第3章 网络层（Network Layer）

3.1 介绍

终于可以介绍 TCP/IP 中的核心组件 IP 了，IP 协议被称为 TCP/IP 中的骡马协议，能看到的 TCP、UDP、ICMP、IGMP、OSPF、BGP、RIP、IPSec、NAT 这些名词都在 IP 上面跑着。IP 相当于一种不挂号的邮件服务，按照术语来说叫 *unreliable*(不可靠传输), *connectionless*(无连接)：

1. unreliable 指的是通过 IP 发送出去的数据，并不保证对方一定可以准确无误地收到，有可能中间丢掉了（丢弃的原因很多，比如链路过载了、转发设备缓存满了之类的），也有可能某部分数据丢失了或被篡改了，如果想要靠谱服务，可以通过更高层次如 TCP、OSPF、IPSec 来实现，目前 TCP/IP 网络得益于链路层的极大进步，丢包是一个频度比较低的事情；
2. connectionless 指的是 IP 在收发数据前并不需要专门协商个会话打个招呼什么的，也不需要维护，发送一方想发就发，接收方也是看到目的地址是监听的就收，很自由，要想高级一点的连接也要靠 TCP 之类的传输层协议实现。

IP 的作用就是在横跨多条链路的网络上实现端到端传输数据，简单、直接、有效、可靠（协议栈可靠，并不是可靠传输）是协议设计者主要考虑的问题。

3.2 头部封装-Header

链路层的数据结构我们称为“帧”frame，比如以太网有 802.3 封装和 Ethernet 封装 2 种，在 TCP/IP 协议栈中，IP 是链路层的上一层，也有自己的数据结构，被称为“*包*”*Packet*，类似于以太网，也有自己的封装结构：

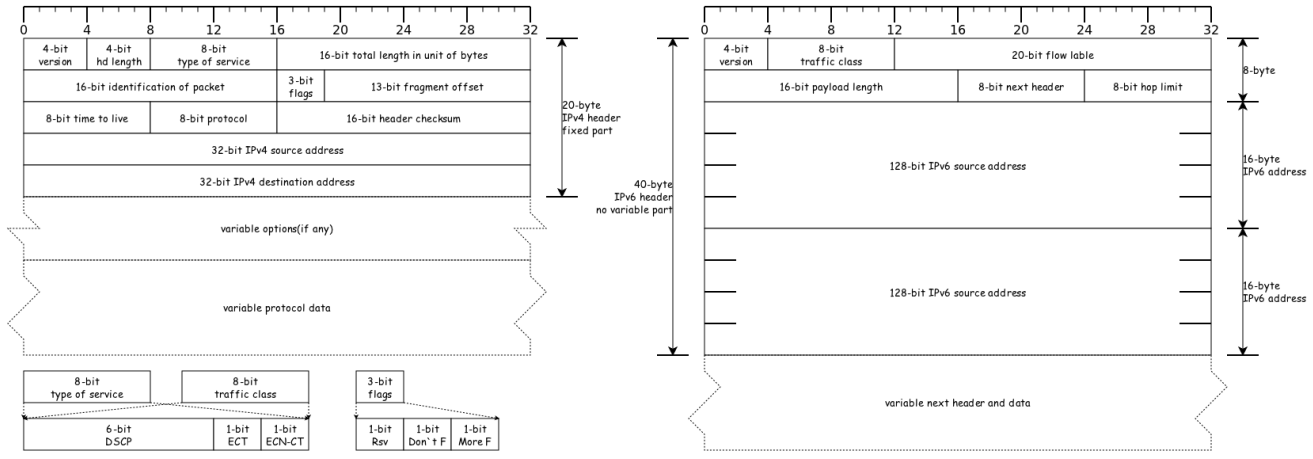


图 3-1 IPv4 和 IPv6 封装结构

IP 目前有 2 个版本：

1. IPv4（IP version 4，互联网协议第四版），是目前的主力，封装如图 3-1 的左半部分，20 字节固定部分，固定部分共 12 个字段:

1) version：版本，4-bit 长，用来表明“IP”（网络互联协议）的版本，老外的设计是会考虑他们当时设计可能不够完善，需要不断改进，所以留了这么一个版本号，目前最为网民所熟知的 IP 地址就是 IPv4，也就是 Version 字段值为 4，可以想象早期版本 1～3 都已经变成古董了；

2) header length：头部长度，4-bit 长，用来表明 IPv4 头部的长度，单位是 4-byte，也就是 32-bit，所以为了方便人们读懂，图 3-1 每行都是 32-bit 长；和以太网不一样，IPv4 头部长度是变化的，所以要有这个字段，IPv4 这个字段通常值是 5，这也是最小值，通常的解读是 IPv4 头部固定部分长度是 20 字节，在图 3-1 右半部分实线部分；

3) type of service：服务类型，8-bit 长，类似于 802.1p 和 802.11i 中的优先级作用，是网络层 QoS 的基础，这个字段是个复合字段，结构如图 3-1 左下角所示：

   (1) DSCP（*Differentiated Services Code Point 差分服务标识码*），6-bit 长，用于标识数据的优先级；

   (2) *ECT*（*Explicit Congestion Notification Capable Transport 显式拥塞通知传输层能力*），1-bit，用于通告支持 ECT，ECT 是一项新兴的端到端 QoS 拥塞控制手段；

   (3) ECN-CE（*ECN Congestion Experienced 显示拥塞通知-发生拥塞*），1-bit 长，当网络中发生了拥塞，这个位就会被置 1；

   (4) 在这里介绍的 type of service 字段是较新版本，比较老的版本是 3-bit 长的优先级，4-bit TOS，1-bit 保留，一切向前看，新版本覆盖老版本，就不多介绍；

4) total length：总长度，16-bit 长，整个 IP 包（包括头和载荷）的长度，单位 byte（header length 字段单位是 4-byte），所以计算 payload 长度也很容易，
   $payload\ length = totoal\ length - Header\ length$ ，最常见的 total length 是 1500，正好是以太网 MTU 长度，这两个值通常相生相伴；

5) identifier：数据包标识，16-bit，可以从图 3-1 中发现这个字段在第二行，这一行都是给 IPv4 数据包分片使用的，数据包分片后就还需要接收方，接收方如何在众多数据包中把合适的分片挑出来重组呢？就是靠数据包标识，相同数据包的分片都拥有相同的标识；关于数据包的分片会结合链路层 MTU 详细介绍；

6) flags：标记，3-bit 长，可以从图 3-1 左下角看到它由 3 个 1-bit 字段组成：

   (1) RSV，reserved，保留位;

   (2) Don't F，do not fragmented，不许分片位，当一个数据包因为超出 MTU 原因需要被分片时，如果看到这个位置 1，那么数据包就不允许被分片，只能被丢弃；

   (3) More F，more fragmentation，更多分片，当一个数据包被分片后，除了最后一个分片没

此位置 0，其余分片都置 1，用于像接收方进行判断；

7) fragment offset：分片偏移量，数据包是被顺序分片的，重组也只能是按照顺序重组，偏移量就是用于许多分片排序使用的；

8) time to live：8-bit 长，更为熟知的是它的简称 TTL，翻译回来叫生存时间，为了避免数据包在网络中无限地传递下去，每经过一台网络层转发设备，TTL 就会减 1，当 TTL 变成 0 后，这个数据包就只能被丢弃，不能再被网络层转发，一般而言网络上是看不到 TTL = 0 的数据包的，因为正常的网络层设备不会犯这种 bug，如果真的看到了，有可能是在测试环境，或者遇到了黑客；

9) protocol：协议，8-bit 长，类似于以太网中的 type 字段，若 protocol = 1，则封装 ICMP，6 则封装 TCP，17 封装 UDP 等等，可以 Google 一下；

10) header checksum：头校验，8-bit 长，对 IPv4 头部（包括固定部分和选项）进行 CRC 校验，防止传输过程中头部损坏；

11) source address：源地址，32-bit 长，正好一行，在端到端通信中用于标识源端位置，就相当于信件通信中寄件人地址的作用；

12) destination address：目的地址，也是 32-bit 长，也正好一行，当然是标识目标端位置，相当于信件通信中收件人地址，网络层转发就是根据目的地址转发；

13) options：选项，变化长度，所以使用虚线勾勒，既然是选项，那就是一般都用不着的；

14) data：也叫 payload，包含协议头和协议数据，变化长度，如果 IPv4 头相当于信封，data 就是信封里的信件；

2. IPv6（顾名思义就是互联网协议第六版），是将来要推广的版本，封装如图 3-1 的右半部分，固定 40 字节的 Header，共 8 个字段，长度比 IPv4 多了 1 倍，但字段少了 4 个，整体看起来还是蛮简洁的：

1) version：也是版本，4-bit 长，兼容 IPv4 的协议号，该值为 6 就表示 IPv6，同样 IPv5 也是一个不成气候的协议，被漠视了；

2) traffic class：流分类，8-bit 长，可以从图 3-1 左下角看出，和 IPv4 的 type of service 是同样作用，换了个更流行的名字而已；

3) traffic label：流标签，20-bit 长，这是一个**全新的字段**，用于标识一条流（流的英文名有许多版本 traffic flow, traffic stream 或者就是 traffic），在 IPv4 中，标识一条流通常要 5 元组（源、目的 IP、协议、源、目的 port），也就是说需要传输层信息，在 IPv6 中，理论上只需要头中的 3 元组（源、目的 IP、流标签）就可以识别一条流，理论上来说要比 IPv4 识别流效率要

高很多，但杯具的是流标签现在怎么使用，既无定论，也无暂行条例，干脆都被置 0，识别 IPv6 流还是要靠老套的 5 元组，每每想到这里，我就无比心痛，这个字段可以还可以做成 IP 无关的标识，大规模应用之间的隔离就可以不用再借助于变化莫测的 IP 地址了，非常方便；

4) payload length：载荷长度，16-bit 长，源于 total length，IPv4 中需要公式计算载荷长度，IPv6 中不需要计算是因为 IPv6 头固定就是 40 字节长，变长的只有 payload，所以只需要这个字段；

5) next-header：下个头类型，8-bit 长，和 IPv4 的 protocol 完全一致，换了个新颖点的名字，不过许多数值都发生了变化，比如 ICMPv6 的值不再是 1，而是 58，细致点也可以 Google 一下；

6) hop-limit：跳限，8-bit 长，和 IPv4 的 TTL 完全一致，IPv4 的 TTL 在设计之初是用来计时的，后来变成计算网络层转发次数，每转发一次就叫 *hop*“*一跳*”，IPv6 干脆就不挂这个羊头，直接叫跳限，这个改名是比较合理的；

7) source address：源地址，128-bit 长，作用和 IPv4 的源地址一致，只不过长一点，占了 4 行；

8) destination address：目的地址，128-bit 长，和 IPv4 的目的地址也一致；

9) payload：和 data 是一个概念，变化长度。

有一些 IPv4 头中的字段被 IPv6 干掉了，大家也肯定比较好奇：

1. header length：因为 IPv6 头不再分固定部分和选项，而是固定 40 字节头，这个字段删除没有疑问；

2. identifier：IPv6 中的分片工作专门由 *fragmentation extended header*（*分片扩展头*）来处理，不需要放在 IPv6 头让所有网络层转发设备都参与分片或者重组，而是有 IPv6 的 source 和 destination 来处理，这个设计也比较合理，理论上能够降低网络层设备成本并提高转发效率，不过现在分片与重组功能已经很成熟，网络设备重组分片也被认为是易受到攻击的安全隐患，这个功能的改变个人认为还是比较有益的；

3. flags：分片功能一部分，搬移到分片扩展头；

4. fragment offset：同上；

5. header checksum：IPv4 已经有 30 多年历史了，30 年前的网络根本无法和现在同日而语，链路层状况差，信号误损率还是比较高的，所以有一个 header checksum 用来识别头部是否受损，到了 IPv6 时代，固定长度的头，再加上链路层传输已经相当稳健，无损率几乎不可能发生，即使发生也可以通过传输层或者应用层手段加以发现和重传，这个字段也被无情地抛弃了；

6. Options：IPv4 的选项在 IPv6 中并没有被抛弃，只是和分片功能一样，被排除出固定头部，通通放

在扩展头里了，以保证 IPv6 头的简洁高效。

IPv6 干掉了 5 个字段，新增 1 个流标签字段，所以总共比 IPv4 减少了 4 个字段，网络进化到 IPv6，协议分工变得更加清晰化，网络转发设备只负责 QoS 处理和转发，其余功能都交给 source 和 destination，通常是扩展头、传输层、应用层处理。

