实验四、应用层协议分析实验报告

组号： 5-6

姓名： 施炎江 学号： 2186113847 班级： 计算机 82

姓名： 高浩翔 学号： 2181411962 班级： 计算机 82

一、 实验目的

分析应用层协议（如 FTP，HTTP）的工作过程，理解应用层与传输层及下层协议的关系。

二、 实验内容

（1）每组同学利用现有实验室网络及云服务器搭建内网、外网环境；

（2）用 Wireshark 截获 HTTP 报文，分析报文结构及浏览器和服务器的交互过程；分析 HTTP 协议的缓存机制。分析应用层协议跟 TCP/DNS 等协议的交互关系。

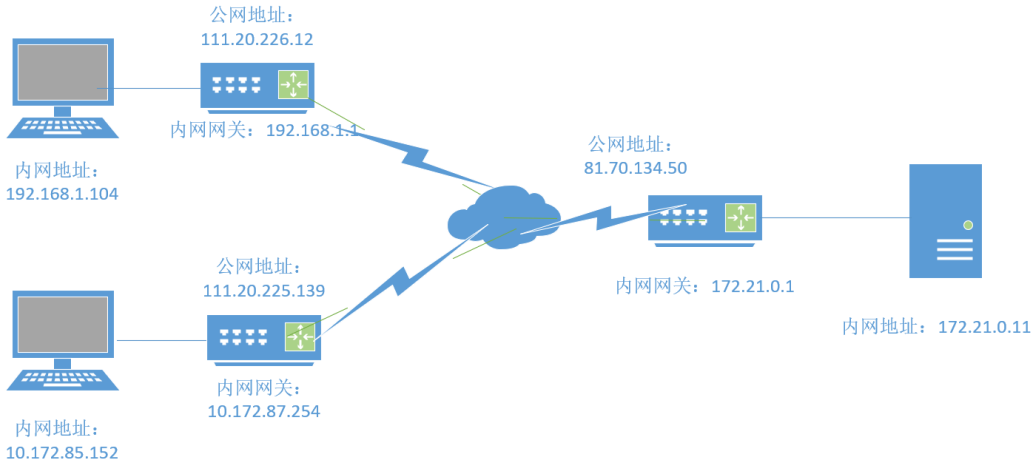
（3）用 Wireshark 截获 FTP 的报文，分析 FTP 协议的连接；分析被动模式，普通模式的区别；分析 NAT 对 FTP 的影响。使用 netcat 工具模拟 FTP 的客户端。

三、 实验环境与分组

每 2 名同学一组，以现有校园网络环境及云服务器搭建内网、外网网络。

四、 实验组网

以各组现有网络实际情况为准，标注内网、公网地址。



五、 实验过程及结果分析

1、HTTP 协议分析

（一）清空缓存后的 ARP，DNS 和 HTTP 协议分析

步骤 1：在计算机终端上运行 Wireshark 截获所有的报文。

步骤 2：清空 ARP，DNS 和 HTTP 浏览器的缓存：

浏览器缓存的清除以 Chrome 浏览器为例，地址栏中输入 chrome://settings/，找到高级选项中的“隐私设置和安全性”，清除浏览数据。

执行“ipconfig /flushdns”清除本地 DNS 缓存。

执行“arp –d”命令清空 arp 缓存。

步骤 3：在浏览器中访问 3 个网址，比如 www.xjtu.edu.cn,www.unb.br, dean.xjtu.edu.cn;

步骤 4：执行完之后，Wireshark 停止报文截获，分析截获的报文。

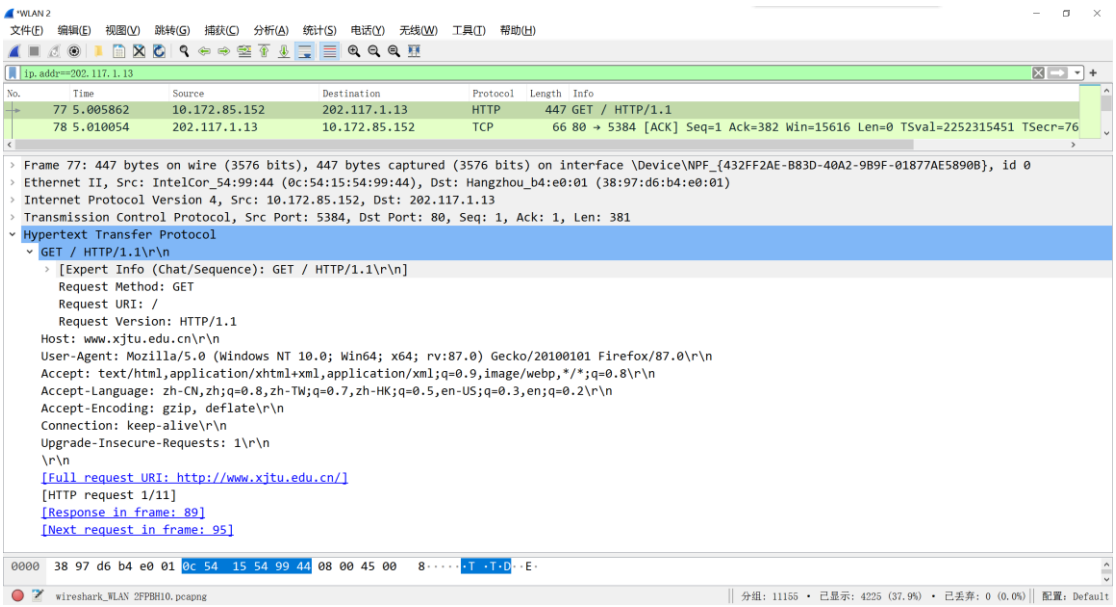
观察几个协议的配合使用，注意访问的延迟情况。特别分析 HTTP 的请求和应答。注意一个网址的访问中，用了几个连接，取了几个对象（HTML，CSS，JS，图片等），有几次 DNS 解析，有没有 Cookie 等。

首先清除本地缓存。

```
C:\WINDOWS\system32>ipconfig/flushdns

Windows IP 配置

已成功刷新 DNS 解析缓存。

C:\WINDOWS\system32>arp -d
```

由于 github 近期暂时无法访问，因此改成访问 www.xjtu.edu.cn、www.unb.br、dean.xjtu.edu.cn 三个网址，同时用 wireshark 进行抓包。

用 ip.addr 过滤出访问 www.xjtu.edu.cn 的数据包，可以看到第一个 http 报文如下：



通过课本内容我们知道，一个 HTTP 请求报文一共包含四个部分，分别是请求行、请求头部、空行和请求数据，现在我们通过上面抓取到的实例具体分析一下。

首先 GET /　HTTP/1.1\r\n 是请求行，GET 代表了请求方法，说明客户端要向服务器请求一个文件。/代表 URL，由于我们访问的是交大的主页，因此使用的是缺省 URL。HTTP/1.1 代表了所使用的 HTTP 协议的版本。后面两个分别是回车符和换行符。

