Лекция 4

https://github.com/IFMO-Android-2016/lesson4

Мобильный интернет

Медленно, дорого, с разрывами

Особенности мобильного интернета…

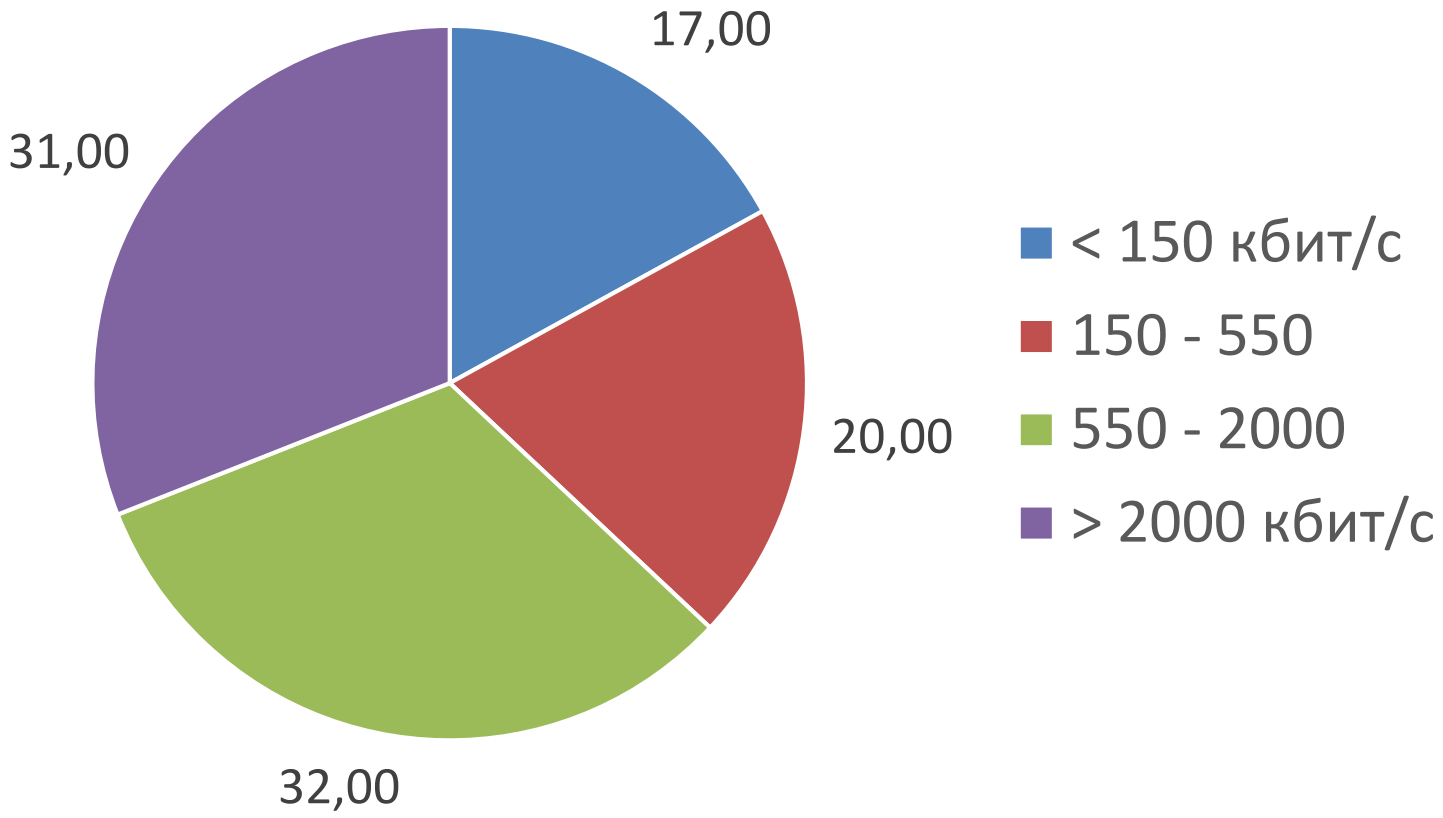
- Низкая доступность – иногда соединения нет вообще, и это норма
- Низкие скорости – в крупных городах с 4G скорости еще нормальные, но чем дальше в лес…
- Большие задержки – даже при высокой скорости (bandwith) много времени уходит просто на ожидание ответа (latency)
- Платный трафик, лимит трафика (особенно в странах СНГ)

И что с ними делать?

