







第5章 网络层






❖ 网络层主要解决的问题

- 路由选择
- 网络互连
- 拥塞控制
- 为上层提供服务

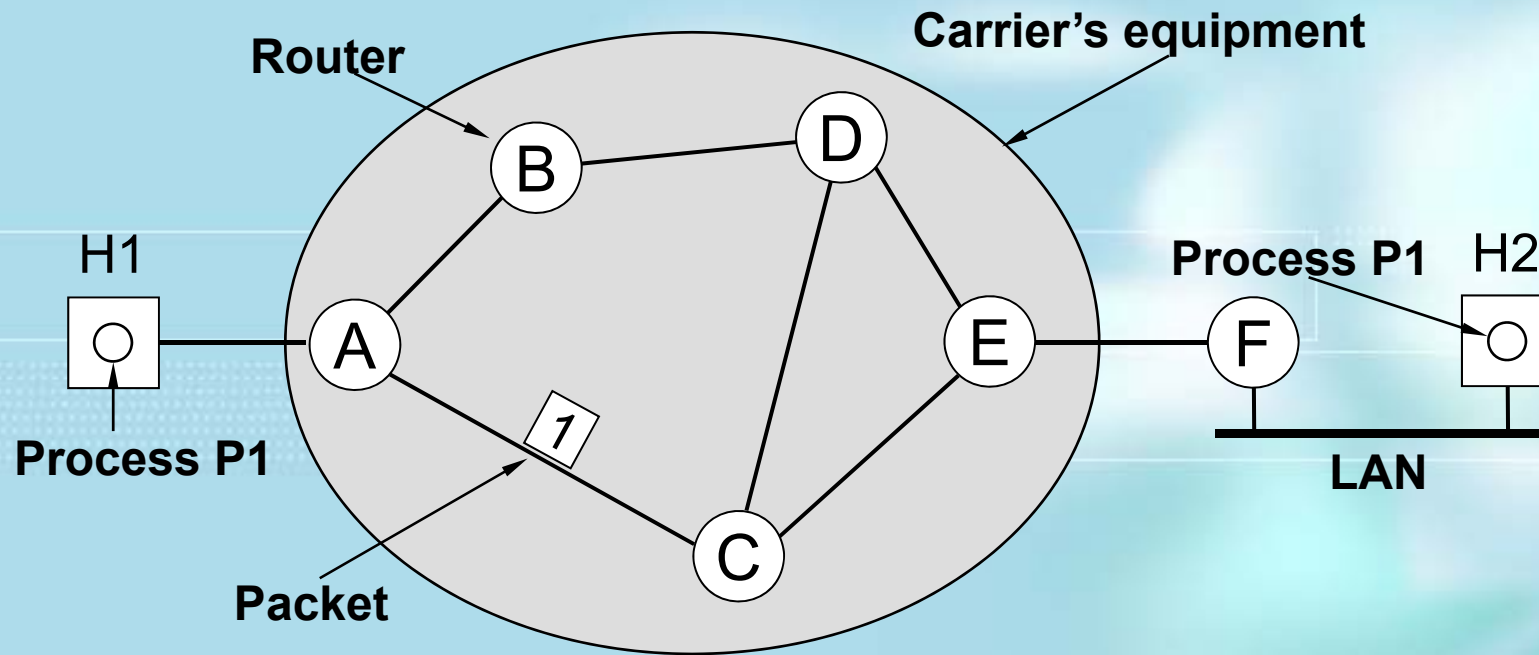
第5章 网络层

- ❖ 网络层设计的相关问题 
- ❖ 路由算法 
- ❖ 拥塞控制 
- ❖ 服务质量 
- ❖ 网络互联 
- ❖ 因特网中的网络层 

网络层的设计






- ❖ 存储转发的数据包交换 
- ❖ 为传输层提供的服务 
- ❖ 数据包子网的实现 
- ❖ 虚电路子网的实现 
- ❖ 虚电路子网和数据报子网的比较 

网络层协议环境



Tnbm P344 Fig. 5-1 网络层协议环境

网络层的设计

- ❖ 存储转发的数据包交换 
- ❖ 为传输层提供的服务 
- ❖ 数据报网络的实现 
- ❖ 虚电路子网的实现 
- ❖ 虚电路子网和数据报子网的比较 

为传输层提供的服务

- ❖ 服务应与路由器技术无关
- ❖ 路由器的数量、类型和拓扑结构对于传输层来说应是不可见的
- ❖ 传输层所能获得的网络地址应采用统一的编址方式，并允许跨越多个**LAN**和**WAN**

网络层提供的服务类型

对于网络层提供的服务有两种观点：

- ❖ 数据报网络：网络是不可靠的，网络服务不应面向连接，分组的排序和流控制应不属于网络层，每个分组都单独寻径，所以必须携带完整的目的地址

如Internet

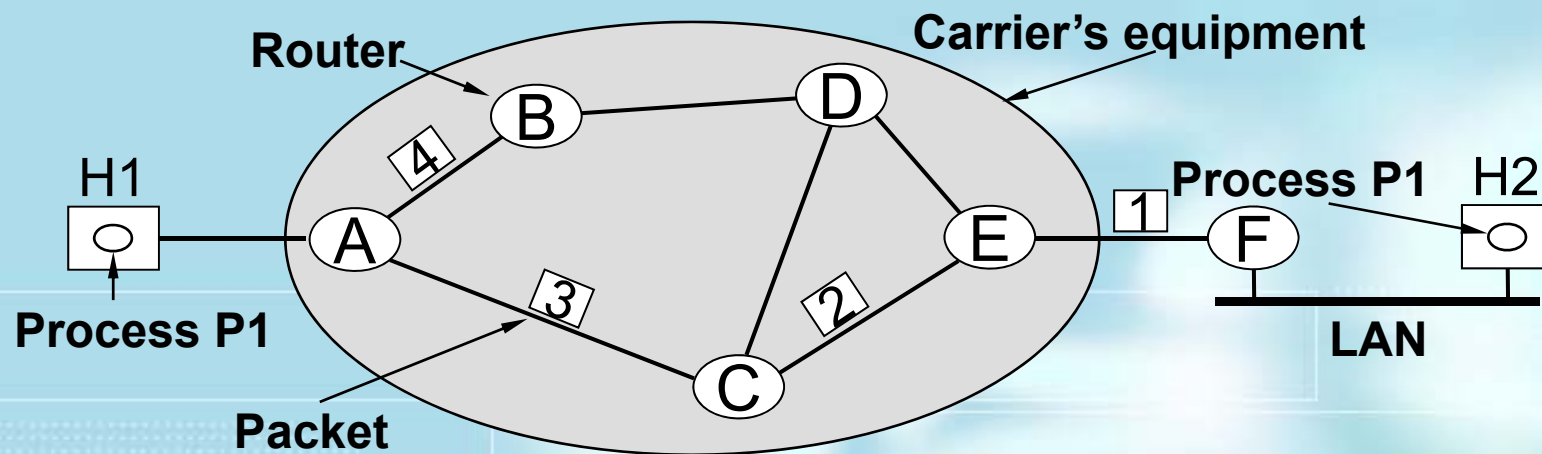
- ❖ 虚电路网络：网络应该提供可靠的、面向连接的服务，否则服务质量将无从谈起，尤其对于多媒体应用

如 ATM

网络层的设计

- ❖ 存储转发的数据包交换
- ❖ 为传输层提供的服务
- ❖ 数据报网络的实现
- ❖ 虚电路网络的实现
- ❖ 虚电路子网和数据报子网的比较

面向无连接服务的实现（数据报子网）



路由器A按左边的路由表运行，后来发现如到E和F应该走B才更好，于是更新路由表

A的路由表

| | |
|---|---|
| A | - |
| B | B |
| C | C |
| D | B |
| E | C |
| F | C |

| | |
|---|---|
| A | - |
| B | B |
| C | C |
| D | B |
| E | B |
| F | B |

C的路由表

| | |
|---|---|
| A | A |
| B | A |
| C | - |
| D | D |
| E | E |
| F | E |

E的路由表

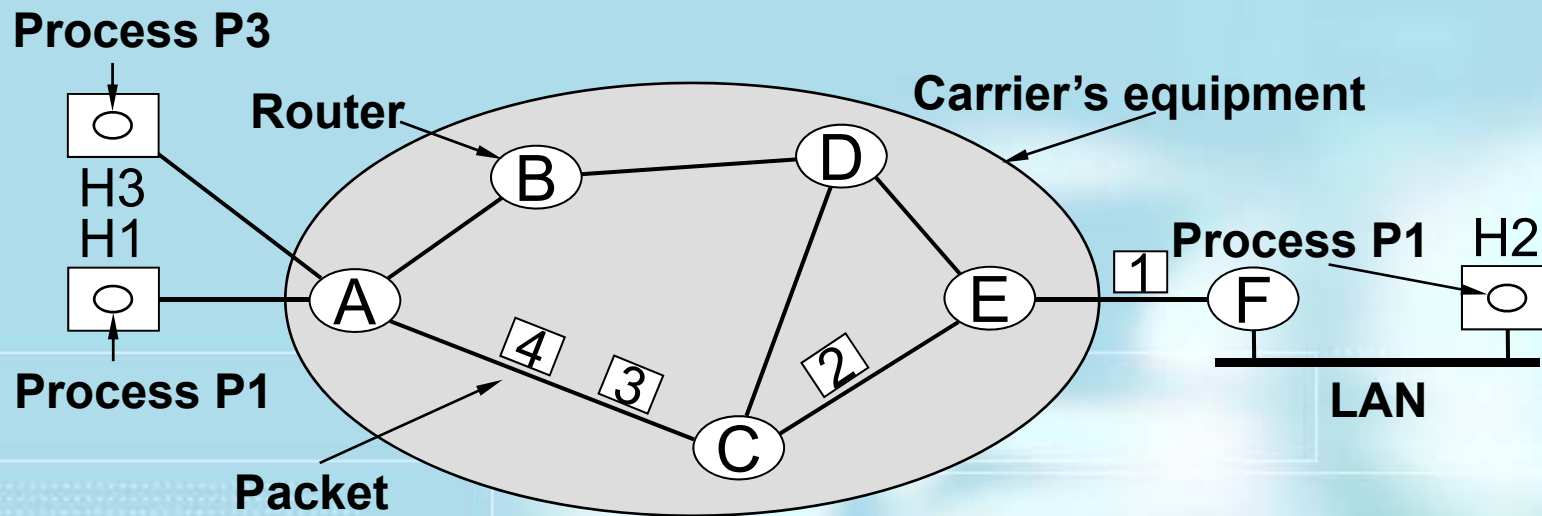
| | |
|---|---|
| A | C |
| B | D |
| C | C |
| D | D |
| E | - |
| F | F |

Tnbm P346 Fig. 5-2 数据报子网中分组的寻径

网络层的设计

- ❖ 存储转发的数据包交换
- ❖ 为传输层提供的服务
- ❖ 面向无连接服务的实现
- ❖ 面向连接服务的实现
- ❖ 虚电路子网和数据报子网的比较

面向连接服务的实现（虚电路子网）



H1和H2已建立了1#连接
H3要和H2建立连接只能是2#

| A的路由表 | | C的路由表 | | E的路由表 | |
|-------|---|-------|---|-------|---|
| H1 | 1 | C | 1 | C | 1 |
| H3 | 1 | C | 2 | C | 2 |
| 入口 | | 出口 | | 入口 | |

Tnbm P348 Fig. 5-3 虚电路子网中分组的寻径

网络层的设计

- ❖ 存储转发的数据包交换
- ❖ 为传输层提供的服务
- ❖ 面向无连接服务的实现
- ❖ 面向连接服务的实现
- ❖ 虚电路子网和数据报子网的比较

虚电路子网和数据报子网的比较

| | 数据报子网 | 虚电路子网 |
|----------|---------------------|--------------------------------|
| 建立电路连接 | 不需要 | 需要 |
| 寻址 | 每个分组包含完整的源和目的地址 | 每个分组包含一个很短的虚电路号 |
| 状态信息 | 路由器不保留连接的状态信息 | 每条虚电路要求为每个连接提供路由表空间 |
| 寻径 | 路由器为每个分组独立寻径 | 寻径在虚电路建立时完成，此后，所以分组按此路径传输 |
| 路由器故障的影响 | 除路由器崩溃，所以分组丢失，否则无影响 | 所有通过该故障路由器的虚电路全部终止 |
| 服务质量 | 困难 | 对每条虚电路，沿途的路由器如有足够的资源可分配，则很容易实现 |
| 拥塞控制 | 困难 | 对每条虚电路，沿途的路由器如有足够的资源可分配，则很容易实现 |

Tnbm P349 Fig. 5-4虚电路子网和数据报子网的比较

虚电路子网/数据报子网的比较(续)

❖ 虚电路子网

- 通过路径选择后建立连接
- 分组按序传输
- 服务质量能得到保证
- 通信后撤销连接
- 适合于实时传输

❖ 数据报子网

- 每个分组分别选择最佳路径，健壮性较好
- 整个网络系统的信道利用率高，成本低
- 差错控制和排序工作由协议高层（主机）完成
- 适合于非实时传输

第5章 网络层

- ❖ 网络层设计的相关问题
- ❖ 路由算法
- ❖ 拥塞控制
- ❖ 服务质量
- ❖ 网络互联
- ❖ 因特网中的网络层



路由算法

- ❖ 路由算法是网络层软件的一个重要部分，它决定进入的分组应从哪一根输出线传输
- ❖ 如果是数据报子网，将在每一个分组到达时作此决定
- ❖ 如果是虚电路子网，是在虚电路建立时决定，该连接上所有分组都将沿此线路传输
- ❖ 路由与转发：路由是寻径，转发是当一个分组到达时发生的动作

路由算法（续）






❖ 路由算法设计必须考虑的问题

正确性 简单性 健壮性 稳定性 公平性 最优性

❖ 路由算法中的度量标准

- 路径长度
- **hop**数
- 延迟时间

路由算法的分类

- ❖ 静态算法 
- ❖ 自适应算法 
- ❖ 拓扑相关的路由算法 
- ❖ 移动节点的路由 
- ❖ **Ad-hoc**网络的路由 

静态算法（static routing）

为路由器配置一张最优的路由表

❖ 最短路径算法（**Dijkstra**）



❖ 扩散法（**flooding**）

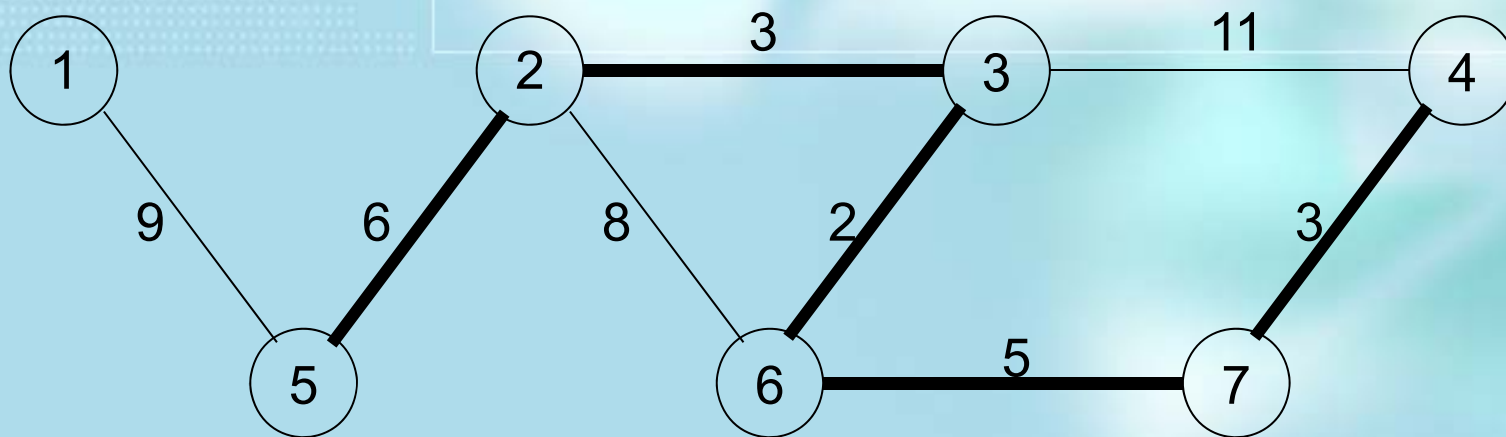


最短路由选择（Dijkstra）

- ❖ **Dijkstra算法（1959）**：通过用边的权值作为距离的度量来计算最短路径，有最少边数的路径不一定是最短路径

如下图：

5和4之间边数最少的路径是5234但最短路径是523674



采用的数据结构

- ❖ 集合**S**: 尚未找到最短路径的节点的集合
- ❖ 数组**R**: **R[i]**为从指定源点去节点*i*的路径上, 节点*i*的前一个节点
- ❖ 数组**D**: **D[i]**为从指定源点到节点*i*的最短距离

算法的初始化

- ❖ 初始化集合**S**为除源节点外的所有节点
- ❖ 初始化数组**D**: 如果从源节点到节点**v**的边存在, 则**D(v)**为该边的权值, 否则为无穷大
- ❖ 初始化数组**R**: 如果从源节点到节点**v**的边存在, 则**R(v)**为源节点, 否则为**0**

算法

WHILE (集合**S**非空)

{ 从**S**中选一节点**u**, 使**D[u]**最小;

 如果 (**D[u]**为无穷大)

 {错误! 无路径存在, 退出}

 把**u**从**S**中删去;

 对(**u,v**)是边的每个节点**v**

 { 如果 (**v**仍在**S**中)

C=D[u]+weight(u,v);

 如果 (**C<D[v]**) /***v**找到了一条更短的路径*/

 {**R[v]= u;** /*替换**v**的最短路径及长度*/

D[v]=C;

 }

 }

 }

从5出发到各个节点的最短路径

| S | R1 | D1 | R2 | D2 | R3 | D3 | R4 | D4 | R6 | D6 | R7 | D7 | u |
|---------------|----|----|----|----|----|----------|----|----------|----|----------|----|----------|---|
| {1,2,3,4,6,7} | 5 | 9 | 5 | 6 | 0 | ∞ | 0 | ∞ | 0 | ∞ | 0 | ∞ | 2 |
| {1,3,4,6,7} | 5 | 9 | 5 | 6 | 2 | 9 | 0 | ∞ | 2 | 14 | 0 | ∞ | 1 |
| {3,4,6,7} | 5 | 9 | 5 | 6 | 2 | 9 | 0 | ∞ | 2 | 14 | 0 | ∞ | 3 |
| {4,6,7} | 5 | 9 | 5 | 6 | 2 | 9 | 3 | 20 | 3 | 11 | 0 | ∞ | 6 |
| {4,7} | 5 | 9 | 5 | 6 | 2 | 9 | 3 | 20 | 3 | 11 | 6 | 16 | 7 |
| {4} | 5 | 9 | 5 | 6 | 2 | 9 | 7 | 19 | 3 | 11 | 6 | 16 | 4 |

静态算法（static routing）

为路由器配置一张最优的路由表

❖ 最短路径算法（**Dijkstra**）

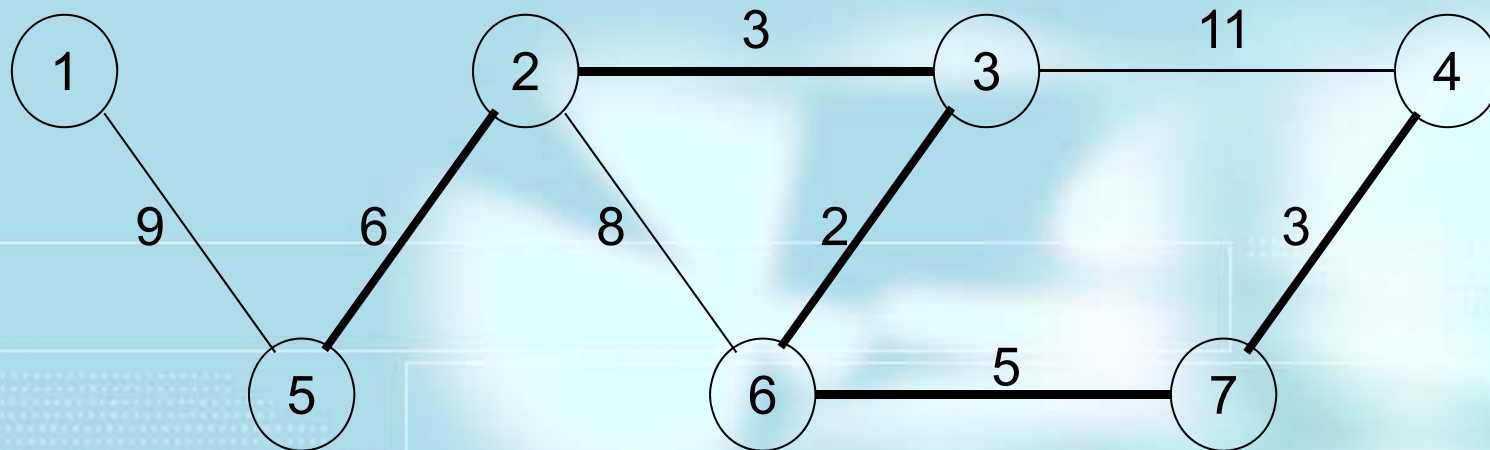


❖ 扩散法（**flooding**）



扩散法 (flooding)

❖ 不计算路径，有路就走



如从**5**出发到**4**:

数据包从**5→1,2**; **2→3,6**; **3→6,4**; **6→3,7**; **7→4**

要解决的问题: 数据包重复到达某一节点, 如**3, 6**

扩散法（续）

❖ 解决方法

- 在数据包头设一计数器初值，每经过一个节点自动减1，计数值为0时，丢弃该数据包
- 在每个节点上建立登记表，则数据包再次经过时丢弃

缺点：重复数据包多，浪费带宽

优点：可靠性高，路径最短，常用于军事

路由算法的分类

❖ 静态算法



❖ 自适应算法



❖ 拓扑相关的路由算法



❖ 移动节点的路由



❖ **Ad-hoc**网络的路由



自适应算法 (adaptive algorithm)

- ❖ 自适应算法是动态的、分布式的算法
- ❖ 实现分布式算法的三要素：
 - **The measurement process** (测量)
 - **The update protocol** (更新协议)
 - **The calculation** (计算)

自适应算法（adaptive algorithm）

路由器动态建立和维护一张最优的路由表

❖ 距离矢量算法（**D-V**）



❖ 链路状态算法（**L-S**）



D-V算法的工作原理

- ❖ 每个路由器用两个向量 D_i 和 S_i 来表示该点到网上所有节点的路径距离及其下一个节点
- ❖ 相邻路由器之间交换路径信息
- ❖ 各节点根据路径信息更新路由表

d_{i1} : 从节点*i* 到节点**1** 的时延向量

d_{i2} : 从节点*i* 到节点**2** 的时延向量

$$D_i = \begin{pmatrix} d_{i1} \\ d_{i2} \\ d_{i3} \\ \dots \\ d_{in} \end{pmatrix}$$

$$S_i = \begin{pmatrix} s_{i1} \\ s_{i2} \\ s_{i3} \\ \dots \\ s_{in} \end{pmatrix}$$

s_{i1} : 从节点*i*到节点**1**的一条最小时延路径上的下一个节点

s_{i2} : 从节点*i*到节点**2**的一条最小时延路径上的下一个节点

其中: **n** —网络中的节点数

D_i —节点*i*的时延向量

d_{ij} —节点*i*到*j*的最小时延的当前估计值

S_i —节点*i*的后继节点向量

s_{ij} —从节点*i*到*j*的最小时延路径上的下一节点

路由表的更新

$$d_{ij} = \min(d_{ix} + d_{xj}) \quad (x \in A)$$

（从i到j的时延取途经每个节点时的时延的最小值）

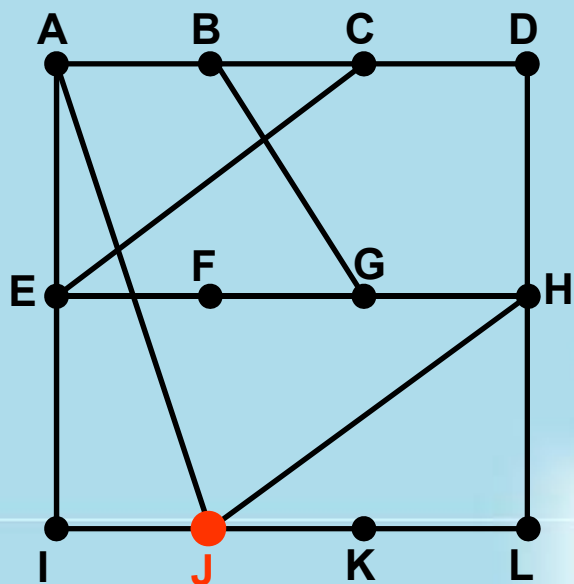
$$S_{ij} = x \quad (\text{从i到j途经的下一个节点为} x)$$

其中：A — 与i相邻的所有节点的集合

d_{ij} — i到j的最短距离

d_{ix} — i到x的最短距离

d_{xj} — x到j的最短距离



注意：A→I为21；I→A为24

因为：往和返的信道流量不一定相同，节点A和I也并非在同一时刻测得，且线路状态是动态变化的

所谓节点即路由器
当前节点为J

J重新估计的延时

| To | 通过A | 通过I | 通过H | 通过K | ↓ ↓ ↓ | 线路 |
|----|-----------------|------------------|------------------|-----------------|--------------|----|
| A | 0 | 24 | 20 | 21 | 8 | A |
| B | 12 | 36 | 31 | 28 | 20 | A |
| C | 25 | 18 | 19 | 36 | 28 | I |
| D | 40 | 27 | 8 | 24 | 20 | H |
| E | 14 | 7 | 30 | 22 | 17 | I |
| F | 23 | 20 | 19 | 40 | 30 | I |
| G | 18 | 31 | 6 | 31 | 18 | H |
| H | 17 | 20 | 0 | 19 | 12 | H |
| I | 21 | 0 | 14 | 22 | 10 | I |
| J | 9 | 11 | 7 | 10 | 0 | - |
| K | 24 | 22 | 22 | 0 | 6 | K |
| L | 29 | 33 | 9 | 9 | 15 | K |
| | J到A 延时 为8 | J到I 延时 为10 | J到H 延时 为12 | J到K 延时 为6 | 节点J的 新路由表 | |

Tnbm P286 Fig. 5-9 D-V算法的路由表更新

D-V算法的缺点

- ❖ 交换的路径信息量大
- ❖ 路径信息不一致
- ❖ 收敛速度慢（坏消息）
- ❖ 不适合大型网络



无穷计算问题

❖ 好消息传播得快，坏消息传播得慢

| A | B | C | D | E | |
|---|----------|----------|----------|----------|--------|
| ● | ● | ● | ● | ● | |
| | ∞ | ∞ | ∞ | ∞ | 初始时 |
| 1 | ∞ | ∞ | ∞ | ∞ | 第1次交换后 |
| 1 | 2 | ∞ | ∞ | ∞ | 第2次交换后 |
| 1 | 2 | 3 | ∞ | ∞ | 第3次交换后 |
| 1 | 2 | 3 | 4 | ∞ | 第4次交换后 |

| A | B | C | D | E | |
|---|----------|----------|----------|----------|--------|
| ● | ● | ● | ● | ● | |
| | 1 | 2 | 3 | 4 | 初始时 |
| | 3 | 2 | 3 | 4 | 第1次交换后 |
| | 3 | 4 | 3 | 4 | 第2次交换后 |
| | 5 | 4 | 5 | 4 | 第3次交换后 |
| | 5 | 6 | 5 | 6 | 第4次交换后 |
| | 7 | 6 | 7 | 6 | 第5次交换后 |
| | 7 | 8 | 7 | 8 | 第6次交换后 |
| | ... | ... | ... | ... | |
| | ∞ | ∞ | ∞ | ∞ | |

A下网了

Tnbn P287 Fig. 5-10 无穷计算问题

克服收敛速度慢的方法

❖ 水平分裂

同距离矢量法，只是到**X**的距离并不是真正的距离，
对下方点通知真正的距离，对上方点，给出无穷大
如上图中的**C**点，它向**D**通知到**A**的是真正距离，而向
B通知到**A**的距离是无穷大

❖ Holddown

当发现不通时，不重新选路径，而是把它设成无穷大
这些方法尚在研究之中

自适应算法 (adaptive algorithm)

路由器动态建立和维护一张最优的路由表

❖ 距离矢量算法 (**D-V**)



❖ 链路状态算法 (**L-S**)



链路状态算法（L-S）

（Link State Routing）

基本思想：

- ❖ 发现它的邻接节点，并得到其网络地址
- ❖ 测量它到各邻接节点的延迟或开销
- ❖ 组装一个分组以告知它刚知道的所有信息
- ❖ 将这个分组发给所有其他路由器
- ❖ 计算到每个其他路由器的最短路径



发现邻接节点

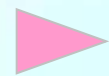
- ❖ 当一个路由器启动后，向每个点到点线路发送**HELLO**分组（携带自己的网络地址），另一端的路由器发送回来一个应答来说明它是谁，即通报其网络地址

链路状态算法（L-S）

（Link State Routing）

基本思想：

- ❖ 发现它的邻接节点，并得到其网络地址
- ❖ 测量它到各邻接节点的延迟或开销
- ❖ 组装一个分组以告知它刚知道的所有信息
- ❖ 将这个分组发给所有其他路由器
- ❖ 计算到每个其他路由器的最短路径








测量线路开销

- ❖ 发送一个**ECHO**分组要求对方立即响应，通过测量一个来回时间再除以**2**，发送方就可以得到一个延迟估计值，想要更精确些，可以重复这一过程，取其平均值
- ❖ 如果链路状态发生过变化，则进入下一步骤

链路状态算法（L-S）

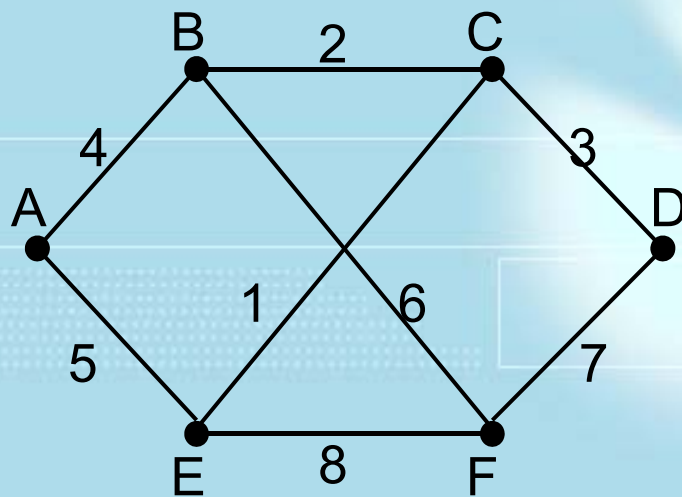
（Link State Routing）

基本思想：

- ❖ 发现它的邻接节点，并得到其网络地址 
- ❖ 测量它到各邻接节点的延迟或开销 
- ❖ 组装一个分组以告知它刚知道的所有信息 
- ❖ 将这个分组发给所有其他路由器 
- ❖ 计算到每个其他路由器的最短路径 

构造分组

子网及其节点到其邻节点（路由器）的线路开销测量值（即延时，假设以ms计）



子网的链路、状态及分组情况：

| A | | B | | C | | D | | E | | F | |
|----|---|----|---|----|---|----|---|----|---|----|---|
| 序号 | | 序号 | | 序号 | | 序号 | | 序号 | | 序号 | |
| 年龄 | | 年龄 | | 年龄 | | 年龄 | | 年龄 | | 年龄 | |
| 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 |

节点A仅与节点B和E相邻

A → B的时延为4ms

A → E的时延为5ms

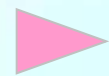
Tnbnm P363 Fig. 5-13 L-S算法的路由表更新

链路状态算法（L-S）

（Link State Routing）

基本思想：

- ❖ 发现它的邻接节点，并得到其网络地址
- ❖ 测量它到各邻接节点的延迟或开销
- ❖ 组装一个分组以告知它刚知道的所有信息
- ❖ 将这个分组发给所有其他路由器
- ❖ 计算到每个其他路由器的最短路径



发布链路状态分组

- ❖ 用扩散法（向邻接的节点）发布链路状态分组（以**B**为例，**B**的邻接点有**A**、**C**、**F**）

源节点**E**的链路状态分组经**A**和**F**到节点**B**，节点**B**必须再将**E**的状态分组转送到**C**，并向**A**和**F**发**ACK**

| 源 | 序号 | 年龄 | 发送标志 | | | ACK标志 | | | 数据 |
|---|----|----|------|---|---|-------|---|---|----|
| | | | A | C | F | A | C | F | |
| A | 21 | 60 | 0 | 1 | 1 | 1 | 0 | 0 | |
| F | 21 | 60 | 1 | 1 | 0 | 0 | 0 | 1 | |
| E | 21 | 59 | 0 | 1 | 0 | 1 | 0 | 1 | |
| C | 20 | 60 | 1 | 0 | 1 | 0 | 1 | 0 | |
| D | 21 | 59 | 1 | 0 | 0 | 0 | 1 | 1 | |

