Instruction of assignment 2 implementation

Shaoqing Yu UPI: syu702

1 Introduction

This is an instruction for the implementation of 711 assignment 2. The project has been compiled and tested on computers in science laboratory.

2 Compiling

The project is written in C#, which requires .Net framework 4.0 or above. The ideal compiler is visual studio 2015, which is the original tool to develop this project. The project has also been successfully compiled on computers in laboratory with visual studio 2013. The only modification you need to do is to choose the first option, which means that changing the compiling version to VS 12, when the dialog pop up after open the solution file. The detail compiling process is like this:

1. Download the A2.zip, unzip CDN folder to any location you can access on your computer. (Don't put it in C:\ since your access might be forbidden).
2. Go into CDN folder, open CDN.sln with visual studio. (If any dialog popping up to require a version change, choose agree to change, which normally is the first option).
3. There are totally 4 projects in this solution. Besides CDNCommon is a dll (dynamic linked library) shared by other projects, other 3 are projects which can produce executable files. Since the dependency has been well configured, what you need to do is to right-click the solution and choose "rebuild solution".
4. If no exception, the build will success. Then you can find client, cache, server in these place respectively:
   Client: CDN\CDNClient\bin\Debug (or Release)\ CDNClient.exe
   Cache: CDN\CDNCache\bin\Debug (or Release)\ CDNCache.exe
   Server: CDN\CDNServer\bin\Debug (or Release)\ CDNServer.exe

3 Running

1. The files on server should be stored at CDN\CDNServer\bin\Debug (Release)\server. If the folder does not exist, create one before running.
2. The segments on caches will be stored at CDN\CDNCache\bin\Debug (Release)\cache. If the folder does not exist, create one before running.
3. The files downloaded from server will be sotored at CDN\CDNClient\bin\Debug (Release)\client. If the folder does not exist, create one before running.
4. Double click CDNServer.exe, CDNCache.exe, CDNClient.exe to run them one by one.

5. Fill target IP address on client based on cache IP; fill target IP on cache based on server IP.
6. Click “refresh” button on client to request file list on server; double clink files in file list can download and update download list; double click download list will display the content of file.
7. Click “clear” button on client will clear all the downloaded files; click “clear” button on cache will clear all the cached segments.

4 Implementation

A typical file request sending by client will trigger a set of communication between client, cache and server like Figure 1.

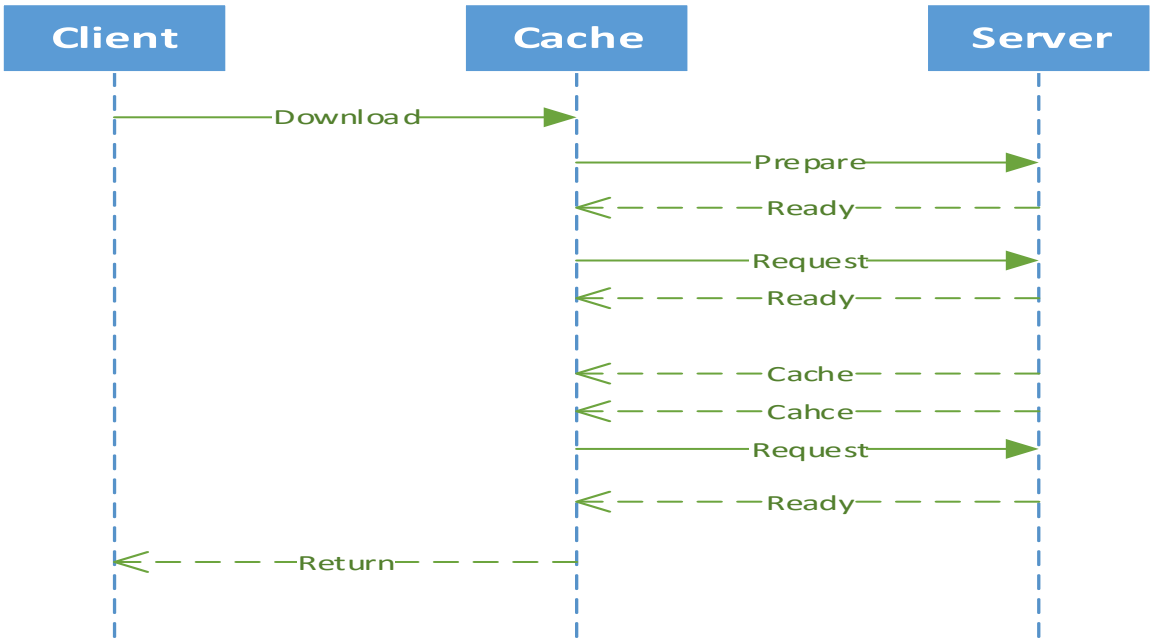

Figure 1 A typical downloading process

a) Once the client request to download a file, it will send a “download” message to cache.
b) When first time the cache receive the “download” message, it will send a “prepare” message to server, informing that “I want this file, please tell me the segments I need”.
c) When server receive “prepare” message from cache, it will index the file and divide it into segments by Rabin function and store all the segments in a list. Each of these segments will have a unique name generated by MD5 algorithm. To follow the file name convention on Windows, prefix “fragment” and suffix “.part” are added to each figment name.

d) After prepare the segments list, the server will send this list back to cache in message “ready”, telling that “to construct the file you request, you need segments in this list”.
e) When the cache get the segment list form server, it will search into its own directory to figure out which segments I have had and which segments I need. Then send a “request” message to server to request the segments which is not cached.
f) When the server receive a “request” message from cache, it will send the segments which are requested to cache, meanwhile send another “ready” message to cache, tell it that “I have sent some segments to you, please check again and request what you still need”.
g) When cache receive a “cache” message, it will retrieve the segment and cache it.
h) Since client, cache and server interact asynchronously, the step e, f, g may run several times to make sure that cache have all the segments to construct the file which is requested by client.
i) Once all the segments have been stored on cache, it will construct the file according to segment list, sending it back to client with message “return”.

5 Results

All the requirements in assignment 2 have been meet in this implementation. The result has been tested on computers in science laboratory.

6 Conclusion

The main technique used in this implementation is Rabin function and MD5. The Rabin function make sure that each segment will have the unique common content. The MD5 algorithm guarantee that the segment which have the same content will have the same name. Therefore, cache and server can exchange the information about which segments have been cached and which segments are requested by compare the segment name with those in segment list.