- Низкая доступность – при разработке всегда учитывать сценари отсутствия соединения
- Низкая скорость – настраивать таймауты, кэшировать тяжелый контент
- Большие задержки – переиспользовать соединение, объединять запросы, HTTP/2
- Платный трафик – проверять тип соединения, не злоупотреблять

Мобильный интернет в России

- Распределение пользователей Android приложения Одноклассники по скорости интернета по данным за 4 октября 2016 г.



Когда сервис не рассчитан на медленный интернет

Android / ANDROID-7516

Аплоад фоток для заметки падает с SocketTimeoutException

Edit | Comment | Assign | More | Reopen

Я столкнулся с этой проблемой в удаленной местности, когда пытался запостить заметку с фотками. При этом использовалось подключение к интернету через wi-fi, который в свою очередь был подключен к мобильному интернету местного оператора. Сколько я ни пытался загрузить заметку, каждый раз аплоад прекращался с одной и той же ошибкой. При этом другие приложения и браузер хоть и медленно, но работали. Я посмотрел в логи – там SocketTimeoutException. Потом посмотрел трафик через прокси и увидел, что фактически запрос POST, который аплоадит фотку, выполняется нормально, и даже приходит успешный ответ от сервера, подтверждающий успешную загрузку фотки, однако наш Http Client все равно бросает SocketTimeoutException. После многочисленных попыток я установил закономерность – SocketTimeoutException бросается только если запрос выполняется дольше 30 секунд.

Дальнейшие исследования:
В офисе в сетке MR_ATC я воспроизвел проблему, записал логи и трафик (tcpdump). По логам видно, что SocketTimeoutException бросается через 30 секунд после окончания чтения загружаемого файла (это не фактическая отправка, а судя по всему, копирование файла в буфер для отправки):

15.01

Updat

26.09

Resol

26.09

Develo

Create

Drag a

Выполнение сетевых запросов

HttpURLConnection – основной класс в Android SDK для выполнения HTTP запросов

Немного истории

- Google выпустил первую версию Android с двумя разными HTTP клиентами в Android SDK:
  - **Apache HTTP Client 3** (сырой API, но работает)
  - **HttpURLConnection** (официально признан негодным из-за багов)
- В версии Android 2.3 Google исправил баги в HttpUrlConnection и перестал поддерживать Apache HTTP Client в составе Android SDK

Немного истории

- Для борьбы с фрагментацией Square выпустила альтернативный клиент **OkHttp**: https://github.com/square/okhttp -- который работал лучше
- Многие разработчики использовали новую версию **Apache HTTP Client 4**, который тоже был неплох (требуется переупаковка кода при помощи JarJar, чтобы не было коллизий с “родным” Apache HTTP Client 3)

Конец истории

- Начиная с версии Android 4.4 OkHttp используется в качестве внутренней реализации HttpURLConnection

Наши дни

- **HttpURLConnection** – подходит для большинства приложений
- OkHttp – если нужен полный контроль (например, тонко настраиваемые таймауты), либо вы поддерживаете версии Android до 2.3 (зачем?!!!)
- Apache Http Client 4 – только если «пришел» с какой-то Java библиотекой

HttpURLConnection

https://developer.android.com/reference/java/net/HttpURLConnection.html

Четыре фазы выполнения запроса:

1. Подготовить запрос: составить URL с параметрами, добавить заголовки, указать метод
2. Отправить запрос: передать тело запроса POST
3. Получить ответ: статус, тело ответа
4. Закрыть HttpURLConnection

Подготовка запроса

- «Собираем» URL при помощи Uri.Builder

```
// https://base.url.com/method/path=param?query=value

String url = Uri.parse("https://base.url.com").buildUpon()
        .appendPath("method")
        .appendEncodedPath("path=param")
        .appendQueryParameter("query", "value")
        .toString();

HttpURLConnection connection =
        (HttpURLConnection) new URL(url).openConnection();
```

Подготовка запроса

- Добавляем заголовки

```
connection.addRequestProperty(
        "User-Agent",
        "Some non-standard User Agent");

connection.addRequestProperty(
        "Custom-Header",
        "Header value");
```

Подготовка запроса

Указываем метод запроса:

- GET – обычный запрос без тела, используется по-умолчанию
- POST – запрос с телом (аплоад)

```
connection.setDoOutput(true);
```

- Другие методы (HEAD, PUT и др)

```
connection.setRequestMethod("HEAD");
```

Отправка запроса

Фактическая отправка запроса происходит:

- Когда явно вызван метод `connect()`
- Когда вызван один из методов передачи или получения даных: `getResponseXXX()`, `getInputStream()`, `getOutputStream()` и др.