Tnbm P365 Fig. 5-14 链路状态分组的转发和确认

存在的问题

- ❖ 状态分组的重复到达
- ❖ 如果序号循环使用，就会发生重复
- ❖ 如果一个路由器被重起，序号将从**0**开始重新计数，但这些分组会被当成过时分组
- ❖ 如果序号发生错误（如序号用**32**位表示，**4**被看成**65540**，第**16**位的**0**被误传成了**1**），则很多分组将被看成过时分组（此时**5~65539**均为过时分组，因为当前的分组序号是**65540**）

解决办法

- ❖ 使用一个**32**位序号，即使每秒钟发送一个分组，**137**年才会循环一次
- ❖ 在每个分组中加一年龄字段（如初值为**60**），每秒钟将年龄减**1**，为**0**后该分组将被丢弃，否则不会被认为是过时分组

链路状态算法（L-S）

（Link State Routing）

基本思想：

- ❖ 发现它的邻接节点，并得到其网络地址
- ❖ 测量它到各邻接节点的延迟或开销
- ❖ 组装一个分组以告知它刚知道的所有信息
- ❖ 将这个分组发给所有其他路由器
- ❖ 计算到每个其他路由器的最短路径

计算新路由

❖ 用**Dijkstra**算法计算到每个节点的路由

得到该节点到每个节点的最短路径

L-S路由算法的优缺点






❖ LSR的优点

- 路由信息的一致性好，坏消息也一样传播得快
- 状态分组的长度较短，仅包含到邻接点的距离、序号和年龄等，与网络规模关系不大，传输所耗用的网络带宽不大，此外，状态分组的扩散，由于年龄参数的设定，不会无限制扩散，所以可适用于大型网络

❖ LSR的缺点

- 每个路由器需要有较大的存储空间，用以存储所收到的每一个节点的链路状态分组
- 计算工作量大，每次都必须计算最短路径

路由算法的分类

- ❖ 静态算法 
- ❖ 自适应算法 
- ❖ 拓扑相关的路由算法 
- ❖ 移动节点的路由 
- ❖ **Ad-hoc**网络的路由 

拓扑相关的路由算法

❖ 分层路由



❖ 广播路由



❖ 多址传输路由选择



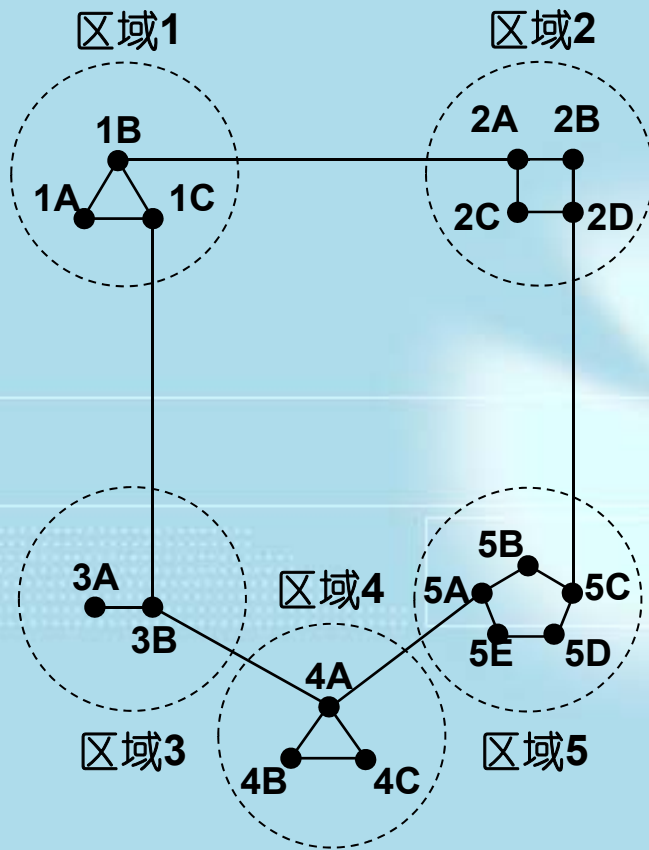
❖ **Peer-to-Peer**网的节点查找



分层路由

- ❖ 随着网络规模的增长，存储和处理路由表所需的资源也急剧增长，从拓扑上分层是解决问题的一个方法
- ❖ 分层的概念：将路由器分成组，每一路由器知道到组内任何一台路由器的路由，以及到其他组的路由，因此可把其他组中所有的路由器抽象成一个，以减少路由表的长度

分层实例



Tnbm P367 Fig. 5-15 分层路由

不分层时1A的路由表

| 目的地 | 下一跳 | 跳数 |
|-----|-----|----|
| 1A | -- | -- |
| 1B | 1B | 1 |
| 1C | 1C | 1 |
| 2A | 1B | 2 |
| 2B | 1B | 3 |
| 2C | 1B | 3 |
| 2D | 1B | 4 |
| 3A | 1C | 3 |
| 3B | 1C | 2 |
| 4A | 1C | 3 |
| 4B | 1C | 4 |
| 4C | 1C | 4 |
| 5A | 1C | 4 |
| 5B | 1C | 5 |
| 5C | 1B | 5 |
| 5D | 1C | 6 |
| 5E | 1C | 5 |

分层时1A的路由表

| 目的地 | 下一跳 | 跳数 |
|-----|-----|----|
| 1A | -- | -- |
| 1B | 1B | 1 |
| 1C | 1C | 1 |
| 2 | 1B | 2 |
| 3 | 1C | 2 |
| 4 | 1C | 3 |
| 5 | 1C | 4 |

分层的层数

- ❖ **Kamoun和Kleinrock发现：** N 个路由器的最优层次数是 $\ln N$ ，每个路由器需要 $e \ln N$ 个路由表项

拓扑相关的路由算法

- ❖ 分层路由 
- ❖ 广播路由 
- ❖ 多址传输路由选择 
- ❖ **Peer-to-Peer**网的节点查找 

广播路由

❖ 逐个向所有节点分别发送报文

缺点：发送量大

需知道网上所有节点的地址

❖ 扩散法

缺点：流量大，消耗大量带宽

某些节点还可能收到重复的报文

广播路由算法

❖ 多目的地路由



❖ 生成树算法



❖ 逆向路径传送



多目的地路由

- ❖ 分组中包含需到达的多个目的地的地址表
- ❖ 到一个节点时，路由器检查所有的目的地址表，确定输出线路集合，路由器为每一条输出线路复制一个新的分组，每个分组中仅含有要用此线路的目的地址表

优点：流量小，节约带宽

缺点：费用承担不公平

广播路由算法

❖ 多目的地路由



❖ 生成树算法



❖ 逆向路径传送



生成树算法

❖ 路由器将分组沿生成树发送（除进入线路之外）

优点：带宽得到最佳利用

缺点：每个路由器必须知道其可用生成树

如链路状态路由算法可得到生成树，距离矢量路由算法却不能得到

广播路由算法

❖ 多目的地路由



❖ 生成树算法

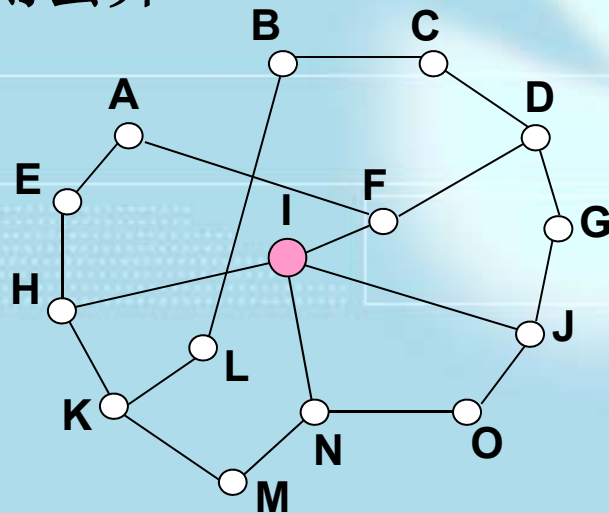


❖ 逆向路径传送

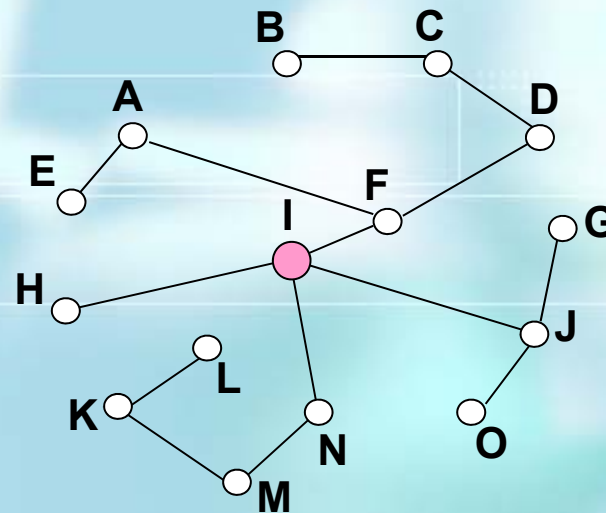


逆向路径传送

- ❖ 基本原理：当某一广播分组到达路由器时，路由器对它进行检查，如该分组来自通常向广播源发送分组的线路，则将该分组转发到除进线以外的其它线路，否则丢弃



(a) 一个子网



(b) 一棵汇集树

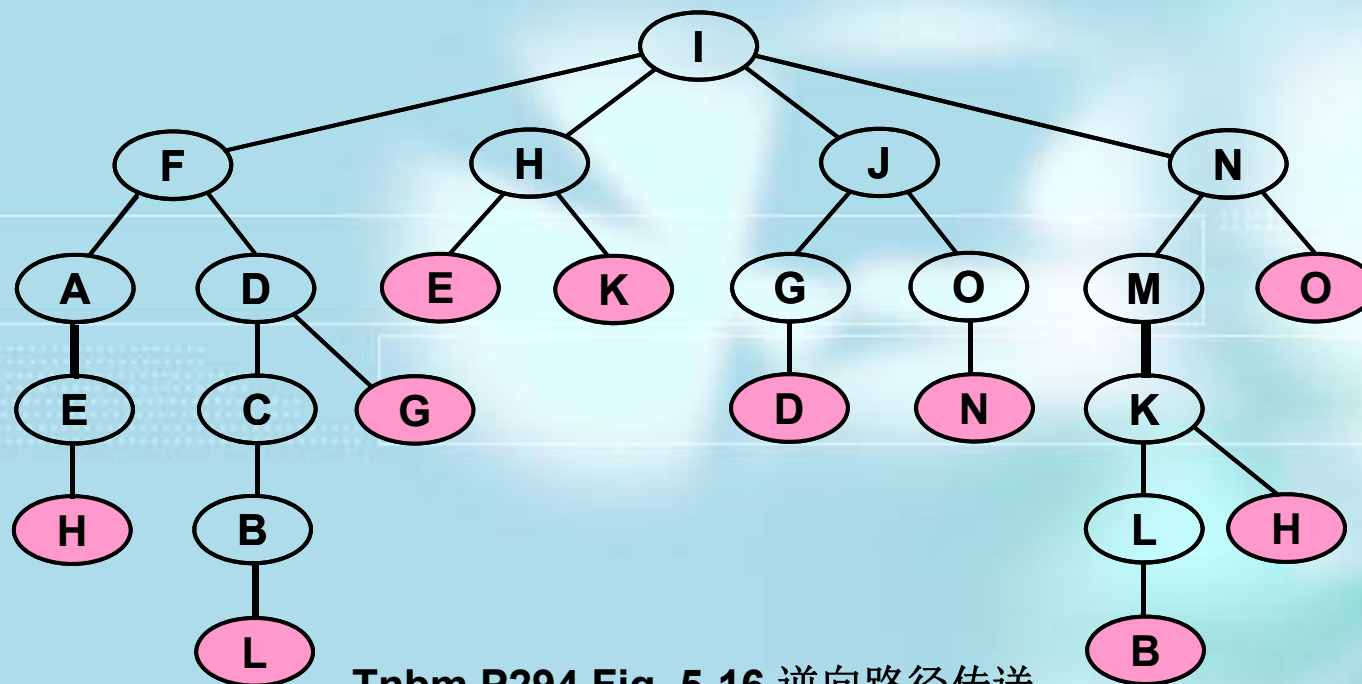
逆向路径传送说明

在(a)的子网中，每个节点都已生成了一张路由表，假设当前每个节点的路由表中到节点I去的路径中的下一跳分别为：

| 当前节点 | A | B | C | D | E | F | G | H | J | K | L | M | N | O |
|----------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 到I去的下一节点 | F | C | D | F | A | I | J | I | I | M | K | N | I | J |

逆向路径传送说明 (续1)

- ❖ 根据上述逆向路径转发的原则，可构造如下一棵树



Tnbm P294 Fig. 5-16 逆向路径传送
(c) 由逆向路径转发构造的树

逆向路径传送说明（续2）

- ❖ 如节点**F**收到一个从**I**来的分组，则立即向节点**A**和节点**D**转发
- ❖ 当节点**D**收到一个经**F**来的节点**I**的分组，它意识到如它有分组要送给**I**的话，是走节点**F**的，所以立即向节点**C**和**G**转发
- ❖ 当节点**G**收到一个经**D**来的节点**I**的分组，它意识到如它有分组要送给**I**的话，是走节点**J**而不是走节点**D**的，所以它不再转发

所以：经过**5**个站点共**24**个分组的转发后，广播算法结束

优点：算法合理，容易实现

缺点：对大型网络不适用

拓扑相关的路由算法

- ❖ 分层路由 
- ❖ 广播路由 
- ❖ 多址传输路由选择 
- ❖ **Peer-to-Peer**网的节点查找 

多址传输路由选择

(Multicast Routing)

❖ 多址传输路由选择 使用IP的D类地址

| | | | | | |
|----|---------|--------|----|----|-------|
| | | 8 | 16 | 31 | |
| A类 | 0 | 前缀 | 后缀 | | 大规模网络 |
| B类 | 1 0 | 前缀 | 后缀 | | 中规模网络 |
| C类 | 1 1 0 | 前缀 | 后缀 | | 小规模网络 |
| D类 | 1 1 1 0 | 多址传送地址 | | | |
| E类 | 1 1 1 1 | 保留将来使用 | | | |

❖ 生成树法

❖ 核心树(core-based tree)

Tnbn P437 Fig. 5-55
IP地址格式



生成树法

- ❖ 组中的每个成员都必须以自己作为出发点建立一棵生成树，根据自己所在小组对生成树进行修剪，得到一棵本小组修剪过的生成树，并告诉同组的其他成员
 - ❖ 多址传输时仅沿相应的生成树转发
- 缺点：存储量大
- 如一个网络有 n 个组，每个组平均有 m 个成员，则要存储 $m*n$ 棵生成树

多址传输路由选择

(Multicast Routing)

❖ 多址传输路由选择 使用IP的D类地址

| | | | | | |
|----|---------|--------|----|----|-------|
| | | 8 | 16 | 31 | |
| A类 | 0 | 前缀 | 后缀 | | 大规模网络 |
| B类 | 1 0 | 前缀 | 后缀 | | 中规模网络 |
| C类 | 1 1 0 | 前缀 | 后缀 | | 小规模网络 |
| D类 | 1 1 1 0 | 多址传送地址 | | | |
| E类 | 1 1 1 1 | 保留将来使用 | | | |

❖ 生成树法

❖ 核心树(core-based tree)

Tnbn P345 Fig. 5-53
IP地址格式

核心树(core-based tree)

- ❖ 每个组只有一棵生成树，它的树根靠近组的中心部位，要发送一个分组给这个组成员，则发给树根，由树根将该分组沿核心树发往各成员，这样，每组只有一棵核心树，节约了存储空间

拓扑相关的路由算法

- ❖ 分层路由 
- ❖ 广播路由 
- ❖ 多址传输路由选择 
- ❖ **Peer-to-Peer**网的节点查找 

Peer-to-Peer网中节点的查找

- ❖ 对等网：每个站点既是客户器又是服务器
- ❖ 对等网种类：
 - 中心化拓扑：用中心化的目录系统
 - 全分布非结构化的**P2P**：采用随机洪泛发现
 - 半分布结构：采用层次性的结构，用一些超级节点记录其他结点的信息
 - 全分布结构化的**P2P**

结构化的P2P-- Chord算法

假设:

- ❖ 系统中有 n 个用户
- ❖ 每个用户都有可提供的信息资源，并且都已作了索引供其他用户使用
- ❖ 每个用户都有一个IP地址，并可被散列成 m 位的一个数字，称为节点标识符（Chord用SHA-1算法来散列），节点标识符为 $0 \sim 2^m-1$
- ❖ 所有 2^m 个标识符按递增次序链接成一个环，而不管节点是否存在
- ❖ 定义函数**successor(k)**: 找出节点 k 后面的第一个存在节点

Chord算法（续）

- ❖ 信息资源名称(**name**)同样被散列为一个**m**位的数字，称为键值**key** (**key=hash(name)**)
- ❖ 信息索引的存储：信息的索引在所有节点上是随机分布的，当一个节点要提供信息**name**时，它首先构造一个二元组 (**name**, **IP地址**)，然后调用 **successor(hash(name))**去存储该二元组，不同节点上的同名信息，其索引将被存在同一节点
- ❖ 信息的查找：当某个用户需要查找名称为**name**的信息时，向**successor(key)**发一个请求，要求返回拥有信息**name**的节点的**IP地址**

Chord算法优化

- ❖ 每个节点保存其直接前趋和直接后继，那么请求可以顺时针传递，也可以逆时针传递，可以在两者之间选择一条较短的路径
- ❖ 每个节点保存一张表，称为**finger table**，该表有**m**个表项，编号从**0**到**m-1**

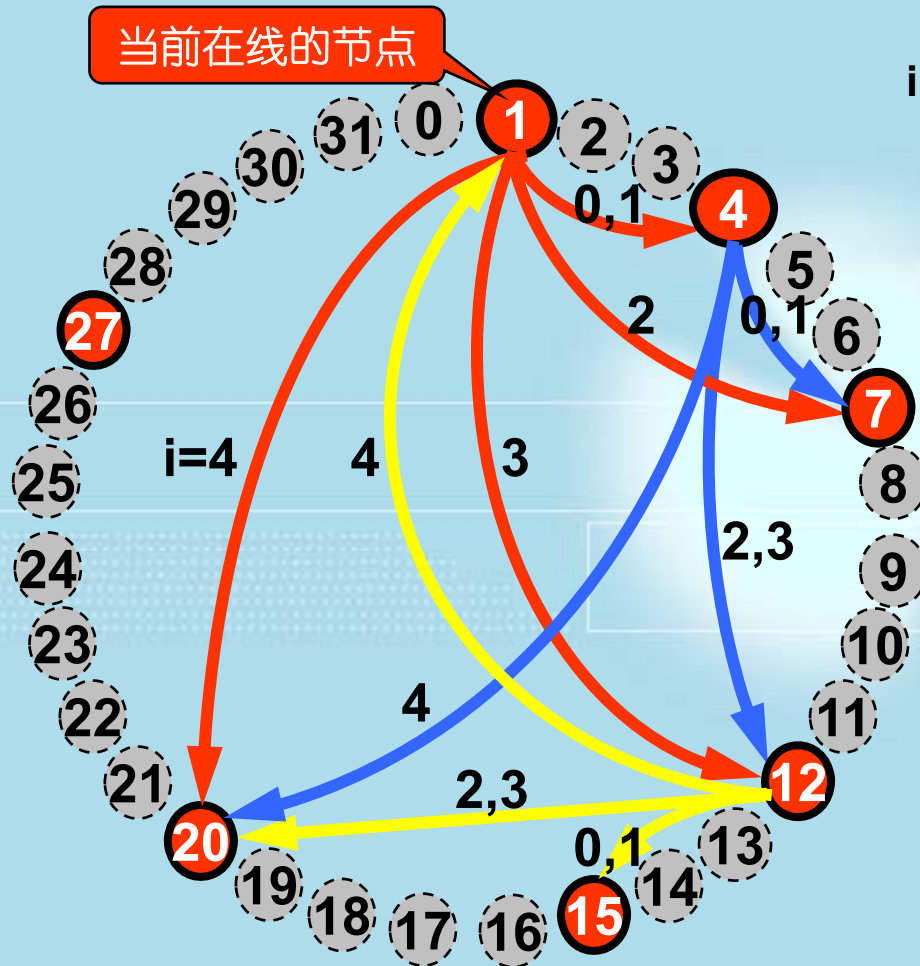
Chord算法优化 (续1)

❖ finger table表的内容

每个表项指向一个实际存在的节点，每个表项有两个字段：**start** 和 **successor(start)** 的IP地址，节点**k**中第**i**个表项的值为：

$$\left\{ \begin{array}{l} \text{Start} [i] = [k + 2^i] \bmod (2^m) [i=0\dots m-1] \\ \text{Successor} (\text{start} [i]) \text{ 的IP地址} \end{array} \right.$$

Chord算法实例



| | | | | | | | | |
|---------|----|----|----|----|----|----|----|----|
| $i = 0$ | 2 | 4 | 5 | 7 | 8 | 12 | 13 | 15 |
| 1 | 3 | 4 | 6 | 7 | 9 | 12 | 14 | 15 |
| 2 | 5 | 7 | 8 | 12 | 11 | 12 | 16 | 20 |
| 3 | 9 | 12 | 12 | 12 | 15 | 15 | 20 | 20 |
| 4 | 17 | 20 | 20 | 20 | 23 | 27 | 28 | 1 |

节点1 节点4 节点7 节点12

| | | | | | | |
|---------|----|----|----|----|----|----|
| $i = 0$ | 16 | 20 | 21 | 27 | 28 | 1 |
| 1 | 17 | 20 | 22 | 27 | 29 | 1 |
| 2 | 19 | 20 | 24 | 27 | 31 | 1 |
| 3 | 23 | 27 | 28 | 1 | 3 | 4 |
| 4 | 31 | 1 | 4 | 4 | 11 | 12 |

节点15 节点20 节点27

Tnbn P382 Fig. 5-24 Chord算法实例

Chord算法优化 (续2)

- ❖ 在节点**k**上查找键值**key**的过程：
 - 如果**key = k**，那么包含**key**信息的节点是**k**，查找结束。
 - 如果**k < key < successor (k)**之间，那么包含**key**信息的节点是**successor (k)**，查找结束
 - 否则，查找**finger**表，找出小于**key**但最接近**key**的**start**值，并直接发一请求给**successor(start)**的IP地址，请求它继续查找

Chord算法实例 (续)

| | | | | | | | | | | | | | | |
|-------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| i = 0 | 2 | 4 | 5 | 7 | 8 | 12 | 13 | 15 | 16 | 20 | 21 | 27 | 28 | 1 |
| 1 | 3 | 4 | 6 | 7 | 9 | 12 | 14 | 15 | 17 | 20 | 22 | 27 | 29 | 1 |
| 2 | 5 | 7 | 8 | 12 | 11 | 12 | 16 | 20 | 19 | 20 | 24 | 27 | 31 | 1 |
| 3 | 9 | 12 | 12 | 12 | 15 | 15 | 20 | 20 | 23 | 27 | 28 | 1 | 3 | 4 |
| 4 | 17 | 20 | 20 | 20 | 23 | 27 | 28 | 1 | 31 | 1 | 4 | 4 | 11 | 12 |

节点1 节点4 节点7 节点12 节点15 节点20 节点27

例1：在节点1上查找key=3 (key=hash(name)=3)

对于节点1，successor(1)=4，key在(1,4)中，返回4，即资源名为name的索引在节点4上

例2：在节点1上查找key=14

由于key不在(1,4)中，于是查节点1的finger table，小于14并最接近14的节点是9，successor(9)=12；于是到节点12查找，successor(12)=15，key在(12,15)中，于是返回15

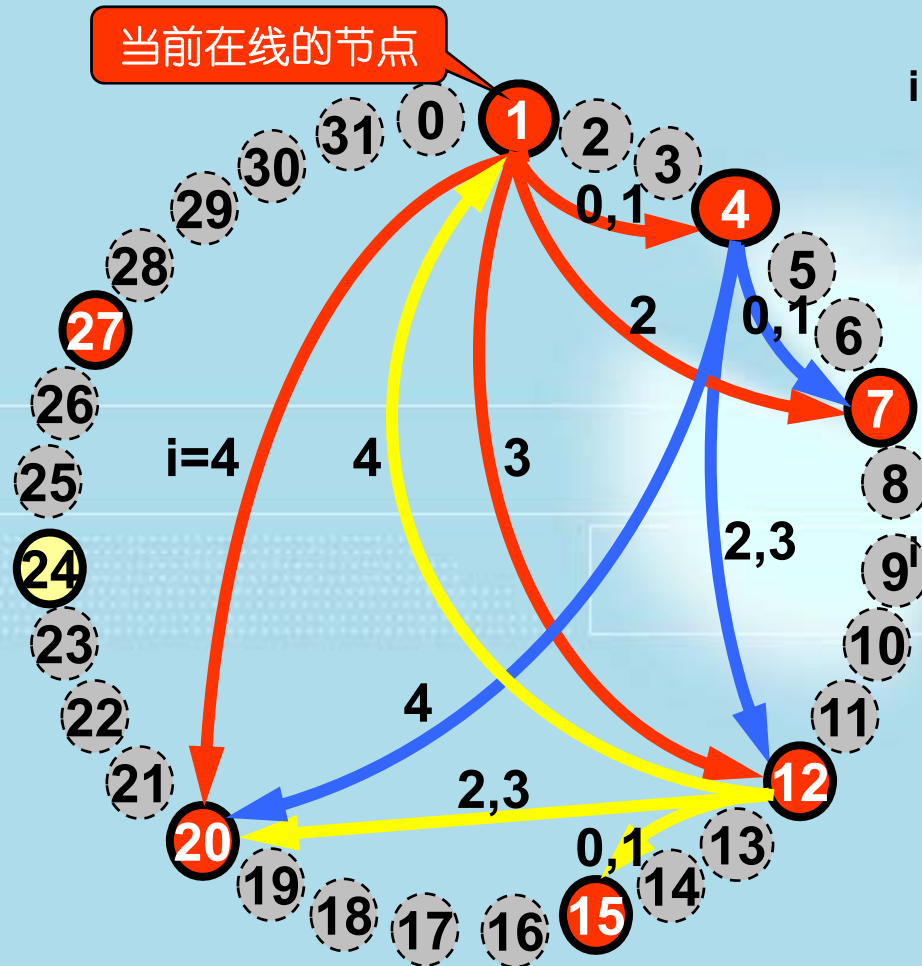
例3：在节点1上查找key=16

key不在(1,4)中，于是查节点1的finger table，小于16并最接近16的节点是9，successor(9)=12，于是到节点12查找，successor(12)=15，key不在(12,15)中，于是查节点12的finger table，小于16并最接近16的节点是14，successor(14)=15，于是到节点15查找，successor(15)=20，key在(15,20)中，于是返回20，即所查找的资源的索引在节点20上

节点的动态添加和删除

- ❖ 初始化：假设开始时节点很少，因此通过节点之间直接交换信息建立初始的环和**finger**表
- ❖ 添加节点**r**:
 - 向任一节点询问**successor(r)**的地址**s**
 - 向**s**询问它的直接前趋**p**
 - 向**s**和**p**申请加入环
 - 节点**s**将**p+1**到**r**的信息转储到**r**上，添加即告结束
 - 其他**finger**表中的问题由定时执行一个后台进程完成
- ❖ 删除节点**r**
 - 将**r**的信息转储到后继**s**
 - 通知前趋**p**，它的后继变为**s**

节点的动态添加举例



Chord算法实例中添加节点24






| | | | | | | | | |
|-------|----|----|----|----|----|----|----|----|
| i = 0 | 2 | 4 | 5 | 7 | 8 | 12 | 13 | 15 |
| 1 | 3 | 4 | 6 | 7 | 9 | 12 | 14 | 15 |
| 2 | 5 | 7 | 8 | 12 | 11 | 12 | 16 | 20 |
| 3 | 9 | 12 | 12 | 12 | 15 | 15 | 20 | 20 |
| 4 | 17 | 20 | 20 | 20 | 23 | 24 | 28 | 1 |

节点1 节点4 节点7 节点12

| | | | | | | | | |
|-------|----|----|----|----|----|----|----|----|
| i = 0 | 16 | 20 | 21 | 24 | 25 | 27 | 28 | 1 |
| 1 | 17 | 20 | 22 | 24 | 26 | 27 | 29 | 1 |
| 2 | 19 | 20 | 24 | 24 | 28 | 1 | 31 | 1 |
| 3 | 23 | 24 | 28 | 1 | 0 | 1 | 3 | 4 |
| 4 | 31 | 1 | 4 | 4 | 8 | 12 | 11 | 12 |

节点15 节点20 节点24 节点27

路由算法的分类

- ❖ 静态算法 
- ❖ 自适应算法 
- ❖ 拓扑相关的路由算法 
- ❖ 移动节点的路由 
- ❖ **Ad-hoc**网络的路由 

移动IPv4 (rfc2002)

- ❖ 移动IP技术，是移动用户在跨网络随意移动和漫游中，不用修改计算机配置的IP地址，享有原网络中一切权限。
- ❖ 支持主机移动或漫游的协议是Mobile IP协议
- ❖ Mobile IP协议中定义了三种功能实体：
 - 移动节点：一个移动的计算机，它改变位置时无需更改其IP设置，仍然可以通过其固定的IP地址进行通信。
 - 主代理：一个移动节点本地网络的主机或路由器，它保存有移动节点的位置信息，当移动节点离开主网络时能够将发往移动节点的数据包传给移动节点。
 - 客代理：移动节点当前所在的外地网络上的一个主机，它能够把由主代理送来的数据包转发给移动节点。

移动IP的工作机制

- ❖ 代理周期性地发送广播，宣告他们与链路之间的关系
- ❖ 移动节点收到这些广播后，判断自己是在本地网还是在其他网。
- ❖ 如在其他网
 - 移动节点向客代理注册，递交自己的**IP**地址，**MAC**地址和一些安全信息
 - 客代理与主代理联系，报告自己的网络地址
 - 主代理告诉客代理接收此消息
 - 客代理保存此移动主机的信息

数据包的转发

当有一个包发给移动主机时

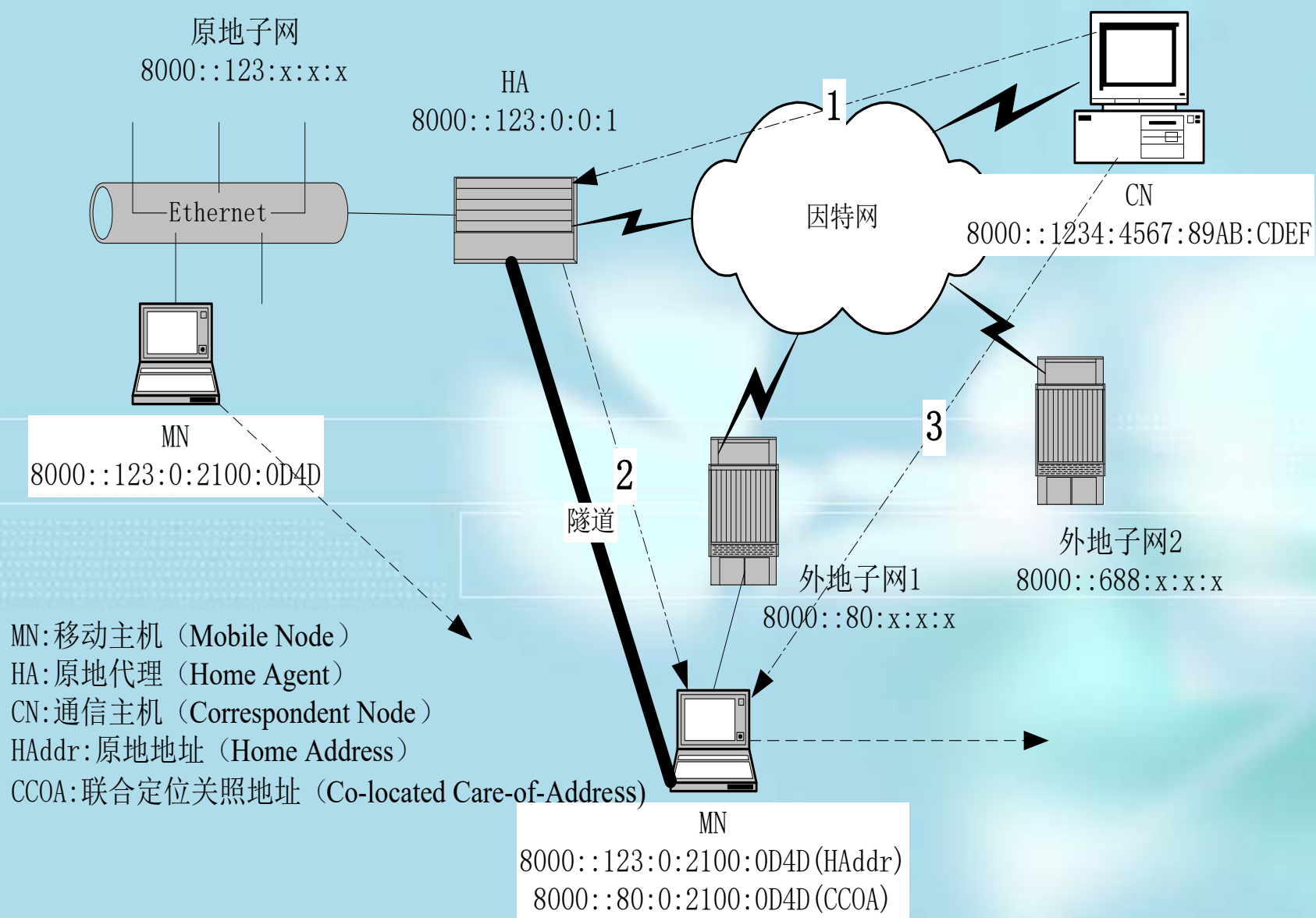
- ❖ 首先，该包会路由到主网络
- ❖ 主代理查找到了移动主机的新住址，以及客代理的地址。
 - 把此包再封装在一个**IP**包中，发给客代理
 - 告诉发送者将包直接发给客代理
- ❖ 客代理解封装数据包，把此数据包用数据链路层协议传给移动主机

