On Content Models for Proximity Services

Dmitry Namiot
Lomonosov Moscow State University
Moscow, Russia
dnamiot@gmail.com

Manfred Sneps-Sneppe
Ventspils University College
Ventspils, Latvia
manfreds.sneps@gmail.com

***Abstract*—First time introduced in Release 12 of the 3GPP specifications, Proximity Services (ProSe in the above-mentioned specification) is a Device-to-Device (D2D) technology that allows two devices to detect each other and to communicate directly without traversing the Base Station or core network. In other words, it is a technology that is oriented (ultimately) on the direct connection of two devices. In this article, we are promoting the idea that proximity services are more than just support for a direct connection (in fact, search for candidates for a direct connection). The paper discusses content models (that is, information dissemination models) based on proximity data. In this case, a direct connection is simply one of the possible options for disseminating information.**

I. Introduction

The 3GPP specification in Release 12 introduces so-called ProSe (Proximity Services). As per 3GPP, ProSe is a D2D (Device-to-Device) technology that allows two devices to detect each other and to communicate directly. It is based on the idea of using several new functional elements for LTE standards and a special air interface for direct connectivity between devices (so-called sidelink) [1].

ProSe could be compared to existing D2D and proximity networking technologies. ProSe offers better scalability, manageability, energy-efficiency, privacy, and security Basic functions for ProSe are D2D discovery and D2D communication. D2D discovery is defined as the continuous process that identifies another device (devices) in proximity [2].

In other words, it is a technology that is oriented (ultimately) on the direct connection of two devices. In this article, we are promoting the idea that proximity services are more than just support for a direct connection (in fact, search for candidates for a direct connection). The paper discusses content models (that is, information dissemination models) based on proximity data. In this case, a direct connection is simply one of the possible options for the dissemination of information.

Proximity, in this case, refers to the distance metric. In fact, this is some distance that we consider (in each concrete situation) to be close. Accordingly, proximity services are services that are provided to users (consumers) that are considered to be located close to the place where the service (service) is provided. In other words, users are located in a certain geographic area, which we consider to be close under given conditions (for this type of services). As a reference point (close to what?), as a rule, the current location of the mobile device is used. For example, for a mobile phone, this will, in fact, be the current location of the person (owner). The difference from the "classic" geo-information service is that there is no geo-computation. In one way or another, we estimate the distance (proximity) without resorting to a query of geo-coordinates and geo-calculations.

The point of using geo-coordinates is that with the advent of global positioning systems and support for working with them in mobile devices, their retrieval (inquiry) is idle time and standardized in mobile operating systems. This extremely simplified all the work and, strictly speaking, became the main reason for the flourishing of geo-information systems. What is the meaning of the fact that we refuse to geo-calculations? There are several reasons for this.

- request for coordinates from global positioning systems is a very energy-intensive process. This is one of the reasons for the emergence of combined methods (such as assisted GPS or inertial systems);
- global positioning technology exists in military and commercial versions. The accuracy of positioning (and proximity) in the available version may not be the greatest. Or, more practically, by other methods, it is possible to achieve greater accuracy in determining the relative position;
- indoor positioning. Access to global positioning may be difficult (or even impossible);
- the last and, perhaps most importantly, moving objects. If we need to bind some actions (data) to proximity to an object that moves itself, then it is obvious that we cannot somehow bind to specific values of longitude and latitude. In this case, we should talk about the proximity to the current location of the reference object, which itself is changing. Note that in this category, for example, most mobile services should fall. If we consider the proximity of two mobile devices, then obviously both can move.

In reality, only a relatively small class of geographic information systems requires geographical coordinates. In most cases, information about proximity is actually requested (nearby ATM, nearby cafe, etc.). In reality, in most services, geographical coordinates are used to organize the storage and retrieval of data. In other words, in general, the data warehouse has the following model:

$$\{latitude, longitude\} \rightarrow \overline{V} \qquad (1)$$

It is a mapping, where each pair of geo-coordinates is assigned a certain vector (data set). In modern implementations - some kind of JSON object that will be

interpreted. A pair of geo-coordinates can be replaced, of course, with a geo-hash. This is the basis for organizing data and computing on such a model is the basis for the provision of services.

In the case of measuring proximity, instead of a pair of geo-coordinates, we will have a certain set of metrics, which will be used to calculate proximity. In other words, this is practically the same as for geographic information systems, but instead of a pair of geo-coordinates there will be a different set of metrics (for example, addresses of wireless nodes and signal strength - RSSI)

$$\vec{M} \to \overline{V} \tag{2}$$

We abandoned the geo-coordinates, but, naturally, we should replace them with some measurable values (or, by analogy with geo-hash, with some derivative code). Naturally, the calculation (confirmation of the fact) of proximity should be based on some measurements. At the same time, measurements need to be understood in an expanded sense - we can both directly receive (measure) some values, and simply register the presence of some kind of signal (phenomena, etc.). Proximity is defined by nearness in place (location), time, and occurrence or relation. So, the proximity to something (to someone) is really based on different factors rather than a discrete location.

