

# 中国地质大学课程论文报告



课程名称	计算机网络报告	
姓 名	常文瀚	
学 院	计算机学院	
班 级	191181 班	
学 号	20181001095	
联系电话	13672173424	
邮 箱	changwh530@gmail.com	
指导老师	陈喆	
报告时间	2021 年 7 月 1 日	

问题及回答记录 (self-test Q&A related to your experience reports, and no less than 5 questions or contents of Q&A nearly full of the sheet as well as handwriting required)
1. 按照资源共享的观点定义的计算机网络应具备几个特征?
A: 1) 计算机网络建立的主要目的是实现计算机资源的共享; 2) 互联的计算机是分布在不同地理位置的多台的“自治计算机”; 3) 联网的计算机之间的通信必须遵循共同的网络协议;
2. 为什么 Host-only 模式下不能上网?
A: Host-Only 模式其实就是 NAT 模式去除了虚拟 NAT 设备, 然后使用 VMware Network Adapter VMnet1 虚拟网卡连接 VMnet1 虚拟交换机来与虚拟机通信的, Host-Only 模式将虚拟机与外网隔开, 使得虚拟机成为一个独立的系统, 只与主机相互通讯。
3. 防火墙的目标和设计目标是什么?
A: 1) 防火墙的目的: 建立可控连接 保护驻地网络免受 Internet 端的攻击 提供单一阻塞点 2) 防火墙的设计目标: 双向的网络流量必须通过防火墙(除非通过防火墙, 否则所有对驻地网络的访问将被物理阻塞) 只有被本地安全策略允许的授权流量能通过防火墙 防火墙自身对渗透免疫(使用具有安全 OS 的可信系统)
4. 防火墙有什么缺陷?
A: 不能防御绕过防火墙的攻击, 不能防御内部攻击, 不能查杀病毒文件
5. 我们学习过了 http 协议, 但是 http 协议和 https 协议有什么区别呢?
A: Http 协议运行在 TCP 之上, 明文传输, 客户端与服务器端都无法验证对方的身份; Https 是身披 SSL(Secure Socket Layer)外壳的 Http, 运行于 SSL 上, SSL 运行于 TCP 之上, 是添加了加密和认证机制的 HTTP。和 HTTP 通信相比, Https 通信会由于加解密处理消耗更多的 CPU 和内存资源, Https 通信需要证书, 而证书一般需要向认证机构购买。
6. TCP 的四次挥手机制, 为什么要有 TIME_WAIT 状态?
A: 因为客户端最后向服务器发送的确认 ACK 是有可能丢失的, 当出现超时, 服务端会再次发送 FIN 报文段, 如果客户端已经关闭了就收不到了。还有一点是避免新旧连接混杂。
7: 假设 A、B 两台主机位于同一网段内, A 想与 B 通信, A 只知道 B 的 IP 地址, 那么 A 通过 ARP 协议就可以获得 B 的 MAC 地址, 进行发送同一网段内的两台主机通信是否需要路由器?
A: 同一网段 A 与 B 通信, 不需要路由器介入, 如果同时满足物理链路连通+网段配置相同的条件的话, 那么两台主机通信可以直接用一根网线直接通信。
8: A 是如何知道自己与 B 在同一网段内的? A 自身进行判断(同一网段的 A 就自己 ARP 寻址, 不是同一网段的就发给路由器)吗?
A: 因为 A-B 之间通信需要对方的 IP 地址, 在主机 A 上实际上有一张路由表, Windows 主机可以在命令行里用 route print 命令打印本机的路由表, 如果 A 要访问的目的地址在 A 所在的网段内, 直接发 ARP 请求, 收到回应以后直接与对方通信。

## 目录

一、Python 网络编程 .....	4
1、应用层编程.....	4
(1) 搭建 Web 服务器.....	4
(2) UDPping 程序.....	4
(3) 邮件客户端.....	5
(4) 多线程 Web 服务器.....	6
(5) TCP 连接程序 .....	7
(6) UDP 连接程序.....	8
(7) RIP 算法的实现.....	8
(8) OSPF 算法的实现 .....	9
二、CentOS7 下服务搭建服务器 .....	10
1、设置虚拟机上网方式及其理解 .....	10
(1) 桥接模式.....	10
(2) NAT 模式.....	11
(3) Host-only 模式.....	11
2、网络文件共享服务.....	12
(1) telnet.....	12
(2) ssh .....	13
(3) 远程桌面.....	14
(4) TFTP 服务 .....	15
(5) Samba 服务.....	15
(6) NFS 服务 .....	16
3、LAMP 环境搭建.....	17
(1) Apache 服务器的安装与设置.....	17
(2) PHP 服务安装配置 .....	18
三、Wireshark 实验 .....	19
四、主要遇到的问题.....	21
1、SSH 无法登录.....	21
2、虚拟机可以 ping 通主机，但是主机无法 ping 到虚拟机 .....	21
3、在安装 PHP 工具时，安装失败.....	22
4、从 Windows 连接 Samba 服务失败.....	22
五、总结.....	23
六、参考与引用.....	23

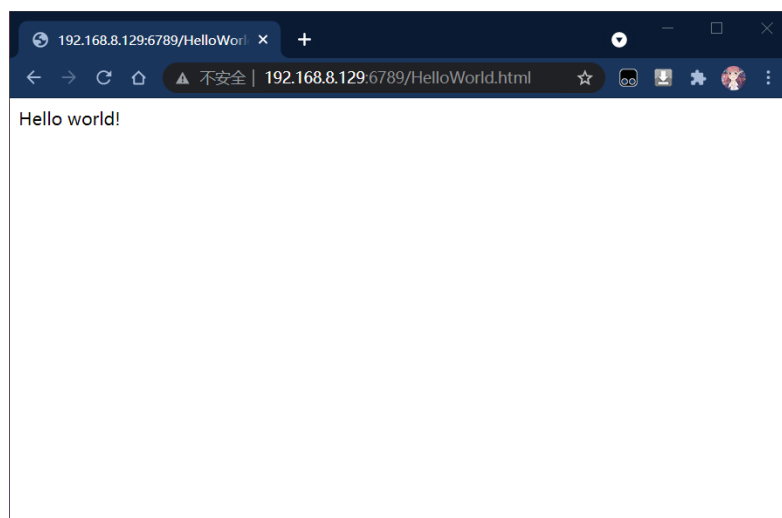
# 一、Python 网络编程

## 1、应用层编程

### (1) 搭建 Web 服务器

Web Server 中文名称叫网页服务器或 web 服务器。WEB 服务器也称为 WWW (WORLD WIDE WEB) 服务器，主要功能是提供网上信息浏览服务 Web 服务器可以解析 (handles) HTTP 协议。当 Web 服务器接收到一个 HTTP 请求 (request)，会返回一个 HTTP 响应 (response)，例如送回一个 HTML 页面。

首先在虚拟机运行 Server 端 python 文件，并将 Server 端 IP 地址写入 Client 端 python 文件。



### (2) UDPPing 程序

ping 是用来探测本机与网络中另一主机之间是否可达的命令，如果两台主机之间 ping 不通，则表明这两台主机不能建立起连接。ping 是定位网络通不通的一个重要手段。

ping 命令是基于 ICMP 协议来工作的，ICMP 全称为 Internet 控制报文协议。ping 命令会发送一份 ICMP 回显请求报文给目标主机，并等待目标主机返回 ICMP 回显应答。因为 ICMP 协议会要求目标主机在收到消息之后，必须返回 ICMP 应答消息给源主机，如果源主机在一定时间内收到了目标主机的应答，则表明两台主机之间网络是可达的。

首先在主机运行 Server 端文件，其次在虚拟机内运行 UDPPinger.py，从虚拟机 ping 主机，可以看到屏幕显示出主机 IP 地址以及 RTT。

```
chang@ubuntu: ~/Downloads/code/UDPPing/source
chang@ubuntu:~/Downloads/code/UDPPing/source$ python3 UDPPinger.py
Sequence 1: Reply from 172.27.125.92 RTT = 0.001s
Sequence 2: Reply from 172.27.125.92 RTT = 0.000s
Sequence 3: Reply from 172.27.125.92 RTT = 0.000s
Sequence 4: Reply from 172.27.125.92 RTT = 0.000s
Sequence 5: Reply from 172.27.125.92 RTT = 0.000s
Sequence 6: Reply from 172.27.125.92 RTT = 0.000s
Sequence 7: Request timed out
Sequence 8: Reply from 172.27.125.92 RTT = 0.001s
Sequence 9: Reply from 172.27.125.92 RTT = 0.000s
Sequence 10: Request timed out
chang@ubuntu:~/Downloads/code/UDPPing/source$ python3 UDPPinger.py
Sequence 1: Reply from 172.27.125.92 RTT = 0.001s
Sequence 2: Reply from 172.27.125.92 RTT = 0.000s
Sequence 3: Reply from 172.27.125.92 RTT = 0.000s
Sequence 4: Reply from 172.27.125.92 RTT = 0.002s
Sequence 5: Reply from 172.27.125.92 RTT = 0.000s
Sequence 6: Reply from 172.27.125.92 RTT = 0.000s
Sequence 7: Reply from 172.27.125.92 RTT = 0.000s
Sequence 8: Reply from 172.27.125.92 RTT = 0.000s
Sequence 9: Reply from 172.27.125.92 RTT = 0.000s
Sequence 10: Reply from 172.27.125.92 RTT = 0.000s
chang@ubuntu:~/Downloads/code/UDPPing/source$ python3 UDPPinger.py
Sequence 1: Reply from 172.27.125.92 RTT = 0.001s
Sequence 2: Reply from 172.27.125.92 RTT = 0.000s
Sequence 3: Reply from 172.27.125.92 RTT = 0.001s
Sequence 4: Reply from 172.27.125.92 RTT = 0.000s
Sequence 5: Reply from 172.27.125.92 RTT = 0.001s
Sequence 6: Request timed out
Sequence 7: Request timed out
Sequence 8: Reply from 172.27.125.92 RTT = 0.002s
Sequence 9: Reply from 172.27.125.92 RTT = 0.002s
Sequence 10: Request timed out
chang@ubuntu:~/Downloads/code/UDPPing/source$
```

### (3) 邮件客户端

SMTP (Simple Mail Transfer Protocol) 即简单邮件传输协议,它是一组用于由源地址到目的地址传送邮件的规则,由它来控制信件的中转方式。

