



武汉大学

WUHAN UNIVERSITY

## 第四章 网络层

---

林海

[Lin.hai@whu.edu.cn](mailto:Lin.hai@whu.edu.cn)



# 网络层

网络层的主要功能为实现  
网络数据端对端的传输

Application
Transport
Network
Link
Physical



## 第 4 章 网络层

- 4.1 网络层提供的两种服务
- 4.2 网际协议 **IP**
- 4.3 划分子网和构造超网
- 4.4 网际控制报文协议 **ICMP**
- 4.5 互联网的路由选择协议
- 4.6 **IPv6**
- 4.7 **IP 多播**
- 4.8 虚拟专用网 **VPN** 和网络地址转换 **NAT**
- 4.9 多协议标记交换 **MPLS**



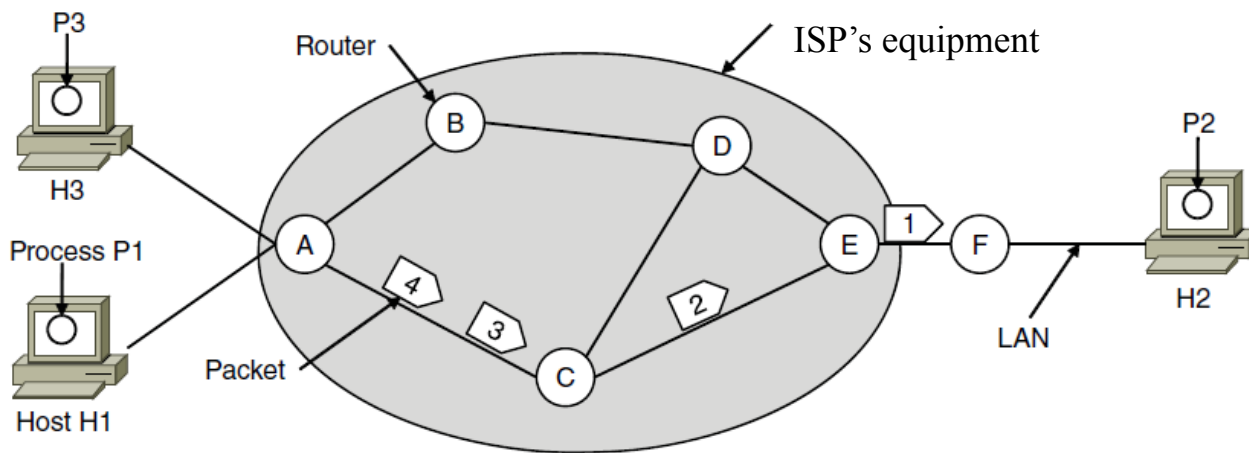
## 4.1 网络层提供的两种服务

- 在计算机网络领域，网络层应该向运输层提供怎样的服务（“**面向连接**”还是“**无连接**”）曾引起了长期的争论。
- 争论焦点的实质就是：在计算机通信中，可靠交付应当由谁来负责？是**网络**还是**端系统**？



# 面向连接：虚电路

- 数据包是通过一条虚电路实现的
  - 在发数据之前把虚电路建好



A's table

H1	1	C	1
H3	1	C	2

C's Table

A	1	E	1
A	2	E	2

E's Table

C	1	F	1
C	2	F	2

In: Line Tag    Line Tag: Out



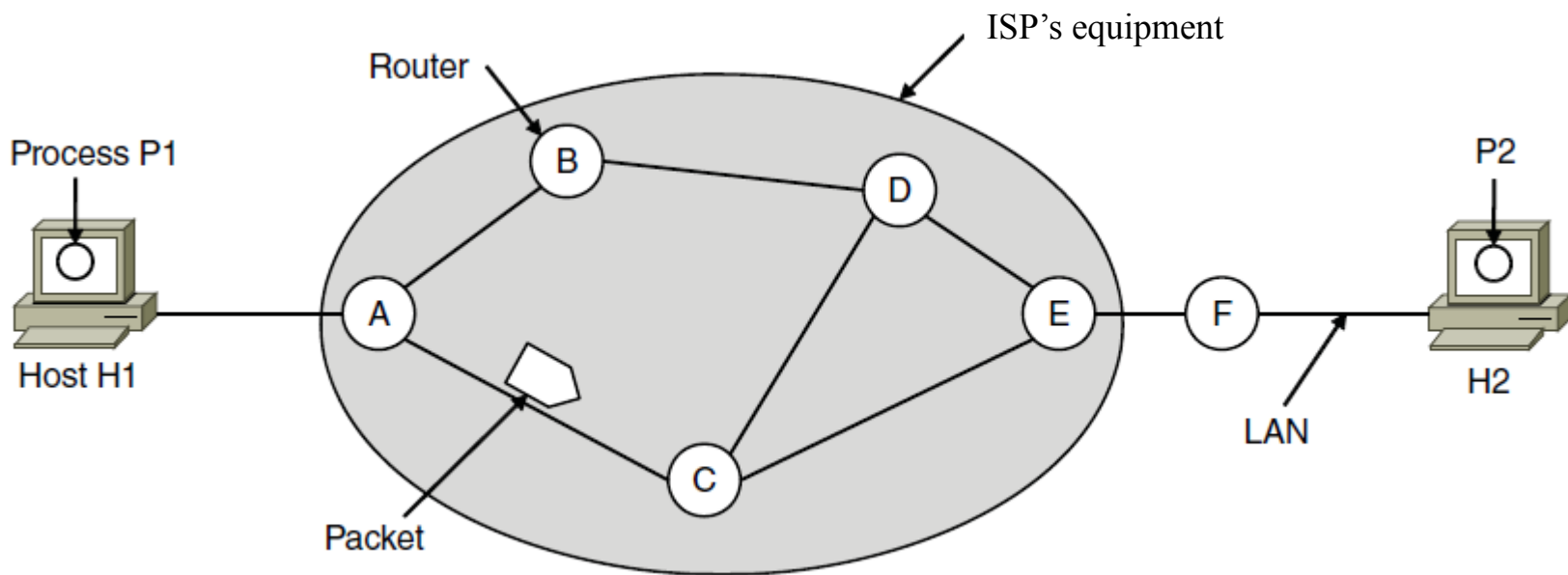
# 虚电路是逻辑连接

- 虚电路表示这只是一条**逻辑上的连接**，分组都沿着这条逻辑连接按照存储转发方式传送，而并不是真正建立了一条物理连接。
- 请注意，电路交换的电话通信是先建立了一条**真正的连接**。因此分组交换的虚连接和电路交换的连接只是类似，但并不完全一样。



# 无连接：包转发

Hosts 将数据包发到网络，网络中的路由  
器转发数据包，最终到达目的地

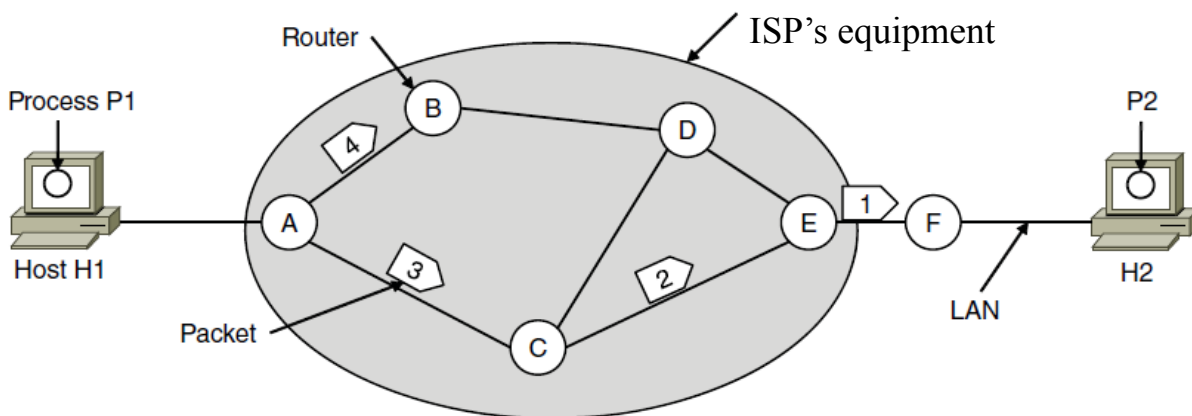




# 无连接：包转发

■ 数据包的转发是通过数据包携带的目的地址来实现的

- 不同的数据包可能会经历不同的路径到达目的地



A's table (initially)

Table	
A	
B	B
C	C
D	B
E	C
F	C
Dest.	Line

A's table (later)

A	
B	B
C	C
D	B
E	D
F	

B

C's Table

A	A
B	A
C	
D	E
E	E
F	E

E's

A	C
B	D
C	C
D	D
E	
F	F





## 因特网采用的设计思路

- 网络层向上只提供简单灵活的、**无连接的、尽最大努力交付的数据报服务**。
- 网络在发送分组时不需要先建立连接。每一个分组（即 IP 数据报）独立发送，与其前后的分组无关（不进行编号）。
- 网络层不提供服务质量的承诺。即所传送的分组可能出错、丢失、重复和失序（不按序到达终点），当然也不保证分组传送的时限。



## 尽最大努力交付的好处

- 由于传输网络不提供端到端的可靠传输服务，这就使网络中的路由器可以做得比较简单，而且价格低廉（与电信网的交换机相比较）。
- 如果主机（即端系统）中的进程之间的通信需要是可靠的，那么就由网络的主机中的运输层负责（包括差错处理、流量控制等）。
- 采用这种设计思路的好处是：网络的造价大大降低，运行方式灵活，能够适应多种应用。
- 因特网能够发展到今日的规模，充分证明了当初采用这种设计思路的正确性。



虚电路服务与数据报服务的对比

对比的方面	虚电路服务	数据报服务
思路	可靠通信应当由网络来保证	可靠通信应当由用户主机来保证
连接的建立	必须有	不需要
终点地址	仅在连接建立阶段使用，每个分组使用短的虚电路号	每个分组都有终点的完整地址
分组的转发	属于同一条虚电路的分组均按照同一路由进行转发	每个分组独立选择路由进行转发
当结点出故障时	所有通过出故障的结点的虚电路均不能工作	出故障的结点可能会丢失分组，一些路由可能会发生变化
分组的顺序	总是按发送顺序到达终点	到达终点时不一定按发送顺序
端到端的差错处理和流量控制	可以由网络负责，也可以由用户主机负责	由用户主机负责



## 4.2 网际协议 IP

---

- 4.2.1 虚拟互连网络
- 4.2.2 分类的 IP 地址
- 4.2.3 IP 地址与硬件地址
- 4.2.4 地址解析协议 ARP
- 4.2.5 IP 数据报的格式
- 4.2.6 IP 层转发分组的流程

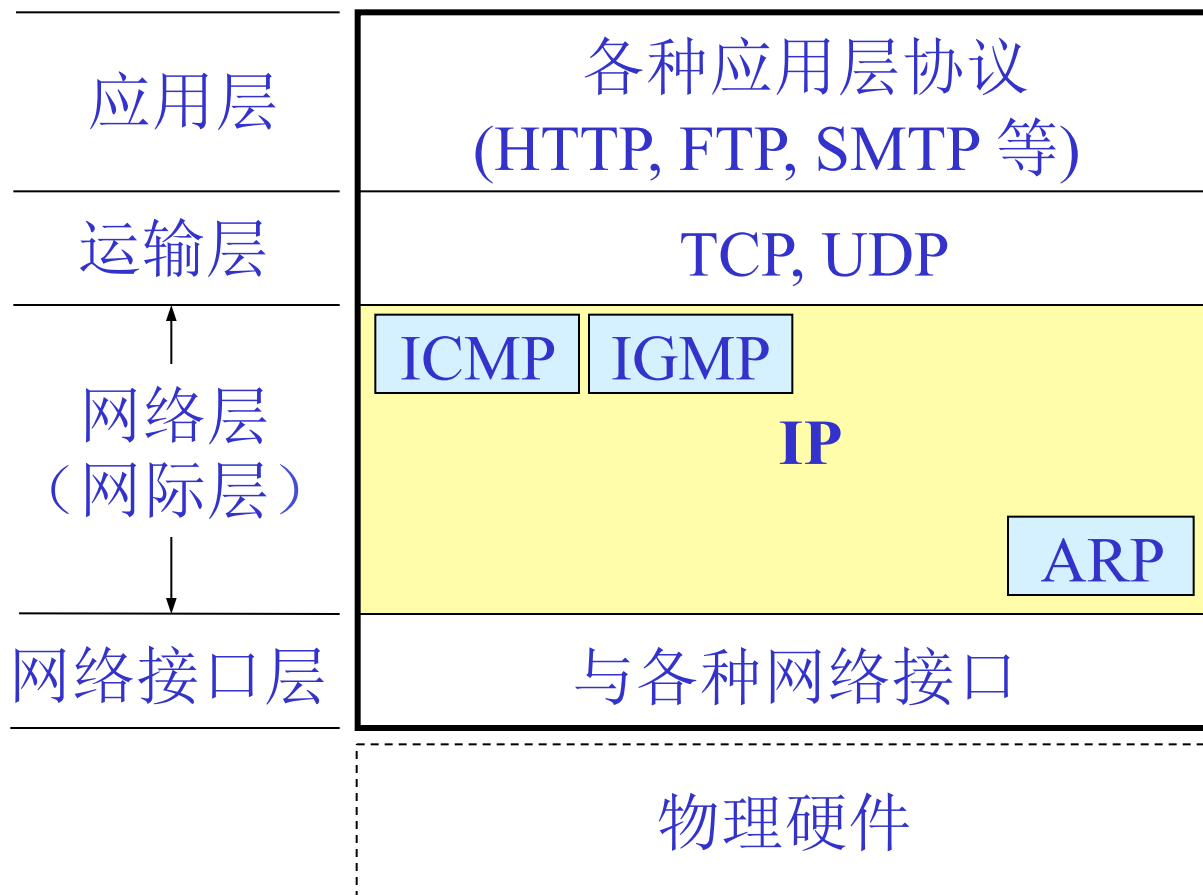


## 4.2 网际协议 IP

- 网际协议 IP 是 TCP/IP 体系中两个最主要的协议之一。
- 与 IP 协议配套使用的还有三个协议：
  - 地址解析协议 ARP  
(Address Resolution Protocol)
  - 网际控制报文协议 ICMP  
(Internet Control Message Protocol)
  - 网际组管理协议 IGMP  
(Internet Group Management Protocol)



# 网际层的 IP 协议及配套协议





## 4.2.1 虚拟互连网络

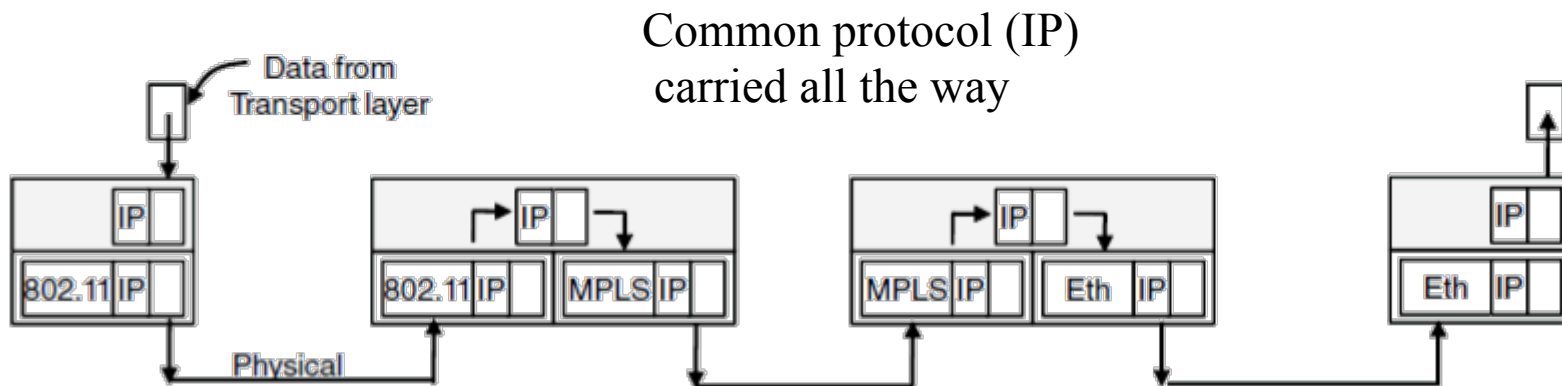
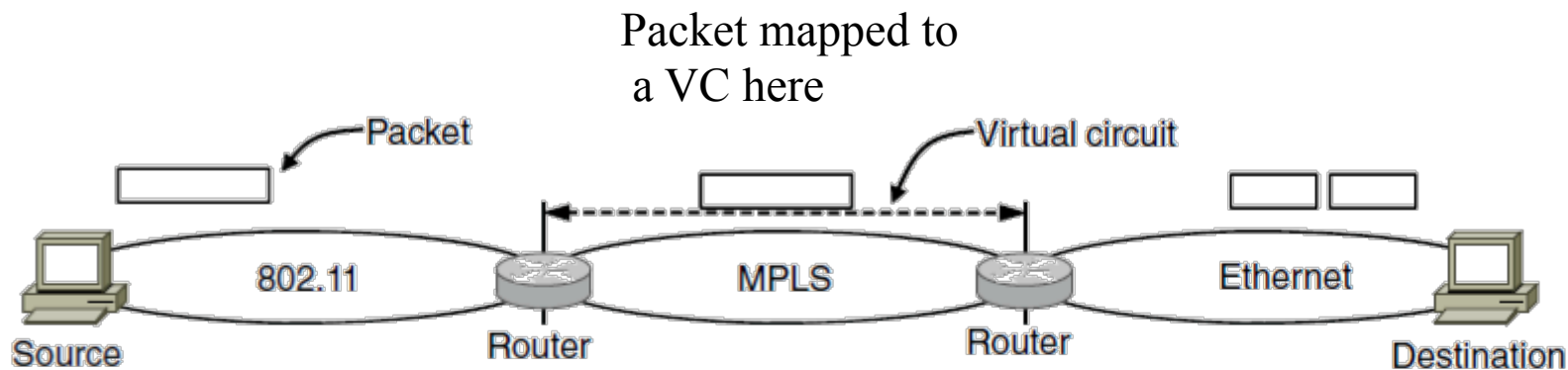
- 将网络互连并能够互相通信，会遇到许多问题需要解决，如：
  - 不同的寻址方案
  - 不同的最大分组长度
  - 不同的网络接入机制
  - 不同的超时控制
  - 不同的差错恢复方法
  - 不同的状态报告方法
  - 不同的路由选择技术
  - 不同的用户接入控制
  - 不同的服务（面向连接服务和无连接服务）
  - 不同的管理与控制方式等

**如何将异构的网络  
互相连接起来？  
网络基于相同的  
网络层-IP**



# 网络互联

网络互联基于相同的网络层 – IP







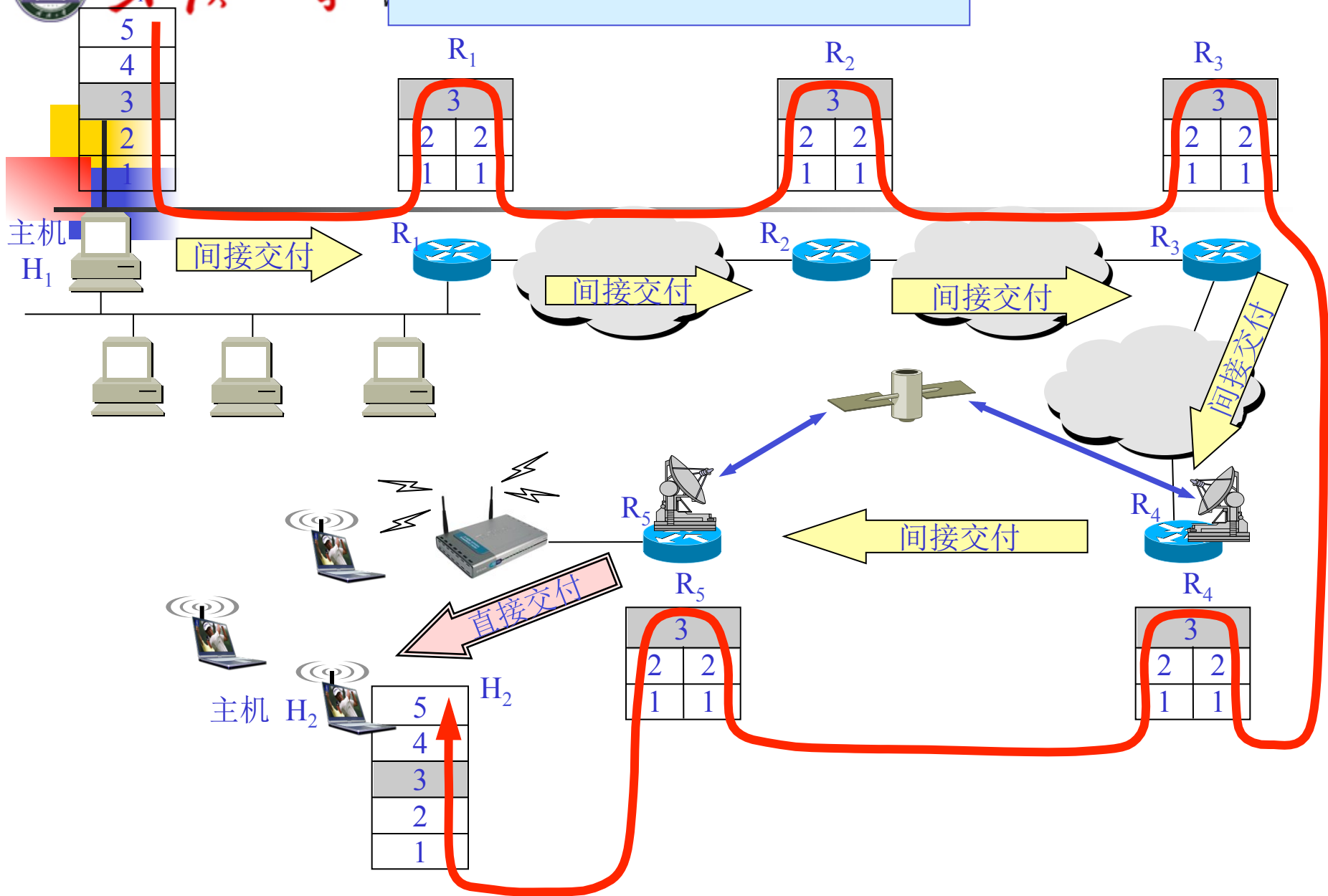
# 虚拟互连网络的意义

- 所谓虚拟互连网络也就是逻辑互连网络，它的意思就是互连起来的各种物理网络的异构性本来是客观存在的，但是我们利用 IP 协议就可以使这些性能各异的网络从用户看起来好像是一个统一的网络。
- 使用 IP 协议的虚拟互连网络可简称为 IP 网。
- 使用虚拟互连网络的好处是：当互联网上的主机进行通信时，就好像在一个网络上通信一样，而看不见互连的各具体的网络异构细节。



武汉大学

# 分组在互联网中的传送





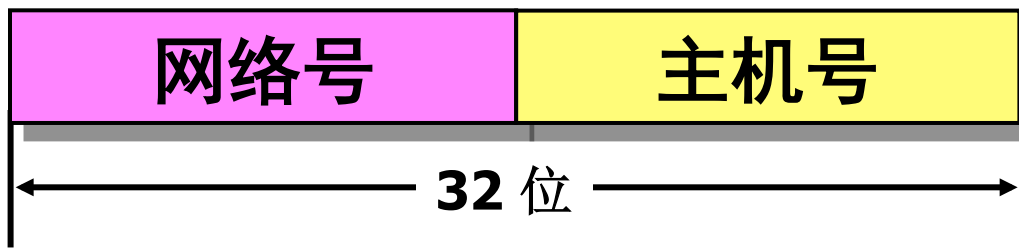
# 1. IP 地址及其表示方法

- 我们把整个因特网看成为一个单一的、抽象的网络。
- IP 地址就是给每个连接在互联网上的主机（或路由器）分配一个在全世界范围是唯一的 32 位的标识符。
- IP 地址现在由互联网名字和数字分配机构 ICANN (Internet Corporation for Assigned Names and Numbers) 进行分配。



## 分类 IP 地址

- 这种两级的 IP 地址结构如下：



**IP 地址 ::= { <网络号>, <主机号> }      (4-1)**

**::= 代表 “定义为”**

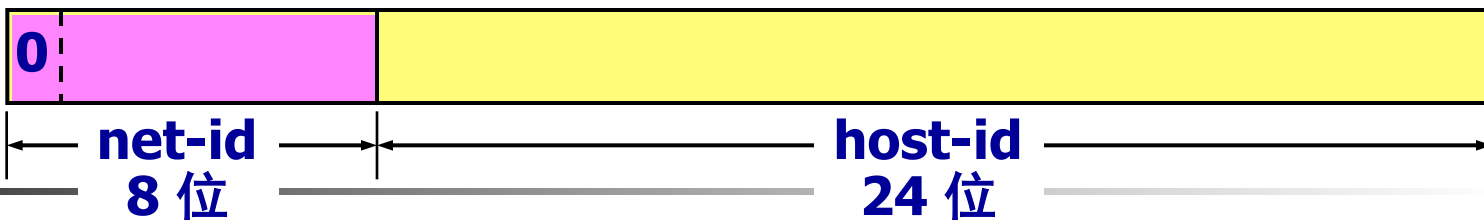


武汉大学

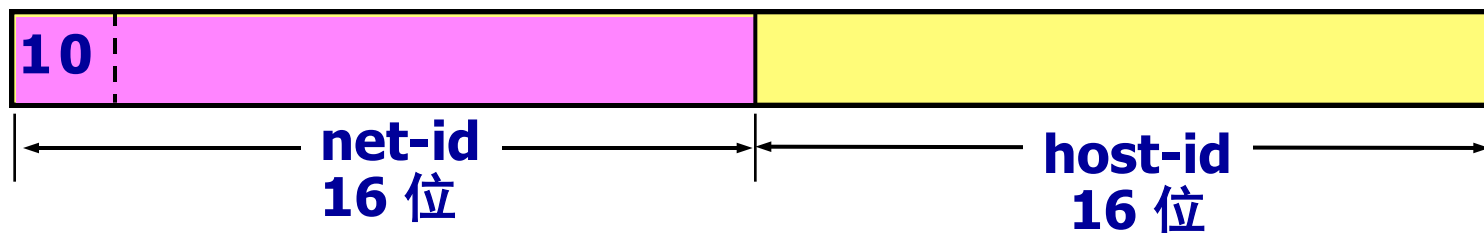
WUHAN UNIVERSITY

# 各类 IP 地址的网络号字段和主机号字段

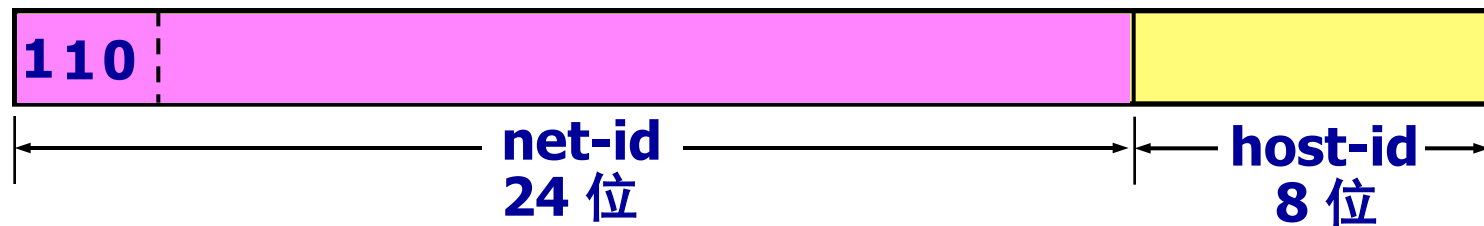
A 类地址



B 类地址



C 类地址



D 类地址



E 类地址





# 点分十进制记法

机器中存放的 IP 地址  
是 32 位 二进制代码

10000000000010110000001100011111

每隔 8 位插入一个空格  
能够提高可读性

10000000 00001011 00000011 00011111

将每 8 位的二进制数  
转换为十进制数

128

11

3

31

采用点分十进制记法  
则进一步提高可读性

128.11.3.31



## 2. 常用的三种类别的 IP 地址

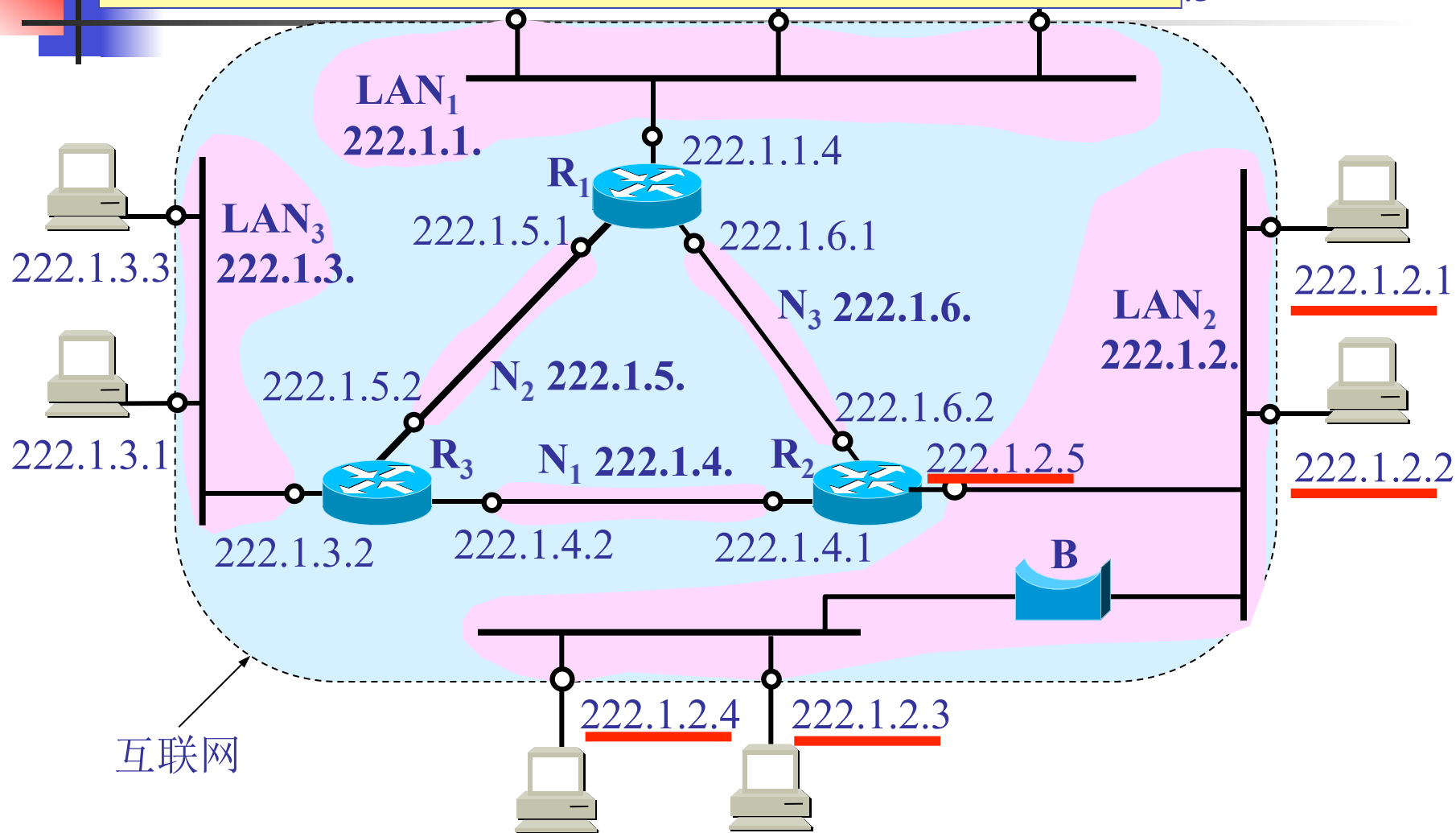
### IP 地址的指派范围

网络类别	最大可指派的网络数	第一个可指派的网络号	最后一个可指派的网络号	每个网络中最大主机数
<b>A</b>	<b><math>126 (2^7 - 2)</math></b>	<b>1</b>	<b>126</b>	<b>16777214</b>
<b>B</b>	<b><math>16383 (2^{14} - 1)</math></b>	<b>128.1</b>	<b>191.255</b>	<b>65534</b>
<b>C</b>	<b><math>2097151 (2^{21} - 1)</math></b>	<b>192.0.1</b>	<b>223.255.255</b>	<b>254</b>