The rest of the work is structured as follows. In section 2, we consider methods for measuring the proximity of mobile devices. In Section 3, we look at the workings of 3GPP. In section 4, we consider the possible services. In Section 5, we give an example implementation.

II. On proximity measurements

First, it is necessary to note the general requirements for such measurements:

- required methods (measurements) should be available for the majority of devices
- standardization of software interfaces is required for access to measurements
- methods for obtaining measurements should not be inconvenient (disruptive) for users. For example, there should be a possibility of background measurement without interrupting the work of other applications.

The first thing we can use here is a proximity sensor on mobile devices. Technically, a proximity sensor is a sensor able to detect the presence of nearby objects without any physical contact. There are many types of proximity sensors: capacitive sensors, sensors based on Doppler effect, inductive, magnetic, including magnetic proximity fuse, optical, sonar, ultrasonic, Hall effect, etc.

It is important to note, that the proximity sensor often has to reflect the radiation signal and the proximity sensor's target is often sensed. It means, that different targets demand different proximity sensors. For example, an inductive proximity sensor works with metallic targets, a capacitive sensor might be suitable for a plastic target, etc. It has been shown that a sensor can be used for a sensor.

Proximity sensors are commonly used on mobile devices [4]. When the target is within nominal range, the device lock screen UI will appear. It is so-called sleep mode. Once the device has awoken from sleep mode, if the proximity sensor's target is still for an extended period of time, the sensor will then ignore it, and the device will eventually revert into sleep mode [5]. Also, during a telephone call, the proximity sensor helps avoid (ignore) accidental touchscreen taps when mobiles are held to the ear. On Android platform proximity sensors usually return the absolute distance, in cm, but some sensors return only *near* and *far* values. It means that proximity sensors return binary values that represent "near" or "far." In this case, the sensor usually reports its maximum range value in the “far” state and a lesser value in the “near” state. Typically, the “far” state is a value more than 5 *cm*, but this can vary from sensor to sensor [6].

The problems that exist here: the differences in sensors and their prevalence, the lack of standards for querying results. One of the main problems - “proximity” should be determined by the service (algorithm) itself, and not by the technical capabilities of the sensor. And of course, proximity sensors could not detect the type of targets. In other words, for the proximity of mobile devices, we could not detect that the target is some mobile device.

In addition to proximity sensors, we can use other available sensors. For example, an accelerometer could be used to determine the actions performed simultaneously on two devices (nearby mobile subscribers perform the same motions with the phones when they see each other). It is so called authentication by shaking - a mechanism for human-assisted authentication and a secure pairing of mobile devices. This mechanism is defined by the use of common movement of devices as a shared secret for their mutual authentication, simultaneous shaking of devices as user method to effect common movement, and embedded sensing of device movement with a single accelerometer [7]. Here, obviously, we see the violation of the principle of non-disruptive proximity detection. It is necessary to interrupt all other actions and go directly to the process of analyzing the synchronicity (similarity) of movements. It is also not clear how, in this case, to determine the proximity of several mobile devices.

A whole group of methods may be related to the fact that we will try to compare measurements on different devices. For example, you can try using a magnetometer and compare readings on several devices (candidates) [8]. There are options in which we will compare the recorded sound (noise) on the candidate devices. All methods leave open questions about the accuracy of measurement.

All this leads to the fact that the most practical approach to determining proximity is the usage of network services (so-called network proximity) [9]. Due to the limited distribution of the signal of wireless networks, the very fact of, for example, the visibility (availability) of any Wi-Fi access point means the location of the mobile device in the vicinity of this

point. The same is true for Bluetooth [10]. A typical example is NFC payments when the phone is located directly at the payment terminal.

Network interfaces are some common denominator for mobile device sensors. Also, mobile operating systems support, naturally, the background operation of these "sensors", application program interfaces for access to which are standardized [11].

In fact, now, proximity for mobile devices in practical implementations is precisely network proximity. The 3GPP standards on ProSe describe exactly network proximity.

III. On ProSe standards

The main motivation to define D2D communication in 3GPP standards is for Public Safety (PS). So, as per initial ideas, authorities such as police, firefighters, and ambulances should be able to use D2D communications to replace relatively old technologies, similar to TETRA [12].

For operators, the main motivation for D2D connectivity is the potential ability to offload traffic from the core network and deploy (in a controlled manner) a new communication paradigm. And the idea of 3GPP with D2D standards is to avoid configuration complexity of current ad-hoc networks, optimize resource usage, and keep everything under the control of operators.

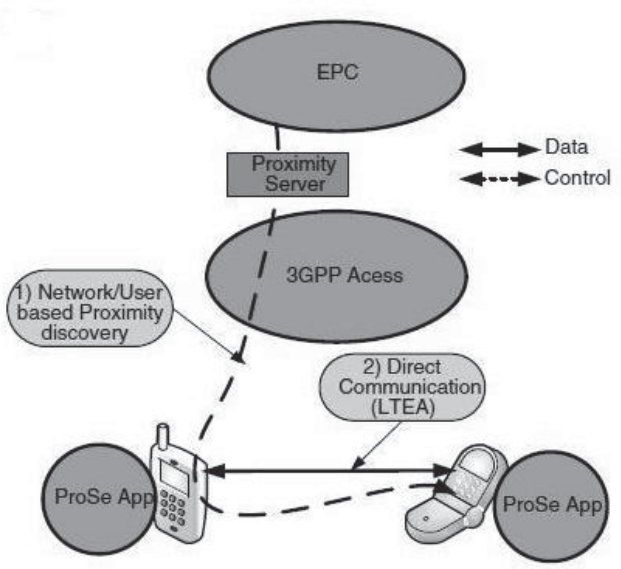

Fig. 1. The code D2D architecture [13].