在运行此代码前,需要修改发送端邮箱和接收端邮箱,打开发送端 SMTP 服务后,修改登录密码,编辑邮件文本。

为了保障用户邮箱的安全,QQ 邮箱设置了 POP3/SMTP/IMAP 的开关。系统缺省设置是“关闭”,在用户需要这些功能时需要“开启”。最后,保存设置,即打开了相应的服务。

在本题中,SMTP 邮件客户端程序的基本流程如下:

与 163 邮件服务器建立 TCP 连接,域名"smtp.126.com",SMTP 默认端口号 25。建立连接后服务器将返回状态码 220,代表服务就绪(类似 HTTP,SMTP 也使用状态码通知客户端状态信息)。

发送"HELO"命令,开始与服务器的交互,服务器将返回状态码 250(请求动作正确完成)。

发送"AUTH LOGIN"命令,开始验证身份,服务器将返回状态码 334(服务器等待用户输入验证信息)。

发送经过 base64 编码的用户名(本例中是 163 邮箱的账号),服务器将返回状态码 334(服务器等待用户输入验证信息)。

发送经过 base64 编码的密码(本例中是 163 邮箱的密码),服务器将返回状态码 235(用户验证成功)。

发送"MAIL FROM"命令,并包含发件人邮箱地址,服务器将返回状态码 250(请求动作正确完成)。

发送"RCPT TO"命令,并包含收件人邮箱地址,服务器将返回状态码 250(请求动作正确完成)。

发送"DATA"命令,表示即将发送邮件内容,服务器将返回状态码 354(开始邮件输入,以"."结束)。

发送邮件内容,服务器将返回状态码 250(请求动作正确完成)。

发送"QUIT"命令,断开与邮件服务器的连接。

```

250-SIZE 73400320
250 OK

334 VXXN1cm5hbwU6

334 UGFzc3dvcnQ6

235 Authentication successful

250 OK.

250 OK

354 End data with <CR><LF>.<CR><LF>.

250 OK: queued as.

```

#### (4) 多线程 Web 服务器

Web 缓存器也叫代理服务器，他是能够代表初始 Web 服务器满足 HTTP 请求的网络实体。Web 缓存器有自己的磁盘存储空间，并在存储空间中保存最近请求给的副本。

Web 缓存器既是服务器又是客户，首先需要讲浏览器端口修改为代理服务器使用的端口。

访问目标 URL，看到控制台显示了一系列的报文：

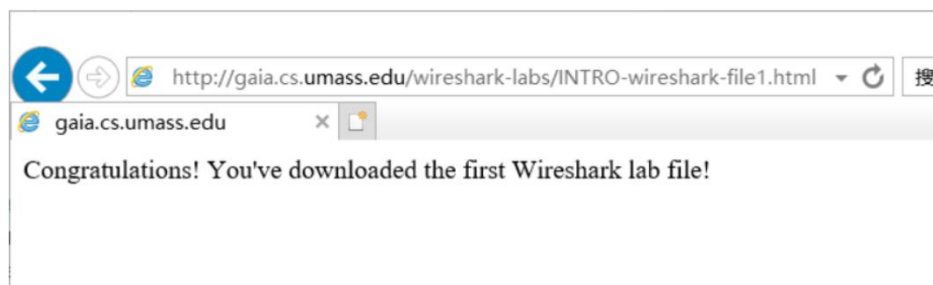
```

准备从客户机接收响应消息...
接收到一个连接，来自： ('127.0.0.1', 54291)
客户机发送过来的消息： GET http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html
HTTP/1.1
Accept: text/html, application/xhtml+xml, image/jxr, */*
Accept-Language: zh-Hans-CN,zh-Hans;q=0.5
User-Agent: Mozilla/5.0 (windows NT 10.0; WOW64; Trident/7.0; rv:11.0) like Gecko
Accept-Encoding: gzip, deflate
Host: gaia.cs.umass.edu
Proxy-Connection: Keep-Alive

文件名: gaia.cs.umass.edu wireshark-labs INTRO-wireshark-file1.html
开始检查代理服务器中是否存在文件: gaia.cs.umass.edu_wireshark-labs_INTRO-wireshark-file1.html文件不
在
Host Name: gaia.cs.umass.edu
套接字连接到主机的80号端口
关闭套接字: tcpCliSock
准备从客户机接收响应消息...
接收到一个连接，来自： ('127.0.0.1', 54292)

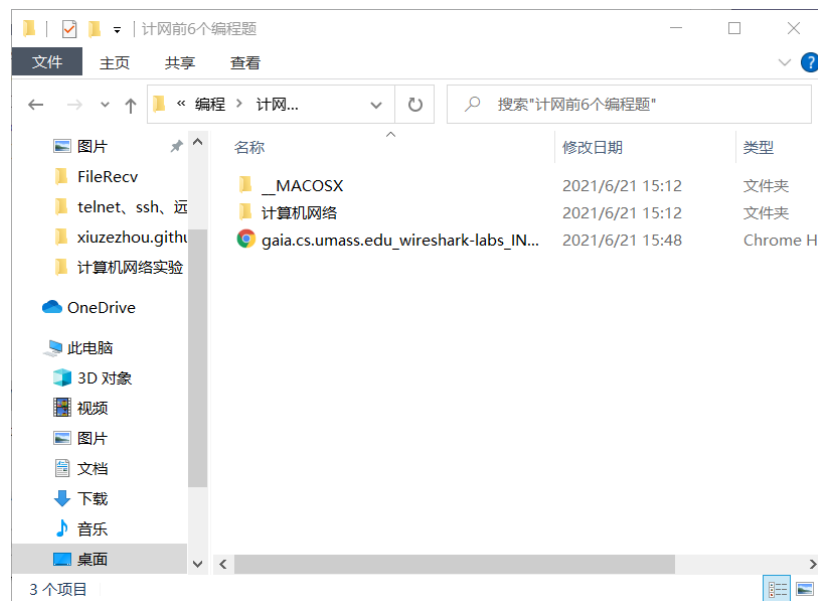
```

访问成功，显示出了目标 html 文件中的文本：



在文件夹中也显出了代理服务器下载的 html 文件，按照官方文档的代码骨架，代理程序接收用户的 GET 请求，若请求的内容发已存在于代理服务器，则直接返回内容，若不存

在，则转发该请求到目标服务器，接收目标服务器的响应，并将响应内容存储为文件，作为缓存，已被之后同样的请求。



## (5) TCP 连接程序

传输控制协议是一种面向连接的、可靠的、基于字节流的传输层通信协议。

应用层向 TCP 层发送用于网间传输的、用 8 位字节表示的数据流，然后 TCP 把数据流分割成适当长度的报文段（通常受该计算机连接的网络的数据链路层的最大传输单元（MTU）的限制）。之后 TCP 把结果包传给 IP 层，由它来透过网络将包传送给接收端实体的 TCP 层。TCP 为了保证不发生丢包，就给每个包一个序号，同时序号也保证了传送到接收端实体的包的按序接收。然后接收端实体对已成功收到的包发回一个相应的确认信息（ACK）；如果发送端实体在合理的往返时延（RTT）内未收到确认，那么对应的数据包就被假设为已丢失并进行重传。TCP 用一个校验和函数来检验数据是否有错误，在发送和接收时都要计算校验和。

TCP 连接程序同样需要在主机和虚拟机分别运行 Client 和 Server 端程序。

```
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 371 bytes 32197 (32.1 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 371 bytes 32197 (32.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

chang@ubuntu:~$ cd TCP
chang@ubuntu:~/TCP$ python3 TCPServer.py
服务器已经启动
('192.168.8.1', 64396)
已经建立连接
从客户端接收到信息为： 常文瀚
给客户端接的回复信息为： Good Evening, Chang
从客户端接收到信息为： 哈哈哈哈哈
给客户端接的回复信息为： hhhhh
从客户端接收到信息为： 你吃饭了么
给客户端接的回复信息为： No
从客户端接收到信息为： ok
给客户端接的回复信息为： █
```



## (6) UDP 连接程序

该实验需要首先运行 Server 端 python 文件，使 Server 端进入等候状态，否则 Client 将会连接失败。

```
PS C:\Users\chang\Desktop\计算机网络实验\编程\UDP> python UDPClient.py
>hello
send the data: hello
receive the reply: Hello, this is udpserver!
>Good Morning
send the data: Good Morning
receive the reply: Hello, this is udpserver!
>
```

在实验中我思考到 UDP 是一个无连接的协议，因此 socket 函数 connect 似乎对 UDP 是没有意义的，然而事实不是这样。对于 UDP 来说，socket 函数建立一个插口；bind 函数指明了本地地址/端口（包括 ADDR\_ANY，通配所有本地网络接口）；connect 可以用来指明目的地址/端口。

一般来说，UDP 客户端在建立了插口后会直接用 sendto 函数发送数据，需要在 sendto 函数的参数里指明目的地址/端口。如果一个 UDP 客户端在建立了插口后首先用 connect 函数指明了目的地址/端口，然后也可以用 send 函数发送数据，因为此时 send 函数已经知道对方地址/端口，用 getsockname 也可以得到这个信息。

UDP 客户端在建立了插口后会直接用 sendto 函数发送数据，还隐含了一个操作，那就是在发送数据之前，UDP 会首先为该插口选择一个独立的 UDP 端口（在 1024-5000 之间），将该插口置为已绑定状态。如果一个 UDP 客户端在建立了插口后首先用 bind 函数指明了本地地址/端口，也是可以的，这样可以强迫 UDP 使用指定的端口发送数据。（事实上，UDP 无所谓服务器和客户端，这里的界限已经模糊了。）

## (7) RIP 算法的实现

输入从 C 发送来的报文，得到如下结果：

新的路由表为：		
目的网络	距离	下一跳路由器
N1	7	A
N2	5	C
N6	5	C
N8	4	E
N9	4	F
N3	9	C

在实验后对 RIP 进行总结，RIP 是一种基于距离矢量（Distance-Vector）算法的协议，它使用跳数（Hop Count）作为度量值来衡量到达目的地址的距离。在 RIP 网络中，缺省情况下，设备到与它直接相连网络的跳数为 0，通过一个设备可达的网络的跳数为 1，其余依此类推。也就是说，度量值等于从本网络到达目的网络间的设备数量。为限制收敛时间，RIP 规定度量值取 0~15 之间的整数，大于或等于 16 的跳数被定义为无穷大，即目的网络或主机不可达。由于这个限制，使得 RIP 不可能在大型网络中得到应用。



