March 9 MOBILE · DATA · BACKEND · ANDROID · NETWORKING AND TRAFFIC · OPTIMIZATION · PERFORMANCE

How we built Facebook Lite for every Android phone and network

Gautam Roy

We rolled out **Facebook Lite**, our version of Facebook for Android built for emerging markets, in June of 2015. Today we're excited to share that the app has hit 100 million monthly active users. It's the fastest-growing version of Facebook to reach 100 million users in under nine months. It has an APK that is less than 1 MB in size, meaning people can download it in seconds on slow connections. The app now supports 56 languages and is most popular in Brazil, India, Indonesia, Mexico, and the Philippines.

Why Facebook Lite

It's important to us for everyone to have a great experience using Facebook on their phone, no matter the device they're using or the connection they're on. Because of various network conditions and types of hardware, experiences can differ. Although 2G mobile networks cover up to 96% of people globally and are used for basic data connectivity by over half the world's population, **at least 1.6 billion people** still live in places where mobile broadband networks (3G and 4G) are not available, making data access difficult. Even for people on 3G networks, the intermittency and stability of the connection are often the biggest barriers to delivering a great mobile experience.

Through our research in emerging markets and in seeing how people use our apps, we know that cost of data and overall data usage is extremely important to people. So we've been working to reduce data usage for people in emerging markets when they want to access Facebook. In addition to continuing to improve how the Facebook for Android app performs on 2G networks, **we introduced** Facebook Lite in 2015 to address those constraints. Our goal when we launched was to deliver a lightweight, fast, and native Facebook experience for people using typical Android phones and network connections in emerging markets.

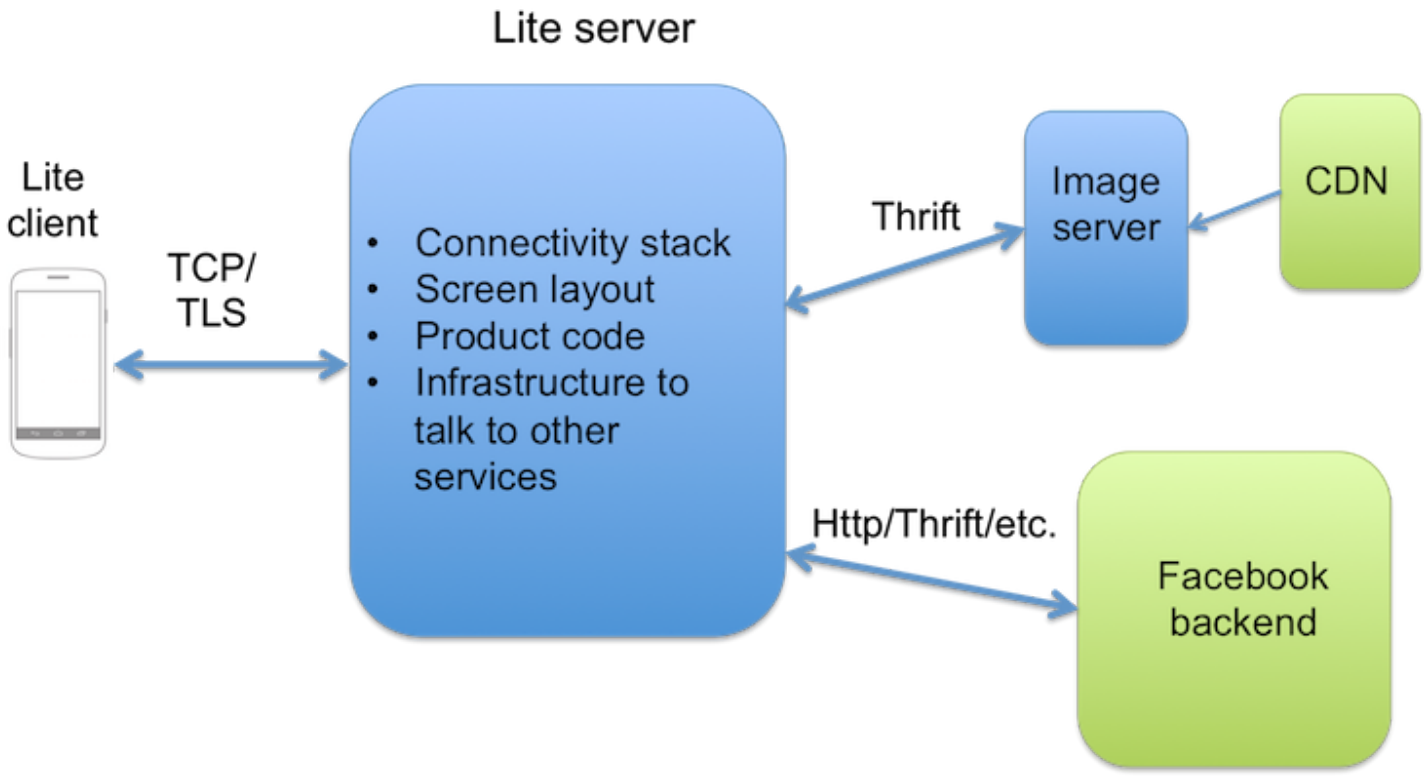
Our goals

Our task boiled down to the following goals:

1. Stay below the 1 MB APK size limit.
2. Design client-server interaction to minimize data usage and work well on 2G networks.
3. Build an app that works on Gingerbread and on a 2009 year class device.