3.3 地址

我很喜欢把 IP 头、以太网头比作信封，其实老外也经常用 Envelope 来描述这些头部，那么我们都知道寄信的时候信封上除了贴邮票外什么最重要？肯定是收件人地址和收件人。IP 头这个信封里最重要的信息也差球不多：

1. destination address：目的地址，就类似于收件人信息，用于在 IP 网络中标识一个位置；

2. source address：源地址，类似于寄件人信息，也是标识一个位置，信封可以写匿名信，因为这种信件是不希望回信的，但 IP 世界里，通信通常是双向的，有源地址，才能回信，源地址必须要填写，填写得对不对有其他方式来处理。

既然地址那么重要，那么我们就来看看地址是怎么构成并表示的。

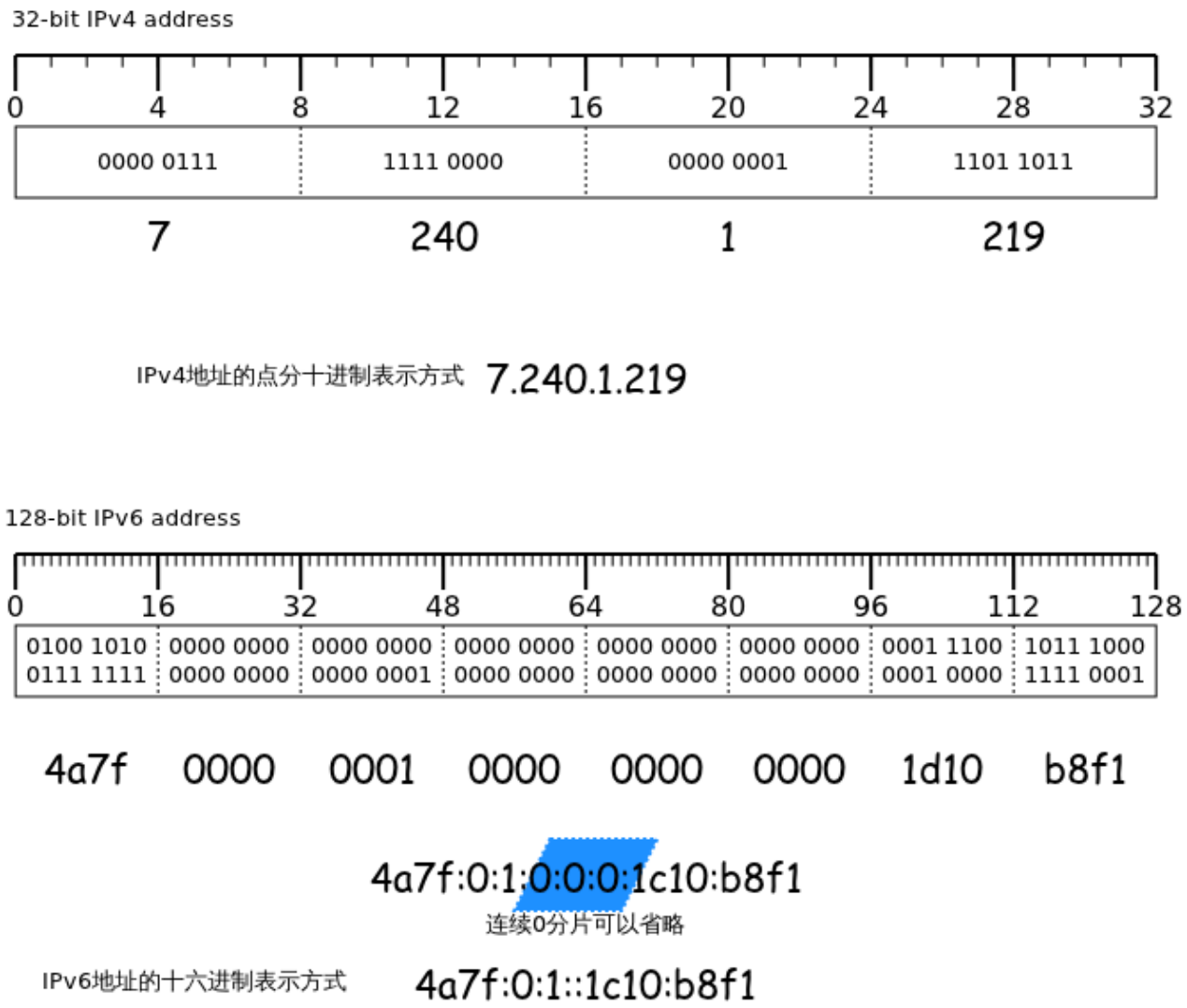


图 3-2 IP 地址表示方式

从图 3-2 可以看到，在链路上传输的都是二进制原生态格式，这种东西非常不利于人们的记忆和交流，所以，必须要有种对人可读的表现格式：

- IPv4 的方式被称为“*点分十进制表示法*”（*dotted-decimal notation*），把 32 位地址三刀下去，切成 4 片，每片 8 位，所以每片的取值范围就是 0～255，每片之间使用点号“.”隔开，如图 3-2 中的“7.240.1.219”；

- IPv6 的方式被称为“*十六进制表示法*”（*hexadecimal notation*），十六进制之间通常使用冒号“:”分割，因为 IPv6 地址太长了，所以按照以前 3 刀下去分 4 片还是不太方便，所以就 7 刀下去分 8 片，每片 16 位，使用十六进制表示每片，所以每片的取值范围是 0000～ffff，如图 3-2 中的 4a7f:0:1:0:0:0:1c10:b8f1，由于 IPv6 地址太长，有的分片可能是全 0，甚至有可能好几个分片都是 0，那么连续的 0 分片就可以用双冒号“::”，如上面中间连续 3 个 0 分片就被缩写为 4a7f:0:1::1c10:b8f1，双冒号并没有长度限制，比如一个地址“ff02::2”，这就表示里面有 6 个分片都是 0，那么怎么知道双冒号表示多少个 0 分片呢？IPv6 地址中只允许出现一次双冒号，假设其

余分片数量是 n，那么双冒号代表的 0 分片就是 8-n，看起来很恼人，但接触多了，就习惯了，地址太长，不得不用一点新鲜的压缩方法啊。在 RFC 5952 中专门对 IPv6 的表示方法做了统一，在这里推广一下：

- 地址中的字母只能是小写；
- 如果不是连续 0 分片，不得使用双冒号，比如 1:2:3:4:5:6:0:8 就不得简写成 1:2:3:4:5:6::8；
- 双冒号必须最大程度简化地址，比如 1:2:3:4:0:0:0:8，必须简写为 1:2:3:4::8，不能简写为 1:2:3:4:0::8；
- 当存在两个或以上长度不一连续 0 分片时，使用双冒号省略最长的连续 0 分片，比如 1:0:0:0:5:0:0:8，只能使用 1::5:0:0:8，不能使用 1:0:0:0:5::8；
- 当所有连续 0 分片长度一致时，只简化最前面的连续分片，比如 1:0:0:4:0:0:7:8，只能被简化成 1::4:0:0:7:8，不能是 1:0:0:4::7:8；
- 如果地址结合 TCP/UDP 端口号是只能采取[ipv6-address]:port 的方式，比如 TCP [1::8]:80，表示 1::8 的 TCP 80 端口，也就是熟知的 HTTP，UDP[1::8]:69，表示 1::8 的 UDP 69 端口，也就是熟知的 TFTP。

介绍完地址的表示后，我们再来探讨地址的分类。分类是为了更加有效地利用和管理地址，IPv4 和 IPv6 两套地址分类手段是不同的，先来看看当前的 IPv4 的分类：

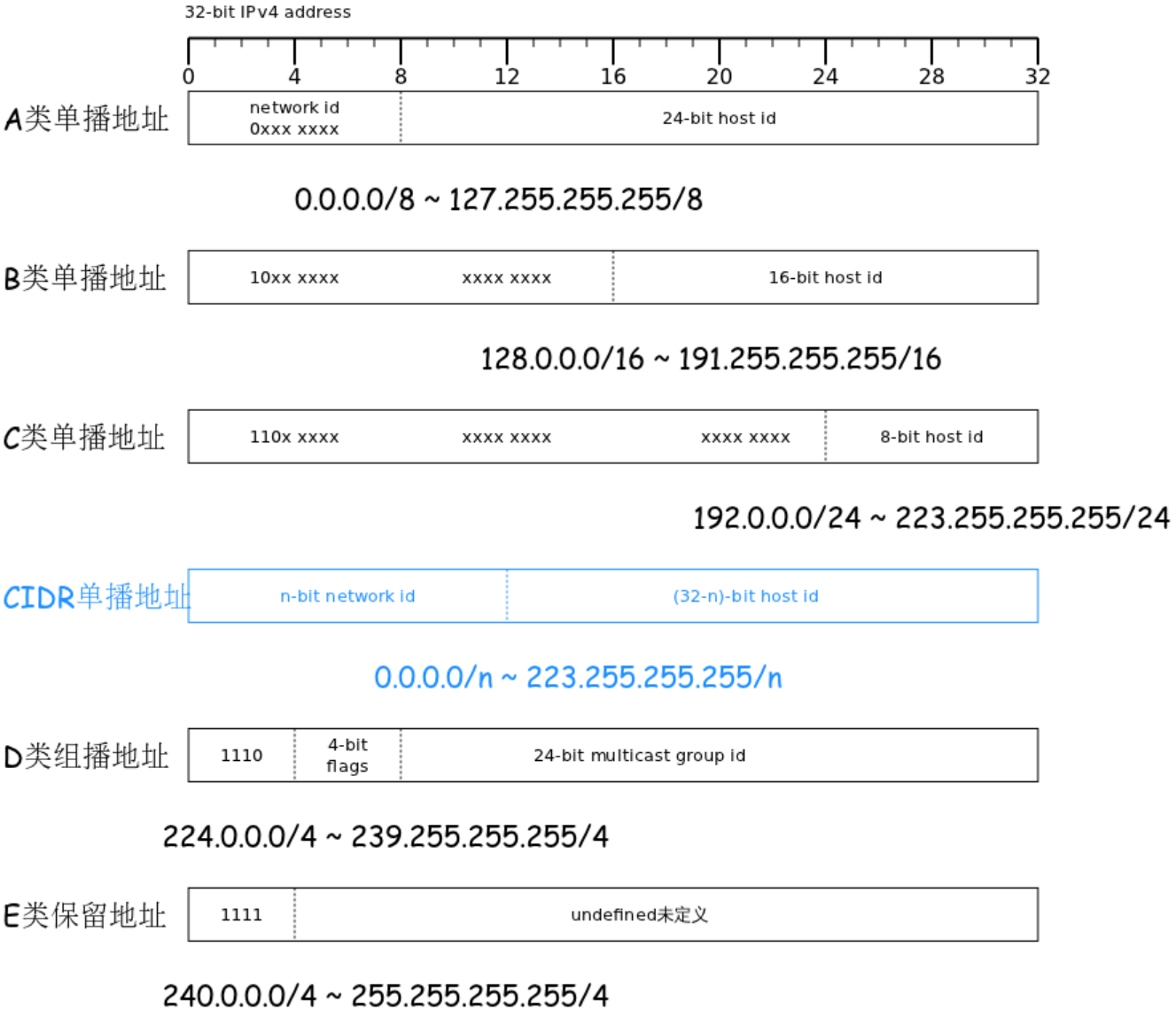


图 3-3 IPv4 地址分类

这是最原始的 IPv4 地址分类，ABC 三类地址被称为 *unicast 单播*地址（单播地址中的一些特殊地址又可以是 *broadcast 广播*地址），D 类地址被称为 *multicast 组播*地址，E 类地址被保留了。平时网民说的 IP 地址通常是指 ABC 类，如果是搞网络基础设施的，那么 D 类地址也是了解的。当今 IPv4 地址已经分配完毕了，个人觉得 E 类地址和剩下的虽然不多，也可以拿出来分了。IPv4 是人类历史上许多第一次，刚开始只是一个实验性质的网络，设计者没能预测到 IP 上竟然能产生
Google、amazon、QQ、Facebook、eBay、Twitter 等丰富多彩的应用，既然是人类的一次尝试，那么实践是检验这些尝试的标准，经过 30 多年的检验，我们得出了一些结论：

- 地址分为*单播地址*（*unicast address*）和*组播地址*（*multicast address*），每个单播地址由*网络号 network id* 和*主机号 host id* 组成，网络号由 IANA 颁发给各个申请者（IANA 主页可以找到地址分配情况），在初始的设计中网络号用于标识一条链路（在第一章中提到，一条链路就是一个网络，网络层协议解决的是 inter networking，也就是网络间通信），主机号由申请者在链路内部自行分配给不同主机，根据 network id 和 host id 不同长度（网络号长度可以使用“/n”表示，如 1.1.1.1/24，也可以使用*掩码 mask* 表示，如 1.1.1.1/255.255.255.0，也就是说地址和掩码转换

