# 暨南大学本科实验报告专用纸

课程名称 计算机网络实验 成绩评定

实验项目名称 Internet应用与应用层协议分析 指导教师 潘冰

实验项目编号 03 实验项目类型 验证 实验地点 b402

学生姓名 邓芷灵 学号 2019051115

学院 智能科学与工程学院 系 专业 信息安全

实验时间2021 年9月14日 上 午～9月14日上 午

## 实验目的

* 理解WWW、DNS服务、FTP服务、SMTP的作用和原理；
* **学会使用wireshark分析HTTP、FTP、SMTP和DNS协议的工作过程，加深对协议格式和工作原理的理解。**

## 实验内容

* 通过域名访问WWW、FTP服务器，分析DNS、WWW、FTP工作过程，并使用WireShark分析相关协议格式；
* 在客户端访问SMTP服务器，使用wireshark分析SMTP、POP3协议的工作过程。（可以在客户端安装outlook或使用QQ邮件服务器或自己编程）

## 实验设备

计算机若干台

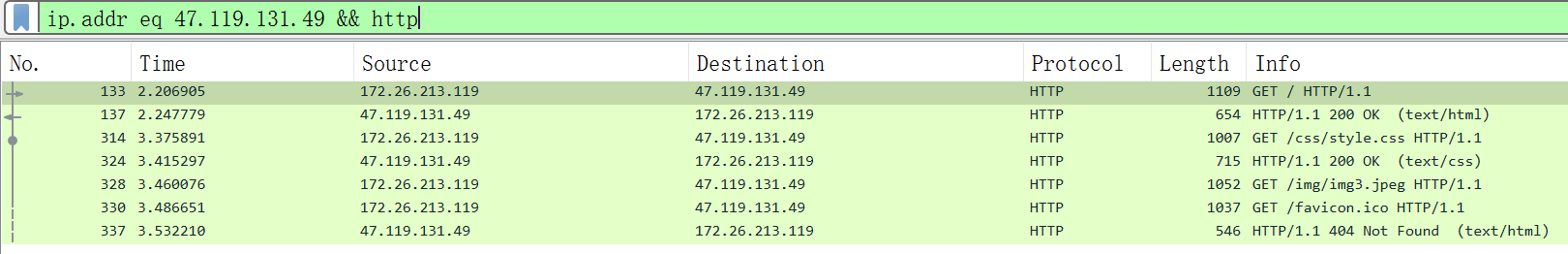
## 实验环境

局部网环境

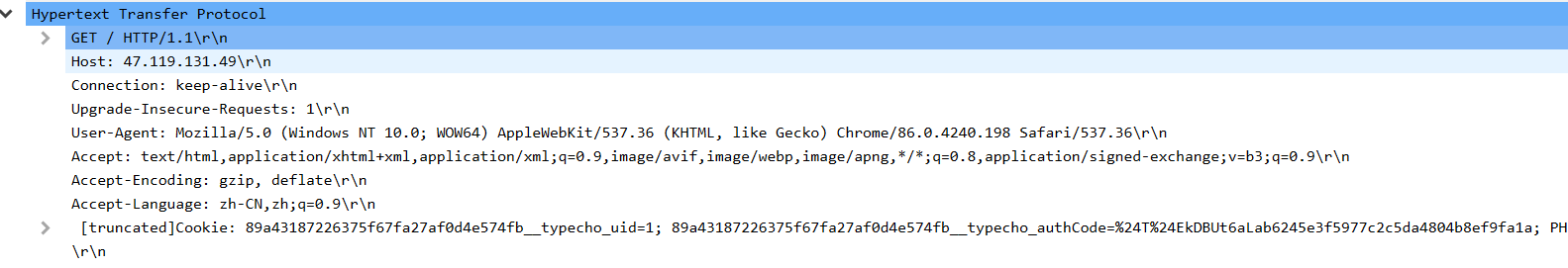
## 实验步骤

### HTTP协议分析

访问此前在阿里云服务器上搭建的简单个人主页：www.mokazure.moe，Wireshark嗅探结果如图：



#### HTTP请求报文分析

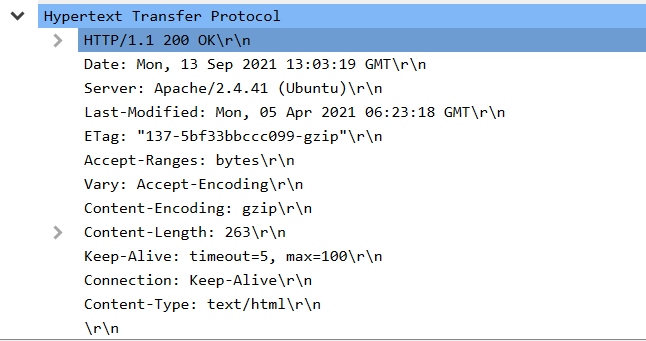


逐行分析如下：

|  |  |
| --- | --- |
| 字段 | 分析 |
| **GET / HTTP/1.1** | 此为请求行，格式为｛方法 URL 版本 CRLF｝（注意空格）。   * GET为方法，表示请求读取由URL所标志的信息； * / 为相对URL，即省略了主机域名，又因为没有子目录，所以只有一个 / ； * HTTP/1.1为HTTP的版本。 |
| **Host:47.119.131.49** | 此行是首部行的开始，给出了主机的域名（可能是因为直接用IP访问，这里给出的是服务器IP）。 |
| **Connection:keep-alive** | Connection字段有2个作用：①控制不再转发给代理的首部字段；②管理持久连接。此处作用为后者，指定keep-alive是为了在旧版本HTTP上也能维持持久连接（HTTP/1.1默认为持久连接）。该字段的另一个值是close，表示明确断开连接。 |
| **Upgrade-Insecure-Requests: 1** | Chrome浏览器在HTTP请求中加入，请求连接升级为HTTPS连接。 |
| **User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.198 Safari/537.36** | User-Agent字段意为用户代理，向服务器说明用户所使用的操作系统、浏览器等。格式为{Mozilla/5.0 (平台) 引擎版本 浏览器版本号}。   * Mozilla/5.0为早期浏览器竞争的历史遗留； * Windows NT 10.0;WOW64表示Windows 10，Win32 on x64，即本机的操作系统； * Chrome/86.0.4240.198为所使用的浏览器内核版本，与浏览器显示的相符；     显示的其余浏览器版本号应该也可以理解为历史遗留产物。 |
| **Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.9** | 表示浏览器接受的响应格式，用分号隔开。q=0.9表示权重优先级，\*/\*表示可以接受任意类型的内容。 |
| **Accept-Encoding: gzip, deflate** | 声明浏览器支持的编码类型为gzip和deflate。 |
| **Accept-Language: zh-CN,zh;q=0.9** | 表示用户希望优先得到中文版本的文档。 |

