day06-网络管理-dhcp-Cluster-chrony

1.网络基本概述

1.1 为什么需要网络

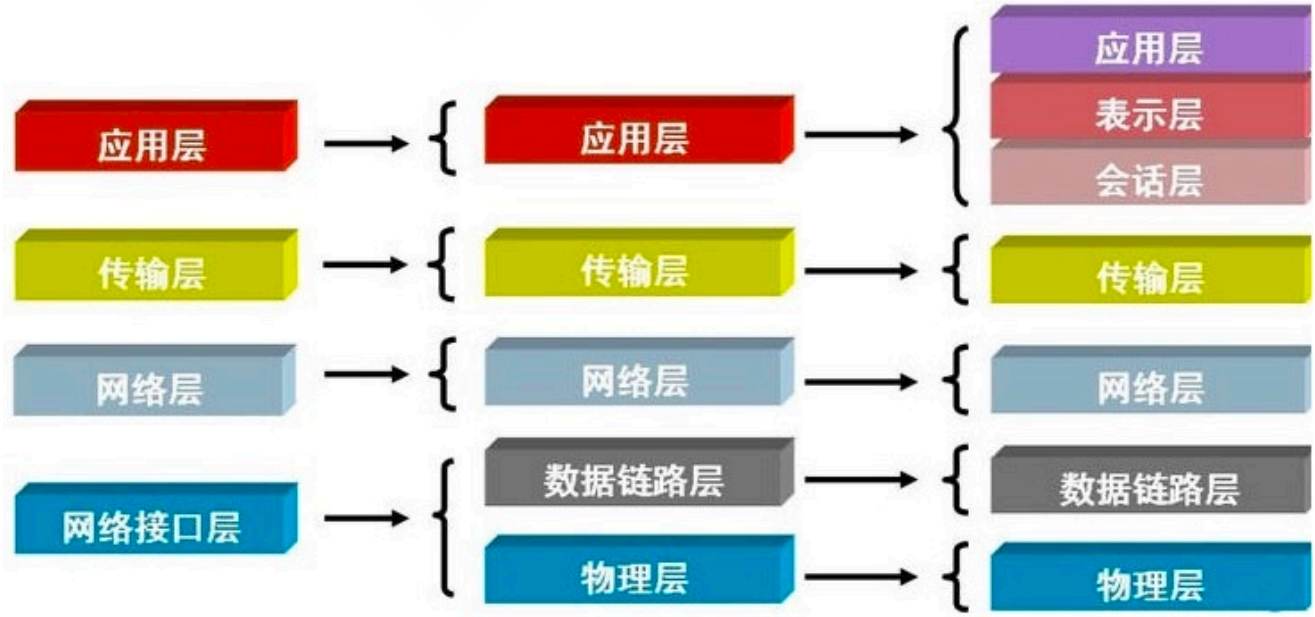
- 假设没有网络：（也就是将所有的计算机网络都关闭）
  - 如果我的计算机上有非常不错的电影，想要进行传输，那就比较的费劲了；
  - 因为我们可能处在不同的城市、或者不同的国家；
- 但如果有了网络：（也就是将所有计算机通过网线连接在一起）
  - 1.打破了地域上数据传输的限制；
  - 2.提高信息之间的传输效率，以便更好的实现”资源的共享“；

1.2 什么是网络

- 网络是由"若干节点"和"连接这些节点"的链路构成，表示诸多对象及其相互联系。
- 网络是信息传输、接收、共享的虚拟平台，通过它把各个信息联系到一起，从而实现这些资源的共享。
- 网络将节点连接在一起，需要实现 ”信息传输“（信息通信）有几个大前提：
  - 1.使用物理连接的介质将所有计算机连接在一起（网卡、网线、交换机、路由器）；
  - 2.双方在通信过程中，必须使用统一的通信标准，也就是通讯协议（互联网通讯协议）；

2.互联网通讯协议

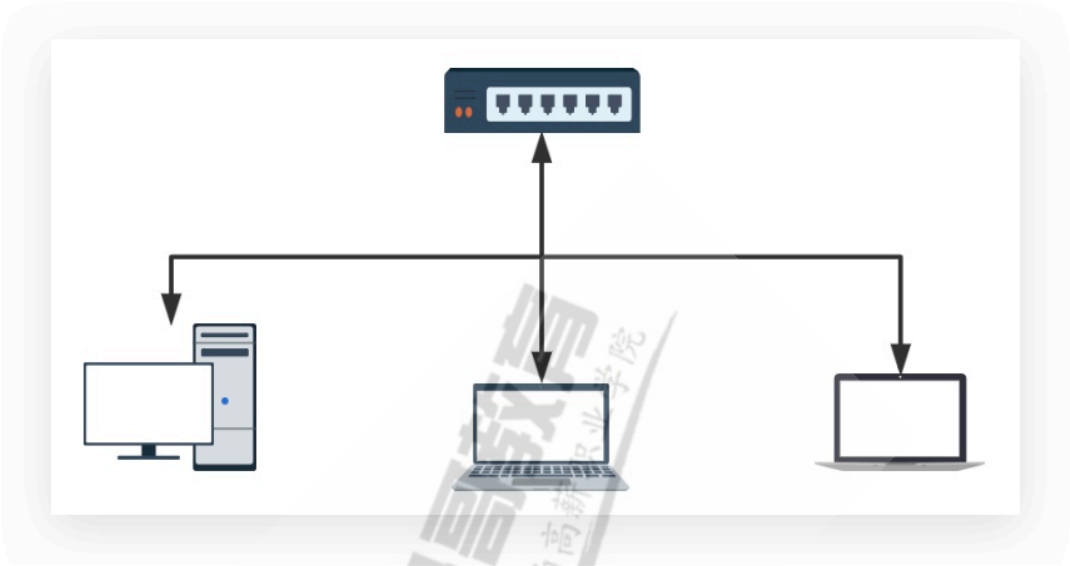
- 协议其实就是规定了一堆标准，用来定义计算机如何接入 `internet` 以及接入 `internet` 的计算机通信的标准；所以计算机都需要学习此标准、遵循此标准来进行信息传输（信息通信）；
- 国际标准化组织：推出了 `OSI` 七层参考模型，将互联网通讯协议分成了不同的层，每一层都有专门的标准，以及组织数据的格式；
  - （应、表、会、传、网、数、物）
- 对于写程序来说，通常会将七层归纳为五层协议；
  - （应、传、网、数、物）
- 所以我们需要学习协议的规定了哪些标准；



以太网封装与解封装动态图

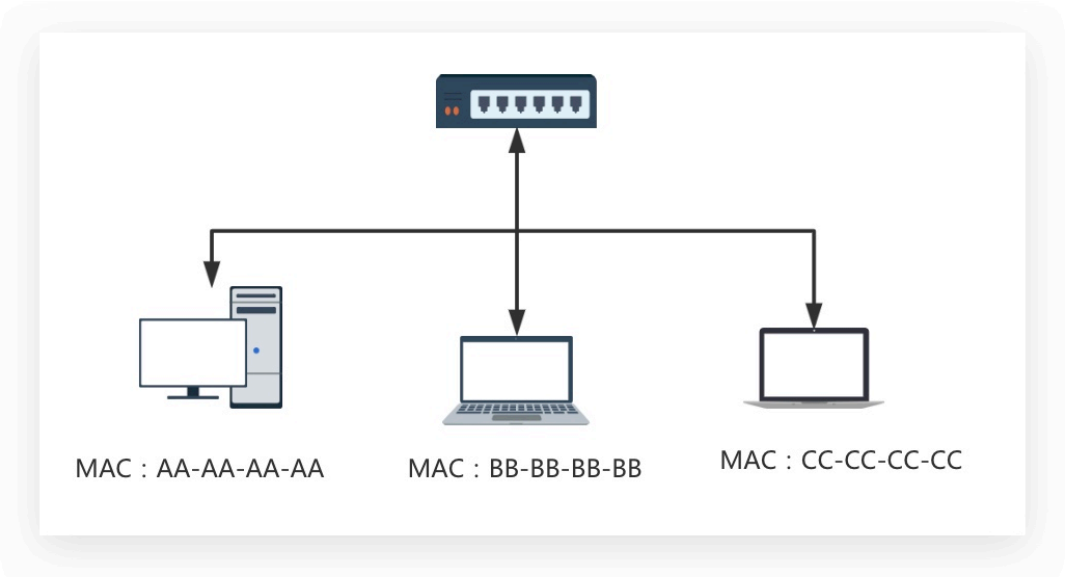
2.1 物理层

物理层：定义物理设备的标准，如网卡网线，传输速率；最终实现数据转成电信号传输；
问题：如果只是单纯发送电信号是没有意义的，因为没有规定开头也没有规定结尾；要想变得有意义就必须对电信号进行分组；比如：xx位为一组、这样的方式去传输，这就需要“数据链路层”来完成了；



2.2 数据链路层

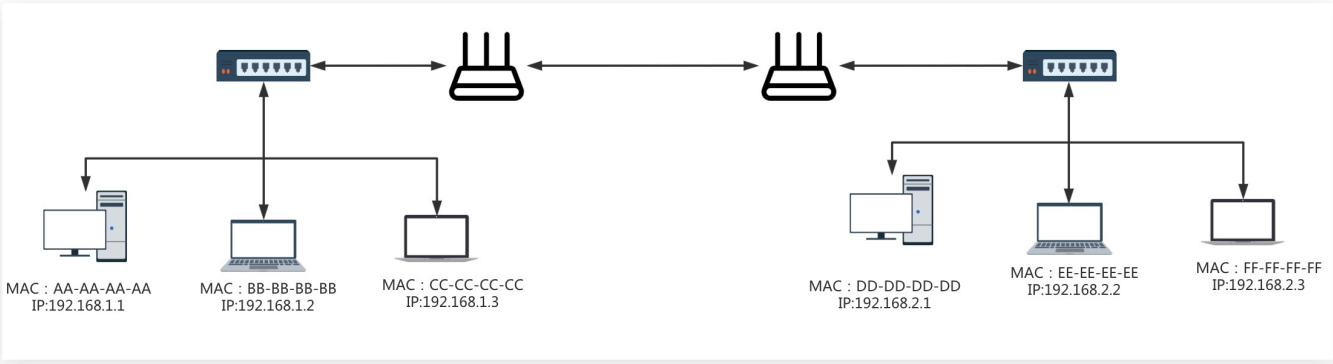
- 数据链路层定义：定义了电信号的分组的标准方式，一组数据称之为一个数据帧，这个标准遵循 `ethernet` 以太网协议，以太网规定了如下几件事；
- 1.数据帧分为 `head` 和 `data` 两部分组成；其中 `head` 长度固定18字节；
  - `head`：发送者/源地址、接收者/目标地址（源地址6字节、目标地址6字节、数据类型6字节）
    - 源地址：`MAC` 地址
    - 目标地址：`MAC` 地址
  - `data`：主要存放是网络层整体的数据，最长1500字节，超过最大限制就分片发送；
- 2.但凡接入互联网的主机必须有一块网卡，网卡烧制了全世界唯一的 `MAC` 地址；
- 3.有了以太网协议规定以后，它能对数据分组、也可以区分数据的意义，还能找到目标主机的地址、就可以实现计算机通信；但是计算机是瞎的，所以以太网通信采用的是"广播"方式；



- 那什么是广播：
  - 假设我们都在一个小黑屋里面，大家互相通信靠吼，假设 `oldxu` 让 `laowang` 买包烟；
    - 1.数据：买烟（类型：干粮）
    - 2.源地址：`oldxu`
    - 3.目标地址：`laowang`
  - 此时屋子里所有人都收到了该数据包，但只有 `laowang` 会接收执行，其他人收到后会丢弃；
- 如果我们将全世界的计算机都接入在一起，理论上是不是就可以实现全世界通信：
  - 首先：无法将全世界计算机放在一个交换机上，因为没有这样的设备；
  - 其次：就算放在同一设备上，每台计算机都广播一下，那设备也无法正常工作；
  - 所以：我们应该将主机划区域，隔离在一个又一个的区域中，然后将多个区域通过"网关/路由"连接在一起；