剩余部分在\r\n 之前的都是请求头部，它是由多个头部字段名-值的二元组组成的，下面逐一进行分析。

Host：请求的主机名，这里我们要访问的是交大的主页，因此主机名即其域名。

User-Agent：用户代理，用来告诉服务器客户端的类型与版本。

Accept：客户端可识别的内容类型。

Accept-Language：客户端所期望的语言类型。

Accept-Encoding：客户端可识别的编码格式类型。

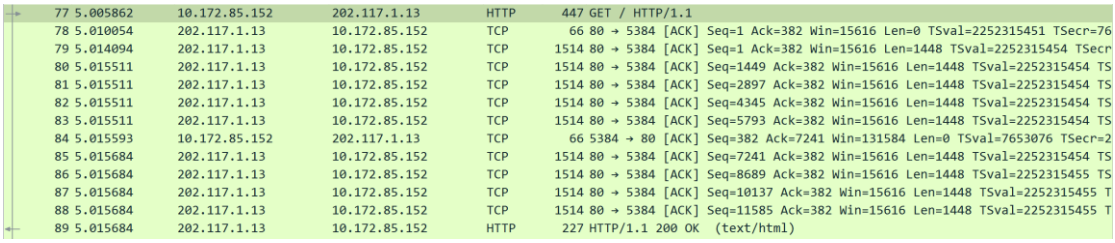
Connection：是否需要持久连接，这里默认的是维持链接。

Upgrade-Insecure-Requests:1：表示客户端向服务器发送一个客户端对 HTTPS 加密和认证响应良好且可以成功处理的信号，这意味着客户端可以请求主机的 HTTPS 资源。

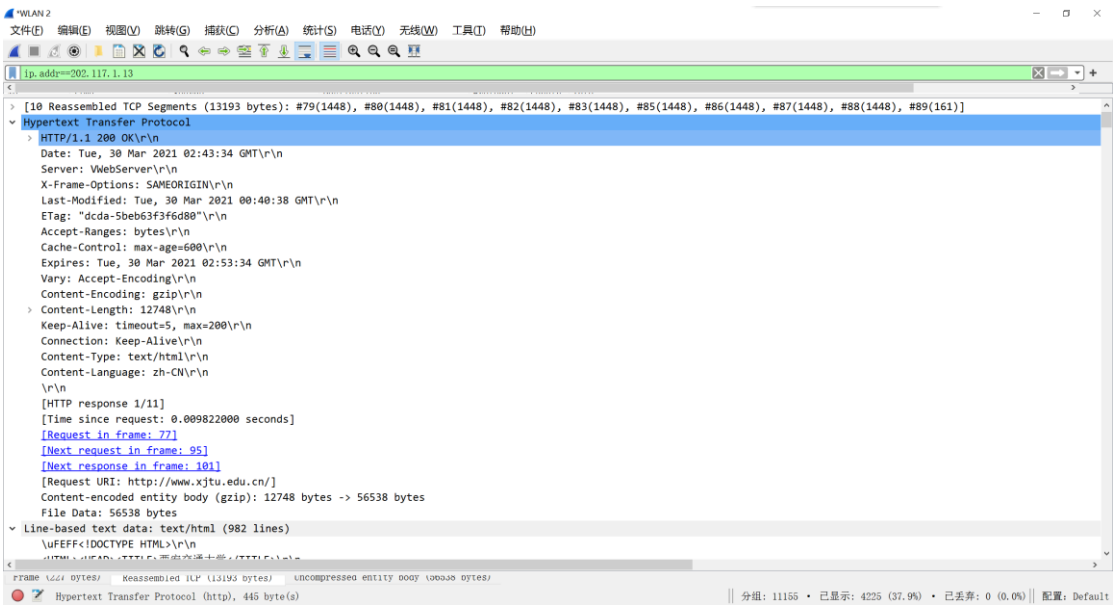
最后一行是空行，用于分隔请求头和请求数据。

由于该 HTTP 是 GET 报文，它只需要服务器返回所需要的文件，因此它并没有请求数据部分。

在这报文发出之后，服务器端接收到请求，于是通过 tcp 报文向主机发送所请求的文件，发送过程见下图。



经过 11 个 tcp 报文的传输，主机所请求的文件发送完成，此时服务器再向主机发送一个 http 应答报文，通知主机所需要的数据已经全部传输到位了。下面具体分析一下应答报文。