为二进制后网络号对应的掩码位是 1，主机号则是 0）可以分为：

- A 类地址：网络号固定 8-bit 长，网络号第 1 位为 0，所以 A 类网络号只有 128 个，主机号就是 24-bit 长，可以支持一个拥有 2^24-2 主机（主机号为全 0 和全 1 都是特殊地址，不能分配给主机）的极大链路（说实话，这种超大型链路应该就从没诞生过），只有 128 个 A 类网络在 IPv4 诞生的初期就被一些大组织或者有名气的单位诸如 IBM、Ford Motor、MIT 之类的瓜分干净；

- B 类地址：网络号固定 16-bit 长，网络号前 2 位为 10，所以 B 类网络号有 2^14 也就是个 16384 个，主机号 16-bit 长，可以支持一个 2^16-2 主机的超大型链路（这个规模的链路也没有诞生过），16384 个网络号分配延续到了 21 世纪才分完；

- C 类地址，网络号固定 24-bit 长，前 3 位 110，所以 C 类网络号有 2^21 个，主机号 8-bit 长，可以支持 2^8-2 主机也就是 254 个主机的大型链路（这个规模比较常见），C 类网络号比较多，所以延续到 2010 年 Q4 才分配完毕，当时许多人都发出了 IPv4 game is over，IPv6 is coming 类似的声音，其实这些 IPv4 地址只是都分配个运营商了，运营商逐步往外分地址，并不是说网民没有地址可用了；

- *CIDR*（*Classless Inter-Domain Routing 无类域间路由*）类地址，名字虽然蛋疼，但目的却很单纯（所以图 3-3 中使用蓝色 CIDR），不再区分 ABC 类地址，所有地址的网络号、主机号都是人为指定长度的（如图 3-3，/n 表示不固定长度网络号），比如一条点到点链路上只有 2 台主机，在没有 CIDR 前，最节约也是使用 C 类地址，C 类地址可以支持 254 主机的链路，用在点到点链路上就浪费了 252 个地址，这些地址还不能用在其它链路上，有了 CIDR 后，就可以分配/30 的网络了，主机位 2 个，正好可以支持 2^2-2=2 个主机；还有一种情况是，有时候几个连续同类地址还可以通过 CIDR 的方式进行聚合，比如 4.0.0.0/8、5.0.0.0/8、6.0.0.0/8、7.0.0.0/8 可以聚合为 4.0.0.0/6 来表示，往细了分的 CIDR 叫 *Subnet*（*子网*），往大了聚合叫 *Supernet*（超网），**目前的互联网基本上都是按照 CIDR 方法使用地址，纯粹的 A、B、C 类地址在使用上因为不够灵活、不够绿色节能已经被废弃**；ABC 类的分法就是被实践证明很糟糕的一个东西，CIDR 则是被证明很棒的思路，各种操作系统都是支持 CIDR 的，虽然默认采用 ABC 分类。

- 组播地址，这是一类特殊的地址，前 4 位固定 1110，后跟 4 位 flags，还剩 24 位 *group id 组号*（组号也是由 IANA 负责分配），单播地址中主机号标识唯一某个链路中的唯一主机，而组播地址的组 id 则标识互联网上多个主机，**组播地址在 IP 头中只能作为目的地址出现，如果作为源地址出现，该 IP 包可以被视为非法包，人人见而得以诛之**，组播的使用如下：

  - 不同的主机都通过一些协议注册到某个组播地址上，也就是说某个组 id 上；

  - 网络上的网络设备通过一些协议维护这个组信息，即这个组有哪些成员；

  - 当某个组成员要往组发送数据时，源地址是组成员，目的地址是组播地址，数据包只需要发送一份；

  - 合理的网络设备会自动將这一份数据包复制成多份发送给其余组成员，以节约网络带宽。

- E 类地址，是单播地址，前 4 位 1111，后 28 位通通保留，IANA 早在 1981 年 9 月就将他们通通 RESERVED 了，理由是 Future use，但我搞不明白的是，现在当年的 ABC 类地址已经分配完毕了，目前就已经是 1981 年的 Future 了，为什么还 RESERVED 呢？当然我解答不了这个答案，这个可以留给大家围观或者自告奋勇地去 IANA 上访。

看完 IPv4 的，再看看 IPv6 的地址分类，相比来说，要简洁很多：

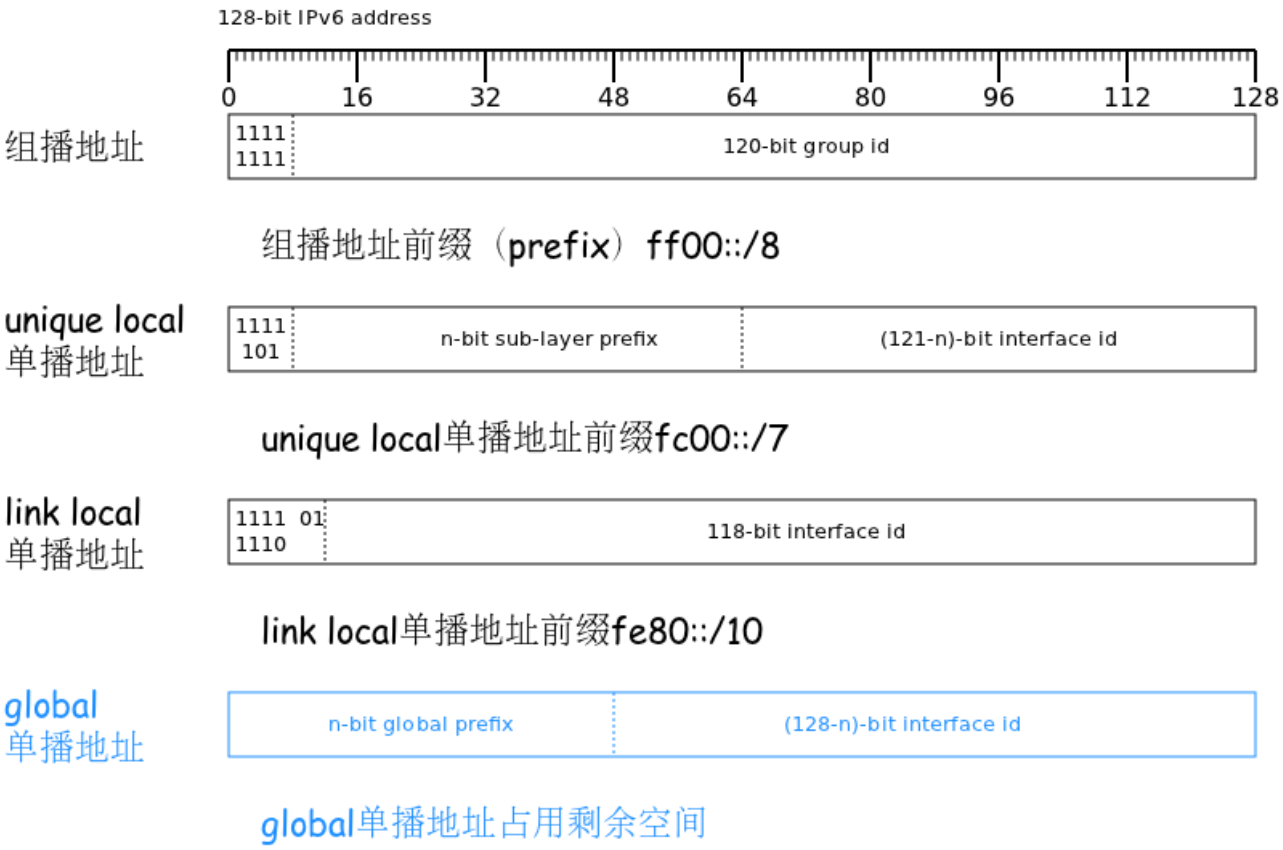


图 3-4 IPv6 地址分类

IPv6 的地址分类会简单很多，把地址分为 *prefix 前缀*和 suffix *后缀*两部分，类似于 IPv4 地址中的 CIDR，直接使用/n 来划分前缀和后缀长度，IPv6 地址可以被简单地划分为组播和单播两部分，后缀名在组播和单播称谓也不一样：

- 组播地址：前 8-bit 都是 1，所以组播地址的公用前缀是 ff00::/8，剩下的 120 位后缀被称为 group-id，也就是组号，这 120 位中有的位已经被 IETF 定义，还有很深的细节待挖掘；

- 单播地址（IPv6 中还会提到 *anycast 任播地址*，任播地址和单播地址共享地址空间，不做单独介绍）：简单来说，除了组播地址外全部都是单播地址，它的后缀被称为 *interface id 接口号*，单播地址可以分为 3 种 *scope 范围*：

  - *link local*：固定前缀 fe80::/10，前 10-bit 如图 3-4 所示，中文的常见翻译叫*链路本地*单播地址，它的作用范围和以太网 MAC 地址类似，只能在一条链路范围内使用，使用 link local 地址的通

信不能被 router（广义上的网络层转发设备，并不单单只是商业意义上的路由器）所转发，那么能不能被 bridge 转发呢？可以，因为 bridge 只是將两条 link 在链路层连接起来，并没有使用网络层连接，link local 地址是每一个启动 IPv6 协议栈接口必须有的地址，基本上每个操作系统实现都会为这个接口自动计算 link local 地址，在后面章节会做一些介绍，需要说明的是 link local 地址 interface id 在图 3-4 中是 118-bit 长，但实际范围内，最常见的 64-bit 长的 interface id（前 54-bit 全 0），118-bit 长的接口号，我们可以在不引起地址冲突的前提下在一个链路范围内任意指定；

- *unique local*：最小前缀 fc00::/7，前 7-bit 也如图 3-4 所示，它的中文翻译还不太好找，原来有叫*本地唯一*单播地址的，但总觉得比较别扭，这里还是叫 unique local 地址吧，这个地址的作用范围不像 link local 那么明确，它的范围是管理范围，通常是一个管理单位内部使用地址，类似于 IPv4 中的 *private address* 私有地址（后面有章节详细介绍私有地址），但它又是需要去向 IANA 申请 n-bit 长 *sub-layer prefix*（*子层前缀*），也就是说不同单位的 unique local 前缀不一样，可以避免 IPv4 那种网络融合时私有地址冲突，unique 也在这一刻得到了体现，(121-n)-bit 的接口号可以由申请单位自行划分，比如再划分 *third-layer prefix*（*三层前缀*），但对于 IANA 来说，这是由使用单位自行分配的事情，并不关注；

- *global unicast*：剩下的地址就都属于*全球*单播地址，它的前后缀划分非常灵活，和 CIDR 完全一致，它的作用范围没有限制，只要 IPv6 网络范围有多远，它就可以跑多远，真正的 Internet 范围地址，这和我们当前去运营商申请下来的地址是一个概念，但要搞这么一个地址就需要考虑成本，通常并不是任何单位都能去申请的，使用单位要通过一些代理机构向 IANA 申请一段长度的前缀，这些单位在图 3-6 中所示。

介绍完地址的分类，我们了解到了网络号、接口号、前缀、后缀，里面有一个单位叫 IANA 的出镜率很高，这是个什么样的单位，这里就要介绍一下地址分配体系。介绍地址分配体系前，想让大家参考一下电话号码的分配体系，电话号码是变化长度的，有国家代码、区域代码、城市代码、分区号码、分机号组成，比如中国大陆的国家带码是 086，美国是 001，这是由 ITU（国际电信联盟）分配的，区域带码以下就是由中国电信部门自行定义，比如北京是 10，广州 20，上海 21，深圳 755 等，189 是电信，188 是移动，186 是联通等等，这就些分配 ITU 就不管不着（ITU 也不愿意管这么宽），再然后的分区号码就由各个城市电信部门分配，最后是我们的分机号，中国分机号往往是 4 位的，这么以来就了解到电话号码有这么一个特点：

- 号码长度是变化的，也就是说理论上蛋糕是按国家发，每个国家有自己的蛋糕，蛋糕要多大，国家自己说了算，如果蛋糕觉得小了可以通过号码扩位的方法使整个蛋糕增长，这是电话号码的独到之处；

- 分级别管理，最高一级 ITU 只分配国家代码，国家内部号码怎么分就有各个国家自己搞，总体上来说，拨打一个号码，只要看国家代码是什么就知道往哪个国家送，看哪个区域号码就往哪个省或者直辖市送，非常高效。

