SCIENCES SORBONNE UNIVERSITÉ

Networks

Renaultᵃ@lrde.epita.fr

Goals

- **Nowadays, most applications require an access to the internet**
  - to advertise
  - to update data
  - to publish on social network
  - …

- **Android provides frameworks for that**
  - Classical HTTP clients
  - Connectivity Manager
  - Volley Framework
  - …

Network Connection

Modify AndroidManifest.xml

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


Choose an HTTP Client

- HttpURLConnection:
  - **Most used component**
  - **Compress Cache**
  - **Post-Froyo applications**
- HttpClient:
  - **Before Froyo and Eclair**
  - **Less configurable**

Test Connectivity

ConnectivityManager

- Check all possible sources (GPS, Wifi, ...)
- Send Intents when connectivity changes

```
ConnectivityManager connMgr = (ConnectivityManager)
                getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();
if (networkInfo != null && networkInfo.isConnected()) {
    // Network available ...
} else {
    // Network not available ...
}
```

- isConnected:
  - **check if a connection exists**
- getActiveNetworkInfo:
  - **informations about existing networks**

How to download a Webpage? (1/2)

Use asynchronous task

```
private class DownloadWebpageTask
        extends AsyncTask<String, Void, String> {
    @Override
    protected String doInBackground(String... urls) {
        // params comes from the execute() call:
        // params[0] is the url.
        try {
            return downloadUrl(urls[0]);
        } catch (IOException e) {
            return "Unable to retrieve URL (maybe invalid).";
        }
    }

    // onPostExecute displays the results of the AsyncTask.
    @Override
    protected void onPostExecute(String result) {
    }
```

How to download a Webpage? (2/2)


```java
private String downloadUrl(String myurl) throws IOException {
    InputStream is = null;
    // Only display the first 500 characters of the retrieved web page content.
    int len = 500;
    try {
        URL url = new URL(myurl);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setReadTimeout(10000 /* milliseconds */);
        conn.setConnectTimeout(15000 /* milliseconds */);
        conn.setRequestMethod("GET");
        conn.setDoInput(true);
        // Starts the query
        conn.connect();
        int response = conn.getResponseCode();
        is = conn.getInputStream();
        // Convert the InputStream into a string
        String contentAsString = readIt(is, len);
        return contentAsString;

    // Makes sure that the InputStream is closed after the app is
    // finished using it.
    } finally {
        if (is != null)
            is.close();
    }
}
```

Volley Framework

Google project dedicated to Android

- Automatic request scheduling
- Simultaneous communications
- Caches
- Request's priority
- Easy-configuration

Adapted for applications

- with a massive use of RPC
- using massively structured datas

Ready-to-use component

- in graddle: 'compile com.mcxiaoke.volley:library:1.0.6'

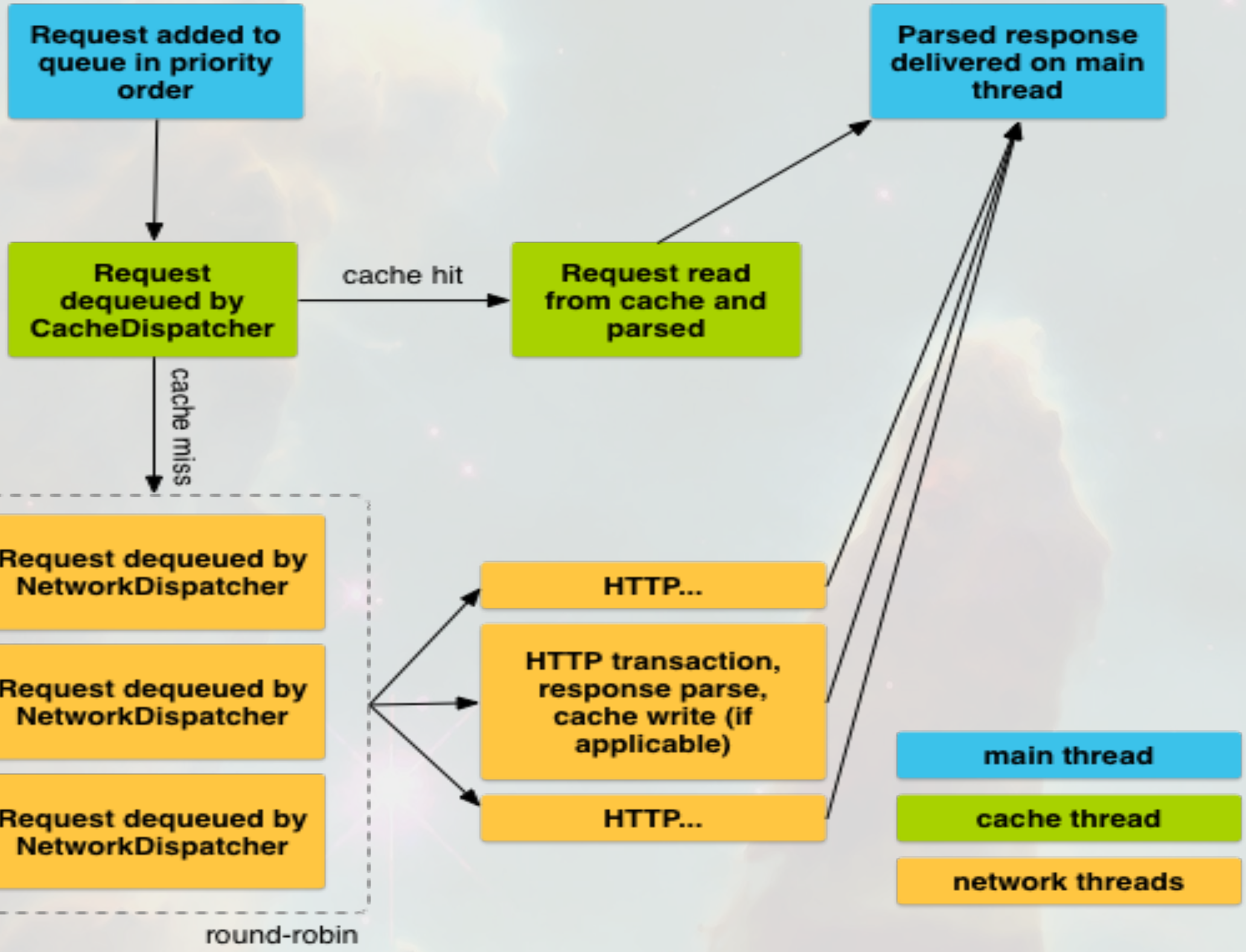
Send/Receive Request


```
RequestQueue queue = Volley.newRequestQueue(this);
String url ="http://www.google.com";

// Request a string response from the provided URL.
StringRequest stringRequest =
    new StringRequest(Request.Method.GET, url,
                      new Response.Listener() {
        @Override
         public void onResponse(Object response) {
            // the response without using AsyncTask!
        }
       }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            //      Error!
        }
    });

// Add the request to the RequestQueue.
queue.add(stringRequest);
```

Request Lifecycle



Summary

- **Many ways to connect to network**
  - Classical HTTP clients
  - Modern Frameworks

- **Do not have long computing in the UI Thread**
  - Otherwise the application will crash

- **Think to check connectivity before sending a request**

- **Manage connectivity all around you application**