A look at the architecture

Given these constraints, we chose a proxy server architecture with a very thin client. We used the successful **Facebook For Every Phone architecture** as the starting point and adapted it for Android.



To reach the APK size target, the Lite APK doesn't have the product code and resources found in a typical Android app. The Lite client is a simple VM that provides various capabilities to interact with the OS (such as read a file, open the camera, create an SQLite database, and so on) and a rendering engine to drive the Android UI. Product code is written on the server and is expressed in terms of the capabilities the client has. Resources are sent down from the server as needed and cached. So it has infinite scalability for building additional product without bloating the APK.

The Lite architecture is designed to let the server side do the heavy lifting, which enables the app to work well on very low-powered devices, like the LG Optimus ME. The server fetches data from the Facebook backend services and sends down screens to the client in the form of a compressed UI tree similar to a DOM, which the client then renders. As a client talks to only a single server in a session, the server can push data to the client in addition to the client requesting data.

Instead of using HTTPS, Lite uses a custom message protocol over TLS (directly over TCP). Compressed message exchange takes place over the persistent TLS connection the client establishes to the server for the duration of the session. This design opens the door for a lot of optimizations that help with reducing data usage and performing on 2G networks.

Lite has a set of image servers that talk to the CDN and other image stores to enable the Lite server to serve exact-size images to the client.

Reaching our goals

Small APK size

Downloading a typical app with 20 MB APK can take more than 30 minutes on a 2G network, and the download is likely to fail before completion, due to the flaky nature of the network. Restricting our APK size makes it easier for people to download it. This also means that people have to use less data to upgrade the app. Thus, we took significant care to minimize the app's APK size. As mentioned before, the app is designed so that the client is a generic VM and the product code is on the server. Elements that tend to bloat APKs, like translations of strings and PNG resources, are sent down from the server on demand and cached rather than built into the APK. In various places, to save data and size, we use Unicode symbols instead of image resources to represent icons.

Optimizing for slower connections and data efficiency

Lite's network stack optimizes for working on 2G networks and reducing data usage. To achieve an extremely byte-efficient wire protocol, instead of using HTTPS, Lite uses a custom message protocol over TLS (directly over TCP). One of the biggest pain points in a 2G network is that establishing a connection can be very slow; it can take multiple seconds. As most Lite traffic flows over a single connection to the backend, this pain point is mitigated in

comparison.

Taking advantage of the fact that the client talks repeatedly to the same server, dynamic shared dictionary compression is applied to messages across a session. We use LZMA2 compression for server-to-client messages due to the great compression ratio as well as the low resource usage in decompression. We use DEFLATE for client-to-server messages.

Tools like **Augmented Traffic Control** and the Internet.org **Innovation Lab** have been instrumental for our engineers to simulate 2G networks for testing in realistic conditions.

Most of the app's data usage comes from images, so this was an area where we wanted to reduce data usage in the app. We control the size of the images to be able to control the amount of data used. The Lite server knows exactly the screen width, height, and screen density of the client. Instead of talking to CDNs directly, the Lite client receives exact-size images over the same TCP connection from the Lite server. The server can adjust the JPEG quality of the images and format images exactly at the size needed, rather than use the few supported sizes in the CDN. The exact size and JPEG quality optimizations make a lot of the images small; when images are large, they are chunked for transport. The Lite image server also provides caching and transcoding of images.

The fact that the client talks to one server allows the server to anticipate and push data in certain situations. Lite has a diff mechanism, so screens that the client has already received and have only a few changes can be sent from server as a diff over the existing screen rather than the entire screen, thus saving data. For example, if a pull-to-refresh is done on News Feed and the only change is the like count of a few stories, only that difference is sent down to the client. Alternatively, if a new comment arrives for a story, the server can update the client with the new comment via the diff mechanism even without the client requesting the data.

Performance on every emerging-market Android device

We set out with a goal to work well on even very low-powered devices that can still be found in emerging markets. A 2009 **year class device** like the Samsung Galaxy Y comes with very restrictive specs of 290 MB RAM, 600 MHz processor, and 180 MB internal memory. Our data also showed that supporting Gingerbread would be of importance. The client architecture ensures minimizing CPU, storage, and RAM usage on the client. Resource-intensive tasks, like screen layout, are done by the Lite server; animations and other heavy UI interactions are

avoided. Cache sizes for images and resources are chosen to stay in the tens of MBs. And the client architecture takes measures to keep memory usage low and release memory when backgrounded.

These design choices help Facebook Lite achieve best-in-class performance metrics on interactions like login, start-up time, pull-to-refresh, and image-loading times on low-bandwidth intermittent networks.

With Facebook Lite, our goal is to provide the best possible Facebook experience to everyone, no matter their device or connection. And we hope that by sharing how we built the app, we can encourage more people to build for the next billion coming online.

More to Read

Connectivity Lab custom-designs two-axis gimbal for air-to-air and air-to-ground laser communications

Related

Building for emerging markets: The story behind 2G Tuesdays