实验 4 应用层协议分析

1.1 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。

1.2 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。

注：HTTP 和 FTP 两个协议二选一。

1.3 HTTP 协议概述[1]

超文本传输协议（HyperText Transfer Protocol，HTTP）是一种用于分布式、协作式和超媒体信息系统的应用层协议。HTTP 是 WWW 的数据通信基础。

HTTP 的发展是由蒂姆·伯纳斯-李于 1989 年在欧洲核子研究组织（CERN）所发起。HTTP 的标准制定由万维网协会（World Wide Web Consortium，W3C）和互联网工程任务组（Internet Engineering Task Force，IETF）进行协调，最终发布了一系列的 RFC，其中最著名的是 1999 年 6 月公布的 RFC 2616，定义了 HTTP 协议中现今广泛使用的一个版本——HTTP 1.1。

2014 年 12 月，互联网工程任务组（IETF）的 Hypertext Transfer Protocol Bis（httpbis）工作小组将 HTTP/2 标准提议递交至 IESG 进行讨论，于 2015 年 2 月 17 日被批准。 HTTP/2 标准于 2015 年 5 月以 RFC 7540 正式发表，取代 HTTP 1.1 成为 HTTP 的实现标准。

[1]这部分内容参考了 CSDN 博主「有抱负的小狮子」的原创文章，有改动。

1.3.1 HTTP 协议概况

HTTP 是一个客户端终端（用户）和服务器端（网站）请求和应答的标准。客户端通常是浏览器，发起一个 HTTP 请求到服务器上指定端口（默认 80）。客户端称为用户代理（user agent）。应答服务器上存储着一些资源，比如 HTML 文件和图像。称为源服务器（origin server）。在用户代理和源服务器中间可能存在多个“中间层”，比如代理服务器、网关或者隧道（tunnel）。

尽管 TCP/IP 协议是互联网上最流行的应用，HTTP 协议中，并没有规定必须使用它或它支持的层。事实上，HTTP 可以在任何互联网协议上，或其他网络上实现。HTTP 假定其下层协议提供可靠的传输。因此，任何能够提供这种保证的协议都可以被其使用。

通常，HTTP 客户端发起一个请求，创建一个到服务器指定端口（默认 80）的 TCP 连接。HTTP 服务器则在该端口监听客户端的请求。一旦收到请求，服务器会向客户端返回一个状态，比如"HTTP/1.1 200 OK"，以及返回的内容，如请求的文件、错误消息、或者其它信息。HTTP 协议的特点是无状态，就是说 HTTP 协议本身不会对请求和响应之间的通信状态做保存。建立在 HTTP 协议之上的应用如果需要状态，可以使用 cookie 来完成。

1.3.2 HTTP 协议的请求

HTTP 协议由 HTTP 请求和 HTTP 响应组成，当在浏览器中输入网址访问某个网站（或点击链接）时， 你的浏览器会将你的请求封装成一个 HTTP 请求发送给服务器站点，服务器接收到请求后会组织响应数据封装成一个 HTTP 响应返回给浏览器。

HTTP 请求包括请求行、请求头、请求体，HTTP 响应包括响应行、响应头、响应体。

HTTP 请求的例子如下图所示：

请求行必须在 http 请求格式的第一行。请求头从第二行开始，到第一个空行结束。请求头和请求体之间存在一个空行（如图中 Cookie 行和 name 行之间）。请求体是可选的，可以为空。

请求头是后面几行（直到 Cookie 行），

请求行的格式：请求方式 资源路径 协议/版本，例如：POST /chapter17/user.html HTTP/1.1。**请求方式**很多，如 GET，POST，HEAD 等等。

GET 请求将请求参数追加在 URL 后面，很简单，没有请求体。但 URL 长度限制 GET 请求方式数据的大小。一般的 HTTP 请求大多都是 GET。常见 GET 请求:地址栏直接访问、<a href="">、<img src="">等。