HTTP 应答报文也包括四部分，分别是 HTTP 协议版本、状态码与状态描述、响应头和响应体。

HTTP/1.1 表示响应报文的协议版本是 1.1，200 是状态码，代表正常返回了客户端所请求的文件。

响应头也是由多个二元组构成，具体分析如下。

Date：返回报文的时间。

Server：服务器名称。

X-Frame-Options：点击劫持保护。

Last-Modified：上一次修改时间。客户端可以根据此信息选择是否更新本地缓存。

ETag：特定版本资源的标识符。

Accept-Ranges：服务器通过 byte serving 支持的部分内容范围类型。

Cache-Control：告诉从服务端到客户端的所有的缓存机制，是否可以缓存这个对象以及可保存的时间。

Expires：响应体的过期时间。可以看出，这个响应体的存活时间是 2 小时左右。

Vary：通知下级代理如何匹配未来的请求头以让其决定缓存的响应是否可用。

Content-Encoding：发送数据的编码方式。

Content-Length：发送数据的长度。

Keep-Alive：保持连接的时间。

Connection：是否保持连接。

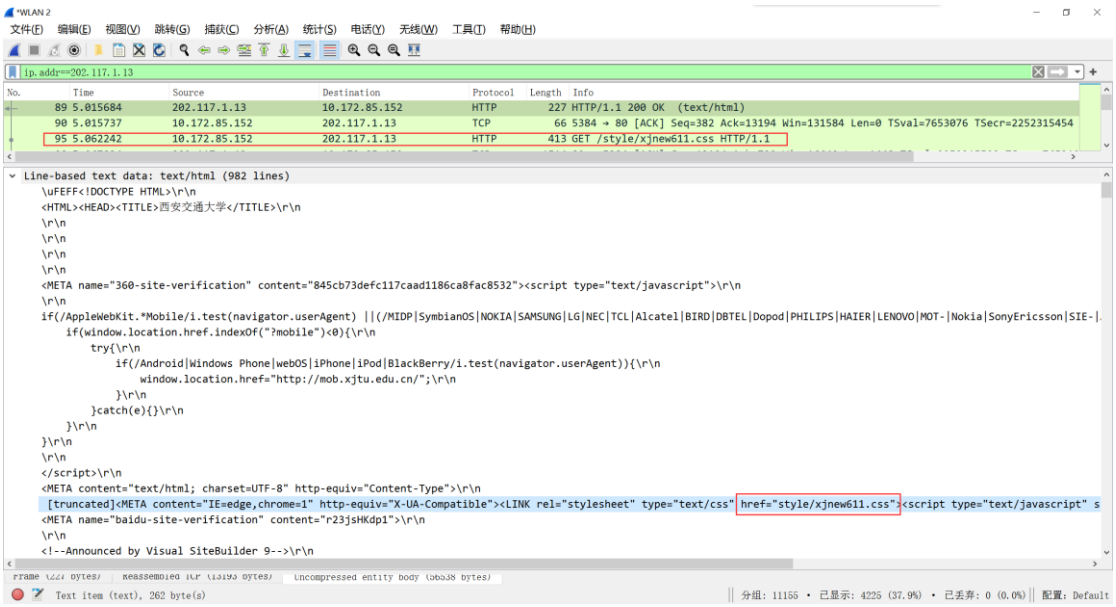
Content-Type：发送数据的文件格式。

Content-Language：发送文件的语言。

空行之后是一个响应体，它包含了客户端请求的文件。由于我们请求了交大的主页，因此服务器返回了 gzip 编码的交大主页的 HTML 文件。如果我们把最后一行的 text data 点开，就能看到原始未经浏览器渲染的 HTML 文件了。

```
Line-based text data: text/html (982 lines)
    \uFEFF<!DOCTYPE HTML>\r\n
    <HTML><HEAD><TITLE>西安交通大学</TITLE>\r\n
    \r\n
    \r\n
    \r\n
    \r\n
    <META name="360-site-verification" content="845cb73defc117caad1186ca8fac8532"><script type="text/javascript">\r\n
    \r\n
    if(/AppleWebKit.*Mobile/i.test(navigator.userAgent) ||(/MIDP|SymbianOS|NOKIA|SAMSUNG|LG|NEC|TCL|Alcatel|BIRD|DBTEL|Dopod|PHILIPS|HAIER|LENOVO|MOT-|Nokia|SonyEricsson|SIE-|
        if(window.location.href.indexOf("?mobile")<0){\r\n
            try{\r\n
                if(/Android|Windows Phone|webOS|iPhone|iPod|BlackBerry/i.test(navigator.userAgent)){\r\n
                    window.location.href="http://mob.xjtu.edu.cn/";\r\n
                }\r\n
            }catch(e){}\r\n
        }\r\n
    }\r\n
    \r\n
    </script>\r\n
    <META content="text/html; charset=UTF-8" http-equiv="Content-Type">\r\n
     [truncated]<META content="IE=edge,chrome=1" http-equiv="X-UA-Compatible"><LINK rel="stylesheet" type="text/css" href="style/xjnew611.css"><script type="text/javascript" s
    <META name="baidu-site-verification" content="r23jsHKdp1">\r\n
    \r\n
    <!--Announced by Visual SiteBuilder 9-->\r\n
    <link rel="stylesheet" type="text/css" href="_sitegray/_sitegray_d.css" />\r\n
    <script language="javascript" src="_sitegray/_sitegray.js"></script>\r\n
    <!-- CustomerNO:77656262657232306975475453525442030500000 -->\r\n
    <link rel="stylesheet" type="text/css" href="index.vsb.css" />\r\n
```

接下来可以看出，在主机得到的主页的 html 文件中，有 style/xjnew611.css 这样一个超链接，当主机浏览器解析到这里时，发现找不到这个文件，于是便会再次向服务器请求这个文件，因此我们就能看到，在响应报文返回不久，主机再次向服务器发送了一个 http 请求报文。该报文的具体格式在上面已经分析过了，故不再赘述。



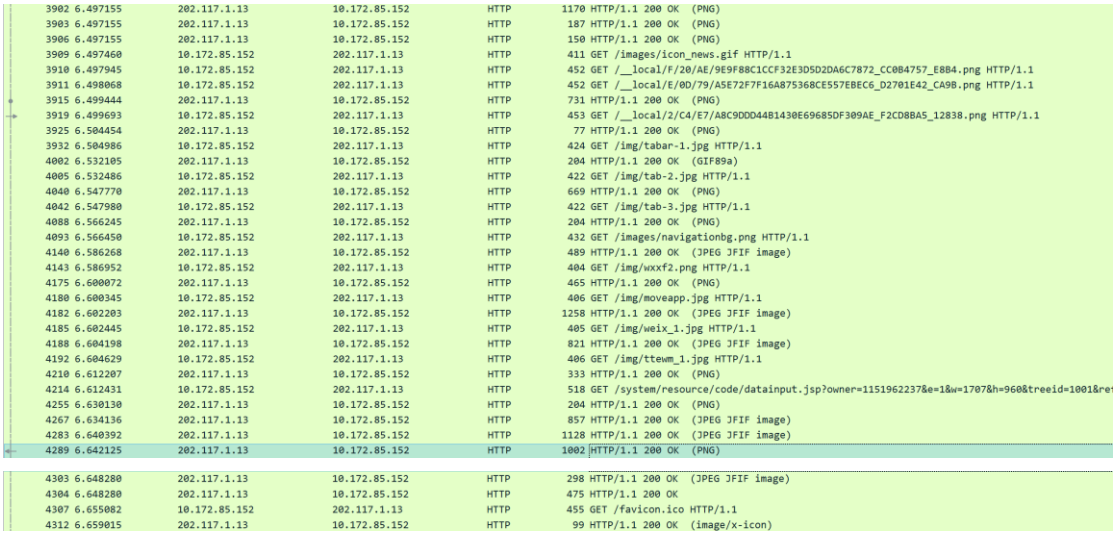
接下来主机就是在递归地请求所有需要的文件，服务器也就根据收到的请求依次发送文件，这里截取了请求交大主页的整个过程中的 http 报文情况。

```
    77 5.005862    10.172.85.152    202.117.1.13     HTTP    447 GET / HTTP/1.1
    89 5.015684    202.117.1.13     10.172.85.152    HTTP    227 HTTP/1.1 200 OK  (text/html)
    95 5.062242    10.172.85.152    202.117.1.13     HTTP    413 GET /style/xjnew611.css HTTP/1.1
   101 5.068080    202.117.1.13     10.172.85.152    HTTP    253 HTTP/1.1 200 OK  (text/css)
   107 5.072453    10.172.85.152    202.117.1.13     HTTP    411 GET /system/resource/js/dynclicks.js HTTP/1.1
   112 5.075055    10.172.85.152    202.117.1.13     HTTP    420 GET /_sitegray/_sitegray_d.css HTTP/1.1
   115 5.076018    10.172.85.152    202.117.1.13     HTTP    402 GET /_sitegray/_sitegray.js HTTP/1.1
   117 5.076985    202.117.1.13     10.172.85.152    HTTP    114 HTTP/1.1 200 OK  (application/javascript)
   123 5.077494    10.172.85.152    202.117.1.13     HTTP    396 GET /js/jquery.min.js HTTP/1.1
   124 5.077617    10.172.85.152    202.117.1.13     HTTP    403 GET /js/jquery.SuperSlide.js HTTP/1.1
   125 5.077809    10.172.85.152    202.117.1.13     HTTP    410 GET /system/resource/js/formfunc.js HTTP/1.1
   128 5.078022    202.117.1.13     10.172.85.152    HTTP    546 HTTP/1.1 200 OK  (text/css)
   130 5.078482    10.172.85.152    202.117.1.13     HTTP    410 GET /system/resource/js/openlink.js HTTP/1.1
   131 5.078738    10.172.85.152    202.117.1.13     HTTP    408 GET /system/resource/js/base64.js HTTP/1.1
   133 5.081184    202.117.1.13     10.172.85.152    HTTP    619 HTTP/1.1 200 OK  (application/javascript)
   134 5.081620    10.172.85.152    202.117.1.13     HTTP    408 GET /index.vsb.css HTTP/1.1
   137 5.082598    202.117.1.13     10.172.85.152    HTTP    909 HTTP/1.1 200 OK  (application/javascript)
   140 5.082598    202.117.1.13     10.172.85.152    HTTP    941 HTTP/1.1 200 OK  (application/javascript)
   142 5.082982    10.172.85.152    202.117.1.13     HTTP    409 GET /system/resource/js/counter.js HTTP/1.1
   143 5.083294    10.172.85.152    202.117.1.13     HTTP    398 GET /js/ddsmoothmenu.js HTTP/1.1
   146 5.084062    202.117.1.13     10.172.85.152    HTTP    150 HTTP/1.1 200 OK  (application/javascript)
   147 5.084062    202.117.1.13     10.172.85.152    HTTP    806 HTTP/1.1 200 OK  (application/javascript)
   149 5.085828    202.117.1.13     10.172.85.152    HTTP    728 HTTP/1.1 200 OK  (text/css)
   150 5.086449    202.117.1.13     10.172.85.152    HTTP   1268 HTTP/1.1 200 OK  (application/javascript)
   163 5.090087    202.117.1.13     10.172.85.152    HTTP    828 HTTP/1.1 200 OK  (application/javascript)
   173 5.094905    202.117.1.13     10.172.85.152    HTTP    401 HTTP/1.1 200 OK  (application/javascript)
   175 5.095261    10.172.85.152    202.117.1.13     HTTP    409 GET /img/logo_pic99.png HTTP/1.1
   176 5.095359    10.172.85.152    202.117.1.13     HTTP    433 GET /images/14/12/11/1tf5znre9c/20210329_02.png HTTP/1.1
   177 5.095435    10.172.85.152    202.117.1.13     HTTP    430 GET /images/14/12/11/1tf5znre9c/20210324.jpg HTTP/1.1
   178 5.095506    10.172.85.152    202.117.1.13     HTTP    430 GET /images/14/12/11/1tf5znre9c/20210325.png HTTP/1.1
```

