实验1：网络概念、协议和常用网络工具

段海新，王浩铭

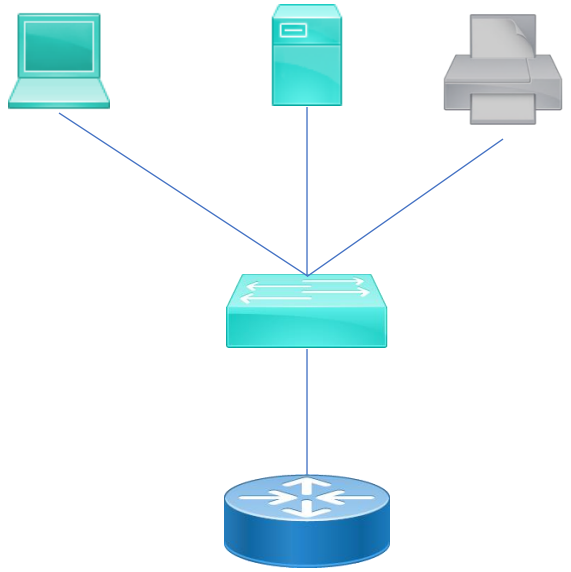
2024/09/12

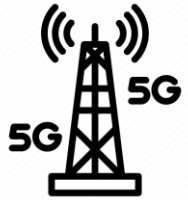
实验目的

- 计算机网络基本概念和协议
  - 计算机网络和互联网
  - TCP/IP协议族和 OSI
  - 常见的网络设备和网络服务
  - 子网划分子网掩码
- 常用的网络工具
  - Wireshark
  - TCPDUMP
  - Scapy
  - Burp Suite

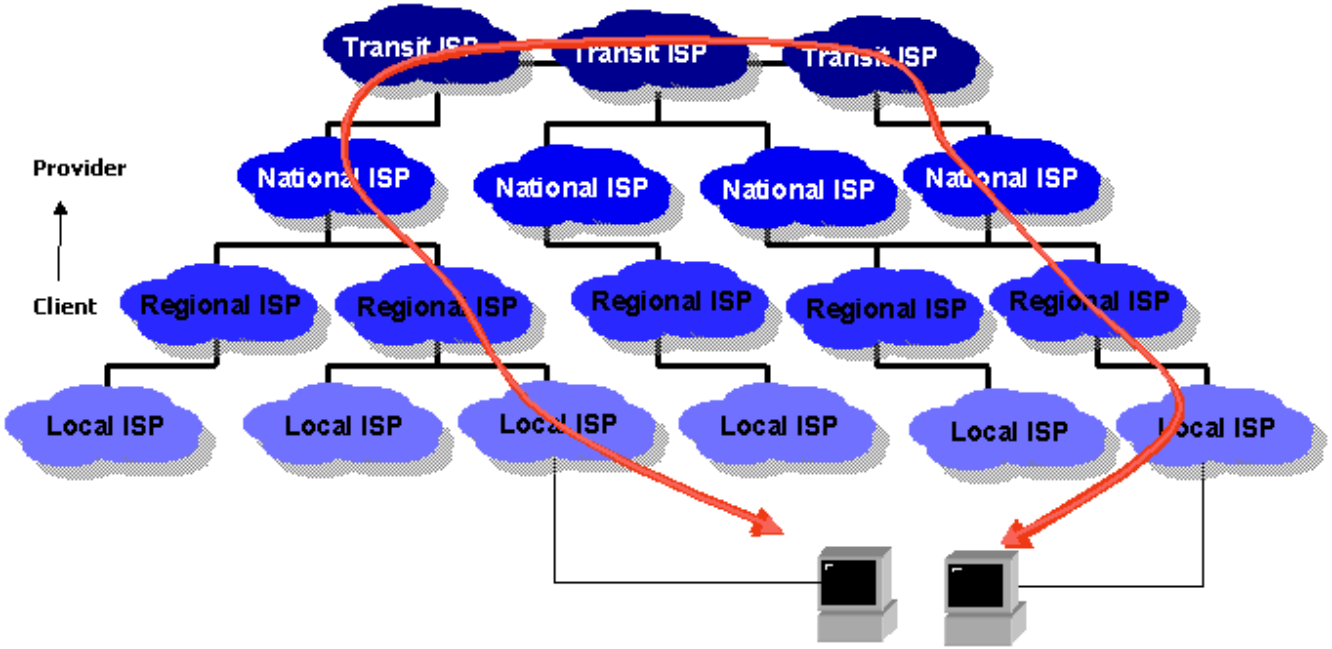
计算机网络

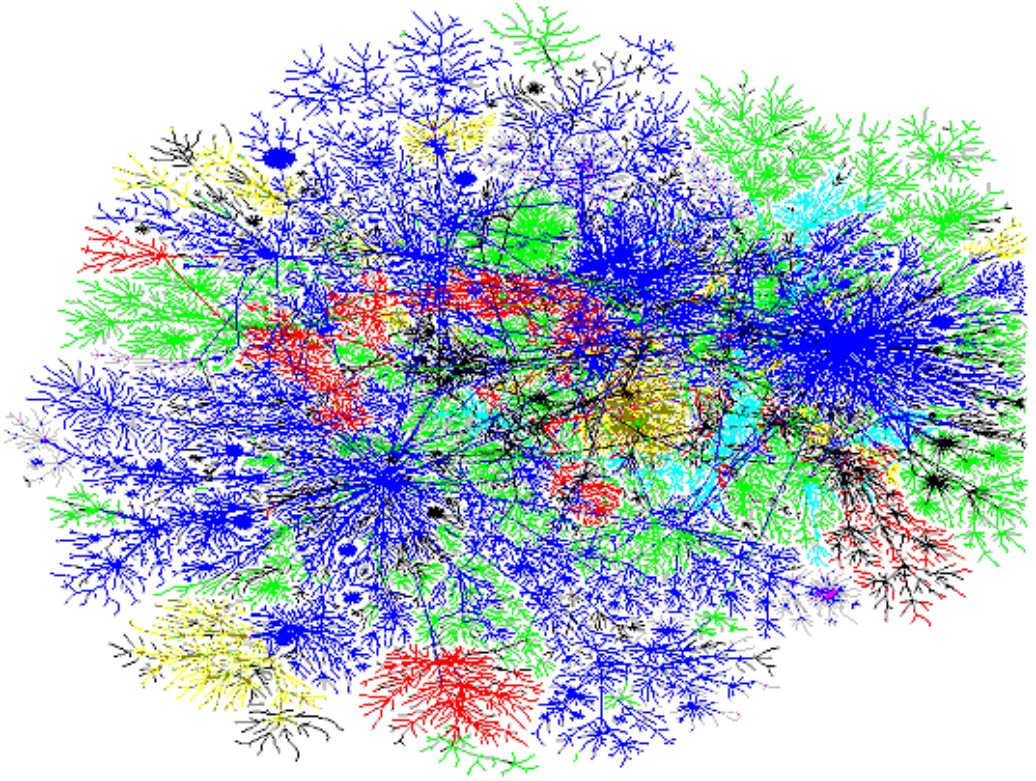
- 计算机网络：将独立的计算机及外部设备，通过通信媒体或网络设备互联，以实现通信或资源共享的目的。
- 通信媒体
  - 有线：同轴电缆、双绞线、光纤
  - 无线：蓝牙，WiFi，3/4/5G、卫星…
- 网络设备
  - 路由器、交换机





互联网(Internet) 世界上最大的“网络的网络”

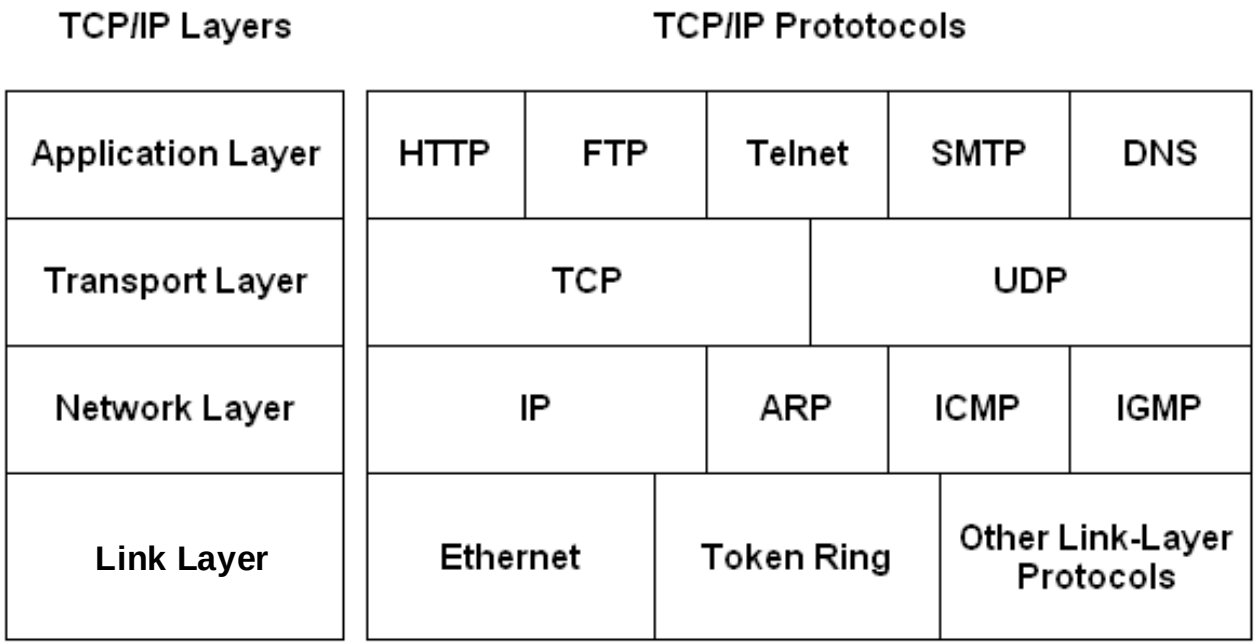




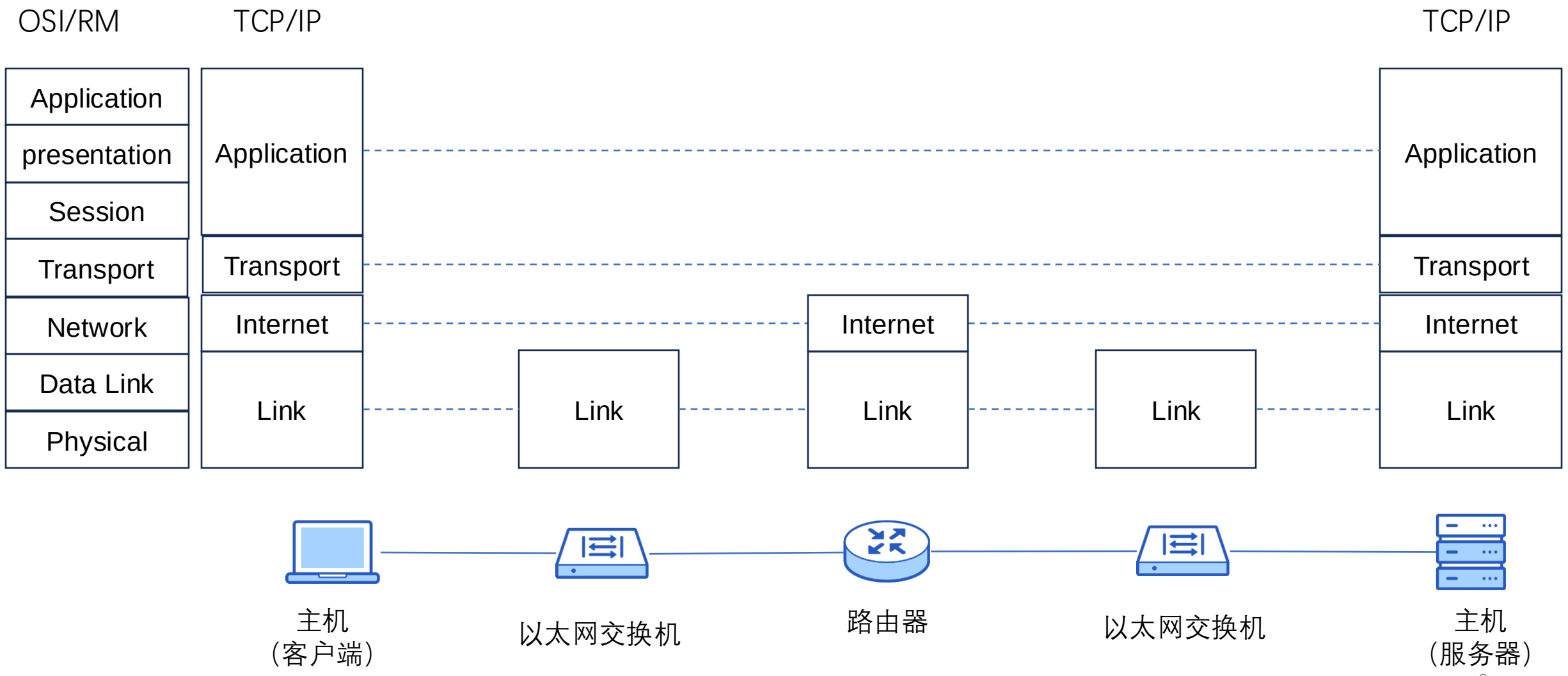
https://www.isoc.org/inet99/proceedings/1e/1e_1.htm