（图中的\r\n即“CR”“LF”，分别代表“回车”和“换行”，开始行及首部行结尾固有）

#### （2）HTTP响应报文分析



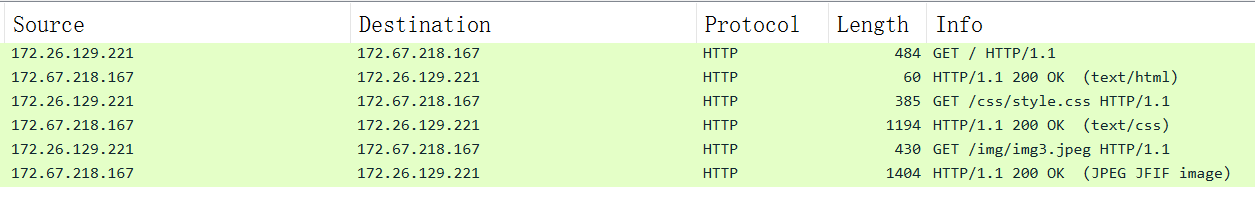
逐行分析如下：

（图中的\r\n即“CR”“LF”，分别代表“回车”和“换行”，开始行及首部行结尾固有）

|  |  |
| --- | --- |
| **字段** | **分析** |
| **HTTP/1.1 200 OK** | 状态行，格式为{版本 状态码 解释状态码的简单短语}。其中状态码都是三位数字，有5大类，其中2xx表示成功。 |
| **Date:Mon,13 Sep 2021 13:03:19 GMT** | 服务器日期，GMT表示格林威治时间。 |
| **Server:Apache/2.4.41 (Ubuntu)** | 处理请求的原始服务器的软件信息。与实际相符。 |
| **Last-Modified: Mon, 05 Apr 2021 06:23:18 GMT** | 所请求资源的最后修改时间。 |
| **ETag:”137-5bf33bbccc099-gzip”** | HTTP 协议规格说明定义ETag为“被请求变量的实体值”，可以连同Last-Modified一起被服务器用于判断页面是否被修改，从而采取不同响应。它允许缓存更高效，并节省带宽，因为如果内容没有改变，Web 服务器不需要发送完整的响应。 |
| **Accept-Ranges: bytes** | 标识自身支持范围请求(partial requests)。字段的具体值用于定义范围请求的单位（这里是以byte为单位）。当浏览器发现 Accept-Ranges 头时，可以尝试继续中断的下载，而不是重新开始。 |
| **Vary: Accept-Encoding** | 列出一个响应字段列表，告诉Cache服务器如何判断应使用一个缓存作为相应还是请求一个新的响应。在这里，响应根据Accept-Encoding这个字段决定，防止因为客户端编码格式出现问题。如对于同一份资源，支持gzip的客户端A先访问，使得Cache服务器缓存了一份gzip格式的资源；随后不支持gzip的客户端B访问同一份资源，即使命中了缓存，Cache服务器也会因为Accept-Encoding的不同而请求一个新的响应，而非直接使用已有的gizp格式的缓存作为响应（如果使用缓存，会导致不支持gzip的客户端B因无法解码而呈现乱码）。 |
| **Content-Encoding: gzip** | 服务器向客户端说明其内容编码格式。这里为gzip格式。 |
| **Content-Length: 263** | 内容长度。 |
| **Keep-Alive: timeout=5, max=100** | 保持TCP连接的同时限制连接时间与最大请求数，超时/超出次数时服务端会关闭连接。这里连接时间为5s，最大请求数为100. |
| **Connection: Keep-Alive** | 表示保持连接，等待后续请求。 |
| **Content-Type: text/html** | 服务端告诉客户端自己响应的类型，这里的text/html表示响应的是HTML。此外还有  text/plain （纯文本格式）、text/xml （XML格式）、image/gif （gif图片格式）、application/pdf（pdf格式）等。 |

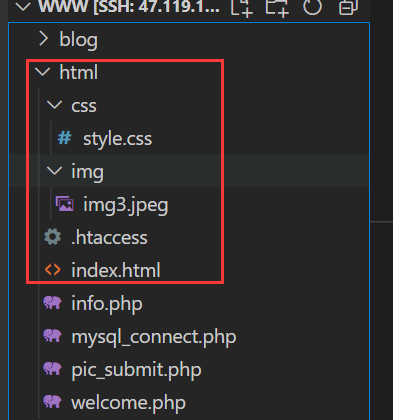
#### HTTP工作过程分析

访问个人主页www.mokazure.moe，Wireshark捕获结果如下：



可以看到，客户端首先向服务器请求网页的HTML文件，再请求CSS文件，最后再请求网页背景图，从而完成对整个页面的加载。在网络比较缓慢的情况下，也可以看到浏览器首先显示出页面文字，再显示出由CSS定义的半透明圆角矩形，最后加载背景图片。

相关资源路径等也与服务器中文件的实际存储路径相符：



### DNS协议分析

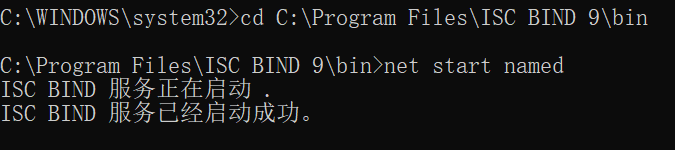
#### 建立DNS服务器

根据这篇[博客](https://www.cnblogs.com/doherasyang/p/14464999.html)完成了ntbind在Windows10下的安装与配置。

