Sync Adapters, Demystified

Ben Oberkfell | @benlikestocode

Ben Oberkfell

- Twitter: @benlikestocode
- Blog: http://benlikestoco.de
- Opinions and views expressed in this talk are my own and do not represent those of my employer

I fight for the users

— *Tron*

Three Guiding Principles

Phones Are Used "In the Now."

Ben Oberkfell | @benlikestocode

Cell Service Can Flake Out.

Ben Oberkfell | @benlikestocode

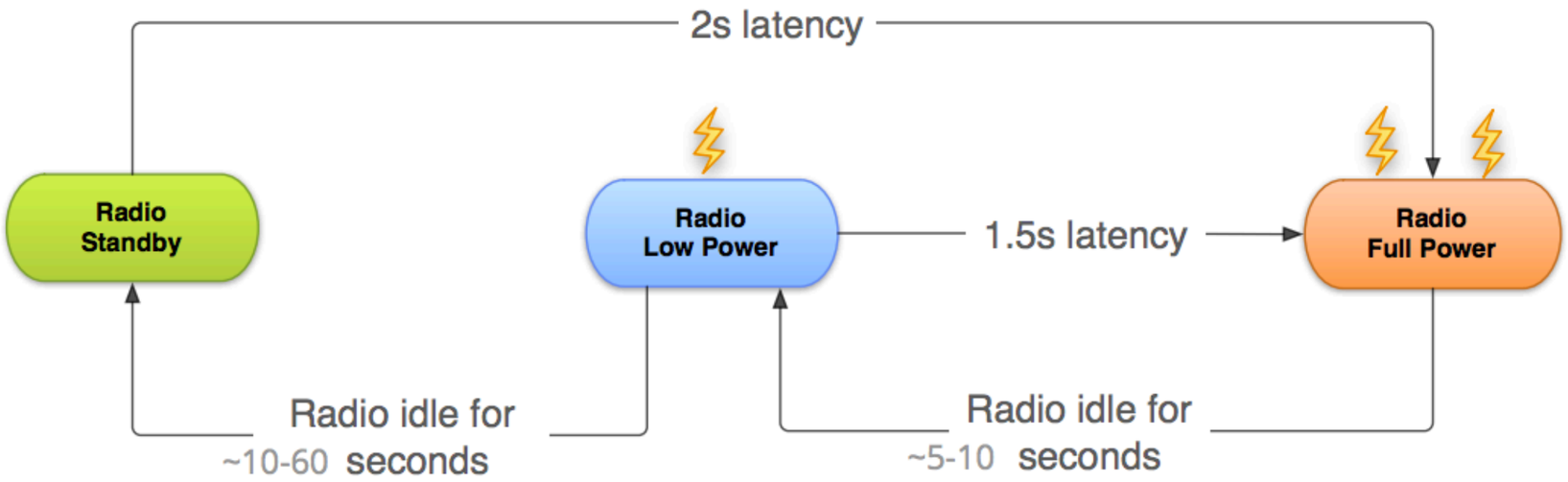
Battery Is Precious.

Ben Oberkfell | @benlikestocode

How can we be nice to users?

- Let's keep data synced to stay up-to-date
- Let's work offline, and re-sync on network availability
- Let's not burn down the whole battery while doing that

A word about battery life



Reto Meier, "Making Good Apps Great" I/O 2012

How can we do all this?

Android gives you a wonderful framework for just this...

The Sync Adapter!

Run your sync logic in the background

- On demand 🔁
- On a schedule 🕒
- When data changes ✏
- When network availability changes 📵

You get your very own thread

So don't forget:

Make your sync logic run synchronously

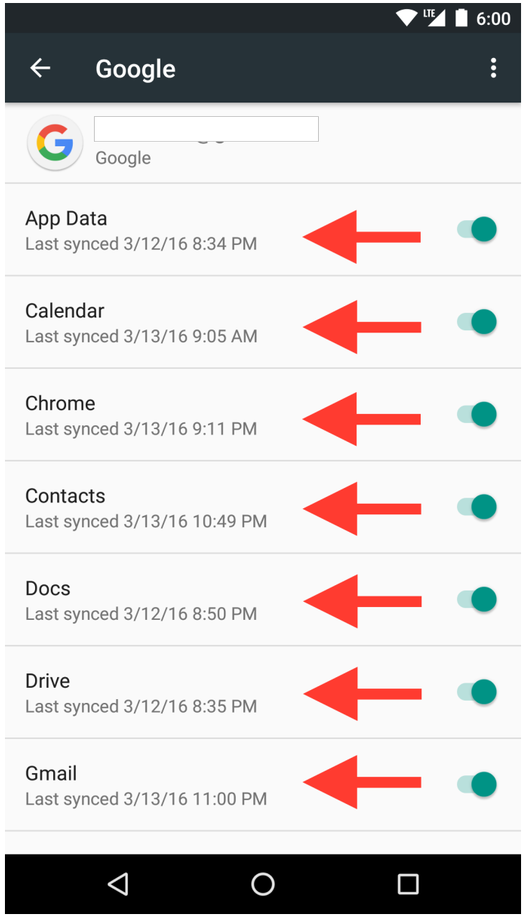
Android handles failures & backoff

Android handles interleaving with network activity

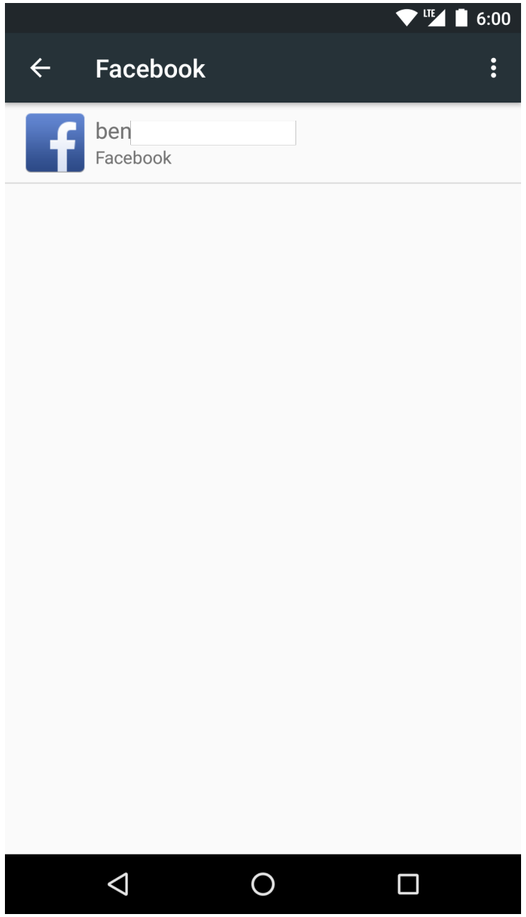
(Remember that battery diagram?)