2.3 网络层

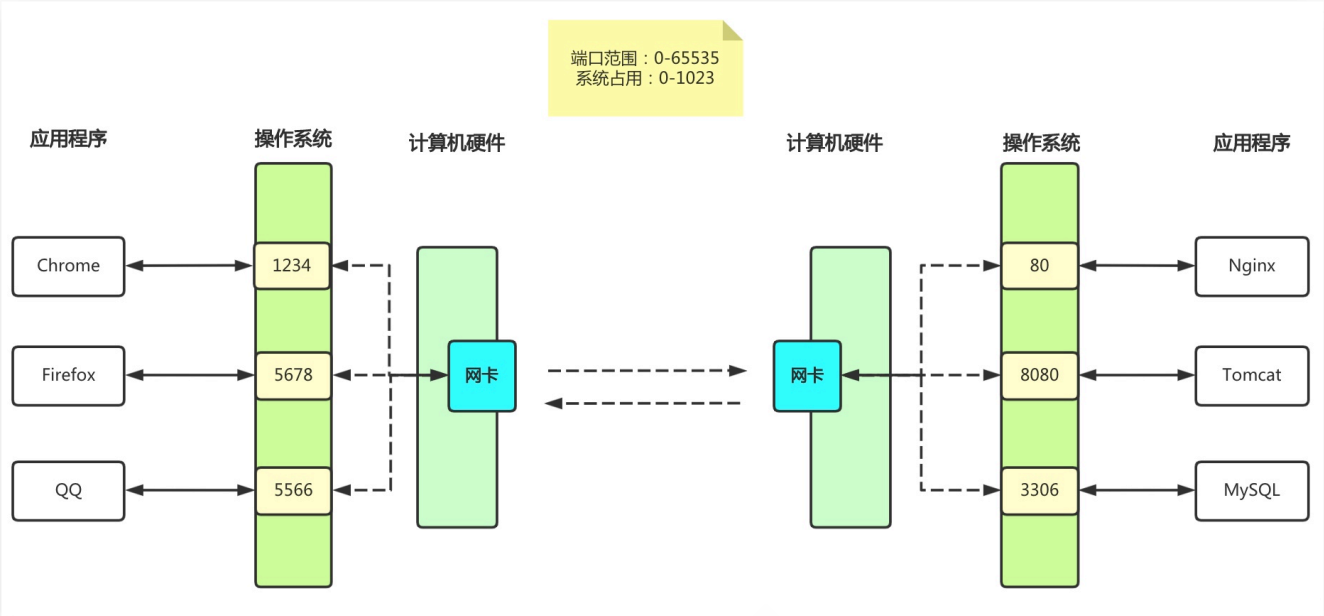
- 网络层定义：用来划分广播域，如果广播域内主机要往广播域外的主机发送数据，一定要有一个"网关/路由"帮其将数据转发到外部计算机；网关和外界通信走的是路由协议（这个我们不做详细阐述）。其次网络层协议规定了如下几件事；
  - 规定1：数据包分成：`head` 和 `data` 两部分组成；
    - `head`：发送者/源地址、接收者/目标地址，该地址为IP地址；
    - `data`：主要存放是传输层整体的数据；
  - 规定2：`IP` 地址来划分广播域，主要用来判断两台主机是否在同一广播域中；
    - 一个合法 `IPV4` 地址组成部分=`ip地址/子网掩码`，在线子网计算器
    - 如果计算出两台地址的广播域一样，说明两台计算机处在同一个区域中；



- 计算两台计算机是否在同一局域网（牵扯到如何发送数据）：
- 假设：现在计算机1要与计算机2通信，计算机1必须拿到计算机2的ip地址；
  - 如果它们处于同一网络（局域网）`10.0.0.1-->10.0.0.100`：
    - 1.本地电脑根据数据包检查目标 IP 如果为本地局域网；
    - 2.直接通过交换机广播MAC寻址；将数据包转发过去；
  - 如果它们处于不同网络（跨局域网）`10.0.0.1-->39.104.16.126`：
    - 1.本地根据数据包检查目标`IP`如果不为本地局域网，则尝试获取网关的`MAC`地址；
    - 2.本地封装数据转发给交换机，交换机拆解发现目标`MAC`是网关，则送往网关设备；
    - 3.网关收到数据包后，拆解至二层后发现请求目标`MAC`是网关本机`MAC`；
    - 4.网关则会继续拆解数据报文到三层，发现目标地址不为网关本机；
    - 5.网关会重新封装数据包，将源地址替换为网关的`WAN`地址，目标地址不变；
    - 6.出口路由器根据自身路由表信息将数据包发送出去，直到送到目标的网关；

2.4 传输层

- 传输层的由来：网络层帮我们区分子网，数据链路层帮我们找到主机，但一个主机有多个进程，进程之间进行不同的网络通信，那么当收到数据时，如何区分数据是那个进程的呢；其实是通过端口来区分；端口即应用程序与网卡关联的编号。
- 传输层的定义：提供进程之间的逻辑通信；
- 传输层也分成：`head`和`data`两部分组成；
  - `head`：源端口、目标端口、协议（TCP、UDP）；
  - `data`：主要存放是应用层整体的数据；



2.5 应用层

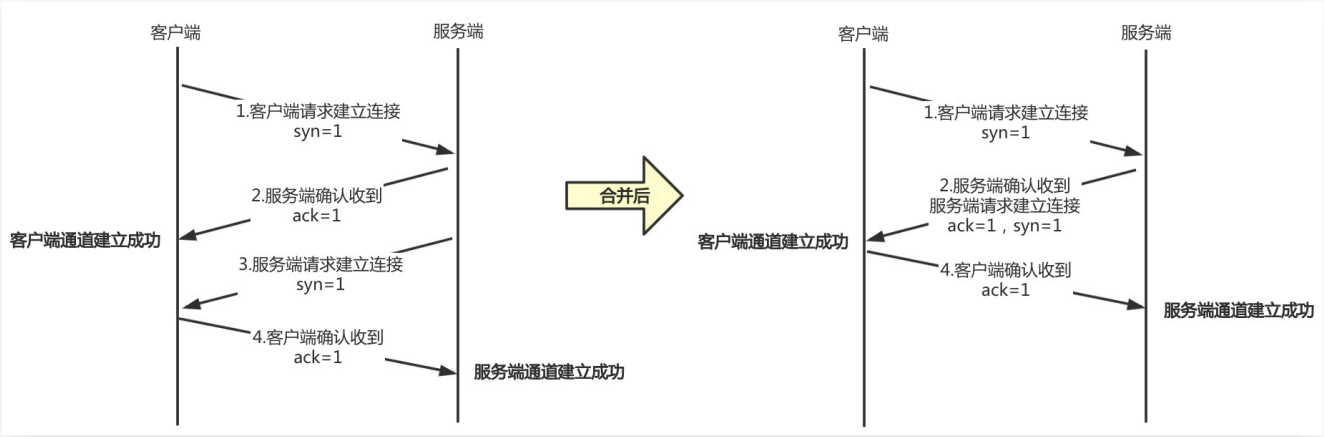
- 应用层定义：为终端应用提供的服务，如我们的浏览器交互时候需要用到的 HTTP 协议，邮件发送的 SMTP，文件传输的 FTP 等。

2.6 TCP协议

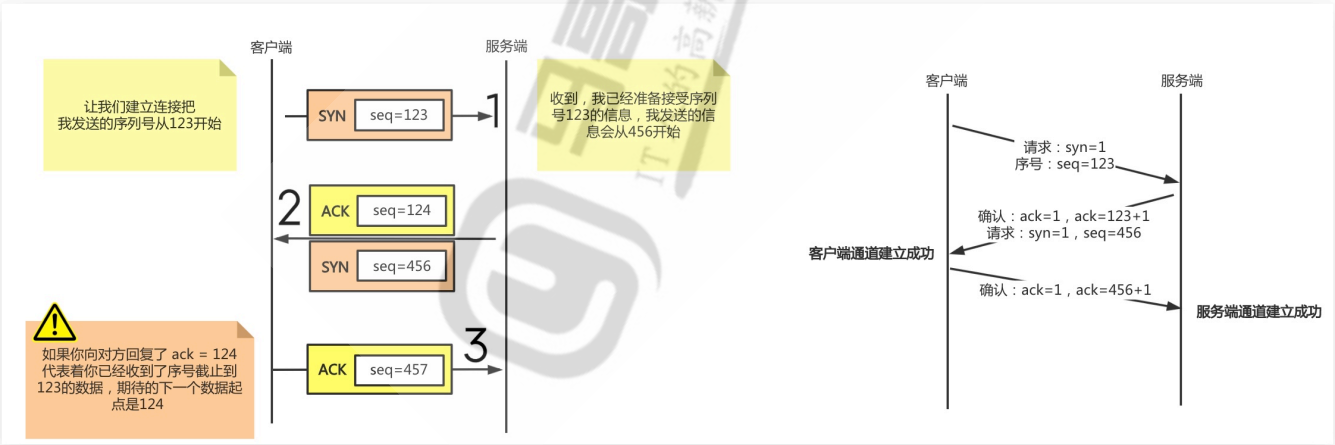
- tcp可靠数据传输协议；为了实现可靠传输，在通信之前需要先建立连接，也叫"双向通路"，就是说客户端与服务端要建立连接，服务端与客户端也需要建立连接，当然建立的这个双向通路它只是一个虚拟的链路，不是用网线将两个设备真实的捆绑在一起；
- 虚拟链路的作用：由于每次通信都需要拿到IP和Port，那就意味着每次都需要查找，建立好虚拟通路，下次两台主机之间就可以直接传递数据；

2.6.1 三次握手

- 第一次：客户端要与服务端建立连接，需要发送请求连接消息；
- 第二次：服务端接收到数据后，返回一个确认操作（至此客户端到服务端链路建立成功）；
- 第三次：服务端还需要发送要与客户端建立连接的请求；
- 第四次：客户端接收到数据后，返回一个确认的操作（至此服务端到客户端的链路建立成功）；
- 由于建立连接时没有数据传输，所以第二次确认和第三次请求可以合并为一次发送；

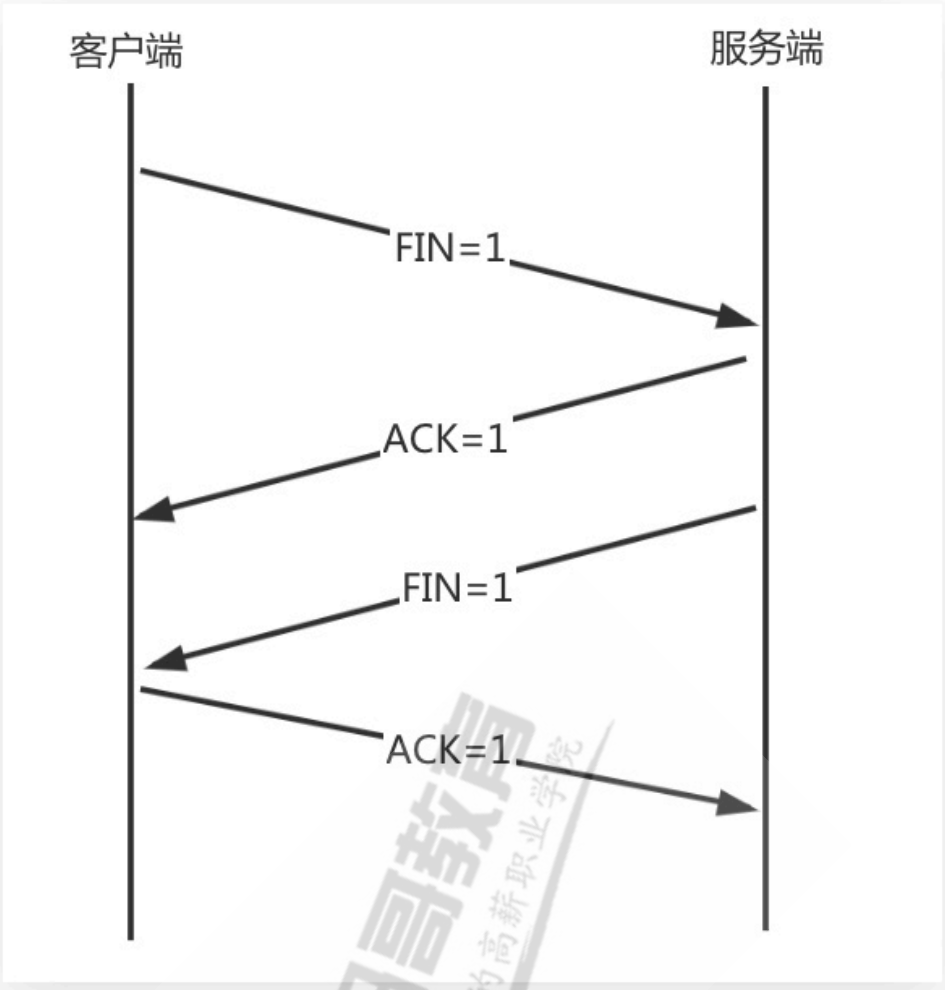


- TCP协议为了实现可靠传输，通信双方需要判断自己已经发送的数据包是否都被接收方收到，如果没收到，就需要重发。为了实现这个需求，就引出序号（seq）和确认号（ack）的使用。
- 举例：发送方在发送数据包时，序列号(假设为 123)，那么接收方收到这个数据包以后， 就可以回复一个确认号（124=123+1）告诉发送方 “我已经收到了你的数据包，你可以发送下一个数据包，序号从 124 开始”，这样发送方就可以知道哪些数据被接收到，哪些数据没被接收到，需要重发。