179	5.095578	10.172.85.152	202.117.1.13	HTTP	414	GET /images/202103240111.png HTTP/1.1
180	5.096191	10.172.85.152	202.117.1.13	HTTP	430	GET /images/14/12/11/1tf5znre9c/20210311.png HTTP/1.1
202	5.103267	202.117.1.13	10.172.85.152	HTTP	1249	HTTP/1.1 200 OK (PNG)
209	5.103911	10.172.85.152	202.117.1.13	HTTP	403	GET /img/zy01.png HTTP/1.1
272	5.107627	202.117.1.13	10.172.85.152	HTTP	119	HTTP/1.1 200 OK (PNG)
276	5.108215	10.172.85.152	202.117.1.13	HTTP	403	GET /img/zy02.png HTTP/1.1
350	5.114063	202.117.1.13	10.172.85.152	HTTP	1337	HTTP/1.1 200 OK (PNG)
354	5.114634	10.172.85.152	202.117.1.13	HTTP	403	GET /img/zy03.png HTTP/1.1
441	5.129922	202.117.1.13	10.172.85.152	HTTP	1290	HTTP/1.1 200 OK (PNG)
443	5.130315	10.172.85.152	202.117.1.13	HTTP	403	GET /img/zy04.png HTTP/1.1
566	5.179119	202.117.1.13	10.172.85.152	HTTP	1305	HTTP/1.1 200 OK (PNG)
568	5.179474	10.172.85.152	202.117.1.13	HTTP	408	GET /images/search.png HTTP/1.1
712	5.238804	202.117.1.13	10.172.85.152	HTTP	714	HTTP/1.1 200 OK (PNG)
713	5.239246	10.172.85.152	202.117.1.13	HTTP	424	GET /img/beijing.jpg HTTP/1.1
973	5.339129	202.117.1.13	10.172.85.152	HTTP	1133	HTTP/1.1 200 OK (JPEG JFIF image)
1000	5.350526	10.172.85.152	202.117.1.13	HTTP	403	GET /img/zy05.png HTTP/1.1
1168	5.418539	202.117.1.13	10.172.85.152	HTTP	1260	HTTP/1.1 200 OK (PNG)
1171	5.418812	10.172.85.152	202.117.1.13	HTTP	452	GET /__local/0/8E/28/41F4105FBAF2B19D014ACB49873_EFA54294_E532.jpg HTTP/1.1
1573	5.583062	202.117.1.13	10.172.85.152	HTTP	1220	HTTP/1.1 200 OK (JPEG JFIF image)
1576	5.583297	10.172.85.152	202.117.1.13	HTTP	425	GET /img/in_xn_18.png HTTP/1.1
1659	5.617053	202.117.1.13	10.172.85.152	HTTP	242	HTTP/1.1 200 OK (JPEG JFIF image)
1664	5.617279	10.172.85.152	202.117.1.13	HTTP	425	GET /img/in_xn_28.png HTTP/1.1
1775	5.661306	202.117.1.13	10.172.85.152	HTTP	1466	HTTP/1.1 200 OK (PNG)
1777	5.661566	10.172.85.152	202.117.1.13	HTTP	425	GET /img/in_xn_20.png HTTP/1.1
1838	5.687865	202.117.1.13	10.172.85.152	HTTP	1459	HTTP/1.1 200 OK (PNG)
1842	5.688284	10.172.85.152	202.117.1.13	HTTP	427	GET /img/tabar-1-on.jpg HTTP/1.1
1954	5.732392	202.117.1.13	10.172.85.152	HTTP	370	HTTP/1.1 200 OK (PNG)
1956	5.732655	10.172.85.152	202.117.1.13	HTTP	452	GET /__local/5/8D/EE/B24A9E3EC8FF4296DA498F897FF_DC240B27_ECAA.png HTTP/1.1
2041	5.765066	202.117.1.13	10.172.85.152	HTTP	605	HTTP/1.1 200 OK (JPEG JFIF image)
2043	5.765278	10.172.85.152	202.117.1.13	HTTP	452	GET /__local/F/CB/0E/8EDE5DD9412B48140269EEA596E_638FE5A8_71F1.jpg HTTP/1.1