Technically, there is inband D2D and outband D2D. Inband D2D means licensed (e.g., cellular) spectrum. The main argument here is controlled interference and QoS provisioning. Inband D2D links could share the same radio resources with cellular communications (underlay category) or use a dedicated set of radio resources (overlay). Outband D2D communications use unlicensed spectrum. The main idea is to eliminate the interference issue between cellular links (licensed spectrum) and D2D (unlicensed spectrum). It means that a mobile device should have an extra interface. Usually, it is some like Bluetooth or Wi-Fi. There is controlled and autonomous outband D2D. It is about controlling (un-controlling) the second interface by cellular.

In D2D communication, by the definition, devices communicate with each other without intermediate nodes. The proximity of mobile devices (UE – user equipment in the spec) may allow for high bit rates, low delays, and low energy consumption. Also, the radio channels could be simultaneously used by cellular and D2D links.

Accordingly, proximity services were considered from the standpoint of determining a nearby device for connection. Our idea that we are promoting in this article is that proximity is a certain metric that can be used as a trigger for various services that do not necessarily imply a direct connection of mobile devices. This does not deny the standard, but rather offers its extension.

As per 3GPP, proximity services can be divided into two stages: proximity discovery and direct communication. Fig. 1 from [13] describes the core architecture for D2D. There is a new network function – Proximity Server. This server should provide the connection between application servers and the mobile network. It could identify the proximity between mobile devices and inform the application servers about the proximity opportunities. D2D sessions could be initiated from the proximity servers by sending an initiation request to the Mobility Management Entity (MME – core signaling node).

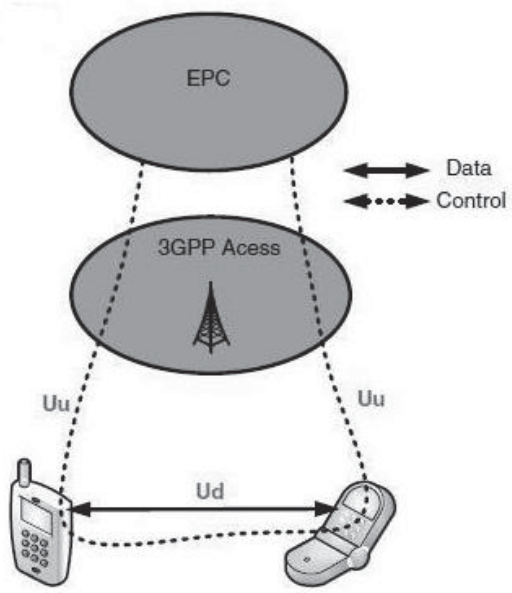

Fig. 2. D2D access model [13].

The MME is then responsible for initiating the D2D radio bearer setup and delivery of IP address for involved devices. In the terms of network access, ProSe uses the cellular link (*Uu* in Fig. 2) for a control plan. For data plan, D2D would need a new Direct Mobile Communication interface (*Ud* in Fig. 2).

3GPP introduces two new System Information Blocks (SIBs) that are transmitted over the LTE RAN (Radio Access Network): SIB18 and SIB19 to support direct communication

and direct discovery functions for D2D enabled devices. D2D model uses the same security architecture that LTE uses to provide USIM and authentication for a network, user, and application domain security [14].

On the spec, 3GPP describes the following scenarios for proximity discovery [15]:

- ProSe direct discovery: a procedure employed by a ProSe-enabled device to discover other ProSe-enabled devices in its vicinity by using only the capabilities of the two devices;
- System-level (originally - EPC-level) ProSe discovery: a process by which the EPC determines the proximity of two ProSe-enabled devices and informs them of their proximity.

Accordingly, there are two communication modes for D2D: the network independent and the network authorized. The network independent mode does not require any network assistance to authorize the connection. In this case, a communication is performed by using only functionality and information available locally to the devices in proximity. The network authorized mode for ProSe direct communication always requires network assistance (from the EPC) to authorize the connection.

In Fig. 3 (from Qualcomm, who started D2D efforts) we provide an application model (original document by Qualcomm points to Samsung for this picture) which is close, in some aspects, to our vision.

