计算机网络

第4章 网络层

目　　录

- 网络层概述
- 虚电路和数据报网络
- 路由器的工作原理（mooc）
- 网际协议:因特网中的转发和编址
- 选路算法
- IP组播介绍

4.1 网络层概述

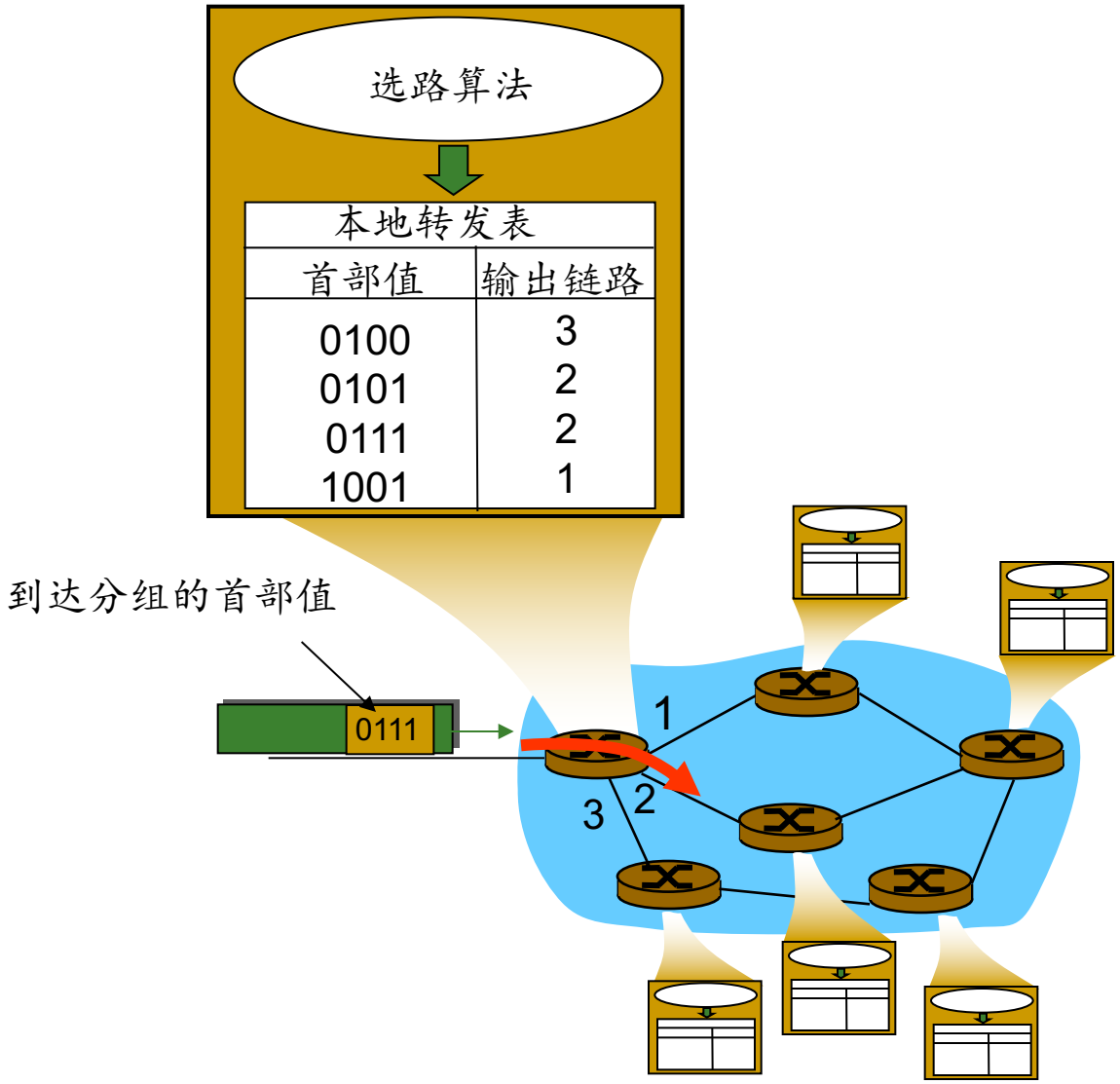
- **网络层的目标**
  - 实现主机到主机的通信
- **网络层在计算机网络中的地位**
  - 为运输层提供支持
    - 运输层实现进程到进程的通信
    - 运输层功能的实现依赖于网络层提供的服务
  - 为实现从源主机到目标主机成功的移动数据分组，整个路径上的每一台分组交换机上均需实现网络层

网络层属于网络核心的功能

从现在开始，讨论从网络边缘进入到了网络核心

4.1 网络层概述

- **网络层的主要功能**
  - 在全局范畴为主机之间的通信进行选路，选路的结果反映为分组交换机上的转发表
  - 分组交换机上的网络层根据转发表以及分组头部信息，将分组向适当链路进行转发
  - 对于面向连接的网络层服务，提供连接建立的功能
    - ATM、X.25和帧中继


选路算法
本地转发表
首部值
输出链路
0100
0101
0111
1001
3
2
2
1
到达分组的首部值
0111
1
2
3

4.1 网络层概述

- **分组交换机的分类**
  - 根据链路层首部信息进行转发的——链路层节点交换机
  - 根据网络层首部信息进行转发的——路由器

4.1 网络层概述

- **网络层可能提供的服务**
  - 确保交付
  - 具有时延上界的确保交付
  - 有序分组交付
  - 确保最小带宽
  - 确保最大时延抖动

  ……

4.1 网络层概述

- **几种实际使用的网络层服务模型**

网络体系结构	服务模型	带宽保证	无丢失保证	排序	定时	拥塞指示
因特网	尽力而为	无	无	无	不维持	无
ATM	CBR	保证恒定速率	是	有序	维持	无拥塞
ATM	ABR	保证最小速率	无	有序	维持	提供指示

4.2 虚电路和数据报网络

- **网络层提供的服务**
  - 面向连接的服务——虚电路，需事先握手
  - 面向无连接的服务——数据报，无需握手
- **网络层与运输层相应服务的区别**
  - 网络层是向运输层提供主机到主机的服务，而运输层是向应用层提供进程到进程的服务
  - 网络层仅提供上述两种服务中的一种，不同时提供两种，而运输层则同时提供两种
  - 运输层的服务在网络边缘的端系统中实现，而网络层的服务则在整个网络中实现，含路由器

4.2 虚电路和数据报网络

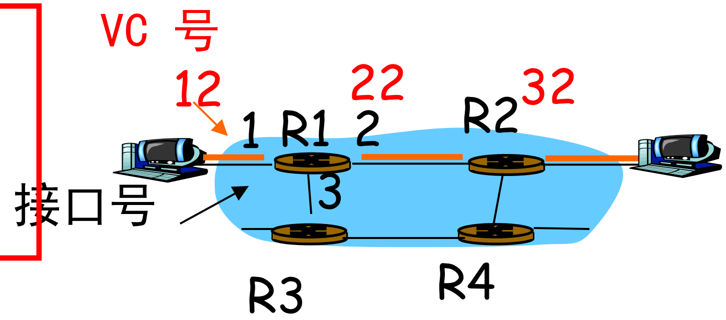
- **虚电路**
  - 目标
    - 使收发双方之间的路径表现得如同电话线路一般
  - 工作机制
    - 数据开始流动之前，呼叫建立；流动结束后要断开
    - 每一个分组携带虚电路的标识 (而不是目的主机的地址)
    - 路径上的每一个路由器必须为进行中的连接维持连接状态信息
      - 传输层的连接仅涉及到两个端系统（end system）
    - 链路, 路由器资源 (带宽、缓冲区) 可以分配给虚电路
      - 目的：为了达到类似线路交换的性能

4.2 虚电路和数据报网络

- 虚电路的组成
  - 从源到目的主机的路径
  - VC 号，沿着该路径的每段链路的一个号码
  - 沿着该路径的每台路由器中的转发表

4.2 虚电路和数据报网络

每当跨越一个路由器建立一个新的VC时，该路由器的转发表就建立一个新项；当该VC拆除时，就删除该路径上所有路由器的相应表项



路由器R1的转发表：

入接口	入 VC #	出接口	出 VC #
1	12	2	22
2	63	1	18
3	7	2	17
1	97	3	87
…	…	…	…

路由器维持连接状态信息！

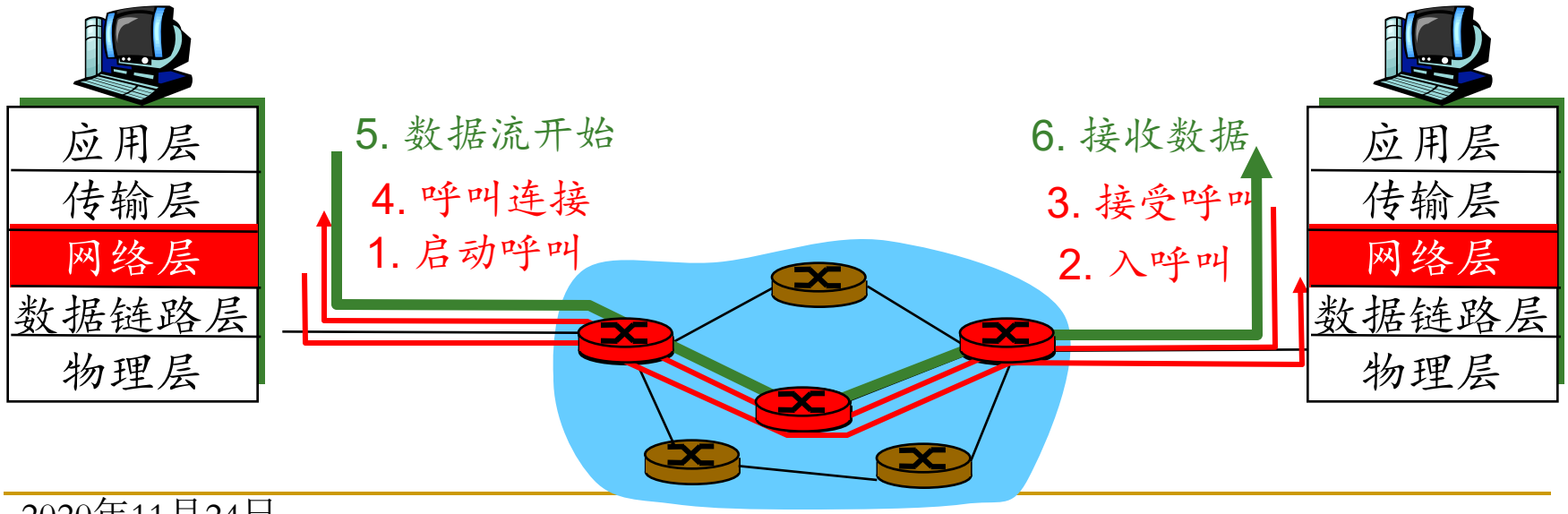
1: 减少VC字段长度！

2: 为 沿着虚电路的每条链路 虚电路号不同！

如果沿着某虚电路所有链路用共同的VC号，路由器不得不发送大量交换报文以确认一个全局VC号

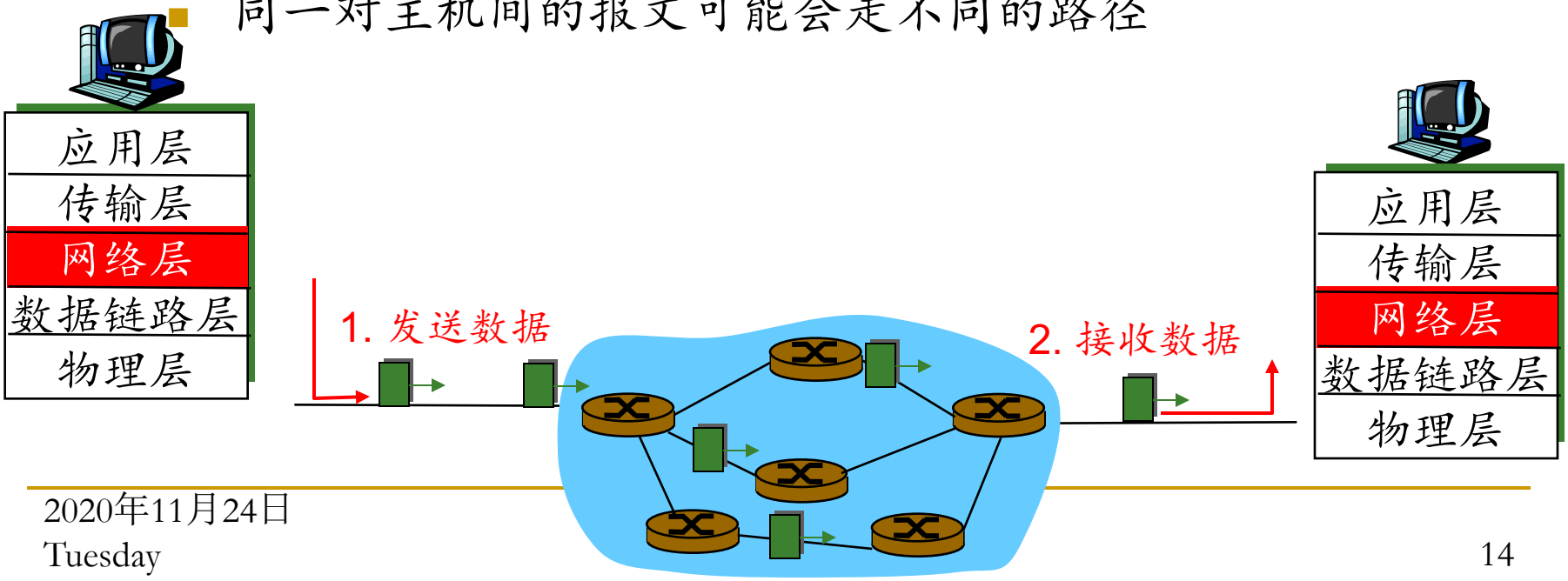
4.2 虚电路和数据报网络

- 信令协议
  - 用于建立、维护以及断开虚电路
  - 用于 ATM, 帧中继, X.25网络
  - 今天的因特网已经不再使用该协议



4.2 虚电路和数据报网络

- **数据报网络**
  - 在网络层没有连接建立过程
  - 路由器：在端到端的连接中不维护连接状态信息
    - 在网络层不存在“联接”的概念
  - 传输报文时使用目的主机地址信息
    - 同一对主机间的报文可能会走不同的路径



4.2 虚电路和数据报网络

可能有超过40亿个地址！如何减小转发表？：采用地址前缀比配法

目的地址范围	链路接口
11001000 00010111 00010000 00000000 到 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 到 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 到 11001000 00010111 00011111 11111111	2
其它	3

4.2 虚电路和数据报网络

- 最长前缀匹配
  - 通过比较地址前缀进行转发，可以大大减小转发表的大小

前缀匹配	接口
11001000 00010111 00010	0
11001000 00010111 00011000	1
11001000 00010111 00011	2
其他	3

例

DA: 11001000 00010111 00010110 10100001　　从哪个接口转发?

DA: 11001000 00010111 00011000 10101010　　从哪个接口转发?

最长前缀匹配原则：向与最长前缀匹配的链路接口转发

数据报还是虚电路: 为什么?

因特网：总的思想是把网络层设计的尽可能简单，把复杂的功能（如按序可靠交付，拥塞控制等放到端系统）。原因：

- 数据交换在计算机之间进行
  - “弹性”服务，没有严格的实时性要求
- “聪明”的端系统（计算机）
  - 可进行自适应，执行控制，出错恢复
  - 网络内部比较简单，“边缘上”比较复杂
- 许多种链路类型
  - 具有不同的特性
  - 统一服务标准十分困难

虚电路：

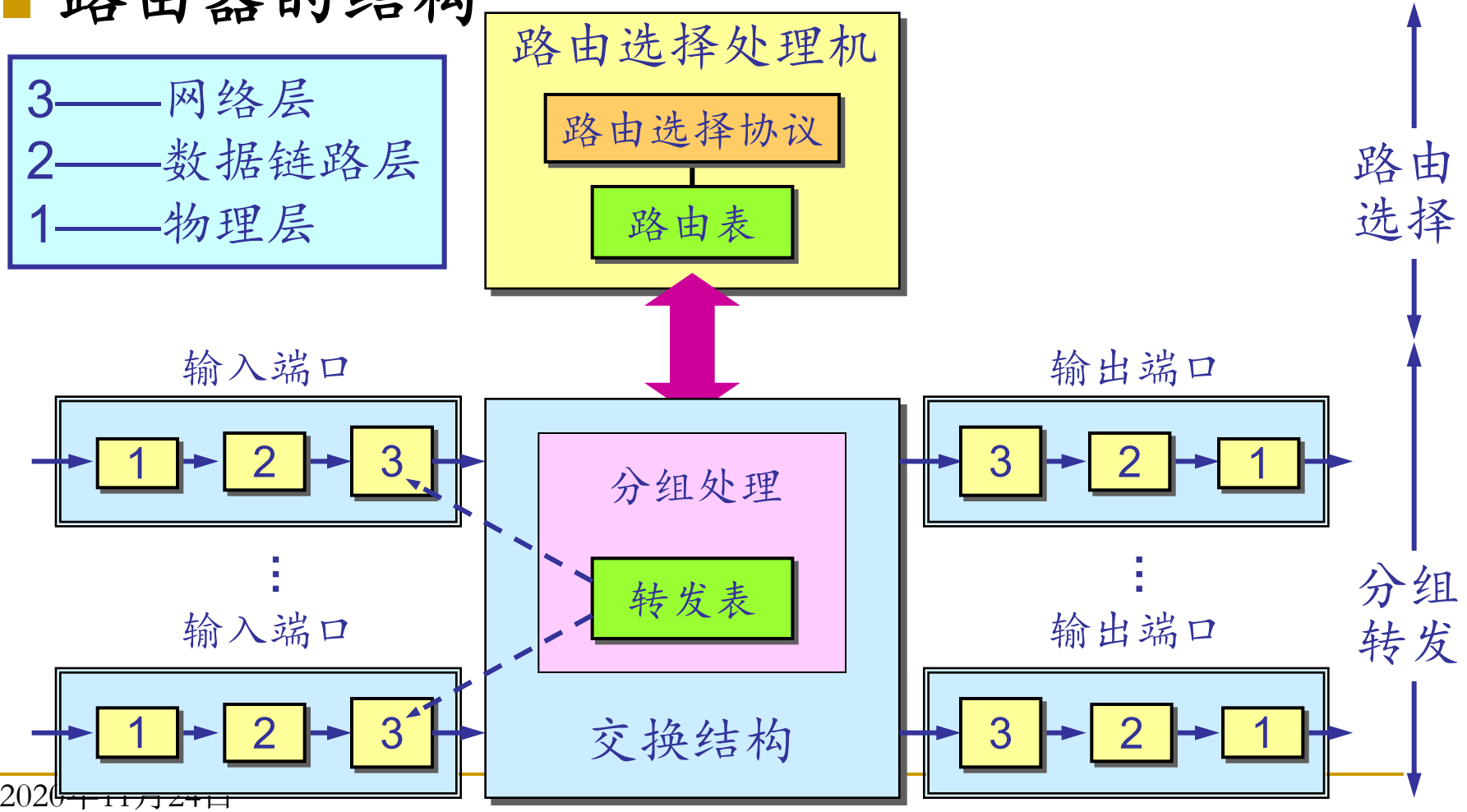
- 电话网络演化而来
- 人们的交流：
  - 严格要求实时性，和可靠
  - 需要服务承诺
- “傻瓜式”的端系统
  - 电话机
  - 复杂性在网络内部

4.2 虚电路和数据报网络

- **虚电路 vs 数据报**
  - 虚电路网络：聪明的网络，愚笨的终端
  - 数据报网络：简单的网络，复杂的终端
    - 互联不同类型的网络更加容易
    - 启用新服务的速度更快，更简单

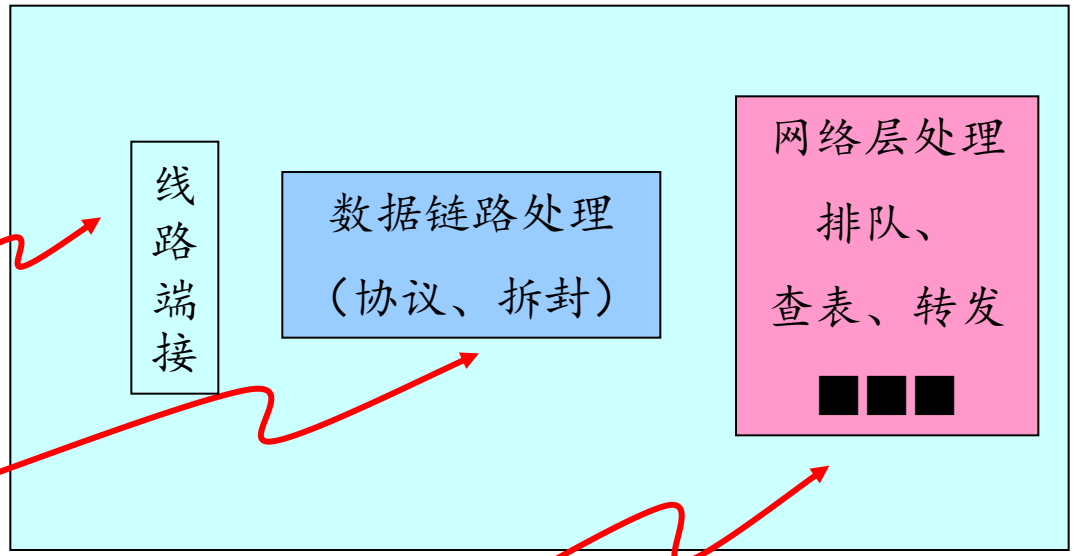
4.3 路由器的工作原理（MOOC）

- **路由器的结构**



4.3 路由器的工作原理

输入端口



物理层:
位流级的接收

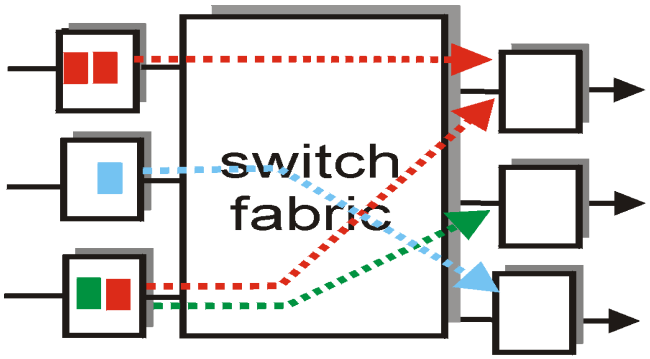
数据链路层:
e.g.,以太网

分散式交换:

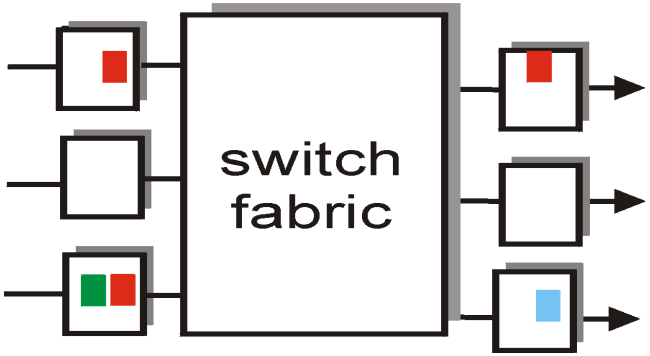
- 按照给出的目的地址,使用输入端口的内存中存储的路由选择表，查找输出端口（注意每个输入端口都有一份转发表的拷贝），
- 目标:以“线路速度”完成输入端口的处理
- 排队：如果数据报到达的速度超过了输入端口将数据报转交给交换结构的速度，则后到的分组会暂时阻塞

4.3 路由器的工作原理

- **输入端口排队**
  - 如果输入端口的处理速率超过了交换结构的速率，输入端口就可能产生排队
  - 线头阻塞:在输入队列中排队的分组必须等待通过交换结构发送，因为它被位于线头的另一个分组阻塞了。即使该分组要转发的输出端口是空闲的（图中绿色的分组）。
  - 输入缓冲区溢出可导致排队时延和丢包!

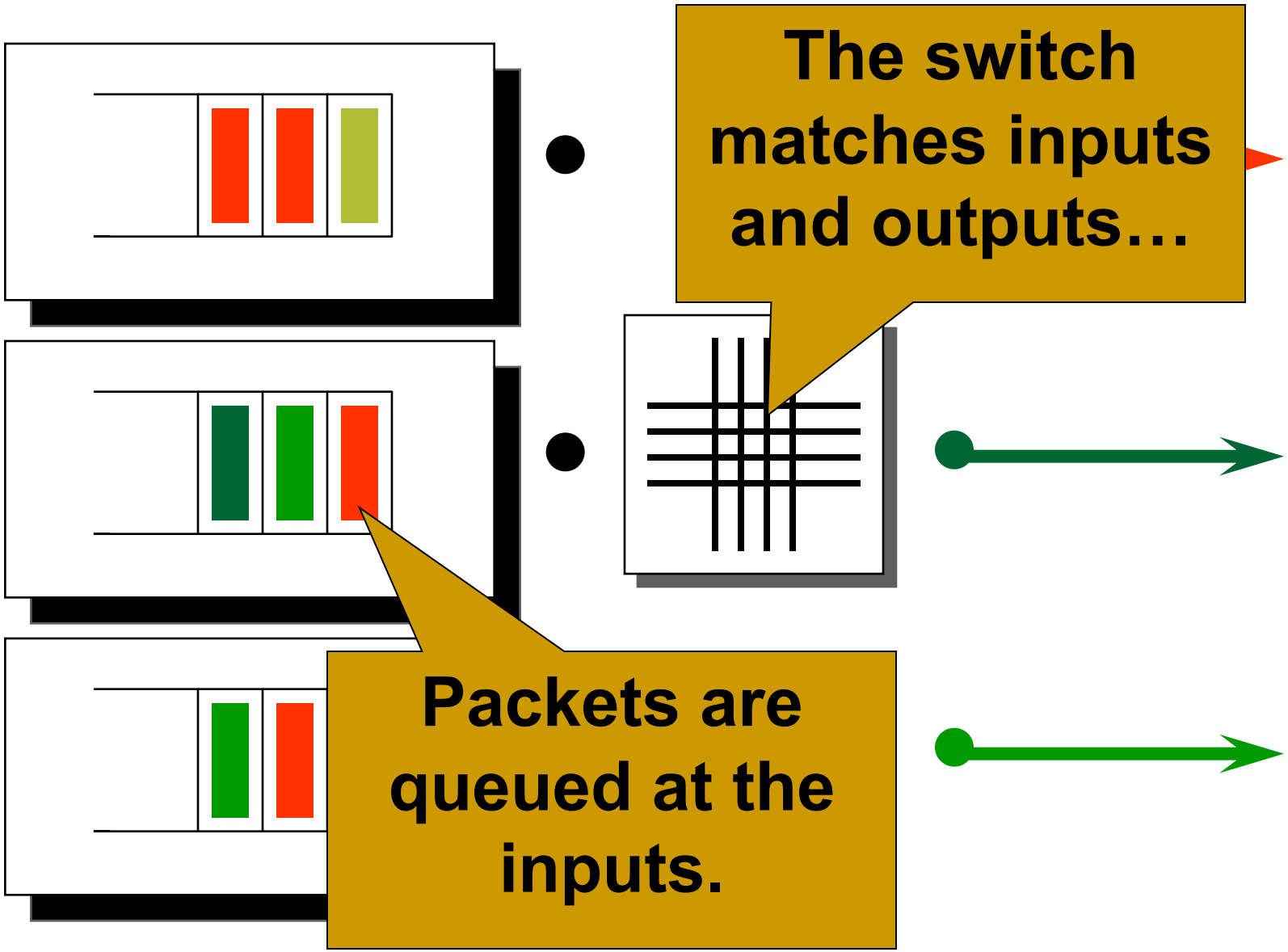


output port contention
at time t - only one red
packet can be transferred

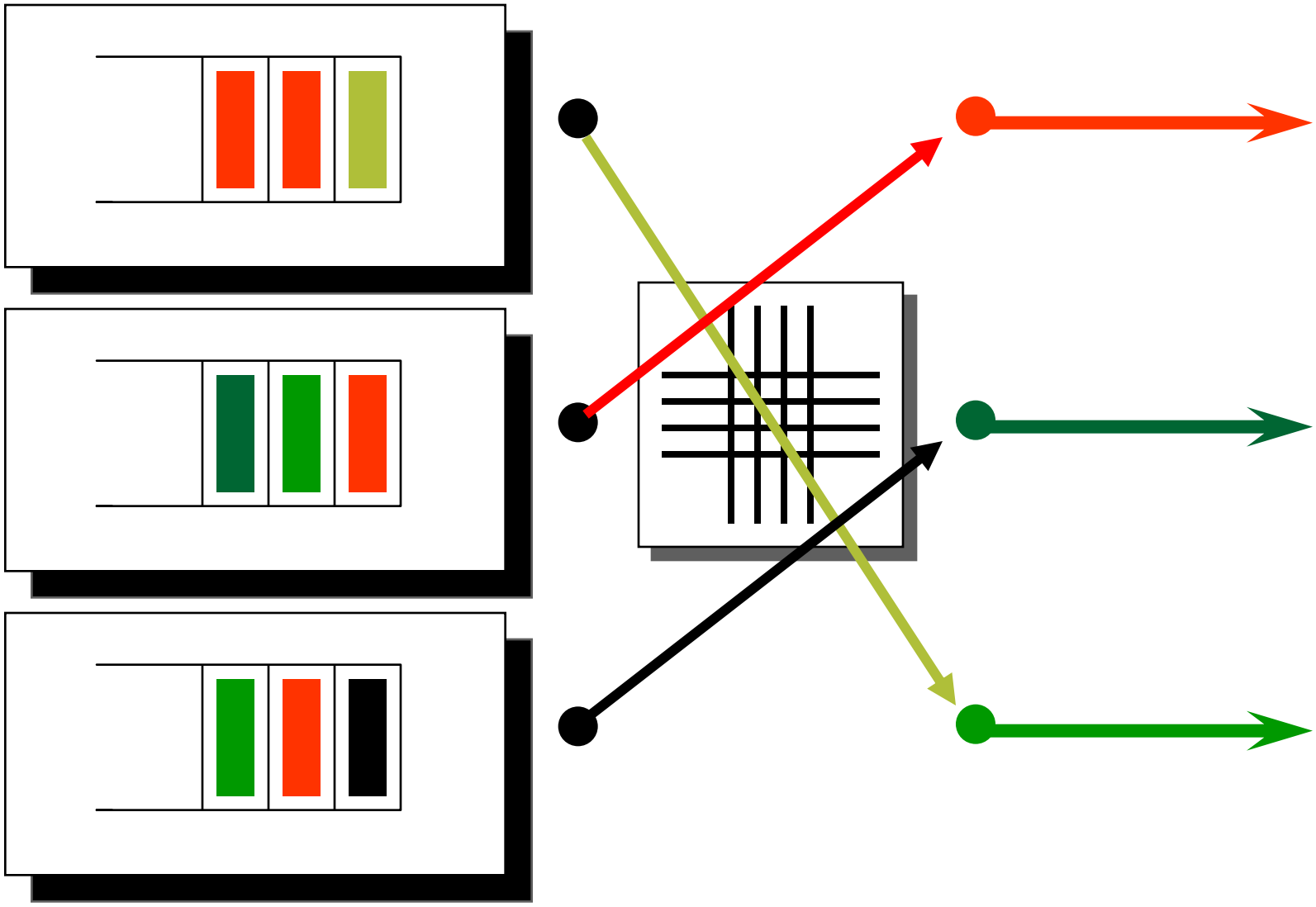


green packet
experiences HOL blocking

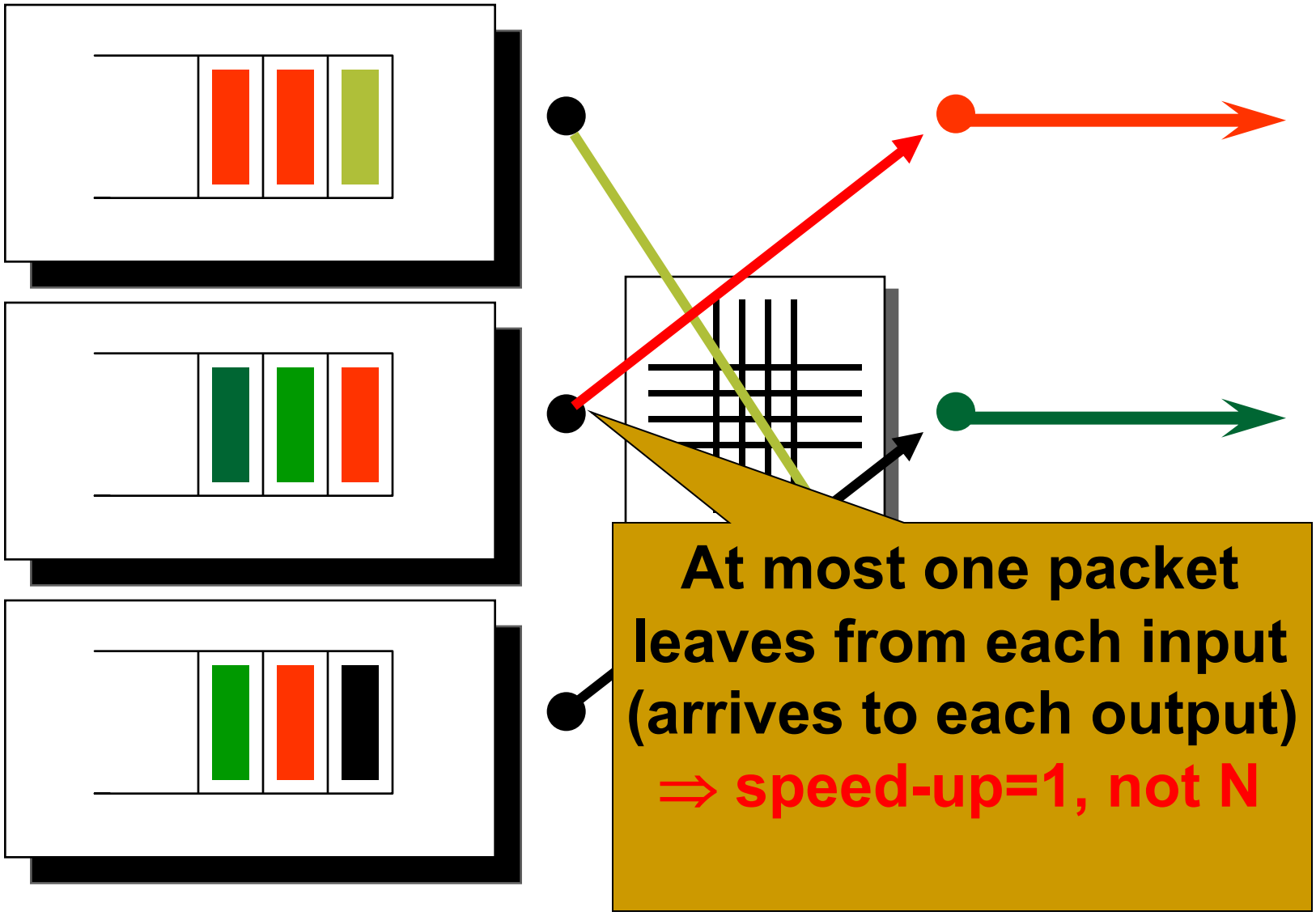
4.3 路由器的工作原理


The switch matches inputs and outputs…
Packets are queued at the inputs.

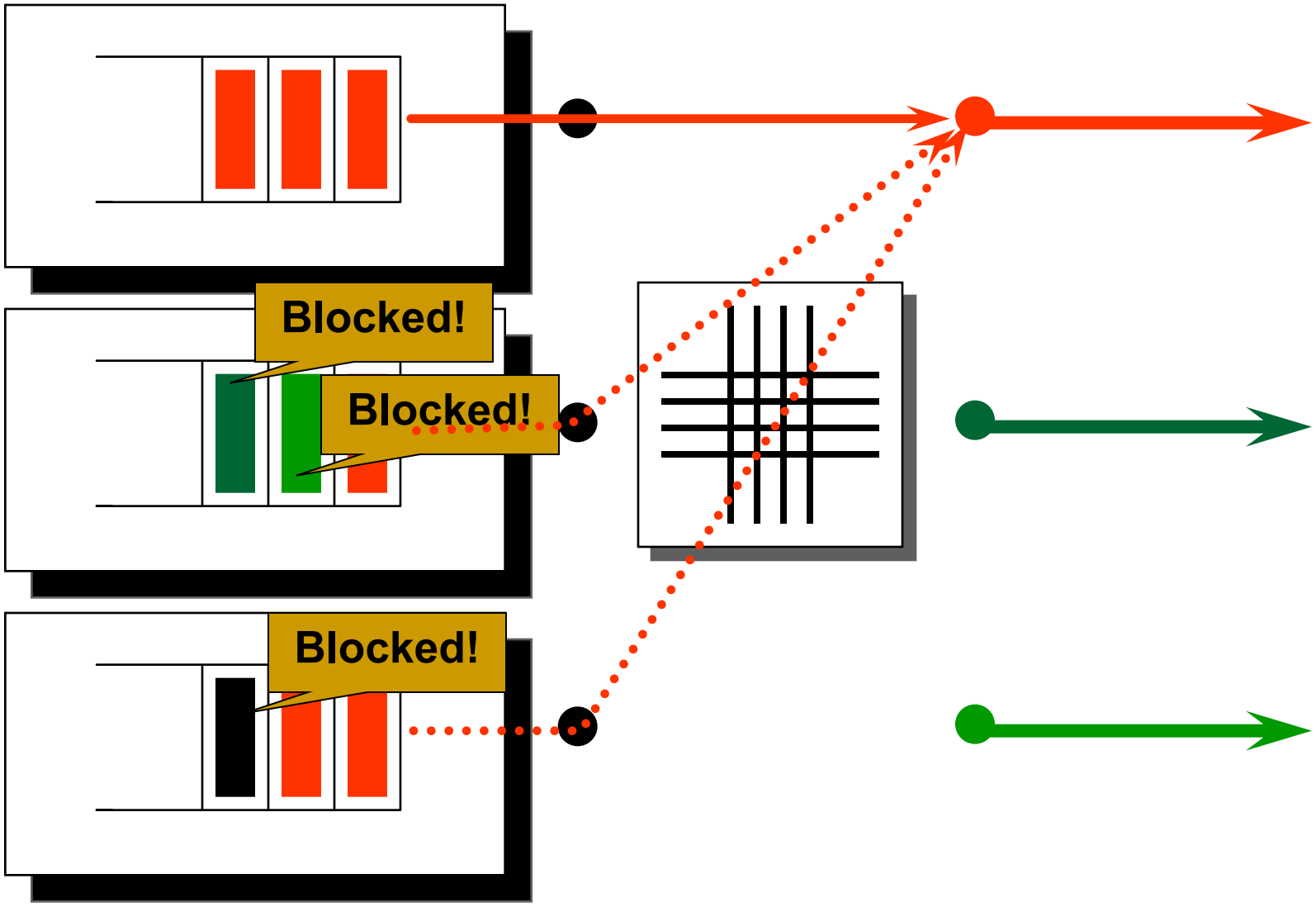
4.3 路由器的工作原理



4.3 路由器的工作原理

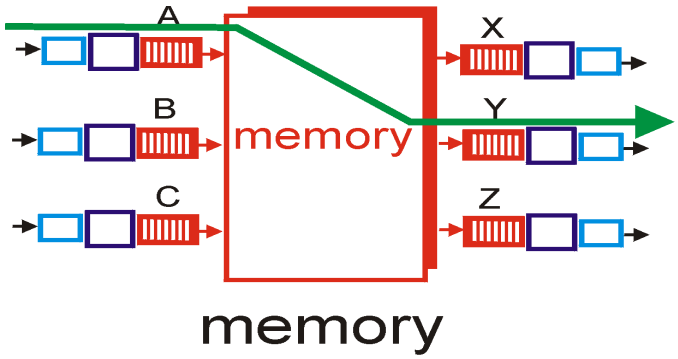

At most one packet
leaves from each input
(arrives to each output)
⇒ speed-up=1, not N

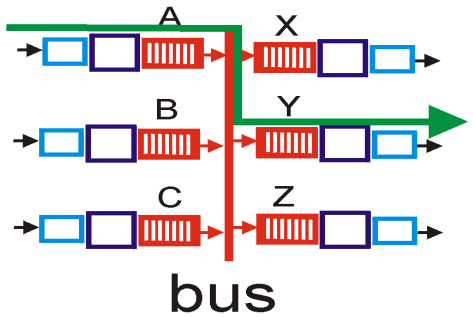
4.3 路由器的工作原理

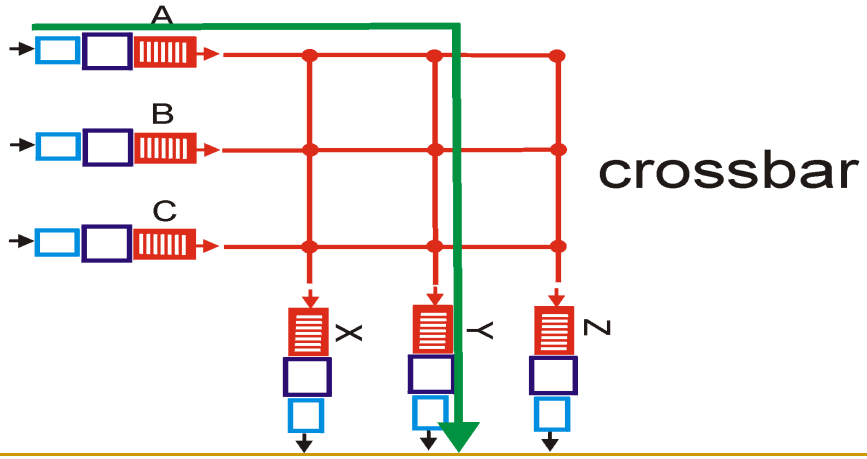


4.3 路由器的工作原理

- **交换结构**

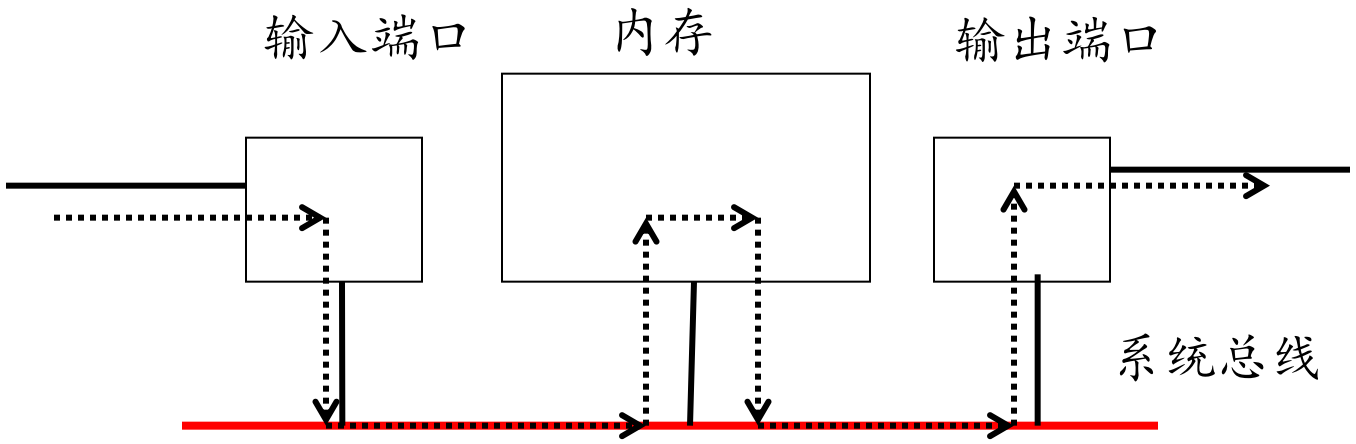






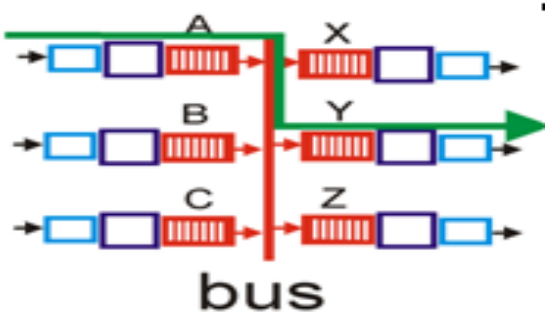
4.3 路由器的工作原理

- 经内存交换
  - 在输入端口和输出端口之间的交换是在CPU（路由处理器）的直接控制下完成的
  - 分组被拷贝到系统内存中，然后在CPU的控制下输送到输出端口
  - 转发速度受限于内存的带宽（每个分组走两次总线）（内存带宽/2）



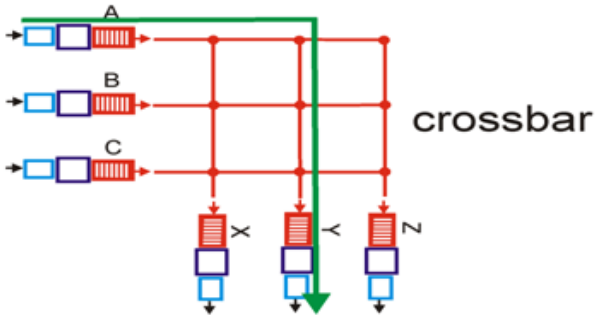
4.3 路由器的工作原理

- 经总线交换
  - 输入端口经一根共享总线将分组直接传送到输出端口
  - 总线交换的问题: 交换速度受限于总线的带宽
  - 现在，总线带宽超过1 Gbps
  - 对于运行在接入网或企业网的路由器，通过总线交换的转发速度是足够的。(区域网和主干网则不行)
  - Cisco 5600：通过一个32Gbps的背板总线来交换分组



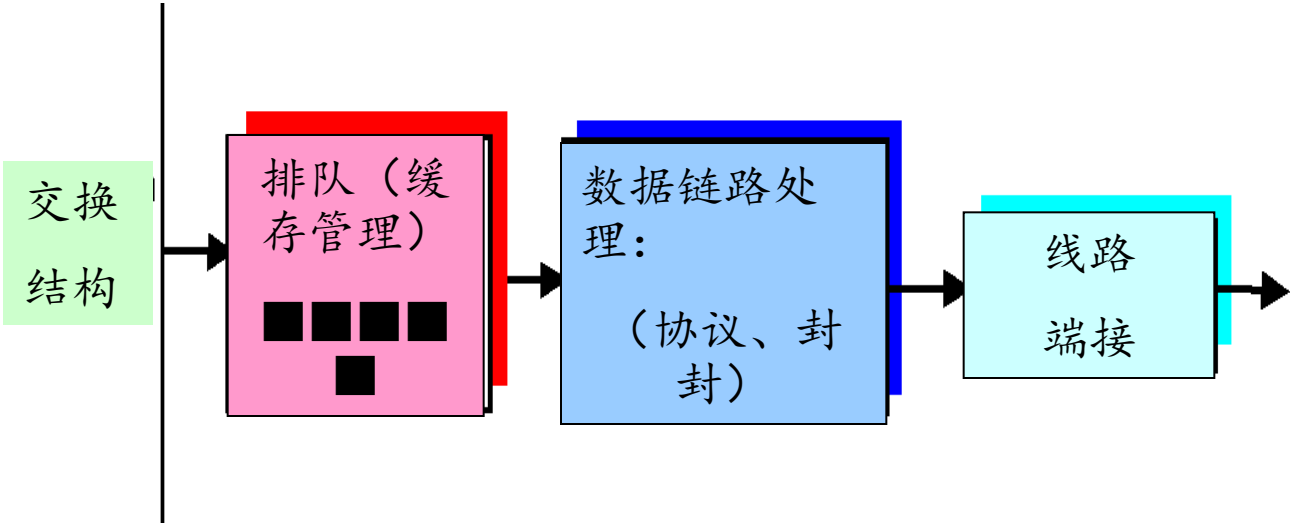
4.3 路由器的工作原理

- 经内联网络
  - 克服总线带宽限制
  - 由2n条总线组成，将n个输入端口和n个输出端口连接
  - 设计先进：将长度变化的IP分组分片成固定尺寸的信元，通过交换结构对信元进行转发
  - Cisco 12000: 通过内联网络交换速度为60Gb/s



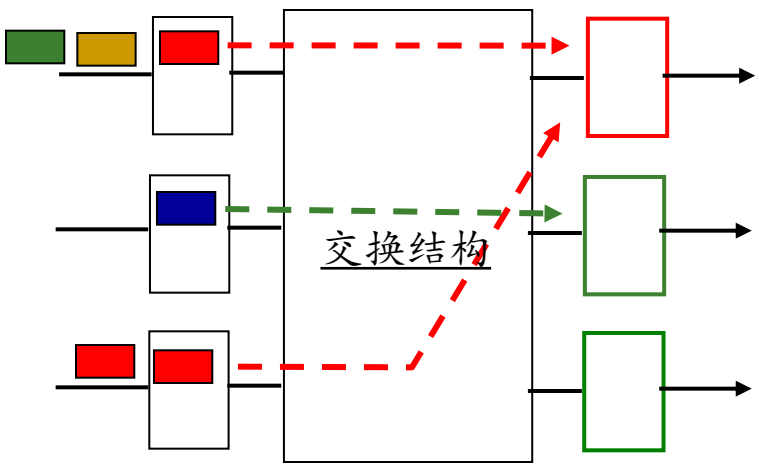
4.3 路由器的工作原理

- **输出端口**
  - 缓存管理：当交换结构将分组交付给输出端口的速率超过输出链路速率时
  - 调度原则：在数据报队列中选择数据报进行传输

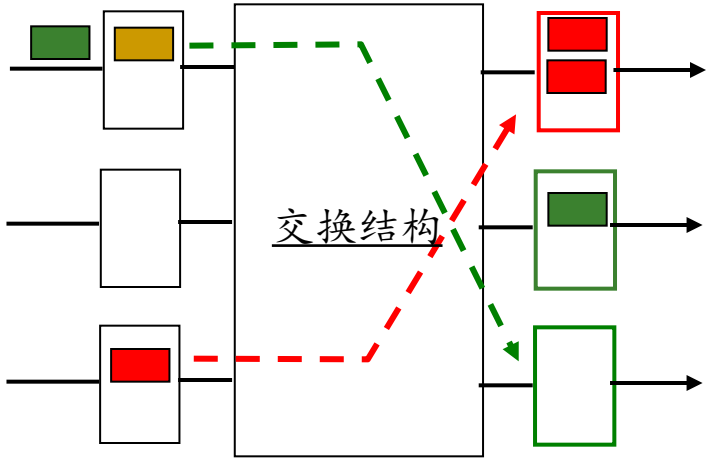


4.3 路由器的工作原理

- **输出端口排队**
  - 当通过交换结构到达的分组速率超过了输出链路的速率时，需要对分组进行缓存
  - 输出端口缓冲区溢出会导致分组的排队和丢失!



t_1时刻分组试图从输入端口去往输出端口



1个分组交换时间之后

4.3 路由器的工作原理

- 缓存区设置多少合适呢？
  - RFC 3439说：$B = RTT \times R$
    - RTT=250ms，R=10Gbps，则B=2.5G bits
  - 最近的理论和试验研究认为对于有N条TCP连接经过的链路而言：

$$B = \frac{RTT \times R}{\sqrt{N}}$$

4.3 路由器的工作原理

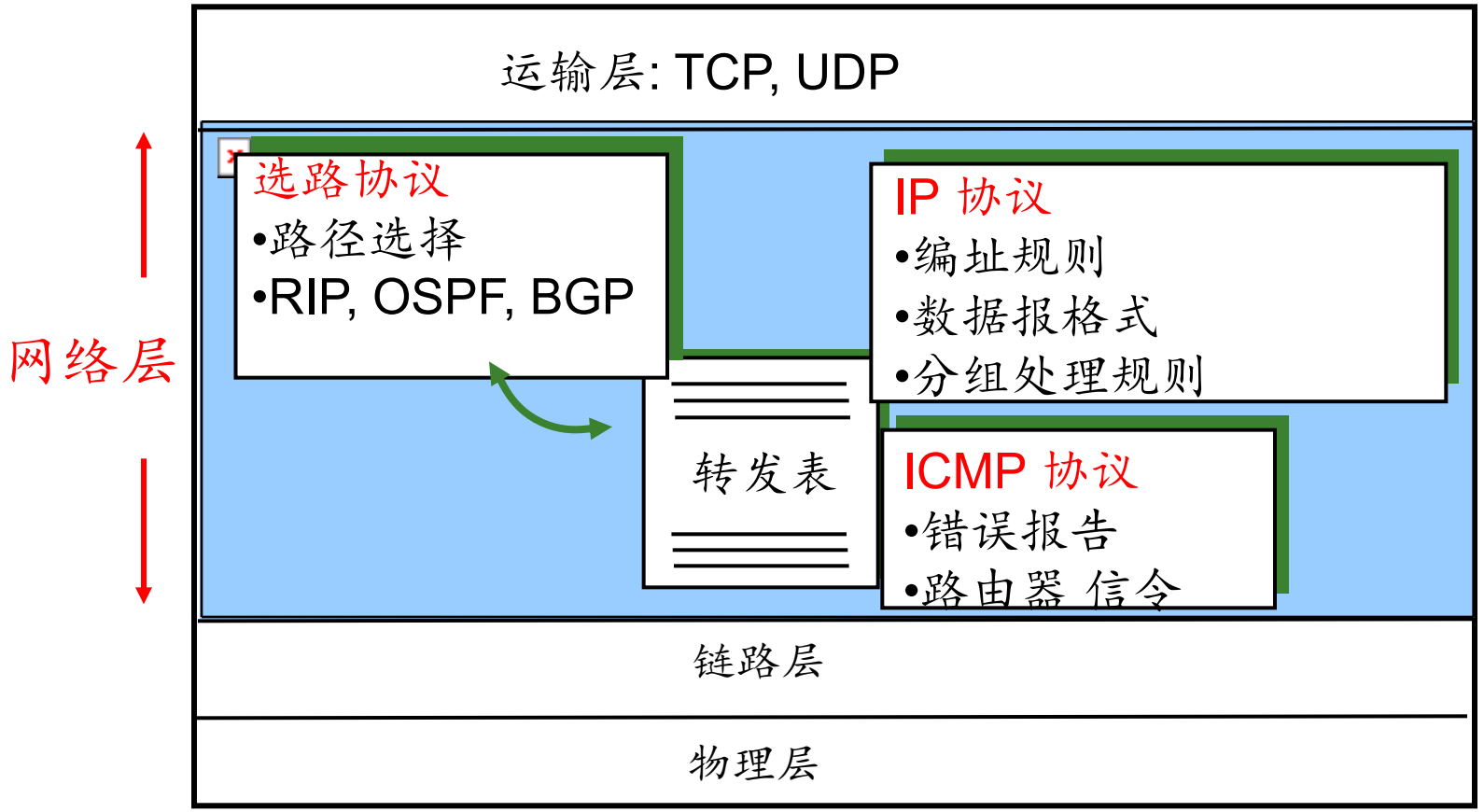
- 输出端口分组调度策略
  - 先来先服务FCFS
  - 加权公平排队WFQ
  - 分组调度程序在提供服务质量保证方面起着关键作用

4.3 路由器的工作原理

- 分组丢弃策略
  - 被动队列管理
    - 弃尾策略
    - 删除一个或者多个已排队分组
  - 主动队列管理——随机早期检测RED
    - 随时计算平均队列长度avg_{th}
    - 最小阈值min_{th}、最大阈值max_{th}
    - avg_{th}小于min_{th}，允许分组入列
    - avg_{th}大于max_{th}，分组被标记或丢弃
    - avg_{th}在min_{th}和max_{th}之间，按照概率标记或丢弃分组

4.4 网际协议:因特网中的转发和编址

- **主机、路由器的网络层组件**



IP数据报的格式


比特 0 1 2 3 4 5 6 7
优 先 级
D
T
R
C
未用
比特 0 4 8 16 19 24 31
版 本
首部长度
服 务 类 型
总 长 度
标 识
标志
片 偏 移
生 存 时 间
协 议
首 部 检 验 和
源 地 址
目 的 地 址
可 选 字 段 （长 度 可 变）
填 充
数 据 部 分
首部
固定部分
可变部分


传送
首 部
数 据 部 分
IP 数据报

IP数据报的格式（续）

一个IP数据报由首部和数据两部分组成。首部的前一部分长度是20字节，后一部分为可变长度。

（1）IP数据报首部固定部分中的各字段

- 版本：4bit，指IP协议的版本。通信双方使用的版本必须保持一致，目前使用的是IPV4。
- 首部长度：4bit。以4字节为一个单位。
- 服务类型：8bit，用来获取更好的服务。前3bit表示优先级，第4bit表示要求更低时延，第5比特表示要求更高吞吐量，第6比特表示要求更高可靠性，第7比特表示要求选择费用更低廉的路由，第8比特未用。

IP数据报的格式（续）

- 数据报长度：指首部和数据之和的长度，单位为字节。
- 标识：是为了使分片后的各数据报片最后能准确地重装
- 标志：3bit，前2bit有意义。
  - 最低位为MF，MF=1即表示后面还有分片的数据报，MF=0表示这已是若干数据报片的最后一个。
  - 第二位记为DF，当DF=0时才允许分片
- 片偏移：13bit，较长的分组在分片后，某个片在原分组中的相对位置。以8字节为单位。
- 寿命：记为TTL（Time To Live），又称生存时间。每当数据报经过一台路由器，该字段值减1。该值为0时被丢弃。

IP数据报的格式（续）

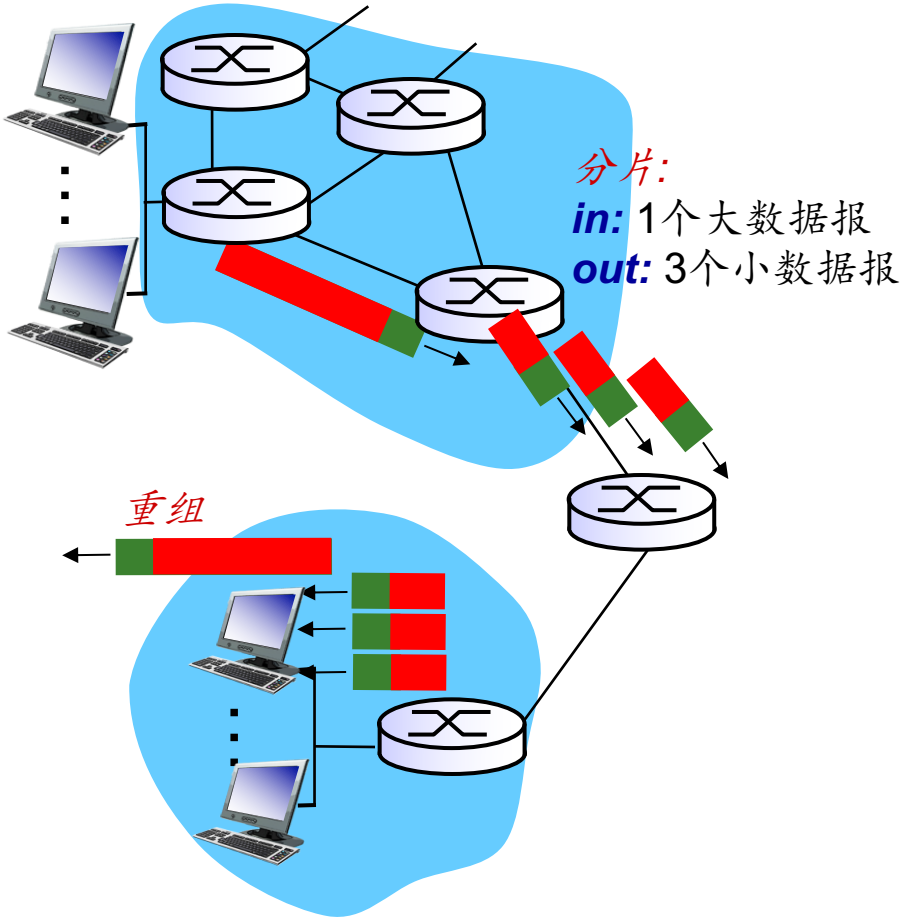
- 协议：8bit，协议字段指出此数据报携带的运输层数据是使用何种协议，以便目的主机的IP层知道应将数据部分上交给哪个处理过程。（如6表示TCP、17表示UDP）
- 首部校验和：此字段只检验数据报首部而不包括数据部分以减少工作量。和运输层（TCP、UDP）一起进行重复校验的原因：校验的内容不同；可能不属于同一个协议栈。
- 源地址和目的地址：各占4个字节。

（2）IP首部的可变部分：

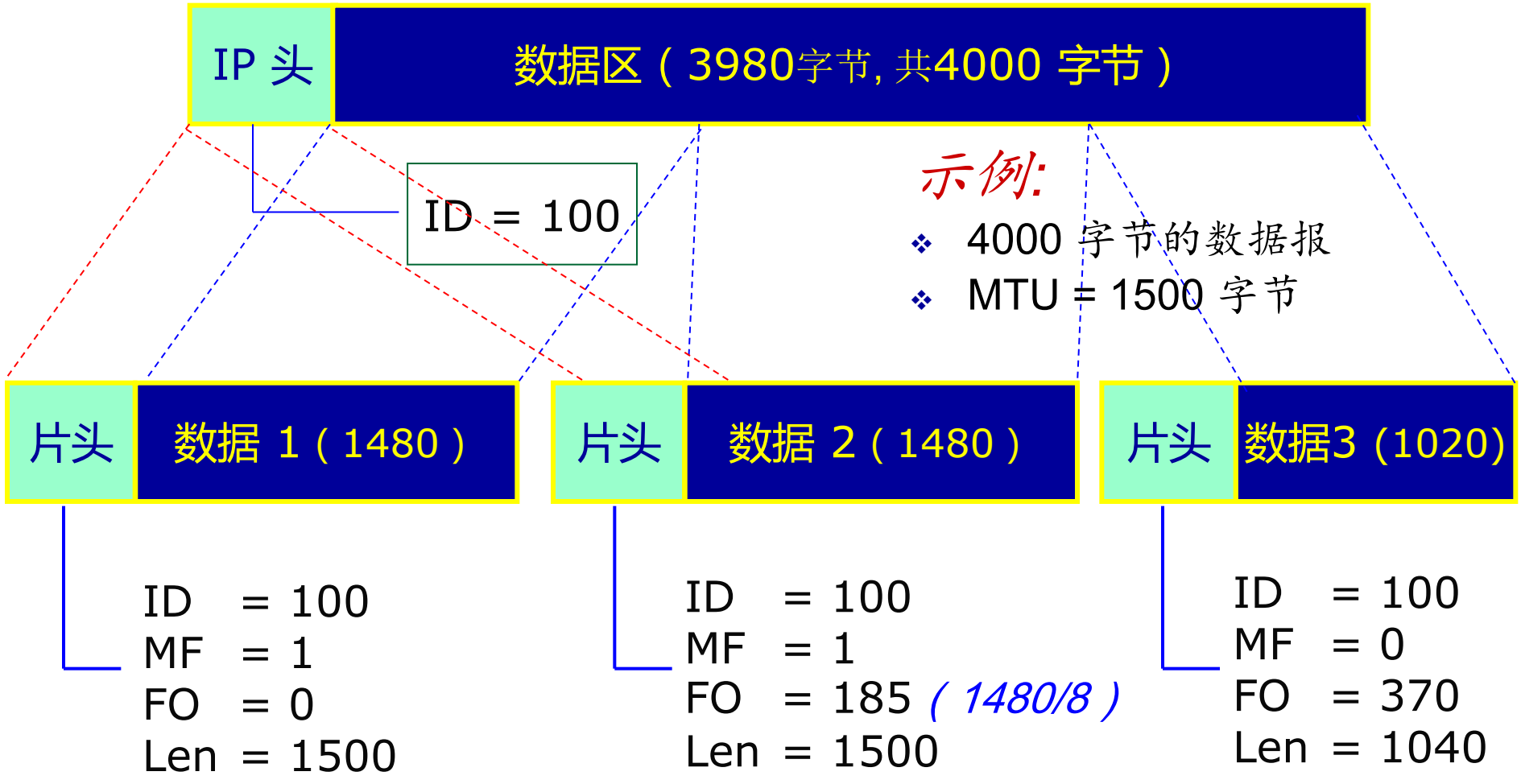
就是一个选项字段，用来支持排错、测量及安全等措施。但由于给服务器的处理带来麻烦，因此一般不使用。

4.4 网际协议:因特网中的转发和编址

- **IP分片和重组**
  - 网络链路具有 MTU (最大传输单位)属性——是由链路层最大帧的限制决定的
    - 不同类型的链路有不同的MTU值
  - 大的IP数据报在网络中会被分成小的分片
    - 一个数据报变成了几个数据报
    - 重组只在目的主机进行
    - 数据报头部的标识、标志以及片偏移字段用于目的主机对接收的分片进行重组



4.4 网际协议:因特网中的转发和编址

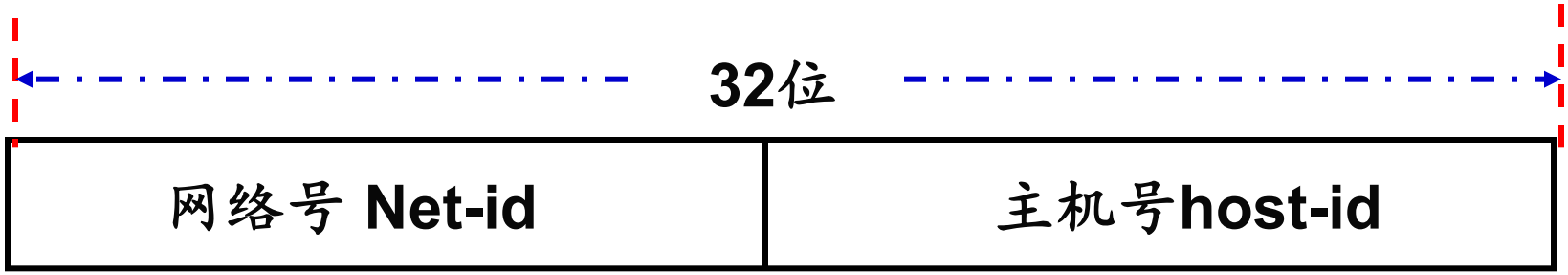


4.4 网际协议:因特网中的转发和编址

- **IP 地址**
  - 32位主机或路由器的接口标志符
  - 接口:连接主机,路由器之间的物理链路
    - 一般说来，路由器有多个接口
    - 主机也有可能有多个接口
  - IP 地址只和接口有关, 而与主机,路由器却没有太多关联

4.4 网际协议:因特网中的转发和编址

- IP地址的结构



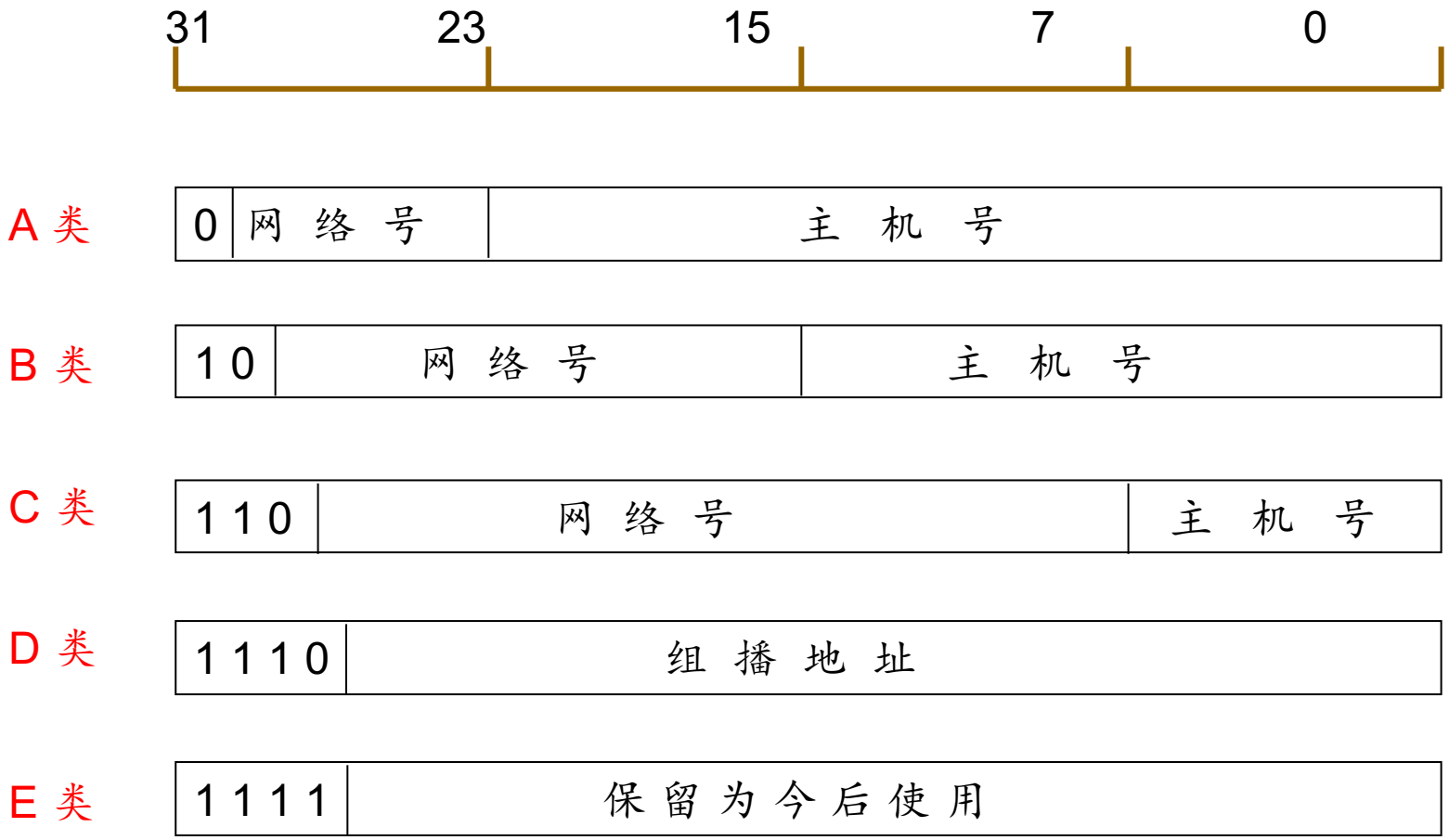
- IP地址的表示方法

223.1.1.1 = 11011111 00000001 00000001 00000001

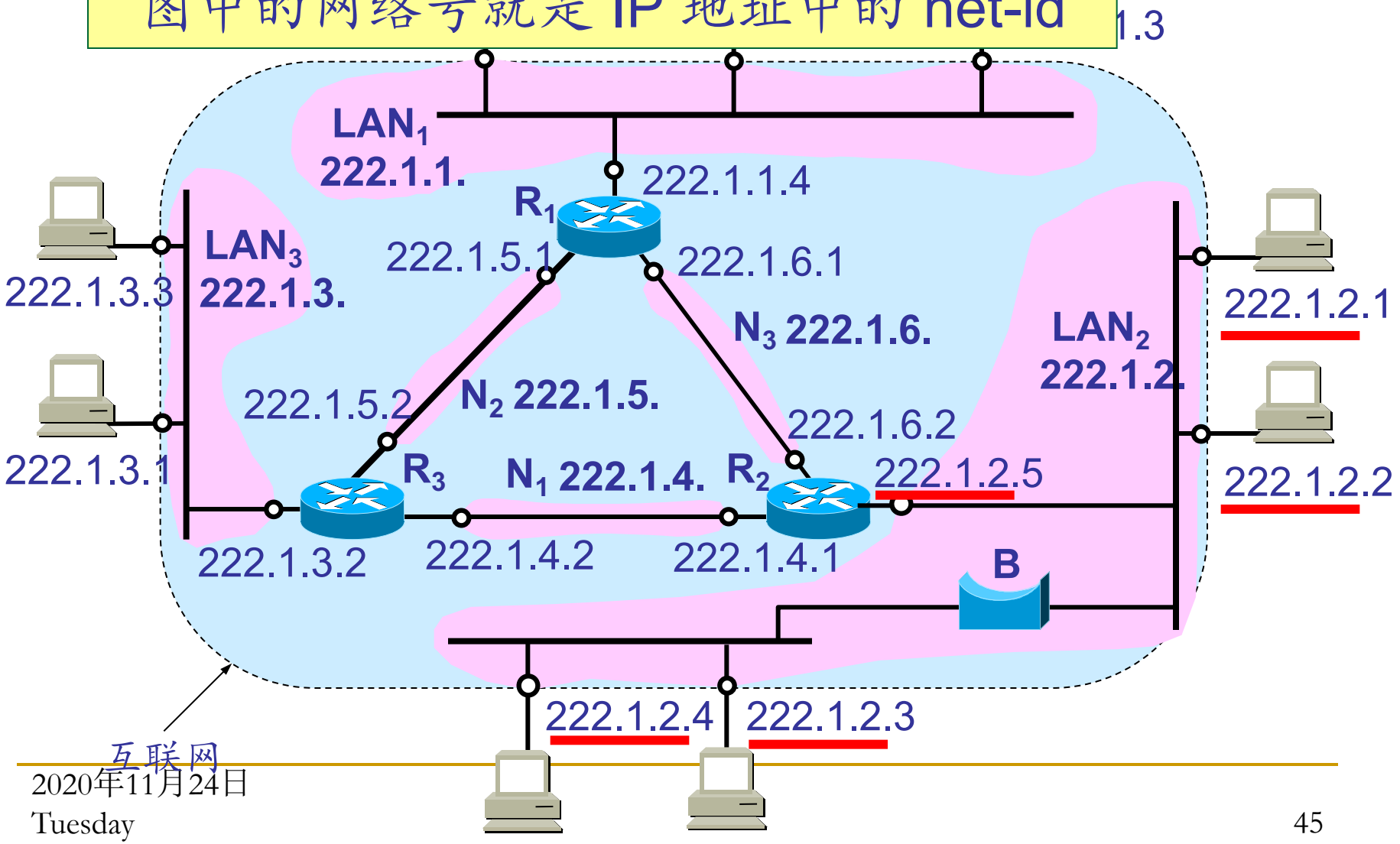
223 1 1 1

4.4 网际协议:因特网中的转发和编址

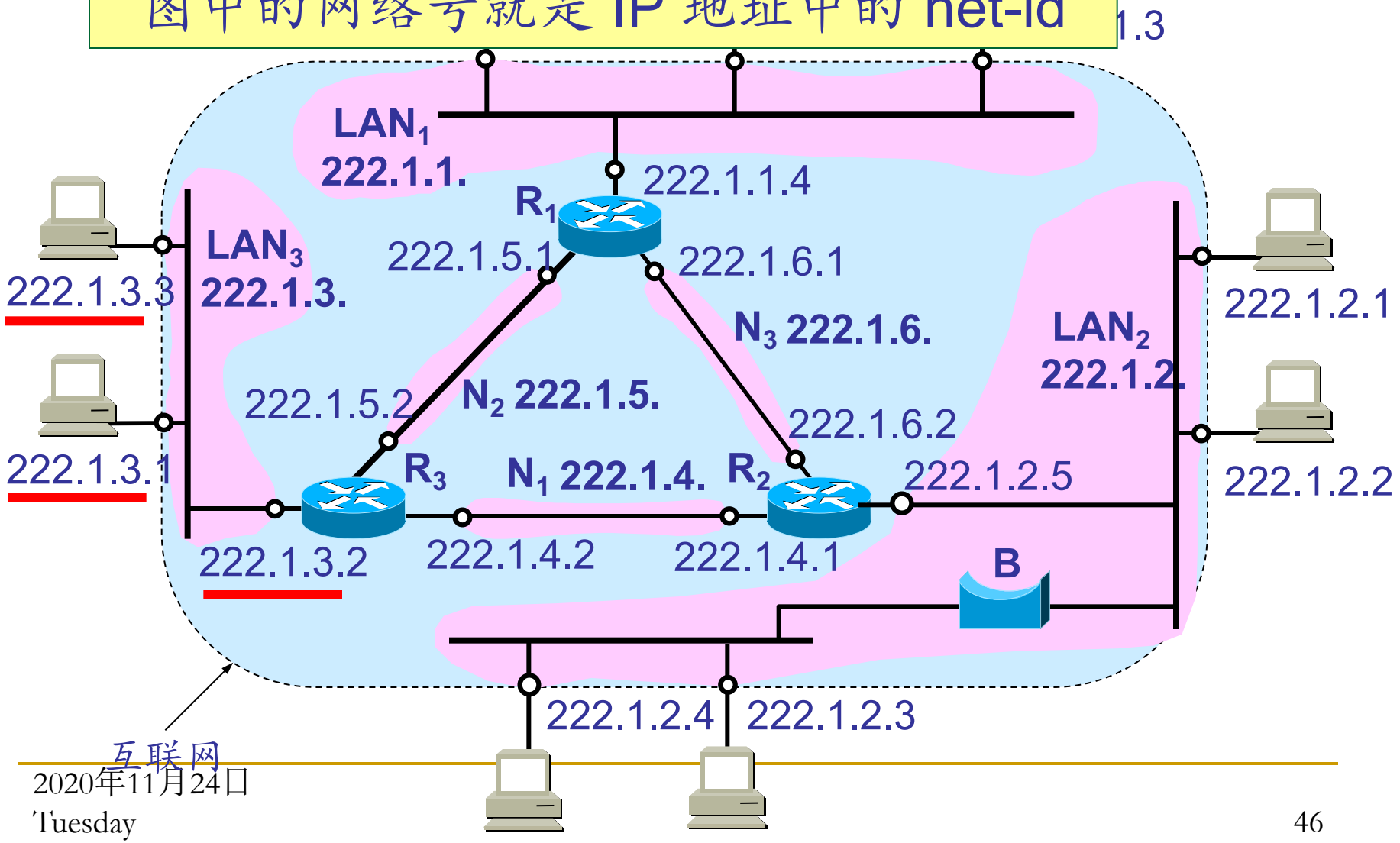
- IP地址的分类



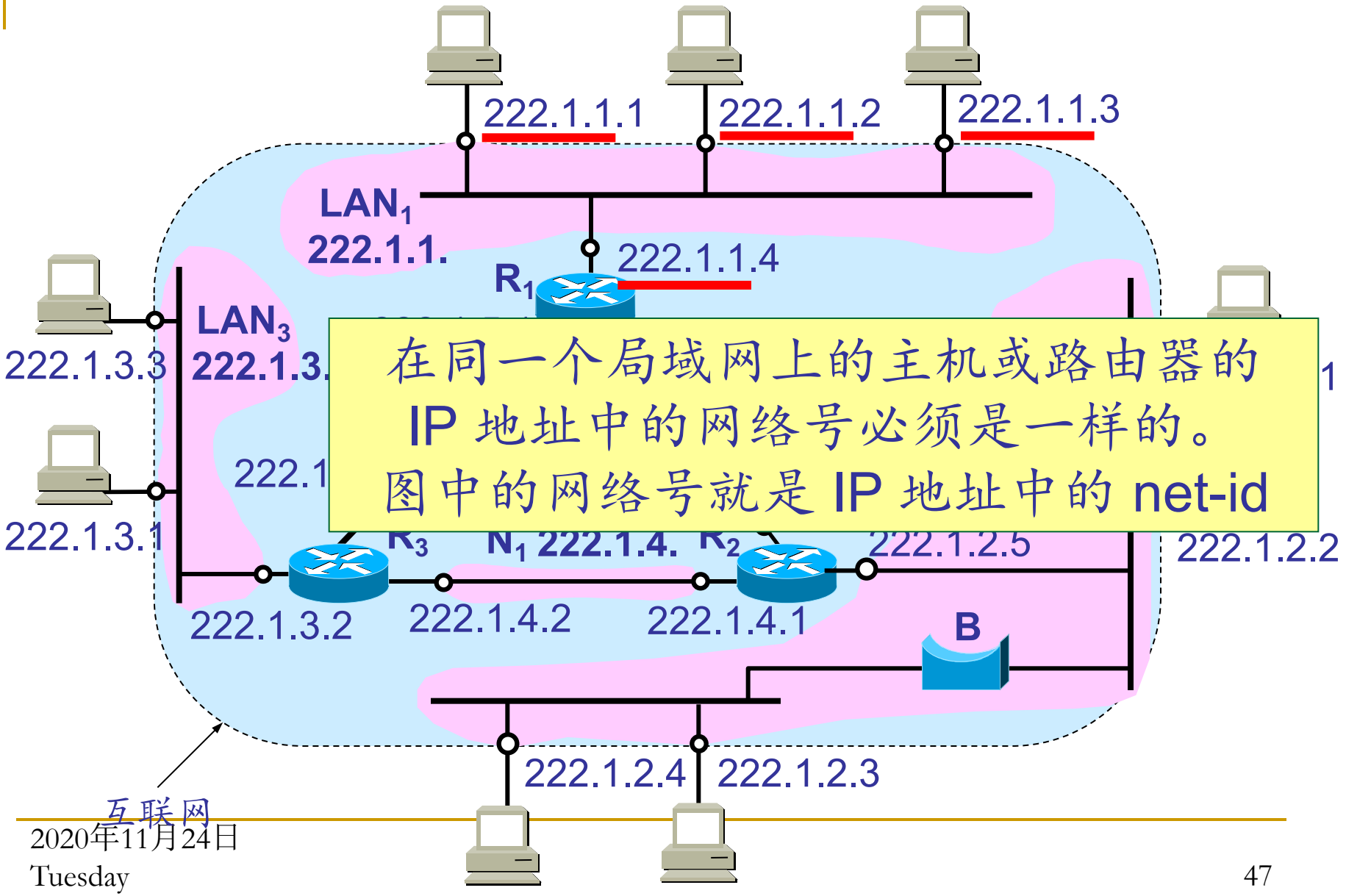
在同一个局域网上的主机或路由器的 IP 地址中的网络号必须是一样的。
图中的网络号就是 IP 地址中的 net-id



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

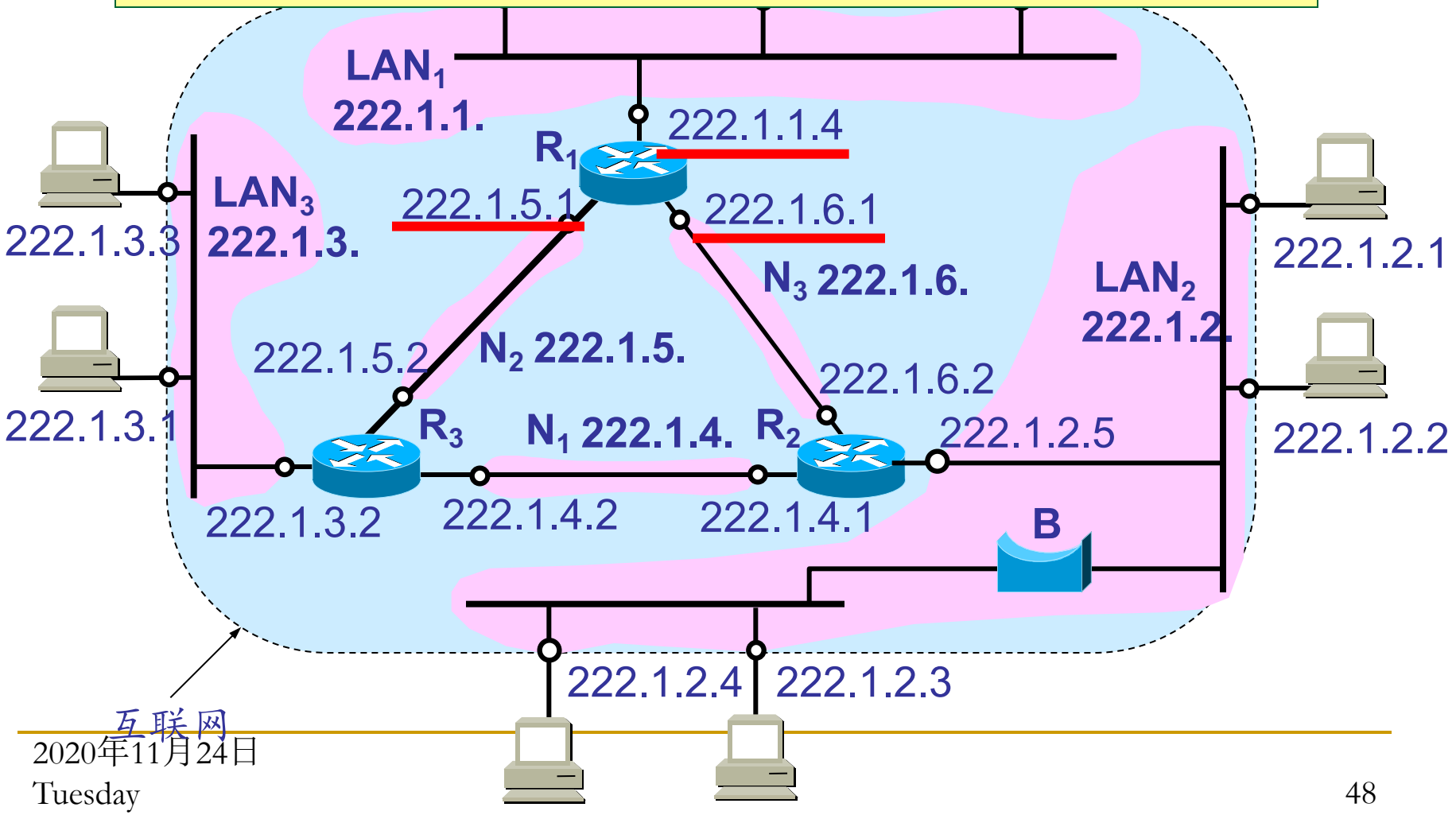


互联网中的 IP 地址

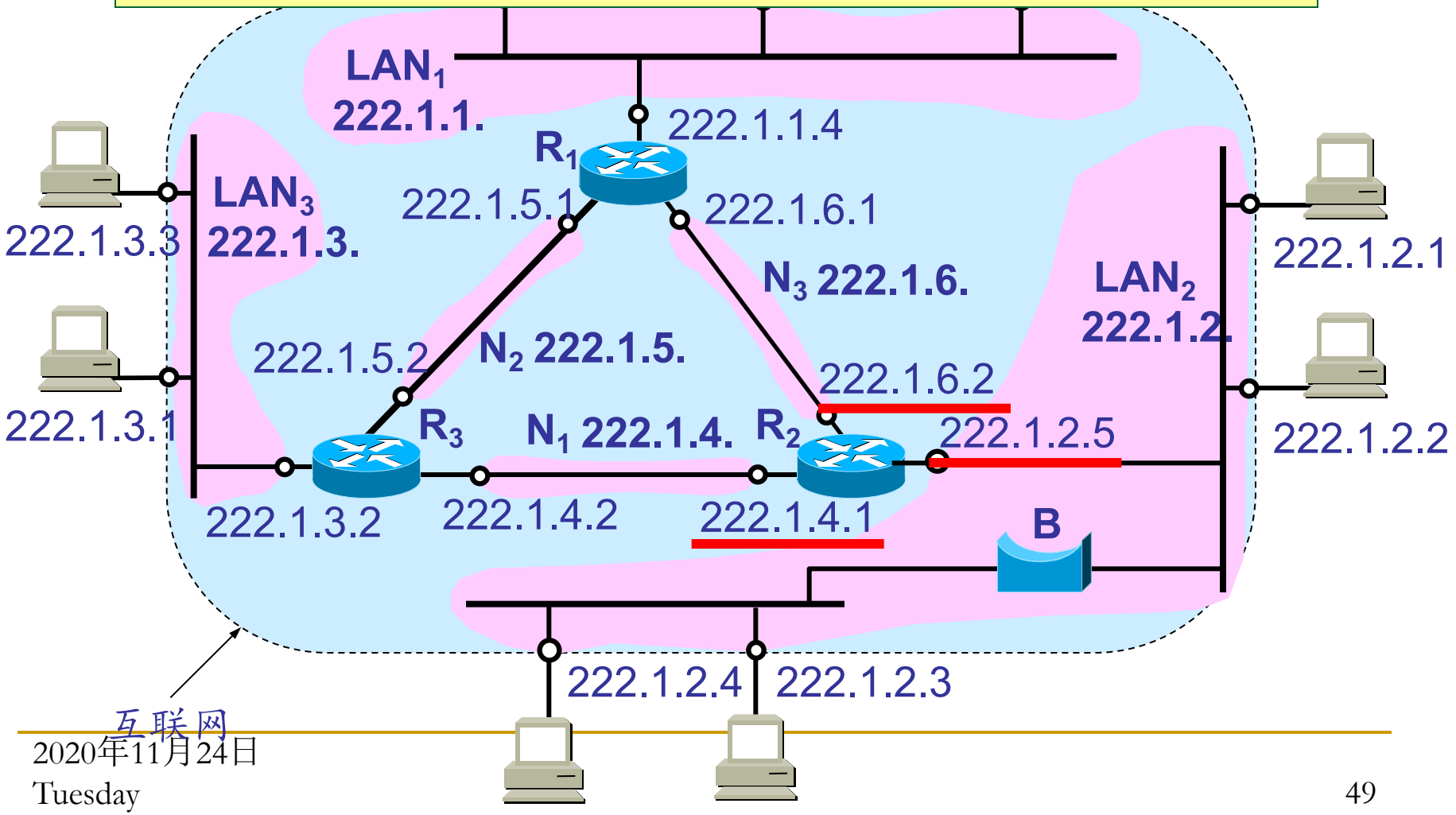


在同一个局域网上的主机或路由器的 IP 地址中的网络号必须是一样的。
图中的网络号就是 IP 地址中的 net-id

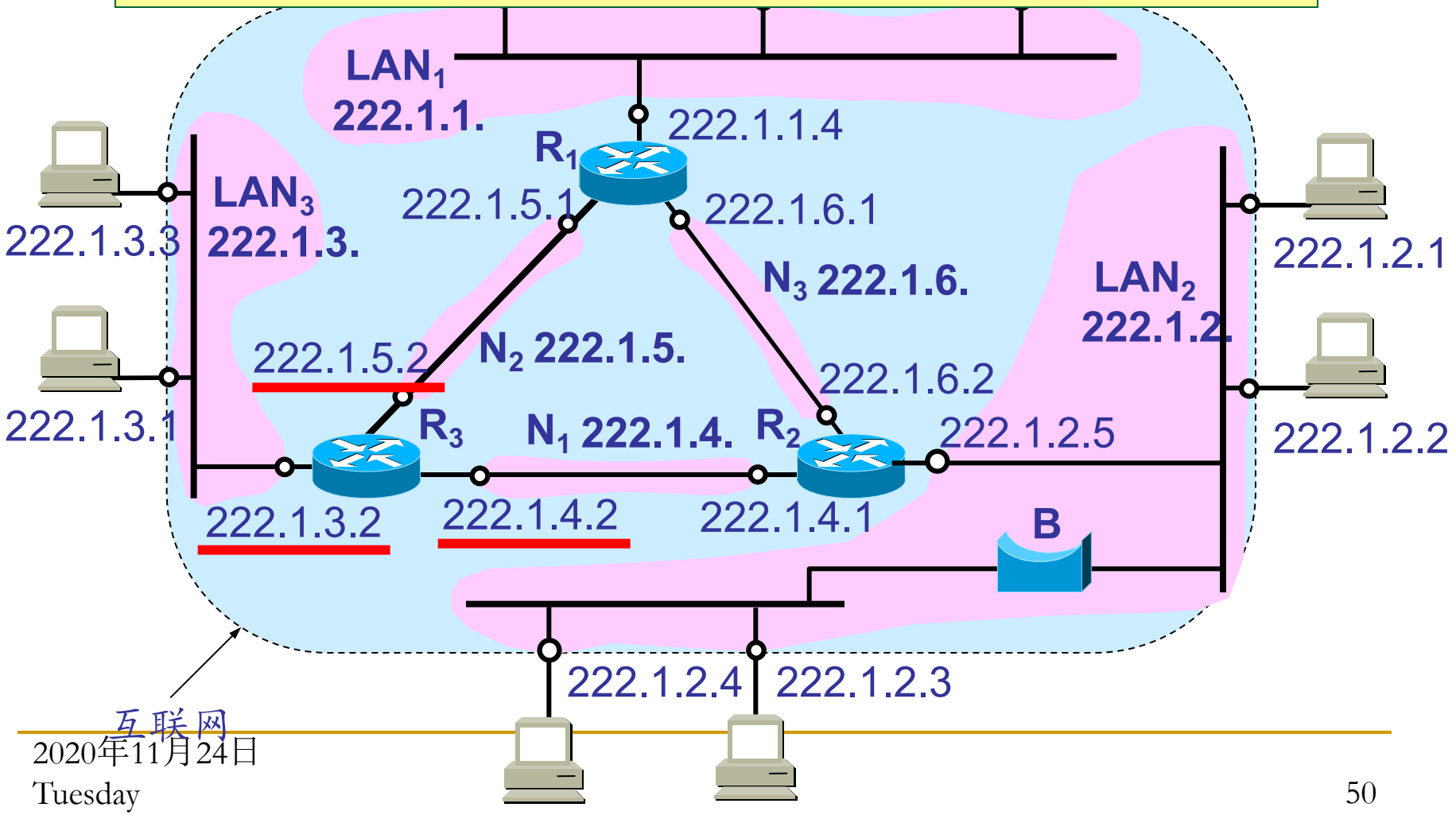
路由器总是具有两个或两个以上的 IP 地址。
路由器的每一个接口都有一个
不同网络号的 IP 地址。



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。



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。



4.4 网际协议:因特网中的转发和编址

子网的划分

问题:

一家跨国公司在海外拥有26家分支机构，包括总部在内，共计27家机构，该公司所有机构共有54320台计算机需要IP地址，总部向ICANN申请了一个B类地址块，现需要分配给所有的机构，如何分配？

★ 由总部管理员统一分配，集中管理 （你愿意当这个管理员么？）

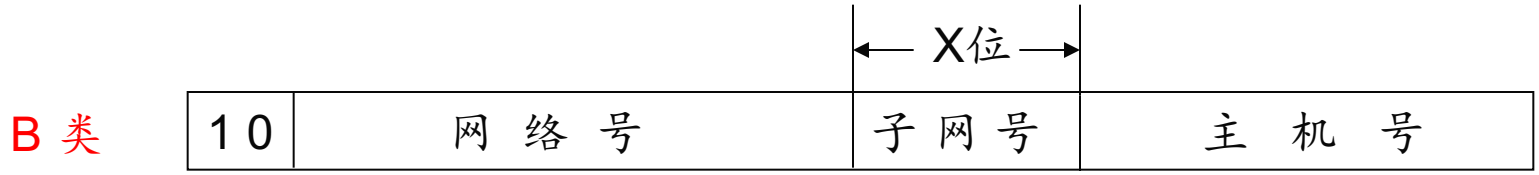
★ 总部管理员将IP地址分成27个子块，每个机构一块，各机构管理员内部自行分配

划分子网

4.4 网际协议:因特网中的转发和编址

- 子网划分的方法

从主机号中借用一部分位数作为子网号

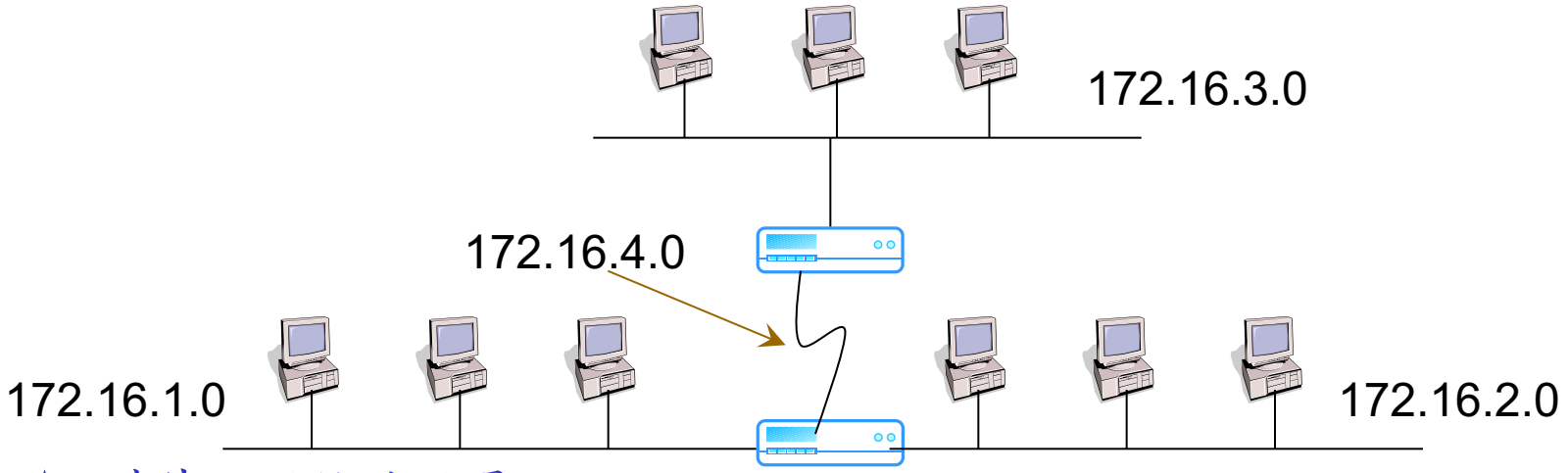


管理员划分好子网后，当用户拿到一个IP地址，用户怎么知道被借用了多少位来作为子网号？

通过子网掩码

4.4 网际协议:因特网中的转发和编址

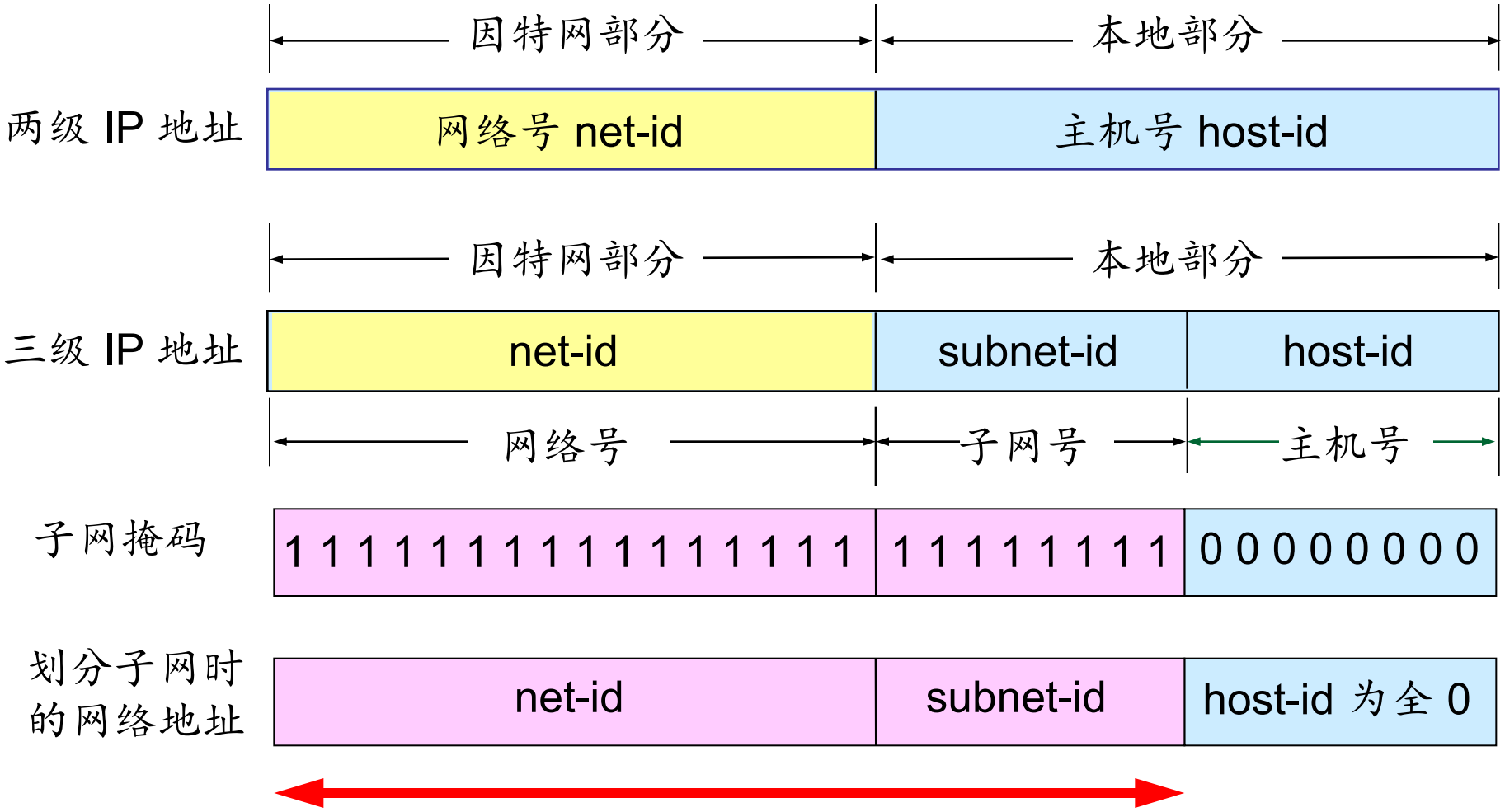
- 子网掩码
  - 获得方法：通过将网络号和子网号相应的位置全置1，主机号相应的位置全置0，即可得到子网掩码
  - 作用：对外隐藏子网的存在，对内指示网络号和子网号的位置



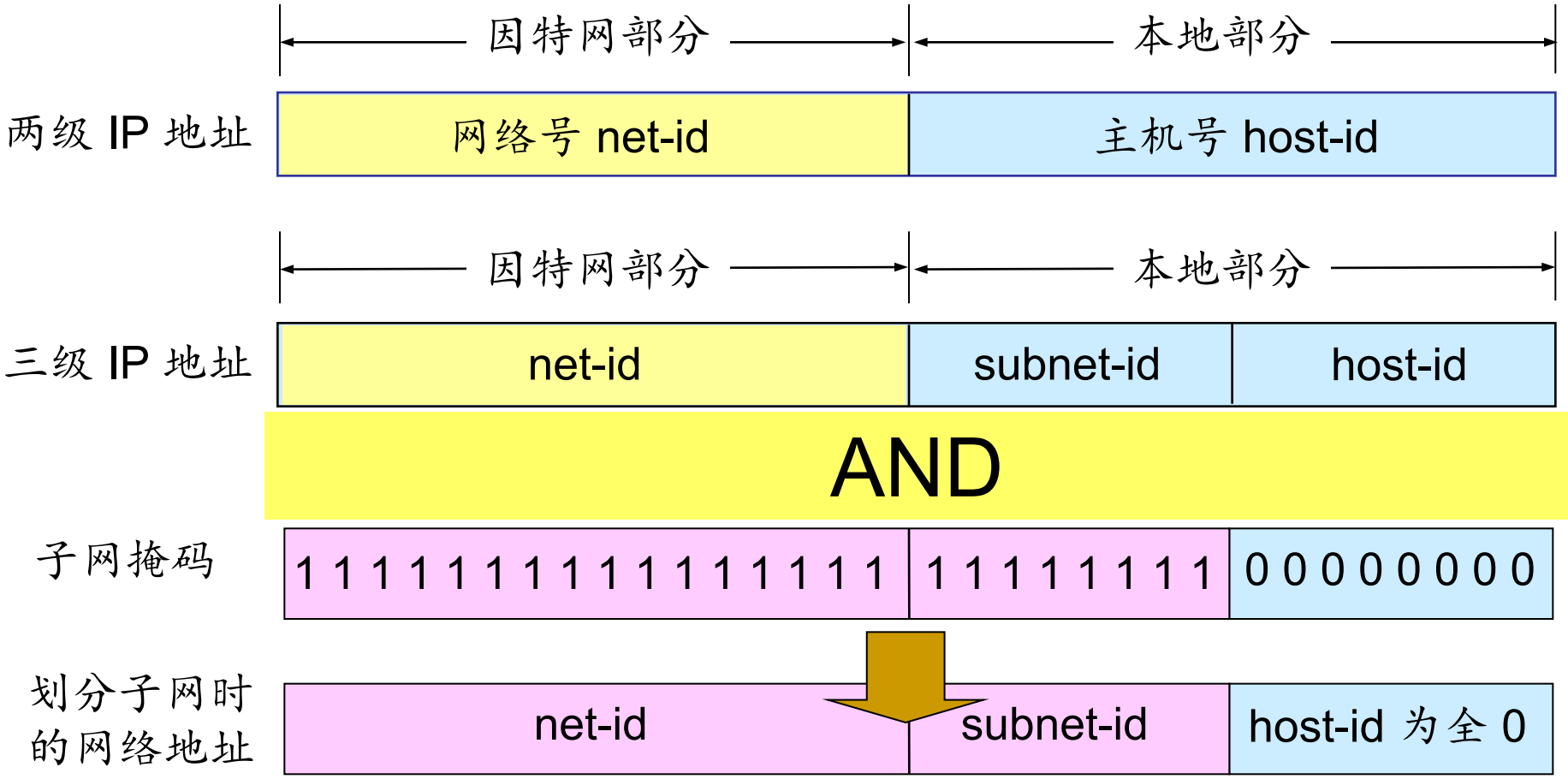
★ 对外，网络地址是 172.16.0.0

★ 对内，存在四个子网，子网掩码为255.255.255.0

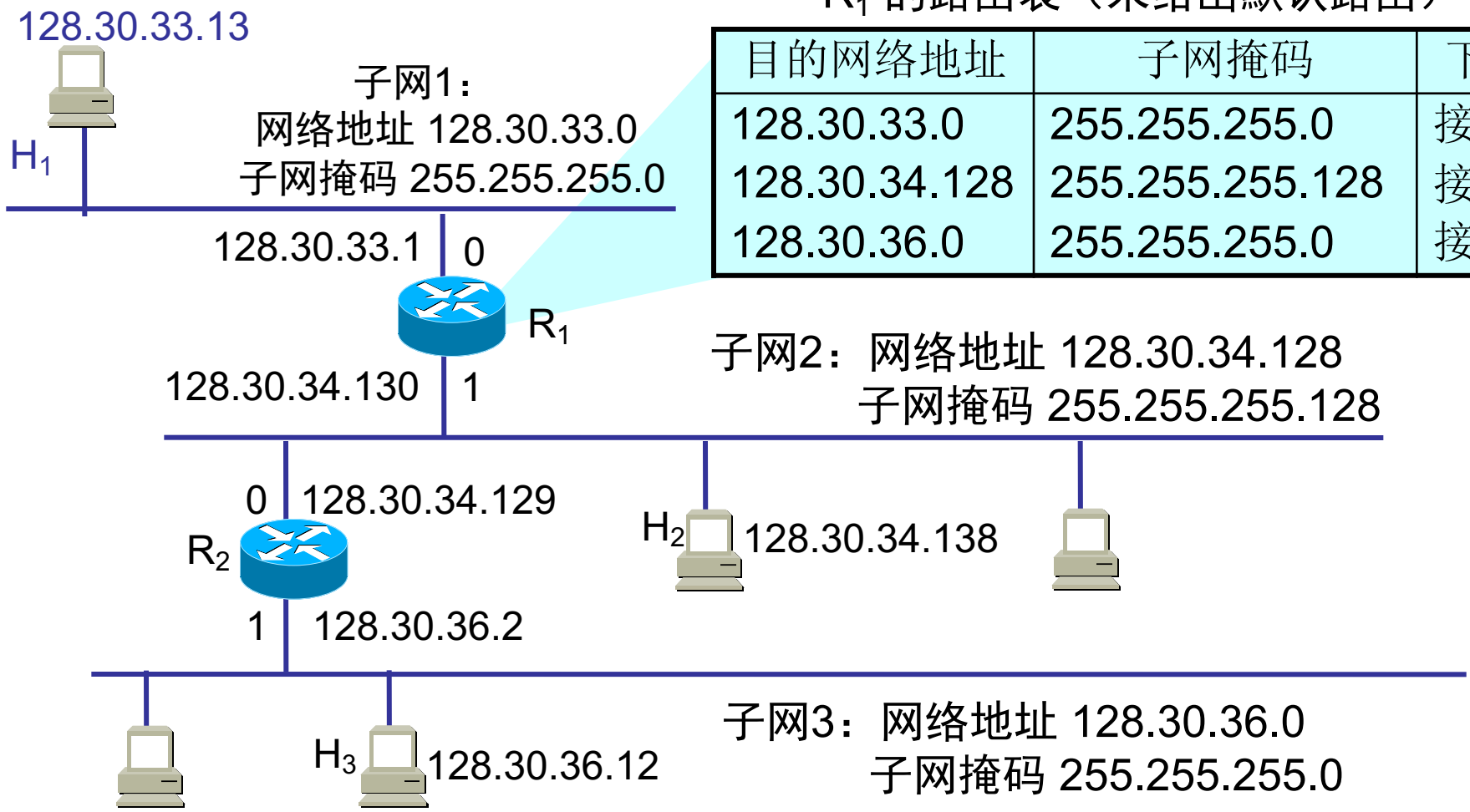
IP 地址的各字段和子网掩码



(IP 地址) AND (子网掩码) = 网络地址



划分子网后分组的转发举例



R_1 的路由表（未给出默认路由）

目的网络地址	子网掩码	下一跳
128.30.33.0	255.255.255.0	接口 0
128.30.34.128	255.255.255.128	接口 1
128.30.36.0	255.255.255.0	接口 1

H_1 并不知道 H_2 连接在哪一个网络上。
H_1 仅仅知道 H_2 的 IP 地址是128.30.34.138

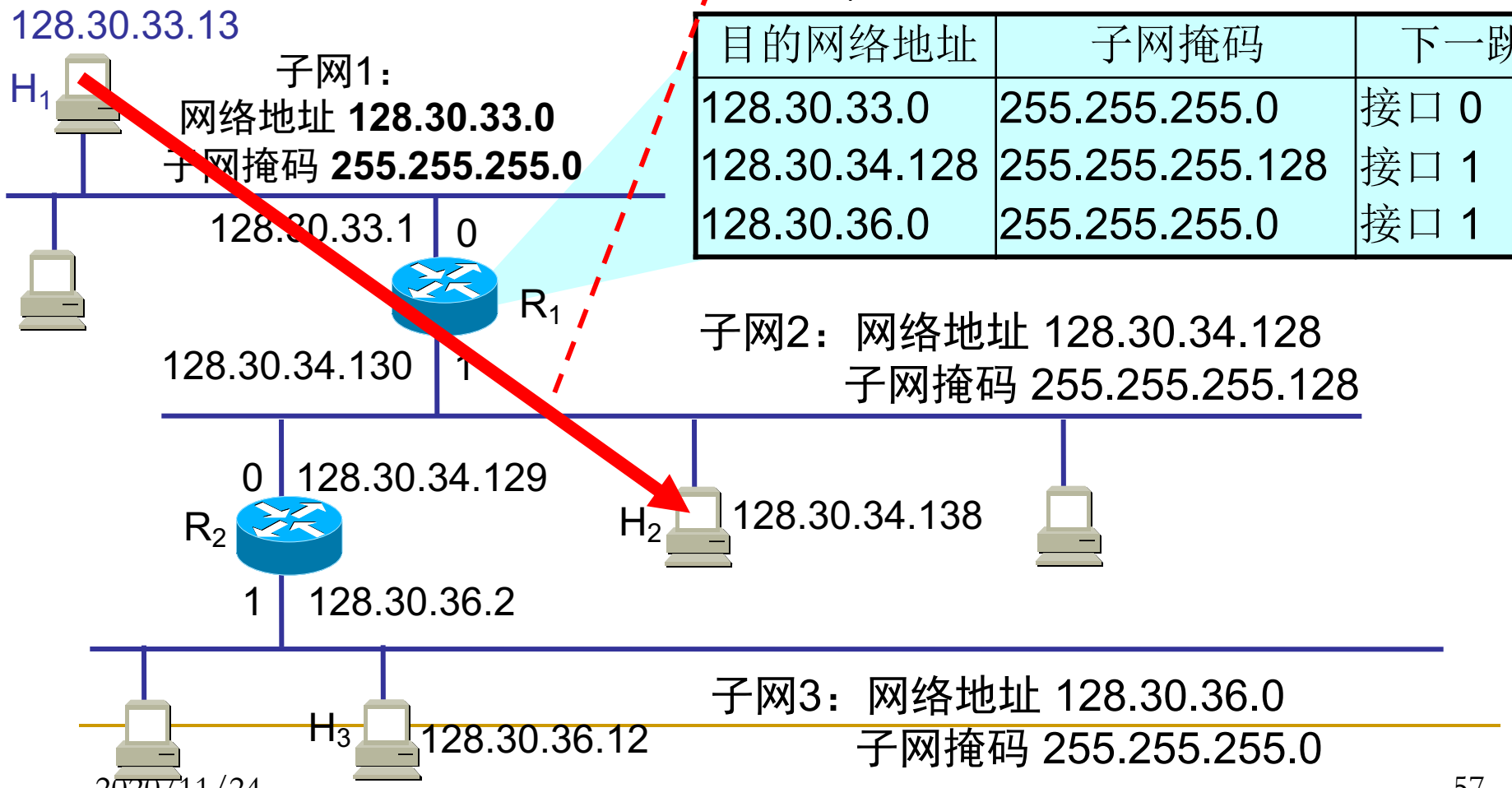
主机 H_1 要发送分组给 H_2

因此 H_1 首先检查主机 128.30.34.138
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。

要发送的分组的目的 IP 地址：128.30.34.138

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	255.255.255.0	接口 0
128.30.34.128	255.255.255.128	接口 1
128.30.36.0	255.255.255.0	接口 1



主机 H_1 首先将本子网的子网掩码 255.255.255.0与分组的 IP 地址 128.30.34.138 逐比特相“与”(AND 操作)

255.255.255.0 AND 128.30.34.138 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 0 AND 138 即可。

0 → 00000000

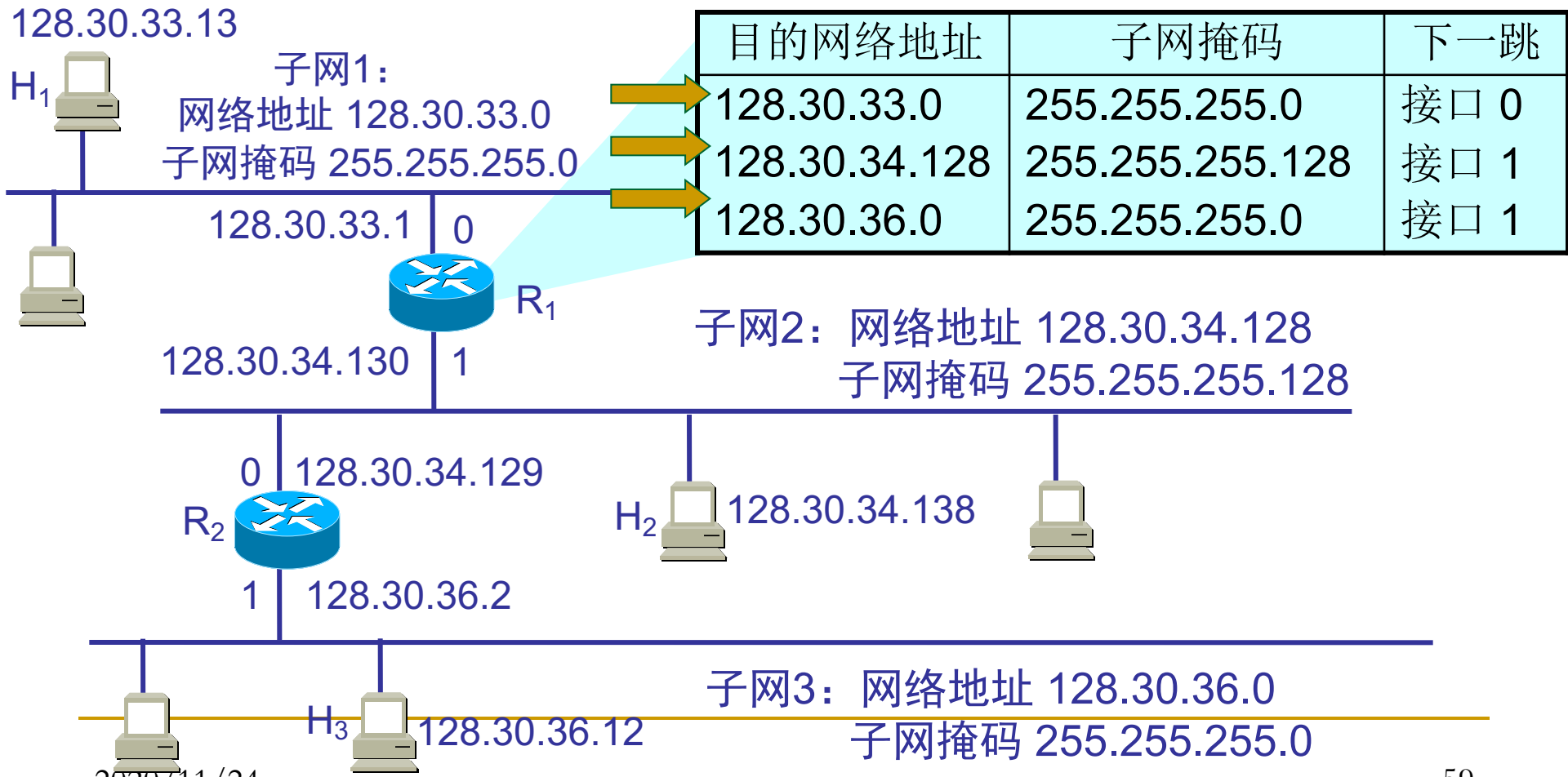
138 → 10001010

逐比特 AND 操作后 00000000 → 0

逐比特 AND 操作	255.255.255.0 128. 30. 34.138
	128. 30. 34.0

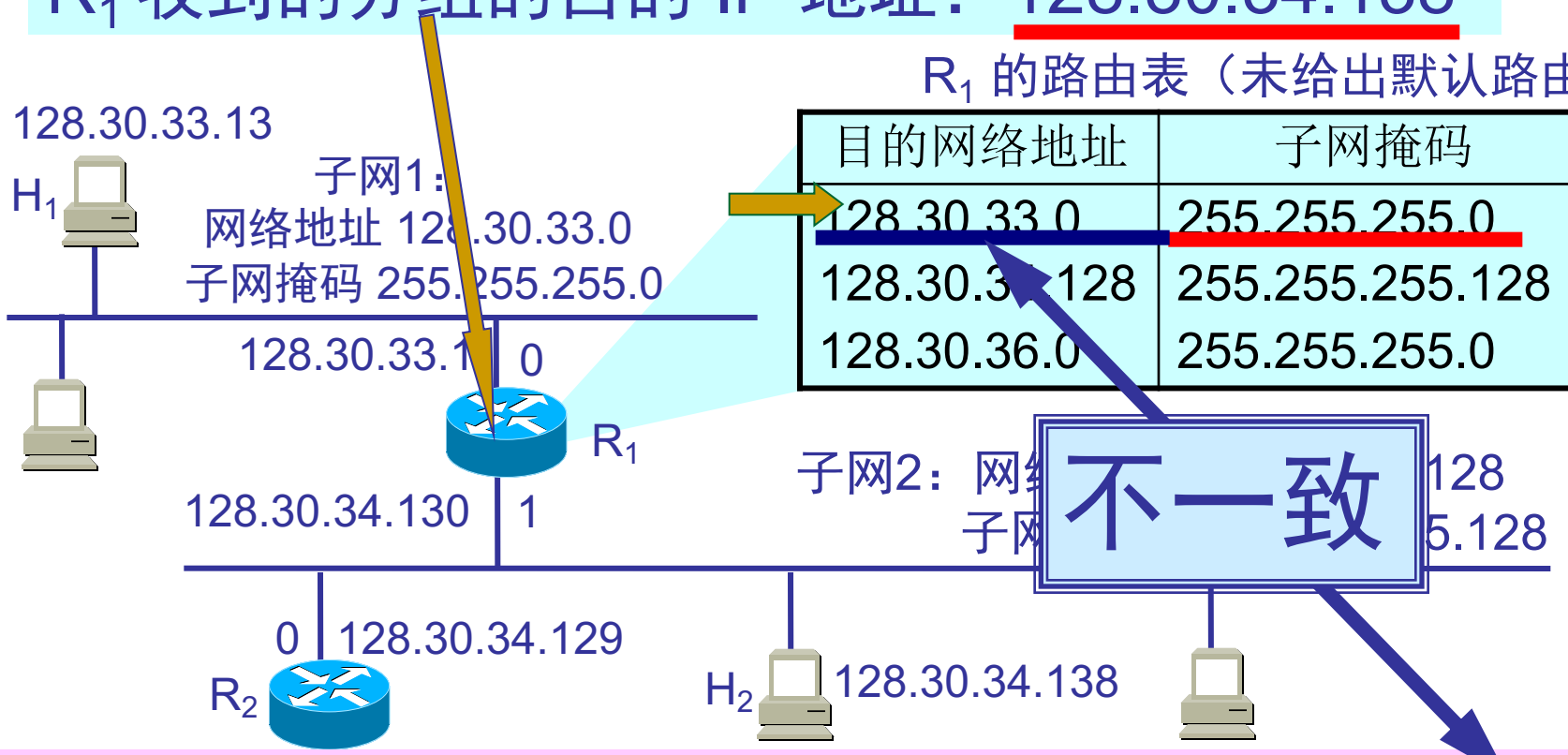
≠ H_1 的网络地址

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 128.30.34.138 逐比特 AND 操作

R_1 收到的分组的目的 IP 地址：128.30.34.138



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	255.255.255.0	接口 0
128.30.3[illegible]128	255.255.255.128	接口 1
128.30.36.0	255.255.255.0	接口 1

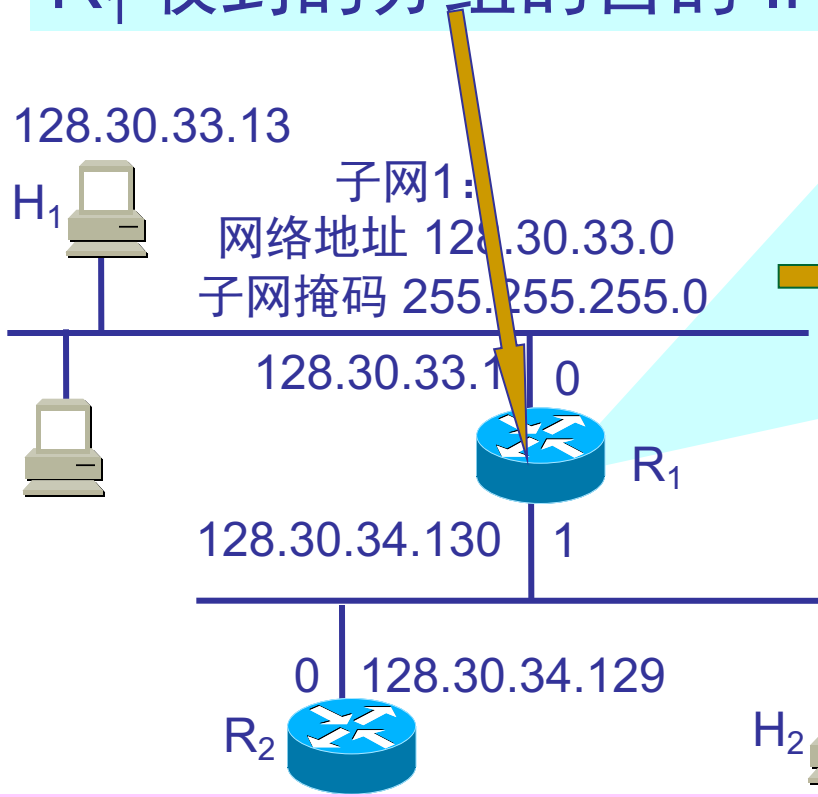
255.255.255.0 AND 128.30.34.138 = 128.30.34.0

不匹配!

（因为128.30.34.0 与路由表中的 128.30.33.0 不一致）

路由器 R_1 再用路由表中第 2 个项目的
子网掩码和 128.30.34.138 逐比特 AND 操作

R_1 收到的分组的目的 IP 地址：128.30.34.138



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	255.255.255.0	接口 0
128.30.34.128	255.255.255.128	接口 1
128.30.36.0	255.255.255.0	接口 1

子网2：网络地址 128.30.34.128
子网掩码 255.255.255.128

一致!

255.255.255.128 AND 128.30.34.138 = 128.30.34.128

匹配!

这表明子网 2 就是收到的分组所要寻找的目的网络（接口1转发）

4.4 网际协议:因特网中的转发和编址

- 一个子网规划的问题
  - 条件：
    - 一个C类地址：201.222.5.0
  - 目标
    - 20 SubNet
    - 5 PC/SubNet

子网掩码：255.255.255.248
255.255.255.11111000

  - 结果
    - 5位子网号、3位主机号
    - 201.222.5.8/255.255.255.248（201.222.5.00001000）
    - 201.222.5.16/255.255.255.248（201.222.5.00010000）

    ……

4.4 网际协议:因特网中的转发和编址

- **子网划分示例**

某公司现获得了一批IP地址，是：

218.103.24.112~218.103.24.127

请问：

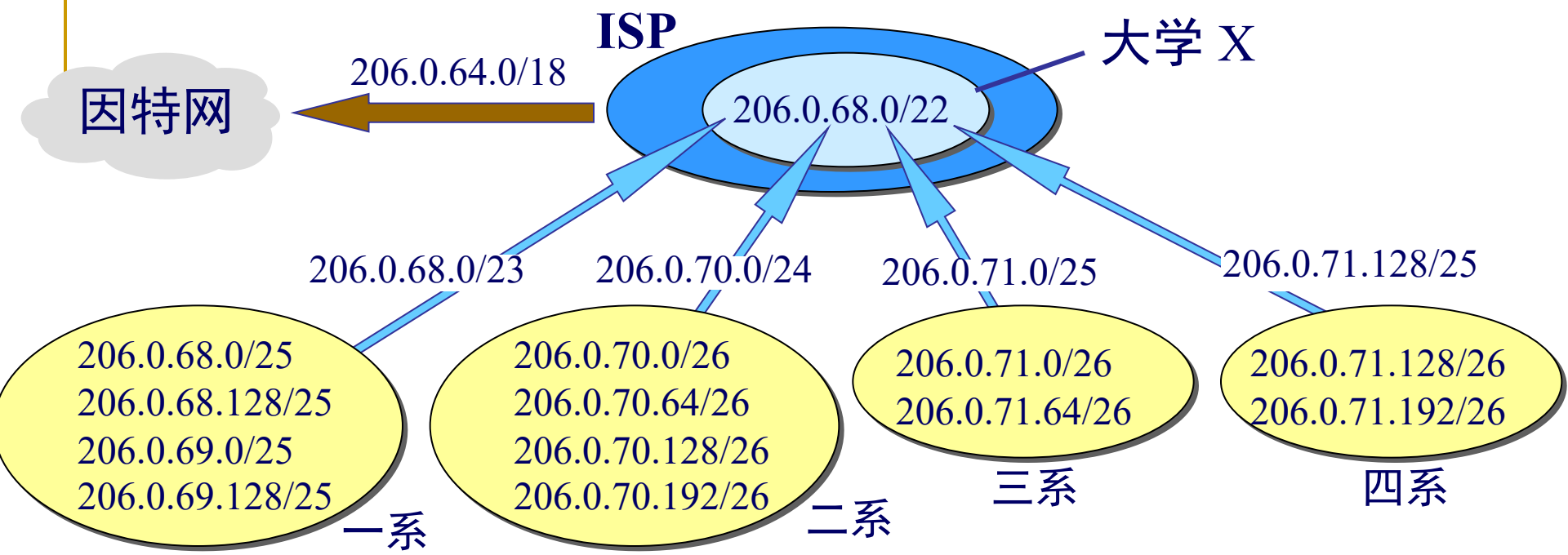
(1) 该批IP地址的网络地址是多少？

(2) 该批IP地址的子网掩码是多少？

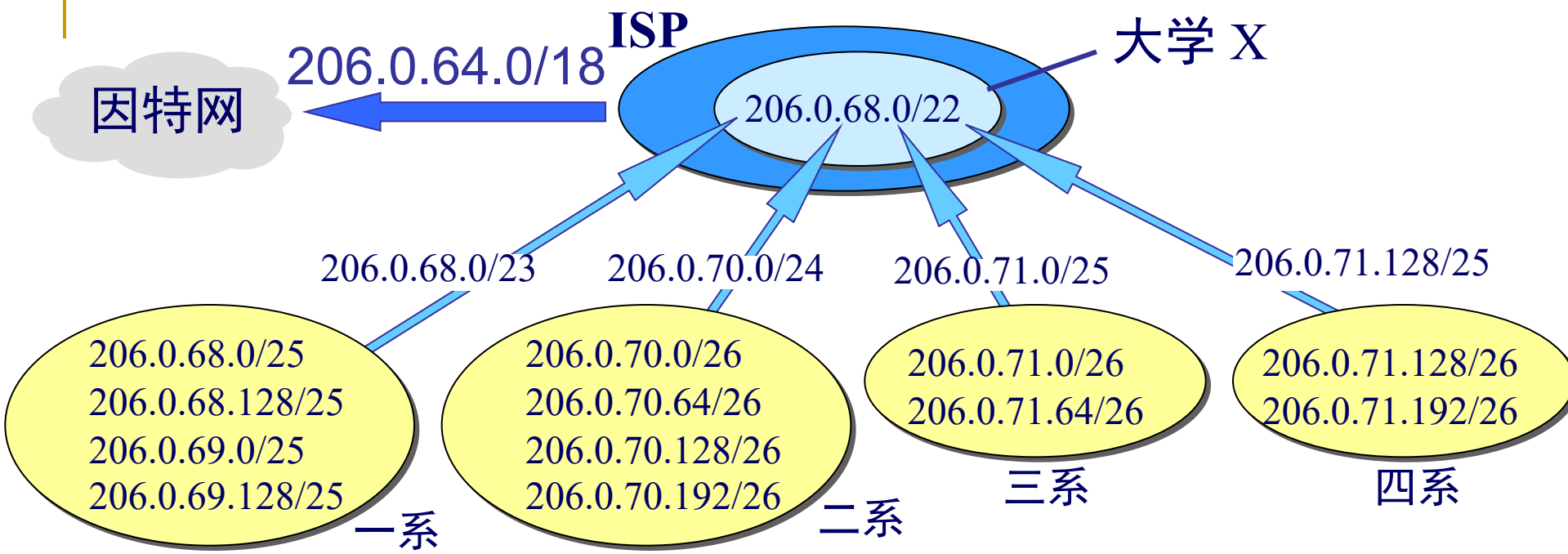
(3) 如果针对该批IP地址进行进一步的子网划分，还可以划分为多少个子网？请用“网络号/子网掩码”来表示这些子网。

4.4 网际协议:因特网中的转发和编址

- **CIDR(Classless InterDomain Routing)**
  - 背景
    - 地址空间的利用率低, 地址空间面临耗尽
    - e.g., 一个B类网址可以容纳65K台主机, 但可能被一个只有2K台主机的单位占据
  - CIDR编址格式
    - IP地址 ::= {<网络前缀>, <主机号>}
    - 地址的网络部分长度任意
    - 地址格式: a.b.c.d/x, 这里的 x表示地址中网络部分的位数 #
    - 斜线记法：192.168.0.1/24
    - 简写记法：10.0.0.0/10　　10/10



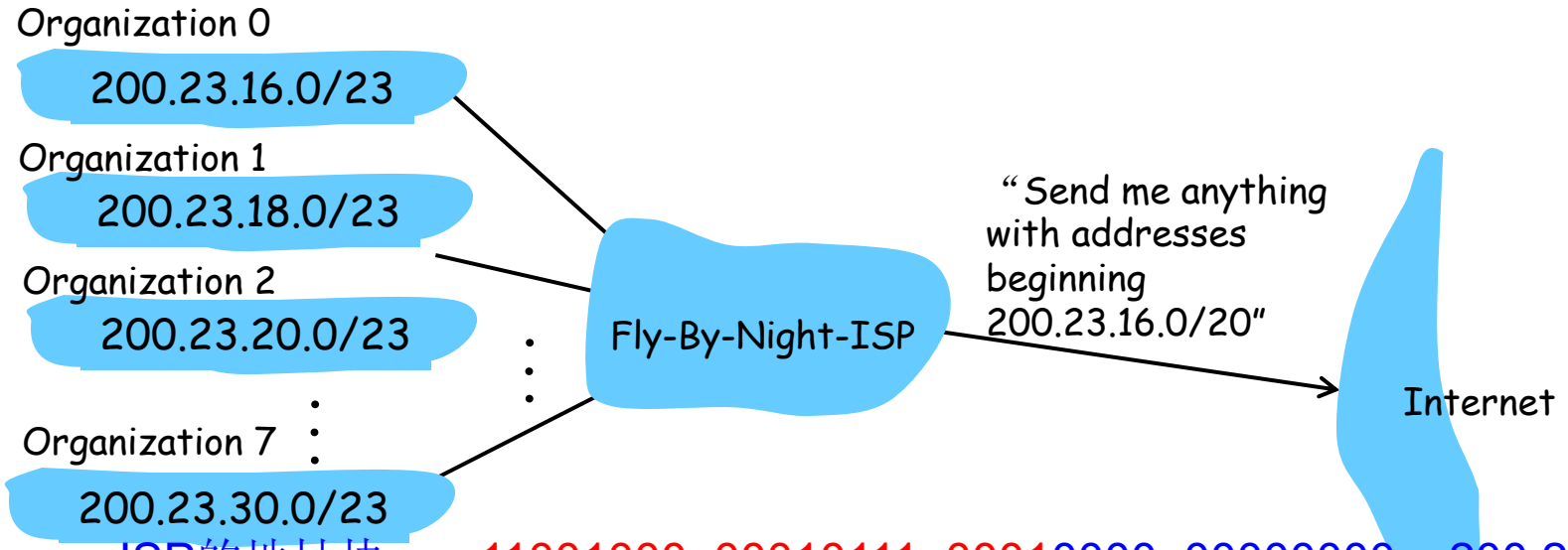
单位	地址块	二进制表示	地址数
ISP	206.0.64.0/18	11001110.00000000.01*	16384
大学	206.0.68.0/22	11001110.00000000.010001*	1024
一系	206.0.68.0/23	11001110.00000000.0100010*	512
二系	206.0.70.0/24	11001110.00000000.01000110.*	256
三系	206.0.71.0/25	11001110.00000000.01000111.0*	128
四系	206.0.71.128/25	11001110.00000000.01000111.1*	128



这个 ISP 共有 64 个 C 类网络。如果不采用 CIDR 技术，则在与该 ISP 的路由器交换路由信息的每一个路由器的路由表中，就需要有 64 个项目。但采用地址聚合后，只需用路由聚合后的 1 个项目 206.0.64.0/18 就能找到该 ISP。

层次编址好处：路由聚合

假设某ISP已经被分配地址块：200.23.16.0/20。ISP将该地址块进一步划分为8块较小的地址块，分配给8个组织。则ISP只需告诉外界（Internet）：向我发送以200.23.16.0/20开始的任何东西。外界不需要知道该地址块内部的细节。这种使用单个网络前缀通告多个网络的能力称为路由聚合。路由聚合可以大大减少路由器中转发表中的项数。

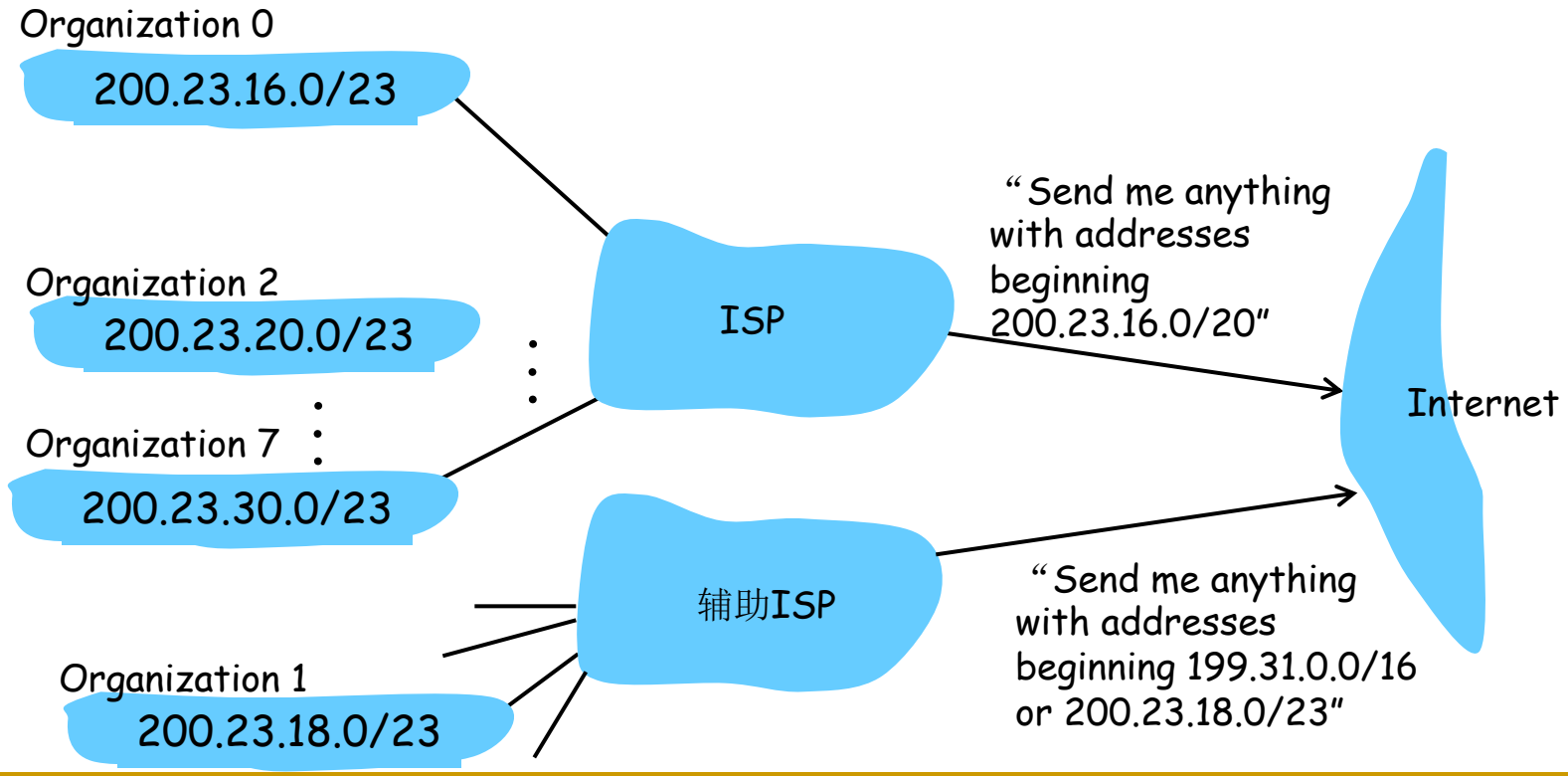


ISP的地址块	11001000	00010111	00010000	00000000	200.23.16.0/20
Organization 0	11001000	00010111	00010000	00000000	200.23.16.0/23
Organization 1	11001000	00010111	00010010	00000000	200.23.18.0/23
Organization 2	11001000	00010111	00010100	00000000	200.23.20.0/23
...	....		....	....	
Organization 7	11001000	00010111	00011110	00000000	200.23.30.0/23

层次编址: 更具体的路由

假设有一个辅助ISP，其地址块为199.31.0.0/16。组织1现在改为通过辅助ISP与Internet相连。

显然组织1的地址块在该地址块以外。一种方法是对组织1内的所有主机和路由器重新进行编址，使其地址在199.31.0.0/16地址块内，但代价很高。更好的办法是由辅助ISP发出通告：向我发送以199.31.0.0/16或200.23.18.0/23开始的任何东西。



4.4 网际协议:因特网中的转发和编址

- 最长前缀匹配
  - 使用 CIDR 时，路由表中的每个项目由“网络前缀”和“下一跳地址”组成。在查找路由表时可能会得到不止一个匹配结果。
  - 应当从匹配结果中选择具有最长网络前缀的路由：最长前缀匹配(longest-prefix matching)。
  - 网络前缀越长，其地址块就越小，因而路由就越具体。
  - 最长前缀匹配又称为最长匹配或最佳匹配。

收到分组的目的地址 D = 206.0.71.142

路由表中项目：206.0.68.0/22　　（ISP）

206.0.71.128/25　（四系）

查找路由表中的第 1 个项目

第 1 个项目 206.0.68.0/22 的掩码 M 有 22 个连续的 1。

M = 11111111 11111111 11111100 00000000

因此只需把 D 的第 3 个字节转换成二进制。

M = 11111111 11111111 11111100 00000000

AND　D = 206. 0. 01000111.10001110

206. 0. 01000100. 0

与 206.0.68.0/22 匹配！

收到分组的目的地址 D = 206.0.71.142

路由表中项目：206.0.68.0/22　　（ISP）

206.0.71.128/25　（四系）

再查找路由表中的第 2 个项目

第 2 个项目 206.0.71.128/25 的掩码 M 有 25 个连续的 1。

M = 11111111 11111111 11111111 10000000

因此只需把 D 的第 4 个字节转换成二进制。

	M =	11111111	11111111	11111111	10000000
AND	D =	206.	0.	71.	10001110
		206.	0.	71.	10000000

与 206.0.71.128/25 匹配

4.4 网际协议:因特网中的转发和编址

D AND (11111111 11111111 11111100 00000000)
= 206.0.68.0/22 匹配

D AND (11111111 11111111 11111111 10000000)
= 206.0.71.128/25 匹配

选择两个匹配的地址中更具体的一个，即选择最长前缀的地址。

4.4 网际协议:因特网中的转发和编址

右表为一个使用CIDR的路由表，请说明下列地址的下一跳各是什么？

(a) C4.5E.13.87 **B**

(b) C4.5E.22.09 **A**

(c) C3.41.80.02 **E**

(d) 5E.43.91.12 **F**

(e) C4.6D.31.2E **C**

(f) C4.6B.31.2E **D**

网络/前缀长度	下一跳点
C4.50.0.0/12	A
C4.5E.10.0/20	B
C4.60.0.0/12	C
C4.68.0.0/14	D
80.0.0.0/1	E
40.0.0.0/2	F
0.0.0.0/2	G

4.4 网际协议:因特网中的转发和编址

- **获取网络地址**

Q: 一个网络如何获得一块地址？

A:从ISP的地址空间中获得。

ISP的地址块	11001000 00010111 00010000	00000000	200.23.16.0/20
Organization 0	11001000 00010111 00010000	00000000	200.23.16.0/23
Organization 1	11001000 00010111 00010010	00000000	200.23.18.0/23
Organization 2	11001000 00010111 00010100	00000000	200.23.20.0/23
...	.....	....	....
Organization 7	11001000 00010111 00011110	00000000	200.23.30.0/23

4.4 网际协议:因特网中的转发和编址

Q: ISP 如何获得整块地址?

A: ICANN: Internet Corporation for Assigned Names and Numbers（因特网名字与号码分配团体）

- 分配IP地址
- 管理 DNS
- 分配域名, 解决域名纠纷

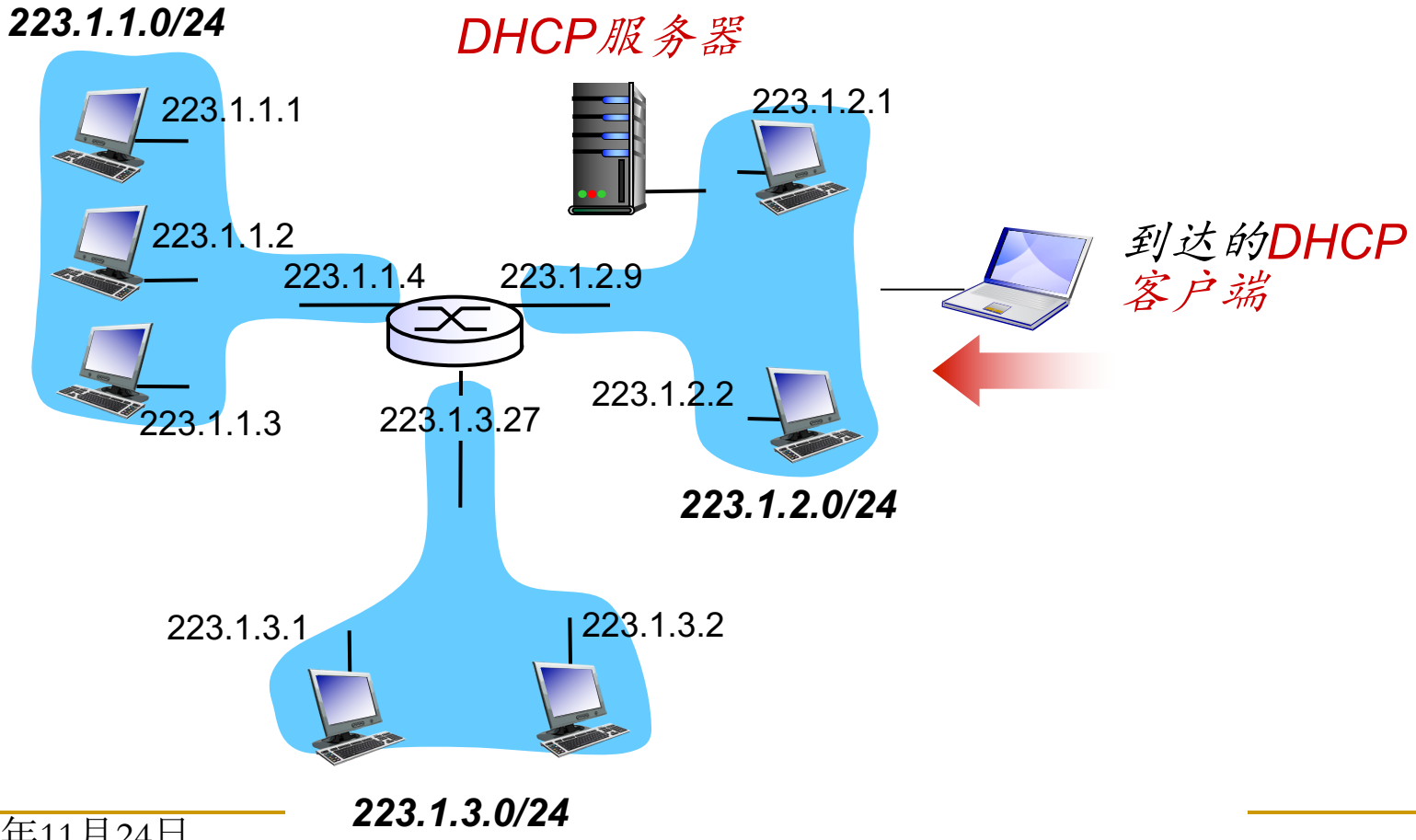
4.4 网际协议:因特网中的转发和编址

Q: 主机如何获得IP地址? (主机部分)

- 手工配置： 系统管理员手工为一台主机配置IP地址
  - Windows: 控制面板->网络->配置->tcp/ip->属性
  - UNIX: /etc/rc.config
- DHCP: Dynamic Host Configuration Protocol 动态主机配置协议:从服务器上动态获取IP地址
  - “即插即用协议”

4.4 网际协议:因特网中的转发和编址

- DHCP工作原理示例

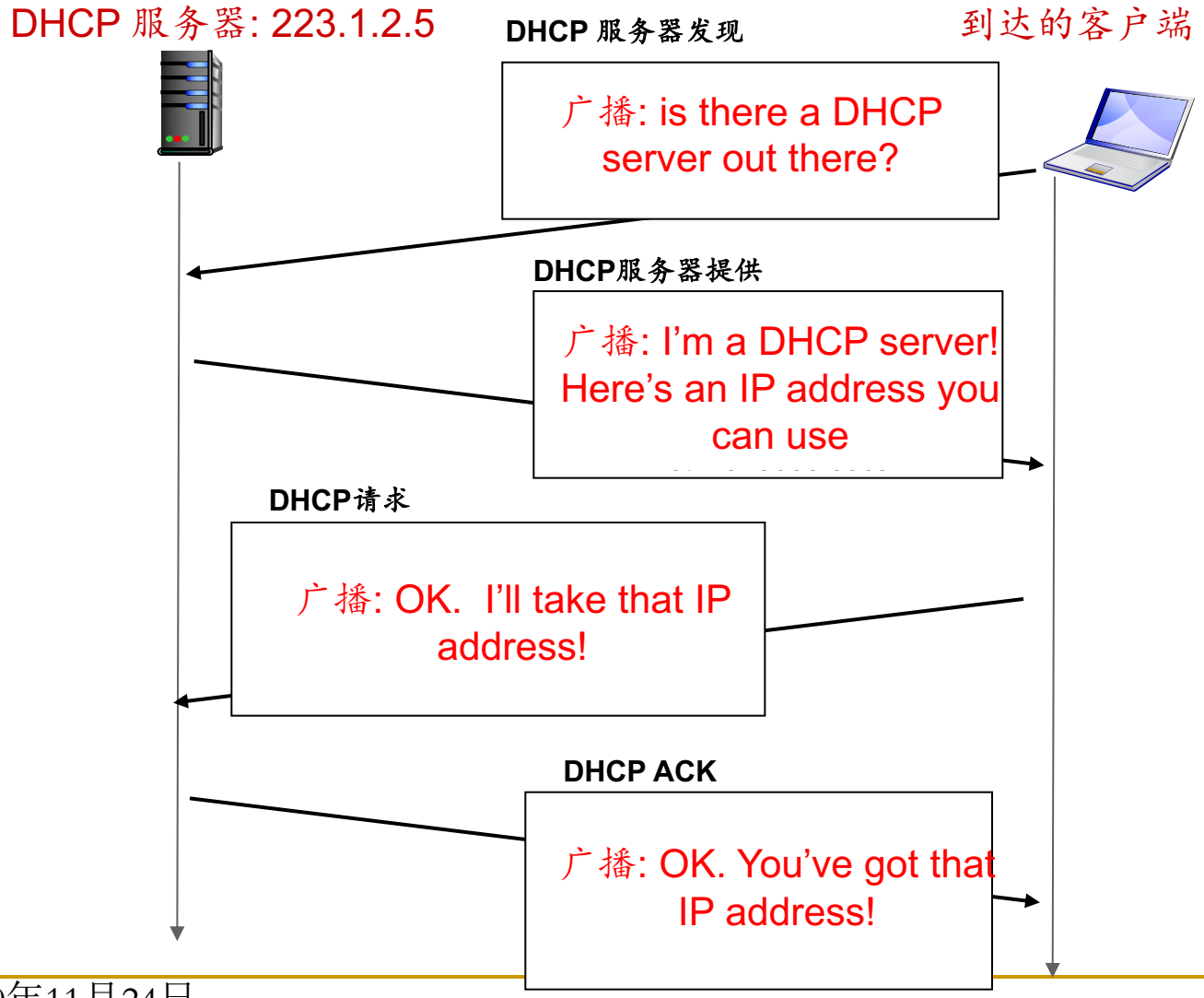


4.4 网际协议:因特网中的转发和编址

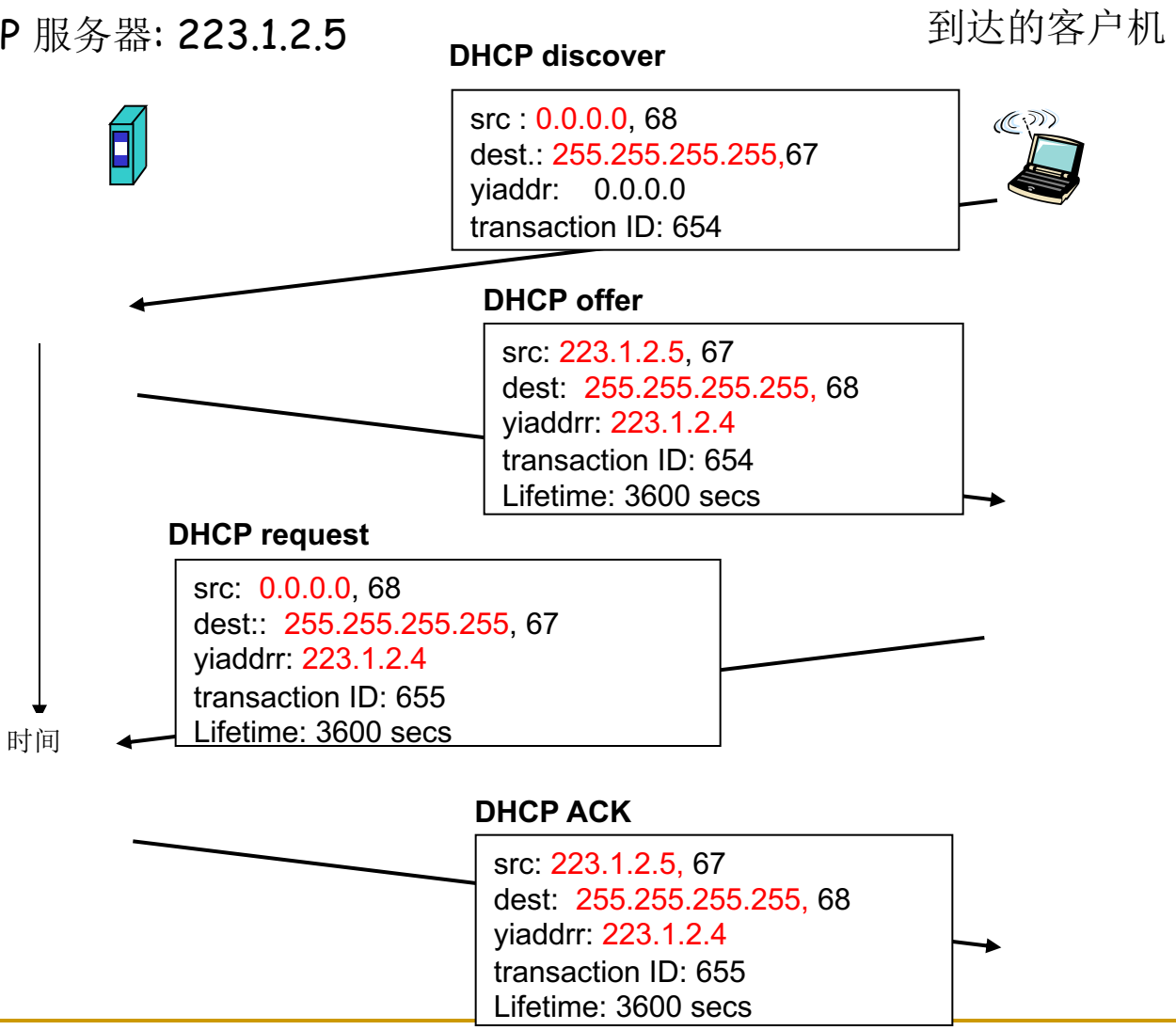
DHCP：动态主机配置协议

- 目标：允许主机加入网络时动态地从网络服务器处获取IP地址(还有默认网关和Local DNS)
  - 允许地址重用
  - 支持希望加入网络的移动用户
- DHCP协议的步骤：
  - 主机广播 “DHCP discover” 报文
  - DHCP服务器响应“DHCP offer” 报文
  - 主机请求IP地址： “DHCP request” 报文
  - DHCP 服务器确认： “DHCP ack” 报文

4.4 网际协议:因特网中的转发和编址



4.4 网际协议:因特网中的转发和编址



4.4 网际协议:因特网中的转发和编址

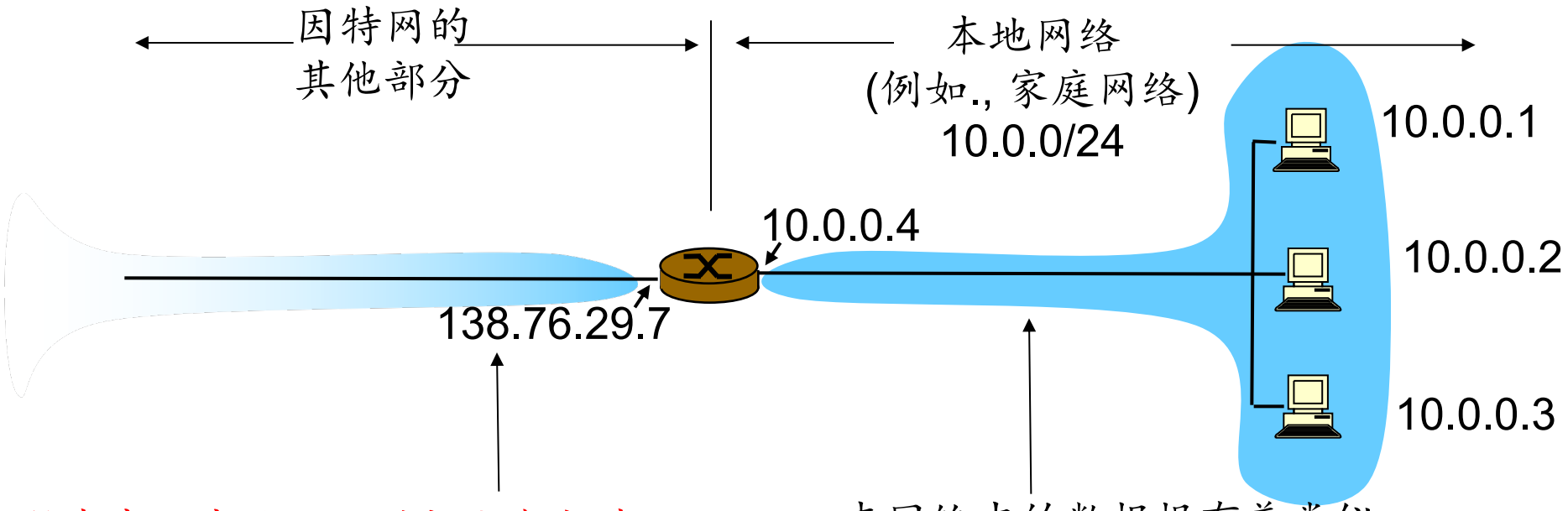
- DHCP协议并不仅仅只能获取IP地址
  - 网关地址
  - DNS地址
  - 子网掩码

4.4 网际协议:因特网中的转发和编址

- **网络地址转换NAT(Network Address Translation)**
  - 动机：本地网络只要使用一个IP地址就可以和外部网络相连 ：
    - 不需要从 ISP处获得大批IP地址：所有设备可以使用同一个 IP地址
    - 可以在不通知外部网络的情况下改变内网主机的IP地址
    - 即使改变了ISP也无须改变内网主机的IP地址
    - 内网主机对外网主机而言是不可见的、不可寻址的。(这也算是一项安全措施).

4.4 网际协议:因特网中的转发和编址

- **NAT(Network Address Translation)**



所有离开本地网络的报文都拥有同一个源IP地址: 138.76.29.7,以及不同的源端口号

本网络中的数据报有着类似10.0.0/24的源或目的IP地址

4.4 网际协议:因特网中的转发和编址

- 实现
  - 发送数据报: 将每个外出报文的源IP地址,端口号替换为NAT IP地址以及新的端口号

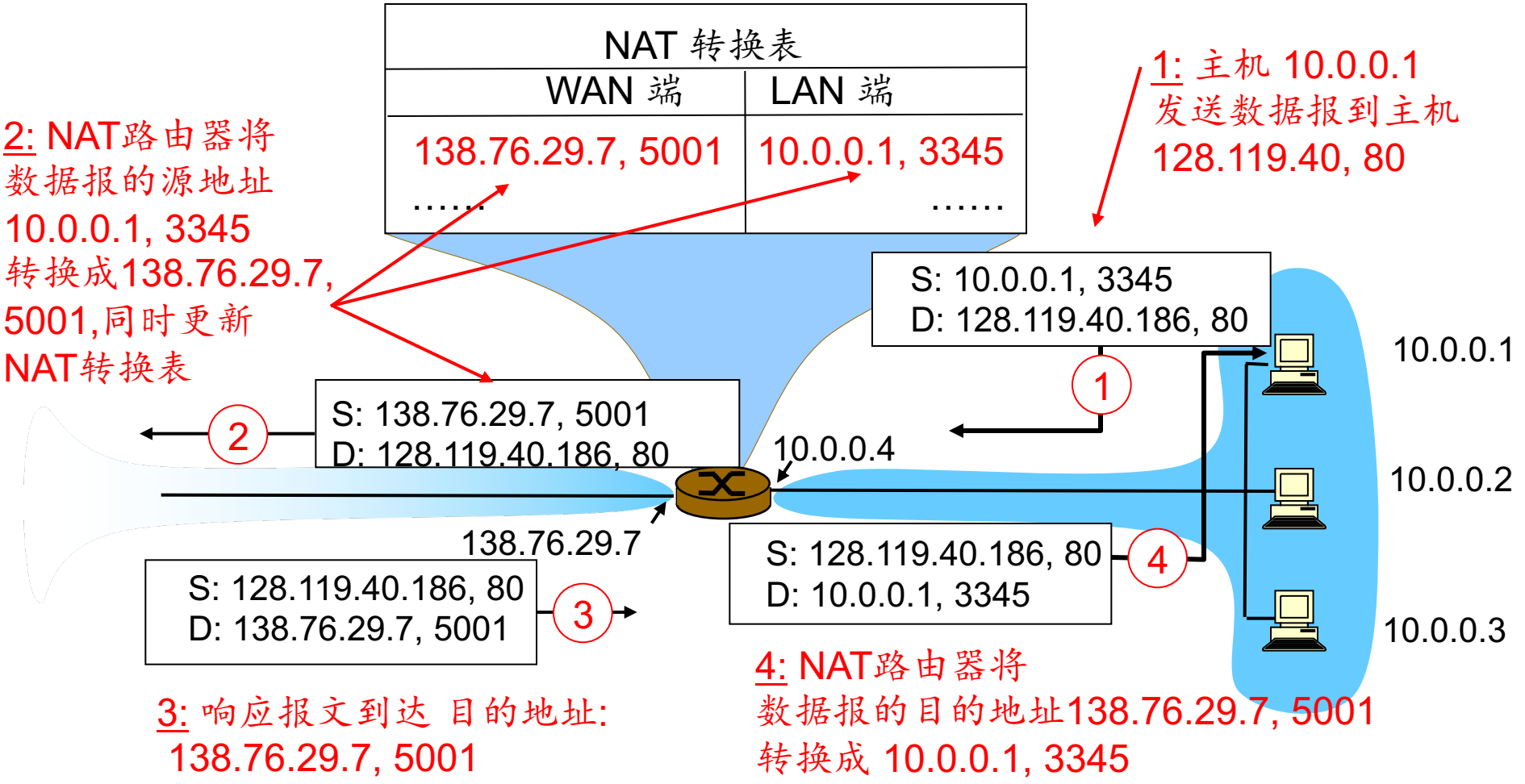
    . . . 远程客户机/服务器将以NAT IP 地址以及新的端口号做为目的地址进行响应.

  - 记住每一个地址转换对 (在 NAT 转换表中)， 即

    源IP地址,端口号→ NAT IP 地址,新的端口

  - 接收数据报:根据NAT转换表将每个进入报文的NAT IP地址,端口号替换为相应的源IP地址以及端口号

4.4 网际协议:因特网中的转发和编址



4.4 网际协议:因特网中的转发和编址

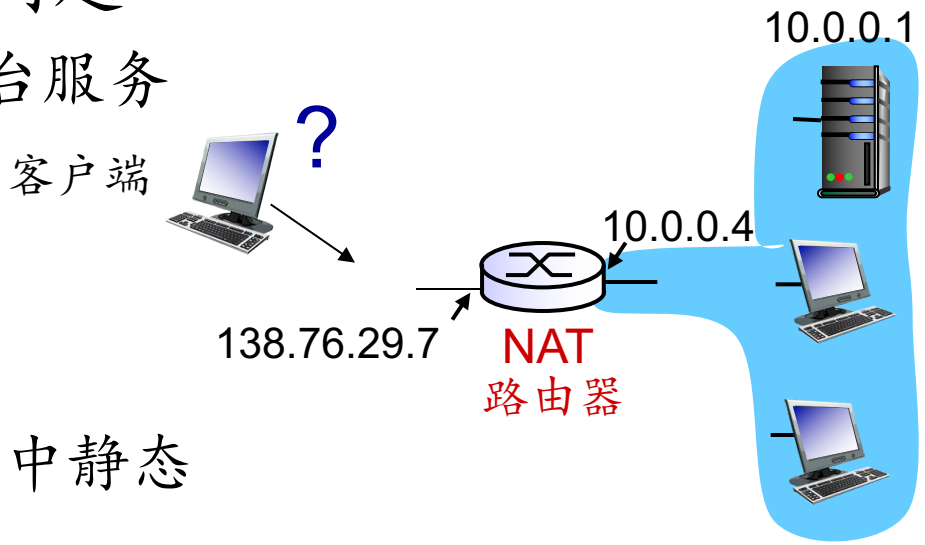
- 两类地址
  - 本地地址
    - 10/8
    - 172.16/12
    - 192.168/16
  - 全球地址
- 三种地址转换方式
  - 静态NAT：一个本地地址对应一个全球地址
  - 动态NAT：一个全球地址对应多个本地地址
  - 端口NAT：一个本地地址的端口对应到一个全球地址的端口

4.4 网际协议:因特网中的转发和编址

- NAT使用中的争议
  - 端口号是用于进程编址的，而不是用于主机编址的
  - 路由器仅应当处理高达第三层的分组
  - NAT协议违反了端到端原则，即主机彼此应相互直接对话，节点不应介入修改IP地址和端口号
  - 应使用IPv6来解决IP地址短缺问题

4.4 网际协议:因特网中的转发和编址

- NAT转换中存在的一个问题
  - 客户端希望与10.0.0.1这台服务器通信



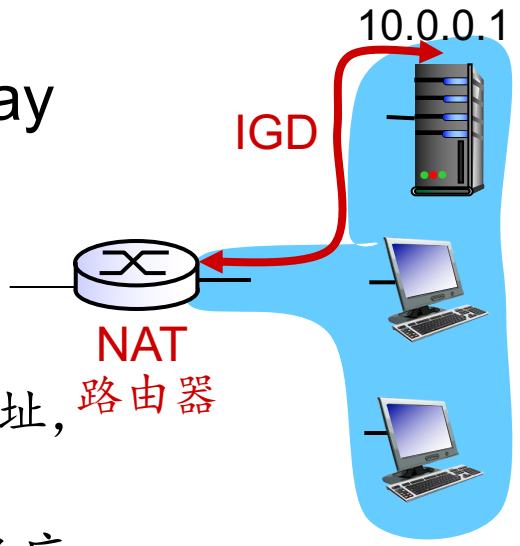
- 解决方案1
  - 采用端口NAT，在路由器中静态的为服务器配置一条记录

如(138.76.29.7,80) 总是指向(10.0.0.1,80)

4.4 网际协议:因特网中的转发和编址

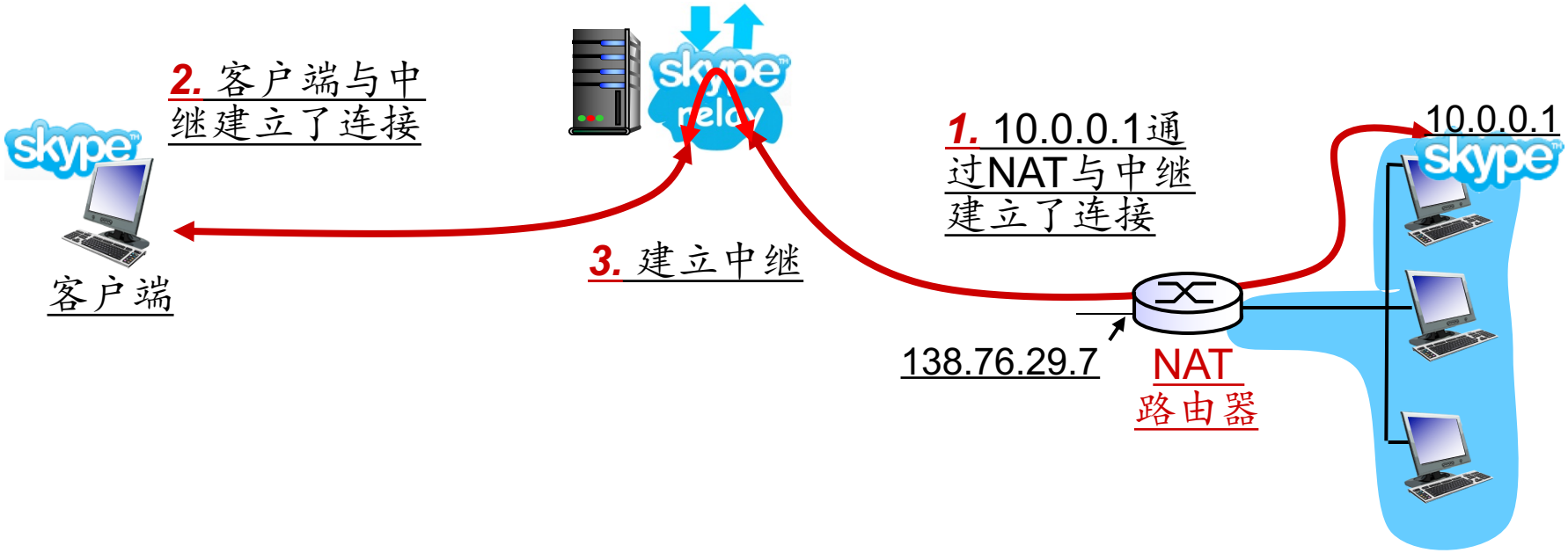
- 解决方案2——通用即插即用UPnP（Universal Plug and Play）
  - 内部主机通过IGD（Internet Gateway Device）协议
    - 了解公共IP地址
    - 向路由器注册/移除映射记录

    （内部IP地址，内部端口号）→（公共IP地址，公共端口号）
  - 内部主机通过某种渠道向外部应用程序公开（公共IP地址，公共端口号）
  - 适用于P2P应用



4.4 网际协议:因特网中的转发和编址

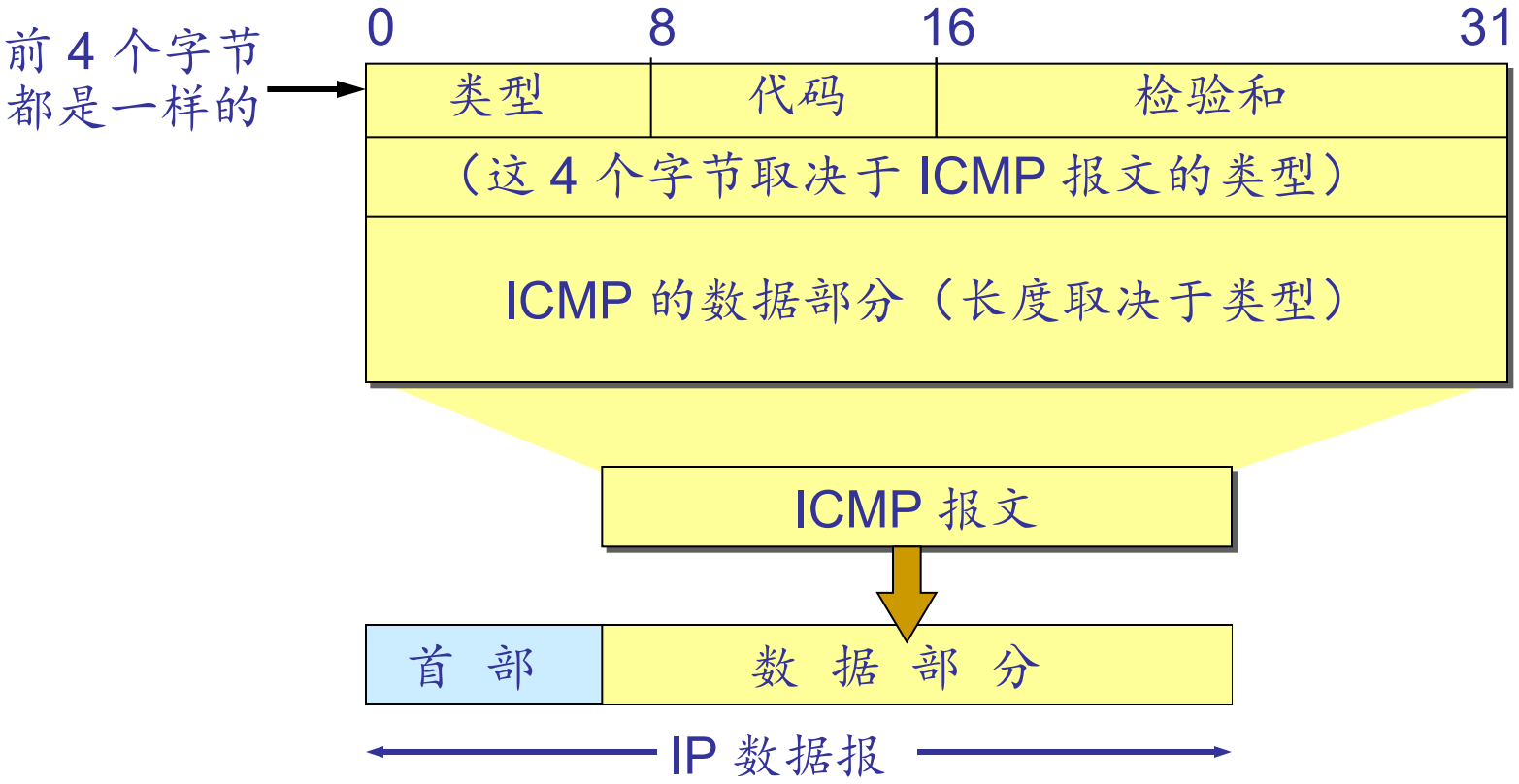
- 解决方案3——中继（用于Skype）



4.4 网际协议:因特网中的转发和编址

- **ICMP：因特网控制报文协议**
  - 用于主机、路由器、网关之间交换网络层信息
    - 错误报告: 如主机、网络、端口、协议不可达等。
    - 回声请求/回答 (用于**ping**应用程序)
  - 从体系结构而言，位于**IP**层之上：
    - **ICMP** 报文封装在**IP**分组中
  - ICMP 消息: 包括一个类型字段和一个编码字段
  - ICMP 报文的种类有两种，即 ICMP 差错报告报文和 ICMP 询问报文

ICMP 报文的格式


前 4 个字节
都是一样的
0
8
16
31
类型
代码
检验和
（这 4 个字节取决于 ICMP 报文的类型）
ICMP 的数据部分（长度取决于类型）
ICMP 报文
首 部
数 据 部 分
IP 数据报

4.4 网际协议:因特网中的转发和编址

类型	代码	描述
0	0	回声回答（对Ping的回答）
3	1	目的主机不可达
3	2	目的协议不可达
3	3	目的端口不可达
3	6	目的网络未知
3	7	目的主机未知
4	0	源抑制 (拥塞控制，未用)
8	0	回声请求 (ping)
9	0	路由器通告
10	0	路由器发现
11	0	TTL 过期
12	0	IP 首部错误

Traceroute和 ICMP

- 源主机向目的主机发送一批UDP报文段
  - 第一个报文的 TTL值为1
  - 第二个报文的 TTL值为2,等
  - 报文段携带一个不可达的UDP端口号
- 当第n个数据报到达第n个路由器时:
  - 路由器丢弃该数据报
  - 同时给源主机发送 ICMP 消息（类型 11，代码 0）
  - 该ICMP消息包含了路由器的名字和IP地址
- 当ICMP消息到达时，源主机计算RTT值
- Traceroute 程序对相同的TTL发送3个一组的分组

停止标准

- UDP报文最终到达了目的主机
- 目的主机返回 “端口不可达” 报文（类型 3,代码 3)
- 源主机收到该ICMP消息时，源主机停止发送UDP报文.

4.4 网际协议:因特网中的转发和编址

- **IPv4面临的问题**
  - 地址空间消耗很快
    - 虽然NAT减少了IP地址的需求量
  - 首部长度不定，中间结点（路由器）需要消耗相当资源用于分组处理
  - 缺少QoS
  - 安全性不够高
- **IPv6**
  - 扩大地址空间- 128位
  - 改革首部格式帮助加速处理/转发-首部长度固定40字节，不支持分割
  - 改革首部实现QoS

4.4 网际协议:因特网中的转发和编址

- **IPv6数据报格式**

<table>
<tr><td>版本
(4bit)</td><td>流量类型
(8)</td><td colspan="3">流标签
(20)</td></tr>
<tr><td colspan="2">有效载荷长度(16)</td><td colspan="2">下一个首部(8)</td><td>跳限制(8)</td></tr>
<tr><td colspan="5">源地址（128比特）</td></tr>
<tr><td colspan="5">目的地址（128比特）</td></tr>
<tr><td colspan="5">数据</td></tr>
</table>

- 无检查和，中间结点无需计算
- 中间结点不再负责分片和重组，由端结点负责
- 首部长度固定，加速中间结点转发速度

4.4 网际协议:因特网中的转发和编址

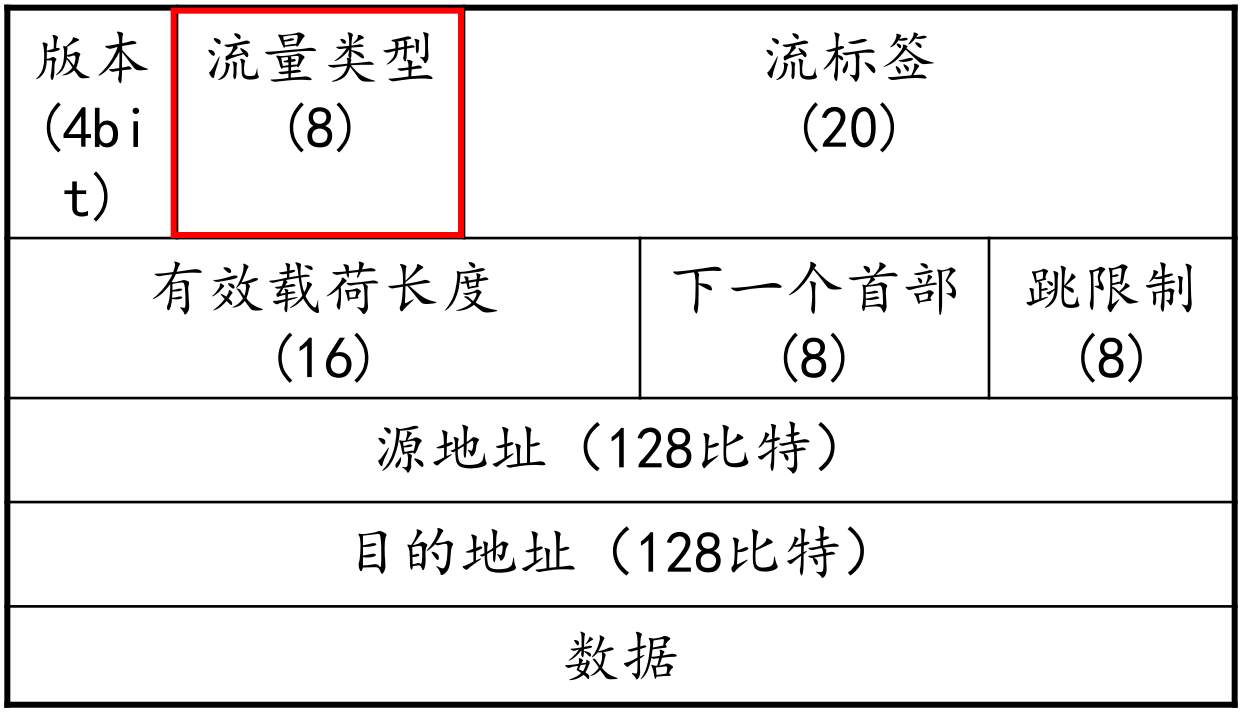
- IPv6数据报格式

<table>
<tr><td>版本
(4bit)</td><td>流量类型
(8)</td><td colspan="2">流标签
(20)</td></tr>
<tr><td colspan="2">有效载荷长度
(16)</td><td>下一个首部(8)</td><td>跳限制(8)</td></tr>
<tr><td colspan="4">源地址（128比特）</td></tr>
<tr><td colspan="4">目的地址（128比特）</td></tr>
<tr><td colspan="4">数据</td></tr>
</table>

版本：IP协议版本号

4.4 网际协议:因特网中的转发和编址

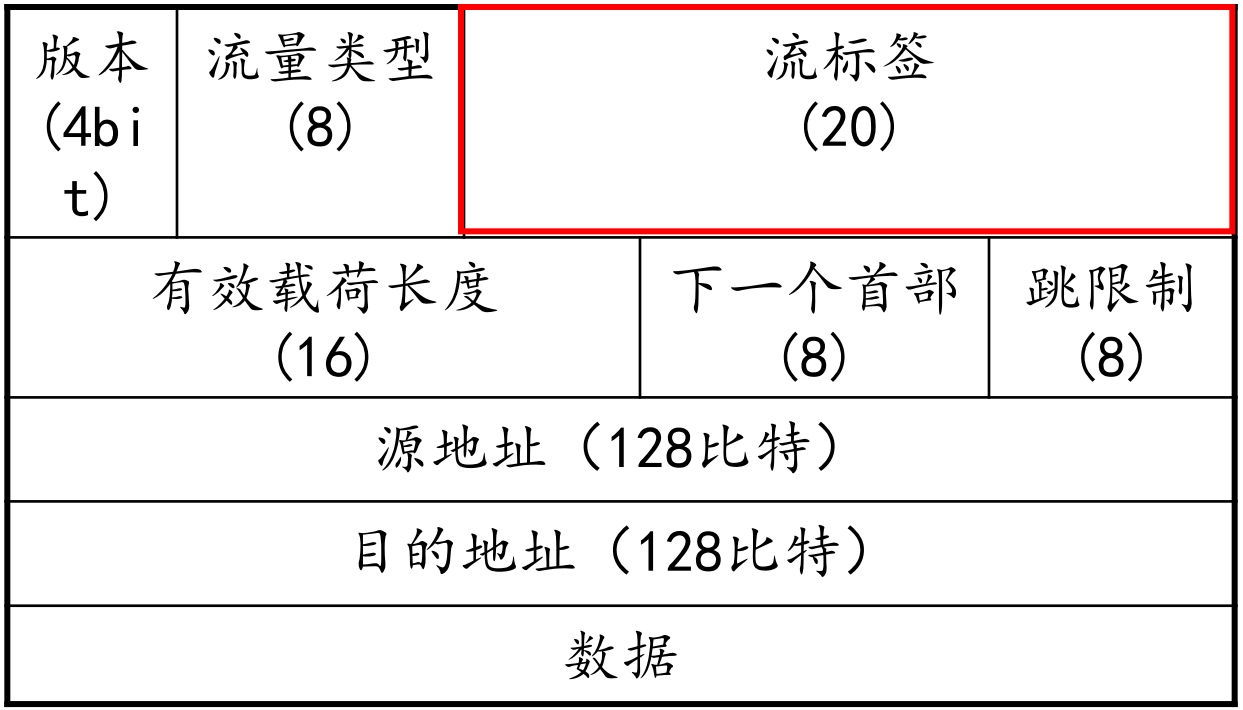
- IPv6数据报格式



流量类型：与IPv4的TOS相似

4.4 网际协议:因特网中的转发和编址

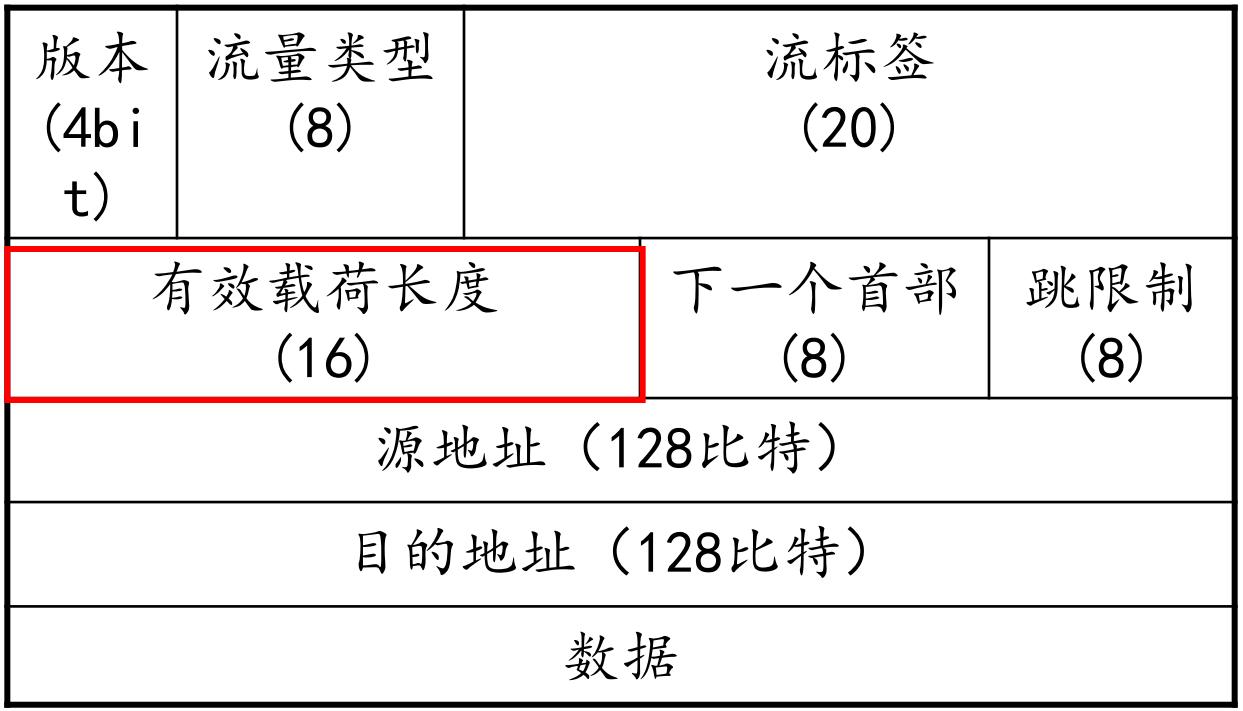
- **IPv6数据报格式**



流标签：给属于特殊流（flow）的分组加上标签。
特殊流指需要特别服务质量的数据流（如实时视频）。
流量类型+流标签用于实现QoS

4.4 网际协议:因特网中的转发和编址

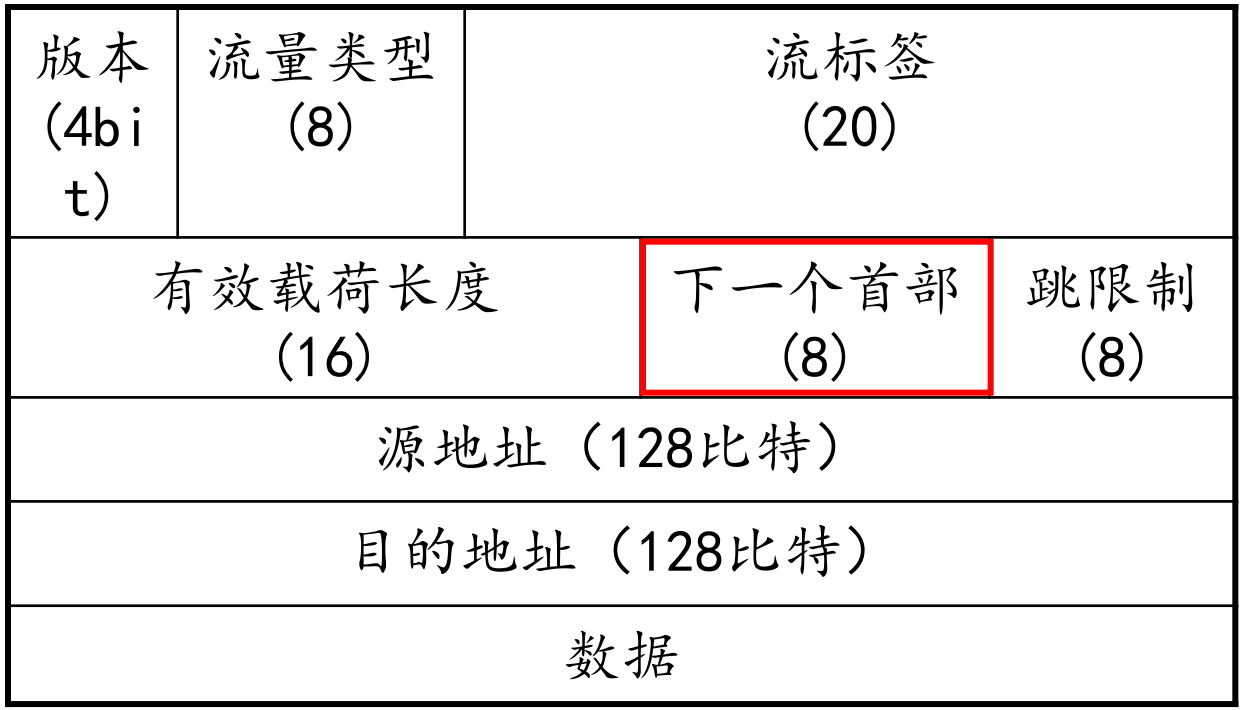
- IPv6数据报格式



有效载荷长度：给出首部后面数据的字节数量。

4.4 网际协议:因特网中的转发和编址

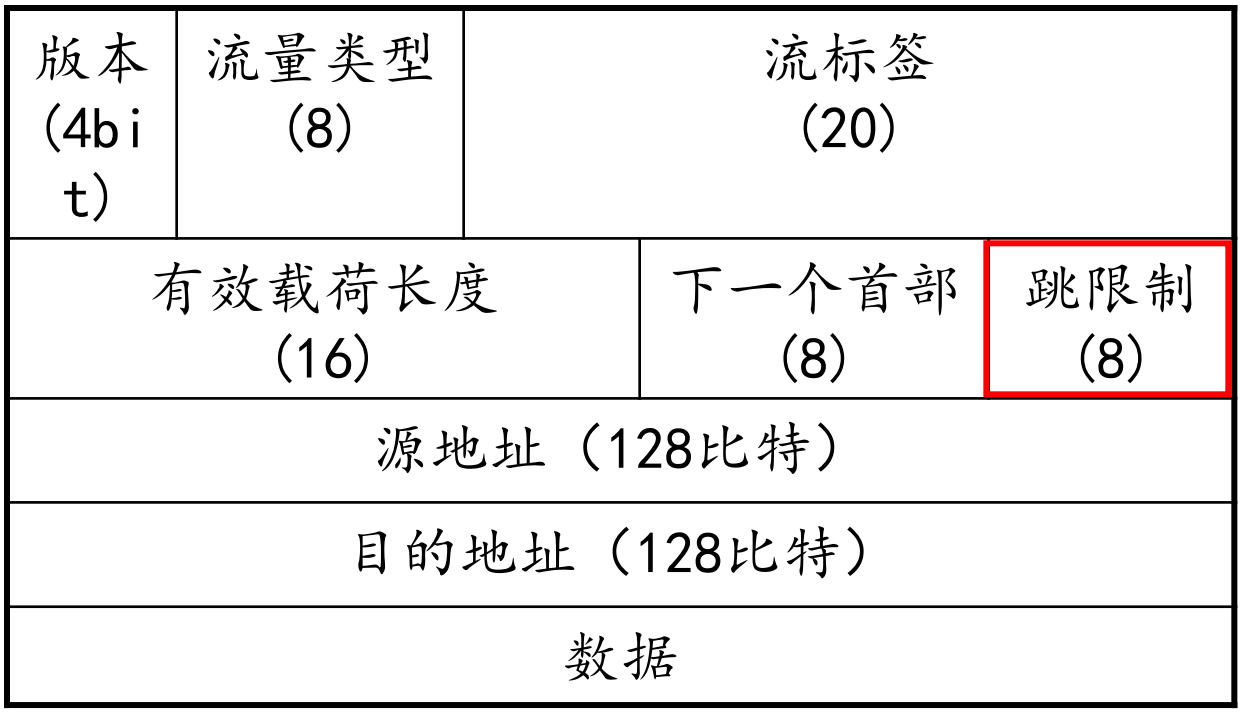
- **IPv6数据报格式**



下一个首部：标识该数据包中内容（数据字段）需要交付给哪个上层协议（如TCP或UDP）。

4.4 网际协议:因特网中的转发和编址

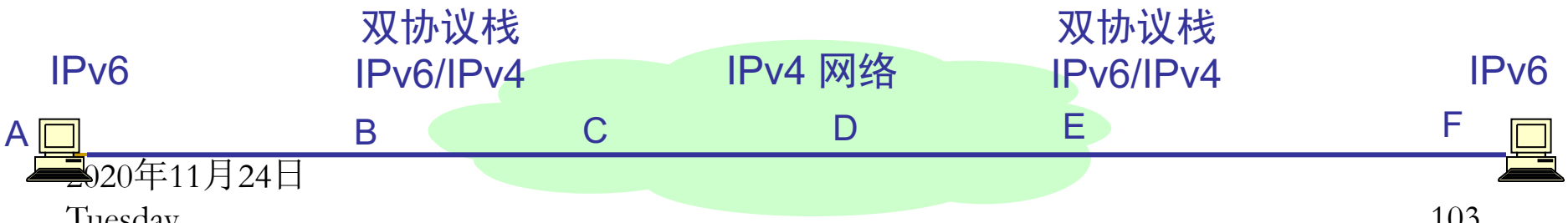
- IPv6数据报格式



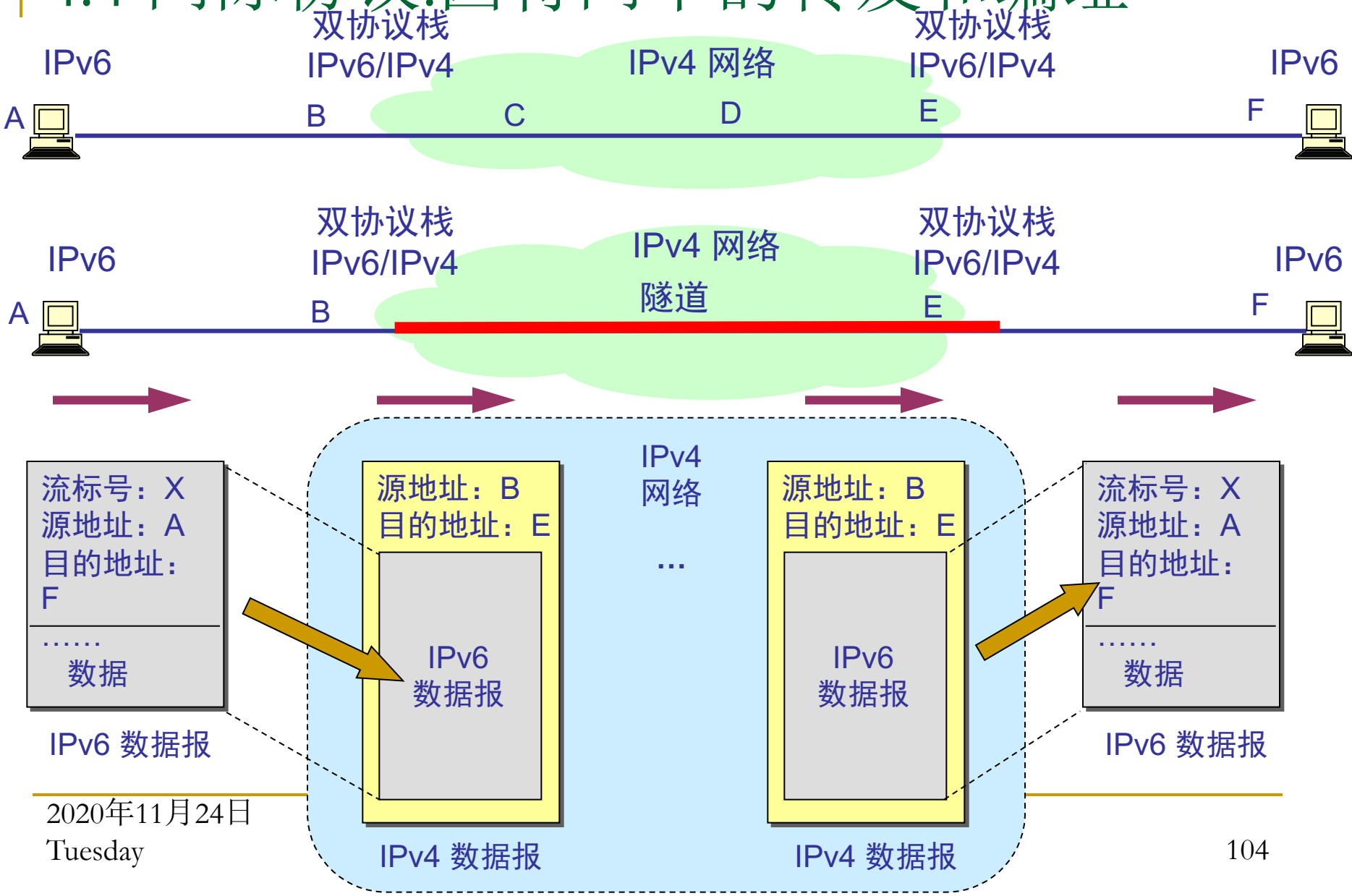
跳限制：转发数据报的每台路由器将该字段内容减一。当跳限制计数到0时，该数据报将被丢弃。同IPV4的TTL

4.4 网际协议:因特网中的转发和编址

- **IPv4到IPv6的迁移**
  - 设立标志日，统一迁移
  - 双栈技术
    - 新加入的设备支持IPv4/IPv6双协议栈
    - 一段链路上，如果源和目标均支持IPv6，则使用IPv6进行通信
    - 如果任一方不支持IPv6，则使用IPv4进行通信
    - 可能会出现信息的丢失



4.4 网际协议:因特网中的转发和编址



4.5 选路算法

- **几个概念**
  - 默认路由器：一台主机“直接”连接到的路由器
  - 源路由器：源主机的默认路由器
  - 目的路由器：目标主机的默认路由器
- **选路算法的目的**
  - 给定一组路由器以及连接路由器的链路，从中找到一条从源路由器到目标路由器“好的”路径
    - “好的”通常指具有最低费用的路径
    - 例外：A和B之间的路径费用很低，但是由于处于两个组织之间，而这两个组织作出的路由策略，不允许通行

4.5 选路算法

抽象图模型

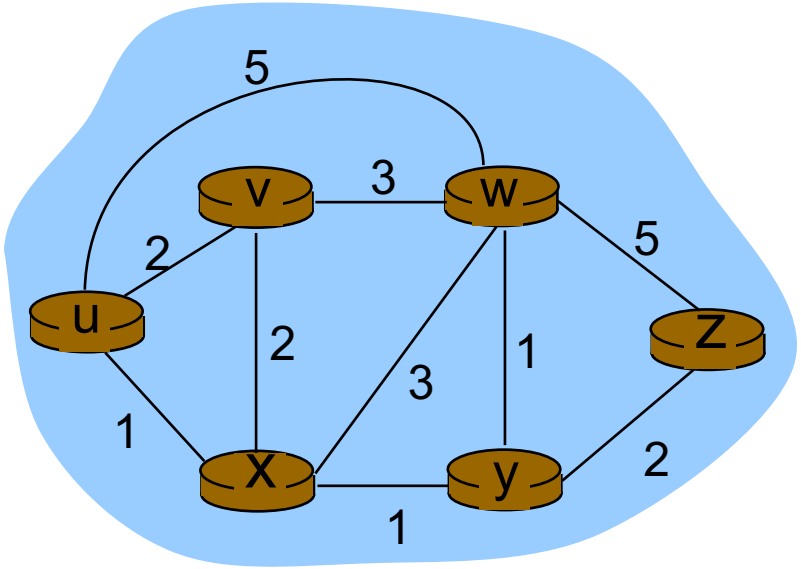


图: G = (N,E)

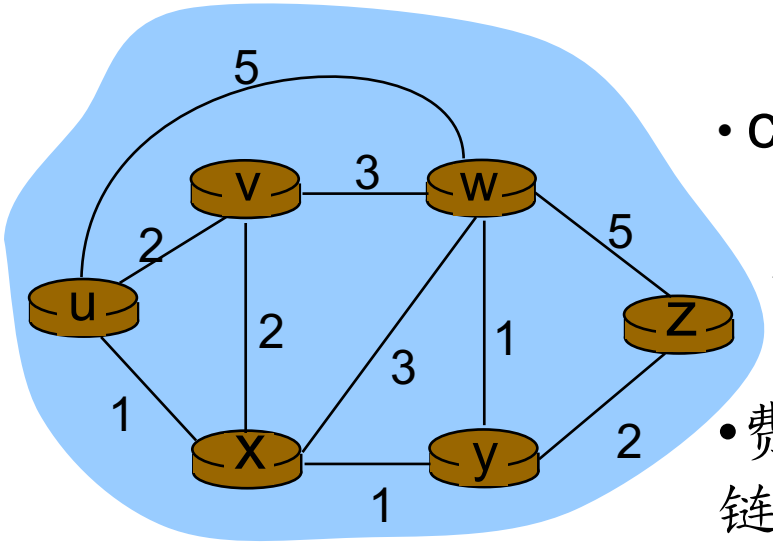
N = 路由器集 = { u, v, w, x, y, z }

E = 链路集 ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

注意: 抽象图模型在其他网络环境中也是很有用的

比如: P2P, 这里N是节点的集合而E是TCP连接的集合

4.5 选路算法



- c(x,x') = 节点X和X' 间边的费用

例如左图中 c(w,z) = 5

- 费用可以是经济的， 但也有可能和链路的带宽以及链路拥塞状况有关

路径费用 $(x_1, x_2, x_3, \ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

问题: 节点U和节点Z之间的最低费用路径是什么？

选路算法: 找出最低费用路径

4.5 选路算法

- **选路算法分类**
  - 根据信息是全局性还是分散式的进行分类
    - 全局选路算法
      - 所有路由器都知道整个网络拓扑图以及链路的费用信息
      - 链路状态算法
    - 分散式选路算法
      - 每个路由器仅有与其相连链路的费用信息
      - 通过迭代计算过程与相邻节点交换信息
      - 距离向量算法

4.5 选路算法

- 根据信息是静态还是动态的进行分类
  - 静态选路算法
    - 随着时间的流逝，路由的变化非常缓
  - 动态选路算法
    - 路由信息可以更快地发生变化
    - 周期性的更新
    - 可以响应拓扑或链路费用的变化

4.5 选路算法

- 根据是否对负载敏感进行分类
  - 负载敏感算法
    - 链路费用会动态地变化以反映出链路的当前状况
  - 负载迟钝算法
    - 链路费用不明显地反映链路的当前状况

4.5 选路算法

链路状态选路算法

Dijkstra's（迪克斯特拉）算法

- 所有节点都知道网络拓扑和链路费用
  - 通过链路状态广播获得信息
  - 所有节点具有该网络的同一个完整的视图
- 计算从某节点到网络中所有其他节点的最低费用
  - 为该节点提供转发表
- 迭代: 经算法的K次迭代后，可知道到K个目的节点的最低费用路径

符号定义:

- c(x,y): 从节点x到节点y的链路费用; 如果x和y不是直连的，则 c(x,y) = ∞
- D(v): 随着算法进行本次迭代，从源节点到目的v的最低费用路径的费用
- p(v): 从源节点到v沿着当前最低费用路径的前一节点
- N': 节点子集。v在N'中，如果从源节点到v的最低费用路径已知

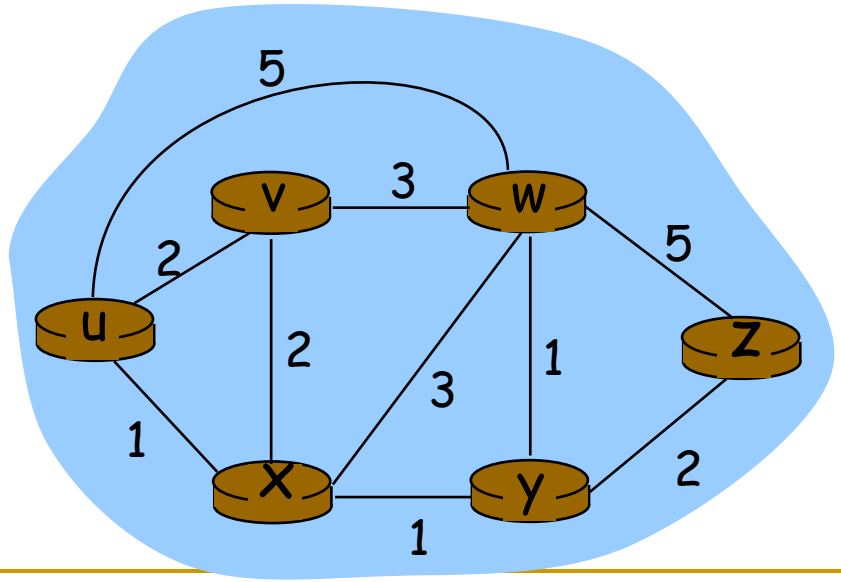
4.5 选路算法

```
1  Initialization:
2    N' = {u}
3    for all nodes v
4      if v adjacent to u
5          then D(v) = c(u,v)
6      else D(v) = ∞
7
8   Loop
9     find w not in N' such that D(w) is a minimum
10    add w to N'
11    update D(v) for all v adjacent to w and not in N' :
12       D(v) = min( D(v), D(w) + c(w,v) )
13    /* new cost to v is either old cost to v or known
14     shortest path cost to w plus cost from w to v */
15  until all nodes in N'
```

更新到所有w的邻居节点v（不在N’中）的D(v)

4.5 选路算法

步骤	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



p(v)：从源节点u到目的节点v沿着当前最低费用路径的前一节点

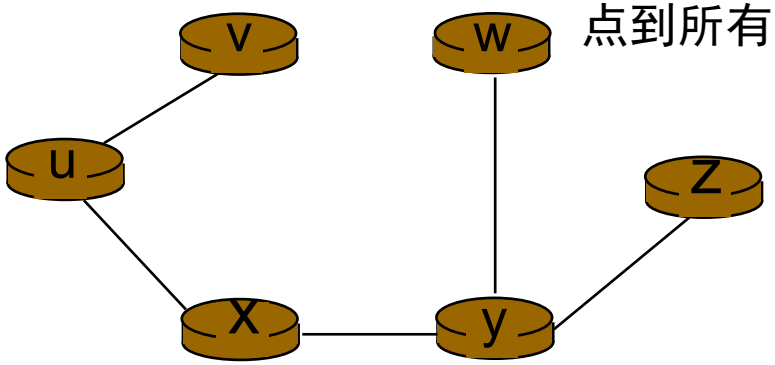
D(v)：随着算法的本次迭代，从源节点u到目的v的最低费用

N’：节点子集。v在N’中，如果从源节点到u的v的最低费用已经知道

4.5 选路算法

在LS算法结束时，对每个节点，我们都得到从源节点沿着它的最低费用路径的前一节点，对每个前一节点，我们又有它的前一节点。以此方式我们可以构建从源节点到所有目的节点的完整路径。

u出发的最短路径树:



u的转发表:

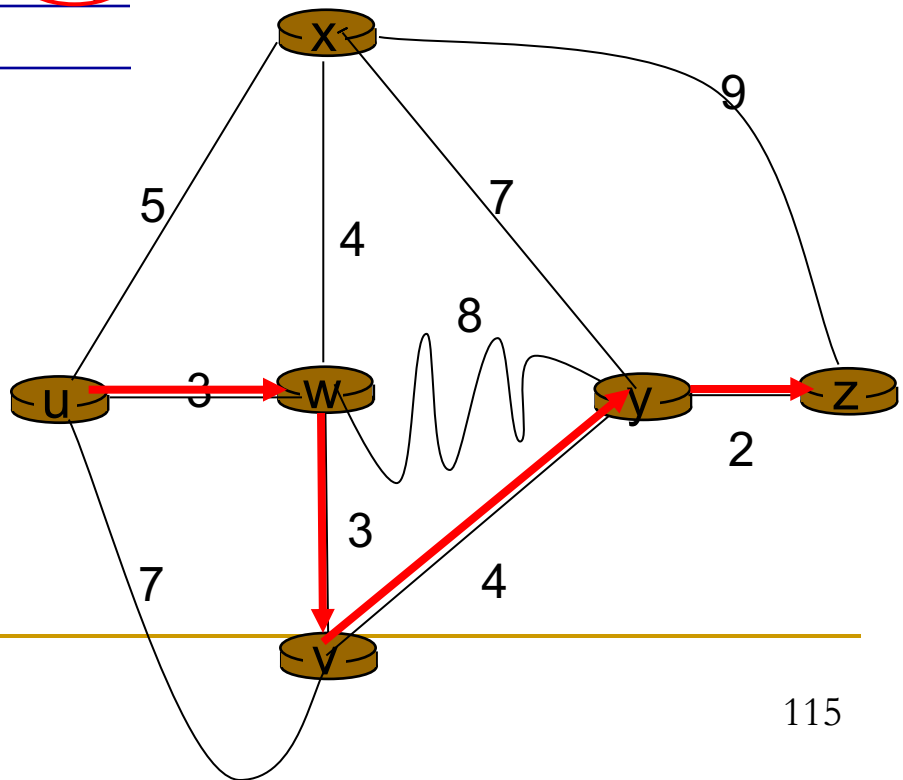
目标	链路
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

4.5 选路算法

步骤	N'	D(v) p(v)	D(w) p(w)	D(x) p(x)	D(y) p(y)	D(z) p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					

注意:

- 通过跟踪前一跳节点可以构造最短路径树

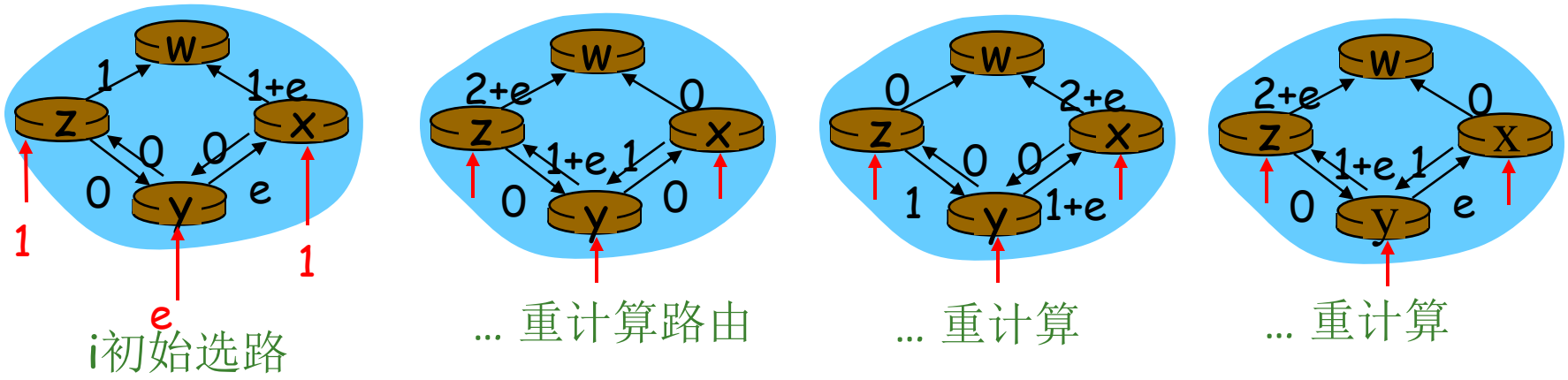


4.5 选路算法

- **Dijkstra’s（迪克斯特拉）算法的复杂性**
  - 对于第一次迭代：需要搜索所有的n个节点以确定出节点w,w不在N’中且具有最低费用
  - 第二次迭代搜索n-1，第三次n-2
  - 在所有迭代中需要搜索的节点总数为n(n+1)/2，所以链路状态算法在最差情况下复杂性为 $O(n^2)$
  - 该算法可采用一种更复杂的实现，使用一种叫做堆的数据结构，其计算复杂性为 O(nlogn)

4.5 选路算法

- 利用链路状态算法选路时可能产生震荡



链路的费用=链路上承载的负载，因此c(u, v)不等于c(v, u)

开始：z向w发送1单元的流量，x向w发送1单元流量，y向w发送e单元流量

当LS算法再次运行后，x、y到w的最低费用路径是顺时针的，导致流量顺时针流动。再次运行LS算法，又导致流量逆时针流动…

4.5 选路算法

- **解决的方案**
  - 强制链路费用不依赖于所承载的流量
    - 无法解决高拥塞的问题，不可接受
  - 确保所有的路由器不同时运行LS算法
    - 因特网上的路由器能够自同步
    - 随机化路由器发送链路通告的时间

4.5 选路算法

- **因特网中的链路状态选路——OSPF协议**
  - “开放”表明 OSPF 协议不是受某一家厂商控制，而是公开发表的。
  - “最短路径优先”是因为使用了 Dijkstra 提出的最短路径算法SPF
  - OSPF 只是一个协议的名字，它并不表示其他的路由选择协议不是“最短路径优先”。
  - 是分布式的链路状态协议。

三个要点

- 向本自治系统中所有路由器发送信息，使用的方法是洪泛法
- 发送的信息就是与本路由器相邻的所有路由器的链路状态
- 只要当链路状态发生变化时，路由器就用洪泛法向所有路由器发送此信息
- 即使链路状态没变化，也要周期性地发送链路状态信息

链路状态数据库

- 由于各路由器之间频繁地交换链路状态信息，因此所有的路由器最终都能建立一个链路状态数据库
- 这个数据库实际上就是全网的拓扑结构图，它在全网范围内是一致的
- OSPF 的链路状态数据库能较快地进行更新，使各个路由器能及时更新其路由表。OSPF 的更新过程收敛得快是其重要优点

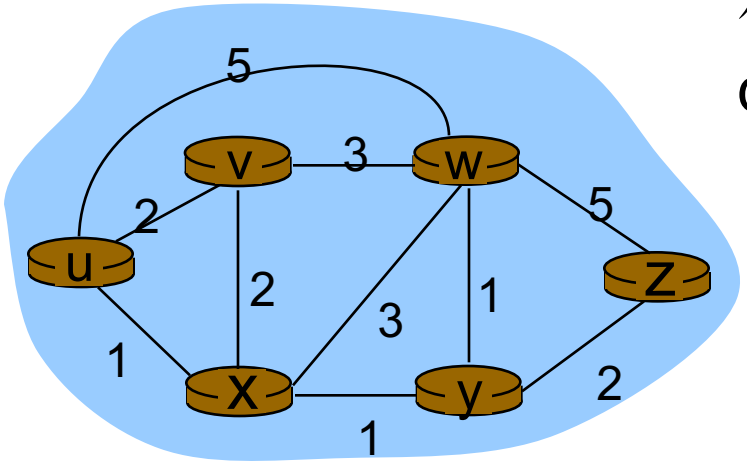
4.5 选路算法

- **OSPF协议的特点**
  - 不强制如何设置链路权值的策略，但提供对给定链路权值集合确定最低费用路径的机制
  - 即使链路状态未发生变化，每30分钟广播一次链路状态
  - 链路状态以OSPF通告的形式封装在OSPF报文中，由IP分组承载（协议号：89）
  - OSPF路由器之间的交换都是经过鉴别的（简单的、MD5的），以确认OSPF通告的真实性，防止伪造和篡改
  - OSPF通告都是有序列号的，以防止重放攻击
  - OSPF中支持多条具有相同费用的路径
  - OSPF支持多播选路和层次路由
  - OSPF使用IP数据报传输报文，是一个网络层协议

4.5 选路算法

- **距离向量选路算法**
  - 特点：迭代、分布、自我终止、异步
  - 思想
    - B-F公式：$d_x(y)=\min_v\{c(x,v)+d_v(y)\}$
      - $d_x(y)$：节点x到y的当前最低费用
      - v是对于x的所有直接邻居
    - 每个路由器中都有一张路由表，包含三个内容：目的网络号、经过的邻居路由器、到目的网络的距离
    - 路由器定期向其邻居路由器传送路由表的拷贝

4.5 选路算法



假设已经知道，

$d_v(z) = 5,\ d_x(z) = 3,\ d_w(z) = 3$

计算B-F公式:

$$d_u(z) = \min \{ c(u,v) + d_v(z),\ c(u,x) + d_x(z),\ c(u,w) + d_w(z) \}$$
$$= \min \{2 + 5,\ 1 + 3,\ 5 + 3\} = 4$$

注意得到最小值即可确定最短路径的下一跳 x→ 转发表

4.5 选路算法

- $D_x(y)$ = x到y的最小费用当前估算值
- 距离向量：$D_x = [D_x(y): y \in N]$
- 每个节点维护下列选路数据：
  - 节点x到每个邻居v的费用： $c(x,v)$
  - 节点x的距离向量：即$D_x = [D_x(y): y \in N]$，包括了x到N中所有目的地最低费用的当前估计值
  - 节点x同时保持它邻居的距离向量
    - 对每个邻居v，x 保持 $D_v = [D_v(y): y \in N]$