2133	5.801305	202.117.1.13	10.172.85.152	HTTP	308	HTTP/1.1 200 OK (PNG)
2135	5.801533	10.172.85.152	202.117.1.13	HTTP	452	GET /__local/5/CA/5F/A00D00C875DD6A75BA769CE795D_4FC8CAAA_FE66.png HTTP/1.1
2259	5.852785	202.117.1.13	10.172.85.152	HTTP	675	HTTP/1.1 200 OK (JPEG JFIF image)
2261	5.853196	10.172.85.152	202.117.1.13	HTTP	427	GET /img/tabar-2-on.jpg HTTP/1.1
2424	5.917547	202.117.1.13	10.172.85.152	HTTP	235	HTTP/1.1 200 OK (PNG)
2428	5.917763	10.172.85.152	202.117.1.13	HTTP	452	GET /__local/2/A7/76/C50B7F0CFD91DF255165A5F8194_BF9A3DAD_BF9D.png HTTP/1.1
2516	5.952023	202.117.1.13	10.172.85.152	HTTP	773	HTTP/1.1 200 OK (JPEG JFIF image)
2519	5.952299	10.172.85.152	202.117.1.13	HTTP	452	GET /__local/6/06/18/4291178CA1D70AB8DF3C6CC24F7_1BCD08B5_8444.png HTTP/1.1
2640	6.000004	202.117.1.13	10.172.85.152	HTTP	431	HTTP/1.1 200 OK (PNG)
2642	6.000262	10.172.85.152	202.117.1.13	HTTP	452	GET /__local/E/FB/D5/409E438D506FCC4FA79567A5318_379ADF15_79A3.png HTTP/1.1
2924	6.113420	202.117.1.13	10.172.85.152	HTTP	1196	HTTP/1.1 200 OK (PNG)
2927	6.113569	10.172.85.152	202.117.1.13	HTTP	427	GET /img/tabar-3-on.jpg HTTP/1.1
2935	6.117216	202.117.1.13	10.172.85.152	HTTP	286	HTTP/1.1 200 OK (JPEG JFIF image)
2940	6.117491	10.172.85.152	202.117.1.13	HTTP	452	GET /__local/A/25/3E/1A67A55877187C7270C3346778E_E97A7580_F01F.png HTTP/1.1
3013	6.147370	202.117.1.13	10.172.85.152	HTTP	1021	HTTP/1.1 200 OK (PNG)
3016	6.147627	10.172.85.152	202.117.1.13	HTTP	452	GET /__local/3/A5/13/86E63A6ED1933B34EF77C664C3D_3B4DEAED_B61F.png HTTP/1.1
3133	6.193042	202.117.1.13	10.172.85.152	HTTP	821	HTTP/1.1 200 OK (JPEG JFIF image)
3135	6.193259	10.172.85.152	202.117.1.13	HTTP	452	GET /__local/6/16/44/FD4C5A2E4CC7D897A1D17AD6364_9C289FCB_F377.png HTTP/1.1
3210	6.223119	202.117.1.13	10.172.85.152	HTTP	395	HTTP/1.1 200 OK (PNG)
3212	6.223302	10.172.85.152	202.117.1.13	HTTP	425	GET /img/in_xn_tb.png HTTP/1.1
3424	6.309599	202.117.1.13	10.172.85.152	HTTP	1120	HTTP/1.1 200 OK (PNG)
3429	6.309835	10.172.85.152	202.117.1.13	HTTP	425	GET /img/in_xn_cw.png HTTP/1.1
3446	6.318271	202.117.1.13	10.172.85.152	HTTP	1214	HTTP/1.1 200 OK (PNG)
3451	6.318483	10.172.85.152	202.117.1.13	HTTP	425	GET /img/in_xn_ky.png HTTP/1.1
3505	6.340209	202.117.1.13	10.172.85.152	HTTP	752	HTTP/1.1 200 OK (PNG)
3508	6.340444	10.172.85.152	202.117.1.13	HTTP	427	GET /img/yidongxjtu.png HTTP/1.1
3604	6.377029	202.117.1.13	10.172.85.152	HTTP	528	HTTP/1.1 200 OK (PNG)
3607	6.377252	10.172.85.152	202.117.1.13	HTTP	425	GET /img/icon_a10.gif HTTP/1.1
3615	6.381047	202.117.1.13	10.172.85.152	HTTP	781	HTTP/1.1 200 OK (PNG)
3618	6.381429	10.172.85.152	202.117.1.13	HTTP	425	GET /img/icon_a11.png HTTP/1.1

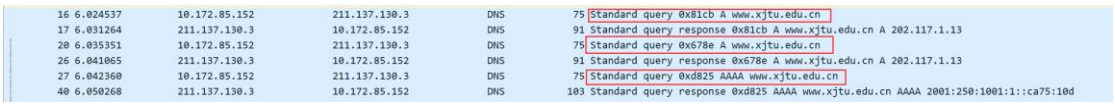
3631	6.387038	202.117.1.13	10.172.85.152	HTTP	795	HTTP/1.1 200 OK (PNG)
3634	6.387322	10.172.85.152	202.117.1.13	HTTP	416	GET /images/20190925chuxin.jpg HTTP/1.1
3672	6.407591	202.117.1.13	10.172.85.152	HTTP	742	HTTP/1.1 200 OK (PNG)
3675	6.407932	10.172.85.152	202.117.1.13	HTTP	407	GET /images/170-2.jpg HTTP/1.1
3682	6.411997	202.117.1.13	10.172.85.152	HTTP	1195	HTTP/1.1 200 OK (PNG)
3687	6.412258	10.172.85.152	202.117.1.13	HTTP	409	GET /images/1234444.png HTTP/1.1
3689	6.413373	202.117.1.13	10.172.85.152	HTTP	608	HTTP/1.1 200 OK (GIF89a)
3694	6.413634	10.172.85.152	202.117.1.13	HTTP	407	GET /images/170-1.jpg HTTP/1.1
3703	6.417073	202.117.1.13	10.172.85.152	HTTP	1136	HTTP/1.1 200 OK (PNG)
3705	6.417408	10.172.85.152	202.117.1.13	HTTP	406	GET /images/zzkj.png HTTP/1.1
3708	6.419513	202.117.1.13	10.172.85.152	HTTP	913	HTTP/1.1 200 OK (PNG)
3711	6.419891	10.172.85.152	202.117.1.13	HTTP	426	GET /images/17/09/04/1mi2amdixo/wmjd.png HTTP/1.1
3728	6.427166	202.117.1.13	10.172.85.152	HTTP	98	HTTP/1.1 200 OK (JPEG JFIF image)
3733	6.427382	10.172.85.152	202.117.1.13	HTTP	426	GET /img/in_app_80.png HTTP/1.1
3737	6.429394	202.117.1.13	10.172.85.152	HTTP	1126	HTTP/1.1 200 OK (JPEG JFIF image)
3739	6.429701	10.172.85.152	202.117.1.13	HTTP	412	GET /images/2020072111.png HTTP/1.1
3765	6.444885	202.117.1.13	10.172.85.152	HTTP	487	HTTP/1.1 200 OK (PNG)
3770	6.444885	202.117.1.13	10.172.85.152	HTTP	1493	HTTP/1.1 200 OK (JPEG JFIF image)
3773	6.445205	10.172.85.152	202.117.1.13	HTTP	430	GET /images/18/01/23/1ay37aq43t/icon_r_1.png HTTP/1.1
3774	6.445528	10.172.85.152	202.117.1.13	HTTP	430	GET /images/18/01/23/1ay37aq43t/icon_r_2.png HTTP/1.1
3825	6.470913	202.117.1.13	10.172.85.152	HTTP	289	HTTP/1.1 200 OK (PNG)
3826	6.470913	202.117.1.13	10.172.85.152	HTTP	1469	HTTP/1.1 200 OK (PNG)
3828	6.471203	10.172.85.152	202.117.1.13	HTTP	430	GET /images/18/01/23/1ay37aq43t/icon_r_3.png HTTP/1.1
3829	6.471508	10.172.85.152	202.117.1.13	HTTP	430	GET /images/18/01/23/1ay37aq43t/icon_r_4.png HTTP/1.1
3831	6.472098	202.117.1.13	10.172.85.152	HTTP	481	HTTP/1.1 200 OK (PNG)
3836	6.472431	10.172.85.152	202.117.1.13	HTTP	430	GET /images/18/01/23/1ay37aq43t/icon_r_6.png HTTP/1.1
3838	6.473277	202.117.1.13	10.172.85.152	HTTP	212	HTTP/1.1 200 OK (PNG)
3843	6.473530	10.172.85.152	202.117.1.13	HTTP	425	GET /img/in_xn_11.png HTTP/1.1
3846	6.475208	202.117.1.13	10.172.85.152	HTTP	190	HTTP/1.1 200 OK (PNG)
3851	6.475476	10.172.85.152	202.117.1.13	HTTP	425	GET /img/in_xn_13.png HTTP/1.1



