cleansing component, the raw data are cleaned of duplicates and unnecessary content. This step also allows us to clean data from blocked phone numbers and e-mail addresses.

The data organization component performs a first alignment of the raw data to the dynamic graph model. Here we look up user profiles, and conversation topics or create them on demand if they did not exist.

Finally, in the graph modelling component, the data are inserted into the graph model as vertices and edges in a single transaction.

The final phase is the data interpretation in which the graph model is used to generate visualizations, recommendations and alerts.

Current research on reality mining tends to focus on one of two aspects: (1) implementing mechanisms and tools that run on mobile devices or (2) implementing methods to analyze data sets in the cloud with low latency.

Our research focuses on dynamic graph models. These models keep their history as the graph changes. In this way we are able to analyse various phenomena, for instance changes in the structure of social networks. On top of these dynamic models we use queries that adapt their results whenever the graph changes in order to reflect the new model.

We are working on integrating methods learned from this research into the design of collaborative applications and in applications of computer supported cooperative work.

Mobile devices have become an important platform for understanding social dynamics and influence, because of their pervasiveness, sensing capabilities and computational powers. The convergence of mobile and cloud computing is forming the breeding ground for real world applications in a research field known as reality mining. This is undoubtedly an interesting field, which is gaining new momentum with the convergence of mobile and cloud computing.

**References:**
[1] B. Begole: "Ubiquitous computing for business", Upper Saddle River, NJ: Financial Times Press; 2011
[2] Nathan Eagle and Alex Pentland: "Reality mining: sensing complex social systems", Personal and Ubiquitous Computing, 2006.

**Please contact:**
Matthias Steinbauer, Ismail Khalil, Gabriele Kotsis
Institute of Telecooperation, Johannes Kepler University Linz, Austria
E-mail: {matthias.steinbauer, ismail.khalil, gabriele.kotsis}@jku.at

# Positioning Terminals in Mobile Computing Networks

by Francisco Barcelo-Arroyo, Israel Martin-Escalona and Marc Ciurana-Adell

***The ability to pinpoint a terminal's position is useful for many applications of mobile computing and for network optimization (eg handovers, tariffs, resource management). A range of techniques are available to obtain a terminal's position [1]. GPS, for example, is used externally to the network and achieves good accuracy outdoors, with the trade off of increased energy consumption. Communication devices, however, are frequently used indoors, connecting to private networks, such as WLAN. Since GPS is inaccurate indoors owing to signal blockage and multipath errors, further research on indoors localization through communication networks is required. Mobile computing is linked to indoors positioning in applications such as: aged care, remote health control and security of buildings such as hospitals.***

The fingerprinting technique is used extensively for WLAN positioning. The terminal collects the received signal strength from several access points and, during a precalibration phase, compares the achieved vector to the vectors previously recorded along with their positions. This technique does not involve modifications to the hardware. Other techniques use the time of flight (ie the time needed by a signal to travel between two nodes) to estimate the distances to several access points at known positions and then apply a trilateration process. The time of flight is more consistent than the signal strength. But in order to avoid modifications to the terminal's hardware, the time of flight must be obtained from communication messages by using only software.

Recent research at the Technical University of Catalonia has led to a procedure to measure distances between terminals [2]. This procedure is aimed at obtaining the time of flight after adding timestamps to messages sent and to the corresponding acknowledgements received. The round trip time (RTT) is computed as the difference between both timestamps, and the distance between the nodes is inferred by considering that the trip occurs at the speed of the radio signal. The software must interact with the link layer of the protocol stack of the device. A simple approach is to use a network interface capable of providing time measurements made by the hardware, but the timestamps performed do not have an acceptable accuracy for many location applications (eg the characteristics of IEEE 802.11 lead to a resolution of one microsecond corresponding to 300 metres in distance.)

