



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
- 向上层提供服务



第5章 网络层

- ❖ 网络层设计的有关问题
- ❖ 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

❖ 网络层提供的服务



❖ 通信子网的结构





网络层的服务模式

- ❖ 服务应与通信子网的技术无关
- ❖ 通信子网的数量、类型和拓扑结构对于传输层来说是隐蔽的
- ❖ 传输层能获得的网络地址应采用统一的编号方式，即使跨越了多个**LAN和WAN**



网络层提供的服务类型

- ❖ 面向连接的服务：需事先建立连接
如X.25, ATM
- ❖ 无连接服务：不需要事先建立连接
IP网



网络层的设计

❖ 网络层提供的服务



❖ 通信子网的结构





通信子网的结构

❖ 虚电路子网

- 连接建立时选择一条路径
- 每个分组包含一个连接号
- 通信结束后，链路撤消

❖ 数据报子网

- 每个数据报包含全部的目的地
址，自行寻找路径



两种子网的比较

❖ 虚电路子网

- 通过路径选择后建立连接
- 到终点后无需重新排序
- 每个分组不需带目的地址，但带虚电路号（较短）
- 通信后撤销连接

❖ 数据报子网

- 每个分组分别选择最佳路径，健壮性较好
- 到终点后需重新排序
- 差错控制和排序工作由协议高层（主机）完成
- 每个分组必须带目的地址



不同的子网结构和服务的组合

子网结构 服务方式	数据报子网	虚电路子网
	面向无连接	面向连接
面向无连接	IP上的UDP	ATM上的IP
面向连接	IP上的TCP	ATM上的AAL1



第5章 网络层

- ❖ 网络层设计的有关问题
- ❖ 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，它决定进入的数据报应从哪一根输出线传输
- ❖ 如果是数据报子网，是在一个数据报到达时作此决定
- ❖ 如果是虚电路子网，是在虚电路建立时决定，数据包将沿此线路传输
- ❖ 路由与转发：路由是决定路线，转发是当一个数据包到达时发生的动作



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

❖ 路由算法的分类

➤ 静态算法

➤ 动态算法

❖ 路由算法中的度量标准

➤ 路径长度

➤ **hop**数

➤ 延迟时间



路由算法介绍

- ❖ 最短路径算法 (**Dijkstra**)
- ❖ 扩散法 (**flooding**)
- ❖ 距离矢量算法 (**D-V**)
- ❖ 链路状态算法 (**L-S**)
- ❖ 分层路由
- ❖ 广播路由
- ❖ 多址传输路由选择
- ❖ 对等网 (**Peer-to-Peer**) 中
节点的查找



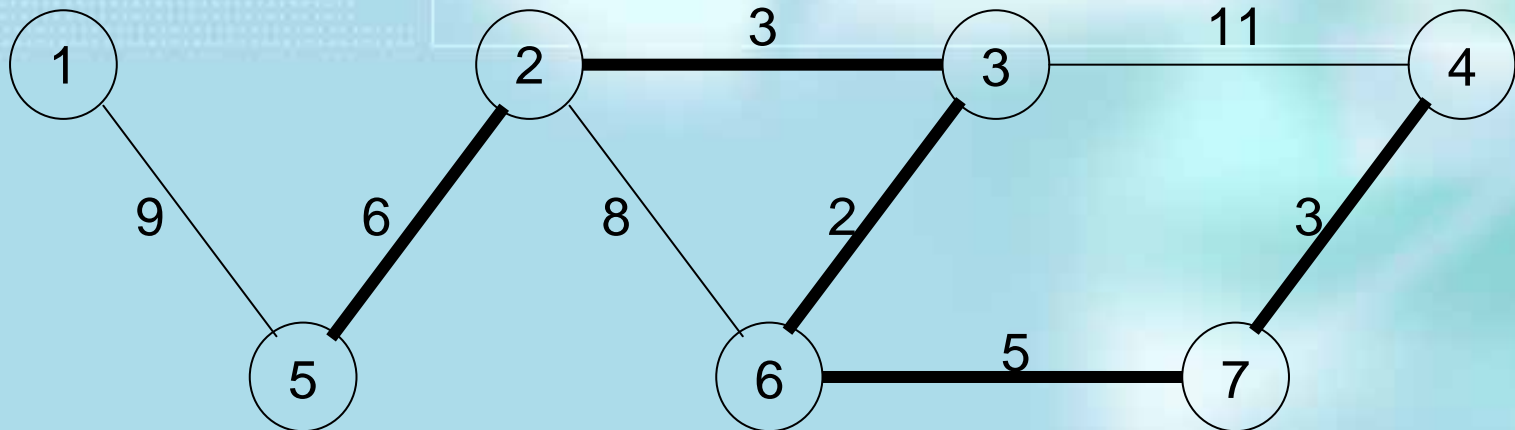


最短路由选择 (Dijkstra)

- ❖ **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**;

}

}

}



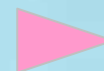
从5出发到各个节点的最短路径

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



路由算法介绍

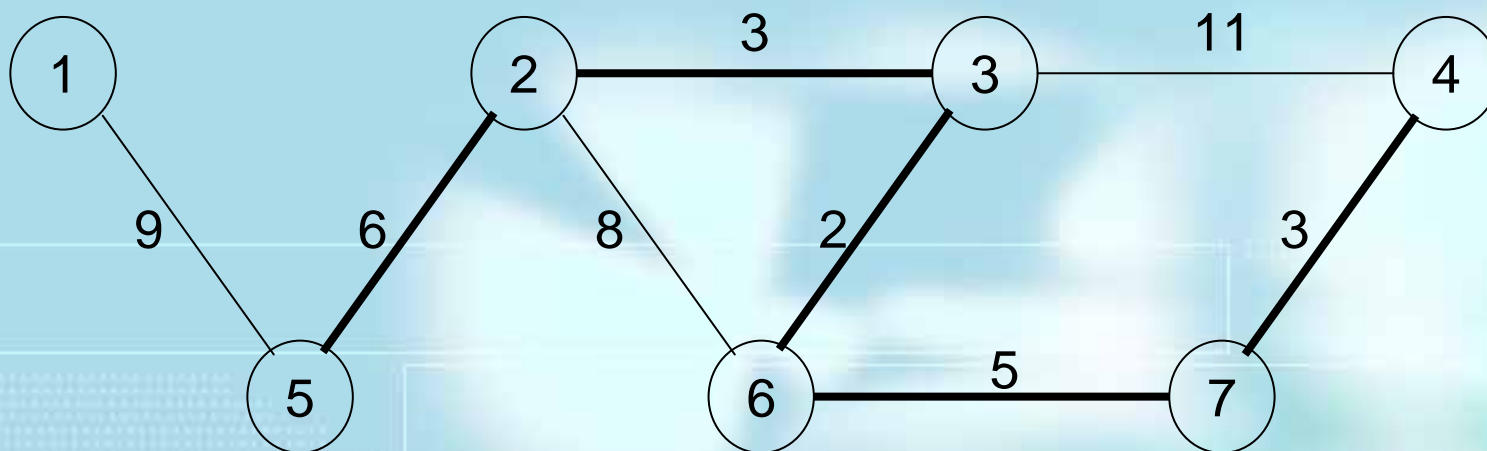
- ❖ 最短路径算法 (**Dijkstra**)
- ❖ 扩散法 (**flooding**)
- ❖ 距离矢量算法 (**D-V**)
- ❖ 链路状态算法 (**L-S**)
- ❖ 分层路由
- ❖ 广播路由
- ❖ 多址传输路由选择
- ❖ 对等网 (**Peer-to-Peer**) 中
节点的查找





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，每经过一个节点自动加1，达到规定值时，丢弃数据包
- 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



路由算法介绍

- ❖ 最短路径算法 (**Dijkstra**)
- ❖ 扩散法 (**flooding**)
- ❖ 距离矢量算法 (**D-V**)
- ❖ 链路状态算法 (**L-S**)
- ❖ 分层路由
- ❖ 广播路由
- ❖ 多址传输路由选择
- ❖ 对等网 (**Peer-to-Peer**) 中
节点的查找





距离矢量算法 (**D-V**)

(**Distance Vector Routing**)

❖ 是动态、分布式算法，**RIP**协议中使用本算法，较小系统中常使用**RIP**

❖ 实现分布式算法的三要素：

The measurement process (测量)

The update protocol (更新邻接点距离矢量)

The calculation (计算)



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

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

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

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)

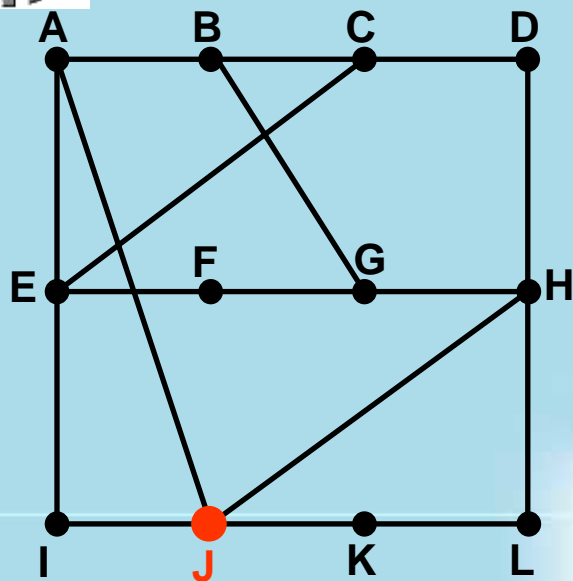
$$S_{ij} = x \quad (\text{从i到j途经的下一个节点为} x)$$

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：节点A和I都是各自测得的距离，且不一定是同一时刻测得的，线路状态是动态变化的


所谓节点即路由器
当前节点为J

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

▼	线路
8	A
20	A
28	I
20	H
17	I
30	I
18	H
12	H
10	I
0	-
6	K
15	K
节点J的 新路由表	



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） 
- ❖ 距离矢量中不考虑带宽因子
- ❖ 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

Tnbnm P359 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



路由算法介绍

- ❖ 最短路径算法 (**Dijkstra**)
- ❖ 扩散法 (**flooding**)
- ❖ 距离矢量算法 (**D-V**)
- ❖ 链路状态算法 (**L-S**)
- ❖ 分层路由
- ❖ 广播路由
- ❖ 多址传输路由选择
- ❖ 对等网 (**Peer-to-Peer**) 中
节点的查找










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，另一端的路由器发送回来一个应答来说明它是谁



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



链路状态算法 (L-S)

(Link State Routing)

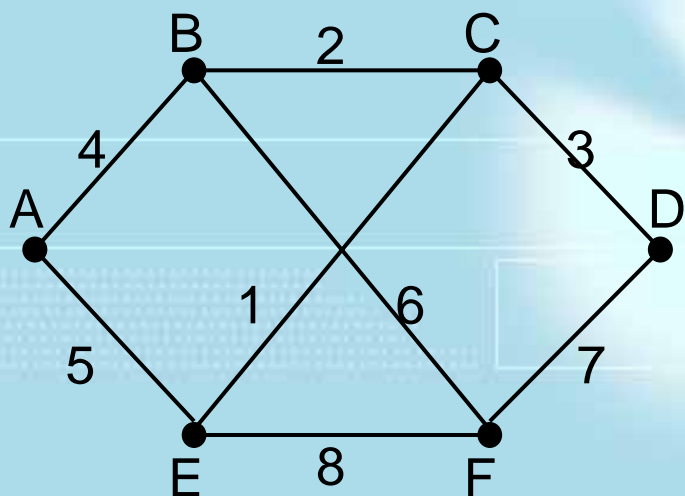
基本思想:

- ❖ 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

- ❖ 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	

Tnbn 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**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

各路由器的路由信息的一致性好

收敛性好，坏消息也一样传播得快

适用于大型网络，报文长度与网络规模关系不大

❖ LSR的缺点

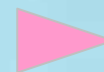
每个路由器需要有较大的存储空间

计算工作量大



路由算法介绍

- ❖ 最短路径算法 (**Dijkstra**)
- ❖ 扩散法 (**flooding**)
- ❖ 距离矢量算法 (**D-V**)
- ❖ 链路状态算法 (**L-S**)
- ❖ 分层路由
- ❖ 广播路由
- ❖ 多址传输路由选择
- ❖ 对等网 (**Peer-to-Peer**) 中
节点的查找



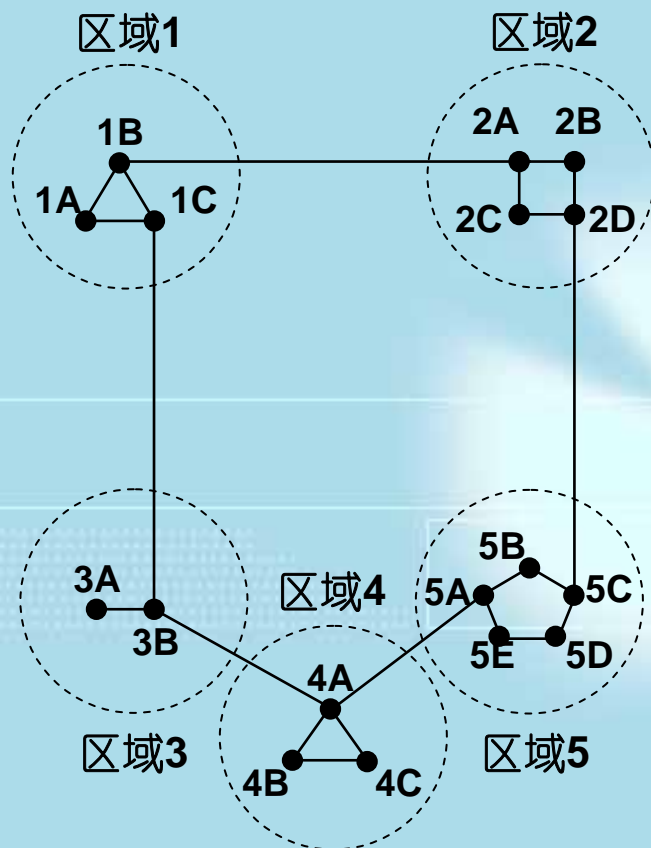


分层路由

- ❖ 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，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



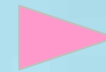
如何决定分层的层数

- ❖ **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 $e \ln N$ 个路由条目



路由算法介绍

- ❖ 最短路径算法 (**Dijkstra**)
- ❖ 扩散法 (**flooding**)
- ❖ 距离矢量算法 (**D-V**)
- ❖ 链路状态算法 (**L-S**)
- ❖ 分层路由
- ❖ 广播路由
- ❖ 多址传输路由选择
- ❖ 对等网 (**Peer-to-Peer**) 中
节点的查找