在这里讨论电话号码似乎是牛头不对马嘴，但实际上还是可以作为一种辅助信息参考，IP 地址分配有如下特点：

- IP 地址长度是定长的，也就是说蛋糕只有 1 个，而且是固定大小，是一种不可再生资源，定长可以让 IP 网络更高效，有利于承载丰富的业务，电话号码则作用只能是打打电话、发发短信、传真、留

言等不多应用，定长的 IP 地址也决定了分级管理方面的一些特点；

- 分级管理，最高一级是 IANA，目前分配体系有点凌乱，不如 ITU 来得层次分明，由于蛋糕就那么点大，谁都想申请多一些，所以申请不是以国家名义申请，而是以组织申请，各个组织通过不同的代理机构向 IANA 提出申请，比如中国电信、中国联通、中国移动、教育网、腾讯、百度、阿里巴巴都通过亚太区代理向 IANA 申请一些地址空间，由于地址空间是有限的，IANA 分配地址一开始也没有经验，IPv4 的 A 类地址是 127 块，很快就分光了，等中国组织去申请的时候，只能少量的 B 类地址和 C 类可选，还有另外一个问题就是中国电话号码都有统一的国家代码，IP 地址确没有，一个国家甚至组织拥有的地址可能是不连续的，分散的，好比一个组织拥有的蛋糕并不是一块完整的大蛋糕，而是好几块小蛋糕拼凑起来的，这会带来另外一个问题——寻址比较困难，不如电话号码用国家代码、区号就可以完成，IP 网络必须通过路由协议来完成这种复杂的寻址；为什么不能像电话号码一样，每个国家采用一个固定前缀呢？因为互联网在世界发展并不平衡，蛋糕有限的情况下不能按照平均方法处理；

- 自助分配接口号，这点和电话号码是一样的，比如一个国家获得国家代码后，可以在国内自行设计区域代码、城市代码等等，和 ITU 没有关系，IP 地址情况类似，比如向 IANA 申请了一个 n-bit IPv6 前缀后，(128-n)-bit 的 interface id 只是对于 IANA 来说的，使用单位还可以进一步划分子层前缀、三层前缀或更多等等；和电话号码、IP 地址分配截然不同的是以太网 MAC 地址（现在也被称为 IEEE 802 地址）的分配，如图 3-5 所示，48-bit 长的 MAC 地址的前 24-bit 被称为 *OUI*（*Organizationally Unique Identifier 组织唯一标识*）和后 24-bit 的 *NIC*（*Network Interface Controller Identifier 网络接口控制器标识*），IEEE 为每个网络接口卡厂商颁发唯一的 OUI，每个厂家为生产的每块网络接口卡分配 NIC，NIC 不存在什么子层、三层等概念，这两部分格式都是固定长度，使用起来远不如 IP 地址和电话号码来得灵活，它的好处在于分配、管理、都要更高效；

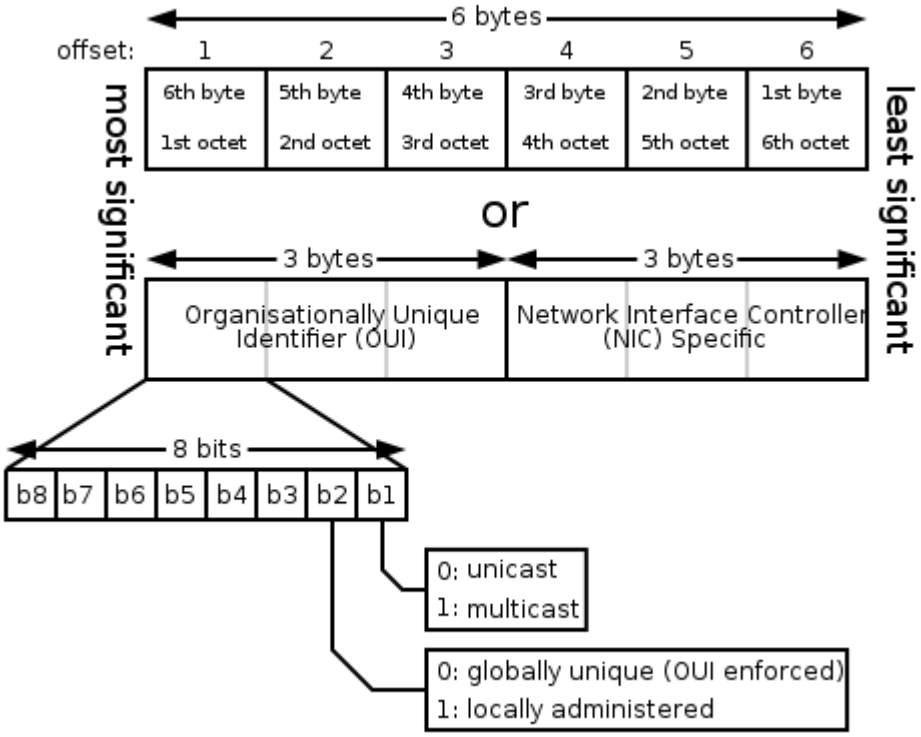


图 3-5 802 地址结构

IANA 在 IP 地址分配、协议号分配、TCP/UDP 端口号上面享有无上权威，所谓分配也就是说一旦地址分配出去，就是申请单位专有，其余单位不得冒用，否则互联网將陷入混乱。刚才说了 IPv4 地址分配在刚开始没什么经验，主要是当时互联网也不像现在，由电信、联通这样的 *ISP*（*Internet Service Provider 互联网服务提供商*）组建，而是各个企业之间互相连，所以地址分配给各个企业，比如像 IBM 这样的跨国企业，同属于一个 A 类地址空间的 2 个地址却分布在不同国家，想想怎么通信可能需要通过一些独特途径，互联网发展到现在，搭建物理上互联的网络是 ISP 更在行，地址分配给 ISP，不同的单位通过连接到不同的 ISP 获得 ISP 再分配的地址可能更合理一些：

- ISP 通过 *RIP*（*Regional Internet Registry 互联网区域注册代理*）向 IANA 提出申请；
- IANA 分配合适大小的地址空间（地址不是按个分的，而是按照*块 block* 分的，通常使用前缀长度来区别不同块的大小）给 ISP；
- ISP 内部再层次化分配更细的块；
- 某个单位连接到某个 ISP 时，ISP 再为这个单位提供一些地址块。

图 3-6 IANA 划分的 5 大 RIP

5 大 RIP 是按照亚太、欧洲、非洲、北美、拉美划分的，不同的 RIP 有不同的分配策略，比较厉害的是 RIPE、ARIN 和 APNIC，基本上是根据经济状况来的，我们来看一下 IANA 在 IPv4 和 IPv6 地址上的分配策略：

- IPv4 地址是按照/8 的前缀分配给不同的组织的，不同组织内部可以划分更细的分块，比如分配 256 个/16 的块；

- IPv6 地址则是在 IETF 规定的前提下划分前缀：

    - 比如 link local 地址 IETF 定义了 fe80::/10 固定前缀，而这个地址又只能在链路范围内使用，就不需要 IANA 分配了；

    - 再比如 unique local 地址 IETF 定义了 fc00::/7 固定部分前缀，那么 IANA 就可以分配子层前缀诸如 fc00:1:2::/48 的前缀给某个单位，甚至可以再细一点，但绝对不能超过 fc00::/7 的范围，用户可以自行划分三层、四层或更多层前缀；

    - 组播地址和 global unicast 地址在上一条已经说了，可以划分不同的子层前缀，用户可以自行划分三层、四层或更多层前缀，如图 3-7 所示一个单位从 IANA 申请 2011::/96 的前缀，可以自由地划分成若干个不同长度的前缀，就好比一块蛋糕中再不停地分割以满足自己的要求，怎么分割更合理是属于网络规划范畴。

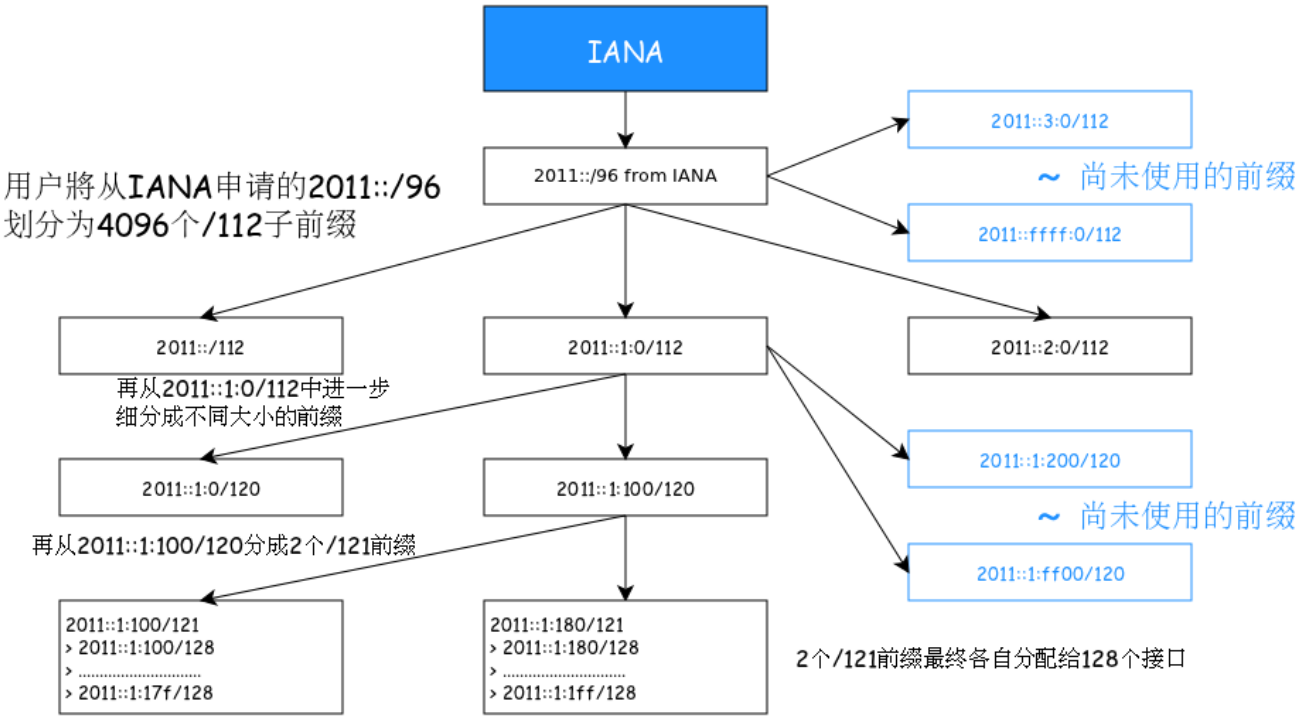


图 3-7 地址划分示意

IANA 主页上有当前 IPv4、IPv6 地址的分配情况和原则，这里就不过多介绍。

在本节最后介绍一些特别地址或前缀的用途，这些地址在后续的章节可能会对这些特殊地址进行引用：

	特殊地址	作为源地址	作为目的地址	作用
IPv4	0.0.0.0/32	yes	no	IPv4 未指定地址
	127.0.0.0/8	yes	yes	IPv4 环回地址
	host id 全 0 地址	no	no	用于表示某个网络号所在网络
	host id 全 1 地址	no	yes	网络号所标识网络的广播地址
	255.255.255.255/32	no	yes	广播地址
	10.0.0.0/8	yes	yes	私有地址，不得出现在互联网
	172.16.0.0/12	yes	yes	同上
	192.168.0.0/16	yes	yes	同上
IPv6	::/128	yes	no	IPv6 未指定地址
	::1/128	yes	yes	IPv6 环回地址

从这张表可以看出点端倪，IPv4 的 host id 中有 2 个是不能用的，一个是全 0，一个是全 1，IPv6 interface id 则没有这么一说，那么我们可以再来计算一下地址划分和地址利用率的问题，假设我们分到了

一个/24 的 IPv4 网段：

- 32 - 24 = 8，host id 有 8 位，也就是 2 ^ 8 = 256 个 host id，全 0 和全 1 不能用，可用 host id 为 254 个，地址利用率为 254 / 256 * 100% = 99.22%；
- 划分成 2 个/25，每个/25 有 host id 7 位，即 128 个 host id，扣除 2 个不能用的，还有 126 个，利用率为 126 * 2 / 256 * 100% = 98.44%；
- 划分成 4 个/26，每个/26 有 host id 6 位，可用 host id 为 62 个，利用率为 62 * 4 / 256 * 100 = 96.88%；
- 划分成 64 个/30，每个/30 有 host id 2 位，可用 host id 为 2 个，利用率为 2 * 64 / 256 * 100 = 50%；
- 划分成 128 个/31，每个/31 有 host id 1 位，可用 host id 为 2 ^ 1 - 2 = 0，利用率为 0；
- 划分成 256 个/32，每个/32 就是一个个主机地址，不区分 host id 全 0 或全 1，理论上利用率是 100%，但这种地址只能作为 loopback 地址，不能用于互联，所以利用率很有限。