## (8) OSPF 算法的实现

OSPF 路由协议是一种典型的链路状态（Link-state）的路由协议，一般用于同一个路由域内。在这里，路由域是指一个自治系统（Autonomous System），即 AS，它是指一组通过统一的路由政策或路由协议互相交换路由信息的网络。在这个 AS 中，所有的 OSPF 路由器都维护一个相同的描述这个 AS 结构的数据库，该数据库中存放的是路由域中相应链路的状态信息，OSPF 路由器正是通过这个数据库计算出其 OSPF 路由表的。

运行代码，输入路由信息：

```
PS C:\Users\chang\Desktop\计算机网络实验\编程\路由算法> python .\OSPF_CN.py
请输入节点，以#结束：A
请输入节点A的链路状态分组(相邻路由器，度量)，以空格隔开，以*结束：B 4
请输入节点A的链路状态分组(相邻路由器，度量)，以空格隔开，以*结束：E 5
请输入节点A的链路状态分组(相邻路由器，度量)，以空格隔开，以*结束：*
请输入节点，以#结束：B
请输入节点B的链路状态分组(相邻路由器，度量)，以空格隔开，以*结束：A 4
请输入节点B的链路状态分组(相邻路由器，度量)，以空格隔开，以*结束：C 2
请输入节点B的链路状态分组(相邻路由器，度量)，以空格隔开，以*结束：F 6
请输入节点B的链路状态分组(相邻路由器，度量)，以空格隔开，以*结束：*
请输入节点，以#结束：C
请输入节点C的链路状态分组(相邻路由器，度量)，以空格隔开，以*结束：B 2
请输入节点C的链路状态分组(相邻路由器，度量)，以空格隔开，以*结束：D 3
请输入节点C的链路状态分组(相邻路由器，度量)，以空格隔开，以*结束：E 1
请输入节点C的链路状态分组(相邻路由器，度量)，以空格隔开，以*结束：*
请输入节点，以#结束：D
请输入节点D的链路状态分组(相邻路由器，度量)，以空格隔开，以*结束：C 3
请输入节点D的链路状态分组(相邻路由器，度量)，以空格隔开，以*结束：F 7
请输入节点D的链路状态分组(相邻路由器，度量)，以空格隔开，以*结束：*
请输入节点，以#结束：E
请输入节点E的链路状态分组(相邻路由器，度量)，以空格隔开，以*结束：A 5
请输入节点E的链路状态分组(相邻路由器，度量)，以空格隔开，以*结束：C 1
请输入节点E的链路状态分组(相邻路由器，度量)，以空格隔开，以*结束：F 8
请输入节点E的链路状态分组(相邻路由器，度量)，以空格隔开，以*结束：*
请输入节点，以#结束：F
请输入节点F的链路状态分组(相邻路由器，度量)，以空格隔开，以*结束：B 6
请输入节点F的链路状态分组(相邻路由器，度量)，以空格隔开，以*结束：D 7
请输入节点F的链路状态分组(相邻路由器，度量)，以空格隔开，以*结束：E 8
请输入节点F的链路状态分组(相邻路由器，度量)，以空格隔开，以*结束：*
请输入节点，以#结束：#
```

查看各个节点的路由表：

```
请输入你想查看路由表的节点：E
节点E的路由表如下：
目的网络  距离  下一跳路由器
A          5      A
B          3      C
C          1      C
D          4      C
F          8      F
请输入你想查看路由表的节点：F
节点F的路由表如下：
目的网络  距离  下一跳路由器
A         10      B
B          6      B
C          8      B
D          7      D
E          8      E
```

当路由器开启 OSPF 后，路由器之间就会相互发送 HELLO 报文，HELLO 报文中包含一些路由器和链路的相关信息，发送 HELLO 报文的目的是为了形成邻居表，然后，路由器

之间就会发送 LSA（LINK STATE ADVERTISEMENT，链路状态通告），LSA 告诉自己的邻居路由器和自己相连的链路的状态，最后，形成网络的拓扑表，其实这个过程是很复杂的，他们经过发 LSA，记录 LSA，装发 LSA，最后形成 LSDB（链路状态数据库，即拓扑表），形成拓扑表之后，在经过 SPF 算法，通过计算 LSDB，最后形成路由表。

形成路由表后，路由器就可以根据路由表来转发数据包，但是，这只是理想情况，如果之后，网络拓扑发生了变化，或是网络链路出现了问题，OSPF 协议还是会经过这三张表来重新计算新的路由，只不过不会这么复杂了，路由器在默认情况下，10S 就会发送一次 HELLO 报文，以检测链路状态，保证链路始终是正常的。

## 二、CentOS7 下服务搭建服务器

### 1、设置虚拟机上网方式及其理解

#### （1）桥接模式

桥接模式就是将主机网卡与虚拟机虚拟的网卡利用虚拟网桥进行通信。在桥接的作用下，类似于把物理主机虚拟为一个交换机，所有桥接设置的虚拟机连接到这个交换机的一个接口上，物理主机也同样插在这个交换机当中，所以所有桥接下的网卡与网卡都是交换模式的，相互可以访问而不干扰。在桥接模式下，虚拟机 ip 地址需要与主机在同一个网段，如果需要联网，则网关与 DNS 需要与主机网卡一致。

使用管理员权限，修改连接方式为桥接模式，并将网卡选择为物理网卡：



Ping 百度，查看延迟：

```
chang@localhost:~  
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)  
[chang@localhost ~]$ ping www.baidu.com  
PING www.baidu.com (220.181.38.150) 56(84) bytes of data.  
64 bytes from 220.181.38.150 (220.181.38.150): icmp_seq=1 ttl=28 time=49.0 ms  
64 bytes from 220.181.38.150 (220.181.38.150): icmp_seq=2 ttl=28 time=38.8 ms  
64 bytes from 220.181.38.150 (220.181.38.150): icmp_seq=3 ttl=28 time=38.9 ms  
64 bytes from 220.181.38.150 (220.181.38.150): icmp_seq=4 ttl=28 time=69.0 ms  
64 bytes from 220.181.38.150 (220.181.38.150): icmp_seq=5 ttl=28 time=632 ms  
64 bytes from 220.181.38.150 (220.181.38.150): icmp_seq=6 ttl=28 time=62.2 ms  
64 bytes from 220.181.38.150 (220.181.38.150): icmp_seq=7 ttl=28 time=47.0 ms  
64 bytes from 220.181.38.150 (220.181.38.150): icmp_seq=8 ttl=28 time=50.0 ms  
64 bytes from 220.181.38.150 (220.181.38.150): icmp_seq=9 ttl=28 time=33.3 ms  
^C  
--- www.baidu.com ping statistics ---  
9 packets transmitted, 9 received, 0% packet loss, time 8068ms  
rtt min/avg/max/mdev = 33.350/112.297/632.077/184.024 ms  
[chang@localhost ~]$
```

## (2) NAT 模式

NAT 的基本工作原理是，当私有网主机和公网主机通信的 IP 包经过 NAT 网关时，将 IP 包中的源 IP 或目的 IP 在私有 IP 和 NAT 的公共 IP 之间进行转换。

创建虚拟机时选择 NAT 上网方式，可以直接上网：



这种方法需要在专用网连接到因特网的路由器上安装 NAT 软件。装有 NAT 软件的路由器叫做 NAT 路由器，它至少有一个有效的外部全球 IP 地址。这样，所有使用本地地址的主机在和外界通信时，都要在 NAT 路由器上将其本地地址转换成全球 IP 地址，才能和因特网连接。

## (3) Host-only 模式

切换上网方式后发现，Host-only 模式下无法上网，Host-Only 模式其实就是 NAT 模式去除了虚拟 NAT 设备，然后使用 VMware Network Adapter VMnet1 虚拟网卡连接 VMnet1 虚

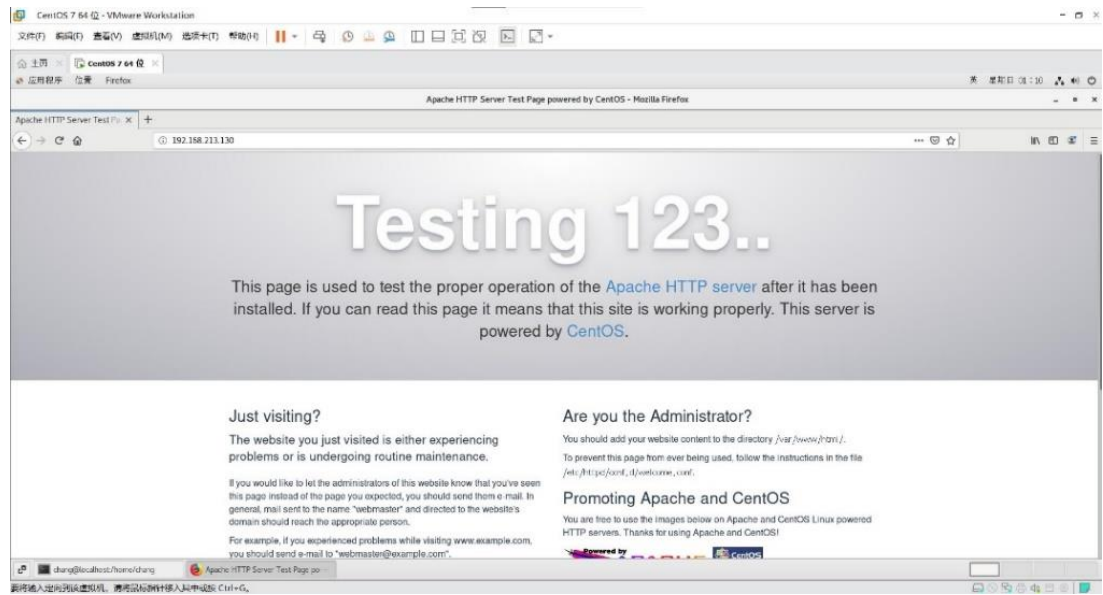
拟交换机来与虚拟机通信的，Host-Only 模式将虚拟机与外网隔开，使得虚拟机成为一个独立的系统，只与主机相互通讯。

```
chang@localhost:~  
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)  
[chang@localhost ~]$ ping www.baidu.com  
ping: www.baidu.com: 未知的名称或服务  
[chang@localhost ~]$ s
```

2、网络文件共享服务

(1) telnet

安装 telnet: 打开防火墙，在 Windows 端添加出入站规则，telnet 登陆成功后，安装 Apache，开启 httpd 服务。  
访问 IP 地址，查看效果，



