TextNow Android - OkHTTP – A replacement for Apache HTTP Library

Links

1. http://square.github.io/okhttp/

Developers

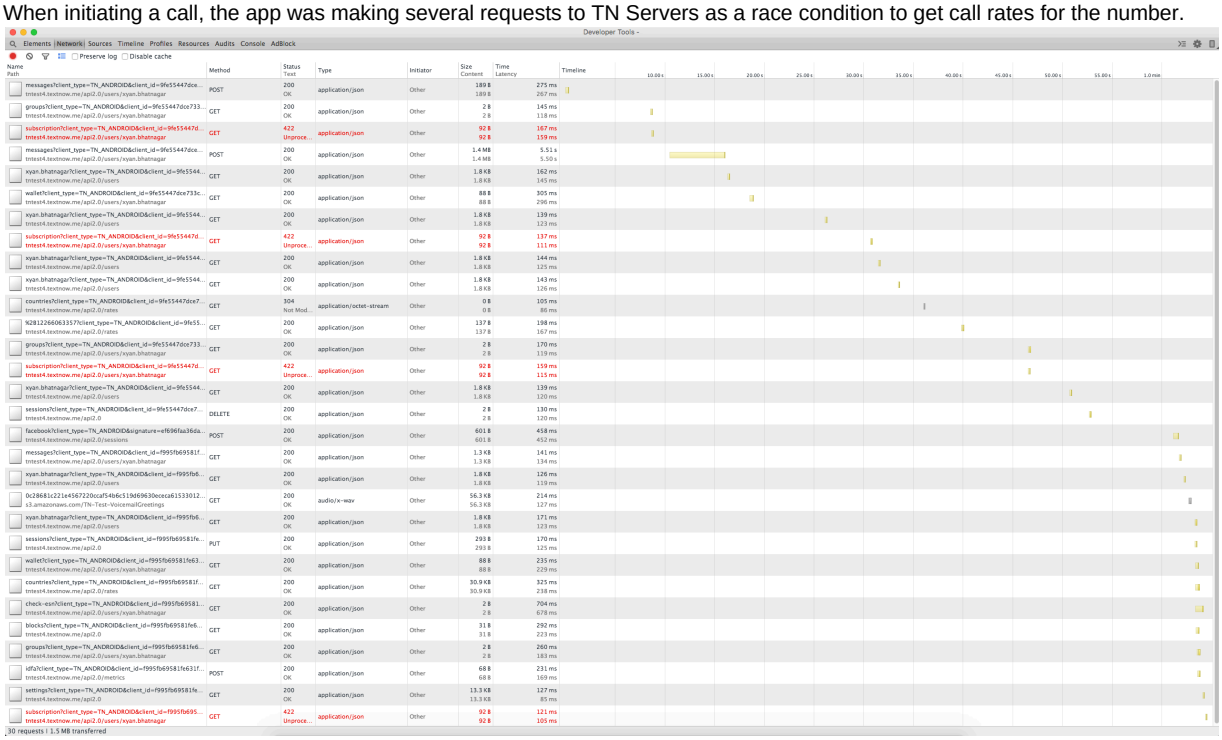
Michael Ye, Xyan Bhatnagar

Introduction

This document is to propose a switch from Apache to OkHTTP, a powerful connection library for Android.

Benefits

1. A lightweight library, especially in comparison to Apache. (~230kb jar file)
2. Completely eliminates Apache HTTP Client from our codebase.
    a. Apache's HTTP Client was deprecated since Gingerbread 2.3
    b. From KitKat 4.4 onwards, OkHTTP is the default library used in Android. https://twitter.com/JakeWharton/status/482563299511250944
    c. From Marshmallow 6.0, the Apache library was removed completely from Android. https://developer.android.com/about/versions/marshmallow/android-6.0-changes.html#behavior-apache-http-client
3. A cleaner implementation. Apache required us to use InputStreams and ByteArrayOutputStreams for basic string body responses. These streams would need to be maintained by us and closed correctly. OkHTTP can automatically convert responses into strings or generate InputStreams for data when needed.
4. Connects to Stetho (http://facebook.github.io/stetho/) which allows us to analyze our TN Server requests exclusively in great detail right from within Chrome. This means better debugging and performance analysis. In its initial testing alone, we found 3 flaws in how we communicate with TN Servers.
    a. For the Country Rates route, a 304 Not Modified response was incorrectly produced, corrupting further responses in the HTTP stream. (Fixed : REDACTED)
    b. The Data Devices route was being called on every transition in and out of MainActivity, even for Free and Premium users who always received a 400 Not Found response. (Fixed : REDACTED)
    c. When initiating a call, the app was making several requests to TN Servers as a race condition to get call rates for the number.



5. Allows us to attach interceptors to modify requests or responses before they are sent out or brought in.
6. Recovers from common network failures. Performance and battery efficient.
7. Attempts to use several address resolution techniques (useful for IPv4 + IPv6)
8. Supports HTTP/2 and modern TLS features (SNI, ALPN). Backward support for TLS 1.0
9. Offers response caching, transparent gzip compression and connection pooling.

Given all the benefits, we should expect to see more reliable and faster connections with TN Servers across most of our app.

Integration Plan

Proposed Stages

1. Code Review
    a. GitHub PR - REDACTED
    b. JIRA - REDACTED
2. Full Regression Testing
3. Dogfooding + Beta Testing
4. Controlled Release to Production

Future Goals

1. Incorporate Retrofit (http://square.github.io/retrofit/) into OkHTTP which eliminates a lot of boilerplate code behind converting HTTP responses into a Java interface.
2. <Completed> Compress redundant Java classes (AbstractHTTPCommand, AbstractCommand, AbstractBaseCommand, ICommand) into a single cleaner implementation
3. Add transaction IDs into requests using Interceptors to prevent retry requests from resulting in duplicacy.
4. <Added Picasso> Analyze which libraries connect well with OkHTTP and can be used advantageously - https://github.com/square/okhttp/wiki/Works-with-OkHttp