Macaco Test-bed Architecture

Tao Peng
Created on 2016-01-28
Updated on 2016-02-04

Overall Architecture

Figure 1 shows the overall architecture of Macaco Test-bed, which involves three tiers: data sensing on the phone, data collection on front servers, and data storage on back-end servers.

MacacoApp senses the context/content data on smartphones, and stores the data temporally in a database on the phone; later MacacoApp sends data to a public server in Toulouse; finally, the data are replicated on two servers in different locations. The personality survey is sent to a server in Switzerland once user compiles it.

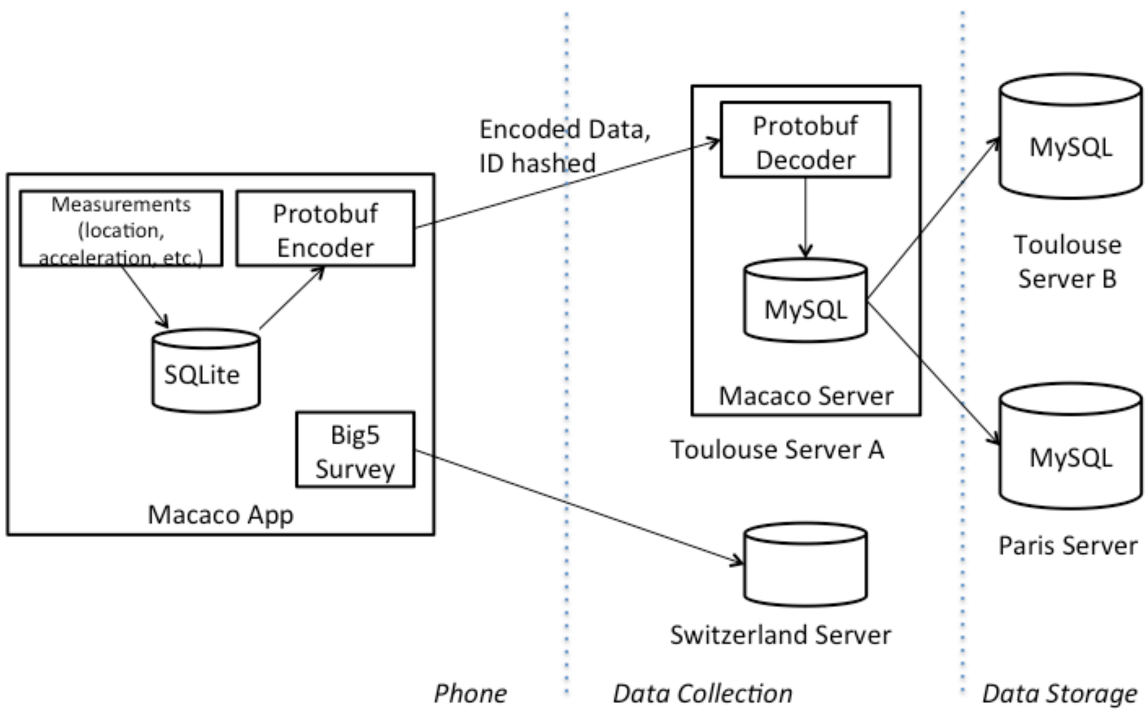

Figure 1 Overall Architecture of Macaco Test-bed

Data sensing – Macaco App

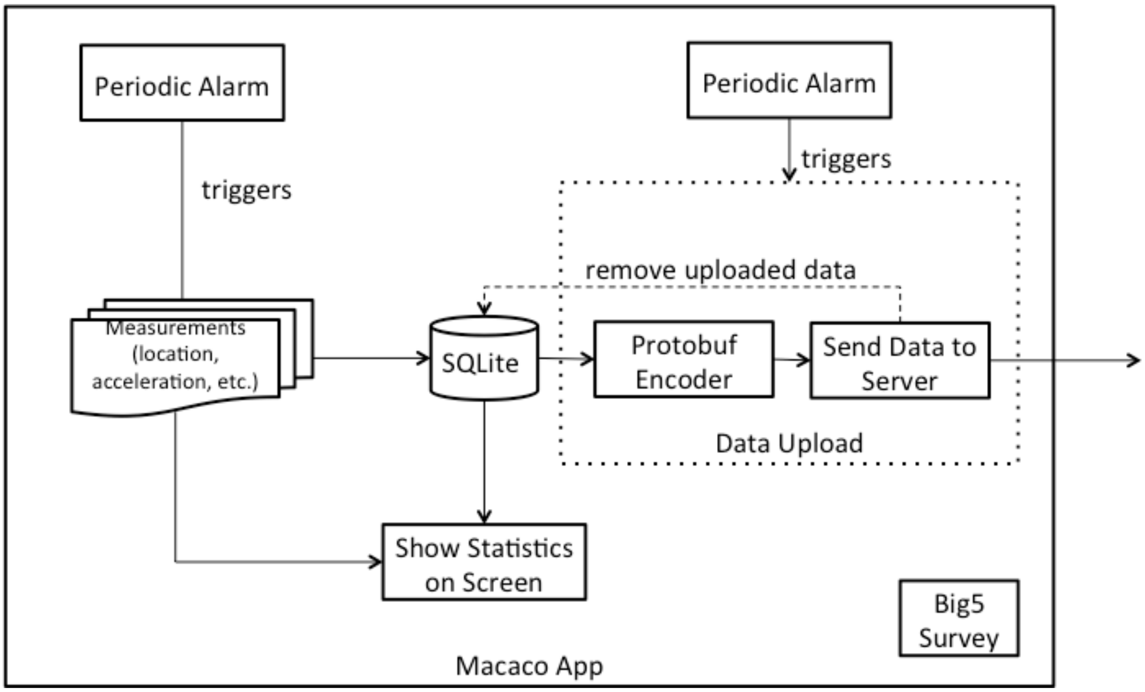

Figure 2 MacacoApp Architecture

We developed MacacoApp as an Android Application, which is installed on the users' phones. MacacoApp periodically senses context/content information (including acceleration, location, running processes, proximity, cellular networks, nearby Wi-Fi Access Points, nearby Bluetooth devices, browser history, battery status, IP address, memory status). The sensed data are first temporally stored in a SQLite database on the phone, and then uploaded to "Toulouse Server A" periodically (less frequent as data sensing). To protect user privacy, we anonymized the device identity.