Telnet 是一种应用层协议，使用于互联网及局域网中，使用虚拟终端的形式，提供双向、以文字字符串为主的命令行接口交互功能。属于 TCP/IP 协议族的其中之一，是互联网远程登录服务的标准协议和主要方式，常用于服务器的远程控制，可供用户在本地主机运行远程主机上的工作。

## (2) ssh

配置 ssh 信息，允许 root 用户登录：查看 ssh 安装和运行状态：

```
chang@localhost:/home/chang
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
[chang@localhost ~]$ su root
密码:
[root@localhost chang]# rpm -qa | grep ssh
libssh2-1.8.0-3.el7.x86_64
openssh-clients-7.4p1-21.el7.x86_64
openssh-7.4p1-21.el7.x86_64
openssh-server-7.4p1-21.el7.x86_64
[root@localhost chang]# vim /etc/ssh/sshd_config
[root@localhost chang]# systemctl start sshd.service
[root@localhost chang]# systemctl status sshd.service
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: en
   Active: active (running) since 日 2021-06-20 00:07:30 PDT; 2h 8min ago
     Docs: man: sshd(8)
           man: sshd_config(5)
    Main PID: 1053 (sshd)
      Tasks: 1
     CGroup: /system.slice/sshd.service
             └─1053 /usr/sbin/sshd -D

6月 20 00:07:30 localhost.localdomain systemd[1]: Starting OpenSSH server da...
6月 20 00:07:30 localhost.localdomain sshd[1053]: Server listening on 0.0.0....
6月 20 00:07:30 localhost.localdomain sshd[1053]: Server listening on :: por...
6月 20 00:07:30 localhost.localdomain systemd[1]: Started OpenSSH server dae...
Hint: Some lines were ellipsized, use -l to show in full.
[root@localhost chang]#
```

开放 22 端口，查看 IP，访问成功：

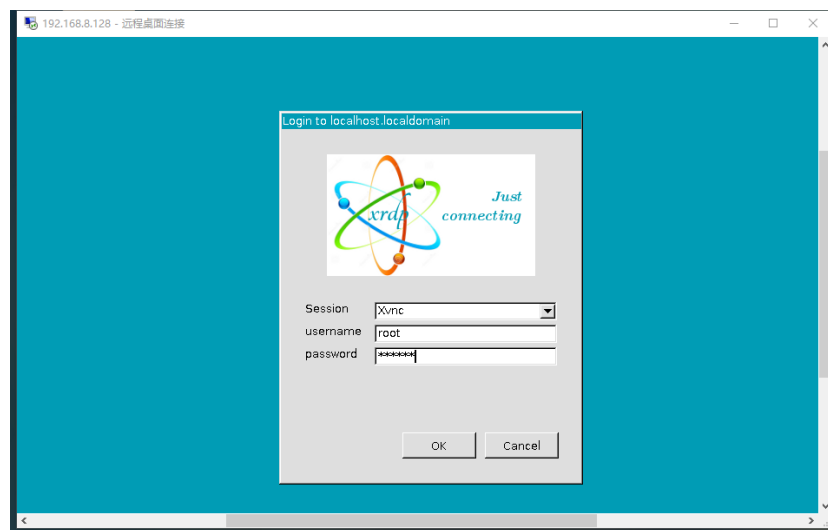


```
root@localhost:~
The authenticity of host '192.168.181.129 (192.168.181.129)' can't be established.
ECDSA key fingerprint is SHA256:tFvpXtlwo4SWJxIxxoWUmScZqOss6cuDAc1AiX6UMsU.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.181.129' (ECDSA) to the list of known hosts.
root@192.168.181.129's password:
Last login: Sun Jun 20 04:38:10 2021 from 192.168.181.1
[root@localhost ~]#
```

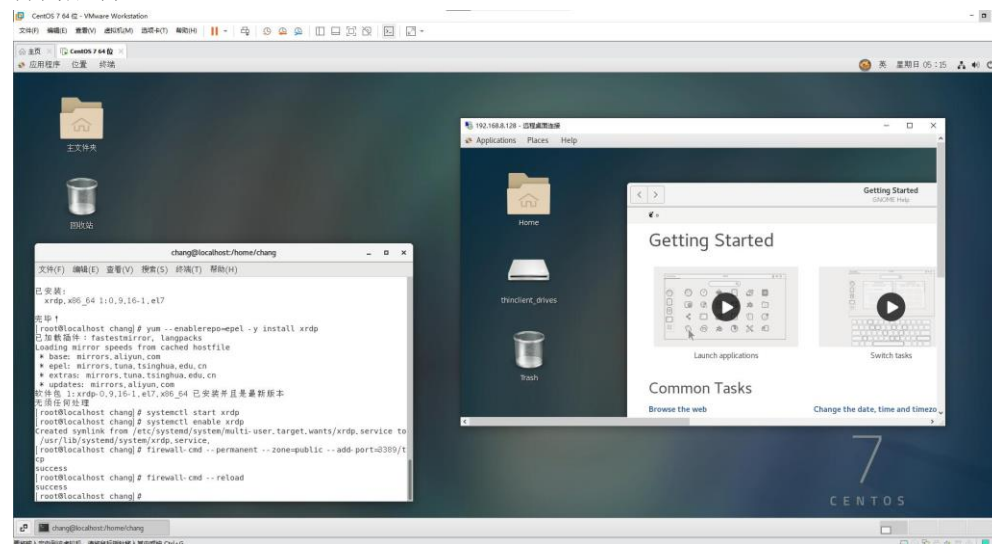
SSH 是英文 Secure Shell 的简写形式。通过使用 SSH，你可以把所有传输的数据进行加密，这样 " 中间人 " 这种攻击方式就不可能实现了，而且也能够防止 DNS 欺骗和 IP 欺骗。使用 SSH，还有一个额外的好处就是传输的数据是经过压缩的，所以可以加快传输的速度。SSH 有很多功能，它既可以代替 Telnet，又可以为 FTP、Pop、甚至为 PPP 提供一个安全的 " 通道 "。

### (3) 远程桌面

安装 xrdp，打开远程桌面需要使用的端口，Windows 端也需要打开，启动远程桌面，输入目标主机的用户名称和密码：



访问成功！



远程桌面的工作原理是客户端和服务端通过 TCP/IP 协议和标准的局域网构架联系，通过客户端终端，客户端的鼠标、键盘的输入传递到服务器端上，再把服务器端的显示传递回客户端。应用程序始终运行在服务器端上，客户端不需要具有计算能力，至多只需提供一定的缓存能力。

简单点说，就是在一台电脑（远程终端）上操作和显示，而实际是在另一台电脑（宿主服务器）上运行程序。终端通过网络传送本地键盘鼠标操作，并接收合成后的画面。画面通常是在服务器 CPU 中虚拟一块显卡来合成，也有能借助物理显卡的技术，但是并不常用。

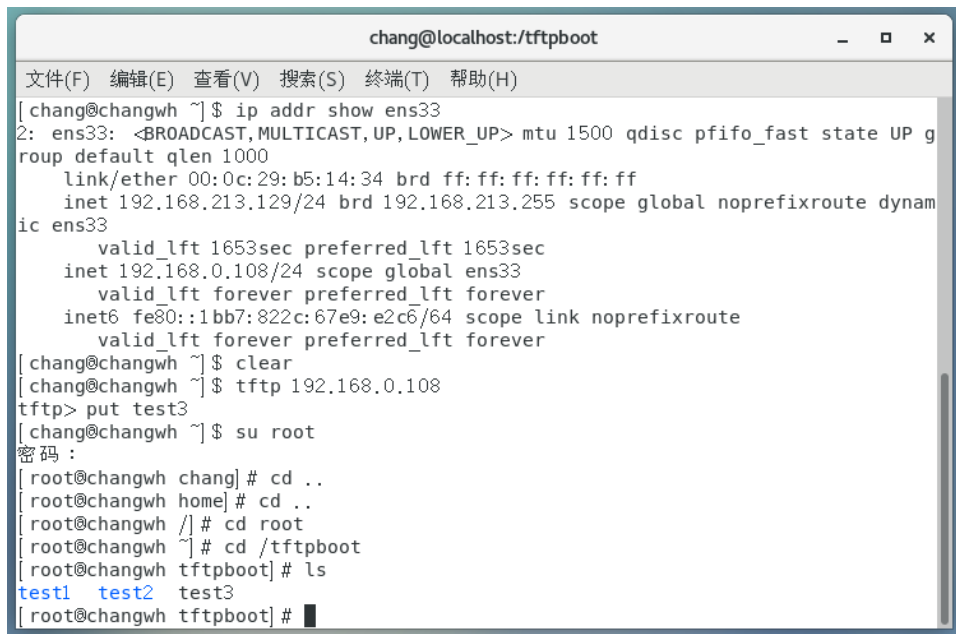


#### (4) TFTP 服务

TFTP(Trivial File Transfer Protocol, 简单文件传输协议)该协议在熟知端口 69 上使用 UDP 服务。TFTP 协议常用于无盘工作站或路由器从别的主机上获取引导配置文件, 由于 TFTP 报文比较小, 能够迅速复制这些文件。

安装 TFTP, 启动服务, 查看状态, 关闭 SELinux, 下载文件。

查看 ip, 向 ip 主机上传 test3 并验证是否存在:

A terminal window titled 'chang@localhost:/tftpboot' showing the following commands and output:

```
chang@changwh ~]$ ip addr show ens33
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:b5:14:34 brd ff:ff:ff:ff:ff:ff
    inet 192.168.213.129/24 brd 192.168.213.255 scope global noprefixroute dynamic ens33
        valid_lft 1653sec preferred_lft 1653sec
    inet 192.168.0.108/24 scope global ens33
        valid_lft forever preferred_lft forever
    inet6 fe80::1bb7:822c:67e9:e2c6/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
[chang@changwh ~]$ clear
[chang@changwh ~]$ tftp 192.168.0.108
tftp> put test3
[chang@changwh ~]$ su root
密码:
[root@changwh chang]# cd ..
[root@changwh home]# cd ..
[root@changwh /]# cd root
[root@changwh ~]# cd /tftpboot
[root@changwh tftpboot]# ls
test1 test2 test3
[root@changwh tftpboot]#
```

