# A Mobile Prototype for Clinical Emergency Calls

Cornelius Wille[1], Thomas Marx[1], and Adam Maciak[2]

[1] University of Applied Sciences, Bingen, Germany
{wille,marx}@fh-bingen.de
[2] Institute for Neuroradiology, University Hospital of Mainz, Germany
am@avallia.com

**Abstract.** In case of an emergency within a hospital, all available doctors get alarmed through a central collecting point. Only the doctor arriving first at the patient undertakes the medical treatment. All other doctors needlessly interrupt their current treatments or standby service. This article presents a prototype, to locate and alarm safely the nearest and available doctor. Mobile devices (smartphones und tablets based on Android or iOS) are used for localization, alarming and confirmation. Beside the localization in closed buildings the daily use of the prototype was tested. This incorporated the smooth integration into clinical information systems, the easy to use interface as well as the availability and robustness of the solution.

**Keywords:** emergency calls, indoor localization, paging, clinical information systems, mobile solution, eHealth.

## 1 Objectives

All business processes in hospitals are based upon *clinical information systems*. Besides data on the hospital's organization structure and on patients, these information systems encompass more and more context related information, like medical knowledge, guidelines or availability of resources. At night and on weekends 2-3 doctors attend 80 to 120 patients. Today, in case of clinical emergencies a notice is generated and distributed through clinical information systems, informing all doctors with a pager. This pager is an additional device, probably replaced by a smartphone in future.

We designed and evaluated a *prototype for clinical emergency calls* [1] which can be integrated into clinical information systems. This prototype locates and safely pages by smartphone only the nearest doctor depending on his availability.

## 2 Prototype

### 2.1 Localization within Buildings

Because of missing GPS signals in buildings, we decided to implement the localization upon Wi-Fi data, assuming complete Wi-Fi coverage for a building. To assure quick and safe alarming, all devices were made to send their current position continuously to the server, automatically excluding "lost" devices from the alarming process on server side.

In order to calculate the most current whereabouts, three different approaches were tested:

1. *Radio cell method*, blending relative signal strengths from different access points together into a location characteristic, see [2].
2. *Triangulation method*, implementing distance measurement via signal propagation time or through signal strength, see [3], [4].
3. *Table based methods* using measured reference points, see [5].

The radio cell method showed inaccurate results, especially for different locations being strongly influenced the same signal. For the triangulation approach the correct compensation of absorption [5] was probed, induced by the different wall consistencies, furniture and other objects. While these absorptions make it difficult to produce accurate results, these phenomena even proved to be helpful for the third method. Comparing the pros and cons of all three approaches, the table based method finally showed the best test results.

To apply the table based method, the buildings have to be metered upfront in the following way:

1. Identify all distinguishing position points within all rooms and floors.
2. Measure and register the Wi-Fi data ("fingerprints") for all reference points (weighted signal intensity to "visible" access points, identified by their MAC address). Within out testing we found out that we need 120 seconds per reference point to balance out fluctuations (median).
3. Built a graph by connecting reference points (vertices) that interlink directly to each other.
4. Annotate the distance between reference points (in meters) to each edge of the graph. Distances within staircase are weighted by factor.

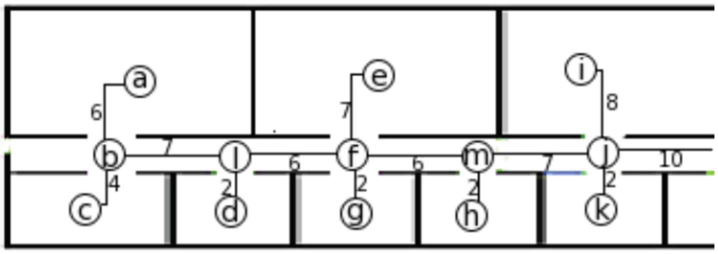

**Fig. 1.** Floor with weighted graph

We built a tiny app for Android and iOS to support the metering. This app sends the reference point data to the server while the weighted edges have to be added manually on server side.