POST 请求：请求参数在请求体处，请求数据大小没有限制，只有表单设置为 method=“POST”才是 POST 请求，其他大都是 GET 请求。

HEAD 请求：跟 GET 相似，不过服务端接收到 HEAD 请求时只返回响应头，不发送响应内容。所以，如果只需要查看某个页面的状态时，用 HEAD 更高效，因为省去了传输页面内容的时间。

其他还有 DELETE，OPTIONS，PUT，TRACE，CONNECT 请求等。

请求头从第二行开始，通常以键值对 key:value 方式传递数据。例如：

```
POST /vk/app/rest/ddp/findModelByType HTTP/1.1
User-Agent: Fiddler
Host: 39.108.107.149:8080
Content-Length: 11

name=城市
```

key 为规范的固定值，value 为 key 对应的取值，通常是一个值，可能是一组。

HTTP 请求报文头属性

常见请求头

Referer：表示这个请求是从哪个 url 跳过来的。通过百度来搜索淘宝网，那么在进入淘宝网的请求报文中，Referer 的值就是 www.baidu.com。如果是直接访问就不会有这个头。常用于防盗链。

Accept：告诉服务端该请求所能支持的响应数据类型，即 MIME 类型。

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
```

MIME 格式：大类型/小类型[;参数]

例如:text/html, text/css，text/javascript，image/*等。

If-Modified-Since:浏览器通知服务器，本地缓存的最后变更时间。与另一个响应头组合控制浏览器页面的缓存。

Cookie：客户端的 Cookie 通过这个属性传给服务端。

User-Agent:浏览器通知服务器，客户端浏览器与操作系统相关信息

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36

Connection:表示客户端与服务连接类型；Keep-Alive 表示持久连接，close 已关闭。

Host:请求的服务器主机名

Content-Length:请求体的长度

POST /vk/app/rest/ddp/iModelServiceImpl/findModelByType HTTP/1.1

User-Agent: Fiddler

Host: 39.108.107.149:8080

Content-Length: 11

name=城市

Content-Type:请求的与实体对应的 MIME 信息。如果是 post 请求,会有这个头,默认值为 application/x-www-form-urlencoded，表示请求体内容使用 url 编码。

Accept-Encoding：浏览器通知服务器，浏览器支持的数据压缩格式。如 GZIP 压缩

