\section{Challenges and Use Cases}

.

Applications mentioned in the previous section have several properties in common:

- they only work in areas where the application's coverage is high;

- they provide services to the users;

- provided services can be consumed on the same mobile device, through a specific application

For such an application to succeed, the provided service has to be in balance with the required contribution, and the application has to be advertised and distributed rapidly. These requirements result in well-polished, feature rich applications that emphasize the provided services, and have high marketing costs. If an application doesn't reach enough users or active user contribution is required which is too costly (see Google's parking app), the service dies out. This is a high cost, high risk scenario.

In the following we present the specific use cases the DroidLab framework fits for, and some of the most important challenges we have to tackle in the design and implementation of the framework.

\subsection{Use Cases}

DroidLab can be used in several ways, with different goals in mind. First, if an application developer needs sensory data for a provided service to work, he can deploy a sensor application over the DroidLab testbed (built from devices running the DroidLab framework) and gather the required information before the launch of his standalone application. Thus, DroidLab is used in this case as a crowd-sensing platform.

Second, if a given service requires mobile sensed data, but doesn't offer mobile content to the users, it is really hard to motivate users to download yet another application that will run in the background and consume valuable resources. However, if a device runs DroidLab, such applications can be deployed seamlessly, without distracting the user. There is no need thus to convince a large group of users to download and run each specific application, they have to be convinced just once, to install the DroidLab framework. Specific incentives and gamification methods will be however needed, as we discuss in the next subsection.

Finally, DroidLab's initial purpose was to be a research platform, a large-scale mobile test network similar to PlanetLab and SensLab, over which researchers can test their developed algorithms and protocols. This is because it is especially difficult to deploy large-scale applications for research purposes, as they usually don't provide any value to the user, operate only for short periods of times, and usually target specific user groups. DroidLab can be regarded thus as a large-scale testing infrastructure for mobile applications.

The majority of the currently sold phones are smartphones; regardless of their gender, age, and technical knowledge, users are persuaded to buy smartphones, resulting in the wide penetration of these devices. We designed DroidLab so as to not require any maintenance. After the framework is set up on the device and configured properly according to the users’ privacy requirements and available resources, it operates in a self-sustained manner, no user interaction being needed. This enables the deployment of DroidLab on devices that are owned by users who are not familiar with modern technology, wideningthe pool of reachable users. With proper marketing and communication, every smart phone user can be targeted. However, the main target group of the DroidLab framework are students and young adults , owning smartphones with mobile broadband access, who travel daily in their home town by different transport means, use multiple WiFi access points and near field communication technologies (Bluetooth, RFID, NFC).

\subsection{Challenges}

There are several challenges that a mobile crowd-sourcing framework has to handle. We will briefly explain how DroidLab tackles them.

\subsubsection{Modularity}

The DroidLab framework, from an Android perspective, consists of a core application and plugin applications. Each plugin can be installed or removed as an Android application. Each plugin requires the permissions that are needed for its use. This enables the user to assemble a setup that he/she is comfortable with, and is suitable for the capabilities of the device. If the user doesn't want to share his/her location, then by omitting that plugin the framework will be limited on the OS level not to use the location services.

The clear benefits of this modular approach come with some drawbacks. The DroidLab distributions will be fragmented, devices will have different plugin sets and even plugin versions. Users will also have the possibility to revoke permissions from the framework and limit the resource usage of plugins.

To address this issue, DroidLab has a dual conformance checking for the applications. When a developer requests devices for his application, he indicates the span of the deployment. He may target a specific user group – based on age, gender, country, urban area or countryside, device type, or user activity type - and may specify whether the application needs inter-user connections or devices can be scattered over a larger area. He can specify some required device capabilities as well: location sensor accuracy, network connectivity type, available sensors, etc. Finally, he should estimate the resources his application will consume on a device during the applications lifetime.

A strong point of DroidLab is the management of the resources. As different sensors have different types of resources, we introduced a resource description system. For each resource we define quotas. Quotas can be associated with the volume of consumed resources and the speed of consumption. A resource can have several associated quotas. These are used to present users with resource management decisions and to limit an application's resource usage.

A plugin interface is defined so that plugins can expose the quotas. Each quota has five usage levels. The user can choose not to grant any of that resource. Low, medium and high are different levels defined by the plugins and labeled by human readable labels. Finally unlimited indicates the user is willing to provide every available resource of the kind to the framework.

Several applications can be present in parallel on a device. The quota limits defined by the user apply for the aggregated resource consumption of the applications. The server is in charge of distributing the applications among devices in a way that each application will be able to access the amount of resources the developer required. This is the first level of conformance.