TCP/IP 协议族

- OSI/RM只是概念，没有实现
  - OSI/RM只是个美丽的梦，而TCP/IP生活在梦一样的现实里
- 但是OSI七层模型的概念被主流网络教材采纳，故课程也使用这些概念，如
  - 链路层是第二层
  - 网络层对应 internet layer
  - 应用层协议是“第七层”协议



OSI/RM的七层模型 和 TCP/IP的四层模型



链路层设备：以太网交换机

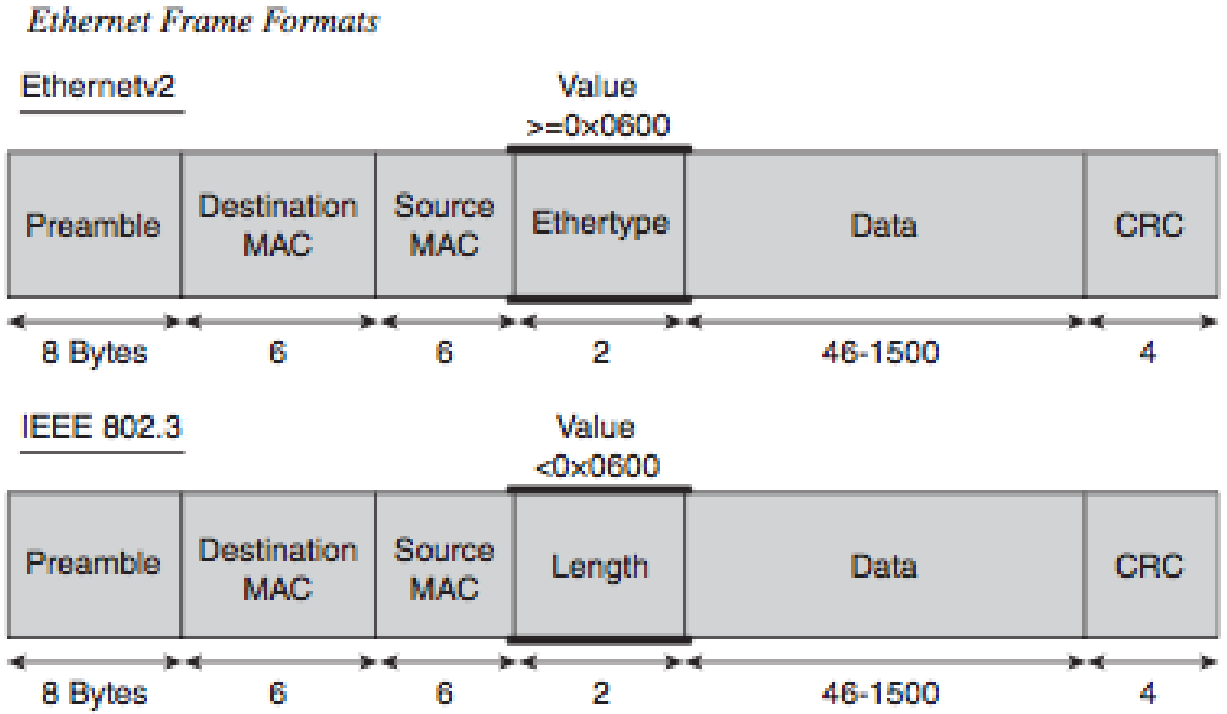
常见的以太网交换机外形

本课程中的图标

以太网帧（Frame）结构

Ethernet Frame： 802.3 , Ethernet II, SNAP

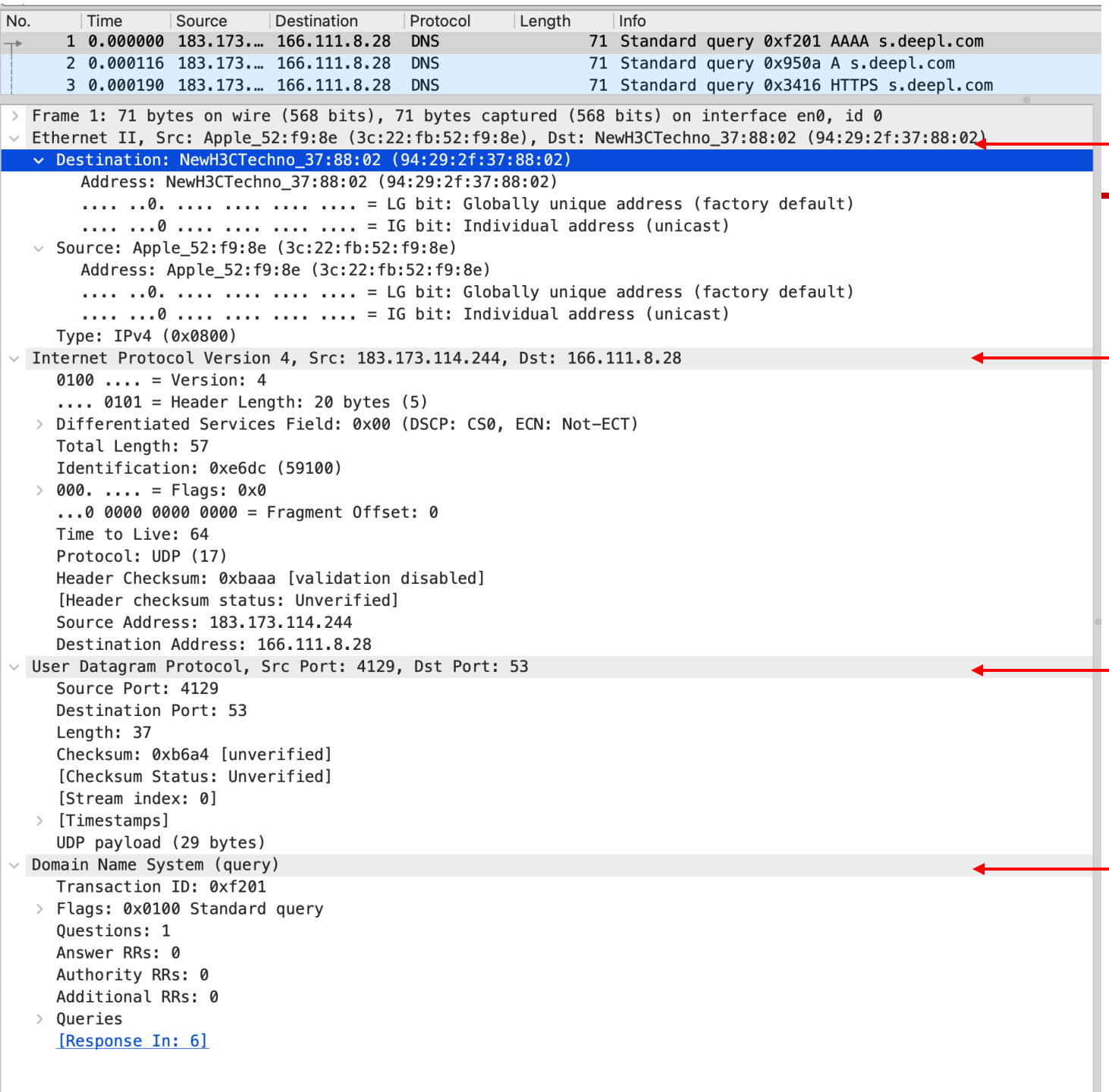
http://standards.ieee.org/getieee802/802.3.html



MAC Address： Burned-In-Address

Broadcast Address： FF-FF-FF

MAC地址可以修改吗？



```
No.     Time       Source          Destination     Protocol  Length  Info
    1   0.000000   183.173.…       166.111.8.28    DNS         71    Standard query 0xf201 AAAA s.deepl.com
    2   0.000116   183.173.…       166.111.8.28    DNS         71    Standard query 0x950a A s.deepl.com
    3   0.000190   183.173.…       166.111.8.28    DNS         71    Standard query 0x3416 HTTPS s.deepl.com

> Frame 1: 71 bytes on wire (568 bits), 71 bytes captured (568 bits) on interface en0, id 0
v Ethernet II, Src: Apple_52:f9:8e (3c:22:fb:52:f9:8e), Dst: NewH3CTechno_37:88:02 (94:29:2f:37:88:02)
  v Destination: NewH3CTechno_37:88:02 (94:29:2f:37:88:02)
      Address: NewH3CTechno_37:88:02 (94:29:2f:37:88:02)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  v Source: Apple_52:f9:8e (3c:22:fb:52:f9:8e)
      Address: Apple_52:f9:8e (3c:22:fb:52:f9:8e)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: IPv4 (0x0800)
v Internet Protocol Version 4, Src: 183.173.114.244, Dst: 166.111.8.28
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 57
    Identification: 0xe6dc (59100)
  > 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 64
    Protocol: UDP (17)
    Header Checksum: 0xbaaa [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 183.173.114.244
    Destination Address: 166.111.8.28
v User Datagram Protocol, Src Port: 4129, Dst Port: 53
    Source Port: 4129
    Destination Port: 53
    Length: 37
    Checksum: 0xb6a4 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 0]
  > [Timestamps]
    UDP payload (29 bytes)
v Domain Name System (query)
    Transaction ID: 0xf201
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  > Queries
    [Response In: 6]
```

数据链路层（以太网）

网络层（IPv4）

传输层（UDP）

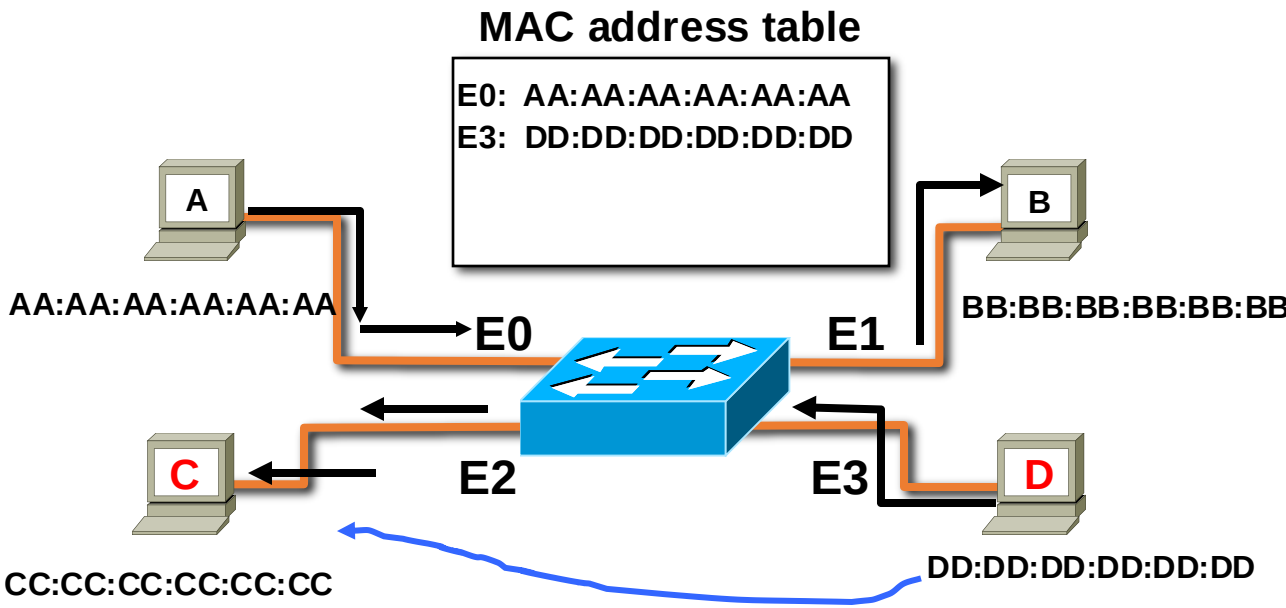
应用层（DNS）

MAC地址的组成： 48比特：厂商+序列号

```
Frame 1: 71 bytes on wire (568 bits), 71 bytes captured (568 bits) on interface en0, id 0
Ethernet II, Src: Apple_52:f9:8e (3c:22:fb:52:f9:8e), Dst: NewH3CTechno_37:88:02 (94:29:2f:37:88:02)
  Destination: NewH3CTechno_37:88:02 (94:29:2f:37:88:02)
     Address: NewH3CTechno_37:88:02 (94:29:2f:37:88:02)
     .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
     .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  Source: Apple_52:f9:8e (3c:22:fb:52:f9:8e)
     Address: Apple_52:f9:8e (3c:22:fb:52:f9:8e)
     .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
     .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 183.173.114.244, Dst: 166.111.8.28
```

以太网交换机工作原理

- 交换机维护地址转发表 <Port, MAC>
- 根据数据帧的目标 MAC 地址查表转发
- 未知则广播（Broadcast, Flood）
- 对于目标地址为FF:FF:FF:FF 的广播



例：

- **A --> D： 广播， 交换机学到 < E0, A>**
- **D-->A： 只发给 A, 交换机学到 <E3, D>**

问题：D→C: 交换机将怎样转发?

目标 MAC	源 MAC	
CC:CC:CC:CC:CC	DD:DD:DD:DD:DD	DATA

FF:FF:FF:FF:FF:FF	DD:DD:DD:DD:DD	DATA

问题：发送方是如何得知接收方的 MAC 地址的?

