Local Area Messaging for Smartphones

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

Abstract — **This paper describes a new model for local messaging. It is mobile mashup combines passive monitoring for smart phones and cloud based messaging for mobile operational systems. Passive monitoring can determine the location of mobile subscribers (mobile phones, actually) without the active participation of the users. Mobile users do not need to mark own location on social networks (check-in). They do not need to run on their phones the location track applications too. Cloud Messaging lets data providers (e.g., local businesses) directly deliver their information to mobile users nearby a selected point. This paper describes how to combine the monitoring and notifications.**

Keywords- Wi-Fi;monitoring;proximity;location;messaging

I. INTRODUCTION

In the first paper that introduced the term 'context-aware' (Schilit and Theimer [1]), authors describe context as location, identities of nearby people and objects, and changes to those objects. There are several definitions from other authors, but most of them define context awareness as a complementary element to location awareness. Location serves as a determinant for the main processes and context adds more flexibility with mobile computing and smart communicators [2].

There are models of applications, where the concept of location can be replaced by that of proximity. At the first hand, this applies to use cases where the detection for exact location is difficult, even impossible or not economically viable [2]. Very often, this change is related to privacy. For example, a privacy-aware proximity detection service determines if two mobile users are close to each other without requiring them to disclose their exact locations [3]. As per developed algorithms for privacy-aware proximity detection methods we can mention papers [4] and [5], for example. They allow two online users to determine if they are close to each other without requiring them to disclose their exact locations to a service provider or other friends. Usually, the main goal for such systems is to generate proximity messages when friends approach each other closer than some predefined distance threshold. Technically, this threshold can be defined individually for each user (for group of users).

Of course, the term "distance" here depends on the metric used for the measurements. The classical example includes shortest path metric and two users on the different sides (banks) of the river. It is anti-pattern. The distance between users could be within the given threshold, but such "proximity" is useless.

Metric measurements for privacy can be replaced with some approximation by wireless proximity (network proximity). For this paper network proximity definition is very intuitive. It is a measure of how mobile nodes are close or far away from the elements of network infrastructure. For example, let us assume that we have one Wi-Fi access point (AP). Network proximity is this case is a measure how our mobile phones are close of far away from this AP. Technically, it is based on the observation that measurements of the wireless channels between a radio frequency source and devices in proximity are highly correlated [6].

There are several systems that can use network proximity as a base for mobile services. At the first hand we can mention here our own system SpotEx (Spot Expert) [7]. According to this model, any existing or even especially created Wi-Fi hot spot could be used as presence sensor that can trigger access for some user-generated information snippets.

The typical application in this area uses collected database of so called Wi-Fi "fingerprints", including MAC addresses and the received signal strengths (RSSI) of nearby access points. This database could be used for Wi-Fi based positioning as well as for discovering the user's behavioral patterns [8]. A classical approach to Wi-Fi fingerprinting [9] involves RSSI (signal strength). The basic principles are transparent. At a given point, a mobile application may hear ("see") different access points with certain signal strengths. This set of access points and their associated signal strengths represents a label ("fingerprint") that is unique to that position. The metric that could be used for comparing various fingerprints is k-nearest-neighbors in signal space. It means that two compared fingerprints should have the same set of visible access points. As the next step they could be compared by calculating the Euclidian distance for signal strengths. At the same time, the need for the collection of fingerprints is the biggest problem for this approach.

II. WI-FI DEVICES AND MONITORING

Problems associated with the collection of fingerprints, are fairly obvious. It is the price of the calibration process, the need for rework after the changes in the network and, most importantly, lack of support for dynamic networks. For example, most of the modern smart phones let users open Wi-Fi access point right on the phone.

We cannot create stable base of fingerprints for dynamic access points. Data linked to such dynamic access points become linked to the phones. It is, in fact, a typical dynamic location based system (LBS). The available services move with the moved phone [2].

In our new service we've decided to use another fingerprints-less model: sniffing for beacon frames. Typically, during the calibration phase of Wi-Fi fingerprinting, beacon frames are collected from nearby access points at each survey position. As the next step, the MAC address, RSSI (signal strength), and timestamp could be are extracted from each beacon. In our system we use the reverse schema. We would like to analyze beacons transmitted by Wi-Fi devices, rather than beacons collected by them.

