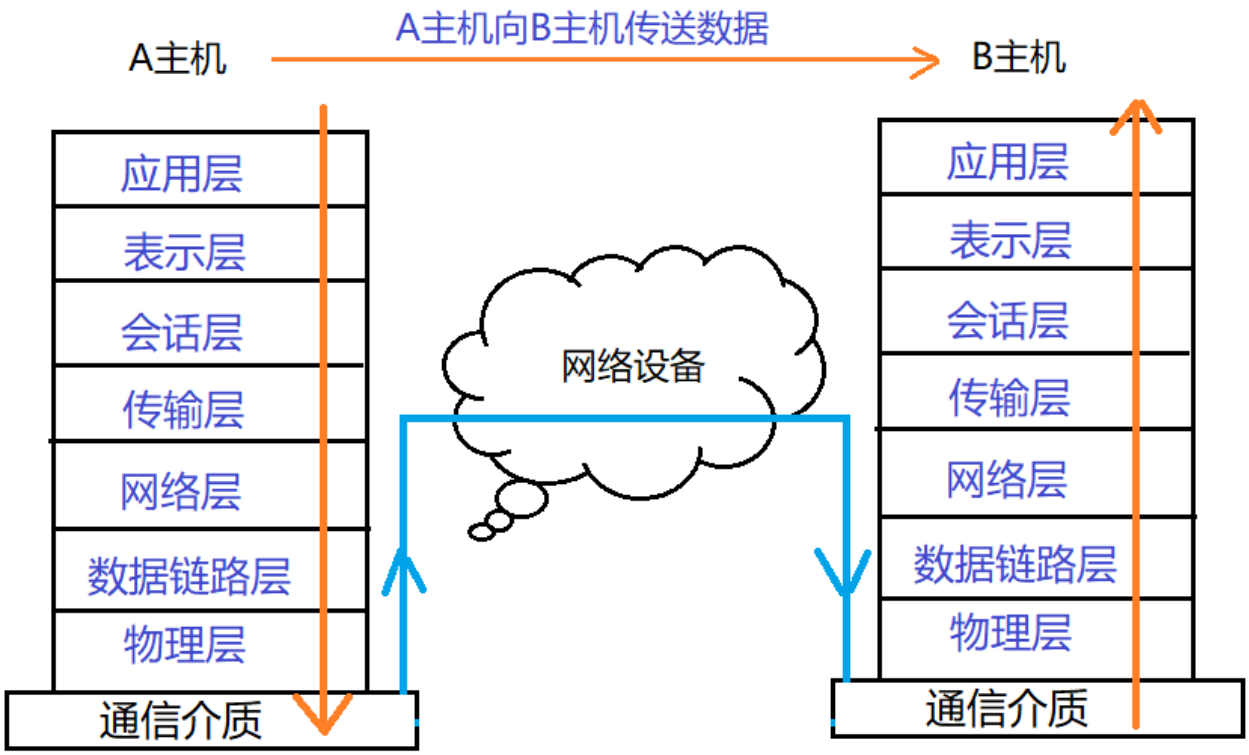
课程目标

- 了解OSI七层模型分层结构
- **了解TCP/IP协议簇四层模型分层结构**
- **能够说出TCP/IP协议簇中运输层、网络层和数据链路层常见的相关协议**
- **能够说出TCP/IP的三次握手四次断开过程**
- 了解Vmware的三种网络模式
- 能够使用客户端工具连接虚拟机

思考：

数据是如何在两台主机之间传输的？

数据传输过程：



一、OSI七层模型

1. 什么是OSI模型

OSI:

- 开放系统互连参考模型,是国际标准化组织(ISO)和国际电报电话咨询委员会(CCITT)联合制定的开放系统互连参考模型。
- **目的**:为开放式互连信息系统提供了一种功能结构的框架和参考。
- 这里所说的开放系统，实质上指的是遵循OSI参考模型和相关协议能够实现互连的具有各种应用目的的计算机系统。
- OSI采用了分层的结构化技术，共分七层：

物理层、数据链路层、网络层、传输层、会话层、表示层、应用层

2. OSI的七层介绍

2.1 应用层

- 应用层是计算机用户，以及各种应用程序和网络之间的接口，其功能是直接向用户提供服务，完成用户希望在网络上完成的各种工作。
- 应用层为用户提供的服务和协议：文件传输服务（FTP）、远程登录服务（ssh）、网络管理服等。
- 上述的各种网络服务由该层的不同应用协议和程序完成。
- 应用层的主要功能如下：
  - 用户接口：应用层是用户与网络，以及应用程序与网络间的直接接口，使得用户能够与网络进行交互式联系。
  - 实现各种服务：该层具有的各种应用程序可以完成和实现用户请求的各种服务。

2.2 表示层

- 表示层是**对来自应用层的命令和数据进行解释，对各种语法赋予相应的含义，并按照一定的格式传送给会话层。**
- 其主要功能是**处理用户信息的表示问题**，如编码、数据格式转换和加密解密等。
- 表示层的具体功能如下：
  - 数据格式处理：协商和建立数据交换的格式，解决各应用程序之间在数据格式表示上的差异。
  - 数据的编码：处理字符集和数字的转换。例如由于用户程序中的数据类型（整型或非整型、有符号或无符号等）、用户标识等都可以有不同的表示方式，因此，在设备之间需要具有在不同字符集或格式之间转换的功能。
  - 压缩和解压缩：为了减少数据的传输量，这一层还负责数据的压缩与解压缩。
  - 数据的加密和解密：可以提高网络的安全性。

2.3 会话层

- 会话层是用户应用程序和网络之间的接口，主要任务是：组织和协调两个会话进程之间的通信，并对数据交换进行管理。
- 当建立会话时，用户必须提供他们想要连接的远程地址。
- 会话层的具体功能如下：
  - 会话管理：

  允许用户在两个实体设备之间建立、维持和终止会话，并支持他们之间的数据交换.