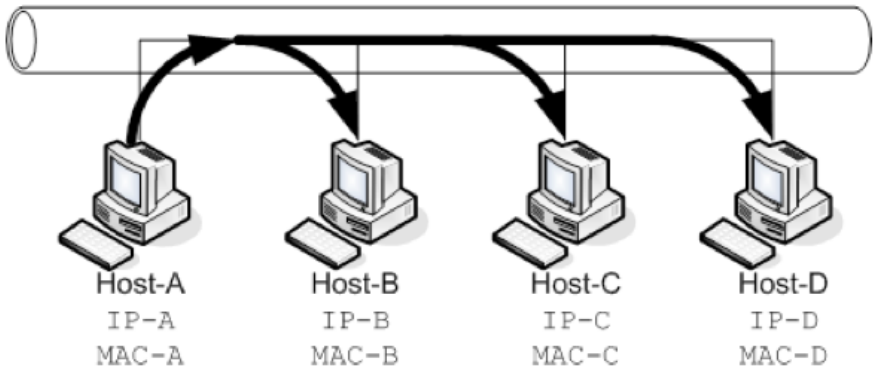
局域网中的标识与认证

- 不同层次的标识：
  - 应用层：userid（本地），全球唯一：email address，domain name
  - 网络层：IP Address（全球唯一，私有地址除外）
  - 数据链路层：MAC address （用于本地寻址）

- Domain name ←→ IP : DNS
  - What is your DNS server?
  - Is the reply from your DNS server true?
- IP ←→ MAC
  - Address Resolution Protocol(ARP)
  - Reverse ARP→ BOOTP → DHCP
- DHCP
  - 根据MAC地址分配：IP, Gateway, DNS, Proxy…

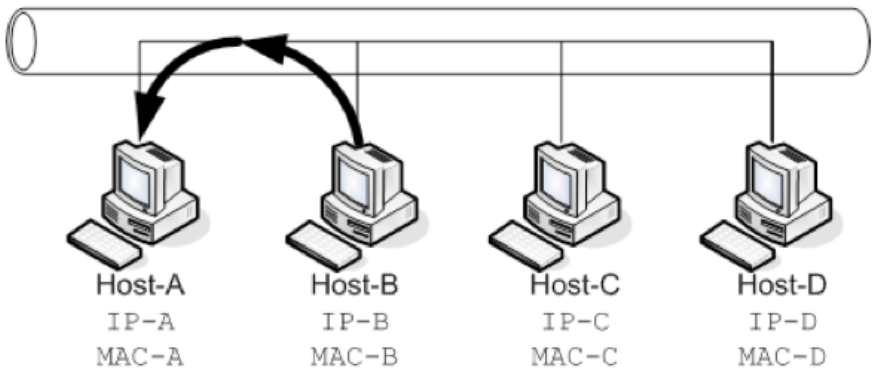
MAC 地址解析： ARP Request and Reply

Ethernet	
Destination:	FF:FF:FF:FF:FF:FF
Source:	MAC-A
Type:	ARP
ARP	
Opcode	**Request**
Sender MAC	MAC-A
Sender IP	IP-A
Target MAC	0:0:0:0:0:0
Target IP	**IP-B**



A > @All : 谁的IP地址是IP-B? 告诉我你的MAC地址

Ethernet	
Destination:	MAC-A
Source:	MAC-B
Type:	ARP
ARP	
Opcode	**Reply**
Sender MAC	**MAC-B**
Sender IP	**IP-B**
Target MAC	MAC-A
Target IP	IP-A

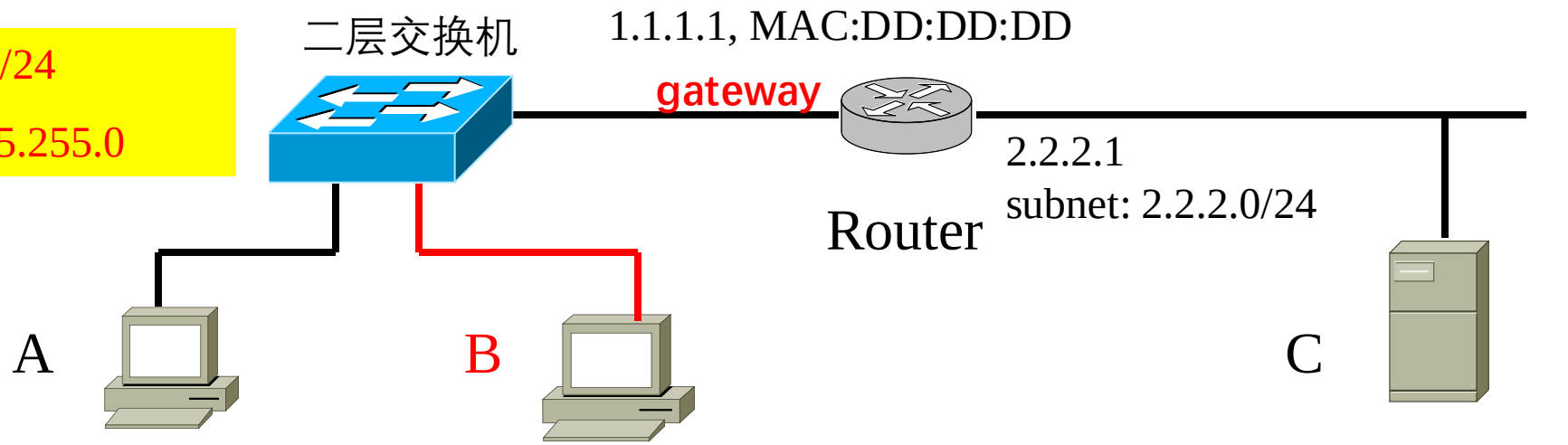


B →A :这是我的MAC地址 MAC-A

什么时候进行ARP 地址解析?

Subnet: 1.1.1.0/24

Net Mask: 255.255.255.0



A> ping 1.1.1.3 #B

A计算目标的子网(根据本地的 subnet mask):

```
    1.  1.  1.  3
&
255.255.255.  0
-----------------
    1.  1.  1.  0
```

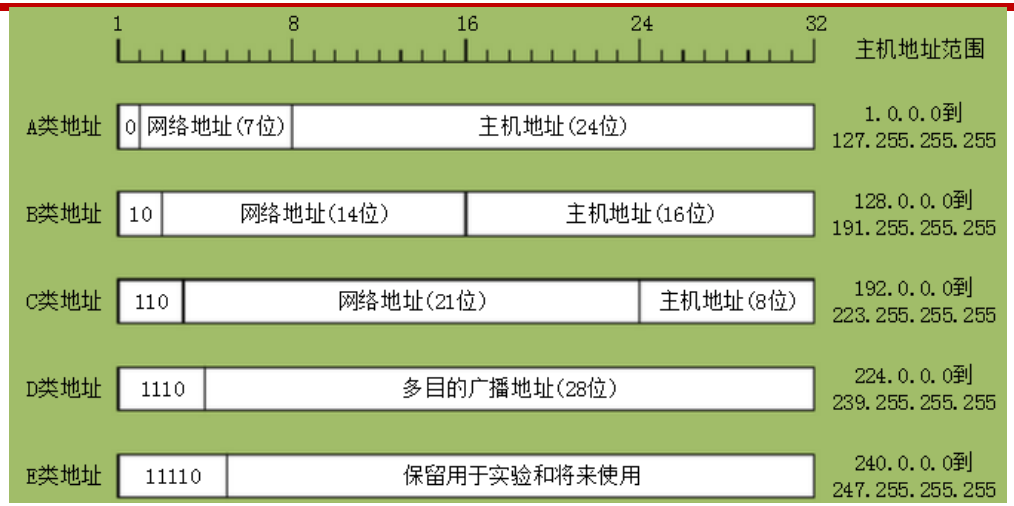
A> ping 2.2.2.20 #C

A计算目标的子网(根据本地的 subnet mask):

```
    2.  2.  2. 20
&
255.255.255.  0
-----------------
    2.  2.  2.  0
```

子网划分和子网掩码

- RFC 791 中的五类 IP 地址
  - A: 1.0.0.0~127.255.255.255
  - B: 128.0.0.0~ 191.255.255.255
  - C: 192.0.0.0 ~ 223.255.255.255
- 子网掩码（Subnet Mask）
  - NetID + HostID
  - 掩码： subnet mask :
  - 子网计算： IP & subnet mask
  - 地址前缀： 子网掩码中 1的位数
- 私有地址(RFC1918)
  - 10.0.0.0/8
  - 172.16.0.0/12
  - 192.168.0.0/16

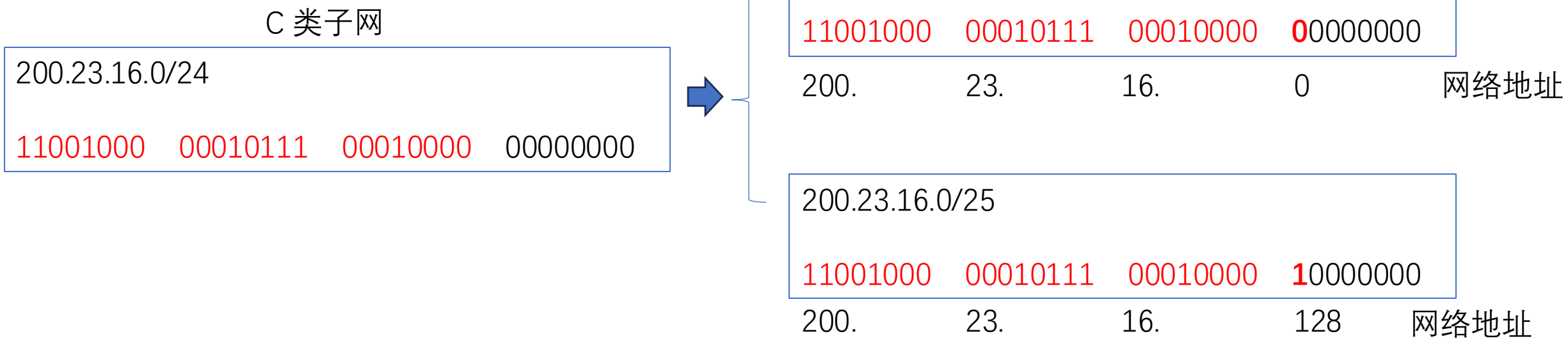


IP Address : 166. 111. 4. 100
Subnet Mask : 255. 255. 0. 0
Subnet : 166. 111. 0. 0
Prefix : /16

IP Address : 202. 112. 5. 1
Subnet Mask : 255. 255. 255. 0
Subnet : ?
Prefix : /?

无分类编址（Classless Inter-Domain Routing）

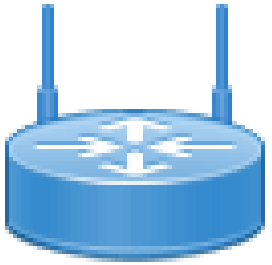
- 互联网规模扩大， IPv4地址空间耗尽
- 提高IPv4 地址空间的效率，消除传统A、B、C 类地址的界限
- 用任意长度的地址前缀取代传统固定长度的掩码
  - a.b.c.d/prefix



网络层设备：路由器

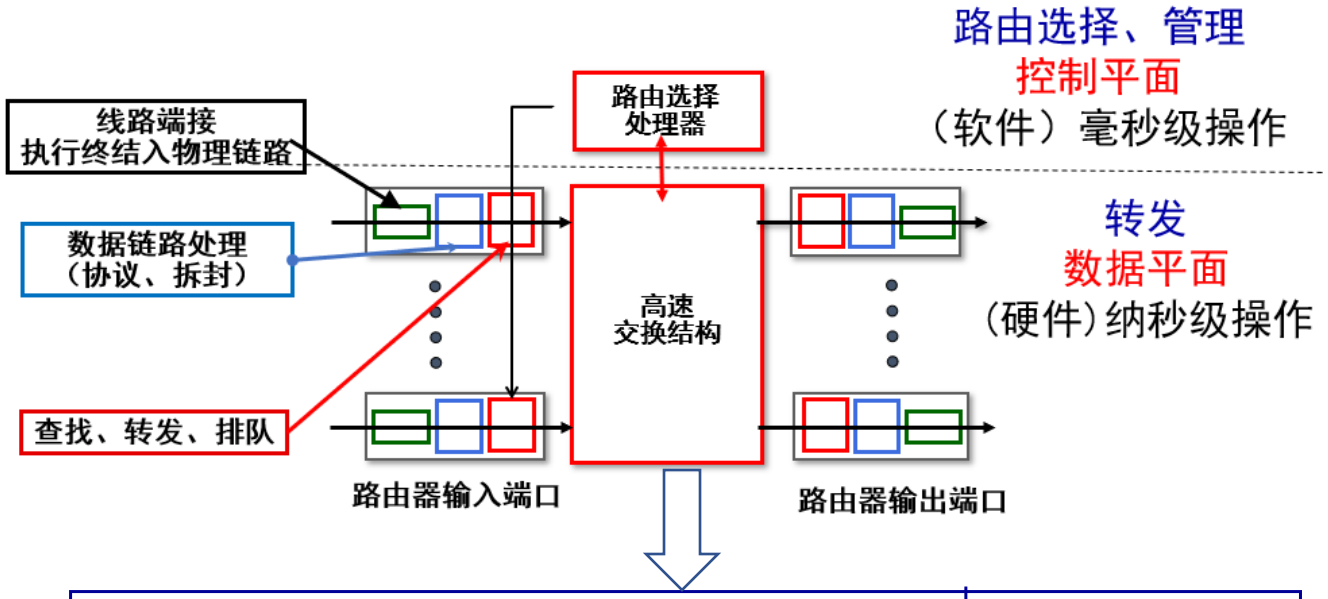
常见的路由器外形

本课程中的图标



路由器的转发功能

- 路由器的转发原理：
  - 根据目的IP地址、查找路由器内部的路由表，把数据包转发到路由器相应的输出端口
- 路由表的构建
  - 静态路由：手动配置路由表
  - 动态路由：路由器之间通过交换的路径信息自动学习
- 未知目的地址：？
- 对于广播地址： 255.255.255.255