广播路由

- ❖ 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

- ❖ 多目的地路由



- ❖ 生成树法 (**spanning tree**)



- ❖ 逆向路径传送



(**reverse path forwarding**)



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

- ❖ 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

- ❖ 多目的地路由

- ❖ 生成树法（**spanning tree**）

- ❖ 逆向路径传送

（**reverse path forwarding**）





生成树法

❖ 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

- ❖ 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

- ❖ 多目的地路由



- ❖ 生成树法（**spanning tree**）



- ❖ 逆向路径传送

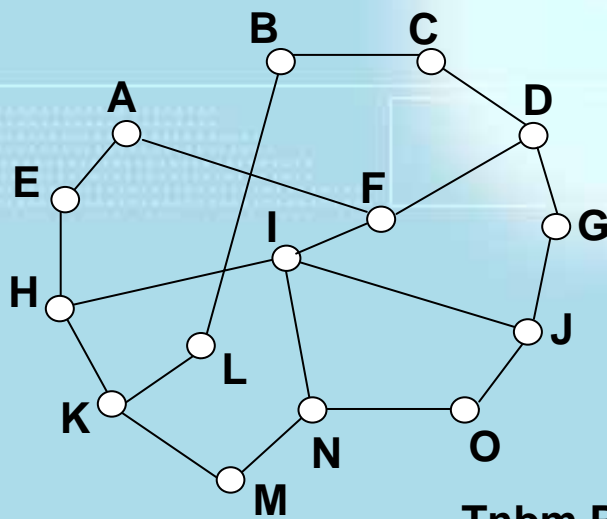


（**reverse path forwarding**）



逆向路径传送

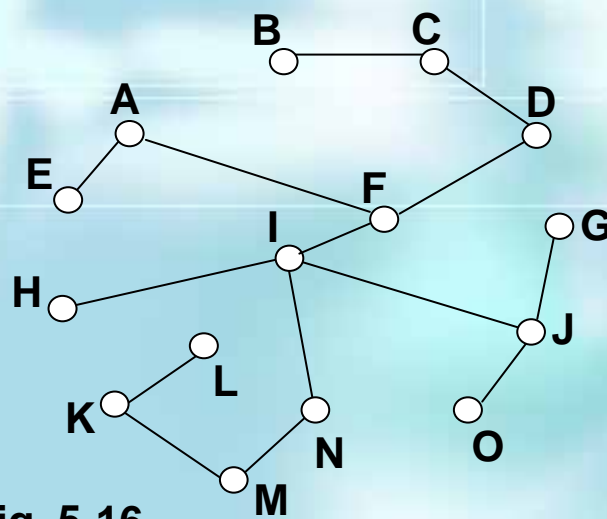
- ❖ 基本原理：当某一广播分组到达路由器时，路由器对它进行检查，如该分组来自通常向广播源发送分组的线路，则将该分组转发到除进线以外的其它线路，否则丢弃



(a) 一个子网

Tnbnm P369 Fig. 5-16

逆向路径传送



(b) 一棵生成树



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，假设每个节点的路由表中到节点I的路径中的下一跳分别为：

A	B	C	D	E	F	G	H	J	K	L	M	N	O
F	C	D	F	A	I	J	I	I	M	K	N	I	J
I	D	F	I	F		I			N	M	I		I
	F	I		I					I	N			
	I									I			

跨一步即到节点I的有F、H、J、N

跨一步即到节点F的有A、D

跨一步即到节点H的没有

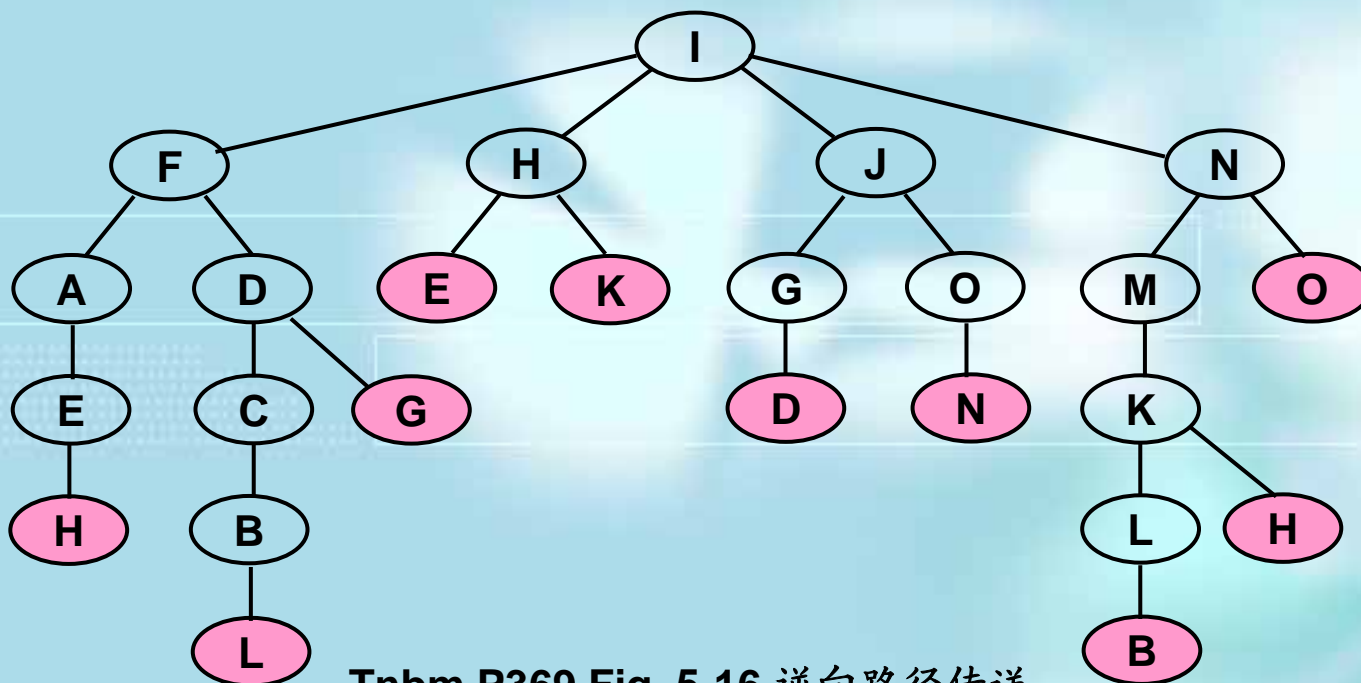
跨一步即到节点J的有G、O

跨一步即到节点N的有M



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 P369 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

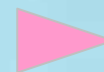
优点：算法合理，容易实现

缺点：对大型网络不适用



路由算法介绍

- ❖ 最短路径算法 (**Dijkstra**)
- ❖ 扩散法 (**flooding**)
- ❖ 距离矢量算法 (**D-V**)
- ❖ 链路状态算法 (**L-S**)
- ❖ 分层路由
- ❖ 广播路由
- ❖ 多址传输路由选择
- ❖ 对等网 (**Peer-to-Peer**) 中
节点的查找





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 P437 Fig. 5-55
IP地址格式



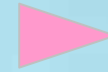
核心树(core-based tree)

- ❖ 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



路由算法介绍

- ❖ 最短路径算法 (**Dijkstra**)
- ❖ 扩散法 (**flooding**)
- ❖ 距离矢量算法 (**D-V**)
- ❖ 链路状态算法 (**L-S**)
- ❖ 分层路由
- ❖ 广播路由
- ❖ 多址传输路由选择
- ❖ 对等网 (**Peer-to-Peer**) 中
节点的查找





对等网 (Peer-to-Peer) 中 节点的查找

- ❖ 对等网：大量的用户通过固定线路接入 Internet，共享资源
- ❖ 对等网的特点：
 - 全分布：所有节点都是对称的，没有中心控制或分层的概念
 - 每个节点都有一些其他用户感兴趣的信息
- ❖ 对等网的路由：没有一个中心数据库，如何发现要寻找的信息



Chord算法

假设:

- ❖ 系统中有 n 个用户
- ❖ 每个用户都有可提供的数据, 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)
- ❖ 所有 2^m 个标识符按递增次序排成一个圈, 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)

- ❖ 信息的名称同样被散列到一个**m**位的数字，称为键值**key**
- ❖ 信息的索引是在所有节点上随机分布的，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将被存在同一节点
- ❖ 信息的查找：当某个用户需要查找名字为**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

- ❖ 每个节点保存直接前驱和直接后继，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**个条目，编号从**0**到**m-1**



Chord算法优化 (续1)

❖ finger table表的内容

每个条目指向一个真实存在的节点，每个条目有两个字段：**start**和**successor(start)**的IP地址，节点**k**中的条目**i**的值为：

$$\text{Start} = k + 2^i \pmod{2^m}$$

Successor(start[i])的IP地址



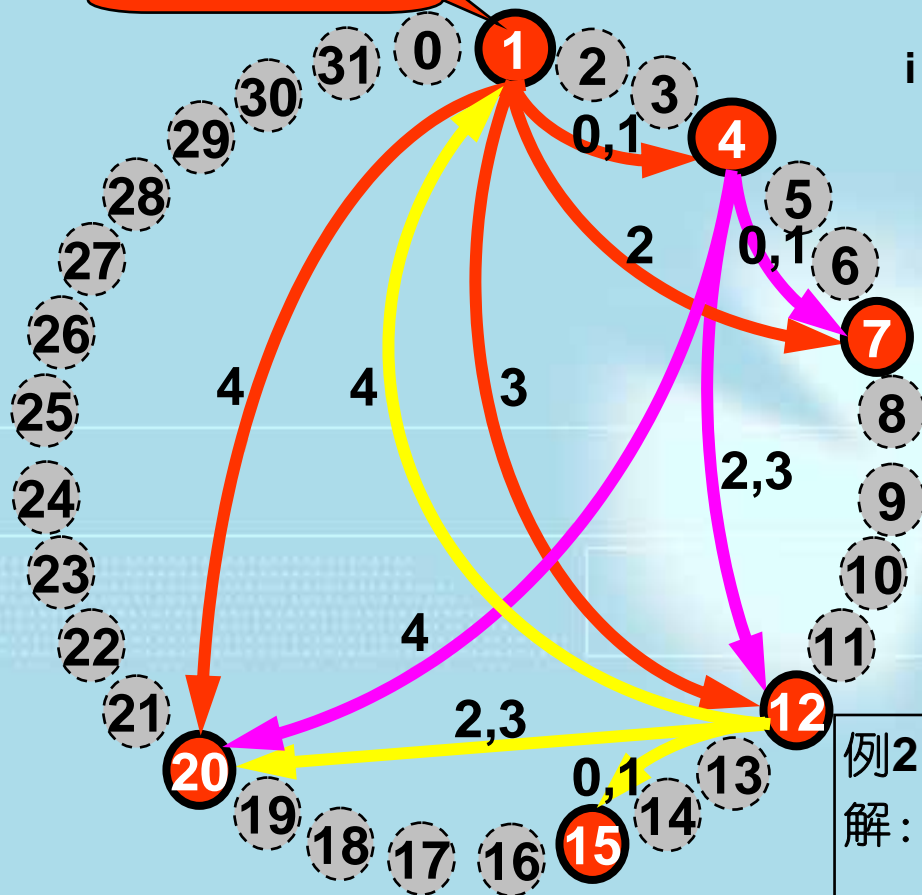
Chord算法优化 (续2)

- ❖ 在节点**k**上查找键值**key**的过程:
 - 如果**key**值介于**k**和**successor(k)**之间, 那么包含**key**信息的节点是**successor(k)**, 查找结束
 - 查找**finger**表, 找出**start**值最接近于**key**的前驱, 直接发一请求给相应的**IP**地址, 请求它继续查找



Chord算法实例

当前在线的节点



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

节点1 节点4 节点12

例1：在节点1上查找键值3
解：键值介于1和4之间，返回4

例2：在节点1上查找键值14
解：键值不介于1和4之间，查找14的最直接的前趋，即9，返回条目9对应的地址12，在节点12，它查到键值是介于它和它的后继15之间，于是返回15

Tnbn P382 Fig. 5-24 Chord算法实例



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**询问它的直接前驱**p**
 - 向**s**和**p**申请加入环
 - **s**将**p+1**到**r**的信息转储到**r**上，添加即告结束
 - 其他**finger**表中的问题有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**
 - 通知前驱**p**，它的后继变为**s**



