# TextNow Android - Optimizing Network Requests with Sync Adapters

## Developers

Xyan Bhatnagar

## Links

Google's Video Explanation: https://www.youtube.com/watch?v=nDHeuEM30ks&list=PLWz5rJ2EKKc9CBxr3BVjPTPoDPLdPIFCE&index=12

Android Training Guide: https://developer.android.com/training/sync-adapters/index.html

## What's happening now?

Currently, TextNow on Android does a lot of network transfer with TN Servers, especially when it is started from a cold start (Force closed and then opened).

Our app in the worst case scenario runs 10 **network tasks simultaneously** every time the app starts.

These tasks are –

1. GetUserInfoTask
2. GetWalletTask
3. GetRatesForCountriesTask
4. CheckESNTask
5. GetSettingsTask
6. ReportIDFATask
7. GetBlockedContactsTask
8. GetNewMessagesTask
9. GetFeatureTogglesTask
10. GetGroupsTask

These were found in TextNowApp and MainActivity's onCreate methods. It's possible that there are more which have been left out.

## What's going wrong?

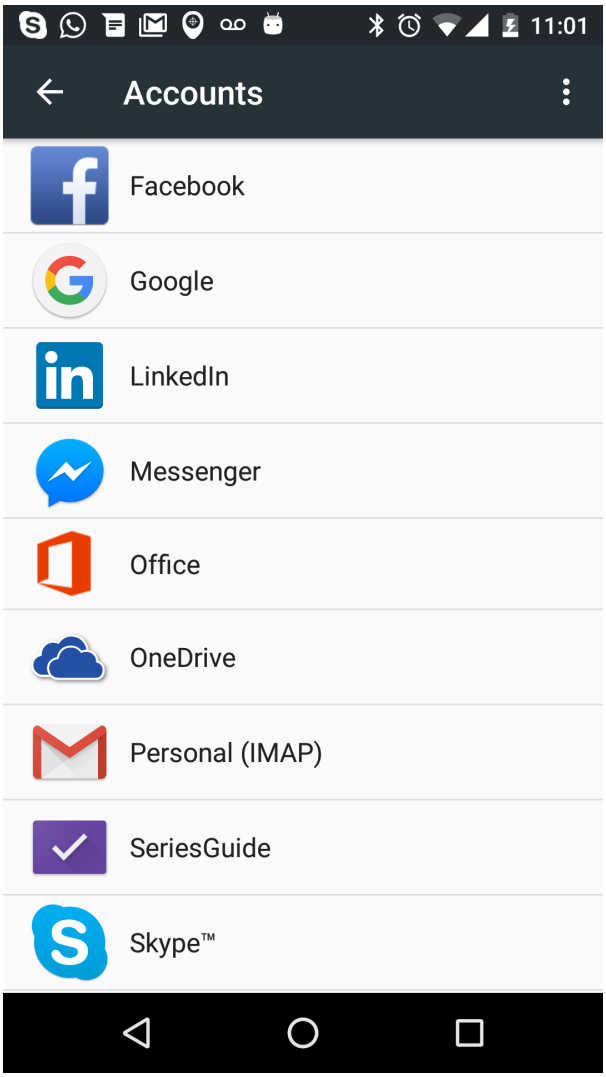
Running these network tasks right before we need information is a **bad idea**.

1. **Several of these tasks are not updated every second**. Things like Settings, Country Rates, Feature Toggles do not change so frequently. Requesting their information right before displaying the UI does not make sense.
2. It has a bad impact to **performance** and **user experience** because we must **force the user to wait** every time we need to get new information. This is very noticeable on our **app's start**, where the user is delayed by several seconds on slower devices in poor network conditions.
3. Our current syncing solutions are not smart. Our network tasks do not take **network availability** into account. There is no point in requesting information when there is no network, but currently we do it anyway!
4. We create custom polling services like TNPullService, where the client periodically polls the server for information. Our polling services use partial wakelocks and keep the system alive. These services can be terminated by Android which means our app may become out of date until the next time it starts.

## What's the solution?

A good solution to this is something that Android has **implemented since Gingerbread**. It's called Sync Adapters. Sync Adapters are currently used by Facebook, WhatsApp, Google Products, Outlook, Skype, LinkedIn and others. Sync Adapters are lightweight plugins to the Android Sync Framework. They allow us to run our data sync tasks **without the app needing to be active**.

Sync Adapters show up in the Accounts section of the Android Settings app. We control what we show to users in the Accounts section of the Android Settings page.




With Sync Adapters, we gain the following advantages.

1. Our app becomes faster and smoother. That is an important step towards better **retention** and **customer satisfaction**!
2. We stop making network requests unnecessarily inside the app and in some cases, we start **refreshing data periodically** rather than just-in-time.
3. We don't need to implement our own services to pull information. Android will do it for us in a more efficient manner!
4. Android will ensure that our Sync Adapters are kept alive and run at the right times, **even if the app is closed**.
5. The code for getting the data is still ours!
6. Sync Adapters are **network aware and user aware**. They will not be triggered if there is no network connection. They can also trigger contextually (for e.g - when the user wakes up).
7. Android triggers Sync Adapters in batches so that the network connection is kept alive for the shortest time possible.
8. Sync Adapters are compatible with Android Marshmallow's Doze functionality which **improves battery life**.
9. If we want settings to be **updated immediately** for our users, we can send a push notification (via GCM) to all devices which will trigger the sync adapter, causing all settings to be refreshed.
10. Our app can query the status of the sync adapter.
11. We can implement several Sync Adapters, each with their own refresh rates. **Exponential back-off** can also be implemented.