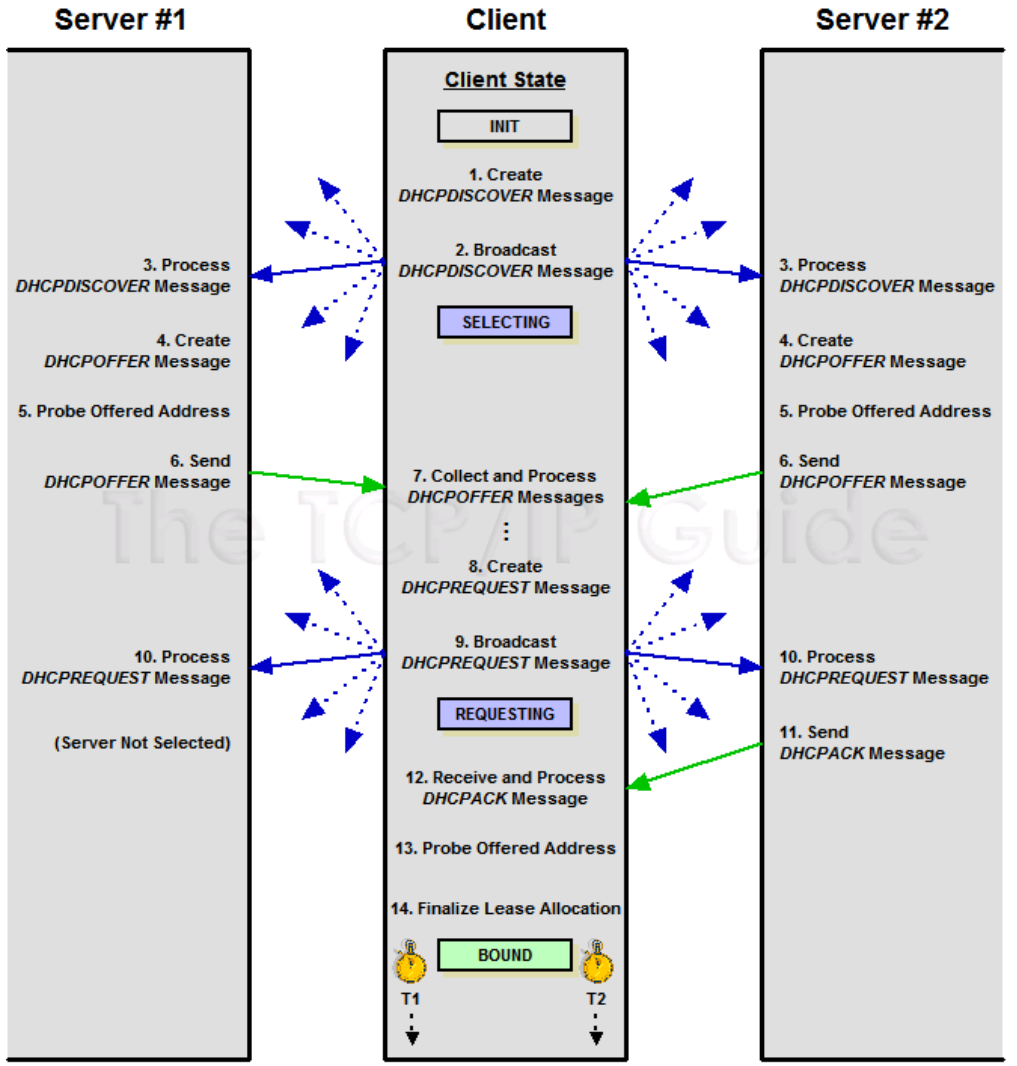
目的地址范围	链路接口
11001000 00010111 00010*** ********	0
11001000 00010111 00011000 ********	1
11001000 00010111 00011*** ********	2
otherwise	3

提问: 11001000 00010111 00011000 10101010
路由器会转发到哪个出口?

基础服务: 动态地址配置服务（DHCP）

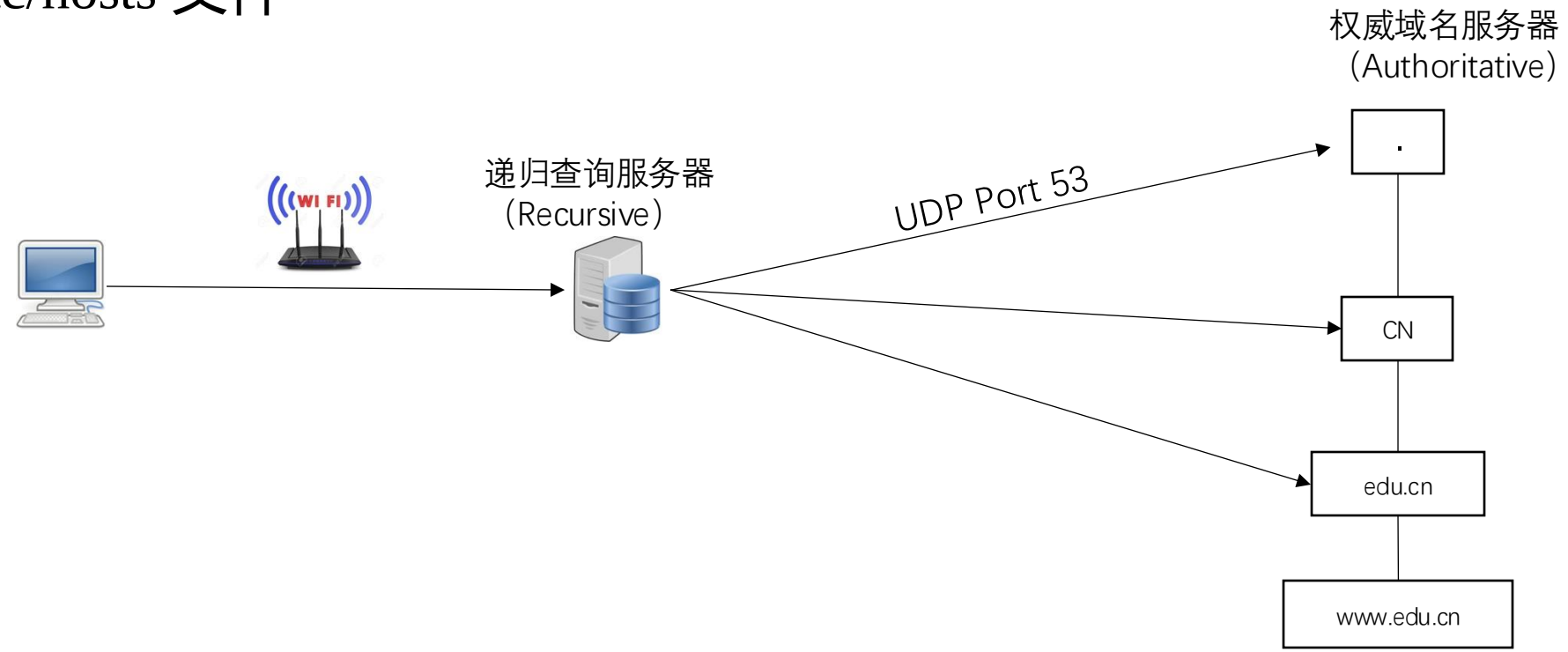
- 为接入的网络终端自动分配IP地址，使得设备可以更方便地连接到网络并获得所需的网络配置。
  - IP 地址及 掩码（netmask）
  - DNS 服务器(name server)
  - 路由器（default gateway）
  - Web 服务代理（Proxy）
  - …
- 客户端接入时靠广播发现 DHCP 服务器

问题：如果局域网中存在多个 DHCP 会怎样?



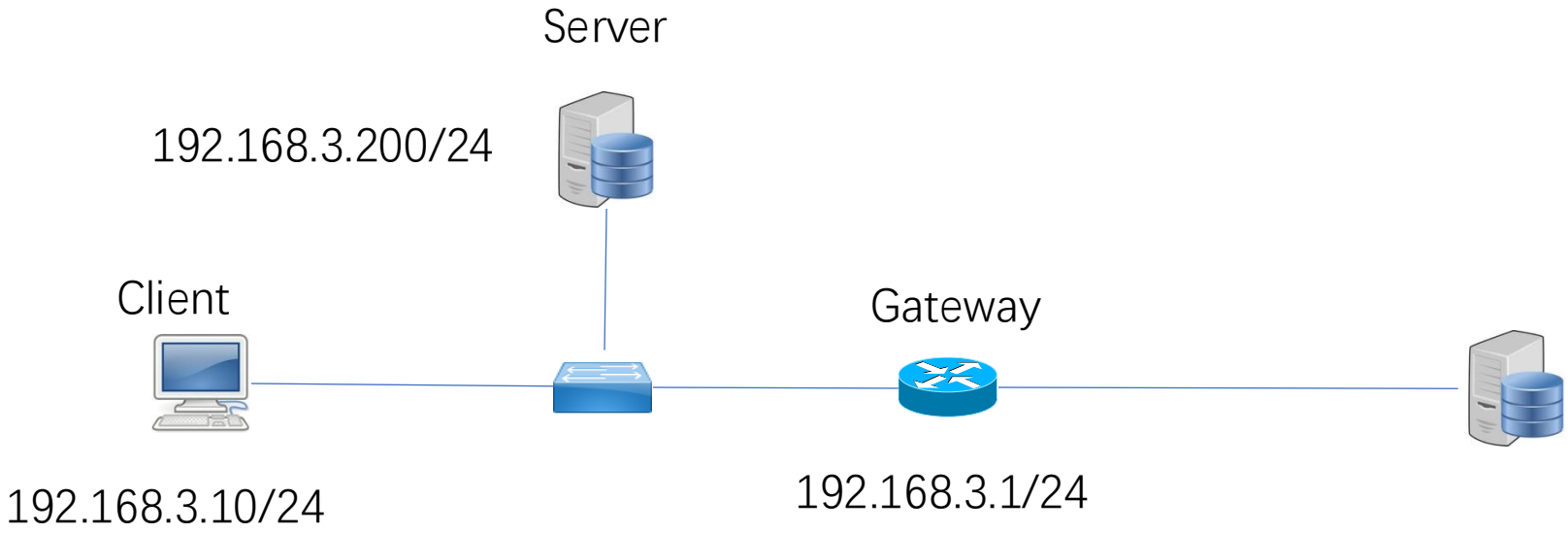
基础服务: 域名服务系统（DNS）

- 域名服务系统（DNS）把用户容易记忆的主机名转化成 IP 地址
- 由DHCP动态分配
- 手工配置 /etc/resolv.conf
- /etc/hosts 文件



实验1 的网络环境中的子网划分

- 这个网络中有几个子网?
- 每个子网的子网掩码是什么?

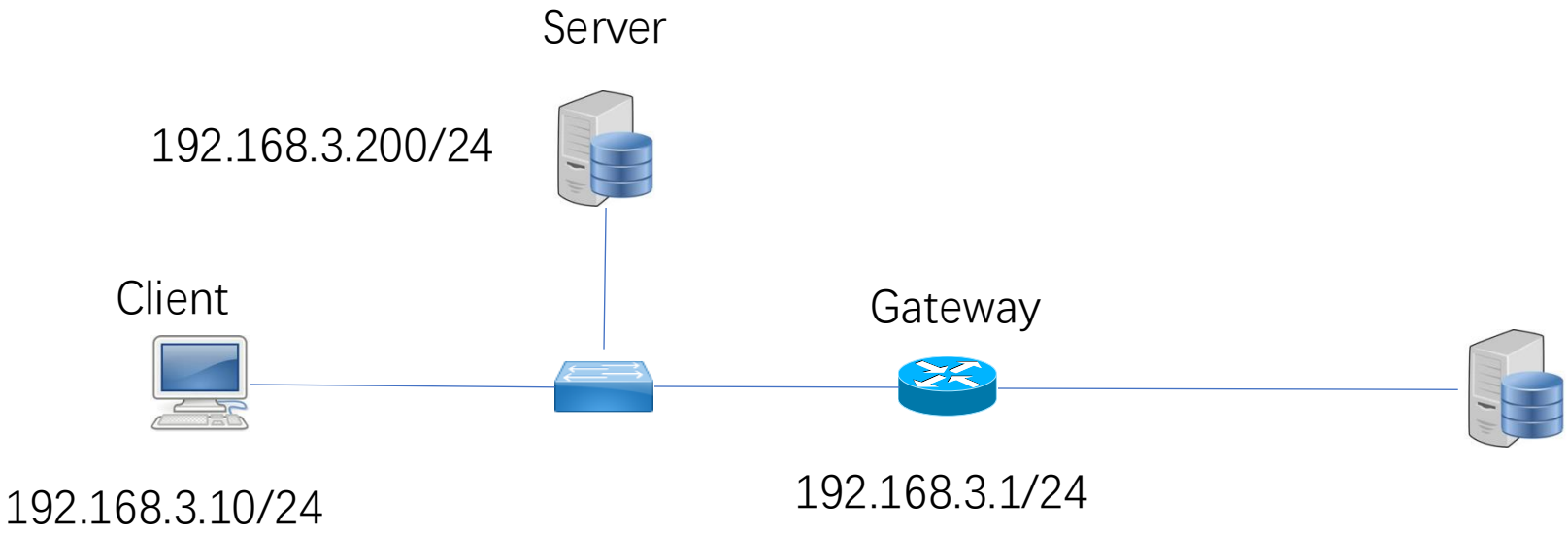


一个ping 命令产生的网络流量

- 在Client （192.168.3.10）上执行
  Client> Ping 192.168.3.200
  会产生什么流量?