* 启用bind服务

以管理员身份运行命令行程序，进入bind9安装目录下的bin文件夹，输入命令net start

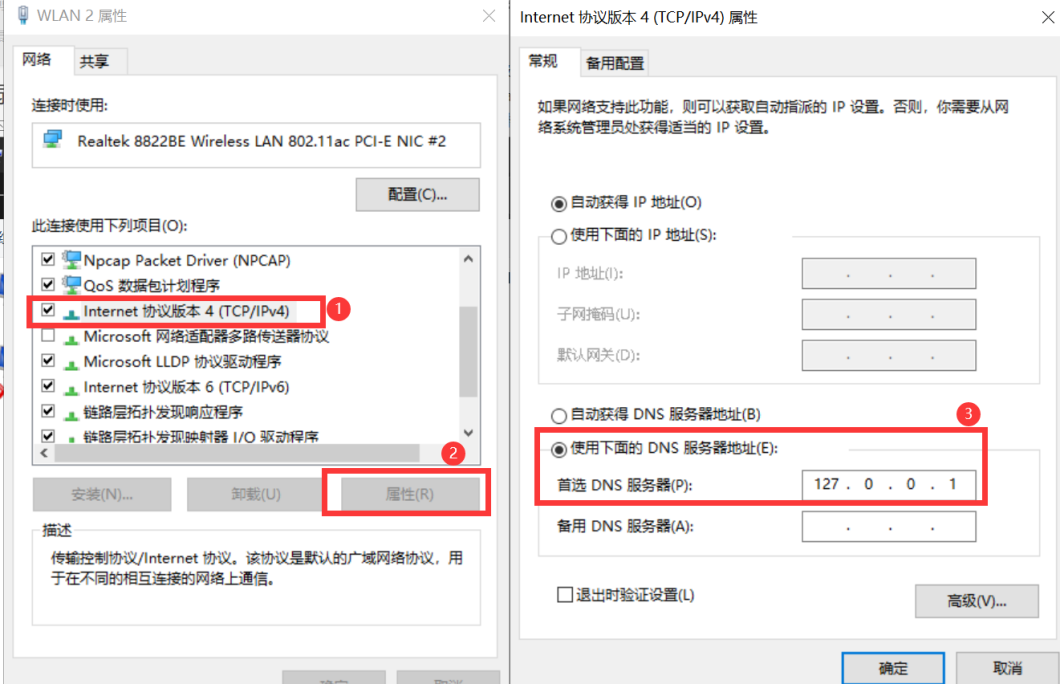
named启用服务，如图所示：



* 更改DNS服务器

打开当前所用网络的属性设置→Internet协议版本4→属性→修改首选DNS服务器地址

为127.0.0.1，即本机环回地址，如图所示：



* 检查解析情况

在命令行程序输入dig ns1.123.com +short，其中dig是一个用于查询DNS相关记录的命

令，ns1.123.com为已经在bind配置文件中建立域名-IP解析关系的一个域名，+short表示只需要返回一个简单结果，如图所示：



这表明ns1.123.com已经被搭建的DNS服务器正确解析为预设的IP：127.0.0.1，搭建的DNS服务器正常工作。

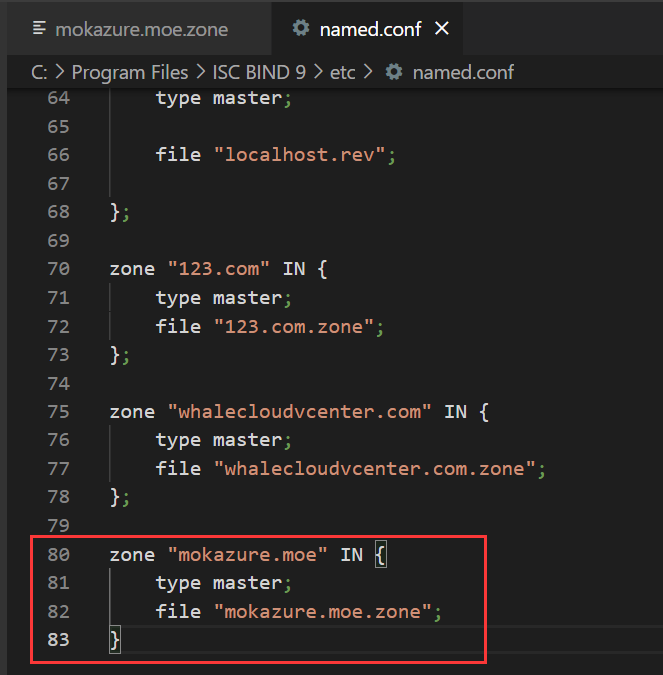
#### 使用DNS

* 在bin9安装目录的etc目录下新增mokazure.com.zone文件，配置如图所示：



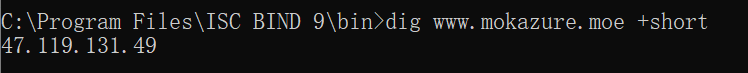
最后一行A记录为个人主页域名mokazure.moe与IP的对应关系。

* 在named.conf文件中补充配置：

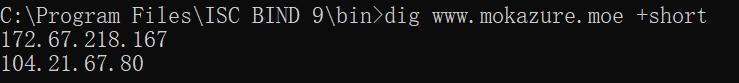


注：修改并保存后出现无法启动BIND服务的问题，通过named-checkconf命令进行语法检查，发现是因为图中最末尾处漏了一个分号。修正之后成功启动服务。

* 检查解析情况：



解析正确。由于mokazure.moe这个域名实际上已经设置为CDN访问，若使用自动配置的DNS服务器，并不会解析到上图中的真实IP，而是：



上图中两个IP均指向CDN服务商CloudFlare，而隐藏了真实IP。

综上，当DNS服务器将www.mokazure.moe解析到真实IP 47.119.131.49时，表明使用

的就是在文件mokazure.com.zone中配置的A记录。

#### DNS协议分析

* DNS报文格式

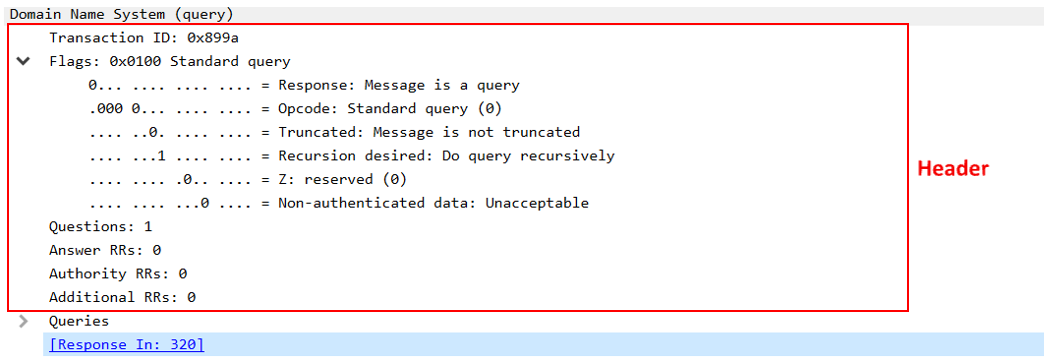
|  |
| --- |
| Header（报头） |
| Question（查询） |
| Answer（应答） |
| Authority（授权应答） |
| Additional（附加信息） |

* Header格式

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| ID | | | | | | | | | | | | | | | |
| QR | Opcode | | | | AA | TC | RD | RA | Z | | | RCODE | | | |
| QDCOUNT | | | | | | | | | | | | | | | |
| ANCOUNT | | | | | | | | | | | | | | | |
| NSCOUNT | | | | | | | | | | | | | | | |
| ARCOUNT | | | | | | | | | | | | | | | |

各字段含义说明如下表：

|  |  |
| --- | --- |
| **字段** | **说明** |
| ID | 16位标识符，标记本条消息 |
| QR | 1位的字段，表明本条消息是查询（0），还是响应（1） |
| Opcode | 4位的字段，表明查询的类型：  0：标准查询；  1：反向查询；  2：服务器状态请求；  3-15：暂未定义，保留。 |
| AA | Authoritative Answer - 只在响应消息中有效，表明该响应是否来自于该域中的权限域名服务器 |
| TC | TrunCation - 表明本条消息是否因为长度过长而被截断 |
| RD | Recursion Desired - 只在查询消息中有效。若置1，则是使域名服务器递归查询（可选） |
| RA | Recursion Available - 只在响应消息中有效，表明域名服务器是否支持递归查询 |
| Z | 保留字段，必须为0. |
| RCODE | Response code - 4位的字段，在响应消息中被设置，各值含义如下：  0：没有错误。  1：报文格式错误(Format error) - 服务器不能理解请求的报文。  2：服务器失败(Server failure) - 因为服务器的原因导致没办法处理这个请求。  3：名字错误(Name Error) - 只有对授权域名解析服务器有意义，指出解析的域名不存在。  4：没有实现(Not Implemented) - 域名服务器不支持查询类型。  5：拒绝(Refused) - 服务器由于设置的策略拒绝给出应答。比如，服务器不希望对某些请求者给出应答，或者服务器不希望进行某些操作（比如区域传送zone transfer）。  6-15：保留。 |
| QDCOUNT | 一个无符号16位整数，表示报文Question段中的问题记录数 |
| ANCOUNT | 一个无符号16位整数，表示报文Answer段中的回答记录数。 |
| NSCOUNT | 一个无符号16位整数，表示报文Authority段中的授权记录数。 |
| ARCOUNT | 一个无符号16位整数，表示报文Additional段中的附加记录数。 |



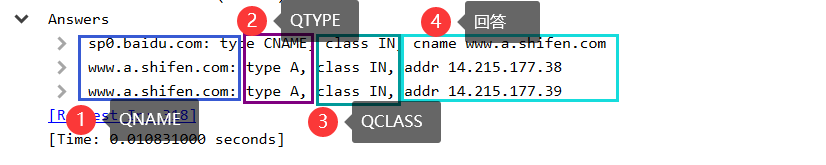
上图中Header部分各字段信息表明：这是一个ID为0x899a的查询消息，类型为标准查询，未被截断，要求递归查询，查询记录数为1.