Collecting traces of Wi-Fi beacons is the well-know approach for getting the locations of Wi-Fi access points (AP). Beacon frames are used to announce the presence of a Wi-Fi network. As a result, an 802.11 client receives the beacons sent from all nearby access points. Client receives beacons even when it is not connected to any network. In fact, even when a client is connected to a specific AP, it periodically scans all the channels to receive beacons from other nearby APs. It lets clients keep track of networks in its vicinity. But in the same time Wi-Fi client periodically broadcasts an 802.11 probe request frame. Client expects to get back an appropriate probe request response from Wi-Fi access point. As Wi-fi spec, a station (client) sends a probe request frame when it needs to obtain information from another station. For example, a radio network interface card would send a probe request to determine which access points are within range.

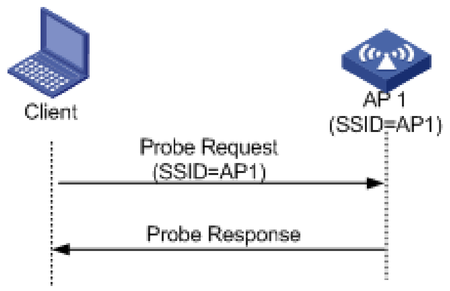

Figure 1. Probe request/response

A Probe Request frame contains two fields: the SSID and the rates supported by the mobile station. Stations that receive Probe Requests use the information to determine whether the mobile station can join the network. To make a happy union, the mobile station must support all the data rates required by the network and must want to join the network identified by the SSID. This may be set to the SSID of a specific network or set to join any compatible network. Drivers that allow cards to join any network use the broadcast SSID in Probe Requests [10].

Technically, probe request frame contains the following information:

- source address (MAC-address)
- SSID
- supported rates
- additional request information
- extended support rates
- vendor specific information

Our access point can analyze received probe request. Obviously, that any new request (any new MAC-address) corresponds to a new wireless customer nearby. Note, that Bluetooth devices could be monitored by the same principles.

Wi-Fi based device detection uses only a part from the above mentioned probe request. It is a device-unique address (MAC address). This unique information lets us re-identify devices (mobile phones) across our monitors.

We should note also, that passive Wi-Fi detection is not 100% reliable. Mobile phones (mobile OS, actually) can actually transmit probe requests at their discretion. Our own experiments with commercially available Wi-Fi probe scanners confirm data from [12]. Monitor detects in average about 70% of passing smartphones.

There are commercial of-the-shelf components that can provide passive Wi-Fi monitoring. For example, it is Meshlium Xtreme [13]. With passive monitoring Wi-Fi devices can be detected without the need of being connected to a specific access point, enabling the detection of any smartphone, laptop or hands-free device which comes into the coverage area.

Here is an example of information saved by Wi-Fi scanner:

DB ID:	53483
Timestamp:	2012-04-24 07:56:25
MAC:	C4:2C:03:96:0E:4A
AP:	Cafe
RSSI:	69
Vendor:	Apple

Note, that key moment here is MAC-address for mobile device. We will use it for re-identification only. It means, by the way, that for keeping the privacy we do not need to save in our database an original address. It is enough just to keep some hash-code for this address.

The typical tasks this approach could be applied for are:

- get a number of people passing daily in a street
- detect an average time of the stance of the people in a street or in a building
- differentiate between residents (daily matches) and visitants (sporadic matches)
- detect the walking routes of people in shopping malls and average time in each area

In general, it could be described as a real analytics for the real places. It is what makes Google Analytics for web sites, but applied for the real places and real visitors.

In this paper we propose a new model (use case) for passive monitoring. It is messaging for the real places and real visitors.

III. CLOUD MESSAGING

Google Cloud Messaging for Android (GCM) is a service that allows you to send data from your server to your users' Android-powered device. This could be a lightweight message telling your app there is new data to be fetched from the server (for instance, a movie uploaded by a friend), or it could be a message containing up to 4kb of payload data (so apps like instant messaging can consume the message directly).

The GCM service handles all aspects of queuing of messages and delivery to the target Android application running on the target device. GCM is completely free no matter how big your messaging needs are, and there are no quotas [14].