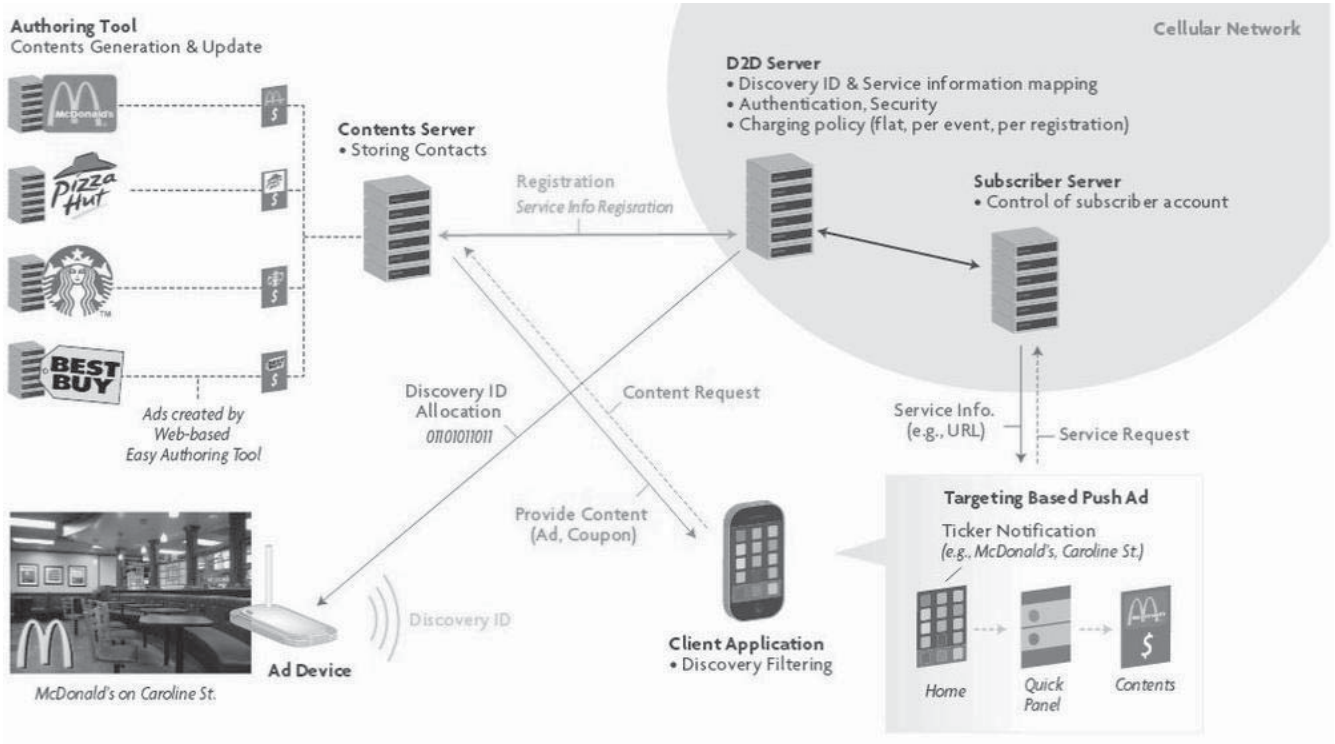

Fig. 3. On application model [15].

Shortly, this model could be described so. A device (Ad Device in Fig. 3) requests so-called Discovery ID for D2D server. This server supports mapping between D2D and content servers. Ad Device distributes obtained Discovery ID. This ID could be received by Client Application (device in proximity). Client Application uses obtained ID (ID, obtained from Ad Device in proximity) and requests content from Contents Server. Note the absence of the direct link (direct connection) between Ad Device and Content Application (between devices in proximity). So, proximity here is just a trigger.

IV. On Proximity Services Models

In this section, we would like to describe what are services based on proximity or how they, in our opinion, can be classified.

Firstly, the proximity of mobile devices opens (by symmetry - closes) access to any information. The model can be imagined as a display of a dynamic web page, the content of which varies depending on the proximity of mobile devices. Different elements of this page can be connected (depend on) the proximity of various devices (one device, group of devices). This is a typical example of a context-sensitive (context-aware) service [16]. Content provide should be able to set the visibility of data depending on the proximity. By the analogue: right now content provides may allow (disallow) access to data depending on geo-location. The same should be true for proximity metrics.

The interface of the application (s) may vary depending on the proximity of other devices (the presence or absence of devices nearby). This is a typical example of ambient intelligence (AMI) [17]. Downloading data in applications

(downloading data) becomes possible depending on the presence (absence) of other nearby devices.

Access to data based on proximity should not be tied to mandatory registration with telephone operators. For example, no registration is required for geo-positioning services. Creating a website and accessing it using mobile devices also does not require registration. Registration of services is understandable from the point of view of the business interests of mobile operators, but it is difficult to explain from the point of view of users.

Proximity-based service should support the ability to receive push notifications when devices are in proximity. A subscription to push-notifications should be tied to proximity metrics. Or, from a practical point of view, proximity metrics should be available to the subscription management service.

Support for a dynamic list of nearby devices. This refers to the work with this list (to support an appropriate API): adding and removing devices, getting their identification, etc. Access to this list should be supported from the devices, included in this list (the ability to can read this information).

Determining proximity should be done without accommodating users and not interfering with the work of other applications.

Establishing a connection between two applications upon determining proximity. Actually, it is about D2D connection.