从这里我们可以发现，IPv4 中地址划分越细，对地址造成的浪费越多，所以 IPv4 地址分配工作也算是网络规划中一个伤不起的基础课题。IPv6 因为没有 interface id 全 0、全 1 不能使用的问题，所以不存在浪费，无论怎么划分，地址利用率都是 100%，这一点就变得很棒了，有没有。

3.4 寻址与路由

前面介绍了 IP 头结构、IP 地址分类与分配，既然把 IP 当作信封邮寄，那我们就来讨论邮递系统中实际的工作，信件的传递工作，这应该是邮递系统中最庞大复杂的一部分，称为*寻址 addressing* 系统，寻址分为*路由 routing* 和*交换 switching* 两大体系，在 IP 领域的寻址方式是*路由 routing*，同样也是 IP 领域中最为核心的一部分。

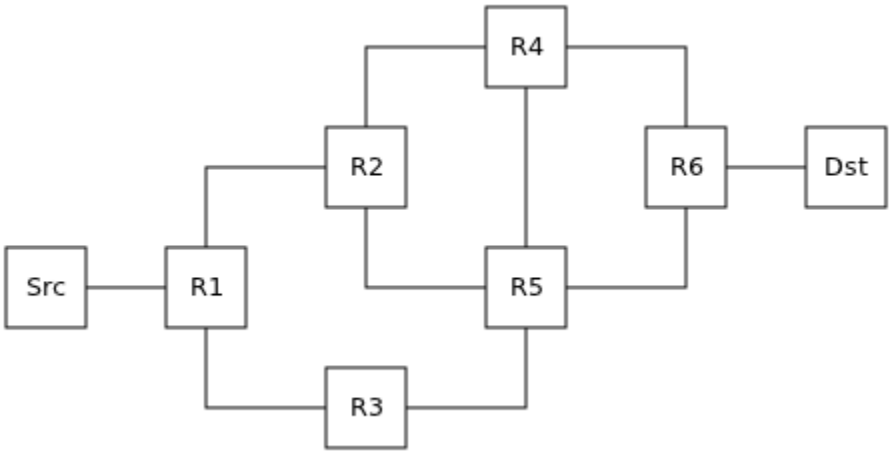


图 3-8 网络层转发拓扑示例

如图 3-8 所示，这个网络中 Src 和 Dst 是两台通信主机，R1～R6 是组成网络的路由器，主机、路由器之间的连线都是一条链路。简单来说就是寻址的目的就是把一个 IP 包从 Src 传递到 Dst 的过程。我们的邮递系统是根据收件人地址信息进行传递，那么 IP 网络的寻址和路由有什么特点呢?

- 默认情况根据目的 IP 进行转发，类似于根据收件人信息传递邮件，也有一些特殊的转发策略可以结合源 IP、协议端口号考虑转发，但这属于另外一个范畴叫*策略路由 policy based route*，这里暂不做考虑；

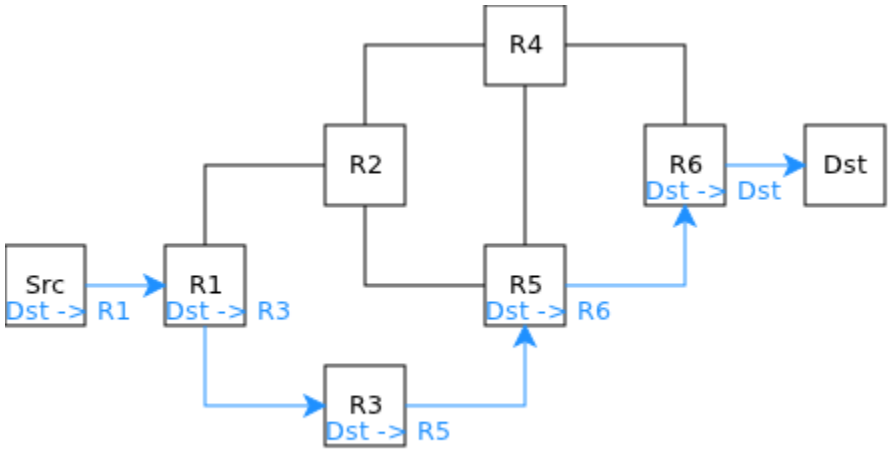


图 3-9 per hop forwarding

- *逐跳转发 per hop forwarding*，如图 3-9 所示，逐跳转发就是指从 Src 到 Dst 传输路径的每一个节点都要做路由工作：
  - Src 发出时，路由查找的结果是 Dst -> R1，意味着这个数据包要交给 R1 才能到达 Dst；
  - R1 收到 IP 包，也要进行路由查找，结果是 Dst -> R3，于是数据包交给了 R3；
  - R3 交给 R5、R5 交给 R6，都做了相同的路由工作；
  - R6 收到 IP 包后，也要进行查找，只不过查找的结果是 Dst -> Dst，意思 Dst 和 R6 是在同一条链路上的节点，也就是说 R6 是这个数据包的*最后一跳 ultimate hop*，最后一跳也是通过查找路由的方式确认 Dst 和自己在同一条链路上的。

也许还会有疑问，R1 同样可以通过 R2 -> R4 -> R6 到达，为何 R1 选择了 R3 呢? 我们来看一下所谓路由的组成，先看看 Linux 系统下的：

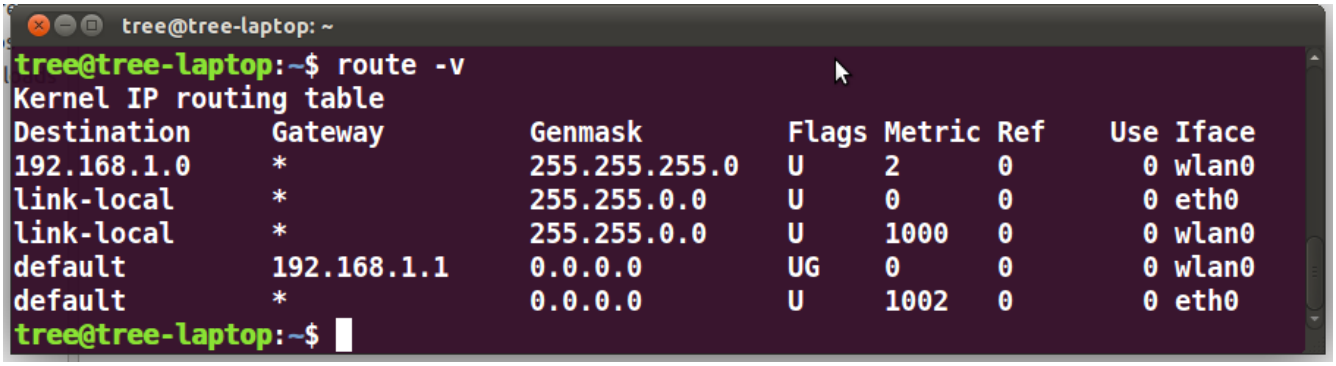


图 3-10 Linux 路由表

可以看出来这是个表，没错，这个玩意叫*路由表 routing-table*，我们来看列的组成：

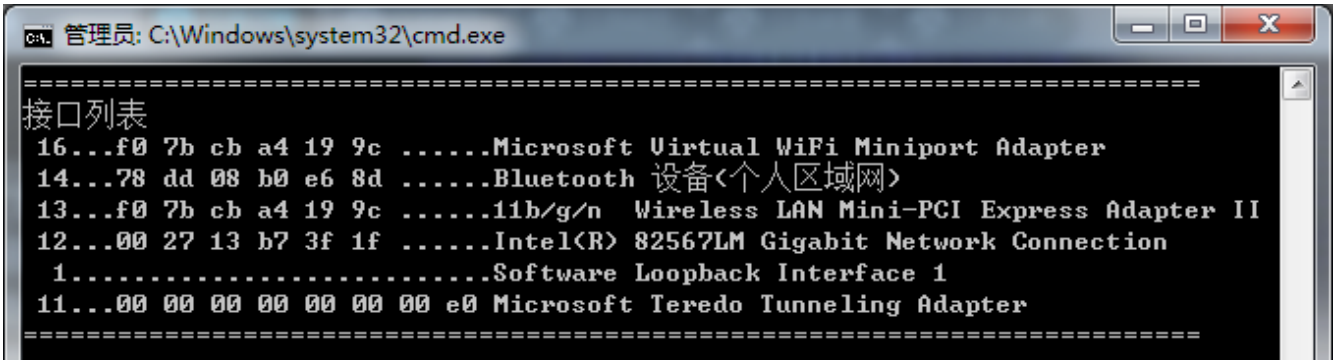
- Destination，表示路由匹配哪个目的地址；
- Gateway，网关，即 next-hop router，下一跳路由器，也就是说前往 Destination 应该把 IP 包交给谁？这个 Gateway 必须是和路由表所有者在同一条链路上的，如果是*则表示该路由 Gateway 是环回接口接口，或者不存在；
- Genmask，掩码，用于计算网络号和主机号，配合 Destination 来表示一下网络前缀，比如路由表的第一条 192.168.1.0 作为 Dst，掩码是 255.255.255.0，那么表示的网络前缀就是 192.168.1.0/24：
  - 如果目的 IP 是 192.168.1.1，和 192.168.1.0/24 有什么关系呢，192.168.1.1 的二进制码前 24 位就是 192.168.1.0，所以我们把这种情况称为 192.168.1.0/24 为 192.168.1.1 的*命中 match* 路由，或者叫匹配路由，反之叫*非匹配路由 dis-match route*，192.168.1.0/24 可以命中 192.168.1.1～192.168.1.254 这么多 IP 地址的，这种路由比较常见，因为可以通过人工的聚合，一条路由命中许多许多地址，以减少路由条目；
  - 同样，192.168.1.1/32 也可以命中 192.168.1.1，而且只能命中 192.168.1.1，这种路由称为*主机路由 host route*，IPv6 中的主机路由是/128 的，因为 IP 地址数量太多，一般不太可能为每个 IP 地址都建立一条主机路由，所以通常采用覆盖量大的路由，比如 IPv4 中常用/30、/24、/20、/19 等等，而 IPv6 可以变得更加灵活；
  - 还有一种路由是 0.0.0.0/0 或::/0，因为任何地址的前 0 位都是 0，所以这种路由可以命中任何地址，被称为 *default route*，中文称呼叫*默认路由*或者*缺省路由*，在每台 PC 上配置网关地址就是配置一条默认路由；
- Flags，标志位，用于说明这条路由的状态（可以复选）：
  - U，表示 Up 或者 Using，这条路由是活跃的，在使用中的，比较常见；
  - H，表示这条路由的掩码是 255.255.255.255 或者/128，这是一条主机路由，只能匹配某一个

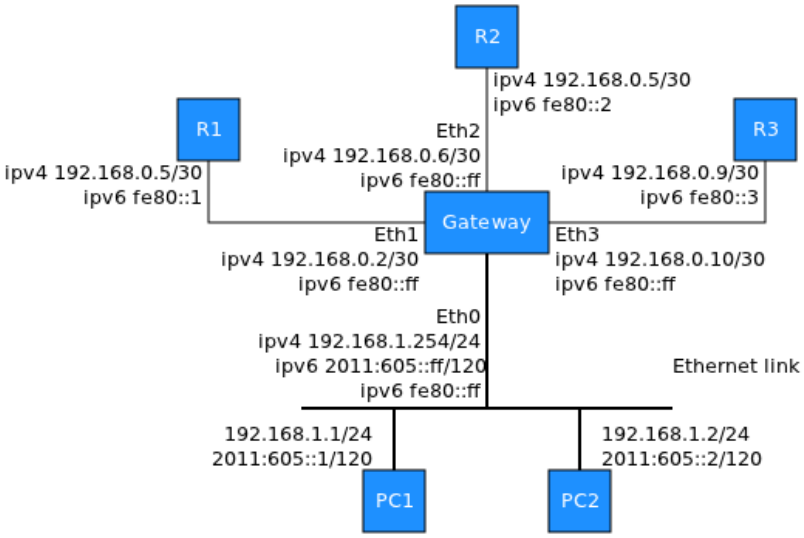
地址，偶尔见到；

- G，表示这条路由掩码是 0.0.0.0 或者/0，即 Destination 可以覆盖任何一个 IP 地址，这条路由被称为 ***default*** *默认或缺省*路由，也比较常见；
- R，Reinstate，动态路由的待命状态；
- D，Dynamic，由路由协议或者重定向安装的路由；
- M，Modified，由路由协议或者重定向修改的路由；
- A，Addrconf，表示由脚本添加的一条路由，不太常见；
- C，表示 Cache，被属于一种比较高级的转发条目，在主机上不太常见，在路由器上很常见；
- ！，表示 Reject，称为黑洞路由，即永远不转发，只丢弃，也不解释原因，常因为安全因素设置；