Apple Push Notification Service (APN) is a robust and highly efficient service for propagating information to devices such as iPhone, iPad, and iPod touch devices. Each device establishes an accredited and encrypted IP connection with the service and receives notifications over this persistent connection. If a notification for an application arrives when that application is not running, the device alerts the user that the application has data waiting for it [15].

Architectures of these push notification services have common features. At the first hand, application servers send a notification message with an intended receiver (or the target mobile device) to one of the cloud-based messaging servers. Messaging servers pushes the message to the target mobile device. The push notification service eliminates the needs of application servers to keep track of the state of a mobile device (i.e., online or offline). Furthermore, mobile devices do not need to periodically probe (poll) the application servers for messages. It reduces the workloads of the application servers and seriously simplifies the mobile application development.

We describe below Google Cloud Messaging Service as a main system used in our development. In the same time principles are the same for all the above-mentioned services.

Here are the primary characteristics of GCM as per Google's manual:

It allows 3rd-party application servers to send messages to their Android applications.

An Android application on an Android device doesn't need to be running to receive messages. The system will wake up the Android application via Intent broadcast when the message arrives, as long as the application is set up with the proper broadcast receiver and permissions.

The first time the Android application needs to use the messaging service, it fires off a registration Intent to a GCM server. This registration Intent) includes the sender ID, and the Android application ID.

If the registration is successful, the GCM server broadcasts an intent which gives the Android application a registration ID.

The Android application should store this ID for later use. To complete the registration, the Android application sends the registration ID to the application server. The application server typically stores the registration ID in a database.

The registration ID lasts until the Android application explicitly un-registers itself, or until Google refreshes the registration ID for your Android application.

For an application server to send a message to an Android application, the following things must be in place:

- The Android application has a registration ID that allows it to receive messages for a particular device.
- The 3rd-party application server has stored the registration ID.
- An API key. This is something that the developer must have already set up on the application server for the Android application. Now it will get used to send messages to the device.

Here is the sequence of events that occurs when the application server sends a message:

- The application server sends a message to GCM servers.
- Google en-queues and stores the message in case the device is offline.
- When the device is online, Google sends the message to the device.

On the device, the system broadcasts the message to the specified Android application via Intent broadcast with proper permissions, so that only the targeted Android application gets the message. This wakes the Android application up. The Android application does not need to be running beforehand to receive the message.

Your Android application cans un-register GCM if it no longer wants to receive messages [14].

IV. SPOTIQUE SERVICE

Based on the above-mentioned description, we can note that receiving the messages requires the registration phase. Android application (read – mobile phone with installed application) should inform GCM about the possibility to obtain messages. Usually, this contract is presented in the form of some ID (registration ID). IDs are stored in database. So, our service can select all the stored IDs and distribute some custom message (messages) to applications.

What if we include into process of registration MAC-address too? This decision lets us simply compare subscription info with the locally detected (presented) mobile subscribers.

The whole schema is actually very transparent.

1) Mobile user informs CGM about his intention to receive messages.

2) Messages are divided by topics. Each topic actually corresponds to some location with passive Wi-Fi monitoring.

3) Our sender saves registration ID, topic and MAC-address in central database.

4) Wi-Fi monitoring detects the presence for mobile phones.

5) Our daemon scans detection log, extracts MAC-addresses and compares them with subscription database

6) As soon as we discover that subscriber is detected (he is somewhere nearby) we can use CGM for delivering some custom messages

Note, that MAC-address in this schema is used for the re-identification only. So, for keeping the privacy, we can replace it with some hash-code (for both processes: monitoring and subscription).

The typical use cases are proximity marketing and news delivery in Smart City projects for example.

The push notification services of other platforms are similar to GCM in the architectural design. When an application launches in a mobile device, it needs to register to the push service to get a unique ID. This ID may have different names in different platforms, e.g., device token in iOS and push URI in Windows, and then sends it to the application server. When the application server wants to send a push notification to an application, it sends the ID together with the payload to a push server. Push server forwards the payload to the application [16].