* Question格式

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| QNAME | | | | | | | | | | | | | | | |
| QTYPE | | | | | | | | | | | | | | | |
| QCLASS | | | | | | | | | | | | | | | |

各字段含义说明如下表：

|  |  |
| --- | --- |
| **字段** | **说明** |
| QNAME | 表示所查询的域名。注意这里的域名以编码表示，如：  sp0 . baidu . com  **03** 73 70 30 **05** 62 61 69 64 75 **03** 63 6f 6d **00**  蓝色部分指示接下来多少字节是有效信息，以00结束，域名中的“.”没有被编码。 |
| QTYPE | 一个16位的字段，表明请求的类型，常用的有：  1：A记录（将主机名映射到主机的记录，返回IPv4地址）  2：NS记录（域的权限服务器）  5：CNAME记录（别名记录）  28：AAAA记录（将主机名映射到主机的记录，返回IPv6地址） |
| QCLASS | 一个16位的字段，表明请求的类（Class）。对于Internet中的请求，QCLASS值为IN（0x0001）。 |



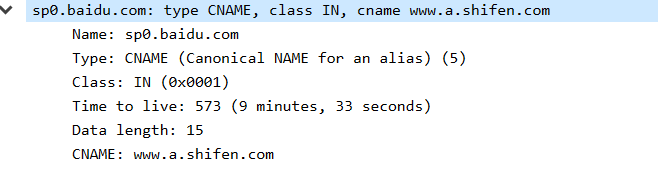
* Resource Record（RR）格式

DNS报文中的三个部分（Answer的详细信息、Authority、Additional）都遵循该格式：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| NAME | | | | | | | | | | | | | | | |
| TYPE | | | | | | | | | | | | | | | |
| CLASS | | | | | | | | | | | | | | | |
| TTL | | | | | | | | | | | | | | | |
| RDLENGTH | | | | | | | | | | | | | | | |
| RDATA | | | | | | | | | | | | | | | |

各字段含义说明如下表：

|  |  |
| --- | --- |
| **字段** | **说明** |
| NAME | 与该资源记录相关的域名 |
| TYPE | 一个16位的字段，表明RDATA中的数据含义。  值所对应的含义与QTYPE相同 |
| CLASS | 一个16位的字段，表明RDATA中的数据的类。  值所对应的含义与QCLASS相同 |
| TTL | 一个32位的字段，表明该记录进行一次缓存的时间间隔（在记录被废弃之前，单位为s）；若为0，则表明记录只适用于处理过程中，不应被缓存。 |
| RDLENGTH | 一个无符号16位整数，表明RDATA中数据的长度 |
| RDATA | 对资源的描述。不同的资源记录有不同的格式。  例如，对于A记录，RDATA是一个32位的IP地址。 |



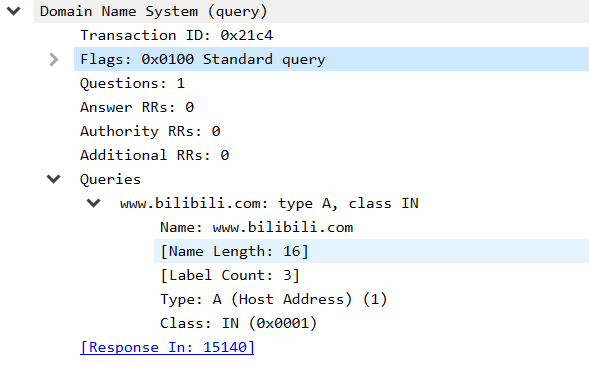
图为一条CNAME记录，映射关系为sp0.baidu.com→www.a.shifen.com

#### DNS工作过程

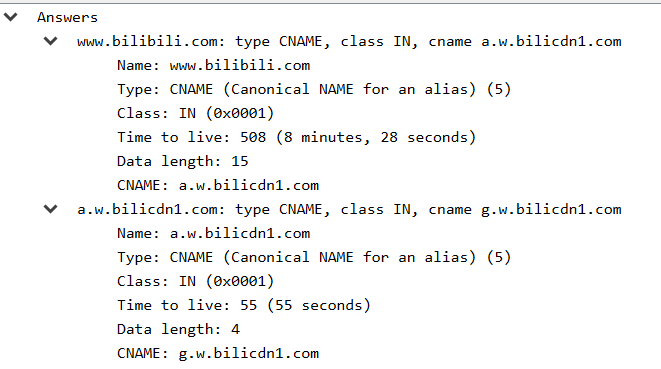
访问www.bilibili.com，Wireshark捕获DNS数据包如下图所示：



① 主机向本地域名服务器192.168.10.8发起查询，报文如下：



② 本地域名服务器响应，Answer中共有7条资源记录：



第一条CNAME记录：[www.bilibili.com→a.w.bilicdn1.com](http://www.bilibili.com→a.w.bilicdn1.com)

第二条CNAME记录：a.w.bilicdn1.com→g.w.bilicdn1.com

然后是五条A记录，g.w.bilicdn1.com可解析到5个不同IP：

139.159.227.6

110.43.33.147

110.43.34.72

139.9.62.5

139.159.246.60

如此便完成了从域名www.bilibili.com到IP的解析。

#### 异常情形

问题：有时候会捕获不到DNS解析过程。

复现：初次访问www.mokazure.moe时，Wireshark能捕获到DNS解析过程；但随后再次访问相同网站时，Wireshark捕获不到。

分析：猜测可能是因为在第一次访问之后，本机（浏览器或系统）进行了DNS缓存，第二次访问时直接从本机缓存中得到IP，不需要再与本地DNS服务器进行通信，因此Wireshark捕获不到解析过程。

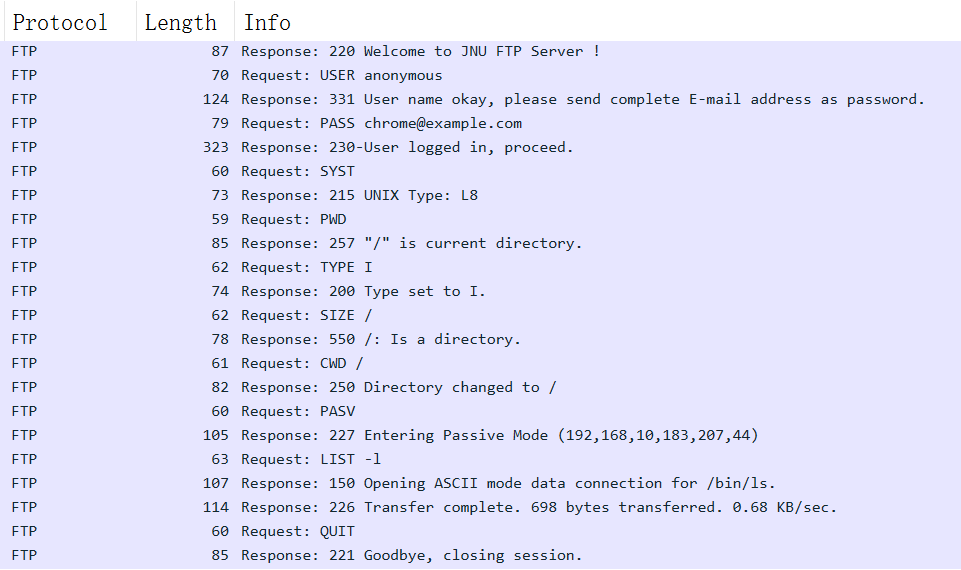
验证：基于上述分析，若在第一次访问之后清除本机缓存，再进行第二次访问，那么两次都应该可以捕获到DNS解析过程。通过重启浏览器的方式清除浏览器的DNS缓存，通过ipconfig/flushdns命令清除系统的DNS缓存。经过实验，以上两种清除操作若只执行其中一种，那么第二次访问仍然无法捕获，只有当两种操作都执行后，第二次访问才能捕获到DNS解析过程。

结论：DNS缓存既存在于浏览器中，也存在于系统中，域名解析将优先查找这两处的缓存；只有当需要解析的域名在这两处地方都没有缓存记录时，才会向本地DNS服务器通信请求解析。此外，浏览器和系统的DNS缓存都不是一直存在的，有设定的TTL，只有在TTL内发起第二次访问，才会无法捕获；若间隔一段时间再访问，不需要清除本机DNS缓存也能捕获解析过程。

### FTP协议分析

#### 协议分析

访问ftp://ftp.jnu.edu.cn，Wireshark捕获结果如下：



图中涉及的命令如下表所示：

|  |  |
| --- | --- |
| **命令** | **说明** |
| USER | USER NAME - 参数为用户名，进行用户认证。通常为建立控制连接之后的第一条命令 |
| PASS | PASSWORD - 参数为用户密码，该命令紧接在USER命令之后 |
| SYST | SYSTEM - 请求服务器操作系统类型 |
| PWD | Print Working Directory - 请求当前工作目录 |
| TYPE | Representation Type - 确定数据的传输方式，参数为标识数据类型的一个或多个ASCII字符，如I（Image），A（ASCII），E（EBCDIC）等 |
| SIZE | （RFC959没有提到这个命令，通过参数指定某个文件，获取文件大小） |
| CWD | Change Working Directory - 把当前目录改为远程文件系统的指定路径 |
| PASV | PASSIVE - 要求服务器端的数据传送进程监听一个数据端口（不是默认的数据端口）并等待连接。对该命令的响应包括服务器地址及正在监听的端口号。 |
| LIST | List - 让服务器发送一份列表 |
| QUIT | 终止连接 |