When the user launches MacacoApp for the first time, he/she is prompted to fill a survey about his/her personality. The result of this survey is sent to a server in SUPSI immediately.

Figure 2 illustrates the architecture of MacacoApp. A periodic alarm triggers the context/content measurement. The sensed data are temporally stored in an SQLite database on the phone. Another periodic alarm triggers the data upload process, which first packs the data using Google Protocol Buffer, and then sends to the Toulouse Server A. After the data are successfully sent, they are removed from the SQLite database on phone. To save cellular data and battery consumption, MacacoApp only uploads data when the phone is connected to Wi-Fi.

To reduce the data volume in transmission and in storage, we squeeze the size of data fields. As an example, latitude/longitude are by default of the type "double" in

Java, we convert it to type "int" [int_value = (int) (double_value * 10^6)] while keeping reasonable precision.

MacacoApp has a screen showing some basic statistics of sensed data, including a list of Apps those consume network traffic, and the number of unsent measurements.

Macaco Server

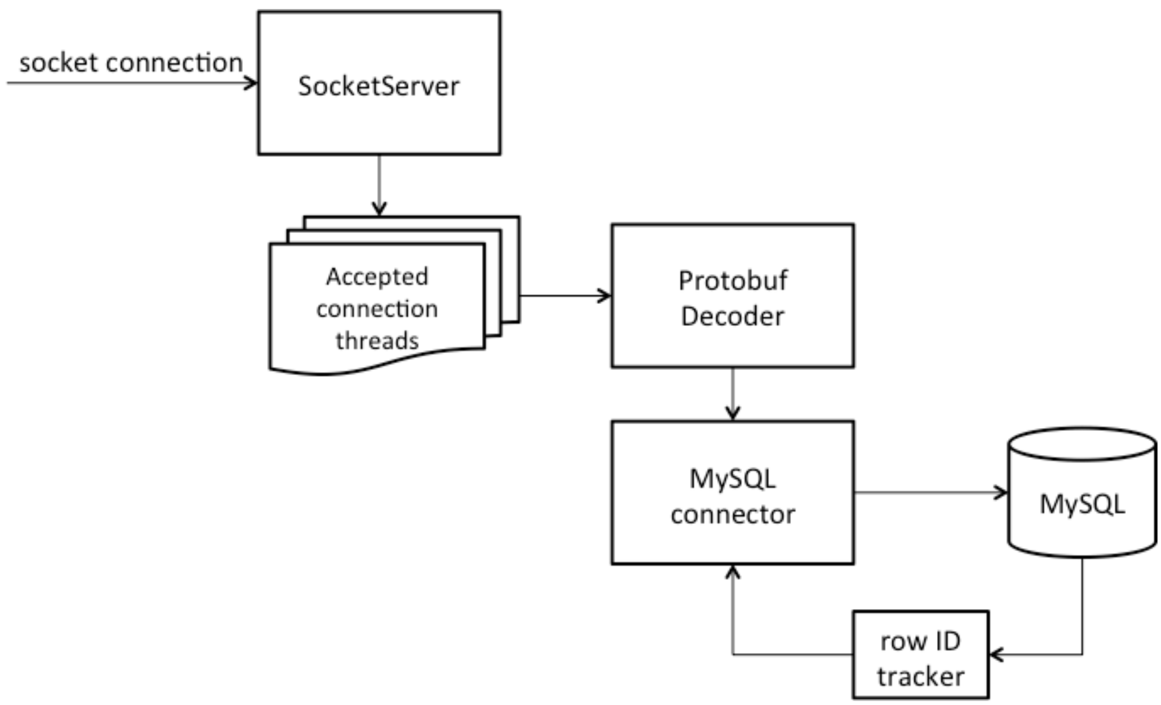

Figure 3 MacacoServer Architecture

We developed MacacoServer as a Java Server Application, which runs on a server located in Toulouse, which serves as a publically accessible front server.

Figure 3 shows the architecture of MacacoServer. MacacoServer listens on a port for incoming socket connections from MacacoApp. When a phone starts to connect, MacacoServer creates a new thread to handle the request. The thread receives the message encoded by Google Protocol Buffer. A decoder converts the message into Java objects holding measurement data. A MySQL connector stores the measurement data in to the MySQL database on Toulouse Server A.

The component "row ID tracker" tracks the max value of the primary key "id" for the tables. Because the data stored on Toulouse Server A is periodically replicated to other servers and then emptied (explained in next section), we need to ensure the later stored data do not restart counting from 1 in the primary keys ("id"), to avoid possible conflicts in successive replications.

Macaco Databases

As shown in Figure 1, the data collected are first stored on Toulouse Server A. A script is scheduled to run every 12 hours, to replicate the data stored in Toulouse Server A to both Toulouse Server B and INRIA server in Paris, and then removed the propagated data from Toulouse Server A. The purpose of data propagation is to better secure the data (Toulouse Server B and Paris Server are not publically accessible), and keep two redundant copies at two different locations.