**Fig.1** shows a floor represented by a weighted graph. A positioning service was implemented to run in the background of the mobile app. Test series showed that scan duration of 10 seconds (delivering approx. 11 scan results) provided satisfactory stable values. Unstable access points dropping out of the 11 scans were weighted by 75%. The measured signals are send continuously (e.g. every 2 minutes) to the server and get matched with the fingerprints. To become more tolerant against minor signal aberrations, our table based method applies *Euklid's distance function*, as proposed in [2], see **Fig.2**.

$$\sqrt{\sum\left(s_{ap}^{reference\ tuple} - s_{ap}^{messured}\right)^2}$$

**Fig. 2.** Euklid's distance function

Within our tests the table based method always identified the correct floor and was off 10 meters in worst case ever observed. We detected no significant differences between iOS and Android based devices. The scans consume only about 2% processor load, leaving the system in energy save mode with no implications to other apps running in parallel.

## 2.2 Notification of the Current Position

Starting our prototype, the user initially confirms his availability within the MainScreen of the app (see **Fig. 3**). Herby, the app connects to the server, synchronizes the settings and starts the background service, see **Fig. 4**. The ConnectedScreen is displayed. Running the main functionality in background allows the user to use its mobile device for any other tasks in parallel. The GUI of the prototype in the ConnectedScreen can be put in background without affecting the functionality of the background service. In addition the user can change its status within the GUI. Whenever the user decides to log off, the background service will terminate as well.

While the user is logged on to the server, the background service continuously scans for access points and sends the data over to the server. In parallel the service listens to incoming emergency calls.

## 2.3 Identification of Doctors Nearby

The identification of the doctor who is (timely) closest to the emergency was understood as a graph problem. Rooms represented vertices while the option to move from one room to another was modeled by vertices plus annotations (distance). To optimize response times the calculation of paths was pre-computed by the *Bellmann-Ford*

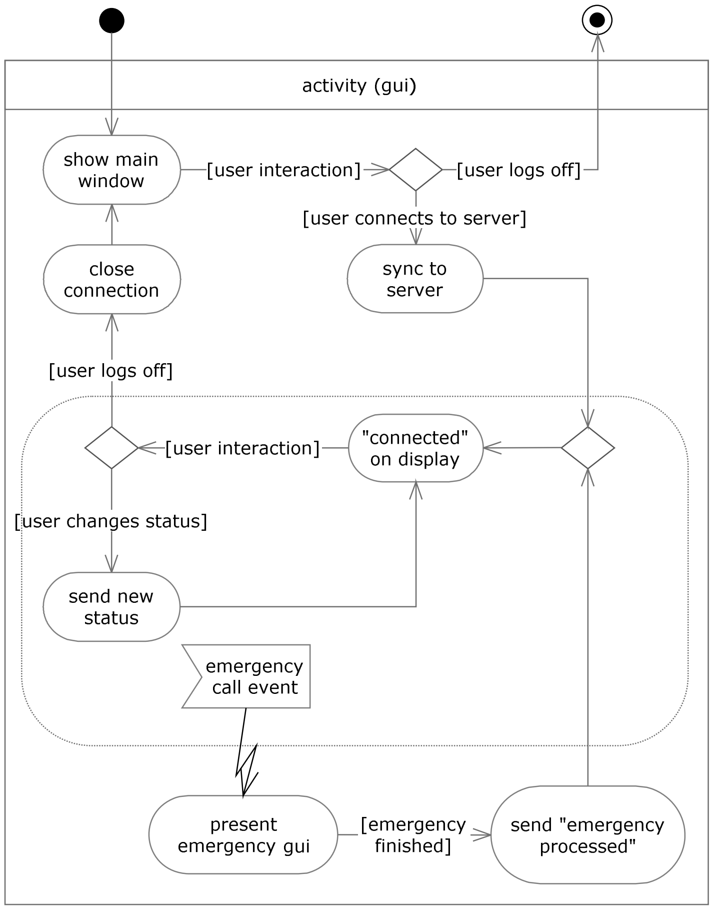

**Fig. 3.** Front end (activities MainScreen and ImageScreen)

*algorithm* [6], another option would have been to apply the *all pairs shortest path* algorithm. Obviously the hospital's graph model and the calculated paths have to be updated regularly to reflect changes, like new doors or construction areas.

## 2.4 Alarming Process

For the alarming process we evaluated two different technologies:

1. Push notification service The push notification service allows to signal incoming messages within an average of 1.4 seconds. It even wakes up iOS devices from version 3.0 and higher from the *enhanced standby mode*. But there are two problems that make the notification service inapplicable: Push notifications are funneled through Apples PUSH server with no guaranteed processing times.