- Metric，开销，有的地方也用 Cost 来表示，用于描述路由前往某网络前缀需要的花费，类似于描述路途多远，这是衡量路由好坏的关键，比如图 3-9 中为何 R1 会选择 R2，而不选择 R3，往往就是因为选择 R3 作为网关的 Metric 比 R2 大，说明选择 R2 离 Dst 更近；
- Ref，Reference，表示这条路有被引用过几次，和转发选择无关；
- Use，表示这条路由被使用过几次，和实现有关，和转发选择无关；
- Iface，interface，接口，也就是说 Gateway 是在哪一条链路上相连的，比如这里有 wlan0 和 eth0 分别是 WLAN 接口 0 和以太网接口 0；

从上面可以看出路由表中和转发关系最大的就是 Destination/Mask、Gateway、Metric、Interface，其余都是一些状态信息，再来看看另外一个操作系统 Windows 下的路由表：



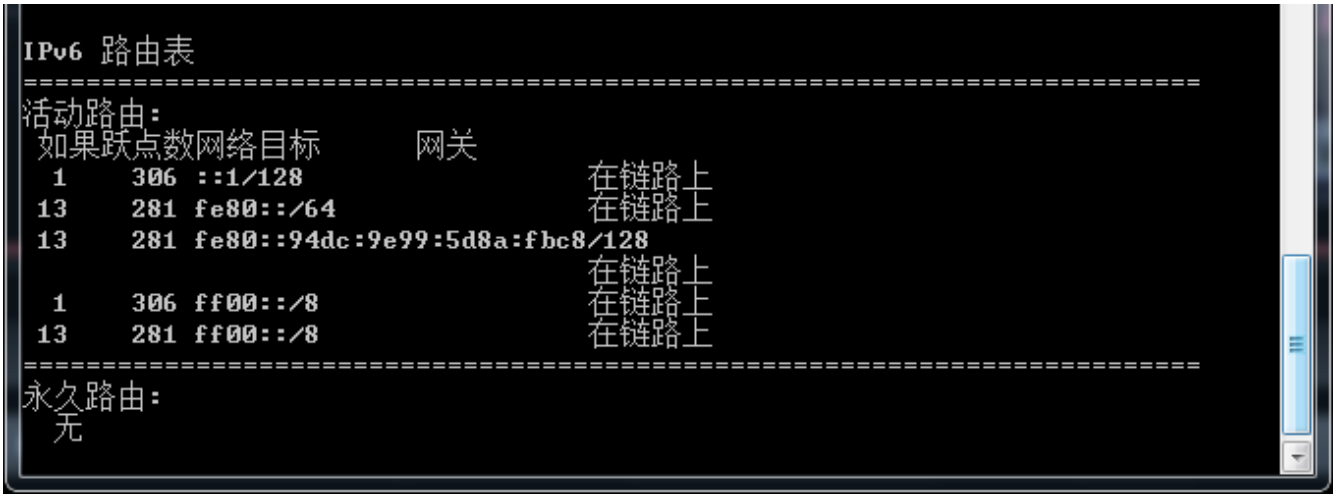


Gateway's ipv4 routing tabel

Dst/prefix	next-hop/outif	metric
192.168.1.1/32	192.168.0.10/eth3	10
192.168.1.0/24	direct/eth0	0
192.168.0.0/16	192.168.0.6/eth2	5
192.168.0.0/16	192.168.0.1/eth1	6
0.0.0.0/0	192.168.0.1/eth1	6

Gateway's ipv6 routing tabel

Dst/prefix	next-hop/outif	metric
2011:605::2/128	fe80::3/eth3	10
2011:605::/120	direct/eth0	0
2011:605::/112	fe80::2/eth2	5
2011:605::/112	fe80::1/eth1	5
::/0	fe80::1/eth1	1



2. Eth1～Eth3 分别连接 R1～R3：

   1) Eth1 接口 IPv4 地址已经介绍，是 192.168.1.254/24，IPv6 地址分 link-local 和 global unique 两个，分别是 fe80::ff 和 2011:605::ff/120；

   2) Eth2 接口 IPv4 地址是 192.168.0.2/30，IPv6 地址只有 link-local，也是 fe80::ff；

3) Eth3 接口 IPv4 地址是 192.168.0.6/30，IPv6 地址只有 fe80::ff；

3. Eth0～Eth3 的 IPv6 地址看起来比 IPv4 奇怪，每个接口都使用了 fe80::ff 这个 link-local 地址，这个地址只在同一个链路起作用，图 3-12 中 Gateway 的 4 个接口被我们划分成了 4 个不同的链路，之间使用网络层隔离，所以这么使用没有问题，1 个设备使用相同的 link-local 地址，便于大家记忆，很好的习惯哟，图 3-12 的 IPv6 路由表的 next-hop 都是 link-local 地址，其实你的 PC 如果在 IPv6 环境中，也可以发现这个现象，规划 link-local 地址变成一项新课题；

4. 只有 Eth0 拥有一个 global unique 地址 2011:605::ff/120，其余接口都没有，似乎和 IPv4 有专门的互联地址很不一样，在 IPv6 中，路由条目的 next-hop 很多时候是一个 link-local 地址，以严格保证路由的逐跳性，只有在一些隧道路由的时候会使用非 link-local 地址，在 3-12 的组网中，使用 link-local 地址互联并不会造成路由表上的混乱。

在旁边可以看到 Gateway 上的 IPv4 和 IPv6 路由表，比如这个时候 Gateway 有个数据包 Dst 是 192.168.1.1，会转发给谁呢?

- 首先 Dst 是 IPv4 地址，所以只能查 IPv4 路由表，根据路由匹配原则，都可以命中 192.168.1.1 啊：
  - 192.168.1.1/32　　192.168.0.10/eth3　　10，也就是说 next-hop 是 R3；
  - 192.168.1.0/24　　direct/eth0　　0，也就是说 next-hop 就是 192.168.1.1，因为是直连的；
  - 192.168.0.0/16　　192.168.0.6/eth2　　5，next-hop 是 R2；
  - 192.168.0.0/16　　192.168.0.1/eth1　　6，next-hop 是 R1；
  - 0.0.0.0/0　　192.168.0.1/eth1　　6，next-hop 是 R1；
- 下面来介绍一下路由表*最长匹配 longest match first* 原则，即在所有命中路由中选择前缀最长的路由，没有路由要比主机路由要更长了，所以在这里会命中第一条路由，将数据包转发给 R3；
- 怎么样确定一条路由是不是最长匹配呢?
  - 把路由表中的所有路由都匹配一遍，总能找到最长的，这种方法称为顺序匹配，这种方法在路由表很庞大时，可以看出来效率会有多么的低下；
  - 先把路由表按照顺序排列好，比如按照前缀从长到短排列，匹配的时候从长开始，那么能匹配的第一条路由就是最长匹配，这种方法提高了一点效率，在图 3-12 中就是按照这种方式先排列，再匹配；
  - hash 查找法，通过设计某种 hash 算法将 IP 地址映射到不同的路由上，保证首次 hash 命中的最

长成功率是设计 hash 算法的关键，hash 是目前的主流；

- 这里有人会问按照上面的 IPv4 路由表，如果目的地址是 192.168.2.1，会匹配哪条路由：
  - 192.168.1.1/32 和 192.168.1.0/24 显然是不能命中了；
  - 192.168.0.0/16　　192.168.0.6/eth2　　5，next-hop 是 R2，可以命中；
  - 192.168.0.0/16　　192.168.0.1/eth1　　6，next-hop 是 R1，也可以命中；
- 两条同样最长匹配时就要关注 metric 了，metric 小的意味着距离更短，更优，所以选择 R2，那么什么时候选择 R1 呢？如果 R2 这条路由因为线路故障原因消失了，就会选择 R1 这条，所以这种方式叫*路由备份 route backup*，这里的路由表只是示例，通常 metric 大的那条路由被称为备份路由，是不会出现在路由表中的，因为路由表默认只显示 *active* 状态的路由，也就是说当前就可以用于转发的路由，备份路由的状态通常是 *inactive*；
- IPv6 路由表的情况基本类似 IPv4 路由表，但细心的我们还是可以发现百花丛中的奇葩：
  - 2011:605::/112　　fe80::2/eth2　5，next-hop 是 R2；
  - 2011:605::/112　　fe80::1/eth1　5，next-hop 是 R1；
- 乖乖，两条路由前缀一样，metric 也一样，就是 next-hop/outif 不一样，这种现象称为*等价路由 equal cost route* 或者叫 *ECMP*（*equal cost multiple path 等价多路径*），如果匹配中 ECMP，通常的策略是在不同的 next-hop 间轮询转发，保证一定的负载均衡，轮询的因子有*逐包 per packet*，和*逐流 per stream*，逐流会更普遍一些，等价路由比路由备份的好处就是既能起到备份的高可靠性效果，还因为同时转发提高链路效率，是一项比较多人研究的网络模型；

我们了解了路由表的构成，也了解了路由查找原理，必然对路由表的产生比较感兴趣，这恰恰是网络建设中比较关键的内容，被称为网络规划，如图 3-12 中的路由表并不是天生如此，而通常是设计人员有意为之，使网络能够满足一些特殊的业务需求，网络规划里面有太多策略、成本方面的因素，很难说这个绝对就好，那个就绝对差，只能说这个是最适合我需求的。这里我们还是讲讲一些路由表产生的基本原理吧，路由条目根据产生途径可以分为：

1. 直接路由，direct route，也就是说伴随着链路产生的，direct 路由在路由表里都有非常明显的特征，比如 next-hop 是一个自身接口地址或者 127.0.0.1 或者直接有个 direct 标记，direct 路由虽然不需要人为操控生成，但接口地址也还是人配置的，比如 eth0 接口配置 2011:605::ff/120，那么接口状态 Up 后，就自动生成了一条直连路由 2011:605::/120，出接口就是 eth0，我们比较好奇，什么样的接口状态称为 Up 呢，命中了直接路由会怎么处理呢：
   1) 如果是共享型链路如以太网，那么只需要接口物理上电信号符合以太网条件，就 Up 了，或者 WLAN，则可以接收到符合条件的无线信号，也可以 Up，当然很多 PC 上的实现另当别论，如

配置了 802.1X 认证，认证失败即使可以接收电信号也不会认为是 Up，这里介绍的还主要是路由器的情况；

2) 对于 PPP 这种点到点链路，Up 的条件是 NCP 协商完成，这个要比以太网高级，因为前提条件是协商成功，只有物理信号收发是远远不够的；

3) 命中直连路由的处理很简单，就是交给出接口所在的链路层，发送出去，因为 IP 地址直接连在了这个接口上，聪明的读者可能会问，如果是以太网接口，链路上连接了好多 PC 呢，链路层是怎么把 IP 层的数据准确地交给其中一个 PC 呢？这个问题会在我们的 ARP、ICMPv6 中得到解答；

2. 间接路由，indirect route，只要不是直接路由就是间接路由，也就是别的主机或者路由器接口上的直接路由，要通过其余路由器才能达到的 Destination 就需要靠间接路由，这是所有路由的重点，间接路由分 2 类：

1) 静态路由，static route，由人工逐条配置方式产生的路由条目称为静态路由，主机上设置网关就是最常见的静态路由，静态路由的特点比较简单，完全由人来操控，在网络非常稳定的情况，还是非常适合的，但是如果网络结构是不断变化的，那就非常的讨厌了，需要人工不停地去修改静态路由，比如图 3-9 全部都是静态路由方式，R5 突然歇菜了，那么 R1 上就要重新修改至 R2 访问 Dst，如果 R1 这边刚改好，R5 由恢复了，那是不是要再改回 R3 呢？没错静态路由相当折腾人，就好比传统地图和电子导航的差别，用去年的地图记得是第 3 个红绿灯左拐的，坑爹的在前面又修了个红绿灯，岔路去吧，电子导航更新及时就不会有这个问题鸟^_^，在道路建设最多的中国，静态根本行不通啊，必须要实时动态；

