南京大学本科生实验报告

课程名称：**计算机网络**　　任课教师：田臣/李文中　　助教：

学院	计算机科学与技术系	专业	计算机科学与技术
学号	185220001	姓名	磯田智明
Email	185220001@smail.nju.edu.cn	开始/完成日期	2021.06.05/2021.06.07

1. 实验名称

Content Delivery Network

2. 实验目的

- 设计CDN中的DNS功能
- 实现CDN中缓存服务器功能

3. 实验内容

Task 1: Preparation

配置实验环境

Task 2: DNS server

实现CDN中的DNS功能

Task 3: Caching server

实现CDN中缓存服务器功能

4. 实验结果

Task 2: DNS server

The features of DNS server

Step 1 : Load DNS Records Table

读入 `dns_table.txt` 即可

Step 2 : Reply Clients' DNS Request

读入的 `dns_table.txt` 要对以下几个部分做处理

- 首先是第一部分的 `Domain Name` 要对通配符 `*` 做处理，对传入的域名首先要与带有通配符的网站作正则匹配，如果能精准匹配到其他网站要以精确匹配到的网战为标准。
- 然后第二部分的 `Record Type` 作处理，如果该值为 `CNAME` 则直接返回第三项的 `Record Values` 即可；如果该值为 `A` ，则寻找 `Record Values` 距离最近的一个ip地址并返回。

Task 3: Caching server

The features of Caching server

Step1: HTTPRequestHandler

通过 `touchItem` 方法得到要发送内容，如果没有目标文件则发送 `HTTPStatus.NOT_FOUND` ；如果有要发送的内容，则通过内置的 `send_header` 函数将 `touchItem` 得到的内容一一发出

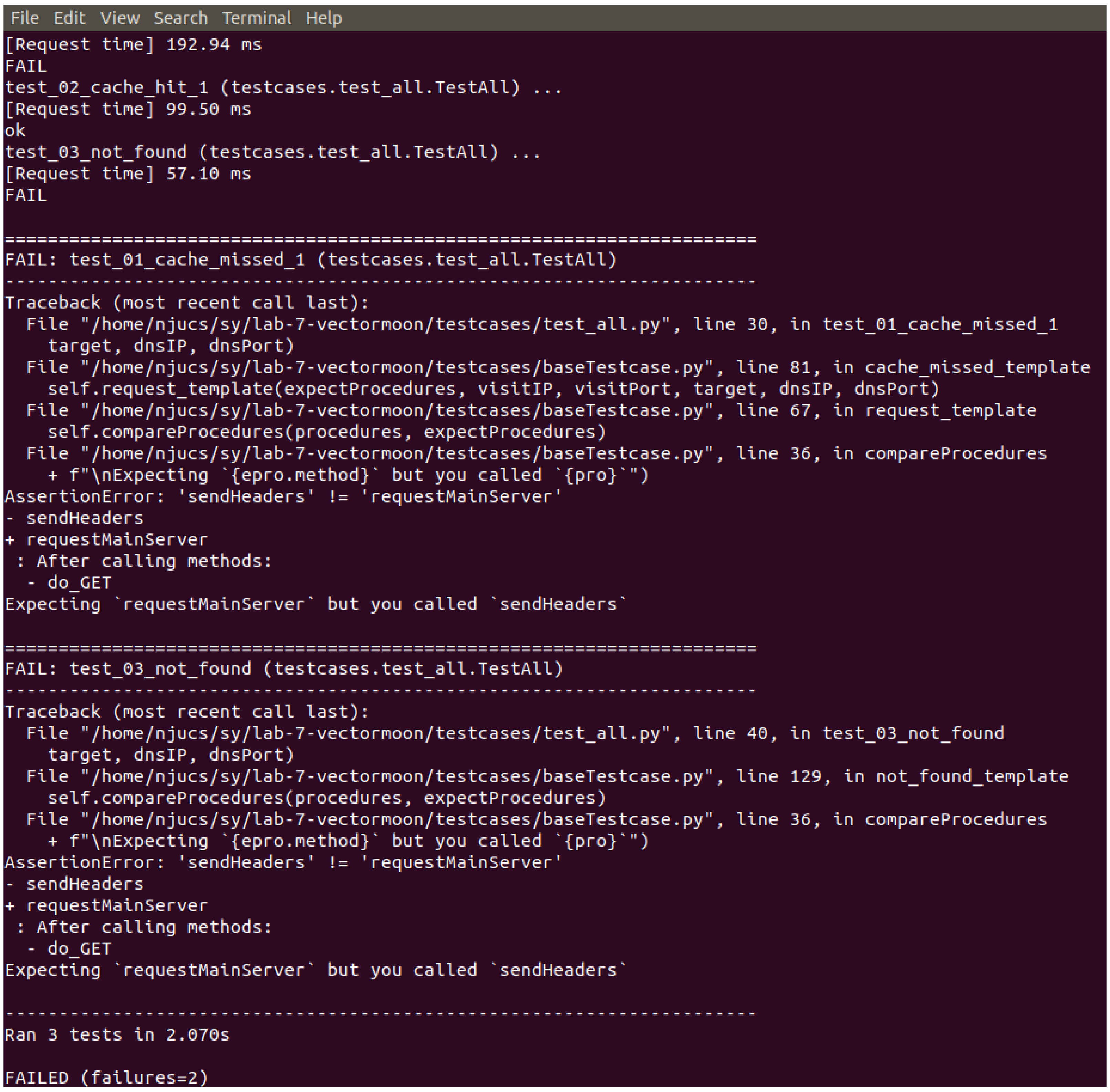
Step2: Caching Server

通过 `touchItem` 先查询 `cacheTable` 中是否存在要查询内容，如果不存在则通过调用 `requestMainServer` 的方法并将得到的内容存入到 `cacheTable` 中