在同一个局域网上的主机或路由器的  
IP 地址中的网络号必须是一样的。  
图中的网络号就是 IP 地址中的 net-id

.3

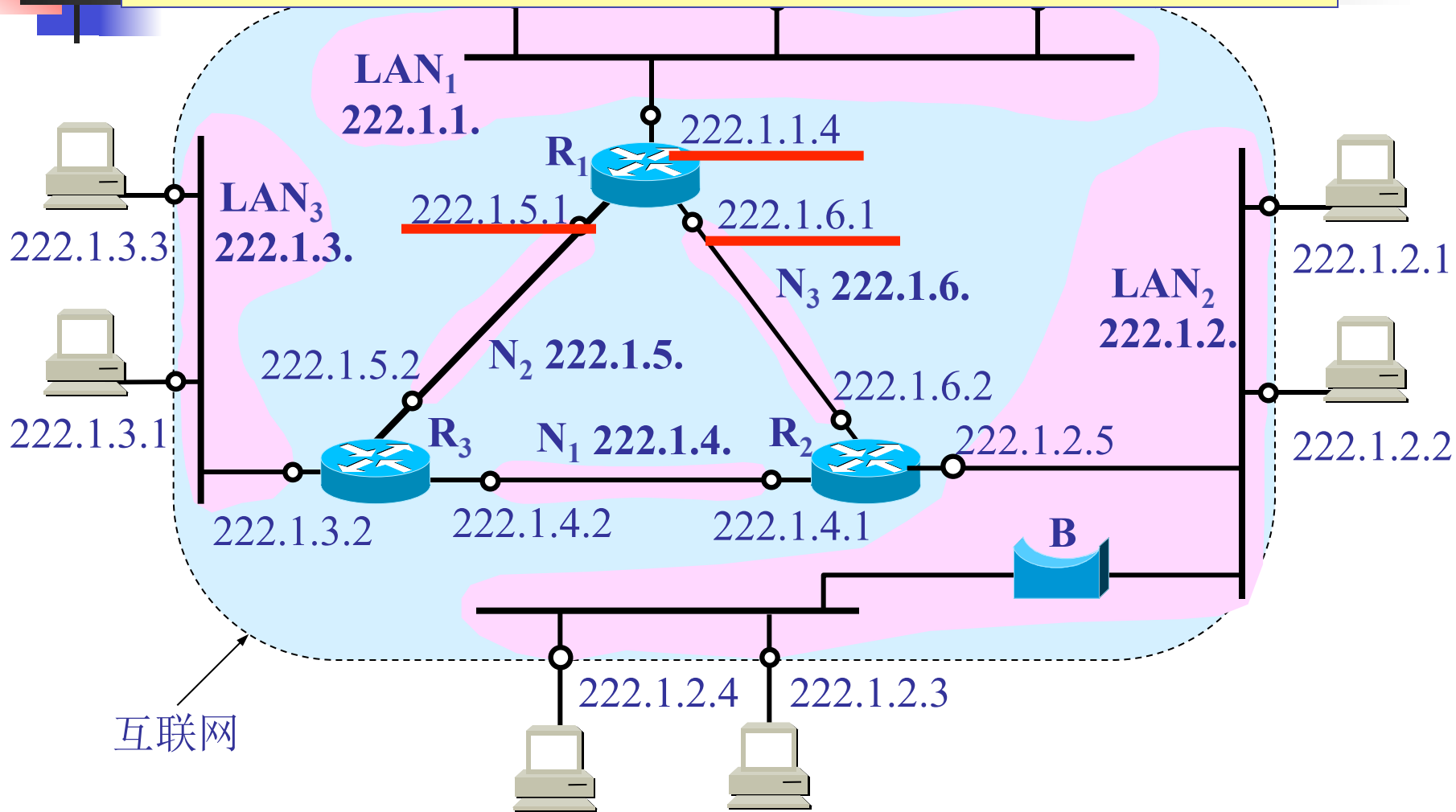






清华大学

路由器总是具有两个或两个以上的 IP 地址。  
路由器的每一个接口都有一个  
不同网络号的 IP 地址。



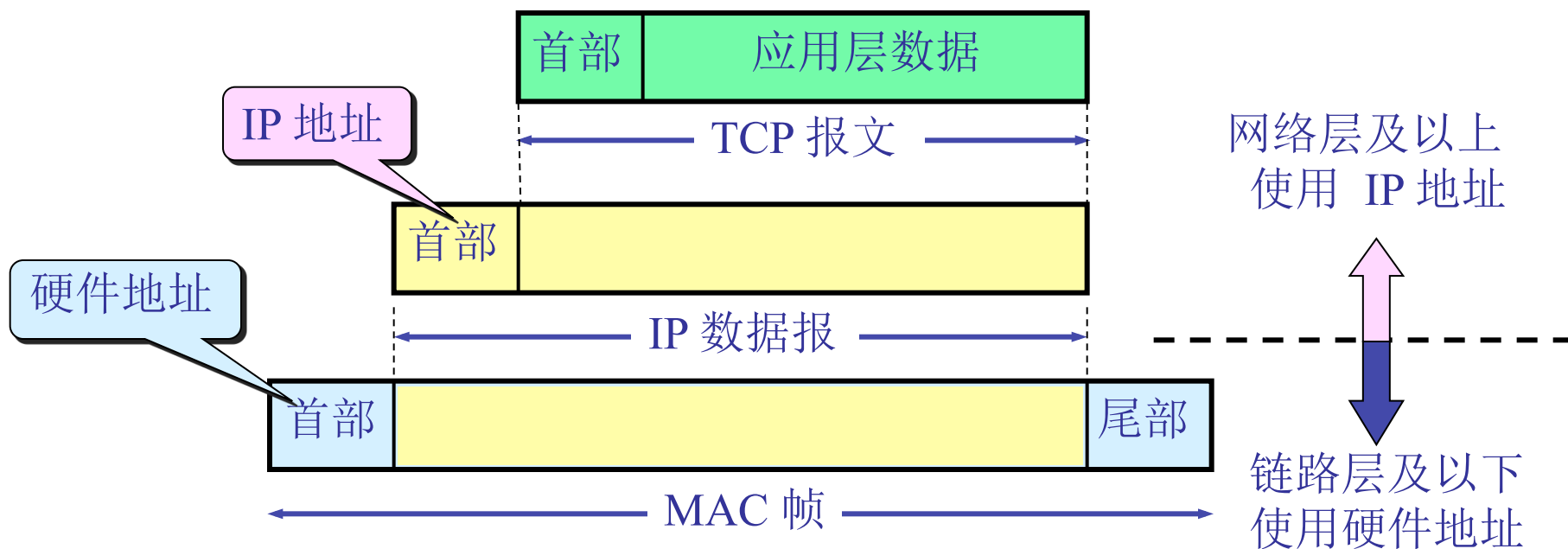


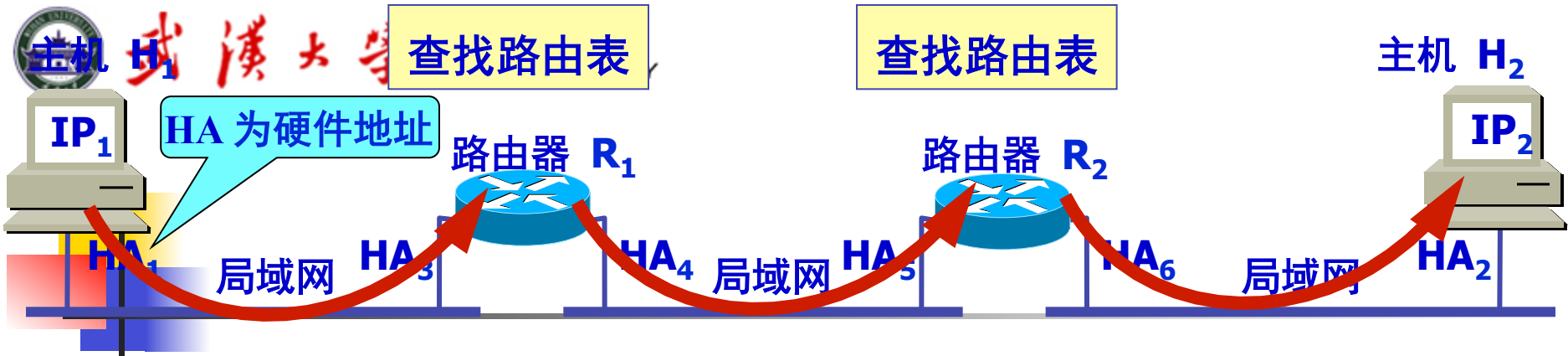
## 4.2.3 IP 地址与硬件地址

- IP 地址与硬件地址是不同的地址。
- 从层次的角度看，
  - 硬件地址（或物理地址）是数据链路层和物理层使用的地址。
  - IP 地址是网络层和以上各层使用的地址，是一种逻辑地址（称 IP 地址是逻辑地址是因为 IP 地址是用软件实现的）。



## 4.2.3 IP 地址与硬件地址

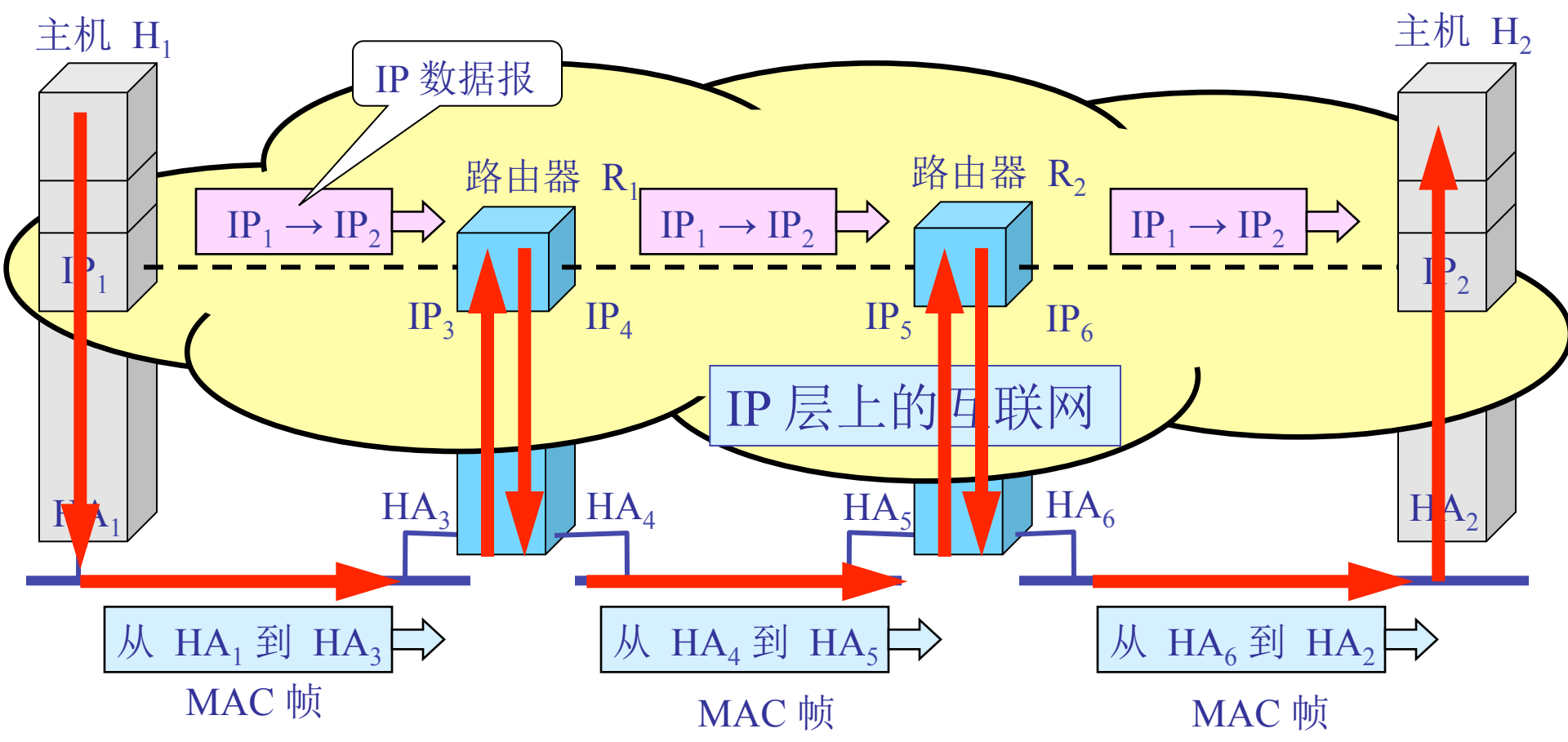




通信的路径：  
H<sub>1</sub> → 经过 R<sub>1</sub> 转发 → 再经过 R<sub>2</sub> 转发 → H<sub>2</sub>



# 从协议栈的层次上看数据的流动





武汉大学

WUHAN UNIVERSITY

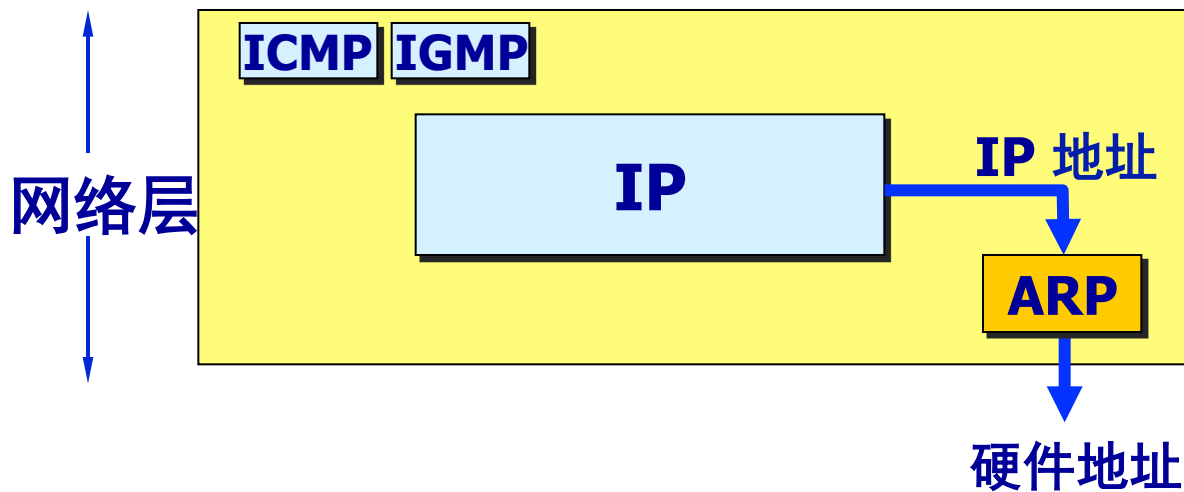
# 主机 $H_1$ 与 $H_2$ 通信中使用的 IP地址 与 硬件地址 HA

	在网络层 写入 <b>IP</b> 数据报首部的地址		在数据链路层 写入 <b>MAC</b> 帧首部的地址	
	源地址	目的地址	源地址	目的地址
从 $H_1$ 到 $R_1$	<b><math>IP_1</math></b>	<b><math>IP_2</math></b>	<b><math>HA_1</math></b>	<b><math>HA_3</math></b>
从 $R_1$ 到 $R_2$	<b><math>IP_1</math></b>	<b><math>IP_2</math></b>	<b><math>HA_4</math></b>	<b><math>HA_5</math></b>
从 $R_2$ 到 $H_2$	<b><math>IP_1</math></b>	<b><math>IP_2</math></b>	<b><math>HA_6</math></b>	<b><math>HA_2</math></b>



# 地址解析协议 ARP 的作用

- 已经知道了一个机器（主机或路由器）的IP地址，如何找出其相应的硬件地址？
- 地址解析协议 ARP 就是用来解决这样的问题的。



ARP 协议的作用

**ARP 作用：**  
从网络层使用的 IP 地址，解析出在数据链路层使用的硬件地址。



# 地址解析协议 ARP 要点

- 不管网络层使用的是什麼协议，在实际网络的链路上传送数据帧时，最终还是必须使用硬件地址。
- 每一个主机都设有一个 **ARP 高速缓存 (ARP cache)**，里面有所在的局域网上的各主机和路由器的 **IP 地址到硬件地址的映射表**。

**< IP address; MAC address; TTL >**

**TTL (Time To Live): 地址映射有效时间 。**





# 地址解析协议 ARP 要点

- 当主机 A 欲向本局域网上的某个主机 B 发送 IP 数据报时，就先在其 ARP 高速缓存中查看有无主机 B 的 IP 地址。
  - 如有，就可查出其对应的硬件地址，再将此硬件地址写入 MAC 帧，然后通过局域网将该 MAC 帧发往此硬件地址。
  - 如没有，ARP 进程在本局域网上广播发送一个 ARP 请求分组。收到 ARP 响应分组后，将得到的 IP 地址到硬件地址的映射写入 ARP 高速缓存。

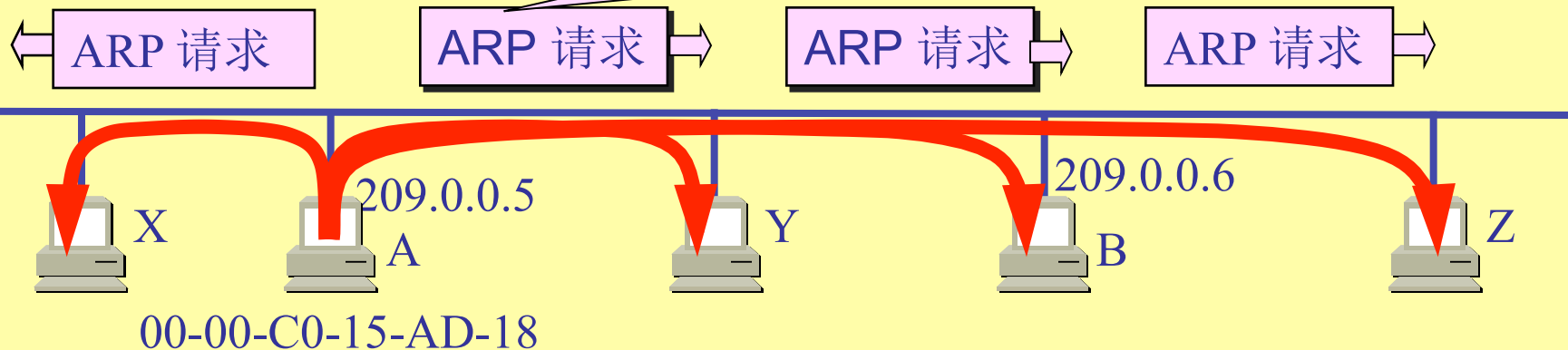


# 地址解析协议 ARP 要点

- **ARP请求分组**：包含发送方硬件地址 / 发送方 IP 地址 / **目标方硬件地址(未知时填0)** / 目标方 IP 地址。
- **本地广播 ARP 请求**（路由器不转发ARP请求）。
- **ARP 响应分组**：包含发送方硬件地址 / 发送方 IP地址 / 目标方硬件地址 / 目标方 IP 地址。
- **ARP 分组封装在物理网络的帧中传输。**

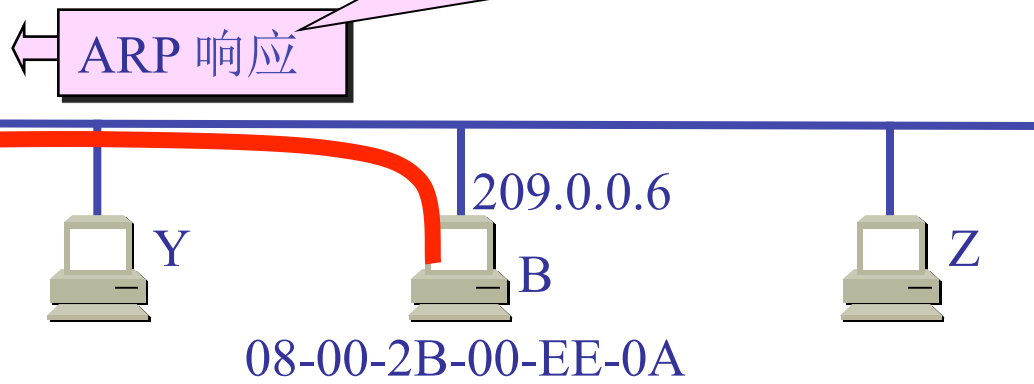
主机 A 广播发送  
ARP 请求分组

我是 209.0.0.5，硬件地址是 00-00-C0-15-AD-18  
我想知道主机 209.0.0.6 的硬件地址



主机 B 向 A 发送  
ARP 响应分组

我是 209.0.0.6  
硬件地址是 08-00-2B-00-EE-0A





# ARP 高速缓存的作用

- 存放最近获得的 IP 地址到 MAC 地址的绑定，以减少 ARP 广播的数量。
- 为了减少网络上的通信量，主机 A 在发送其 ARP 请求分组时，就将自己的 IP 地址到硬件地址的映射写入 ARP 请求分组。
- 当主机 B 收到 A 的 ARP 请求分组时，就将主机 A 的这一地址映射写入主机 B 自己的 ARP 高速缓存中。这对主机 B 以后向 A 发送数据报时就更方便了。



## 应当注意的问题

- ARP 是解决同一个局域网上的主机或路由器的 IP 地址和硬件地址的映射问题。
- 如果所要找的主机和源主机不在同一个局域网，那么就要通过 ARP 找到一个位于本局域网上的某个路由器的硬件地址，然后把分组发送给这个路由器，让这个路由器把分组转发给下一个网络。剩下的工作就由下一个网络来做。



## 应当注意的问题（续）

- 从IP地址到硬件地址的解析是自动进行的，主机的用户对这种地址解析过程是不知道的。
- 只要主机或路由器要和本网络上的另一个已知 IP 地址的主机或路由器进行通信，ARP 协议就会自动地将该 IP 地址解析为链路层所需要的硬件地址。



# 什么我们不直接使用硬件地址进行通信？

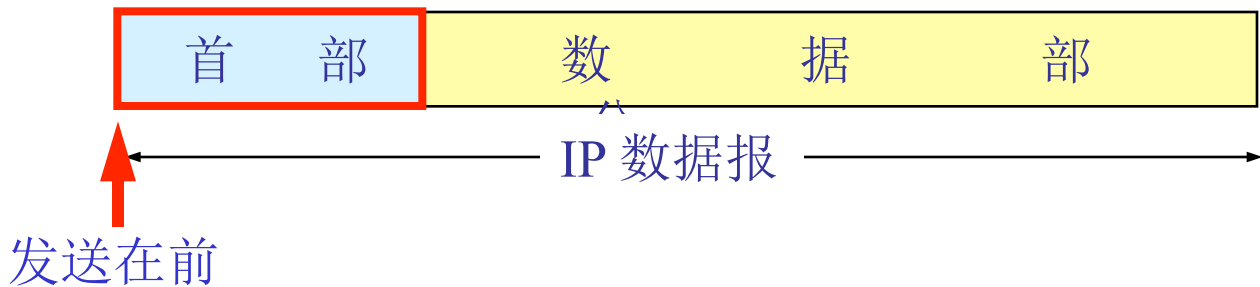
- 由于全世界存在着各式各样的网络，它们使用不同的硬件地址。要使这些异构网络能够互相通信就必须进行非常复杂的硬件地址转换工作，因此几乎是不可能的事。
- 连接到因特网的主机都拥有统一的 IP 地址，解决上述问题。
- 无法解决移动性问题
- 造成大量的路由信息

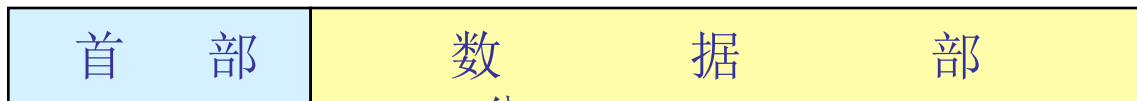


## 4.2.5 IP 数据报的格式

- 一个 IP 数据报由首部和数据两部分组成。
- 首部的前一部分是固定长度，共 20 字节，是所有 IP 数据报必须具有的。
- 在首部的固定部分的后面是一些可选字段，其长度是可变的。

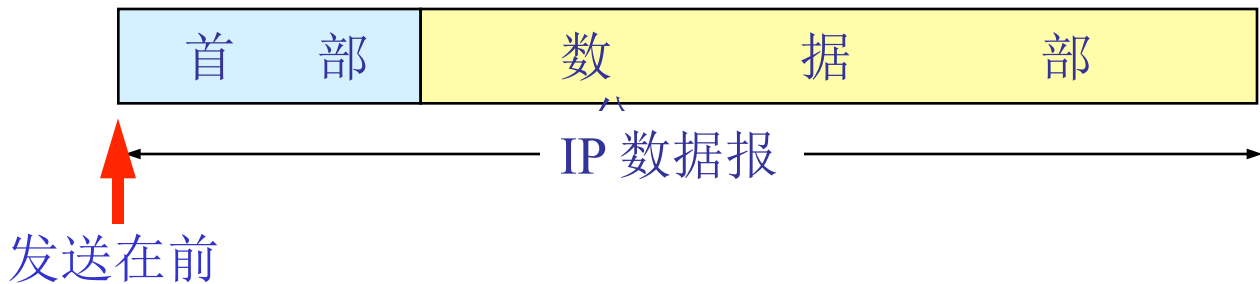






IP 数据报

发送在前





# 1. IP 数据报首部的固定部分中的各字段



版本——占 4 位，指 IP 协议的版本  
目前的 IP 协议版本号为 4 (即 IPv4)



首部长度——占 4 位，可表示的最大数值是 15 个单位(一个单位为 4 字节)  
因此 IP 的首部长度的最大值是 60 字节。



区分服务——占 8 位，用来获得更好的服务  
在旧标准中叫做服务类型，但实际上一直未被使用过。  
1998 年这个字段改名为区分服务。  
只有在使用区分服务（DiffServ）时，这个字段才起作用。  
在一般的情况下都不使用这个字段



总长度——占 16 位，指首部和数据之和的长度，单位为字节，因此数据报的最大长度为 65535 字节。  
总长度必须不超过最大传送单元 MTU。



标识(identification) 占 16 位，  
它是一个计数器，用来产生数据报的标识。





标志(flag) 占 3 位，目前只有前两位有意义。

标志字段的最低位是 **MF** (More Fragment)。

**MF** = 1 表示后面“还有分片”。**MF** = 0 表示最后一个分片。

标志字段中间的一位是 **DF** (Don't Fragment) 。

只有当 **DF** = 0 时才允许分片。



片偏移(12 位)指出：较长的分组在分片后  
某片在原分组中的相对位置。  
片偏移以 8 个字节为偏移单位。



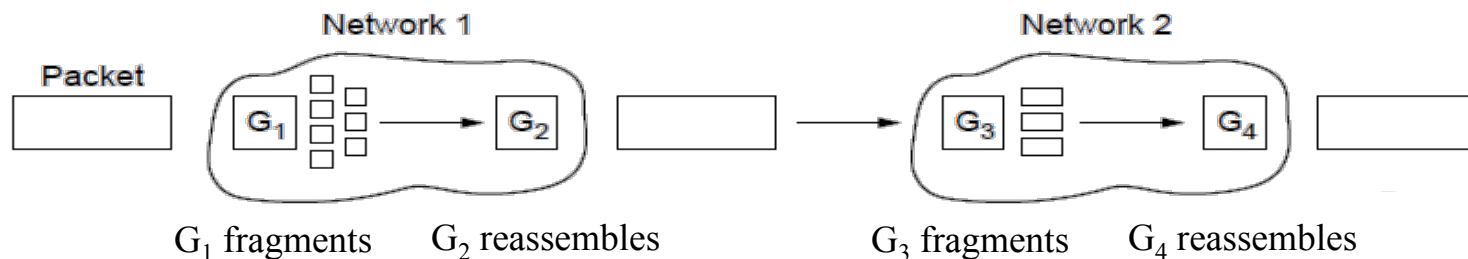
# 数据包分段 (1)

- 现在看一下如下场景
  - 一个大小为 $a$ 的数据包希望进入一个最大只允许大小为 $b$  ( $a > b$ ) 的数据包的网络
- 解决方案:
  - 先找出从源到目的路径上所允许的最大包大小(MTU)
    - 问题是包有可以经由不同的路径
  - 采用分段/重组 fragmentation & reassembly 技术

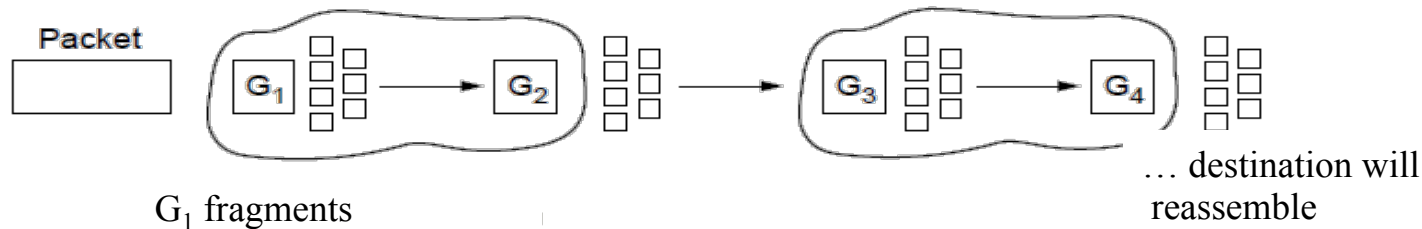


## 数据包分段 (2)

- 分段/重组 fragmentation & reassembly
- 透明：分段/重组都在中间网络网络完成 (数据包必须经由相同的路径)
- 非透明：分段在路由器中进行，但重组在目的节点完成



Transparent – packets fragmented / reassembled in each network



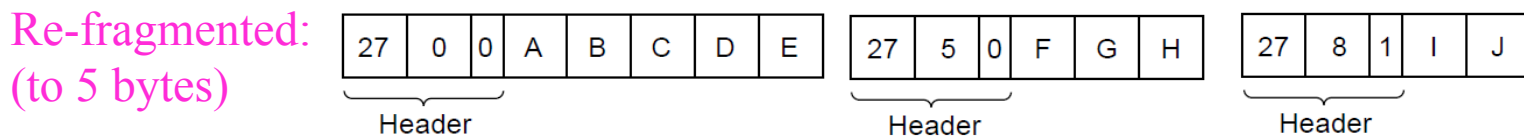
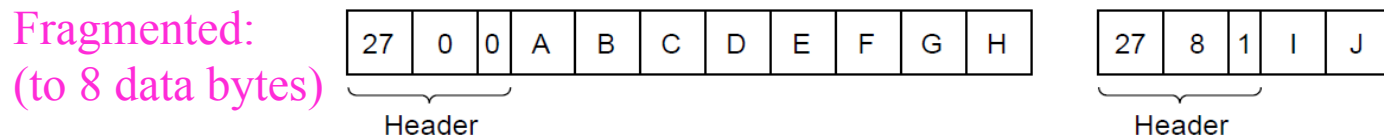
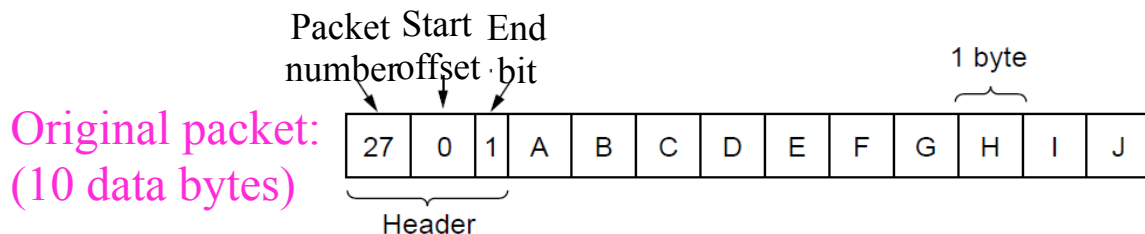
Non-transparent – fragments are reassembled at destination



Packet number=>标识  
Start offset=>偏移  
量（注：IP的偏移量  
是以8个字节为单位）  
End bit=>MF“的否”

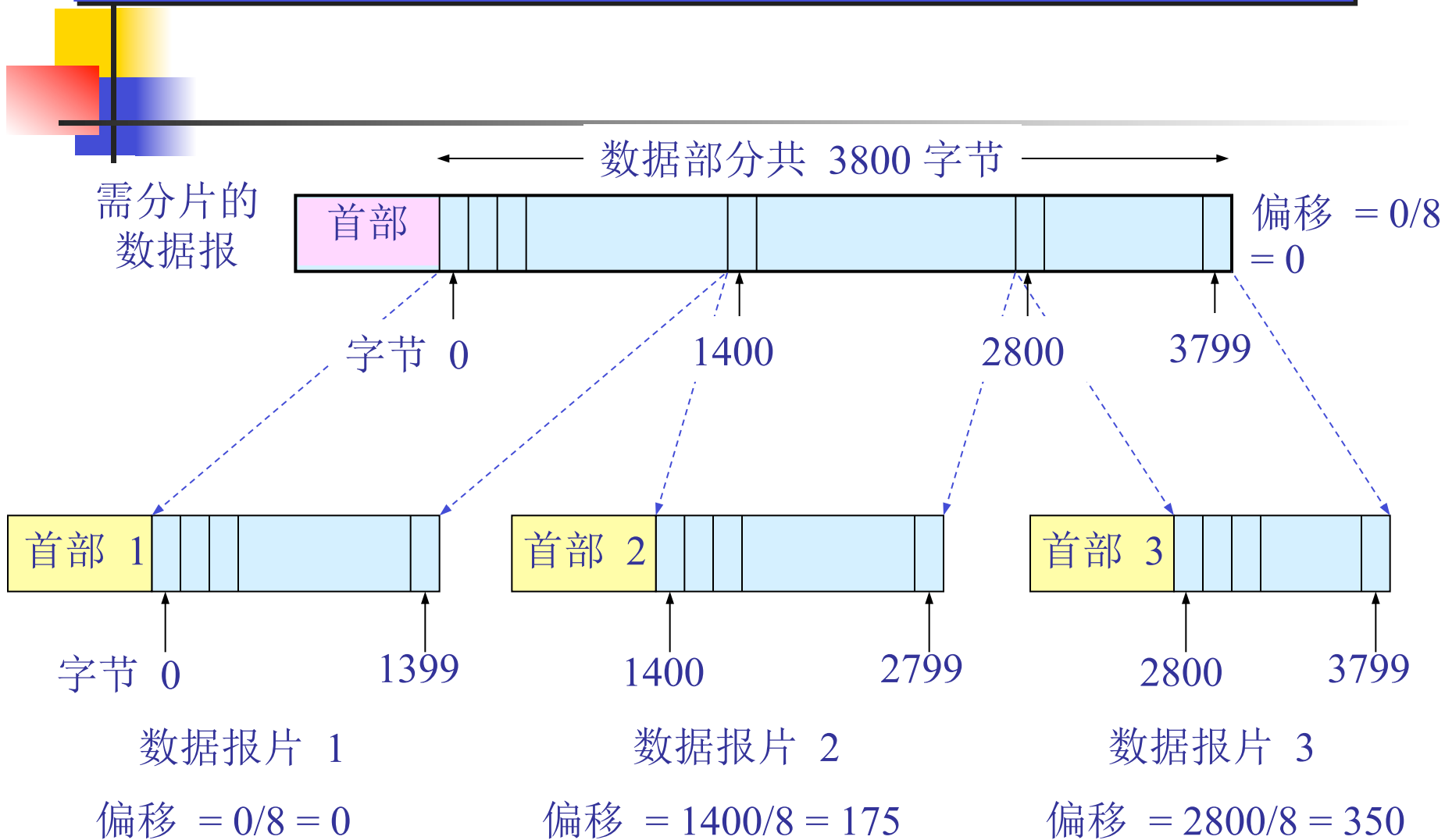
## 数据包分段 (3)

- 怎么分段呢？ 看一个IP包的分段：
  - Start offset和End bit域





# 【例4-1】 IP 数据报分片





# 【例4-1】 IP 数据报分片

IP 数据报首部中与分片有关的字段中的数值

	总长度	标识	MF	DF	片偏移
原始数据报	3820	12345	0	0	0
数据报片1	1420	12345	1	0	0
数据报片2	1420	12345	1	0	175
数据报片3	1020	12345	0	0	350



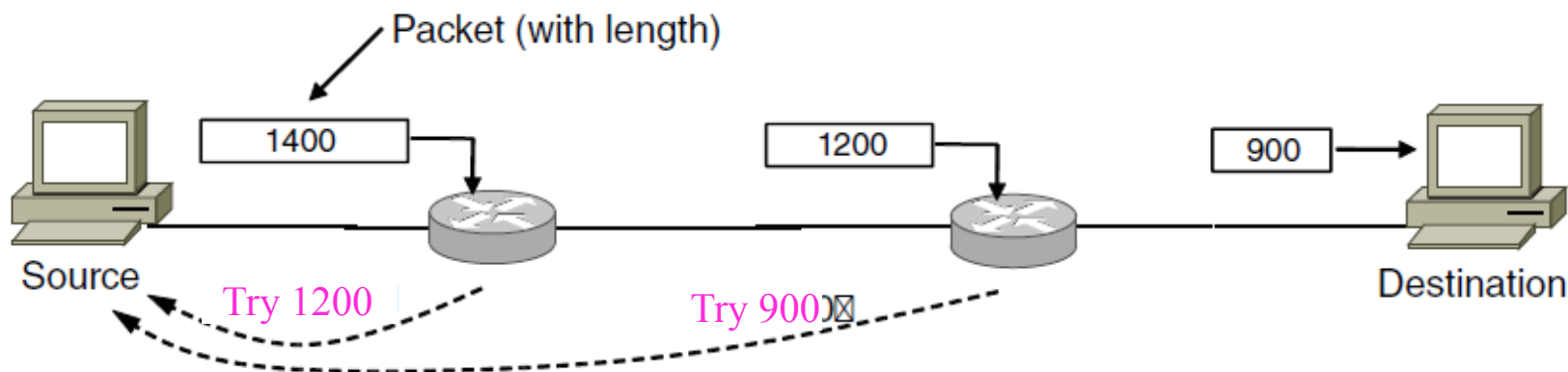
## 数据包分段 (4)

分段的问题:

- 只要一个分段丢掉，整个数据包都要重传
- 添加新的字段

所以我们要避免分段

- 只好再求助与MTU (Max. Transmission Unit) 机制, 但采用不同的策略.
- 源节点按照正常的数据包发送，当这个数据包无法通过某个路由器时，路由器回馈MTU给源并丢掉此数据包，源节点再根据回馈的MTU设置大小



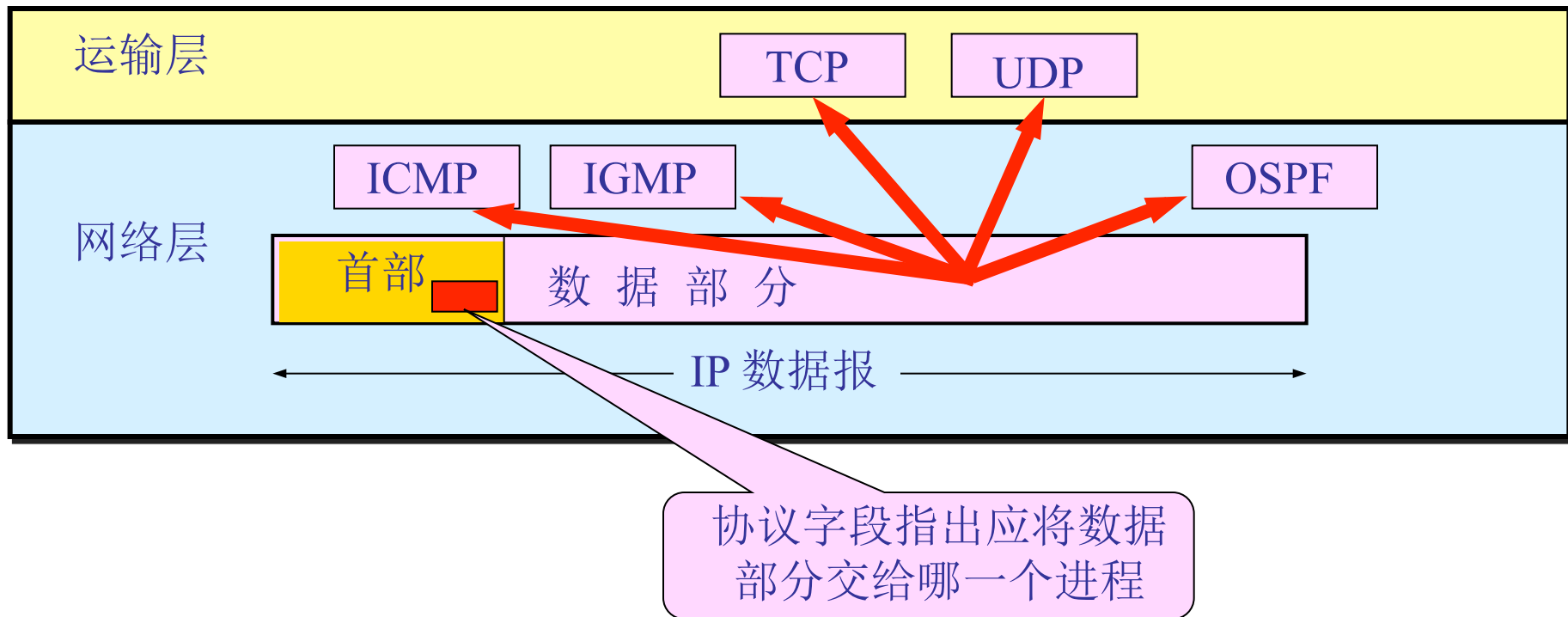
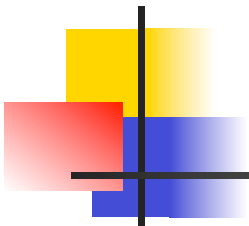