  比如:提供单方向会话或者双方向会话时，管理会话中的发送顺序，以及会话所占用时间的长短。

  - 会话流量控制：提供会话流量控制和交叉会话功能。
  - 寻址：使用远程地址建立会话连接。
  - 差错控制：

  从逻辑上讲会话层主要负责**数据交换的建立、保持和终止**，但**实际的工作是接收来自传输层的数据**，并负责纠正错误。但需要注意，该层检查的错误不是通讯介质的错误，而是磁盘空间、打印机缺纸之类的高级错误。

2.4 传输层

- OSI上3层：应用层、表示层、会话层的主要任务是数据处理——**资源子网**

- OSI下3层：网络层、数据链路层、物理层的主要任务是数据通讯——**通讯子网**
- 传输层是OSI模型的第4层，它是通信子网和资源子网的接口和桥梁，起到承上启下的作用
- 传输层的主要任务是：**向用户提供可靠的端到端的差错和流量控制，保证报文的正确传输**

报文：报文(**message**)是网络中交换与传输的[1]

报文段:组成报文的每个分组。我们将运输层分组称为报文段(**segment**)

2.5 网络层

- 主要任务是：**数据链路层的数据在这一层被转换为[2]，然后通过路径选择、分段组合、顺序、进/出路由等控制，将信息从一个网络设备传送到另一个网络设备。**
- 一般情况下，数据链路层是解决**同一网络**(局域网)内节点之间的通信，而网络层主要解决**不同子网**间的通信。

2.6 数据链路层

在计算机网络中由于各种干扰的存在，物理链路是不可靠的。因此，这一层的主要功能是:

- 在物理层提供的**比特流**的基础上，通过差错控制、流量控制方法，使有差错的物理线路变为无差错的数据链路，即提供可靠的通过物理介质传输数据的方法。包含LLC子层和MAC子层。
    - LLC子层的主要功能包括：
        - 传输可靠性保障和控制
        - 数据包的分段与重组
        - 数据包的顺序传输。
    - MAC子层的主要功能包括：
        - 数据帧的封装与解封装
        - 帧的寻址和识别，帧的接收与发送，链路的管理，帧的差错控制等
        - MAC子层的存在屏蔽了不同物理链路种类的差异性。
- 具体工作是：接收来自物理层的位流（比特流）形式的数据，并封装成[3]，传送到上一层；同样，也将来自上层的数据帧，拆装为位流形式的数据转发到物理层；并且，还负责处理接收端发回的确认帧的信息，以便提供可靠的数据传输。

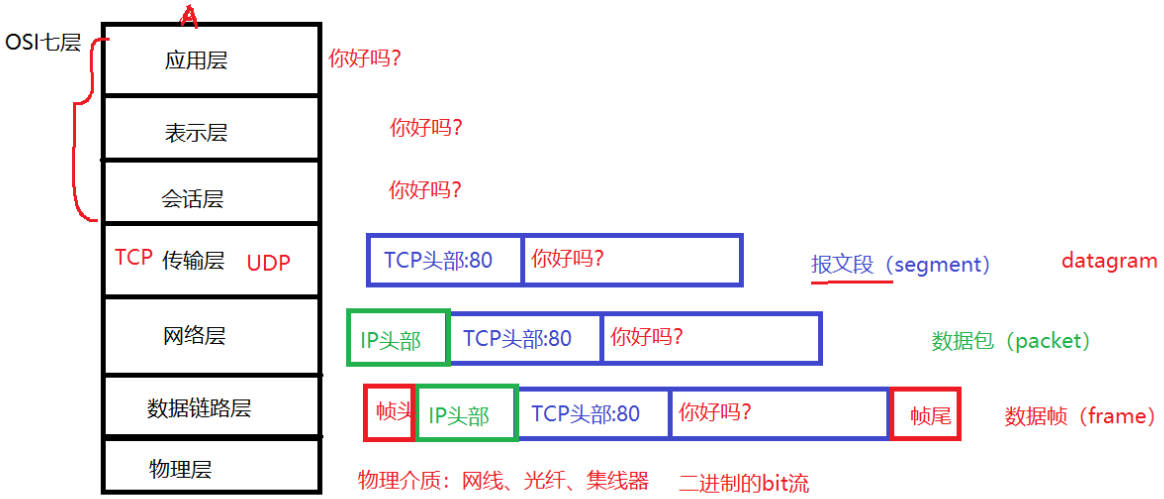
帧：帧(**frame**)是数据链路层的传输单元。将上层传入的数据添加一个头部和尾部，组成了帧.

2.7 物理层

- 主要功能是：利用传输介质为数据链路层提供物理连接，实现比特流的透明传输。尽可能屏蔽掉具体传输介质和物理设备的差异。

3. 总结

- 由于OSI是一个理想的模型，因此一般网络系统只涉及其中的几层，很少有系统能够具有所有的7层，并完全遵循它的规定。
- 在7层模型中，每一层都提供一个特殊的网络功能。
- 从网络功能的角度观察：
    - 物理层、数据链路层、网络层：主要提供**数据传输和交换功能**，即以节点到节点之间的通信为主；
    - 传输层（第4层）：作为上下两部分的桥梁，是整个网络体系结构中最关键的部分；
    - 会话层、表示层和应用层：以提供**用户与应用程序之间的信息和数据处理**功能为主；