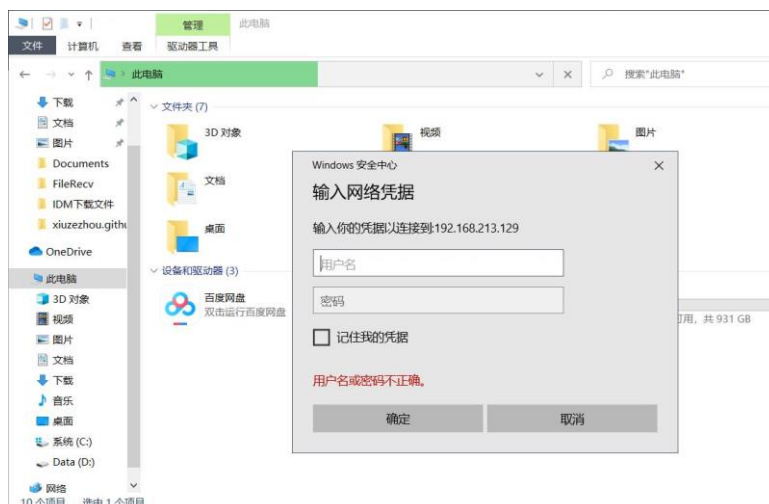
TFTP 协议是基于服务器和客户端之间的传输协议。一开始客户端向服务器发出连接请求, 服务器应答后两者连接建立。然后客户端向服务器发出文件传输请求, 服务器将客户端需要的文件分割成多个小块, 依次传递给客户端, 客户端每收到一个小块后向服务器发出应答, 收到应答后服务器再发送下一个小块, 当所有文件块传输完毕后, 两者连接断开。

#### (5) Samba 服务

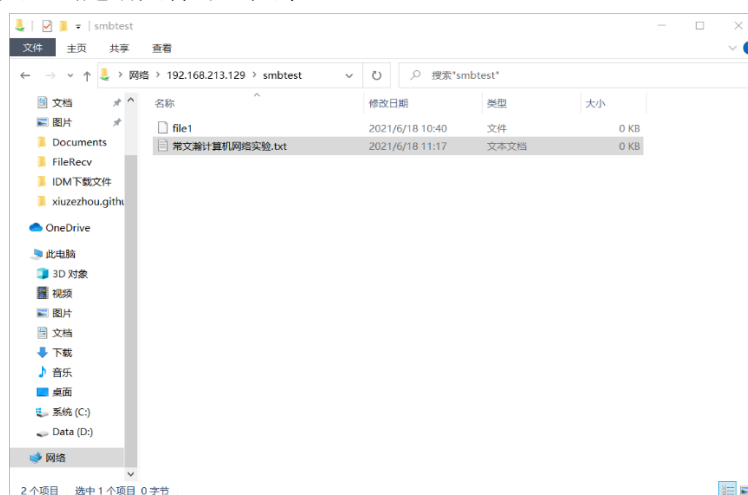
Samba 服务功能强大, 这与其通信基于 SMB/CIFS 协议有关。SMB 不仅提供目录和打印机共享, 还支持认证、权限设置。在早期, SMB 运行于 NBT 协议 (NetBIOS over TCP/IP) 上, 使用 UDP 协议的 137、138 及 TCP 协议的 139 端口; 后期 SMB 经过开发, 可以直接运行于 TCP/IP 协议上, 没有额外的 NBT 层, 使用 TCP 协议的 445 端口。

设置 Samba 配置文件, 添加 Samba 用户, 创建 Samba 共享文件夹, 开启本地 Windows 的 Samba 服务, 访问虚拟机 Samba:



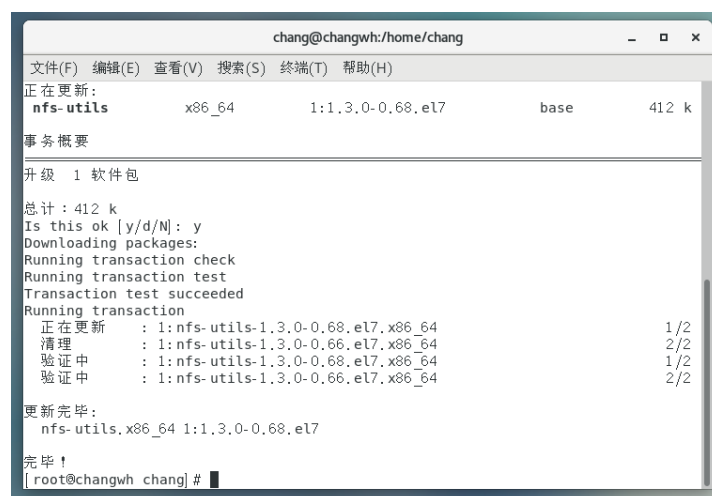


查看文件夹，创建新文件即可共享：

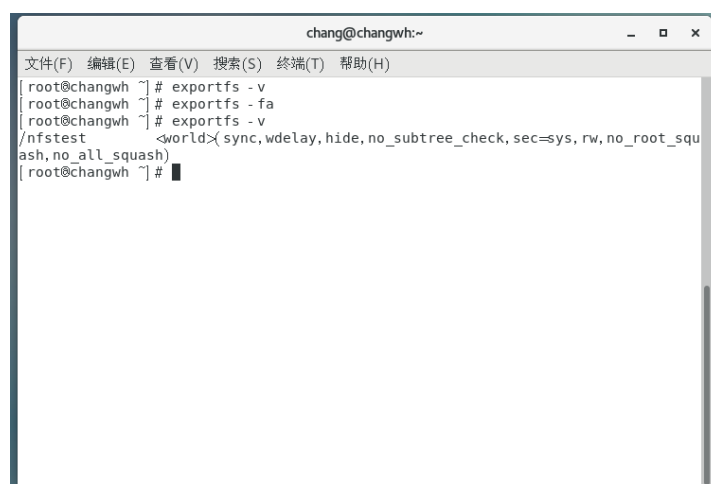


## (6) NFS 服务

安装 NFS：



创建共享文件夹，修改访问权限，查看目录导出情况：



```
chang@changwh:~  
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)  
[root@changwh ~]# exportfs -v  
[root@changwh ~]# exportfs -fa  
[root@changwh ~]# exportfs -v  
/nfstest->world(x,sync,wdelay,hide,no_subtree_check,sec=sys,rw,no_root_squash,no_all_squash)  
[root@changwh ~]#
```

NFS 的主要功能是通过网络让不同的机器系统之间可以彼此共享文件和目录。NFS 服务器可以允许 NFS 客户端将远端 NFS 服务器端的共享目录挂载到本地的 NFS 客户端中。在本地的 NFS 客户端的机器看来，NFS 服务器端共享的目录就好像自己的磁盘分区和目录一样。一般客户端挂载到本地目录的名字可以随便，但为方便管理，我们要和服务器端一样比较好。


当我们在 NFS 服务器设置好一个共享目录/data 后，其他的有权访问 NFS 服务器的 NFS 客户端就可以将这个目录挂载到本地。并且能够看到服务端/data 的所有数据。因为挂载在本地的/data 目录，其实就是服务器端的/data 目录。如果服务器端配置的客户端只读，那么客户端就只能够只读。如果配置读写，客户端就能够进行读写。

NFS 客户端又是如何知道 NFS 服务器端到底使用的是哪个端口呢？其实 NFS 服务器时通过远程过程调用（remote procedure call 简称 RPC）协议/服务来实现的。也就是说 RPC 服务会统一管理 NFS 的端口，客户端和服务端通过 RPC 来先沟通 NFS 使用了哪些端口，之后再利用这些端口（小于 1024）来进行数据的传输。

### 3、LAMP 环境搭建

#### （1）Apache 服务器的安装与设置

首先查看 httpd 版本：

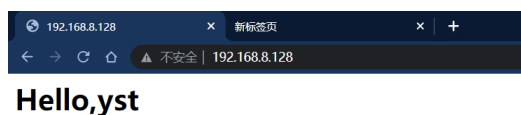


```
chang@localhost:~  
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)  
[chang@localhost ~]$ su root  
密码：  
[root@localhost chang]# cd ..  
[root@localhost home]# cd ..  
[root@localhost /]# cd root  
[root@localhost ~]# rpm -qa httpd  
httpd-2.4.6-97.el7.centos.x86_64
```