2) 动态路由，dynamic route，它的出现就是为了解决静态路由太折腾人的问题，静态路由设置正确有两个条件：

(1) 人首先要收集如图 3-8 那样的整个网络拓扑，就相当于先画一张地图；

(2) 在每一台路由器上为需要通信的非直接网段设置一条路由，并且保证路由是无环的，比如图 3-13 中对于 Dst 路由设置是 R1->R3，R3->R5，R5->R4，R4->R2，R2->R1 这就是一个环路，数据包从 R1 发出后又绕回 R1 了，最简单的环路是 R1->R3、R3->R1，所有路由可以不是最优的，但必须是无环的，环路的危害很严重，一是无法將数据准确送达目的，其次还消耗链路带宽，IP 头部的 TTL 和 hop limit 字段都是为了防止环路的措施，没经过一个 hop，TTL 和 hop limit 就减 1，当这个数值为 0 时就不允许再发出去了，因此一个数据包不可能在环路中无限地转发。

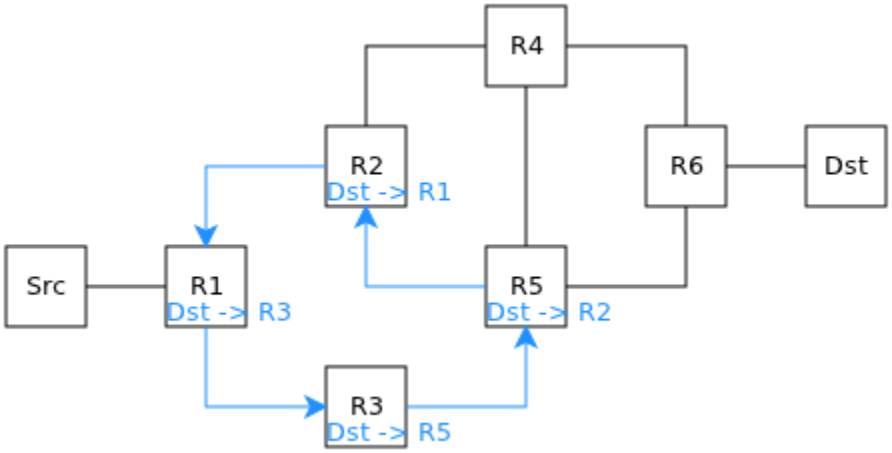


图 3-13 路由环路

既然静态路由是靠人来完成的，那计算机也可以完成，所以人在静态路由的基础上设计了动态路由协议，比如 *RIP*（*Route Information Protocol 路由信息协议*）、*OSPF*（*Open Shortest Path First 开放最短路径优先*）、*IS-IS*（*Inter System - Inter System 中间系统 - 中间系统*）、*BGP*（*Border Gateway Protocol 边界网关协议*），每种路由协议都有自己的规则，目的就是在不同的路由器之间交换直连路由，通过一致的规则计算出无环路由，由于是自动运行的协议，所以一旦配置 OK 后，它最大的特点就是适应网络变化，解放了人，人可以干些更牛 B 的工作。不同的动态路由协议有不同的原理，比如 RIP 和 OSPF 就属于不同原理的两种，但殊途同归都是通过同一条链路上传播不同路由器的直接路由，如图 3-14 就是以 R1 上直接路由 Src 传播到 R2～R6 的例子，这个图展示的是 RIP 的部分原理，RIP 简单地说是不画地图的，简单地向邻居打听路途远近，挑选一个最近的，OSPF 比较高级一些，要先画地图再选路。图 3-14 中“R1 have src”表示可以通过 R1 访问 src，“Src -> R1 @ x”表示通过 R1 访问 src 的 Metric 是 x，路由协议是网络规划的基础，在网络原理部分不作深入探讨。

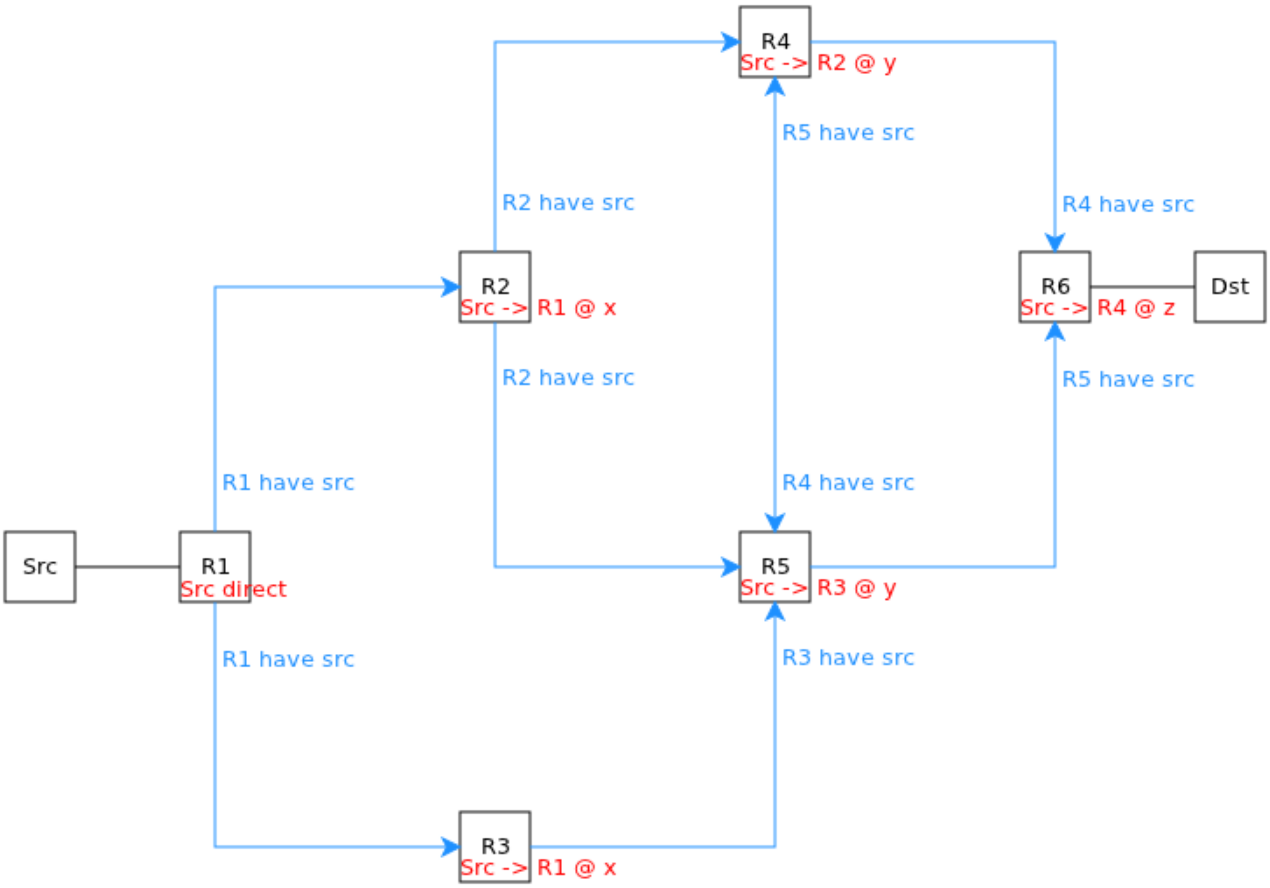


图 3-14 动态路由基本原理

了解完非直接路由，可能会产生个疑问，假设人工设置了一条 2011:605::/120 的静态路由，还有 IS-IS 发现的一条 2011:605::/120 的动态路由，怎么办?

- 首先，这两条路由的关键属性 destination/prefix 是完全一致的，需要我们判断的是否可以称为等价路由或者路由备份，如果不一致，那就是不同的路由，不存在等价或者备份关系；

- 不同种类的路由通过 *preference* 来 pk 优先级别，有的地方也叫 *distance 管理距离*，这个值和 metric 类似，越小越棒；

- 不同的操作系统对于不同种类路由默认优先级设置不一样，比如思科设备中，静态路由的 preference 是 0，和直接路由一样，要比动态路由协议都要高，那么在刚才的假设中，静态路由就会胜出，添加到路由表中，IS-IS 路由落选，只有在静态路由失效后才能进入路由表，变成活跃路由，不同系统的默认值可以使用 Google 搜索一下；

- 既然有默认值，也就意味着这个值是可以调整的，规划人员可以调整不同路由协议的优先级，以实现自己的目的，但唯有直接路由在所有的操作系统中都是 0，也就是最优的，即使將非直接路由优先级都调整成 0，内部实现上也不会让非直接路由和直接路由形成等价，同样即使非直接路由之间 preference 配置成一致，也是会在实现上有个严格的座次，没有并列，也就要求规划人员必须为不同

的路由规划一个唯一的值，经验上来说可以在实验网络中玩玩全部配置成一样，在实际网络中这么玩只会困惑网络维护者，他们可能会不停地咒骂网络规划人员；

- 聊完 preference 或者 distance，肯定有人会说这和刚才的 metric 似乎很像啊，神马区别?

    - 都是针对相同 destination/prefix 时使用，比如 192.168.1.1/32 和 192.168.1.0/24 这样就没法比，只有都是 192.168.1.1/32 或者都是 192.168.1.0/24 才可能比，试想想 IPv6 的 2011:605::/120 和 2011:605::/112 是否有可比性呢？答案显然也是不可比，因为前缀长度不一样；

    - preference/distance 是在不同种类路由协议之间 pk 用的，pk 只有胜利和失败，没有平手，胜利一方进军活跃路由表，败北的在路由表门外等候；

    - metric 是在同种路由内部 pk 用的，pk 有胜利、失败和平手，胜利方称为 active 路由，失败方变成 inactive 的备份路由，如果平手就称为等价路由；

如果命中直接路由是直接从出接口链路层发出，如果命中的是非直接路由肿么办?

- 路由条目中转发信息是 next-hop/outif，也就是说从 outif 中发送给 next-hop 指定的地址喽；

- 还是交给 outif 所在的链路层，链路层接收方是 next-hop 的链路层，比如 next-hop/outif 是 fe80::1/eth1，那么链路层接收方就是 fe80::1 对应的以太网 MAC 地址，这里又要涉及到 ARP、ICMPv6 知识了；

- 总体来说，直连路由、非直连路由在转发上并没有多大区别，如果某条路由只指定 next-hop，没有 outif 肿么办？许多操作系统都有解决这个烦人的问题，叫做迭代查询：

    - 如图 3-12 我们在 Gateway 上新建一条路由 2011:606::/120，只指定了 next-hop 是 2011:605::ff00，这个 next-hop 不在 Gateway 的任何一个接口所在链路上；

    - 系统会使用 2011:605::ff00 作为匹配条件去查询路由表，可以命中 2011:605::/112，有 2 个 next-hop/outif，分别是 fe80::2/eth2 和 fe80::1/eth1，那么迭代查询结束，系统会自动將新建的 2 条路由这么修改：

        - 2011:606::/120　　fe80::2/eth2　　metric

        - 2011:606::/120　　fe80::1/eth1　　metric

    - 路由迭代其实是非逐跳路由的一种，迭代会带来额外的计算工作量，影响日常网络运维效率，运维人员咒骂规划人员坑爹的事情又得发生，所以 IPv6 路由都使用 link-local 地址作为 next-hop 有天然的逐跳性，避免迭代，在这里还是强烈顶一下。

假设没有一条路由命中，怎么办？系统会直接把数据包直接丢弃，附加的动作是向 Source 发送个消息告诉

Source 因为没有路由的原因丢弃了，但现在许多不良 Source 都是故意发送一些明知不可达的 Destination，以消耗路由器资源，学聪明的路由器往往就直丢弃，什么都不通知了。

关于路由，原理部分就介绍这么多，可能还有人会问路由器和交换机到底什么差别，这个问题的突破点太多了，以至于变成一个开放式问题，现在很多交换机都是具备路由功能的，也是路由器，但交换机接口多，同等接口情况下要比路由器便宜很多，所以现在许多交换机在抢路由器的市场，路由器上现在也有了交换板卡，也具备交换机接口多的特点，总体上路由器市场份额是日益缩减，我们没有必要纠结于路由器和交换机谁要打倒谁，聚焦于原理，该交换就要懂交换，该路由就要懂路由。但是另外一个问题，关于 2 大寻址系统路由和交换优劣却不是开放问题：