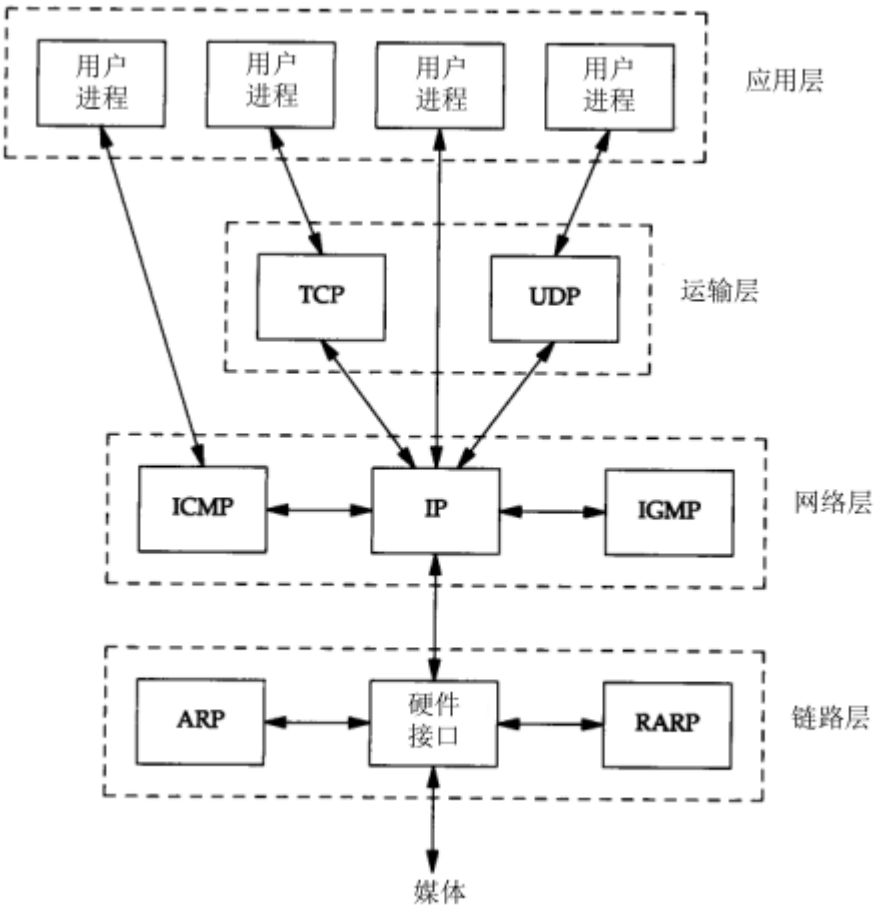


二、TCP/IP协议模型

1. 什么是TCP/IP模型

- **TCP/IP协议模型**（Transmission Control Protocol/Internet Protocol），包含了一系列构成互联网基础的网络协议，是Internet的核心协议，通过20多年的发展已日渐成熟，并被广泛应用于**局域网和广域网**中，目前已成为一种**国际标准**。
- TCP/IP协议簇是一组不同层次上的**多个协议**的组合，该协议采用了4层的层级结构，每一层都呼叫它的下一层所提供的协议来完成自己的需求，与OSI的七层模型相对应。
- 尽管通常称该协议族为TCP/IP，但TCP和IP只是其中的两种协议而已（该协议族的另一个名字是Internet协议族(Internet Protocol Suite)）

2. TCP/IP的分层结构



2.1 链路层

OSI的物理层和数据链路层

- **ARP**（地址解析协议**IP-MAC**）和**RARP**（逆地址解析协议**MAC-IP**）是某些网络接口（如以太网）使用的特殊协议，用来转换IP层和网络接口层使用的地址。

2.2 网络层

OSI网络层

- 也称作互联网层或网际层，处理分组在网络中的活动，例如分组的选路。
- 在TCP/IP协议族中，网络层协议包括IP协议（网际协议），ICMP协议（Internet互联网控制报文协议），以及IGMP协议（Internet组管理协议）。
    - **IP**是一种网络层协议，提供的是一种不可靠的服务，它只是尽可能快地把分组从源结点送到目的结点，但是并不提供任何可靠性保证。同时被TCP和UDP使用。
    - TCP和UDP的每组数据都通过端系统和每个中间路由器中的IP层在互联网中进行传输。
    - **ICMP**是IP协议的附属协议。IP层用它来与其他主机或路由器**交换错误报文和其他重要信息。**它主要是用来提供有关通向目的地址的路径信息。Ping和Traceroute工具，它们都使用了ICMP协议。
    - **IGMP**是Internet组管理协议。它用来把一个UDP数据报多播到多个主机。该协议运行在主机和组播路由器之间。

2.3 运输层

OSI传输层

主要为两台主机上的应用程序提供端到端的通信。在TCP/IP协议族中，有两个互不相同的传输协议：

TCP（传输控制协议）和UDP（用户数据报协议） **TCP协议：**为两台主机提供高可靠性的数据通信。TCP是面向连接的通信协议，通过**三次握手**建立连接，通讯完成时要断开连接，由于TCP是面向连接的所以只能用于端到端的通讯。TCP提供的是一种可靠的数据流服务，采用**"带重传的肯定确认"**技术来实现传输的可靠性。也就是TCP数据包中包括序号（seq）和确认（ack），所以未按照顺序收到的包可以被排序，而损坏的包可以被重传。 **UDP协议：**则为应用层提供一种非常简单的服务。它是面向无连接的通讯协议，UDP数据包括目的端口号和源端口号信息，由于通讯不需要连接，所以可以实现广播发送。 UDP通讯时不需要接收方确认，不保证该数据报能到达另一端，属于不可靠的传输，可能会出现丢包现象。UDP与TCP位于同一层，但它不管数据包的顺序、错误或重发。