As you can see, device connection (in fact, data exchange in applications) is just one of the options. Here it is important to note one more thing - a direct connection in D2D communications, in modern terms, there is a connection (data exchange) with some unknown device. It is one of the serious objections to the direct connection of devices (applications) in services. In an era of heightened attention (and real problems) with security, direct connections (loading data from another device) are rare exceptions in mobile services. These are safety issues. It is possible that this approach will work in corporate applications, but for mass services, direct data download from random third-party devices is taboo.

And the listed services should be interpreted, first of all, as a set of program interfaces for their inclusion in applications.

V. ON PROXIMITY SERVICE IMPLEMENTATION

In this section, we discuss the practical implementation of proximity-based content models that follow the above-mentioned principles.

It is a mobile application on the Android platform that presents a simple way to exchange text information between mobile phones in proximity without the connectivity between devices. It uses such features as Bluetooth extended inquiry response packet and Bluetooth advertising packets and does not require pairing and security issues associated with connectivity. It can work with classic Bluetooth and BLE.

Usage models: chats with mobile users in proximity, quick announces for business, mobile classified, badges (e.g., announce a link to own profile if the social network), etc. Sometimes we need a communication method with a short time limit and strong localization (regional restrictions). For example in shopping malls or buses, we want to know the promotions of each store or we want to chat anonymously with the customers or passengers around us. The method should not require a web operator or server, not require sign up or some authentication. It is just like people's voices, convenient and quick, only send to people nearby.

The idea is very simple. A Bluetooth device does not need to be paired to get the names of other devices. Therefore mobile phones can exchange data via Bluetooth device name. Mobile phones save messages as device names and advertise any message (actually – own name) to other mobile devices phones in network proximity. The same mobile application can both announce (broadcast) its own name (message) by creating dynamically a node in Bluetooth and read announcements from applications on other mobile devices. And the network proximity provides here a weak positioning, reduces the exposure of location privacy, and provides the same effective location service. The network proximity method naturally provides a location information filter based on wireless network propagation range, just like people's voices [18].

The main screen is illustrated in Fig. 4.

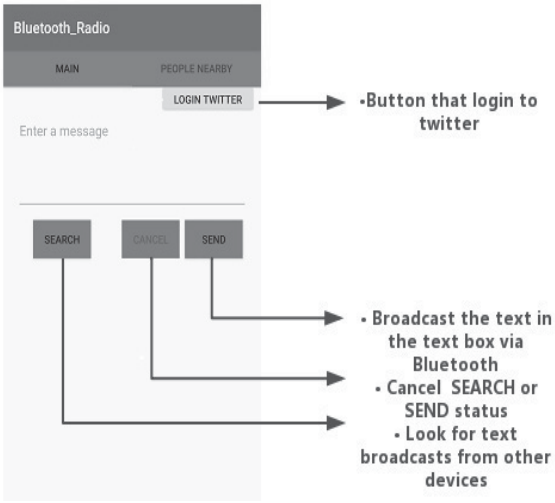

Fig.4. Bluetooth Radio application

A mobile user can type some text and broadcast it via a periodically changed name of Bluetooth node on the own device. Via *Search* button mobile user can scan (receive) broadcasts from other users. By default, messages are anonymous. Alternatively, a user can login to Twitter (it could be Facebook or LinkedIn too) and confirm own identity. His name in Twitter will be distributed together with custom messages. Note also that the resulting text during its output can be analyzed (parsed), and the detected standard fragments (phone, email, URL, etc.) will be displayed as clickable objects (hyperlinks), just as it is done, for example, when displaying SMS.

After users click *Send* button, the software opens a background service. It firstly detects users' text length, returns directly if the upper limit is exceeded, and segments it if it

does not exceed the upper limit. After that, the process adds the serial number, the number of segments, and the ID to each segment. After this segmentation, the service enters the loop process. It changes the public name of the Bluetooth node at the device at regular intervals until the time runs out, or sender cancels it.

And users can cancel the sending process at any time, or within a certain period of time. The recipient only needs to open the software, click search to start scanning the information. This process can be canceled at any time, or automatically stopped after a period of time, which can save the energy of the mobile phone. For the recipient, after clicking *Search* button, the program will open a background service, continue to search for nearby devices, and try to get their name. It stores the identifiable code, MAC or UUID of the discovered class, and creates an instance of each device with the name attribute to store its information. Whenever the system receives the "notification of device discovery", it checks whether it has a name attribute, whether the length of the name is less than the minimum value, and whether there is software-defined coding information in the header. If this newly discovered name is compounded, the storage class is called to store her. The storage class will detect whether the stored ID and the ID of the message are the same in the class corresponding to the device code. If they are consistent, the device starts to send a new set of messages, so the device will discard the original storage and perform the class element. Format and save the message. Whether there is already the same numbered message, if not, it will not be stored, if not, it will be stored and counted. After receiving all the pieces of the text, the UI will be updated (Fig. 5).