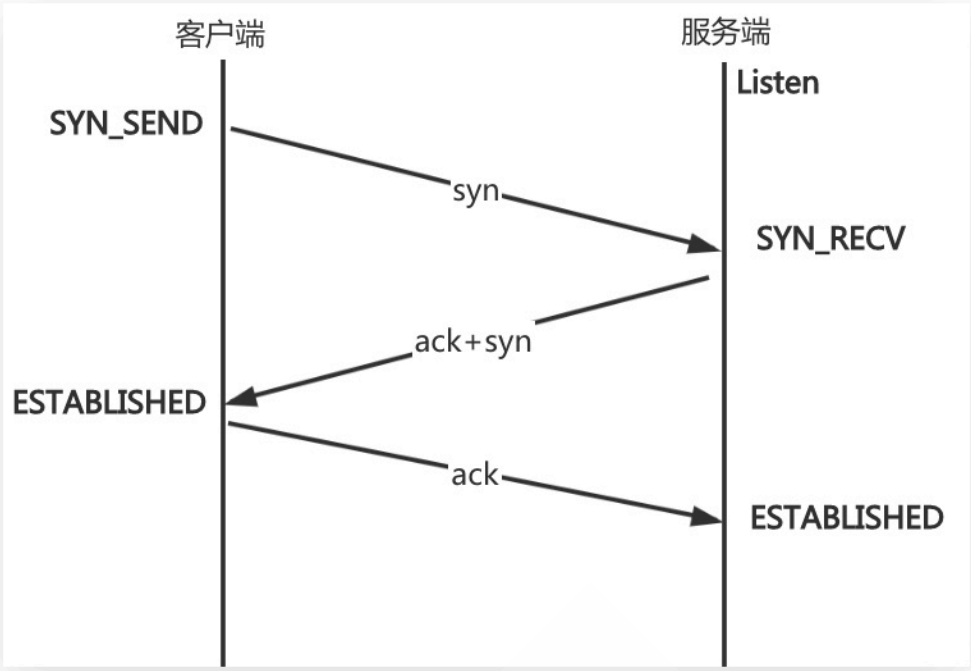
2.6.2 四次挥手

- 第一次挥手：客户端（服务端也可以主动断开）向服务端说明想要关闭连接；
- 第二次挥手：服务端会回复确认。但不是立马关闭，因为此时服务端可能还有数据在传输中；
- 第三次挥手：待到服务端数据传输都结束后，服务端向客户端发出消息，我要断开连接了；
- 第四次挥手：客户端收到服务端的断开信息后，给予确认。服务端收到确认后正式关闭。

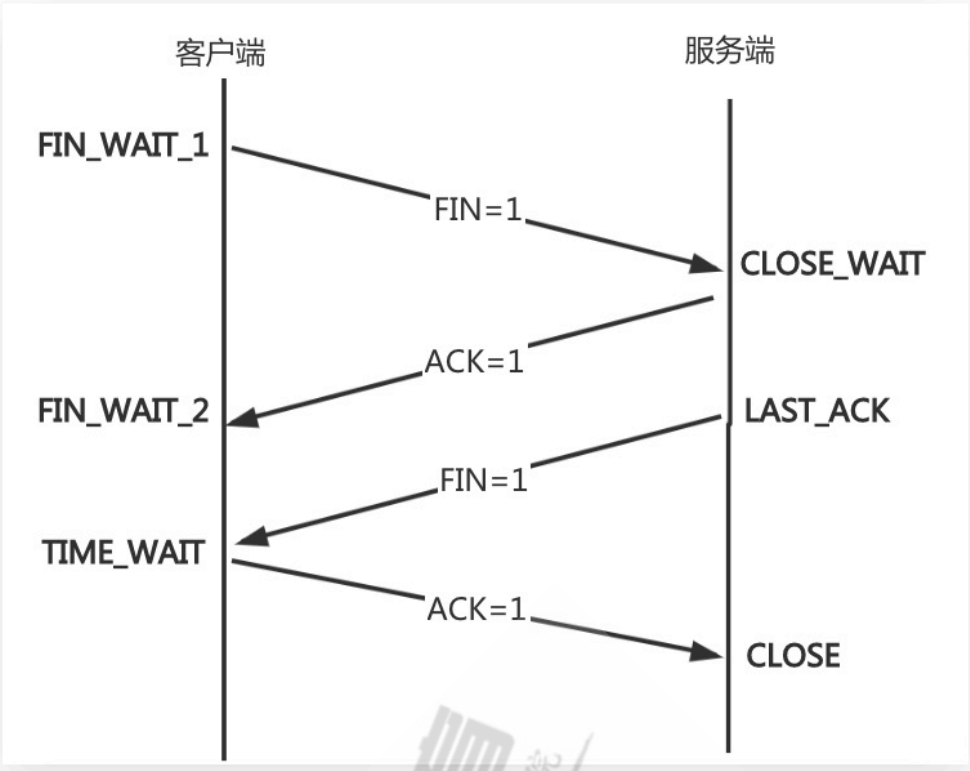


2.6.3 转换状态

- 三次握手状态转换：
- 1.客户端发送syn包向服务端请求建立 `TCP` 连接，客户端进入 `SYN_SEND` 状态；
- 2.服务端收到请求之后，向客户端发送 `SYN+ACK` 的合成包，同时自身进入 `SYN_RECV` 状态；
- 3.客户端收到回复之后，发送 `ACK` 信息，自身进入 `ESTABLISHED` 状态；
- 4.服务端收到ACK数据之后，进入 `ESTABLISHED` 状态。



- 四次挥手过状态转换：
- 1.客户端发送完数据之后，向服务器请求断开连接，自身进入 `FIN_WAIT_1` 状态；
- 2服务端收到 `FIN` 包之后，回复 `ACK` 包表示已经收到，但此时服务端可能还有数据没发送完成，自身进入 `CLOSE_WAIT` 状态，表示对方已发送完成且请求关闭连接，自身发送完成之后可以关闭连接；
- 3.服务端数据发送完成后，发送 `FIN` 包给客户端，自身进入 `LAST_ACK` 状态，等待客户端 `ACK` 确认；
- 4.客户端收到 `FIN` 包之后，回复一个 `ACK` 包，并进入 `TIME_WAIT` 状态；
- 注意： `TIME_WAIT` 状态比较特殊，当客户端收到服务端的 `FIN` 包时，理想状态下，是可以直接关闭连接了；但是有几个问题：
  - 问题1：网络是不稳定的，可能服务端发送的一些数据包，比服务端发送的 `FIN` 包还晚到；
  - 问题2：.如果客户端回复的ACK包丢失了，服务端就会一直处于 `LAST_ACK` 状态，如果客户端没有关闭，那么服务端还会重传 `FIN` 包，然后客户端继续确认；
- 所以客户端如果 `ACK` 后立即关闭连接，会导致数据不完整、也可能造成服务端无法释放连接。所以此时客户端需要等待2个报文生存最大时长，确保网络中没有任何遗留报文了，再关闭连接；
- 如果机器 `TIME_WAIT` 过多，会造成端口会耗尽，可以修改内核参数 `tcp_tw_recycle=1` 端口重用；



2.7 UDP协议

- `udp`是不可靠传输协议；不可靠指的是传输数据时不可靠；
- `udp`协议不需要先建立连接，只需要获取服务端的`ip+port`，发送完毕也无需服务器返回`ack`；
- `udp`协议如果在发送在数据的过程中丢了，那就丢了；

3.Linux网络配置

3.1网卡命名规则

- `Centos6`网卡命名规则是`eth0、eth1....`
- `Centos7`网卡命名规则是`ens32、ens33...`
- 由于这种无规则的命名方法给维护带来了困难，所以需要将网卡命名规则修改为`eth0、eth1...`

3.1.1 已安装系统网卡名称修改

1.已安装`Linux7`系列操作系统, 修改网卡命名规则为`eth0 eth1`

1.修改网卡配置文件

```
[root@linux-node2~]# cd /etc/sysconfig/network-scripts/
[root@linux-node2network-scripts]# mv ifcfg-ens32 ifcfg-eth0
[root@linux-node2 network-scripts]# vim ifcfg-eth0
NAME=eth0
DEVICE=eth0
```

2.修改内核启动参数，禁用预测命名规则方案，将 `net.ifnames=0 biosdevname=0` 参数关闭

```
[root@xuliangwei~]# vim /etc/sysconfig/grub
GRUB_CMDLINE_LINUX="...net.ifnames=0 biosdevname=0 quiet"
[root@xuliangwei~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

3.重启系统并检查修改结果

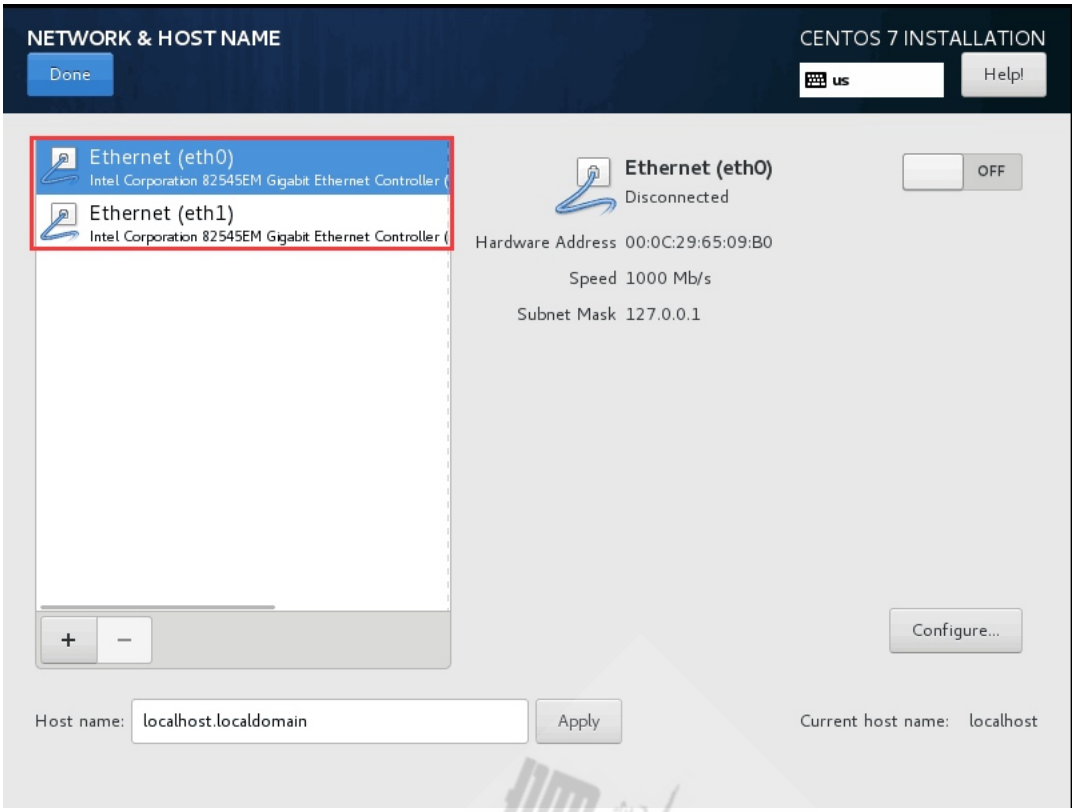
```
[root@xuliangwei~]# reboot
[root@xuliangwei~]# ifconfig
eth0:flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
inet 192.168.56.12  netmask 255.255.255.0  broadcast 192.168.56.255
inet6 fe80::20c:29ff:fe5c:7bb1  prefixlen 64
scopeid 0x20<link>
ether 00:0c:29:5c:7b:b1  txqueuelen 1000  (Ethernet)
    RX packets 152  bytes 14503 (14.1 KiB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 98  bytes 14402 (14.0 KiB)
    TX errors 0  dropped 0 overruns 0  carrier 0    collisions 0
```

3.1.2 新安装系统指定网卡名称

1.在安装系统选择 `Install Centos7` 按下 `Tab` 设定 `kernel` 内核参数；

2.增加内核参数: `net.ifnames=0 biosdevname=0` ;

3.检查是否修改成功，成功后可继续安装系统；



3.2 网络接口查询

3.2.1 ifconfig

- 使用 `ifconfig` 当前处于活动状态的网络接口