第5章 网络层

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





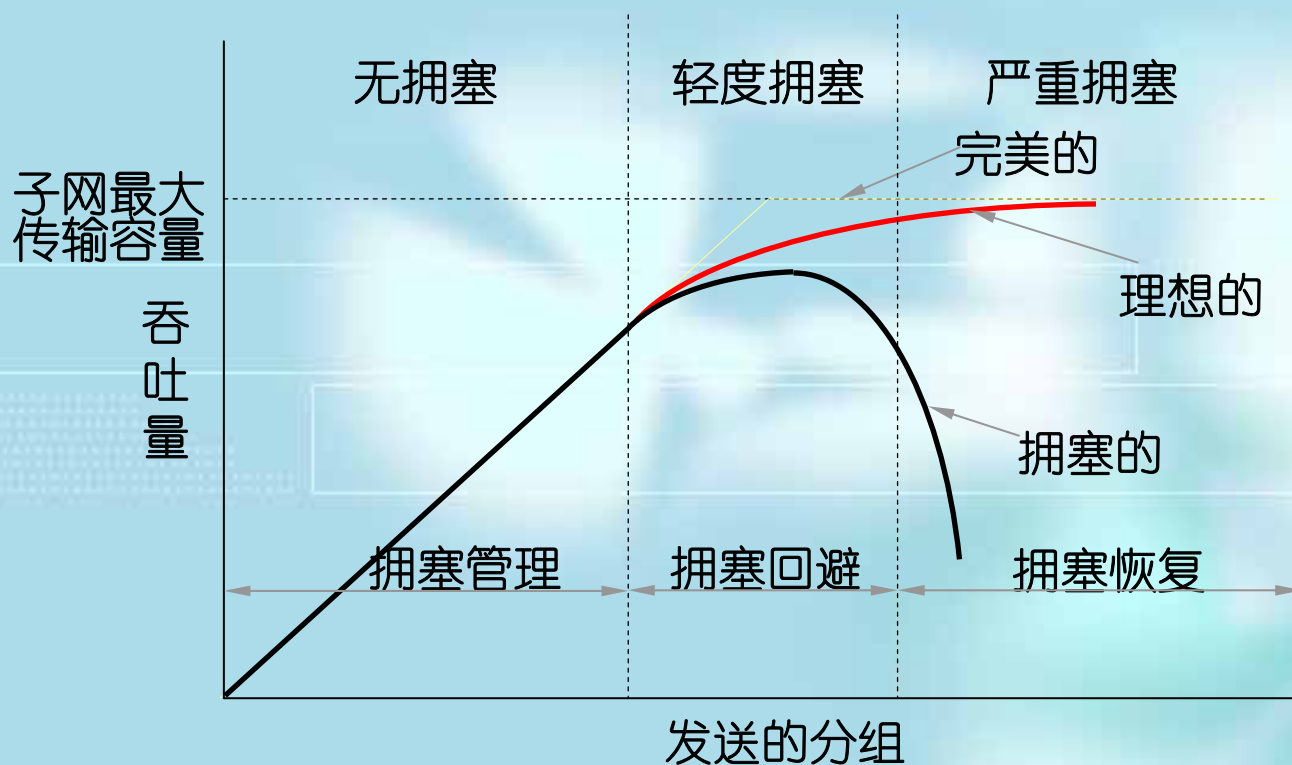
拥塞控制

- ❖ 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

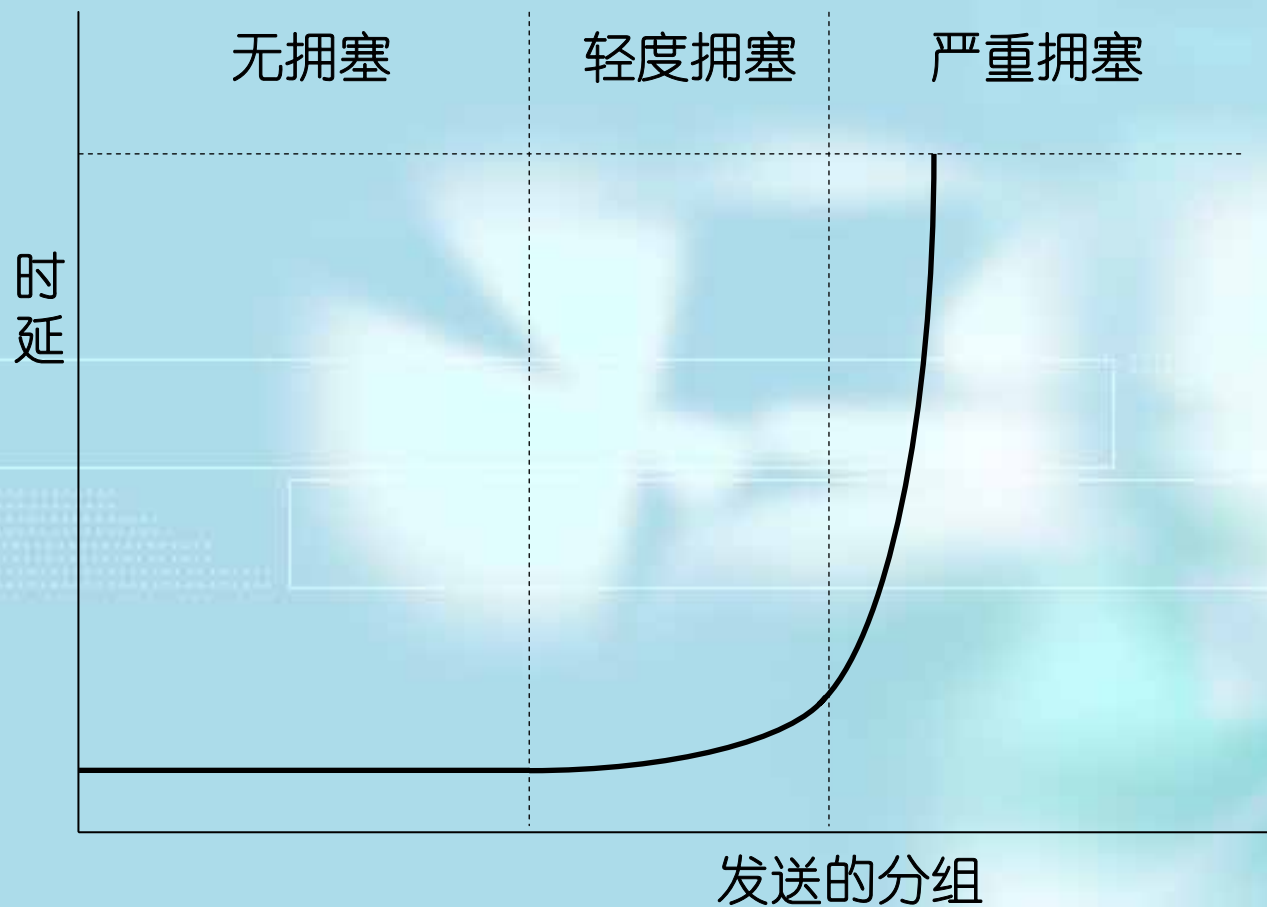
❖ 当通信量太大时，发生拥塞，性能显著降低



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
- ❖ 数据包子网的拥塞控制
- ❖ 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流
 - 什么时候开始丢弃数据报，丢弃那些数据报？
 - 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
- ❖ 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包文
- ❖ 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
- ❖ 数据包子网的拥塞控制
- ❖ 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影响拥塞

层 次	策 略
传输层	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	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、
数据链路层	滑动窗口策略、差错恢复策略、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 
- ❖ 数据包子网的拥塞控制 
- ❖ 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
- ❖ 允许建立新的虚电路，但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
- ❖ 数据包子网的拥塞控制
- ❖ 载荷脱落
- ❖ 抖动控制





数据包子网的拥塞控制

在路由器上监视每条输出线的利用率，当利用率超过某一域值时，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，在数据包头上设置某一位警告位，当数据包到达接收方时，传输实体将警告位拷贝到**ACK**上，发送方能得知拥塞的发生，以调整它的发送速率

❖ 阻塞数据包

拥塞的路由器直接发一个数据包给发送源

❖ Hop-by-Hop阻塞包

直接发阻塞包给发送源的方法反应太慢，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，一直到发送源



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
- ❖ 数据包子网的拥塞控制
- ❖ 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) :

在情况还没有太糟之前就采取行动

- 何时开始丢弃？通过观察队列的长度，当长度超过阈值时，表示拥塞即将发生
- 丢弃哪个数据包？在队列中随机选取一个

❖ 如何通知发送源拥塞的发生？

- 在丢包的同时，直接发送一个阻塞包
- 丢包时不发送阻塞包，当发送方没有收到**ACK**，则知道数据包丢失，可能是拥塞发生了



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
- ❖ 数据包子网的拥塞控制
- ❖ 载荷脱落
- ❖ 抖动控制





抖动控制

- ❖ 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

- ❖ 网络层设计的有关问题
- ❖ 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数据包称为流
- ❖ 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

Tnbnm 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，均衡路由是把数据分散到多条路径上去



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

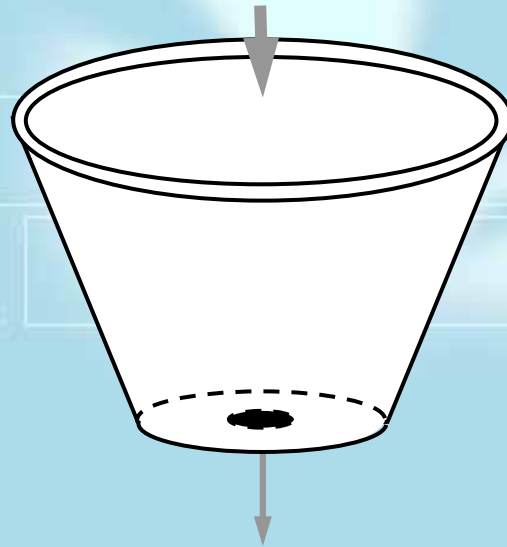
- ❖ 数据包调度





漏桶算法 (Leaky Bucket Algorithm)

❖ 平滑输入流量，控制进入网络的流量

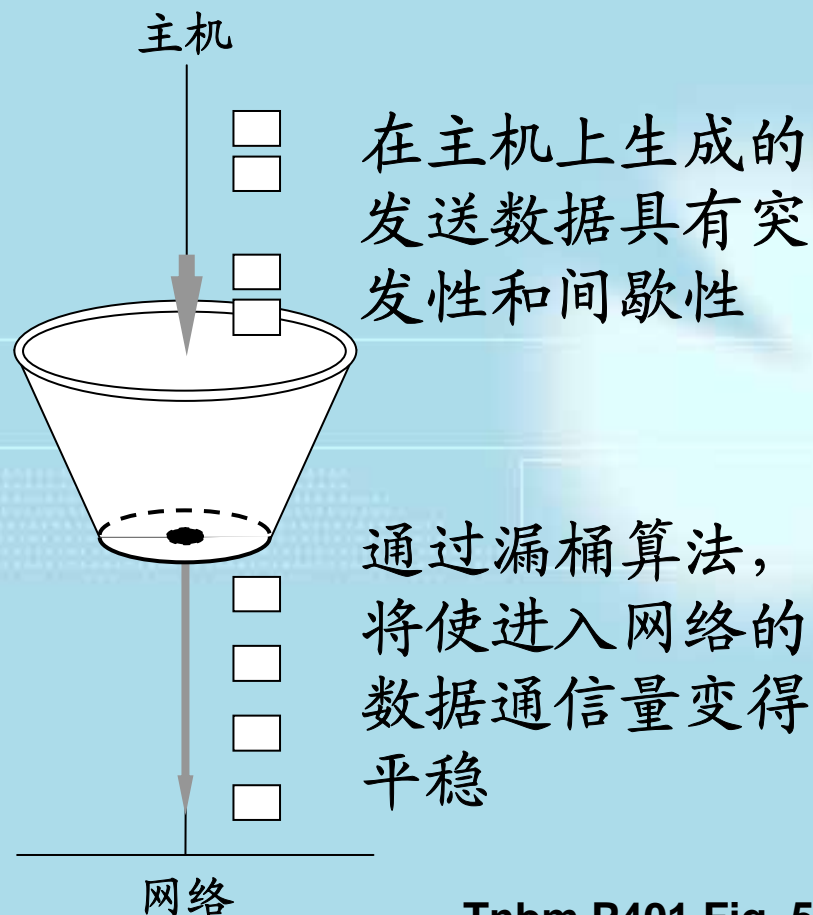


每单位时间泄漏一定量的字节数

Tnbn P401 Fig. 5-32 (a)漏桶



漏桶算法中的主机与网络



漏桶深 如 C=1M byte	即缓冲区的容量
数据生成速率为 25M byte/s	突发数据通信量为 1M byte 如超过 1M 将溢出
路由器的最佳工作速率是 2M byte/s	输出速率恒定 如 1M 的漏桶需传输 500ms

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

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 $C = 1\text{M byte}$	$C = 1\text{M}$
路由器的最佳工作速率是 2M byte/s	$\rho = 2\text{M}$
数据生成速率为 25M byte/s $r = 25\text{M/s}$	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

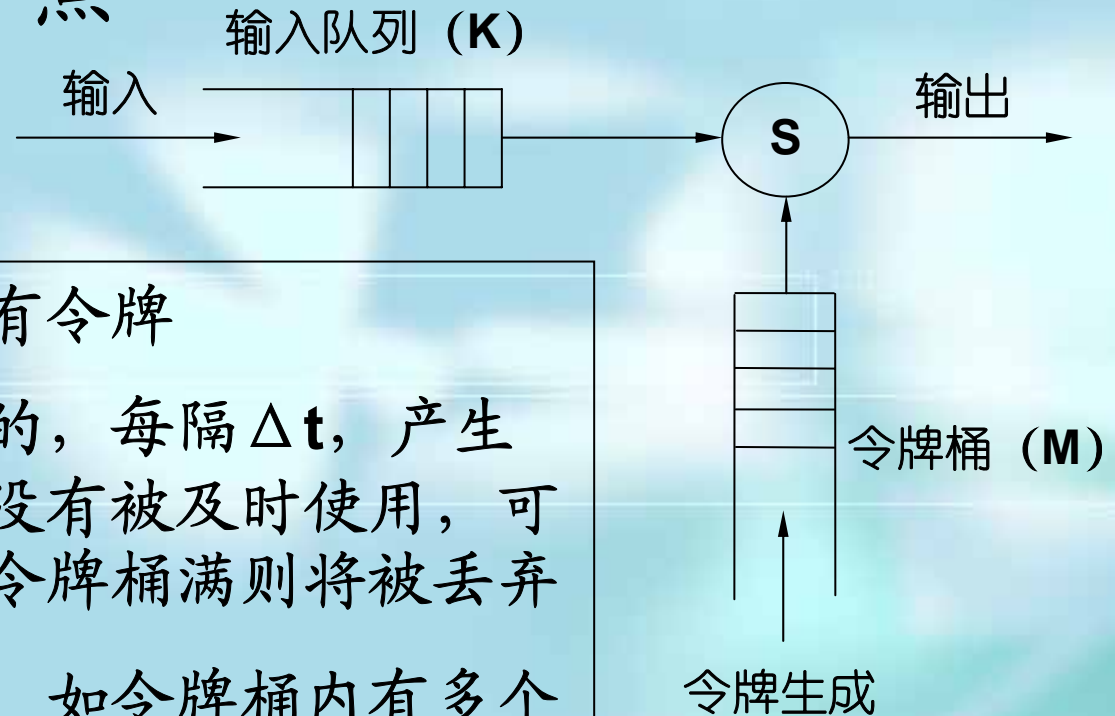
- ❖ 数据包调度





令牌桶算法(Token bucket Algorithm)

- ❖ 希望当大的突发流量到来时，输出也能相应加快一点



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，则突发数据可及时输出



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)

$$= 250/23 \text{ (ms)} \quad \text{约为} 11 \text{ ms}$$



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蕴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
最小分组大小	最小分组的字节数
最大分组大小 (字节数)	最大分组能有多大
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 说明了每秒钟注入桶中多少字节及桶的大小
令牌桶大小 (字节数)	
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，即说明了令牌桶为空的最短时间间隔