- 电话网络中寻址系统就是交换机制，交换最大的特点是不需要顺序查找，不存在最长匹配问题，交换表只有唯一匹配，所以效率很高，以太网交换机的交换芯片是个包含若干介质的二层网桥，它的寻址系统也是交换机制，所有表项都是 48-bit 以太网 MAC 查找，不存在 24-bit 或者其余长度表项，所以交换芯片可以做到即高效又便宜，这也是交换机为何如火如荼的原因，交换虽然是唯一匹配，但并不是说不能与 ECMP 共存，意思是匹配条件唯一，outif 可以有多个，交换表项非常简单，以至于比较少人关注了，交换的问题是除了电话分层交换体系外，交换受到表容量限制，比如 48-bit 以太网 MAC 交换表，通常芯片的容量在 32 000 或者 64 000 以内，如果 IPv4 要做成交换模式，至少 2^32 的容量，目前的实现还做不到那么大的交换容量；

- 路由机制最大的特点就是在芸芸路由表中寻找最长匹配，只有全部寻找才能确定那个“最”，效率很低啊，路由虽然原理上不高效，但因为 IP 地址分配方式的灵活，不得不采用路由的方式，路由还可以聚合，节约条目，对于 IP 这种大规模网络只能采用路由模式，合理的路由规划实际效率和交换寻址媲美，现实生活中使用的地图寻址也算是路由模式，前往一个目的地往往有许多许多路径，每条路径的匹配模式并不一样，需要人工去选择，尽管遭到交换寻址派的鄙视，路由还是 IP 网络层的必选项，目前也已经非常成熟，想要消除它，那请把 IP 毁灭了先。

虽然路由和交换是寸有所长，尺有所短的双头龙，许多专家学者设计了一些融合两者有点的东西，其代表就是 *MPLS*（*Multiple Protocol Label Switch 多协议标签交换*），这是另外一个大话题了，在本书中也不做过多探讨。

3.5 转发示例

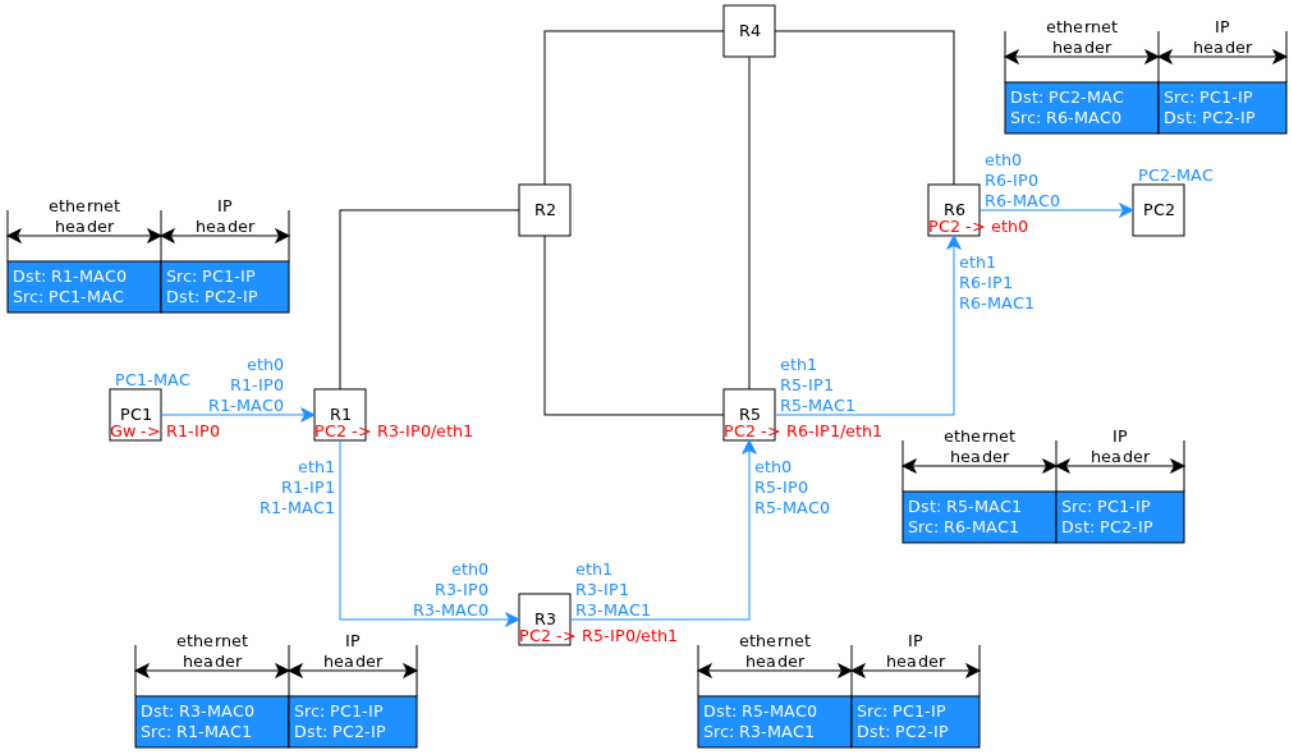


图 3-15 转发封装示意

如图 3-15 所示为 PC1 -> PC2 IP 层通信，蓝色箭头表示数据传送方向，红色字体为每个节点匹配 PC2 的最长路由，蓝色字体为各个接口编号、IP 地址、MAC 地址信息，蓝底百字的是 PC1 -> PC2 的封装，由于通信所有的接口都是以太网，所以都是以太网 - IP 封装（IPv4 和 IPv6 均可），完整的过程描述如下：

1. PC1 上要和 PC2 进行 IP 层通信，首先回去查匹配 PC2 的路由，常见情况就像图 3-15 所示，匹配静态路由 Gw -> R1-IP0，即默认路由的网关地址是 R1-IP0，也就是说 next-hop 是 R1-IP0 这个 IP 地址，如果是 IPv6，这个 R1-IP0 大多数是个 link-local 地址，R1-IP0 甭管是 IPv4 还是 IPv6，都必须通过以太网链路发送给 R1：

   1) ethernet header 的 Src MAC 好理解，必定是 PC1-MAC，因为数据在这条链路上是 PC1 发的；

   2) Dst MAC 是 R1-MAC，不必奇怪，这里讨论的是链路层的事，在 PC1 - R1 这条链路上，R1 收，那 Dst MAC 就是 R1 的 MAC——R1-MAC；

   3) 为何 IP Dst 是 PC2，而 ethernet Dst 是 R1-MAC 呢，就是因为 R1-IP0 是路由表中 PC2 的 next-hop，而 R1-MAC 则是 R1-IP0 所在接口的链路层地址，如何获取这 2 者的对应关系是通过 ARP 或者 ICMPv6 实现的；

2. PC1 按照如上封装发送给 R1 后：

1) R1 发现目的 MAC 是 R1-MAC，就要解封装，把 ethernet header 拆掉；

2) ethernet header 拆掉后，就看到 IP header，这个时候检查 IP dst 是 PC2，不是 R1 的 IP，那就要继续查路由表；

3) 路由表如图所示，最长匹配的最佳路由 next-hop 是 R3-IP0/eth1，也就意味着这个 IP 包要从 eth1 这个接口的链路发送到 R3-IP0；

4) 经过 PC1 上费解的解释，相信大家应该差不多明白了，R1 - R3 这条链路上 R3 收，R1 发，那么 ethernet header 的 Dst 和 Src 就分别是 R3-MAC0 和 R1-MAC1；

3. 这个 IP 包在各个路由器上就像接力棒一样，依次传递到了 R6：

   1) R6 也是要查找路由表，只不过这次路由 next-hop 是 direct/eth0，也就是 eth0 接口的直接路由；

   2) 那就直接从 eth0 接口继续玩链路层 R6 发，PC2 收的游戏，最后一个 ethernet header 就像图 3-15 画的那样，Dst 是 PC2-MAC，Src 是 R6-MAC0；

4. PC2 收到这个包如何处理：

   1) 以太网拆封装，只剩下 IP 封装；

   2) 看到 IP header 里的 Dst 是 PC2 的 IP 地址，那么就进入到第二章图 2-67 所提及的 loopback 处理环节了。

整个过程就完毕了，这幅图是很经典的一个考题，考察知识点主要是对路由、链路层传送原理，从这张图中我们可以看到 IP 地址在 IP header 中的作用就是建立一条从 PC1 到 PC2 的路径，这条路径被称为端到端路径，因此 IP 被称为提供端到端通信服务，但是实际上传递还是在一条接着一条的链路上由链路层完成的，链路层职责就在链路上传递数据。IP 层和链路层的衔接则是下一章的内容。

3.6 练习

1. 检查一下你的电脑上是否同时具备 IPv4 和 IPv6 地址，分别是什么？

2. 看看是否通过 wireshark 或者 tcpdump 可以抓到 IPv4 和 IPv6 的包，观察一下它们的 header 区别有哪些？

3. 到 google 上搜索一下如果 IPv4 分片丢失，重组时需要做什么？如果路由器做重组是否会有什么隐患？

4. 把刚才查看到的电脑上 IPv4 和 IPv6 地址窗口关掉，是否可以把它重写下来，是否可以记住它们的

特点?

5. 检查一下你的 IPv4 地址和 IPv6 地址分别属于哪一类地址，IPv4 地址是私有地址吗，如果是查看一下你的?

6. 如果现在有一条链路上有 40 台 PC，计划为每台 PC 分配一个 IP 地址，IPv4 的 host id 和 IPv6 的 interface id 至少需要几位，是否有什么问题?

7. 现在给定一个前缀空间 2011:609::/112，现在有 3 个部门，要求同一个部门的 PC 都在相同的链路和上，第 1 个部门有 60 台 PC、第 2 个部门有 120 台、第 3 个部门则有 200 台 PC，怎么划分前缀空间满足要求，并满足最高的地址利用率?

8. 一个美国公司通过 ARIN 申请到了一个 2011:609::/96 的前缀供访问互联网使用，因为业务拓展到了中国，该前缀中的一部分 2011:609::217:0/112 被划分出来给中国分部使用，也发布到互联网，那么互联网上是不是存在 2011:609::/96 前缀范围内的两块路由，小的一块指向中国，大的一块指向美国，有什么办法解决这种路由碎片?

9. 路由表条目的多少取决于地址的分配或者划分还是取决于某种技术？为什么?

10. 链路层使用网络层隔开后通过路由方式互联和不同链路使用网桥互联有何区别?

11. 路由的 outif 是一个共享型链路如 ethernet 时为何需要指定 next-hop，如果没有会怎样？如果这个以太网链路是一个点到点介质是否还需要指定 next-hop？如果是像 PPP 这样的点到点链路又如何?这 3 者之间的区别在哪里?

12. 在图 3-15 中，PC2 -> PC1 的路径是否可以是 PC2 -> R6 -> R4 -> R5 -> R2 -> R1 -> PC1，为什么?

13. 检查一下电脑的路由表，分别查看 IPv4 和 IPv6 部分，看看网关地址和出接口是否能够和 PC 上众多网络接口对应起来?

14. 为什么图 3-12 中 Gateway 不同以太网接口之间可以使用相同的 link-local 地址?

15. 为什么路由器之间 IPv6 互联接口只需要 link-local 地址就 OK 了?

16. 路由器 IPv4 互联为何需要专门为接口指定唯一的地址，而不能像 IPv6 那样各个接口使用相同的 link-local 地址?

17. 如果在某个路由器的某个接口上配置了 IPv4 地址 6.9.2.17/24 和 IPv6 地址 2011:609:217::/120，是否都成功？IPv4 网络号、主机号、IPv6 的前缀、接口号分别是多少？接口如果状态是 Up 的话，产生的直接路由分别是什么?

18. 默认路由 0.0.0.0/0 和::/0 是间接路由还是直接路由？为什么？

19. 路由匹配如果取消最长匹配模式，是否会产生环路？

20. 在间接路由中，为何需要为不同来源的路由指定不同的 preference 或 distance？如果网络中的路由器都使用了多种来源的间接路由学习相同的路由，且各来源的 preference/distance 排列顺序不一致，是否会导致环路？

21. 为何 IPv6 路由 next-hop 是 link-local 后可以避免路由迭代？

22. 等价路由是否要求路由来源一致，同时 metric 也一致？那么路由备份呢？

23. 现在有一个网络共有 30 000 台 PC，假设全部连接在同一条以太网链路上，使用交换方式互联，那么这个以太网交换机的交换容量应该满足多少条 MAC？如果划分成 300 条以太网链路，每条链路 100 台 PC，并分配一个 IP 前缀，300 条链路之间使用路由方式相连，那么这个路由器的路由表容量应该满足多少条路由？