Once an application is pushed to a device, each delivered event and called method is monitored by the framework and authorized by the resource manager running on the device. Once the application's quota level is reached or the overall quota limit has expired, the framework denies the request.

\subsubsection{Security}

DroidLab ensures that the user is protected from both outside attacks and malicious applications. We identified a number of high risk points in the architecture.

We rely on Android Play Store's security measures to guarantee that neither the framework nor the plugin codes can be compromised. Both the framework and the plugins can be downloaded from the Play Store. This ensures that no third party application will be able to use the same package name on the device. The Android user management system ensures code integrity; thus, DroidLab isn't more fragile than any Android application.

An attacker could compromise though the DroidLab client by pushing to the client an application that either has altered quotas and permissions, or contains code that was not checked and compiled on the DroidLab server. To prevent such an intrusion, every application and accompanying resource descriptor is signed by DroidLab's asymmetrical key. The application has the public key compiled inside itself. This signature is checked every time the application is loaded from local storage.

Applications are written in Java, compiled by the server, and sent to the client. To make sure the client framework has full control over the application, only a limited set of the Java language can be used. Developers are limited to java.lang, java.util and java.math packages, excluding threading, Timer and class loading. Developers are allowed to use DroidLab's interfaces package, but cannot reference any class from the framework. Source code is checked for these imports and compiled without any external library. This ensures that applications cannot reach outside of the designated sandbox.

Finally, DroidLab uses intents to communicate on the device between the different components. Android permissions are defined and enforced for these intents, limiting senders and receivers of the intent to the trusted components of the client (framework and plugins). It is up to the user to keep his device safe, any installed Android application has to be authorized, and permissions have to be granted manually.

\subsubsection{Privacy}

One of the main user concerns is privacy. A crowd-sourcing framework can be successful only if the users can be certain that the data shared through the application doesn't violate their privacy. The required level of privacy varies from person to person, and DroidLab is designed to allow different levels of privacy.

The common basis of privacy is device anonymity. Each device generates a unique identifier and every data is associated to that identifier. The identifier is independent of any other device or user identifier. That means that if a user reinstalls the application, the two installations will have different identifiers and cannot be associated. As the framework only handles the task of storing log lines generated by the applications, other privacy issues are taken in account on the plugin level. We design every plugin with user privacy in mind. Permissions defined by the plugins are designed in a way to accommodate different privacy needs.

Log files have to be stored on the external storage, as most current phones have limited internal storage available. These files can be read and altered by any application having permission to the SD card. To protect the sensitive information, we use a XOR symmetric key coding and sign the saved files. This guarantees integrity and privacy. Files will be uploaded through a secure channel either manually or automatically. In case of manual settings the user can review file contents before deciding to delete or upload them.

Files uploaded to the server are stored in the user's private space. Users can access the information that will be shared by them. Three sharing options will be available to the user. Always share will grant developers access to the files immediately, speeding up the data processing. Share if not removed will grant developers access to the files after a given period, if the user didn’t delete them Finally, the strictest option, share when permitted, keeps files private until the user doesn't give explicit permission to share the files with the developer. Files can be reviewed but not altered by the user.

\subsubsection{Seamlessness and resource management}

DroidLab will run on user equipment, tablets and phones that have to perform well, in a seamless manner their usual tasks, even when DroidLab is running resource-hungry sensing applications in the background. Moreover, these devices are mostly mobile, with limited or costly network access. Thus, all these aspects (limited battery, CPU and communication resources vs. seamless and efficient operation) have to be taken into account when designing the framework.

Our preliminary experiments show that the framework in itself doesn't influence battery life. We created three passive monitoring applications: a battery meter, a CPU utilization capturer, and a running application capturer, all scheduled to run each second. Running these applications didn't increase the battery usage considerably, battery lifetime was not compromised.

On the other hand, active sensing like the use of the gyroscope, the accelerometer, WiFi or Bluetooth discovery, or active bandwith measurements will naturally consume more battery. Plugins can be optimized to reduce power consumption. To prevent the framework from draining the battery, thresholds can be set by the user, when to disable battery intensive tasks, and when to stop the framework altogether.

Another battery intensive task is the periodic upload of the collected sensing data. As this usually is not a time sensitive task, the framework takes in consideration the user's network preference, and also his indications on the minimum battery level at which the framework should upload the gathered data.

Current Android distributions don't support application level CPU limits and quotas, so our options were limited in terms of processor usage management. As the DroidLab application source code is checked to prevent starting new threads, the lifecycle of the application is managed by the framework. We are measuring the runtime of each application method call and flagging applications that take too long to return. However, plugin methods initiated by the applications are asynchronous, hence we do not measure CPU load resulting from a plugin call directly. An estimation of the CPU usage of CPU intensive plugin method calls will be incorporated in the quotas defined by the plugins.