We observed delays up to 6 minutes. Secondly only the iPhone family supports push notifications, iPads have to stay online discharging the battery. In context of Android no such restrictions apply.

2. Background service Fortunately with iOS version 4.0 Apple introduced the option to run (limited) background processes from an app, just like Google's Android always did. With iOS so called *local notifications* must be send to a background process in special mode, to wake up a device from stand by. Putting these kinds of apps onto the App Store, apple reserves the right to revoke the app without reasons. In context of Android there are no such restrictions.

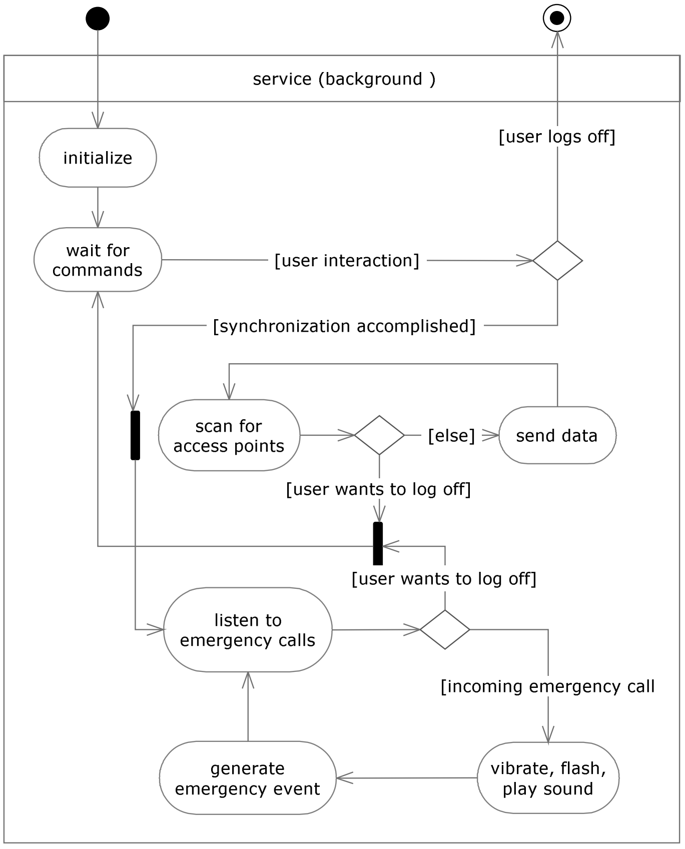

**Fig. 4.** Background service

The background service of our prototype waits for incoming notifications from *NetworkManager*. In case of arrival, see **Fig. 4**, the user is alarmed (sound, vibration) plus the *EmergencyScreen* is displayed by the front end activity. The background service continues to wait for notifications right after notifying the front end. The *EmergencyScreen* requests the user to accept or decline the call and will only stop after having received an input. The result is given back to the server that may alarm additional doctors.

## 3 Case Study

### 3.1 Mobile Devices and Clinical Information Systems

More and more mobile devices are used for so called *mHealth applications* in hospitals. Here, specific demands apply, like the ability to be sanitized, certificate on non-interference, usability, availability, battery performance etc. on to a reasonable price to allow general provisioning.

For our *eHealth prototype* [7] we evaluated standard devices (iPad), dedicated devices (Dell Streak) and design studies (Intel Motion Computing C5). We diagnosed significant differences. Especially the operating system has significant influence on professional use, because vendors of clinical information systems might lack native clients.

### 3.2 Integration of mHealth-Applications

Specific requirements for *mHealth applications* within clinical use are: complete encryption, strict authentication, authorization, perfect usability and self-explaining interfaces. Medical staff got to be assisted in its daily work, supported by up to date contextual data and operations that are executed directly identically as upon workstations.

In corporation with our partner, the University Hospital Mainz, three different mHealth solutions were evaluated to integrate our prototype: *drchrono iPad app* [8], *checkPAD MED* [9] und *myCare2x* [10].

Besides evaluating the aforementioned criteria the applications were rated upon openness and ease of integration, using common medical standards, like the communication standard *HL7* [11], the image communication standard *DICOM* [12] or the *LOINC-Code-System* [13].

All three products implement well the core requirements for clinical environments. *checkPAD MED* and *drchrono* showed best usability and integration option for our prototype [7], allowing our server to receive information emergency call including the location.