2.4 应用层

OSI会话层、表示层、应用层

应用层负责处理特定的应用程序细节。

3. 数据封装过程

- **数据格式**

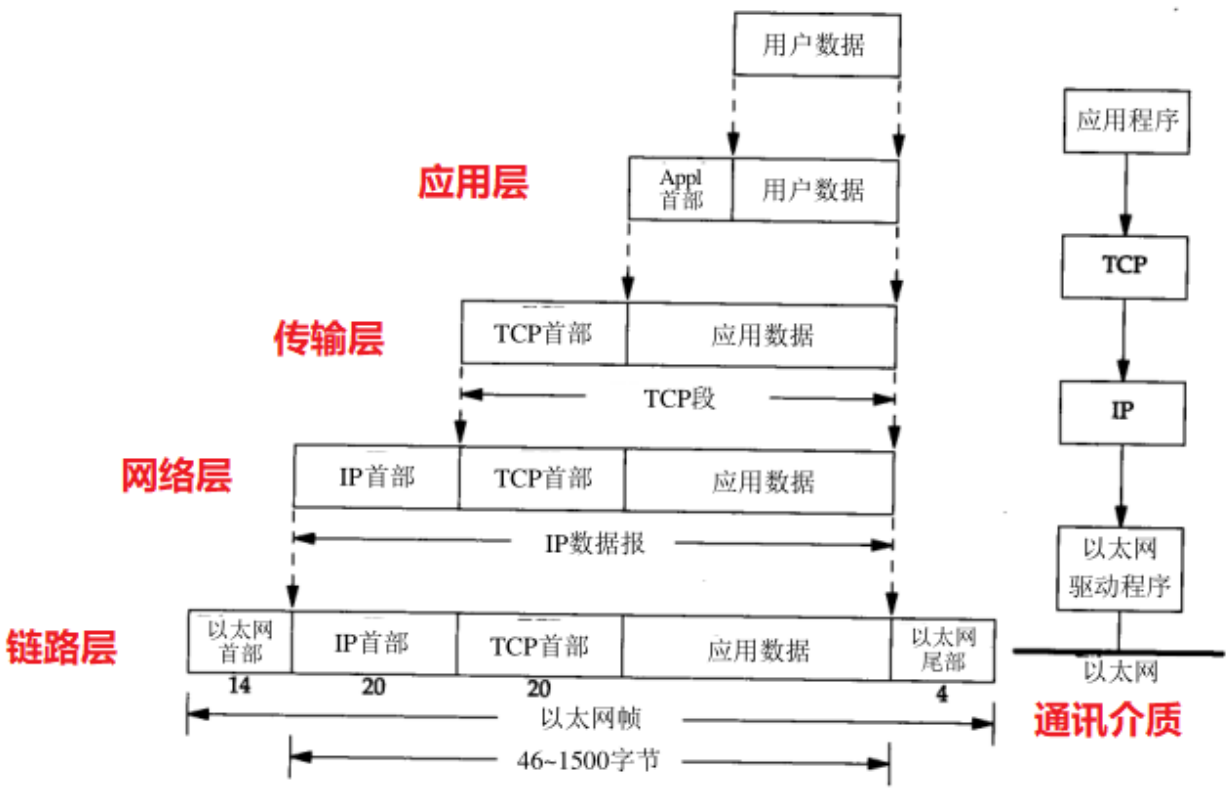
  TCP数据信息：TCP头部+实际数据 (TCP头包括源和目标主机**端口号**、顺序号、确认号、校验字等)

  IP数据包：IP头部+TCP数据信息（IP头包括源和目标主机**IP地址**、类型、生存期等）

  数据帧：帧头+IP数据包+帧尾 （帧头包括源和目标主机**MAC初步地址**及类型，帧尾是校验字）

- **数据的封装与解封装：** 封装：数据要通过网络进行传输，要从高层一层一层的向下传送，如果一个主机要传送数据到别的主机，先把数据装到一个特殊协议报头中，这个过程叫-----封装。 解封装：上述的逆向过程

  当数据以TCP/IP协议传输时的封装与街封装过程如下图：



数据进入协议栈时的封装过程

三、TCP/IP三次握手四次断开

1. 了解相关名词

序列号：Seq序号，占32位，用来标识从TCP源端向目的端发送的字节流，发起方发送数据时对此进行标记。
确认序号：Ack序号，占32位，只有ACK标志位为1时，确认序号字段才有效，Ack=Seq+1。

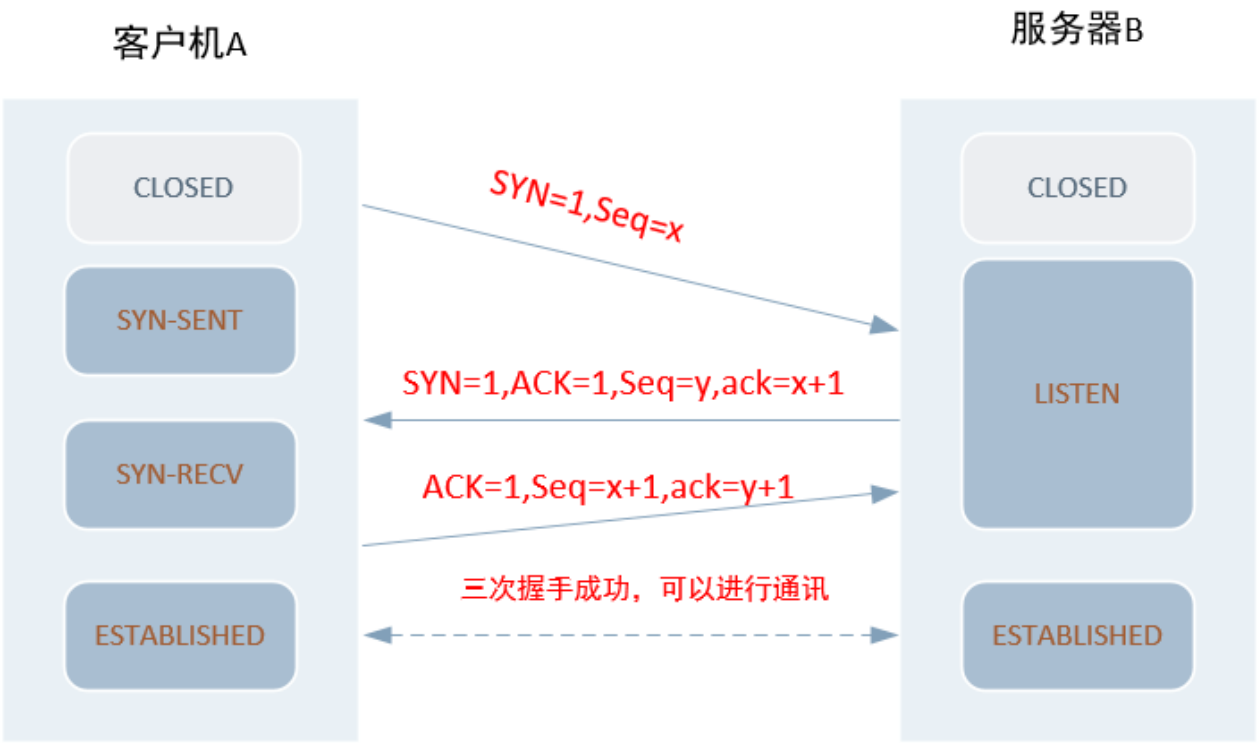
常见的标志位：
ACK：确认序号有效。
SYN：发起一个新连接。
FIN：释放一个连接。