安全问题






- ❖ 采用优化路由的主要障碍是安全问题。
- ❖ 当通信对端直接通过隧道与移动节点通信时，对端必须知道移动节点当前的优化地址，这个问题与移动IP注册相似。移动IP注册的目的就是通知家乡代理移动节点当前的转交地址。一个“坏家伙”只需送一个伪造的注册给通信对端就可以中断移动节点和对端的通信，这对移动IP中任何试图优化路由的努力来说都是一个挑战。

移动IPv6

- ❖ 无客代理
- ❖ 同时支持隧道和源路由技术
- ❖ 主要内容
 - 移动主机在外地网上获取一个转交地址
 - 将转交地址通知主代理和他的通信伙伴
 - 知道转交地址的伙伴和直接发送信息
 - 不知道转交地址的伙伴可发送给主代理，由主代理再发送到转交地址

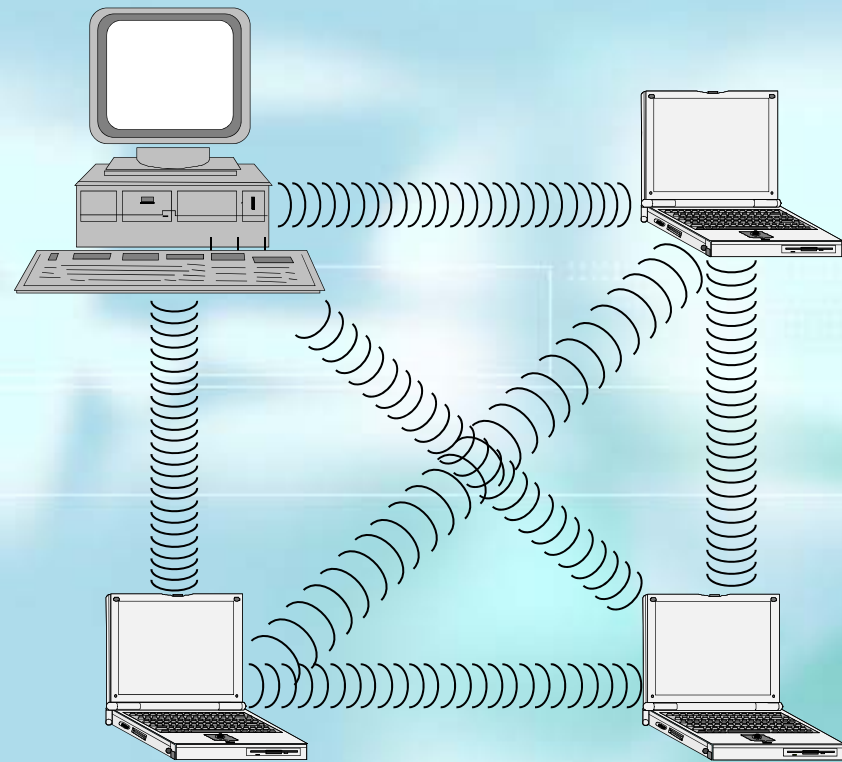


路由算法的分类

- ❖ 静态算法 
- ❖ 自适应算法 
- ❖ 拓扑相关的路由算法 
- ❖ 移动节点的路由 
- ❖ **Ad-hoc**网络的路由 

Ad-Hoc模式

- ❖ 早期的**Ad-Hoc**模式用于多个移动主机（无线站点）之间的直接通信，毋需接到有线网络，每个站点必须“看”到其它所有站点
- ❖ 现在的**Ad-Hoc**，移动主机并不要求相互都能“看”到，但，源主机到目的主机之间至少存在一条通路，途中的站点（移动主机）同时也将兼作路由器用，也可以与有线网络连接



Ad-Hoc的路由

- ❖ 主动路由（表驱动）：每个节点都周期性广播路由信息，主动地发现并维护路由表

 - **DSDV (Destination Sequenced Distance Vector)** 

- ❖ 按需路由（事件驱动）：只有在需要发送数据时，源站点才寻找路由

 - **AODV (Ad Hoc On-demand Distance Vector)** 

 - **DSR (Dynamic Source Routing)** 

主动路由 DSDV

- ❖ **DSDV (Destination Sequenced Distance Vector)** 是一种采用距离矢量算法的路由协议
- ❖ 网络中每个节点都维护一张路由表，路由表包含所有可能的目的节点、距离信息以及下一跳等
- ❖ 每个节点都周期地广播其全部的路由信息（对于拓扑变化相对较慢的网络只需广播其更新的路由信息），每个节点都依据所收到的信息来维护它们的路由表
- ❖ **DSDV**算法依赖于每个节点路由信息周期性的广播，在**Ad-Hoc**网络中，由于其网络拓扑是动态变化的，维护一张路由表意味着耗用大量的带宽和**CPU**资源

Ad-Hoc的路由

- ❖ 主动路由（表驱动）：每个节点都周期性广播路由信息，主动地发现并维护路由表

 - **DSDV (Destination Sequenced Distance Vector)** 

- ❖ 按需路由（事件驱动）：只有在需要发送数据时，源站点才寻找路由

 - **AODV (Ad Hoc On-demand Distance Vector)**

 - **DSR (Dynamic Source Routing)** 



按需路由AODV说明

- ❖ **AODV (Ad Hoc On-demand Distance Vector)**
- ❖ **AODV**是**DSDV**算法的改进，与**DSDV**的区别在于只是在需要时 (**on-demand**) 才寻找路由，而不必如**DSDV**一样需随时维护一张包含全部节点的路由表
- ❖ 当源节点想发送信息给某个目的节点时，如当前没有路径，则启动**Path Discovery**过程，广播一个**Route Request (RREQ)** 路径请求分组给相邻节点，收到此请求分组的邻节点再转发给它的相邻节点并建立到源节点的反向路径，直到一个知道目的节点路由信息的中间节点或目的节点本身，然后回复**Route Reply (RREP)** 路径应答包给源节点
- ❖ **AODV**假设每一条链路都是双向对称的 (**Symmetric**)

按需路由AODV算法

- ❖ **AODV的请求分组和应答分组** 
- ❖ **AODV的Path Discovery过程** 
- ❖ **AODV的路径维护** 

AODV中的请求分组和应答分组

❖ AODV中的请求分组

| Source Add | Request ID | Destination Add | Source Seq. # | Destination Seq. # | Hop Count |
|------------|------------|-----------------|---------------|--------------------|-----------|
|------------|------------|-----------------|---------------|--------------------|-----------|

Tnbn P377 Fig. 5-21 Route Request分组的格式

❖ AODV中的应答分组

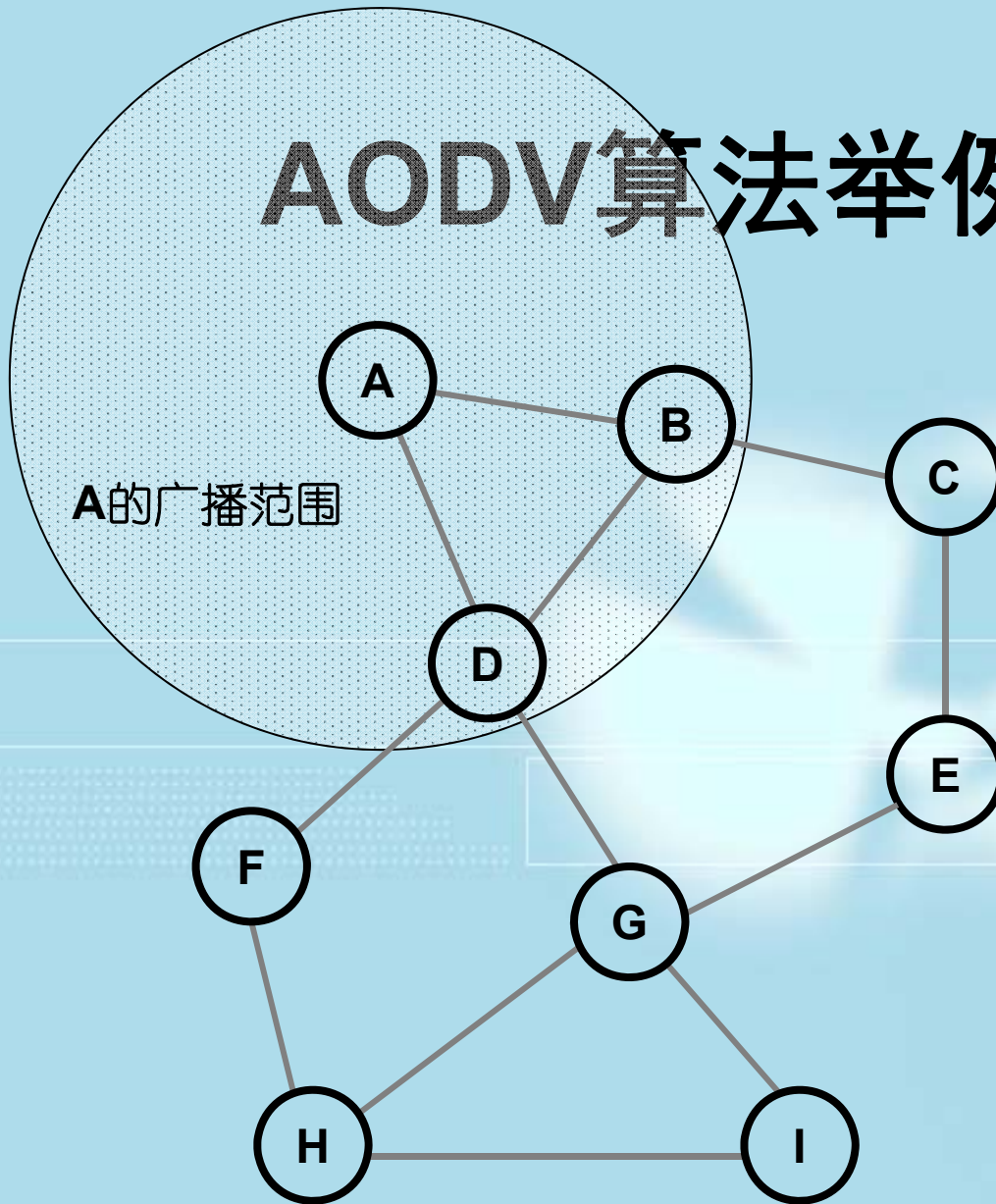
| Source Add | Destination Add | Destination Seq. # | Hop Count | LifeTime |
|------------|-----------------|--------------------|-----------|----------|
|------------|-----------------|--------------------|-----------|----------|

Tnbn P378 Fig. 5-22 Route Reply分组的格式

按需路由AODV算法

- ❖ **AODV的请求分组和应答分组** 
- ❖ **AODV的Path Discovery过程** 
- ❖ **AODV的路径维护** 

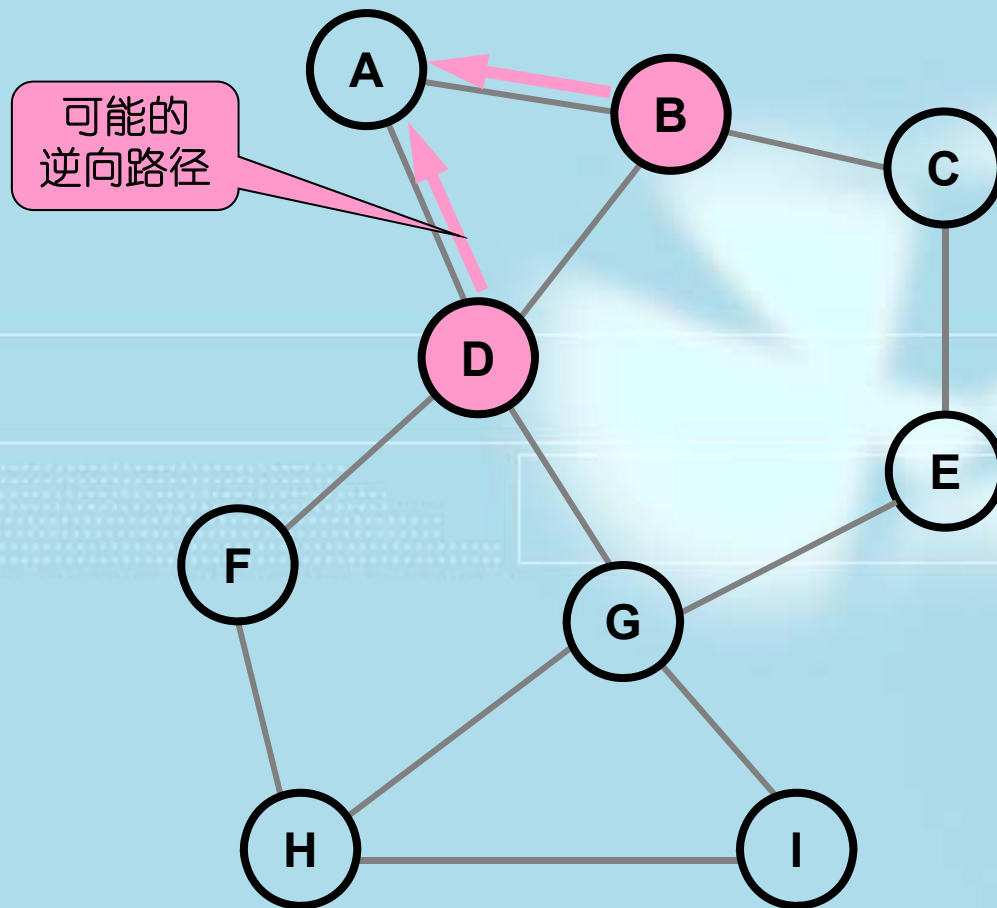
AODV算法举例—环境



Tnbnm P376 Fig.5-20(a) A的广播范围

- 图中每个节点都是一台具有路由功能的移动主机
- 图为A~I九个节点当前的位置及其相互的连接状态
- 节点A的广播范围覆盖了节点B和节点D，所以A和B以及A和D之间有连接，其它节点间的连接关系如图，并假设连接是双向的
- 当前A准备向I发送分组，并且当前没有到达I的表项
- 于是A发送一个RREQ广播分组，分组中包含源和目的地址（A和I的IP地址）

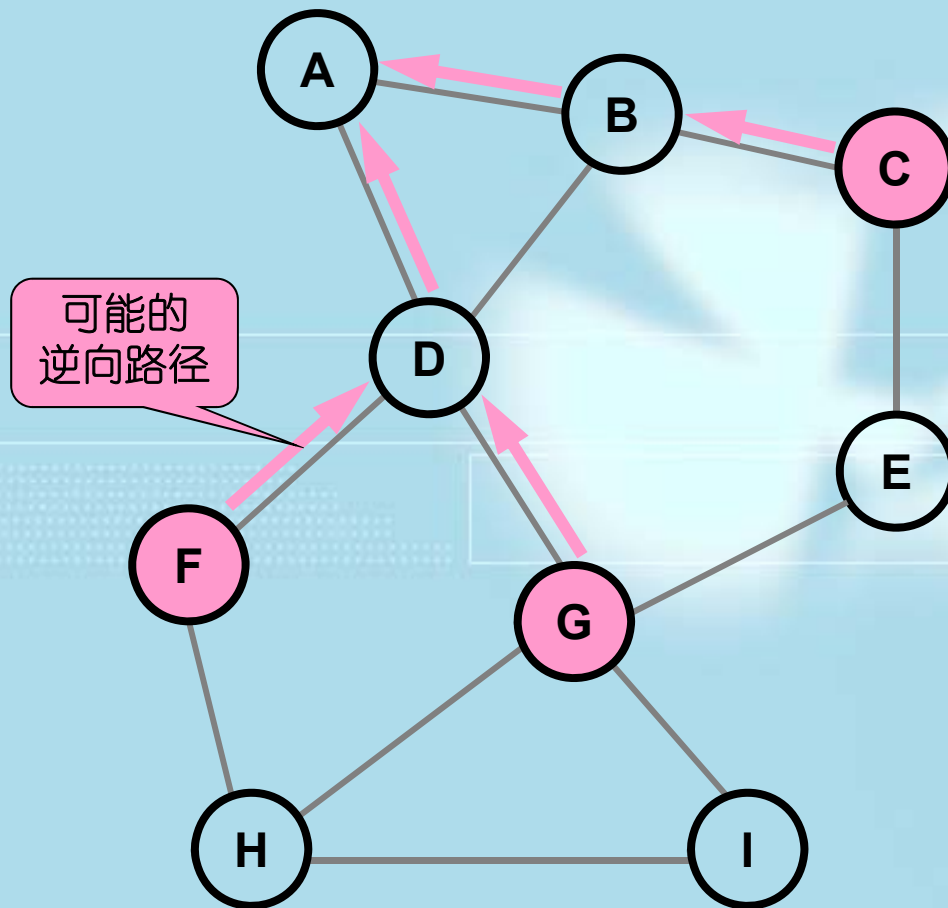
路径发现过程—Route Request1



- 当**B**和**D**接收到**A**的**RREQ**广播分组，于是查各自的路由表，首先判断是否重复，然后查找是否有较新的到达目的端的路径信息
- 如果有，则向源节点回送一个**RREP**分组
- 如果没有，则继续广播该**RREQ**分组，并建立一逆向路由的表项，以备返回的**RREP**分组途经本节点时使用，同时启动一个定时器（届时如果不是逆向路径的途经节点，则丢弃）

Tnbnm P376 Fig.5-20(b) B和C接收到A的广播信息之后

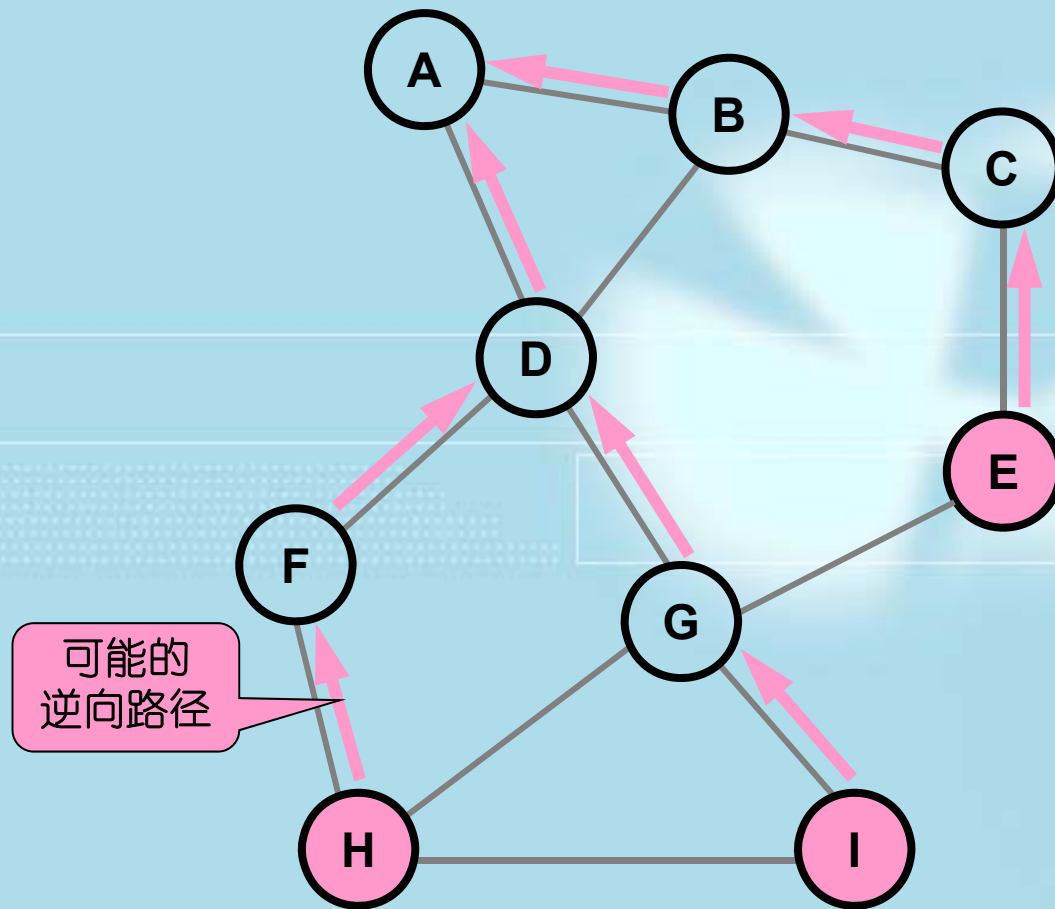
路径发现过程—Route Request2



- 当B收到D继续广播的A的RREQ分组，经查是重复的，丢弃，当F、G收到A的RREQ分组，经查不是重复的，然后查自己的路由表，假设也都没有较新的到达I的路径信息
- 所以，F和G也继续广播该RREQ分组，并建立一逆向路由的表项，同时启动一个定时器
- C和D收到B继续广播的分组，D丢弃，C的处理与F、G类似

Tnbm P376 Fig.5-20(c) C、F和G接收到A的广播信息之后

路径发现过程—Route Request3



Tnbnm P376 Fig.5-20(d) E、H和I 接收到A的广播信息之后

- 当E和H收到A的RREQ分组，也同样检查是否重复，再查自己的路由表
- E和H的处理与F和G等节点类似，即也继续广播，并建立一逆向路由的表项，同时启动一个定时器等
- I 在收到后，由于它是目的节点，所以立即创建一个RREP分组作为应答，其中源、目的地址从RREQ分组中copy，此外根据当前的计数值填写Dest.Seq#、清Hop count为0，并对Lifetime置一适当的初值
- 该RREP分组是一单播分组

路径发现过程—Route Reply

- ❖ I 所创建的**RREP**分组是一个单播的分组，将按逆向路径传送给源节点**A**
- ❖ 沿途的节点（本例中的**G**、**D**）都将免费得到一条到达**I**的路径（如果原来没有到达**I**的路径则添加，如果有则替代或更新）
- ❖ 非沿途的中间节点（本例中的**B**、**C**、**E**、**F**和**H**），在定时器**TimeOut**后，原暂时保留的逆向路径将丢弃

路径发现过程的优化

- ❖ **AODV**算法中的**RREQ**分组采用的是广播方式，所以会产生大量的广播分组
- ❖ 如果将**RREQ**分组的**TTL**（**Time to Life**）限定在一个有限的网络半径之内，将是有效的
- ❖ 如果发送方在生成**RREQ**分组时，先把**TTL**设为**1**，如在合理的时间内没有**RREP**分组返回，则再生成下一个**RREQ**分组，并把**TTL**逐次设为**2**、**3**、**4... ..**，将有效地限制广播分组的传输范围

按需路由AODV算法

- ❖ **AODV的请求分组和应答分组** 
- ❖ **AODV的Path Discovery过程** 
- ❖ **AODV的路径维护** 

AODV的路径维护

- ❖ 每个节点都定期广播一个**HELLO**消息，并期待其邻居节点的应答，如果没有应答，则说明该邻居节点已经不再有效
- ❖ 如果某个邻居节点无效，必须及时更新自己的路由表
- ❖ 如果无效的邻居节点是活动邻居节点，那么还必须检查路由表中与活动邻居节点相关的路径，并通知相关节点（递归）
- ❖ 活动邻居：在最近的 ΔT 时间内曾经给它发送过到达该目的节点的分组

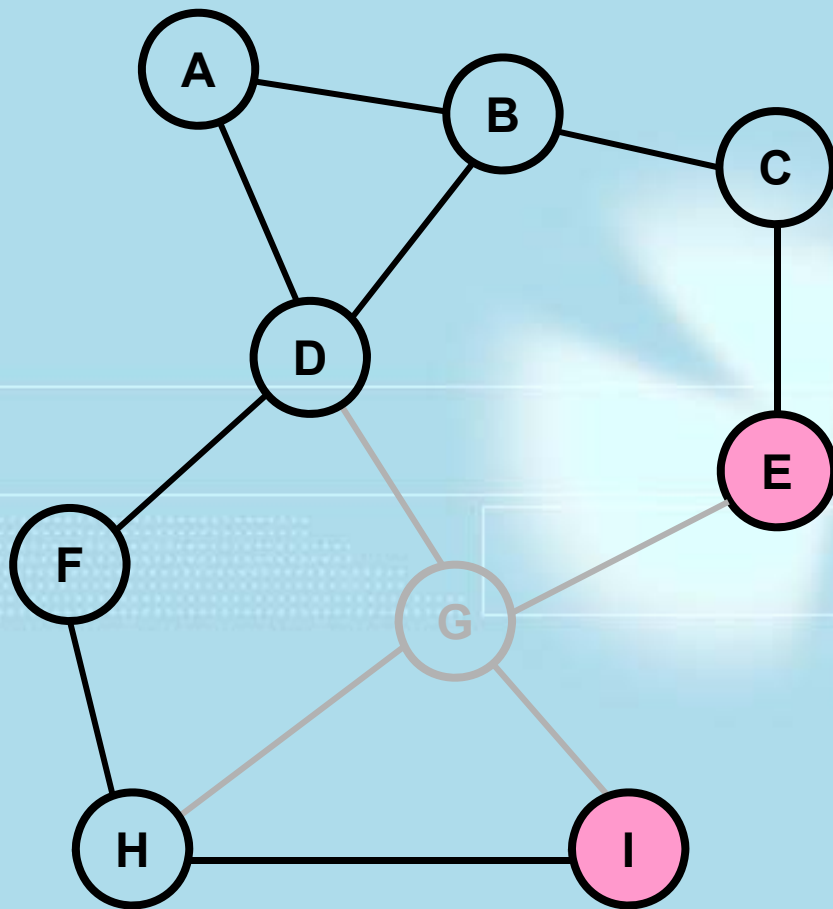
AODV的路径维护

❖ 以节点**D**的路由表为例

| Dest. | Next hop | Distance | Active Neighbors | Other fields |
|-------|----------|----------|------------------|--|
| A | A | 1 | F、G |  |
| B | B | 1 | F、G | |
| C | B | 2 | F | |
| E | G | 2 | | |
| F | F | 1 | A、B | |
| G | G | 1 | A、B | |
| H | F | 2 | A、B | |
| I | G | 2 | A、B | |
| I | G | 2 | A、B | |

Tnbn P379 Fig.5-23(a) 在G停机之前D的路由表

AODV的路径维护




- ❖ **D**发现其邻居节点**G**已不复存在
- ❖ 根据路由表得知，到目的节点**E**、**G**、**I**的下一跳是节点**G**，所以在**D**的路由表中删除目的地址为**E**、**G**、**I**的表项
- ❖ 同时，**D**知道节点**A**和**B**的某些路径依赖节点**G**，所以立即通知节点**A**和**B**

Tnbnm P379 Fig.5-23(b) 在G停机之后的拓扑图

Ad-Hoc的路由

- ❖ 主动路由（表驱动）：每个节点都周期性广播路由信息，主动地发现并维护路由表

 - **DSDV (Destination Sequenced Distance Vector)** 

- ❖ 按需路由（事件驱动）：只有在需要发送数据时，源站点才寻找路由

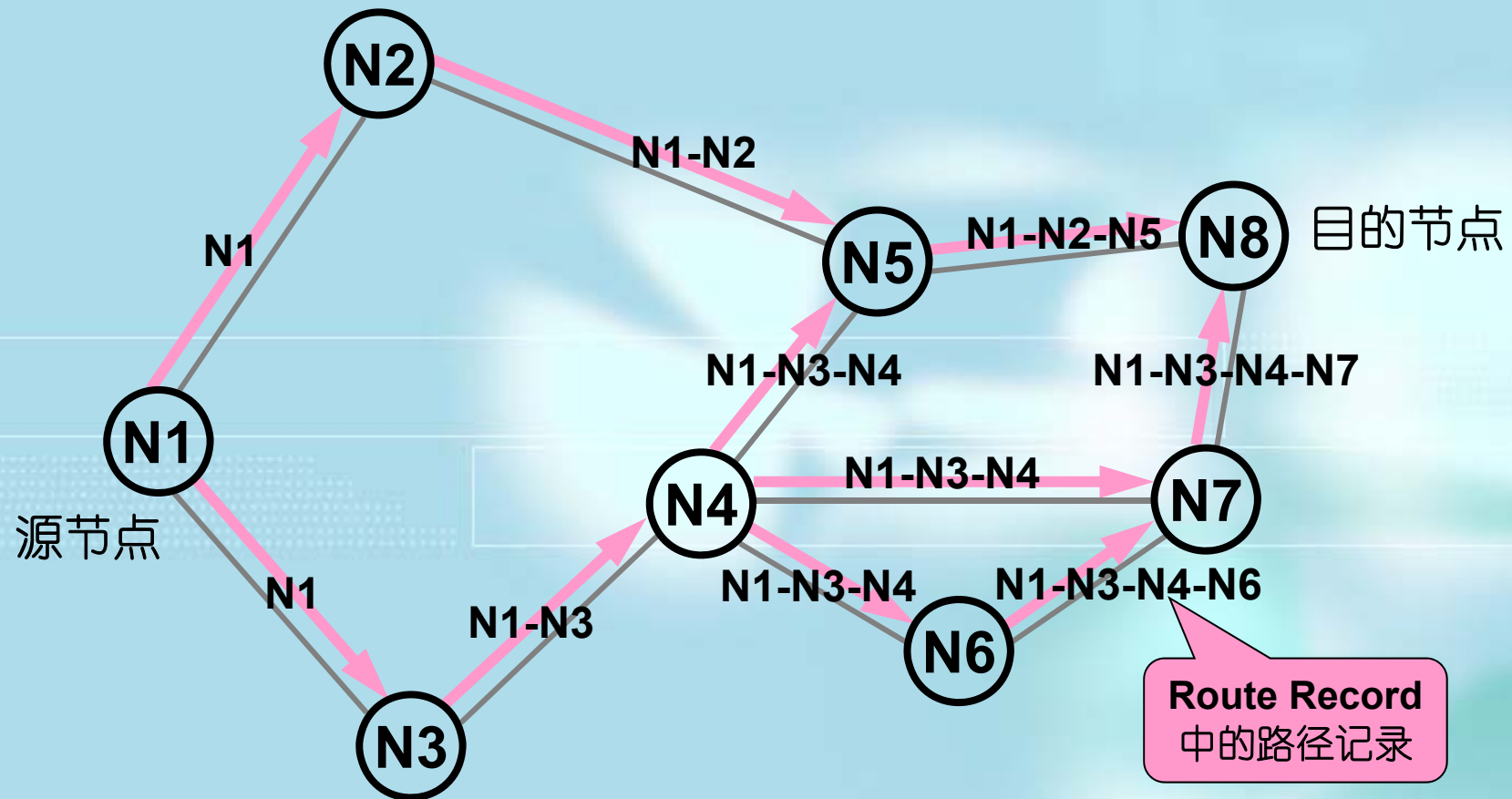
 - **AODV (Ad Hoc On-demand Distance Vector)** 

 - **DSR (Dynamic Source Routing)** 

按需路由DSR算法

- ❖ **DSR (Dynamic Source Routing)**
- ❖ **DSR**也是一种按需 (**on-demand**) 的路由协议，是一种基于源路由的按需路由协议，它使用源路由算法，每个节点都有一个**Route Cache**，记录路由的路径轨迹
- ❖ 当某源节点要发送数据给某一目的节点时，首先查看自己的**Route Cache**中是否存在现成的路径可以使用（有时甚至保存有多条），如果不存在，则启动**Route Discovery**过程，即广播一个**RREQ**请求分组，其中包含目的节点地址、源节点地址、**Request ID**和一个**Route Record**（用于按顺序逐个记录此**RREQ**所途经的节点地址）