Accept-Encoding: gzip, deflate

Accept-Language：浏览器通知服务器，浏览器支持的语言。各国语言(国际化 i18n)

Accept-Language: zh-CN,zh;q=0.9

Cache-Control：指定请求和响应遵循的缓存机制

对缓存进行控制，如一个请求希望响应返回的内容在客户端要被缓存一年，或不希望被缓存就可以通过这个报文头达到目的。

Cache-Control: no-cache

请求体

当请求方式是 POST 时，请求体有请求的参数，格式如下：

username=zhangsan&password=123

1.3.3 HTTP 协议的应答

HTTP 应答（也称响应）也由三部分组成（响应行+响应头+响应体）。

响应行：

①报文协议及版本；

例如：HTTP/1.1 200 OK

②状态码及状态描述；

状态码：由 3 位数字组成，第一个数字定义了响应的类别

1xx：指示信息，表示请求已接收，继续处理

2xx：成功，表示请求已被成功接受，处理。

200 OK：客户端请求成功

204 No Content：无内容。服务器成功处理，但未返回内容。一般用在只是客户端向服务器发送信息，而服务器不用向客户端返回什么信息的情况。不会刷新页面。

206 Partial Content：服务器已经完成了部分 GET 请求（客户端进行了范围请求）。响应报文中包含 Content-Range 指定范围的实体内容

3xx：重定向

301 Moved Permanently：永久重定向，表示请求的资源已经永久的搬到了其他位置。

302 Found：临时重定向，表示请求的资源临时搬到了其他位置

303 See Other：临时重定向，应使用 GET 定向获取请求资源。303 功能与 302 一样，区别只是 303 明确客户端应该使用 GET 访问

307 Temporary Redirect：临时重定向，和 302 有着相同含义。POST 不会变成 GET

304 Not Modified：表示客户端发送附带条件的请求（GET 方法请求报文中的 IF…）时，条件不满足。返回 304 时，不包含任何响应主体。虽然 304 被划分在 3XX，但和重定向没有关系。

4xx：客户端错误

400 Bad Request：客户端请求有语法错误，服务器无法理解。

401 Unauthorized：请求未经授权，这个状态代码必须和 WWW-Authenticate 报头域一起使用。

403 Forbidden：服务器收到请求，但是拒绝提供服务

404 Not Found：请求资源不存在。比如，输入了错误的 url

415 Unsupported media type：不支持的媒体类型

5xx：服务器端错误，服务器未能实现合法的请求。

500 Internal Server Error：服务器发生不可预期的错误。