生存时间(8 位)记为 TTL (Time To Live)  
数据报在网络中可通过的路由器数的最大值。



协议(8 位)字段指出此数据报携带的数据使用何种协议以便目的主机的 IP 层将数据部分上交给哪个处理过程





首部检验和(16 位)字段只检验数据报的首部  
不检验数据部分。

这里不采用 CRC 检验码而采用简单的计算方法。



源地址和目的地址都各占 4 字节



## 2. IP 数据报首部的可变部分

- IP 首部的可变部分就是一个选项字段，用来支持排错、测量以及安全等措施，内容很丰富。
- 选项字段的长度可变，从 1 个字节到 40 个字节不等，取决于所选择的项目。
- 增加首部的可变部分是为了增加 IP 数据报的功能，但这同时也使得 IP 数据报的首部长度成为可变的。这就增加了每一个路由器处理数据报的开销。
- 实际上这些选项很少被使用。



# IPv4 协议

0000 45 00 01 d4 30 52 00 00 80 11 a3 da 0a 2f a8 0e

0010 0a 2f a8 80

请说明：版本、头长度、总长度、Identification、fragment offset、TTL、上层协议、源地址、目的地址

```

> Internet Protocol version 4, Src: 10.47.168.14 (10.47.168.14), Dst: 10.47.168.128 (10.47.168.128)
  Version: 4
  Header Length: 20 bytes
  Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    0000 00.. = Differentiated Services Codepoint: Default (0x00)
    .... ..00 = Explicit Congestion Notification: Not-ECT (Not ECN-Capable Transport) (0x00)
  Total Length: 468
  Identification: 0x3052 (12370)
  Flags: 0x00
  Fragment offset: 0
  Time to live: 128
  Protocol: UDP (17)
  Header checksum: 0xa3da [correct]
  Source: 10.47.168.14 (10.47.168.14)
  Destination: 10.47.168.128 (10.47.168.128)

```



## 4.2.6 IP 层转发分组的流程

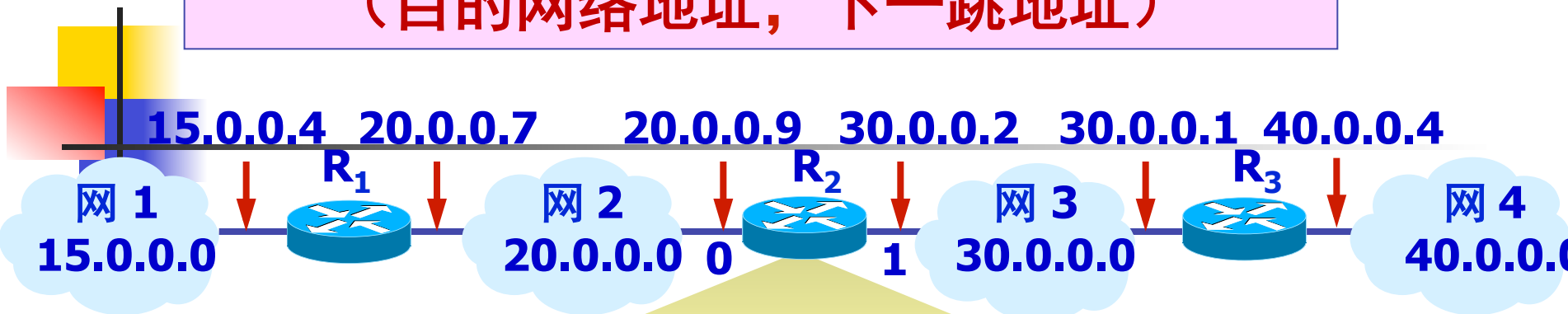
- 假设：有四个 A 类网络通过三个路由器连接在一起。每一个网络上都可能有成千上万个主机。
- 可以想像，若按目的主机号来制作路由表，每一个路由表就有 4 万个项目，即 4 万行（每一行对应于一台主机），则所得出的路由表就会过于庞大。
- 但若按主机所在的网络地址来制作路由表，那么每一个路由器中的路由表就只包含 4 个项目（每一行对应于一个网络），这样就可使路由表大大简化。





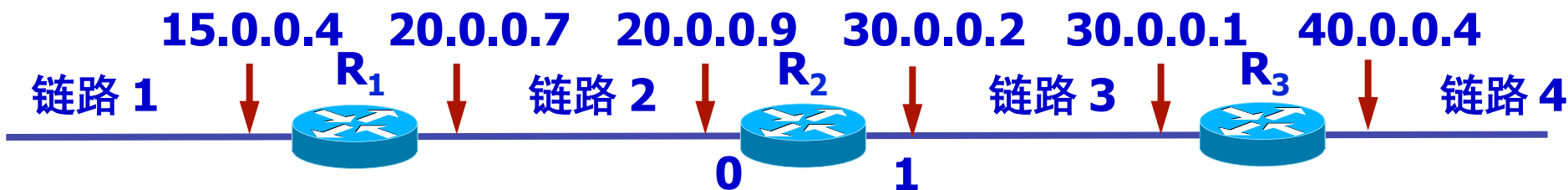
武汉大学

在路由表中，对每一条路由，最主要的是  
(目的网络地址，下一跳地址)



路由器 R<sub>2</sub> 的路由表

目的主机所在的网络	下一跳地址
20.0.0.0	直接交付，接口 0
30.0.0.0	直接交付，接口 1
15.0.0.0	20.0.0.7
40.0.0.0	30.0.0.1





## 关于路由表

- 路由表没有给分组指明到某个网络的完整路径。
- 路由表指出，到某个网络应当先到某个路由器（即下一跳路由器）。
- 在到达下一跳路由器后，再继续查找其路由表，知道再下一步应当到哪一个路由器。
- 这样一步一步地查找下去，直到最后到达目的网络。



## 默认路由(default route)

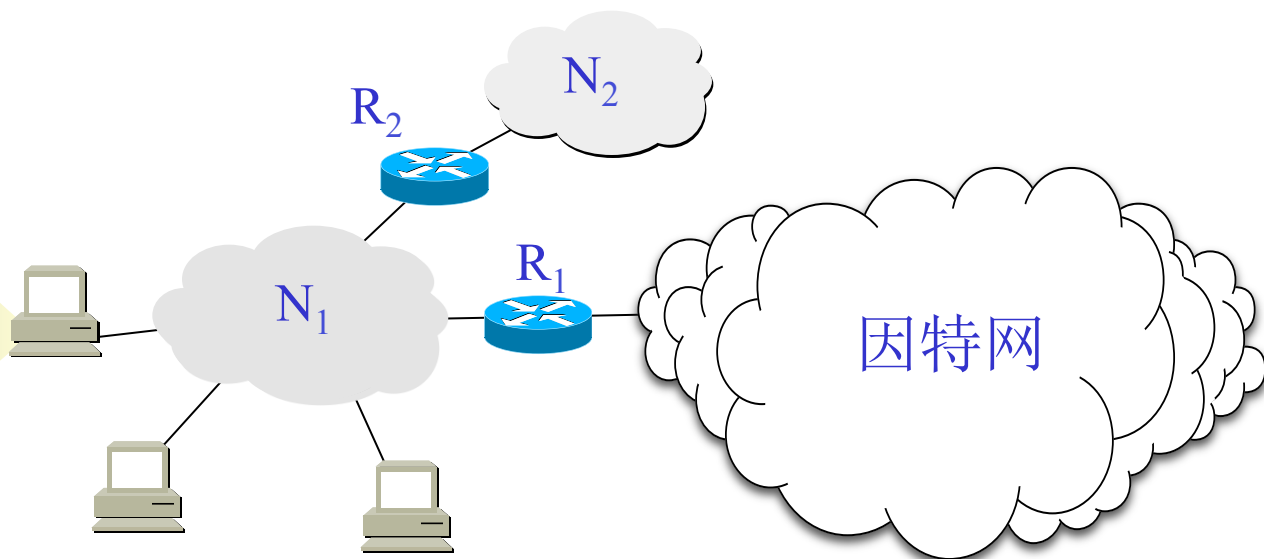
- 路由器还可采用**默认路由**以减少路由表所占用的空间和搜索路由表所用的时间。
- 这种转发方式在一个网络只有很少的对外连接时是很有用的。
- 默认路由在主机发送 IP 数据报时往往更能显示出它的好处。
- 如果一个主机连接在一个小网络上，而这个网络只用一个路由器和因特网连接，那么在这种情况下使用默认路由是非常合适的。



只要目的网络不是  $N_1$  和  $N_2$ ，  
就一律选择默认路由，  
把数据报先间接交付路由器  $R_1$ ，  
让  $R_1$  再转发给下一个路由器。

路由表

目的网络	下一跳
$N_1$	直接
$N_2$	$R_2$
默认	$R_1$





# 分组转发算法

- (1) 从数据报的首部提取目的主机的 IP 地址  $D$ , 得出目的网络地址为  $N$ 。
- (2) 若网络  $N$  与此路由器直接相连, 则把数据报**直接交付**目的主机  $D$ ; 否则是**间接**交付, 执行(3)。
- (3) 若路由表中有目的地址为  $D$  的特定主机路由, 则把数据报传送给路由表中所指明的下一跳路由器; 否则, 执行(4)。
- (4) 若路由表中有到达网络  $N$  的路由, 则把数据报传送给路由表指明的下一跳路由器; 否则, 执行(5)。
- (5) 若路由表中有一个默认路由, 则把数据报传送给路由表中所指明的默认路由器; 否则, 执行(6)。
- (6) 报告转发分组出错。



## 4.3 划分子网和构造超网

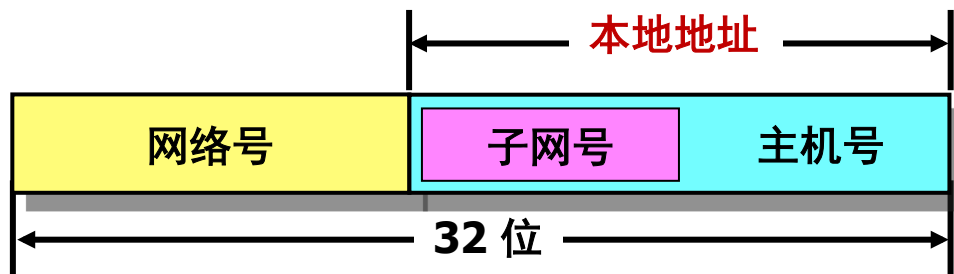
---

- 4.3.1 划分子网
- 4.3.2 使用子网时分组的转发
- 4.3.3 无分类编址 **CIDR**（构造超网）



# 划分子网的基本思路

- 划分子网纯属一个单位内部的事情。单位对外仍然表现为没有划分子网的网络。
- 从主机号借用若干个位作为子网号 subnet-id, 而主机号 host-id 也就相应减少了若干个位。



**IP地址 ::= {<网络号>, <子网号>, <主机号>} (4-2)**



## 划分子网的基本思路（续）

- 凡是从其他网络发送给本单位某个主机的 IP 数据报，仍然是根据 IP 数据报的 **目的网络号 net-id**，先找到连接在 **本单位网络上的路由器**。
- 然后 **此路由器** 在收到 IP 数据报后，再按 **目的网络号 net-id** 和 **子网号 subnet-id** 找到目的子网。
- 最后就将 IP 数据报直接交付目的主机。

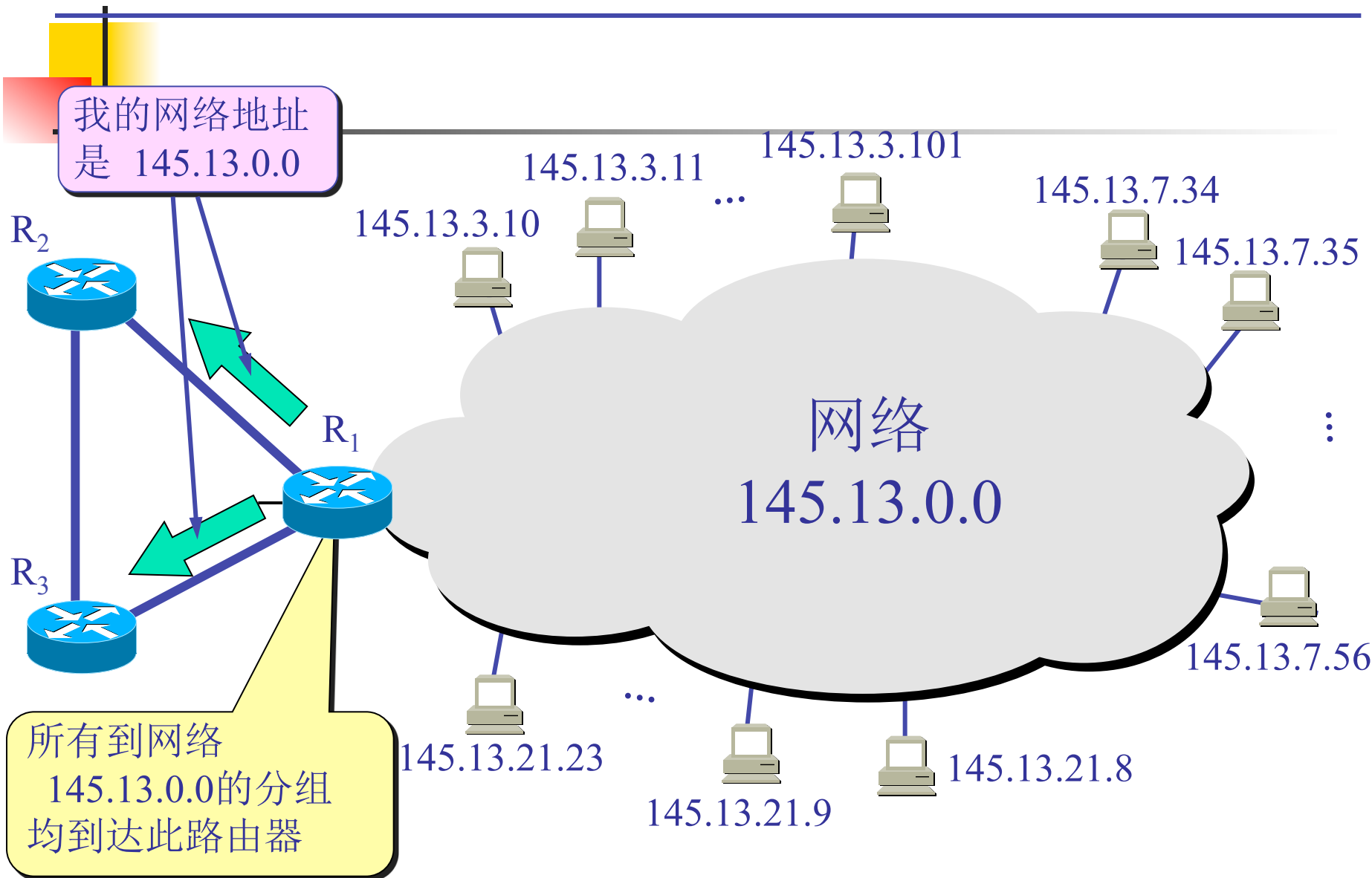




武汉大学

WUHAN UNIVERSITY

# 一个未划分子网的 B 类网络 145.13.0.0



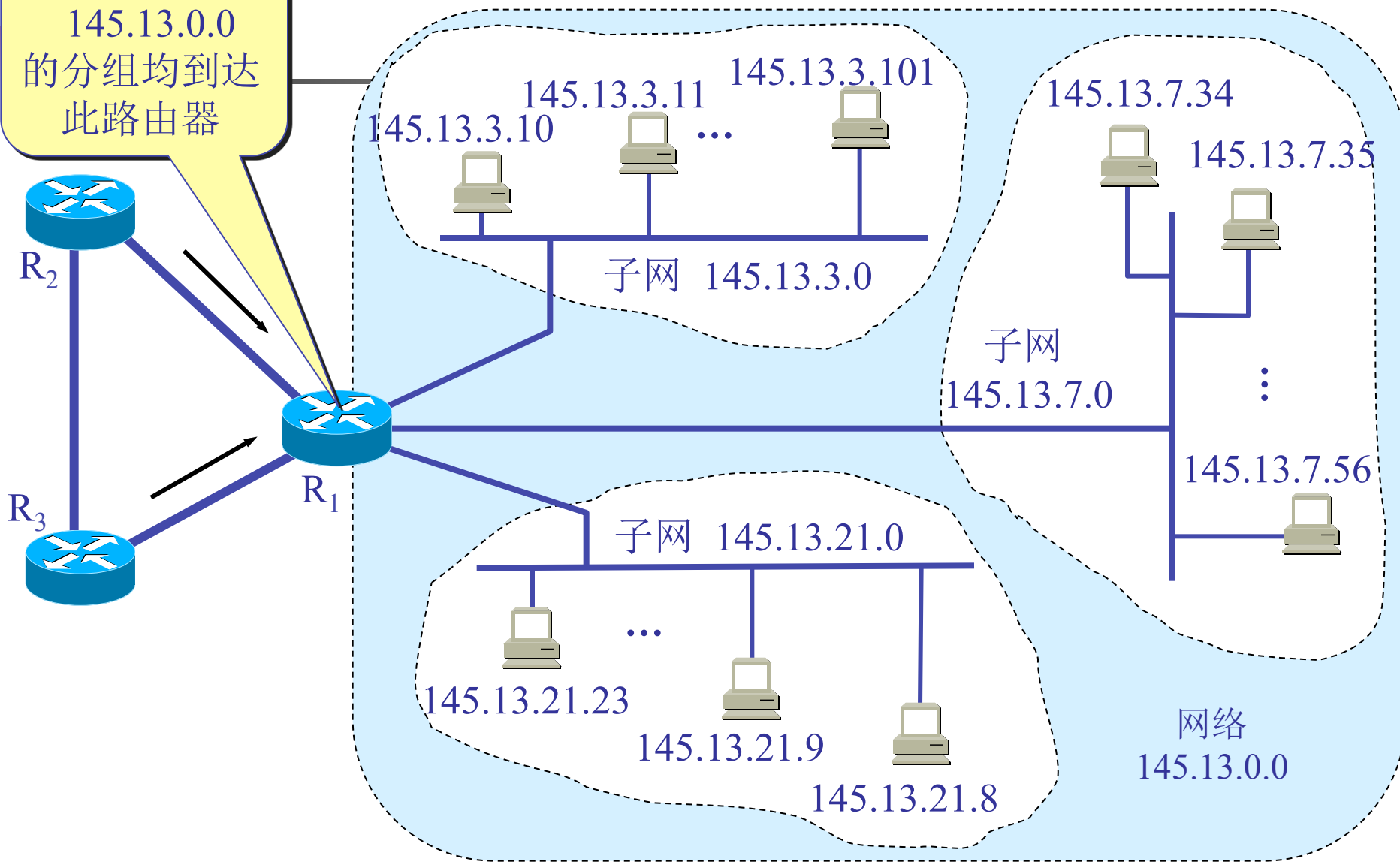


武汉大学

WUHAN UNIVERSITY

# 划分为三个子网后对外仍是一个网络

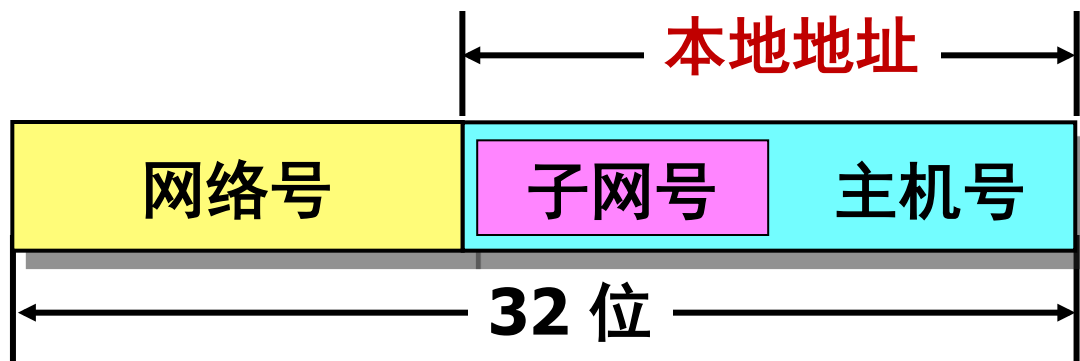
所有到达网络  
145.13.0.0  
的分组均到达  
此路由器





## 划分子网后变成了三级结构

- 当没有划分子网时，IP 地址是两级结构。
- 划分子网后 IP 地址就变成了三级结构。
- 划分子网只是把 IP 地址的主机号 host-id 这部分进行再划分，而不改变 IP 地址原来的网络号 net-id。





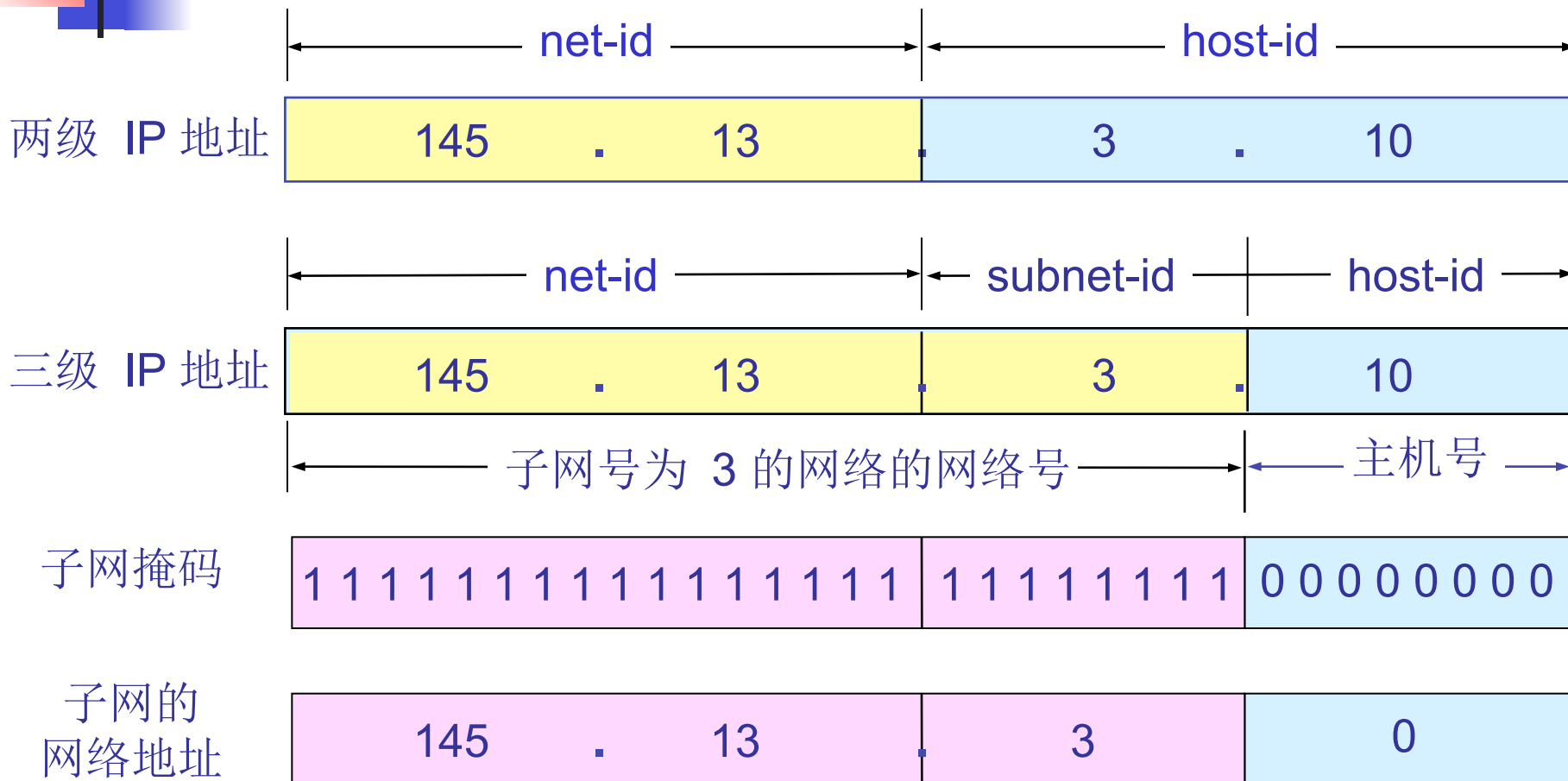
## 2. 子网掩码

- 从一个 IP 数据报的首部并无法判断源主机或目的主机所连接的网络是否进行了子网划分。
- 使用子网掩码 (subnet mask) 可以找出 IP 地址中的子网部分。

规则：

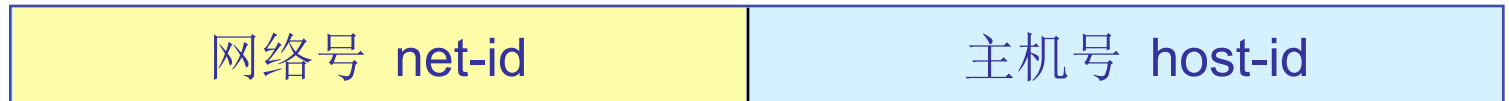
- 子网掩码长度 = 32 位
- 某位 = 1: IP地址中的对应位为网络号和子网号
- 某位 = 0: IP地址中的对应位为主机号

# IP 地址的各字段和子网掩码

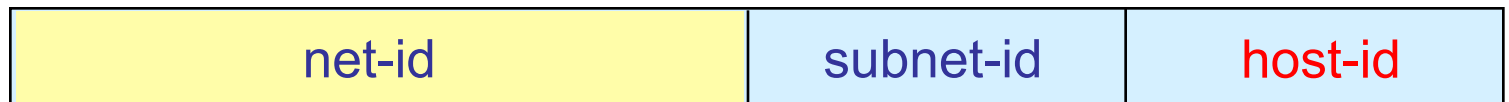


# (IP 地址) AND (子网掩码) = 网络地址

两级 IP 地址

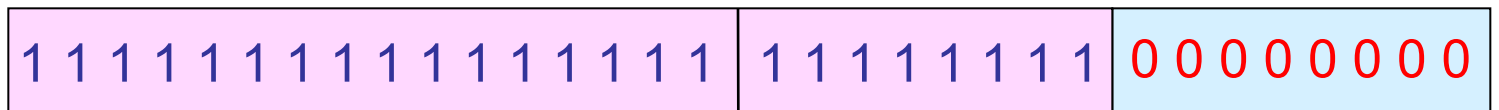


三级 IP 地址

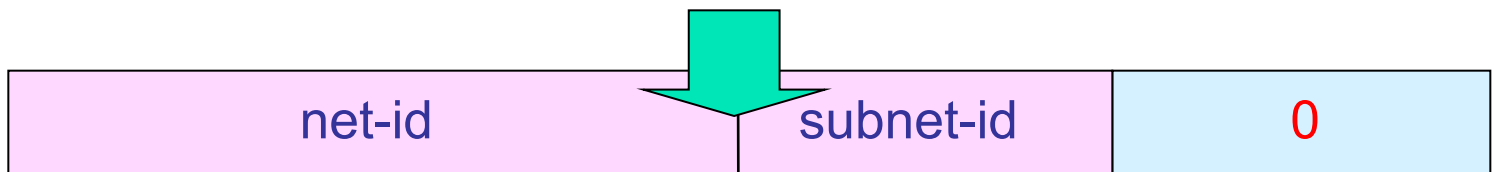


逐位进行 AND 运算

子网掩码



子网的  
网络地址





# 子网掩码是一个重要属性

- 子网掩码是一个网络或一个子网的重要属性。
- 路由器在和相邻路由器交换路由信息时，必须把自己所在网络（或子网）的子网掩码告诉相邻路由器。
- 路由器的路由表中的每一个项目，除了要给出目的网络地址外，还必须同时给出该网络的子网掩码。
- 若一个路由器连接在两个子网上就拥有两个网络地址和两个子网掩码。

**【例4-2】** 已知 IP 地址是 141.14.72.24，子网掩码是 255.255.192.0。试求网络地址。

(a) 点分十进制表示的 IP 地址

141 . 14 . 72 . 24

(b) IP 地址的第 3 字节是二进制

141 . 14 . 01001000 . 24

(c) 子网掩码是 255.255.192.0

11111111 11111111 11000000 00000000

(d) IP 地址与子网掩码逐位相与

141 . 14 . 01000000 . 0

(e) 网络地址（点分十进制表示）

141 . 14 . 64 . 0





## 4.3.2 使用子网掩码的分组转发过程

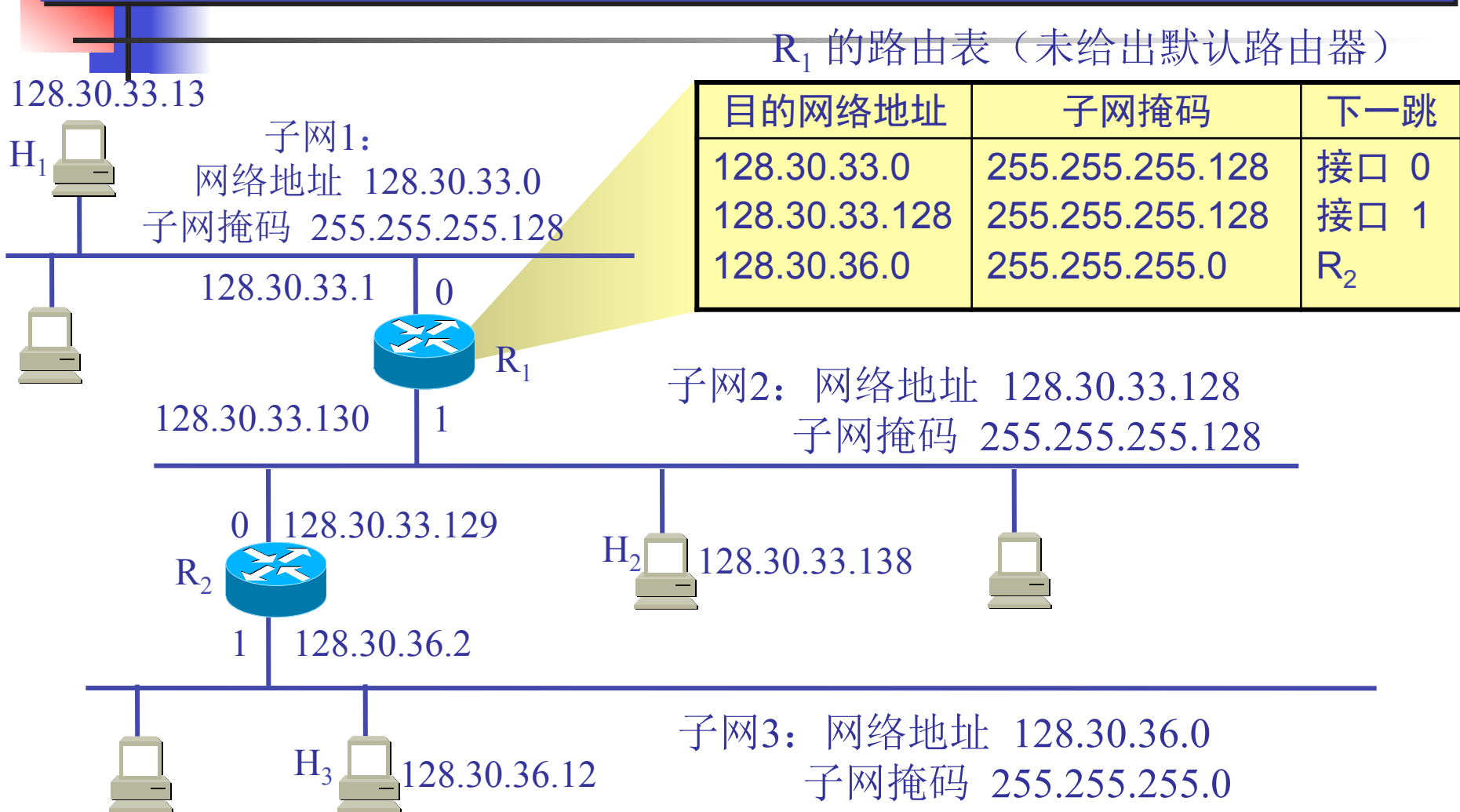
---

- 在不划分子网的两级 IP 地址下，从 IP 地址得出网络地址是个很简单的事。
- 但在划分子网的情况下，从 IP 地址却不能唯一地得出网络地址来，这是因为网络地址取决于那个网络所采用的子网掩码，但数据报的首部并没有提供子网掩码的信息。
- 因此分组转发的算法也必须做相应的改动。

# 在划分子网的情况下路由器转发分组的算法

- (1) 从收到的分组的首部提取目的 IP 地址  $D$ 。
- (2) 先用各网络的子网掩码和  $D$  逐位相“与”，看是否和相应的网络地址匹配。若匹配，则将分组直接交付。否则就是间接交付，执行(3)。
- (3) 若路由表中有目的地址为  $D$  的特定主机路由，则将分组传送给指明的下一跳路由器；否则，执行(4)。
- (4) 对路由表中的每一行的子网掩码和  $D$  逐位相“与”，若其结果与该行的目的网络地址匹配，则将分组传送给该行指明的下一跳路由器；否则，执行(5)。
- (5) 若路由表中有一个默认路由，则将分组传送给路由表中所指明的默认路由器；否则，执行(6)。
- (6) 报告转发分组出错。

【例4-4】已知互联网和路由器  $R_1$  中的路由表。主机  $H_1$  向  $H_2$  发送分组。试讨论  $R_1$  收到  $H_1$  向  $H_2$  发送的分组后查找路由表的过程。

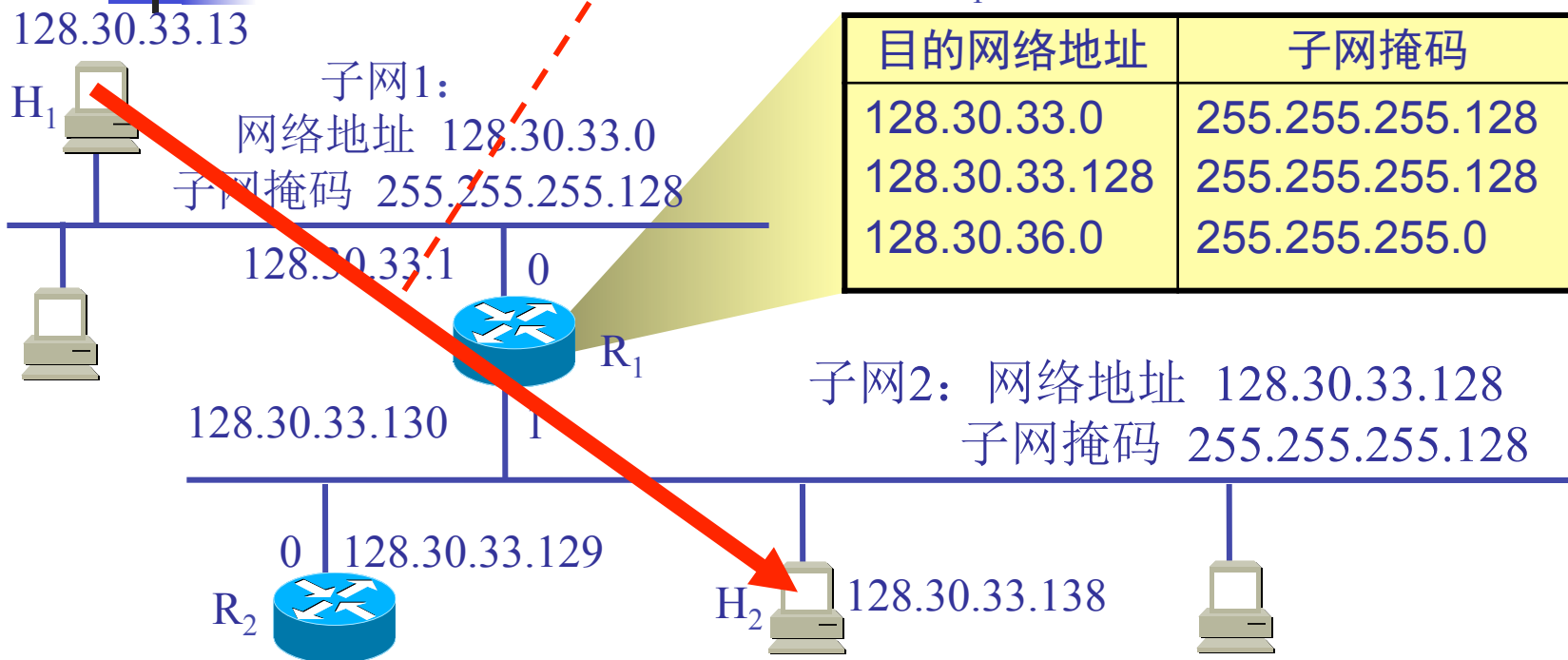


# 主机 $H_1$ 要发送分组给 $H_2$

要发送的分组的目的地 IP 地址：128.30.33.138

$R_1$  的路由表（未给出默认路由器）

目的网络地址	子网掩码	下一跳
128.30.33.0	255.255.255.128	接口 0
128.30.33.128	255.255.255.128	接口 1
128.30.36.0	255.255.255.0	$R_2$