DSR的Route Discovery过程1

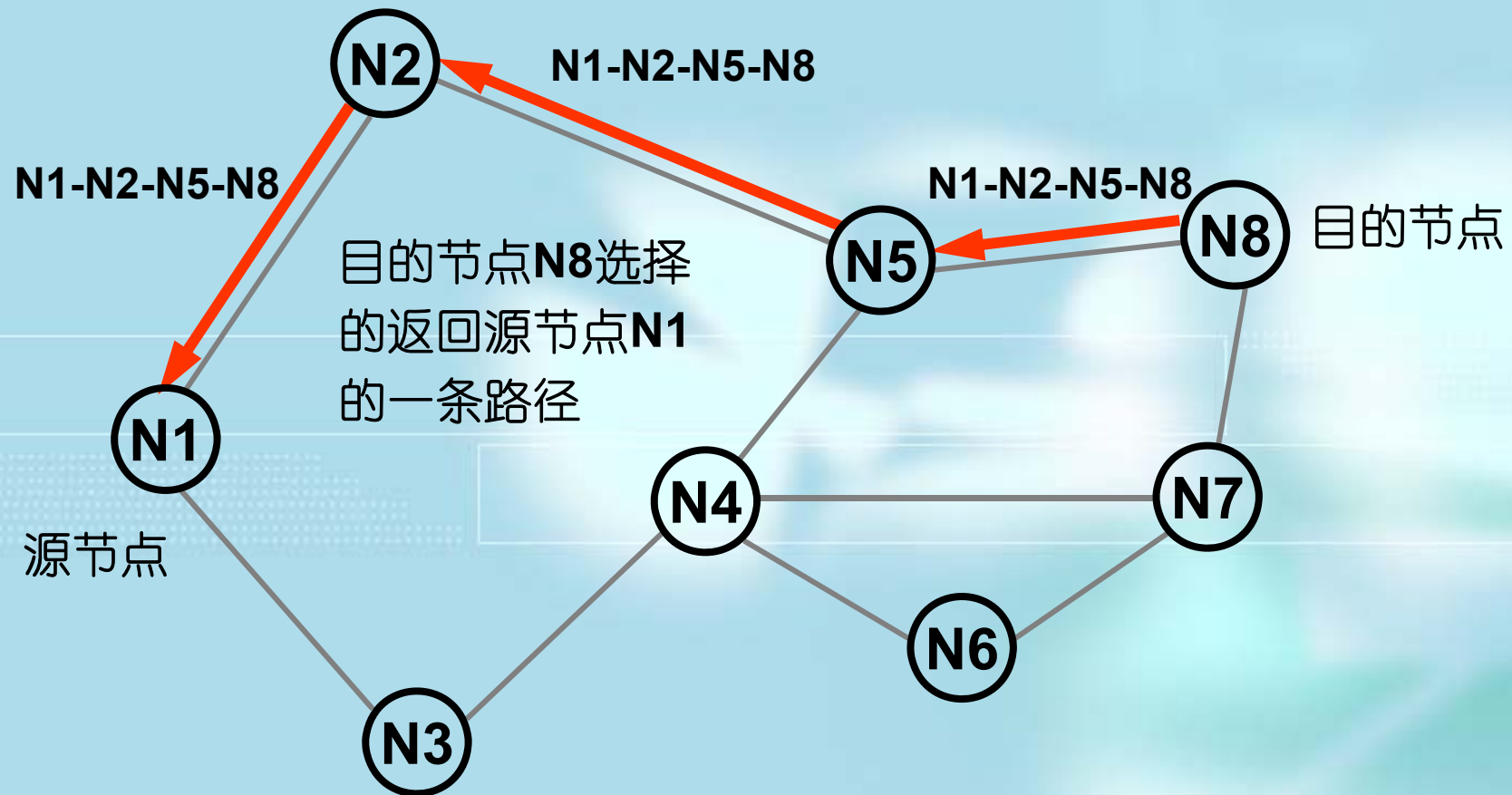


Route Discovery过程中的Route Request

当节点收到Route Request时

- ❖ 如果在此之前已收到过同一源节点、同一ID的**RREQ**（表示这是重复的），则该请求分组丢弃
- ❖ 如果在**Route Cache**中发现已经有到目的节点的路径（表示自己可提供到达目的节点的路径），则把此路径加入**Route Record**并创建 **RREP**应答分组，回复源节点并丢弃该请求分组
- ❖ 如果自己是目的节点，此时**Route Record**中记录的就是从源节点到目的节点的路径信息，把此路径加入**RREP**应答分组，回复给源节点，并丢弃该请求分组
- ❖ 否则，代表自己是中间节点，把自己的地址附加在**Route Record**中，然后再继续广播

DSR的Route Discovery过程2



Route Discovery过程中的Route Reply 过程

AODV和DSR算法的比较 1

AODV和**DSR**都有类似的**Route Discovery**过程，都有着按需建立路径的特性，都强调是在源节点需要的时候才会去建立到目的地节点的路径，但是两者仍然有相当的差别

- ❖ **AODV**假设链路是双向的；而**DSR**支持单向链路，并且**DSR**是基于源路由的

在**AODV**中，目的节点生成的**RREP**分组是按原路径返回，所以链路必须是双向的，但未必所有的链路都是双向的；在**DSR**中，目的节点所创建的 **RREP**分组是单播的，如果链路是双向的，可以按原路径返回，也可以再启动一个**Route Discovery**过程，将先前所得到的路径附在新的**RREQ**分组中，所以**DSR**支持单向链路

AODV和DSR算法的比较 2

- ❖ 在**AODV**的路由表中，每个目的节点只有一条路径记录，而在**DSR**中，采用的是**Route Cache**，所以对每个目的节点都可能保留有多条路径记录

在**AODV**中，在处理源节点的**RREQ**分组时，只处理一笔路径信息，所以目的节点也只回应最先到的**RREQ**，其余的一律丢弃；在**DSR**中，因为是使用**Route Cache**，目的节点会回应所有的**RREQ**分组，即使是重复的，这就使源节点可以在**Route Cache**中保留多条到同一目的地的路径，所以当最短路径失效时，可能还有另一个选择，这种机制减少了启动**Route Discovery**过程的机会，但是回应每一个**RREP**却也增加了网路流量

AODV和DSR算法的比较 3

- ❖ 在一次**Route Discovery** 过程中，与**AODV**相比较，**DSR**所得到的路径信息更多







DSR是基于源路由的，在**RREQ**请求分组中，除包含目的节点地址、源节点地址等以外，还包括一个**Route Record**域，用于按顺序逐个记录所途经的节点地址，所以在完成一次**Route Discovery**后，源节点可以获得目的节点和所有中间节点的路径信息，每一个中间节点也可以获得到达其他中间节点的路径信息；而**AODV**得到的路径信息则是有限的，只可得到相邻节点的路径信息，所以可能会造成节点常常启动**Route Discovery**过程，而使得网路流量增大

AODV和DSR算法的比较 4

- ❖ 路径信息的更新，和由于某个节点失效而导致的路由信息的维护，两者所采用的机制和策略有所不同

在**AODV**中，路径的时效是由**Sequence Number**来决定，并根据所得到的消息随时更新，如果过一段时间后此路径没被用过，则将从**Routing Table**中删除，此外，一旦发现某邻节点失效，发现节点将维护自己的路由表，并根据路由表查看并通知与失效节点相关的路径中所涉及到的节点；在**DSR**中，则无法判断现有的路径信息是否有效，只有等到**Route Error**分组返回时，才会把**Route Cache**中无效的路径信息删除，如果发现某邻节点失效，因为是基于源路由的，所以也只会通知唯一的上游节点，直到源节点为止，而不会通知不在此路径上的其它节点

第5章 网络层

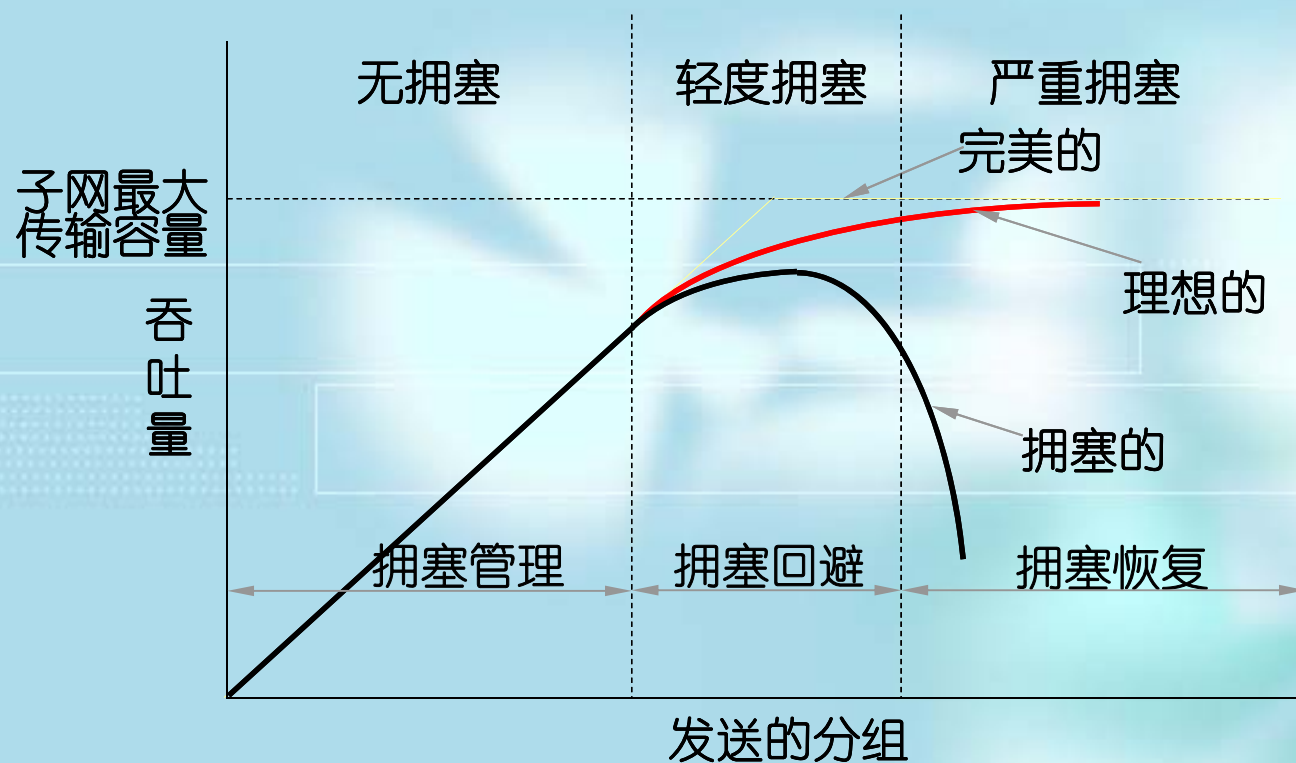
- ❖ 网络层设计的相关问题 
- ❖ 路由算法 
- ❖ 拥塞控制 
- ❖ 服务质量 
- ❖ 网络互联 
- ❖ 因特网中的网络层 

拥塞控制

- ❖ 当通信子网中有太多的分组，导致其性能降低，这种情况叫拥塞
- ❖ 拥塞控制和流量控制的区别
 - 全局性问题和局部性问题
- ❖ 造成拥塞的原因
 - 节点存储容量（缓冲区）不够
 - 处理机速度太低
 - 线路容量（带宽）不够

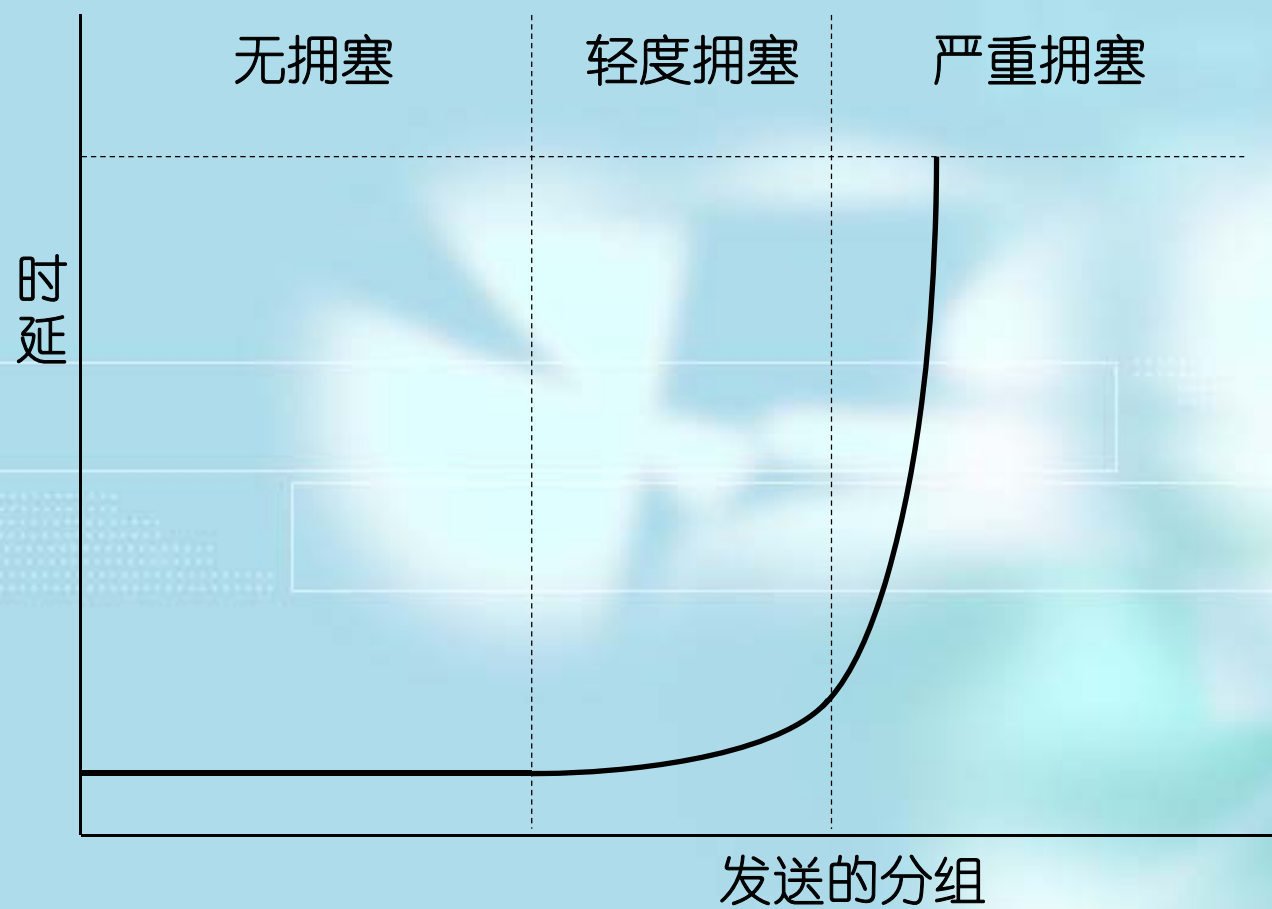
拥塞示例

❖ 当通信量太大时，发生拥塞，性能显著降低



Tnbm P385 Fig. 5-25 吞吐量增大将发生拥塞

拥塞对延迟的影响



拥塞控制原理和一般方法

- ❖ 拥塞控制的基本原理
- ❖ 拥塞预防策略
- ❖ 虚电路子网中的拥塞控制
- ❖ 数据报子网的拥塞控制
- ❖ 载荷脱落
- ❖ 抖动控制



拥塞控制的基本原理

❖ 开环控制 

❖ 闭环控制 

开环控制

- ❖ 通过良好的设计来避免问题的出现，确保问题在一开始就不会出现
- ❖ 开环控制的方法：
 - 什么时候接受新的数据报
 - 什么时候开始丢弃分组，丢弃哪些分组
 - 制定网络中各个节点的调度策略

所有这些决定与网络中的当前状态无关

拥塞控制的基本原理

❖ 开环控制 

❖ 闭环控制 

闭环控制

建立在反馈的基础上，由三部分组成：

- ❖ 监视系统，检测何时何地发生了拥塞
 - 检测的指标可以是包丢失率、平均队列长度、由于超时引起的重发包数和数据包延迟抖动等
- ❖ 将此信息传送到可能采取行动的地方
 - 直接发包给相关节点
 - 利用包中的某一**bit**将拥塞通知邻居节点
 - 每个节点周期性地发出探测报文
- ❖ 调整系统操作以更正系统

闭环控制方式

- ❖ 显式反馈：当某一节点发现拥塞时，它发一个回答帧给相关节点，通知它们网络拥塞了
- ❖ 隐式反馈：通过定时器方法，发送端每发一个帧，就启动一个定时器，如在规定的时间内没有收到相应的**ACK**，则认为该帧丢失，如丢失率相当高，则认为网络发生了拥塞

闭环控制一般不适合于高速网

因为等反馈的控制信号到达，早已时过境迁

拥塞控制原理和一般方法

- ❖ 拥塞控制的基本原理
- ❖ 拥塞预防策略
- ❖ 虚电路子网中的拥塞控制
- ❖ 数据报子网的拥塞控制
- ❖ 载荷脱落
- ❖ 抖动控制









拥塞预防策略

❖ 传输层、网络层和数据链路层的传输策略都会对拥塞产生影响

| 层 次 | 策 略 |
|-------|--|
| 传输层 | <ul style="list-style-type: none">➢重传策略➢错序保存策略➢应答策略、➢流量控制策略➢超时决定 |
| 网络层 | <ul style="list-style-type: none">➢子网是虚电路子网还是数据报子网➢数据包排队和服务策略➢数据包丢弃策略➢路由算法➢数据包生存时间管理 |
| 数据链路层 | <ul style="list-style-type: none">➢重传策略➢错序保存策略➢应答策略➢流量控制策略 |

拥塞控制原理和一般方法

- ❖ 拥塞控制的基本原理 
- ❖ 拥塞预防策略 
- ❖ 虚电路子网中的拥塞控制 
- ❖ 数据报子网的拥塞控制 
- ❖ 载荷脱落 
- ❖ 抖动控制 

虚电路子网中的拥塞控制

- ❖ 通过准入控制：一旦拥塞出现，不允许建立新的虚电路
- ❖ 如果允许建立新的虚电路，则在路由选择时要非常小心地绕过拥塞地段
- ❖ 当虚电路建立时，在主机和子网之间协商一个协议，该协议指出了新建流的量、特征、服务质量和其他参数，子网将为该数据流预留资源

拥塞控制原理和一般方法

- ❖ 拥塞控制的基本原理
- ❖ 拥塞预防策略
- ❖ 虚电路子网中的拥塞控制
- ❖ 数据报子网的拥塞控制
- ❖ 载荷脱落
- ❖ 抖动控制



数据报子网的拥塞控制

在路由器上监视每条输出线的利用率，当利用率超过某一阈值时，采取某些行动

❖ 警告位方法

这是**DECnet**和帧中继网络上采用的方法，当发生拥塞时，在分组的头部设置一位警告位，当该分组到达目的主机时，传输实体将警告位拷贝到**ACK**上，源主机能得知拥塞的发生，以调整它的发送速率

❖ 抑制分组

拥塞的路由器直接发一个抑制分组给源主机

❖ **Hop-by-Hop**阻塞包

直接发抑制分组给源主机可能反应太慢，取而代之的是发给它的前一站路由器，通知它放慢速率，前一站再发给它的前一站，一直到源主机

拥塞控制原理和一般方法

- ❖ 拥塞控制的基本原理
- ❖ 拥塞预防策略
- ❖ 虚电路子网中的拥塞控制
- ❖ 数据报子网的拥塞控制
- ❖ 载荷脱落
- ❖ 抖动控制



载荷脱落

当路由器来不及处理数据包时，就把数据包给扔掉

❖ **RED (Random Early Detection)** 随机早期检测

在情况还不太糟糕时就采取动作

- 随时观察数据包队列的长度，一旦长度超过阈值，表示拥塞即将发生，立即开始在队列中随机选取一个分组丢弃，并继续观察

❖ 拥塞发生的信息反馈

- 显式反馈：在丢包的同时，发送一个抑制分组
- 隐式反馈：丢包时不发送抑制分组，当源主机没有收到 **ACK**，则表示分组丢失，可能已发生拥塞

拥塞控制原理和一般方法







- ❖ 拥塞控制的基本原理
- ❖ 拥塞预防策略
- ❖ 虚电路子网中的拥塞控制
- ❖ 数据报子网的拥塞控制
- ❖ 载荷脱落
- ❖ 抖动控制



抖动控制

- ❖ 在音频和视频应用中应考虑抖动（**jitter**）控制
- ❖ 在每个节点上控制预期的时间，当到达得太早，则在此节点上多留一会儿，如到达得太晚，则加快处理速度
- ❖ 在接收方设置一缓冲区，将收到的数据包放入缓冲区，然后按一定的速率回放

第5章 网络层

- ❖ 网络层设计的相关问题 
- ❖ 路由算法 
- ❖ 拥塞控制 
- ❖ 服务质量 
- ❖ 网络互联 
- ❖ 因特网中的网络层 

服务质量的指标

- ❖ 从同一源到同一目的地的一串分组流（**stream**）称为流（**flow**）
- ❖ 流的服务质量有四个指标
 - 可靠性
 - 延迟
 - 抖动
 - 所需带宽

不同的服务需要不同的质量

| 服务 | 可靠性 | 延迟 | 抖动 | 所需带宽 |
|---------------|-----|----|----|------|
| E-mail | 高 | 低 | 低 | 低 |
| ftp | 高 | 低 | 低 | 中 |
| Web | 高 | 中 | 低 | 中 |
| rlogin | 高 | 中 | 中 | 低 |
| 音频点播 | 低 | 低 | 高 | 中 |
| 视频点播 | 低 | 低 | 高 | 高 |
| 电话 | 低 | 高 | 高 | 低 |
| 电视会议 | 低 | 高 | 高 | 高 |

Tnbm P397 Fig. 5-30不同的服务需要不同的质量

服务质量

- ❖ 保证服务质量的技术
- ❖ 集成服务
- ❖ 区分服务
- ❖ 标签交换和**MPLS**



所谓服务质量

❖ 提供充足的资源

包括：路由器的处理能力、缓冲区和带宽
在接收端提供缓冲能力，消除抖动

❖ 提供均衡路由

当源站点到目的站点有多条路径时，通常的做法是选一条最佳路径，均衡路由是把流量分散到多条路径上，效果可能更好

保证服务质量的技术

通常保证服务质量的技术包括：

- ❖ 流量整形

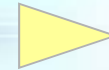
- 漏桶

- 令牌桶

- ❖ 资源预留

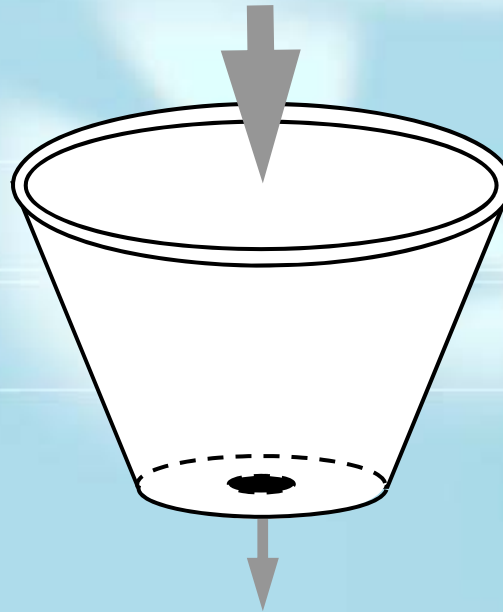
- ❖ 准入控制

- ❖ 数据包调度



漏桶算法 (Leaky Bucket Algorithm)

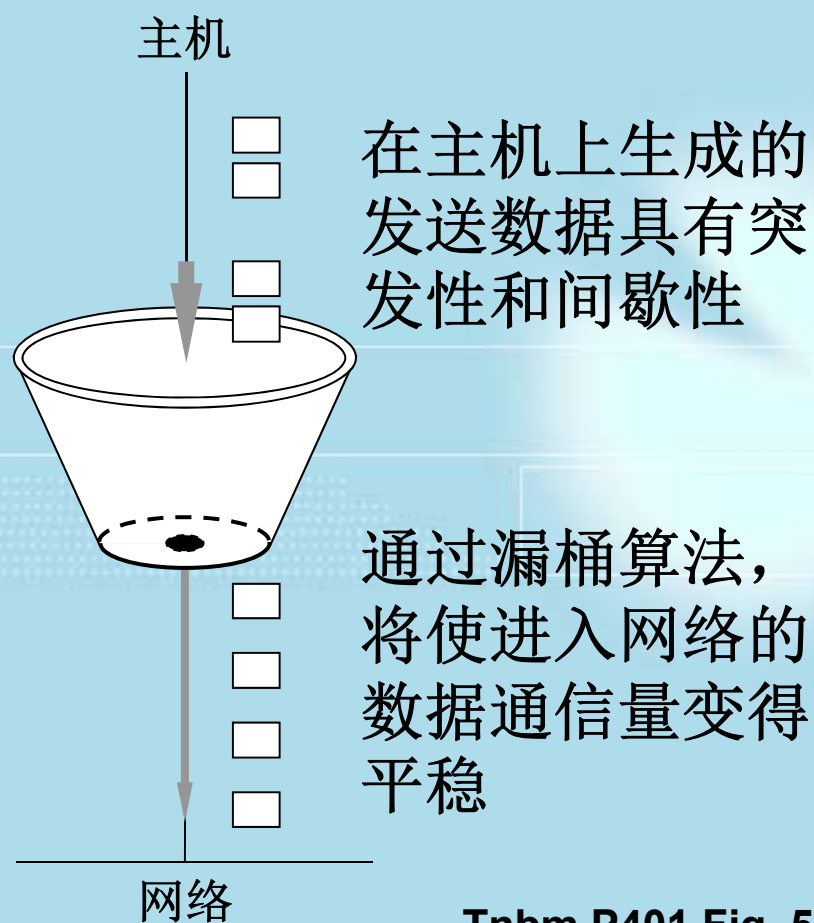
❖ 平滑输入流量，控制进入网络的流量



ρ : 每单位时间泄漏一定量的字节数

Tnbm P401 Fig. 5-32 (a)漏桶

漏桶算法中的主机与网络



| | |
|---|---|
| 漏桶深 如 $C=1\text{M byte}$ | 即缓冲区的容量 |
| 路由器的最佳工作速率是 $\rho = 2\text{M byte/s}$ | 输出速率恒定 如 1M 的漏桶需传输 500ms |
| 数据生成速率为 25M byte/s | 约 40ms 就能把漏桶灌满，否则漏桶将溢出 |

Tnbm P401 Fig. 5-32 (b)数据包漏桶

漏桶算法突发长度的计算

设**C**为漏桶容量 (**Byte, cell**数)

ρ 为漏桶的输出速率 (**Byte/s**)

则突发速率**r**与允许突发时间的长度**S**的关系为:

$$S = \frac{C}{r - \rho}$$

上例中突发长度的计算

| | |
|--|--------------------------------------|
| 漏桶深 C = 1M byte | C = 1M |
| 路由器的最佳工作速率是 $\rho = 2\text{M byte/s}$ | $\rho = 2\text{M}$ |
| 数据生成速率为 25M byte/s r = 25M/s | 允许有 43.5ms 的突发数据 |

$$\begin{aligned}\text{允许突发时间的长度 } S &= C / (r - \rho) \\ &= 1 / (25 - 2) \\ &= 43.5(\text{ms})\end{aligned}$$

漏桶算法的不足之处

- ❖ 漏桶满时，造成数据丢失
- ❖ 漏桶算法强迫输出保持一个固定的平均速率，不能体现通信量的突发

漏桶方法的改进

- ❖ 增加排队缓冲区，提高系统的吞吐量
- ❖ 许可证控制（令牌桶）
- ❖ 跳跃式窗口
 - 通过反馈通知主机缩小发送窗口的大小
- ❖ 信元标注法
 - 将违约信元加标识，进入网络，如网络较空，则通过，否则丢弃
- ❖ 信元按优先级划分
 - 若网络拥挤，先丢掉优先级低的信元

保证服务质量的技术

通常保证服务质量的技术包括：

- ❖ 流量整形

- 漏桶

- 令牌桶

- ❖ 资源预留

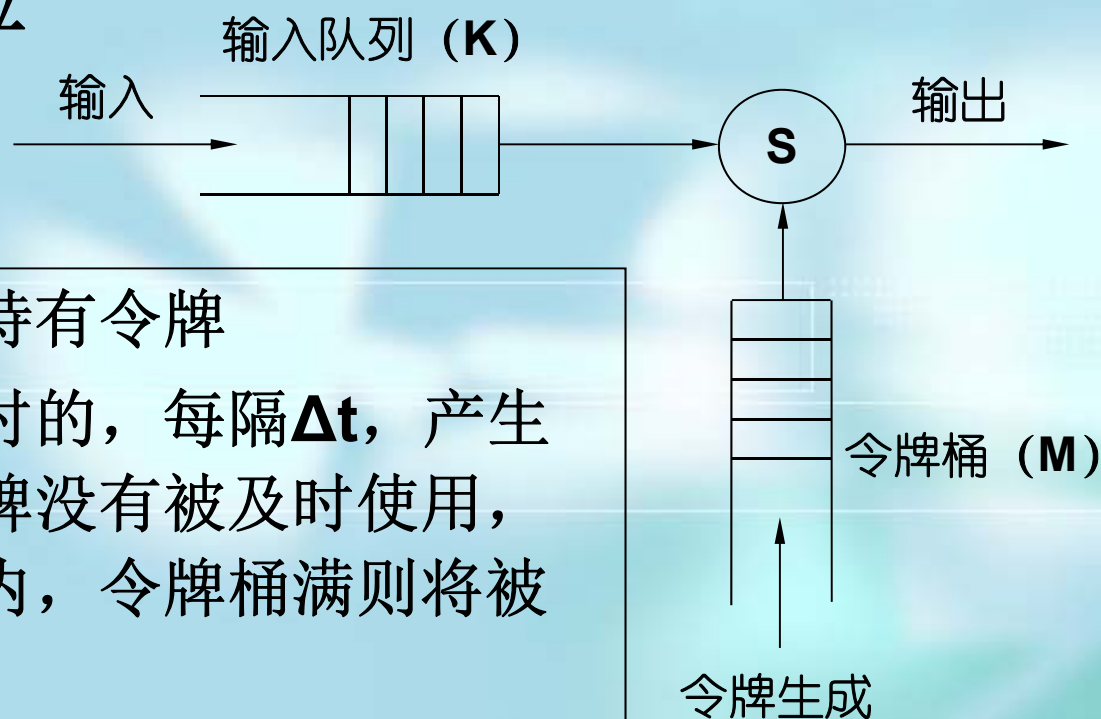
- ❖ 准入控制

- ❖ 数据包调度



令牌桶算法(Token bucket Algorithm)

- ❖ 希望当大的突发流量到来时，输出也能有适当响应



- 数据的输出必须持有令牌
- 令牌的产生是定时的，每隔 Δt ，产生一个令牌，如令牌没有被及时使用，可以存在令牌桶内，令牌桶满则将被丢弃
- 当突发数据到达时，如令牌桶内有多个令牌，则突发数据可按令牌数输出

令牌桶突发时间长度的计算

如: **C**: 为令牌桶的容量

ρ : 为令牌到达速率

M: 为最大的输出速率 (数据到达速率 $> M$)

则最大的突发时间**S**为:

$$C + \rho S = MS \quad \text{即} \quad S = C / (M - \rho)$$

当: 令牌桶的容量**C** = 250K byte

令牌到达速率 ρ = 2M byte/s

最大的输出速率**M** = 25M byte/s时,

如令牌桶已满, 则**S** = 250K/(25M - 2M) (ms)

= 250/23 (ms) 约为11 ms

保证服务质量的技术

通常保证服务质量的技术包括：

- ❖ 流量整形

- 漏桶

- 令牌桶

- ❖ 资源预留

- ❖ 准入控制

- ❖ 数据包调度



资源预留

- ❖ 流量整形是保证服务质量的良好开端，然而这些信息的使用隐含着所有的数据包必须沿着同一路径
- ❖ 一旦路径定下来，必须为该数据流在沿途预留一定的资源，包括带宽、缓冲区和**CPU**的时间

保证服务质量的技术

通常保证服务质量的技术包括：

- ❖ 流量整形

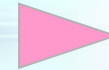
- 漏桶

- 令牌桶

- ❖ 资源预留

- ❖ 准入控制

- ❖ 数据包调度



准入控制

- ❖ 网络根据流量特征及所需的资源决定是否接受该数据流
- ❖ 流说明：一组参数，用来描述流的特征，网络根据这些特征确定所需的资源
- ❖ 当这组参数沿着路径传播时，沿途的路由器都会修改这组参数，当到达接收方，就知道是否准入