503 Server Unavailable：服务器当前不能处理客户端的请求，一段时间后可能恢复正常，

响应头：

③响应头，也是由多个属性组成；也是用键值对 k: v 表示。

服务器通过响应头来控制浏览器的行为，不同的头浏览器操作不同。

常见响应头	描述
Location	指定响应的路径，需要与状态码 302 配合使用，完成跳转。
Content-Type	响应正文的类型（MIME 类型） 取值：text/html;charset=UTF-8
Content-Disposition	通过浏览器以下载方式解析正文 取值：attachment;filename=xx.zip
Set-Cookie	与会话相关技术。服务器向浏览器写入 cookie
Content-Encoding	服务器使用的压缩格式 取值：gzip
Content-length	响应正文的长度
Refresh	定时刷新，格式：秒数;url=路径。url 可省略，默认值为当前页。 取值：3;url=www.itcast.cn　//三秒刷新页面到 www.itcast.cn
Server	服务器名称
Last-Modified	服务器通知浏览器，文件的最后修改时间。与 If-Modified-Since 一起使用。
Cache-Control	响应输出到客户端后，服务端通过该报文头属告诉客户端如何控制响应内容的缓存。常见的取值有 private、public、no-cache、max-age，no-store，默认为 private。缓存时间为 31536000 秒（365 天）

响应体：

④响应体，就是服务器发送给浏览器的正文，即我们真正要的内容（如网页，图片等）；

响应体，响应体是服务器回写给客户端的页面正文，浏览器将正文加载到内存，然后解析渲染显示页面内容

1.4 FTP 协议概述

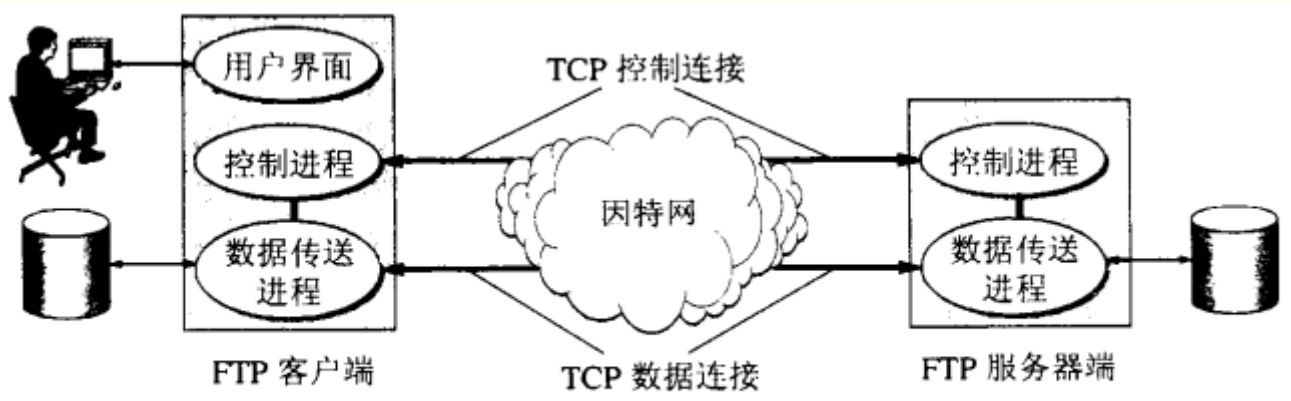
文件传输协议 FTP(File Transfer Protocol)是 Internet 早期使用最广泛的文件传输协议。FTP 使用交互式的访问，允许客户指定文件的类型和格式(如指明是否使用 ASCII 码)，并允许文件具有存取权限(如访问文件的用户必须经过授权，并输入有效的口令)。

1.4.1 FTP 基本工作原理

FTP 只提供文件传送的一些基本服务，它使用 TCP 可靠地运输服务。FTP 使用客户端-服务器模型，一个 FTP 服务器进程可以为多个客户进程提供服务。通常 FTP 服务器有两大部分组成：一个主进程，负责接受新的请求；还有若干从属进程，负责处理单个请求。主进程工作步骤：

1. 打开熟知端口(21)，使客户进程能够连接上
2. 等待客户进程发送连接请求
3. 启动子进程处理客户进程发送的连接请求，子进程处理完请求后结束。子进程在运行期间可能根据需要可创建其他一些子进程
4. 回到等待状态，继续接受其他客户进程发起的请求，主进程与子进程的处理是并发进行的