2. 了解netstat中的网络状态

```
CLOSED          初始（无连接）状态。
LISTEN          侦听状态，等待远程机器的连接请求。
SYN_SEND
在TCP三次握手中，主动连接端发送了SYN包后，进入SYN_SEND状态，等待对方的ACK包。
SYN_RECV
在TCP三次握手中，主动连接端收到SYN包后，进入SYN_RECV状态。
ESTABLISHED
完成TCP三次握手后，主动连接端进入ESTABLISHED状态。此时，TCP连接已经建立，可以进行通信。

FIN_WAIT_1      在TCP四次断开时，主动关闭端发送FIN包后，进入FIN_WAIT_1状态。
FIN_WAIT_2      在TCP四次断开时，主动关闭端收到ACK包后，进入FIN_WAIT_2状态。
TIME_WAIT       在TCP四次断开时，主动关闭端发送了ACK包之后，进入TIME_WAIT状态。
CLOSE_WAIT      在TCP四次断开时，被动关闭端收到FIN包后，进入CLOSE_WAIT状态。
LAST_ACK        在TCP四次断开时，被动关闭端发送FIN包后，进入LAST_ACK状态，等待对方的ACK包。
```

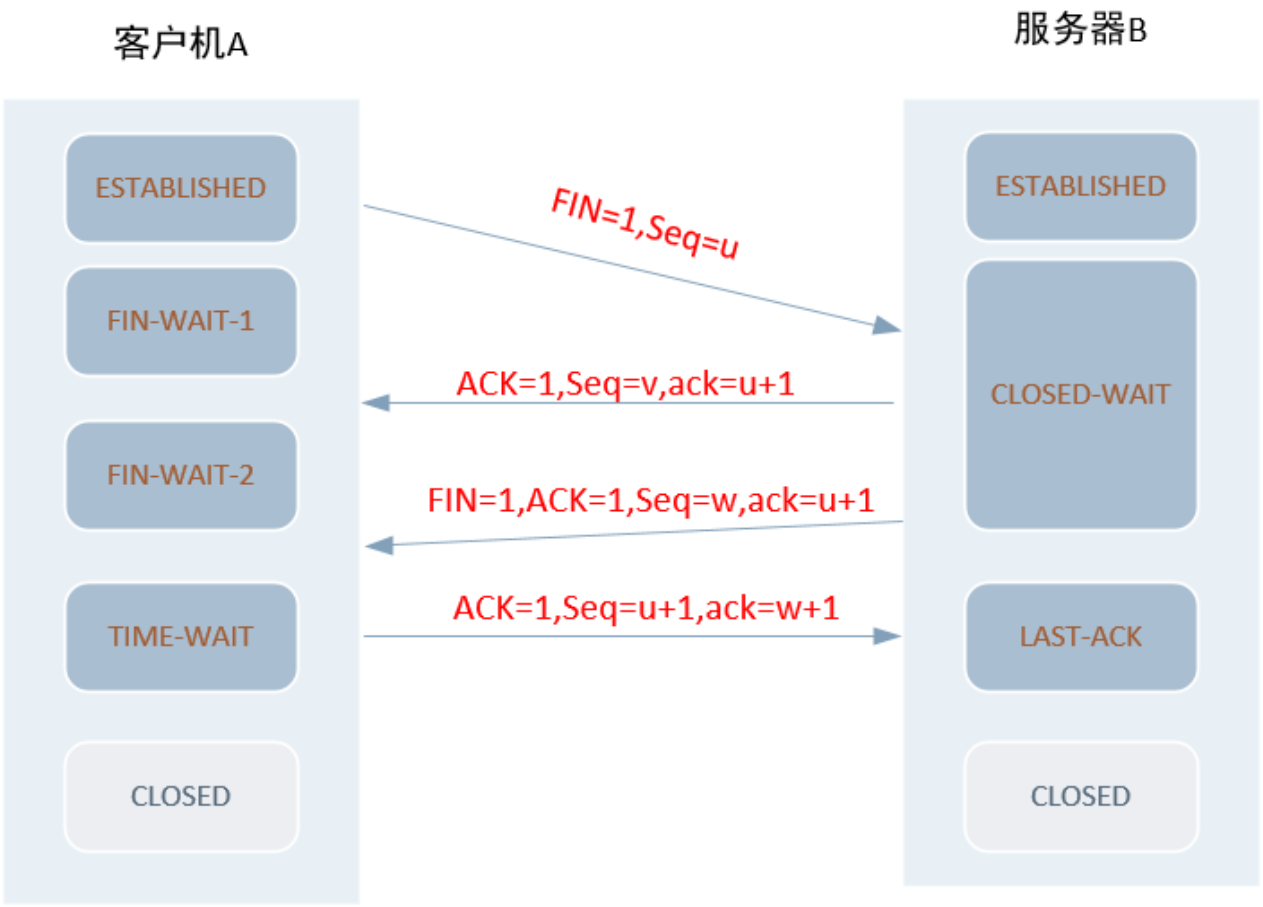
3. TCP/IP三次握手



TCP三次握手的过程如下：

1. 客户机A端（主动连接端）发送一个SYN包给服务器B端（被动连接端）；
2. 服务器B端（被动连接端）收到SYN包后，发送一个带ACK和SYN标志的包给客户机A端（主动连接端）；
3. 客户机A端（主动连接端）发送一个带ACK标志的包给服务器B端（被动连接端），握手动作完成。