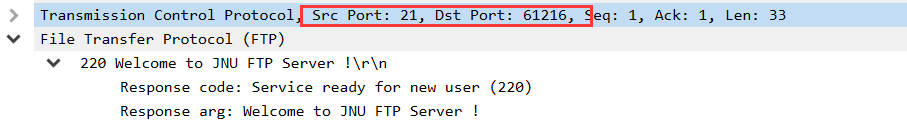
图中涉及的响应码如下表所示：

|  |  |
| --- | --- |
| **响应代码** | **说明** |
| 220 | 为新用户的服务就绪 |
| 331 | 用户名无误，要求密码 |
| 230 | 用户登录成功，继续 |
| 215 | 系统类型回复 |
| 257 | 路径名（“PATHNAME”）建立 |
| 200 | 成功 |
| 550 | 文件不可用 |
| 250 | 文件行为完成 |
| 227 | 进入被动模式（h1,h2,h3,h4,p1,p2）  （h1,h2,h3,h4,p1,p2为32位的地址（host address）和16位的端口号（port address），十进制表示） |
| 150 | 文件状态无误，准备打开数据连接 |
| 226 | 结束数据连接 |
| 221 | 服务器关闭控制连接 |

另响应码有一定格式表明响应的类型，如：

|  |  |
| --- | --- |
| **代码格式** | **说明** |
| 1yz | 成功，开始处理请求。通常会有第二条响应表示处理结果，在此之前不会处理新的请求 |
| 2yz | 成功，请求处理完成 |
| 3yz | 成功，但处理请求需要用户提供进一步信息，被暂时搁置以等待用户 |
| 4yz | 失败（暂时的），用户可以重新发起相同请求，再一次尝试 |
| 5yz | 失败（永久的），即使再一次发起相同请求也会失败，需要检查请求是否正确。如拼写错误或目录状态不符合要求等 |
| x0z | 语法 - 请求存在语法错误、不存在的命令或多余的命令 |
| x1z | 信息 - 请求需要用户提供进一步信息 |
| x2z | 连接 - 与控制连接或数据连接有关的响应 |
| x3z | 认证 - 与用户登录有关的响应 |
| x4z | 保留，暂未定义 |
| x5z | 文件系统 - 有关文件传输或其他行为的响应 |

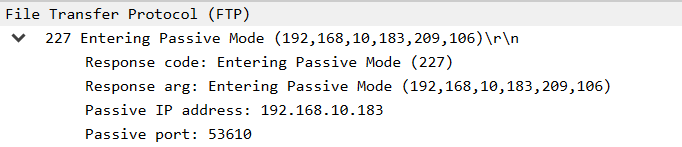
结合命令说明及响应说明，由抓包结果所知的FTP访问流程为：连接成功→发送用户名→发送密码→查看服务器系统信息（UNIX）→查看当前工作目录（/）→确定数据类型为I（Image）→查看根目录的大小（失败，因为不是文件）→改变工作目录（/）→进入被动模式→请求传输列表→终止连接。



由图中可知，建立**控制连接**的服务器端口为21，客户端端口为61216.

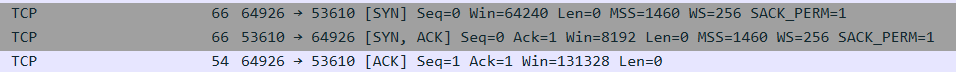
#### 被动模式

对于命令PASV，服务器给出的响应详细信息如图所示：

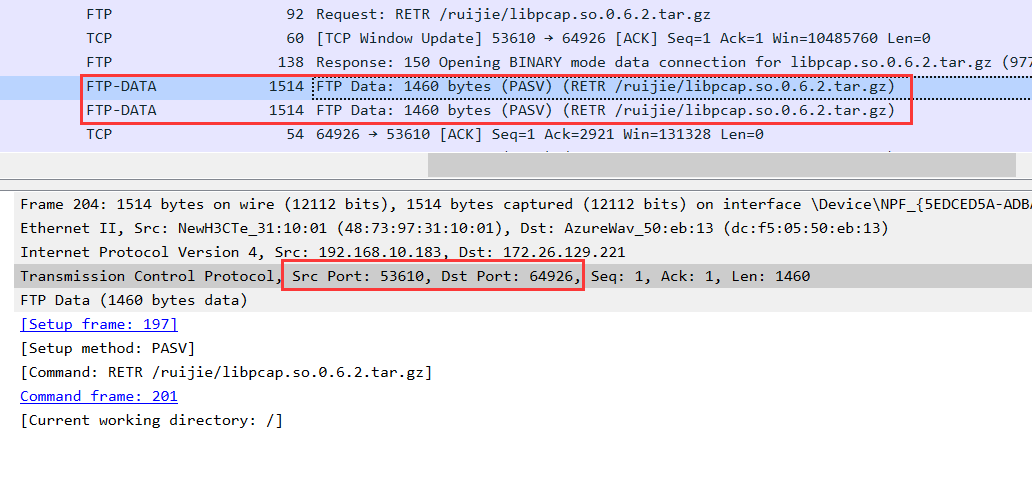


其中（192,168,10,183,207,44）依次为（h1,h2,h3,h4,p1,p2），转换后即服务器地址为192.168.10.183，端口号为53610（注意209对应二进制数是高八位，106对应二进制数是低八位，换算即可）。正如被动模式的工作原理，当客户端发出PASV命令后，服务器随机开放一个端口（1024以上）并告知客户端。

