EXperimental Learning

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Big Data and Social Analytics certificate course

MODULE 2 UNIT 1
Video 4 Transcript

MIT BDA Module 2 Unit 1 Video 4 Transcript

Speaker key

AS: Arek Stopczynski

HY: Hapyak

AS: Hello again. Now, we know smartphones are powerful sensing devices. The question is, how do we actually collect this high-resolution behavioral data? We want to create applications that are able to run in the background for months or even years without actually bothering the user, without consuming too much battery life, without crashing. Think about your phone. Think about all the times that the application crashed on your phone. Now imagine that you need to have someone that is running continuously for weeks or months. Hopefully you will never remember that it's actually there because this is the ultimate goal of sensing: to be effortless.

00:00:50

And to create something like that there is a set of common functionalities that we want. The data needs to be captured from multiple sensors. It needs to be stored somewhere. It needs to be uploaded to the server in most cases. We want to be able to configure the phone preferably remotely and in real time or almost real time. We want to think about the privacy of the user. So, we want a way to incorporate user consent into the entire pipeline. We want to think about the encryption of the data so in case you lose your phone it doesn't suddenly expose all the behavioral data that has been collected for months or years. And this set of common requirements is the reason why here at MIT Media Lab in Human Dynamics Group we have built a library called Funf and this is a library that takes care of all this heavy lifting that you can use if you want to capture high-resolution behavioral data. Once you take the library, what you need to build on top is… you just need a thin UI that takes care of user flows and presenting the information to the user if any, but all the heavy lifting, encryption, data collection, uploading to the server, configuration, and so on and so forth is taken care of and can be very extensively configured.

HY: Funf does most of the heavy lifting, what is meant by this?

Capturing, storing, uploading to server, remote configuration, consent and encryption.

AS: And we have tested it. We have been testing this library for millions of man-hours data captured in Copenhagen Network study that I will also mention in the upcoming video.

00:02:17

So, how does one build a system or a library that allows for a collection of behavioral data? Let's take a look at a very high-level overview of the architecture of Funf. The core concept of Funf is a probe and a probe is an object, a piece of code that gets woken up and it's supposed to collect the data and pass the data up. That's all its responsibility and this is what it's supposed to do. It's supposed to do it in an energy-efficient way, shut down as quickly as the data is complete, but otherwise it doesn't have to bother with what happens to data later. It just needs to wake up and it needs to grab the data and pass the data further.

What wakes up the probe is a scheduler and this is an extensible service that is running and is actually able to see the probes that are configured and registered and wake them up at certain intervals or even allow them to capture data in a passive mode. So, every time any other application in Android captures some data point, we can also see that and store that. This is the passive mode of the probes.

HY: What has the job of waking up probes?

a. Sensor
b. Wifi
c. **Scheduler**
d. GPS

AS: Once the data is captured because the scheduler woke up the probes, what is happening periodically is archiving. So, the data gets transferred into small databases that live directly on the SD card, or internal memory of your phone, and those little databases… they can be JSON databases or they can be SQLite databases. They can be encrypted and they can be compressed. So, they are relatively small and they are also encrypted. So, again, if you lose your phone, the data doesn't get exposed.

00:04:00

Once we have the data in the internal memory, another part of the process is upload to the server. So, we can configure the phone in a way that periodically… every hour, every two hours, or maybe every 12 hours – it doesn't matter – to actually push those files into a predefined server and then there is the server responsibility to take care of that and if you are doing a research project, probably you will build such server yourself. If you're just using Funf for your personal use, you might hook it up to your Dropbox so actually your data files will appear in your own Dropbox on your own computer.

Another piece of functionality is exactly configuration. So, phones can download config of the server and this config has all the things that the phone should be doing: how often it should be uploading, how often the Wi-Fi probe should fire, should the Bluetooth be enabled, and so on and so forth. So, again, you can create it for yourself or if you're running a research deployment you can actually specify that part of the population should be collecting this type of data and other parts should be collecting other parts of data.

Another interesting thing that Funf can do is geofencing. So, it can actually only wake up and collect data within certain pre-specified geographical areas. So, for example, if you are studying students, you may only limit the collection of the data to the university campus or if you are thinking about doing something with employees, you can only limit that to the buildings of your workplace and this allows users this extra level of comfort that… it's a very clear contract where and when the data gets collected and outside it is not.

HY: In relation to Funf, geofencing refers to the ability of the scheduler to wake up the probe in order to collect data when the user is in a specified geographical location.

True.

AS: Question is, what can we actually collect with Funf? Once you download it off GitHub and start using that or developing for it, what is there off the shelf? So, Funf comes with over 40 probes that are prebuilt or built in that are ready to be used and those probes include all the important types of behavioral data such as location, Bluetooth, Wi-Fi, temperature, those physical sensors, but also virtual sensors such as your contacts list, your call and text metadata, the screen going on and off, and – very interestingly – new probes can be developed very easily. We are talking about a few hours of development rather than a few days if you want to build an entire system from scratch because again remember probes just need to wake up and collect the data and pass the data and they don't have to worry about anything else and if we want to develop the new probes, it's not just that it's easy to do that. It's not just easy to add in probes. Also the way the data gets handled upstream… it's easy. You just need to specify the schema for this data and suddenly your server will start seeing new data type appearing and it might be of a new amazing sensor that no one thought before and suddenly it is added because the entire infrastructure of collecting, storing, uploading, making sure that the configuration is proper is already there in place for you to use. Of course this is an open source library. License is LGPL and it's hosted on GitHub. You can see the address and feel free to use it but also to develop for it. We are always happy to accept contributions.

00:07:24

It's important to know that Funf is not trying to cover the entire lifetime of the data. Funf is really responsible on the phone to collect the data, to capture it, store it, and then upload it to the server and at that point it's happy. There's another project also coming from MIT Media Lab which is OpenPDS – Open Personal Data Store – and this is the backend solution that fits like a hand in a glove with Funf that can take over the data that Funf uploaded and then handle it in a privacy-responsible way for research but also for personal use.

HY: What steps are covered by Funf and OpenPDS in the data analysis process?

Funf deals with the collection, storage and uploading, while OpenPDS allows for the backend storage and analysis of the data.

AS: So, please check this project if you are interested in the full end-to-end solution of data collection and storage and analysis.

So, this is the overall architecture of Funf and what it can do for you. Think about… if there is this architecture that we are talking about, what do you think could be changed or what would you do differently or you think is missing? If you are now given the library and you're supposed to start using that, what are other common things that you think everyone struggles with but they are not there yet in Funf?