4. TCP/IP四次断开



TCP四次断开的过程如下：
1. 客户机A端（主动连接端）发送一个FIN包给服务器B端（被动连接端）请求断开连接；
2. 服务器B端（被动连接端）收到FIN包后，发送一个ACK包给客户机A端（主动连接端）；
3. 服务器B端（被动连接端）发送了ACK包后，再发送一个FIN包给客户机A端（主动连接端）确认断开；
4. 客户机A端（主动连接端）收到FIN包后，发送一个ACK包，当服务器B端（被动连接端）收到ACK包后，四次断开动作完成，连接断开。

四、路由表

思考:

什么是交换,什么是路由,什么是路由表?

1. 交换是指**同网络访问**（两台机器连在同一个交换机上，配置同网段的不同ip就可以直接通讯)
2. 路由就是**跨网络访问**(路径选择)
3. 路由表是**记录路由信息的表**，在Linux中首先是一张可见的,可更改的表,它的作用就是当数据包发到Linux的时候，系统（或者说内核)就根据这张表中定义好的信息来决定这个数据包接下来该怎么走.

1. 查看路由表信息

```
//route命令用来查看和设置路由表信息

[root@server ~]# route -n     //查看路由表信息
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.0.0     0.0.0.0         255.255.255.0   U     0      0        0 eth1
10.1.1.0        0.0.0.0         255.255.255.0   U     0      0        0 eth0
0.0.0.0         10.1.1.254      0.0.0.0         UG    0      0        0 eth0
目标网络        网关            子网掩码        路由标志            网卡
                                    Up:启动状态
                                    UG:该网关为路由器
```

2. 读懂路由信息

```
讨论1:
按上面的路由表来看，如果我ping一个公网IP（如ping 14.200.151.38），应该怎么走?

1）先看目标ip是否为本地ip，如果是，则直接访问本地;如果不是，则找路由表里是否有你访问的网段.
2）如果路由表有则从这个路由条目后面指定的网卡出去；如果路由表里没有你访问的网段，则会找默认路由
   （也就是网关）。
3）如果网关也没有的话，则会报错网络不可达。
connect: Network is unreachable


讨论2:
按上面的路由表来看，如果我ping一个局域网IP为10.1.1.10，会怎么走?

ping 10.1.1.10不会走网关，而是走本地路由从eth0网卡出去（因为路由表里有10.1.1.0/24的路由）。

讨论3:
如何加网关和删除网关，加网关有什么要求?
route add default gw  x.x.x.x     临时加网关，马上生效
route del default gw  x.x.x.x        临时删网关，马上生效
永久增加网关:
vim  /etc/sysconfig/network-scripts/ifcfg-eth0
GATEWAY=x.x.x.x
或者
vim /etc/rc.local    操作系统开机最后读取的一个文件
..
route add default gw xxxxx

注意事项:
1. 加网关只能加你已经有的路由网段里的一个IP才行（此IP不一定存在）
2. 加网关可以不用指定子网掩码（因为是已有的一个网段的ip，所以掩码已经确认了）

讨论4:
一个linux服务器上能有几个有效网关?
准确来说：一个路由表上可以加多个网关，但只有一个生效。

讨论5:
我一台linux上如果有双物理网卡，请问可不可以两个网卡配置同网段的不同IP呢?
eth0 10.1.1.1/24
```

```
eth1 10.1.1.2/24

如果两个网卡同网段，则会有下面两条路由
10.1.1.0        0.0.0.0         255.255.255.0   U     0      0        0 eth0
10.1.1.0        0.0.0.0         255.255.255.0   U     0      0        0 eth1

结果：
它会实现从两张网卡进来的包，却从一张网卡出去，问题就产生了。假设eth0网卡有问题时，路由表里匹配到第一条后，依然会走eth0网卡，而不会走eth1。
也有解决方法（比如多路由表或者双网卡绑定），这里不涉及。
```

五、路由选择实验

1. route命令介绍

```
route -n  查看路由,显示ip,不解析
route del default gw 10.1.1.254    删除默认路由
route add default gw 192.168.1.110    添加一个默认网关，把所有不知道的网络交给网关来转发
route add -net 192.168.2.0/24 dev eth0    对一个网络添加一个新的路由（另一个网段）
route del -net 192.168.2.0/24
```

2. 实验要求

```
环境准备：
server:10.1.1.250
作为网关服务器，开启路由转发功能/proc/sys/net/ipv4/ip_forward
node1:192.168.0.254
node2:10.12.0.254
要求：
实现不同网络(10.12.0.0/24和192.168.0.0/24)之间的互通，使用第三方主机server作为路由进行转发
```

3. 具体步骤

```
思路：
1. 中间人server服务器开启路由转发功能；中间人需要和node1和node2主机互通
2. node1主机和node2主机的默认网关应该是中间人server服务器的IP地址

步骤：
1. server服务器（中间人）完成以下任务
1）开启路由转发功能
临时开启：
[root@server ~]# cat /proc/sys/net/ipv4/ip_forward
0
[root@server ~]# echo 1 > /proc/sys/net/ipv4/ip_forward
[root@server ~]# cat /proc/sys/net/ipv4/ip_forward
1
永久开启：
[root@server ~]# vim /etc/sysctl.conf
# Controls IP packet forwarding
net.ipv4.ip_forward = 1
```