```
[root@xuliangwei ~]# yum install net-tools -y
[root@xuliangwei ~]# ifconfig

#仅查看eth0网卡状态信息
[root@xuliangwei ~]# ifconfig eth0

#查看所有网卡状态信息, 包括禁用和启用
[root@xuliangwei ~]# ifconfig -a
#UP: 网卡处于活动状态 BROADCAST: 支持广播 RUNNING: 网线已接入
#MULTICAST: 支持组播 #MTU: 最大传输单元(字节),接口一次所能传输的最大包
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        #inet: 显示IPv4地址行
        inet 10.0.0.100  netmask 255.255.255.0  broadcast 10.0.0.255
        #inet6: 显示IPv6地址行
        inet6 fe80::a879:62cf:396c:e7d9  prefixlen 64  scopeid 0x20<link>
        inet6 fe80::22a2:cb:8a69:bf63  prefixlen 64  scopeid 0x20<link>
        #enther: 硬件(MAC)地址 txqueuelen: 传输缓存区长度大小
        ether 00:0c:29:5f:6b:8a  txqueuelen 1000  (Ethernet)
        #RX packets: 接收的数据包
        RX packets 3312643  bytes 4698753634 (4.3 GiB)
        RX errors 0  dropped 0  overruns 0  frame 0
```

```
        #TX packets: 发送的数据包
        TX packets 235041  bytes 20504297 (19.5 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
#errors: 总的收包的错误数量
#dropped: 拷贝中发生错误被丢弃
#collisions: 网络信号冲突情况, 值不为0则可能存在网络故障
```

3.2.1 ip命令

- 1.使用 `ip` 命令查看当前地址

```
[root@xuliangwei ~]# ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,①UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
qlen 1000
    ②link/ether 00:0c:29:5f:6b:8a ff:ff:ff:ff:ff:ff
    ③inet 10.0.0.100/24 brd④ 192.168.69.255 scope global ens32
       valid_lft forever preferred_lft forever
    ⑤inet6 fe80::bd23:46cf:a12e:c0a1/64 scope link
       valid_lft forever preferred_lft forever

#①: 活动接口为UP
#②: Link行指定设备的MAC地址
#③: inet行显示IPv4地址和前缀
#④: 广播地址、作用域和设备名称在此行
#⑤: inet6行显示IPv6信息
```

2.使用 `ip -s link show eth0` 命令查看网络性能的统计信息, 比如: 发送和传送的数据包、错误、丢弃

```
[root@xuliangwei ~]# ip -s link show eth0
3: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT
    link/ether 14:18:77:35:0d:f5 brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped overrun mcast
    518292951  4716385  0       0       0       709280
    TX: bytes  packets  errors  dropped carrier collsns
    23029861512 15391427 0       0       0       0
```

3.3 配置系统网络

- `CentOS7` 系统默认采用 `NetworkManager` 来提供网络服务，这是一种动态管理网络配置的守护进程，能够让网络设备保持连接状态。`NetworkManager` 提供的命令行和图形配置工具对网络进行设定，设定保存的配置文件在 `/etc/sysconfig/network-scripts` 目

录下，工具有 `nmcli、nmtui`

- `NetworkManager` 有如下两个概念需要了解：
  - `device` 物理设备，例如：`enp2s0,virbr0,team0`
  - `connection` 连接设置，具体网络配置方案
  - 一个物理设备 `device` 可以有多套逻辑连接配置，但同一时刻只能使用一个 `connection` 连接配置；

3.3.1 nmcli查看网络

1.使用 `nmcli device` 命令查看设备情况

```
# 查看所有设备
[root@xuliangwei ~]# nmcli device
DEVICE      TYPE      STATE      CONNECTION
eth0        ethernet  connected  eth0
lo          loopback  unmanaged  --

# 指定查看某个设备的详细状态
[root@xuliangwei ~]# nmcli dev show eth0
```

2.使用 `nmcli connection` 命令查看连接状态

```
#查看连接状态
[root@xuliangwei ~]# nmcli connection
NAME    UUID                                  TYPE            DEVICE
eth0    a4319b27-80dc-4d63-a693-2927ea1018e7  802-3-ethernet  eth0


# 查看所有活动连接的状态
[root@xuliangwei ~]# nmcli con show --active

# 查看指定连接状态
[root@xuliangwei ~]# nmcli con show "eth0"
```

3.3.2 nmcli创建dhcp连接

- 使用 `nmcli` 创建一个 `dhcp` 的连接，实质是添加 `/etc/sysconfig/network-scripts/ifcfg-ens33-dhcp` 配置文件

```
[root@linux-node1 ~]# nmcli connection add \
con-name eth0-dhcp ifname eth0 autoconnect yes \
type ethernet ipv4.method auto
```

```
[root@linux-node1 ~]# nmcli connection
NAME        UUID                                  TYPE            DEVICE
eth0        000b9696-19d5-4ade-bca6-7ee0266ddcf0  802-3-ethernet  eth0
eth0-dhcp   33bcddf0-9cc4-47fe-9acf-ede449757d8a  802-3-ethernet  --
```

3.3.2 nmcli创建static连接

- 使用 `nmcli` 创建一个 `static` 的连接，配置IP、掩码、网关等
  - 1) 添加一个连接配置，并指定连接配置名称
  - 2) 将连接配置绑定物理网卡设备
  - 3) 配置网卡的类型，网卡是否开机启动
  - 4) 网卡使用什么模式配置IP地址(静态、dhcp)
  - 5) 配置网卡的IP地址、掩码、网关、DNS等等*

```
[root@linux-node1 ~]# nmcli connection add con-name eht0-static ifname eth0 \
type ethernet autoconnect yes \
ipv4.method manual \
ipv4.addresses 10.0.0.222/24 \
ipv4.gateway 10.0.0.254 \
ipv4.dns 233.5.5.5 \
+ipv4.dns 8.8.8.8

#激活eht1-static的连接
[root@linux-node1 ~]# nmcli connection up eht0-static
[root@xuliangwei ~]# nmcli connection show
NAME         UUID                                  TYPE            DEVICE
eht0-static  6fdebe6e-5ef0-4a05-8235-57e317fdada0  802-3-ethernet  eth0
```

3.3.2 nmcli修改网络配置信息

1.取消 `eht1-static` 连接开机自动激活网络

```
[root@xuliangwei ~]# nmcli connection modify eht0-static \
autoconnect no
```

2.修改 `eht1-static` 连接的 `dns` 配置

```
[root@xuliangwei ~]# nmcli connection modify eht0-static \
ipv4.dns 8.8.8.8
```

3.给连接再增加 `dns` 有些设定值通过 `+/-` 可以增加或则移除设定

```
[root@xuliangwei ~]# nmcli connection modify eht0-static \
+ipv4.dns 8.8.8.8
```

4.替换连接的静态IP和默认网关

```
[root@xuliangwei ~]# nmcli connection modify eht0-static \
ipv4.addresses 10.0.0.111/24 ipv4.gateway 10.0.0.254
```

5. `nmlci` 仅仅修改并保存了配置，要激活更改，需要重激活连接

```
[root@linux-node1 ~]# nmcli connection down eht1-static && \
nmcli connection up eht1-static
```

6.删除自建的 `connection`

```
[root@xuliangwei ~]# nmcli connection delete eht1-static
```

3.3.3 nmcli管理网卡配置文件

- 使用 `nmcli` 管理 `/etc/sysconfig/network-scripts/` 配置文件，其实就是自定义一个网卡的配置文件，然后加入至 `NetworkManager` 服务进行管理；
  - 1) 新增物理网卡
  - 2) 拷贝配置文件(可以和设备名称一致)
  - 3) 修改配置,UUID、连接名称、设备名称、IP地址
  - 4) 重新加载网络配置
  - 5) 启用连接,并检查

1.添加一个物理设备，进入 `/etc/sysconfig/network-script/` 目录拷贝一份网卡配置文件；

```
[root@xuliangwei network-scripts]# cp ifcfg-eth0-static ifcfg-eth1-static
```

2.修改网卡配置文件如下；

```
[root@xuliangwei network-scripts]# cat ifcfg-eth1-static
TYPE=Ethernet
BOOTPROTO=none
IPADDR=10.0.0.222
PREFIX=24
```

```
DEFROUTE=yes
NAME=eth1-static
DEVICE=eth2
ONBOOT=yes
```

3.重载 `connetction` 连接，让 `NetworkManager` 服务能够识别添加自定义网卡配置；

```
[root@xuliangwei network-scripts]# nmcli connection reload
NAME         UUID                                  TYPE      DEVICE
eth0         5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03  ethernet  eth0
eth1-static  8f105ed6-1361-8e14-51fd-dedb8ef3510a  ethernet  eth1
```

4. `eth1-static` 连接配置已经关联了 `eth1` 物理设备，如果希望修改 `IP` 地址，可以用如下两种方式；

```
# 方式一、nmcli modify方式修改然后重载配置
[root@xuliangwei ~]# nmcli modify eth1-static ipv4.address 10.0.0.233/24
[root@xuliangwei ~]# nmcli down eth1-static && nmcli up eth2-static

# 方式二、vim修改，先reload，然后重载
[root@xuliangwei network-scripts]# cat ifcfg-eth1-static
...
IPADDR=10.0.0.234
...
[root@xuliangwei ~]# nmcli connection reload
[root@xuliangwei ~]# nmcli connection down eth1-static && nmcli connection up e
th1-static
```

4.网关/路由概念

4.1 什么是路由

- 路由是指路由器从一个LAN接口上收到数据包，根据数据包的"目的地址"进行定向并转发到另一个WAN接口的过程。（跨网络访问的路径选择）
- 路由工作包含两个基本的动作：
  - 1、确定最佳路径
  - 2、通过网络传输信息
- 在路由的过程中，后者也称为（数据）交换。交换相对来说比较简单，而选择路径很复杂。

4.2 为什么需要路由

- 如果没有路由，就没有办法实现，不同地域的主机互联互通了；

4.3 如何配置路由

- linux系统配置路由使用 `route` 命令；可以使用 `route` 命令来显示和管理路由表；
- `route` 命令语法示例：
- `route [add|del] [-host|-net|default] [address[/mask]] [netmask] [gw] [dev]`
  - `[add|del]`：增加或删除路由条目；
  - `-host`： 添加或删除主机路由；
  - `-net`： 添加或删除网络路由；
  - `default`： 添加或删除默认路由；
  - `address`： 添加要去往的网段地址 由 `ip+netmask` 组成；
  - `gw`： 指定下一跳地址，要求下一跳地址必须是能到达的，且一般是和本网段直连的接口。
  - `dev`： 将添加路由与对应的接口关联，一般内核额会自动判断路由应该关联哪个接口；
- `route` 添加路由命令示例：