因此  $H_1$  首先检查主机 128.30.33.138 是否连接在本网络上  
如果是，则直接交付；  
否则，就送交路由器  $R_1$ ，并逐项查找路由表。

# 本子网的子网掩码 255.255.255.128 与分组的 IP 地址 128.30.33.138 逐比特相“与”(AND 操作)

255.255.255.128 AND 128.30.33.138 的计算

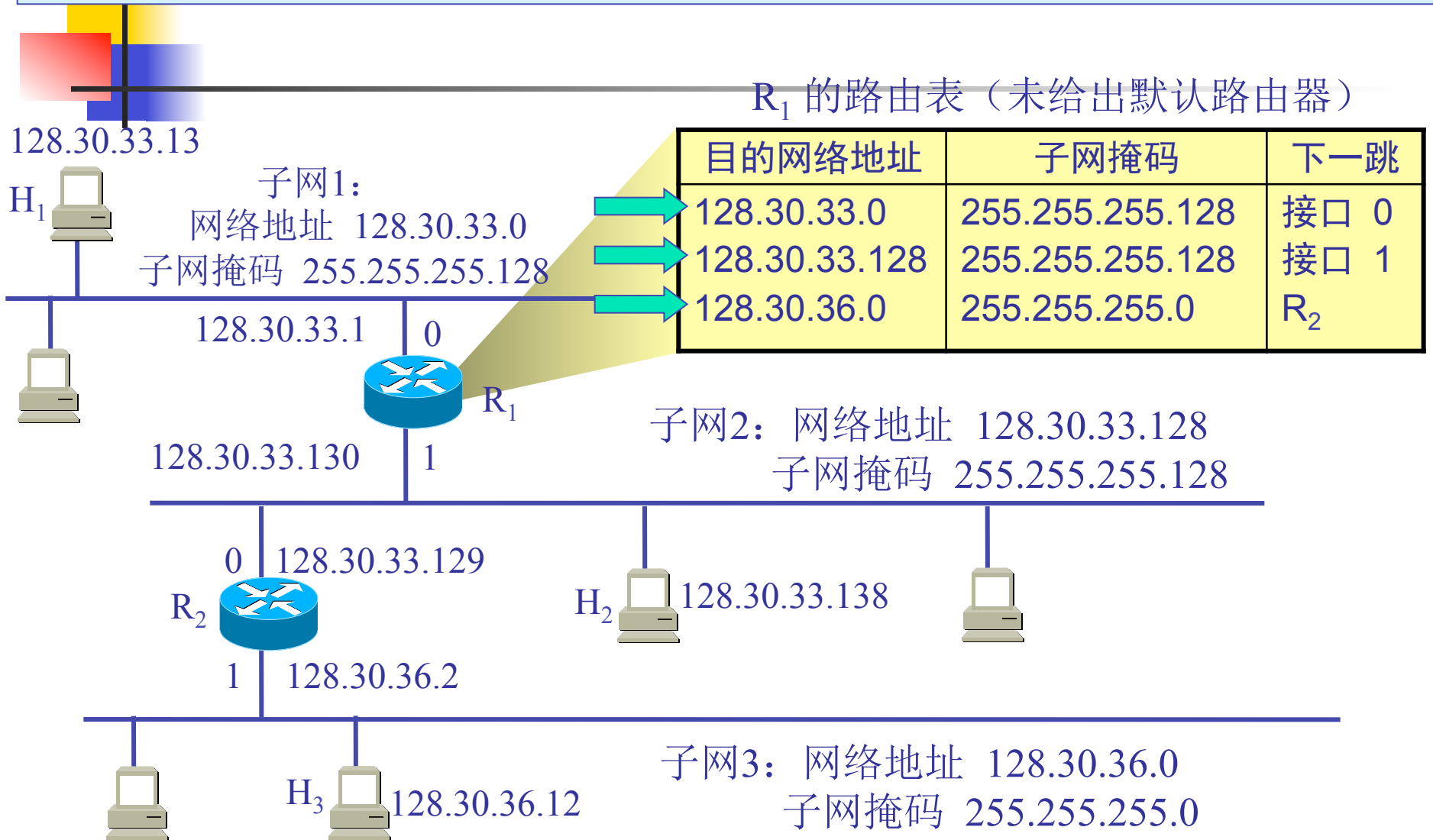
255 就是二进制的全 1，因此 255 AND xyz = xyz，  
这里只需计算最后的 128 AND 138 即可。

128	→	10000000
138	→	10001010
<hr/>		
逐比特 AND 操作后：	10000000	→ 128

逐比特 AND 操作	255.255.255.128
	128. 30. 33.138
	<hr/>
	128. 30. 33.128

≠ H<sub>1</sub> 的网络地址

因此  $H_1$  必须把分组传送到路由器  $R_1$   
然后逐项查找路由表

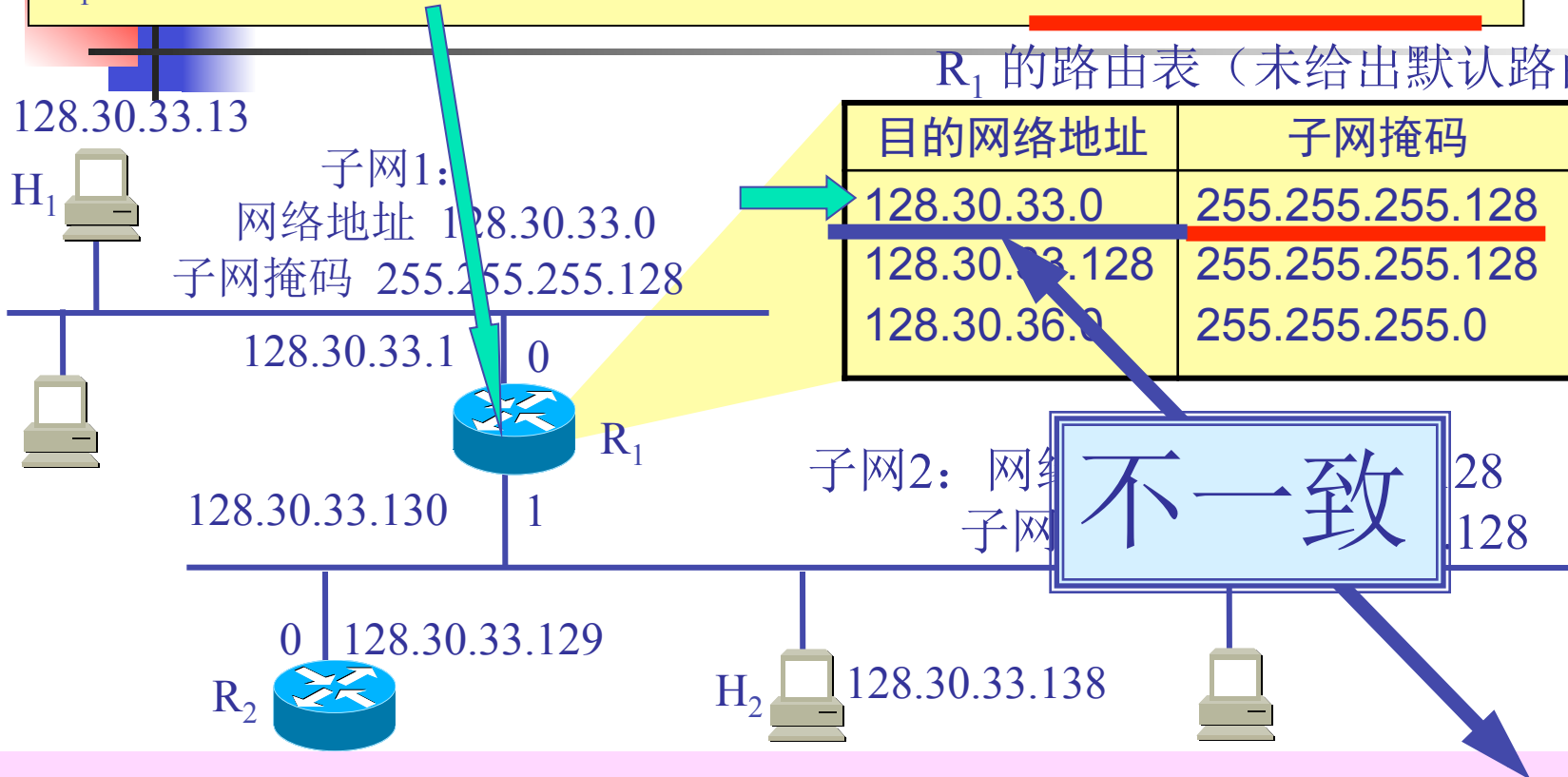


路由器  $R_1$  收到分组后就用路由表中第 1 个项目的子网掩码和 128.30.33.138 逐比特 **AND** 操作

$R_1$  收到的分组的目的 IP 地址: 128.30.33.138

$R_1$  的路由表 (未给出默认路由器)

目的网络地址	子网掩码	下一跳
128.30.33.0	255.255.255.128	接口 0
128.30.33.128	255.255.255.128	接口 1
128.30.36.0	255.255.255.0	$R_2$



$255.255.255.128 \text{ AND } 128.30.33.138 = 128.30.33.128$

不匹配!

(因为 128.30.33.128 与路由表中的 128.30.33.0 不一致)

# 路由器 R<sub>1</sub> 再用路由表中第 2 个项目的子网掩码和 128.30.33.138 逐比特 AND 操作

R<sub>1</sub> 收到的分组的目的 IP 地址: 128.30.33.138

R<sub>1</sub> 的路由表 (未给出默认路由器)

目的网络地址	子网掩码	下一跳
128.30.33.0	255.255.255.128	接口 0
128.30.33.128	255.255.255.128	接口 1
128.30.36.0	255.255.255.0	R <sub>2</sub>

128.30.33.13



子网1:

网络地址 128.30.33.0

子网掩码 255.255.255.128

128.30.33.1

0



R<sub>1</sub>

128.30.33.130

1

0

128.30.33.129

R<sub>2</sub>



子网2: 网络地址

子网掩码

一致!

H<sub>2</sub>



128.30.33.138



255.255.255.128 AND 128.30.33.138 = 128.30.33.128

匹配!

这表明子网 2 就是收到的分组所要寻找的目的网络





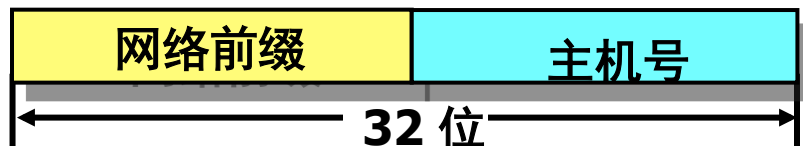
# CIDR 最主要的特点

- CIDR（**无分类域间路由选择**）消除了传统的 A 类、B 类和 C 类地址以及划分子网的概念，因而可以更加有效地分配 IPv4 的地址空间。
- CIDR 使用各种长度的“**网络前缀**” (network-prefix) 来代替分类地址中的网络号和子网号。
- IP 地址从三级编址（使用子网掩码）又回到了两级编址。



# 无分类的两级编址

- 无分类的两级编址的记法是：



**IP地址 ::= {<网络前缀>, <主机号>} (4-3)**

- CIDR 使用“**斜线记法**” (slash notation), 它又称为 **CIDR 记法**, 即在 IP 地址面加上一个斜线 “/”, 然后写上网络前缀所占的位数 (这个数值对应于三级编址中子网掩码中 1 的个数)。例如: **220.78.168.0/24**



# CIDR 地址块

- CIDR 把网络前缀都相同的连续的 IP 地址组成“**CIDR 地址块**”。
- 128.14.32.0/20 表示的地址块共有  $2^{12}$  个地址（因为斜线后面的 **20** 是网络前缀的位数，所以这个地址的主机号是 12 位）。
  - 这个地址块的起始地址是 128.14.32.0。
  - 在不需要指出地址块的起始地址时，也可将这样的地址块简称为“**/20 地址块**”。
  - 128.14.32.0/20 地址块的最小地址：128.14.32.0
  - 128.14.32.0/20 地址块的最大地址：128.14.47.255
  - 全 0 和全 1 的主机号地址一般不使用。

# 128.14.32.0/20 表示的地址 ( $2^{12}$ 个地址)

最小地址

所有地址  
的 20 位  
前缀都是  
一样的

最大地址



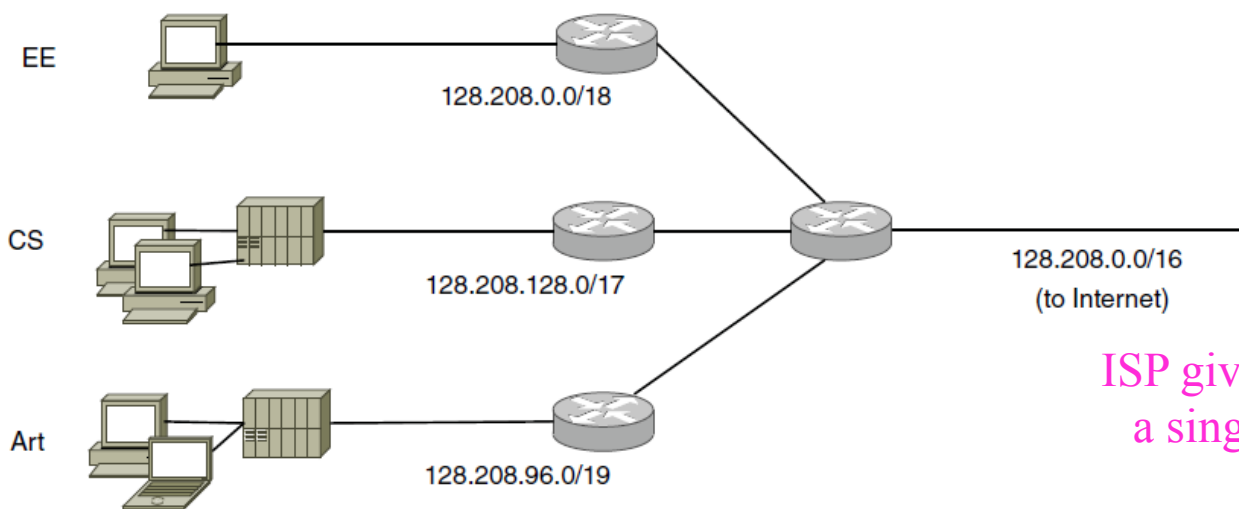
```
10000000 00001110 00100000 00000000
10000000 00001110 00100000 00000001
10000000 00001110 00100000 00000010
10000000 00001110 00100000 00000011
10000000 00001110 00100000 00000100
10000000 00001110 00100000 00000101
    . . .
    . . .
10000000 00001110 00101111 11111011
10000000 00001110 00101111 11111100
10000000 00001110 00101111 11111101
10000000 00001110 00101111 11111110
10000000 00001110 00101111 11111111
```



## IP 地址 (2) - 子网

某大学拥有如下地址128.208.0.0/16. 这些地址被分配给三个学院 (EE, CS, Art) .

- 平均分配: 等分4份, 三个学院分配到: 128.208.0.0/18, 128.208.64.0/18, 128.208.128.0/18, 剩余128.208.192.0/18
- 非平均: 128.208.128.0/17(一半), 128.208.0.0/18 (四分之一), 128.208.96.0/19 (八分之一), 128.208.64.0/19 (另外的八分之一)
- 但对于外部, 只有一个128.208.0.0/16



ISP gives network  
a single prefix

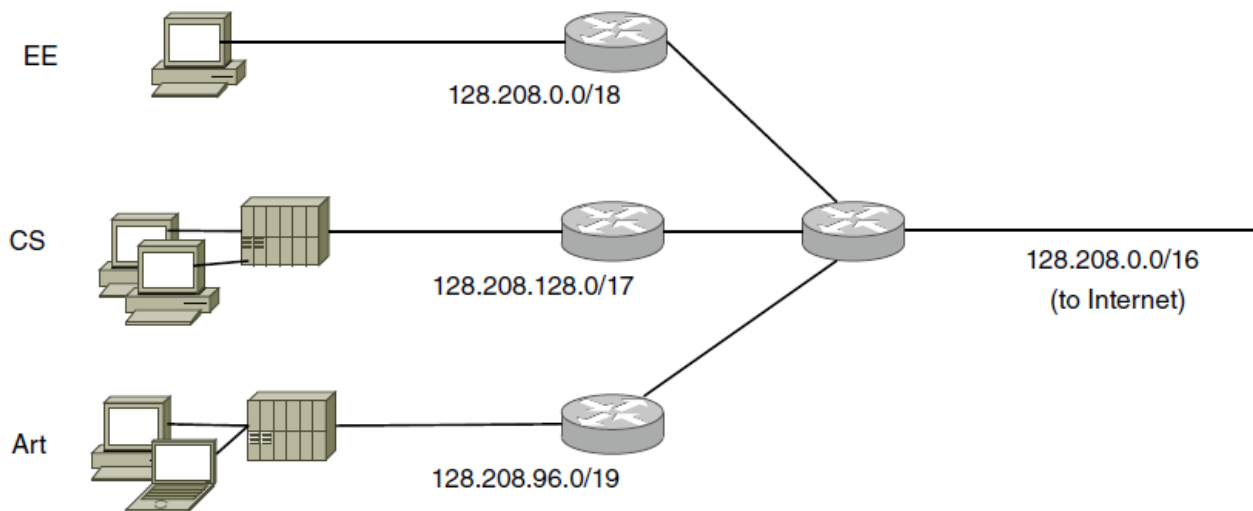
Network divides it into subnets internally



## IP 地址 (2) - 子网

路由器是如何转发一个到达的数据包的呢？（如图，并且目的地址为128.208.2.51）

- 路由器会根据目的地址来查询路由表
- 路由器将目的地址和网络的掩码进行AND操作：与CS网络的掩码进行操作得结果为128.208.0.0，这个并不是CS网络的prefix，则这个数据包不是发给CS网络。同样，与EE网络的结果为128.208.0.0，确实是E网络的prefix，所以这个包是给EE网络
- 路由器将包发给EE网络





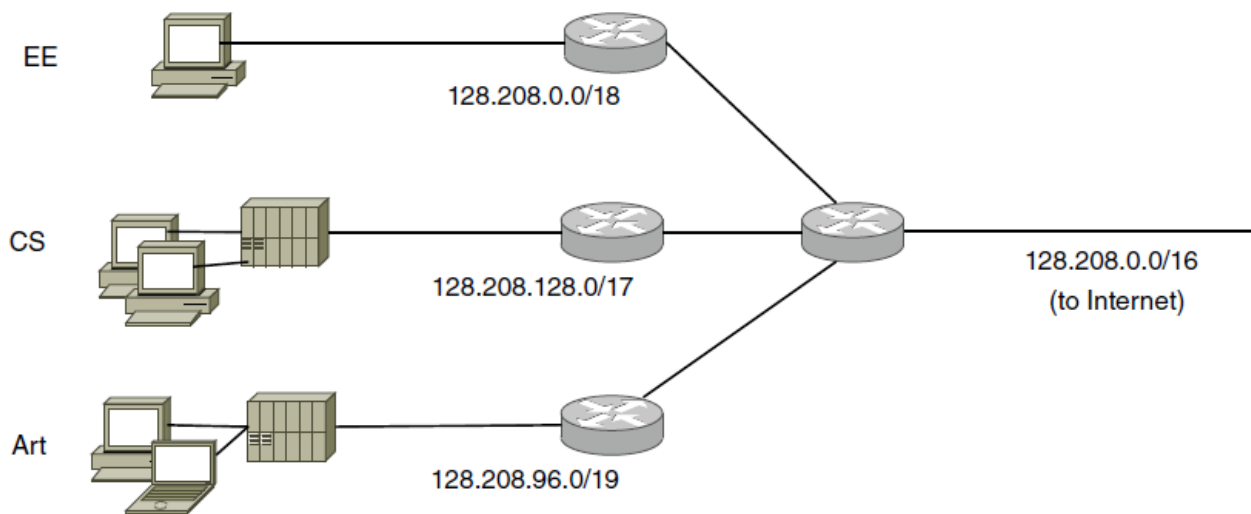
## IP地址 (3) – 聚合

我们学习了如何将网络地址划分子网，同时我们也发现，子网的地址（**prefix**）不会被路由器向外发布（通告）。路由器只会发布一个聚合的地址。

128.208.0.0/18

128.208.128.0/17  $\Rightarrow$  128.208.0.0/16

128.208.96.0/19





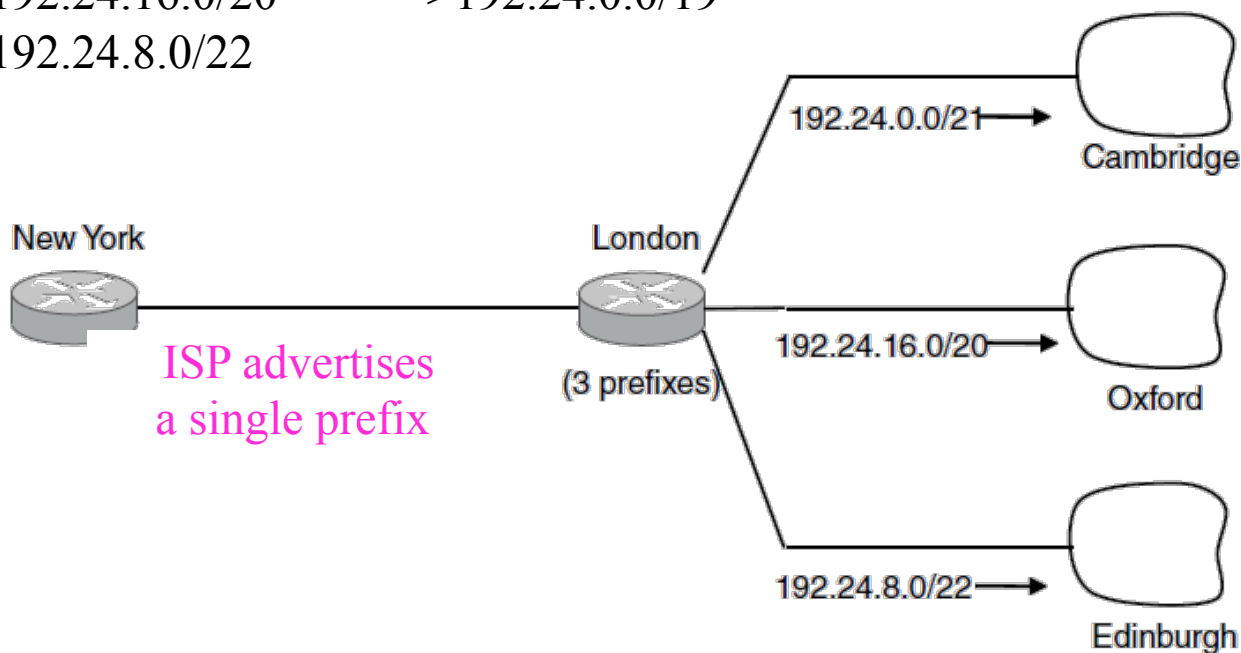
## IP地址 (3) – 聚合

聚合由于将多个地址prefix合成一个，所以可以很大程度减少路由表的大小

192.24.0.0/21

192.24.16.0/20      => 192.24.0.0/19

192.24.8.0/22



ISP customers have different prefixes

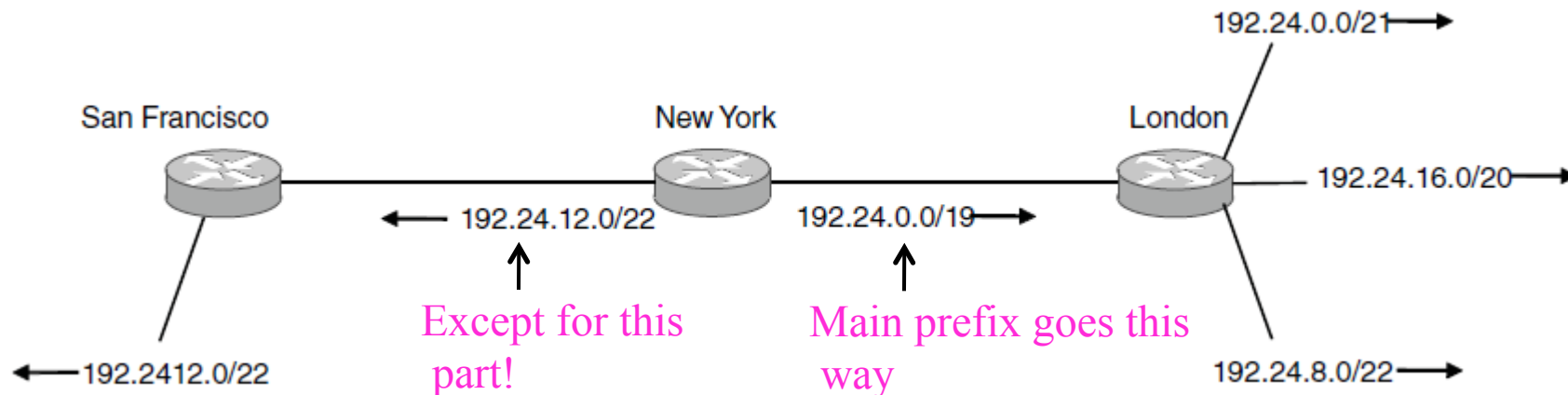




## IP地址(4) – 最长匹配前缀

聚合提高性能，但存在一个问题：如果192.24.0.0/19 的另外一部分被分配给另外一个网络，如何处理？

- 如图192.24.12.0/22 分给San Francisco，那么一个数据包目的地为192.24.15.0即匹配london又匹配San Francisco，怎么转发？
- 数据包应该被发送给最长匹配原则得出的网络，即如果有多个网络都匹配数据包的目的地址，那么这个数据包应该被发往具有最长网络地址的网络。
- 所以数据包192.24.15.0应该被发送到San Francisco。





### 3. 使用二叉线索查找路由表

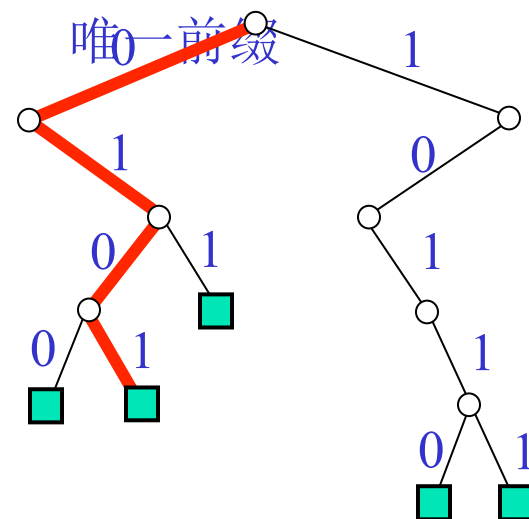
- 当路由表的项目数很大时，怎样设法减小路由表的查找时间就成为一个非常重要的问题。
- 为了进行更加有效的查找，通常是将无分类编址的路由表存放在一种层次的数据结构中，然后自上而下地按层次进行查找。这里最常用的就是二叉线索 (binary trie)。



# 用 5 个前缀构成的二叉线索

32 位的 IP 地址

01000110 00000000 00000000 00000000	0100
01010110 00000000 00000000 00000000	0101
01100001 00000000 00000000 00000000	011
10110000 00000010 00000000 00000000	10110
10111011 00001010 00000000 00000000	10111

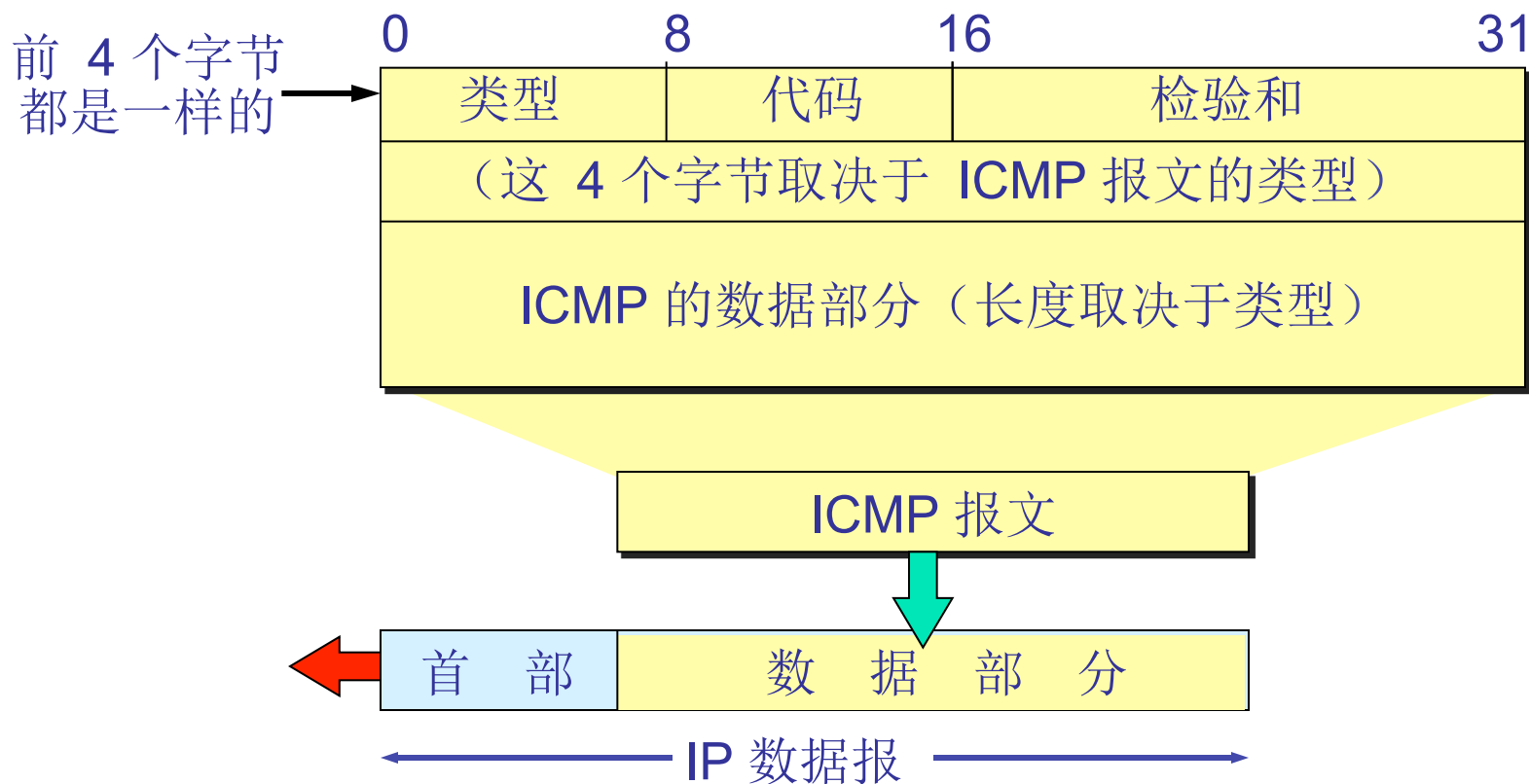




## 4.4 网际控制报文协议 ICMP

- 为了提高 IP 数据报交付成功的机会，在网际层使用了网际控制报文协议 ICMP (Internet Control Message Protocol)。
- ICMP 允许主机或路由器报告差错情况和提供有关异常情况的报告。
- ICMP 不是高层协议，而是 IP 层的协议。
- ICMP 报文作为 IP 层数据报的数据，加上数据报的首部，组成 IP 数据报发送出去。

# ICMP 报文的格式





## 4.4.1 ICMP 报文的种类

---

- ICMP 报文的种类有两种，即 ICMP 差错报告报文和 ICMP 询问报文。
- ICMP 报文的前 4 个字节是统一的格式，共有三个字段：即类型、代码和检验和。接着的 4 个字节的内容与 ICMP 的类型有关。

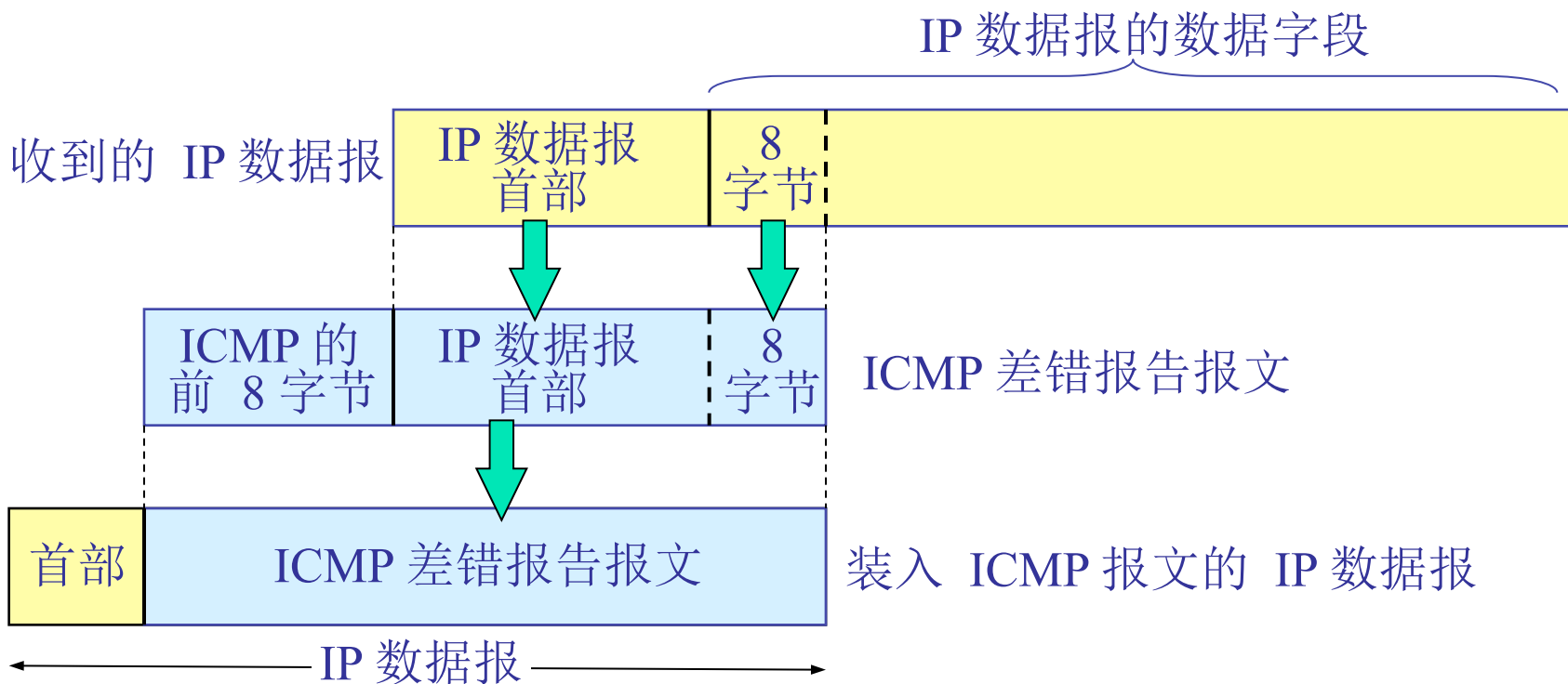


# ICMP 差错报告报文共有 4 种

- 终点不可达 (Destination unreachable)
- 时间超过 (Time exceeded): TTL为零
- 参数问题 (Parameter problem): 包头设置有问题
- 改变路由 (重定向) (Redirect):
- 源点抑制(Source quench) : Choke packe



# ICMP 差错报告报文的数据字段的内容







# 不应发送 ICMP 差错报告报文的几种情况

- 对 ICMP 差错报告报文不再发送 ICMP 差错报告报文。
- 对第一个分片的数据报片的所有后续数据报片都不发送 ICMP 差错报告报文。
- 对具有多播地址的数据报都不发送 ICMP 差错报告报文。
- 对具有特殊地址（如127.0.0.0 或 0.0.0.0）的数据报不发送 ICMP 差错报告报文。



# ICMP 询问报文有两种

---

- 回送请求和回答报文 (Echo and Echo reply)
- 时间戳请求和回答报文 (Timestamp request/reply): 同时, 但有时间戳