然后客户端用一个新端口64926与服务器告知的端口53610建立**数据连接**：



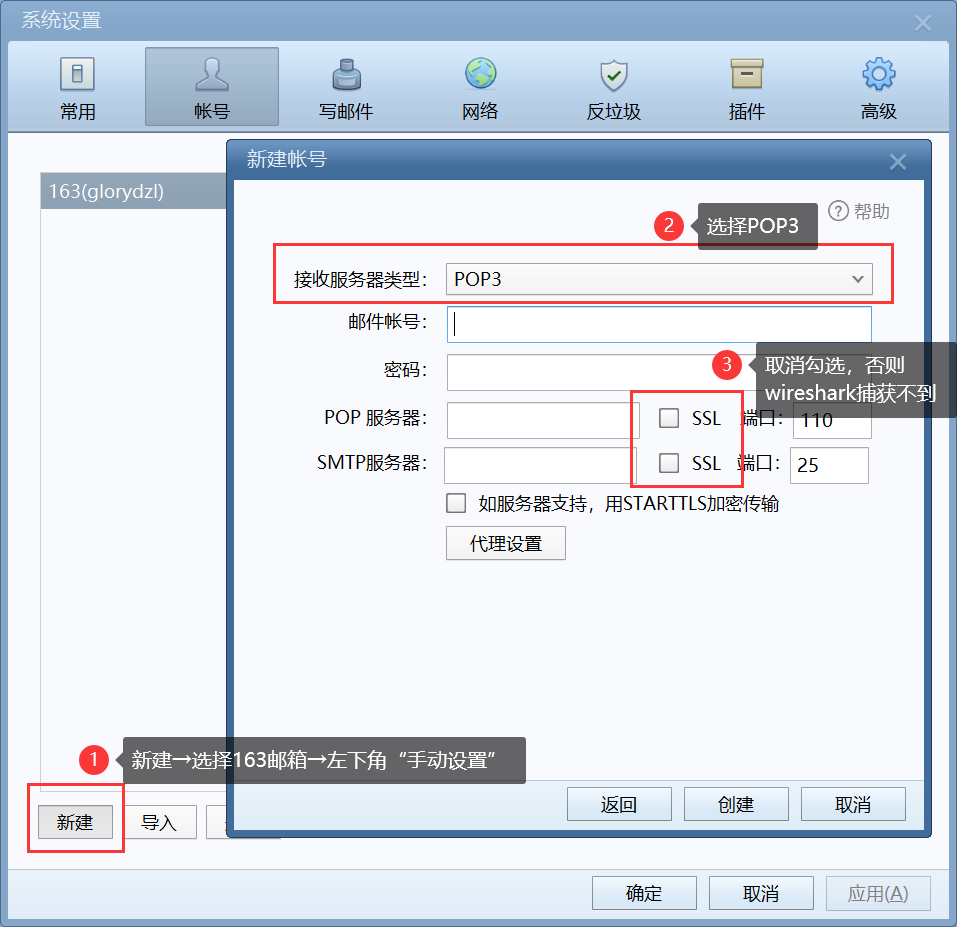
然后通过这个数据连接开始传输数据（53610→64926）：



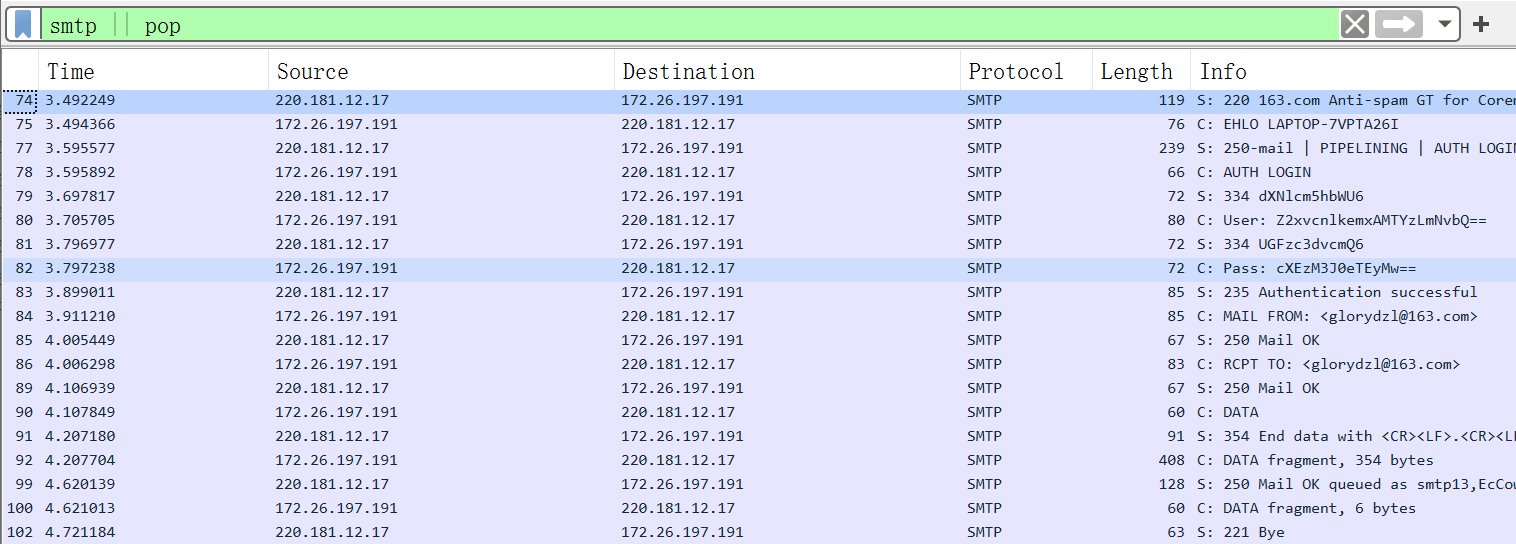
### SMTP和POP协议分析

#### 下载并设置Foxmail

（注：163邮箱需要手动开启POP3，先登录163邮箱进行设置，再用授权密码在Foxmail中登录，而不是用邮箱密码）



打开Wireshark，写邮件发送给自己，捕获结果如下：



#### SMTP协议分析

* 阶段一：建立连接

|  |  |  |
| --- | --- | --- |
| **协议类型** | **INFO** | **分析** |
| TCP | 64261 → 25 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK\_PERM=1 | SMTP客户端口就熟知端口25向SMTP服务器发送SYN标记的包，请求建立TCP连接 |
| TCP | 25 → 64261 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0 MSS=1460 SACK\_PERM=1 WS=128 | SMTP服务器发送对SYN包的确认包 |
| TCP | 64261 → 25 [ACK] Seq=1 Ack=1 Win=131328 Len=0 | SMTP客户收到确认包，再发送一个ACK确认包，通知服务器连接已建立 |
| SMTP | S: 220 163.com Anti-spam GT for Coremail System (163com[20141201]) | * 220 163.com为应答信息，表示服务器就绪 * 其他为服务器的欢迎信息 |
| SMTP | C: EHLO LAPTOP-7VPTA26I | * HELO为客户端向服务器表明用户身份 * EHLO为HELO的扩展，支持用户认证 |
| TCP | 25 → 64261 [ACK] Seq=66 Ack=23 Win=14720 Len=0 |  |
| SMTP | S: 250-mail | PIPELINING | AUTH LOGIN PLAIN | AUTH=LOGIN PLAIN | coremail 1Uxr2xKj7kG0xkI17xGrU7I0s8FY2U3Uj8Cz28x1UUUUU7Ic2I0Y2Ur34xFrUCa0xDrUUUUj | STARTTLS | 8BITMIME |  |
| SMTP | S: 334 dXNlcm5hbWU6 | 334表示接受，等待用户输入验证信息；“dXNlcm5hbWU6”是经过BASE64加密的“Username:” |
| SMTP | C: AUTH LOGIN | 客户端发送用户登录命令 |
| SMTP | C: User: Z2xvcnlkemxAMTYzLmNvbQ== | 客户端发送BASE64加密的用户名，解密后可知是邮箱地址“glorydzl@163.com” |
| SMTP | S: 334 UGFzc3dvcmQ6 | 334表示接受，等待用户输入验证信息；“UGFzc3dvcmQ6”是经过BASE64加密的“Password:” |
| SMTP | C: Pass: cXEzM3J0eTEyMw== | 客户端发送BASE64加密的密码 |
| SMTP | S: 235 Authentication successful | 235表示接受，认证成功 |