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留的带宽

❖ 公平排队



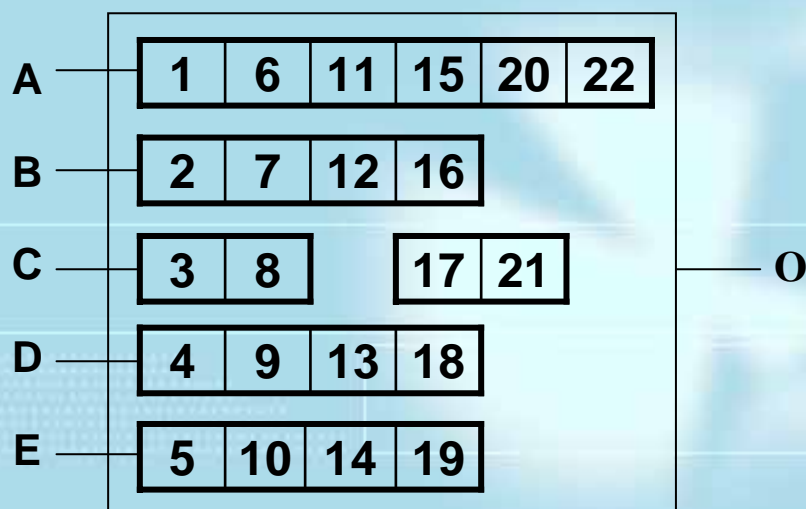
❖ 加权公平排队





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)

Tnbn 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

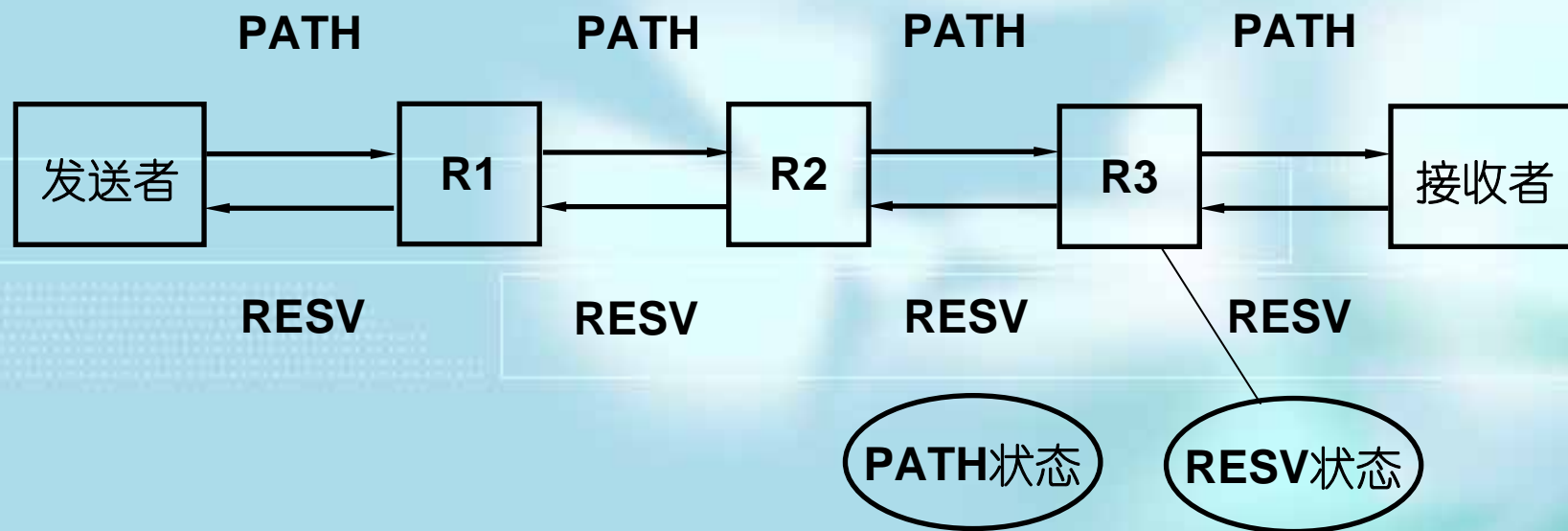
- ❖ 最常用的资源预留协议是**RSVP**
(**Resource reSerVation protocol**)

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

❖ 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，
给每个数据包一个丢弃概率：高、中、低，组合
起来，一共有**12**种类别

➤ 数据包的处理：





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)

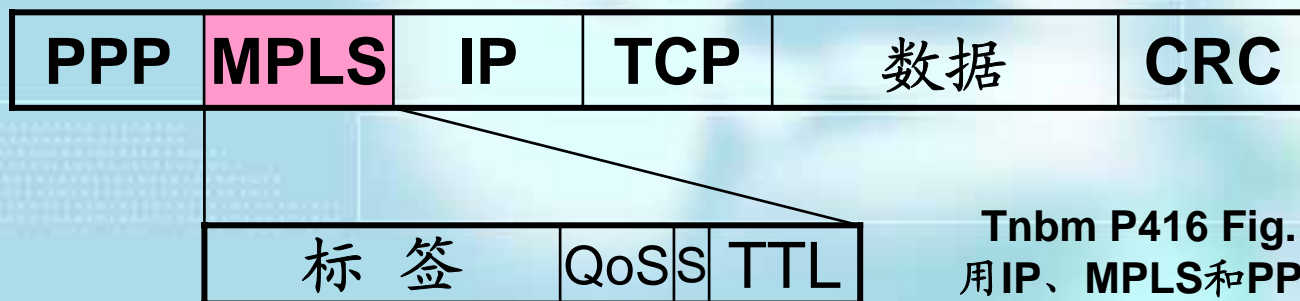
- ❖ 类似于虚电路方式：在数据包头上加一个标签，一般为转发表中的索引，在中间路由器上根据标签而不是根据目的地址作路由，这样可加速转发速度
- ❖ **Label switching**也被称为**tag switching**



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



Tnbn 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

- ❖ 网络层设计的有关问题
- ❖ 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

❖ 网络互联的类型

LAN-LAN LAN-WAN

WAN-WAN LAN-WAN-LAN

❖ 网络互联的设备

中继器 网桥 路由器 传输网关 应用网关
交换机



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
编址	平面编址还是层次编址
多址传输	支持还是不支持
数据包大小	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
拥塞控制	漏桶、令牌桶
安全	加密方法
参数	不同的超时设置，流量说明
计费	根据连接时间、数据包数、字节数



互联网络的相关技术

❖ 互联方式



❖ 互联网络的路由



❖ **Packet**的分段与重组





互联方式

❖ 连锁虚电路

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

- 不能保证数据包按顺序到达
- 分组格式的转换
- 不同网络有不同的地址编制方法

设计一个通用的互联网分组和地址

❖ 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

❖ 互联方式



❖ 互联网络的路由



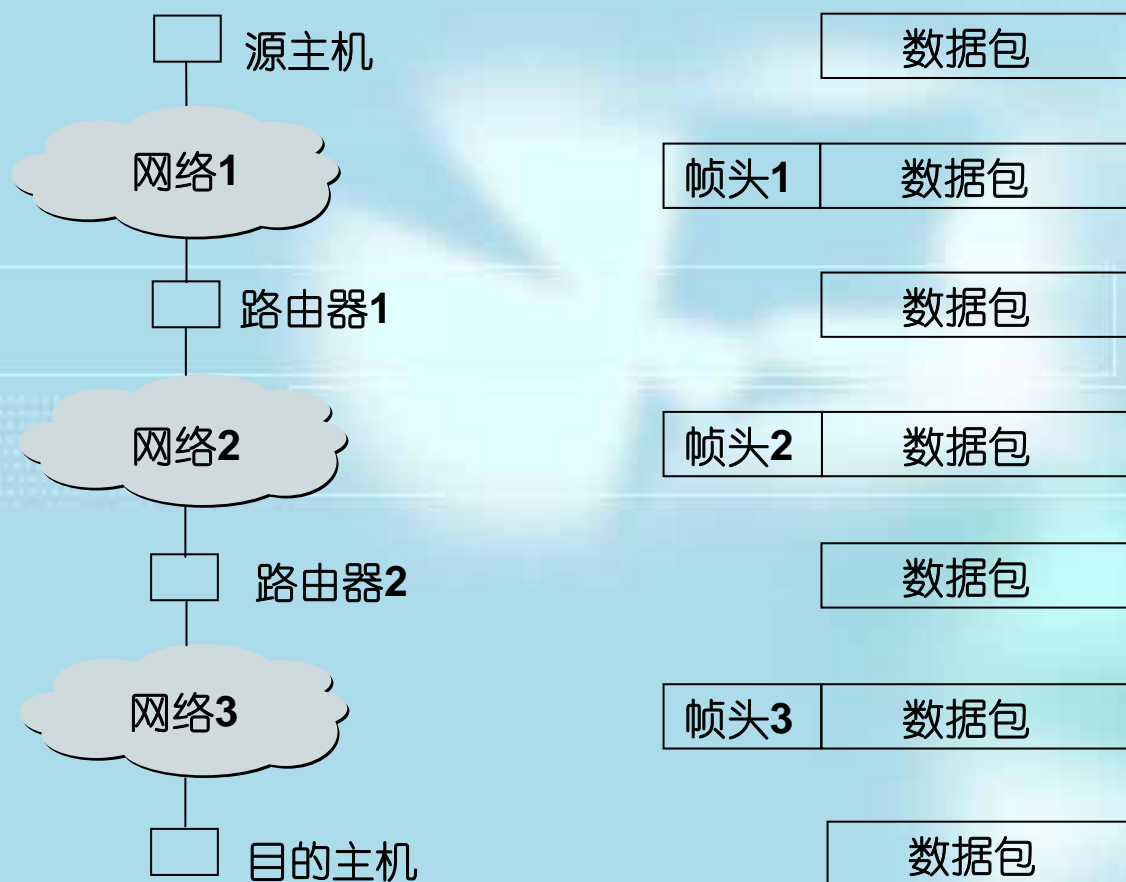
❖ **Packet**的分段与重组





Packet的分段与重组

❖ 不同的网络其帧的最大长度**MTU**也不同

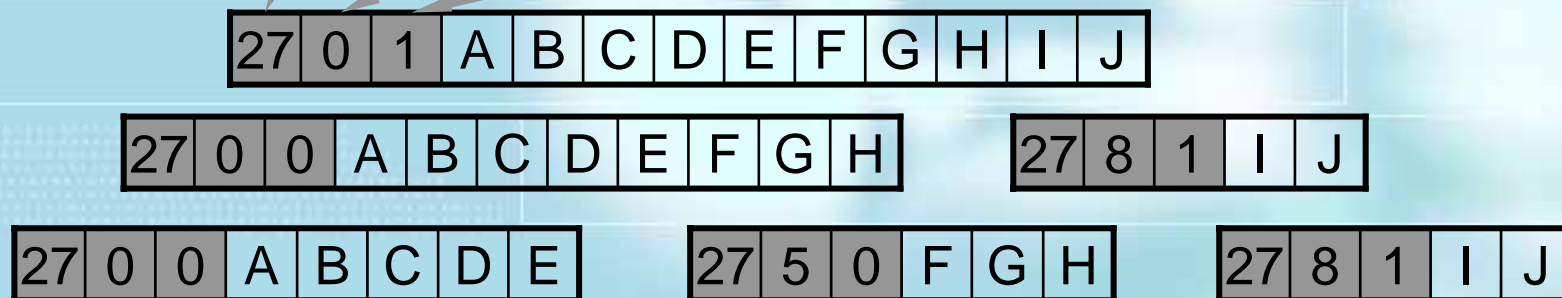




分段

每个报头都包含三个字段：

分组号，偏移量和分组结束标志





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

- ❖ 网络层设计的有关问题
- ❖ 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

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
安全	指出数据包的加秘方式
严格源路由	数据包必须严格按此路径转发
松散源路由	必须经过给出的这些路由器
记录路由	每个途径的路由器把它的IP地址存入此数据包
时间戳	每个途径的路由器把它的IP地址和经过时间存入此数据包

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：
 - 属原来的哪个段
 - 属原来段中的第几个分段
 - 哪一个为段尾



通过标志、标识和段偏移实现

❖ 标识（**identifier**）：16位

发送方每发送一个报文编号加一

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

0	1	2
保留	DF	MF

DF = 0 允许分段

= 1 不允许分段 如刚启动时，即**Boot**时不允许分段

MF = 0 最后一段

= 1 段未结束

❖ 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**，现要传输一个数据报（其报头为**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)

- ❖ 每个分段的报头其基本部分（如源地址、目的地址等）是**copy**原数据报的报头，与分段相关的域则应重新生成

		原报头	分段1报头	分段2报头	分段3报头
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

$$75 \times 8 = 600$$



数据报分段的重组

- ❖ 重组是在各分段都到达目的地后才进行
每个分段可以走不同的路径
减少路由器中保存的信息量及路由器的工作量
- ❖ 途中的任意一个路由器都无法重组
重组必须在所有的分段全部收到后，才可进行
- ❖ 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

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

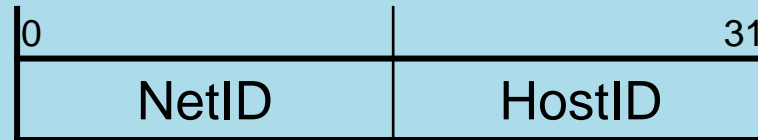
❖ NAT-Network Address



Translation



IP地址的层次结构和分类



❖ IP地址分为A、B、C、D、E类

	0	8	16	31	
A类	0	前缀	后缀		大规模网络
B类	10	前缀	后缀		中规模网络
C类	110	前缀	后缀		小规模网络
D类	1110	多址传送地址			
E类	1111	保留将来使用			

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
网络ID	全0	网络	标识一个网络
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
127	任意	回送	测试 (127.0.0.1)