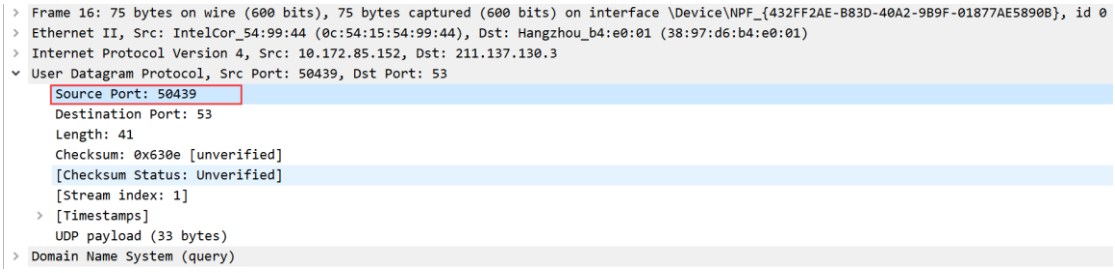
对于请求过程中协议配合，具体来说就是先由 http 协议提出对某文件的请求，然后服务器收到请求后通过 TCP 协议将所需的文件传输给主机，再向主机发送一个 HTTP 报文表示完成请求。整个过程中取了 html、css、js、png、jpg、gif 共 6 种对象。

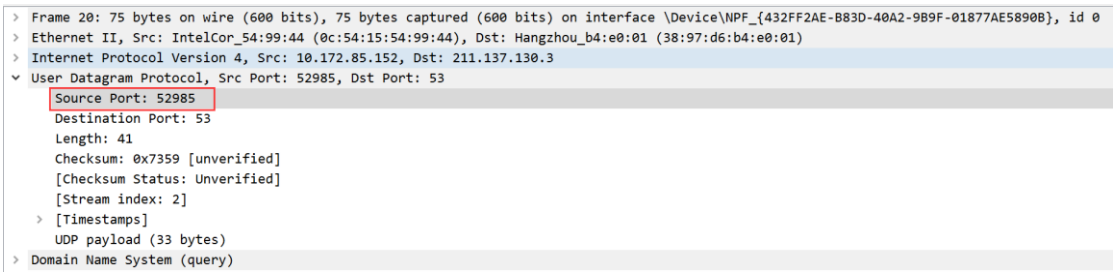
同时，分析访问网站时的 DNS 报文，可以发现，每个网址都会进行 3 次 DNS 查询，其中有 2 次是类型为 A 的 DNS 查询，用于查找域名对应的 ipv4 的地址；1 次是类型为 AAAA 的 DNS 查询，用于查找域名对应的 ipv6 的地址。这里 2 次 ipv4 地址查询的源端口号不同。同时，如果返回的主页的 html 文件中还存在其他的网站的域名，主机便会递归地对这些域名进行 DNS 查询。举例来说，www.xjtu.edu.cn 的主页中有包含 en.xjtu.edu.cn 的超链接存在，那么主机便会自动的继续查询该域名对应的 ip 地址。

下图是对 www.xjtu.edu.cn 的三次 DNS 查询以及响应。



下图是上面两个查询 ipv4 地址的 DNS 报文的详细信息，可以看出，这两个报文的源端口号其实并不一样。实验过程中我还抓到了同一个源端口发送了两个 ipv4 地址查询的 DNS 报文，只是两个目的地址不同（因为我的主机默认有 2 个 DNS 服务器地址可供查询）。





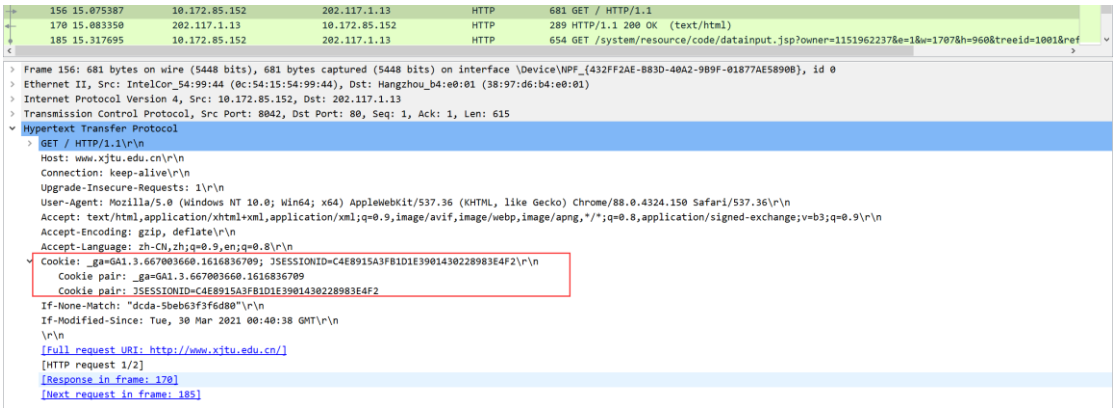
下面就是很多递归地 DNS 查询，由于数量太多这里只列出部分查询报文和响应报文。

同时，由于在抓包之前所有缓存都已经被清空，因此报文中并没有附带 cookie 信息。

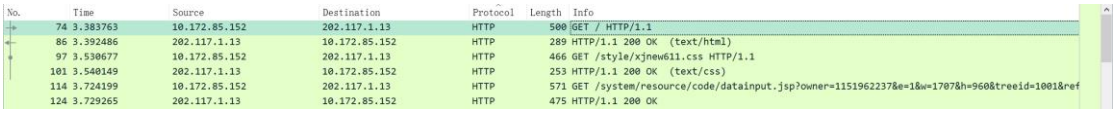
（二）带缓存的 ARP，DNS 和 HTTP 协议分析

照着 1.7.1 中的步骤 1-4 再次执行一遍，但不执行步骤 2。观察缓存的使用和带来的好处。

重新执行后，由于缓存的存在，访问各个网页的响应时间都有了较大的减少，同时在 GET 请求报文中，可以看见 cookie 信息的存在。



同时也能发现，由于缓存的存在，大部分未经修改的文件可以直接通过本地缓存来获得，因此主机请求的 HTTP 报文也明显减少了。下图是再次访问 www.xjtu.edu.cn 是抓到的所有 HTTP 报文，可以看出数量和没有缓存时相比有了显著减少。