What are the advantages for this approach? At the first hand, it is so-called passive monitoring. We do not need to develop some special applications for mobile subscribers. We do not need to ask for any special actions from mobile subscribers, like running some application, checking-in in social networks etc. The messaging will target only subscribers physically presented in the covered area. The process for subscription and un-subscription is very straightforward. The “check-in” process (passive discovering) is secure. It does not keep records in social networks like ordinary check-ins in Foursquare, Facebook, etc. It does not require user’s identification too.

What are the disadvantages? At the first hand, the passive monitoring (as we wrote above) is not 100% reliable. Push messaging delivery requires internet connectivity. But in the same time, installing active Wi-Fi access point on-site, mobile users can connect to, will improve the discovery process.

The future development will introduce customized messaging servers for passive Wi-Fi monitoring.

V. Conclusion

This article presents a new mashup based on passive Wi-Fi monitoring for mobile devices and cloud based notifications. Passive monitoring uses probe requests from Wi-Fi specifications for detecting nearby clients. Notification module uses cloud messaging (push notifications) from mobile operational systems. This approach does not require from mobile subscribers use special applications for setting own location or publish location info in the social network. Practical use cases for this application are proximity marketing and Smart City projects. The proposed approach automatically guaranties that custom messages will target online subscribers in the nearby area only.

References

[1] G. Schilit and B. Theimer Disseminating Active Map Information to Mobile Hosts. IEEE Network, 8(5) (1994) pp. 22-32

[2] D. Namiot and M. Sneps-Sneppe “Context-aware data discovery” Intelligence in Next Generation Networks (ICIN), 2012 16th International Conference on, 2012 pp. 134 – 141, DOI: 10.1109/ICIN.2012.6376016

[3] L.Šikšnys, J. Thomsen, S. Šaltenis, and M. Yiu Private and Flexible Proximity Detection in Mobile Social Networks, Mobile Data Management (MDM), 2010 Eleventh International Conference on, 2010, pp. 75 - 84

[4] S. Mascetti, C. Bettini, D. Freni, X. S. Wang, and S. Jajodia, “Privacy-aware proximity based services,” in MDM, 2009, pp. 31–40.

[5] P. Ruppel, G. Treu, A. Küpper, and C. Linnhoff-Popien “Anonymous User Tracking for Location-Based Community Services,” in LoCA, 2006, pp. 116–133.

[6] S.Mathur, R.Miller, A.Varshavsky, W. Trappe, and N.Mandayam “ProxiMate: Proximity-based Secure Pairing using Ambient Wireless Signals”, MobiSys '11 Proceedings of the 9th international conference on Mobile systems, applications, and services, 2011, pp. 211-224, DOI: 10.1145/1999995.2000016

[7] 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 14-16 Sept. 2011 pp. 48-53 DOI: 10.1109/NGMAST.2011.19

[8] J. Rekimoto, T. Miyaki, and T. Ishizawa. LifeTag: WiFi-based Continuous Location Logging for Life Pattern Analysis. in LOCA. 2007

[9] Y. Chen, Y. Chawathe, A. LaMarca, and J. Krumm. Accuracy characterization for metropolitan-scale Wi-Fi localization. In ACM MobiSys, 2005.

[10] M.Gast 802.11 Wireless Networks: The Definitive Guide O'Reilly Media, Inc., 2005, 654 p.

[11] Wi-Fi probe request http://wiresharklab.blogspot.ru/ Retrieved: Jan, 2013

[12] A. Musa and J.Eriksson Tracking Unmodified Smartphones Using Wi-Fi Monitors, SenSys’12, November 6–9, 2012, Toronto

[13] Lubelium routers http://www.libelium.com Retrieved: Jan, 2013

[14] Google Cloud Messaging http://developer.android.com/google/gcm/index.html Retrieved: Jan, 2013

[15] Apple Push Notification Service http://developer.apple.com/library/mac/#documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/ApplePushService/ApplePushService.html Retrievd: Jav, 2012

[16] S. Zhao, P.Lee, J.Lui, X.Guan, X.Ma, and J.Tao Cloud-Based Push-Styled Mobile Botnets: A Case Study of Exploiting the Cloud to Device Messaging Service ACSAC ’12 Dec.3-7, 2012, Orlando, Florida USA