```
[root@xuliangwei ~]# route add -host 1.1.1.1/32 dev eth0
[root@xuliangwei ~]# route add -net 1.1.1.1/32 dev eth1
[root@xuliangwei ~]# route add -net 1.1.1.1/32 gw 1.1.1.2
[root@xuliangwei ~]# route add default gw 1.1.1.2
```

4.4 路由的分类

4.4.1 主机路由

- 主机路由作用：指明到某台主机具体应该怎么走； `Destination` 精确到某一台主机
- Linux上如何配置主机路由：

```
# 去往1.1.1.1主机，从eth0接口出
[root@xuliangwei ~]# route add -host 1.1.1.1/32 dev eth0

# 去往1.1.1.1主机，都交给10.0.0.2转发
[root@xuliangwei ~]# route add -host 1.1.1.1/32 gw 10.0.0.2
```

4.4.2 网络路由

- 网络路由作用：指明到某类网络怎么走；`Destination`精确到某一个网段的主机
- Linux上如何配置网络路由：

```
# 去往2.2.2.0/24网段，从eth0接口出
[root@xuliangwei ~]# route add -net 2.2.2.0/24 dev eth0

# 去往2.2.2.0/24网段，都交给10.0.0.2转发
[root@xuliangwei ~]# route add -net 2.2.2.0/24 gw 10.0.0.2
```

4.4.3 默认路由

- 默认路由：如果匹配不到主机路由、网络路由的，全部都走默认路由（网关）；
- Linux上如何配置网络路由：

```
[root@xuliangwei ~]# route add -net 0.0.0.0  gw 10.0.0.2
[root@xuliangwei ~]# route add default gw 10.0.0.2
```

4.4.4 永久路由

- 使用`route`命令添加的路由，属于临时添加；那如何添加永久路由条目；
- 在`/etc/sysconfig/network-scripts`目录下创建`route-ethx`的网卡名称，添加路由条目

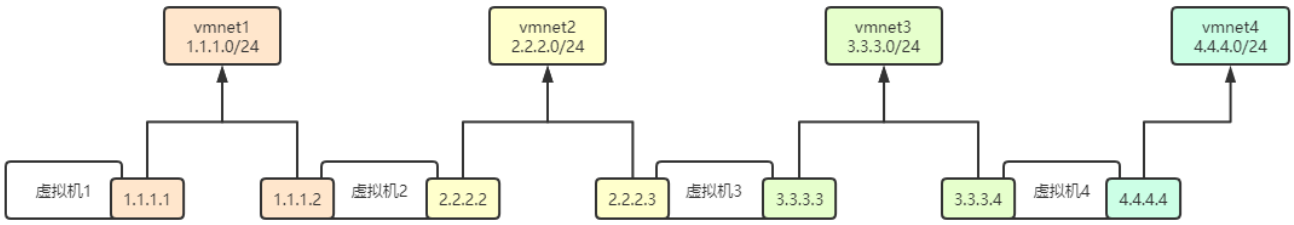
```
[root@dns-master ~]# cat   /etc/sysconfig/network-scripts/route-eth0
1.1.1.0/24 dev eth0
1.1.1.0/24 via 1.1.1.2

[root@dns-master ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
1.1.1.0         0.0.0.0         255.255.255.0   U     100    0        0 eth0
1.1.1.0         1.1.1.2         255.255.255.0   UG    100    0        0 eth0
```

4.5 路由项目案例

- 一台Linux主机能够被当成路由器用需要三大前提：
  - 1.至少有两块网卡分别连接两个不同的网段；
  - 2.开启路由转发功能`/proc/sys/net/ipv4/ip_forward`；
  - 3.在`linux`主机添加网关指向该服务器；

4.5.1 环境准备

- 实验环境



- 虚拟机网段配置

4.5.2 虚拟机1网卡配置

1.eth0网卡

```
[root@vm1 ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
TYPE=Ethernet
BOOTPROTO=static
DEFROUTE=yes
NAME=eth0
DEVICE=eth0
ONBOOT=yes
```

```
IPADDR=10.0.0.100
PREFIX=24
```

2.eth1网卡

```
[root@vm1 ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth1
TYPE=Ethernet
BOOTPROTO=static
DEFROUTE=yes
NAME=eth1
DEVICE=eth1
ONBOOT=yes
IPADDR=1.1.1.1
PREFIX=24
```

3.路由信息

```
[root@vm1 ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
1.1.1.0         0.0.0.0         255.255.255.0   U     101    0        0 eth1
10.0.0.0        0.0.0.0         255.255.255.0   U     100    0        0 eth0
```

4.5.3 虚拟机2网卡配置

1.eth0网卡配置

```
[root@vm2 ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
TYPE=Ethernet
BOOTPROTO=static
DEFROUTE=yes
NAME=eth0
DEVICE=eth0
ONBOOT=yes
IPADDR=1.1.1.2
PREFIX=24
```

2.eth1网卡配置

```
[root@vm2 ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth1
TYPE=Ethernet
BOOTPROTO=static
```

```
DEFROUTE=yes
NAME=eth1
DEVICE=eth1
ONBOOT=yes
IPADDR=2.2.2.2
PREFIX=24
```

3.路由信息

```
[root@vm2 ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
1.1.1.0         0.0.0.0         255.255.255.0   U     100    0        0 eth0
2.2.2.0         0.0.0.0         255.255.255.0   U     101    0        0 eth1
```

4.5.4 虚拟机3网卡配置

1.eth0网卡配置

```
[root@vm3 ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
TYPE=Ethernet
BOOTPROTO=static
DEFROUTE=yes
NAME=eth0
DEVICE=eth0
ONBOOT=yes
IPADDR=2.2.2.3
PREFIX=24
```

2.eth1网卡配置

```
[root@vm3 ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth1
TYPE=Ethernet
BOOTPROTO=static
DEFROUTE=yes
NAME=eth1
DEVICE=eth1
ONBOOT=yes
IPADDR=3.3.3.3
PREFIX=24
```

3.路由信息

```
[root@vm3 ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
2.2.2.0         0.0.0.0         255.255.255.0   U     100    0        0 eth0
3.3.3.0         0.0.0.0         255.255.255.0   U     101    0        0 eth1
```

4.5.5 虚拟机4网卡配置

1.eth0网卡配置

```
[root@vm4 ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
TYPE=Ethernet
BOOTPROTO=static
DEFROUTE=yes
NAME=eth0
DEVICE=eth0
ONBOOT=yes
IPADDR=3.3.3.4
PREFIX=24
```

2.eth1网卡配置

```
[root@vm4 ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth1
TYPE=Ethernet
BOOTPROTO=static
DEFROUTE=yes
NAME=eth1
DEVICE=eth1
ONBOOT=yes
IPADDR=4.4.4.4
PREFIX=24
```

3.路由信息

```
[root@vm4 ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
3.3.3.0         0.0.0.0         255.255.255.0   U     100    0        0 eth0
4.4.4.0         0.0.0.0         255.255.255.0   U     101    0        0 eth1
```

4.5.6 场景示例1

- 问：1.1.1.1地址能否与1.1.1.2 互通；
- 可以通，因为本机1.1.1.1与目标主机1.1.1.2 两台机器处于一个LAN中，并且两台机器上的路由表里具有Destination指向对方的网段路由条目

```
[root@vm1 ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        0.0.0.0         255.255.255.0   U     100    0        0 eth0
1.1.1.0         0.0.0.0         255.255.255.0   U     101    0        0 eth1

[root@vm2 ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
1.1.1.0         0.0.0.0         255.255.255.0   U     100    0        0 eth0
2.2.2.0         0.0.0.0         255.255.255.0   U     101    0        0 eth1
```

- 问：1.1.1.1地址能否与2.2.2.2地址互通
- 答：不能；因为数据包只能送到1.1.1.2，而无法送达2.2.2.2
- 所以需要添加一条去往2.2.2.0/24网段的路由，从eth1接口发出即可；

```
[root@vm1 ~]# route add -net 2.2.2.0/24 dev eth1
[root@vm1 ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
1.1.1.0         0.0.0.0         255.255.255.0   U     101    0        0 eth1
2.2.2.0         0.0.0.0         255.255.255.0   U     0      0        0 eth1
10.0.0.0        0.0.0.0         255.255.255.0   U     100    0        0 eth0

[root@vm1 ~]# ping 2.2.2.2
PING 2.2.2.2 (2.2.2.2) 56(84) bytes of data.
64 bytes from 2.2.2.2: icmp_seq=1 ttl=64 time=0.602 ms
64 bytes from 2.2.2.2: icmp_seq=2 ttl=64 time=1.60 ms
```

4.5.7 场景示例2

- 问：1.1.1.1地址能否与2.2.2.3地址互通；
- 答：不能；因为数据包只能送到vmnet2交换机，送不到vmnet3交换机
- 解决方案：将去往2.2.2.0/24网段的数据包交给1.1.1.2这台主机帮我们转发给2.2.2.3这台主机；

```
# vm1添加路由
[root@vm1 ~]# route add  -net 2.2.2.0/24 gw 1.1.1.2
[root@vm1 ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
```

```
1.1.1.0         0.0.0.0         255.255.255.0   U     101    0        0 eth1
2.2.2.0         1.1.1.2         255.255.255.0   UG    0      0        0 eth1

# vm2开启内核转发（由于vm2上有去往2.2.2.0/24网段路由，所以添加）
[root@vm2 ~]# echo "1" >  /proc/sys/net/ipv4/ip_forward
[root@vm2 ~]# sysctl -p

# vm1测试是否能ping通
[root@vm1 ~]# ping 2.2.2.3
PING 2.2.2.3 (2.2.2.3) 56(84) bytes of data.
64 bytes from 2.2.2.3: icmp_seq=1 ttl=63 time=0.690 ms
64 bytes from 2.2.2.3: icmp_seq=2 ttl=63 time=1.98 ms
```

4.5.8 场景示例3

- 问：1.1.1.1地址能否与3.3.3.3地址互通；
- 答：不能；因为数据包只能送到vmnet2交换机，送不到vmnet3交换机
- 解决方案：
  - 1.在vm1主机上将去往3.3.3.0/24网段的数据包交给1.1.1.2，由这台主机帮我们转发给3.3.3.3;
  - 2.在vm2上需要添加到3.3.3.0/24网段的路由，然后开启转发功能，否则数据包无法转发，会被丢弃；
  - 3.数据包到达vm3主机，但是无法送回来，所以还需要在vm3主机上添加去往1.1.1.0/24网段的数据包走2.2.2.2这台主机转发；

```
# vm1添加路由
[root@vm1 ~]# route add -net 3.3.3.0/24 gw 1.1.1.2
[root@vm1 ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
3.3.3.0         1.1.1.2         255.255.255.0   UG    0      0        0 eth1

# vm2开启转发，添加路由
[root@vm2 ~]# echo "1" >  /proc/sys/net/ipv4/ip_forward
[root@vm2 ~]# sysctl -p
[root@vm2 ~]# route add -net 3.3.3.0/24 eth1
[root@vm2 ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
3.3.3.0         0.0.0.0         255.255.255.0   U     0      0        0 eth1

# vm3添加回包路由
[root@vm3 ~]# route add -net 1.1.1.0/24 gw 2.2.2.2
[root@vm3 ~]# route -n
Kernel IP routing table
```

```
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
1.1.1.0         2.2.2.2         255.255.255.0   UG    0      0        0 eth0

#vm1主机测试
[root@vm1 ~]# ping 3.3.3.3
PING 3.3.3.3 (3.3.3.3) 56(84) bytes of data.
64 bytes from 3.3.3.3: icmp_seq=1 ttl=63 time=0.788 ms
64 bytes from 3.3.3.3: icmp_seq=2 ttl=63 time=0.440 ms
```

4.5.9 场景示例4

- 问：1.1.1.1地址能否与3.3.3.4地址互通；
- 答：不能；因为数据包只能从vmnet2交换机送往vmnet3交换机，无法达到vmnet4交换机；

- 解决方案：
  - 1.在vm1主机上将去往3.3.3.0/24网段的数据包交给1.1.1.2，由这台主机帮我们转发给3.3.3.4;
  - 2.在vm2上开启转发功能，然后添加到3.3.3.0/24网段的路由，由2.2.2.3帮我们转发给3.3.3.4;
  - 3.在vm3上开启转发功能；
  - 4.数据包到达vm4主机，但是无法送回来，所以还需要在vm4主机上添加去往1.1.1.0/24网段的数据包走3.3.3.3这台主机转发；