It's easily self contained and testable

Background sync can be user controlled

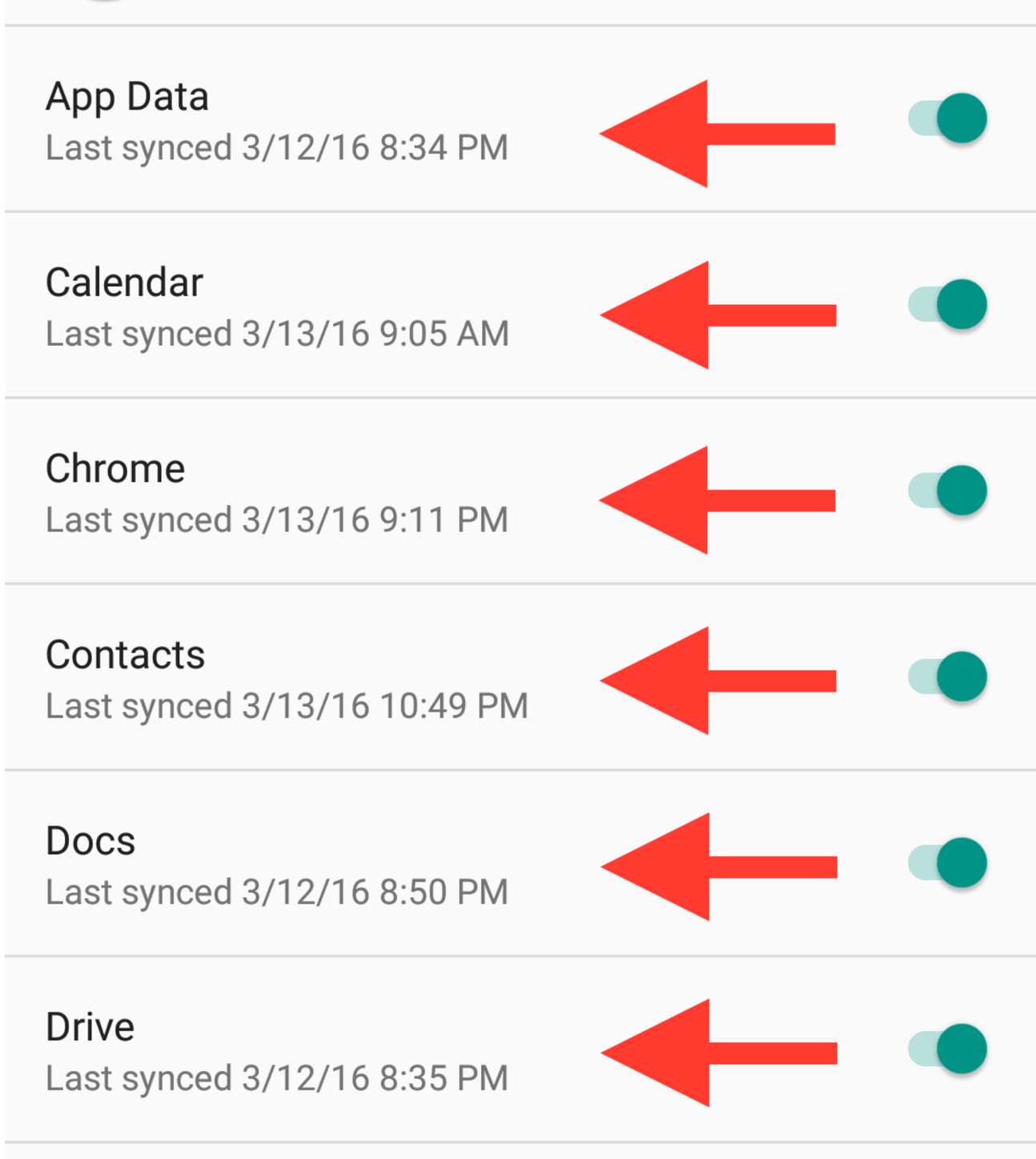


Or not.



Under the hood

A `SyncAdapter` subclass syncs data for a given `Account` and `ContentProvider`



Under the hood

A content provider is defined by an "authority" string, which should be globally unique.

Best practice: use a reverse domain name string

e.g. `com.example.syncdemo.content.tweets`

Wait just a sec...

I have to implement all this? 😱

NO!

Stubbing is totally OK.

- For your `AuthenticatorService/AccountAuthenticator`
- And your `ContentProvider`
- Unless you need the benefits, just stub it.

Sync Logic

- Define a SyncAdapter subclass
- Implement `onPerformSync` -- runs synchronously!!!
- Declare it in the manifest

```
public class DemoSyncAdapter extends AbstractThreadedSyncAdapter {

    // ...

    @Override
    public void onPerformSync(Account account,
                              Bundle extras,
                              String authority,
                              ContentProviderClient provider,
                              SyncResult syncResult) {

            MessagesSyncer syncer = new MessagesSyncer(Injector.provideRealm(getContext()),
            prefsHelper, messageService);
            MessageSyncResult result = syncer.performSync();

            // Apply the potential soft errors from our result to the sync
            syncResult.stats.numIoExceptions += result.numIoExceptions;

            // report the number of values inserted
            // if we had an error, but made progress, it will immediately
            // retry
            syncResult.stats.numInserts += result.numInserts;
        }
}
```

```java
public MessageSyncResult performSync() {
        long recentSyncTime = prefsHelper.getRecentSyncTime();

        MessageSyncResult syncResult = new MessageSyncResult();

        // We need to get our Realm on the thread that onPerformSync happens in.
        try {
            int valuesAdded = 0;

            MessageListResponse response = messageService.recentMessages(recentSyncTime).execute().body();
            realm.beginTransaction();
            for (Message m : response.getMessages()) {
                SyncMessage savedMessage = new SyncMessage();
                savedMessage.setMessage(m.getMessage());
                savedMessage.setUserName(m.getUserName());
                savedMessage.setUserAvatarUrl(m.getAvatarUrl());
                savedMessage.setTime(m.getTime());
                realm.copyToRealm(savedMessage);
                valuesAdded++;
            }
            realm.commitTransaction();

            prefsHelper.setRecentSyncTime(response.getTimestamp());
            syncResult.numInserts = valuesAdded;
        } catch (IOException e) {
            syncResult.numIoExceptions++;
        } finally {
            realm.close();
        }

        return syncResult;
    }
```