地址解析协议（ARP）
ARP Request:
C ->FF:FF:FF： Who-has the IP 192.168.3.200 ?

ARP Reply:
S ->C: 192.168.3.200 is-at S



一个 Web 访问上网的实例

- 在浏览器中输入 www.imool.com.cn
- 域名解析：得到目标域名的 IP
  - 49.7.50.131
- 网关的MAC 地址解析
  - ARP : who has the IP ?
- 建立 TCP 连接：三次握手
- HTTP 请求与相应
  - GET

常用网络相关命令

- Linux 系统
  - ifconfig　　//查看网卡信息
  - ping　　//发起ICMP请求
  - traceroute　　//分析网络路径
  - netstat　　//查看端口信息
  - dig　　//发起DNS请求
  - tcpdump　　//流量抓取
  - arp　　//查看arp信息
  - nmap　　//扫描目标主机开放端口

常用网络工具1：tcpdump

tcpdump [option] expression

```
tcpdump -i eth0 -w file.pcap host 8.8.8.8
```

Option

-D：列出所有网卡的名字和当前状态。

-c count：收到指定的count个数据包以后命令退出。

-e: 显示数据包的链路层头信息，。

-i interface：指定监听的网卡名称。

-n：不把 IP地址反向解析成域名。

-r file：从文件file中读取数据包，而不是从网卡监听数据。

-s snaplen： 从每个数据包中剪切 snaplen 字节的数据。

-v：在解析和打印时，会产生（稍多一点的）冗长输出。

-vv 和 -vvv：打印更加详细的输出。

-w file：捕获的数据直接写入file文件。

Expression : Berkeley Packet Filter(BPF)语法

```
net 192.168.1.0/24 and udp port 53
```

每个表达式包含一个或多个原语（primitive)。每个原语包含一个或多个限定词（qualifier)，然后跟着一个ID名字或者数字

限定词	说明	实例
type	指出名字或数字的意义	host, net, port
dir	传输的方向或来源	src, dst
proto	限定词所匹配的协议	ether, ip, tcp, udp, http, ftp,…

详细信息，请参考TCP MANUAL：

HTTPS://WWW.TCPDUMP.ORG/MANPAGES/TCPDUMP.1.HTML

流量分析利器： tcpdump 分析 HTTP 流量

curl http://www.edu.cn

tcpdump -i en0 -n –v –X -host www.edu.cn and port 80

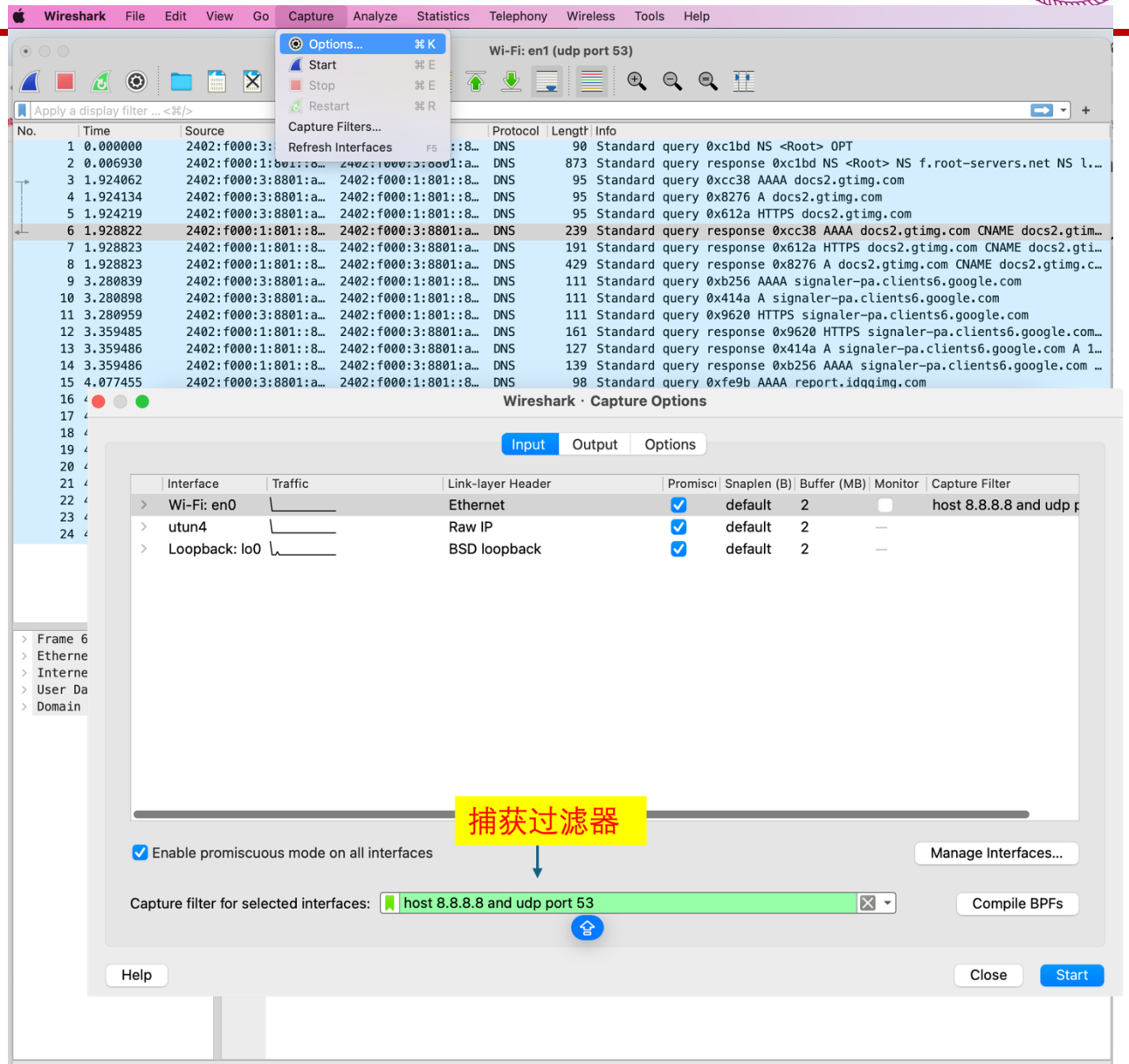
```
.                       104281  IN      NS      i.root-servers.net.
.                       104281  IN      NS      l.root-servers.net.

;; ADDITIONAL SECTION:
a.root-servers.net.     104281  IN      A       198.41.0.4
b.root-servers.net.     104281  IN      A       199.9.14.201
c.root-servers.net.     104281  IN      A       192.33.4.12
d.root-servers.net.     104281  IN      A       199.7.91.13
e.root-servers.net.     104281  IN      A       192.203.230.10
f.root-servers.net.     104281  IN      A       192.5.5.241
g.root-servers.net.     104281  IN      A       192.112.36.4
h.root-servers.net.     104281  IN      A       198.97.190.53
i.root-servers.net.     104281  IN      A       192.36.148.17
j.root-servers.net.     104281  IN      A       192.58.128.30
k.root-servers.net.     104281  IN      A       193.0.14.129
l.root-servers.net.     104281  IN      A       199.7.83.42
m.root-servers.net.     104281  IN      A       202.12.27.33
a.root-servers.net.     104281  IN      AAAA    2001:503:ba3e::2:30
b.root-servers.net.     104281  IN      AAAA    2001:500:200::b
c.root-servers.net.     104281  IN      AAAA    2001:500:2::c
d.root-servers.net.     104281  IN      AAAA    2001:500:2d::d
e.root-servers.net.     104281  IN      AAAA    2001:500:a8::e
f.root-servers.net.     104281  IN      AAAA    2001:500:2f::f
g.root-servers.net.     104281  IN      AAAA    2001:500:12::d0d
h.root-servers.net.     104281  IN      AAAA    2001:500:1::53
i.root-servers.net.     104281  IN      AAAA    2001:7fe::53
j.root-servers.net.     104281  IN      AAAA    2001:503:c27::2:30
k.root-servers.net.     104281  IN      AAAA    2001:7fd::1
l.root-servers.net.     104281  IN      AAAA    2001:500:9f::42
m.root-servers.net.     104281  IN      AAAA    2001:dc3::35

;; Query time: 76 msec
;; SERVER: 192.168.3.1#53(192.168.3.1)
;; WHEN: Mon Jul 10 09:12:01 CST 2023
;; MSG SIZE  rcvd: 823

duanhx@MBP-abai Python % tcpdump
tcpdump: ioctl(SIOCIFCREATE): Operation not permitted
duanhx@MBP-abai Python % pwd
/Users/duanhx/Learn/Python
duanhx@MBP-abai Python % cd /Users/duanhx/Library/Mobile\ Documents/com~apple~CloudDocs/Teach/2023-致理书院/Slides/Data

duanhx@MBP-abai Data % pwd
/Users/duanhx/Library/Mobile Documents/com~apple~CloudDocs/Teach/2023-致理书院/Slides/Data
duanhx@MBP-abai Data % ls
duanhx@MBP-abai Data % sudo -s
Password:
root@MBP-abai Data # curl
curl: try 'curl --help' or 'curl --manual' for more information
root@MBP-abai Data # curl http://www.edu.cn
<a href="https://www.edu.cn/">Moved Permanently</a>.

root@MBP-abai Data # dig www.edu.cn

; <<>> DiG 9.10.6 <<>> www.edu.cn
```

```
root@MBP-abai Data # tcpdump -r edu.cn.pcap -n -v -X
reading from file edu.cn.pcap, link-type RAW (Raw IP)
09:34:06.334196 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 64)
    10.6.6.2.53995 > 202.205.109.212.80: Flags [S], cksum 0xb544 (correct), seq 3287166284, win 65535, options [mss 1260,nop,wscale 6,nop,nop,TS val 2076834197 ecr 0,sackOK,eol], length 0
        0x0000:  4500 0040 0000 4000 4006 f20e 0a06 0602  E..@..@.@.......
        0x0010:  cacd 6dd4 d2eb 0050 c3ee 2d4c 0000 0000  ..m....P..-L....
        0x0020:  b002 ffff b544 0000 0204 04ec 0103 0306  .....D..........
        0x0030:  0101 080a 7bc9 f995 0000 0000 0402 0000  ....{...........
09:34:06.346079 IP (tos 0x0, ttl 57, id 0, offset 0, flags [DF], proto TCP (6), length 52)
    202.205.109.212.80 > 10.6.6.2.53995: Flags [S.], cksum 0x2508 (correct), seq 3171814918, ack 3287166285, win 29200, options [mss 1380,nop,nop,sackOK,nop,wscale 10], length 0
        0x0000:  4500 0034 0000 4000 3906 f91a cacd 6dd4  E..4..@.9.....m.
        0x0010:  0a06 0602 0050 d2eb bd0e 0e06 c3ee 2d4d  .....P........-M
        0x0020:  8012 7210 2508 0000 0204 0564 0101 0402  ..r.%......d....
        0x0030:  0103 030a                                ....
09:34:06.346167 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 40)
    10.6.6.2.53995 > 202.205.109.212.80: Flags [.], cksum 0xc79d (correct), ack 1, win 4096, length 0
        0x0000:  4500 0028 0000 4000 4006 f226 0a06 0602  E..(..@.@..&....
        0x0010:  cacd 6dd4 d2eb 0050 c3ee 2d4d bd0e 0e07  ..m....P..-M....
        0x0020:  5010 1000 c79d 0000                      P.......
09:34:06.346218 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 114)
    10.6.6.2.53995 > 202.205.109.212.80: Flags [P.], cksum 0x849c (correct), seq 1:75, ack 1, win 4096, length 74: HTTP, length: 74
        GET / HTTP/1.1
        Host: www.edu.cn
        User-Agent: curl/7.64.1
        Accept: */*

        0x0000:  4500 0072 0000 4000 4006 f1dc 0a06 0602  E..r..@.@.......
        0x0010:  cacd 6dd4 d2eb 0050 c3ee 2d4d bd0e 0e07  ..m....P..-M....
        0x0020:  5018 1000 849c 0000 4745 5420 2f20 4854  P.......GET./.HT
        0x0030:  5450 2f31 2e31 0d0a 486f 7374 3a20 7777  TP/1.1..Host:.ww
        0x0040:  772e 6564 752e 636e 0d0a 5573 6572 2d41  w.edu.cn..User-A
        0x0050:  6765 6e74 3a20 6375 726c 2f37 2e36 342e  gent:.curl/7.64.
        0x0060:  310d 0a41 6363 6570 743a 202a 2f2a 0d0a  1..Accept:.*/*..
        0x0070:  0d0a                                     ..
09:34:06.352157 IP (tos 0x0, ttl 57, id 11918, offset 0, flags [DF], proto TCP (6), length 40)
    202.205.109.212.80 > 10.6.6.2.53995: Flags [.], cksum 0xd736 (correct), ack 75, win 29, length 0
        0x0000:  4500 0028 2e8e 4000 3906 ca98 cacd 6dd4  E..(..@.9.....m.
        0x0010:  0a06 0602 0050 d2eb bd0e 0e07 c3ee 2d97  .....P........-.
        0x0020:  5010 001d d736 0000                      P....6..
09:34:06.352403 IP (tos 0x0, ttl 57, id 11919, offset 0, flags [DF], proto TCP (6), length 256)
    202.205.109.212.80 > 10.6.6.2.53995: Flags [P.], cksum 0xe504 (correct), seq 1:217, ack 75, win 29, length 216: HTTP, length: 216
        HTTP/1.1 301 Moved Permanently
        Content-Type: text/html; charset=utf-8
        Location: https://www.edu.cn/
        Date: Mon, 10 Jul 2023 01:34:06 GMT
        Content-Length: 54

        <a href="https://www.edu.cn/">Moved Permanently</a>.

        0x0000:  4500 0100 2e8f 4000 3906 c9bf cacd 6dd4  E.....@.9.....m.
        0x0010:  0a06 0602 0050 d2eb bd0e 0e07 c3ee 2d97  .....P........-.
        0x0020:  5018 001d e504 0000 4854 5450 2f31 2e31  P.......HTTP/1.1
        0x0030:  2033 3031 204d 6f76 6564 2050 6572 6d61  .301.Moved.Perma
        0x0040:  6e65 6e74 6c79 0d0a 436f 6e74 656e 742d  nently..Content-
        0x0050:  5479 7065 3a20 7465 7874 2f68 746d 6c3b  Type:.text/html;
```