## 4.4.2 ICMP 的应用举例

### PING (Packet InterNet Groper)

- **PING** 用来测试两个主机之间的连通性。
- **PING** 使用了 **ICMP** 回送请求与回送回答报文。
- **PING** 是应用层直接使用网络层 **ICMP** 的例子，它没有通过运输层的 **TCP** 或 **UDP**。



## PING 的应用举例

```
C:\Documents and Settings\XXR>ping mail.sina.com.cn

Pinging mail.sina.com.cn [202.108.43.230] with 32 bytes of data:

Reply from 202.108.43.230: bytes=32 time=368ms TTL=242
Reply from 202.108.43.230: bytes=32 time=374ms TTL=242
Request timed out.
Reply from 202.108.43.230: bytes=32 time=374ms TTL=242

Ping statistics for 202.108.43.230:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 368ms, Maximum = 374ms, Average = 372ms
```

用 PING 测试主机的连通性



## 4.4.2 ICMP 的应用举例

### Traceroute 的应用举例

- 在 Windows 操作系统中这个命令是 `tracert`。
- 用来跟踪一个分组从源点到终点的路径。
- 它利用 IP 数据报中的 **TTL** 字段和 **ICMP** 时间超过差错报告报文实现对从源点到终点的路径的跟踪。



## 4.4.2 TCMP 的应用举例

```
C:\Documents and Settings\XXR>tracert mail.sina.com.cn
```

```
Tracing route to mail.sina.com.cn [202.108.43.230]  
over a maximum of 30 hops:
```

1	24 ms	24 ms	23 ms	222.95.172.1
2	23 ms	24 ms	22 ms	221.231.204.129
3	23 ms	22 ms	23 ms	221.231.206.9
4	24 ms	23 ms	24 ms	202.97.27.37
5	22 ms	23 ms	24 ms	202.97.41.226
6	28 ms	28 ms	28 ms	202.97.35.25
7	50 ms	50 ms	51 ms	202.97.36.86
8	308 ms	311 ms	310 ms	219.158.32.1
9	307 ms	305 ms	305 ms	219.158.13.17
10	164 ms	164 ms	165 ms	202.96.12.154
11	322 ms	320 ms	2988 ms	61.135.148.50
12	321 ms	322 ms	320 ms	freemail43-230.sina.com [202.108.43.230]

```
Trace complete.
```

用 **tracert** 命令获得到目的主机的路由信息



## 4.5 互联网的路由选择协议

- 4.5.1 有关路由选择协议的几个基本概念
- 4.5.2 内部网关协议 RIP
- 4.5.3 内部网关协议 OSPF
- 4.5.4 外部网关协议 BGP
- 4.5.5 路由器的构成



# 路由

- 什么是路由算法? 是网络层软件的一部分, 它负责确定一个入境数据包应该被发送到哪一条输出线路上.
- 路由器的两个进程:
  - 路由表的创建: 创建和更新路由表
    - 路由表指出到达的包应该通过那个端口 (线路) 转发出去
  - 转发: 根据路由表来将数据包发送到相应的端口 (线路)

路由算法的两大类:

- 非自适应算法 (静态路由)
- 自适应算法 (动态路由)





# 分层次的路由选择协议

- 互联网采用分层次的路由选择协议。
  - 内部域的路由（**AS**内的路由）
  - 域之间的路由（**AS**之间的路由）
- 优点
  - 可扩展性
  - 隐私保护



# 自治系统 AS (Autonomous System)

- **自治系统 AS 的定义**：在单一的技术管理下的一组路由器，而这些路由器使用一种 **AS** 内部的路由选择协议和共同的度量以确定分组在该 **AS** 内的路由，同时还使用一种 **AS** 之间的路由选择协议用以确定分组在 **AS**之间的路由。
- 现在对自治系统 **AS** 的定义是强调下面的事实：尽管一个 **AS** 使用了多种内部路由选择协议和度量，但**重要的是一个 AS 对其他 AS 表现出的是一个单一的和一致的路由选择策略。**



# 互联网有两大类路由选择协议

- 内部网关协议 IGP (Interior Gateway Protocol)
  - AS内部使用的路由。
  - RIP 和 OSPF 协议。
- 外部网关协议 EGP (External Gateway Protocol)
  - AS之间的路由。
  - BGP协议。

# 自治系统和 内部网关协议、外部网关协议



自治系统之间的路由选择也叫做  
**域间路由选择**(interdomain routing),  
在自治系统内部的路由选择叫做  
**域内路由选择**(intradomain routing)



## 4.5.2 内部网关协议 RIP

### 1. 工作原理

- RIP 是一种分布式的、基于距离向量的路由选择协议。
- 路由器通过和其邻居交换路由信息来得出到整个AS域其他所有节点（路由器）的最短路径（距离最短），注：RIP仅仅知道下一跳信息
- RIP协议中的“距离”也称为“跳数” (hop count)，因为每经过一个路由器，跳数就加 1
- RIP基于距离矢量路由



## 5.2.4 距离矢量路由 (1)

距离矢量 是一种分布式的路由算法

- 最短路径的计算是由各节点共同完成，即每个节点只做一跳的计算，最短路径由所有一条的总和

算法：

- 每个节点通过包交换来知道其所有的邻居节点以及到邻居节点的开销
- 每个节点广播邻居节点信息给其所有的邻居节点
- 收到这些信息后，每个节点记录新的节点的信息
- 每当节点信息更新时，每个节点需要将这些更新的信息再次广播给其邻居

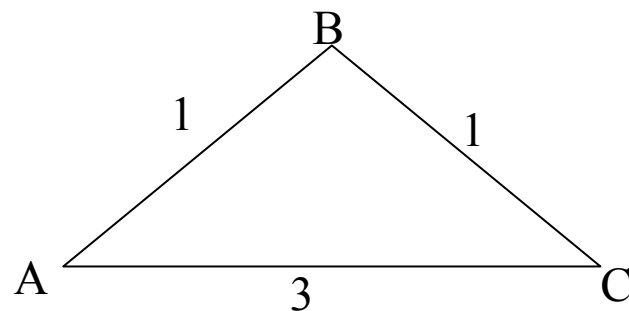


## 距离矢量路由 (2)

**Example:** 网络的拓扑如图，请描述距离矢量算法的过程（即路由表是如何更新的）。

Step 1: A: (B, 1, B), (C, 3, C)  
B: (A, 1, A), (C, 1, C)  
C: (B, 1, B), (A, 3, A)

Step 2: A: (B, 1, B), (C, 2, B)  
B: (A, 1, A), (C, 1, C)  
C: (B, 1, B), (A, 2, B)




注：每个节点只广播它到其他节点的距离，而不会广播下一跳



To	A	I	H	K
A	0	24	20	21
B	12	36	31	28
C	25	18	19	36
D	40	27	8	24
E	14	7	30	22
F	23	20	19	40
G	18	31	6	31
H	17	20	0	19
I	21	0	14	22
J	9	11	7	10
K	24	22	22	0
L	29	33	9	9

JA delay is 8	JI delay is 10	JH delay is 12	JK delay is 6
------------------------	-------------------------	-------------------------	------------------------


  
 Vectors received from  
 J's four neighbors

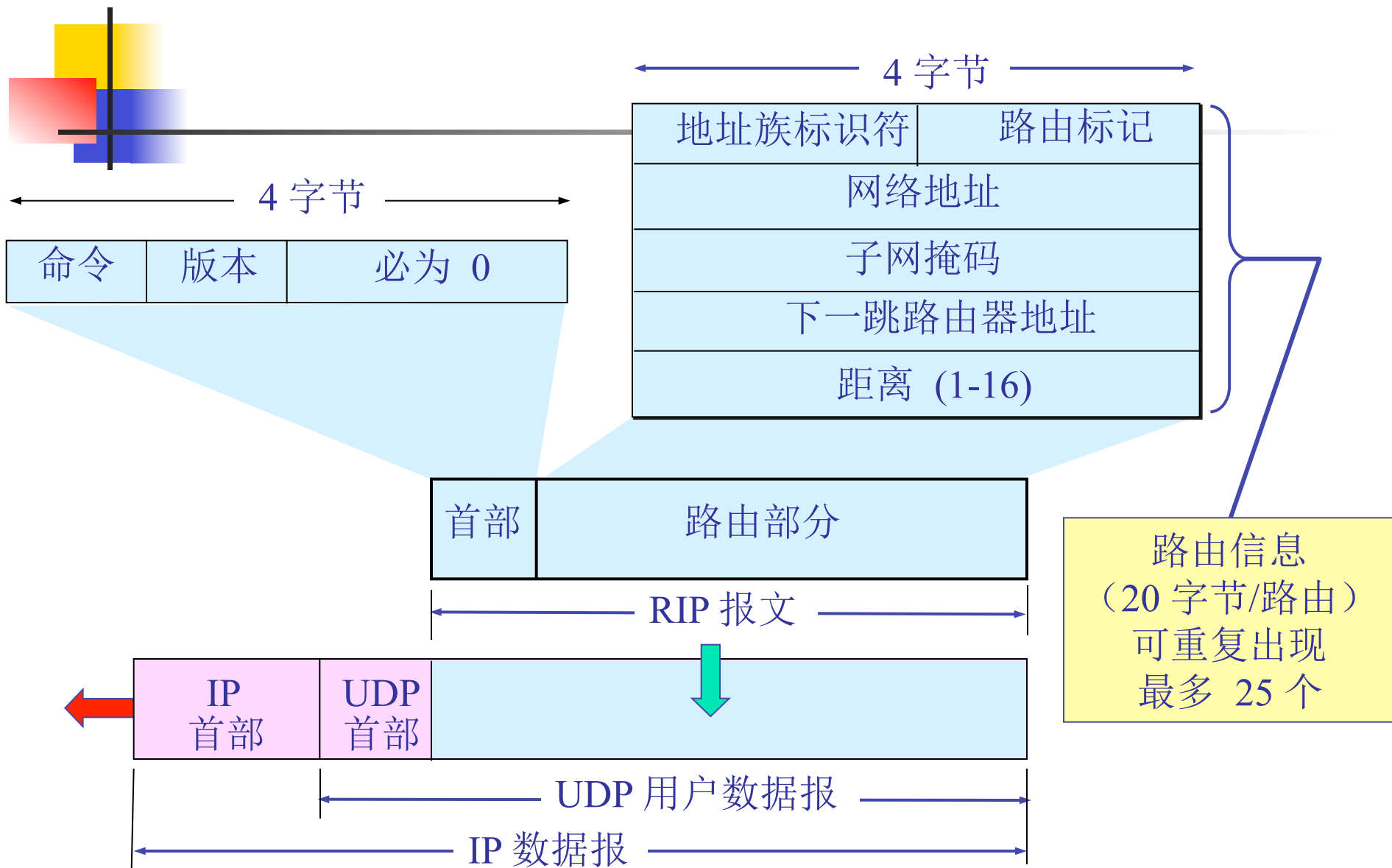
A->G:18, J->A:8=>J->G:26  
I->G:31, J->I:10=>J->G:41  
H->G:6, J->H:12=>J->G:18  
K->G:31, J->K:6=>J->G:37

Vectors received at J from  
Neighbors A, I, H and K





# 3. RIP2 协议的报文格式





# RIP2 的报文

## 由首部和路由部分组成。

- RIP2 报文中的路由部分由若干个路由信息组成。每个路由信息需要用 20 个字节。地址族标识符（又称为地址类别）字段用来标志所使用的地址协议。
- 路由标记填入自治系统的号码，这是考虑使RIP 有可能收到本自治系统以外的路由选择信息。再后面指出某个网络地址、该网络的子网掩码、下一跳路由器地址以及到此网络的距离。



路由器收到相邻路由器（其地址为 **X**）的一个 **RIP** 报文：

**(1)** 先修改此 **RIP** 报文中的所有项目：把“下一跳”字段中的地址都改为 **X**，并把所有的“距离”字段的值加 **1**。

**(2)** 对修改后的 **RIP** 报文中的每一个项目，重复以下步骤：

- 若项目中的目的网络不在路由表中，则把该项目加到路由表中。
- 否则
  - 若下一跳字段给出的路由器地址是同样的，则把收到的项目替换原路由表中的项目。
  - 否则，若收到项目中的距离小于路由表中的距离，则进行更新，否则，什么也不做。

**(3)** 若 **3** 分钟还没有收到相邻路由器的更新路由表，则把此相邻路由器记为不可达路由器，即将距离置为 **16**（表示不可达）。

**(4)** 返回。



【例4-5】已知路由器  $R_6$  有表 4-9(a) 所示的路由表。现在收到相邻路由器  $R_4$  发来的路由更新信息，如表 4-9(b) 所示。试更新路由器  $R_6$  的路由表。

表 4-9(a) 路由器  $R_6$  的路由表

目的网络	距离	下一跳路由器
Net2	3	$R_4$
Net3	4	$R_5$
...	...	...

表 4-9(b)  $R_4$  发来的路由更新信息

目的网络	距离	下一跳路由器
Net1	3	$R_1$
Net2	4	$R_2$
Net3	1	直接交付

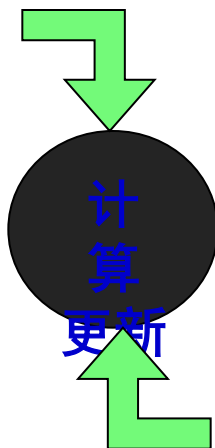
距离加 1

表 4-9(c) 修改后的表

目的网络	距离	下一跳路由器
Net1	4	$R_4$
Net2	5	$R_4$
Net3	2	$R_4$

表 4-9(d) 路由器  $R_6$  更新后的路由表

目的网络	距离	下一跳路由器
Net1	4	$R_4$
Net2	5	$R_4$
Net3	2	$R_4$
...	...	...





# 无穷计算问题

距离矢量算法简单，但存在一个问题：

- 好消息和坏消息
- 坏消息导致DV无穷计算问题

A	B	C	D	E	
•	•	•	•	•	Initially
	•	•	•	•	After 1 exchange
	1	•	•	•	After 2 exchanges
	1	2	•	•	After 3 exchanges
	1	2	3	•	After 4 exchanges
	1	2	3	4	After 5 exchanges

Good news of a path to  
A spreads quickly

主要问题出在第二步的B身上，那么我们可以这么改进吗：说当B收到C的距离矢量信息时，因到A的路径是经过B的，所以并不采用？

A	B	C	D	E	
•	•	•	•	•	Initially
	1	2	3	4	After 1 exchange
	3	2	3	4	After 2 exchanges
	3	4	3	4	After 3 exchanges
	5	4	5	4	After 4 exchanges
	5	6	5	6	After 5 exchanges
	7	6	7	6	After 6 exchanges
	7	8	7	8	After 7 exchanges
	•	•	•	•	After 8 exchanges
	•	•	•	•	After 9 exchanges
	•	•	•	•	After 10 exchanges
	•	•	•	•	After 11 exchanges
	•	•	•	•	After 12 exchanges
	•	•	•	•	After 13 exchanges
	•	•	•	•	After 14 exchanges
	•	•	•	•	After 15 exchanges
	•	•	•	•	After 16 exchanges
	•	•	•	•	After 17 exchanges
	•	•	•	•	After 18 exchanges
	•	•	•	•	After 19 exchanges
	•	•	•	•	After 20 exchanges
	•	•	•	•	After 21 exchanges
	•	•	•	•	After 22 exchanges
	•	•	•	•	After 23 exchanges
	•	•	•	•	After 24 exchanges
	•	•	•	•	After 25 exchanges
	•	•	•	•	After 26 exchanges
	•	•	•	•	After 27 exchanges
	•	•	•	•	After 28 exchanges
	•	•	•	•	After 29 exchanges
	•	•	•	•	After 30 exchanges
	•	•	•	•	After 31 exchanges
	•	•	•	•	After 32 exchanges
	•	•	•	•	After 33 exchanges
	•	•	•	•	After 34 exchanges
	•	•	•	•	After 35 exchanges
	•	•	•	•	After 36 exchanges
	•	•	•	•	After 37 exchanges
	•	•	•	•	After 38 exchanges
	•	•	•	•	After 39 exchanges
	•	•	•	•	After 40 exchanges
	•	•	•	•	After 41 exchanges
	•	•	•	•	After 42 exchanges
	•	•	•	•	After 43 exchanges
	•	•	•	•	After 44 exchanges
	•	•	•	•	After 45 exchanges
	•	•	•	•	After 46 exchanges
	•	•	•	•	After 47 exchanges
	•	•	•	•	After 48 exchanges
	•	•	•	•	After 49 exchanges
	•	•	•	•	After 50 exchanges
	•	•	•	•	After 51 exchanges
	•	•	•	•	After 52 exchanges
	•	•	•	•	After 53 exchanges
	•	•	•	•	After 54 exchanges
	•	•	•	•	After 55 exchanges
	•	•	•	•	After 56 exchanges
	•	•	•	•	After 57 exchanges
	•	•	•	•	After 58 exchanges
	•	•	•	•	After 59 exchanges
	•	•	•	•	After 60 exchanges
	•	•	•	•	After 61 exchanges
	•	•	•	•	After 62 exchanges
	•	•	•	•	After 63 exchanges
	•	•	•	•	After 64 exchanges
	•	•	•	•	After 65 exchanges
	•	•	•	•	After 66 exchanges
	•	•	•	•	After 67 exchanges
	•	•	•	•	After 68 exchanges
	•	•	•	•	After 69 exchanges
	•	•	•	•	After 70 exchanges
	•	•	•	•	After 71 exchanges
	•	•	•	•	After 72 exchanges
	•	•	•	•	After 73 exchanges
	•	•	•	•	After 74 exchanges
	•	•	•	•	After 75 exchanges
	•	•	•	•	After 76 exchanges
	•	•	•	•	After 77 exchanges
	•	•	•	•	After 78 exchanges
	•	•	•	•	After 79 exchanges
	•	•	•	•	After 80 exchanges
	•	•	•	•	After 81 exchanges
	•	•	•	•	After 82 exchanges
	•	•	•	•	After 83 exchanges
	•	•	•	•	After 84 exchanges
	•	•	•	•	After 85 exchanges
	•	•	•	•	After 86 exchanges
	•	•	•	•	After 87 exchanges
	•	•	•	•	After 88 exchanges
	•	•	•	•	After 89 exchanges
	•	•	•	•	After 90 exchanges
	•	•	•	•	After 91 exchanges
	•	•	•	•	After 92 exchanges
	•	•	•	•	After 93 exchanges
	•	•	•	•	After 94 exchanges
	•	•	•	•	After 95 exchanges
	•	•	•	•	After 96 exchanges
	•	•	•	•	After 97 exchanges
	•	•	•	•	After 98 exchanges
	•	•	•	•	After 99 exchanges
	•	•	•	•	After 100 exchanges

Bad news of no path to A is  
learned slowly



## 4.5.3 内部网关协议 OSPF

- 开放最短路径优先 OSPF (Open Shortest Path First)是为克服 RIP 的缺点在 1989 年开发出来的。
- OSPF基于链路状态路由



# 链路状态路由

链路状态 是另外一种路由算法：

- 计算复杂但拓扑结构变化时能够立即收敛
- 在Internet广泛使用 (OSPF, ISIS)

算法：

- 发现邻居（地址）
- 设置到邻居的开销
- 将邻居信息泛洪到网络的每一个节点
- 每个节点将收到其他所有节点的邻居信息，从而构建出网络拓扑
- 每个节点使用Dijkstra算法计算到其他所有节点的最短路径

和距离矢量比较：

- 节点的邻居信息需要被泛洪到每个节点
- 每个节点都知道网络拓扑



# 泛洪

一种简单的方法将数据包在整个网络上传播

工作原理：每个节点在一条链路上收到一个数据包，后将此数据包在其他所有链路上转发出去

如何防止多次转发：每个节点都需要将泛洪数据进行记录；

- Advantage: Robust
- Disadvantage: too expensive

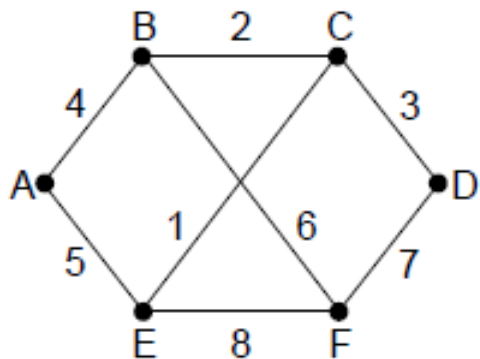




# 泛洪的数据包：LSPs

邻居信息被放在一个称为LSP (Link State Packet) 的数据包中泛洪给其他节点

- LSP包含所有的邻居，以及到邻居的开销



Network

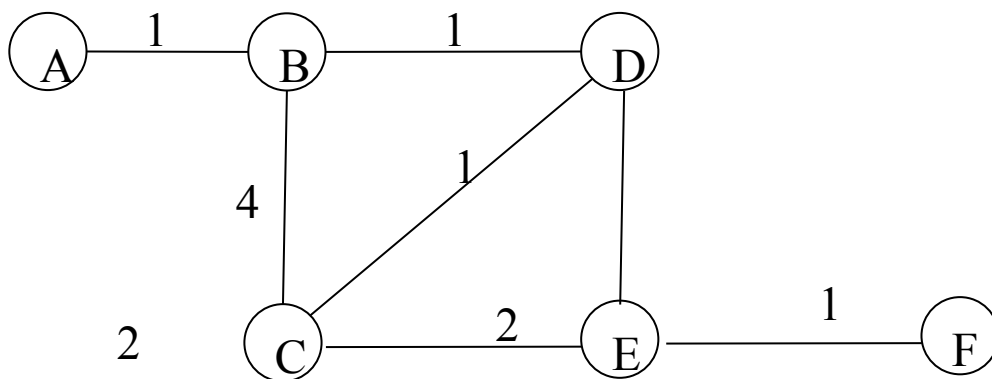
A	B	C	D	E	F
Seq.	Seq.	Seq.	Seq.	Seq.	Seq.
Age	Age	Age	Age	Age	Age
B   4	A   4	B   2	C   3	A   5	B   6
E   5	C   2	D   3	F   7	C   1	D   7
	F   6	E   1		F   8	E   8

LSP for each node



## 得出拓扑结构

Example: 节点 A 收到如下的LSP, 请画出网络拓扑结构,  
A: (B,1); B: (A,1), (C,4), (D,1); C: (B,4), (D,1), (E,2);  
D: (B,1), (C,1), (E,2); E: (C,2), (D,2), (F,1); F: (E, 1)





# 最短路径算法

怎么样得到一条最优（最短）路径: Dijkstra's algorithm:

- 每个链路都有权值
- 最短路径即权值总和最小的路径

算法:

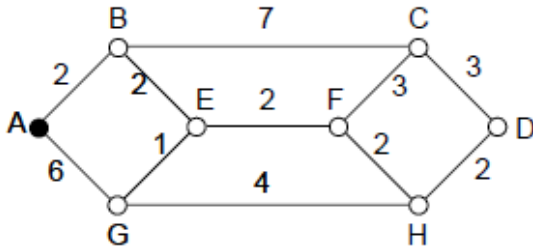
- 从汇集节点出发，将所有的节点到汇集节点的开销设置为无穷。将和汇聚节点直接相连的节点做暂时性标记（标记为到汇集节点的开销）
- 比较所有暂时性节点的开销，选取最小的节点
- 将此最小节点设置成为永久节点，此节点为新的工作节点，将所有和新的工作节点相连的节点，设置暂时性标记或者修改暂时性标记
- 重复第二步和第三步，直到所有的节点都被加入到汇集树



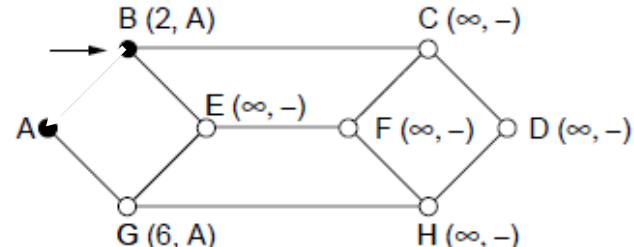
武汉大学

WUHAN UNIVERSITY

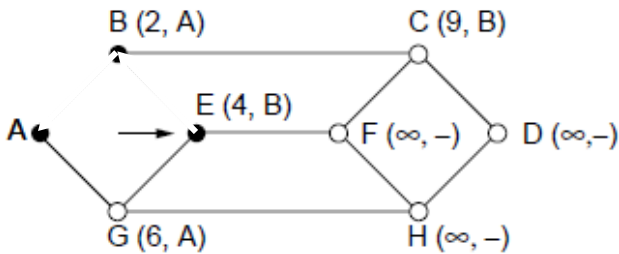
# 最短路径算法



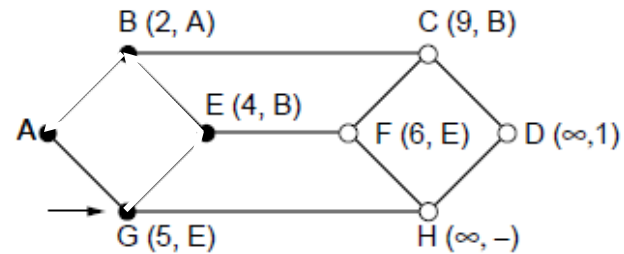
(a)



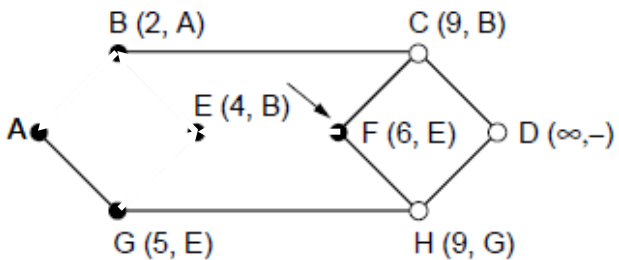
(b)



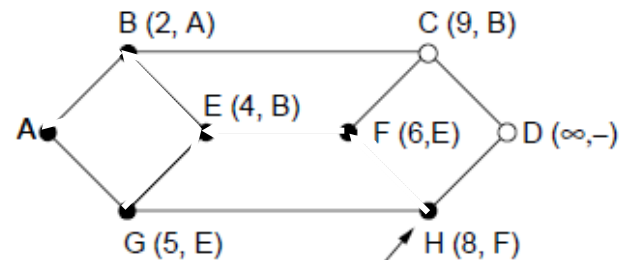
(c)



(d)



(e)



(f)

A network and first five steps in computing the shortest paths from A to D. Pink arrows show the sink tree so far.

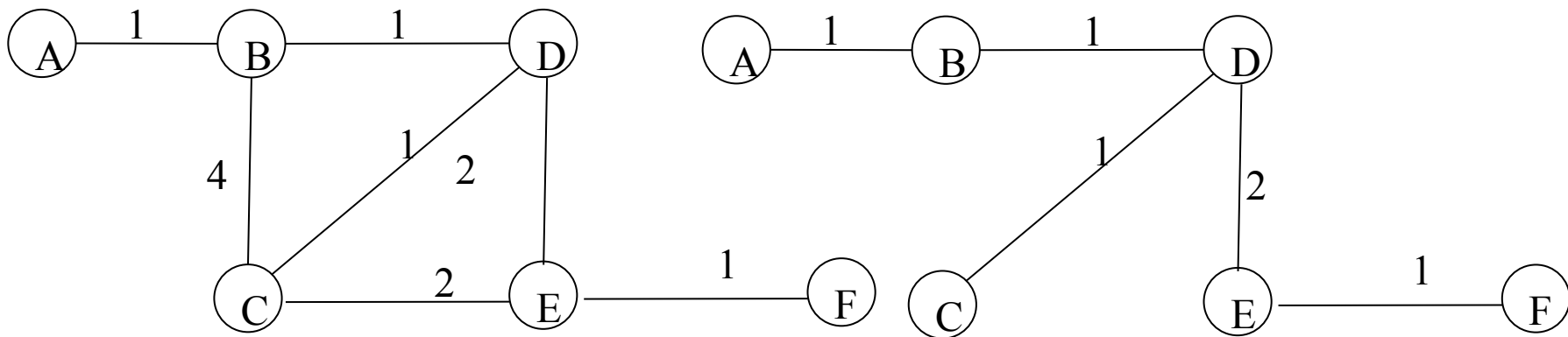


## 得出最短路径

Example: 节点 A 收到如下的LSP，请画出网络拓扑结构，并使用Dijkstra算法来计算到其他节点的最短路径

A: (B,1); B: (A,1), (C,4), (D,1); C: (B,4), (D,1), (E,2);

D: (B,1), (C,1), (E,2); E: (C,2), (D,2), (F,1); F: (E, 1)

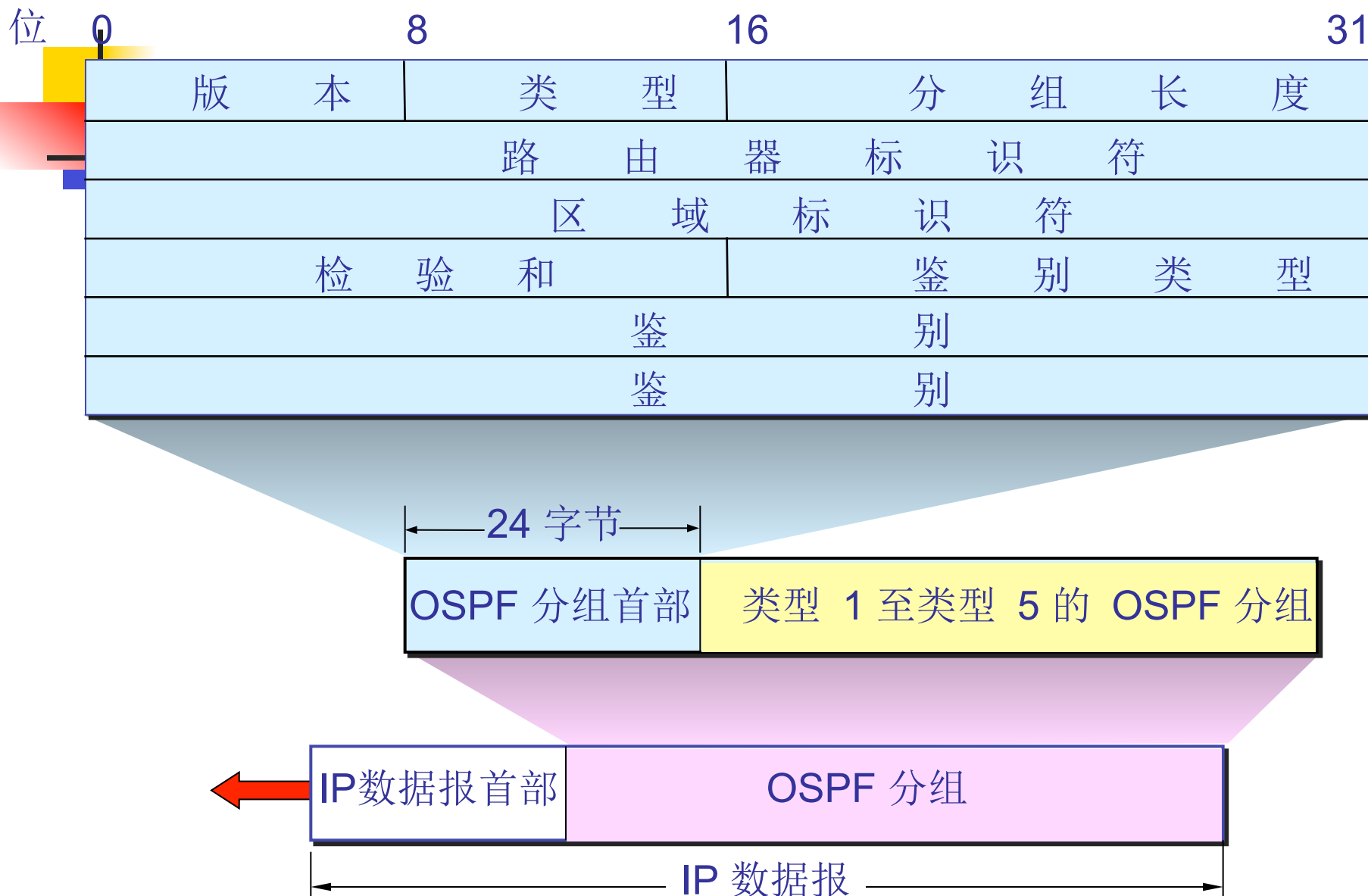




# OSPF 直接用 IP 数据报传送

- OSPF 不用 UDP 而是直接用 IP 数据报传送。
- OSPF 构成的数据报很短。这样做可减少路由信息的通信量。
- 数据报很短的另一好处是可以不必将长的数据报分片传送。分片传送的数据报只要丢失一个，就无法组装成原来的数据报，而整个数据报就必须重传。

# OSPF 分组





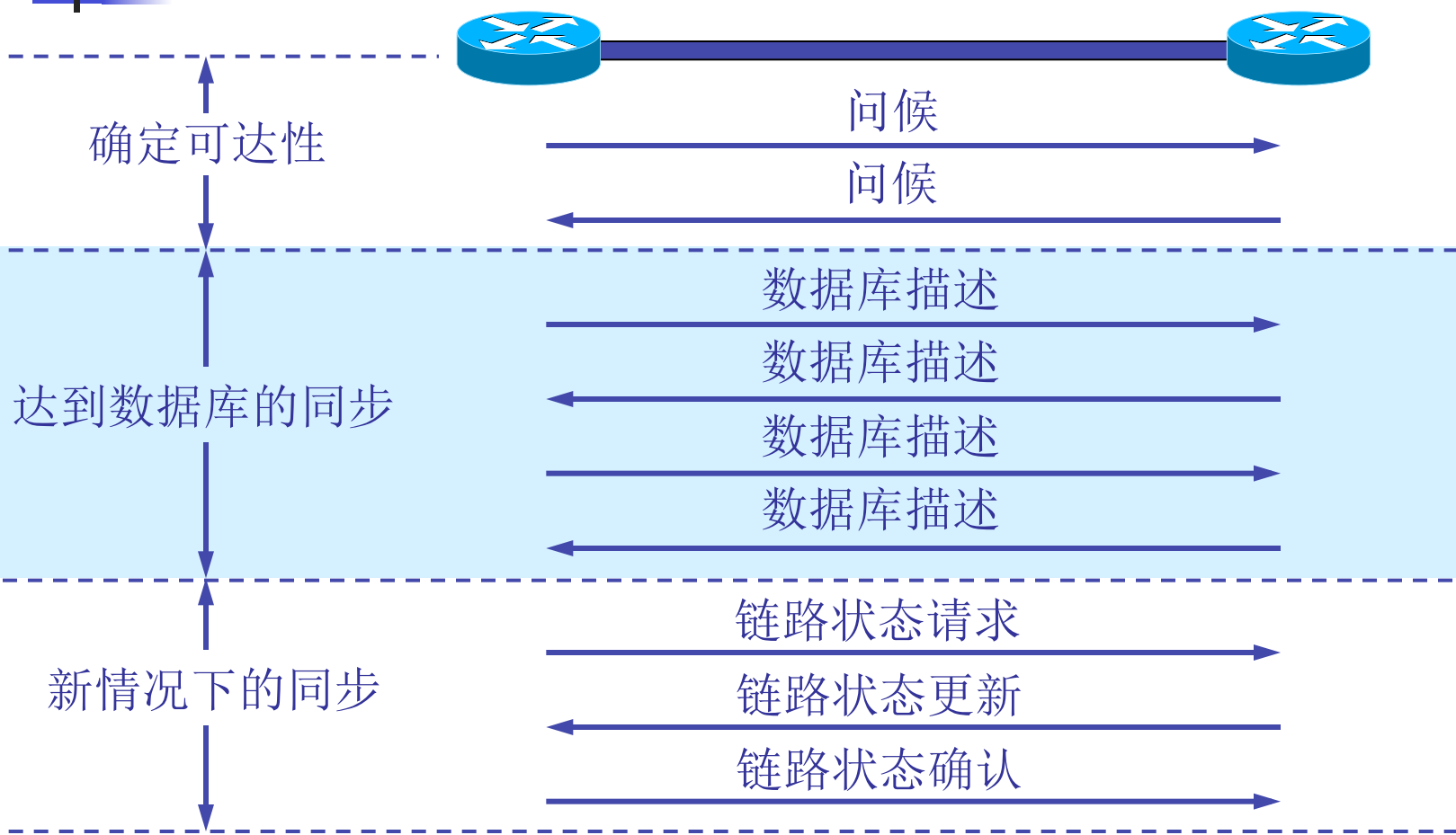
## 2. OSPF 的五种分组类型

---

- 类型1，问候(Hello)分组。
- 类型2，数据库描述(Database Description)分组。
- 类型3，链路状态请求(Link State Request)分组。
- 类型4，链路状态更新(Link State Update)分组，  
用洪泛法对全网更新链路状态。
- 类型5，链路状态确认(Link State Acknowledgment)分组。



# OSPF的基本操作

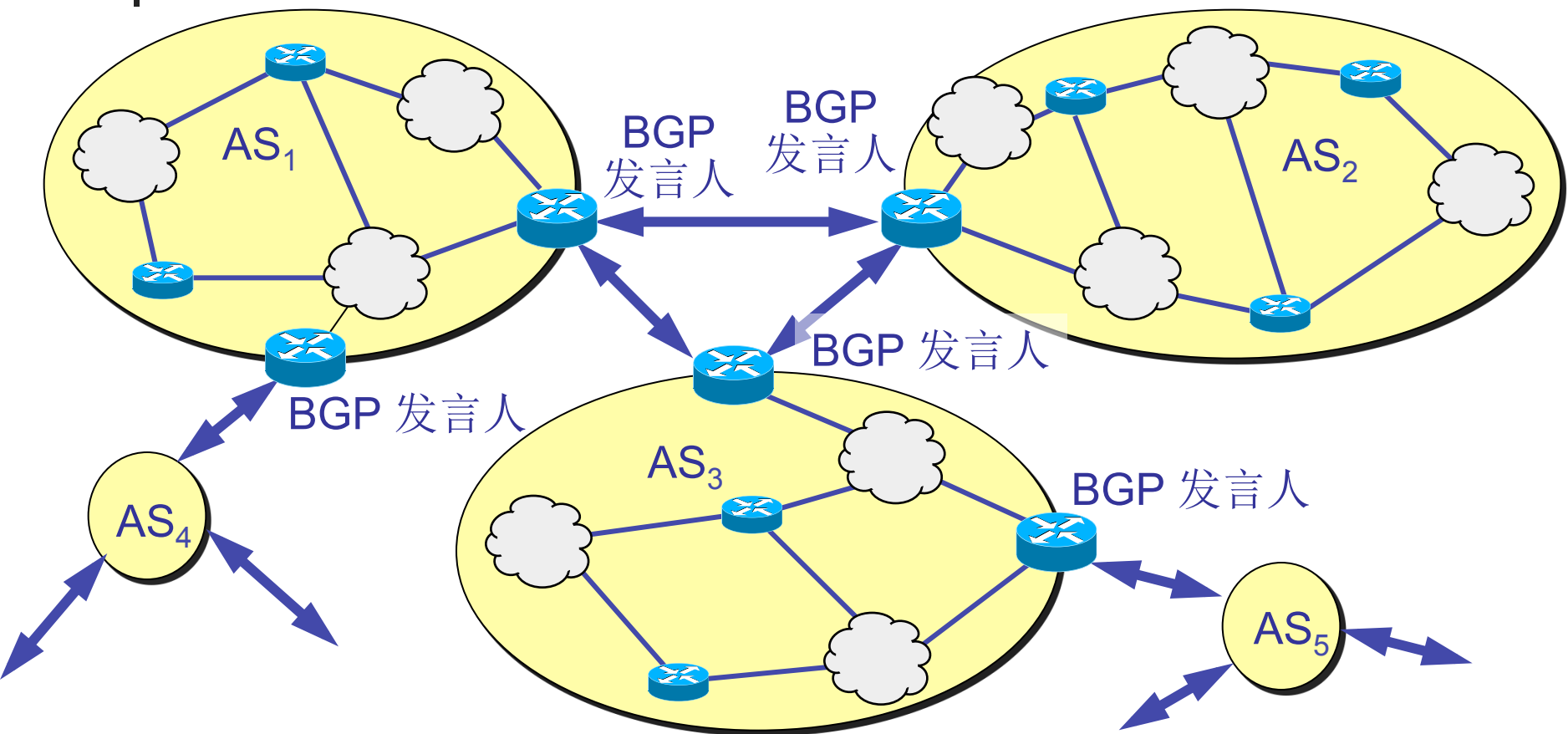




## 4.5.4 外部网关协议 BGP

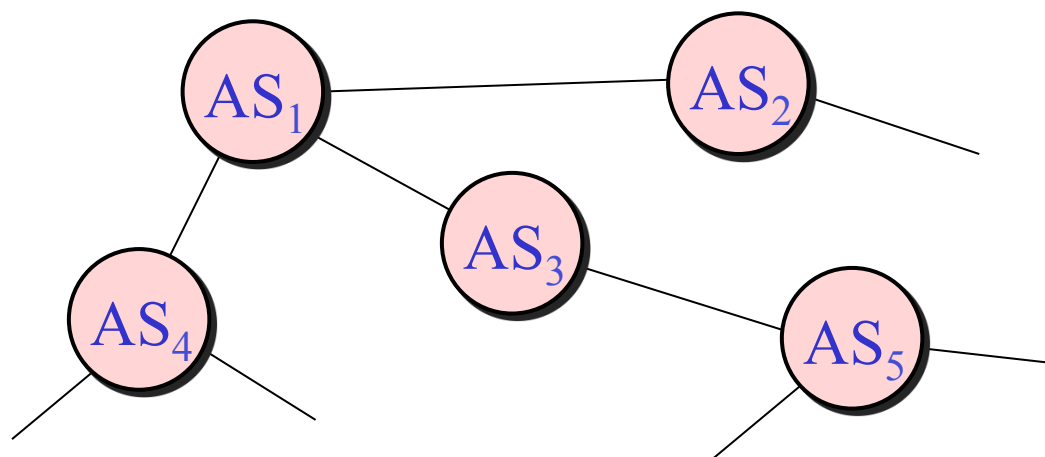
- BGP 是不同AS(自治系统)的路由器之间交换路由信息的协议。
- 互联网的规模太大，使得自治系统之间路由选择非常困难。并且需要考虑政治因素
- 边界网关协议 BGP 只能是力求寻找一条能够到达目的网络且比较好的路由（不能兜圈子），而并非要寻找一条最佳路由。
- 怎么做：
  - AS之间交换路由消息
  - 由AS中与其他AS互联的路由器作为发言人（BGP 发言人）来交换消息
  - 使用TCP连接来进行消息交换

# BGP 发言人和 自治系统 AS 的关系



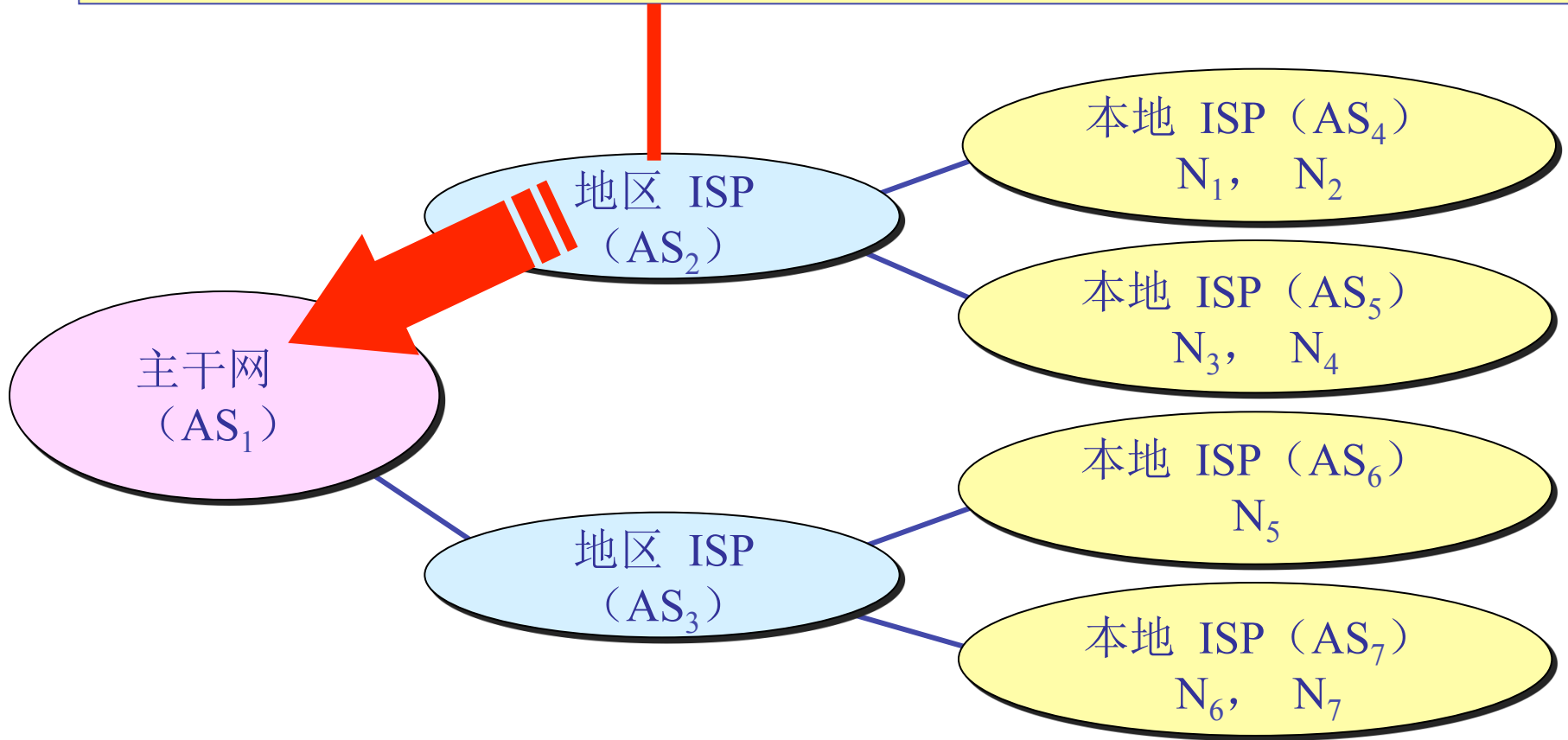
# AS 的连通图举例

- BGP 所交换的网络可达性的信息就是要到达某个网络所要经过的一系列 AS。
- 当 BGP 发言人互相交换了网络可达性的信息后，各 BGP 发言人就根据所采用的策略从收到的路由信息中找出到达各 AS 的较好路由。



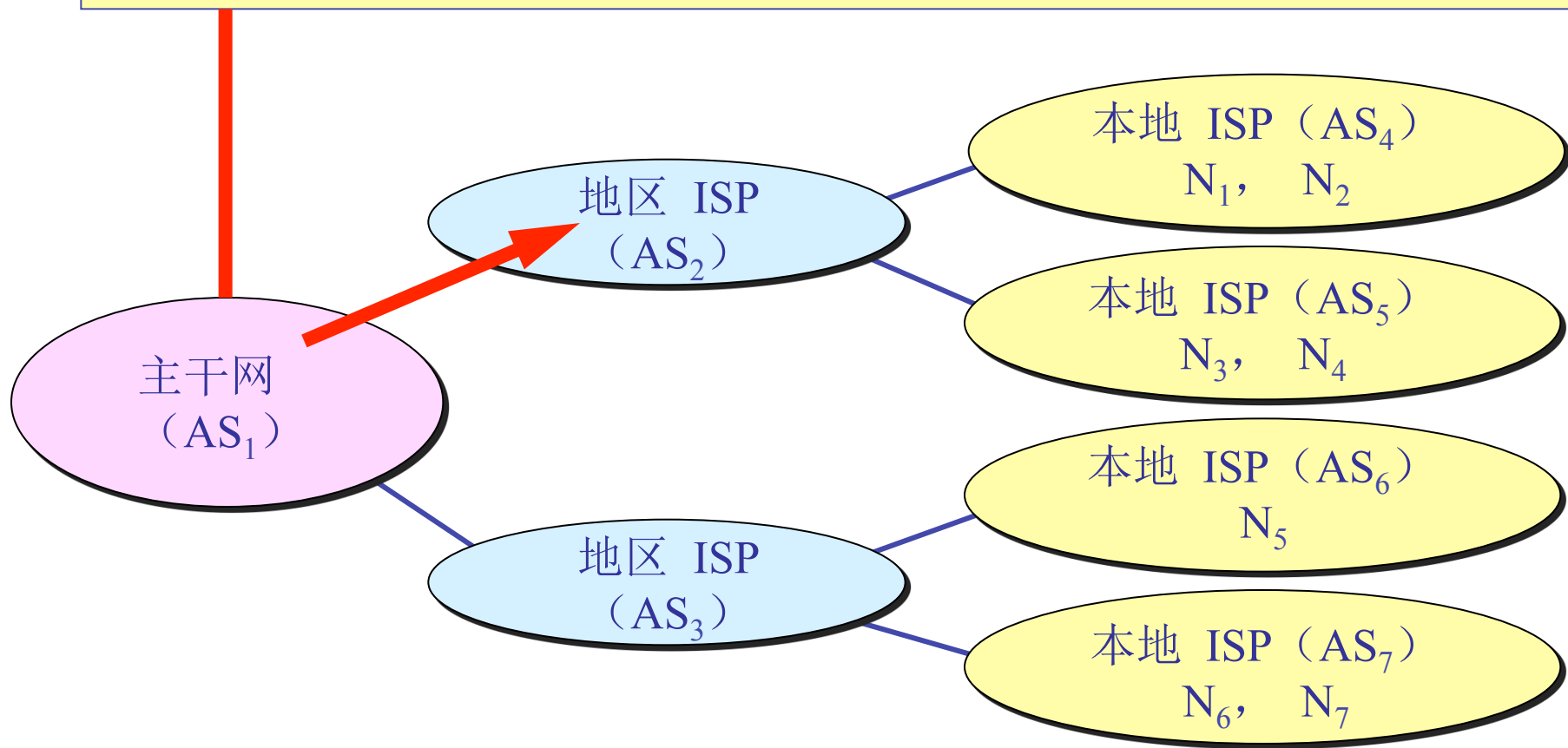
# BGP 发言人交换路径向量

自治系统  $AS_2$  的 BGP 发言人通知主干网的 BGP 发言人：“要到达网络  $N_1, N_2, N_3$  和  $N_4$  可经过  $AS_2$ 。”



# BGP 发言人交换路径向量

主干网还可发出通知：“要到达网络  $N_5, N_6$  和  $N_7$  可沿路径  $(AS_1, AS_3)$ 。”



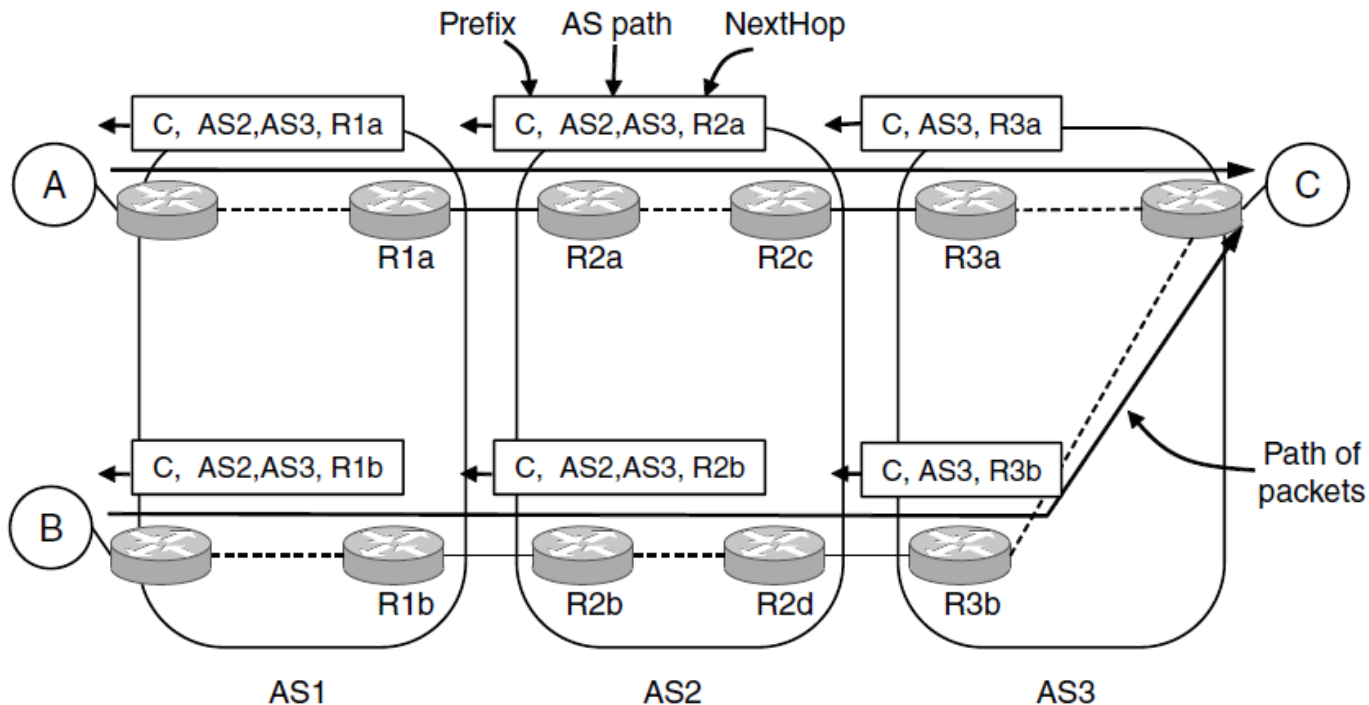


武汉大学

WUHAN UNIVERSITY

# BGP— Exterior Routing Protocol

- BGP 建立路由是通过“通告”的方式，“通过”必须沿着符合政策的路径传播路由消息
  - 消息包含prefix, AS 路径(避免loops)和下一跳(to send over the local network)





# BGP 协议的特点

- BGP 协议交换路由信息的结点数量级是自治系统数的量级，这要比这些自治系统中的网络数少很多。
- 每一个自治系统中 BGP 发言人（或边界路由器）的数目是很少的。这样就使得自治系统之间的路由选择不致过分复杂。





# BGP 协议的特点

- BGP 支持 CIDR，因此 BGP 的路由表也就应当包括目的网络前缀、下一跳路由器，以及到达该目的网络所要经过的各个自治系统序列。
- 在BGP 刚刚运行时，BGP 的邻站是交换整个的 BGP 路由表。但以后只需要在发生变化时更新有变化的部分。这样做对节省网络带宽和减少路由器的处理开销方面都有好处。



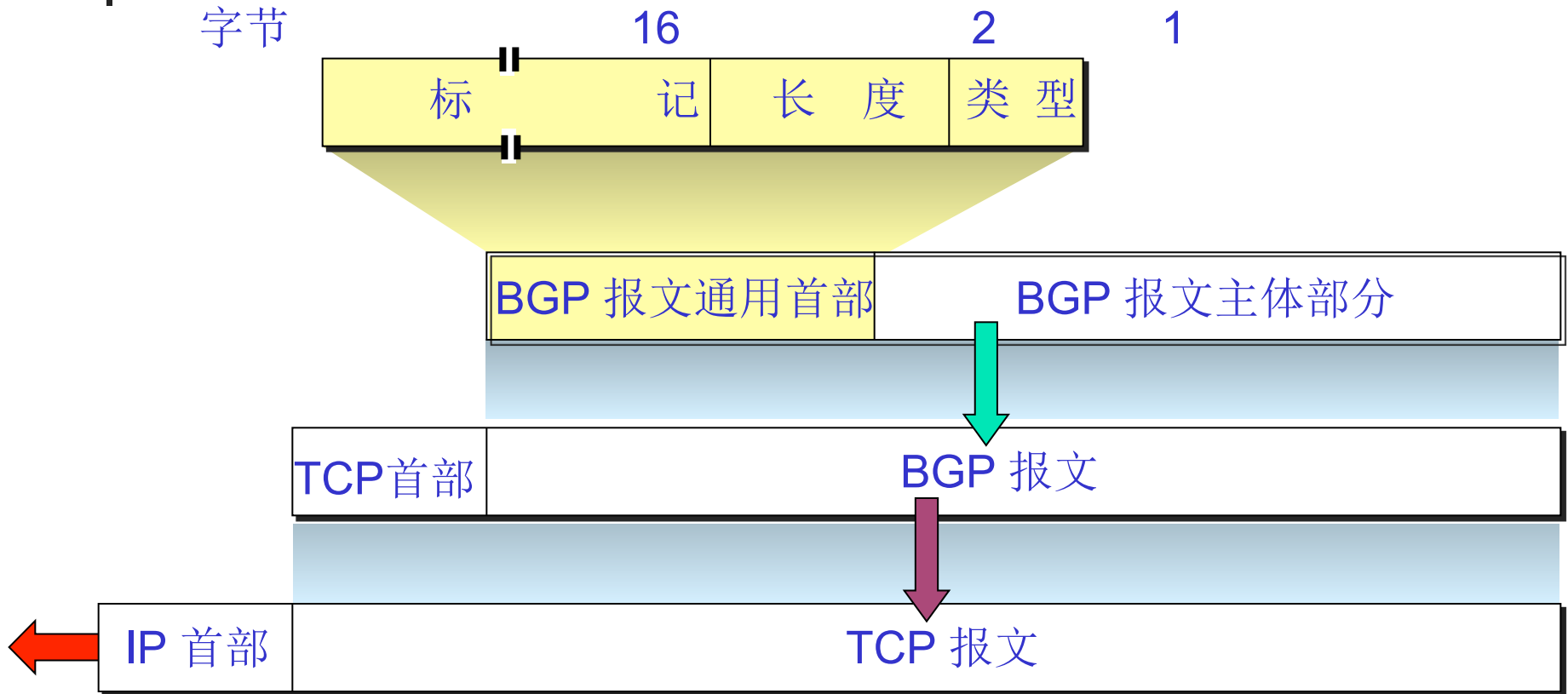
# BGP-4 共使用四种报文

---

- (1) 打开(**OPEN**)报文，用来与相邻的另一个BGP发言人建立关系。
  - (2) 更新(**UPDATE**)报文，用来发送某一路由的信息，以及列出要撤消的多条路由。
  - (3) 保活(**KEEPALIVE**)报文，用来确认打开报文和周期性地证实邻站关系。
  - (4) 通知(**NOTIFICATION**)报文，用来发送检测到的差错。
- 在 RFC 2918 中增加了 ROUTE-REFRESH 报文，用来请求对等端重新通告。

# BGP 报文具有通用的首部

字节





## 4.5.5 路由器的构成

- 整个的路由器结构可划分为两大部分：
  - 路由选择部分
    - 建立和维护路由表
  - 分组转发部分
    - 根据路由表对分组进行转发

# 典型的路由器的结构

3——网络层  
2——数据链路层  
1——物理层

路由选择处理机

路由选择协议

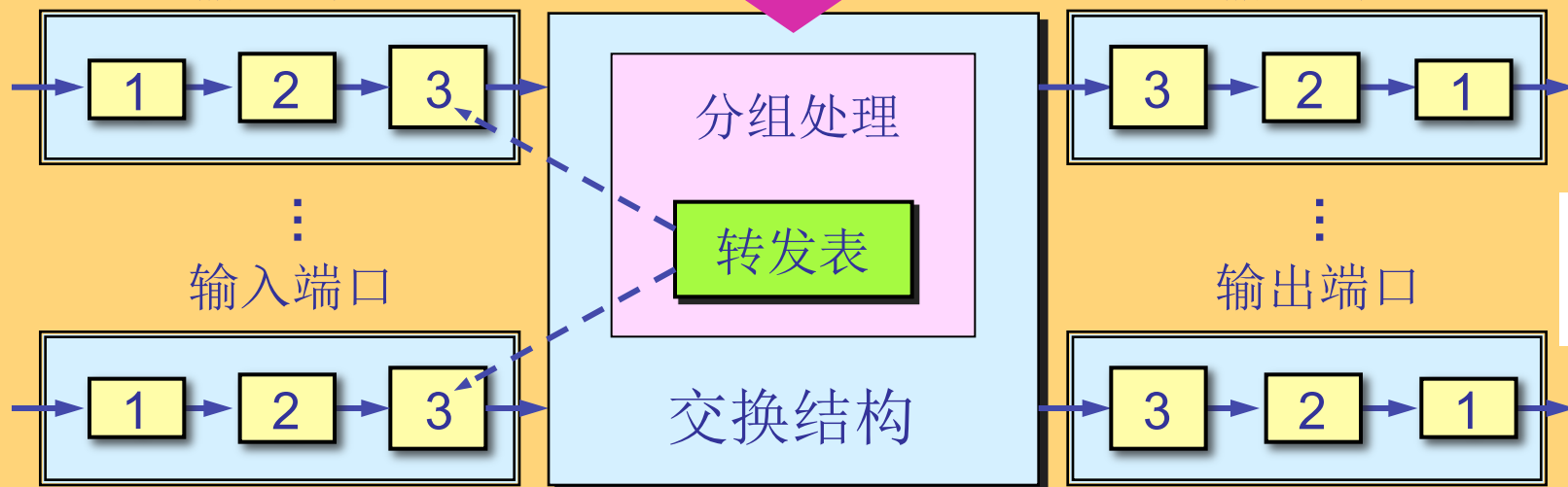
路由表

路由选择

输入端口

输出端口

分组转发





# “转发” 和 “路由选择” 的区别

---

- “**转发**” (forwarding)就是路由器根据转发表将用户的 IP 数据报从合适的端口转发出去。
- “**路由选择**” (routing)则是按照分布式算法, 根据从各相邻路由器得到的关于网络拓扑的变化情况, 动态地改变所选择的路由。
- 路由表是根据路由选择算法得出的。而转发表是从路由表得出的。
- 在讨论路由选择的原理时, 往往不去区分转发表和路由表的区别,



## 分组丢弃

- 若路由器处理分组的速率赶不上分组进入队列的速率，则队列的存储空间最终必定减少到零，这就使后面再进入队列的分组由于没有存储空间而只能被丢弃。
- 路由器中的输入或输出队列产生溢出是造成分组丢失的重要原因。



## 4.6 IPv6

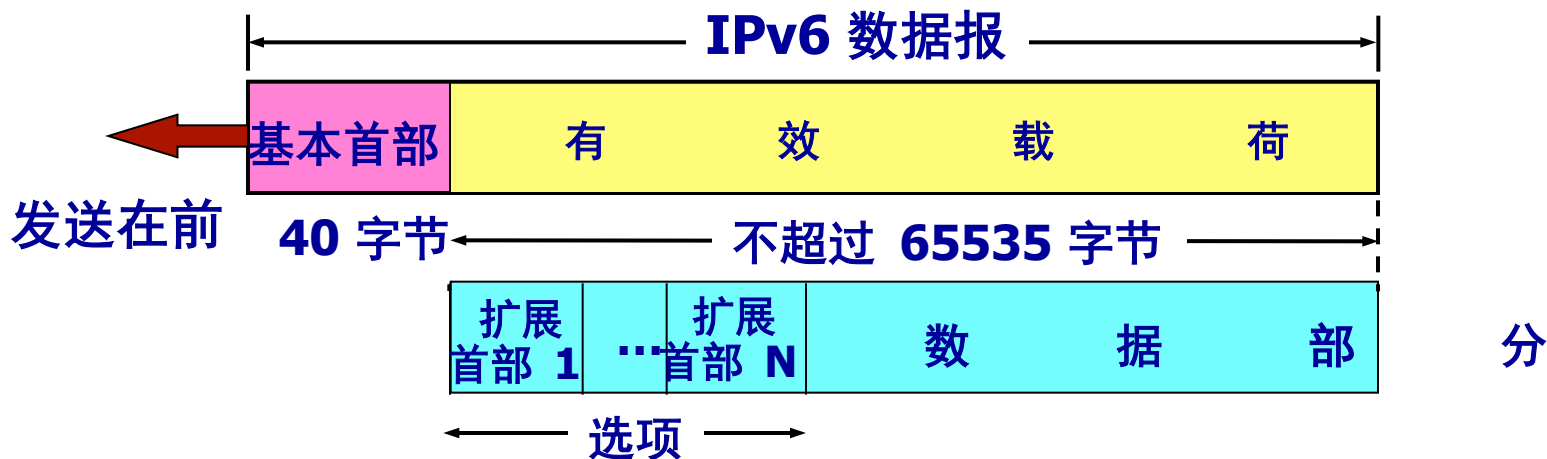
- 解决 IP 地址耗尽的根本措施就是采用具有更大地址空间的新版本的 IP，即 IPv6
- 所引进的主要变化如下：
  - 更大的地址空间。IPv6 将地址从 IPv4 的 32 位 增大到了 128 位。
  - 扩展的地址层次结构。
  - 改进的选项。IPv6 允许数据报包含有选项的控制信息，其选项放在有效载荷中。
  - 支持即插即用（即自动配置）。因此 IPv6 不需要使用 DHCP。





# IPv6 数据报的一般形式

- IPv6 数据报由两大部分组成：
  - 基本首部 (base header)
  - 有效载荷 (payload)。有效载荷也称为净负荷。有效载荷允许有零个或多个扩展首部 (extension header)，再后面是数据部分。



具有多个可选扩展首部的 IPv6 数据报的一般形式



# IPv6 数据报的基本首部

- IPv6 将首部长度变为**固定的 40 字节**，称为**基本首部**。
- 把首部中不必要的功能取消了，使得 IPv6 首部的字段数减少到只有 8 个。
- IPv6 对首部中的某些字段进行了**更改**：



## IPv6简介

### ■ IPv4 vs. IPv6 Header

■ 20字节 vs 40字节

IPv4 Header

0	4	8	12	16	20	24	28	31
Version	IHL	Type of Service		Total Length				
Identification				Flags	Fragment Offset			
Time to Live		Protocol		Header Checksum				
Source Address								
Destination Address								

IPv6 Header

0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	63
Version	Traffic Class		Flow Label					Payload Length					Next Header		Hop Limit	
Source Address																
Destination Address																

- 取消了首部长度的字段，因为首部的长度是固定的 **40** 字节；
- 取消了服务类型字段；
- 取消了总长度字段，改用有效载荷长度字段；

- 把 **TTL** 字段改称为跳数限制字段；
- 取消了协议字段，改用下一个首部字段；
- 取消了检验和字段；
- 取消了选项字段，而用扩展首部来实现选项功能。



## IPv6简介



通信量类(traffic class)—— 8 位。这是为了区分不同的 IPv6 数据报的类别或优先级。目前正在进行不同的通信量类性能的实验。

流标号(flow label)—— 20 位。“流”是互联网络上从特定源点到特定终点的一系列数据报，“流”所经过的路径上的路由器都保证指明的服务质量。所有属于同一个流的数据报都具有同样的流标号。

有效载荷长度(payload length)—— 16 位。它指明 IPv6 数据报除基本首部以外的字节数（所有扩展首部都算在有效载荷之内），其最大值是 64 KB。

下一个首部(next header)—— 8 位。它相当于 IPv4 的协议字段或可选字段。

Hop Limit—— 8 位。它相当于TTL



# IPv6 的扩展首部

- IPv6 把原来 IPv4 首部中选项的功能都放在**扩展首部**中，并将扩展首部留给路径两端的源站和目的站的主机来处理。
- 数据报途中经过的路由器都不处理这些扩展首部（只有一个首部例外，即逐跳选项扩展首部）。
- 这样就**大大提高了路由器的处理效率**。



# 六种扩展首部

在 RFC 2460 中定义了六种扩展首部：

- (1) 逐跳选项
- (2) 路由选择
- (3) 分片
- (4) 鉴别
- (5) 封装安全有效载荷
- (6) 目的站选项

每一个扩展首部都由若干个字段组成，它们的长度也各不相同。但所有扩展首部的第一个字段都是8位的“下一个首部”字段。此字段的值指出了在该扩展首部后面的字段是什么。



## 4.6.2 IPv6 的地址

- IPv6 数据报的目的地址可以是以下三种基本类型地址之一：
  - (1) **单播** (unicast): 传统的点对点通信。
  - (2) **多播** (multicast): 一点对多点的通信。
  - (3) **任播** (anycast): 这是 IPv6 增加的一种类型。任播的目的站是一组计算机，但数据报在交付时只交付其中的一个，通常是距离最近的一个。



# IPv6简介

## ■ IPv4 vs. IPv6

零压缩

注：只能使用一次零压缩

An IPv4 address (dotted-decimal notation)

**172 . 16 . 254 . 1**



10101100.00010000.11111110.00000001



One byte = Eight bits

Thirty-two bits (  $4 * 8$  ), or 4 bytes

An IPv6 address (in hexadecimal)

**2001:0DB8:AC10:FE01:0000:0000:0000:0000**



**2001:0DB8:AC10:FE01::** Zeroes can be omitted



1000000000000001:0000110110111000:1010110000010000:1111111000000001:

0000000000000000:0000000000000000:0000000000000000:0000000000000000





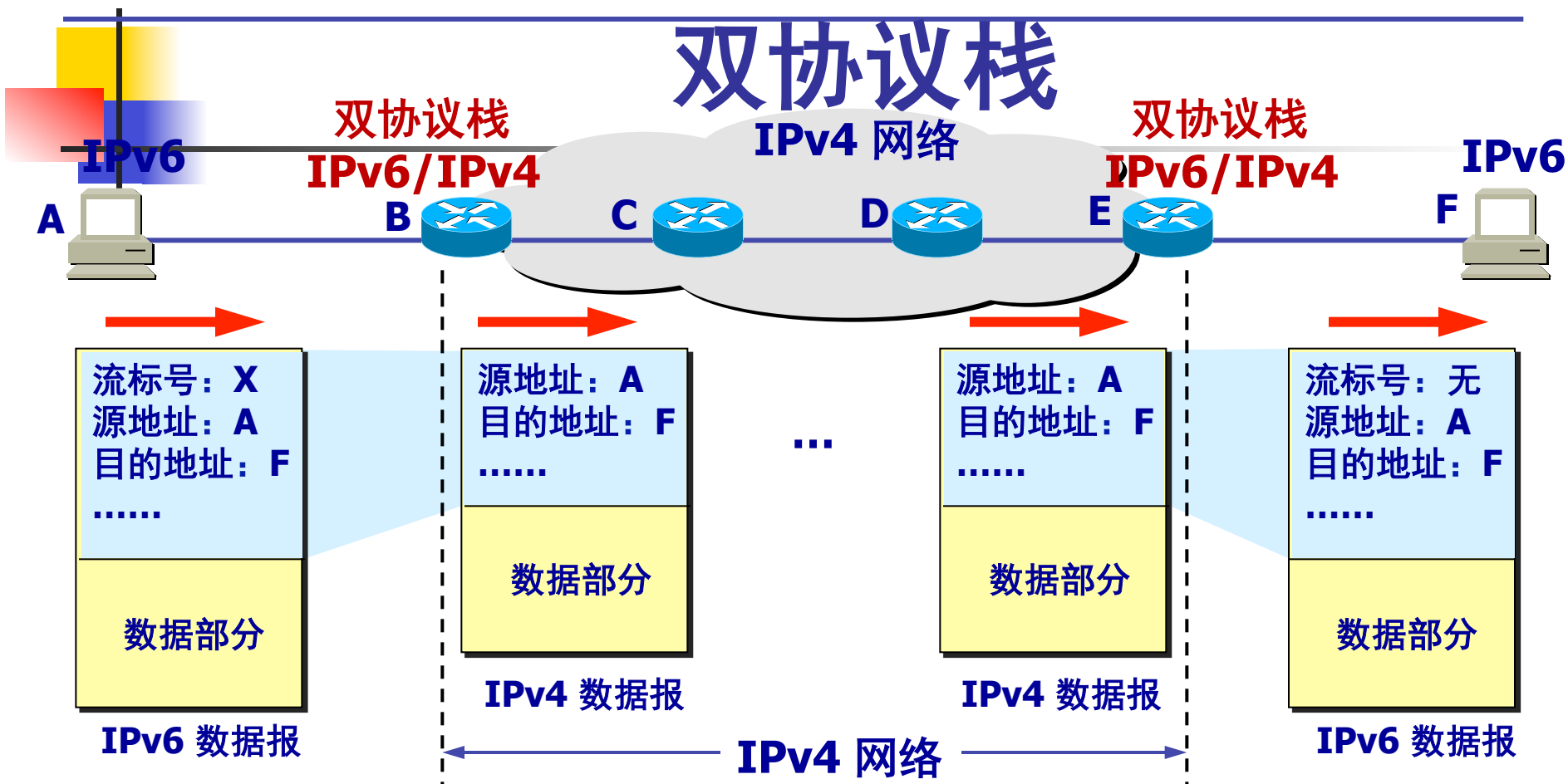
# 点分十进制记法的后缀

- 冒号十六进制记法可结合使用点分十进制记法的后缀，这种结合在 IPv4 向 IPv6 的转换阶段特别有用。
- 例如：0:0:0:0:0:0:128.10.2.1  
再使用零压缩即可得出： ::128.10.2.1
- CIDR 的斜线表示法仍然可用。
- 例如：60 位的前缀 12AB00000000CD3 可记为：  
12AB:0000:0000:CD30:0000:0000:0000:0000/60  
或 12AB::CD30:0:0:0:0/60 （零压缩）  
或 12AB:0:0:CD30::/60 （零压缩）



## 4.6.3 从 IPv4 向 IPv6 过渡

- 向 IPv6 过渡只能采用逐步演进的办法，同时，还必须使新安装的 IPv6 系统能够向后兼容：IPv6 系统必须能够接收和转发 IPv4 分组，并且能够为 IPv4 分组选择路由。
- 两种向 IPv6 过渡的策略：
  - 使用双协议栈
  - 使用隧道技术