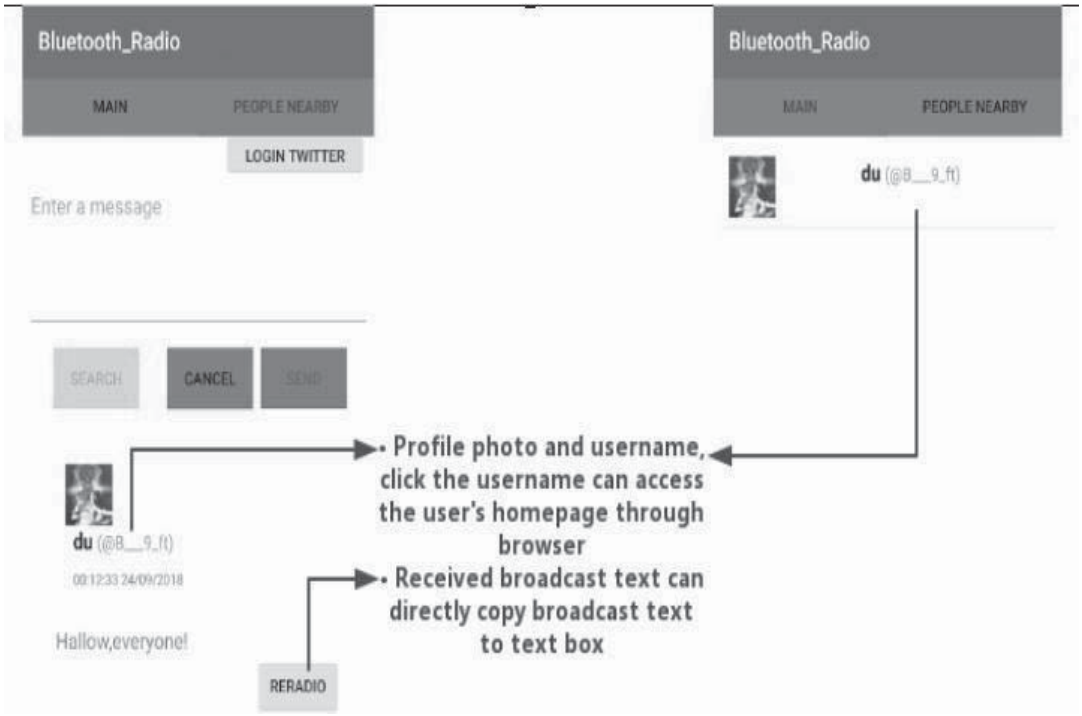

Fig. 5. UI screens

After receiving the content, the app can automatically extract the first address that conforms to the home page format through the regular expression and removes the URL address from the text content. Then use the extracted home page address to get the html page, and use the regular expression to locate the tag, extract the user's avatar and friendly user name. The username and profile photo are displayed on the page as the title of each message. Each displayed Twitter name has a hyperlink. The user can jump to the browser to open the user's home page by clicking on the Twitter username displayed on the page. If it is the first occurrence of a twitter username, it will also be added to the list of people nearby on the second page. The user can open a nearby person Twitter on the second page and contact him by Twitter. In a similar manner, it will work with any social network. Note that this is not a classic check-in [19] in a social network. The application does not publish anything in social networks and, accordingly, does not request such opportunities. The social network is used only to identify the user.

Technically, in Bluetooth communication, two steps are usually required. The inquiry process, where a master device discovers slave devices in proximity, and secondly, the communication process, where connections between them are established. In this paper, we don't require an actual pairing connection, so we focus on the first step. This is consistent with the principles stated above when we try to avoid direct connection. And this model is quite general since the process of searching for neighboring nodes is present in all wireless networks. And it is the search process (announcement) in wireless networks that can be used for the actual transfer of information. In this application, we implemented this approach with Classic Bluetooth and BLE. In a similar manner, it will work with Wi-Fi Direct and Wi-Fi Aware.

In the inquiry process, an originating device sends ID packet in 32 out of 79 frequencies. Any ID packet includes the device access code (DAC) or inquiry access code (IAC). It has a fixed length of 68 bits. The slave device continuously scans in the scan window (11.25*ms*), which enough for the master device to hop 16 times. Then the slave device goes to sleep, which is a time period that can be set by the manufacturer.

In Classic Bluetooth, valid Bluetooth names are a maximum of 248 bytes using UTF-8 encoding. However, the length of the name that the actual remote device can obtain is not necessarily equal to 248. For example, in BLE4.0, because of the limitation of the data packet, only a maximum of 30 bytes can be sent at a time [21]. The name is fragmented over one or more DM1 packets. The payload DM1 packet is 17 bytes. In the packet carries the name offset, the name length, and the name fragment. So, we can use MAC addresses to identify fragments from the same user and add 6 signal bytes in the fragment to ensure the integrity of the message. Among them, 3 bytes used to store a special signal (*0xef, 0xbf, 0xbd*) determining whether the name that was found in a scan is a fragment of our message. 2 bytes used to store serial numbers and total numbers, a value of which can range from *0x20* to *0x80*, for determining whether we received all fragments of the message. 1 byte used to store ID number for determining whether the fragment comes from the same *Send* button click.

