Android networking problem areas:

Thread pool management

(and many others)

Origins of Volley:

Internally developed to support Google play refactoring

Subsequently released to general public

Features

Memory and disk caching

JSON, image and text management

Android networking Model

Main thread: ui thread, should never perform blocking operations (ui gets stuck)

Solution: spawn background thread

Volley class architecture

1. Request queue
   1. Manages request objects
2. Request objects
   1. Post/get
   2. Error/success handlers

Before Volley

Use background thread + http url stack

If version <2.3 then it is advisable to use the Apache Stack

Advantages of Volley

Automatic stack management

Ease of migration

Requests related to destroyed view are deleted automatically

Requests are executed in parallel

Response is accessible from ui thread

Supported Android versions ALL OF THEM ☺

Downsides of Volley

No default caching system available

Unsuitable for massive upload/download operations, use internal Android services (what services?)

No push notification support(use gcm, parse($)?)

Volley tutorial APP architecture

1. Main class
   1. Image loader
   2. Request queue
2. Base activity (view?)
   1. Login
   2. Wall activity <-> wall adapter (binds ui svc data)

Service architecture

1. Login (response format json)
2. Get wall (ditto)
3. Post wall (ditto)

Initialization

Queue inner workings

1. Thread check memory then disk
2. Miss? - > signal to dedicated thread that calls ws
3. (ask about caching system)

<http://goo.gl/x2Wbw1>

response listener

stringrequest instead of jsonrequest since not all servers reply using json

adapterview problems:

cannot cancel requests

Wall app architecture

1. Activity

Goo.gl/z40GQJ

Set maximum input length on textbox widget and save yourself a headache