* 阶段二：邮件传送

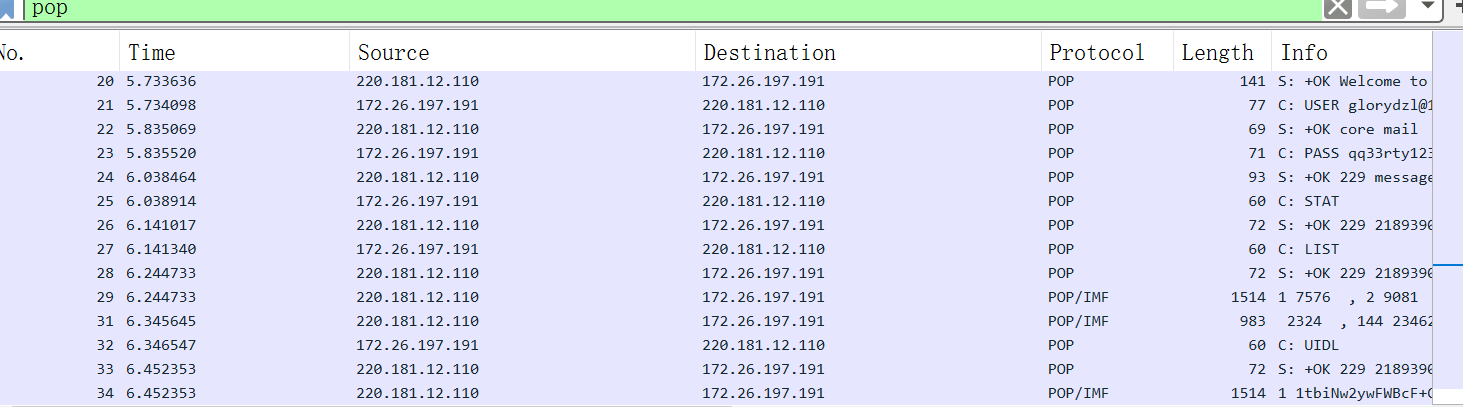
|  |  |
| --- | --- |
| **INFO** | **分析** |
| C: MAIL FROM: [<glorydzl@163.com>](mailto:<glorydzl@163.com>) | 声明邮件来源邮箱 |
| S: 250 Mail OK | 250表示要求的邮件操作完成 |
| C: RCPT TO: [<glorydzl@163.com>](mailto:<glorydzl@163.com>) | 声明邮件目的邮箱 |
| S: 250 Mail OK | 250表示要求的邮件操作完成 |
| C: DATA | 请求发送数据命令 |
| S: 354 End data with <CR><LF>.<CR><LF> | 354表示开始邮件输入。以一行.结束输入 |
| C: DATA fragment, 354 bytes |  |
| S: 250 Mail OK queued as smtp13,EcCowAD3FY5t9mRh5WGrRA--.20094S2 1634006638 | 250表示要求的邮件操作完成，邮件已进入发送队列 |
| C: DATA fragment, 6 bytes | 一行“QUIT”，请求与服务器断开连接 |
| S: 221 Bye | 221表示服务关闭 |

* 阶段三：释放连接

|  |  |
| --- | --- |
| **INFO** | **分析** |
| 64261 → 25 [FIN, ACK] Seq=1465 Ack=455 Win=130816 Len=0 | 客户端主动关闭与服务器的数据传送 |
| 25 → 64261 [FIN, ACK] Seq=464 Ack=1465 Win=18688 Len=0 | 服务器收到并返回确认 |
| 25 → 64261 [ACK] Seq=465 Ack=1466 Win=18688 Len=0 | 服务器主动关闭与客户端的数据传送 |
| 64261 → 25 [RST, ACK] Seq=1466 Ack=464 Win=0 Len=0 | 客户端收到并返回确认 |

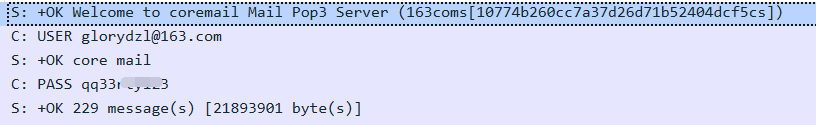
#### POP协议分析

在Foxmail客户端点击“收取”，Wireshark嗅探结果如图：



POP3协议中有三种状态，认证状态，处理状态（TRANSACTION State），和更新状态。

* 认证状态

① 客户端与服务器成功建立TCP连接后，服务器发送一条欢迎信息，POP3会话处于认证状态。

② 图中客户端向服务器依次发送用户名及密码进行认证。

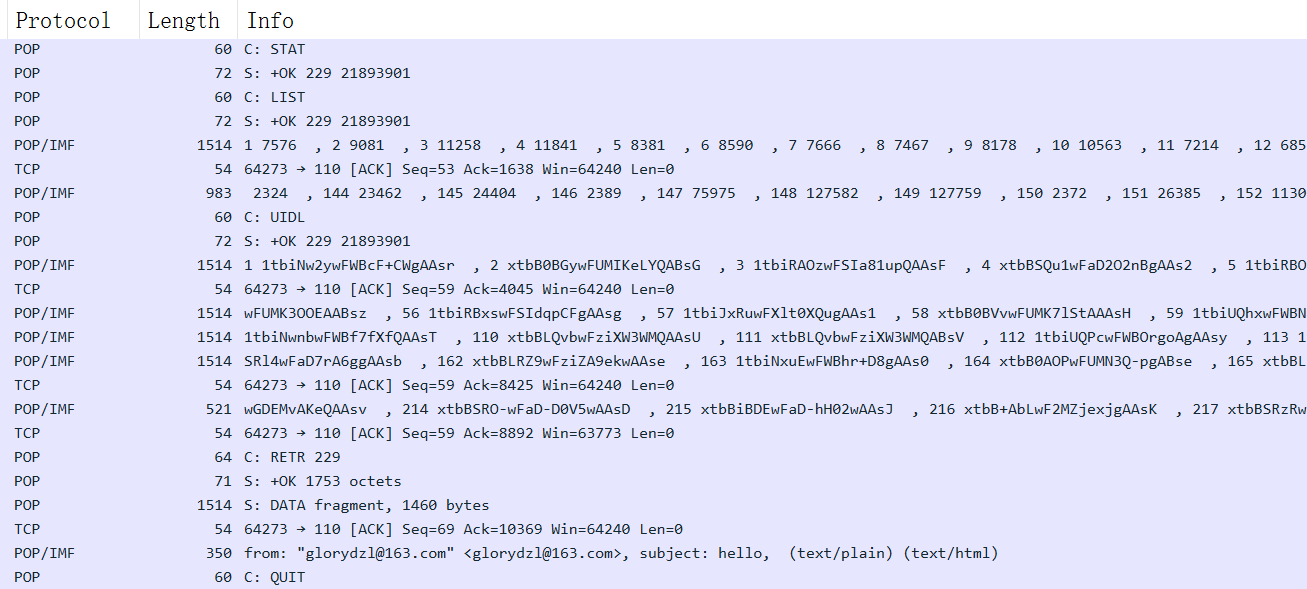
③ 认证成功，转入处理状态。

服务器响应的第一行以一个状态码“+OK”或“-ERR”开头，分别表示操作成功或失败，后面再接一些文本说明。

图中涉及的命令如下表所示：

|  |  |  |  |
| --- | --- | --- | --- |
| **命令** | **参数** | **在何种状态下使用** | **描述** |
| USER | Username | 认证 | 当客户端使用“USER”和“PASS”命令组合进行认证时，必须先发送“USER”命令 |
| PASS | Password | 认证 | 发送密码让服务器确认并授权 |