Our application design guidelines suggest a timer or event based operation, with small tasks that aim to filter sensor data or change the state of the application. DroidLab was not intended to be a distributed mobile computation platform.

Finally, the Android platform distinguishes between internal and external storage. Internal storage is private to the Android application; using it for downloaded DroidLab applications adds another layer of security. Available storage capacity varies widely from device to device, thr usual bottleneck being the internal storage. Our internal storage needs are way more modest than those of most Android applications. However, caching application logs before uploading them may result in the overuse of external storage.

\subsubsection{Incentives}

Many crowd sourced applications provide value-added services to the users in return for their collaboration. Another approach is the introduction of micro-payment systems to incentivize users in participating. A micropayment system in an automated sensing scenario opens up however many issues. As opposed to easy to evaluate tasks, like answering a question or taking a picture at a given location, it is hard to put a price tag on a periodic or event based sensing task. A monetary reward would also attract cheaters. We believe thus that other extrinsic motivators should be used to increase public involvement. We propose a gamified approach.

The framework in itself is just a delivery system for applications. Our goal is to be able to run more applications on more devices, and to be able to target specific user groups. This requires that users install more plugins,give more permissions and higher quotas. Moreover, they should provide detailed user profiles about themselves.

On the other hand each application benefits from different user behaviors. A pothole detecting application needs car users. Drivers using roads that are not yet part of the database are more valuable than users taking the same path every day. For an application that maps open WiFi networks, people not switching on their WiFi connection, or devices used only at home are useless.

We plan on implementing a system that has both a unified reward system and an application level reward system. The gamified system will be composed of points, leaderboards and badges. Points will be awarded for quotas that the applications spend. As developers have limited quota available, applications will be optimized to use fewer resources. So a quota spent by the application should be useful for the developer. For the users to increase the spent quota they have to make their setup desirable to the applications, meaning more plugins, permissions and higher quota limits. Leaderboars will be based on the points awarded to the users. With the increase of the user count, we will create different leaderboards for different time windows, and social leaderboards allowing users to compare themselves to their peers.

Badges will be the parts where the application developers can task and reward their users. Applications have the possibility to define badges that will be part of the system. Specifications of the task that has to be accomplished to earn the badge can be secret or known to the user. In the latter case, the framework will present the user with the description of the task. The application will be responsible for checking whether the requirements have been fulfilled or not. The framework will also define badges, which will be part of the tutorial introducing the user to DroidLab. User tasks will guide the user through the settings, and give the user some insight about the system.

\subsubsection{Scaling}

As opposed to crowd sensing applications, DroidLab will not have bootstrapping issues. DroidLab doesn't provide a service that relies on sensing data, thus the first user will receive the same user experience as a user joining the system when it becomes popular. In fact, early adopters will benefit from their history when it will come to leaderboards or badges: as badges are associated to applications, if an application is available for a limited time only, badges earned during that period will be more valuable. DroidLab's success won't depend on its startup speed.

DroidLab's server requests needed for its operation scale linearly with the number of devices. Most of the server interactions consist of uploading or downloading files related to a single user or a single application. Gamification related server requests will also involve a single user's information. Leaderboards will be cached and evaluated periodically. This makes the cloud an ideal platform for DroidLab. A more complex task is the distribution of applications among devices. Finding an optimal solution is a hard problem. We will use thus sub-optimal algorithms to find acceptable associations. This will result in sub-optimal device quota usage, but will conserve server resources.

\section{The DroidLab Architecture}

, includingSC,simply Aover DroidLab

currently operational.

\section{Conclusions}

\label{sec:conclusion\_and\_future\_work}

\section{Acknowledgment}

\label{sec:acknowledgment}

http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats/a\#subscribers

http://au.businessinsider.com/comparing-app-downloads-across-countries-2013-6

http://www.androidauthority.com/google-labs-open-spot-a-useful-application-that-no-one-uses-15186/

http://www.waze.com/

http://live.hopstop.com/

http://www.bizjournals.com/sanjose/news/2013/05/31/whitepages-acquires-crowdsourced-spam.html

http://www.lvmt.fr/ewgt2012/compendium

http://www.communitysensing.org/

https://play.google.com/store/apps/details?id=com.jiwire.android.finder

http://www.rootmetrics.com/

\end{document}

%%% Local Variables:

%%% mode: latex

%%% TeX-master: t

%%% End: