网络层 – 控制平面

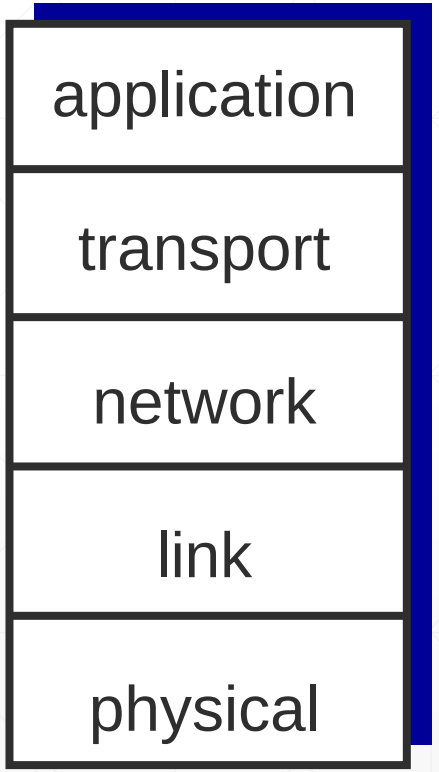
Network layer – control plane

本课内容

- 了解网络层控制平面的原理
  - 控制平面 – 路由!
  - 传统的各种路由协议
  - ICMP
  - MPLS
- 数据平面和控制平面
- 路由协议的分类
- 内部网关协议RIP
- 内部网关协议OSPF
- 外部网关协议BGP
- ICMP
- MPLS

网络层的位置

- 网络层
- Network layer
- 以后每一次课件都会有这张图!

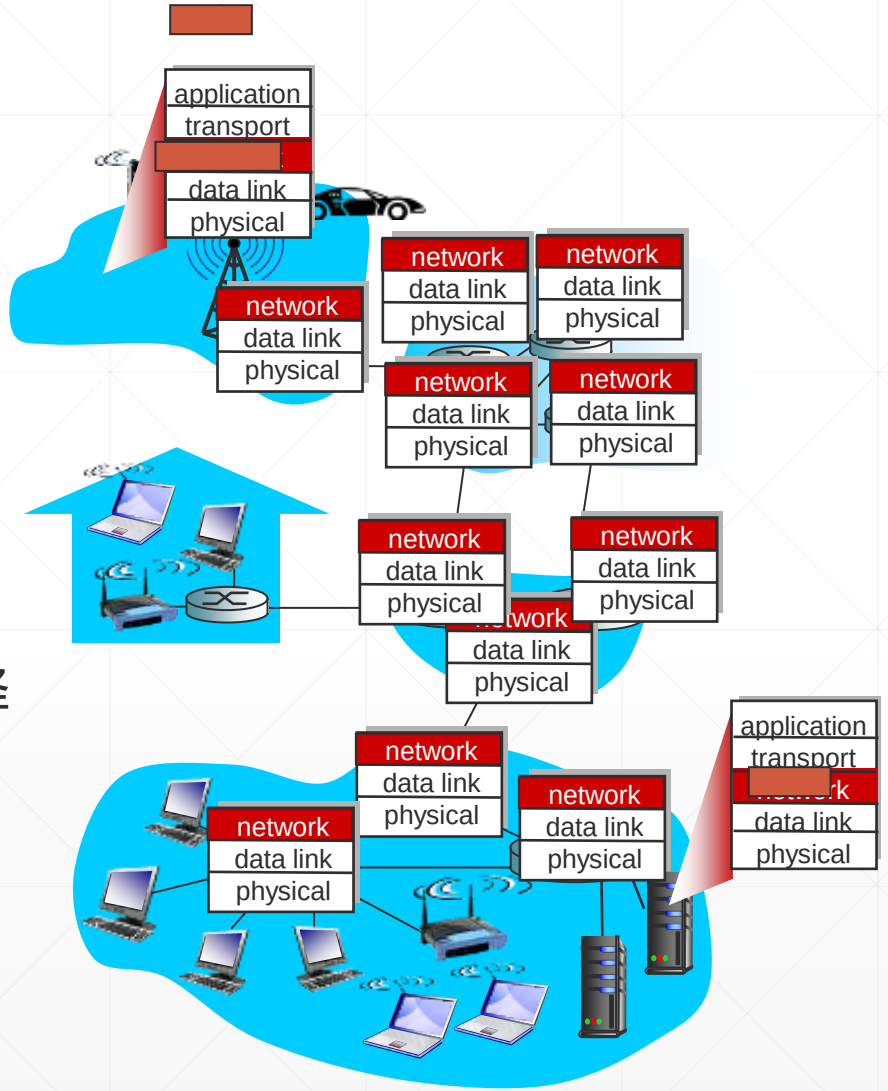


数据平面和控制平面

Data plane and control plane

网络层

- 从发送端到接收端传输数据段（segment）
- 发送端：将传输层的数据段包装成数据报
- 接收端：将收到的数据报解包，然后传送到传输层
- 每个端/主机、路由都需要部署网络层
- 路由器检查每个IP数据报的首部，并根据路由表选择不同路径发送IP数据报



无连接的数据报服务

- 在计算机网络领域，网络层应该向运输层提供怎样的服务（“面向连接”还是“无连接”）曾引起了长期的争论。
- 争论焦点的实质就是：在计算机通信中，可靠交付应当由谁来负责？是网络还是端系统?
- 现在的互联网：“无连接” “尽最大努力的”数据报服务
  - Connectionless
  - Best-effort service

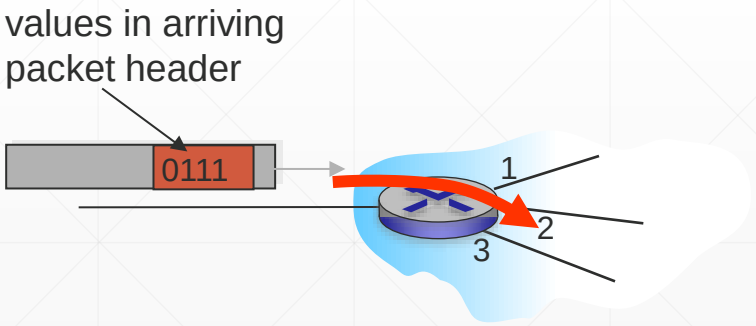
两个网络层的关键作用 – 路由和转发

- **网络层关键作用：**
- **转发**：将分组从路由器的输入端口转发到合适的输出端口
- **路由**：决定分组从发送端到接收端的的具体路径
  - 用到路由算法

- **类比，旅行：**
- **转发**：在机场，从入口进入，从哪个登机口出去？
- **路由**：计划旅行过程，包括从出发到回家的整个过程

数据平面和控制平面

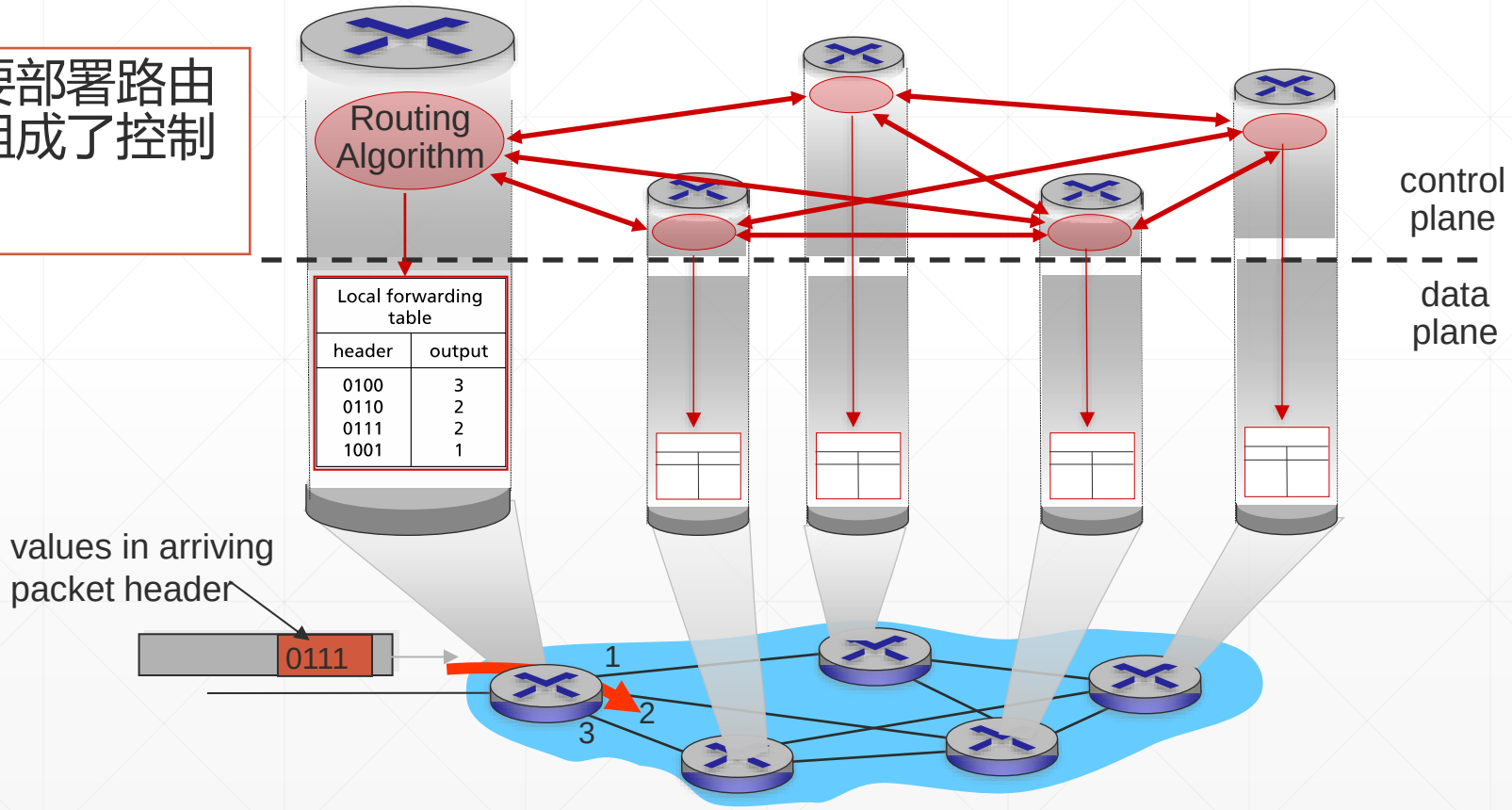
- **数据平面**：
- 本地、基于单个路由器的行为
- 主要行为：转发



- **控制平面**：
- Network-wide的行为
- 主要行为：路由
- 两种控制平面的路由做法：
  - 传统的路由算法：在每个路由器上运行路由算法，决定分组的路由
  - SDN（软件定义网络Software-defined Networking）：在 中心服务器上定义路由算法。

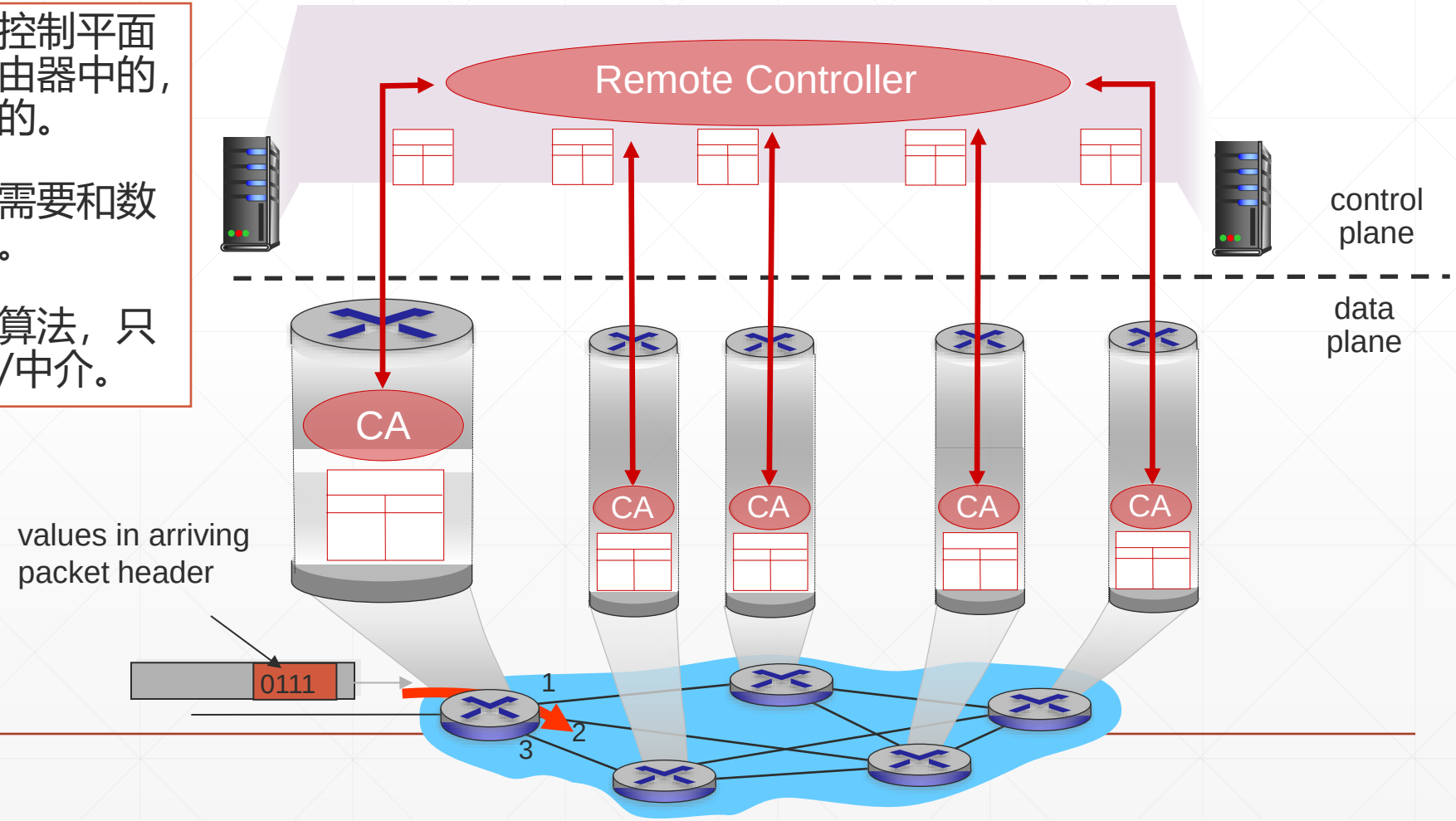
传统路由方法 per-router control plane

- 每一个路由器上都需要部署路由算法。这些路由算法组成了控制平面。



SDN的路由 logically centralized control plane

- 和传统路由不同，控制平面不是部署在每个路由器中的，而是逻辑上中心化的。
- 控制平面往往并不需要和数据平面部署在一起。
- 路由器中不再部署算法，只是控制平面的代理/中介。



路由协议 – 基本概念、术语

Routing protocols – concepts and terminologies

路由协议/算法

- 在计算机网络、网络层这个语境下，通常意思是一样的
- 路由协议的目的：决定从起点（source address, sending host）到终点（destination address, receiving host）的，经过若干个路由器的，最佳路径。
  - 路径：从给定的起点到终点，分组将会经过的路由器组成的序列
  - 最佳？Least cost, fastest, least congested。。。取决于具体的目的
- 路由协议/算法是计算机网络中最常见的挑战和研究问题之一。

有关路由选择协议的几个基本概念

1. 理想的路由算法

- 算法必须是正确的和完整的。
- 算法在计算上应简单。
- 算法应能适应通信量和网络拓扑的变化，这就是说，要有自适应性。
- 算法应具有稳定性。
- 算法应是公平的。
- 算法应是最佳的。

关于“最佳路由”

- 不存在一种绝对的最佳路由算法。
- 所谓“最佳”只能是相对于某一种特定要求下得出的较为合理的选择而已。
- 实际的路由选择算法，应尽可能接近于理想的算法。
- 路由选择是个非常复杂的问题
  1. 它是网络中的所有结点共同协调工作的结果。
  2. 路由选择的环境往往是不断变化的，而这种变化有时无法事先知道。

从路由算法的自适应性考虑

- **静态**路由选择策略——即**非自适应路由选择**，其特点是简单和开销较小，但不能及时适应网络状态的变化。
- **动态**路由选择策略——即**自适应路由选择**，其特点是能较好地适应网络状态的变化，但实现起来较为复杂，开销也比较大。

2. 分层次的路由选择协议

- 互联网采用分层次的路由选择协议。这是因为：

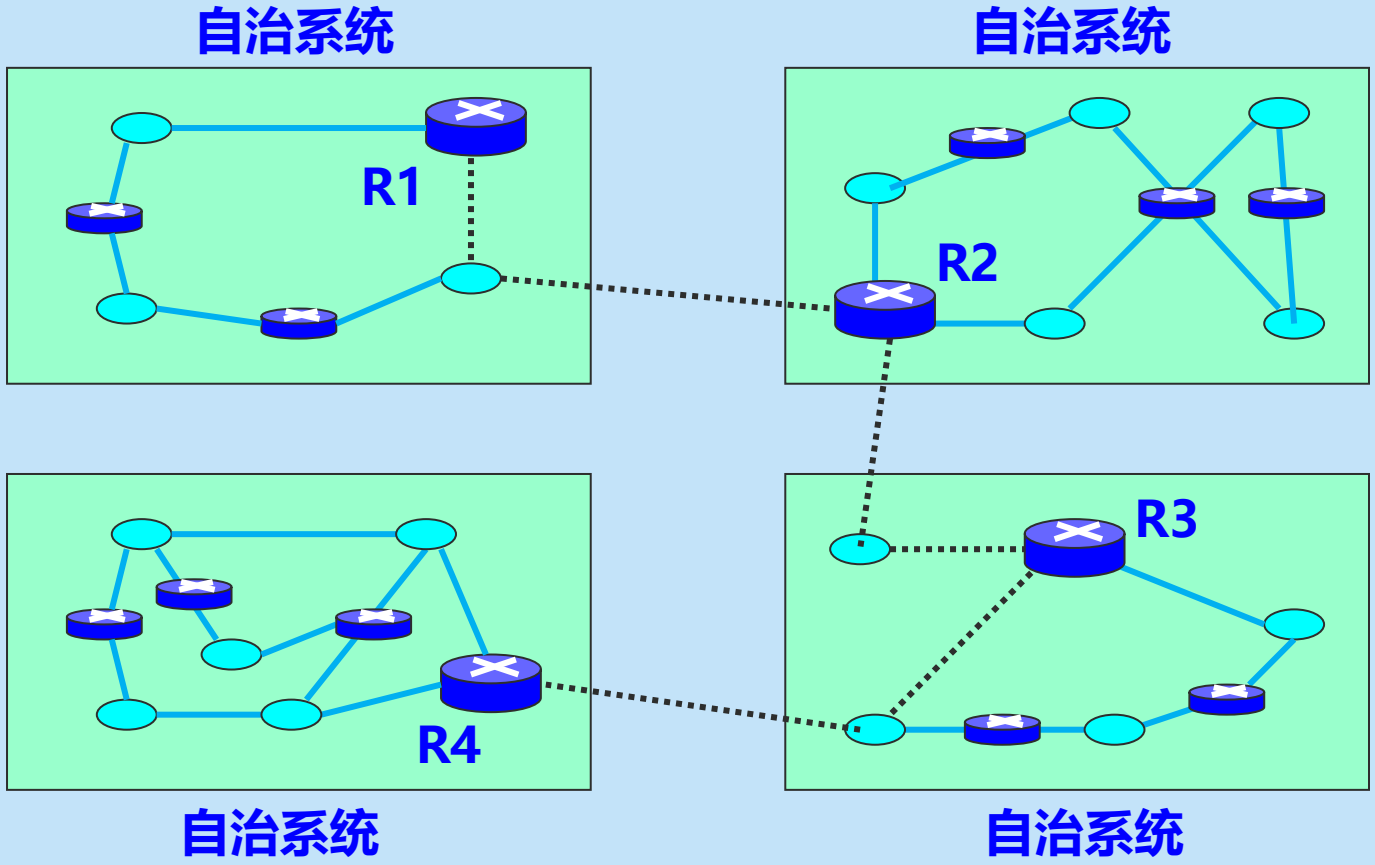
  (1) 互联网的规模非常大。如果让所有的路由器知道所有的网络应怎样到达，则这种路由表将非常大，处理起来也太花时间。而所有这些路由器之间交换路由信息所需的带宽就会使互联网的通信链路饱和。

  (2) 许多单位不愿意外界了解自己单位网络的布局细节和本部门所采用的路由选择协议（这属于本部门内部的事情），但同时还希望连接到互联网上。

自治系统 AS (Autonomous System)

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

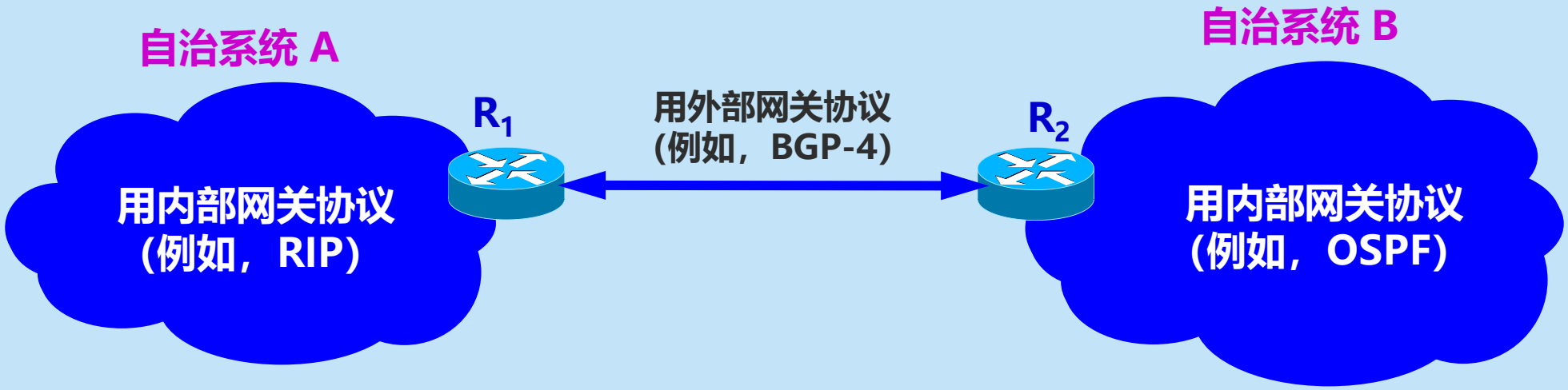
自治系统 AS



互联网有两大类路由选择协议

- **内部网关协议** IGP (Interior Gateway Protocol)
  1. 在一个自治系统**内部使用**的路由选择协议。
  2. 目前这类路由选择协议使用得最多，如 RIP 和 OSPF 协议。
- **外部网关协议** EGP (External Gateway Protocol)
  1. 若源站和目的站处在不同的自治系统中，当数据报传到一个自治系统的边界时，就需要使用一种协议**将路由选择信息传递到另一个自治系统中**。这样的协议就是外部网关协议 EGP。
  2. 在外部网关协议中目前使用最多的是 BGP-4。

自治系统和内部网关协议、外部网关协议



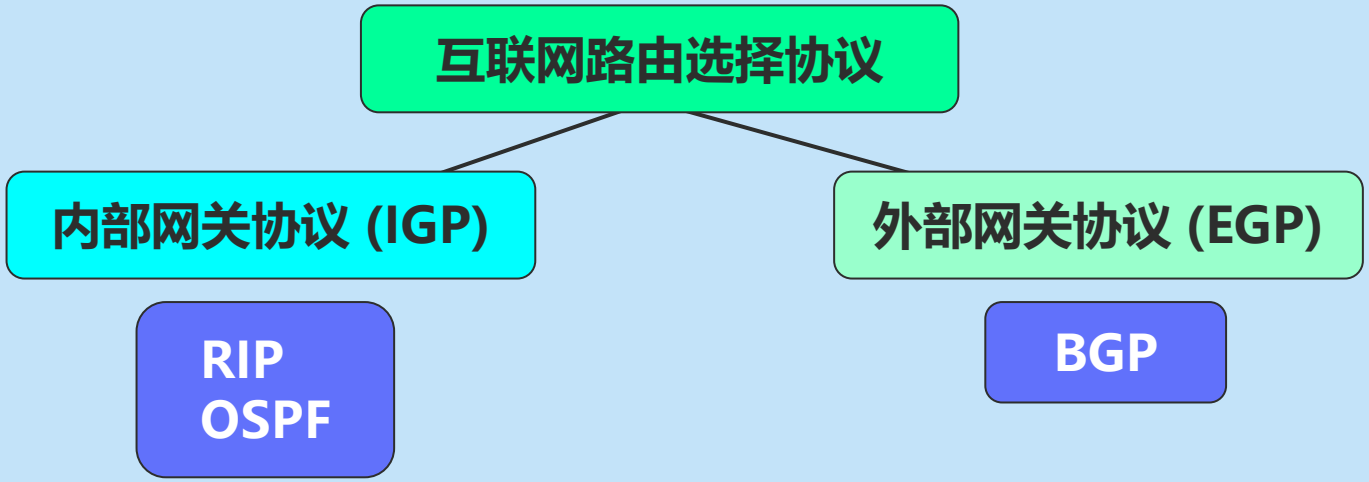
自治系统之间的路由选择也叫做域间路由选择 (inter-domain routing)，在自治系统内部的路由选择叫做域内路由选择 (intra-domain routing)。

这里要指出两点

- 互联网的早期 RFC 文档中未使用“**路由器**”而是使用“**网关**”这一名词。但是在新的 RFC 文档中又使用了“路由器”这一名词。应当把这两个术语当作**同义词**。
- **IGP 和 EGP 是协议类别的名称。**但 RFC 在使用 EGP 这个名词时出现了一点混乱，因为最早的一个外部网关协议的协议名字正好也是 EGP。因此在遇到名词 EGP 时，应弄清它是指旧的协议 EGP 还是指外部网关协议 EGP 这个类别。

互联网的路由选择协议

- **内部网关协议 IGP**：具体的协议有多种，如 RIP 和 OSPF 等。
- **外部网关协议 EGP**：目前使用的协议就是 BGP。

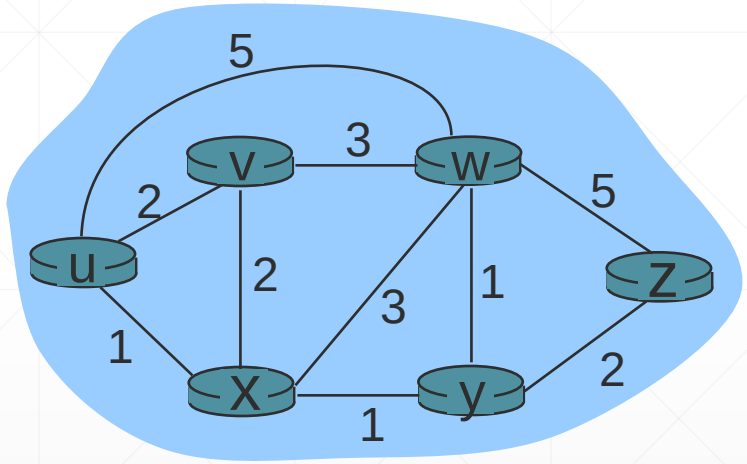


两类基本路由算法

Routing algorithm classification

网络的图抽象

- 讨论路由算法的基础
- 节点
- 链路
- 除了路由算法，其他的网络问题经常用到这种抽象。



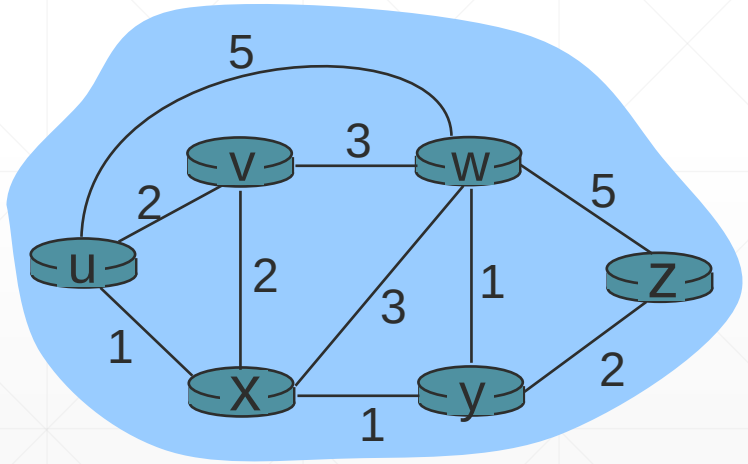
graph: G = (N,E)