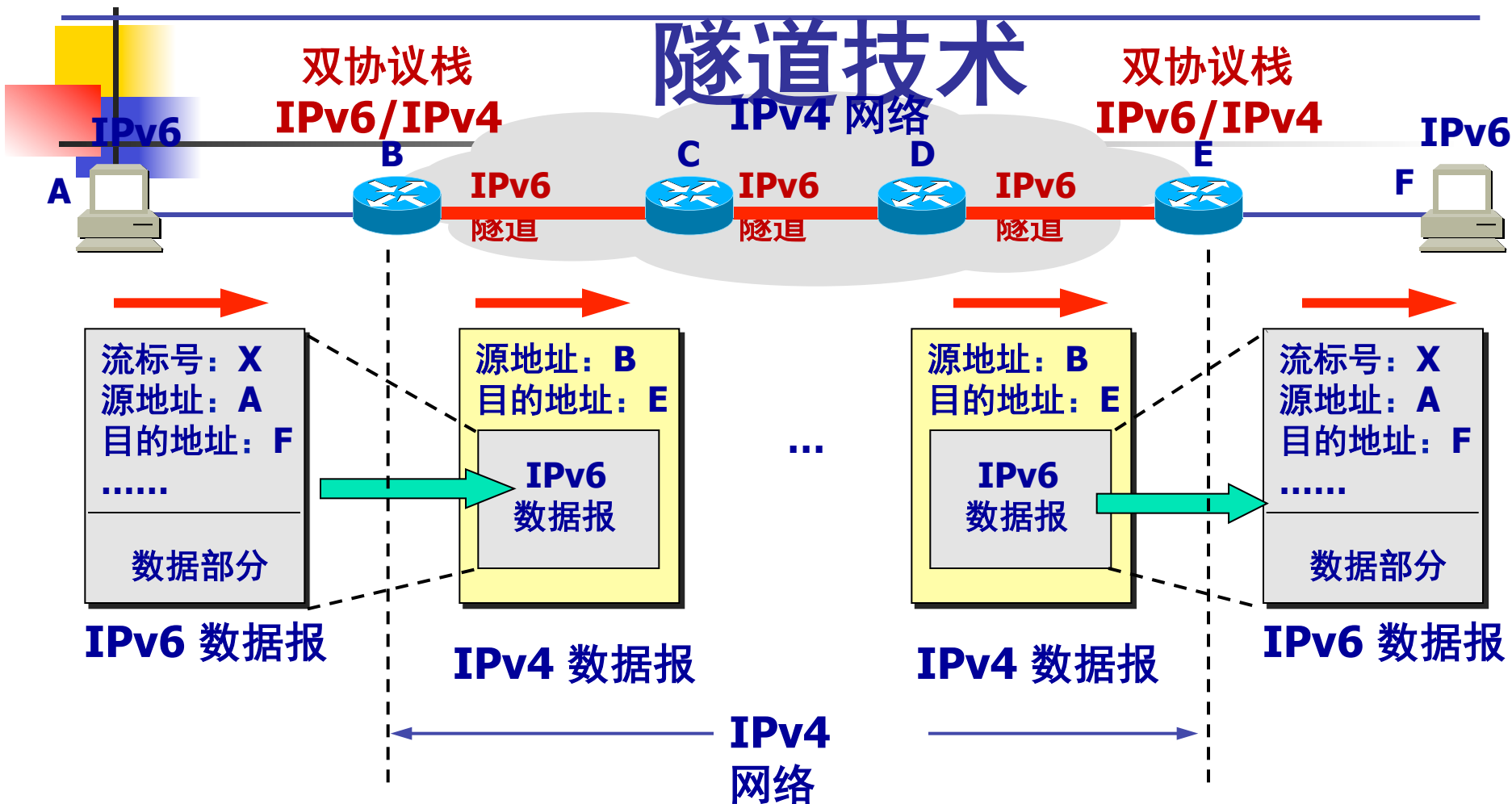


使用双协议栈进行从 **IPv4** 到 **IPv6** 的过渡



# 隧道技术

- 在 IPv6 数据报要进入 IPv4 网络时，把 IPv6 数据报封装成为 IPv4 数据报，整个的 IPv6 数据报变成了 IPv4 数据报的数据部分。
- 当 IPv4 数据报离开 IPv4 网络中的隧道时，再把数据部分（即原来的 IPv6 数据报）交给主机的 IPv6 协议栈。



使用隧道技术进行从 **IPv4** 到 **IPv6** 的过渡



## 4.7.1 IP 多播的基本概念

- IP 多播 (multicast, 以前曾译为组播) 已成为互联网的一个热门课题。
- 目的: 更好第支持一对多通信。
- 一对多通信: 一个源点发送到许多个终点。
  - 例如, 实时信息的交付 (如新闻、股市行情等), 软件更新, 交互式会议及其他多媒体通信。

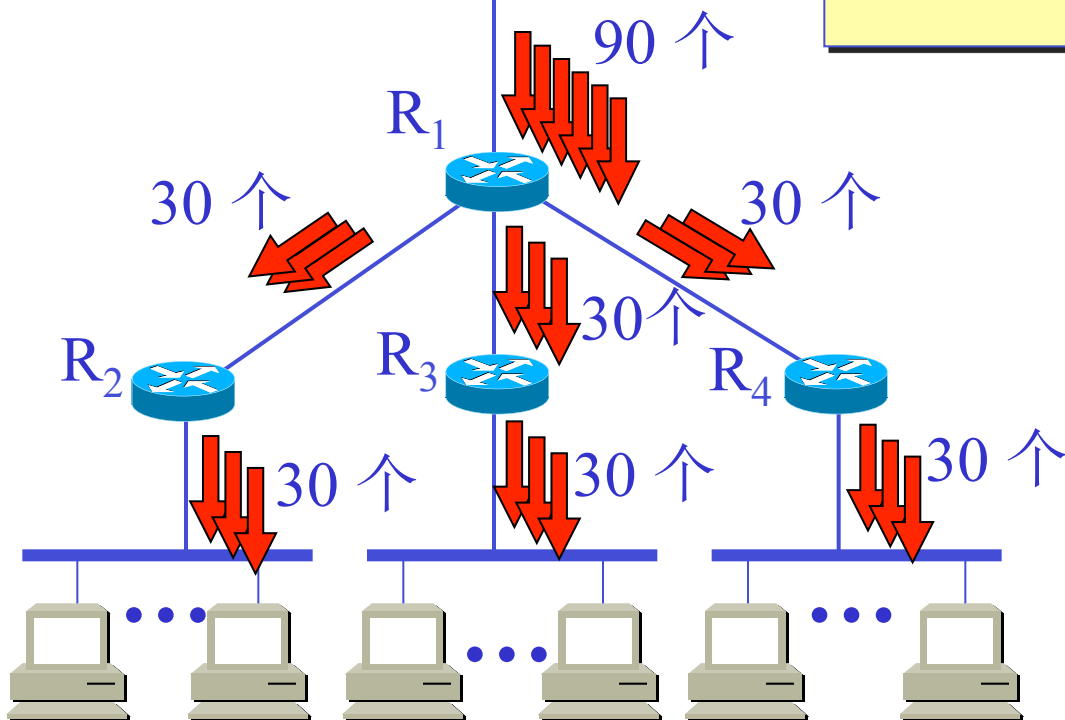


## 4.7.1 IP 多播的基本概念

视频服务器



不使用多播时需要  
发送 90 次单播



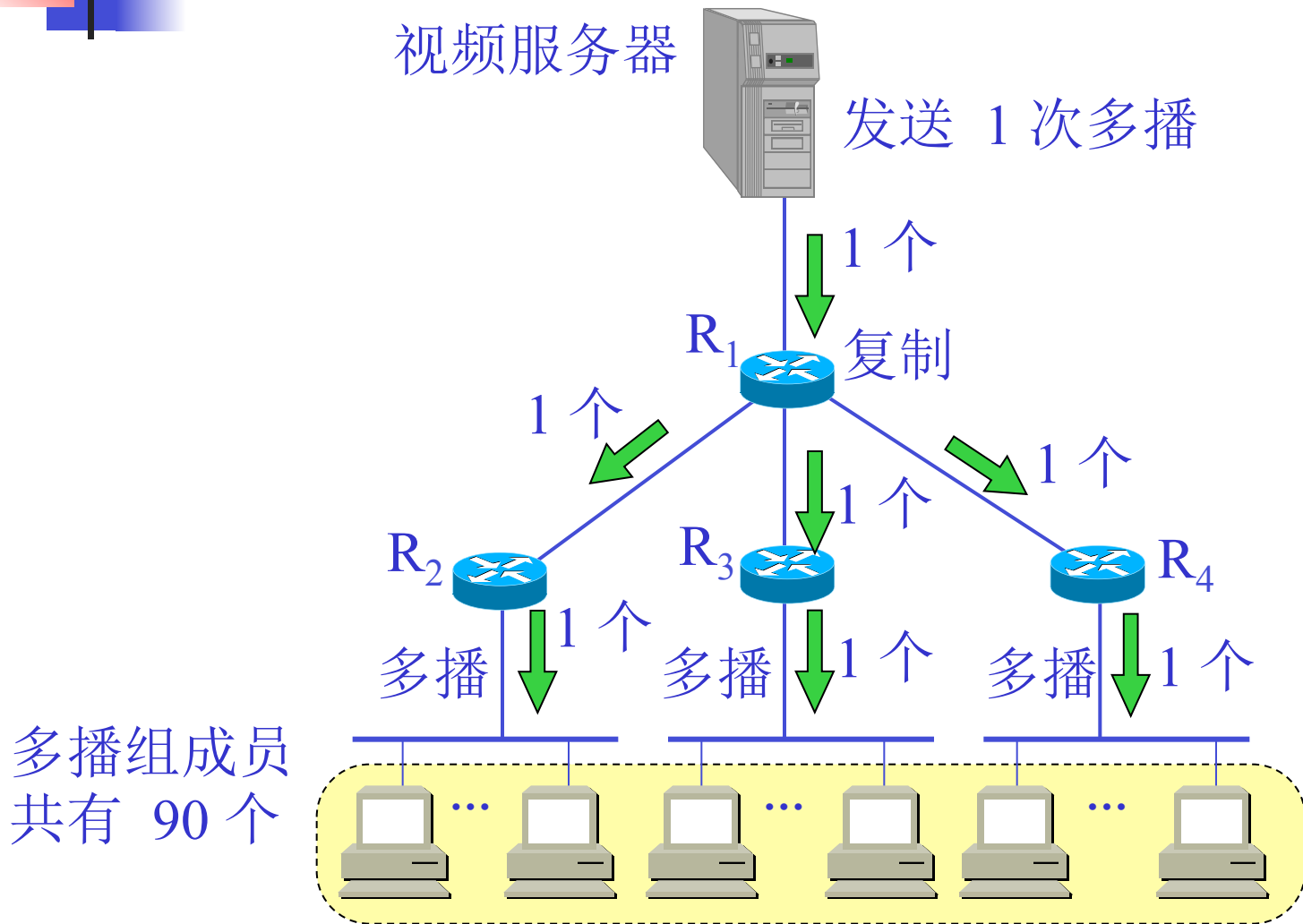
共有 90 个主机接收视频节目



# 多播可明显地减少 网络中资源的消耗

视频服务器

发送 1 次多播







# IP 多播的一些特点

- (1) 多播使用组地址——IP 使用 D 类地址支持多播。多播地址只能用于目的地址，而不能用于源地址。
- (2) 永久组地址——由因特网号码指派管理局 IANA 负责指派。
- (3) 动态的组成员
- (4) 使用硬件进行多播



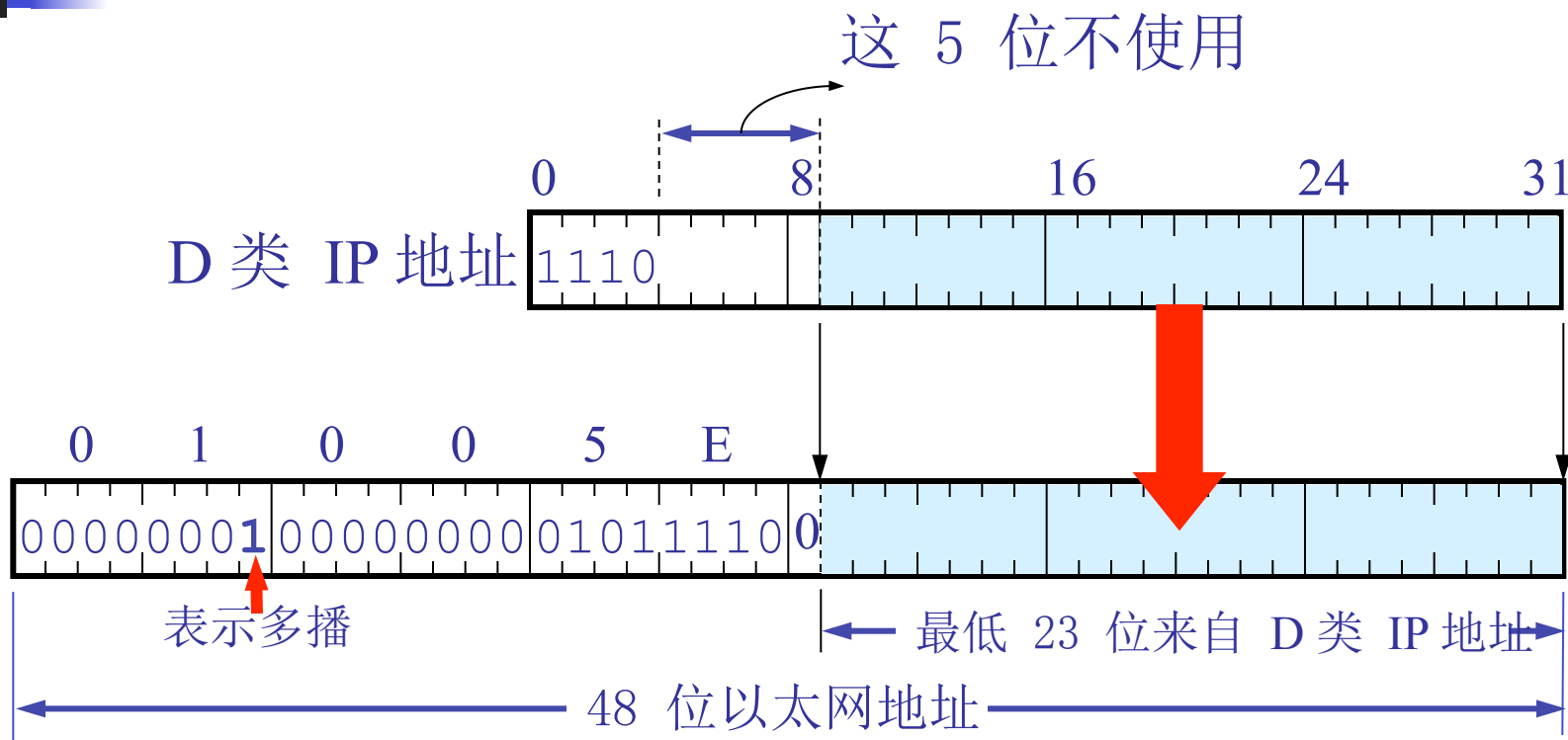
## 4.6.2 在局域网上进行硬件多播

- 因特网号码指派管理局 IANA 拥有的以太网地址块的高 24 位为 00-00-5E。
- 因此 TCP/IP 协议使用的以太网多播地址块的范围是：从 00-00-5E-00-00-00  
到 00-00-5E-FF-FF-FF
- D 类 IP 地址可供分配的有 28 位，在这 28 位中的前 5 位不能用来构成以太网硬件地址。



# D 类 IP 地址

## 与以太网多播地址的映射关系



由于多播 IP 地址与以太网硬件地址的映射关系不是唯一的，因此收到多播数据报的主机，还要在 IP 层利用软件进行过滤，把不是本主机要接收的数据报丢弃。



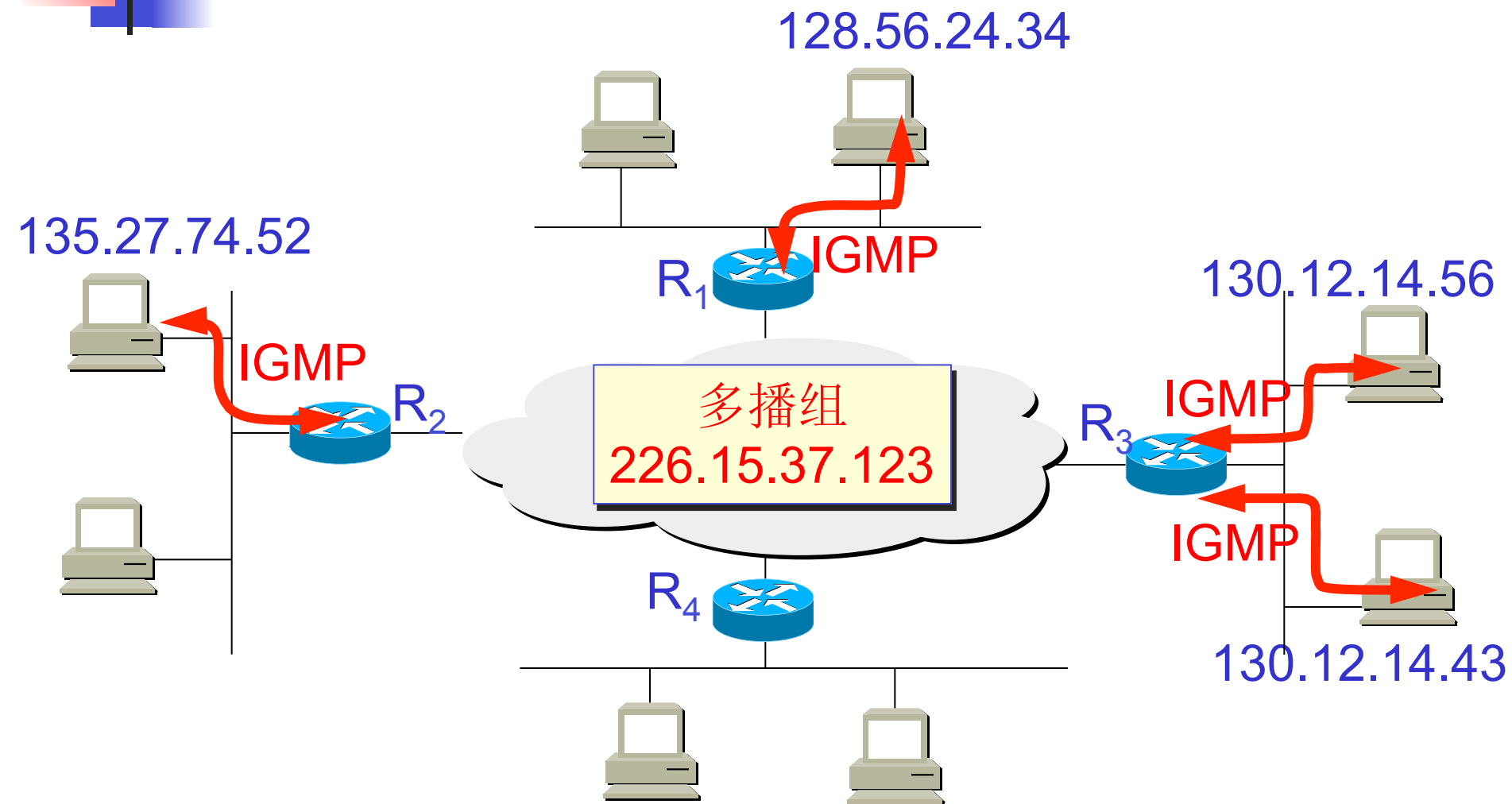
## 4.7.3 网际组管理协议 IGMP 和多播路由选择协议

---

### 1. IP 多播需要两种协议

- **网际组管理协议 IGMP** (Internet Group Management Protocol): 让路由器知道成员信息, 所以IGMP的作用范围只在主机和其路由器之间。
- **多播路由选择协议**: 多播路由器还必须和因特网上的其他多播路由器协同工作, 以便把多播数据报传送给所有的组成员

# IGMP 使多播路由器 知道多播组成员信息





# IGMP 的本地使用范围

---

- IGMP 并非在因特网范围内对所有多播组成员进行管理的协议。
- IGMP 不知道 IP 多播组包含的成员数，也不知道这些成员都分布在哪些网络上。
- IGMP 协议是让连接在本地局域网上的多播路由器知道本局域网上是否有主机（严格讲，是主机上的某个进程）参加或退出了某个多播组。



# IGMP 是整个网际协议 IP 的一个组成部分

---

- 和 ICMP 相似，IGMP 使用 IP 数据报传递其报文（即 IGMP 报文加上 IP 首部构成 IP 数据报），但它也向 IP 提供服务。
- 因此，我们不把 IGMP 看成是一个单独的协议，而是属于整个网际协议 IP 的一个组成部分。



# IGMP 工作可分为两个阶段

- 第一阶段：加入多播组。
  - 当某个主机加入新的多播组时，该主机应向多播组的多播地址发送 **IGMP** 报文，**声明**自己要成为该组的成员。
  - 本地的多播路由器收到 **IGMP** 报文后，将组成员关系转发给互联网上的其他多播路由器。





# IGMP 可分为两个阶段

- 第二阶段： 探测组成员变化情况。
  - 因为组成员关系是动态的，因此本地多播路由器要周期性地探测本地局域网上的主机，以便知道这些主机是否还继续是组的成员。
  - 只要对某个组有一个主机响应，那么多播路由器就认为这个组是活跃的。
  - 但一个组在经过几次的探测后仍然没有一个主机响应，则不再将该组的成员关系转发给其他的多播路由器。



### 3. 多播路由选择

- 多播路由选择实际上就是要找出以源主机为根结点的多播转发树。
- 多播路由选择协议在转发多播数据报时使用三种方法：
  - (1) 洪泛与剪除
  - (2) 隧道技术 (tunneling)
  - (3) 基于核心的发现技术



## (1) 洪泛与剪除

- 这种方法适合于较小的多播组，而所有的组成员接入的局域网也是相邻接的。
- 一开始，路由器转发多播数据报使用洪泛的方法（这就是广播）。
- 为了避免兜圈子，采用了叫做反向路径广播 **RPB** (Reverse Path Broadcasting) 的策略。

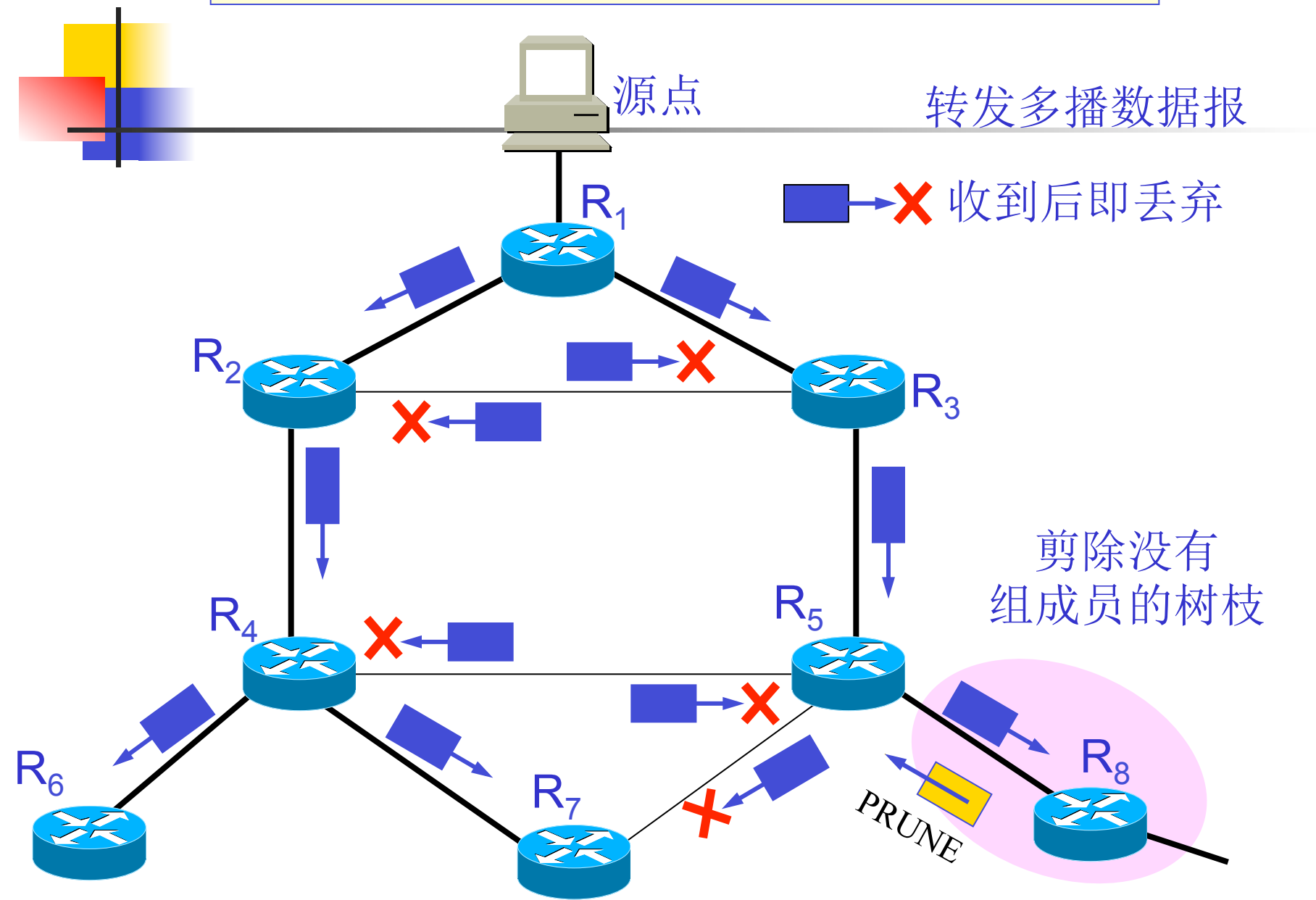


# RPB 的要点

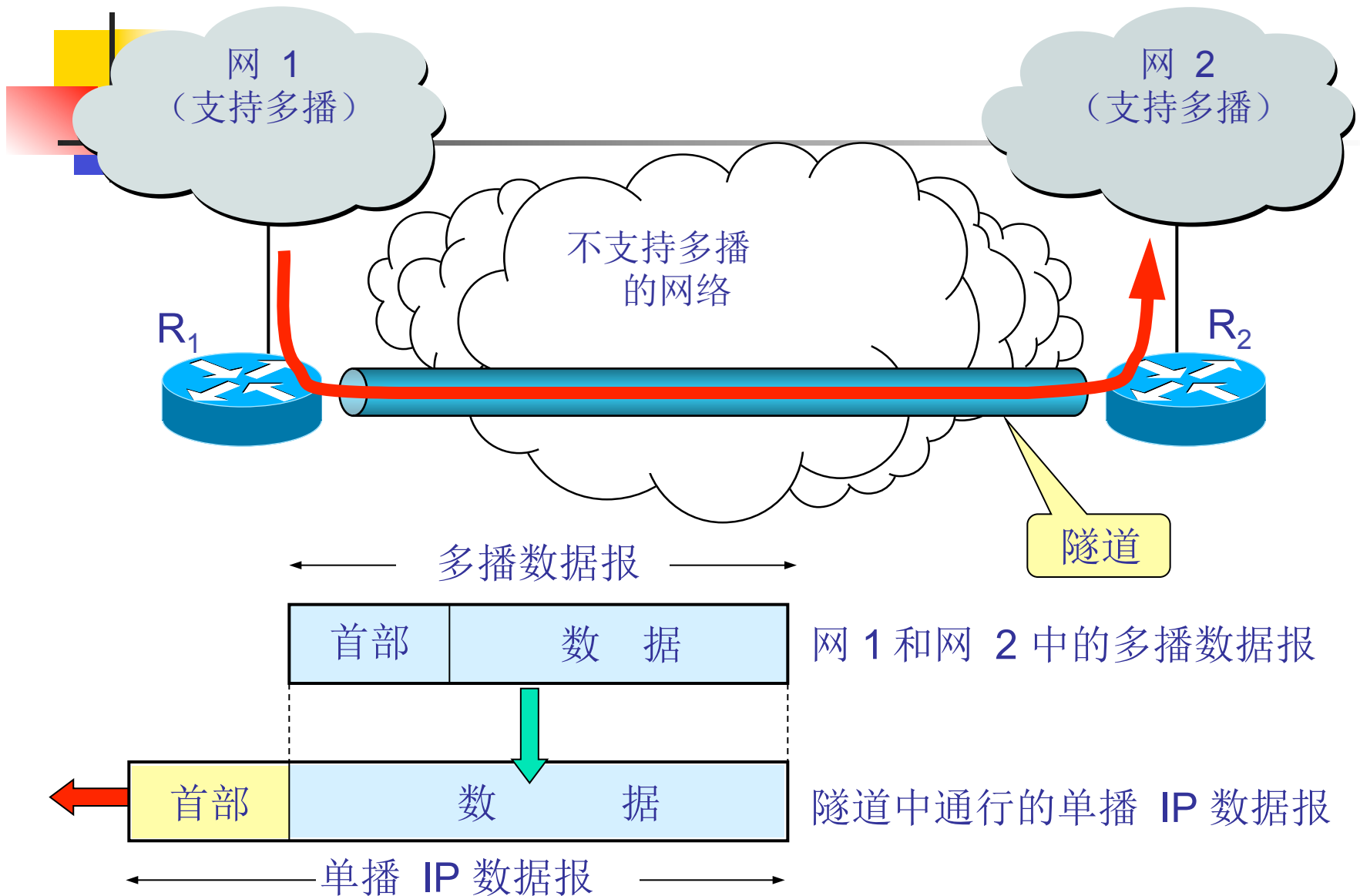
---

- 路由器收到多播数据报时，先检查是否从源点经最短路径传送来的。
- 若是，就向所有其他方向转发刚才收到的多播数据报（但进入的方向除外），否则就丢弃而不转发。

# 反向路径广播 RPB 和剪除

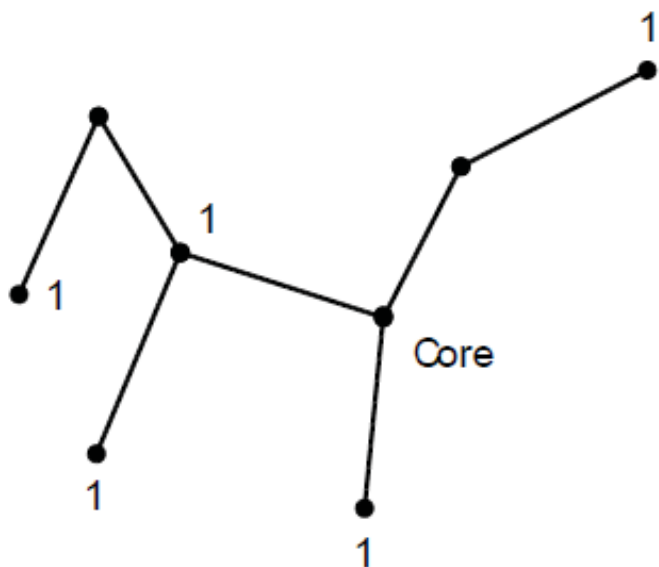


## (2) 隧道技术(tunneling)

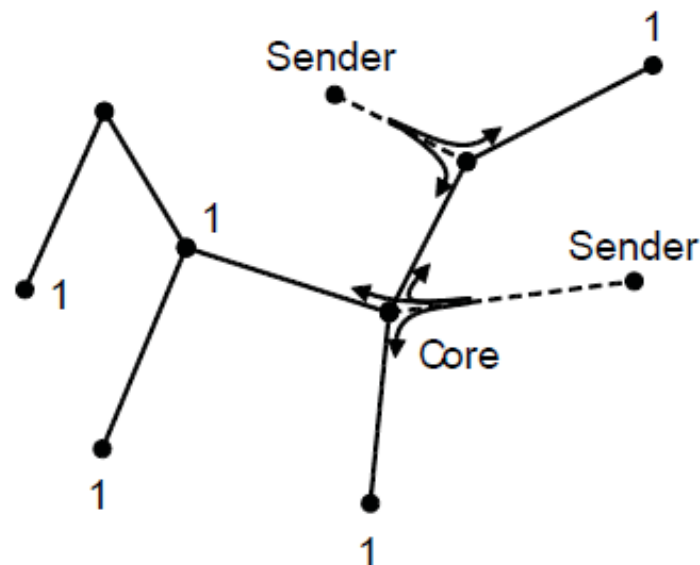


### (3) 基于核心的发现技术

- 每个组里的节点都可能成为源节点，这样就需要为每个节点都建立一个生成树，开销很大。
- CBT (Core-Based Tree) 使用一颗树给所有的组员
  - CBT是一个以core节点为根、包含所有组员的生成树
  - 当一个组员要发送数据时，先将数据沿着树发送给core，再通过core到达所有其他组员



Sink tree from core to group 1



Multicast is send to the core then down when it reaches the sink tree



# 几种多播路由选择协议

---

- 距离向量多播路由选择协议 DVMRP (Distance Vector Multicast Routing Protocol)
- 基于核心的转发树 CBT (Core Based Tree)
- 开放最短通路优先的多播扩展 MOSPF (Multicast Extensions to OSPF)
- 协议无关多播-稀疏方式 PIM-SM (Protocol Independent Multicast-Sparse Mode)
- 协议无关多播-密集方式 PIM-DM (Protocol Independent Multicast-Dense Mode)



## 4.7 虚拟专用网 VPN 和网络地址转换 NAT

### 4.7.1 虚拟专用网 VPN

---

- **本地地址**——仅在机构内部使用的 IP 地址，可以由本机构自行分配，而不需要向因特网的管理机构申请。
- **全球地址**——全球唯一的IP地址，必须向因特网的管理机构申请。



# RFC 1918 指明的专用地址 (private address)

---

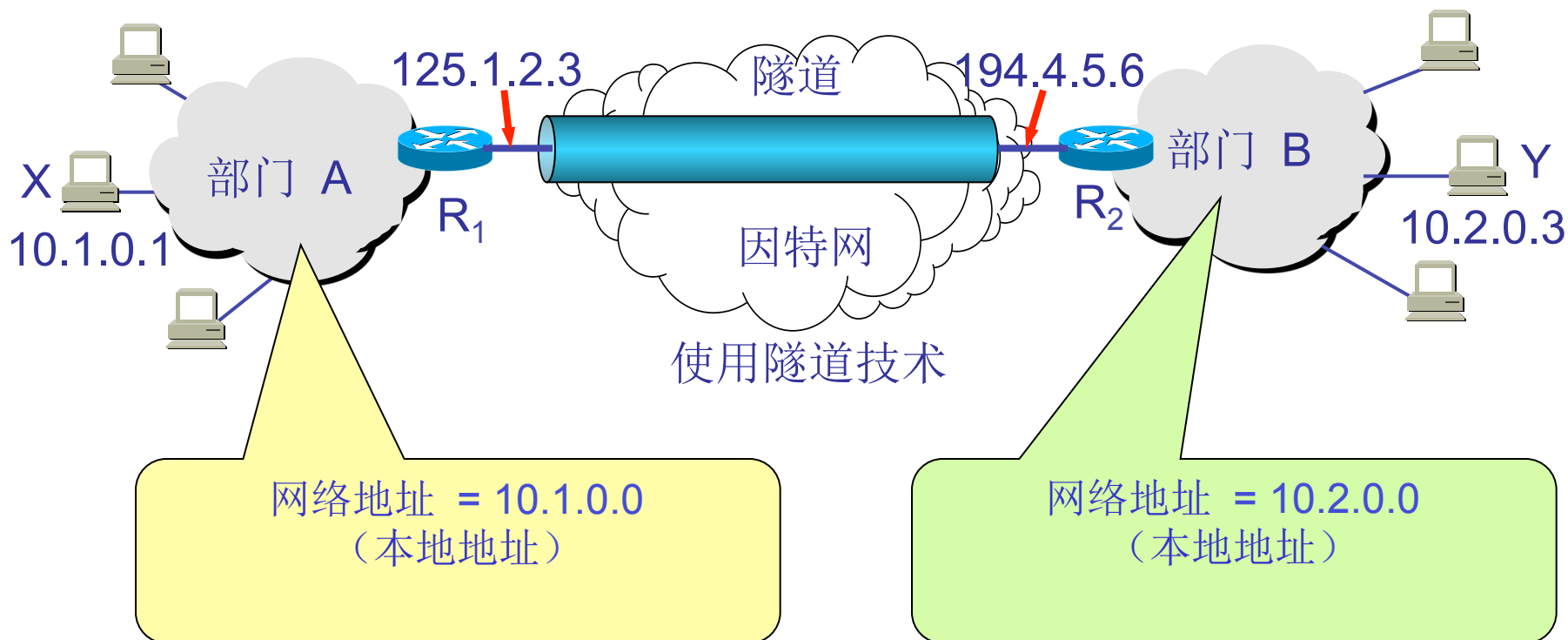
- 10.0.0.0 到 10.255.255.255
- 172.16.0.0 到 172.31.255.255
- 192.168.0.0 到 192.168.255.255
- 这些地址只能用于一个机构的内部通信，而不能用于和因特网上的主机通信。
- 专用地址只能用作本地地址而不能用作全球地址。在因特网中的所有路由器对目的地址是专用地址的数据报一律不进行转发。