```
#vm1添加路由
[root@vm1 ~]# route add -net 3.3.3.0/24 gw 1.1.1.2
[root@vm1 ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Ifa
ce
3.3.3.0         1.1.1.2         255.255.255.0   UG    0      0        0 eth
1

#vm2开启转发，添加路由规则
[root@vm2 ~]# echo "1" >  /proc/sys/net/ipv4/ip_forward
[root@vm2 ~]# sysctl -p
[root@vm2 ~]# route  add -net 3.3.3.0/24 gw 2.2.2.3
[root@vm2 ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Ifa
ce
3.3.3.0         2.2.2.3         255.255.255.0   UG    0      0        0 eth
1
```

```
#vm3开启转发
[root@vm3 ~]# echo "1" >  /proc/sys/net/ipv4/ip_forward
[root@vm3 ~]# sysctl -p

#vm4添加回包路由
[root@centos7 ~]# route add -net 1.1.1.0/24 gw 3.3.3.3
[root@centos7 ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Ifa
ce
1.1.1.0         3.3.3.3         255.255.255.0   UG    0      0        0 eth
0

#vm1测试
[root@vm1 ~]# ping 3.3.3.4
PING 3.3.3.4 (3.3.3.4) 56(84) bytes of data.
64 bytes from 3.3.3.4: icmp_seq=80 ttl=62 time=0.506 ms
64 bytes from 3.3.3.4: icmp_seq=81 ttl=62 time=0.594 ms
```

4.6 路由条目优化

- 以虚拟机1为例，除了第一个路由条目外，其他的路由条目其实都需要由1.1.1.2来转发；
- 所以我们可以统一用一条路由规则；（配置默认路由）

1.删除vm1上无用的路由；

```
[root@vm1 ~]# route del -net 2.2.2.0/24
[root@vm1 ~]# route del -net 2.2.2.0/24     # 需要删除两次；因为添加了两次；
[root@vm1 ~]# route del -net 3.3.3.0/24
[root@vm1 ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
1.1.1.0         0.0.0.0         255.255.255.0   U     101    0        0 eth1
10.0.0.0        0.0.0.0         255.255.255.0   U     100    0        0 eth0
```

2.配置默认路由

```
[root@vm1 ~]# route add -net 0.0.0.0/0 gw 1.1.1.2
[root@vm1 ~]# #route add default gw 1.1.1.2
[root@vm1 ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
```

```
0.0.0.0         1.1.1.2         0.0.0.0         UG    0      0        0 eth1
```

3.测试效果

```
[root@vm1 ~]# ping 1.1.1.2
[root@vm1 ~]# ping 2.2.2.2
[root@vm1 ~]# ping 2.2.2.3
[root@vm1 ~]# ping 3.3.3.3
[root@vm1 ~]# ping 3.3.3.4
[root@vm1 ~]# ping 4.4.4.4
```

4.其他虚拟机按如上方式进行优化即可；

5.DHCP动态地址服务

1.DHCP基础知识

1.1 什么是DHCP

- 在大型企业网络环境中，会有大量的主机或设备需要获取IP地址等网络参数；如果采用手动配置，工作量大且不好管理，如果有用户擅自修改网络参数，还可能造成地址冲突；
- 使用动态主机配置协议 `DHCP（Dynamic Host Configuration Protocol）` 来分配IP地址等网络参数，可以减少管理员的工作量；避免手动配置网络参数造成地址冲突；

1.2 DHCP应用场景

- 只有有网络环境都会涉及到`DHCP`（WIFI、家庭、企业）；
- 云计算环境中，地址无需固定，则需要使用 `DHCP` 动态分配地址；
- 注意：在传统集群环境下，可能地址不能来回变动，所以任然会采用手动配置方式；

2.DHCP工作报文

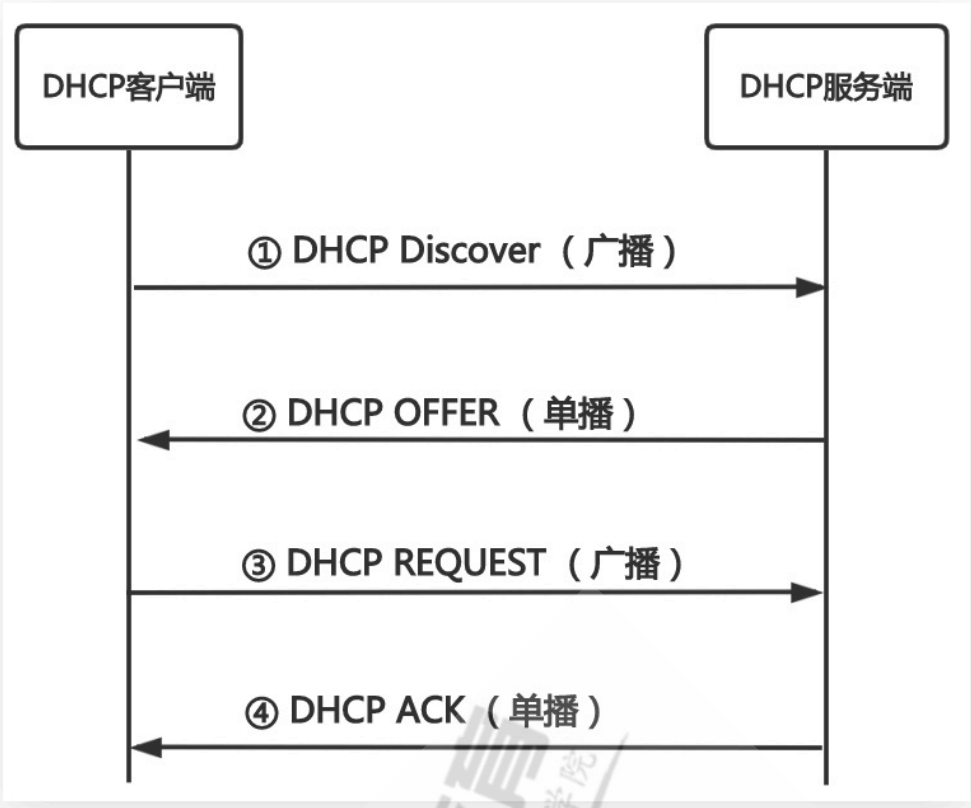
- 了解 `DHCP` 报文类型，有助于我们理解 `DHCP` 工作原理；

报文类型	含义
DHCP DISCOVER	客户端用来寻找DHCP服务器
	DHHCP服务器用来响应DHCP DISCOVER报文，此报文携带了各种

DHCP OFFER	配置信息；
DHCP REQUEST	客户端请求配置确认，或者续借租期；
DHCP ACK	服务器对 REQUEST 报文的确认响应；
DHCP NAK	服务器对 REQUEST 报文的拒绝响应；
DHCP RELEASE	客户端要释放地址时用来通知服务器；

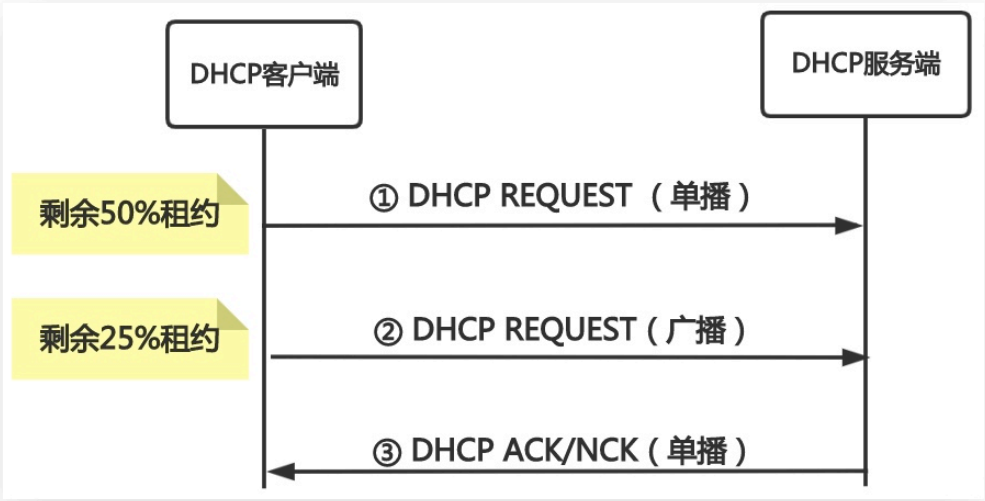
2.1 DHCP工作原理

- 第一步：客户端首先发送 `dhcp discover` 广播报文来发现局域网内的 `dhcp` 服务器；
- 第二步：服务器接收到客户端发送的报文后，根据自己地址池剩下的地址，分配给客户端一个地址；
- 第三步：客户端根据先收到的 `Offer` 报文来决定选用哪个服务器提供的 `DHCP` 的地址；
  - 之所以是广播，是因为可能会存在多个 `DHCP` 服务器，假设存在两个 `DHCP` 服务器；
  - 第一台服务器会响应 `ACK`，确定报文；
  - 第二台服务器会响应 `NAK`，拒绝报文；
- 第四步：服务器收到 `REQUEST` 报文以后，确认地址池中的这个地址没有被分配，如果没有被分配就回复 `ACK` 报文，如果被分配了，就会回复 `NAK` 报文，告诉客户端地址已经被分配了；



2.2 DHCP租期更新

- 当 `IP` 租约期限达到 `50%` 时， `DHCP` 客户端会请求更新 `IP` 地址租约；（默认租期为10分钟）
  - 第一步：客户端发送单播 `Request` 报文，请求服务器更新租期；
  - 第二步：服务器收到以后，如果该地址可用会响应 `ACK`，如不可用则响应 `NAK`
- 当 `IP` 租约期限达到 `75%` 时， `DHCP` 客户端会再次请求更新 `IP` 地址租约；
  - 客户端发送广播 `Request` 报文给 `DHCP` 服务器以便请求继续租用原来使用的地址；
    - 如果服务器确认客户端可以使用该地址，则回复一个 `ACK` 确认的消息；
    - 如果此地址已经无法再分配，则回复一个 `NCK` 否认的消息；
    - 当客户端收到的不是 `ACK`，而收到的是 `NCK`，则会获得`169.254.0.1~169.254.255.254` 之间的地址，然后每隔 `5min` 尝试更新租约；



2.4 DHCP地址释放

- 如果 `IP` 租约到期前都没有收到服务器响应，客户端停止此 `IP` 地址；
- 如果客户端不再使用分配的地址，也可以主动像服务器发送 `DHCP RELEASE` 报文，释放该IP地址；

3.DHCP服务配置

3.1 DHCP服务端

1.安装 `DHCP` 服务

```
[root@dhcp-server ~]# yum install dhcp -y
```

2.配置 `DHCP` 服务

```
[root@dhcp-server ~]# cat /etc/dhcp/dhcpd.conf
# 如果此DHCP服务器是本地服务器的正式DHCP服务器,应取消注释
```

```
#authoritative;

# 日志，默认存储/var/log/boot.log，也可自行调整
log-facility local7;

# 动态分配地址
subnet 172.16.1.0 netmask 255.255.255.0 {
  range 172.16.1.100 172.16.1.200;          # 动态分配地址池范围;
  option routers 172.16.1.254;              # 为客户端分配默认网关;
  option domain-name-servers 223.5.5.5;     # 为客户端分配DNS;
  option broadcast-address 172.16.1.255;    # 广播地址;
  default-lease-time 600;                   # 默认租期为10分钟;
  max-lease-time 7200;                      # 最大租期为2小时;
}
```

3.启动 `DHCP` 服务

```
[root@dhcp-server ~]# systemctl start dhcpd
[root@dhcp-server ~]# systemctl enable dhcpd
```

4.默认服务器如果是双网卡则提供全局 `DHCP` 服务；也可以通过如下方式指定特定接口提供 `DHCP` 服务；

```
[root@dhcp-server ~]# cat /etc/sysconfig/dhcpd
# DHCPDARGS="eth1"