4.5 选路算法

基本思想：

- 每个节点周期性地发送它自己的距离向量给邻居
- 当节点x从邻居那里收到收到一个新的距离向量，它将用B-F方程更新自己的距离向量

$$D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\} \quad y \in N$$

- 最终$D_x(y)$ 估算值将收敛到实际的最低费用路径的费用。

4.5 选路算法

迭代、异步：

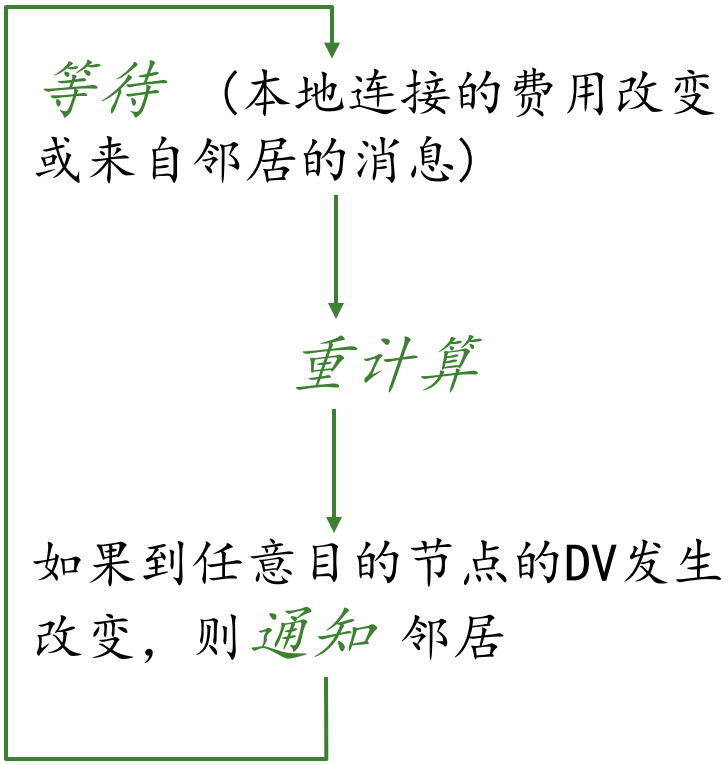
每个本地迭代由下面的事件引起：

- 本地连接的费用改变
- 从邻居而来的DV更新消息

分布的：

- 每个节点都要从一个或多个邻居接收距离向量更新消息
  - 在执行计算后，如果有必要，该节点再将计算结果发回给邻居

每个节点：



$D_x(y) = \min\{c(x,y) + D_y(y),\ c(x,z) + D_z(y)\}$
$= \min\{2+0\ ,\ 7+1\} = 2$

$D_x(z) = \min\{c(x,y) + D_y(z),\ c(x,z) + D_z(z)\}$
$= \min\{2+1\ ,\ 7+0\} = 3$

节点x表

费用到

从	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

从	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

从	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

节点y表

费用到

从	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

从	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

从	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

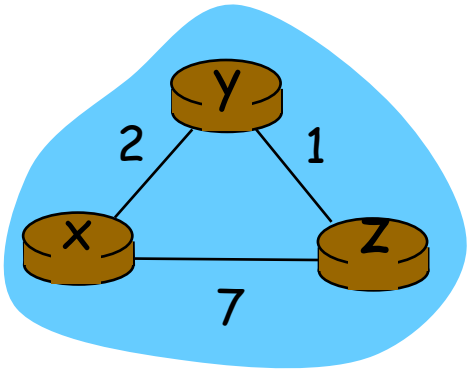
节点z表

费用到

从	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

从	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

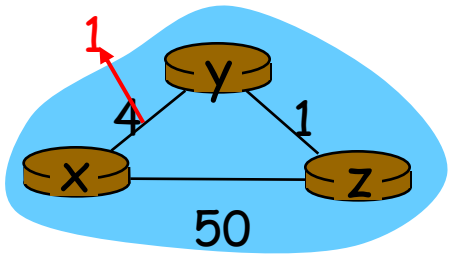
从	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0



时间

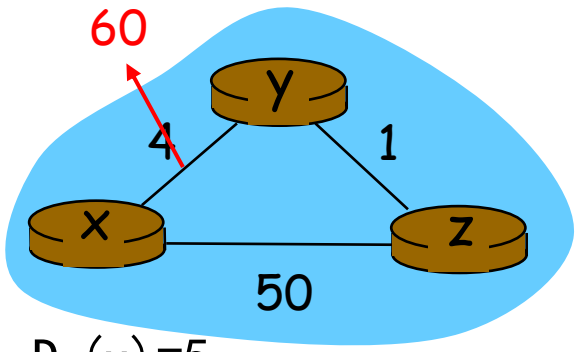
4.5 选路算法

- 链路状态改变时的特点
  - 好消息传的快
    - 在 *t0* 时刻，*y* 检测到链路费用变化，更新自己的距离向量，同时将这个变化通知给它的邻居
    - 在 *t1*时刻，*z* 收到来自 *y* 的更新报文并更新了自己的距离向量表，计算出到x的新的最低费用，并向邻居发送它的新距离向量
    - 在 *t2*时刻，*y* 收到自z的更新并更新其距离向量表，Y的最低费用未变，因此y不发送任何报文给z



该算法二次迭代就收敛，x与y之间的费用降低的好消息迅速传播

4.5 选路算法



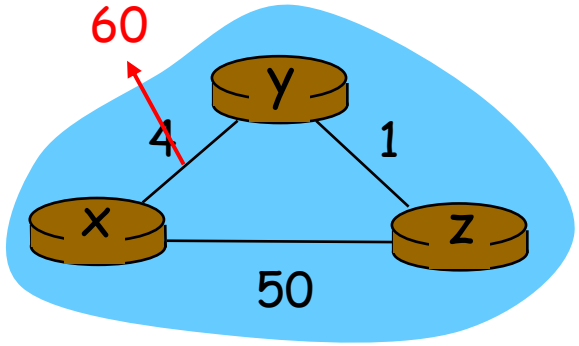
- 坏消息传播的慢
  - 在链路费用变化前，$D_y(x)=4$，$D_y(z)=1$，$D_z(x)=5$
  - t_0时刻，y与x之间的费用增加到60
  - y重新计算$D_y(x)$

    $D_y(x)=\min\{c(y,x)+D_x(x),\ c(y,z)+D_z(x)\}$

    $=\min\{60+0,\ 1+5\}=6$
  - 尽管y到x的最小距离是51，但DV算法不能马上算出
  - y到x的费用从5->6，y将新的向量通知z
  - z收到更新的距离向量后，重新计算$D_z(x)$

    $D_z(x)=\min\{c(z,x)+D_x(x),\ c(z,y)+D_y(x)\}=\{50+0,1+6\}=7$
  - y、z之间的报文需要交换44次，直到z最终算出$D_z(x)=50$
  - 会导致无穷计数问题

4.5 选路算法



- 解决方案——毒性逆转
  - 如果z通过y选路到达目的地x ：
    - z将通告y，它到x的距离是无穷大（所以 y不会通过z到达x）
    - t_0时刻y与x之间的费用增加到60
    - y更新其距离向量，由于$D_z(x)=\infty$，因此新的$D_y(x)=60$
    - z在t_1时刻收到新的$D_y(x)=60$后，马上算出新的$D_z(x)=50$
    - 收到来自z的更新后，y算出新的$D_y(x)=51$
  - 由于涉及到3个或更多节点的环路不能被毒性逆转技术检测到，因此毒性逆转没有解决不可记数问题！

4.5 选路算法

- **链路状态路由选择算法 vs 距离向量路由选择算法**
  - 报文的复杂性
    - LS: n 个节点, E 条链路,需要发送 O(nE)个报文
    - DV: 只在直连的邻居之间交换报文
  - 收敛速度
    - 算法收敛时间依赖于许多因素，因而是可变的
    - LS: 是一个要求O(nE)个报文的$O(n^2)$ 算法
      - 可能有震荡
    - DV: 收敛时间不确定
      - 可能会遇到选路环路
      - 记数到无穷问题

4.5 选路算法

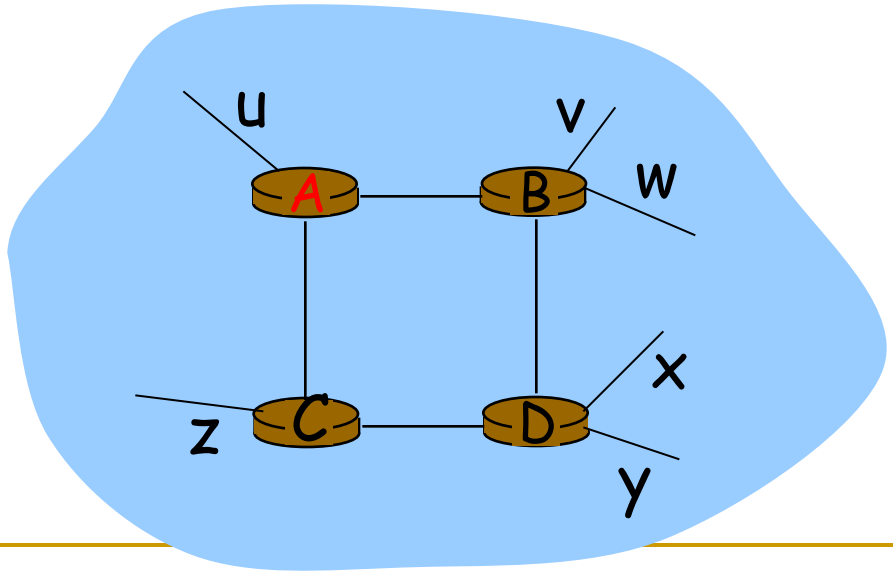
- 健壮性
  - LS:
    - 节点能够向其连接的链路广播不正确费用
    - 每个节点只计算自己的转发表，路由计算在某种程度上是分离的，有一定的健壮性
  - DV
    - 一个节点可向邻居节点通告其不正确的最低费用路径
    - 每个节点的计算都会传递给它的邻居
      - 错误会通过网络进行传播

4.5 选路算法

- **因特网上的距离向量算法——RIP协议**
  - 相邻两点间链路上的费用定义为1，即只考虑源到目标经过多少个路由器，或多少“跳”
  - 一条路径的最大费用限制为15
  - 选路更新消息每30s在邻居之间以RIP响应报文（RIP通告）的形式进行交换
  - 路由器经过180s没有收到来自某个邻居的RIP通告，则认为该邻居已离线，修改选路表，向其它邻居广播
  - RIP是一个运行在UDP上的应用层协议（端口520）

RIP (选路信息协议)

- 距离向量算法
- 包含在支持TCP/IP的1982年UNIX伯克利软件版本（BSD）中
- 距离尺度：以跳数 作为度量，也就是每条链路费用为1(最大值为15跳)。跳(hop)是沿着从源路由器到目的子网的最短路径所经过的子网数量（包括目的子网）

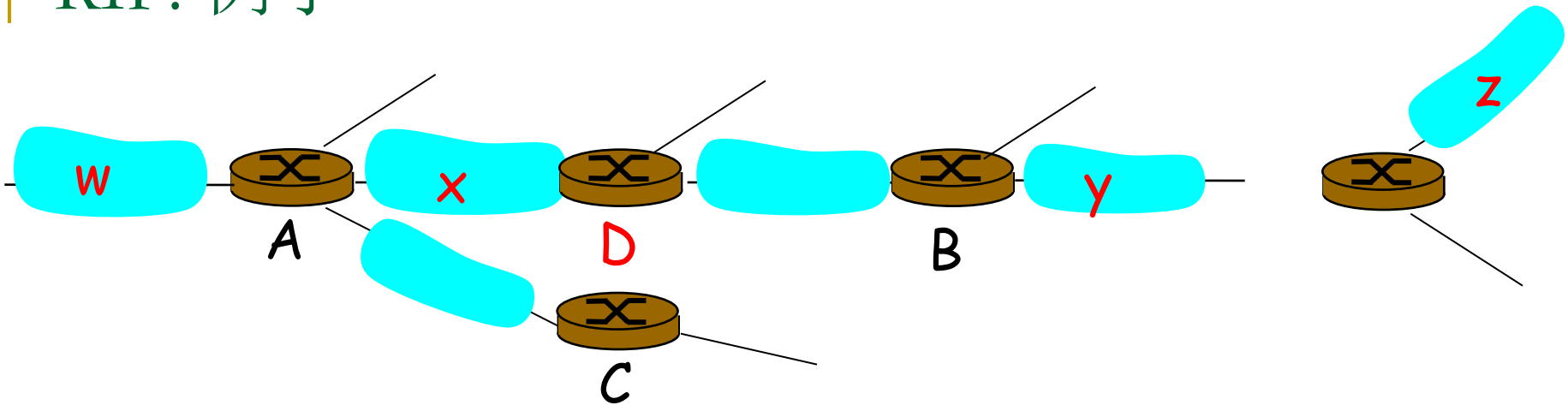


目的地	跳数
u	1
v	2
w	2
x	3
y	3
z	2

RIP 通告

- 更新距离向量：选路更新信息在邻居之间通过RIP响应报文（也称为通告）交换，每30秒相互交换一次
- 通告：通告包含了目的子网列表，下一跳路由器、以及发送方到其中每个子网的距离的信息

RIP: 例子



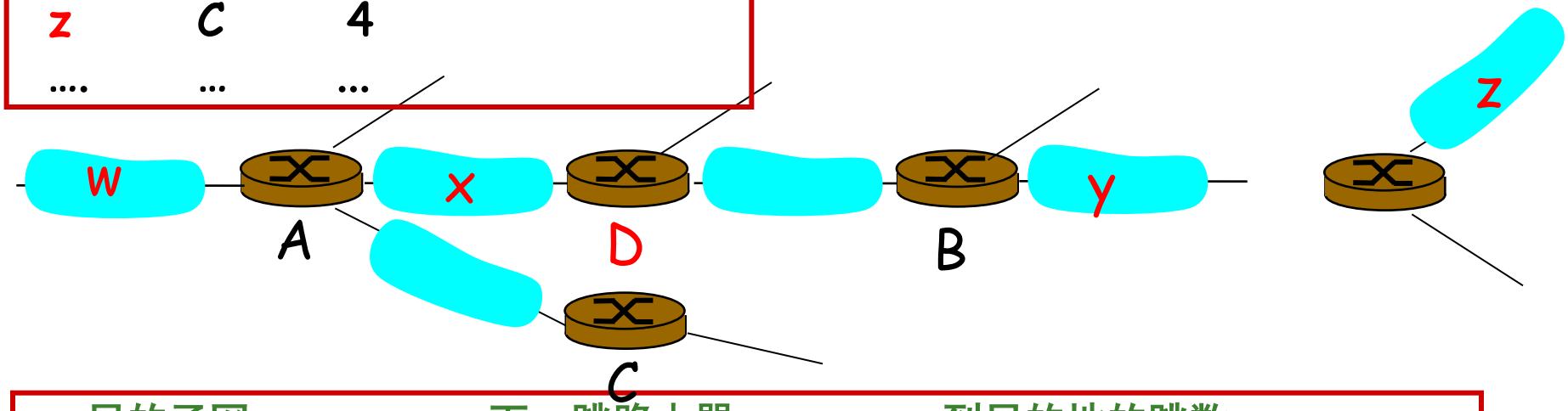
目的子网	下一跳路由器	到目的地的跳数.
w	A	2
y	B	2
z	B	7
x	--	1
....	....	....

收到来自路由器A的通告之前路由器D中的转发表

RIP: 例子

目的子网	下一跳	到目的地的跳数
w	-	-
x	-	-
z	C	4
....	...	...

来自A的通告



目的子网	下一跳路由器	到目的地的跳数.
w	A	2
y	B	2
z	B ~~A~~	7 ~~5~~
x	--	1
....	....	....

收到来自路由器A的通告之后路由器D中的转发表

层次路由

前面讨论的路由算法是非常理想化：

- 所有路由器都是相似的
- 网络是平面的（无层次结构），实际当中不是这样的

4.5 选路算法

- **层次路由**
  - 问题背景
    - 因特网规模过大——数亿个目标网络
      - 路由器无法存储每台主机的选路信息
      - 路由表更新的报文广播将导致无剩余带宽供发送数据使用
    - 管理自治
      - 因特网 = 网络的网络
      - 每个网络管理员可能希望能够按照自己的愿望运行和管理其网络

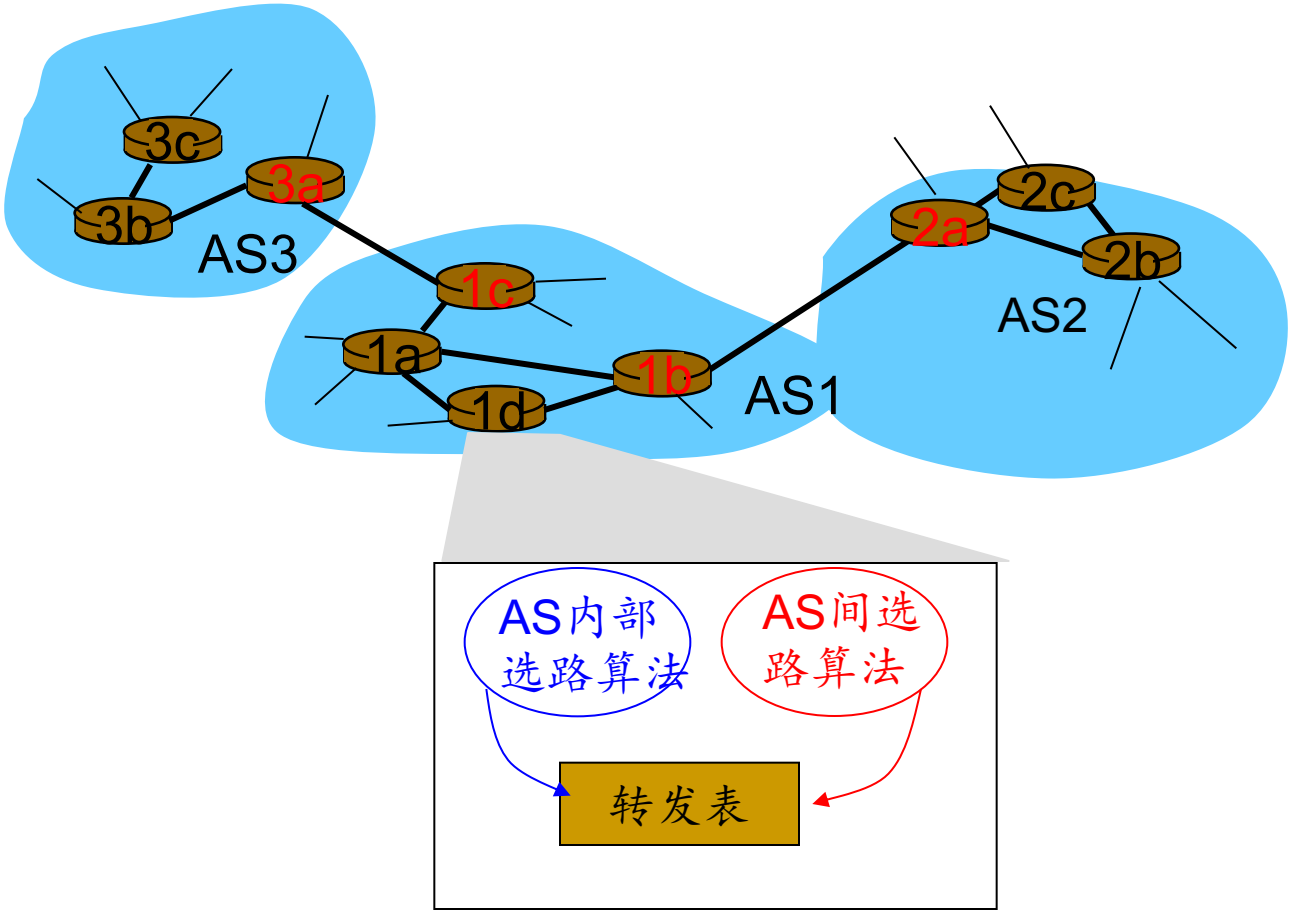
4.5 选路算法

- 解决方法
  - 将路由器聚合到一个区域,“自治系统” (AS)
  - 在相同AS内的路由器可全部运行同样的选路算法
    - 自治系统内部选路协议
      - 内部网关协议 IGP (Interior Gateway Protocol)　目前这类路由选择协议使用得最多，如 RIP 和 OSPF 协议。
    - 在不同AS内的路由器可以运行不同的自治系统内部选路协议
  - 不同AS之间的选路：外部网关协议EGP
    - 边界网关协议BGP

网关路由器:

- 通常位于AS的边缘
- 具有连接到其它AS的链路

4.5 选路算法



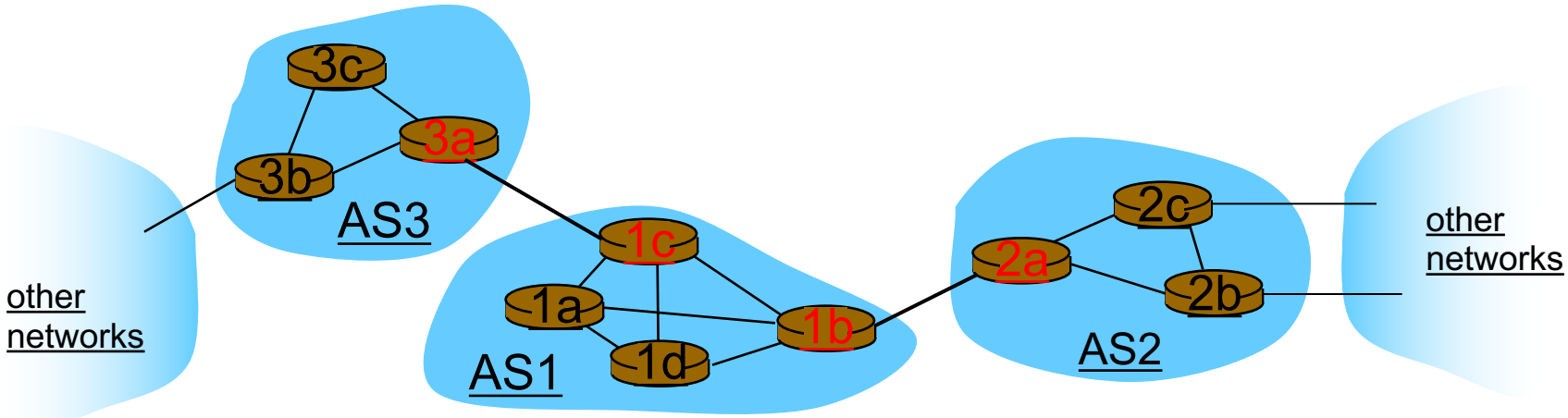
- **转发表是由AS内部选路算法和AS间选路算法共同决定的**
  - AS内部选路算法为内部目的地址设置转发表信息
  - AS内部选路算法和AS间选路算法共同为外部目的地址设置转发表信息

4.5 选路算法

- 自治系统间路由器的任务
  - 假设AS1中的路由器收到一个数据报，其目的地址在AS1之外
    - 该路由器应该将该数据报转发到某一个网关路由器，但是是哪一个呢？

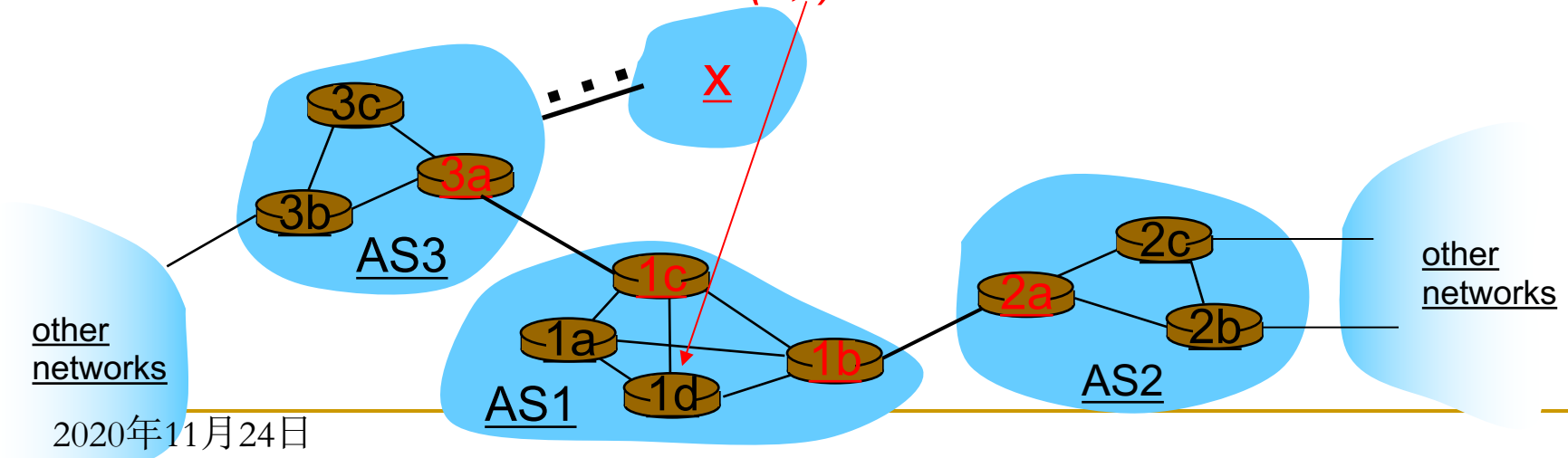
AS1需要：

1. 需要知道通过AS2可以到达哪些目的地，通过AS3可以到达哪些目的地
2. 将这些可达性信息向AS1中的所有路由器传播



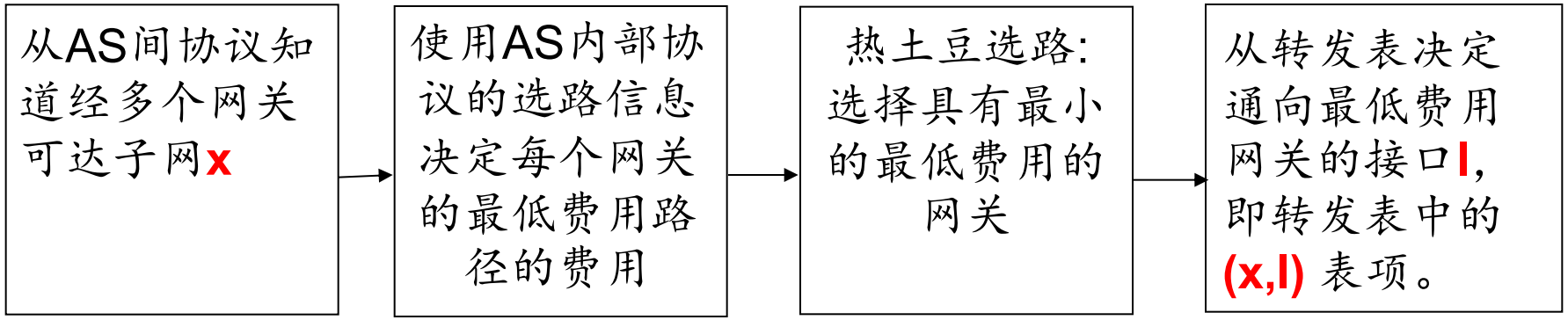
4.5 选路算法

- 场景1：从源到目标仅有一条路可选
  - 假设AS1从AS间选路协议知道子网 *x* 经AS3 (网关1c) 可达，但是通过AS2不可达。
  - AS1向它的所有路由器广播该可达信息。
  - 路由器1d 知道，它的接口 *I* 在到路由器1c的最低费用路径上。
  - 从而路由器1d将表项 *(x,I)* 放入*其转发表*。



4.5 选路算法

- 场景2:从源到目标有多条路可选
  - 现在假设AS1知道子网 *x* 可以通过 AS3 *和* AS2到达.
  - 为了配置转发表, 路由器 1d 必须决定通过哪个网关路由器转发报文（1b或1c）。
  - 这也是自治系统间选路协议必须解决的问题!
  - 热土豆选路: 将报文发送到最近的路由器。

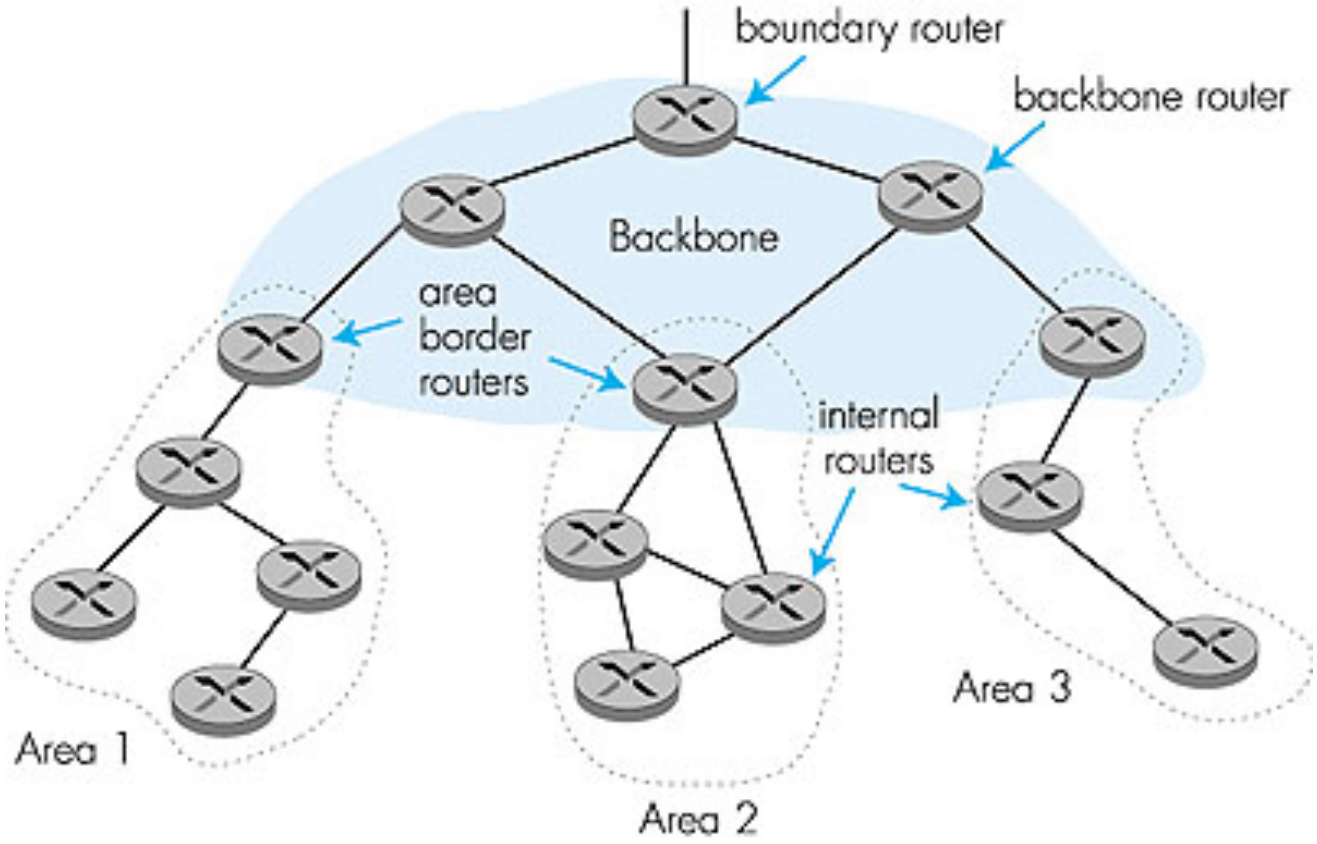


4.5 选路算法

- **因特网上的AS内层次路由——层次OSPF**
  - 为了使 OSPF 能够用于规模很大的网络，OSPF 将一个自治系统再划分为若干个更小的范围，叫作区域
  - 每一个区域都有一个 32 bit 的区域标识符
  - 区域也不能太大，在一个区域内的路由器最好不超过 200 个

4.5 选路算法

- **因特网上的AS内层次路由——层次OSPF**



划分区域

- 划分区域的好处就是将利用洪泛法交换链路状态信息的范围局限于每一个区域而不是整个的自治系统，这就减少了整个网络上的通信量。
- 在一个区域内部的路由器只知道本区域的完整网络拓扑，而不知道其他区域的网络拓扑的情况。
- OSPF 使用层次结构的区域划分。在上层的区域叫作主干区域(backbone area)。主干区域的标识符规定为0.0.0.0。主干区域的作用是用来连通其他在下层的区域。

4.5 选路算法

- **因特网上的AS间路由——BGP4**
  - 因特网的规模太大，使得自治系统之间路由选择非常困难。
    - 对于自治系统之间的路由选择，要寻找最佳路由是很不现实的。
    - 自治系统之间的路由选择必须考虑有关策略
  - BGP 为每个AS提供一种手段，以处理
    - 从相邻AS获取子网可达性信息
    - 向该AS内部的所有路由器传播这些可达性信息
    - 基于该可达性信息和AS策略，决定到达子网的“好”路由
      - 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

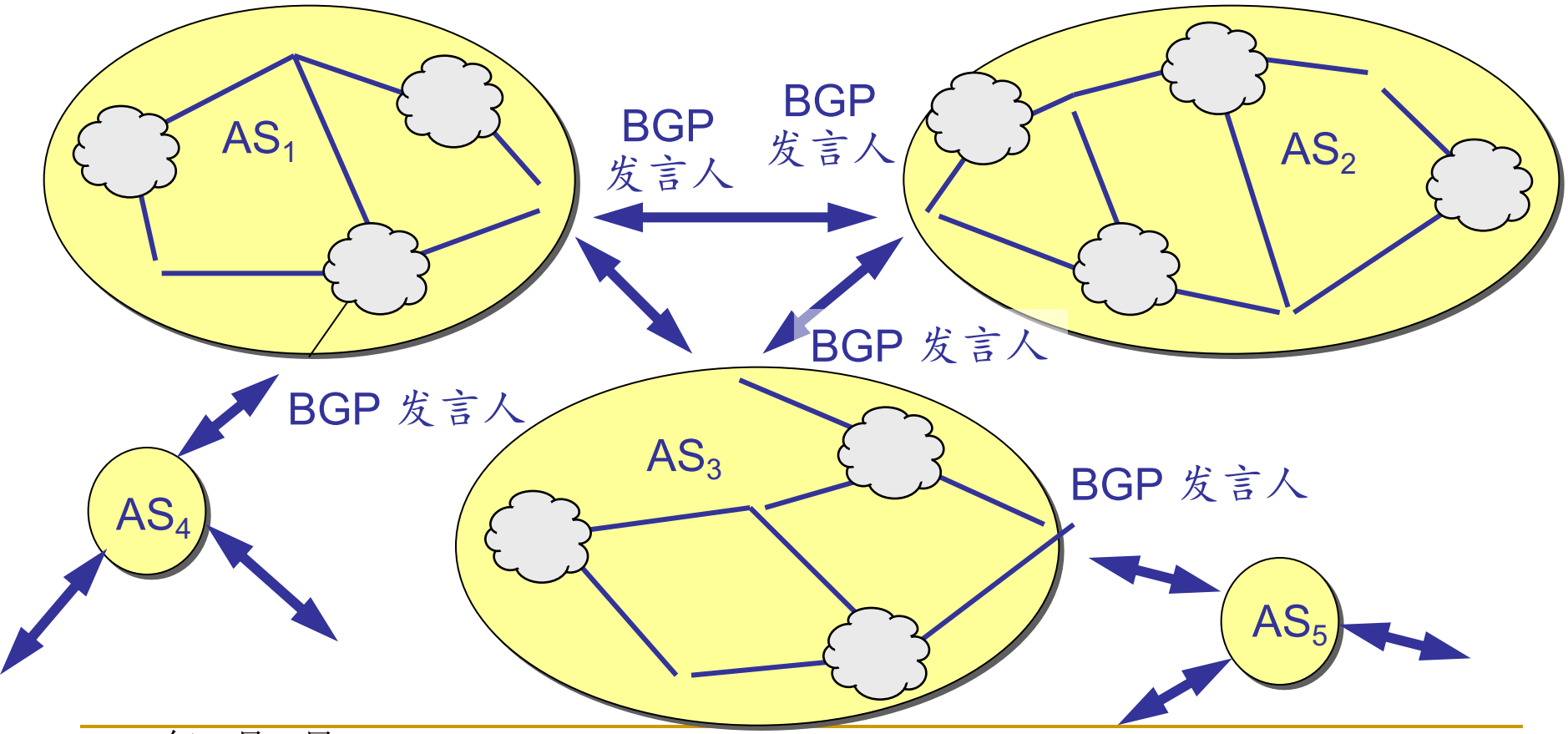
BGP 发言人

- 每一个自治系统的管理员要选择至少一个路由器作为该自治系统的“BGP 发言人”。
- 一般说来，两个 BGP 发言人都是通过一个共享网络连接在一起的，而 BGP 发言人往往就是 BGP 边界路由器