When the device received all fragments, our program will combine them into a complete message and display this message on the screen. It is illustrated in the following code fragment:

```
while(messageSerial < messageTotal) {
while (0x80 == (0xC0 & byteMessage_Be[lang + contentSize])) { contentSize--; }
byte[] s = new byte[contentSize+baseByteMsg];
System.arraycopy(byteMessage_Be, lang, s, baseByteMsg, contentSize);
System.arraycopy(Signal, 0, s, 0, Signal.length);
s[baseByteMsg - 3] = (byte)messageSerial;
s[baseByteMsg - 2] = (byte)messageTotal;
s[baseByteMsg - 1] = (byte)messageID;
list.add(s);
lang+=contentSize;
messageSerial++;
contentSize=__size;
}
```

Therefore, in order to send a complete message, the Bluetooth discovery process may be performed multiple times. But this progress is a time-consuming process. According to the test in our Android phones, the maximum time of device discovery and name request response is 15 seconds. Once the device missed a certain part of the message, it had to wait for the entire message to be resent, it will waste more time. Therefore, we choose to change the name every 15 seconds.

Bluetooth LE adds a non-connectable mode. This mode does not require connection or discovery, no connection is required, and only a Bluetooth transmitter can be used without a receiver. Figures 6 and 7 illustrate BLE advertising and scanning [22].

In Bluetooth LE, advertiser broadcast advertising packets, called ADV_IND PDUs, on 3 advertising channel. Scanner scans in the advertising channel, listens to the ADV_IND PDUs, and sends back a SCAN_REQ when it receives an ADV_IND PDU.

In Android Source Code [23], we found some parameter settings about Bluetooth LE: advertising interval, advertising delay, scan interval, and scan window. It is illustrated in Fig.6.

```
/**
 * Scan params corresponding to regular scan setting
 */
private static final int SCAN_MODE_LOW_POWER_WINDOW_MS = 500;
private static final int SCAN_MODE_LOW_POWER_INTERVAL_MS = 5000;
private static final int SCAN_MODE_BALANCED_WINDOW_MS = 2000;
private static final int SCAN_MODE_BALANCED_INTERVAL_MS = 5000;
private static final int SCAN_MODE_LOW_LATENCY_WINDOW_MS = 5000;
private static final int SCAN_MODE_LOW_LATENCY_INTERVAL_MS = 5000;
```

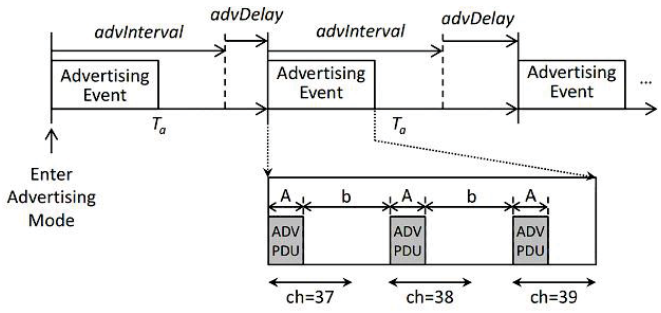

Fig. 6. BLE advertising [22]

The size of the advertising packet is 31 bytes. And Since Android 5.0 (API Level 23), we can only get a randomized MAC address of external devices via Bluetooth LE scan, and this MAC changed in each advertising packet. So, we can not use MAC addresses to identify fragments from the same user as we did in Classic Bluetooth. We choose to add a randomized UUIDs in advertising packet instead of MAC, this UUID changed in each time when click *Send* button. This information will occupy 16 bytes. It is illustrated in Fig. 7.

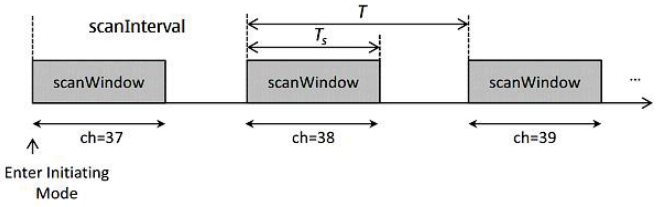

Fig. 7. BLE scanning [22]

We also need 2 bytes for serial numbers, total numbers like in Classic Bluetooth. According to BLE packet format, 4 bytes are left for ADType and length. At least 9 bytes left for the data of our message. We choose *ADVERTISE_MODE_LOW_LATENCY* and *MODE_LOW_LATENCY* for the best result. According to analysis [22], the discovery latency does not exceed twice the advertising interval (200 *ms*). And every iteration of the loop in our code can consume 100 *ms* on average.

Bluetooth 5.0 provides more flexibility for non-connected applications, enabling devices to carry more data in advertising packets. Now the advertising channels are abstracted into two types: primary advertising channel and secondary advertising channel. If we want to advertise data, which is more than 254 bytes, we can use the *AUX_CHAIN_IND* PDUs. *AUX_CHAIN_IND* PDU can chain multiple *ADV_EXT_IND* PDU [24]. Android 8.0 has already provided support for these new features of Bluetooth 5.0. In Bluetooth 5.0 data capacity and transmission speed of advertising packets are seriously enhanced. With Bluetooth 5.0 such localized advertising communications will be more efficient. This method may not only quickly share text, but also share binary data (e.g., audio).