[root@dhcp-server ~]# systemctl restart dhcpd
```

3.2 DHCP客户端

1.修改客户端网卡与服务端网卡在同一局域网，然后调整地址自动获取；

```
[root@dhcp-client ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth1
TYPE=Ethernet
BOOTPROTO=dhcp
DEFROUTE=yes
NAME=eth1
DEVICE=eth1
ONBOOT=yes
```

2.查看客户端获取的地址，以及路由、DNS是否正常；

```
# IP地址
[root@dhcp-client ~]# ifconfig eth1
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.16.1.139  netmask 255.255.255.0  broadcast 172.16.1.255

# 网关
[root@dhcp-client ~]# route -n
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         172.16.1.254    0.0.0.0         UG    101    0        0 eth1

# DNS
[root@dhcp-client ~]# cat /etc/resolv.conf
nameserver 223.5.5.5
```

3.检查服务端日志；

```
[root@dhcp-server ~]# tail -f /var/log/boot.log
dhcpd: DHCPDISCOVER from 00:0c:29:aa:8d:38 via eth1
dhcpd: DHCPOFFER on 172.16.1.139 to 00:0c:29:aa:8d:38 via eth1
dhcpd: DHCPREQUEST for 172.16.1.139 (172.16.1.6) from 00:0c:29:aa:8d:38 via eth
1
dhcpd: DHCPACK on 172.16.1.139 to 00:0c:29:aa:8d:38 via eth1
```

3.3 客户端重新获取地址

- 问题描述：
  - 局域网内一台客户端是通过 `DHCP` 自动获取 `IP` 地址，和另一台手动配置的静态 `IP` 产生冲突
- 解决方法：
  - 客户端释放当前 `IP` 地址，重新获取 `IP` 地址；

1.客户端重新获取IP地址；

```
[root@dhcp-server ~]# dhclient -r       # 释放
[root@dhcp-server ~]# dhclient          # 获取
```

2.检查服务器端日志；

```
[root@dhcp-server ~]# tail -f /var/log/boot.log
# DHCP RELEASE
dhcpd[3151]: DHCPRELEASE of 172.16.1.139 from 00:0c:29:aa:8d:38 via eth1 (found
)
```

```
# 重新获取
dhcpd[3151]: DHCPDISCOVER from 00:0c:29:aa:8d:38 via eth1
dhcpd[3151]: DHCPOFFER on 172.16.1.139 to 00:0c:29:aa:8d:38 via eth1
dhcpd[3151]: DHCPREQUEST for 172.16.1.139 (172.16.1.6) from 00:0c:29:aa:8d:38 v
ia eth1
dhcpd[3151]: DHCPACK on 172.16.1.139 to 00:0c:29:aa:8d:38 via eth1
```

3.4 为客户端分配固定地址

- 对于公司的打印机、文件服务器等，都需要手动进行IP地址分配，避免来回变化造成服务不可用；

1.配置服务端，追加如下配置；

```
[root@dhcp-server ~]# cat  /etc/dhcp/dhcpd.conf
# dhcpd.conf

# 为主机指定固定IP地址，这些地址不应列为可用于动态分配
host Server-A {                          # 指定节点名称；
  hardware ethernet 00:0c:29:aa:8d:38;   # 指定节点MAC地址；
  fixed-address 172.16.1.123;            # 指定分配的固定IP（必须存在地址池中）；
}
```

2.客户端重新获取 `IP` 地址；

```
[root@dhcp-server ~]# dhclient -r
[root@dhcp-server ~]# dhclient
[root@dhcp-client ~]# ifconfig eth1
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.16.1.123  netmask 255.255.255.0  broadcast 172.16.1.255
```

3.检查服务端日志；

```
[root@dhcp-server ~]# tail -f /var/log/boot.log
# RELEASE阶段
dhcpd: DHCPRELEASE of 172.16.1.139 from 00:0c:29:aa:8d:38 via eth1 (found)

# DISCOVER
dhcpd: DHCPDISCOVER from 00:0c:29:aa:8d:38 via eth1
dhcpd: DHCPOFFER on 172.16.1.123 to 00:0c:29:aa:8d:38 via eth1

# 静态分配阶段
```

```
dhcpd: Dynamic and static leases present for 172.16.1.123.
dhcpd: Remove host declaration Server-A or remove 172.16.1.123
dhcpd: from the dynamic address pool for 172.16.1.0/24

# 响应与确认阶段
dhcpd: DHCPREQUEST for 172.16.1.123 (172.16.1.6) from 00:0c:29:aa:8d:38 via eth
1
dhcpd: DHCPACK on 172.16.1.123 to 00:0c:29:aa:8d:38 via eth1
```

6.集群架构概述

本章课程内容大纲

1.架构基本术语
2.已知架构模型分析
3.未知架构模型分析
4.架构的访问流程-->用户视角
5.架构的维护流程--->运维视角（安全、监控、日志、自动化配置、自动化上线）
6.架构的运行环境--->运维视角
7.架构的产品开发流程-->开发视角
8.架构对应的工具与IP地址规划

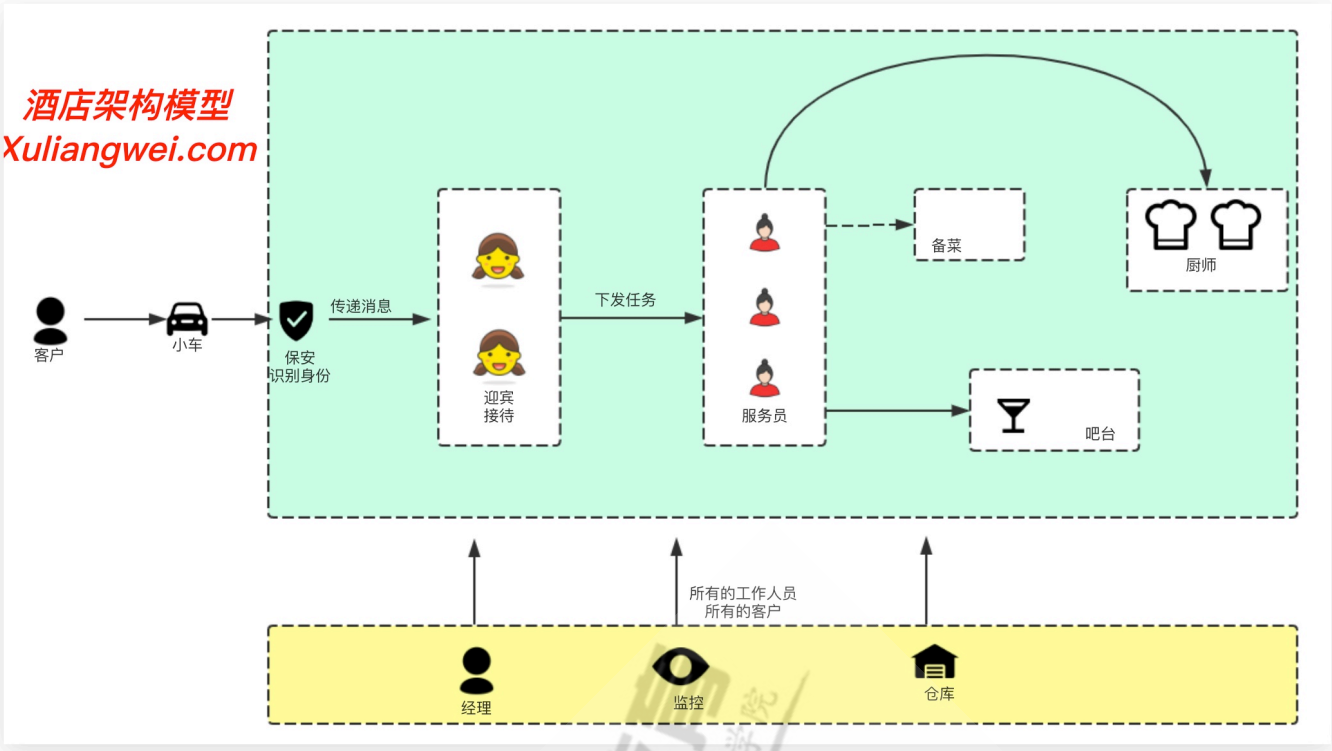
1.架构基础知识

1.什么是项目, 类似于手机的app，每一个app都可以算做一个项目。
2.什么是架构, 维护一个项目使用的一套服务器。(一套服务器可能会有很多角色。)
3.什么是集群, 为解决某个特定问题将多台计算机组合起来形成的单个系统。
4.什么是高可用, 当一台服务器不可用，另一台服务器自动接管，保证业务不down机,

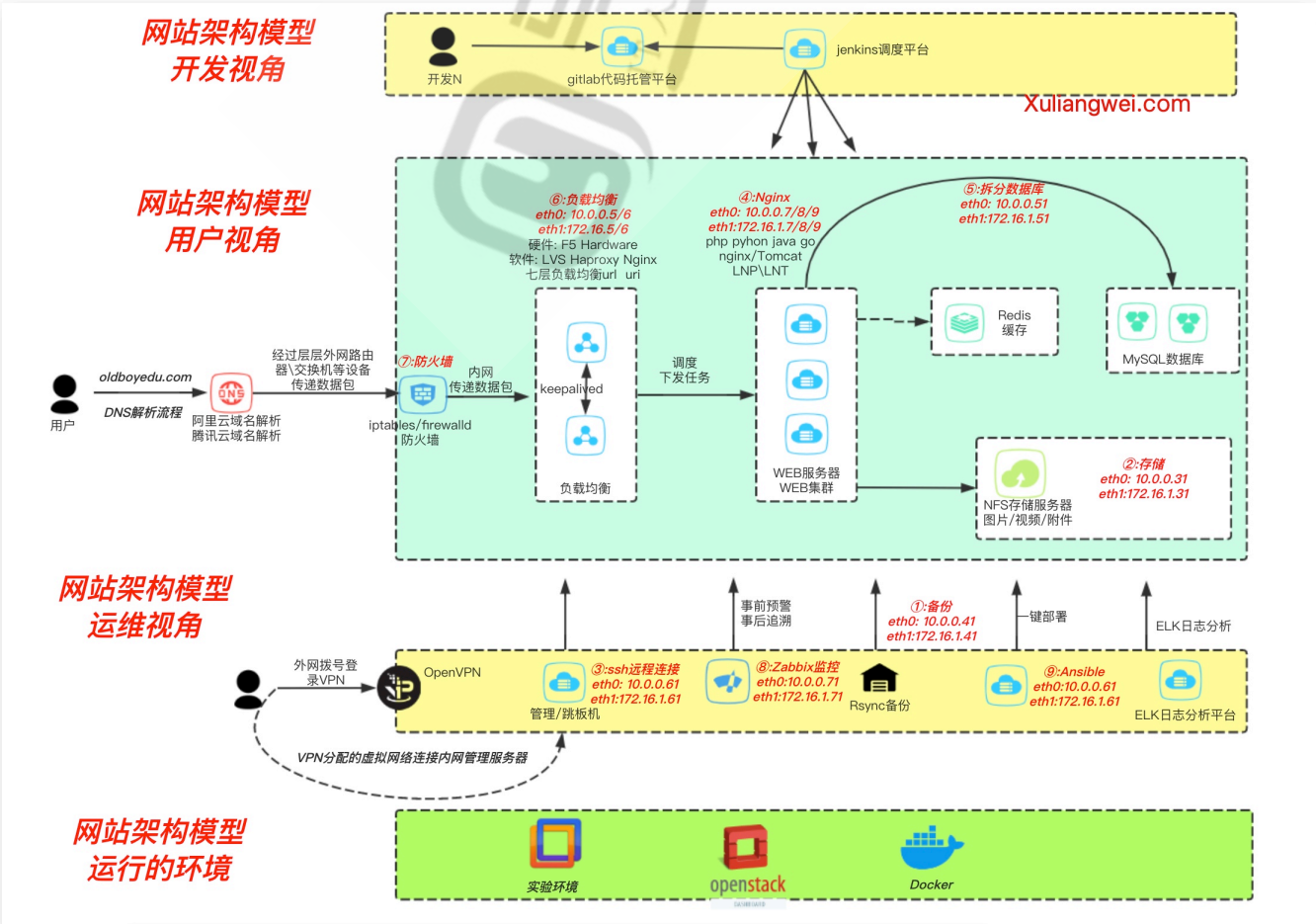
2.已知架构模型

开车 ---> 各种公路和高速路-->抵达目的地-->酒店
保安 ---> 验证身份
迎宾 ---> 接待工作
服务员 --> 满足客人需求
后厨厨师 --> 提供具体的菜品
吧台 --> 存放烟酒
备菜 --> 提前准备好-->快
仓库 --> 存放杂物，或者存放重要的文件或者手机
经理 --> 管理保安、服务员、厨师等等

监控 --> 用来监控 服务员、厨师等状态信息，用来事件的回溯



3.未知架构模型



总结：
一个项目涵盖了一套架构，一套架构又涵盖了不同的角色（高可用、负载均衡、web集群、缓存、数据库、存储、备份、监控、自动化、日志收集）
五层架构模型--> 负载均衡 web服务 存储服务 缓存服务 数据库服务（通过tcp连接）

4.整体环境规划