The SyncResult

- Report hard and soft error conditions
- `delayUntil` to force a delay in running again
- report progress to inform potential future retries

Sync Logic Thoughts

- Use a helper, to make it easy to unit test
- Send up pending data, pull down fresh data
- Use a different sync adapter for different kinds of data

In the Manifest, define the SyncAdapter

```
<service
    android:name=".sync.DemoSyncService"
    android:exported="true">
        <intent-filter>
            <action android:name="android.content.SyncAdapter"/>
        </intent-filter>
        <meta-data android:name="android.content.SyncAdapter"
            android:resource="@xml/syncadapter" />
</service>
```

Metadata for the SyncAdapter

```
<sync-adapter xmlns:android="http://schemas.android.com/apk/res/android"

    android:contentAuthority="com.example.syncdemo.provider"
    android:accountType="com.example.syncdemo"
    android:userVisible="false"
    android:supportsUploading="false"
    android:allowParallelSyncs="false"
    android:isAlwaysSyncable="true"
    />
```

Requesting Syncs

OK, great. We've defined the sync logic. But we want to actually run it.

Requesting a Periodic Sync

```
public void requestPeriodicSyncWithMinuteFrequency(Account account, String contentAuthority, int frequency) {
    ContentResolver.setSyncAutomatically(account, contentAuthority, true);
    ContentResolver.addPeriodicSync(account, contentAuthority, Bundle.EMPTY, frequency * 60);
}
```

Requesting a Sync On Demand

```
public void requestImmediateSync(Account account, String contentAuthority) {
    Bundle syncSettingsBundle = new Bundle();
    syncSettingsBundle.putBoolean(ContentResolver.SYNC_EXTRAS_MANUAL, true);
    syncSettingsBundle.putBoolean(ContentResolver.SYNC_EXTRAS_EXPEDITED, true);

    ContentResolver.requestSync(account, contentAuthority, syncSettingsBundle);
}
```

Stubbing the Authenticator & ContentProvider

In the Manifest, define the AuthenticatorService

```
<service
    android:exported="false"
    android:name=".sync.DemoAuthenticatorService">
        <intent-filter>
            <action android:name="android.accounts.AccountAuthenticator"/>
        </intent-filter>
    <meta-data
        android:name="android.accounts.AccountAuthenticator"
        android:resource="@xml/authenticator" />
</service>
```

Specify the name/icon for the account type

```
<account-authenticator
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:accountType="com.example.syncdemo"
    android:icon="@mipmap/ic_launcher"
    android:smallIcon="@mipmap/ic_launcher"
    android:label="@string/app_name"/>
```

Stub out the code

```
public class DemoAuthenticatorService extends Service {

    private DemoAuthenticator authenticator;

    @Override
    public void onCreate() {
        // Create a new authenticator object
        authenticator = new DemoAuthenticator(this);
    }

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return authenticator.getIBinder();
    }
}
```

Stub the ContentProvider

In the Manifest:

```
<provider
    android:name=".sync.DemoContentProvider"
    android:authorities="com.example.syncdemo.provider"
    android:exported="false"
    android:syncable="true"/>
```