上传 html 文件，设置个人主页到 localhost，并访问：

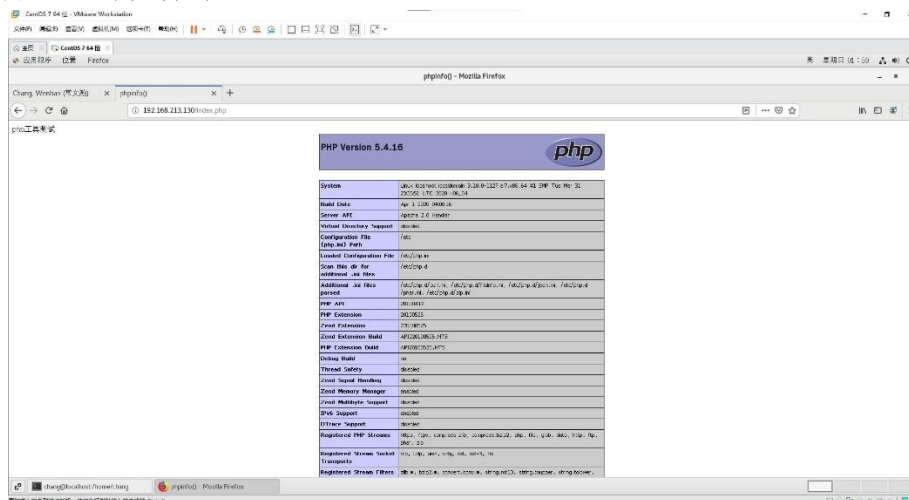


设置 index.html 到 Apache 的访问目录，设置域名，查看语法无误后，访问目标地址：

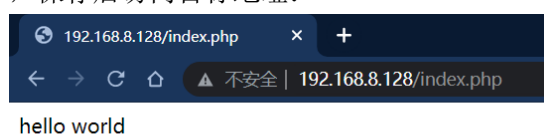


## (2) PHP 服务安装配置

安装 PHP，测试效果：



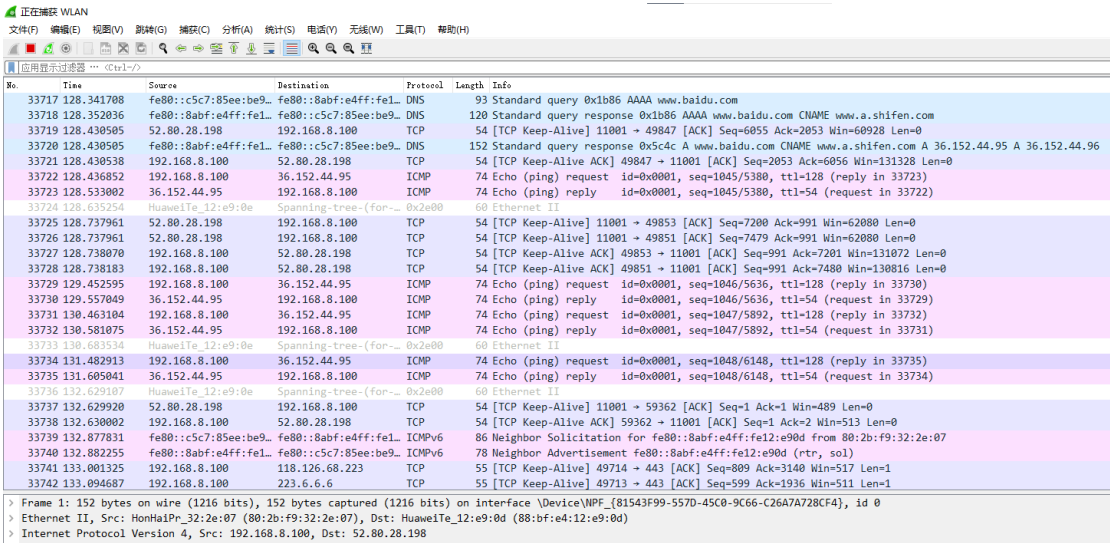
编写目标 php 文件，保存后访问目标地址：



### 三、Wireshark 实验

作为一款高效免费的抓包工具，Wireshark 可以捕获并描述网络数据包，其最大的优势就是免费、开源以及多平台支持，在 GNU 通用公共许可证的保障范围下，用户可以免费获取软件和代码，并拥有对其源码修改和定制的权利，如今其已是全球最广泛的网络数据包分析软件之一。

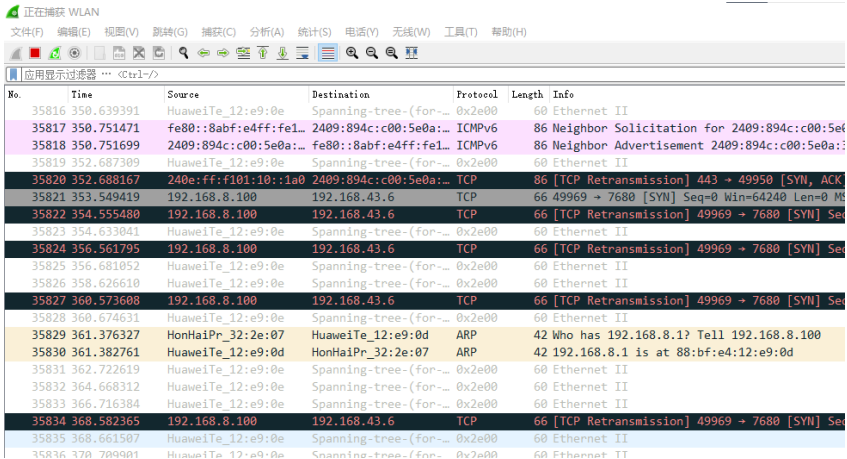
(1) 首先 ping 一下 [www.baidu.com](http://www.baidu.com)，可以看到 Wireshark 捕获到了 ICMP 协议的报文，在图中的颜色是粉色：



The screenshot shows a Wireshark capture of network traffic on the wlan interface. The packet list on the left shows several ICMP Echo (ping) requests and replies. The packet details pane on the right shows the structure of an ICMP Echo request, including the type, code, identifier, and sequence number. The packet bytes pane at the bottom shows the raw data of the packet.

No.	Time	Source	Destination	Protocol	Length	Info
33717	128.341708	fe80::c5c7:85ee:be9...	fe80::8abf:e4ff:fe1...	DNS	93	Standard query 0x1b86 AAAA www.baidu.com
33718	128.352036	fe80::8abf:e4ff:fe1...	fe80::c5c7:85ee:be9...	DNS	120	Standard query response 0x1b86 AAAA www.baidu.com CNAME www.a.shifen.com
33719	128.430505	52.80.28.198	192.168.8.100	TCP	54	[TCP Keep-Alive] 11001 → 49847 [ACK] Seq=6055 Ack=2053 Win=60928 Len=0
33720	128.430505	fe80::8abf:e4ff:fe1...	fe80::c5c7:85ee:be9...	DNS	152	Standard query response 0x5c4c A www.baidu.com CNAME www.a.shifen.com A 36.152.44.95 A 36.152.44.96
33721	128.430538	192.168.8.100	52.80.28.198	TCP	54	[TCP Keep-Alive ACK] 49847 → 11001 [ACK] Seq=2053 Ack=6056 Win=131328 Len=0
33722	128.436852	192.168.8.100	36.152.44.95	ICMP	74	Echo (ping) request id=0x0001, seq=1045/5380, ttl=128 (reply in 33723)
33723	128.533002	36.152.44.95	192.168.8.100	ICMP	74	Echo (ping) reply id=0x0001, seq=1045/5380, ttl=54 (request in 33722)
33724	128.635254	HuaweiTe_12:e9:0e	Spanning-tree-(for-...	0x2e00	60	Ethernet II
33725	128.737961	52.80.28.198	192.168.8.100	TCP	54	[TCP Keep-Alive] 11001 → 49853 [ACK] Seq=7200 Ack=991 Win=62080 Len=0
33726	128.737961	52.80.28.198	192.168.8.100	TCP	54	[TCP Keep-Alive] 11001 → 49851 [ACK] Seq=7479 Ack=991 Win=62080 Len=0
33727	128.738070	192.168.8.100	52.80.28.198	TCP	54	[TCP Keep-Alive ACK] 49853 → 11001 [ACK] Seq=991 Ack=7201 Win=131072 Len=0
33728	128.738183	192.168.8.100	52.80.28.198	TCP	54	[TCP Keep-Alive ACK] 49851 → 11001 [ACK] Seq=991 Ack=7480 Win=130816 Len=0
33729	129.452595	192.168.8.100	36.152.44.95	ICMP	74	Echo (ping) request id=0x0001, seq=1046/5636, ttl=128 (reply in 33730)
33730	129.557049	36.152.44.95	192.168.8.100	ICMP	74	Echo (ping) request id=0x0001, seq=1046/5636, ttl=54 (request in 33729)
33731	130.463104	192.168.8.100	36.152.44.95	ICMP	74	Echo (ping) request id=0x0001, seq=1047/5892, ttl=128 (reply in 33732)
33732	130.581075	36.152.44.95	192.168.8.100	ICMP	74	Echo (ping) reply id=0x0001, seq=1047/5892, ttl=54 (request in 33731)
33733	130.683534	HuaweiTe_12:e9:0e	Spanning-tree-(for-...	0x2e00	60	Ethernet II
33734	131.482913	192.168.8.100	36.152.44.95	ICMP	74	Echo (ping) request id=0x0001, seq=1048/6148, ttl=128 (reply in 33735)
33735	131.605041	36.152.44.95	192.168.8.100	ICMP	74	Echo (ping) reply id=0x0001, seq=1048/6148, ttl=54 (request in 33734)
33736	132.629107	HuaweiTe_12:e9:0e	Spanning-tree-(for-...	0x2e00	60	Ethernet II
33737	132.629920	52.80.28.198	192.168.8.100	TCP	54	[TCP Keep-Alive] 11001 → 59362 [ACK] Seq=1 Ack=1 Win=489 Len=0
33738	132.630002	192.168.8.100	52.80.28.198	TCP	54	[TCP Keep-Alive ACK] 59362 → 11001 [ACK] Seq=1 Ack=2 Win=513 Len=0
33739	132.877311	fe80::c5c7:85ee:be9...	fe80::8abf:e4ff:fe1...	ICMPv6	86	Neighbor Solicitation for fe80::8abf:e4ff:fe12:e90d from 80:2b:f9:32:2e:07
33740	132.882255	fe80::8abf:e4ff:fe1...	fe80::c5c7:85ee:be9...	ICMPv6	78	Neighbor Advertisement fe80::8abf:e4ff:fe12:e90d (rtr, sol)
33741	133.001325	192.168.8.100	118.126.68.223	TCP	55	[TCP Keep-Alive] 49714 → 443 [ACK] Seq=809 Ack=3140 Win=517 Len=1
33742	133.094687	192.168.8.100	223.6.6.6	TCP	55	[TCP Keep-Alive] 49713 → 443 [ACK] Seq=599 Ack=1936 Win=511 Len=1

(2) 电脑连接华为路由器后，在捕获的列表中看到了 ARP 协议的路由，其源头的名字开头是 Huawei，这应当是路由器使用 ARP 协议时的信息，该报文的显示在了图片下端：



The screenshot shows a Wireshark capture of network traffic on the wlan interface. The packet list on the left shows several ARP requests and replies. The packet details pane on the right shows the structure of an ARP request, including the type, code, identifier, and sequence number. The packet bytes pane at the bottom shows the raw data of the packet.