Wireshark： 流量分析

- 开源、免费
- 源于1998年的Ethereal
- 支持上千种协议
- 支持Plugin 扩展新协议

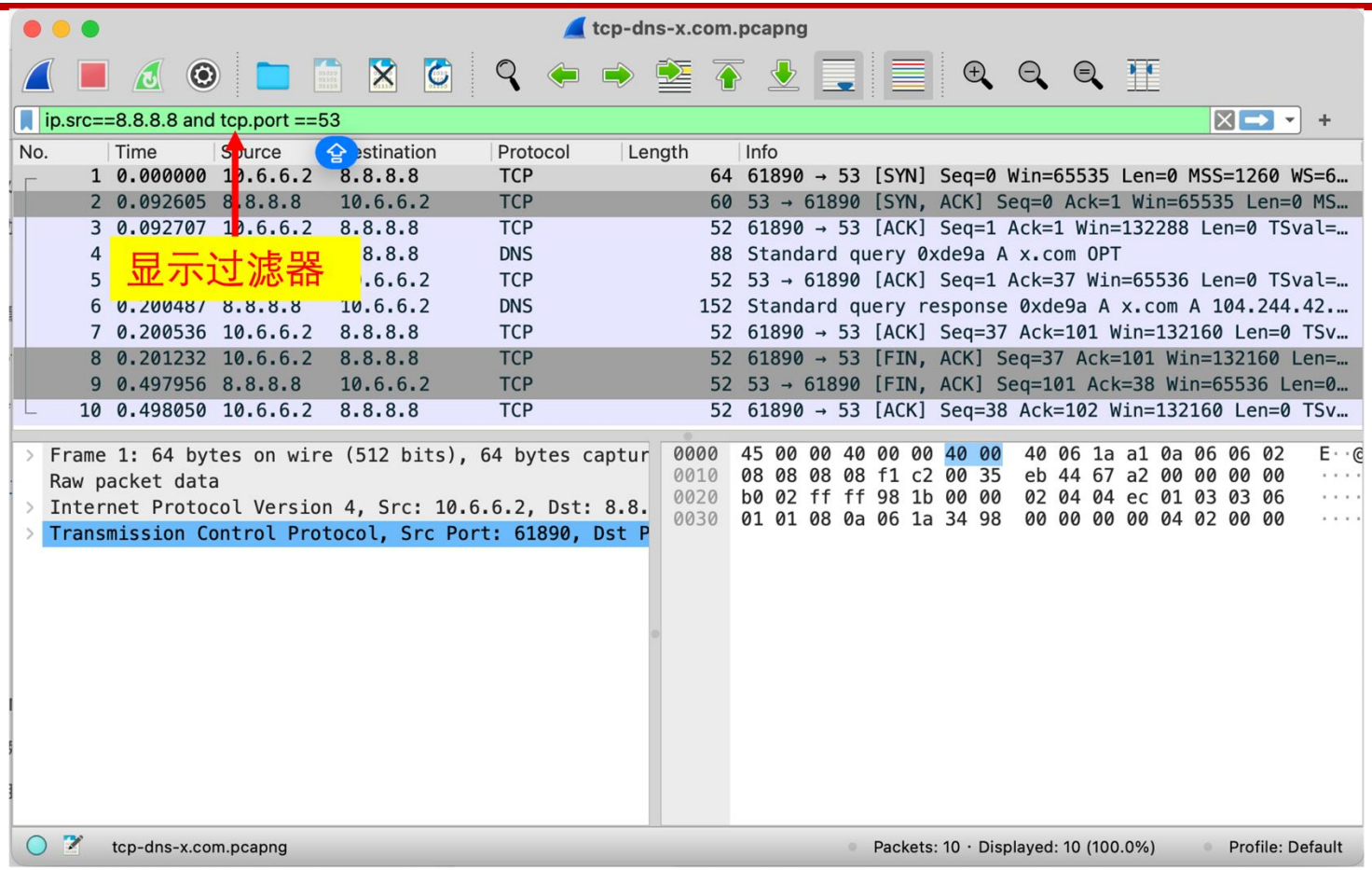
捕获过滤器语法规则 与使用BPF过滤规则，与TCPDUMP相同



Wireshark 使用

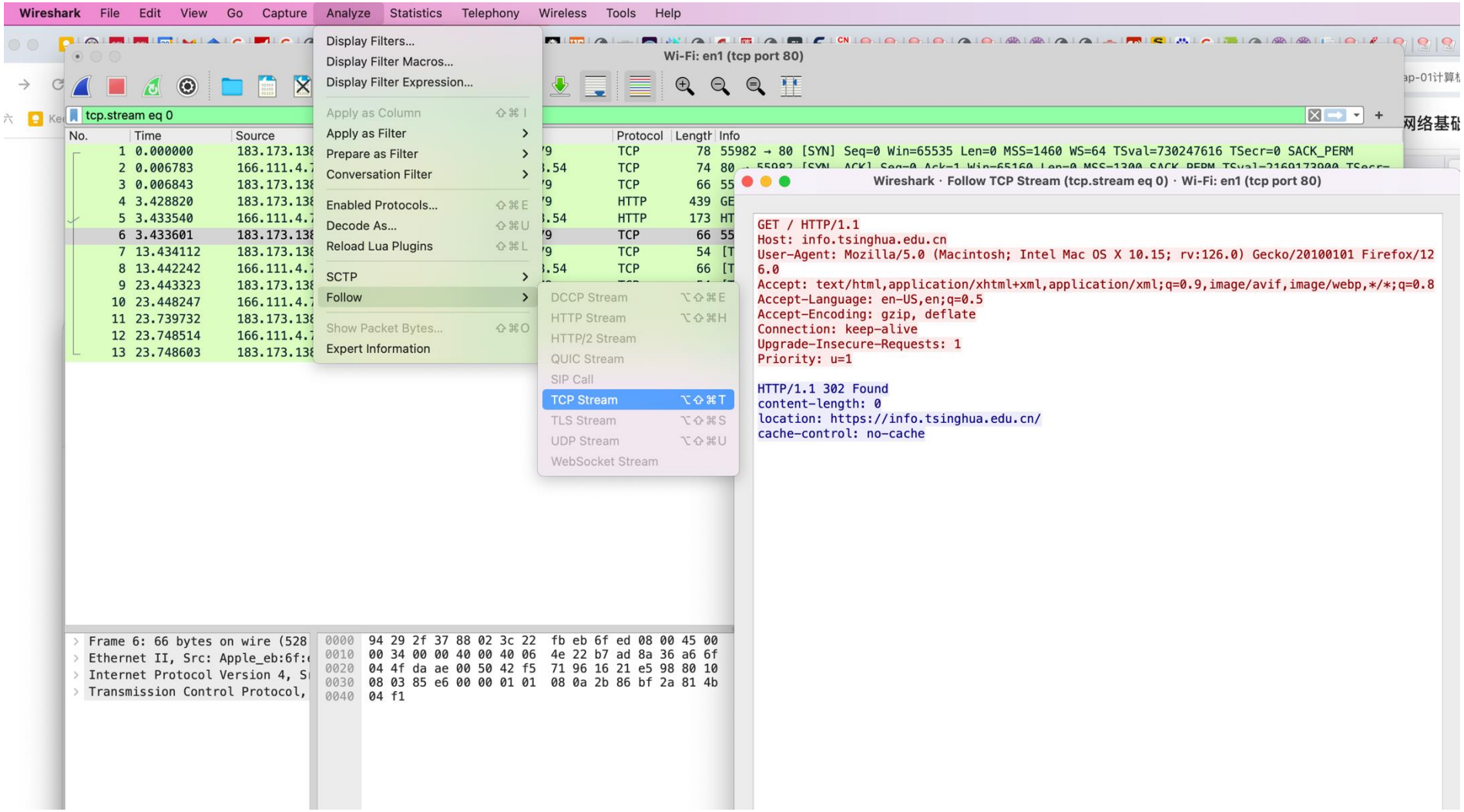
显示过滤器与捕获过滤器使用不同的语法规则。

- ip.src == 192.168.2.10 and ip.dst == 8.8.8.8
- tcp and tcp.scrport != 22
- smtp || pop|| imap



Wireshark 使用

- 分析一个tcp流



用 Wireshark 分析常用的应用层协议

- 超文本传输协议（HTTP）：Web浏览器和Web服务器之间传输超文本文档

- 电子邮件协议（SMTP、POP3、IMAP）：用于发送和接收电子邮件

- 域名系统（DNS）：用于将域名转换为IP地址，实现域名解析功能

网络数据包构造：Scapy

- Scapy 是一个用 Python 语言写的交互式数据包构造和解析程序，可以伪造或解码大量协议的数据包，捕获数据包、匹配请求和响应。

https://scapy.net/?try=1

```
>>> target = "8.8.8.8/30"
>>> packets=IP(dst=target)/ICMP()
>>> [p for p in packets]
[<IP  frag=0 proto=1 dst=8.8.8.8 |<ICMP  |>>, <IP  frag=0 proto=1 dst=8.8.8.9 |<ICMP  |>>, <IP  frag=0
proto=1 dst=8.8.8.10 |<ICMP  |>>, <IP  frag=0 proto=1 dst=8.8.8.11 |<ICMP  |>>]
>>> packets[0]
<IP  frag=0 proto=1 dst=Net("8.8.8.8/30") |<ICMP  |>>
>>> packets[0].show()
WARNING: No route found (no default route?)
WARNING: No route found (no default route?)
WARNING: more No route found (no default route?)
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = 1
  chksum    = None
  src       = 0.0.0.0
```

发送数据包（send和sendp）

```
>>> send (IP(dst="8.8.8.8/30")/ICMP())
....
Sent 4 packets.
```

把四个ICMP数据包发到网络上，scapy使用操作系统的路由选择合适的网卡，并且填充数据链路层的信息。

如果要自己构造数据链路层的信息，比如设定数据帧的MAC地址，可以使用sendp() 函数，例如，下面的命令可以指定发送数据帧的目标地址为广播地址，并指定从网卡eth1发出去：

```
>>>sendp(Ether(dst="FF:FF:FF:FF:FF:FF")/IP(dst="1.2.3.4",ttl
=(1,4)), iface="eth1") )
```

发送并接受数据包（sr）

```
>>> p=sr1(IP(dst="www.slashdot.org")/ICMP()/"XXXXXXXXXXX")
Begin emission:
...Finished to send 1 packets.
.*
Received 5 packets, got 1 answers, remaining 0 packets
>>> p
<IP version=4L ihl=5L tos=0x0 len=39 id=15489 flags= frag=0L ttl=42 proto=ICMP
 chksum=0x51dd src=66.35.250.151 dst=192.168.5.21 options='' |<ICMP type=echo-reply
 code=0 chksum=0xee45 id=0x0 seq=0x0 |<Raw load='XXXXXXXXXXX'
 |<Padding load='\x00\x00\x00\x00' |>>>>
>>> p.show()
---[ IP ]---
version   = 4L
ihl       = 5L
tos       = 0x0
len       = 39
id        = 15489
flags     =
frag      = 0L
ttl       = 42
proto     = ICMP
chksum    = 0x51dd
src       = 66.35.250.151
dst       = 192.168.5.21
options   = ''
---[ ICMP ]---
   type      = echo-reply
   code      = 0
```

sr() 函数用于发送数据包的同时，接收相应的应答数据包。该函数返回若干应答数据和未应答的数据包。

函数 sr1() 是sr()的变体，它只返回一个应答了已发送数据包（或数据包集）的数据包。与send（）相同，数据包必须是网络层数据包（如IP、ARP 等）。

如果在数据链路层发送数据同时接受响应，可以使用函数 srp()函数。

执行网络嗅探（sniffing)

```
>>> sniff(filter="icmp and host 8.8.8.8", count=2, iface="eth0")
<Sniffed: TCP:0 UDP:0 ICMP:2 Other:0>
>>> pkts=_    # 下划线_表示上面sniff()函数的返回值
>>> pkts.summary()
Ether / IP / ICMP 8.8.8.8 > 192.168.0.103 echo-reply 0 / Raw
Ether / IP / ICMP 192.168.0.103 > 8.8.8.8 echo-request 0 / Raw
>>> pkts[1]
<Ether  dst=d0:76:e7:d8:35:43 src=3c:22:fb:52:f9:8e type=IPv4 |<IP  version=4 ihl=5 tos=0x0
len=84 id=43085 flags= frag=0 ttl=64 proto=icmp chksum=0x13d src=192.168.0.103 dst=8.8.8.8
|<ICMP  type=echo-request code=0 chksum=0xa60e id=0x9281 seq=0xa unused='' |<Raw
load='e\\xe3\x0fG\x00\r_+\x08\t\n\x0b\x0c\r\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17
\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567' |>>>>
```