Stub the ContentProvider

```
public class DemoContentProvider extends ContentProvider {
    @Nullable
    @Override
    public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder) {
        return null;
    }

    @Nullable
    @Override
    public Uri insert(Uri uri, ContentValues values) {
        return null;
    }

    @Override
    public int delete(Uri uri, String selection, String[] selectionArgs) {
        return 0;
    }

    @Override
    public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs) {
        return 0;
    }

    //....
}
```

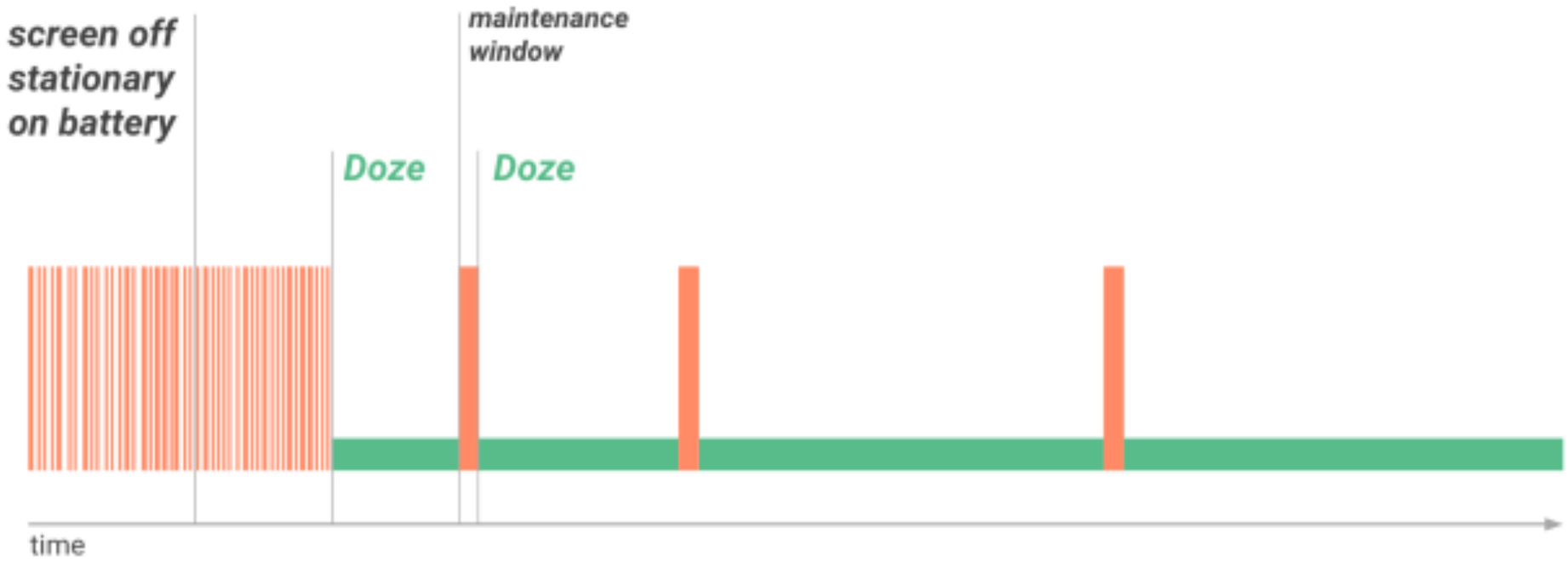
Maybe I Don't Want to Stub?

Do you want to

* Share content among other apps
* Share account credentials with other apps

Maybe the Account & ContentProvider is right for you after all!

Doze



Your syncs will only happen in the maintenance windows

Doze

- Need an immediate sync? Use a high priority GCM message to trigger an on-demand sync.

Sample Code

http://bit.ly/sync-adapters-demystified

Q&A

image credits

- Smartphone in hand: https://flic.kr/p/i7SdQF CC-BY 2.0
- BART station: https://flic.kr/p/gBPGR CC BY-NC 2.0
- Dead Battery: https://flic.kr/p/4MMsxZ CC BY-NC-ND 2.0
- Battery life diagram: Reto Meier, "Making Good Apps Great", Google I/O 2012