（三）使用 ncat 工具访问 HTTP 服务

参考 1.7.1 中的步骤 1-4 和分析结果，在命令窗口执行 ncat -C xxx.xxx.xxx.xxx 80，ncat 连接上 HTTP 服务器后，根据协议输入合适的请求。其中 xxx.xxx.xxx.xxx 为服务器地址。

```
┌──(syj㉿NetExp)-[~/Desktop]
└─$ ncat -C 81.70.134.50 80
GET / HTTP/1.1
Host: 81.70.134.50

HTTP/1.1 200 OK
Date: Tue, 30 Mar 2021 14:55:47 GMT
Server: Apache/2.4.6 (CentOS)
Last-Modified: Tue, 23 Mar 2021 08:32:43 GMT
ETag: "69-5be3006a9ca1b"
Accept-Ranges: bytes
Content-Length: 105
Content-Type: text/html; charset=UTF-8

<a href="./download/Zelda.rar">DOWNLOAD(3.2MB)</a>
<br>
<a href="./download/FF7.rar">DOWNLOAD(500KB)</a>
```

上图是利用 ncat 请求 http 报文的结果，其中第一行为云服务器 IP 地址及端口号，第二行为 HTTP 请求头，第三行为主机名。由于服务器没有设定主机名，故用 IP 地址代替。下面的部分为云服务器返回的 HTTP 响应报文。

六、 互动讨论主题

1、HTTP 协议的缓存，DNS 的缓存；缓存对网络访问速度的影响。

HTTP 协议缓存机制：

HTTP 协议通过配置文件头中的关键字来告知浏览器该如何缓存，相应头中的关键字主要一下几个：

Cache-Control、*ETag*、*Last-Modified*、*Expires*

下面逐个介绍功能：

Cache-Control：这个关键字说明是的浏览器是否被允许缓存资源，缓存类型，缓存的时间。主要有以下几个值：

a. no-store: 不能缓存。

b. no-cache：这种情况下本地允许缓存，但是每次有请求的时候必须向服务器发送请求报文检查更新是否过期，以及及时更新。

c. public: 代表该 http 响应是公共的，可以被任何中间服务器或者代理缓存。

d. private（默认值）：代表该 http 响应是个人的，不允许被中间服务器或代理缓存。

e. max-age=<seconds>：表示缓存能够被缓存的最大时间。

f. must-revalidate：表示缓存可能以及过期，下次访问必须验证活性。

ETag：针对每个文件的特定标识，可以用来校验文件是否更新。

Last-Modified：该缓存最后的更新时间。

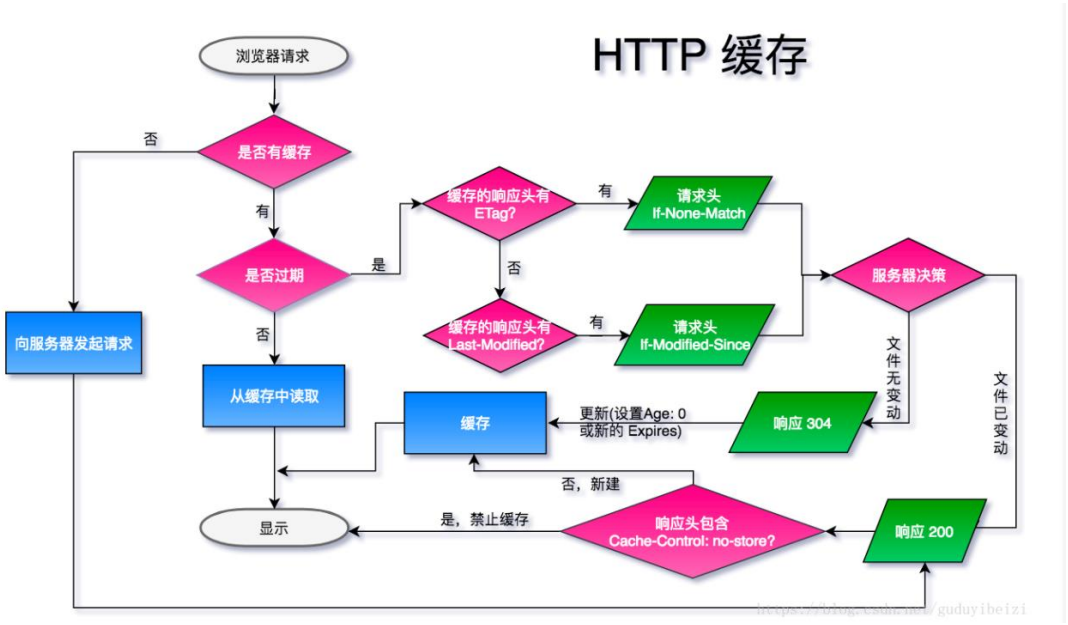
Expires：缓存过期的时间。

相应的，请求报文中也有几个关键字与缓存机制有关：

If-None-Match：上次缓存响应头中的 Etag 数值，用来精确校验文件是否更新。

If-Modified-Since：上次缓存响应头中的 Last-Modified 数值，用来模糊校验文件是否更新。

每次请求时流程大致如下：



DNS 缓存机制：

DNS 缓存在本地浏览器以及操作系统中均存在，用浏览器访问域名时会优先访问浏览器中缓存，如果为命中就会访问操作系统中的缓存。浏览器为每个缓存设置一个固定的生存时间（2 到 30 分钟），同样的，操作系统中的缓存也有生存时间。

在路由器以及代理服务器也有缓存缓存，如果本地未命中就会进行递归或者迭代查询。

缓存机制可以大量减少资源的请求、相应环节，使得网络的访问与下载速度提升，也可以使服务的的效率大大提升。

2、NAT 对 FTP 传输的影响，比较 HTTP 与 FTP 的特点；

HTTP 与 FTP 都是基于 TCP/IP 协议，许多方面其实相差不大。

HTTP 的特点：持续链接，相对于传统的文件传输，部署方便，web 服务器即可。低时延。

FTP 的特点：使用两个链接，每次传输都要建立链接，可以上传下载整个文件夹的内容，但时延较大。FTP 起初优势在于 HTTP 在大文件传输方面的弊端，但是随着协议优化，二者大文件传输效率差别已经基本不大了，甚至 HTTP 更好一点，因此 FTP 已经逐步退出市场。

七、 进阶自设计

1、用 nmap 的 ncat 来模拟 https 客户端，访问 1-2 个网站。

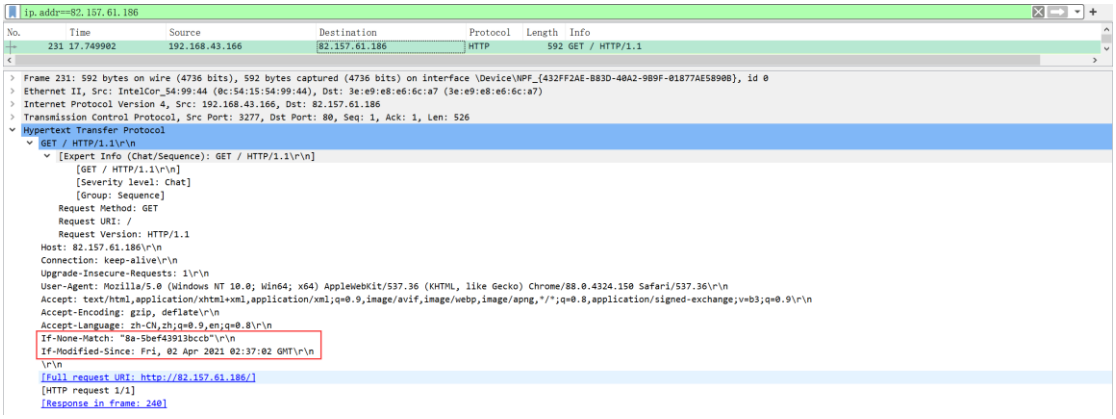
这一步类似用 ncat 模拟请求 HTTP 报文，只要在输入命令是加上--ssl 参数即可。这里利用 ncat 模拟 HTTPS 客户端，访问了 mail.qq.com。