Отправка запроса

Опциональная отправка данных в методе POST (на примере отправки данных из файла):

```
InputStream fileIn = new BufferedInputStream(
                             new FileInputStream(file));
OutputStream out = new BufferedOutputStream(
                             connection.getOutputStream();

byte[] buffer = new byte[8192];
int readBytes;

while ((readBytes = fileIn.read(buffer)) >= 0) {
    out.write(buffer, 0, readBytes);
}

fileIn.close();
out.close;
```

Получение ответа

```
connection.getResponseCode()
```

- 200 – HTTP OK, варианты 20X
- 30X – редирект, при выполнении запроса автоматически выполняется до 5 редиректов
- 40X – ошибка запроса (404 – документ не найден)
- 50X – ошибка сервера

Если не вызвать этот метод, то в случае ошибочного кода будет брошен `IOException`

Получение ответа

Чтение тела ответа в строку:

```
ByteArrayOutputStream baos = new ByteArrayOutputStream();
InputStream in = new BufferedInputStream(
                                        connection.getInputStream());
byte[] buffer = new byte[8192];
int readSize;

while ((readSize = in.read(buffer)) >= 0) {
    baos.write(buffer, 0, readSize);
}

in.close();

String content = baos.toString("UTF-8"); out.close;
```

Permissions

- Объявляются в `AndroidManifest.xml`

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="ru.ifmo.android_2016.lesson_4">
    <uses-sdk android:targetSdkVersion="24"/>

    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name=
                    "android.permission.ACCESS_NETWORK_STATE"/>

    <application ...>
        <!-- Объявления компонентов приложения -->
    </application>
</manifest>
```

- Если не задекларировать пермиссии – приложение упадет с `SecurityException`

JSON

JavaScript Object Notation

JSON

- Наиболее часто используемый формат передачи данных в API интернет сервисов
- Более компактный и более читаемый по сравнению с XML
- Поддерживается в Android:
  - `JSONObject` – DOM parser
  - `JsonReader` – Pull parser

JSON

Типичный JSON:

```
{
  "firstName": "Евгений",
  "lastName": "Лукашин",
  "isMarried" : false,
  "children" : null,
  "address": {
     "streetAddress": "3-я ул. Строителей, д. 25",
     "city": "Москва",
     "postalCode" : 123456
   },
   "phoneNumbers": [ "095 123-1234", "812 123-4567" ]
}
```

DOM парсер JSONObjet

```
// например, прочитали из ответа на запрос
String content = ...

JSONObject json = new JSONObject(content);

// бросает JSONException, если поле не найдено
String name = json.getString("name");

// Возвращает дефолтное значение, если поле не найдено
int age = json.optInt("age", 0);

JSONArray array = json.getJSONArray("pnoneNumbers");
for (int i = 0; i < array.length(); i++) {
    String phoneNumber = array.optString(i);
}
```

DOM парсер JSONObjet

https://developer.android.com/reference/org/json/JSONObject.html

- Простой API позволяет быстро накидать парсер для прототипа
- Требует загрузить весь ответ в память, даже если нужно извлечь одно поле – плохо сказывается на производительности.
- **Не рекомендуется** в production коде

Pull парсер JsonReader

```
InputStream in = ...
JsonReader reader = new JsonReader(in, "UTF-8");

reader.beginObject();
while (reader.hasNext()) {
    final String name = reader.nextName();
    if (name == null) {
        reader.skipValue();
        continue;
    }
    switch (name) {
        case "id": id = reader.nextString(); break;
        case "title": title = reader.nextString(); break;
        case "image": imageUrl = parseImage(reader); break;
        default: reader.skipValue(); break;
    }
}
reader.endObject();
```

Pull парсер JsonReader

https://developer.android.com/reference/android/util/JsonReader.html

- Более сложный API, приходится писать больше кода
- Потоковая обработка – не нужно загружать весь ответ в память. Делает меньше аллокаций, более быстрый
- **Рекомендуется** в production коде