Tnbn P438 Fig. 5-56
特殊IP地址

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



私有网络的IP地址

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

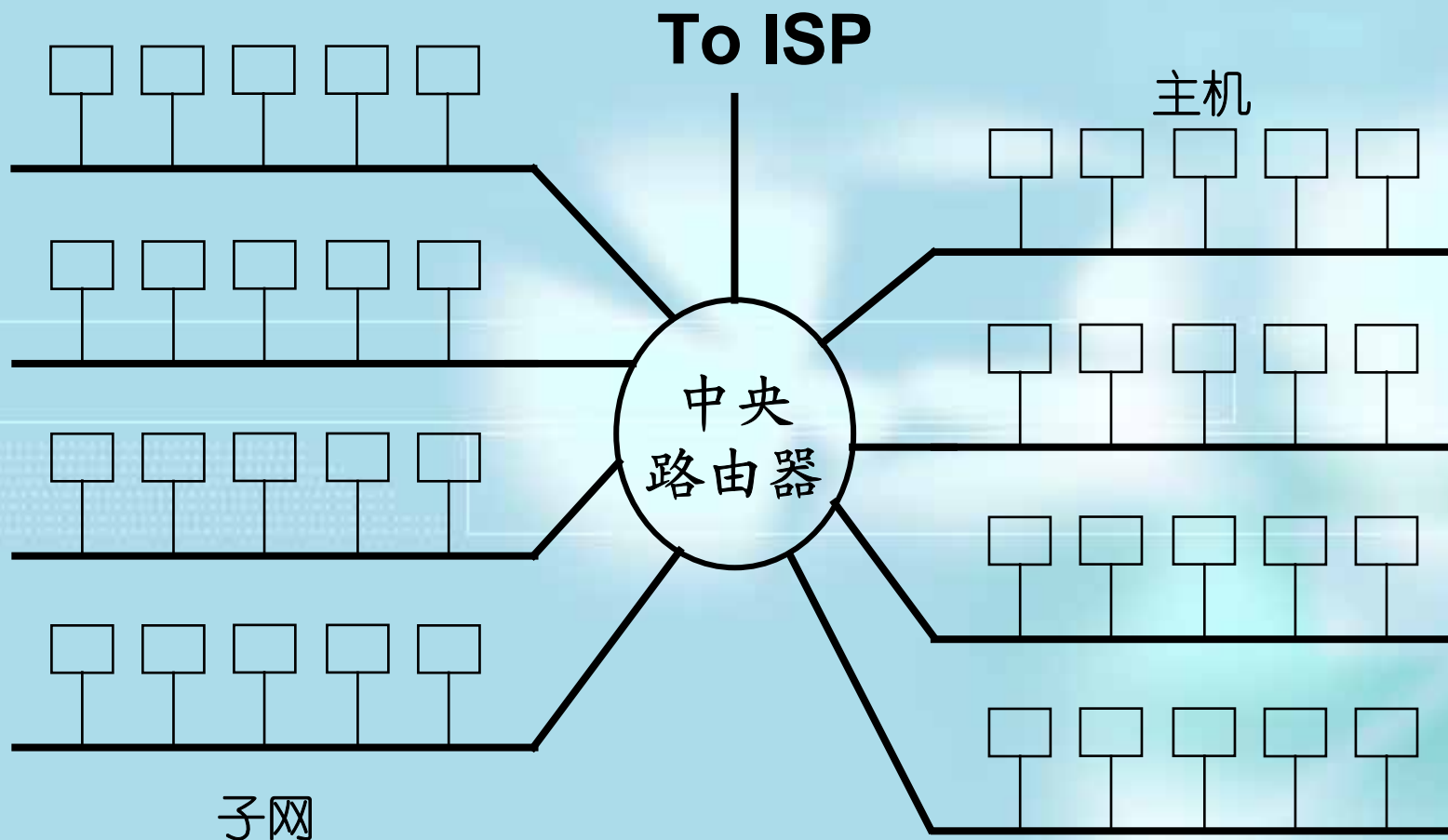


子网划分

- ❖ 随着网络规模的扩大，虽然能提供大量的**IP**地址（如**B**类地址），但一个以太网网段支撑不了那么多的主机
- ❖ 如果再申请一个网络地址，那将造成地址的浪费
- ❖ 问题的关键在于一个网络地址只能对应于一个网络，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几个部分，而对外仍像一个网络



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，所以告知设备地址的哪一部分是包含子网的网络地址段，地址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**

子网地址 = **011**的子网掩码是: **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 \wedge 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
掩 码	11111111	11111111	11111111	111	00000
	255	255	255		224

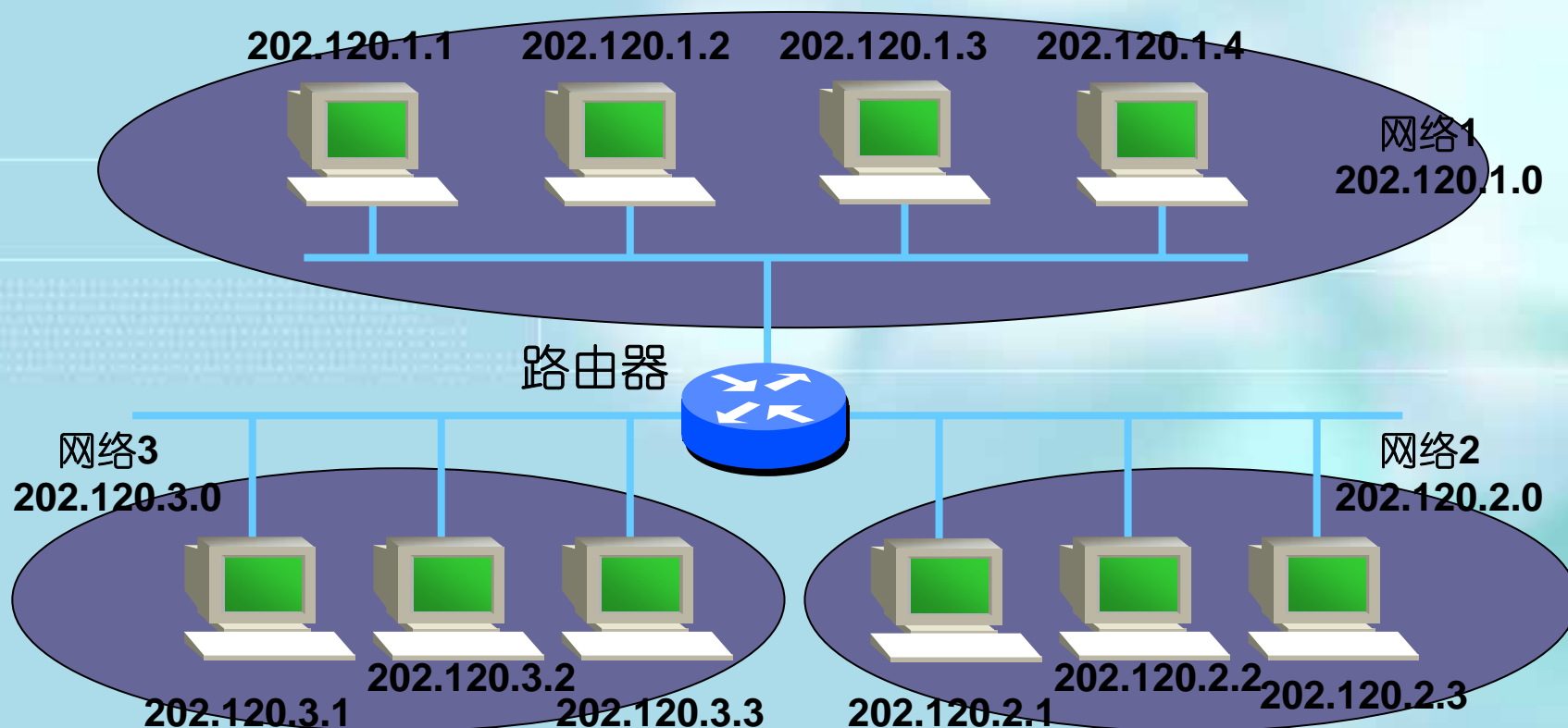
掩码也可用更简洁的方式表示:

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，由于每个部门的主机数都比较少，因此只需要三个**C**类地址（没有比**C**类地址更小的了）



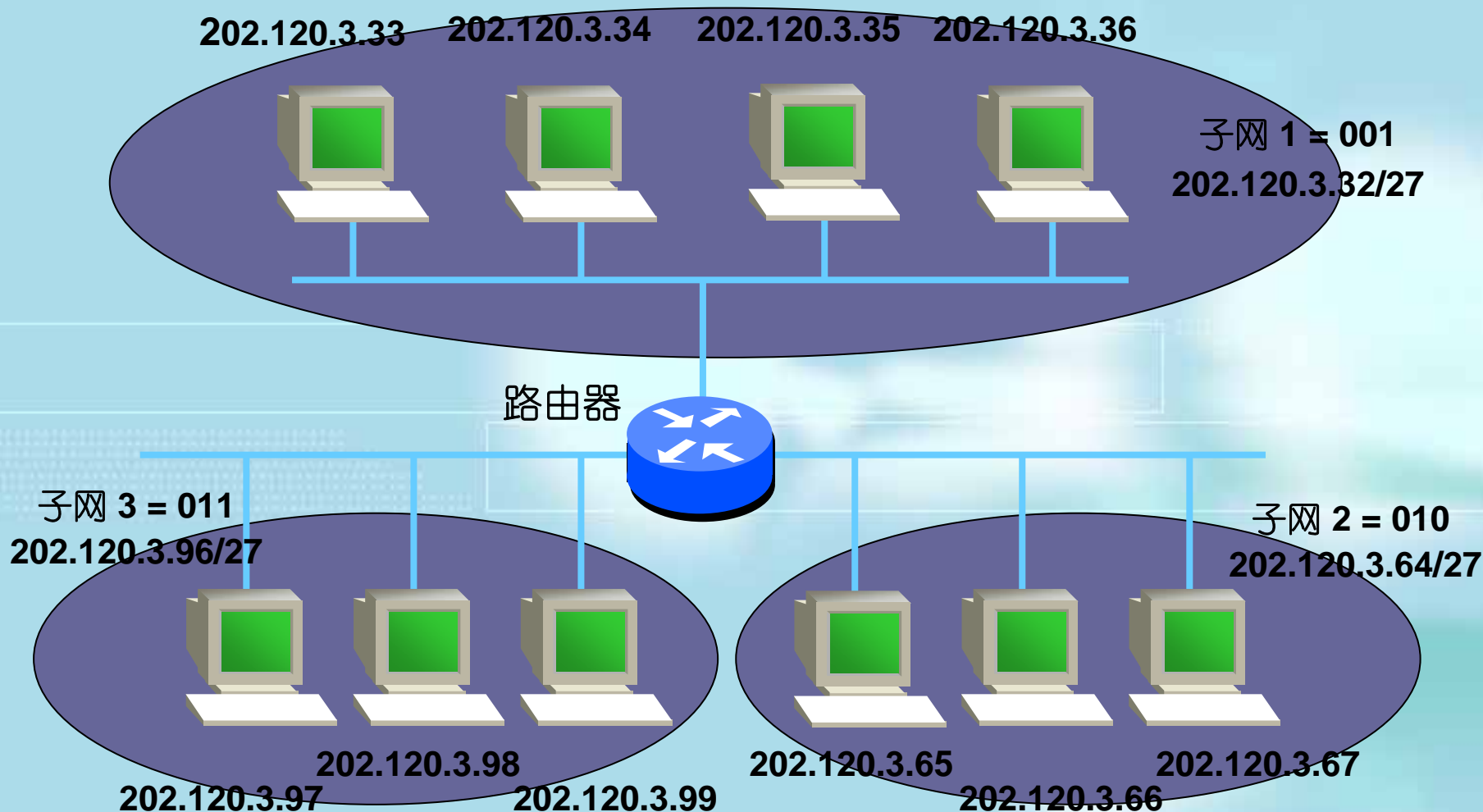


选择子网号长度

- ❖ 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





子网和子网掩码

例：某单位有一C类地址 **202.10.23.0**，该单位有多个部门，每个部门的机器数为**20**个左右，问如何确定子网掩码？最多能有几个子网？每个子网的主机数为多少？

掩码：**255.255.255.224** 最多的子网个数：**8(6)**个

某个主机**IP**地址及其掩码也可写成：**202.10.23.47/27**

每个子网的主机数为： **$32 - 2 = 30$** （台）



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）合成一个组地址，作为一个分配单元
- 例如：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

我校地址： **202.120.0.0 ~ 202.120.63.0**



CIDR举例

- ❖ 剑桥大学的一台计算机，其IP地址为**194.24.6.112**，掩码为**255.255.248.0**，牛津大学的一台计算机，其IP地址为**194.24.23.112**，掩码为**255.255.240.0**，爱丁堡大学的一台计算机，其IP地址为**194.24.10.112**，掩码为**255.255.252.0**

单位	地址数	地址范围	掩码
剑桥	2048	194.24.0.0 ~ 194.24.7.255	255.255.248.0
牛津	4096	194.24.16.0 ~ 194.24.31.255	255.255.240.0
爱丁堡	1024	194.24.8.0 ~ 194.24.11.255	255.255.252.0



CIDR举例 (续)

- ❖ 在路由器中，按掩码中含1的位数最多的优先做匹配，以确定该分组送哪一个路由器

Eg. 目的地址为**194.24.17.4**的分组到达时的处理

	194.24.17.4	11000010	00011000	00010001	00000100
爱丁堡掩码	255.255.252.0	11111111	11111111	11111100	00000000
	网络号	11000010	00011000	00010000	00000000
		194.24.16.0不是爱丁堡的起始网络号			
剑桥掩码	255.255.248.0	11111111	11111111	11111000	00000000
	网络号	11000010	00011000	00010000	00000000
		194.24.16.0不是剑桥的起始网络号			
牛津掩码	255.255.240.0	11111111	11111111	11110000	00000000
	网络号	11000010	00011000	00010000	00000000
		194.24.16.0是牛津的起始网络号			