* 处理状态



①客户端认证成功且服务器锁定并打开对应的邮筒（Maildrop）后，POP3会话处于处理状态。

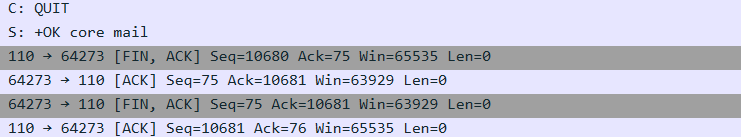
②服务器持续响应客户端发送的命令，直到收到“QUIT”命令。

③客户端发送“QUIT”命令，POP3会话转入更新状态。

图中涉及的命令如下表所示（Message不知怎么准确翻译，也许是邮件吧）：

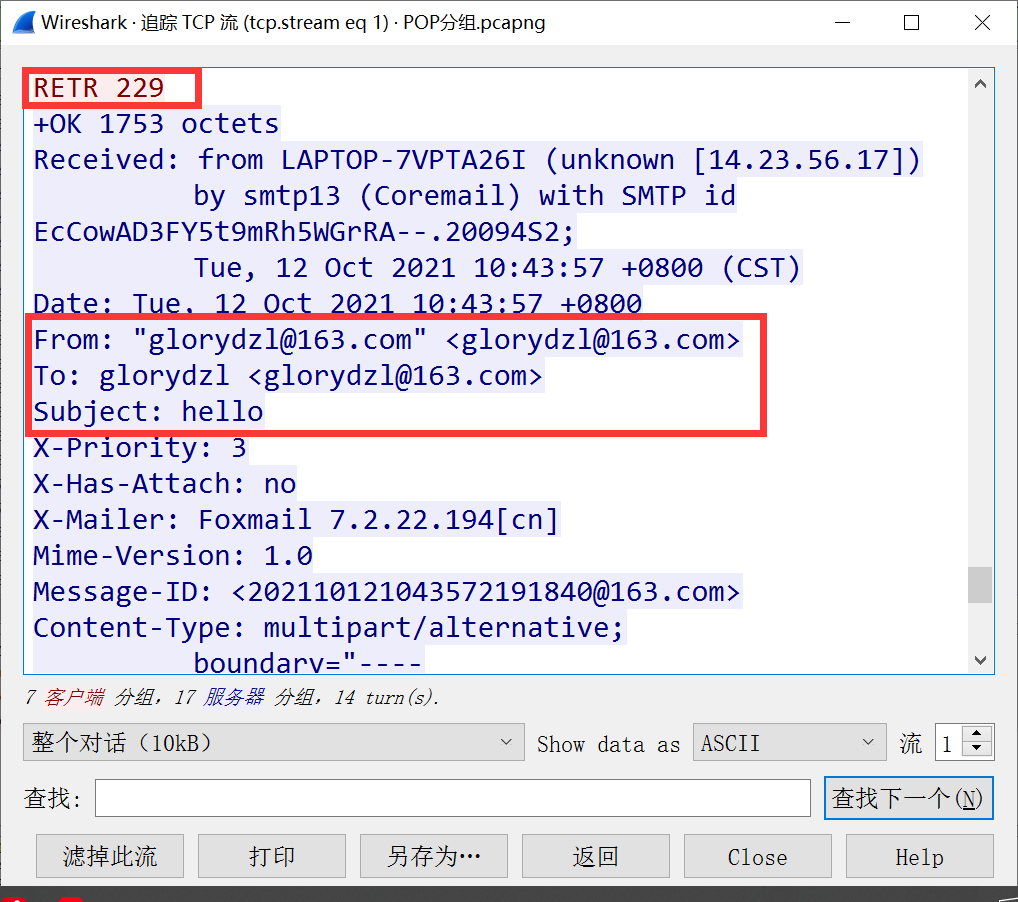
|  |  |  |  |
| --- | --- | --- | --- |
| **命令** | **参数** | **在何种状态下使用** | **描述** |
| STAT | 无 | 处理 | 服务器响应邮筒信息，也叫drop listing，包含邮筒中的message数量、以8bit表示的邮筒大小。 |
| LIST | Message number（可选） | 处理 | 若无参数，服务器会发出多行响应，第一行以“+OK”开始，余下各行为邮筒中每条message的number及以两个字节表示的准确大小，每条一行。 |
| UIDL | Message number（可选） | 处理 | 若无参数，服务器会发出多行响应，第一行以“+OK”开始，余下各行为邮筒中每条message的number及唯一ID，每条一行。 |
| RETR | Message number（必须） | 处理 | 检索指定的message。服务器响应为message number对应的Message的全部内容。 |
| QUIT | 无 | 任何 | POP3会话转入更新状态，服务器移除邮筒中被标记为“删除”的message，释放在处理状态下申请的资源并关闭TCP连接。 |

* 更新状态

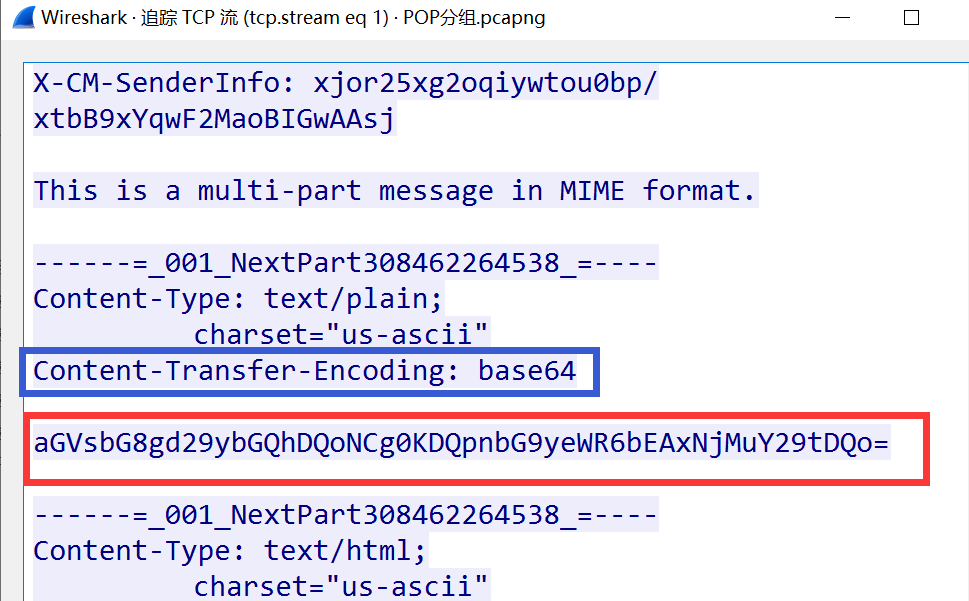


客户端发出QUIT命令，服务器响应“+OK”后发起TCP连接关闭请求（110是服务器的POP3熟知端口），四次挥手后关闭TCP连接。

此外，在Wireshark中追踪TCP流可以看到：



也就是说Message number为229的Message就是实验中所发的那封邮件。往下可找到base64加密的邮件内容（右图为解密结果）：

## 实验总结

本次实验分析了HTTP、DNS、FTP、SMTP和POP3五个协议。HTTP定义了万维网文档在浏览器及万维网服务器之间的请求与传送，由工作过程分析可以知道浏览器显示页面的过程，其实就是浏览器向服务器多次请求HTML、CSS、图片等资源的过程；DNS允许互联网用户以域名代替具体的IP，同时还有均衡负载的作用。由于DNS本质上是域名到IP的映射，只需要得到对应记录就算成功，因此可以将记录缓存在本地DNS服务器、缓存在系统、缓存在浏览器，避免每次解析都舍近求远，既缓解了根服务器等的压力，也能加快解析速度；FTP则定义了文件传输方式，通过建立控制连接、数据连接这两条TCP连接完成文件数据的传输。FTP的被动模式是服务器打开端口，被客户端所连接；主动模式则是客户端主动去连接服务器端口；POP3和SMTP分别定义了邮件的收与发的方式，都需要进行用户认证以及一系列命令与服务器交互。做实验的时候使用了Foxmail客户端，收与发都是在客户端界面一键完成，似乎还可以在Linux下纯手动操作，但因为时间原因没有尝试。

本次实验报告在讲到HTTP之前就开始写了，断断续续写了将近三个星期，是花费时间较多但也是收获最大的一次实验。需要分析的数据包很快就能捕获，但协议分析需要一直查找资料，看很多篇博文，还要从其中筛选出有用的、想要了解的信息。这个过程其实相当浪费时间，因为大多数博文要么是简单的复制粘贴产物、要么排版极其混乱（虽说本实验报告的排版也好不到哪里去）、要么内容零碎，总之多数时间都耗费在浏览和整合上了。但凡能在短时间内搜索到一篇高质量博文也不至于如此。

直到看到一篇博文给出了POP3的RFC链接，才转而通过协议文档去了解协议。协议的工作过程很容易百度了解，但具体到Wireshark所给出的某一个数据包的某一个命令或状态码，却不容易看到详细的整理。而RFC文件则都给出了详尽的说明。当然一个翻译得好的中文版文档也同样难找，而且基本上是节选，如果不给原文术语的话也很难和Wireshark的结果一一对应起来，所以最好的方式还是直接阅读RFC文件原文。虽然翻译需要一点时间，还要与一些博文对照补充，但了解协议以及写实验报告的速度总体上加快了。