BGP 交换路由信息

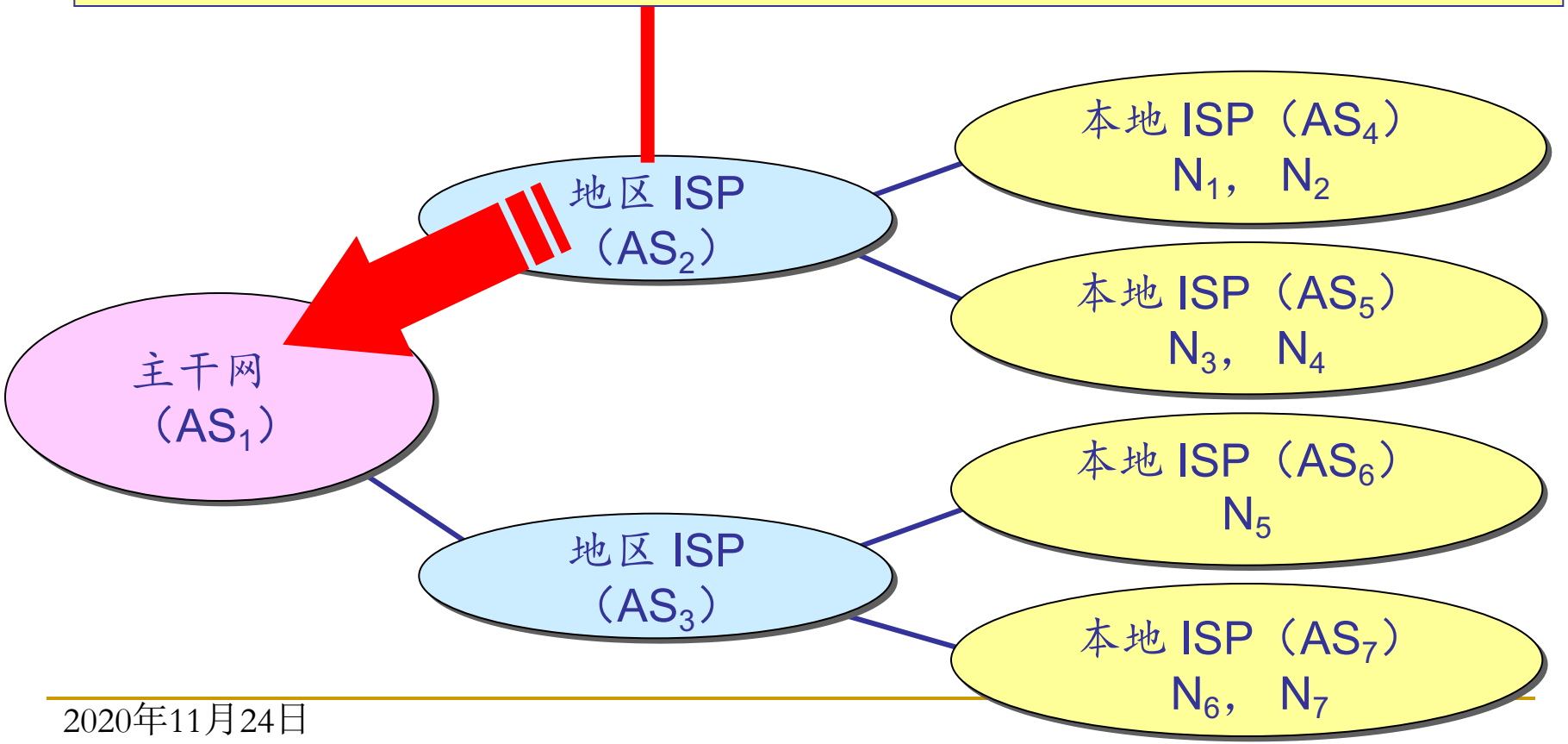
- 一个 BGP 发言人与其他自治系统中的 BGP 发言人要交换路由信息，就要先建立 TCP 连接，然后在此连接上交换 BGP 报文以建立 BGP 会话(session)，利用 BGP 会话交换路由信息。
- 使用 TCP 连接能提供可靠的服务，也简化了路由选择协议。
- 使用 TCP 连接交换路由信息的两个 BGP 发言人，彼此成为对方的邻站或对等站。

BGP 发言人和自治系统 AS 的关系



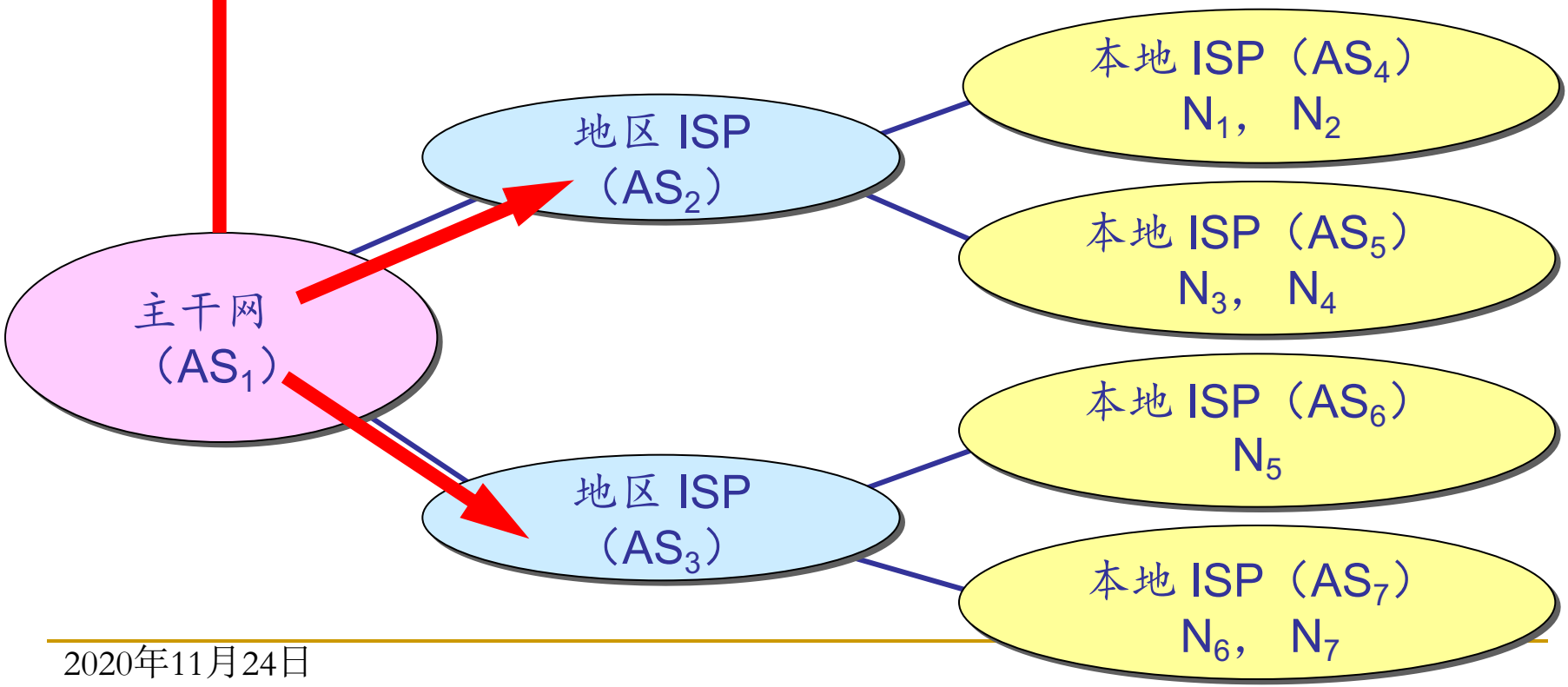
BGP 发言人交换路径向量

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。”



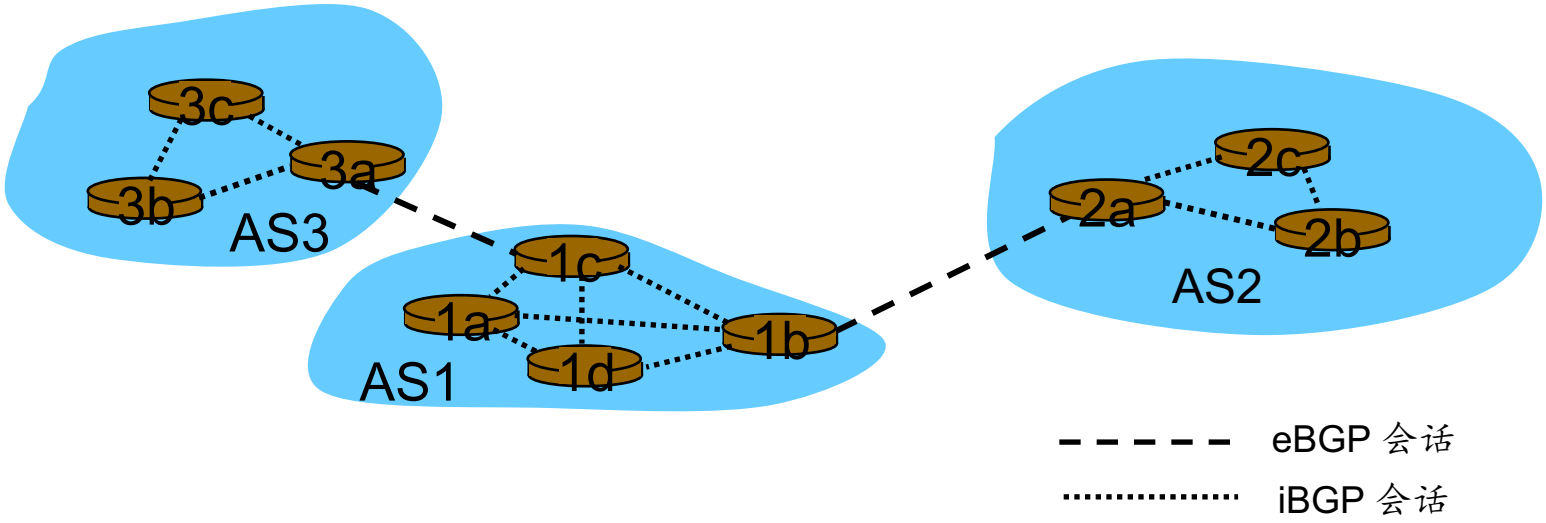
BGP 发言人交换路径向量

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）。”



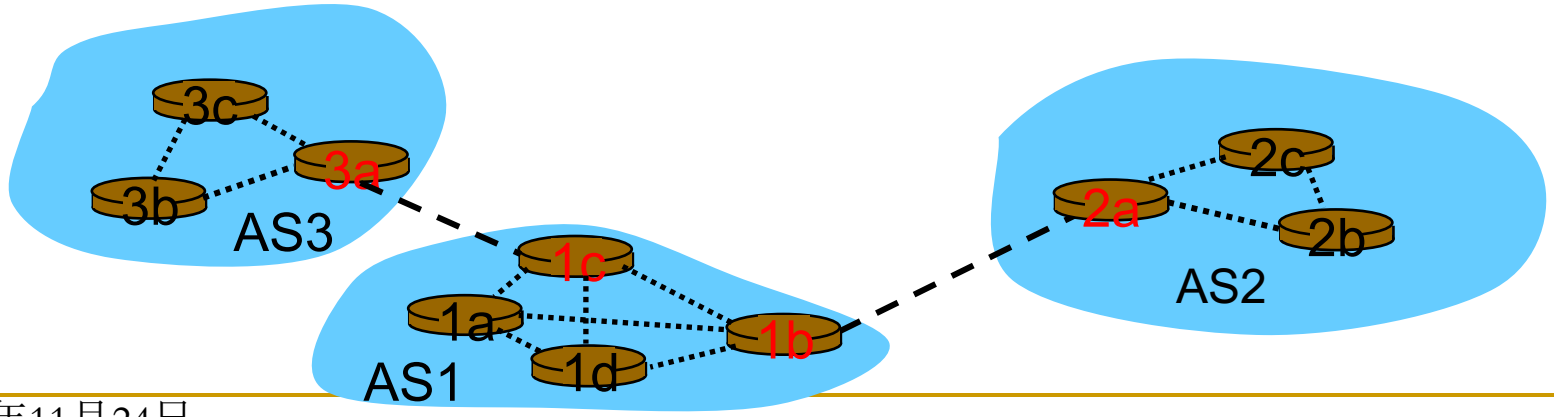
4.5 选路算法

- BGP路由通告
  - AS2可以聚合多个前缀为一个，并使用BGP向AS1通告单一前缀，即AS2承诺它将转发指向该前缀的任何数据报



4.5 选路算法

- 发布前缀可达性消息
  - 在网关路由器3a和1c之间使用eBGP, AS3向AS1发送一个自AS3可达的前缀列表.
  - 1c 则使用iBGP会话向AS1中的其他路由器发布这些前缀
  - 1b 则将和路由器2a间使用eBGP会话学习到的前缀信息发布到AS1
  - 当一个路由器得知一个新前缀，它为该前缀在其转发表中创建一个表项



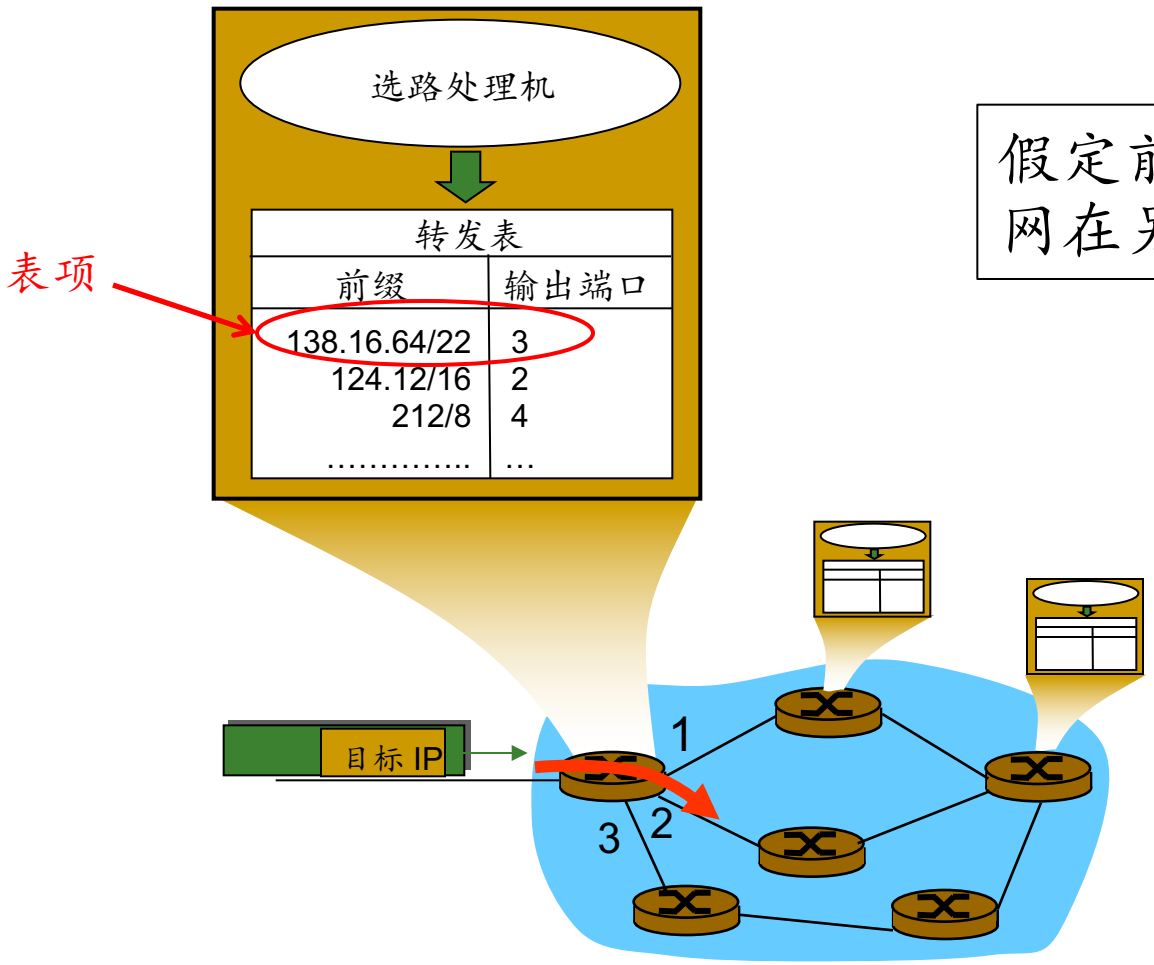
4.5 选路算法

- 路径和BGP路由
  - 当路由器通告一个前缀时，它随着前缀包含一些BGP属性
    - 前缀 +属性 = “路由”
  - 2个重要的属性:
    - AS-PATH: 该属性包含了前缀的通告已经通过的那些AS: AS2 AS1
    - NEXT-HOP: 指明到下一跳AS的具体的路由器 (因为可能从当前AS到下一跳AS之间可能有多条链路)
  - 当一台网关路由器接收到一个路由器通告时，它使用输入策略决定是否接收或过滤该路由

4.5 选路算法

- BGP路由选择
  - 一台路由器可能知道到一条前缀的多条路由路，路由器必须在可能的路由中选择一条。
  - 消除规则:
    - 本地偏好值: 策略决定。具有最高本地偏好值的路由将被选择。
    - 最短AS-PATH：在余下的路由中，具有最短AS-PATH的路由将被选择。
    - 从余下的路由中，选择具有最靠近NEXT-HOP路由器的路由:热土豆路由。

表项如何进入路由器转发表



假定前缀所指示子网在另一个AS中

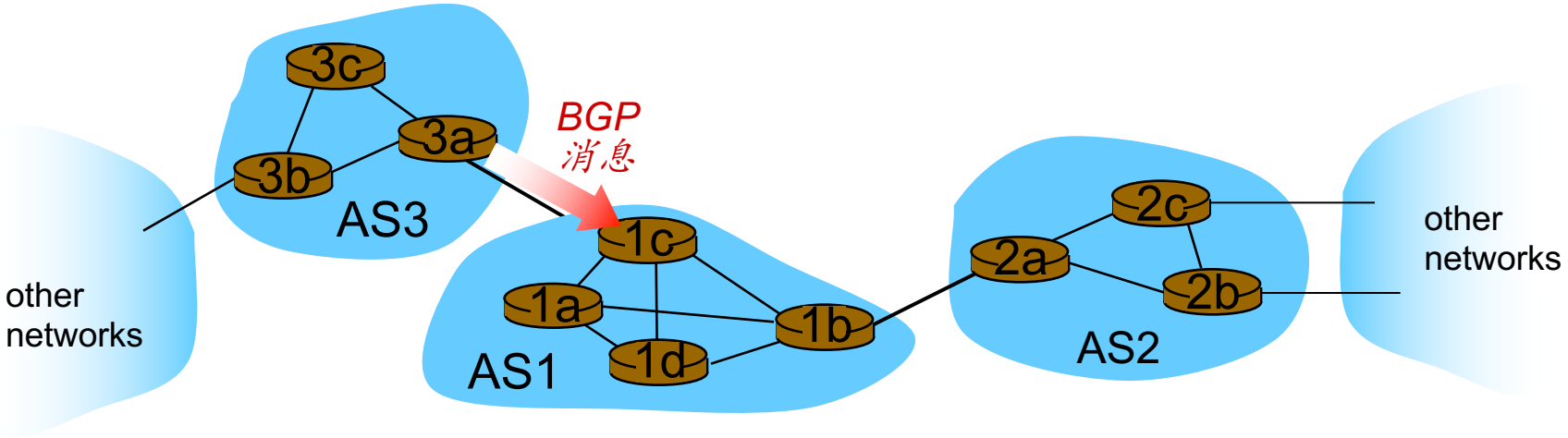
表项如何进入路由器转发表

步骤总览

1. 路由器要知道前缀的存在
2. 路由器要确定适当的输出端口
3. 路由器将“前缀-端口”对放入转发表中

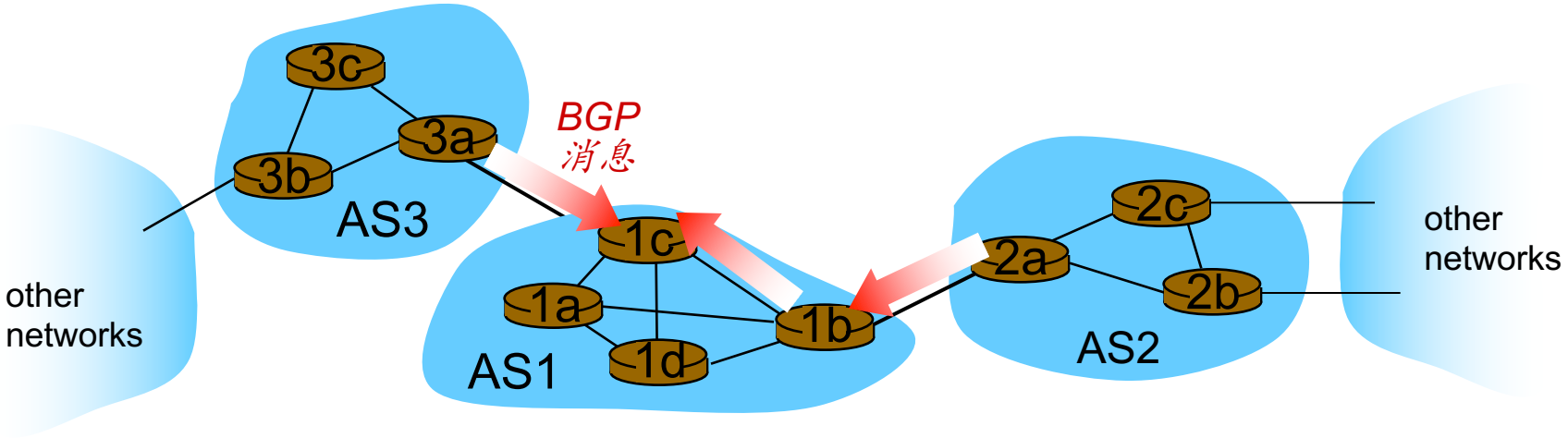
表项如何进入路由器转发表

1. 路由器如何知道前缀的存在



- BGP消息中会包含“路由”
- “路由”包括前缀和若干属性： AS-PATH、NEXT-HOP、…
- 例如： Prefix:138.16.64.0/22 ; AS-PATH: AS3 AS131 ; NEXT-HOP: 201.44.13.125

表项如何进入路由器转发表



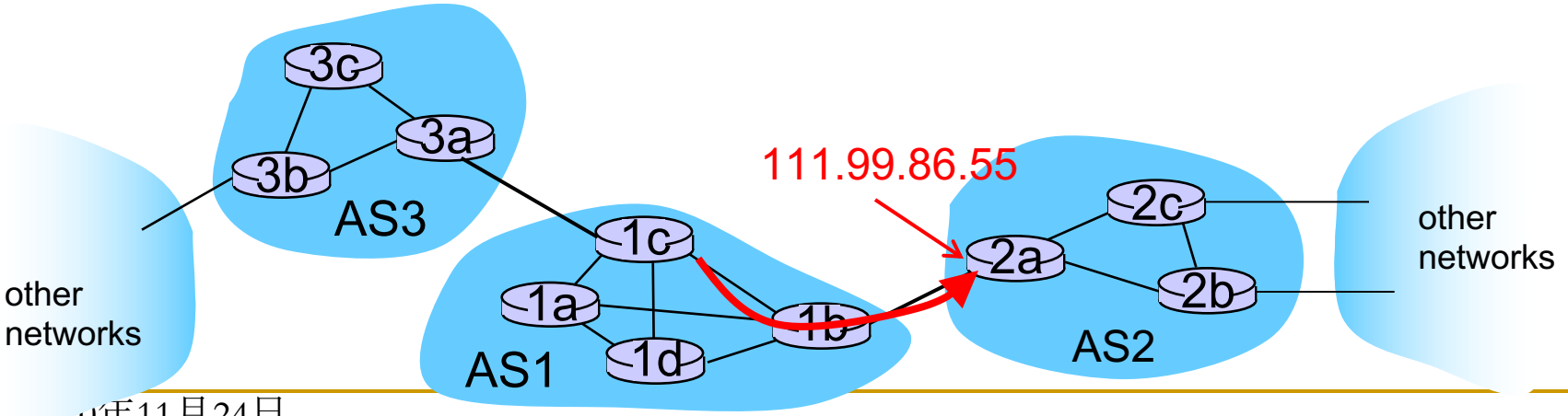
- 路由器可能会收到去往同一前缀的多个“路由”
- 应该选择哪一个路由呢？

表项如何进入路由器转发表

- 路由器可以基于最短AS-PATH进行选择
- 例如:
  - AS2 AS17 to 138.16.64.0/22
  - AS3 AS131 AS201 to 138.16.64.0/22

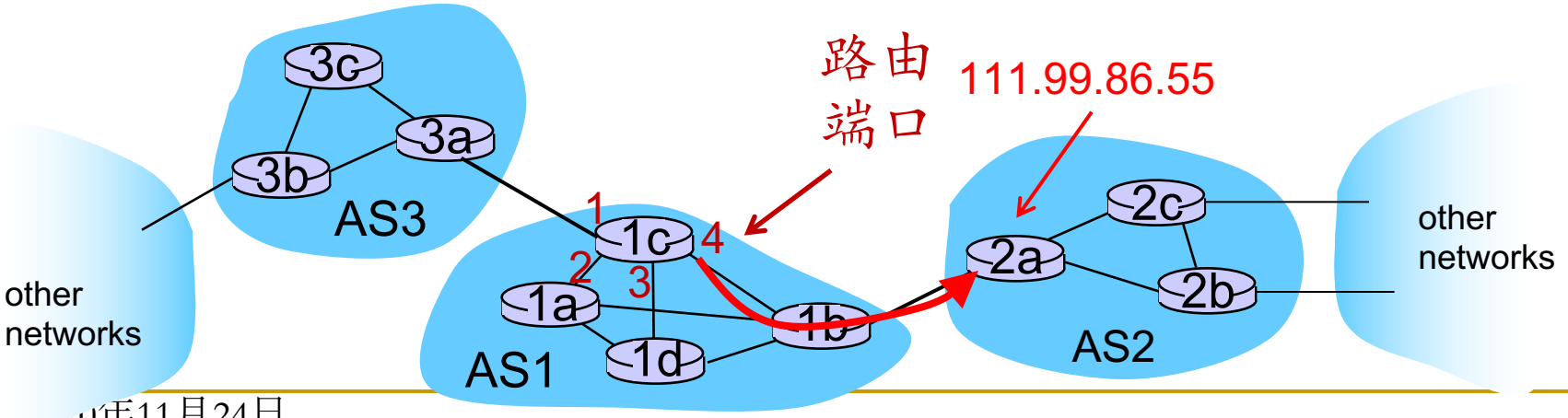
表项如何进入路由器转发表

- 获取选定“路由”的NEXT-HOP属性
  - “路由”的NEXT-HOP属性是AS PATH中第一个路由器的接口IP地址
- 例如:
  - AS-PATH: AS2 AS17 ; NEXT-HOP: 111.99.86.55
- 路由器1c使用OSPF或者RIP找到从1c去往111.99.86.55的最短路径



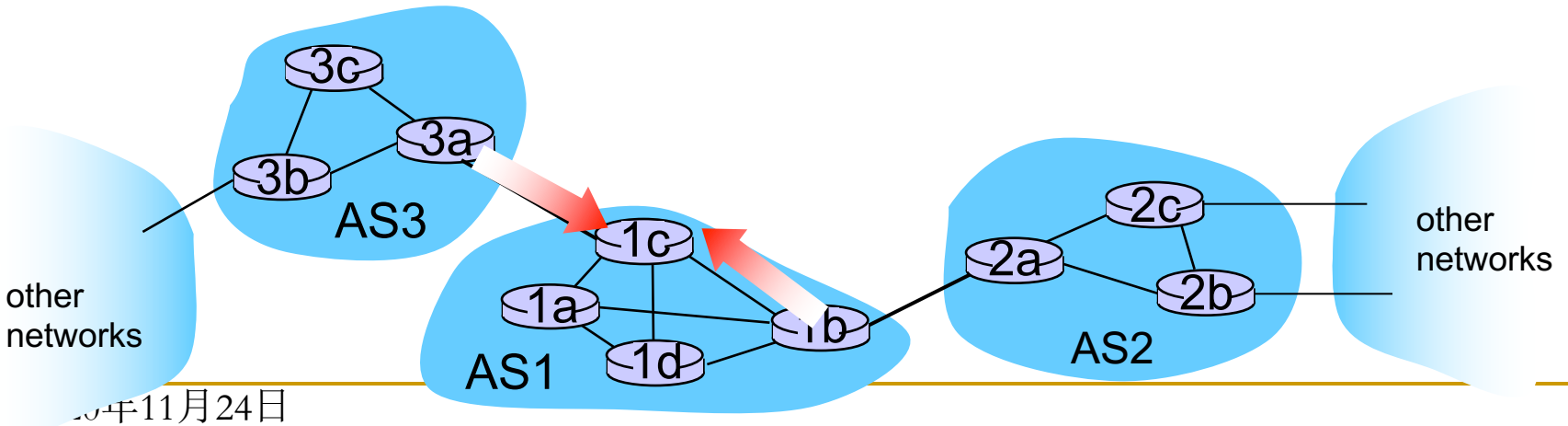
表项如何进入路由器转发表

- 路由器确定去往最短路径的端口
- 向转发表中增加表项:
  - (138.16.64.0/22 , port 4)



表项如何进入路由器转发表

- 假设有两条或者更多条最佳域间路由
- 那么就采用热土豆思想选择距离NEXT-HOP最近的路由
  - 采用IGP确定哪个边界路由器是最近的
  - Q: 从1c来说, 选择AS3 AS131还是AS2 AS17?
  - A: 显然应该选择AS3 AS131（离3a更近）



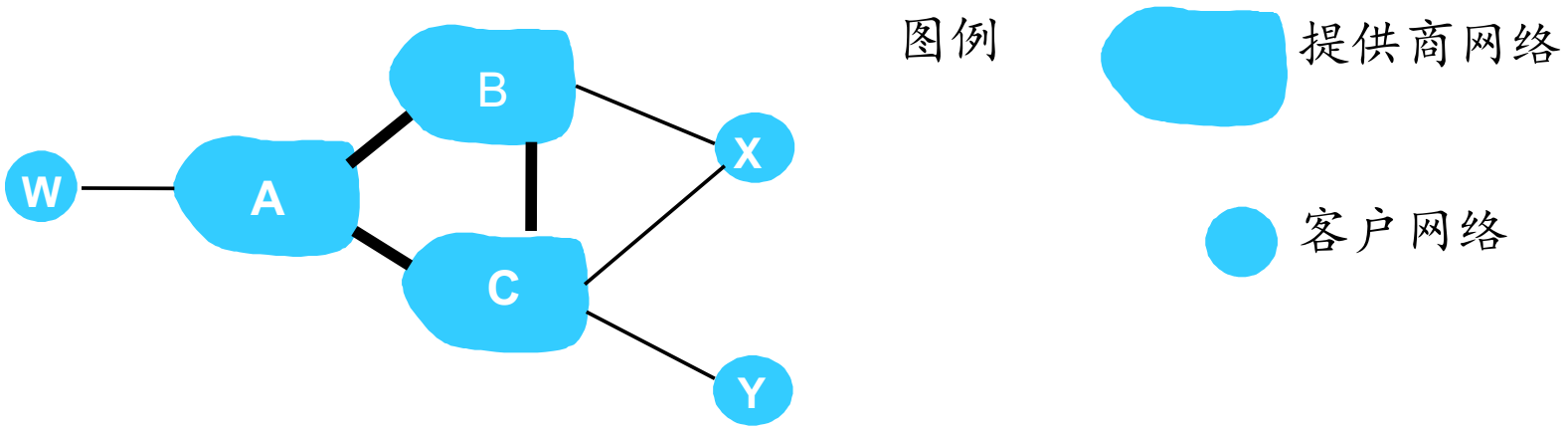
表项如何进入路由器转发表

- 总结
  - 路由器知晓前缀的存在性
    - 通过BGP通告得知
  - 确定此前缀的转发端口
    - 使用BGP路由选择确定最佳域间路由
    - 使用IGP路由选择确定最佳域内路由
    - 确定最佳路由的转发端口
  - 将（前缀，端口）表项放入转发表中

4.5 选路算法

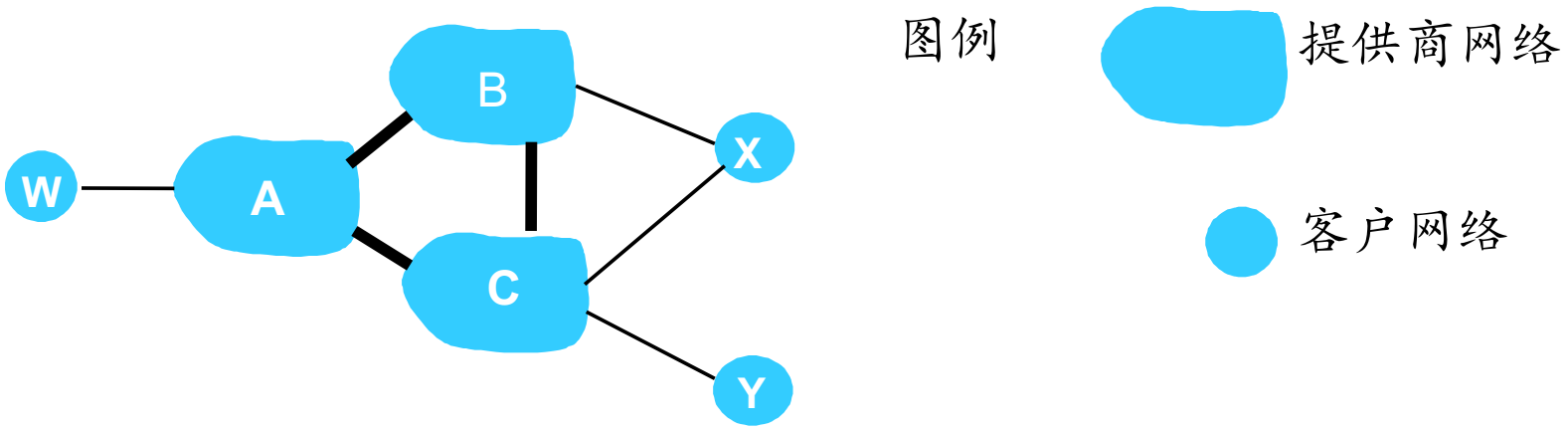
- BGP路由策略——情景1
  - A,B,C是提供商网络
  - X,W,Y是客户网络
  - X是一个双宿桩网络：隶属于2个网络
    - X不希望B通过X到达C （不希望成为流量经过的网路）
    - 因此X不会向B通告到C的路由信息

桩网路：所有进入该网络的流量必定以该网络为目的地；所有离开该网络的流量必定源于该网络



4.5 选路算法

- BGP路由策略——情景2
  - A告诉B，A有一条路径AW到W
  - B想向它的客户X通告路径BAW
  - B应该将路径BAW通告给C吗?
    - 应该不会! B 不会因此而得到任何回报，因为W和C都不是它的客户。
    - B希望C到W的流量通过A，而不是自己!
    - B希望任何流经自己的通信流必须是其源或目的在B的某个客户网络中!



4.5 选路算法

- 路由协议小结

	RIP	OSPF	BGP
协议类型	IGP	IGP	EGP
信息表达格式	距离—向量协议	链路—状态协议	路径向量
交换信息范围	相邻路由器	自治系统或区域内路由器	BGP发言人
交换信息内容	路由表	链路状态	路径向量
交换信息时间	30S	当链路状态发生变化	有变化
原则	最短路径	最小代价	可达性
收敛过程	较快	快	快
传输协议	UDP	IP	TCP
适用网络类型	小型网络	大型网络	自治系统之间
衡量标准	距离	可有多种度量标准	无

课后思考题

- 复习题 4.3、4.22、4.25、5.4、5.8
- 习　　题 4.1、4.3、4.5、4.10-12、4.14、4.16、5.3、5.14、39

作 业

- 习题
  - 4.5、4.12、5.3、5.14