流说明举例

| 输入的特性 | 说 明 |
|--------------|--------------------------------|
| 令牌到达速率（字节/秒） | 通信量将被以字节方式工作的令牌桶算法整形 |
| 令牌桶大小（字节数） | 说明了每秒钟有多少字节允许输出及桶的大小，即令牌存储的最大值 |
| 最大传输速率（字节/秒） | 主机在任何条件下可以产生的最高速率 |
| 最小分组大小（字节数） | 最小分组的字节数 |
| 最大分组大小（字节数） | 最大分组的字节数 |

Tnbn P407 Fig. 5-35 流说明举例

保证服务质量的技术

通常保证服务质量的技术包括：

- ❖ 流量整形

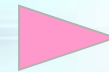
 - 漏桶

 - 令牌桶

- ❖ 资源预留

- ❖ 准入控制

- ❖ 数据包调度



数据包调度

如何强迫每个流只得到它自己预订的带宽

❖ 公平排队

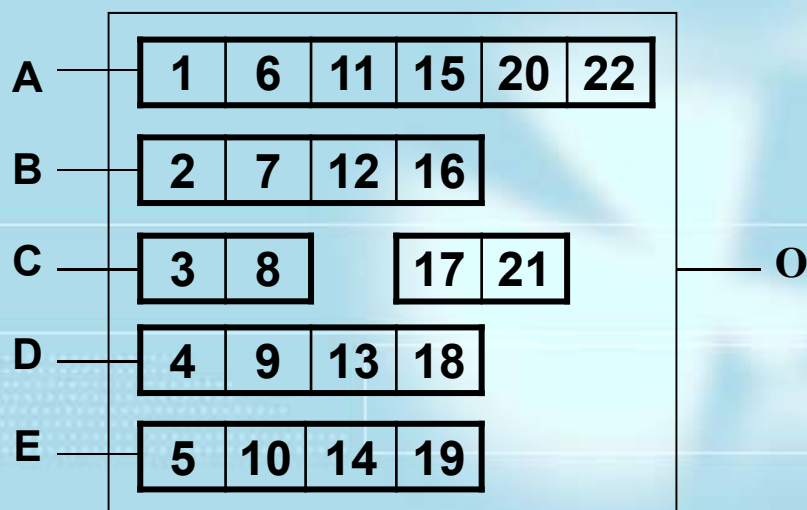


❖ 加权公平排队



公平排队

❖ 公平排队：强迫各发送源公平地占用信道



(a)

| 分组 | 结束时间 |
|----|------|
| C1 | 8 |
| B | 16 |
| D | 18 |
| E | 19 |
| C2 | 21 |
| A | 22 |

(b)

Tnbm P409 Fig. 5-36 (a) 线路O有5个分组队列的路由器
(b) 6个分组的结束时间

公平排队算法

$$F(p_f^j) = \max\{v(A(P_f^j)), F(P_f^{j-1})\} + \frac{l_f^j}{r} \quad j \geq 1$$

其中： P_f^j 为流 f 的第 j 个数据报

l_f^j 为流 f 的第 j 个数据报的长度

$v(A(P_f^j))$ 为数据报 P_f^j 到达的系统虚拟时间

$F(P_f^j)$ 为数据报 P_f^j 的离开时间

r 为线路带宽

主要目的是无论数据流到达时的速率怎样，都必须按传输信道的速率，每个流一个数据块地输出

数据包调度

如何强迫每个流只得到它自己预留的带宽

❖ 公平排队



❖ 加权公平排队



加权公平排队

- ❖ 每个发送源有不同的优先级，即不同的流有不同的带宽

$$F(p_f^j) = \max\{v(A(P_f^j)), F(P_f^{j-1})\} + \frac{l_f^j}{r_f} \quad j \geq 1$$

其中： r_f 为分配给流 f 的线路带宽

如每个流都已分配了不同的输出信道带宽，则排队中必须体现信道带宽的权值，即加权公平排队

服务质量

- ❖ 保证服务质量的技术
- ❖ 集成服务
- ❖ 区分服务
- ❖ 标签交换和**MPLS**



集成服务 (Integrated Service)

- ❖ 提供两类服务

- 保证服务

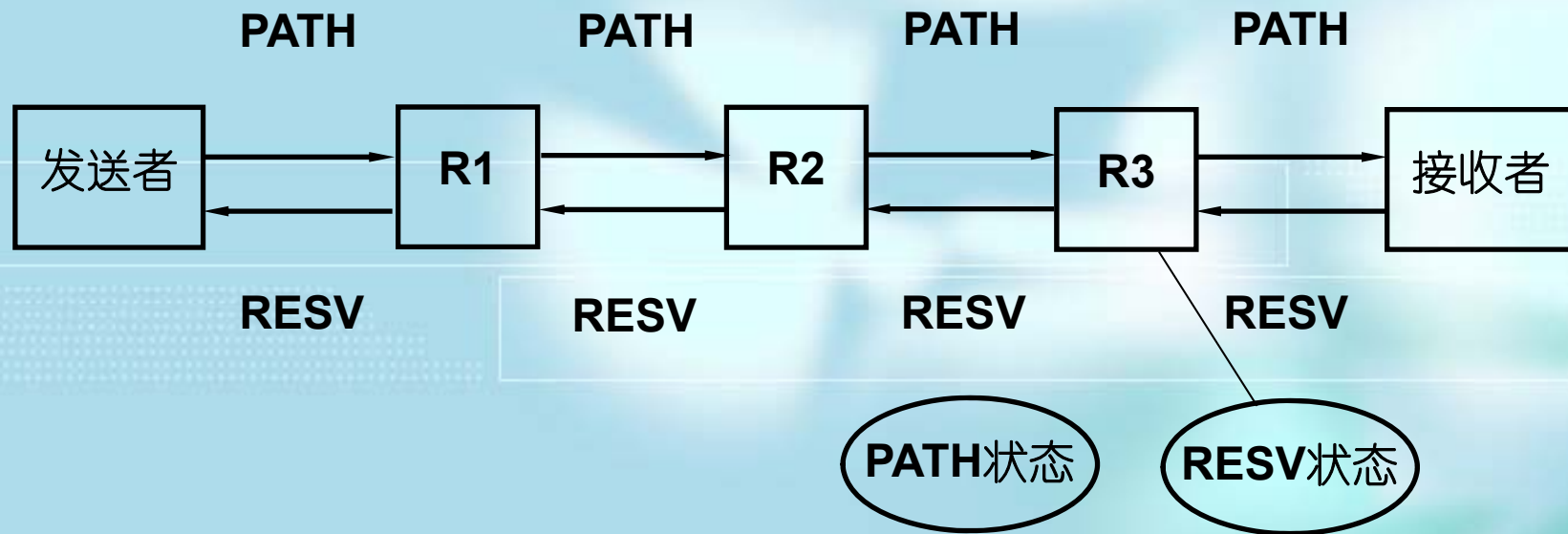
- 负载可控服务

- ❖ 实现手段：通过资源预留

资源预留协议RSVP

- ❖ 最常用的资源预留协议是**RSVP**
(**Resource reSerVation protocol**)
协议将在沿途的路由器上预留一定的资源，包括带宽、缓冲区、表空间等
- ❖ **RSVP**是一种基于接收端，并由接收端发起的资源预留协议

RSVP的工作过程



服务质量

- ❖ 保证服务质量的技术
- ❖ 集成服务
- ❖ 区分服务
- ❖ 标签交换和**MPLS**



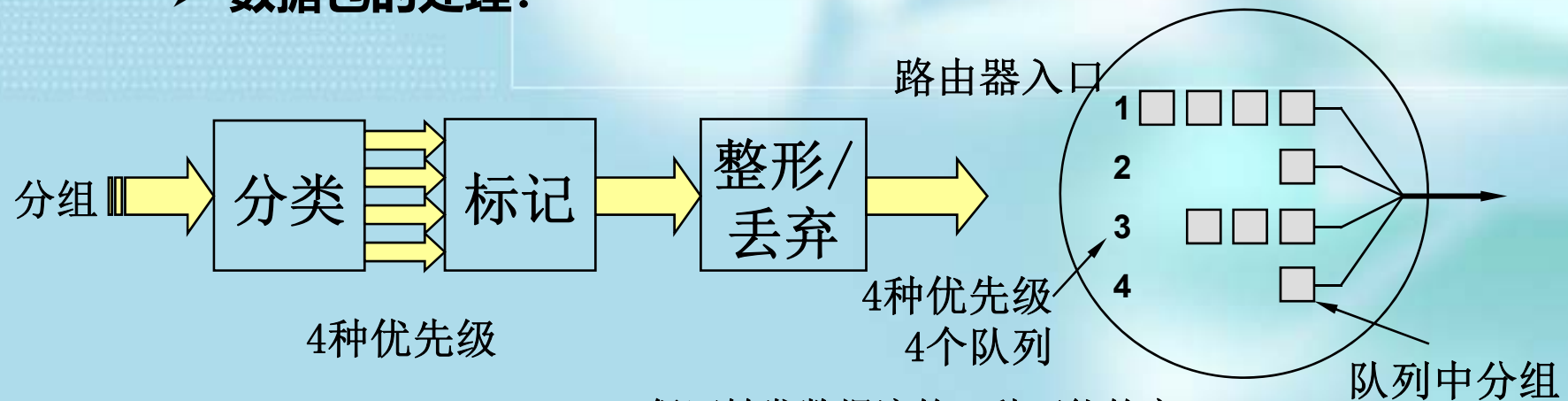
区分服务

集成服务要在沿途的每个路由器上保存每个流的状态，因而可扩展性较差，区分服务就是针对此问题提出

- ❖ **区分服务中，一组路由器形成一个管理域**
- ❖ **每个域中定义一组服务级别，即一组转发规则**
- ❖ **当一客户登录到某一域时，它的数据包必须携带一服务类型子段，用来表示所需的服务级别**
- ❖ **不需要为每一流进行端到端的协商和资源预留**
- ❖ **QBONE: Internet上的区分服务的测试床**

已定义的服务类别

- ❖ **加速转发 (expedited forwarding) :**
 - 专线模拟：定义两个队列（快速、常规）
- ❖ **保证转发 (assured forwarding) :**
 - 分为四个优先级，每个优先级有它自己的资源，每个数据包定义一个丢弃概率：高、中、低，组合起来，一共有12种类别
 - 数据包的处理：



Tnbn P414 Fig. 5-40 保证转发数据流的一种可能的实现

服务质量

- ❖ 保证服务质量的技术
- ❖ 集成服务
- ❖ 区分服务
- ❖ 标签交换和**MPLS**



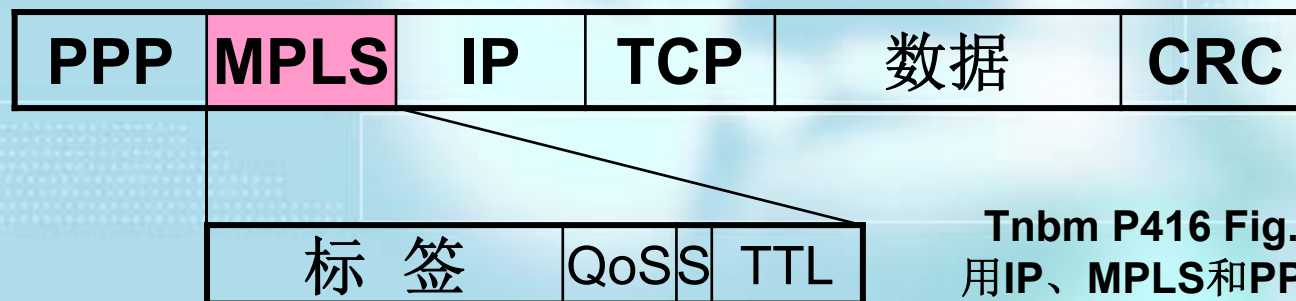
标签交换（Label Switching）

- ❖ 类似于虚电路方式：在数据包头上加一个标签，一般为转发表中的索引，在中间路由器上根据标签而不是根据目的地址作路由，这样可加速转发速度
- ❖ **Label Switching**也被称为**Tag Switching**

MPLS

(MultiProtocol Label Switching)







- ❖ IETF提出了标签交换的标准
- ❖ 标签交换的格式：在IP头前增加一个MPLS头



Tnbm P416 Fig. 5-41
用IP、MPLS和PPP传输
一个TCP数据报

- ❖ 转发表的建立：采用数据驱动的方式，即当第一个数据包经过时，建立转发表项

第5章 网络层

- ❖ 网络层设计的相关问题 
- ❖ 路由算法 
- ❖ 拥塞控制 
- ❖ 服务质量 
- ❖ 网络互联 
- ❖ 因特网中的网络层 

网络互联

❖ 网络互联（**internet**）的背景

不同类型的局域网的发展

Ethernet FDDI 802.11 ATM

TCP/IP SNA NCP/IPX AppleTalk

❖ 网络互联的类型

LAN-LAN LAN-WAN

WAN-WAN LAN-WAN-LAN

所谓网络的不同

| | |
|-------|---------------------------|
| 提供的服务 | 面向连接还是面向非连接 |
| 协议 | IP、IPX、SNA、ATM 等 |
| 编址 | 平面编址还是层次编址 |
| 多址传输 | 支持还是不支持 |
| 分组大小 | 每个网络有它自己的最大值 |
| 服务质量 | 支持还是不支持，支持几种 |
| 差错处理 | 可靠的有序传送还是无序传送 |
| 流量控制 | 滑动窗口、速率控制等，还是没有控制 |
| 拥塞控制 | 漏桶、令牌桶、 RED 、抑制分组等 |
| 安全 | 加密方法等 |
| 参数 | 不同的超时设置，流量说明等 |
| 计费 | 按连接时间、数据包数、字节数或不计费 |

Tnbm P420 Fig. 5-43 不同网络的部分区别

互联网络的相关技术

❖ 互联设备与方式



❖ 互联网络的路由



❖ **Packet**的分段与重组



互联方式

❖ 级联虚电路

要求沿途的网络都能提供可靠传输的保证

❖ 无连接的网络互联

每个分组独立地选择路由

- 不同网络的 分组格式可能不尽相同
- 不同网络可能采用不同的地址编制方法
- 此外，不能保证分组按顺序到达

必须设计一个通用的互联网分组格式和编址方法

❖ 隧道



隧道

❖ 隧道作用

网络互联的一种较简单的方式

VPN—在公用网上建立自己的专用网

❖ 隧道协议

➤ 第二层隧道协议

PPTP—point to point tunneling protocol

L2TP—layer 2 tunneling protocol

➤ 第三层隧道协议: **IPSec (IP Security)**

互联网络的相关技术

❖ 互联设备与方式



❖ 互联网络的路由






❖ **Packet**的分段与重组



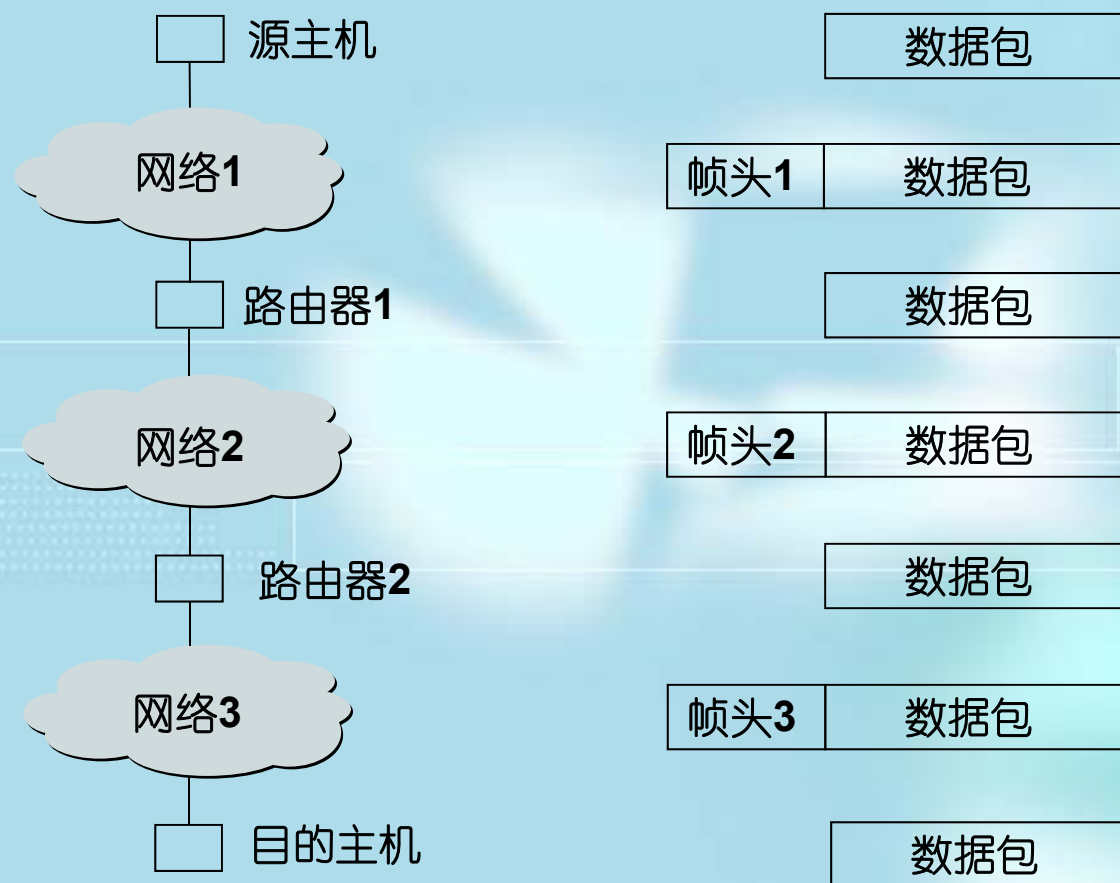
互联网络的路由

- ❖ 将网络分成一个个的**AS**（**autonomous system** 自治系统），**AS**之间由路由器连接
- ❖ 每个自治系统受单一管理机构控制，由一组网络构成（如校园网就是一个**AS**）
- ❖ **AS**中由内部网关协议**IGP**（**Interior Gateway protocol**）处理
- ❖ **AS**间由外部网关协议**EGP**（**Exterior Gateway protocol**）进行处理

互联网络的相关技术

- ❖ 互联设备与方式 
- ❖ 互联网络的路由 
- ❖ **Packet**的分段与重组 

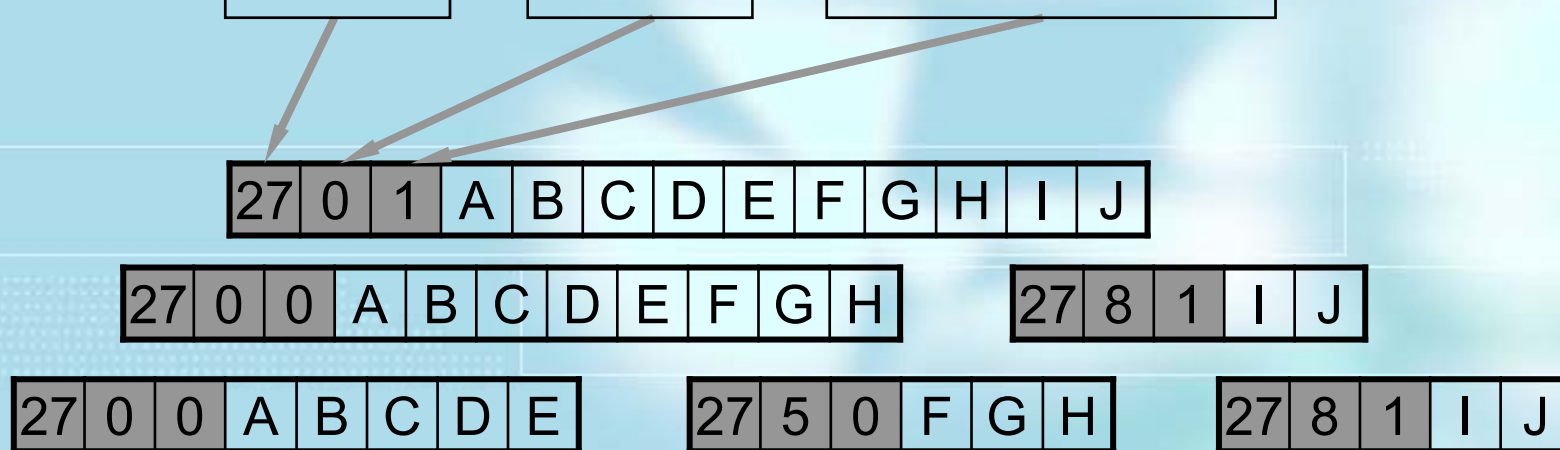
简单的网络互联



分段

每个报头都包含三个字段：

分组号、偏移量和分组结束标志



Tnbn P430 Fig. 5-51 一个分段的例子

重组







- ❖ 透明分段——在入口网关分段，由出口网关重组

出口网关必须知道什么时候本分组的所有分段都已收到，并必须对每个分段进行存储，此外所有分段必须汇集到出口网关

- ❖ 非透明分段——由目的主机重组

要求每一主机都有重组功能，由于每个分段都必须增加一个头部，所以增加了每一分组的开销

第5章 网络层

- ❖ 网络层设计的相关问题 
- ❖ 路由算法 
- ❖ 拥塞控制 
- ❖ 服务质量 
- ❖ 网络互联 
- ❖ 因特网中的网络层 

因特网中的网络层

- ❖ Internet 综述 
- ❖ IP 协议 
- ❖ IP 控制协议 
- ❖ IP 路由 
- ❖ IPv6 

Internet的发展

1970年：第一个分组交换网**ARPA**诞生，连接了四所大学

1972年：**ARPA**网有**40**个网点，应用有**e-mail**、**rlogin**、**FTP**，至此，网络的核心技术产生并开始研究网络的互联

1974年：产生了两个基本的**Internet**协议：**IP**协议和**TCP**协议

80年代后期：产生了**NSF**网，它连接了美国所有的超级计算中心，并逐步取代了**ARPA**网

技术特点

- ❖ 使用**TCP/IP**协议
- ❖ 网络互联结构：各网之间一般采用路由器加专线连接
- ❖ 层次结构的域名及网络管理
- ❖ 分布式的管理模式，没有一个**Internet**管理中心
- ❖ 开发了通用的应用技术

TCP/IP分层模型



Internet的发展方向

- ❖ **Internet 2:** 美国各大学，主要解决 QoS
- ❖ **Ipng: (Next Generation Internet)**
美国政府组建机构研究，主要解决已有的**Internet**的一些限制，如地址、安全和多媒体传输等问题

因特网中的网络层

❖ Internet 综述 

❖ IP 协议 

❖ IP 控制协议 

❖ IP 路由 

❖ IPv6 

IP 协议

❖ IP包格式



❖ IP包的分段与重组



❖ IP地址



IP包格式

| | | | | | |
|-------------|-----|------|-------|-----|-----|
| 0 | 4 | 8 | 16 | | 31 |
| 版本 | 头部长 | 服务类型 | 总长 | | |
| 标识 | | | 标志 | 段偏移 | |
| 生存时间 | 类型 | | 头部校验和 | | |
| 源IP地址 | | | | | |
| 目的IP地址 | | | | | |
| IP可选项（可以省略） | | | | | 充填域 |
| 数据开始... .. | | | | | |

Tnbm P434 Fig. 5-53 IPv4分组格式

- ❖ 版本为**4**（**Ipv4**）
- ❖ 头部长以**32**位字长为单位

IP包格式 (续)

❖ 服务类型：8位（两位保留）

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|-----|---|---|---|---|---|----|---|
| 优先级 | | | D | T | R | 未用 | |

由用户指定数据报的优先级
7→0

↓
Throughput
↓
Delay

↓
Reliability

由路由器选择哪个最优先，但通常都忽略

- ❖ 总长：包括报头和数据报，最长 $2^{16}-1$ ，即**65535**个字节
- ❖ 标识、标志、段偏移：用于数据报的分段
- ❖ 生存时间：以秒为单位
也可以经过路由器的个数为单位

IP包格式 (续)

❖ 类型或协议

| | | | |
|------|---|------|----|
| TCP | 6 | UDP | 17 |
| ICMP | 1 | OSPF | 89 |

- ❖ 头部校验和：按**16**位相加，结果求反
- ❖ 源和目的地址：**32**位
- ❖ **IP**可选项：用于控制和测试
- ❖ 充填域：凑成**32**位的整倍数

IP可选项

用于增加在原始设想中没有考虑到的工作

| 可选项 | 描述 |
|------------------------------|----------------------------------|
| Security | 规定数据包的加密方式 |
| Strict source routing | 分组必须严格按此路径转发 |
| Loose source routing | 分组必须经过给出的路由器 |
| Record route | 每台途经的路由器都必须附上它的 IP 地址 |
| Timestamp | 每台途经的路由器都必须附上它的 IP 地址和时间戳 |

Tnbn P436 Fig. 5-54 IP可选项

IP 协议

❖ IP包格式



❖ IP包的分段与重组



❖ IP地址



分段过程

- ❖ 按**MTU**的及数据包的实际负载长度计算所需段数并划分，分段应满足两个条件：
 - 各段在不大于**MTU**的前提下，尽可能地大
 - 段的长度为**8**的整倍数
- ❖ 原数据包的报头作为每段的数据包报头，并修改其中的某些字段，指明：
 - 属原来的哪个分组
 - 属原来段中的第几个分段
 - 哪一个是段尾

通过标志、标识和段偏移实现

❖ 标识（**identifier**）：16位

发送方每发送一个分组编号加一

各分段的标识相同

源地址加标识来区分各个分段

❖ 标志（**flag**）：3位（一位保留）

| 0 | 1 | 2 |
|----|-----------|-----------|
| 保留 | DF | MF |

DF = 0 表示本分组允许分段

= 1 表示本分组必须完整到达，途中不允许分段

MF = 0 表示本段为本分组的最后一个段

= 1 表示本段后面还有更多的分段

❖ 段偏移（**fragmentation offset**）：13位

实际偏移量 = 段偏移值 x 8 Byte

IP包分段举例

- ❖ 一个物理网络的**MTU为1500B**，现要传输一个**IP分组**（其**IP分组头为20B**，数据区长度为**1400B**）到**MTU为620B**的另一个物理网络，其分段情况为：

| | | | |
|-------|-----|-----|-----|
| 原IP报头 | 600 | 600 | 200 |
| 分段1报头 | 600 | | |
| 分段2报头 | 600 | | |
| 分段3报头 | 200 | | |

IP包分段举例（续）

- ❖ 每个分段的头部其基本部分（如源地址、目的地址等）是**copy**原IP分组的头部，与分段相关的域则应重新生成

| | | 原头部 | 分段1头部 | 分段2头部 | 分段3头部 |
|----|-----|-------|-------|-------|-------|
| ID | 标识 | 30303 | 30303 | 30303 | 30303 |
| M | 标志 | 0 | 1 | 1 | 0 |
| OS | 段偏移 | 0 | 0 | 75 | 150 |
| TL | 总长 | 1420 | 620 | 620 | 220 |

段未结束

$$75 \times 8 = 600$$

分段的重组

- ❖ 途中的任意一台路由器都无法重组，重组必须在各分段都到达目的地之后才能进行
 - 每个分段可以走不同的路径
 - 每个分段不能保证按顺序到达
 - 减少路由器中保存的信息量及路由器的工作量
- ❖ 互联网层是遵循尽力而为来传送**IP**包的，也存在力不从心的时候，此时只能丢弃
 - IP协议采用非透明重组，主机将遵循：要么重组成功，要么全部丢弃的原则**

IP 协议

❖ IP包格式



❖ IP包的分段与重组



❖ IP地址



IP地址

- ❖ **IP地址分类与表示**



- ❖ **子网划分**



- ❖ **CIDR-Classless InterDomain**



Routing

- ❖ **NAT-Network Address**

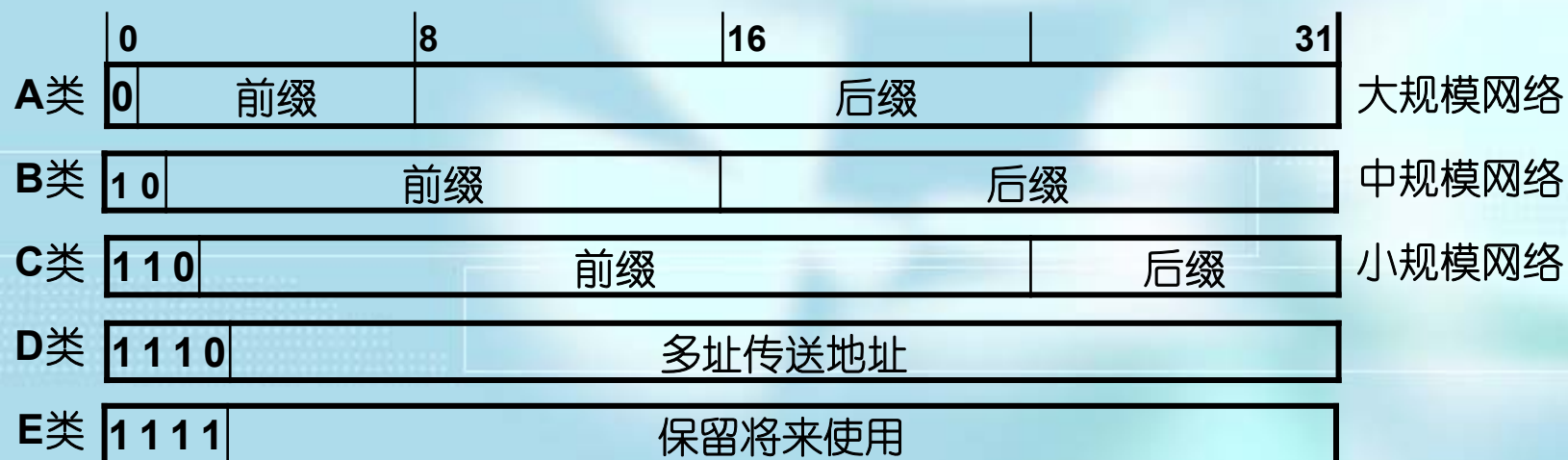


Translation

IP地址的层次结构和分类



❖ IP地址分为A、B、C、D、E类



| 地址类别 | 网络数 | 主机数 |
|------|------------------|------------|
| A | 0~127（128） | 16777216-2 |
| B | 128~191（16384） | 65536-2 |
| C | 192~223（2097152） | 256-2 |

Tnbn P437 Fig. 5-55
IP地址格式

IP地址的表示

- ❖ 点分十进制表示 如: **202.120.1.154**
- ❖ 特殊**IP**地址

| 前缀 | 后缀 | 地址类型 | 用途 |
|--------------|--------------|------|--|
| 全 0 | 全 0 | 本机 | 启动时使用 |
| 全 0 | 主机 ID | 主机 | 标识本网络中的主机 |
| 网络 ID | 全 1 | 直接广播 | 在指定网上广播 |
| 全 1 | 全 1 | 有限广播 | 在本地网上广播 无盘工作站在启动时尚不知道自己所处的网络 ID ，所以用 32 为全 1 地址广播，请求回答 |
| 127 | 任意 | 回送 | 测试 (127.0.0.1) |

Tnbm P438 Fig. 5-56 特殊IP地址

私有网络的IP地址

❖ **IANA** (Internet Assigned Number Authority)

保留给私有网络的**IP**地址段

| 地址类别 | 地 址 |
|-----------|--------------------------------------|
| A类 | 10.0.0.0 - 10.255.255.255 |
| B类 | 172.16.0.0 - 172.31.255.255 |
| C类 | 192.168.0.0 - 192.168.255.255 |