用Scapy可以完成网络嗅探或监听的功能，类似tcpdump 。由于同样基于libpcap实现，Scapy的过滤表达式语法和tcpdump相同。可以指定所监听网卡的名称，如果没有指定，则使用配置变量conf.iface的值。以下脚本监听源地址或目标地址为8.8.8.8的ICMP数据包。

在自己的Python脚本中使用Scapy

```
#! /usr/bin/env python3
import sys
from scapy.all import DNS, DNSQR, IP, sr1, UDP
dns_req = IP(dst='http://8.8.8.8') / UDP(dport=53) /
    DNS(rd=1, qd=DNSQR(qname=sys.arg[1]))
answer = sr1(dns_req, verbose=0)
print(answer[DNS].summary())
```

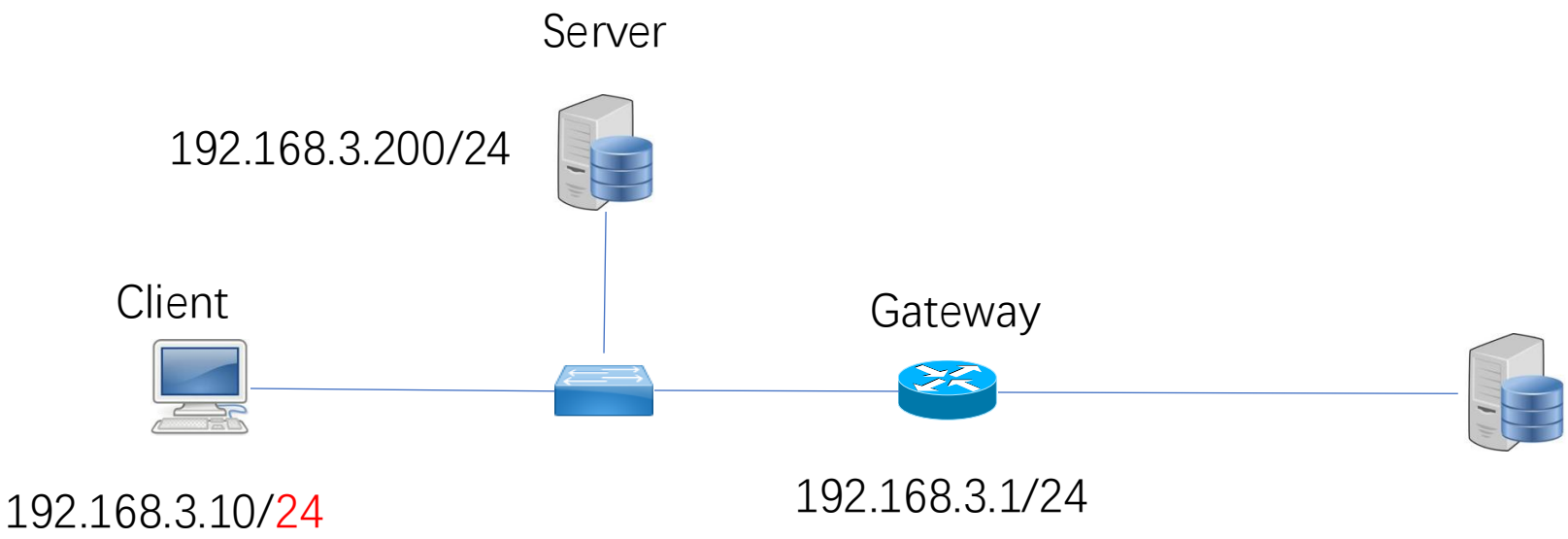
实验1：使用tcpdump分析ICMP流量

1.通过实验操作和网络流量分析，深入理解交换机和路由器的工作原理。

2.掌握子网划分和子网掩码的概念和应用，理解其在网络通信中的重要性。

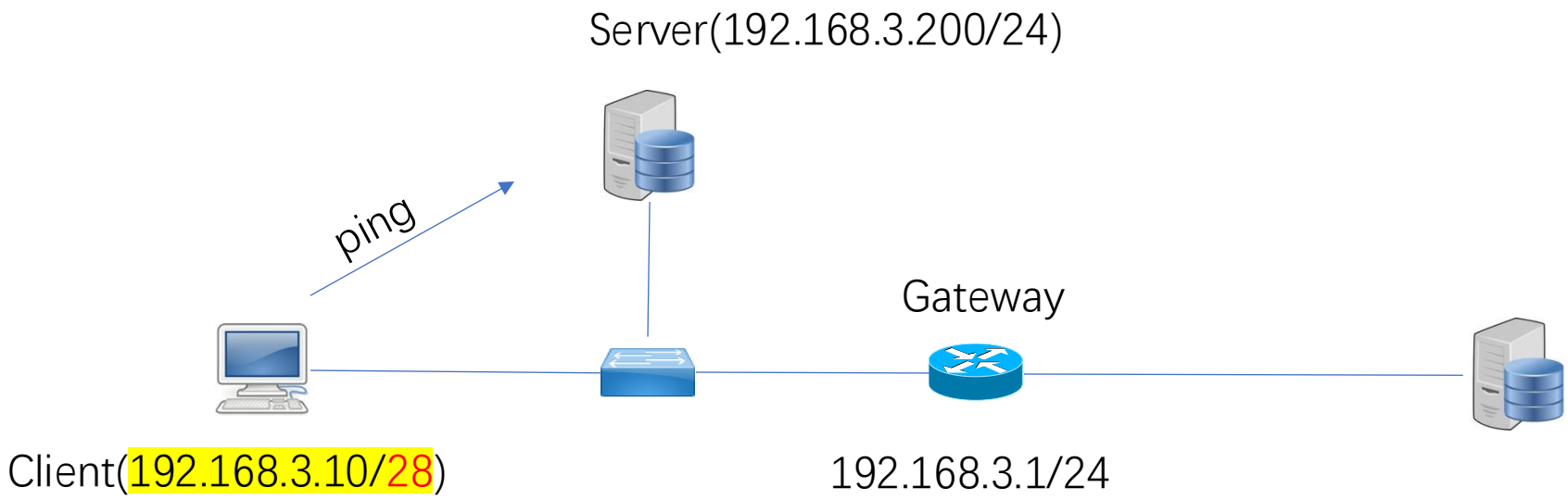
3.熟练使用tcpdump工具，掌握获取和分析网络流量的技能。

4.为后续章节的学习和深入理解网络原理奠定基础，提升解决实际网络问题的能力



子网掩码配置错了会发生什么

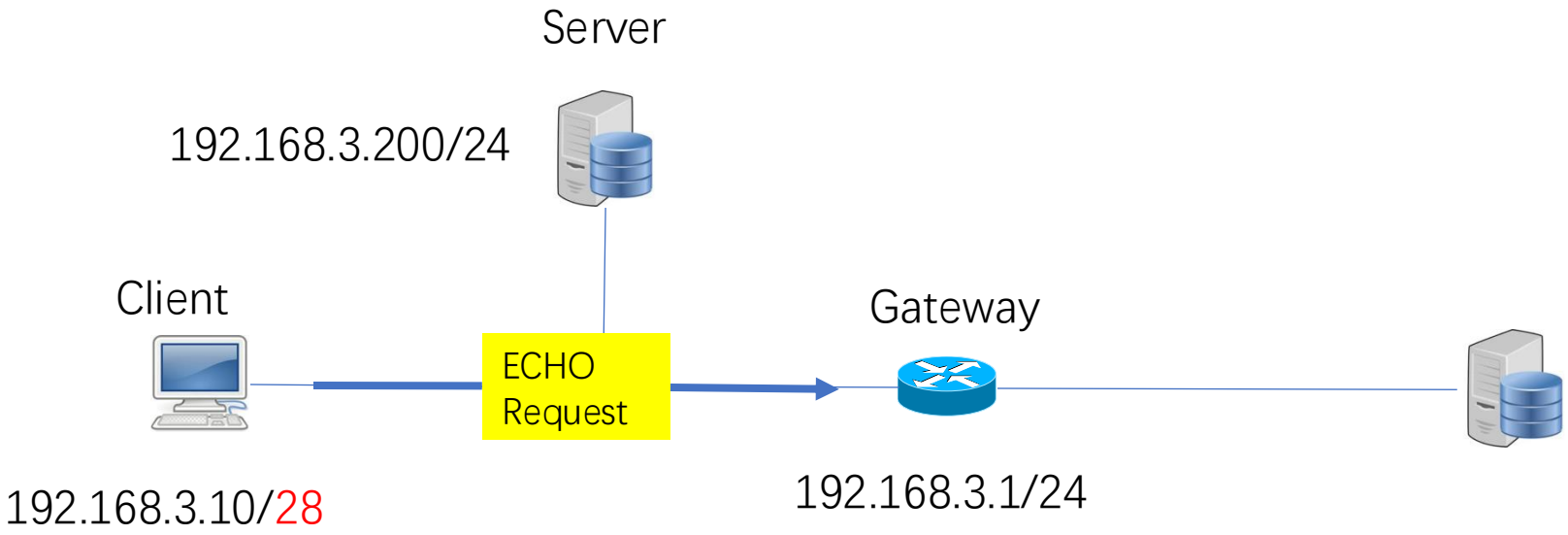
- 在一个C类子网（192.168.3.0/24）中，Client的IP地址配错了，本来应该是 192.168.3.10/24，但写成了 192.168.3.10/28
- 在Client 上执行 ping 192.168.3.200，能收到响应吗?
- 与正确配置的环境下，网络上的流量有什么不同?



子网掩码配置错了会发生什么

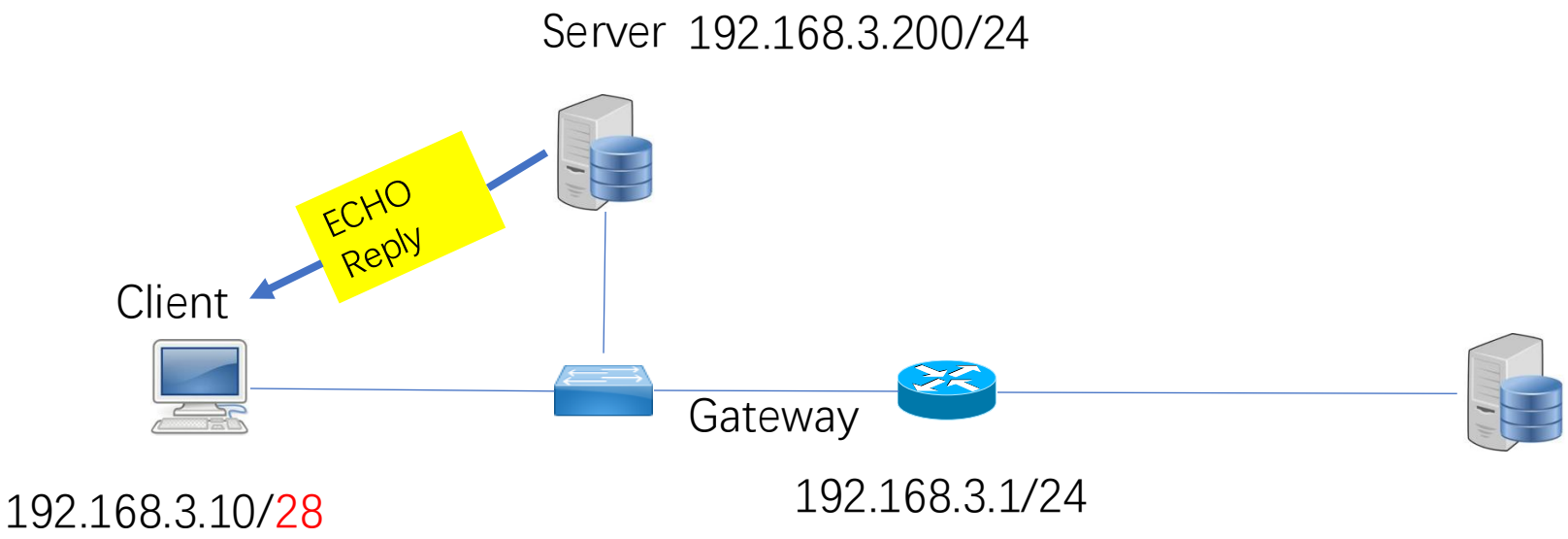
- Ping(ICMP Echo Request)
- Client> ping 192.168.3.200

message	srcMAC	dstMAC	srcIP	dstIP
ECHO Request				



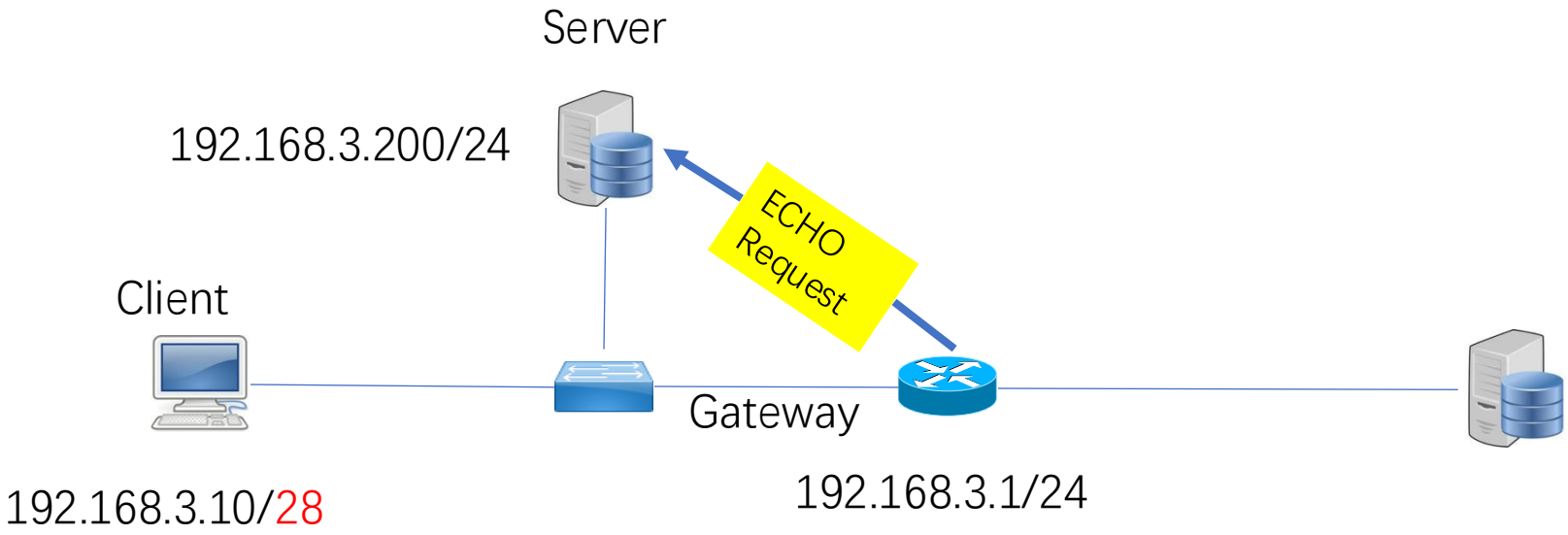
子网掩码配置错了会发生什么

- 在一个C类子网（192.168.3.0/24）中，Client的IP地址配错了，本来应该是 192.168.3.10/24，但写成了 192.168.3.10/28
- 在Client 上执行 ping 192.168.3.200，能收到响应吗?
- 与正确配置的环境下，网络上的流量有什么不同?