No.	Time	Source	Destination	Protocol	Length	Info
35816	350.639391	HuaweiTe_12:e9:0e	Spanning-tree-(for-...	0x2e00	60	Ethernet II
35817	350.751471	fe80::8abf:e4ff:fe1...	2409:894c:c00:5e0a:...	ICMPv6	86	Neighbor Solicitation for 2409:894c:c00:5e0a:...
35818	350.751699	2409:894c:c00:5e0a:...	fe80::8abf:e4ff:fe1...	ICMPv6	86	Neighbor Advertisement 2409:894c:c00:5e0a:...
35819	352.687309	HuaweiTe_12:e9:0e	Spanning-tree-(for-...	0x2e00	60	Ethernet II
35820	352.688167	240e:ff:f101:10:1a0...	2409:894c:c00:5e0a:...	TCP	86	[TCP Retransmission] 443 → 49950 [SYN, ACK]
35821	353.549419	192.168.8.100	192.168.43.6	TCP	66	49969 → 7680 [SYN] Seq=0 Win=64240 Len=0
35822	354.555480	192.168.8.100	192.168.43.6	TCP	66	[TCP Retransmission] 49969 → 7680 [SYN] Seq=0
35823	354.633041	HuaweiTe_12:e9:0e	Spanning-tree-(for-...	0x2e00	60	Ethernet II
35824	356.561795	192.168.8.100	192.168.43.6	TCP	66	[TCP Retransmission] 49969 → 7680 [SYN] Seq=0
35825	356.681052	HuaweiTe_12:e9:0e	Spanning-tree-(for-...	0x2e00	60	Ethernet II
35826	358.626610	HuaweiTe_12:e9:0e	Spanning-tree-(for-...	0x2e00	60	Ethernet II
35827	360.573608	192.168.8.100	192.168.43.6	TCP	66	[TCP Retransmission] 49969 → 7680 [SYN] Seq=0
35828	360.674631	HuaweiTe_12:e9:0e	Spanning-tree-(for-...	0x2e00	60	Ethernet II
35829	361.376327	HonHaiPr_32:2e:07	HuaweiTe_12:e9:0d	ARP	42	Who has 192.168.8.1? Tell 192.168.8.100
35830	361.382761	HuaweiTe_12:e9:0d	HonHaiPr_32:2e:07	ARP	42	192.168.8.1 is at 88:bf:e4:12:e9:0d
35831	362.722619	HuaweiTe_12:e9:0e	Spanning-tree-(for-...	0x2e00	60	Ethernet II
35832	364.668312	HuaweiTe_12:e9:0e	Spanning-tree-(for-...	0x2e00	60	Ethernet II
35833	366.716384	HuaweiTe_12:e9:0e	Spanning-tree-(for-...	0x2e00	60	Ethernet II
35834	368.582365	192.168.8.100	192.168.43.6	TCP	66	[TCP Retransmission] 49969 → 7680 [SYN] Seq=0
35835	368.661507	HuaweiTe_12:e9:0e	Spanning-tree-(for-...	0x2e00	60	Ethernet II
35836	370.709001	HuaweiTe_12:e9:0e	Spanning-tree-(for-...	0x2e00	60	Ethernet II

(3) 在浏览器内访问 [www.baidu.com](http://www.baidu.com)，可以看到计算机首先访问到了 DNS 服务器：

41204	694.174607	192.168.8.1	192.168.8.100	DNS
41205	694.174607	192.168.8.1	192.168.8.100	DNS
41206	694.177842	192.168.8.1	192.168.8.100	DNS
41207	694.177842	192.168.8.1	192.168.8.100	DNS
41208	694.177842	192.168.8.1	192.168.8.100	DNS
41209	694.177842	192.168.8.1	192.168.8.100	DNS
41210	694.178521	192.168.8.1	192.168.8.100	DNS
41211	694.178521	192.168.8.1	192.168.8.100	DNS

(4) 因为华为路由器内插了中国移动的卡，所以可以看到路由器同时使用了 IPV6 寻址：

43602	902.827833	fe80::c5c7:85ee:be9...	fe80::8abf:e4ff:fe1...	DNS
43603	902.828219	fe80::c5c7:85ee:be9...	fe80::8abf:e4ff:fe1...	DNS
43604	902.862190	192.168.8.100	185.199.110.153	TLSv1.3
43605	902.920899	192.168.8.100	192.168.8.1	DNS
43606	902.920899	192.168.8.100	192.168.8.1	DNS

(5) 在数据传输的过程中也能看到存在 TCP 数据重传的现象，网络可能出现了拥塞的现象：

43966	969.952203	192.168.8.100	172.217.160.74	TCP
43967	969.983163	192.168.8.100	216.58.200.234	TCP
43968	970.436513	192.168.8.100	216.58.200.234	TCP
43969	970.673403	HuaweiTe_12:e9:0e	Spanning-tree-(for-...	0x2e00
43970	970.943310	192.168.8.100	216.58.200.234	TCP
43971	971.426576	192.168.8.100	216.58.200.234	TCP
43972	971.946426	192.168.8.100	216.58.200.234	TCP
43973	971.993065	192.168.8.100	216.58.200.234	TCP
43974	972.354265	192.168.8.100	120.253.255.97	TCP
43975	972.414136	120.253.255.97	192.168.8.100	TCP
43976	972.720904	HuaweiTe_12:e9:0e	Spanning-tree-(for-...	0x2e00
43977	973.954056	192.168.8.100	216.58.200.234	TCP
43978	973.954065	192.168.8.100	172.217.160.74	TCP
43979	974.445783	192.168.8.100	216.58.200.234	TCP
43980	974.667122	HuaweiTe_12:e9:0e	Spanning-tree-(for-...	0x2e00
43981	975.431841	192.168.8.100	216.58.200.234	TCP
43982	975.999174	192.168.8.100	216.58.200.234	TCP

(6) 当我打开一个视频时，可以看到有大量的数据传输：

正在捕获 WLAN				
文件(F) 编辑(E) 视图(V) 网络(N) 捕获(C) 分析(A) 统计(S) 电话(T) 无线(W) 工具(T) 帮助(H)				
应用显示过滤器: <Ctrl>F				
No.	Time	Source	Destination	Protocol Length Info
57915	1349.782344	2409:8c4c:c00:322:3...	2409:894c:c00:5e0a:...	TCP 1294 443 → 53221 [ACK] Seq=5669935 Ack=1459 Win=27136 Len=1220 [TCP segment of a reassembled PDU]
57916	1349.782398	2409:894c:c00:5e0a:...	2409:8c4c:c00:322:3...	TCP 74 53221 → 443 [ACK] Seq=1459 Ack=5671155 Win=529408 Len=0
57917	1349.832331	2409:8c4c:c00:322:3...	2409:894c:c00:5e0a:...	TLSv1.2 655 Application Data
57918	1349.881163	2409:8c4c:c00:322:3...	2409:894c:c00:5e0a:...	TLSv1.2 1294 Application Data
57919	1349.881163	2409:8c4c:c00:322:3...	2409:894c:c00:5e0a:...	TCP 1294 443 → 53221 [ACK] Seq=5672956 Ack=1459 Win=27136 Len=1220 [TCP segment of a reassembled PDU]
57920	1349.881260	2409:894c:c00:5e0a:...	2409:8c4c:c00:322:3...	TCP 74 53221 → 443 [ACK] Seq=1459 Ack=5674176 Win=529408 Len=0
57921	1349.881555	2409:8c4c:c00:322:3...	2409:894c:c00:5e0a:...	TCP 1294 443 → 53221 [ACK] Seq=5674176 Ack=1459 Win=27136 Len=1220 [TCP segment of a reassembled PDU]
57922	1349.881555	2409:8c4c:c00:322:3...	2409:894c:c00:5e0a:...	TCP 1294 443 → 53221 [ACK] Seq=5675396 Ack=1459 Win=27136 Len=1220 [TCP segment of a reassembled PDU]
57923	1349.881665	2409:894c:c00:5e0a:...	2409:8c4c:c00:322:3...	TCP 74 53221 → 443 [ACK] Seq=1459 Ack=5676616 Win=529408 Len=0
57924	1349.882130	2409:8c4c:c00:322:3...	2409:894c:c00:5e0a:...	TCP 1294 443 → 53221 [ACK] Seq=5676616 Ack=1459 Win=27136 Len=1220 [TCP segment of a reassembled PDU]
57925	1349.882130	2409:8c4c:c00:322:3...	2409:894c:c00:5e0a:...	TCP 1294 443 → 53221 [ACK] Seq=5677836 Ack=1459 Win=27136 Len=1220 [TCP segment of a reassembled PDU]
57926	1349.882172	2409:894c:c00:5e0a:...	2409:8c4c:c00:322:3...	TCP 74 53221 → 443 [ACK] Seq=1459 Ack=5679056 Win=529408 Len=0
57927	1349.883310	2409:8c4c:c00:322:3...	2409:894c:c00:5e0a:...	TLSv1.2 1294 Application Data [TCP segment of a reassembled PDU]
57928	1349.883351	2409:894c:c00:5e0a:...	2409:8c4c:c00:322:3...	TCP 74 53221 → 443 [ACK] Seq=1459 Ack=5680276 Win=529408 Len=0
57929	1349.922597	2409:8c4c:c00:322:3...	2409:894c:c00:5e0a:...	TLSv1.2 1294 Application Data, Application Data, Application Data
57930	1349.942341	2409:8c4c:c00:322:3...	2409:894c:c00:5e0a:...	TLSv1.2 1800 Application Data, Application Data, Application Data
57931	1349.942386	2409:894c:c00:5e0a:...	2409:8c4c:c00:322:3...	TCP 74 53221 → 443 [ACK] Seq=1459 Ack=5682502 Win=529408 Len=0
57932	1349.945130	49.235.252.228	192.168.8.100	TLSv1.3 491 Application Data
57933	1349.945181	2409:8c4c:c00:322:3...	2409:894c:c00:5e0a:...	TLSv1.2 1294 Application Data
57934	1349.945181	2409:8c4c:c00:322:3...	2409:894c:c00:5e0a:...	TCP 1294 443 → 53221 [ACK] Seq=5683722 Ack=1459 Win=27136 Len=1220 [TCP segment of a reassembled PDU]
57935	1349.945210	2409:894c:c00:5e0a:...	2409:8c4c:c00:322:3...	TCP 74 53221 → 443 [ACK] Seq=1459 Ack=5684942 Win=529408 Len=0
57936	1349.945793	2409:8c4c:c00:322:3...	2409:894c:c00:5e0a:...	TCP 1294 443 → 53221 [ACK] Seq=5684942 Ack=1459 Win=27136 Len=1220 [TCP segment of a reassembled PDU]
57937	1349.945793	2409:8c4c:c00:322:3...	2409:894c:c00:5e0a:...	TCP 1294 443 → 53221 [ACK] Seq=5686162 Ack=1459 Win=27136 Len=1220 [TCP segment of a reassembled PDU]
57938	1349.945840	2409:894c:c00:5e0a:...	2409:8c4c:c00:322:3...	TCP 74 53221 → 443 [ACK] Seq=1459 Ack=5687382 Win=529408 Len=0
57939	1349.946313	2409:8c4c:c00:322:3...	2409:894c:c00:5e0a:...	TCP 1294 443 → 53221 [ACK] Seq=5687382 Ack=1459 Win=27136 Len=1220 [TCP segment of a reassembled PDU]
57940	1349.946313	2409:8c4c:c00:322:3...	2409:894c:c00:5e0a:...	TCP 1294 443 → 53221 [ACK] Seq=5688602 Ack=1459 Win=27136 Len=1220 [TCP segment of a reassembled PDU]
> Frame 1: 152 bytes on wire (1216 bits), 152 bytes captured (1216 bits) on interface \Device\NPF_{81543F99-5570-45C0-9C66-C26A7A728CF4}, id 0				
> Ethernet II, Src: HontasPr_32:2e:07 (80:2b:f9:32:2e:07), Dst: HuaweiTe_12:e9:0d (88:bf:e4:12:e9:0d)				
> Internet Protocol Version 4, Src: 192.168.8.100, Dst: 52.80.28.198				
0000	88 bf e4 12 e9 0d 80 2b f9 32 2e 07 08 00 45 00	.....+.2...E		
0010	00 8a d5 71 40 00 00 06 0a da c0 a0 00 64 34 50	...@.....dAP		
0020	1c c6 f3 25 2a f9 70 54 5c 61 2c bd 7e b5 50 18	...%*pT \a...P		
0030	02 01 82 c8 00 00 14 bf 2d 5a 8f 1c 98 7d d4 f1	.....-Z...).		
0040	08 3a ca 92 a7 4a ab 20 14 20 13 dc 15 c7 e1 3d	.....J.....		
0050	8f 26 fd 33 67 87 01 63 26 e9 44 f0 e4 22 fb d6	.&3g..c &D...*		
0060	81 58 ee 75 fd e3 f6 de f4 f9 41 13 66 21 fd 20	X.u.....-A.f1.		
0070	db b7 9c ea 2a 56 f9 29 00 38 3b 31 80 62 fd d6	...V...) &1-b-		
0080	a7 0d be 80 a9 06 7d c3 78 76 d5 cc af 16 83 73	.....}. xv.....S		
0090	46 e8 7e a9 d2 1b 32 c6	F.....2.		