IP地址

❖ **IP地址分类与表示**



❖ **子网划分**



❖ **CIDR-Classless InterDomain**



Routing

❖ **NAT-Network Address**



Translation



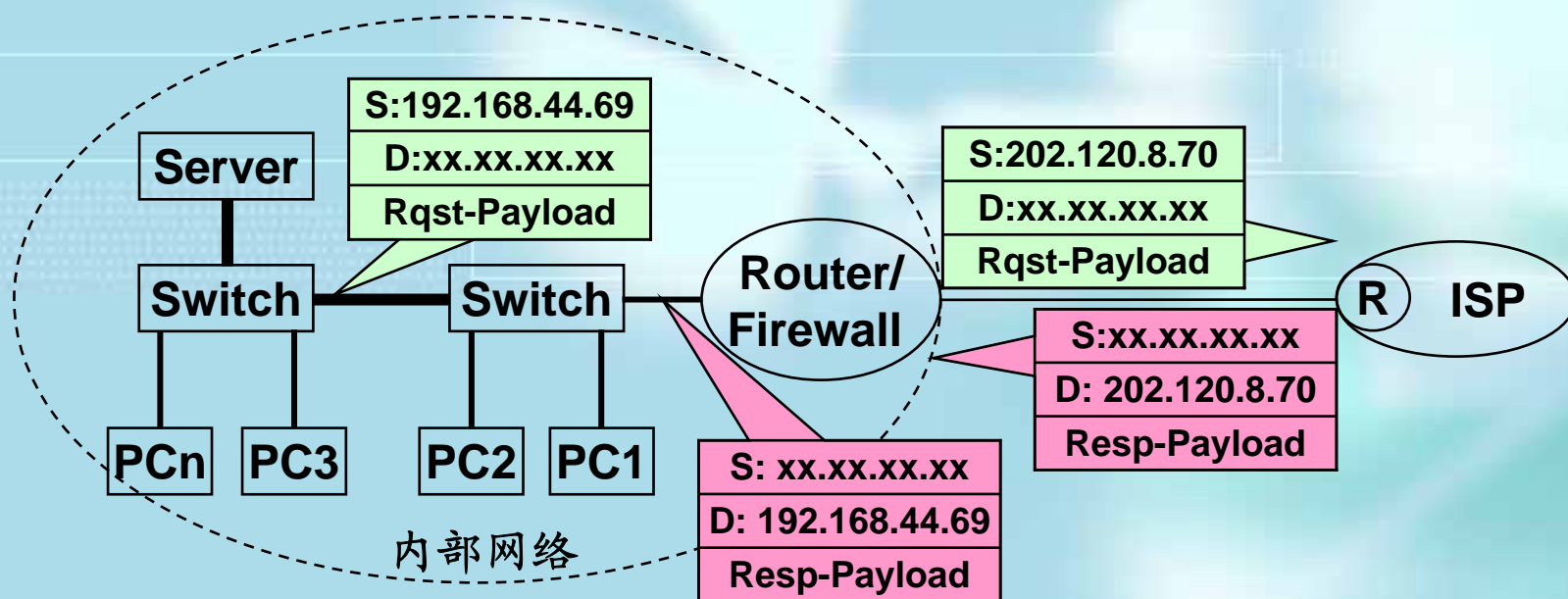
网络地址转换NAT

- ❖ 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网络中的身份，但一个单位只能分配到少量的公开**IP**地址，所以单位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**，**PC3**的内部IP地址为**192.168.44.69**
- ❖ **PC3**发出一个请求通过**ISP**访问**Internet**
- ❖ **Router(Firewall)**将对请求IP包和返回的应答IP包作**NAT**





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		



回来的包如何到达源端

- ❖ 当外出的**IP**包到达**NAT**时，将**IP**包中的源**IP**地址和源端口号信息填入映射表，同时将对应的映射表表项编号取代**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**字段，重新计算校验和并发送
- ❖ 当**IP**包从**ISP**发回来时，**NAT**将抽取该**IP**包中源端口号字段，查映射表将映射表中对应的内容放回源**IP**字段和源端口号字段，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的信息
- ❖ **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

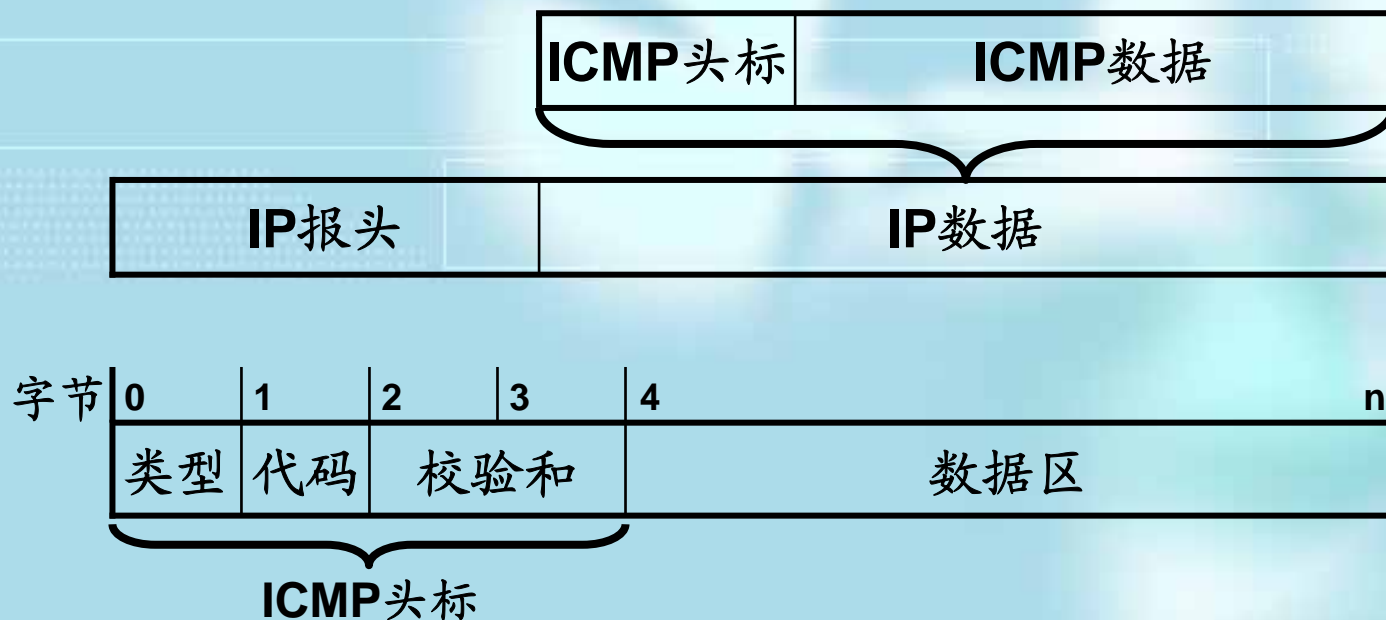
- **ICMP**报文格式
- **ICMP**报文主要类型
- **ICMP**应用举例





ICMP报文格式

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



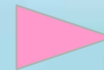


ICMP

(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
	17	地址掩码请求
	18	地址掩码应答



ICMP差错报文

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
	17	地址掩码请求
	18	地址掩码应答



目的站点不可达

❖ 目的站点不可达 **Destination Unreachable**

- 目的站点已关机
- 目的地址不存在
- 不知道目的站点的路径

❖ 目的站点不可达报文的格式

0	8	16	31
类型 (3)	代码	校验和	
(0~12) 用 (全0)			
出错数据报报头 + 前64 bit数据			
... ..			



目的站点不可达 (续)

❖ 目的站点不可达报文类型

代码值	意义
0	网络不可达
1	主机不可达
2	端口不可达
3	协议不可达
4	需分段, 但 DF=1 (不允许分段)
5	源寻径失败
6	目的站点网络未知
7	目的主机网络未知
8	原主机被隔离
9	与目的站点网络的通信被禁止
10	与目的站点主机的通信被禁止
11	对请求的服务类型, 网络不可达
12	对请求的服务类型, 主机不可达



ICMP差错报文

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
	17	地址掩码请求
	18	地址掩码应答



数据报超时 Time Exceeded

❖ 数据报超时报文的格式

0	8	16	31
类型（11）	代码（0/1）	校验和	
未用（全0）			
出错数据报报头 + 前64 bit数据			
... ..			

代码为**0**: **TTL**超时

代码为**1**: 分段重组超时



ICMP差错报文

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
	17	地址掩码请求
	18	地址掩码应答



数据报参数错

❖ 数据报参数错报文的格式

0	8	16	31
类型（12）	代码（0/1）	校验和	
未用（全0）			
出错数据报报头 + 前64 bit数据			
... ..			

代码为**0**: 指出数据报中现有的某参数出错, 其
指针域指向数据报中引起故障的字节

代码为**1**: 指出数据报中缺少某一必须的选项
(代码为**1**时无指针域)



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
	17	地址掩码请求
	18	地址掩码应答



ICMP控制报文

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
	17	地址掩码请求
	18	地址掩码应答



源抑制Source Quench

- ❖ 拥塞是无连接传输机制可能遇到的严重问题，由于路由器的缓冲区有限，大量数据报的涌入，缓冲区可能被“淹没”，形成拥塞
- 拥塞控制是全局性的，其数据来自不同的数据源
- 流量控制是局部性的，属于点对点的数据传输速率的匹配问题



原抑制方式分三个阶段

- ❖ 一旦发生拥塞，发**ICMP**源抑制报文
 - 输出队列满，或原设置的队列上限超界，将丢弃新来的数据报，凡丢弃一个，则向源站点发一个源抑制报文
 - 在较复杂的源抑制策略中，对不同的源站点的数据报的传输速率作统计，在队列超界时，有选择地向源站点发源抑制报文
- ❖ 源站点收到源抑制报文后将降低数据报传输速率
 - 通常降低传输速率后的一段时间内将不再理会源抑制报文
- ❖ 拥塞解除后，源站点主机恢复数据报的传输速率
 - 这与**ICMP**无关，主机在一段时间内没有再收到源抑制报文，则认为拥塞解除，将逐渐恢复数据报的传输速率



源抑制报文的格式

0	8	16	31
类型（4）	代码（0）	校验和	
未用（全0）			
出错数据报报头 + 前64 bit数据			
... ..			

其中的代码恒为0



ICMP控制报文

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
	17	地址掩码请求
	18	地址掩码应答



重定向 Redirect

- ❖ **ICMP**重定向机制将保证主机拥有一个动态的既小且优的路由表，但对路由器之间的路由表优化无能为力



重定向报文的格式

0	8	16	31
类型（5）	代码（0 ~ 3）	校验和	
网关地址			
出错数据报报头 + 前64 bit数据			
... ..			

其中的“ 网关**IP**地址”是告诉源站点“ 去目的站点最优路径中第一个网关的地址”

代码值	意 义
0	对网络的重定向报文 (已不用)
1	对主机的重定向报文
2	对服务类型和网络的重定向报文
3	对服务类型和主机的重定向报文



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
	17	地址掩码请求
	18	地址掩码应答



ICMP请求/应答报文

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 
	17	地址掩码请求 
	18	地址掩码应答 



回应请求/应答报文的格式

❖ 回应请求/应答用于测试目的站点的可达性

0	8	16	31
类型（8或0）	代码（0）	校验和	
标识符		序号	
任选数据			
... ..			

通常：某源站点主机向某目的站点主机发出一个回应请求，包含一段任选数据，如与收到的应答报文中的任选数据相同，则证明目的站点可达



ICMP请求/应答报文

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 
	17	地址掩码请求 
	18	地址掩码应答 



时间戳请求/应答

- ❖ 在互联网中，所有的计算机（包括路由器）都是独立运行的，为避免相互的时钟相差太大，**TCP/IP**提供了几个用于时钟同步的协议，其中最简单的是使用**ICMP**时间戳请求/应答报文

0	8	16	31
类型(13或14)	代码 (0)	校验和	
标识符		序号	
初始时间戳			
接收时间戳			
发送时间戳			

由于互联网（尤其是大型互联网）中数据报往返的延时的随机性很大，所以用**ICMP**时间戳的方法来同步时钟，其实用性有限



ICMP请求/应答报文

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 
	17	地址掩码请求 
	18	地址掩码应答 



地址掩码请求/应答

- ❖ 任何主机在启动时，都会广播其子网地址掩码请求报文，路由器收到后会发回一个地址掩码应答报文，其中包含本网使用的**32**位子网掩码

0	8	16	31
类型（1或8）	代码（0）	校验和	
标识符		序号	
地址掩码			



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



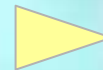
ping命令

❖ 路由跟踪命令



tracert (**Unix**下为**traceroute**) 命令

❖ 得到路径中最小的**MTU**



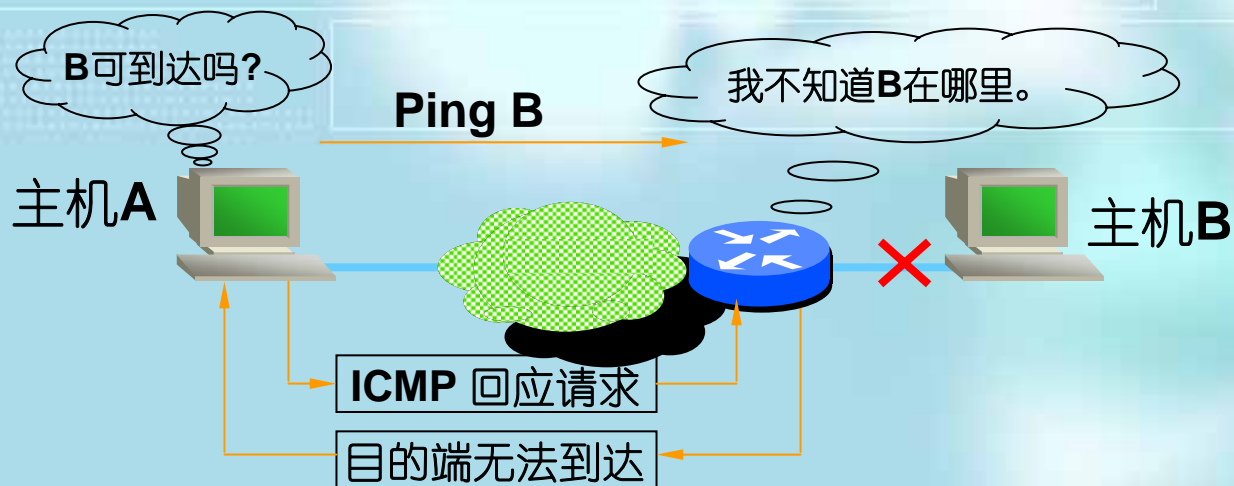


ping命令

- ❖ 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命令



❖ 路由跟踪命令

tracert (**Unix**下为**traceroute**) 命令



❖ 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命令

❖ 路由跟踪命令

tracert (**Unix**下为**traceroute**) 命令

❖ 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，如途径某个网络的**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**地址来定位机器