### 3.3 Testing within the University Hospital of Mainz

For a case study we implemented the following setting: The 5th and 6th floor of the University of Mainz at Staudinger Weg 9 was chosen. All rooms, floors, the staircase, Wi-Fi access points were identified and became part of the building's model. No additional technical infrastructure was added. Reference points were determined by metering the individual Wi-Fi perception plus the distance to nearby reference points. A weighted graph was built upon these reference points. Three testees were equipped with Android devices (different HW and OS versions) plus the prototype app. A notebook running the server engine was added to the network. An inspector constantly compared the testee's location to the reported one on the server. Test alarms were send to different devices and locations. The case study ran on a single day.

The case study delivered the following findings: The app constantly delivered acceptable location data without crashing. Only once we observed a significant fault of 10 meters which was still acceptable. The alarming process, identifying the doctor close by, worked very reliable without any problems. During the complete tests, the doctors approved an easy to use interface. There were brief instructions given beforehand, no tutorial. Neither client nor server software were able to detect connection loss in time (limit set to 3 minutes). Finally the battery consumption of competing apps and phone usage became critical for our test plus we were not able to detect user failures, e.g. whenever a doctor left his device in his jacket.

## 4 Summary and Outlook

We presented a full functional prototype for an intelligent paging of doctors on clinical emergency calls. The prototype worked on Android and iOS devices and integrated well with clinical information systems. A case study in a real hospital infrastructure proved that the prototype was able to alarm the doctor being close-by the emergency.

Tests showed that there is still some room for improvement, e.g. on calculating the exact location. Here, the correction of measuring faults and to achieve a better affinity degree *Spearmans rank correlation coefficient* [14] should be investigated on, using different antennas simultaneously. The observed sporadic skipping between rooms might get fixed by including historical data into calculations. Beside these tweaks it is important, that the Wi-Fi fingerprints get regularly update to respect changes.

Future tests should include additional building types, because the tested University Hospital of Mainz is a homogenous, mainly column supported building. In production scenarios the demand of selective alarming should be relaxed to probably three doctors at the same time. Features like end-to-end encryption, anonymization of data and battery performance protection have to be added, too.

Besides these improvements there is still one critical problem to solve: it still might take too much time to recognize whenever clients loose server connection. Constant keep alive signals or other brute force approaches will quickly discharge the battery.

## References

1. Drchrono Inc., Electronic Medical Records (2012), `https://drchrono.com/` (retrieved February 11, 2012)
2. Friedrich, R.: eHealth - KlinischeInformationssysteme (Bachelor Thesis). Mainz: Joh. Gutenberg-University Mainz (2011)
3. Healthcare Consulting GmbH, mycare3x (2007), `http://www.mycare2x.de/` (retrieved February 11, 2012)

4. HL7 Deutschland e.V, HL7 (2012), http://www.hl7.de/standard/standards.php (retrieved February 11, 2012)
5. Jehl, G.: Deterministic WLAN Position Determination, Universität Freiburg (February 7, 2007), http://de.pdfsb.com/readonline/596c5a41664174305748423141483568555 13d3d-4508901 (retrieved February 11, 2012)
6. Lange, K.: Taschenbuch der Hochfrequenztechnik, 5th edn. Springer, Heidelberg (1992)
7. Leinberger, M.: EinNotrufsystemmit Andoid Smartphones (Bachelor Thesis). Joh. Gutenberg University Mainz, Mainz (2011)
8. Lohmann & Birkner Health Care Consulting GmbH (2012), http://www.lohmann-birkner.de/lohmann/wDeutsch/HP_Checkpad/Index.php (retrieved February 11, 2012)
9. Meyer, M.: Kommunikationstechnik, Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, 2nd edn (2002)
10. National Electrical Manufacturers Association, Digital Imaging and Communications in Medicine (2012), http://medical.nema.org/ (retrieved February 11, 2012)
11. Dornbusch, P., Zündt, M.: Realisierung von Positionsortungen in WLAN. Dresden (2002), http://www.fireflow.de/publikationen.htm (retrieved February 11, 2012)
12. Regenstrief Institute, Inc., Logical Observation Identifiers Names and Codes (2012) http://loinc.org/ (retrieved February 11, 2012)
13. Cormen, T.H., Leiserson, C.E.: Algorithmen – EineEinführung, 2nd edn. Oldenbourg Verlag (2007)