2）分别添加到node1和node2两台主机所在的网络

```
[root@server ~]# route add -net 192.168.0.0/24 dev eth1
[root@server ~]# route add -net 10.12.0.0/24 dev eth1
[root@server ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.12.0.0       0.0.0.0         255.255.255.0   U     0      0        0 eth1
192.168.0.0     0.0.0.0         255.255.255.0   U     0      0        0 eth1
10.1.1.0        0.0.0.0         255.255.255.0   U     0      0        0 eth1
169.254.0.0     0.0.0.0         255.255.0.0     U     1003   0        0 eth1
0.0.0.0         10.1.1.254      0.0.0.0         UG    0      0        0 eth1
```

2. 分别配置node1和node2的IP和网关

node1: 192.168.0.254

```
[root@node1 network-scripts]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.0.0     0.0.0.0         255.255.255.0   U     1      0        0 eth0

[root@node1 network-scripts]# route add default gw 10.1.1.250
[root@node1 network-scripts]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.0.0     0.0.0.0         255.255.255.0   U     1      0        0 eth0
10.1.1.0        0.0.0.0         255.255.255.0   U     0      0        0 eth0
0.0.0.0         10.1.1.250      0.0.0.0         UG    0      0        0 eth0
```

node2:10.12.0.254

```
[root@node2 network-scripts]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.12.0.0       0.0.0.0         255.255.255.0   U     1      0        0 eth1
[root@node2 network-scripts]# route add -net 10.1.1.0/24 dev eth1
[root@node2 network-scripts]# route add default gw 10.1.1.250
[root@node2 network-scripts]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.12.0.0       0.0.0.0         255.255.255.0   U     1      0        0 eth1
10.1.1.0        0.0.0.0         255.255.255.0   U     0      0        0 eth1
0.0.0.0         10.1.1.250      0.0.0.0         UG    0      0        0 eth1
```

3. 测试验证

1）中间主机分别ping node1和node2

```
[root@server ~]# ping 192.168.0.254
PING 192.168.0.254 (192.168.0.254) 56(84) bytes of data.
64 bytes from 192.168.0.254: icmp_seq=1 ttl=64 time=0.410 ms
64 bytes from 192.168.0.254: icmp_seq=2 ttl=64 time=0.273 ms

[root@server ~]# ping 10.12.0.254
PING 10.12.0.254 (10.12.0.254) 56(84) bytes of data.
64 bytes from 10.12.0.254: icmp_seq=1 ttl=64 time=19.8 ms
64 bytes from 10.12.0.254: icmp_seq=2 ttl=64 time=0.263 ms
```

```
2) node1和node2相互ping
[root@node2 ~]# ping 192.168.0.254
PING 192.168.0.254 (192.168.0.254) 56(84) bytes of data.
64 bytes from 192.168.0.254: icmp_seq=1 ttl=64 time=0.239 ms
64 bytes from 192.168.0.254: icmp_seq=2 ttl=64 time=0.275 ms

[root@node1 ~]# ping 10.12.0.254
PING 10.12.0.254 (10.12.0.254) 56(84) bytes of data.
64 bytes from 10.12.0.254: icmp_seq=1 ttl=64 time=0.250 ms
64 bytes from 10.12.0.254: icmp_seq=2 ttl=64 time=0.323 ms
```

六、Vmware网络模式

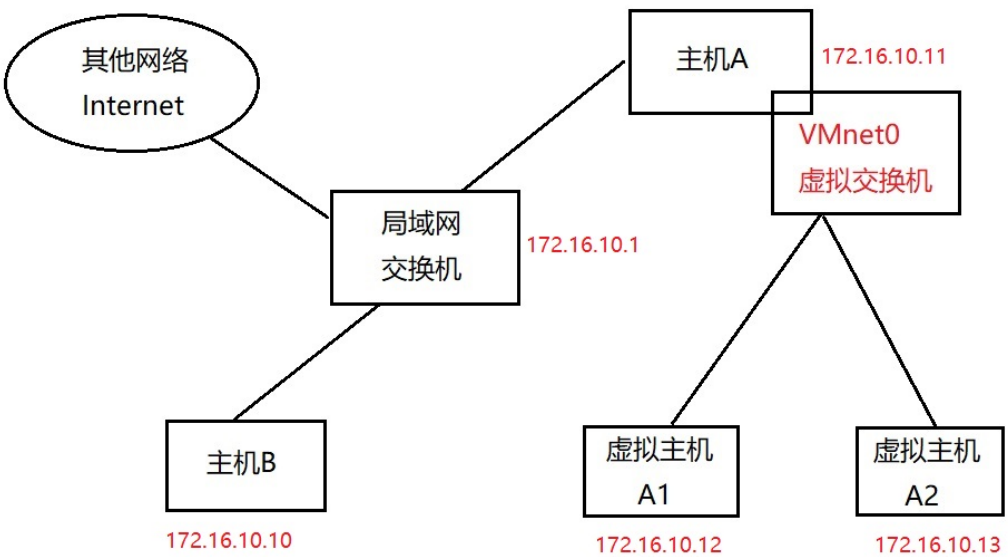
1. 虚拟设备

> VMnet0：用于虚拟桥接网络下的虚拟交换机
>
> VMnet1：用于虚拟Host-Only网络下的虚拟交换机
>
> VMnet8：用于虚拟NAT网络下的虚拟交换机
>
> VMware Network Adepter VMnet1：Host用于与Host-Only虚拟网络进行通信的虚拟网卡　　VMware Network Adepter VMnet8：Host用于与NAT虚拟网络进行通信的虚拟网卡