VI. Conclusion

In this article, we looked at some content models related to proximity services. The appearance of proximity services in standards seems to us a very positive point. Services, based on proximity rather than geographical location, are the basis of context-sensitive models for mobile services (from a practical point of view, the basis of all mobile services). Our proposals relate to the fact that direct connections should, if possible, be separated from proximity services. Direct connections should be one of the possible applications of proximity services, but not the only one. Services based on proximity, in our opinion, should be focused primarily on data presentation (content), and not on connections. The paper describes also the implementation of a proximity-oriented mobile service based on the corresponding model proposed by us. As per directions for further work, we can offer the standardization for interfaces described in Fig. 3 at the level of the software framework.

Acknowledgments

The authors are grateful to the master student Du Zeiu (Open Information Technologies Lab, Lomonosov Moscow State University) for the work on the software implementation.

References

[1] Lin, Xingqin, et al. "An overview of 3GPP device-to-device proximity services." *IEEE Communications Magazine* 52.4 (2014): 40-48.
[2] 3GPP Release 12 http://www.3gpp.org/specifications/releases/68-release-12 Retrieved: Jan, 2018
[3] Su, Xing, Hanghang Tong, and Ping Ji. "Activity recognition with smartphone sensors." Tsinghua science and technology 19.3 (2014): 235-249.
[4] Lane, Nicholas D., et al. "A survey of mobile phone sensing." *IEEE Communications magazine* 48.9 (2010).
[5] Proximity Sensors http://www.wikiwand.com/en/Proximity_sensor Retrieved: Jan, 2019
[6] Android Development: Position Sensors https://developer.android.com/guide/topics/sensors/sensors_position#sensors-pos-prox Retrieved: Jan, 2019
[7] Mayrhofer, Rene, and Hans Gellersen. "Shake well before use: Intuitive and secure pairing of mobile devices." *IEEE Transactions on Mobile Computing* 8.6 (2009): 792-806.
[8] Gozick, Brandon, et al. "Magnetic maps for indoor navigation." IEEE Transactions on Instrumentation and Measurement 60.12 (2011): 3883-3891.
[9] Namiot, Dmitry, and Manfred Sneps-Sneppe. "Geofence and network proximity." In *Internet of Things, Smart Spaces, and Next Generation Networking*. Springer, Berlin, Heidelberg, 2013. 117-127.
[10] Namiot, Dmitry, and Manfred Sneps-Sneppe. "Wireless networks sensors and social streams." Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on. IEEE, 2013.
[11] Namiot, Dmitry, and Manfred Sneps-Sneppe. "On software standards for smart cities: API or DPI." In *ITU Kaleidoscope Academic Conference: Living in a converged world-Impossible without standards*, Proceedings of the 2014. IEEE, 2014.
[12] Mikulic, Marijan, and Borivoj Modlic. "General system architecture of TETRA network for public safety services." In *ELMAR, 2008. 50th International Symposium*. Vol. 1. IEEE, 2008.
[13] Mumtaz, Shahid, and Jonathan Rodriguez, eds. Smart device to smart device communication. Switzerland: Springer International Publishing, 2014.
[14] 3GPP TS 23.303 Proximity-based services (ProSe); Stage 2(Release 15), v15.1.0, 2018 https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=840 Retrieved: Feb, 2019
[15] Enabling the Next Generation of Proximal Services https://www.qualcomm.com/media/documents/files/whitepaper-expanding-horizons.pdf Retrieved: Jan, 2019
[16] Namiot, Dmitry, and Manfred Sneps-Sneppe. "Context-aware data discovery." In *16th International Conference on Intelligence in Next Generation Networks. IEEE*, 2012.
[17] Sneps-Sneppe, Manfred, and Dmitry Namiot. "On physical web models." In *Control and Communications (SIBCON), 2016 International Siberian Conference on. IEEE*, 2016.
[18] Namiot, Dmitry. "Context-Aware Browsing--A Practical Approach." In *Next Generation Mobile Applications, Services and Technologies (NGMAST), 2012 6th International Conference on*. IEEE, 2012.
[19] Namiot, Dmitry, and Manfred Sneps-Sneppe. "Customized check-in procedures." In *Smart Spaces and Next Generation Wired/Wireless Networking*. Springer, Berlin, Heidelberg, 2011. 160-164.
[20] Duflot, Marie, et al. "A formal analysis of Bluetooth device discovery." International journal on software tools for technology transfer 8.6 (2006): 621-632.
[21] Bluetooth, S. I. G. "Bluetooth core specification version 4.0." Specification of the Bluetooth System (2010).
[22] Liu, Jia, Canfeng Chen, and Yan Ma. "Modeling and performance analysis of device discovery in bluetooth low energy networks." In *Global Communications Conference (GLOBECOM)*, 2012 IEEE. IEEE, 2012.
[23] AndroidXref - Android Source Code Cross http://androidxref.com Retrieved: Jan, 2019
[24] Collotta, Mario, et al. "Bluetooth 5: A concrete step forward toward the IoT." *IEEE Communications Magazine* 56.7 (2018): 125-131.