. Jones.:Context-aware retrieval: Exploring a new environment for information retrieval and information filtering. Personal Ubiquitous Computing, 5(4): pp. 253–263, 2001.

[19] Wi-Fi chat http://wifichat.linkstore.ru Retrieved: Mar, 2012

[20] Web Intents: http://webintents.org/ Retrived: Feb, 2012

[21] Wi-Fi CERTIFIED Wi-Fi Direct http://www.cnetworksolution.com/uploads/wp_Wi-Fi_Direct_20101025_Industry.pdf Retrieved: Feb, 2012