典型的 FTP 应用会用到两种连接：控制连接和数据连接。FTP 控制连接在整个会话期间都保持打开，只用来发送 FTP 命令及其应答。当客户进程向服务器发送连接请求时，连接服务器进程的熟知端口 21，就是一个控制连接。当双方需要交换数据（如传文件，列表文件等）时，这些数据的交换要在数据连接中传输。所以双方还要协商建立一个数据连接。



FTP 使用的两个 TCP 连接

1.4.2 FTP 的数据表示

FTP 协议规定了控制协议传送与存储的多种选择，在以下 4 个方面必须做出一个选择。

- 文件类型：ASCII 码文件(默认的)/ 二进制的文件类型
- 格式控制：该选项针对 ASCII 类型文件适用，非打印(默认选择，文件中不包含垂直格式信息)/ 远程登录格式控制
- 结构：文件结构(默认选择，文件被认为是一个连续的字节流)/ 记录结构
- 传输方式：流方式(模式选择，文件以字节流方式传输，对于文件结构，发方在文件尾提示关闭数据连接，对于记录结构，有专用的两字节序列码记录结束和文件结束)/ 块方式(文件以一系列块来传送，每块前面有一个或多个首部字节)/ 压缩方式

1.4.3 FTP 的命令和应答

FTP 协议的命令和应答在客户和服务器的控制连接上以 ASCII 码形式传送。这就要求在每行结尾都要返回 CR、LF 对（也就是每个命令或每个应答）。这些命令都是 3 或 4 个字节的大写 ASCII 字符，其中一些带选项参数。从客户向服务器发送的 FTP 命令超过 30 种。下图是比较常用的几种命令：

命　令	说　明
ABOR	放弃先前的FTP命令和数据传输
LIST *filelist*	列表显示文件或目录
PASS *password*	服务器上的口令
PORT *n*1,*n*2,*n*3,*n*4,*n*5,*n*6	客户端IP地址（*n*1.*n*2.*n*3.*n*4）和端口（*n*5×256+*n*6）
QUIT	从服务器注销
RETR *filename*	检索（取）一个文件
STOR *filename*	存储（放）一个文件
SYST	服务器返回系统类型
TYPE *type*	说明文件类型：A表示ASCII码，I表示图像
USER *username*	服务器上用户名

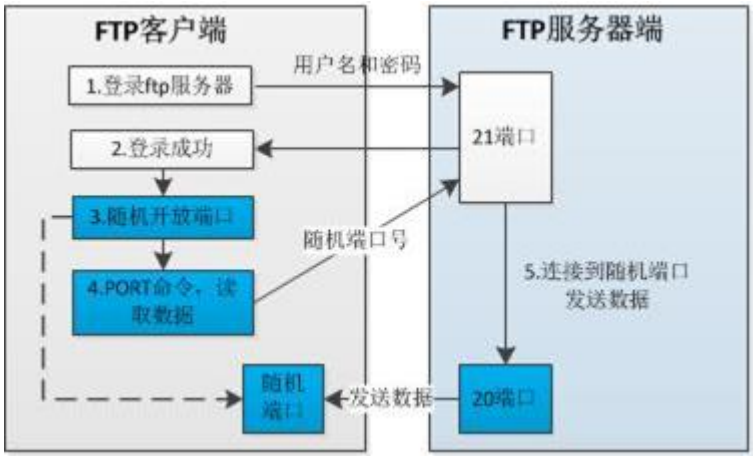
应答都是 ASCII 码形式的 3 位数字，并跟有报文选项。其原因是软件系统需要根据数字代码来决定如何应答，而选项串是面向人工处理的。由于客户通常都要输出数字应答和报文串，一个可交互的用户可以通过阅读报文串（而不必记忆所有数字回答代码的含义）来确定应答的含义。

应答	说　　明
1yz	肯定预备应答。它仅仅是在发送另一个命令前期待另一个应答时启动
2yz	肯定完成应答。一个新命令可以发送
3yz	肯定中介应答。该命令已被接受，但另一个命令必须被发送
4yz	暂态否定完成应答。请求的动作没有发生，但差错状态是暂时的，所以命令可以过后再发
5yz	永久性否定完成应答。命令不被接受，并且不再重试
x0z	语法错误
x1z	信息
x2z	连接。应答指控制或数据连接