IP地址

- ❖ **IP地址分类与表示**



- ❖ **子网划分**



- ❖ **CIDR-Classless InterDomain**



Routing

- ❖ **NAT-Network Address**

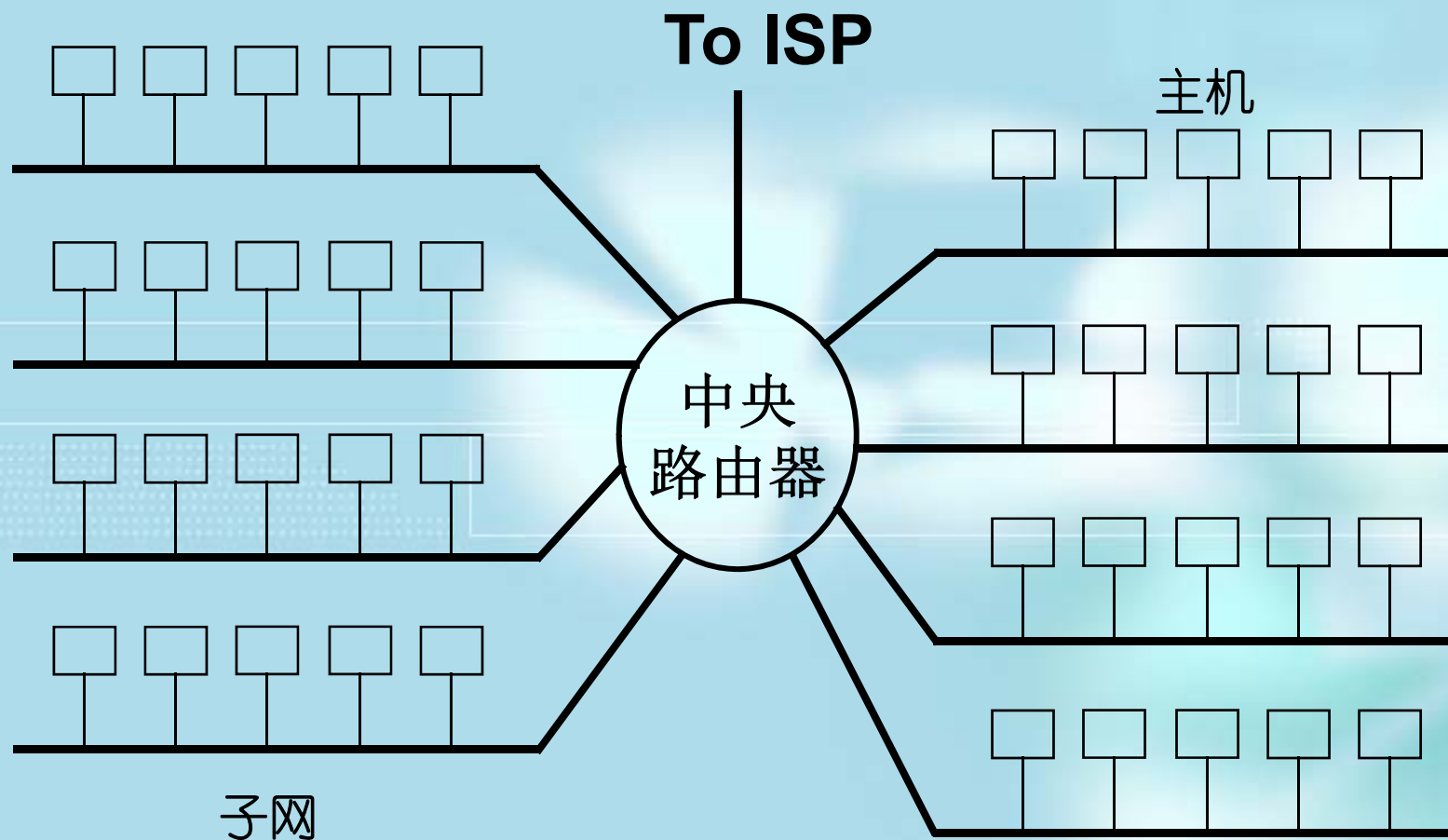


Translation

子网划分

- ❖ 一个网络中的主机都必须具有相同的网络ID
- ❖ 随着网络规模的扩大，虽然能为一个网络提供大量的IP地址（如B类地址），但通常在一个网络（如以太网）内支撑不了那么多的主机
- ❖ 如果再申请一个IP地址，那将造成地址的浪费
- ❖ 问题在于一个网络ID，只能对应一个网络，而不是一组网络
- ❖ 解决的方法是允许在内部将一个网络分成若干个子网，而对外仍像一个网络
- ❖ 然而子网之间的数据传输，必须通过路由器

含子网划分的网络拓扑



Tnbm P439 Fig. 5-57 由部门LAN组成的园区网络

主机的IP地址分配

❖ 任意指派

对**B**类网络，主路由器上的路由表中至少有**65536**个表项

❖ 按子网划分

从**IP**地址的主机号的高位部分取出若干位作为子网的标识，其余作为子网中的主机标识，在子网中，主机标识可任意指定

| | | |
|-----|-----|-----|
| 网络号 | 子网号 | 主机号 |
|-----|-----|-----|

子网掩码

❖ 子网掩码的作用

因为子网地址长度不是固定的，所以必须说明设备地址中的哪一部分是包含子网的网络地址段，地址中哪一部分是主机地址段

❖ 子网掩码使用与**IP**编址相同格式

子网掩码的网络地址部分和子网地址部分全为**1**，它的主机部分全为**0**

❖ 一个缺省**C类IP**地址的掩码为：

255.255.255.0

子网掩码 (续1)

- ❖ 一个C类主机地址为**202.120.3.99**，所属子网号为**011**，其掩码应为：**255.255.255.224**

| | | | | | |
|--------|----------|----------|----------|------|-------|
| | 网络地址 | | | 子网地址 | 主机地址 |
| C类IP地址 | 11001010 | 01111000 | 00000011 | 011 | 00011 |
| | 202 | 120 | 3 | 99 | |
| 掩 码 | 11111111 | 11111111 | 11111111 | 111 | 00000 |
| | 255 | 255 | 255 | 224 | |

子网掩码 (续2)

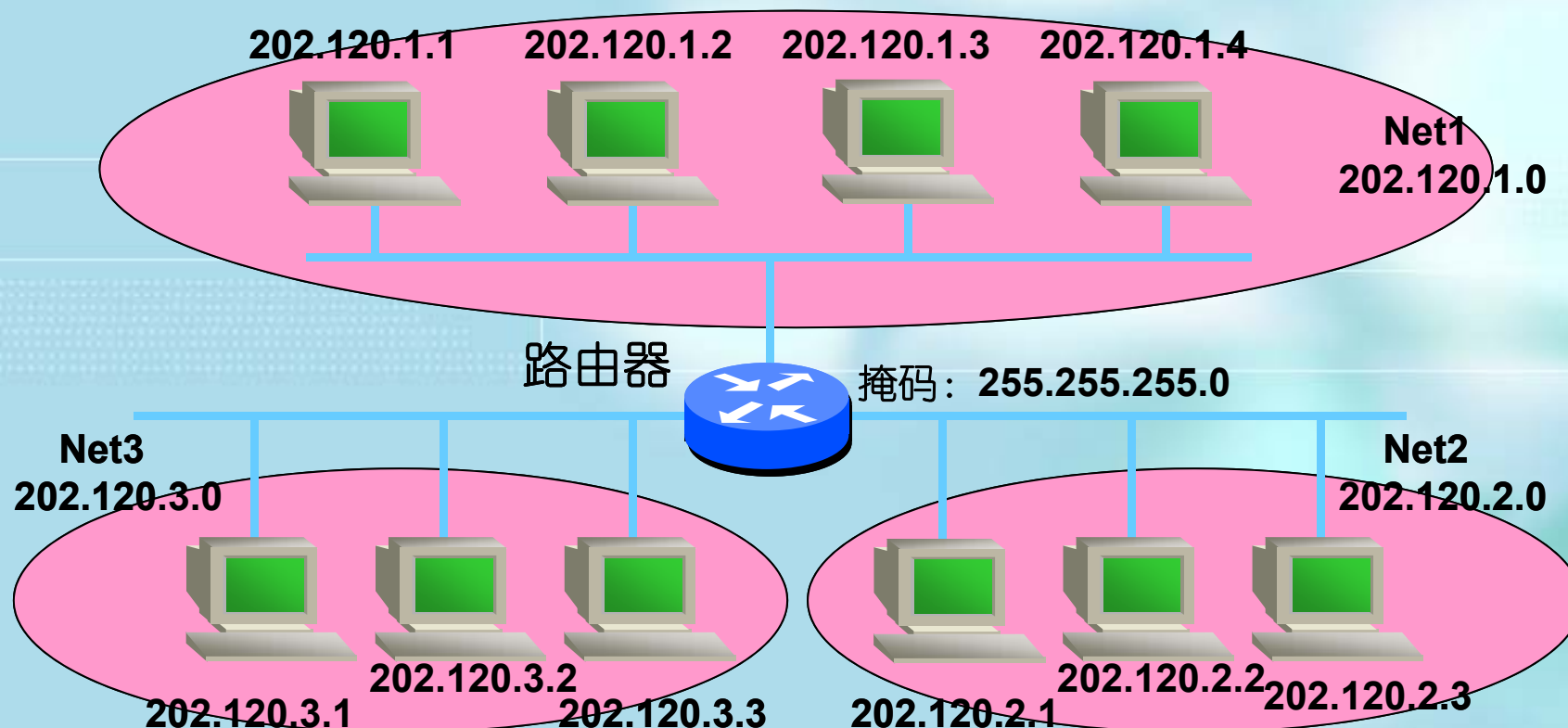
❖ 包含子网地址的网络号 = IP地址 ^ 掩码

| | | | | | |
|--------|----------|----------|----------|------|-------|
| | 网络地址 | | | 子网地址 | 主机地址 |
| C类IP地址 | 11001010 | 01111000 | 00000011 | 011 | 00011 |
| | 202 | 120 | 3 | | 99 |
| 子网掩码 | 11111111 | 11111111 | 11111111 | 111 | 00000 |
| | 255 | 255 | 255 | | 224 |

掩码也可用更简洁的方式表示：**202.120.3.99/27**，
其中**27**表示掩码中**1**的个数

子网划分实例1

某单位有三个部门，分别有自己的网络，虽然每个部门的主机数都比较少，但也需要三个**C**类地址（因为没有比**C**类地址更小的了），掩码为：**255.255.255.0**

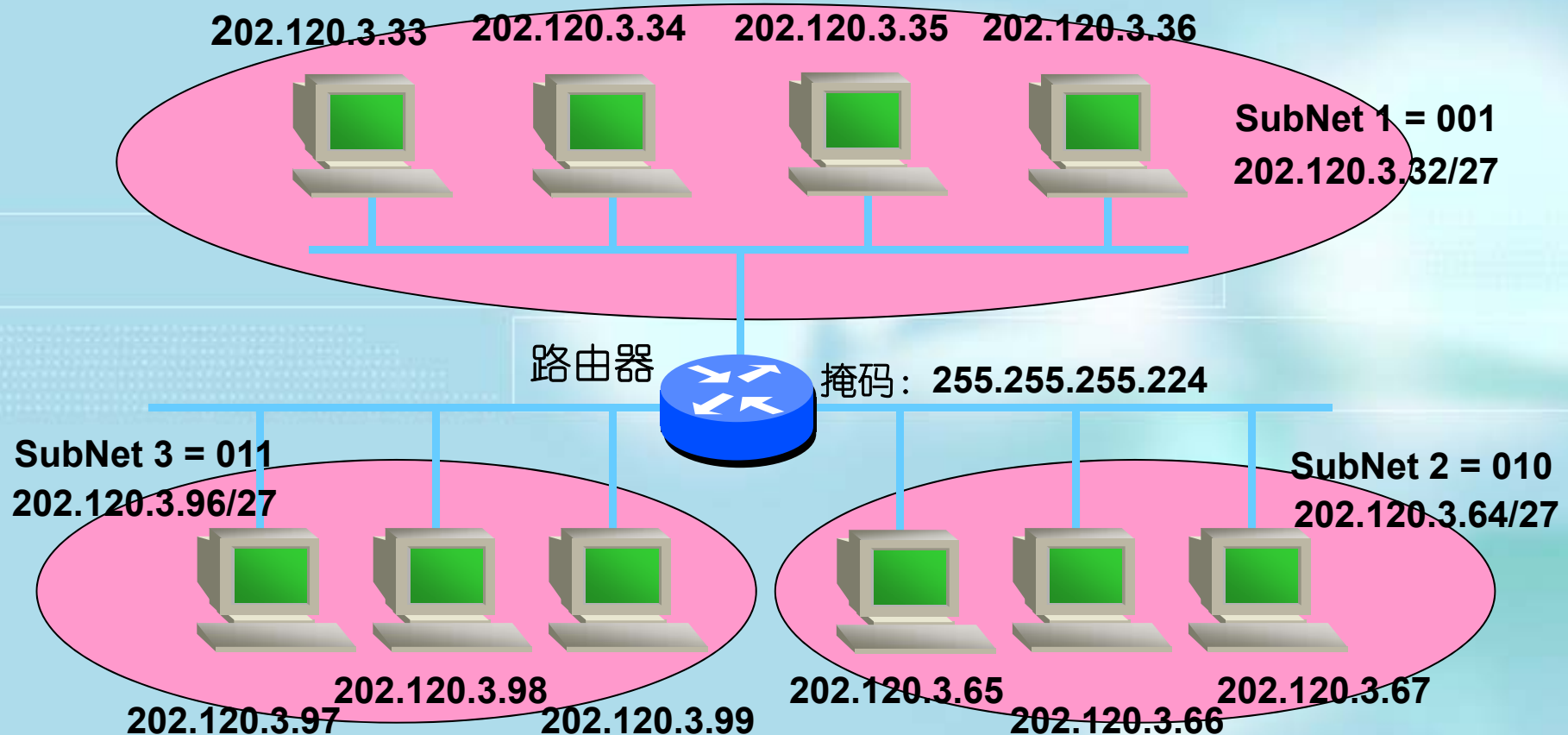


选择子网号长度

- ❖ 三个子网一共只有**10**台主机，因此一个**C**类地址就足够了
- ❖ 子网和主机所需位数的分配：可以根据子网数或者每个子网中的主机数来决定
- ❖ 如按子网数来决定：**3**个子网需要**2**位编码：**00**，**01**，**10**，**11**，但一般**00**和**11**保留，因此**3**个子网需要**3**位编码，则子网掩码为：**202.120.1.0/27**

子网划分实例2

采用子网划分技术，可节约IP地址，图中掩码为：**255.255.255.224**



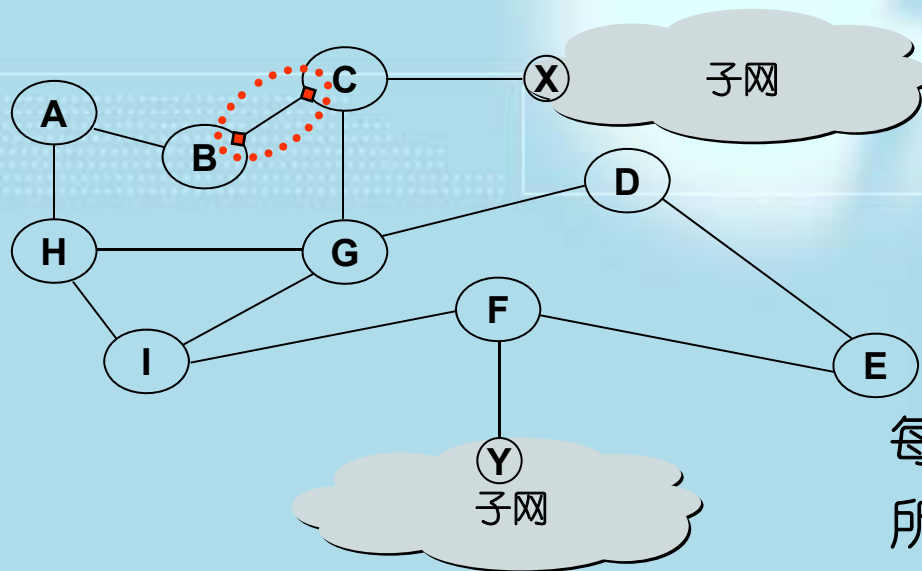
子网划分实例3

例：某单位有多个部门，每个部门的机器数为**20**个左右，现申请有一个**C**类地址 **202.10.23.0**，问：
就目前而言，如何划分子网较为合理，请确定子网掩码，每个子网中主机数最多为多少台？

用**3**位表示子网号，**5**位表示主机号是合理的，所以：
掩码为：**255.255.255.224**；最多的子网个数：**8(6)**个
每个子网中主机数最多可接： **$32 - 2 = 30$** （台）
某个主机**IP**地址及其掩码也可写成：**202.10.23.47/27**

子网划分实例4

例：某主干网（通信子网）使用一个C类地址**200.120.8.0**，主干网的结构如下图所示，图中有**A、B、C、... H、I**等**9**个路由器，所示的两个子网分别通过其边界路由器**X**和**Y**与主干网连接，应如何设计主干网的子网划分策略，即确定掩码，以分配路由器的端口地址



两个原则：

- 两个路由器上相互连接的两个端口必须在同一个子网
- 该子网中不可能包含第三个端口

每个子网中仅有两个有效地址
所以，掩码为：**255.255.255.252**

IP地址

- ❖ **IP地址分类与表示**



- ❖ **子网划分**



- ❖ **CIDR-Classless InterDomain**



Routing

- ❖ **NAT-Network Address**



Translation

CIDR无类域间路由

❖ 为什么要引入CIDR

- 一个**B**类地址对大多数机构来说还是太大，**C**类地址又太小，网络地址本身又非常紧缺

❖ 解决办法：（**RFC1519**）

- 以可变长分块的方式分配所剩的**C**类网络，把若干个**C**类地址（必须是连续的）捆绑成一个组地址，作为一个分配单元
- 甚至也可把**256**个**C**类地址合成一个**B**类地址

❖ 引入CIDR的好处

- 提高**IP**地址的利用率
- 缩短路由表

RFC1519中还规定

- ❖ 在**RFC1519**中，把世界被划分成四个大区，每个大区分配 **2^{17}** 个**C**类地址，分配如下

| | |
|------------------------------------|-------|
| 194.0.0.0 ~ 195.255.255.255 | 欧洲 |
| 198.0.0.0 ~ 199.255.255.255 | 北美 |
| 200.0.0.0 ~ 201.255.255.255 | 中美、南美 |
| 202.0.0.0 ~ 203.255.255.255 | 亚太 |
| 204.0.0.0 ~ 223.255.255.255 | 保留 |

交大拥有六大块，共**168**个**C**类地址，其中：

| | |
|--|------------------|
| 202.120.0.0 ~ 202.120.63.255 | 徐汇、闵行等行政、教学科研等使用 |
| 218.193.176.0 ~ 218.193.193.255 | 闵行四、五期学生宿舍使用 |
| 219.228.96.0 ~ 219.228.127.255 | 闵行六、七期学生宿舍使用 |

CIDR举例

- ❖ 剑桥大学、牛津大学和爱丁堡大学分别提出IP地址申请，分别申请**2048**个、**4096**个和**1024**个IP地址，有关部门从地址**194.24.0.0**开始作了如下的分配：

| 学校 | 地址数 | 地址范围 | 掩码 |
|-----|------|-----------------------------|---------------|
| 剑桥 | 2048 | 194.24.0.0 ~ 194.24.7.255 | 255.255.248.0 |
| 爱丁堡 | 1024 | 194.24.8.0 ~ 194.24.11.255 | 255.255.252.0 |
| 保留 | 1024 | 194.24.12.0 ~ 194.24.15.255 | 255.255.252.0 |
| 牛津 | 4096 | 194.24.16.0 ~ 194.24.31.255 | 255.255.240.0 |

CIDR举例 (续2)

- ❖ 在路由器中，通常应按掩码中含1位数最多的掩码优先做匹配，以确定该分组应从哪一个路由器端口输出

Eg. 目的地址为**194.24.17.4**的分组到达时的处理

| | | | | | |
|-----------|----------------------|---------------------------------|-----------------|-----------------|-----------------|
| IP包中的目的地址 | 194.24.17.4 | 11000010 | 00011000 | 00010001 | 00000100 |
| 爱丁堡大学掩码 | 255.255.252.0 | 11111111 | 11111111 | 11111100 | 00000000 |
| 得到的网络号 | | 11000010 | 00011000 | 00010000 | 00000000 |
| | | 194.24.16.0 不属于爱丁堡大学的网络号 | | | |
| 剑桥大学掩码 | 255.255.248.0 | 11111111 | 11111111 | 11111000 | 00000000 |
| 得到的网络号 | | 11000010 | 00011000 | 00010000 | 00000000 |
| | | 194.24.16.0 不属于剑桥大学的网络号 | | | |
| 牛津大学掩码 | 255.255.240.0 | 11111111 | 11111111 | 11110000 | 00000000 |
| 得到的网络号 | | 11000010 | 00011000 | 00010000 | 00000000 |
| | | 194.24.16.0 属于牛津大学的网络号 | | | |

IP地址

- ❖ IP地址分类与表示
- ❖ 子网划分
- ❖ **CIDR-Classless InterDomain**

Routing

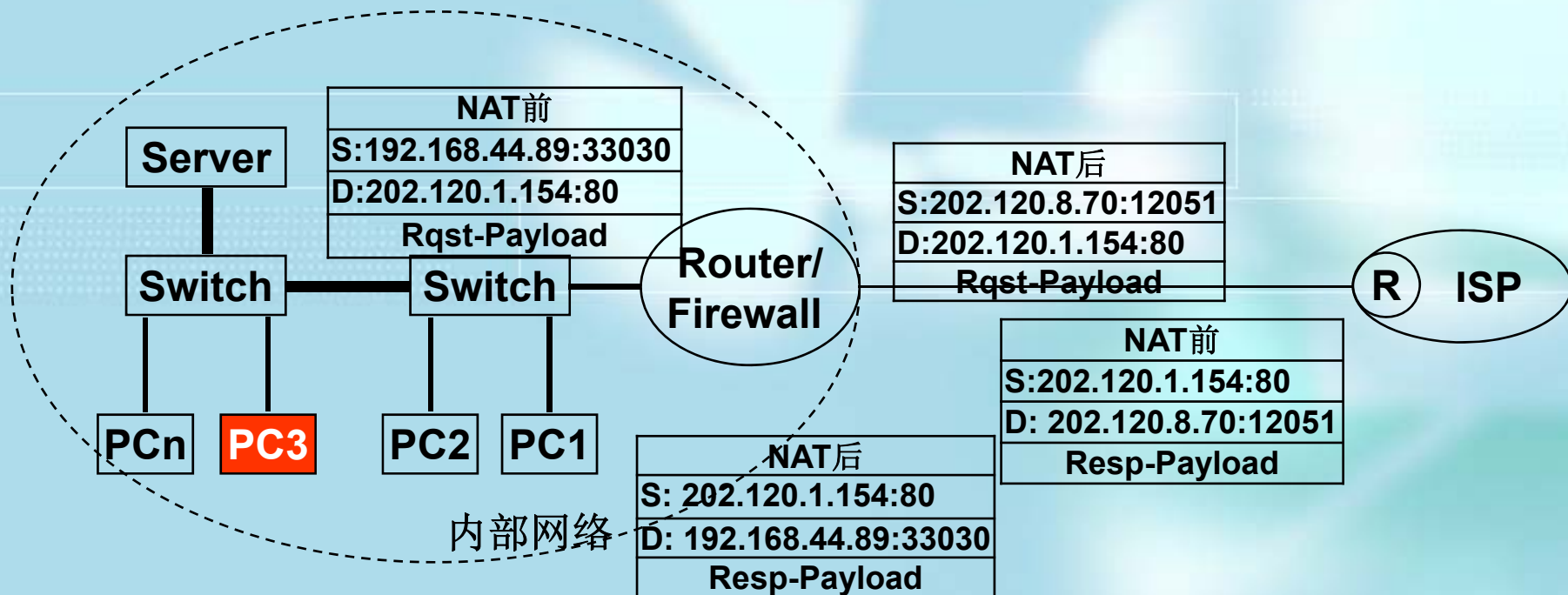
- ❖ **NAT-Network Address
Translation**

网络地址转换NAT

- ❖ 一般来说每台主机都有一个固定的**IP**地址用于表示自己在**Internet**中的身份，但一个单位只能分配到少量的公开**IP**地址，同时也为了安全而不向**Internet**公开单位内部的地址，所以企业内部的主机通常都使用私有地址，当内部主机要访问**Internet**时，必须进行“网络地址转换”，即把私有**IP**地址转换成公开**IP**地址

NAT的工作过程

- ❖ 假设内部网络使用内部网络地址**192.168.44.0**，并已申请到公开IP地址为**202.120.8.65-202.120.8.79**，内部主机**PC3**的IP地址为**192.168.44.89**
- ❖ **PC3**发出一个请求通过**ISP**访问**Internet**，目的IP地址为**202.120.1.154**
- ❖ **Router (Firewall)** 将对请求IP包和返回的应答IP包作**NAT**



Tnbm P446 Fig. 5-60 NAT的工作过程

回来的IP包如何到达源端

- ❖ 利用传输层的源端口号字段
- ❖ 在**NAT**中有一张映射表

| | | |
|-------|---------------|------|
| 0 | 原始的IP地址 | 源端口号 |
| 1 | 192.168.44.33 | 1053 |
| 2 | 192.168.44.69 | 1141 |
| | | ... |
| 151 | 192.168.44.89 | 3033 |
| | | ... |
| 65535 | | |

请求IP包和应答IP包的NAT

- ❖ 当请求**IP**包到达路由器时，将**IP**包中的源**IP**地址和源端口号信息填入**NAT**表（映射表），同时将对应的**NAT**表表项序号取代**IP**包中的源端口号字段，并将选定的合法地址（即公开**IP**地址）取代**IP**包中的源**IP**地址字段，重新计算校验和并发送
- ❖ 当应答**IP**包从**ISP**发回来时，路由器将抽取该应答**IP**包中目的端口号字段，查**NAT**表将表中对应的内容放回目的**IP**地址字段和目的端口号字段，发回源节点

NAT存在的问题

- ❖ **NAT违反了IP的体系结构**

IP规定一个**IP**地址对应一个网络接口，而在**NAT**中，某个地址可能会对应成千上万个地址

- ❖ **NAT将Internet的无连接服务变成了有连接服务**

因为**NAT**必须包含所有经过它的连接信息

- ❖ **NAT违反了网络分层原则**

在第三层用了第四层协议格式中的字段

- ❖ **如果IP包的载荷不是TCP或UDP，则NAT无法工作**

因为在其他协议的信息中不包含源端口号

- ❖ **在有些应用中NAT可能会出错**

在有些应用中，发送方会将源**IP**地址嵌在它的消息中，接收方必须从信息中取用源**IP**地址，但**NAT**不作此类处理，因而将导致错误

因特网中的网络层

❖ Internet 综述 

❖ IP 协议 

❖ IP 控制协议 

❖ IP 路由 

❖ IPv6 

IP控制协议

❖ **IP**协议只负责传送**IP**数据包，无法监视和控制网络中出现的一些问题，这些工作由**Internet**的控制协议来完成

➤ **ICMP**




➤ **ARP**

➤ **RARP**、**BOOTP**和**DHCP**



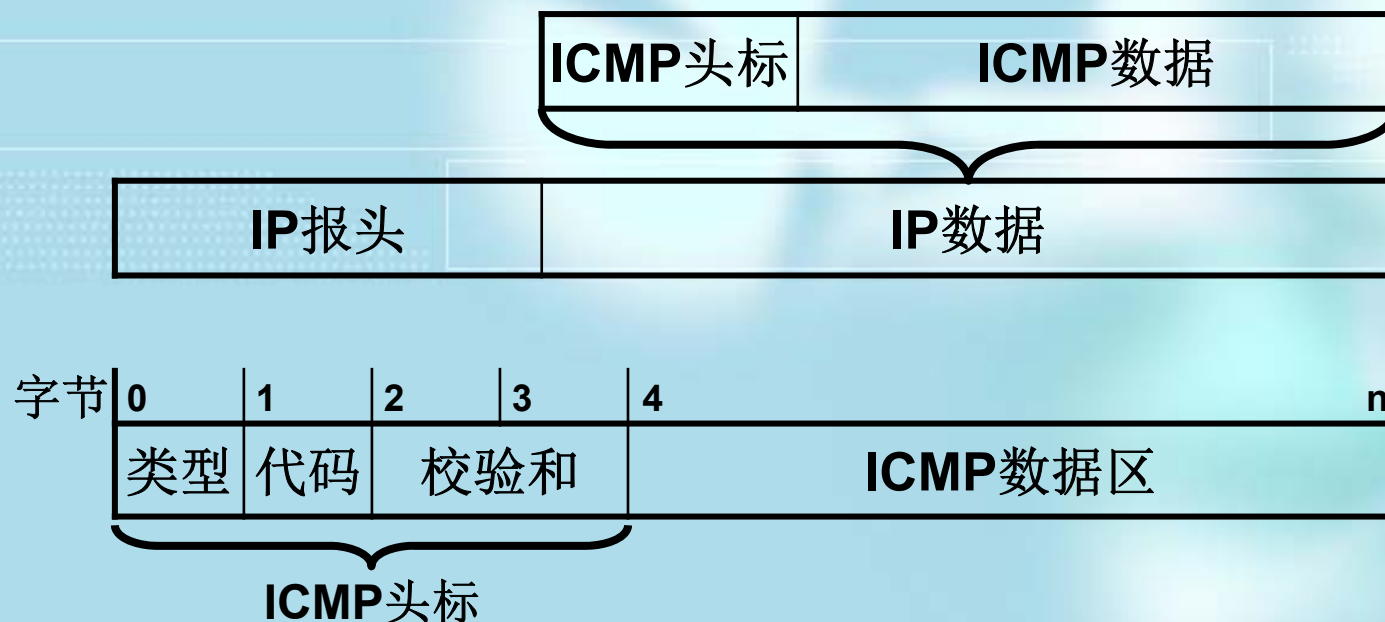
ICMP

(Internet Control Message Protocol)

- ❖ IP协议提供的是尽力而为的通信服务
- ❖ 数据报的丢失、重复、延迟、乱序在所难免
- ❖ **ICMP**提供了一种把通信服务中的差错向源站点报告的机制
 - **ICMP**报文格式 
 - **ICMP**报文主要类型 
 - **ICMP**应用举例 




ICMP报文格式

- ❖ **ICMP**数据连同它的头标一起封装在一个**IP**分组中，但在**TCP/IP**模型的协议分层中，并未把**ICMP**单列为一层



ICMP

(Internet Control Message Protocol)




- ❖ IP协议提供的是尽力而为的通信服务
- ❖ 数据报的丢失、重复、延迟、乱序在所难免
- ❖ **ICMP**提供了一种把通信服务中的差错向源站点报告的机制
 - **ICMP**报文格式 
 - **ICMP**报文主要类型 
 - **ICMP**应用举例 

ICMP报文主要类型

| 类型 | 类型域 | ICMP报文类型 |
|-------------|-----------|----------|
| 差错报文 | 3 | 目的站点不可达 |
| | 11 | 数据报超时 |
| | 12 | 数据报参数错 |
| 控制报文 | 4 | 源抑制 |
| | 5 | 重定向 |
| 请求/应答 报文 | 8 | 回应请求 |
| | 0 | 回应应答 |
| | 13 | 时间戳请求 |
| | 14 | 时间戳应答 |

ICMP

(Internet Control Message Protocol)

- ❖ IP协议提供的是尽力而为的通信服务
- ❖ 数据报的丢失、重复、延迟、乱序在所难免
- ❖ **ICMP**提供了一种把通信服务中的差错向源站点报告的机制
 - **ICMP**报文格式 
 - **ICMP**报文主要类型 
 - **ICMP**应用举例 

ICMP使用举例

- ❖ 测试报文的可达性

ping



- ❖ 路由跟踪

tracert (Unix下为traceroute)



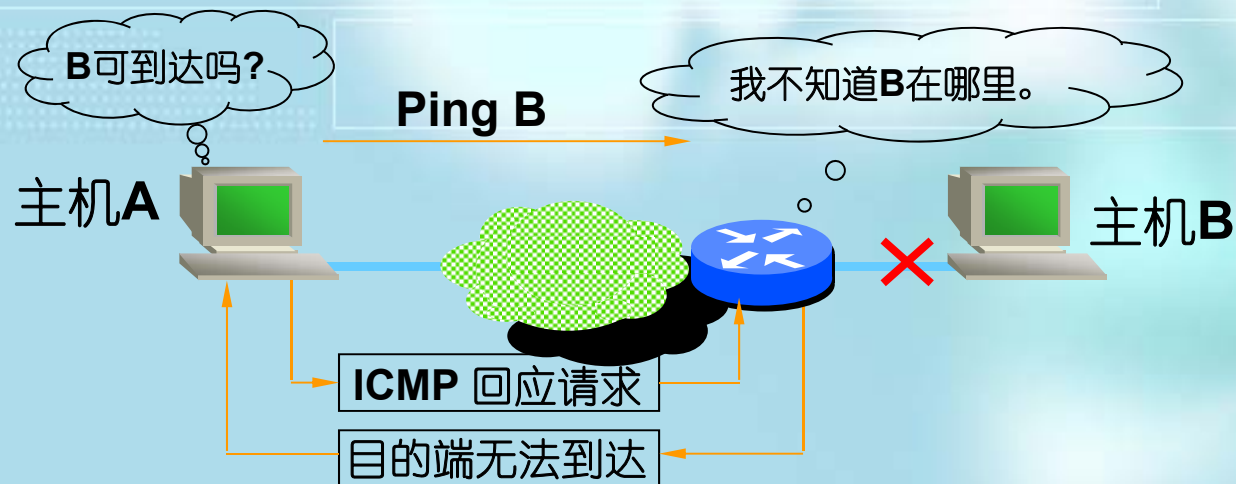
- ❖ 得到路径中最小的**MTU**



ping命令

- ❖ 使用**ping**命令（即调用**ping**过程）时，将向目的站点发送一个**ICMP**回应请求报文（包括一些任选的数据），如目的站点接收到该报文，必须向源站点发回一个**ICMP**回应应答报文，源站点收到应答报文（且其中的任选数据与所发送的相同），则认为目的站点是可达的，否则为不可达

ping命令的实现



ICMP使用举例

- ❖ 测试报文的可达性

ping



- ❖ 路由跟踪

tracert (Unix下为traceroute)