Task4: Deployment

Test all

出错的原因是在 `do_GET` 中调用 `sendHeaders` ，再由 `sendHeaders` 调用 `touchItem` 在执行其他内容，导致和测试程序函数调用顺序有出入



```
[Request time] 192.94 ms
FAIL
test_02_cache_hit_1 (testcases.test_all.TestAll) ...
[Request time] 99.50 ms
ok
test_03_not_found (testcases.test_all.TestAll) ...
[Request time] 57.10 ms
FAIL

======================================================================
FAIL: test_01_cache_missed_1 (testcases.test_all.TestAll)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/njucs/sy/lab-7-vectormoon/testcases/test_all.py", line 30, in test_01_cache_missed_1
    target, dnsIP, dnsPort)
  File "/home/njucs/sy/lab-7-vectormoon/testcases/baseTestcase.py", line 81, in cache_missed_template
    self.request_template(expectProcedures, visitIP, visitPort, target, dnsIP, dnsPort)
  File "/home/njucs/sy/lab-7-vectormoon/testcases/baseTestcase.py", line 67, in request_template
    self.compareProcedures(procedures, expectProcedures)
  File "/home/njucs/sy/lab-7-vectormoon/testcases/baseTestcase.py", line 36, in compareProcedures
    + f"\nExpecting `{epro.method}` but you called `{pro}`")
AssertionError: 'sendHeaders' != 'requestMainServer'
- sendHeaders
+ requestMainServer
 : After calling methods:
  - do_GET
Expecting `requestMainServer` but you called `sendHeaders`

======================================================================
FAIL: test_03_not_found (testcases.test_all.TestAll)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/njucs/sy/lab-7-vectormoon/testcases/test_all.py", line 40, in test_03_not_found
    target, dnsIP, dnsPort)
  File "/home/njucs/sy/lab-7-vectormoon/testcases/baseTestcase.py", line 129, in not_found_template
    self.compareProcedures(procedures, expectProcedures)
  File "/home/njucs/sy/lab-7-vectormoon/testcases/baseTestcase.py", line 36, in compareProcedures
    + f"\nExpecting `{epro.method}` but you called `{pro}`")
AssertionError: 'sendHeaders' != 'requestMainServer'
- sendHeaders
+ requestMainServer
 : After calling methods:
  - do_GET
Expecting `requestMainServer` but you called `sendHeaders`

----------------------------------------------------------------------
Ran 3 tests in 2.070s

FAILED (failures=2)
```

5. 核心代码

Task 2: DNS server

```python
if (regex_flag):
    for key in self.table.keys():
        if "*" in key:
            key = key[1:]
            if (re.search(key, request_domain_name) is not None):
                request_domain_name = "*" + key
if (request_domain_name in self.table):
    if (self.table[request_domain_name][1] == "CNAME"):
        response_type = "CNAME"
        response_val = self.table[request_domain_name][2]
    elif (self.table[request_domain_name][1] == "A"):
        response_type = "A"
        table_len = len(self.table[request_domain_name])
        min_distant = 0x3f3f3f3f
        if (table_len == 3):
            response_val = self.table[request_domain_name][2]
        else:
            if (IP_Utils.getIpLocation(client_ip) is None):
                random_load_flag = random.randint(2, table_len - 1)
                response_val = self.table[request_domain_name]
[random_load_flag]
            else:
                client_ip_location = IP_Utils.getIpLocation(client_ip)
                i = 2
                min_distant_flag = 0
                while (i < table_len):
                    server_ip_location =
IP_Utils.getIpLocation(self.table[request_domain_name][i])
                    if (server_ip_location is not None):
                        res = self.calc_distance(server_ip_location,
client_ip_location)
                        if (res < min_distant):
                            min_distant = res
                            min_distant_flag = i
                        i += 1
                    if min_distant_flag == 0:
                        response_val = None
                    else:
                        response_val = self.table[request_domain_name]
[min_distant_flag]
```

Task 3: Caching server

```
def sendHeaders(self):
    ''' Send HTTP headers to client'''
    # TODO: implement the logic of sending headers
    headers, body = self.server.touchItem(self.path)
    if (headers is None and body is None):
        self.send_response(HTTPStatus.NOT_FOUND)
        self.end_headers()
        return None
    else:
        self.send_response(HTTPStatus.OK)
        for header in headers:
            self.send_header(header[0], header[1])
        self.end_headers()
        return body

def touchItem(self, path: str):
    ''' Touch the item of path.
        This method, called by HttpHandler, serves as a bridge of server and
        handler.
        If the target doesn't exsit or expires, fetch from main server.
        Write the headers to local cache and return the body.
        '''
    # TODO: implement the logic described in doc-string
    ct = CacheTable()
    if (ct.getHeaders(path) is not None):
        headers = ct.getHeaders(path)
        body = ct.getBody(path)
        return headers, body
    elif (ct.expired(path) or ct.getHeaders(path) is None):
        response = self.requestMainServer(path)
        if (response is None):
            return None, None
        else:
            headers = response.getheaders()
            body = response.read()
            ct.setHeaders(path, headers)
            ct.appendBody(path, body)
            return headers, body
```