Assignment 9

Automating Recovery after Failure

# Introduction

Nowadays, networking technology became a part of people’s life. With network, people can read latest news like sports, weather and national news. Friends can chat online and they can share photo. However, networking on a mobile device is not stable and reliable. WWAN (wireless wide area network) is worse than WIFI. When people in the elevator or behind large buildings, they will lose signal. If the network is under a low bandwidth, it may fail due to timeout.

To improve user experience on networking operation, application should automatically detect the network status. It will re-try, when networking is available or on WIFI rather than 3G. Saving cache can help save networking resource and battery.

# Failure Recovery

In the mobile device, the low bandwidth and poor signal may cause failure on networking operation.

There are 2 solutions to detect the network status, one is notification service, and another one is NSTimer class to perform the action by schedule. Second solution may need more CPU resource. To observe a notification is better, because performed action will occur only by network status is changed.

## Download failure (client side failure)

Usually, user spends more download resource than upload resource. Twitter application use REST http API to communication between client and server. The API supports the GET, POST, DELETE send user’s request. When request is timeout, the application should retry later when the network status is available. If the network is broken during the data transfer, application should save all data source from Internet to the persistence storage as cache.

When download action started, application only need to download the rest of data. It saves the network resource and system resource. If application fails download data at all, it should retry in the background with threads, while user is reading tweets offline.

## Upload failure

YouTube is a most popular video website in the world. People may upload their life video to YouTube. Most websites do not support the resume uploading. To improve the better user experience, application should able to record the description of a failed download. When network status is available, it should read all information from disk storage and display on the screen. Then restart again.

## Retry rules

To resume interpreted download, application should consider following rules.

* If the network operation fails, application could retry immediately, but it can’t retry indefinitely, Application should have some maximum retry count, then abort. If network transfer fails, it can retry with limited times. If it still fails, it will only try when networking is available or user bring app from background to foreground.
* To distinguish errors those are worth retry. Like account/password errors and DNS errors is not worth to retry. The timeout errors are most worth to retry.
* Avoid duplicate downloads, when retry failure.

## Gaming and video stream (UDP)

With UDP protocol client can receive the data stream from server. It is faster than TCP. UDP is used to gaming and video application. User may receive an uncompleted data source. Some packets may be lost. If a gaming or a video sharing application failed when try to access data source on the Internet. It should retry immediately. If a game cannot get the opponent statistic, it will stop. With a large number of retry, if it is still failed, application should inform user “internet is broken”.

## Determine the network

Most mobile devices are support WIFI and WWAN. WWAN is little slowly than WIFI.

If applications try to transfer large file, it should send via WIFI. However if there is no available WIFI near user, the data should be compress.

# Drawbacks

In the old design, I used NSTimer to check the network status every 5 seconds. If the network is reachable, application will try to download data again. This is a bad design. An NSTimer object will run forever during application life cycle. It wastes lots of resource. To improve this implementation, I used notification to check network states. The notification will only happen, when the network states is changed.

Another problem is I used main thread to download data source from the Internet. A serious problem is the application freeze. User cannot control the interface. To fix this problem, I used

[request startAsynchronsou];

to request http data source. This request will obtain the data via background thread.

The App should only retry when a timeout error occurred. To implement with this method:

[request setNumberOfTimesToRetryOnTimeOut:2];

# Conclusion

In conclusion, following steps can improve user experience on a mobile device.

* Limited retry networking request
* Save downloaded data with cache, avoid duplicated request
* Observe the network status by notification
* If request user’s timeline by twitter API, cannot request too much and too less data
* Finally reloadData to refresh UITableView

# References

Apple’s Reachability Sample Project, Apple, accessed on 8th November 2011, <[http://developer.apple.com/](http://developer.apple.com/library/ios/#samplecode/Reachability/Introduction/Intro.html)library[/ios/#](http://developer.apple.com/library/ios/#samplecode/Reachability/Introduction/Intro.html)samplecode[/Reachability/Introduction/Intro.html](http://developer.apple.com/library/ios/#samplecode/Reachability/Introduction/Intro.html)>

Copsey, B 2011, *ASIHttpRequest documentation*, All-Seeing Interactive, accessed on 7th November 2011, <[http://](http://allseeing-i.com/ASIHTTPRequest/)allseeing-i[.com/ASIHTTPRequest/](http://allseeing-i.com/ASIHTTPRequest/)>

*Effective iOS Network Programming Techniques,* SliderShare, 2011,Accessed on 9th November 2011, <http://www.slideshare.net/subdigital/effective-ios-network-programming-techniques>

# Appendix

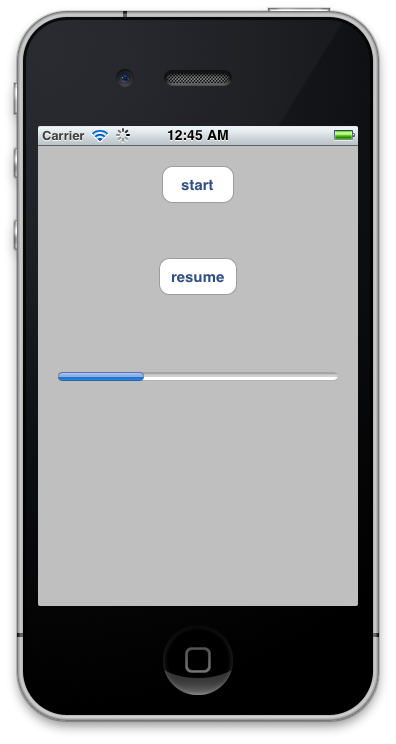


Figure 1 - Screen Shot while downloading