A more sophisticated approach is presented in Figure 1. The protocol stack of the terminal is updated by introducing two software layers that are registered in each terminal. The registration process is run once and replaces the network interrupt handler (responsible for handling the events related with the net-

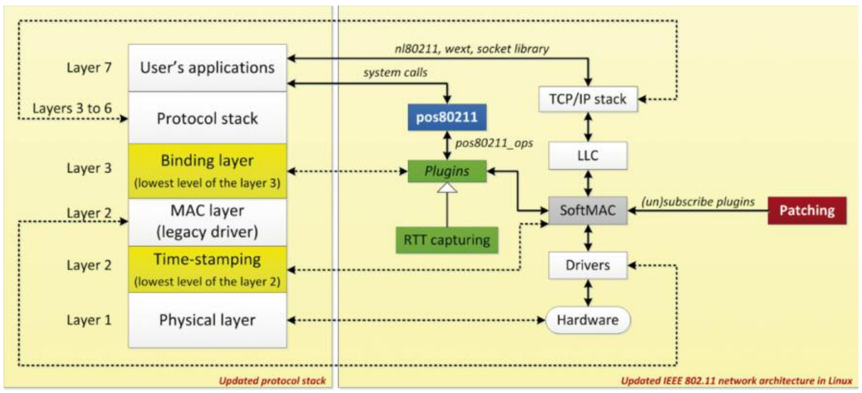

*Figure 1: Proposed Linux implementation of the measurement system.*

work interface) with a new one. This new handler analyses the traffic transmitted and received and adds timestamps to certain messages so that the RTT can be finally computed. Figure 1 also shows the network architecture including the interrupt handler in a Linux-based device. The sources of the Linux kernel have been patched to allow location metrics (ie RTT) to be observed. These changes alter the mac80211 subsystem, which implements most of the common MAC features in Linux. The goals of these changes are 1) to allow the location-related capabilities to be registered and released and 2) to add timestamps to the messages exchanged between the terminal and the access point. The capabilities are implemented as plugins, so that each works as standalone. This allows isolation of the bugs and extension of the capabilities without impacting those that are already working. An RTT plugin has been developed in order to calculate the RTT between a node and an IEEE 802.11 access point. This plugin is responsible for most of the tasks developed by the interrupt handler. Specifically, it filters the traffic not suitable for location purposes and matches the transmission and reception messages involved in an RTT, so that the RTT can finally be computed. The interaction between the user's applications and the RTT plugin is done by means of system calls to a new module named pos80211 [3]. This module provides the computed and buffered RTTs to the user's applications.

Currently, the proposed implementation has been prototyped and tested at distances shorter than 30 metres with good results. Future work includes more testing (eg different scenarios, longer distances) and developing new plugins for hyperbolic, instead of circular, trilateration. The current plugin provides the RTT which is useful for circular trilateration, while time differences are appropriate for hyperbolic. The use of this proposal in ad hoc networks is also investigated.

**References:**

[1] Y. Gu, A. Lo, I. Niemegeers: "A survey of indoor positioning systems for wireless personal networks", IEEE Commun. Surveys & Tutorials, vol. 11, no. 1, pp. 13-32, 1Q 2009

[2] F. Barcelo-Arroyo, M. Ciurana, I. Martin-Escalona: "Process and system for calculating distances between wireless nodes", U.S. Patent 8 289 963, October 26, 2012

[3] J. Corbet, A. Rubini, G. Kroah-Hartman: "Linux Device Drivers", O'Reilly Media, Third edition, 2005

**Link:** http://grxca.upc.edu

**Please contact:**
Francisco Barcelo-Arroyo, Universitat Politecnica de Catalunya, Spain
Tel: +34 934016010
E-mail: barcelo@entel.upc.edu

# DarkDroid - Exposing the Dark Side of Malicious Mobile Applications

by Engin Kirda

***The Android operating system is a burgeoning platform for deploying mobile applications to users, with more than 550,000 activations per day and an approximate 75% share of the global smartphone market that eclipses the once-dominant Apple iOS [1,3]. This trend is expected to continue, considering that Android's liberal licensing structure, open development environment, wide adoption across multiple hardware manufacturers and carriers, and modern end-user experience make it an attractive platform for both civilian and military use.***

The Android security model is a particularly compelling reason for security-conscious organizations to adopt Android as a platform for developing and distributing mobile applications. In this security model, applications must declare a set of permissions that describes the set of privileged actions that might be performed during execution. Examples of such actions include accessing camera data, determining the location of a device, or placing phone calls. Prior to installation, the user must explicitly approve the set of permissions requested by the application. During execution, the Android OS is responsible for ensuring that the application only performs those privileged actions that have been approved by the user.

This containment-based approach to preventing malicious behavior has the benefit of potentially high scalability, since Android applications are not necessarily subject to a manual review process (as is the case for iOS applications distributed in Apple's App Store). Unfortunately, the discovery of Android-based malware in the wild has prompted concerns that the Android security model is inadequate, especially if Android devices are to be adopted in high security environments (eg, the US