Sync Adapters can be triggered in several ways

1. When the server has new data (Via GCM Push Notifications)
2. When the client has new data
3. Over customizable intervals
4. Manually, inside the app

## What is the difficulty involved?

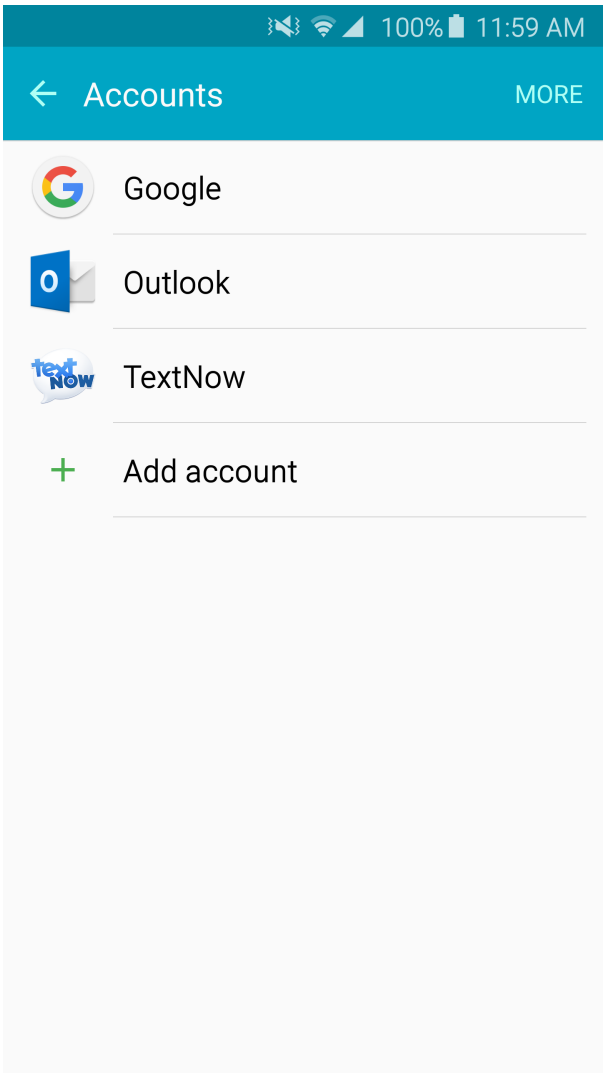
Sync Adapters are easy to implement in our app. They can also **run side-by-side with our current implementation and can be feature toggled**, allowing us to phase out our old practices. This is because of the way we have coded our application with TNTasks. Since **Sync Adapters get application context**, all we have to do is trigger the above TNTasks inside the Sync Adapter.

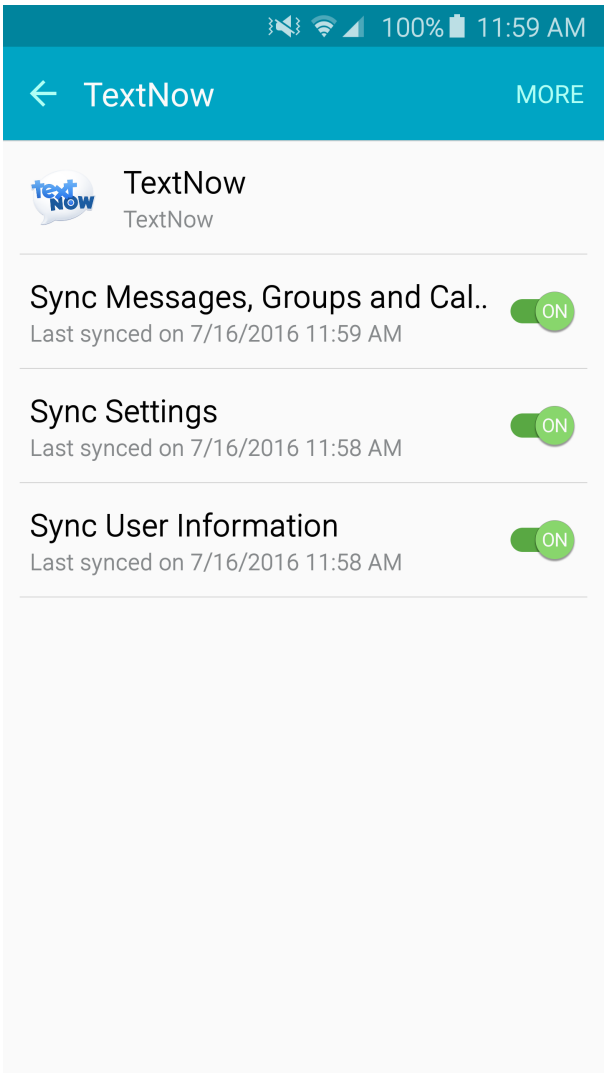
However, before implementing this change, we may need to conduct an investigation into the current syncing patterns for our app to see which network tasks qualify for periodic syncing, immediate syncing or both.

**Implementing Sync Adapters may qualify as a short time, large impact change.**

## Do you have a test implementation?

Yes!




Refresh Rates are set as follows

1. Settings, Feature Toggles, Rates, IDFA - Every hour
2. User Info, Wallet, Subscription, Device Data - Every 30 minutes
3. Messages, Groups - Every 5 minutes

These adapters also refresh every time the app is started.

Sync Adapter requests can be logged by filtering for "TNSyncAdapter" through Logcat.

**Note** - This test build was made in just a day - proof that it is an easy change to make.

Download the test build from here - REDACTED

Check the GitHub branch - REDACTED