- ❖ 得到路径中最小的**MTU**



tracert命令

- ❖ **tracert**过程是通过**ICMP**数据报超时报文来得到一张途经路由器端口列表的
- ❖ 源主机向目的主机发一个**IP**报文，并置**hop**为**1**，到达第一个路由器时，**hop**减**1**，为**0**，则该路由器回发一个**ICMP**数据报超时报文，源主机取出路由器的**IP**地址即为途经的第一个路由端口地址
- ❖ 接着源主机再向目的主机发第二个**IP**报文，并置**hop**为**2**，然后再发第三个、第四个**IP**数据报，... ..直至到达目的主机

但互联网的运行环境状态是动态的，每次路径的选择有可能不一致，所以，只有在相对较稳定（相对变化较缓慢）的互联网中，**tracert**才是有意义的

ICMP使用举例

- ❖ 测试报文的可达性

ping

- ❖ 路由跟踪

tracert（**Unix**下为**traceroute**）

- ❖ 得到路径中最小的**MTU**

得到路径中最小的MTU

- ❖ 源主机发送一系列的探测IP数据报，并置**DF = 1**，即不允许分段，如途经某个网络的**MTU**较小，则路由器将丢弃该数据报并发回一个**ICMP**数据报参数错，要求分段，源主机则逐步减小数据报长度，并仍置**DF = 1**，直至某个探测报文成功到达目的主机，即得到路径中的最小**MTU**

IP控制协议

❖ **IP**协议只负责传送**IP**数据包，无法监视和控制网络中出现的一些问题，这些工作由**Internet**的控制协议来完成

➤ **ICMP**

➤ **ARP**

➤ **RARP**、**BOOTP**和**DHCP**



ARP (RFC 1512)

(Address Resolution Protocol)

❖ 地址解析的作用



❖ 地址解析技术



❖ 地址解析协议**ARP**



地址解析的作用

❖ 协议地址

软件提供的抽象地址，如**IP**地址，它使整个互联网看成一个网络，但真正的物理网络并不能通过**IP**地址来定位主机

❖ 物理地址

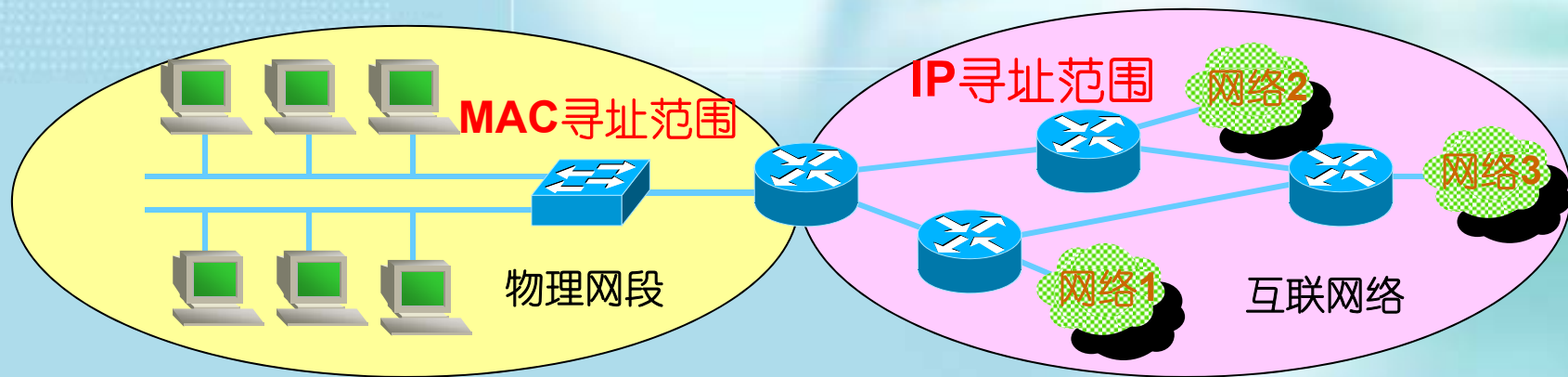
硬件地址，如**MAC**地址

❖ 地址解析

协议地址和物理地址之间的转换，如**IP**地址和**MAC**地址之间的转换

地址解析

- ❖ 地址解析必须在某一物理网络中进行，一台主机在向同一物理网络上的另一台计算机发送数据时，应先做地址解析，然后按物理地址直接发送数据帧



ARP (RFC 1512)

(Address Resolution Protocol)

❖ 地址解析的作用






❖ 地址解析技术



❖ 地址解析协议**ARP**



地址解析技术

- ❖ 查表 
- ❖ 相近形式计算 
- ❖ 消息交换法 

查表

- ❖ 一个物理网络，即**IP**的一个子网，对应一张地址解析表




| IP地址 | 硬件地址（MAC地址） |
|---------------|-------------------|
| 202.120.1.102 | 0A:07:4B:12:82:36 |

- ❖ 一个物理网络，即**IP**的一个子网，它的**IP**地址中的网络号均相同，因此在实际使用时，可省略**IP**地址中的网络号部分

优点：通用、实现简单

缺点：需要人工设置对应表

地址解析技术

- ❖ 查表 
- ❖ 相近形式计算 
- ❖ 消息交换法 




相近形式计算

适用于动态硬件地址，它在指定**IP**地址和硬件地址时，使它们保持一定的关系，在解析时可通过**IP**地址计算出硬件地址

如：硬件地址与**IP**地址的最后一个字节相同，

则：硬件地址 = **IP**地址 & 0xff

地址解析技术

- ❖ 查表 
- ❖ 相近形式计算 
- ❖ 消息交换法 

消息交换法

- ❖ 服务器方式：由服务器提供解析结果（**ATM**网络）
- ❖ 分布式方法：每台计算机负责对本机地址的解析
- ❖ 相比较而言，前者对地址的配置和管理较容易，但需要额外的服务器，一旦网络繁忙程度加大，服务器会成为瓶颈

ARP (RFC 1512)

(Address Resolution Protocol)

❖ 地址解析的作用



❖ 地址解析技术



❖ 地址解析协议**ARP**



地址解析协议ARP

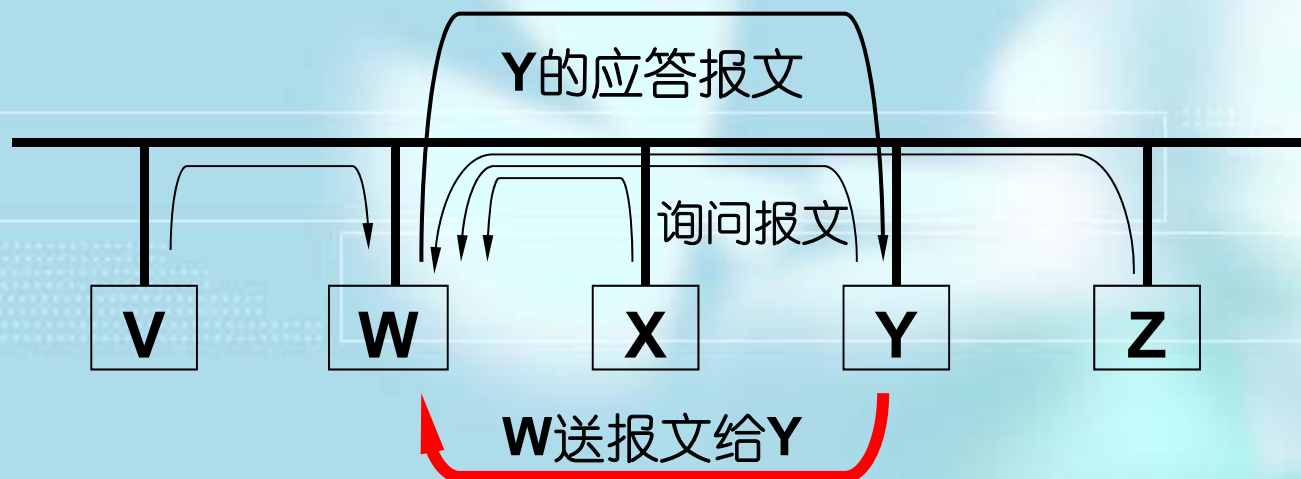
- ❖ 工作原理 
- ❖ **ARP**的报文格式 
- ❖ 暂存**ARP**应答 
- ❖ 处理接收的**ARP**消息 

工作原理（以IP网络为例）

- ❖ 一个**ARP**请求消息是一个数据帧，其中包含发送站点的**MAC**地址和**IP**地址，以及目的站点的**IP**地址，并把此数据帧在本物理网络内广播
- ❖ 一个**ARP**应答消息是一个数据帧，其中包含应答站点的**MAC**地址和**IP**地址，以及原发送站点的**IP**地址，并把此数据帧发送给原发送站点

工作原理（续）

- ❖ 站点**W**有数据发送给目的站点**Y**，但目前尚不知道站点**Y**的**MAC**地址，无法组成数据帧



ARP的询问报文中包含目的站点**Y**的**IP**地址

ARP的应答报文中包含目的站点**Y**的**MAC**地址

地址解析协议ARP

- ❖ 工作原理 
- ❖ **ARP**的报文格式 
- ❖ 暂存**ARP**应答 
- ❖ 处理接收的**ARP**消息 

ARP的报文格式

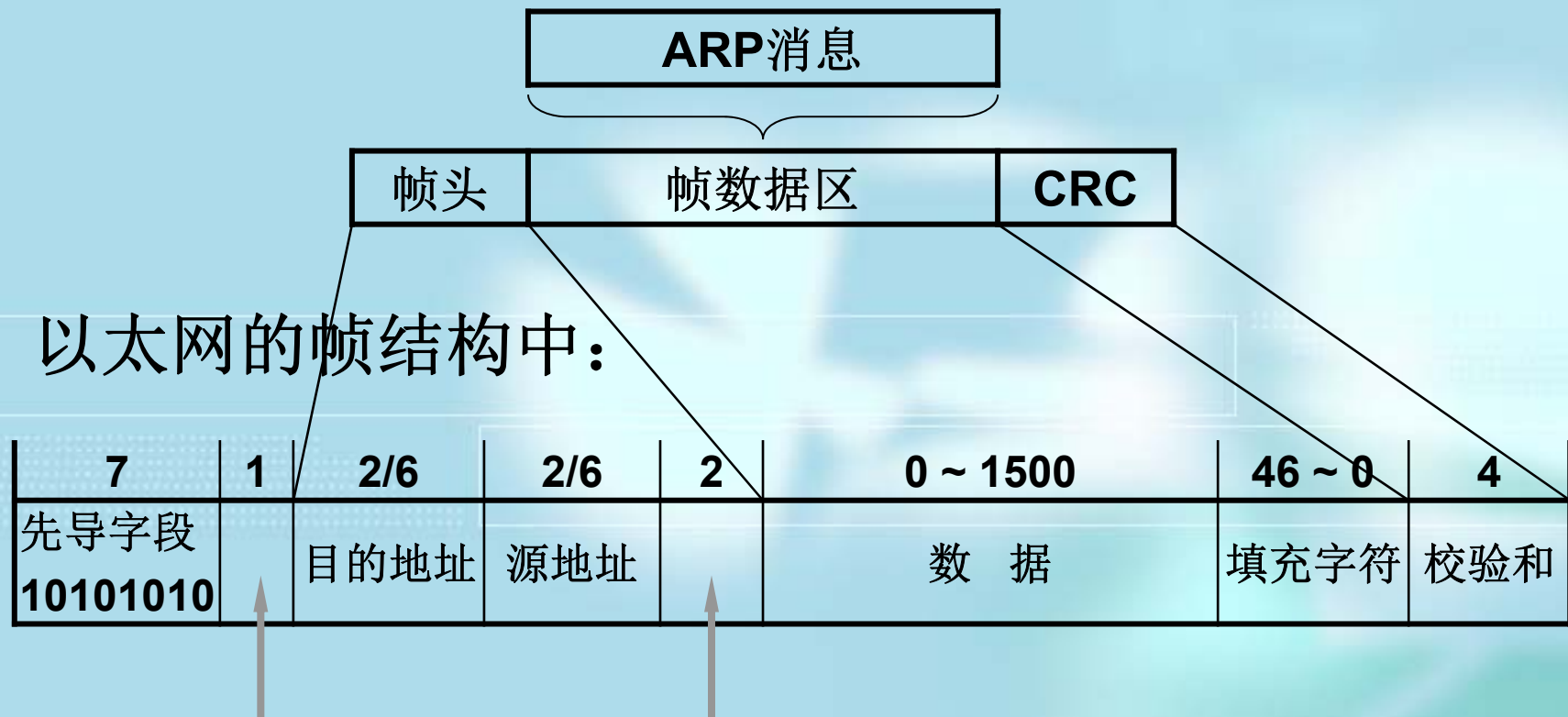
| | | | |
|---------------------------|--------|-------------------------|----|
| 0 | 8 | 16 | 31 |
| 硬件地址类型 | | 协议地址类型 | |
| 硬件地址长度 | 协议地址长度 | 操 作 | |
| 发送站硬件地址（字节 0~3 ） | | | |
| 发送站硬件地址（字节 4~5 ） | | 发送站协议地址（字节 0~1 ） | |
| 发送站协议地址（字节 2~3 ） | | 目的站硬件地址（字节 0~1 ） | |
| 目的站硬件地址（字节 2~5 ） | | | |
| 目的站协议地址全部（字节 0~3 ） | | | |

以协议地址是IP地址、硬件地址 是以太网MAC地址为例

| | |
|-------------------------|---------------|
| 发送站 ARP 请求报文中填入： | |
| 硬件地址类型 | 1 |
| 协议地址类型 | 0X0800 |
| 硬件地址长度 | 6 |
| 协议地址长度 | 4 |
| 操作 | 1（请求） |
| 发送站硬件地址 | MAC地址 |
| 发送站协议地址 | IP地址 |
| 目的站硬件地址 | 全0 |
| 目的站协议地址 | IP地址 |

| | |
|-------------------------|---------------|
| 目的站 ARP 应答报文中填入： | |
| 硬件地址类型 | 1 |
| 协议地址类型 | 0X0800 |
| 硬件地址长度 | 6 |
| 协议地址长度 | 4 |
| 操作 | 2（应答） |
| 目的站硬件地址 | MAC地址 |
| 目的站协议地址 | IP地址 |
| 原发送站硬件地址 | MAC地址 |
| 原发送站协议地址 | IP地址 |

ARP消息在以太网中



帧开始字符10101011 ARP消息帧的类型值为0X806

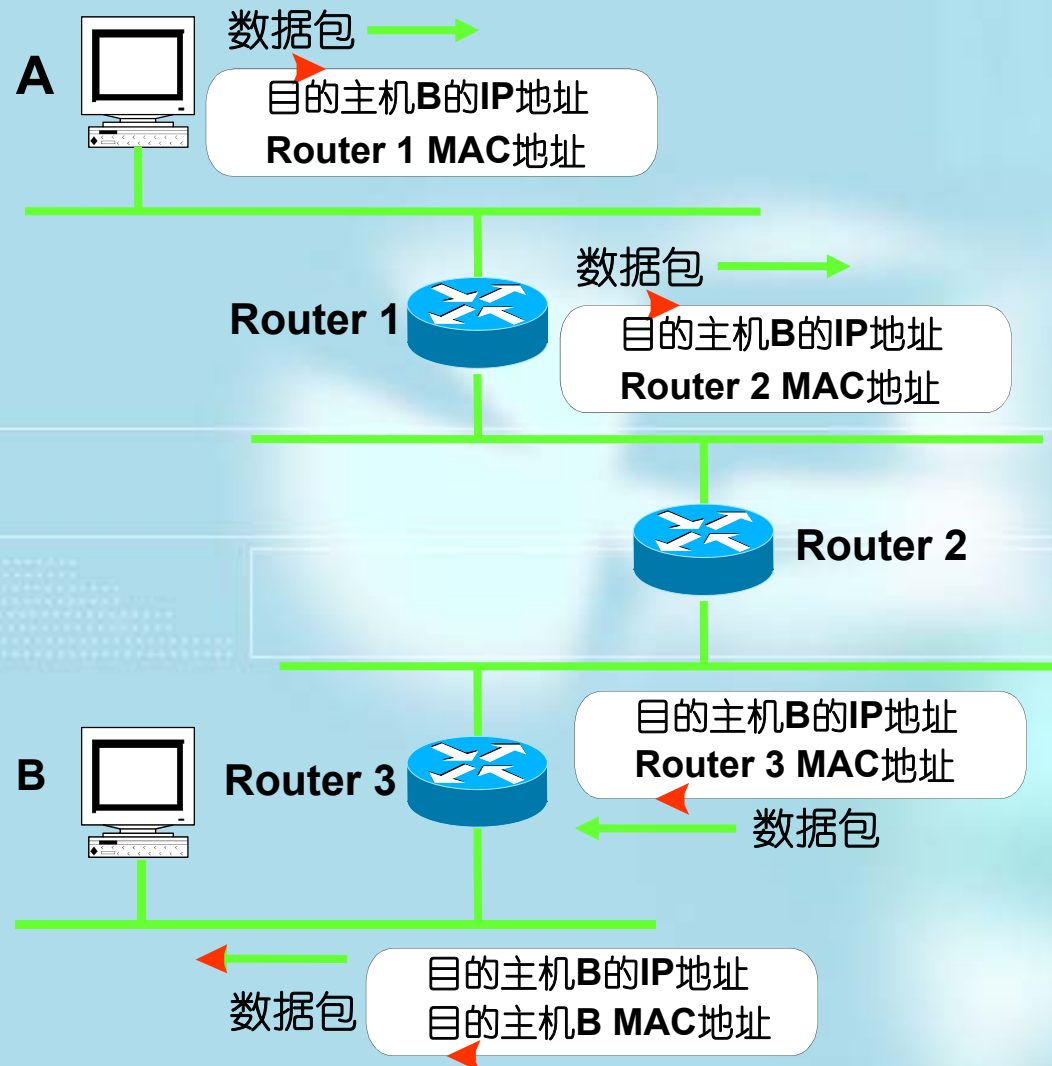
地址解析协议ARP

- ❖ 工作原理 
- ❖ **ARP**的报文格式 
- ❖ 暂存**ARP**应答 
- ❖ 处理接收的**ARP**消息 

暂存ARP应答和 处理接收的ARP消息

- ❖ **ARP**应答暂存于**Cache**或内存中，以后即可查表，不必再发询问报文，以减少网络的通信量
- ❖ 从消息中取出发送方的协议地址和硬件地址，更新**cache**中已有的信息
- ❖ 检查消息是请求还是应答，若是应答，则接收；若是请求，检查是否为发送给本站的，如是，则发应答消息

ARP地址解析和数据包在网间的传递



IP控制协议

❖ **IP**协议只负责传送**IP**数据包，无法监视和控制网络中出现的一些问题，这些工作由**Internet**的控制协议来完成

➤ **ICMP**

➤ **ARP**

➤ **RARP**、**BOOTP**和**DHCP**



RARP、BOOTP和DHCP

❖ RARP 

❖ BOOTP 

❖ DHCP 

RARP (RFC 903)

(Reverse Address Resolution Protocol)

给出一个以太网地址，如何找到相应的**IP**地址？
(如在无盘工作站中)

- ❖ 每个子网需要一个**RARP**服务器，当工作站要得到其**IP**地址时，将它的以太网地址广播出去，**RARP**服务器得到此信息，在其配置文件中找到该以太网地址，把对应的**IP**地址返回该工作站
- ❖ **RARP**的消息不能跨路由器，因此每个子网中都必须有一个**RARP**的服务器

RARP、BOOTP和DHCP

❖ RARP 

❖ BOOTP 

❖ DHCP 

BOOTP (Bootstrap Protocol)

- ❖ **BOOTP**是一个高层的程序，基于**UDP**，它可取代**RARP**服务器得到**IP**地址、启动文件地址和配置信息
- ❖ **BOOTP**协议采用客户/服务器工作方式，需要得到配置信息的一方称为**BOOTP**客户，提供配置信息的一方称为**BOOTP**服务器
- ❖ 客户方首先用局部广播（如果子网中有**BOOTP**服务器）或全局广播（**BOOTP**服务器不在同一子网中）的**IP**地址发送一条请求报文，如果客户知道自己的**IP**地址，可将此地址放入请求报文，否则在请求报文中的发送方**IP**地址部分放入全**0**，表示客户方需要得到**IP**地址，**BOOTP**的服务器用广播地址返回一条包含客户的**IP**地址及其他启动信息的应答报文

BOOTP的问题

- ❖ 需要手工配置**IP**地址和**MAC**地址的对应表，
当一台新的主机加入某一**LAN**时，它不能立即用**BOOTP**启动，必须等到管理员赋给它一个**IP**地址，并把该**IP**地址和**MAC**地址的对应关系手工写入**BOOTP**的配置文件

RARP、BOOTP和DHCP

❖ RARP 

❖ BOOTP 

❖ DHCP 

DHCP

(Dynamic Host Configuration Protocol)

- ❖ 可以静态或动态地为主机分配**IP**地址
- ❖ 基于客户/服务器的工作方式
- ❖ 不需要为每个**LAN**设置一台**DHCP**服务器，多个**LAN**可共享一台**DHCP**服务器，但每个**LAN**必须有一个**DHCP**中继代理，负责转发客户请求

DHCP工作过程

- ❖ 客户首先广播一条包含它自己的客户编号的报文（**DHCP DISCOVER**报文），以声明自己的出现
- ❖ 每个收到客户**DHCP DISCOVER**的服务器检查用户的配置文件决定是分配一个静态地址或动态地址，如果是要一个动态地址，服务器从“地址池”里选择一个**IP**地址，如果是要静态地址，服务器从配置文件中取出该客户的静态地址，将该地址放在一条**DHCP OFFER**报文中，送回用户
- ❖ 客户收到**DHCP OFFER**报文后，在众多的服务器中选择一个服务器，这通常是根据**DHCP OFFER**报文中提供的选项来决定

DHCP工作过程（续）

- ❖ 客户再广播一条**DHCP REQUEST**报文，指出选择了哪一个服务器，并申请使用该服务器提供的**IP**地址
- ❖ 当服务器收到**DHCP REQUEST**报文，并且该报文指出使用了它提供的地址，则服务器将该地址标记为租用，如服务器收到的报文指出客户接受了另一服务器提供的地址，则将地址退回给地址池，如果在一段时间内没有收到报文，服务器也将地址退回地址池，选中的服务器发送一条应答报文**DHCP ACK**
- ❖ 客户确定该配置信息是否合法，接受了合法的租用后，客户指定一个绑定**binding**服务器的声明，继续使用**该IP地址和选项**

因特网中的网络层

❖ Internet 综述 

❖ IP 协议 

❖ IP 控制协议 

❖ IP 路由 

❖ IPv6 

IP路由

- ❖ 路由器和路由表
- ❖ 静态路由和动态路由
- ❖ 分层路由
- ❖ **RIP**协议
- ❖ **OSPF**协议
- ❖ **BGP**协议
- ❖ **IP**多址传输

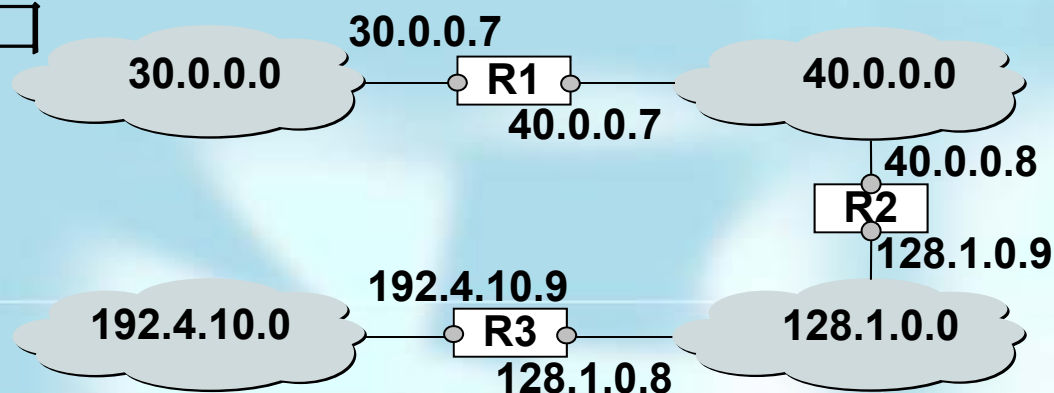


路由器和路由表

- ❖ 路由器是网络层的一个智能设备，承担了路由选择的任务，选择路由的依据是一张路由表，路由表指明了要到达某个地址该走哪一条路径
- ❖ 在路由表中，并非为每一个具体的目的主机的**IP**地址指明路径，而是为目的主机所在的网络指明路径，这样路由表的大小才落在可操作的范围内，因此查找路由表的依据是目的主机的网络地址
- ❖ 路由器对每一个接收到的分组，取出它的目的**IP**地址，然后根据目的**IP**地址中的网络地址查找路由表，确定下一步的传输路径，并从相应的路由器端口将分组送出，传送的路径轨迹是由所经过的路由器一步一步确定的

路由表

- ❖ 决定到某个子网去，下一站该送交哪一个路由器端口



R2的路由表为:

| 目的地 | 掩码 | 下一站 |
|------------|---------------|-----------|
| 30.0.0.0 | 255.0.0.0 | 40.0.0.7 |
| 40.0.0.0 | 255.0.0.0 | 直接传送 |
| 128.1.0.0 | 255.255.0.0 | 直接传送 |
| 192.4.10.0 | 255.255.255.0 | 128.1.0.8 |

IP地址的掩码

- ❖ 掩码用来对某主机的**IP**地址作“**与**”操作后可得到该目的主机的网络号

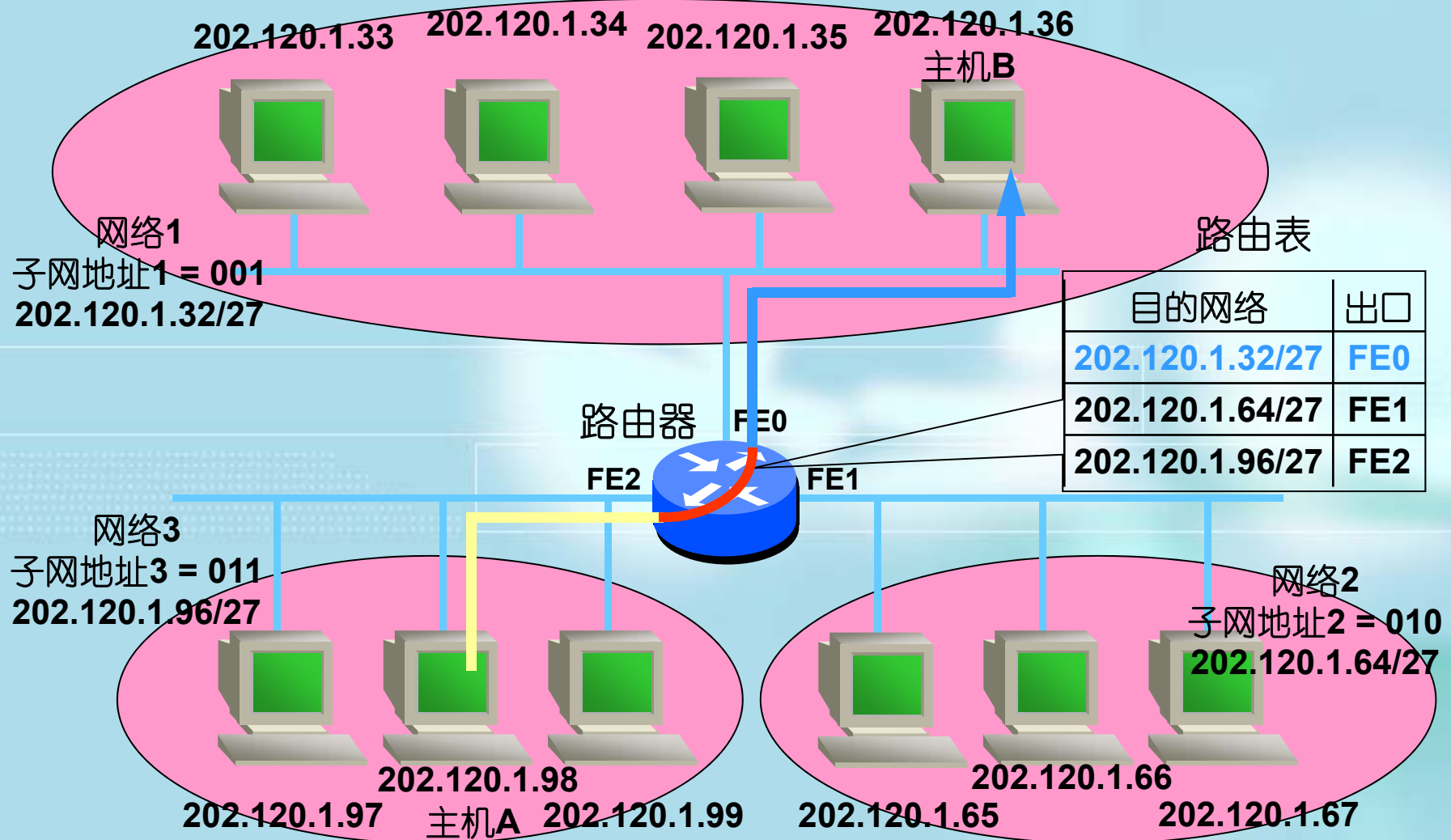
即：目的主机的网络号

= 目的主机的**IP**地址 **&** 该地址的掩码

路径的选择：

if ((Mask[i]&D) == Destination[i]) forwarding to NextHop[i]

路由选择实例



IP路由

- ❖ 路由器和路由表
- ❖ 静态路由和动态路由
- ❖ 分层路由
- ❖ **RIP**协议
- ❖ **OSPF**协议
- ❖ **BGP**协议
- ❖ **IP**多址传输



静态路由和动态路由

❖ 静态路由（**Static Route**）

—— 人工在路由器上配置路由表

优点：路由器不必为路由表项的生成和维护花费大量时间，有时可以抑制路由表的增长

缺点：人工配置开销大，网络拓扑结构变更时需重新配置路由表，一般只在小型网络或部分链路上使用

❖ 动态路由（**Dynamic Route**）

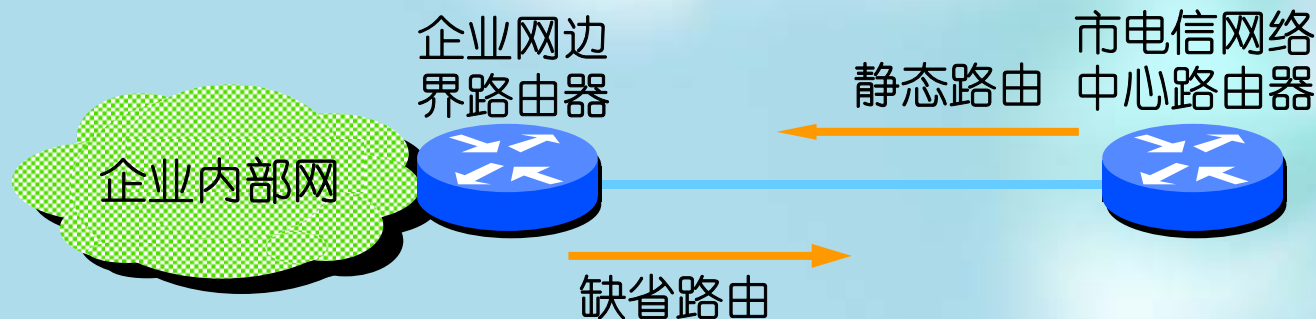
—— 由动态路由协议自动建立并维护路由表

优点：网络拓扑发生变化时，动态路由协议将自动更新路由表

缺点：路由器路由计算开销大

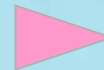
静态路由和缺省路由

- ❖ 缺省路由是静态路由的一个特例，也需要人工配置
- ❖ 互联网上有太多的网络和子网，受路由表大小的限制，路由器不可能也没有必要为互联网上所有网络和子网指明路径
- ❖ 凡是在路由表中无法查到的目标网络，在路由表中明确指定一个出口，这种路由方法称之为缺省路由