```
# IP
```

5.环境准备

```
1.安装全新Centos7系统，配置网卡为eth0及eth1命名模式
    1.第一块网卡为NAT模式[公网环境]，配置的网段为10.0.0.0网段
    2.第二块网卡为LAN模式[私网环境]，配置的网段为172.16.1.0网段
    3.优化安装好的Centos7虚拟机，安装常用软件、关闭防火墙等等
2.优化步骤
        #1.配置yum仓库
rm -f /etc/yum.repos.d/*
curl -o /etc/yum.repos.d/CentOS-Base.repo http://mirrors.aliyun.com/repo/Centos
-7.repo
curl -o /etc/yum.repos.d/epel.repo http://mirrors.aliyun.com/repo/epel-7.repo

        #2.安装基础软件包
yum install net-tools vim tree htop iftop \
iotop lrzsz sl wget unzip telnet nmap nc psmisc \
dos2unix bash-completion bash-completion-extras sysstat \
rsync nfs-utils httpd-tools -y

        #3.关闭防火墙firewalld
systemctl disable firewalld
systemctl stop firewalld

        #4.关闭selinux
sed -i '/^SELINUX=/c SELINUX=disabled' /etc/selinux/config

        #5.调整单个进程最大能打开文件的数量
echo '* - nofile 65535' >> /etc/security/limits.conf

3.基于优化后的虚拟机进行克隆
    1.连接克隆（需要依赖于母体）
    2.完整克隆（完完全全的复制一份，占用磁盘空间）

4.对新克隆后的主机进行如下操作：
    1.修改主机名  hostnamectl set-hostname backup
```

```
    2.修改IP地址  sed -i 's#200#41#g' /etc/sysconfig/network-scripts/ifcfg-eth[0
1]
    3.重启服务器

5.创建xshell标签->测试连接服务器是否成功
```

7.Chrony时间同步

1.时间同步基本概念

1.1 什么是时间同步

时间同步就是通过对本地时钟的某些操作，达到为分布式系统提供一个统一时间的过程。
在集中式系统中，由于所有进程都可以从系统唯一的全局时钟获取时间，因此系统内任何两个事件都有着明确的先后关系。而在分布式系统中，由于物理上的分散性，系统无法为彼此间相互独立的模块提供一个统一的全局时钟，而由各个进程各自维护它们的本地时钟。
由于这些本地时钟的计时速率、运行环境不一致性，因此所有本地时钟在某一时刻都被校准，一段时间后，这些本地时钟也会出现不一致。为了这些本地时钟再次达到相同的时间值，所以需要进行时间同步的操作；

1.2 为什么需要时间同步

在运维工作的场景当中，存在着众多主机协同完成不同的任务；比如 `LNMP` 架构，也是可以分别部署在三台不同的主机上；那么这三台主机在工作时，由于分别位于不同的主机之上，它们需要根据文件或者数据流所生成的时间，来决定我们响应给客户端的结果该如何进行展示；此时就需要同一网络中的主机时间一致；

但这个时间一致并不是说一定得是正确的，如果现在当前时间是下午2点，但是这三台主机的时间精确一致是昨天凌晨5点，这也没有什么问题；
但对于有些场景中时间不正确也不行，比如 `https` 应用；客户端与服务端通讯时，如果客户端时间是准确的，而服务端时间来自昨天，或者来自未来的响应，则会提示存在风险，而不予接受；

1.3 时间同步是如何完成

- 假设服务器启动起来后，发现时间慢了24小时，那么他如何将自己的时间调整正确呢
  - 如果是手表该如何校对时间呢？（波动表针，调整时间的正常逻辑）
  - 如果是 `date` 命令是如何校对时间呢？（直接跳跃时间，跳跃的过程中造成部分文件出现空白段）

1.3.1 NTP

- 逻辑：让时间校对像手表一样波动的快一点，而不是像 `date` 命令直接跳跃过去：其他服务器一分钟60s，而ntp一分钟30s，来实现时间的校对；
- 问题：为了赶上慢的24小时，可能需要花费非常长的时间来进行校对；

1.3.2 Chrony

- 逻辑：`Chrony` 是 `NTP` 的替代品，能更精确的时间和更快的速度同步时钟，传统 `ntp` 需要几小时，而 `chrony` 仅需要数秒种或数毫秒即可完成时间同步；（调整时间的速度就像波动表针的速度一样快）

2.Chrony时间服务

2.1 Chrony介绍

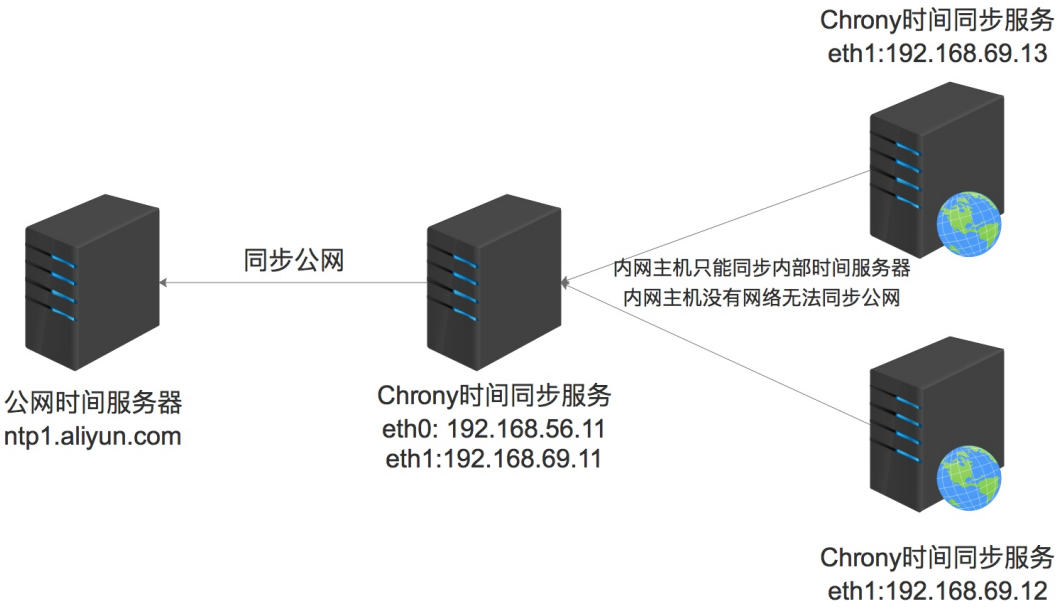
`chrony` 是一个 `ntp` 协议的实现程序，既可以当做服务端，也可以充当客户端；它专为间歇性互联网连接的系统而设计，当然也能良好应用于持久互联网连接的环境；

- `chrony` 有三个时间参考：硬件时钟、实时时钟以及手动同步；
- `chrony` 是 `NTP` 的替代品，能更精确的时间和更快的速度同步时钟；
- `chrony` 占用系统资源少，只有被唤起时才占用少部分 `CPU`， `chrony` 兼容 `ntpdate`；
- `chrony` 允许本地网络其他主机像本地进行时间同步；

2.2 Chrony优势

- 更快的同步，最大程度减少了时间和频率误差，对于并非全天运行的虚拟计算机非常有用
- 能够更好地响应时钟频率的快速变化，对于不稳定的虚拟机或导致时钟频率发生变化的技术有用
- 在初始同步后，它不会停止时钟，以防对需要系统时间保持单调的应用程序造成影响
- 在应对临时非对称延迟时（例如，在大规模下载造成链接饱和时）提供了更好的稳定性
- 无需对服务器进行定期轮询，因此具备间歇性网络连接的系统仍然可以快速同步时钟

2.3 Chrony时间同步



- 问题：所有服务器直接同步公网上的时间服务器不就可以了吗?为何需要自己搭建一台时间服务器呢?
  - 如果每台服务器都去同步公网时间服务器，且服务器较多
    - 造成延迟
    - 浪费带宽
  - 解决方法：搭建内网时间服务器，来同步公网时间,然后所有服务器来与这台服务器进行时间同步
    - 减小误差，提升同步速度
    - 减少网络带宽损耗
    - 统一规范管理时间

2.3.1 Chrony安装

```
[root@chrony ~]# yum install chrony -y
```

- 主配置文件：`/etc/chrony.conf`
- 客户端程序：`/usr/bin/chronyc`
- 服务端程序：`/usr/sbin/chronyd`

2.3.2 Chrony服务端

- 默认配置

```
[root@chrony ~]# cat /etc/chrony.conf

#使用同步的远程时钟源，理论上可以同步无限个
server 0.centos.pool.ntp.org iburst
server 1.centos.pool.ntp.org iburst
server 2.centos.pool.ntp.org iburst
```

```
server 3.centos.pool.ntp.org iburst

#根据实际时间计算出服务器增减时间的比率，然后记录到一个文件中，在系统重启后为系统做出最佳时间补偿调整
driftfile /var/lib/chrony/drift

#如果系统时钟的偏移量大于1秒，则允许系统时钟在前三次更新中步进
makestep 1.0 3

#启用实时时钟（RTC）的内核同步
rtcsync

#通过使用 hwtimestamp 指令启用硬件时间戳
#hwtimestamp *

#增加调整所需的可选择源的最小数量
#minsources 2

# 允许指定网络的主机同步时间，不指定就是允许所有，默认不开启。
allow 192.168.0.0/16

# 默认情况下本地服务器无法同步互联网时间时，可能会出现不精确，所以会拒绝提供授时服务；
# 开启此选项，则表示允许接受不精确时间，继续为客户端提供授时服务；
local stratum 10

#指定包含 NTP 身份验证密钥的文件
#keyfile /etc/chrony.keys

#指定日志文件
logdir /var/log/chrony

#选择日志文件要记录的信息
log measurements statistics tracking
```

1. Chrony 服务端配置，修改 `/etc/chrony.conf` 文件三处，设定外部时间服务器、允许内网同步此服务端、设置断网继续同步

```
[root@chrony ~]# vim /etc/chrony.conf
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
server ntp.aliyun.com iburst

# Allow NTP client access from local network.
allow 172.16.1.0/24

# Serve time even if not synchronized to a time source.
local stratum 10
```

2.重启 `Chrony` 服务

```
[root@chrony ~]# systemctl restart chronyd
```

2.3.3 Chrony客户端

1.客户端使用 `ntpdate` 或 `chronyc` 命令的方式进行手动同步

```
# ntpdate
[root@chrony ~]# yum install ntpdate -y
[root@chrony ~]# ntpdate 172.16.1.5

# chronyc
[root@chrony ~]# chronyc -a makestep
200 OK
```

2.客户端使用 `chrony` 守护进程方式进行时间自动化同步

```
[root@chrony ~]# yum install chrony -y
[root@chrony ~]# vim /etc/chrony.conf
# 指向至服务端
server 172.16.1.5 iburst

[root@chrony ~]# systemctl restart chronyd
```

3.查看时间同步是否正常

```
[root@chrony ~]# chronyc sources
210 Number of sources = 1
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^* 172.16.1.5                    3   6    77    24   -926us[-2077us] +/-   19ms
[root@chrony ~]# chronyc sources -v
```