x3z	鉴别和记帐。应答用于注册或记帐命令
x4z	未指明
x5z	文件系统状态

1.4.4 FTP 数据连接的工作模式

FTP 有两种工作模式，分别是主动模式(PORT)和被动模式(PASV)两种模式，这两种模式是按照 FTP 服务器的“角度”来说的，更通俗一点说就是：在传输数据时，如果是服务器主动连接客户端，那就是主动模式；如果是客户端主动连接服务器，那就是被动模式。

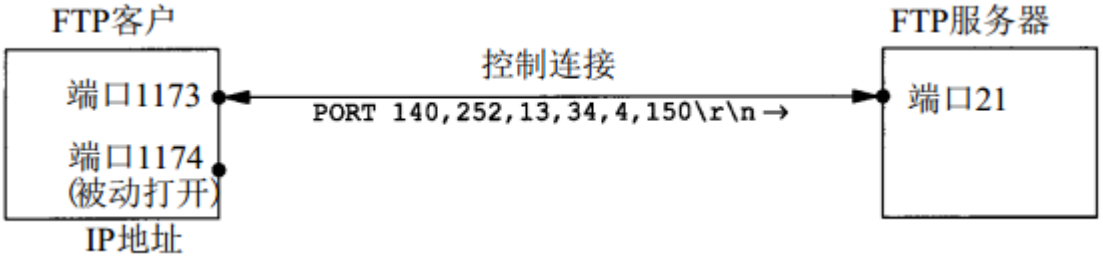
PORT 中文称为主动模式，工作的原理： FTP 客户端连接到 FTP 服务器的 21 端口，发送用户名和密码登录，登录成功后要 list 列表或者读取数据时，客户端随机开放一个端口（1024 以上），发送 PORT 命令到 FTP 服务器，告诉服务器客户端采用主动模式并开放端口；FTP 服务器收到 PORT 主动模式命令和端口号后，通过服务器的 20 端口和客户端开放的端口连接，发送数据，原理如下图：



主动模式其一般过程如下：

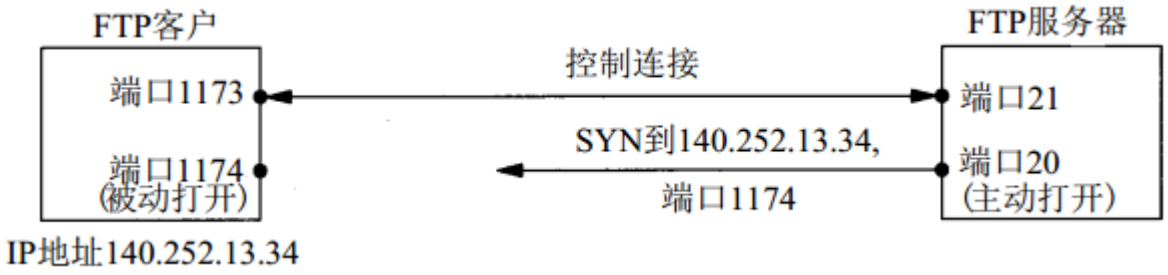
1. 客户端监听在一个临时端口号，使用 PORT 命令从控制连接上把端口号发向服务器。

2. 服务器接收到端口号，使用端口 20 向客户端的这个端口发送一个连接请求，建立起数据连接。

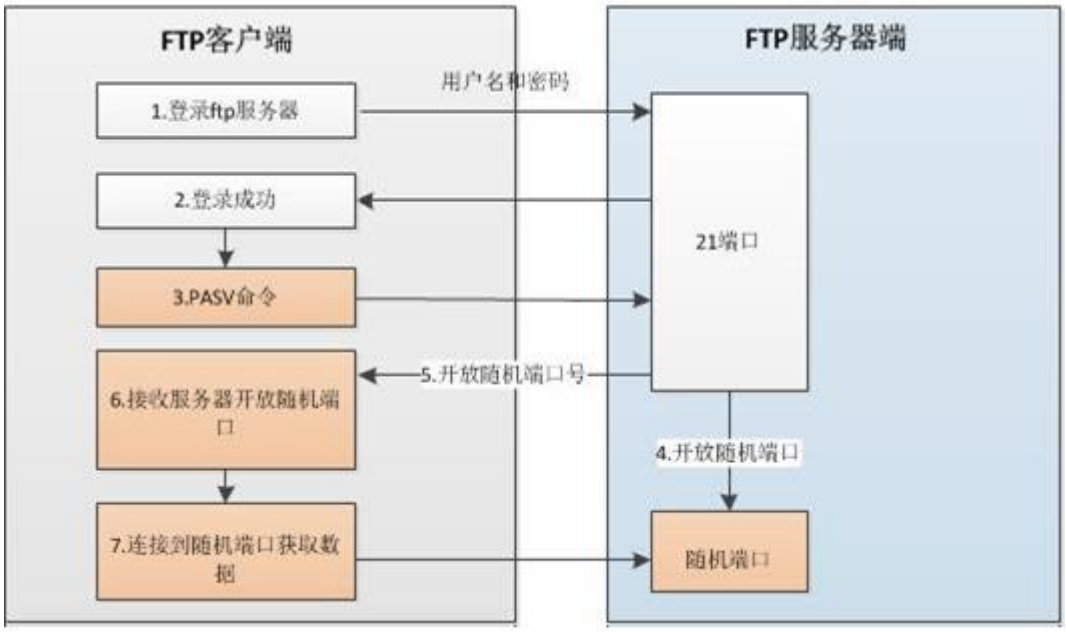


上图给出了第 1 步执行时的连接状态。假设客户用于控制连接的临时端口是 1173，客户用于数据连接的临时端口是 1174。客户发出的 PORT 命令，其参数是 6 个 ASCII 编码的十进制数字，它们之间由逗点隔开。前面 4 个数字指明客户上的 IP 地址，服务器将向它发出主动打开（本例是 140.252.13.34），而后两位指明 16 bit 端口地址。由于 16 bit 端口地址是从这两个数字中得来，所以其值在本例中就是 4×256+150=1174。

下图给出了服务器向客户所在数据连接端发起主动打开时的连接状态。服务器的端点是端口 20。



被动模式其一般过程如下（参考下图）：



1. 客户端使用 PASV 命令从控制连接上告诉服务器，希望用被动模式传输数据。

2. 服务器接收到 PASV 命令，监听在某个随机的端口上，并以应答（比如 227 Entering Passive Mode (202,117,1,2,172,17).）告诉客户端该端口号（比如 44049）。

3. 客户端新建立一个 TCP 连接到服务器的 44049 端口。

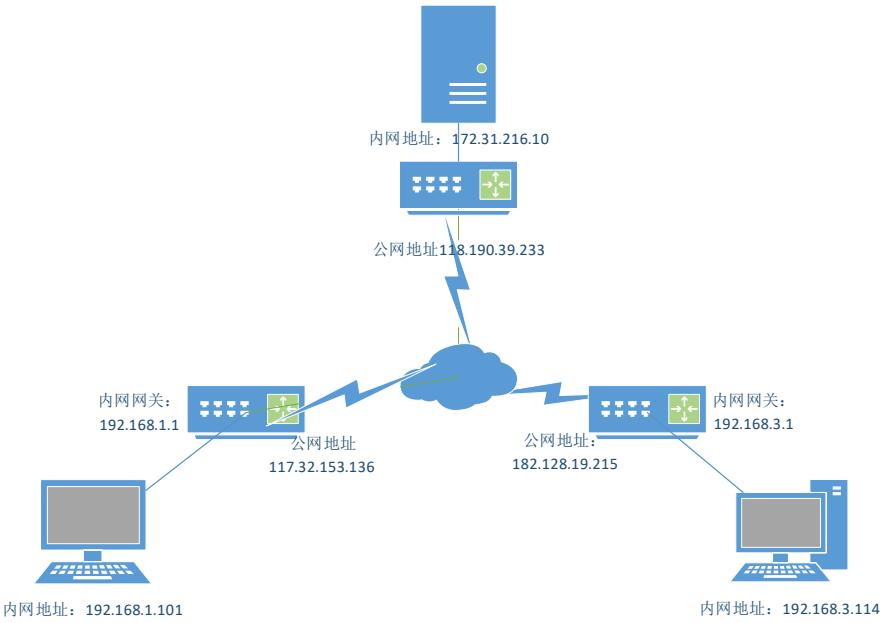
今天的互联网中大部分客户端机器都藏在 NAT 和防火墙后，所以主动模式通常不能工作，需要使用被动模式来建立连接。如果服务器也是在 NAT 盒子后面，那么还需要扩展的被动模式（命令是 EPSV）来完成数据连接建立。

1.5 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。