2、在云服务器上搭建 Apache2（或其他 WEB 服务器），并测试修改 HTML 或图片文件，看客户端能否及时访问到更新的内容。注意抓包分析。

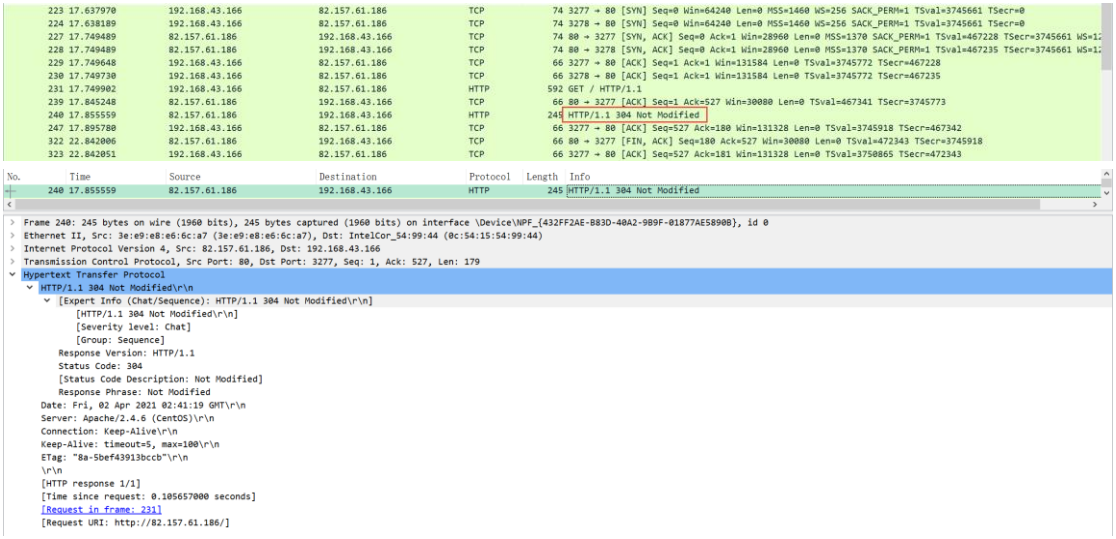
初始的 index.html 文件

当再次访问同一网页时，由于缓存的存在，主机发送的 GET 报文中会包含最新缓存的修改时间，服务器收到这条报文后，会拿这个时间和要请求资源（在本例中，就是 index.html）的最后修改时间相比较，如果在最新缓存时间后这个资源并没有被修改，就说明主机的缓存就是当前最新的资源，因此不需要再次传输数据，只要返回一个 Not Modified 即可。

下图是主机发送的 GET 报文：

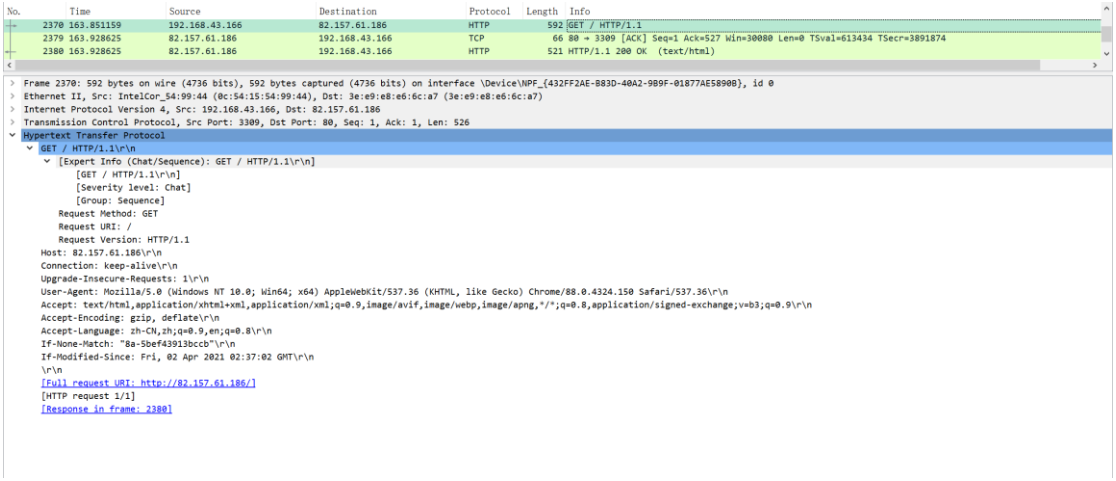


下图是服务器返回的 Not Modified 报文：

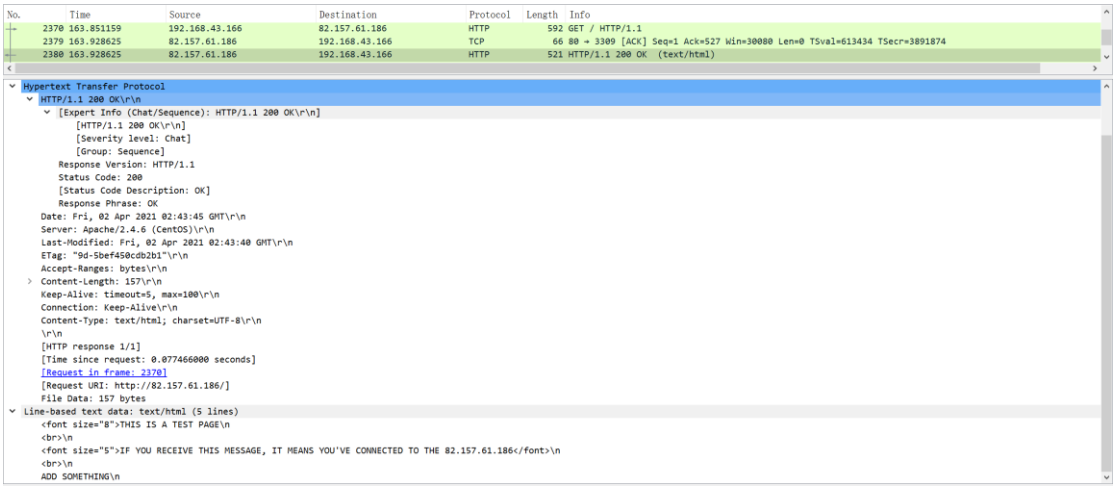


这样做减少了数据的传输，提高了效率。这时我们再把 index.html 文件进行修改。

此时如果不主动刷新页面，页面将保持原状不变。现在我们再主动刷新一次页面并进行抓包。



可以看出，GET 报文和之前的基本相同。



但是由于我们修改了请求的文件，因此服务器端通过比较两个时间，发现在缓存时间之后文件又被进行了修改，因此服务器将直接把新的文件发送到主机。

最终结果：