IP路由

- ❖ 路由器和路由表
- ❖ 静态路由和动态路由
- ❖ 分层路由
- ❖ **RIP**协议
- ❖ **OSPF**协议
- ❖ **BGP**协议
- ❖ **IP**多址传输

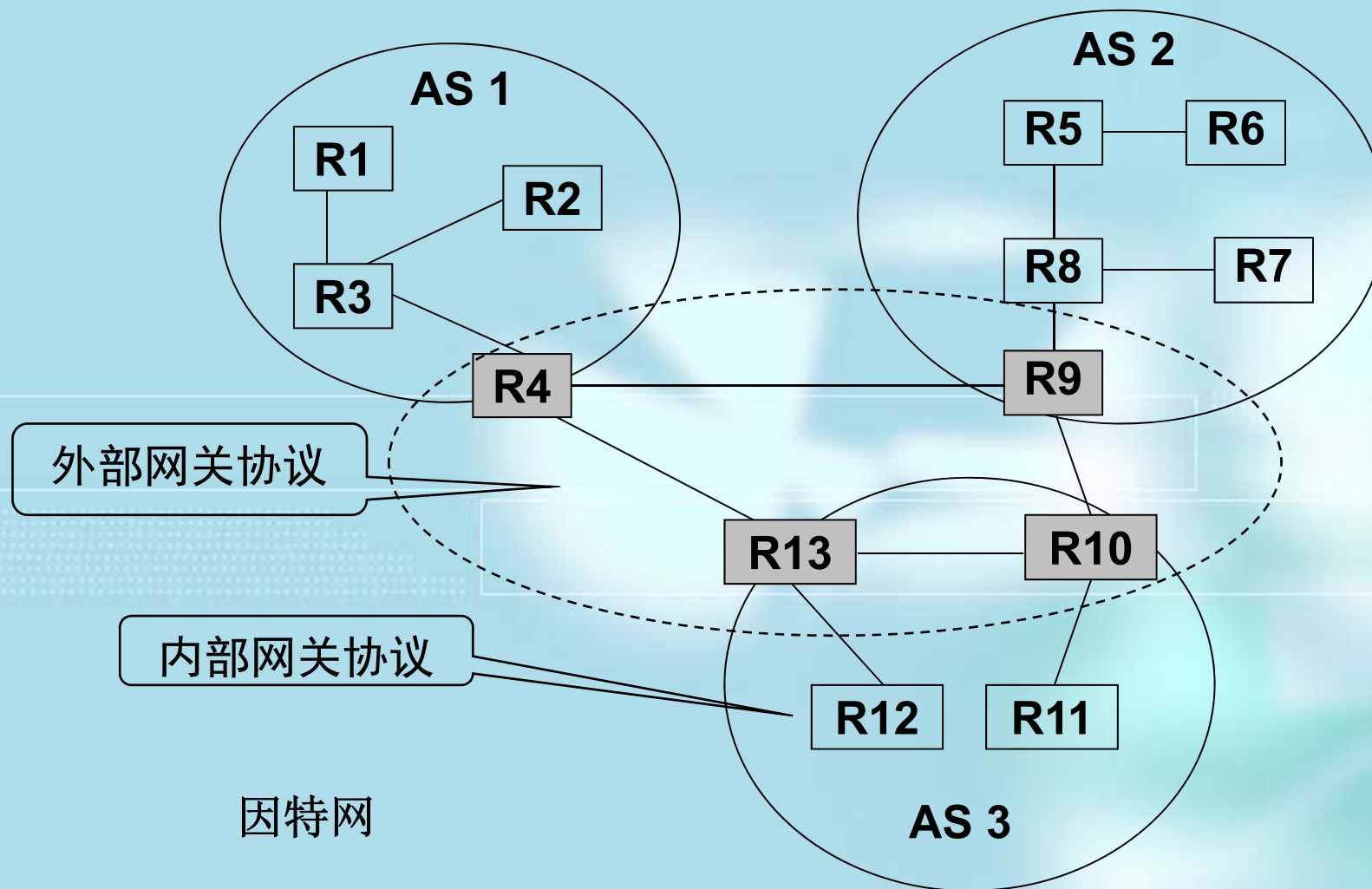


分层路由

- ❖ IP网采用分层路由的做法
- ❖ 为什么采用分层路由
 - 因特网规模太大
 - 许多单位不愿意外界了解自己单位的组网的细节和所采用的路由协议

自治系统 AS

- ❖ 一个自治系统由一组路由器组成，它们通过相同的路由选择协议交换信息
- ❖ 一个自治系统是由一个组织管理的一整套路由器和网络
- ❖ 除了发生故障的情况以外，一个自治系统是连通的（从图论的角度看），即在任意一对节点之间都存在一条通路



IGP和EGP

- ❖ **IGP**: 内部网关协议。在一个自治区域内所用的路由协议。常用的有**RIP**和**OSPF**
- ❖ **EGP**: 外部网关协议。自治区域之间的路由协议。常用的有**BGP4**

为什么要区分IGP和EGP

- ❖ AS由不同的机构管理运行
- ❖ 在AS内部，路由选择的标准很单一，即路径最优
- ❖ AS之间的路由，要选择最佳是不现实的，因为每个AS中的代价的含义不同。因此AS间的路由只交换可达性
- ❖ AS之间的通信会受到很多人为因素的影响，这些因素称为路由策略。例如，A到B之间最佳的线路是通过C，但C不愿意帮他们传输

IP路由

- ❖ 路由器和路由表
- ❖ 静态路由和动态路由
- ❖ 分层路由
- ❖ **RIP**协议
- ❖ **OSPF**协议
- ❖ **BGP**协议
- ❖ **IP**多址传输



RIP协议

- ❖ **RIP**采用**D-V**路由算法，是**Internet**的一个主要路由协议
- ❖ 传输层采用**UDP**协议
- ❖ 用跳数作为距离
- ❖ **RIP**协议有两个报文：请求包和响应包

RIP协议的格式

❖ 报文格式:

| | | |
|--------|-------|----|
| 命令 | 版本号 | 0 |
| 网络类型标志 | | |
| 网络地址 | | |
| 掩码 | | |
| | 路由器地址 | |
| 0 | | 距离 |

V1中为0, v2中为
自治区域标记

最多重
复**25**次

命令: **1**—请求包

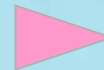
2—响应包

版本号: 有**1**和**2**两种, **V2**支持**VLSM** (可变长子网掩码)

距离: 一般为**hop**数, **<=15**

IP路由

- ❖ 路由器和路由表
- ❖ 静态路由和动态路由
- ❖ 分层路由
- ❖ **RIP**协议
- ❖ **OSPF**协议
- ❖ **BGP**协议
- ❖ **IP**多址传输



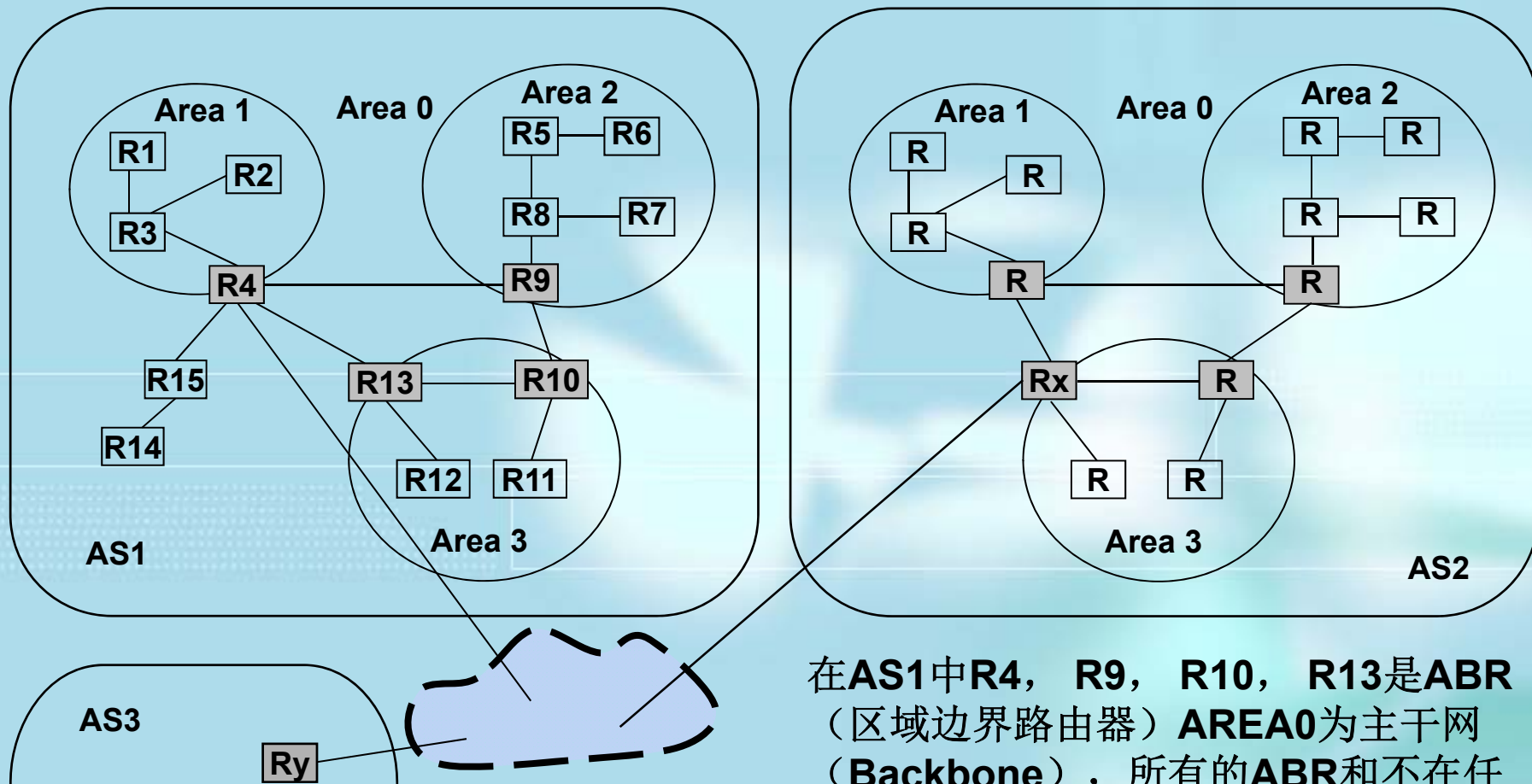
OSPF协议

- ❖ **OSPF**协议采用链路状态法，直接**IP**协议传送
- ❖ **OSPF**协议中的距离可以是真正的距离，也可以是费用、时延、带宽等
- ❖ 网络拓扑信息存放在网络拓扑数据库
- ❖ 只有当链路状态变化时，才发送信息

分层路由

- ❖ **OSPF**采用分层结构。使之能适合较大规模的网络
- ❖ 将整个网络分成若干个小区域
- ❖ 每个区域有一个区域边界路由器与其他区域相连
- ❖ 所有的区域边界路由器连接起来的区域称为区域**0**

区域划分举例



Tnbm P458 Fig. 5-65 在OSPF中, AS、Backbone、Area之间的关系

OSPF的路由器类型

- ❖ 内部路由器:

该路由器连接的网络在同一区域中，这些路由器有相同的网络拓扑信息

- ❖ 区域边界路由器**ABR**:

指连接两个或多个区域的路由器

- ❖ **AS边界路由器ASBR** :

连接多个**AS**的路由器

路由表的计算

- ❖ 路由器根据它所在的区域的数据库，计算到域内各网络的路由
- ❖ 路由器要根据**ABR**向本域散发的域外网络综合**LSA**来计算到本**AS**内的其他各区域子网的可达性信息
本路由器到本**AS**内目的网络的距离
$$= \text{本路由器到ABR的距离} + \text{ABR到目的网络的距离}$$
- ❖ 路由器根据与之邻接的**ABR**送来的**ASBR**掌握的本**AS**外网络综合**LSA**来计算到本**AS**外各网络的可达性
本路由器到本**AS**外目的网络的距离
$$= \text{本路由器到ASBR的距离} + \text{ASBR到目的网络的距离}$$

OSPF报文

- ❖ **HELLO**
- ❖ **Link State Update:** 链路状态更新发现相邻结点有变化时，向所有节点发送此报文
- ❖ **Link State ACK:** 确认链路状态更新消息
- ❖ **Database Description:** 数据库描述
- ❖ **Link State Request:** 链路状态请求

HELLO

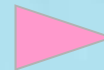
- ❖ 用来发现邻居路由器
- ❖ 相邻路由器每隔**10**秒交换一次**hello**分组
- ❖ 如**40**秒没有收到某个相邻路由器发来的**hello**，则认为该路由器是不可达的

指定路由器

- ❖ 若**N**个路由器连在一个以太网上，则每个路由器要向其他**N-1**个路由器发送链路状态，因而有 **$N(N-1)$** 个链路状态在网上传输
- ❖ 指定路由器代表该以太网上所有的路由器向与该以太网相连的个路由器发送链路状态，从而减少了广播信息

IP路由

- ❖ 路由器和路由表
- ❖ 静态路由和动态路由
- ❖ 分层路由
- ❖ **RIP**协议
- ❖ **OSPF**协议
- ❖ **BGP**协议
- ❖ **IP**多址传输



BGP协议（RFC 1654）

- ❖ **BGP（Border Gateway Protocol）** 考虑**AS**之间的路由，将一个**AS**看成一个节点
- ❖ 使用的是**D-V**算法，但只在路由状态发生变化时才发送变化信息
- ❖ 使用**TCP**类连接传输层，进行信息交换

BGP的配置

- ❖ 每个**AS**要选举一个代表，即**AS**边界路由器
- ❖ 创建一个**TCP**连接，用**179**端口

BGP-4的消息

BGP的当前版本称为**BGP-4**（RFC 1771）

- ❖ **OPEN:**

用于打开与另一个路由器的邻站关系

- ❖ **KEEPALIVE:**

用于确认一个**OPEN**报文，并且周期性地对邻站关系加以证实

- ❖ **UPDATE:**

用于传输有关一条路由信息和（或）列出多条被取消的路由

- ❖ **NOTIFICATION:**

在检测到错误状态时发送该报文

IP路由

- ❖ 路由器和路由表
- ❖ 静态路由和动态路由
- ❖ 分层路由
- ❖ **RIP**协议
- ❖ **OSPF**协议
- ❖ **BGP**协议
- ❖ **IP**多址传输



IP多址传输

❖ 两类组地址

永久地址：由Internet中央管理机构分配，永久存在

临时地址：根据需要创建，当成员数为**0**时被撤消

❖ 多目地址（D类地址）格式

| | | | | | |
|---|---|---|---|------|----|
| 0 | 1 | 2 | 3 | 4 | 31 |
| 1 | 1 | 1 | 0 | 组标地址 | |

即: 224.0.0.0 ~ 239.255.255.255

❖ 永久地址实例

224.0.0.1 – LAN上所有系统

224.0.0.2 – LAN上所有路由器

224.0.0.5 – LAN上所有OSPF路由器







224.0.0.6 – LAN上所有OSPF指定路由器

因特网中的网络层

- ❖ Internet 综述 
- ❖ IP 协议 
- ❖ IP 控制协议 
- ❖ IP 路由 
- ❖ IPv6 

IPv6

RFC1883 RFC1884







- ❖ IPv4的不足 
- ❖ IPv6的主要改进 
- ❖ IPv6数据报 
- ❖ IPv6的分段与重组 
- ❖ IPv6地址 
- ❖ IPv4到IPv6的过渡 

IPv4的不足

- ❖ 地址基本耗尽，这是当前最棘手的问题。
- ❖ 路由表越来越大
- ❖ 功能不足，缺少对多媒体信息传输的支持
- ❖ 缺少对高速传输的支持
- ❖ 缺少对安全的支持
- ❖ 缺少对主机漫游的支持

IPv6

RFC1883 RFC1884







- ❖ IPv4的不足 
- ❖ IPv6的主要改进 
- ❖ IPv6数据报 
- ❖ IPv6的分段与重组 
- ❖ IPv6地址 
- ❖ IPv4到IPv6的过渡 

IPv6的主要改进

- ❖ 更大的地址空间：**16**字节，**128**位
- ❖ 首部的 简化：只有**7**个固定域，撤消了有关分段的域和校验和域，以便更快地处理分组，提高路由器的吞吐量，缩短延时
- ❖ 更好地支持选项：选项是有次序的，以便路由器可简单地跳过与它无关的选项，加快分组的处理速度
- ❖ 增强了安全性：认证和隐私是关键特征
- ❖ 更加关注服务质量：以支持**Internet**上日益增长的多媒体应用

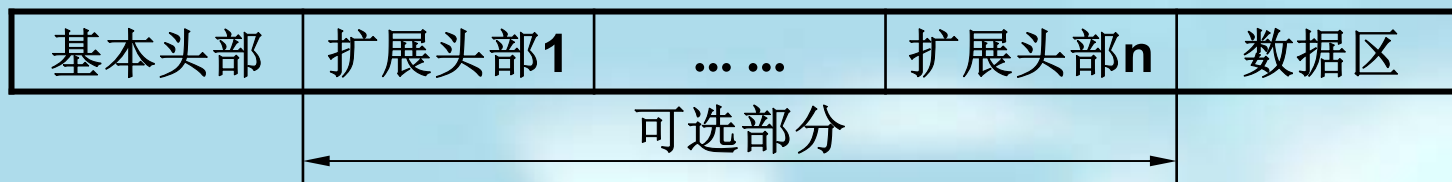
IPv6

RFC1883 RFC1884

- ❖ IPv4的不足 
- ❖ IPv6的主要改进 
- ❖ IPv6数据报 
- ❖ IPv6的分段与重组 
- ❖ IPv6地址 
- ❖ IPv4到IPv6的过渡 

IPv6数据报格式

❖ 基本头部和扩展头部



❖ 基本头部格式

| 0 | 4 | 8 | 16 | 24 | 31 |
|---------|-------------------|-------|---------|---------|----|
| 版 本 | 流量类别 | 流 标 签 | | | |
| 负 载 长 度 | | | 下 一 头 部 | 驿 站 限 制 | |
| | 源 地 址 (16字节) | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | 目 的 地 址 (16字节) | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

Tnbm P467 Fig. 5-68 IPv6的固定头部

格式说明

- ❖ 版本（**Version**）：0 ~ 3位
- ❖ 流量类别（**Traffic class**）：4 ~ 7位
 - 0 – 7：可进行拥塞控制，拥塞时可放慢传输速率
 - 8-15：实时应用，不可进行拥塞控制，不重发

| | |
|----|--|
| 0 | 应用层未指明数据包的优先级 |
| 1 | 如 USENET 报文 |
| 2 | 如电子邮件 |
| 4 | 大块数据传送，如 FTP 或 HTTP |
| 6 | 用于交互性业务，如 Telnet |
| 7 | Internet 控制业务，如 SNMP 和路由报文 |
| 8 | 丢弃数据包产生的影响最小，如高保真视频 |
| 15 | 丢弃数据产生的影响最大，如低保真音频 |
| 其他 | 保留今后使用 |

格式说明（续1）

❖ 流标签（**Flow label**）：8 ~ 31位

把在时间上敏感的一串报文，打上同一个标记，路由可优先通过

❖ 负载长度（**Payload length**）：16位

除基本报头以外的长度

格式说明（续2）

❖ 下一个头部（**Next header**）：8位

指出扩展头部是什么类型

一个**IPv6**报文可以带有零个、一个或多个扩展头，由前一个头中的下一个头部域进行说明，如有扩展头部，则说明扩展头部的类型，如没有其它扩展头部，则说明数据报中携带的数据类型，如：

| | | | |
|----------------------|----------------------------|------------|------|
| Base NEXT=TCP | TCP header | data | |
| Base NEXT=Routing | Routing header NEXT=TCP | TCP header | data |

格式说明（续3）

- ❖ 驿站限制（**Hop Limit**跳数限制）：**8位**
类似**IPv4**的生存时间（**Time To Live**），在经过每个节点转发时减一，当**Hop**数被减至零时则抛弃该报文
- ❖ 源、目的地址：**128位**
如按**IPv6**的**128位**地址均匀分配，意味着地球上每**M²**平均分配的地址数有 **7×10^{25}**

IPv6格式的扩展头部

IPv6定义的扩展头部应按顺序排列

| | |
|-----------------------------------|----------------|
| Hop by hop option | 给路由器的其它信息 |
| Destination options | 传送控制信息给目的地址 |
| Routing | 指定路由（必须经过的路由器） |
| Fragmentation | 作分段处理 |
| Authentication | 身份认证，安全性功能 |
| Encrypted security payload | 数据加密，安全性功能 |

Tnbn P470 Fig. 5-69 IPv6的扩展头部

IPv6

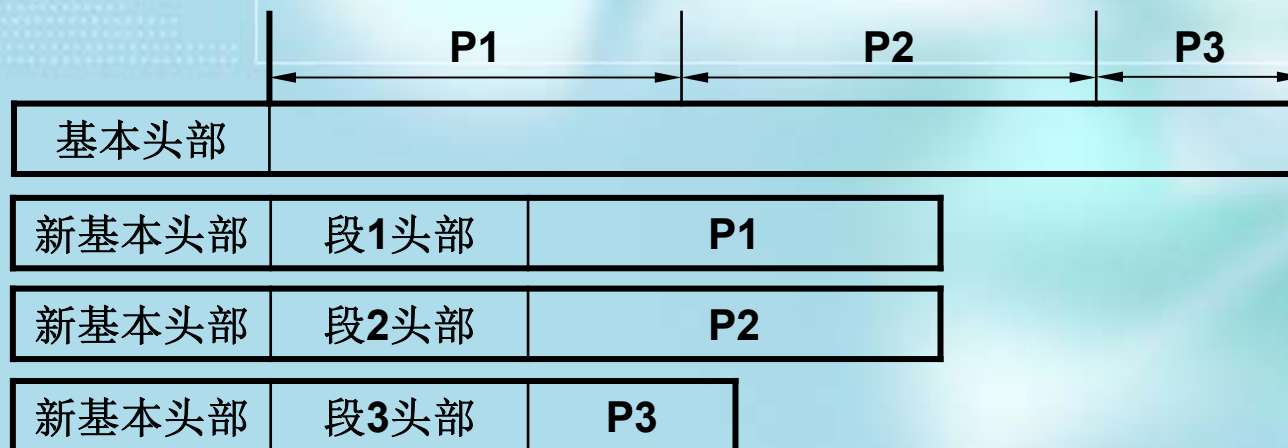
RFC1883 RFC1884

- ❖ IPv4的不足 
- ❖ IPv6的主要改进 
- ❖ IPv6数据报 
- ❖ IPv6的分段与重组 
- ❖ IPv6地址 
- ❖ IPv4到IPv6的过渡 

IPv6的分段与重组







- ❖ 每个分段将增加一个分段头部，并紧随基本头部之后
- ❖ 每个分段的基本头部即原基本头部，但其中的负载长度需作修改

一般情况下，要求报文长度小于**576**字节，若**>576**字节，由源站负责分段工作，沿途路由器不作分段和重组工作，路由器发现一个大于**MTU**的数据包，则丢弃，并发一**ICMP**消息给源站，由源站重新划分数据包并重发



IPv6

RFC1883 RFC1884

- ❖ IPv4的不足 
- ❖ IPv6的主要改进 
- ❖ IPv6数据报 
- ❖ IPv6的分段与重组 
- ❖ IPv6地址 
- ❖ IPv4到IPv6的过渡 

IPv6地址表示法

❖ 冒分十六进制表示法

X:X:X:X:X:X:X:X 其中X表示地址中16位二进制数的十六进制值

例：**FEDC:BA98:7654:3210:FEDC:BA98:7654:3210**

❖ 零压缩法

如其中有多于个连续的零，则可用零压缩法

如：**1080:0000:0000:0000:0008:0800:200C:417A**

可写成：**1080::8:800:200C:417A**

❖ IPv4地址在IPv6中的表示

IPv4地址202.120.5.100 可写成 ::202.120.5.100

IPv6地址类型

- ❖ 单目地址 **Unicast** 
- ❖ 多目地址（组地址） **Multicast** 
- ❖ **Anycast**地址 

单目地址 Unicast

❖ 标识一个接口的标识符，发送的数据报将被送到该地址所标识的接口上

❖ 地址格式：

| 3bits | 5bits | (16) | (24) | (32) | (48) |
|-------|-------|--------|-------|-------|-------|
| 010 | 注册 | 提供者标识符 | 用户标识符 | 子网标识符 | 节点标识符 |

单址传输

❖ 保留了两个地址：

| | | |
|------------------------|---------------------|--------------------------------------|
| 0:0:0:0:0:0:0:0 | 未定义地址 ::0 | 正初始化的主机在不知道自己的地址时，可将未定义地址填入数据报的源地址域中 |
| 0:0:0:0:0:0:0:1 | 回送地址 ::1 | 站点的发送数据报给自己时使用回送地址 |

IPv6地址类型

- ❖ 单目地址 **Unicast** 
- ❖ 多目地址（组地址） **Multicast** 
- ❖ **Anycast**地址 

多目地址（组地址） Multicast

- ❖ 标识一组接口（一般属于不同的站点）的标识符，发送的数据报将被送到该地址所标识的一组接口上
- ❖ **IPv6**没有广播地址，其功能由多目地址了实现

| | | | |
|----------|------|------|------|
| 8位 | 4位 | 4位 | 112位 |
| 11111111 | flag | scop | 组ID |

| | | |
|-------|------|--------------------------------|
| flag= | 0000 | 权威地址分配机构指定的永久分配（well know）多目地址 |
| | 0001 | 非永久分配的多目地址 |
| scop= | 0 | 保留 |
| | 1 | 本节点范围（node-local） |
| | 2 | 本链接范围（link-local） |
| | 5 | 本站点范围（site-local） |
| | 8 | 本组织机构范围（organization-local） |
| | E | 全球范围（global） |
| | F | 保留 |
| | 其余 | 未定义 |

IPv6地址类型

- ❖ 单目地址 **Unicast** 
- ❖ 多目地址（组地址） **Multicast** 
- ❖ **Anycast**地址 

Anycast地址







- ❖ 标识一组接口（一般属于不同的站点）的标识符，发送到一个**Anycast**地址的数据报将被送到该地址所标识的一组接口中的任意一个接口上（根据路由协议实测距离最近的一个接口）
- ❖ **multicast**与**anycast**地址的区别：**multicast**必须将数据报送到该组的每一个成员；**anycast**地址只需送到组中的任意一个地址，一般为最近的一个
- ❖ 所有类型的**Ipv6**地址均指定给接口，而不是节点，通常一个单目地址只能分配给一个接口，而一个接口却可以拥有多个任意类型（单目、多目、**Anycast**）的**Ipv6**地址

Anycast地址

| n bit | 128-n bits |
|-------|------------|
| 子网号 | 全0 |

IPv6

RFC1883 RFC1884

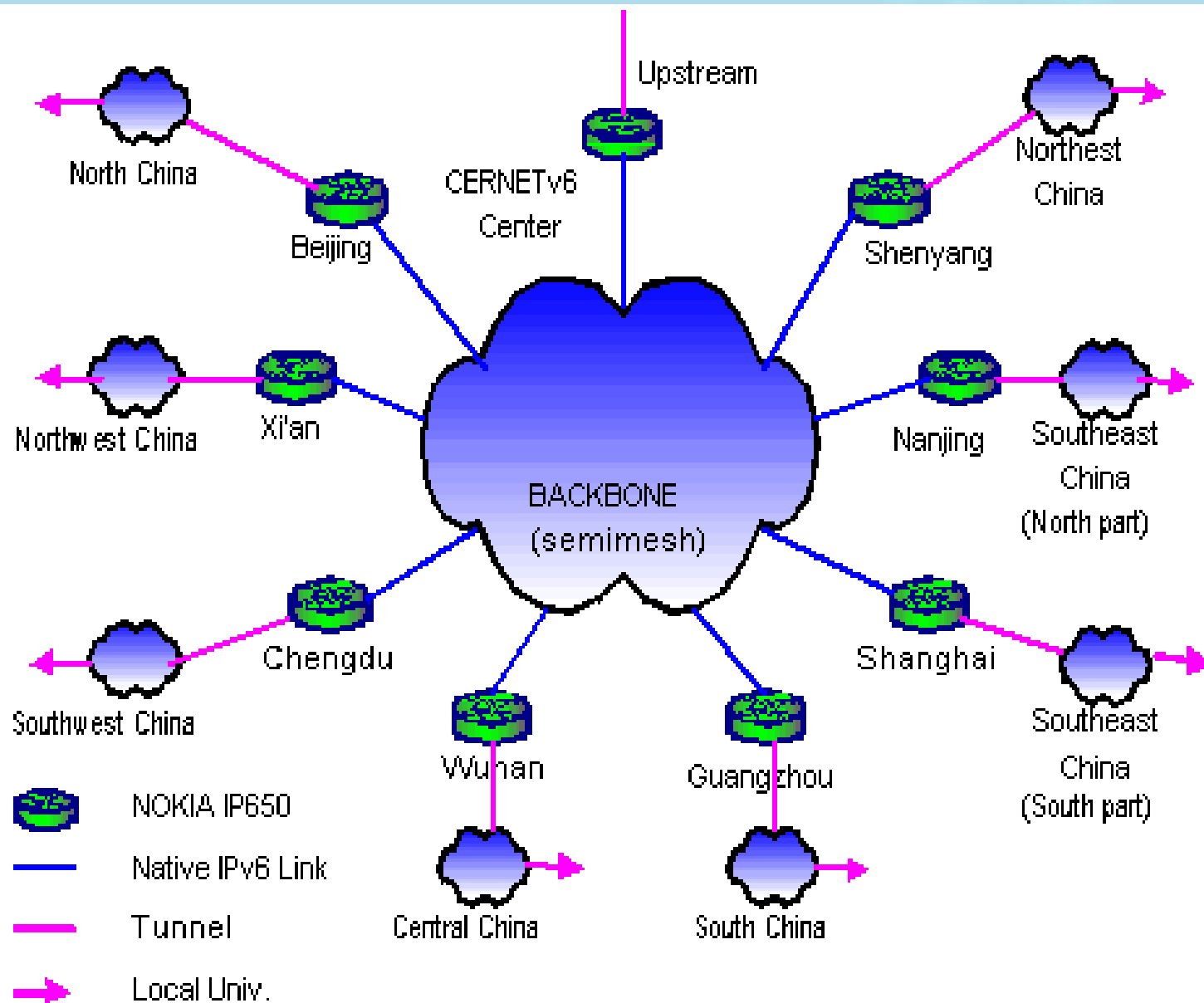
- ❖ IPv4的不足 
- ❖ IPv6的主要改进 
- ❖ IPv6数据报 
- ❖ IPv6的分段与重组 
- ❖ IPv6地址 
- ❖ IPv4到IPv6的过渡 

IPv4到IPv6的过渡

- ❖ 双协议站：过渡时期，站点必须同时支持**IPv4**和**IPv6**
- ❖ 隧道技术：**IPv6**主机之间通信必须使用**IPv4**的隧道
- ❖ 首部转换：用于发送方使用**IPv6**，而接收方使用**IPv4**

IPv6试验网——6bone

- ❖ 作为向下一代互联网络协议过渡的重要步骤，国际的**IPv6试验网——6bone**在**1996**年成立了。现在，**6bone**已经扩展到全球**50**多个国家和地区，成为**IPv6**研究者、开发者和实践者的主要平台。
- ❖ **CERNET(中国教育和科研网)**国家网络中心于**1998年6月**加入**6bone**，同年**11月**成为其骨干网成员。**1999**年，**CERNET**在国内教育网范围内组建了**IPv6**试验床，并从**6bone**获得**p-TLA (pseudo-Top Level Aggregation, 伪顶级聚类)** **3FFE:3200::/24**的地址空间；并且建立了**5**条以隧道为基础的国际**IPv6**虚拟链路，直接通达美国、英国和德国的**IPv6**网络，间接地与几乎所有现有的**6bone**成员互连。



本章作业

- ❖ 简述网络层的主要功能。
- ❖ 阐述静态路由算法中最短路径算法，自适应路由算法中的距离矢量算法及链路状态算法，拓扑相关路由算法中的逆向路径传送算法，**Ad-hoc**中的**AODV**按需路由算法。
- ❖ 简述拥塞控制的原理及一般方法。
- ❖ 阐述服务质量控制中的漏桶算法及令牌桶算法。
- ❖ 简述**IP**地址的表示和分类，子网掩码的主要功能。
- ❖ 简述**IPv4**和**IPv6**的主要特点。

本章复习

- ❖ **5.1 面向连接以及无连接服务的实现**
- ❖ **5.2 理解各种路由算法**
- ❖ **5.3 理解拥塞控制算法：途径 准入控制
流量调节等**
- ❖ **5.4 服务质量：流量整形**
- ❖ **5.5&5.6 IP地址分类； IPv4 IPv6**