1.6 实验网拓扑结构

以各组现有网络实际情况为准，标注内网、公网地址。



1.7 HTTP 协议分析

1.7.1 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.github.com, www.unb.br；

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。

1.7.2 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。

1.7.3 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。

1.8 FTP 协议分析

1.8.1 FTP 协议的分析

步骤 1：在远程的云服务器上开启 ftp 服务，并在云服务器控制台把 21 端口开放。在云服务器上运行报文截获工具（如 Linux 的 tcpdump，Windows 的 Wireshark）截获 FTP 报文。

步骤 2：在计算机终端上运行 Wireshark 截获报文，使用 IE 浏览器来访问该 FTP 服务器。比如在地址栏输入 ftp://xxx.xxx.xxx.xxx/ 其中 xxx.xxx.xxx.xxx 是该云服务器的 IP 地址。

步骤 3：进入 FTP 服务器中的某个目录中，下载一个文件。结束后，停止报文截获。

分析该 FTP 的过程。注意对照服务器和客户端的一起分析，注意 NAT 的影响。观察是否使用了被动模式，如果是主动模式并且工具允许，使用被动模式再做一次下载，并分析。

如果 FTP 不能正常工作，请仔细抓包并分析原因。NAT 穿越问题可能是原因之一。

1.8.2 使用 ncat 工具来访问 FTP 服务

步骤 1：提前把用到的 FTP 命令准备好（写在 notepad++中）。

步骤 2：在云服务器上运行报文截获工具准备截获 FTP 报文。在命令窗口执行 ncat –C xxx.xxx.xxx.xxx 21，ncat 连接上 FTP 服务器后，根据协议输入合适的命令。其中 xxx.xxx.xxx.xxx 为云服务器地址。

步骤 3：解析完毕后停止报文截获。把过程整理记录到报告中。

在 ncat 模拟 FTP 客户端的过程中，请查看服务器的文件列表，并下载一个不大的文本文件。过程中，必要时用 netstat 命令观察双方的（新开）端口监听情况。

1.9 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。

（2）NAT 对 FTP 传输的影响，比较 HTTP 与 FTP 的特点；

1.10 进阶自设计

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

（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。