子网掩码配置错了会发生什么

- 在一个C类子网（192.168.3.0/24）中，Client的IP地址配错了，本来应该是 192.168.3.10/24，但写成了 192.168.3.10/28
- 在Client 上执行 ping 192.168.3.200，能收到响应吗?
- 与正确配置的环境下，网络上的流量有什么不同?



子网掩码配错时的三角转发

Client Gateway Server

- 第一个ICMP (Echo Request)， 源MAC 是 Client(IP 192.168.3.10）， 目标 MAC 是网关（IP:192.168.3.1)
- 第二个ICMP (Echo Request)， 源MAC 是 网关（192.168.3.1)， 目标MAC 是 server (192.168.3.200)。也就是说，这是网关转发的 ICMP包；
- 第三个ICMP (Echo Reply)， 源MAC 是 Server， 目标MAC 是 Client

也就是说，这个ping 经历了一个三角： Client -> Gateway -> Server ->Client

```
reading from file test.pcap, link-type EN10MB (Ethernet)
tcpdump: expression rejects all packets
root@ubuntu-20-desktop:/home/imool# tcpdump -n -r test.pcap -e icmp or ar
reading from file test.pcap, link-type EN10MB (Ethernet)
19:21:56.172605 02:0a:5a:5e:88:e9 > 02:99:c2:70:22:6c, ethertype ARP (0x0
19:21:56.172616 02:99:c2:70:22:6c > 02:0a:5a:5e:88:e9, ethertype ARP (0x0
19:22:06.154205 02:99:c2:70:22:6c > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 42: Request who-has 192.168.3.1 tell 192.168.3.10, length 28
19:22:06.159102 02:0a:5a:5e:88:e9 > 02:99:c2:70:22:6c, ethertype ARP (0x0806), length 60: Reply 192.168.3.1 is-at 02:0a:5a:5e:88:e9, length 46
19:22:12.945393 02:99:c2:70:22:6c > 02:0a:5a:5e:88:e9, ethertype IPv4 (0x0800), length 98: 192.168.3.10 > 192.168.3.200: ICMP echo request, id 6, seq 1, length 64
19:22:12.948786 02:0a:5a:5e:88:e9 > 02:82:00:d8:48:8e, ethertype IPv4 (0x0800), length 98: 192.168.3.10 > 192.168.3.200: ICMP echo request, id 6, seq 1, length 64
19:22:12.951073 02:82:00:d8:48:8e > 02:99:c2:70:22:6c, ethertype IPv4 (0x0800), length 98: 192.168.3.200 > 192.168.3.10: ICMP echo reply, id 6, seq 1, length 64
19:22:13.946624 02:99:c2:70:22:6c > 02:0a:5a:5e:88:e9, ethertype IPv4 (0x0800), length 98: 192.168.3.10 > 192.168.3.200: ICMP echo request, id 6, seq 2, length 64
19:22:13.948582 02:0a:5a:5e:88:e9 > 02:99:c2:70:22:6c, ethertype IPv4 (0x0800), length 126: 192.168.3.1 > 192.168.3.10: ICMP redirect 192.168.3.200 to host 192.168.3.20
0, length 92
19:22:13.948613 02:99:c2:70:22:6c > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 42: Request who-has 192.168.3.200 tell 192.168.3.10, length 28
19:22:13.952192 02:82:00:d8:48:8e > 02:99:c2:70:22:6c, ethertype IPv4 (0x0800), length 98: 192.168.3.200 > 192.168.3.10: ICMP echo reply, id 6, seq 2, length 64
19:22:13.952524 02:82:00:d8:48:8e > 02:99:c2:70:22:6c, ethertype ARP (0x0806), length 60: Reply 192.168.3.200 is-at 02:82:00:d8:48:8e, length 46
19:22:14.948469 02:99:c2:70:22:6c > 02:0a:5a:5e:88:e9, ethertype IPv4 (0x0800), length 98: 192.168.3.10 > 192.168.3.200: ICMP echo request, id 6, seq 3, length 64
19:22:14.950471 02:0a:5a:5e:88:e9 > 02:99:c2:70:22:6c, ethertype IPv4 (0x0800), length 126: 192.168.3.1 > 192.168.3.10: ICMP redirect 192.168.3.200 to host 192.168.3.20
0, length 92
19:22:14.951719 02:82:00:d8:48:8e > 02:99:c2:70:22:6c, ethertype IPv4 (0x0800), length 98: 192.168.3.200 > 192.168.3.10: ICMP echo reply, id 6, seq 3, length 64
19:22:15.949872 02:99:c2:70:22:6c > 02:82:00:d8:48:8e, ethertype IPv4 (0x0800), length 98: 192.168.3.10 > 192.168.3.200: ICMP echo request, id 6, seq 4, length 64
19:22:15.952435 02:82:00:d8:48:8e > 02:99:c2:70:22:6c, ethertype IPv4 (0x0800), length 98: 192.168.3.200 > 192.168.3.10: ICMP echo reply, id 6, seq 4, length 64
19:22:16.951606 02:99:c2:70:22:6c > 02:82:00:d8:48:8e, ethertype IPv4 (0x0800), length 98: 192.168.3.10 > 192.168.3.200: ICMP echo request, id 6, seq 5, length 64
19:22:16.953486 02:82:00:d8:48:8e > 02:99:c2:70:22:6c, ethertype IPv4 (0x0800), length 98: 192.168.3.200 > 192.168.3.10: ICMP echo reply, id 6, seq 5, length 64
```

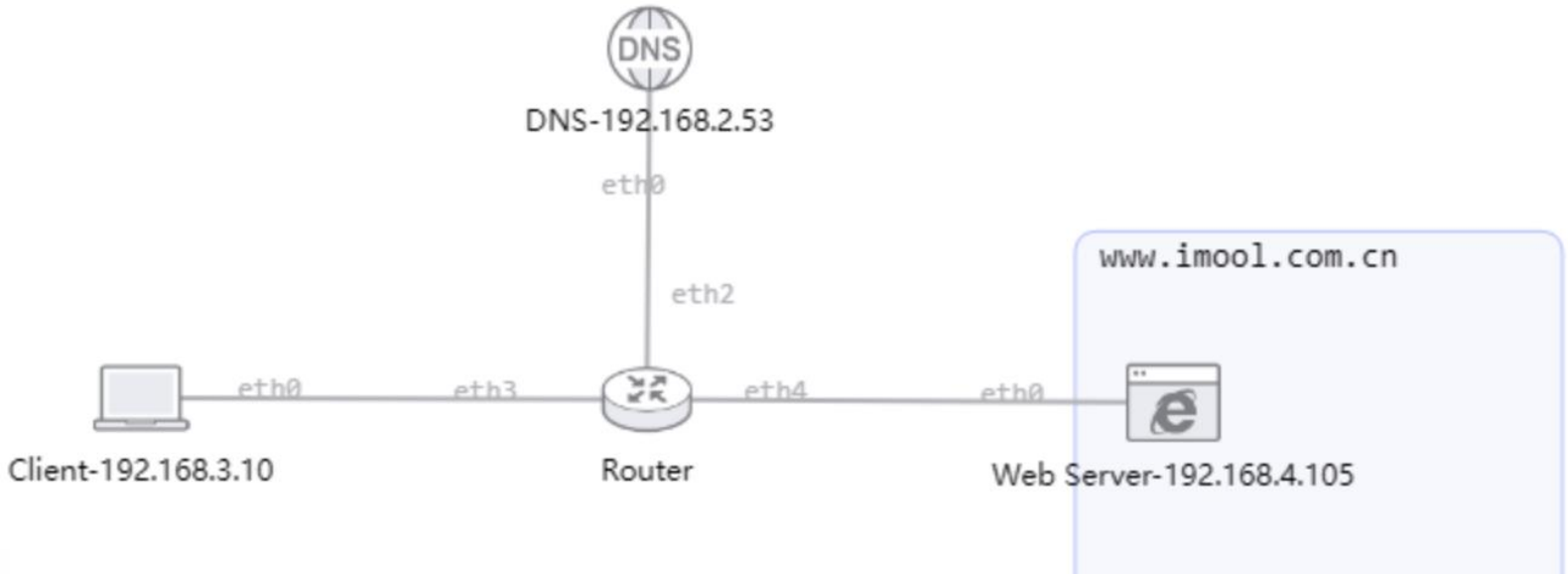
网关告诉Client，重定向到本地

Client ARP解析Server的MAC

以后的Echo Request都用 Server 的MAC地址封装

实验2:使用wireshark分析Web访问过程流量

1. 学习并掌握Wireshark在网络流量分析中的作用，以及如何通过它来捕获和分析网络数据包;
2. 通过实际操作，观察并分析域名解析过程中涉及的IP包，理解DNS请求和响应的格式和内容,以及它是如何将域名转换为IP地址的;
3. 观察并解释Client与www.imool.com.cn之间TCP连接的建立和拆除过程，包括三次握手和四次挥手;
4. 使用Wireshark的协议流追踪功能，提取并分析Web访问过程中产生的Cookie信息。

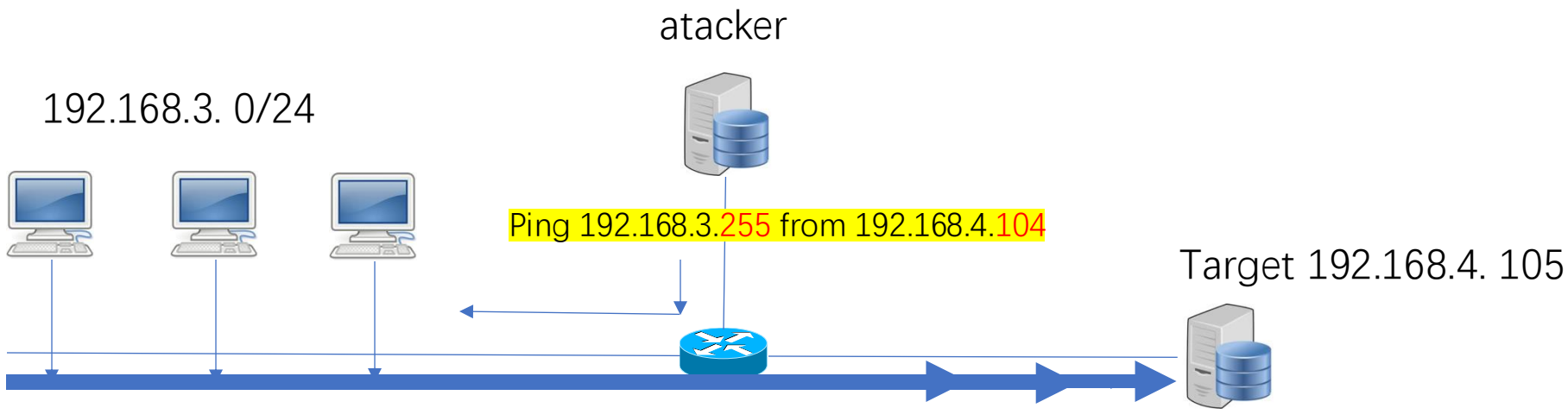


实验4: 用Scapy构造ICMP 数据包

1.学习并掌握Scapy工具的基本使用方法;

2.网络数据包的伪造与发送，发送伪造的ICMP数据包。

3.实现一个流量反射放大攻击 Smurf



提交内容

- 实验报告书
  - 每个实验步骤完成后的验证截图
  - 实验步骤中对应的配置、命令（截图或代码）
  - 流量抓取的简单分析：对于报文中不同网络协议层的头部以及应用层数据的简单分析

清華大學
Tsinghua University

谢谢