2. 三种网络模式

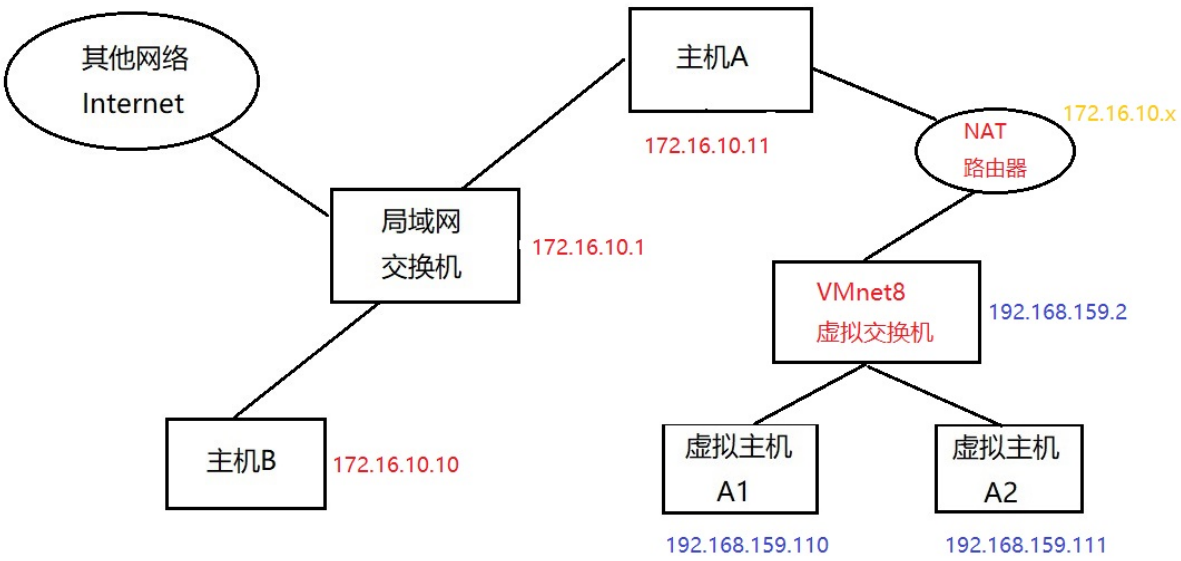
- 桥接网络

> 桥接网络是指虚拟网卡通过VMnet0虚拟交换机和本地物理网卡进行桥接，那么物理网卡和虚拟网卡就相当于处于同一个网段，虚拟交换机就相当于一台现实网络中的交换机。所以要想虚拟机也可以连接到互联网中，那么两个网卡的IP地址也要设置为同一网段。



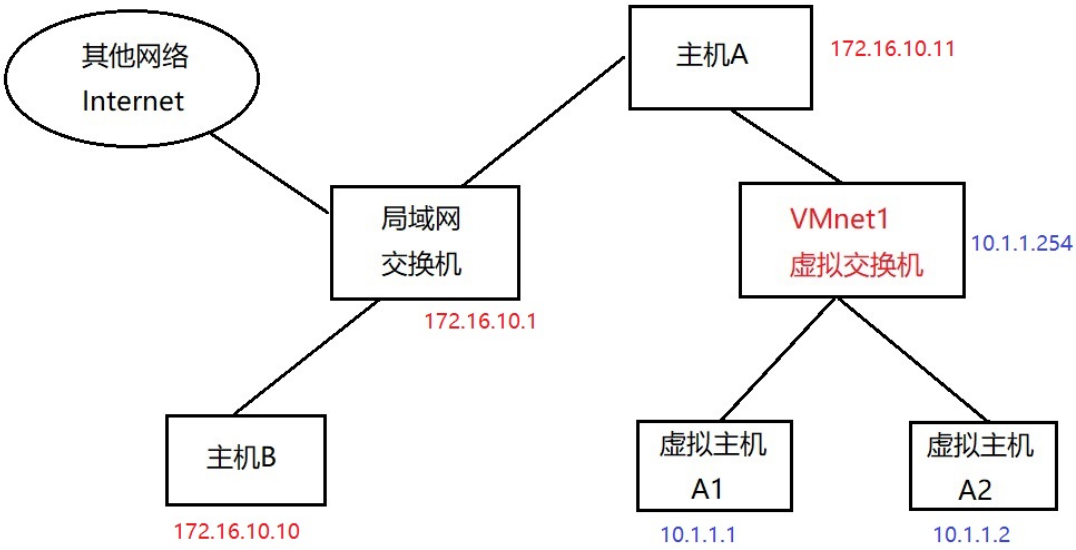
- NAT网络

> 在NAT网络中，会用到VMware Network Adepter VMnet8虚拟网卡，主机上的VMware Network Adepter VMnet8虚拟网卡被直接连接到VMnet8虚拟交换机上与虚拟网卡进行通信。VMware Network Adepter VMnet8虚拟网卡的作用仅限于和VMnet8网段进行通信，它不给VMnet8网段提供路由功能，所以虚拟机虚拟一个NAT服务器，使虚拟网卡可以连 接到Internet。 VMware Network Adepter VMnet8虚拟网卡的IP地址是在安装VMware时由系统指定生成的，我们尽量不要修改这个数值，否则可能会使主机和虚拟机无法通信。



- Host-Only模式

> 在Host-Only模式下，虚拟网络是一个全封闭的网络，它唯一能够访问的就是物理真机。其实Host-Only网络和NAT网络很相似，不同的地方就是Host-Only网络没有NAT服务，所以虚拟网络不能连接到Internet。主机和虚拟机之间的通信是通过VMware Network Adepter VMnet1虚拟网卡来实现的。



1. 数据单元是网络信息传输的基本单位。一般网络连接不允许传送任意大小的数据包，而是采用分组技术将一个数据分成若干个很小的数据包，并给每个小数据包加上一些关于此数据包的属性信息.↩

2. 包(Packet)是TCP/IP协议通信传输中的数据单位，一般也称“数据包”。↩

3. 帧是数据链路层的传输单元。它将上层传入的数据添加一个头部和尾部，组成了帧.↩