# 用隧道技术实现虚拟专用网

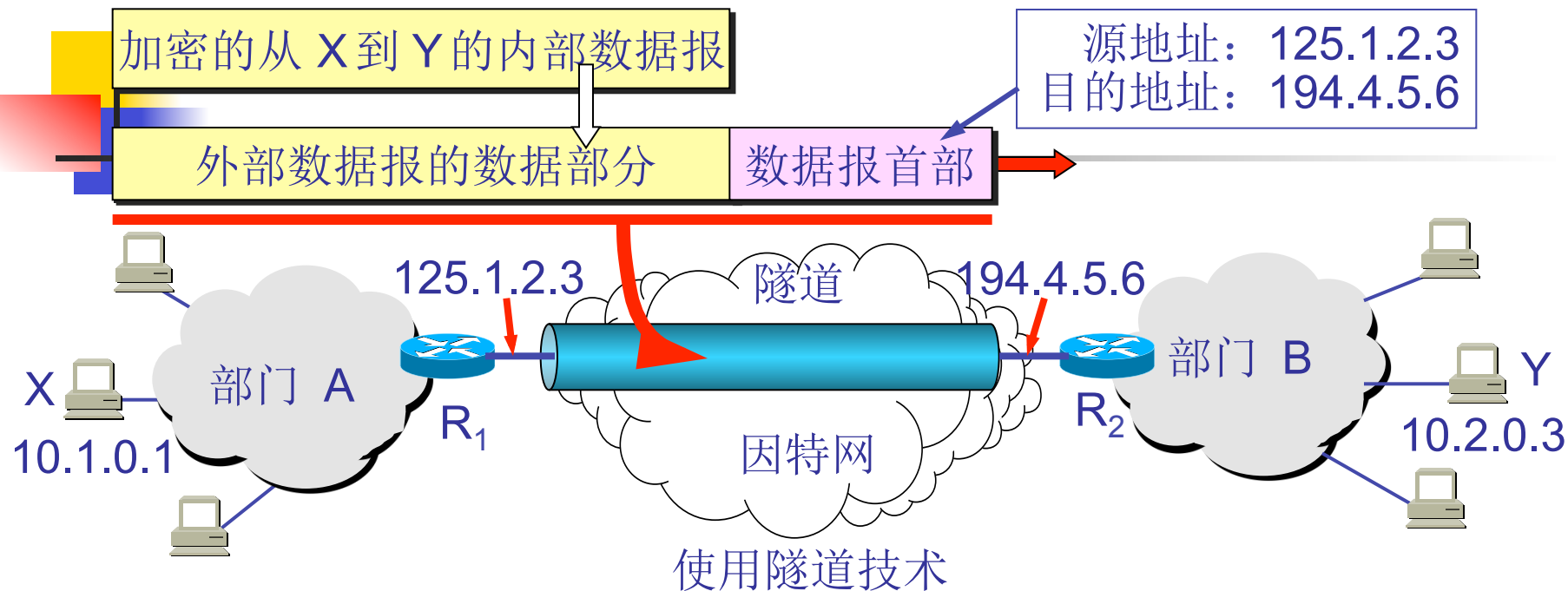
本地地址

全球地址

本地地址



# 用隧道技术实现虚拟专用网



# 内联网 intranet 和外联网 extranet

(都是基于 TCP/IP 协议)

- 由部门 A 和 B 的内部网络所构成的虚拟专用网 VPN 又称为**内联网**(intranet), 表示部门 A 和 B 都是在**同一个**机构的内部。
- 一个机构和某些**外部机构**共同建立的虚拟专用网 VPN 又称为**外联网**(extranet)。





# 远程接入VPN

## (remote access VPN)

---

- 有的公司可能没有分布在不同场所的部门，但有很多流动员工在外地工作。公司需要和他们保持联系，远程接入 VPN 可满足这种需求。
- 在外地工作的员工拨号接入因特网，而驻留在员工 PC 机中的 VPN 软件可在员工的 PC 机和公司的主机之间建立 VPN 隧道，因而外地员工与公司通信的内容是保密的，员工们感到好像就是使用公司内部的本地网络。

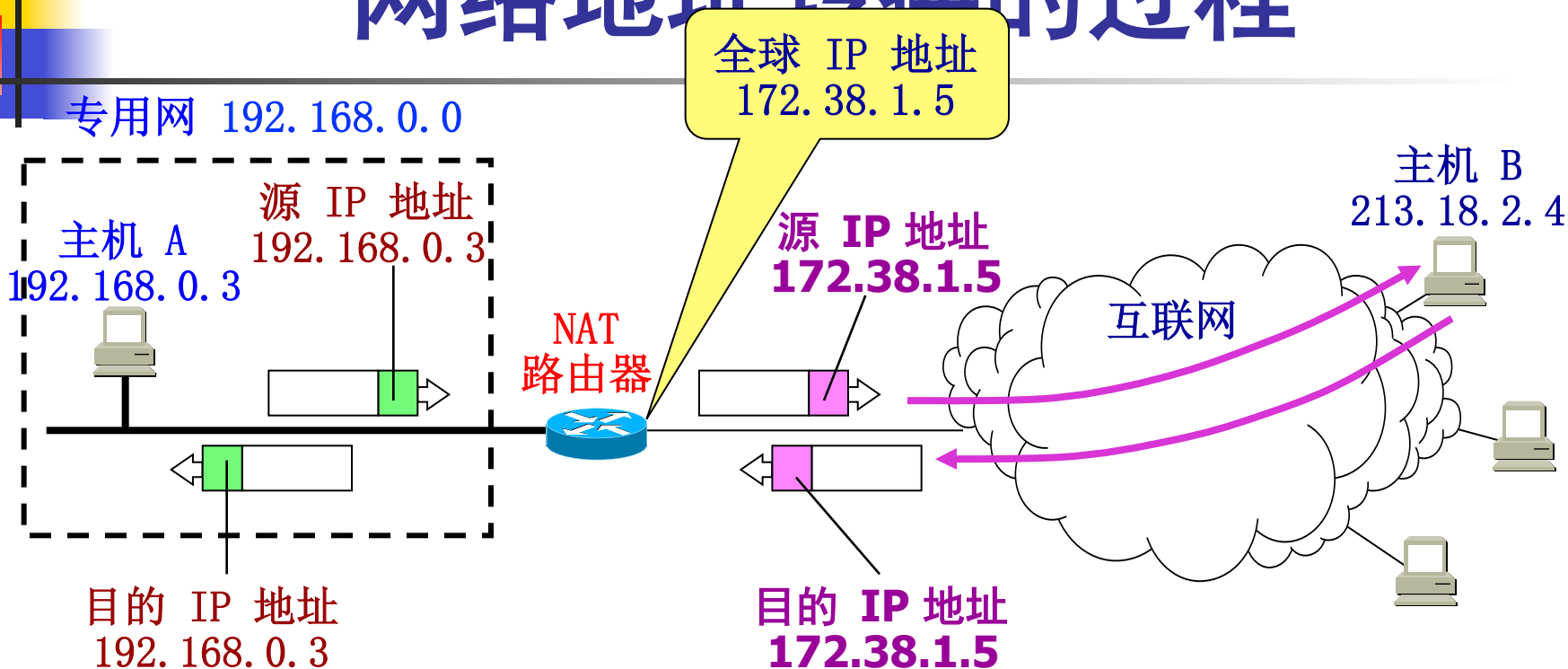


## 4.8.2 网络地址转换 NAT

- **问题：**在专用网上使用专用地址的主机如何与互联网上的主机通信（并不需要加密）？
- **解决：**
  - (1) 再申请一些全球 **IP** 地址。但这在很多情况下是不容易做到的。
  - (2) 采用网络地址转换 **NAT**。这是目前使用得最多的方法。



# 网络地址转换的过程



NAT 路由器的工作原理





# NAT 地址转换表

## NAT 地址转换表举例

方向	字段	旧的IP地址和端口号	新的IP地址和端口号
出	源IP地址:TCP源端口	192.168.0.3:30000	172.38.1.5:40001
出	源IP地址:TCP源端口	192.168.0.4:30000	172.38.1.5:40002
入	目的IP地址:TCP目的端口	172.38.1.5:40001	192.168.0.3:30000
入	目的IP地址:TCP目的端口	172.38.1.5:40002	192.168.0.4:30000

**NAPT**把专用网内不同的源 **IP** 地址，都转换为同样的全球 **IP** 地址。但对源主机所采用的 **TCP** 端口号（不管相同或不同），则转换为不同的新的端口号。因此，当 **NAPT** 路由器收到从互联网发来的应答时，就可以从 **IP** 数据报的数据部分找出运输层的端口号，然后根据不同的目的端口号，从 **NAPT** 转换表中找到正确的目的主机。



## 4.9 多协议标记交换 MPLS

---

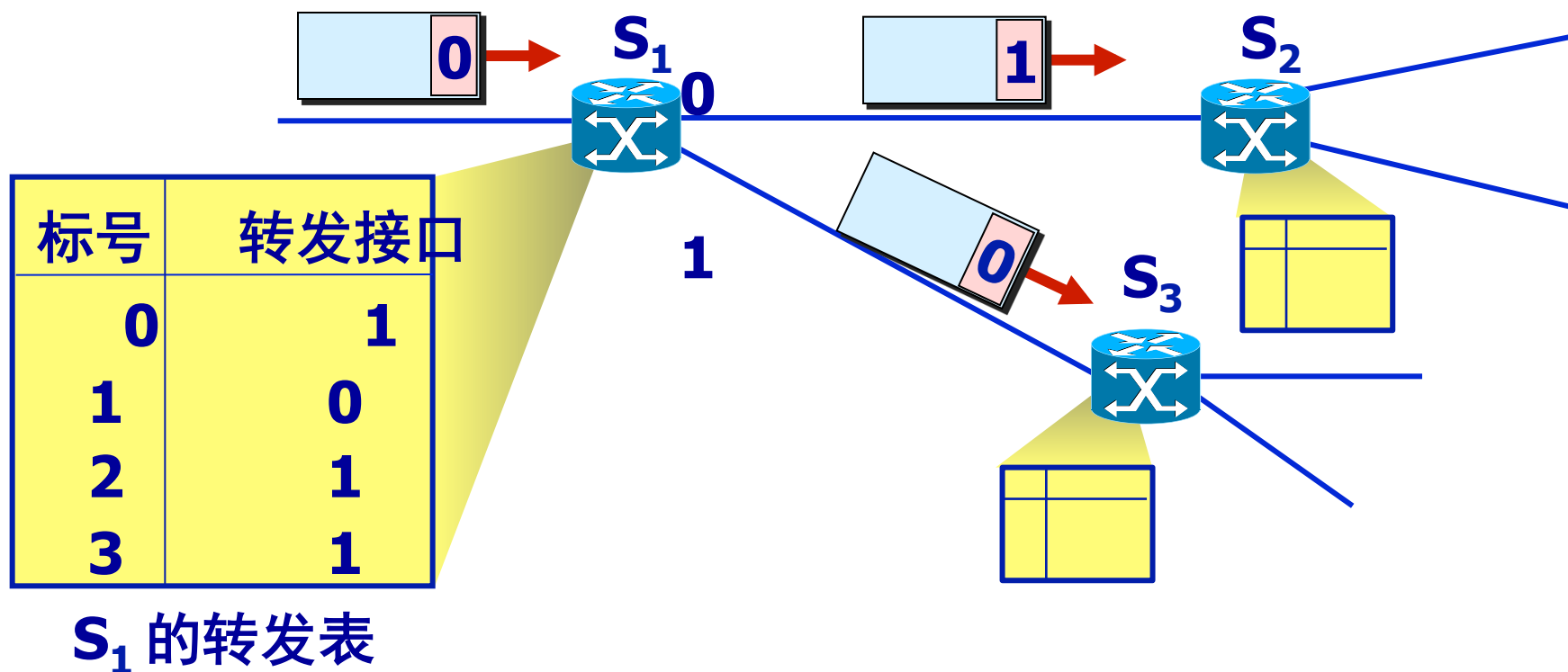
- 4.9.1 MPLS 的工作原理
- 4.9.2 MPLS 首部的位置与格式



## 4.9 多协议标记交换 MPLS

- IETF于1997年成立了 MPLS 工作组，开发出一种新的协议——多协议标记交换 MPLS (MultiProtocol Label Switching)。
- “多协议”表示在 MPLS 的上层可以采用多种协议，例如：IP，IPX；可以使用多种数据链路层协议，例如：PPP，以太网，ATM 等。
- “标记”是指每个分组被打上一个标记，根据该标记对分组进行转发。

为了实现交换，可以利用面向连接的概念，  
使每个分组携带一个叫做**标记 (label)** 的小整数。  
当分组到达交换机（即**标记交换路由器**）时，  
交换机读取分组的标记，  
并用标记值来检索分组转发表。  
这样就比查找路由表来转发分组要快得多。





# MPLS 特点

- MPLS 并没有取代 IP，而是作为一种 IP 增强技术，被广泛地应用在互联网中。
- MPLS 具有以下三个方面的特点：
  - (1) 支持面向连接的服务质量；
  - (2) 支持流量工程，平衡网络负载；
  - (3) 有效地支持虚拟专用网 VPN。



## 4.9.1 MPLS 的工作原理

### 1. 基本工作过程

#### ■ IP 分组的转发

- 在传统的 IP 网络中，分组每到达一个路由器后，都必须提取出其目的地址，按目的地址查找路由表，并按照“最长前缀匹配”的原则找到下一跳的 IP 地址（请注意，前缀的长度是不确定的）。
- 当网络很大时，查找含有大量项目的路由表要花费很多的时间。
- 在出现突发性的通信量时，往往还会使缓存溢出，这就会引起分组丢失、传输时延增大和服务质量下降。



# MPLS 协议的基本原理

- 在 **MPLS 域**的**入口**处，给每一个 **IP** 数据报打上**固定长度“标记”**，然后对打上标记的 **IP** 数据报用**硬件进行转发**。
- 采用硬件技术对打上标记的 **IP** 数据报进行转发就称为**标记交换**。
- “交换”也表示在转发时不再上升到第三层查找转发表，而是**根据标记在第二层（链路层）用硬件进行转发**。



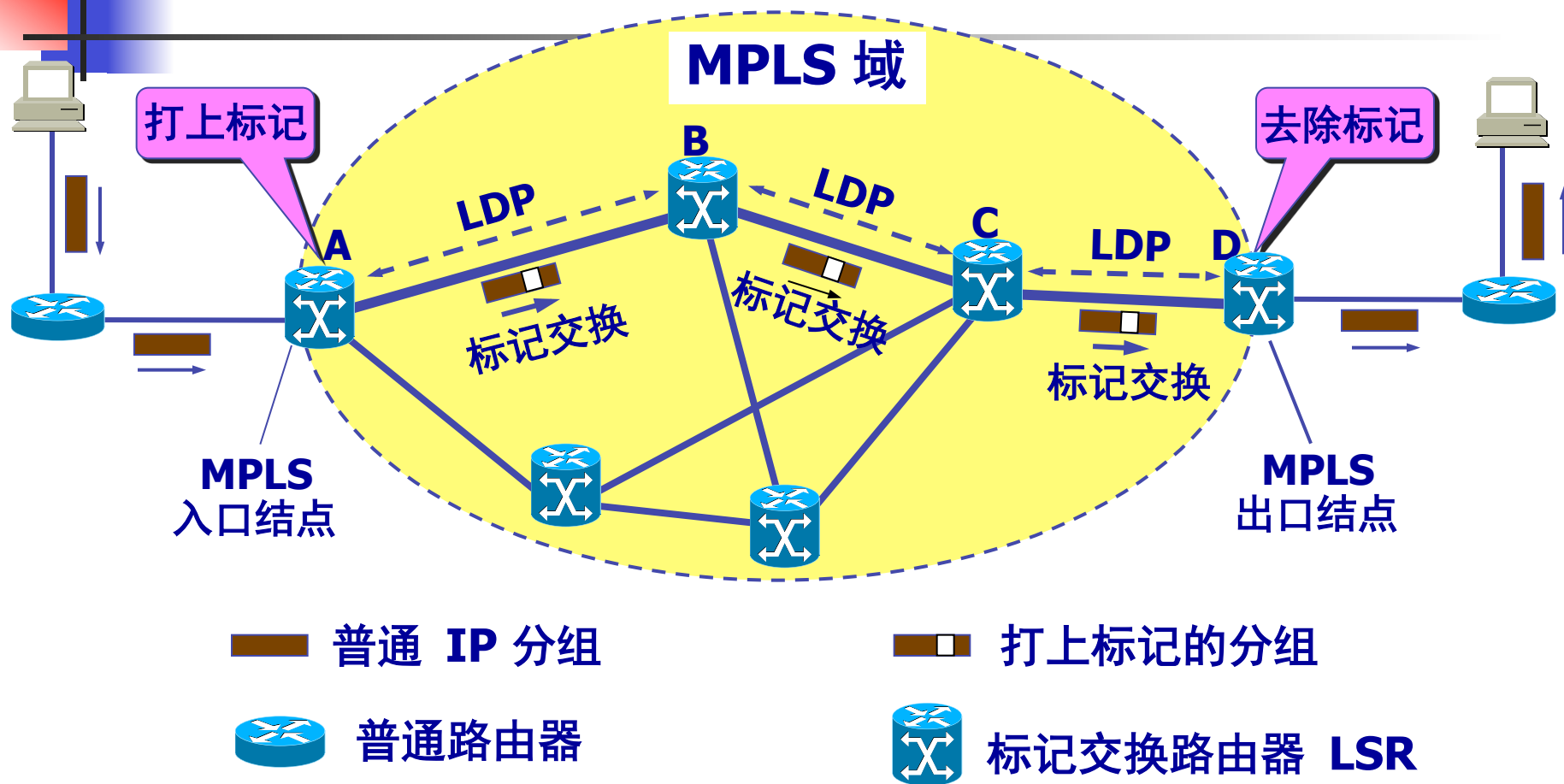
# MPLS 协议的基本原理

- MPLS 域 (MPLS domain) 是指该域中有许多彼此相邻的路由器，并且所有的路由器都是支持 MPLS 技术的**标记交换路由器 LSR (Label Switching Router)**。
- LSR 同时具有标记交换和路由选择这两种功能，**标记交换功能是为了快速转发**，但在这之前**LSR** 需要使用路由选择功能构造转发表。





# MPLS 协议的基本原理



MPLS 协议的基本原理



# MPLS 的基本工作过程

- (1) MPLS 域中的各 LSR 使用专门的**标记分配协议 LDP** 交换报文，并找出**标记交换路径 LSP**。各 LSR 根据这些路径构造出**分组转发表**。
- (2) 分组进入到 MPLS 域时，**MPLS 入口结点把分组打上标记**，并按照转发表将分组转发给下一个 LSR。给 IP 数据报打标记的过程叫做**分类 (classification)**。



# MPLS 的基本工作过程

- (3) 一个标记仅仅在两个标记交换路由器 LSR 之间才有意义。分组每经过一个 LSR，LSR 就要做两件事：一是转发，二是更换新的标记，即把入标记更换成为出标记。这就叫做标记对换 (label swapping)。

转发表

入接口	入标记	出接口	出标记
0	3	1	1

项目含义：从入接口 0 收到一个入标记为 3 的 IP 数据报，转发时，应当把该 IP 数据报从出接口 1 转发出去，同时把标记对换为 1。



## MPLS 的基本工作过程

(4) 当分组离开 MPLS 域时，MPLS 出口结点把分组的标记去除。再以后就按照一般分组的转发方法进行转发。

上述的这种“由入口 LSR 确定进入 MPLS 域以后的转发路径”称为显式路由选择 (explicit routing)，它和互联网中通常使用的“每一个路由器逐跳进行路由选择”有着很大的区别。



## 2. 转发等价类 FEC

- MPLS 有个很重要的概念就是转发等价类 FEC (Forwarding Equivalence Class)。
- “转发等价类”就是路由器按照同样方式对待的分组的集合。

**“按照同样方式对待”**表示：从同样接口转发到同样的下一跳地址，并且具有同样服务类别和同样丢弃优先级等。

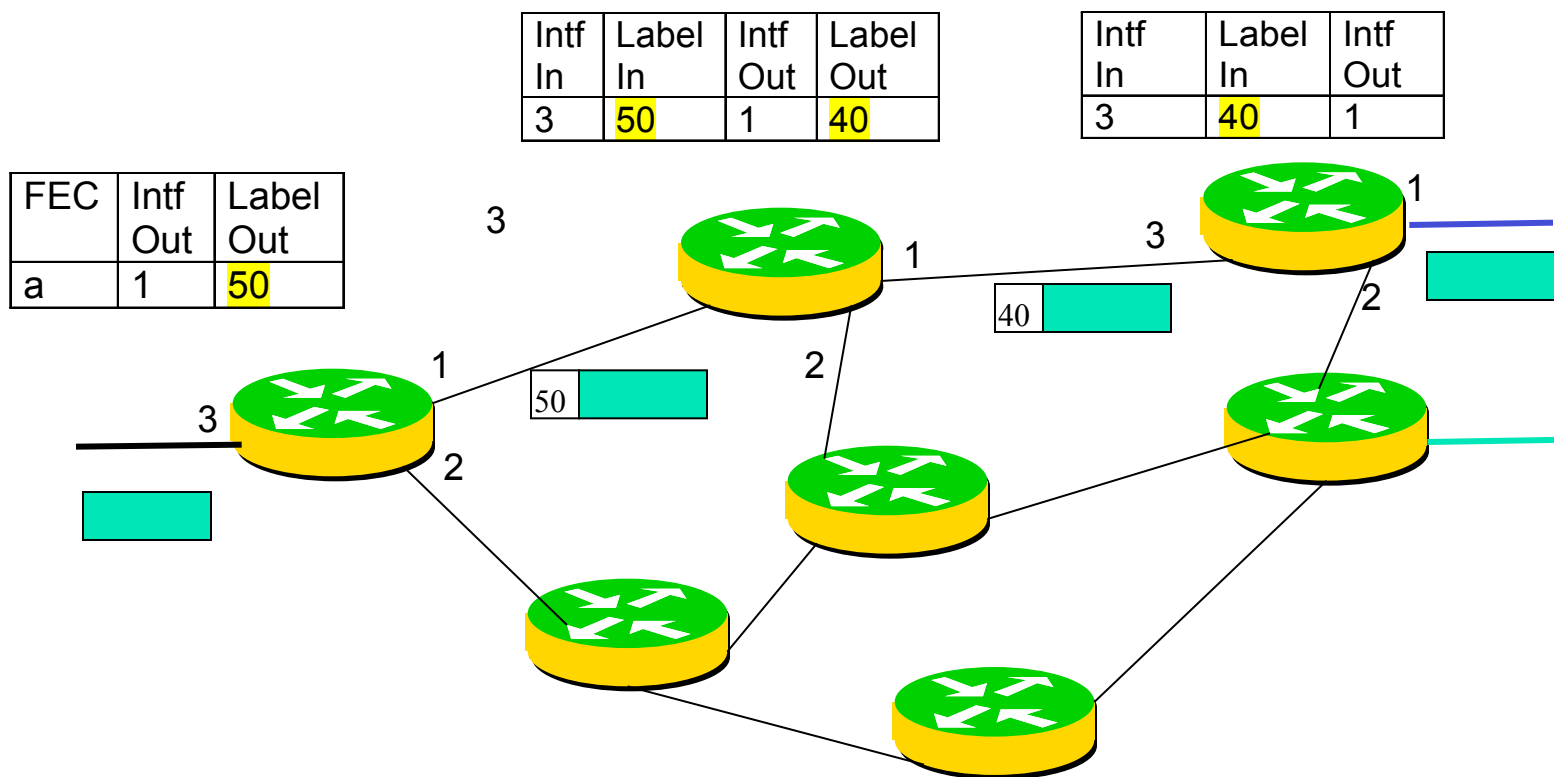


## 2. 转发等价类 FEC

- 划分 **FEC** 的方法不受什么限制，这都由网络管理员来控制，因此非常灵活。
  - 源/目的地址；源/目的端口
  - Diffserv
- 入口结点并不是给每一个分组指派一个不同的标记，而是将属于同样 **FEC** 的分组都指派同样的标记。
- **FEC** 和标记是一一对应的关系。



# MPLS Operation





# Label分配

- 每个LSR（有信令完成，如RSVP-TE）
  - 根据FEC分配一个Label
  - 通知上游节点
  - 同理，下游节点会将其分配的Label通知本节点



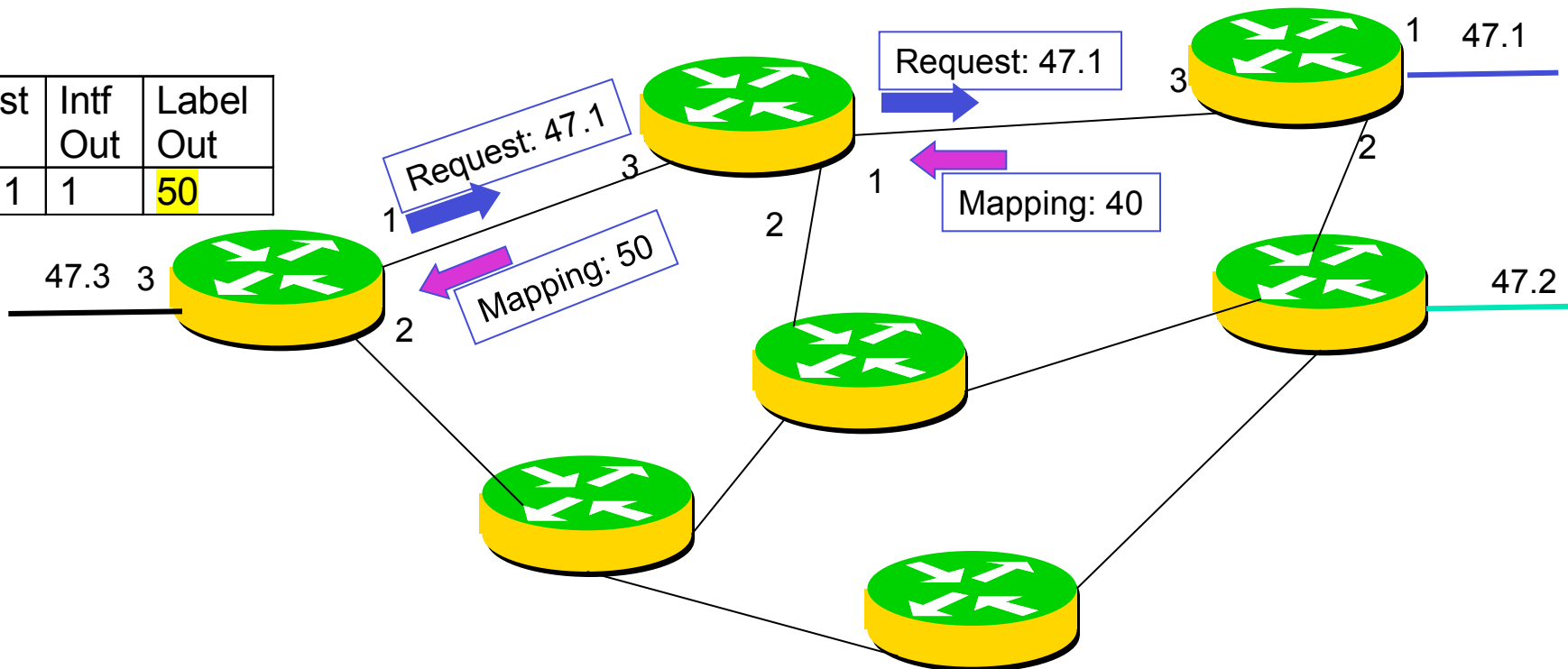


# Label 分配

Intf In	Label In	Intf Out	Label Out
3	50	1	40

Intf In	Label In	Intf Out
3	40	1

Dest	Intf Out	Label Out
47.1	1	50



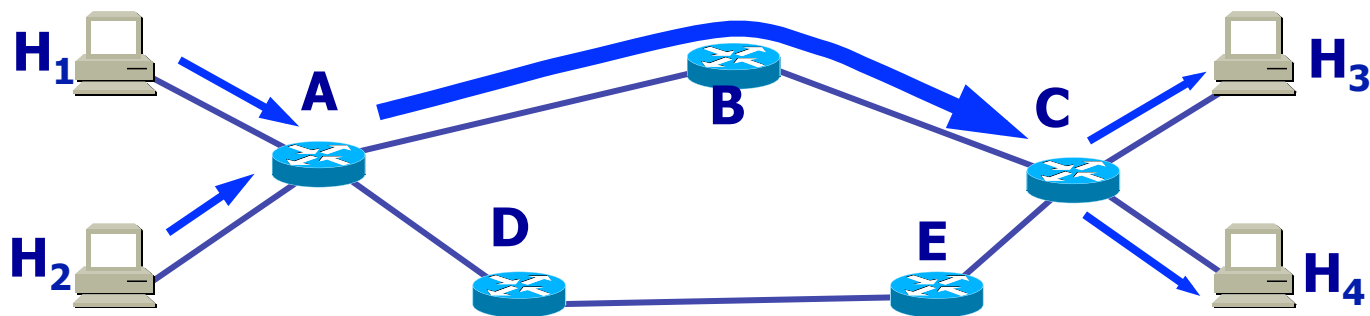


# LSP路径选择

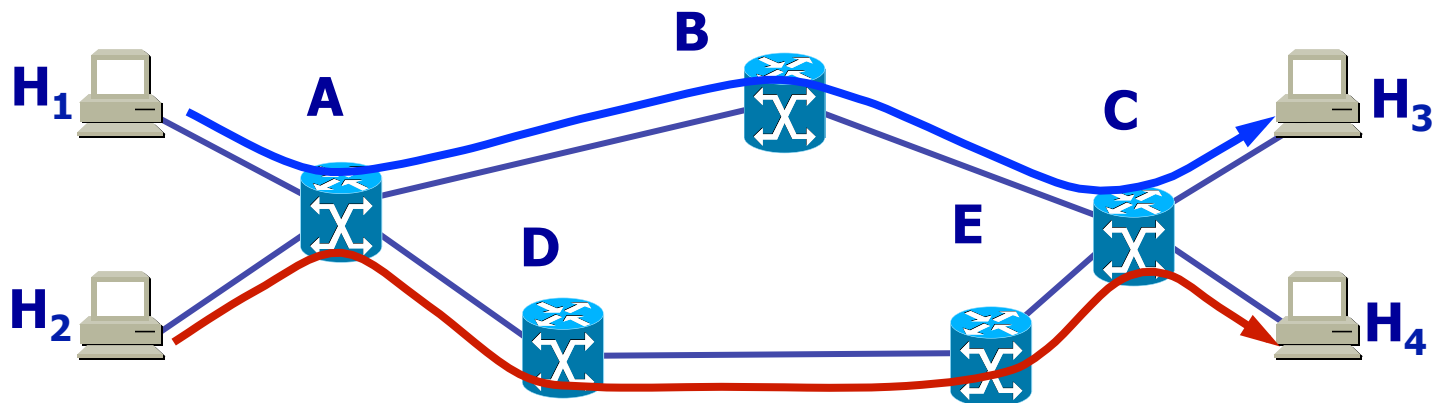
- 通过动态路由来建立LSP
  - 根据路由来转发RSVP-TE
- 显示路由（**Explicit routing**）
  - 有入口LSR决定一条LSP
  - 并通过源路由来转发RSVP-TE



# FEC 用于负载均衡



(a) 传统路由选择协议使最短路径  $A \rightarrow B \rightarrow C$  过载

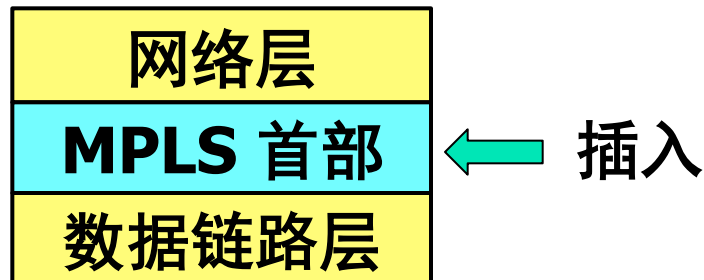


(b) 设置两种 FEC, 利用 FEC 使通信量分散



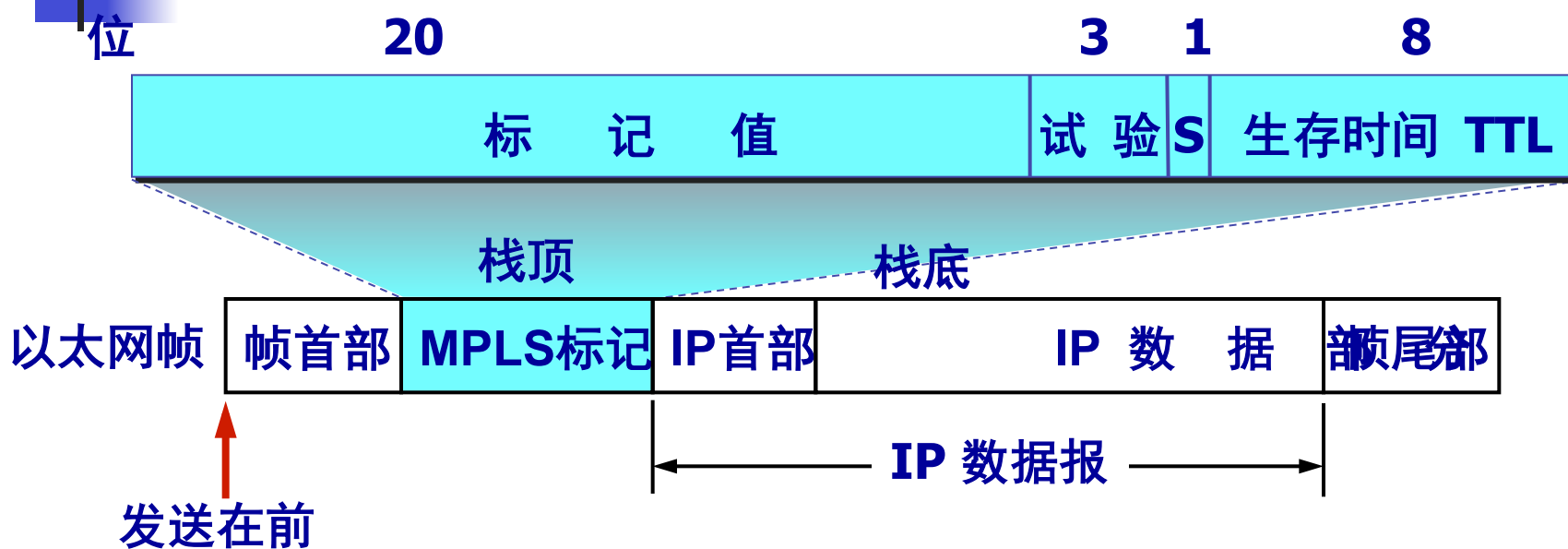
## 4.9.2 MPLS 首部的位置与格式

- MPLS 并不要求下层的网络都使用面向连接的技术。
- 下层的网络并不提供打标记的手段，而 IPv4 数据报首部也没有多余的位置存放 MPLS 标记。
- 这就需要使用一种封装技术：在把 IP 数据报封装成以太网帧之前，先要插入一个 MPLS 首部。
- 从层次的角度看，MPLS 首部就处在第二层和第三层之间。





# MPLS 首部的格式



“给 IP 数据报打上标记” 其实就是在以太网的帧首部和 IP 数据报的首部之间插入一个 4 字节的 **MPLS 首部**。



# MPLS 首部的格式

- MPLS 首部共包括以下四个字段：
  - (1) 标记值（占 20 位）。可以同时容纳高达  $2^{20}$  个流（即 1048576 个流）。实际上几乎没有哪个 MPLS 实例会使用很大数目的流，因为通常需要管理员人工管理和设置每条交换路径。
  - (2) 试验（占 3 位）。目前保留用作试验。
  - (3) 栈S（占 1 位）。在有“标记栈”时使用。
  - (4) 生存时间TTL（占 8 位）。用来防止 MPLS 分组在 MPLS 域中兜圈子。



# 作业

- P195
- 4-02, 4-03, 4-07, 4-09, 4-16  
    , 4-17, 4-18, 4-20, 4-21, 4-26  
    , 4-29, 4-30, 4-31, 4-40, 4-41  
    , 4-44, 4-45, 4-47, 4-54, 4-56  
    , 4-67