❖ 物理地址

硬件地址，如**MAC**地址

❖ 地址解析

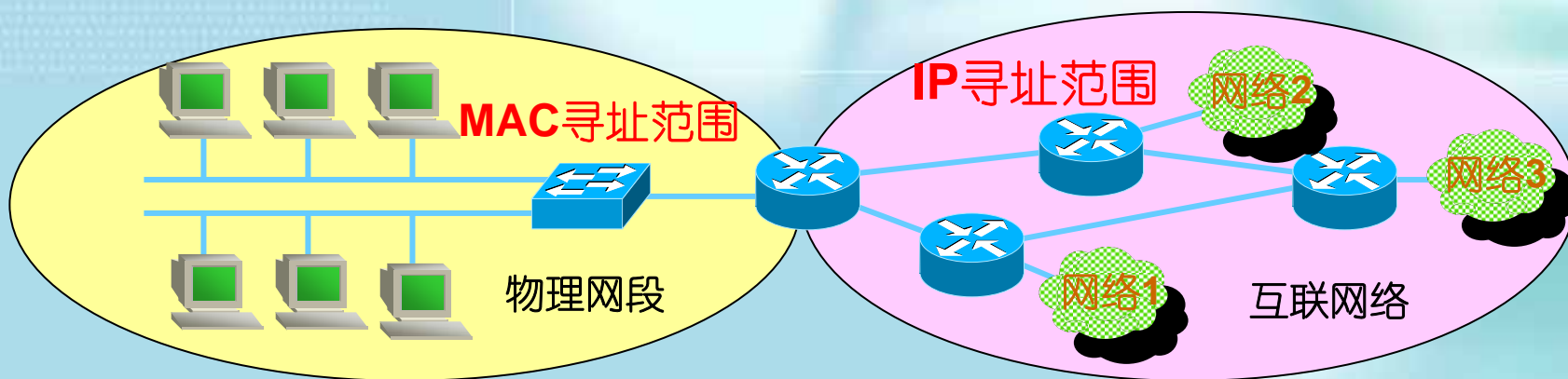
协议地址和物理地址之间的转换，如**IP**地址和**MAC**地址之间的转换

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，先做地址解析，然后按物理地址直接发送数据帧



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，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



❖ 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



❖ 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 



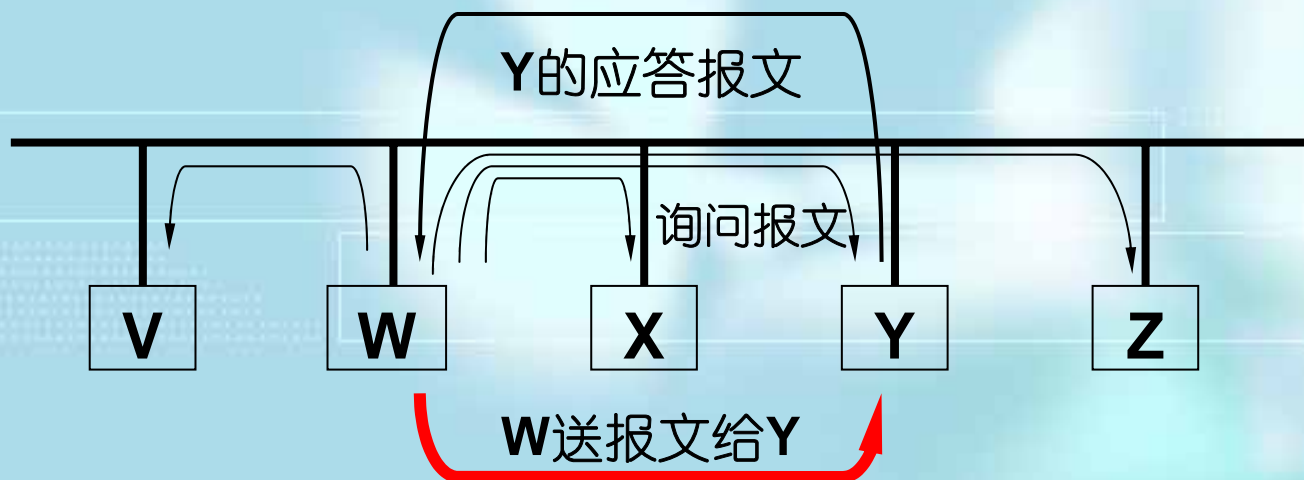
工作原理（以IP网络为例）

- ❖ 一个**ARP**请求消息是一个数据帧，其中包含发送站点的硬件地址和协议地址，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硬件地址和协议地址，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



以太网的帧结构中:

7	1	2/6	2/6	2	0 ~ 1500	46 ~ 0	4
先导字段 10101010		目的地址	源地址		数 据	填充字符	校验和

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 

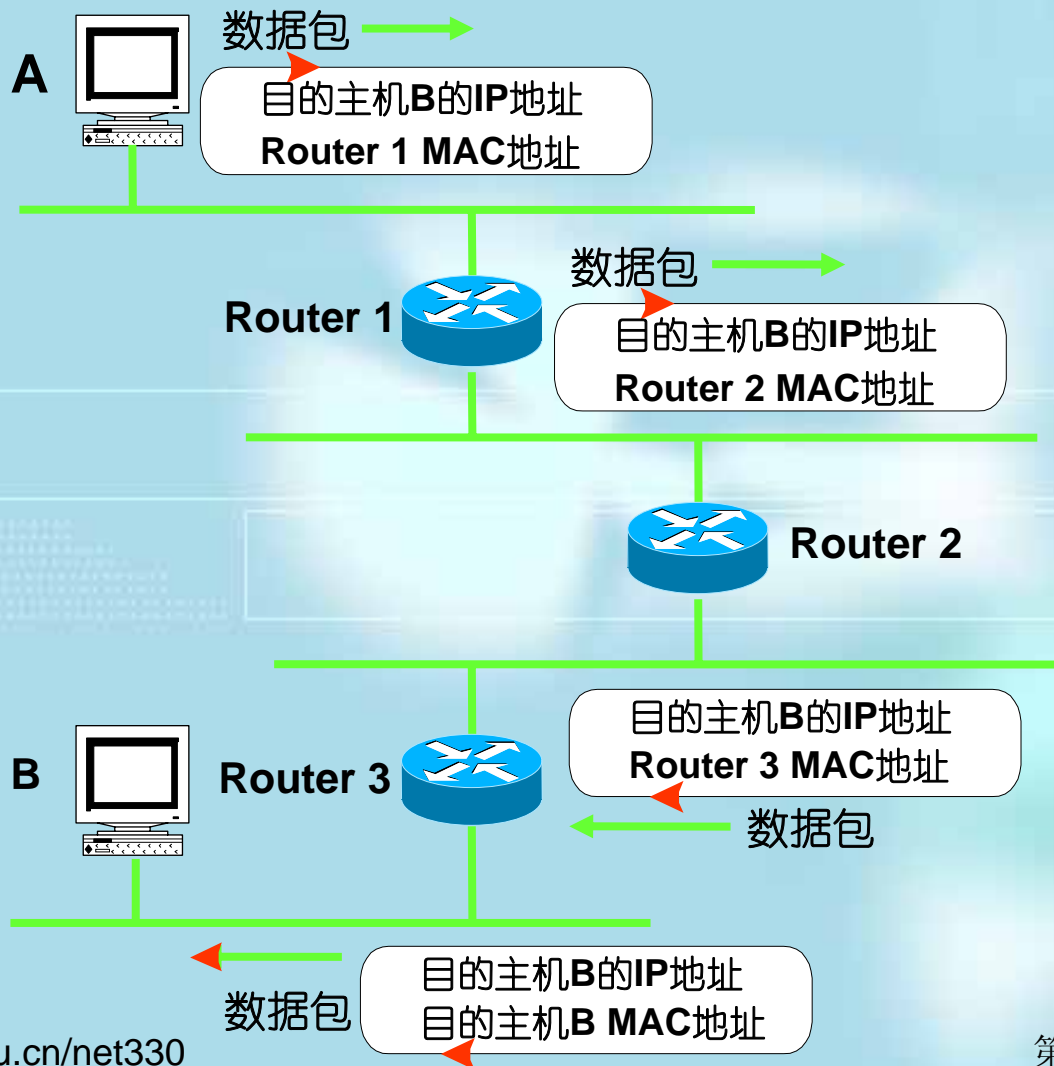


暂存ARP应答和 处理接收的ARP消息

- ❖ **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？
(如在全盘工作站中)

- ❖ 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，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**必须有一台**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
- ❖ **RIP协议 RFC 1058/1388**
(Routing Information protocol)
- ❖ **OSPF协议 RFC 1247**
(Open Shortest Path First)
- ❖ **BGP协议 RFC 1654**
(Border Gateway protocol)
- ❖ **IP多址传输**





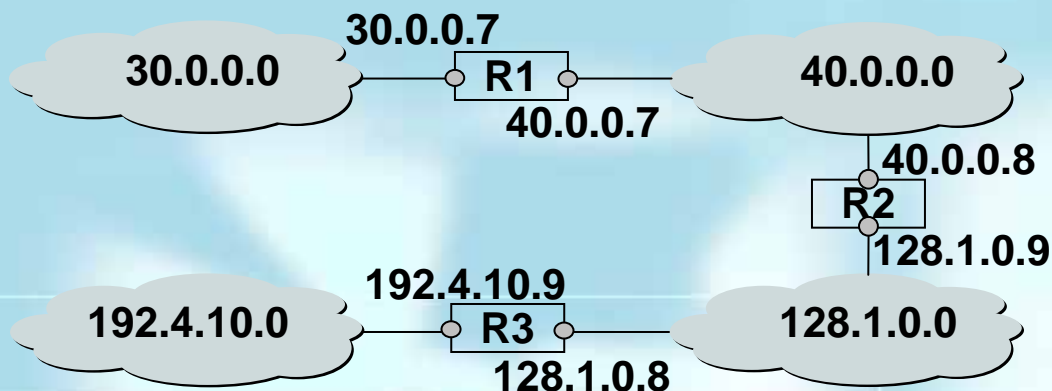
路由器和路由表

- ❖ 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标**IP**地址指明路径，而是为目标**IP**地址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标主机的网络地址
- ❖ 路由器对每一个接收到的分组，取出它的目标**IP**地址，然后根据目标**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，传送路径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

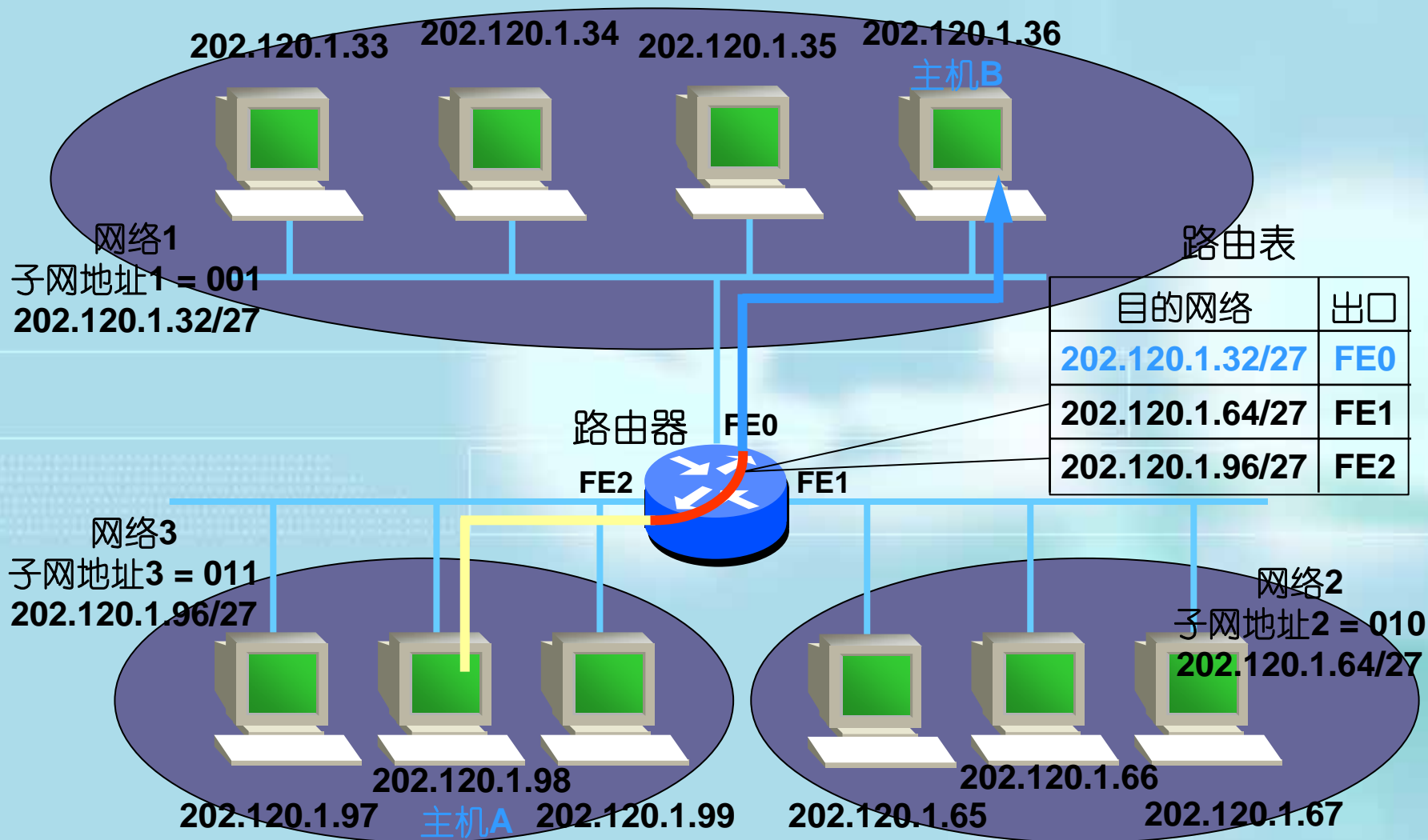
= 目的主机的**IP**地址 & 该地址的掩码

路径的决定：

if $((\text{Mask}[i] \& D) == \text{Destination}[i])$ forwarding to $\text{NextHop}[i]$



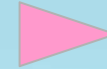
路由选择实例





IP路由

- ❖ 路由器和路由表
- ❖ 静态路由和动态路由
- ❖ **RIP协议 RFC 1058/1388**
(Routing Information protocol)
- ❖ **OSPF协议 RFC 1247**
(Open Shortest Path First)
- ❖ **BGP协议 RFC 1654**
(Border Gateway protocol)
- ❖ **IP多址传输**
(**multicast address**)