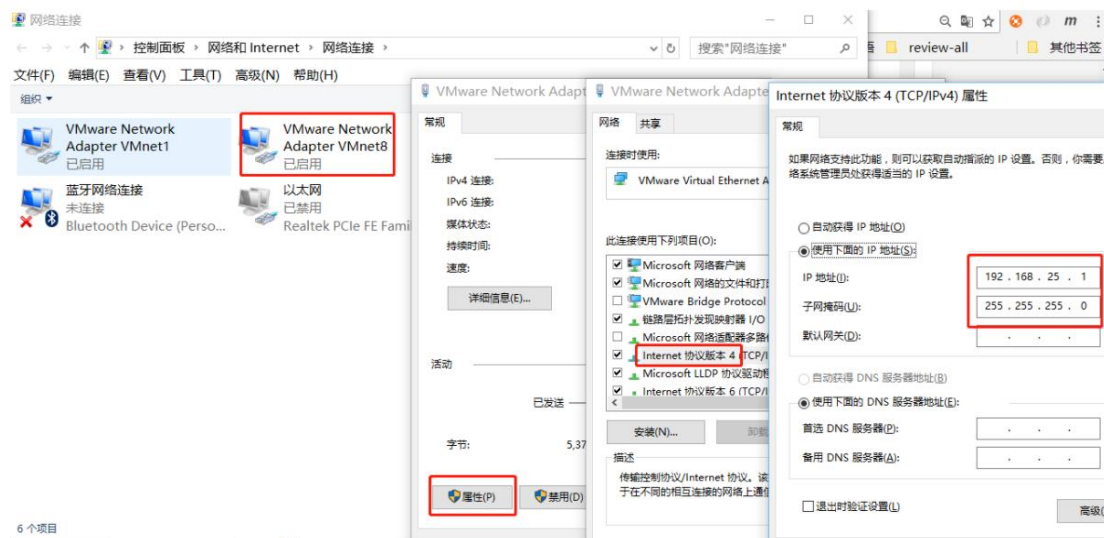
## 四、主要遇到的问题

### 1、SSH 无法登录

问题描述：在从机房回到宿舍连接上热点后，发现按照整合步骤无法从电脑 Windows 系统通过 SSH 登录到虚拟机。

问题原因：在从机房回到宿舍后，当电脑连接上热点时，IP 地址发生了变化，并且和虚拟机的 IP 不在一个网段了，所以主机不能连接到虚拟机。

解决方法：查看网卡和 Windows 的 IP 地址，对 VMware 虚拟机的网卡修改 IP 地址，使其与主机在同一网段。



### 2、虚拟机可以 ping 通主机，但是主机无法 ping 到虚拟机

问题描述：电脑从连接校园网变为连接热点后，主机 ping 不通虚拟机，但是虚拟机 ping 通主机。

问题原因：原因同上，当路由器更换时，可能会出现 IP 网段变化的情况，这时候如果想继续是研究需要手动更换虚拟机 IP。

解决方法：虚拟机网络连接方式选择 Nat，关闭 Linux 防火墙命令：service iptables stop / service firewalld stop，查看 Linux 防火墙状态命令：service iptables status / service firewalld status。关闭 windows 防火墙，查看网卡和 Windows 的 IP 地址，对 VMware 虚拟机的网卡修改 IP 地址，使其与主机在同一网段。



### 3、在安装 PHP 工具时，安装失败

问题描述：安装 PHP 工具时报错，无法正常安装。

问题原因：PHP 工具与 CentOS 版本不匹配，可以通过更新 CentOS 的 yum 工具来解决。

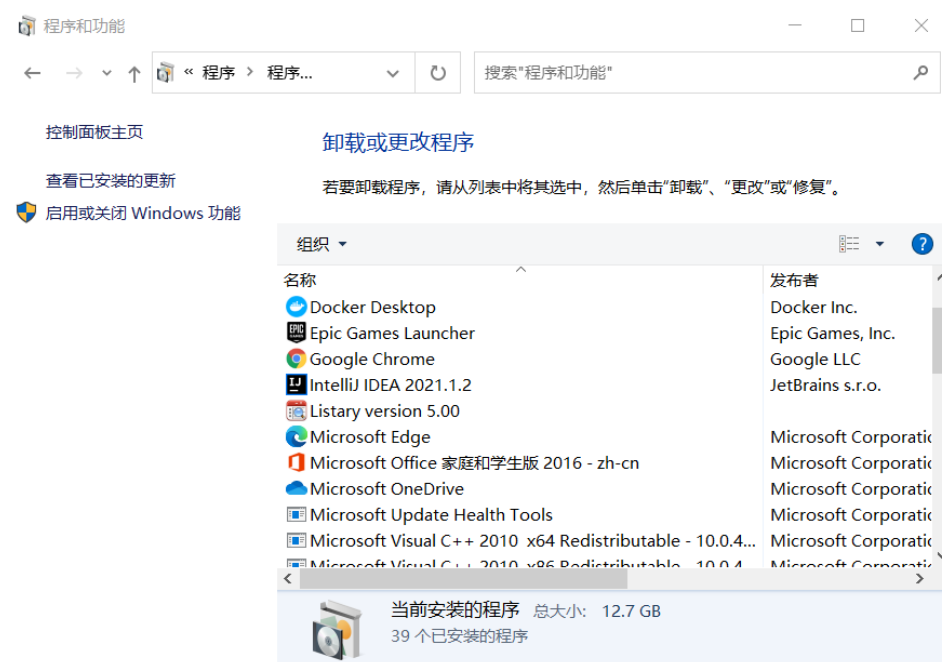
解决方法：因为更新 yum 的过程中断网，导致更新失败，因此直接使用 yum install PHP，同样安装成功了 PHP 工具，可以直接使用！

### 4、从 Windows 连接 Samba 服务失败

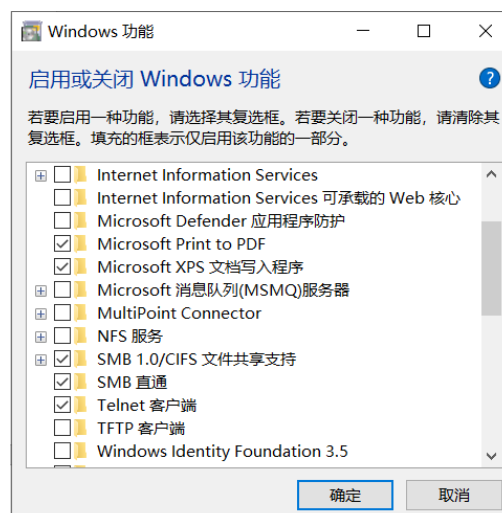
问题描述：完成了 Samba 服务的安装和配置后无法从文件夹访问 Samba

问题原因：没有开启 Windows 端的 Samba 服务，需要双方都打开。

解决方法：打开程序与功能：选择启用或关闭 Windows 功能



打开 Samba 服务，重启后，可以成功访问！





## 五、总结

通过这次实践，我对计算机网络有了必要的认识，这对我以后的学习有很大的帮助，研究 TCP/IP 协议，使我们对 TCP/IP 协议有了深入的了解，并且熟悉它在网络中的应用的问题。正是有了 TCP/IP 协议，才有了今天“地球村”因特网的巨大发展。TCP/IP 协议作为网络层中最基本也是最重要的协议，随着现代科技的进步与发展，协议中的许多不足也在不断地体现出来，用户的增加是 TCP/IP 首要解决的问题，同时由于用户的激增，传输速率也成为人们关心的热点，以及传输的有效数据。

我们也学会了构建基本的网络结构，能够实现各个主机之间的通信。通过查资料和敲命令行代码，不仅对于各种协议真正理解了他的作用所在，而且对主机、虚拟机、路由器的配置和一些路由算法有了更清晰的认识。对于服务器也深刻理解了它的做用。

在查阅了各种资料后，我也意识到计算机网络技术已经渗透到生活中的各个领域，给我们的生活带来了极大的便利，未来计算机网络将更加人性化，更加适应人们的生活。网络技术的高速发展为各行业的生命注入了新的血液，同时对各行业的发展也是一个考验，人们将更加离不开网络，而计算机也将更好地服务于人类，使人们的生活更加丰富。

## 六、参考与引用

[1] <https://cloud.tencent.com/developer/article/1626729>

[2] <https://github.com/moranzcw/Computer-Networking-A-Top-Down-Approach-NOTES>

[3] <https://blog.csdn.net/lisayh/article/details/108055516>

[4] <https://blog.csdn.net/TJU355/article/details/7098372>

[5] <https://zhuanlan.zhihu.com/p/346933068>

[6] [https://blog.csdn.net/qq\\_18662865/article/details/51454136](https://blog.csdn.net/qq_18662865/article/details/51454136)