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生成路由表

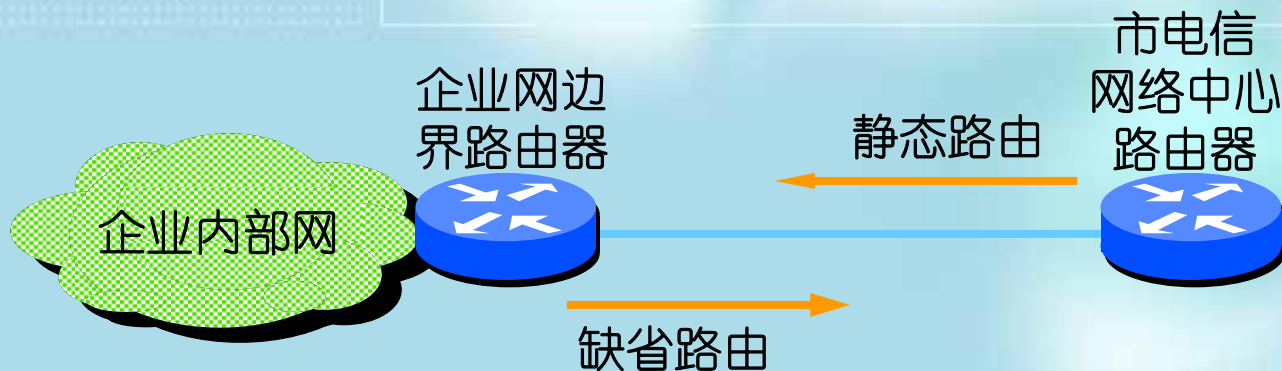
优点：网络拓扑发生变化时，动态路由协议自动更新路由表

缺点：路由器路由计算开销大



静态和缺省路由

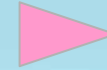
- ❖ 缺省路由是静态路由的一个特例，也需要人工配置；
- ❖ 互联网上有太多的网络和子网，受路由表大小的限制，路由器不可能也没有必要为互联网上所有网络和子网指明路径
- ❖ 凡是在路由表中无法查到的目标网络，在路由表中明确指定一个出口，这种路由方法称之为缺省路由





IP路由

- ❖ 路由器和路由表
- ❖ 静态路由和动态路由
- ❖ **RIP协议 RFC 1058/1388**
(Routing Information protocol)
- ❖ **OSPF协议 RFC 1247**
(Open Shortest Path First)
- ❖ **BGP协议 RFC 1654**
(Border Gateway protocol)
- ❖ **IP多址传输**





RIP协议的格式

❖ **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

❖ 报文格式：

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

最多重复25次

命令：1—请求包

2—响应包

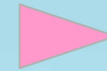
版本号：有1和2两种，V2支持VLSM（可变长子网掩码）

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
- ❖ **RIP协议 RFC 1058/1388**
(Routing Information protocol)
- ❖ **OSPF协议 RFC 1247**
(Open Shortest Path First)
- ❖ **BGP协议 RFC 1654**
(Border Gateway protocol)
- ❖ **IP多址传输**
(**multicast address**)





OSPF协议

❖ 自治系统 AS (Autonomous System)

- 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

OSPF是**Internet**上主要的内部网关协议，负责**AS**内部路由

1988年开始制定，**1990**年成为标准，采用**L-S**路由算法



OSPF支持三类网络

❖ 点对点的网络:

两个路由器直接连接

❖ 带广播的多访问网络:

如大多数局域网

❖ 不带广播的多访问网络:

大多数分组交换广域网，网上有多台路由器，这些路由器之间都能直接通信



OSPF的说明

❖ OSPF中的一些概念

OSPF采用分层管理的方法，将一个**AS**分成若干个区域，一个区域内的路由器之间交换所有的信息而对于同一**AS**内的其他区域则隐藏其详细拓扑结构

❖ 网络拓扑数据库

存储网络拓扑信息，网络用有向图来表示



OSPF的说明 (续)

❖ 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:

ABR (Area Border Router)

指连接两个或多个区域的路由器

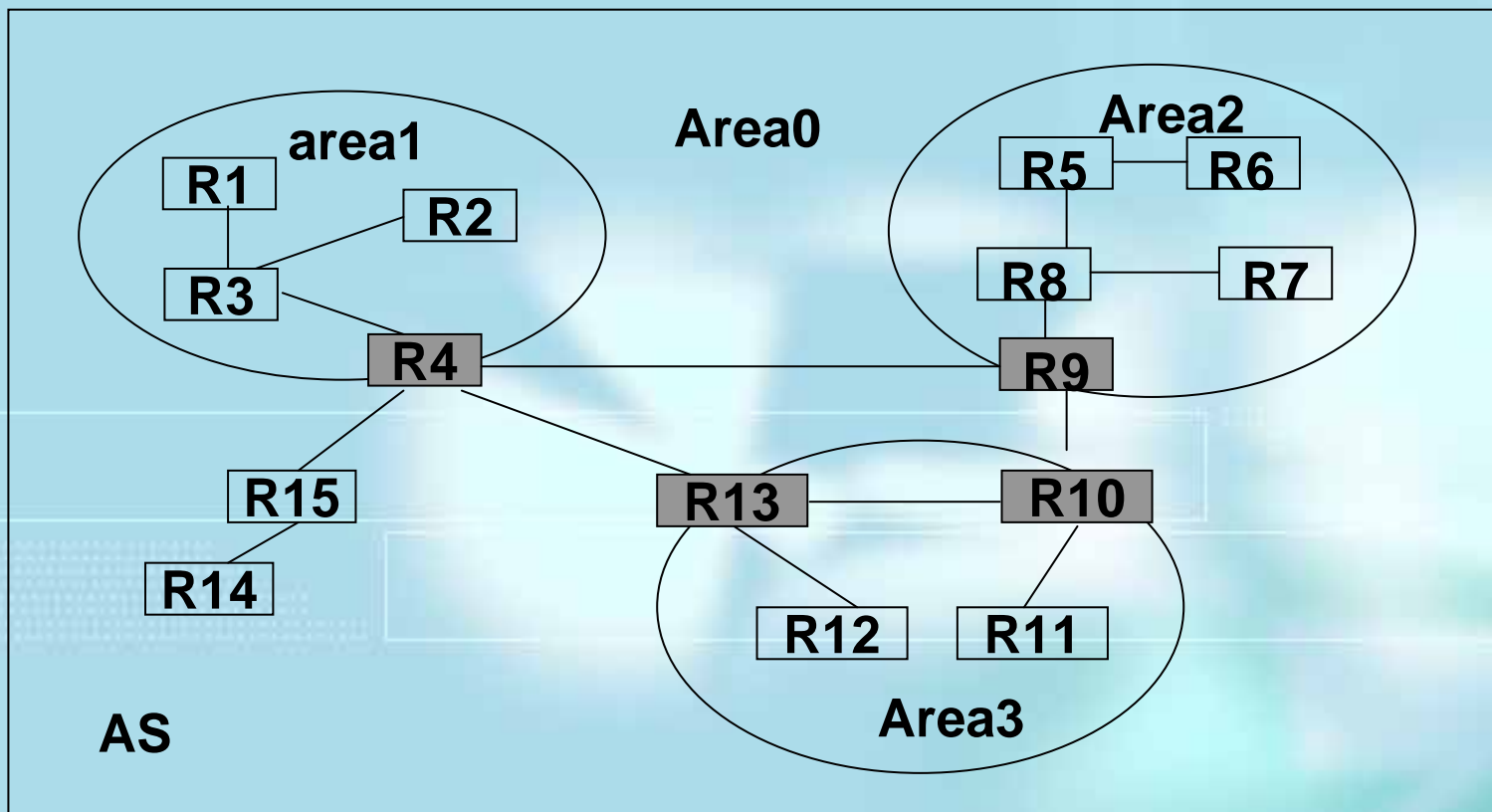
➤ AS边界路由器ASBR:

ASBR (Autonomous System Border Router)

连接多个AS的路由器



AS自治域举例



R4, R9, R10, R13 是ABR (区域边界路由器)

AREA0为主干网 (Backbone), 所有的**ABR**和不在任何区域中的网络和路由器都属于主干网



链路状态公告LSA和邻接关系

- ❖ **OSPF**是通过邻接路由器之间交换**LSA**（**Link State Advertisement**）来实现
- ❖ 邻接路由器与邻居并不相同，邻接路由器必须通过邻接关系来定义
- ❖ 为节约带宽，每个多访问的网络中要定义一台指定路由器**DR**（**Designated Router**）和备份指定路由器**BDR**（**Backup Designated Router**），**DR**与**LAN**中的其它路由器有邻接关系



OSPF报文

- ❖ **HELLO:** 用来发现相邻的路由器
- ❖ **Link State Update:** 链路状态更新，发送新的**LSA**
- ❖ **Link State ACK:** 链路状态确认，确认链路状态更新报文
- ❖ **Database Description:** 数据库描述，描述发送者当前拥有的新的**LSA**信息
- ❖ **Link State Request:** 链路状态请求，从邻接路由器要求新的**LSA**的请求信息



OSPF 的工作过程

- ❖ 与相邻的路由器建立邻接关系：
定期向各个接口发**HELLO**报文，在**HELLO**报文中，包含它的地址、用于选择**DR**的优先权、已知的**DR**和**BDR**、相邻路由表，根据优先权选择**DR**
- ❖ 数据库同步：
邻接路由器交换**LSA**信息
- ❖ 发**LSA**更新报文：
路由器在其链路发生变化或收到其他路由器发来的**LSA**更新报文后，向相邻路由器发**LSA**更新报文



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

= 本路由器到**ABR**的距离 + **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

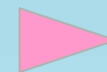
本路由器到本**AS**外目的网络的距离

= 本路由器到**ASBR**的距离 + **ASBR**到目的网络的距离



IP路由

- ❖ 路由器和路由表
- ❖ 静态路由和动态路由
- ❖ **RIP协议 RFC 1058/1388**
(Routing Information protocol)
- ❖ **OSPF协议 RFC 1247**
(Open Shortest Path First)
- ❖ **BGP协议 RFC 1654**
(Border Gateway protocol)
- ❖ **IP多址传输**
(**multicast address**)





BGP协议 (RFC 1654)

考虑**AS**之间的路由，将一个**AS**看成一个节点，使用的是距离矢量算法，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

❖ 为什么要区分**IGP**和**EGP**

AS是由不同的机构管理运行，**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，由**AS**管理者设置

❖ 路由策略的分类



❖ **BGP-4**的消息





路由策略的分类

- ❖ 控制本**AS**到其他**AS**的路径
如禁止本**AS**发出的数据经过某一个中间**AS**
- ❖ 控制本**AS**是否为某一相邻的**AS**传递过境数据
如出于经济原因
- ❖ **AS**内部的协调
决定出境数据的最佳路径



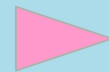
BGP协议 (RFC 1654)

考虑**AS**之间的路由，将一个**AS**看成一个节点，使用的是距离矢量算法，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

❖ 为什么要区分**IGP**和**EGP**

AS是由不同的机构管理运行，**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，由**AS**管理者设置

❖ 路由策略的分类



❖ **BGP-4**的消息





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

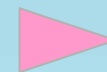
❖ **NOTIFICATION:**

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
- ❖ **RIP协议 RFC 1058/1388**
(Routing Information protocol)
- ❖ **OSPF协议 RFC 1247**
(Open Shortest Path First)
- ❖ **BGP协议 RFC 1654**
(Border Gateway protocol)
- ❖ **IP多址传输**
(**multicast address**)





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 







IPv6的主要改进

- ❖ 更大的地址空间：128位
- ❖ 灵活的首部格式：用一系列固定格式的扩展首部取代了IPv4中可变长度的选项字段
- ❖ 简化了协议：如取消了首部的校验和字段，分段只能在源端进行
- ❖ 允许对网络资源的预分配，支持实时图象等要求保证一定的带宽和时延的应用
- ❖ 允许协议继续演变，增加新的功能



IPv6

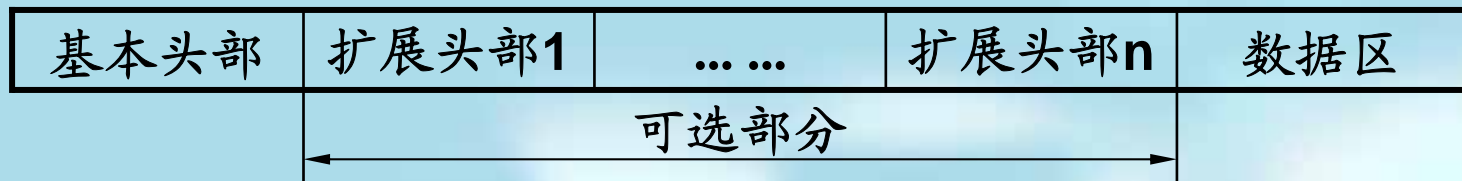
RFC1883 RFC1884

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



IPv6数据包格式

❖ 基本头部和扩展头部



❖ 基本头部格式

0	4	8	16	24	31
版本	优先级	流 标 记			
负 载 长 度			下一头部	驿站限制	
	源 地 址				
	目 的 地 址				



格式说明

- ❖ 版本: 0 ~ 3位
- ❖ 优先级: 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)

❖ 流标记: **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: **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^{23}**



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 



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 



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

如其中有多多个连续的零，则可用零压缩法

如：**1080:0:0:0:8:800:200C:417A** 可写成**1080::8:800:200C:417A**

❖ IPv4地址在IPv6中的表示

X:X:X:X:X:X:d.d.d.d 最后**32**位地址为**IPv4**的地址点分十进制表示

如：**0:0:0:0:0:0:202.120.5.100** (称为**IPv4兼容IPv6**地址)

0:0:0:0:0:FFFF:202.120.5.100 (称为**IPv4映射IPv6**地址)

也可以压缩成**0::202.120.5.100** **0::FFFF: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



第5章习题

Tnbnm P474

#6 (#6)、 #9 (#8)、 #11 (#10)

#20、 #22 (#16)、 #27 (#20)

#30、 #35、 #37 (#28)、 #52



v3中没有的习题

20. Consider the Chord circle of Fig.5-24. Suppose that node 10 suddenly goes on line. Does this affect node 1's finger table, and if so, how?
30. The CPU in a router can process 2 million packets/sec. The load offered to it is 1.5 million packets/sec. If a route from source to destination contains 10 routers, how much time is spent being queued and serviced by the CPUs?
35. A router is blasting out IP packets whose total length(data plus header) is 1024 bytes. Assuming that packets live for 10 sec, what is the maximum line speed the router can operate at without danger of cycling through the IP datagram ID number space?
52. The *Protocol* field used in the IPv4 head is not present in the fixed IPv6 head. Why not?