N = set of routers = { u, v, w, x, y, z }

E = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

网络的图抽象 – 链路成本

- 链路的成本 cost/权值/开销
- 计算路由的基本衡量方式
- 从u到z的最小成本的路径?
- 这就是路由算法需要解决的问题。



c(x,x') = cost of link (x,x')
e.g., c(w,z) = 5

cost could always be 1, or inversely related to bandwidth, or inversely related to congestion

cost of path $(x_1, x_2, x_3, \ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

路由算法的分类

- 不谈具体的IGP/EGP、或者RIP/OSPF/BGP，根据路由算法是需要中心化的、全局的信息，还是去中心化的、局部的信息，可以分为两类：

链路状态算法

- 所有路由器都知道完整的拓扑结构，以及链路的成本信息
- Link state algorithms

距离向量算法

- 每个路由器只知道和自己相邻的路由器的存在，以及和相邻路由器间的链路的成本信息
- 需要进行循环迭代计算
- 需要相邻路由器间交换信息
- Distance vector algorithms

链路状态算法

- **Dijkstra算法**
- 网络拓扑已知，链路成本已知
  - 可以通过“链路状态广播”等方式
  - 所有节点有相同信息
- 计算从一个点（源节点）到其他所有节点的最小成本路径
  - 这就给出了该节点的“转发表”
- 循环迭代
  - 经过k论迭代，可以指导前往k个节点的最佳路径

- **用到的符号**
- $c(x,y)$: 从x到y的cost。如果x和y不相邻，则为无穷
- $D(v)$: 从源节点到v的当前cost
- $p(v)$: 从源节点到v的路径上，v的上一个节点
- N': 已经确定了最佳路径的节点构成的集合。

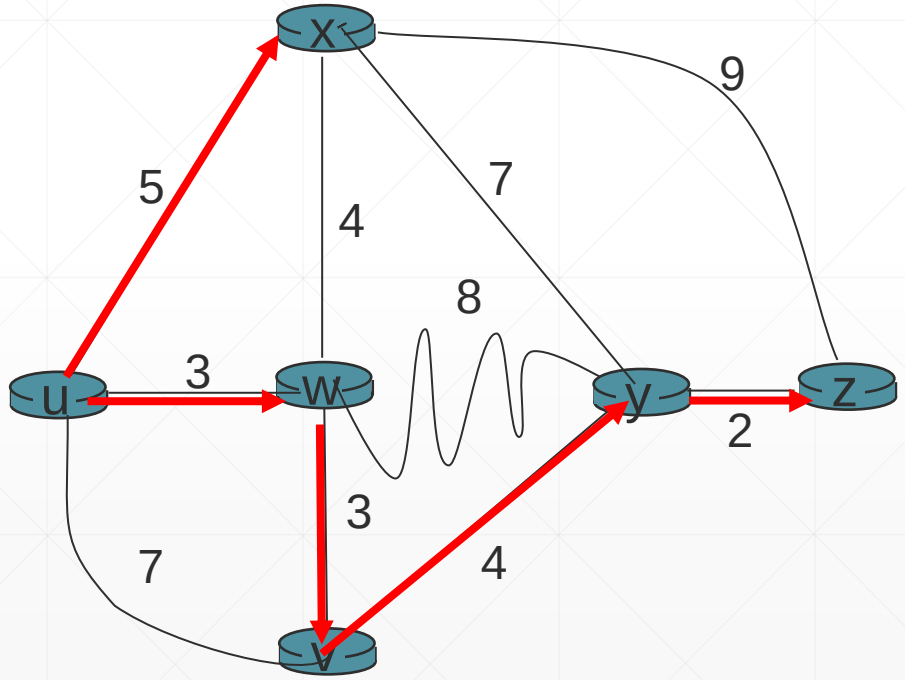
Dijsktra算法

- 源节点是u
- 计算源节点到所有其他节点的最佳路径。

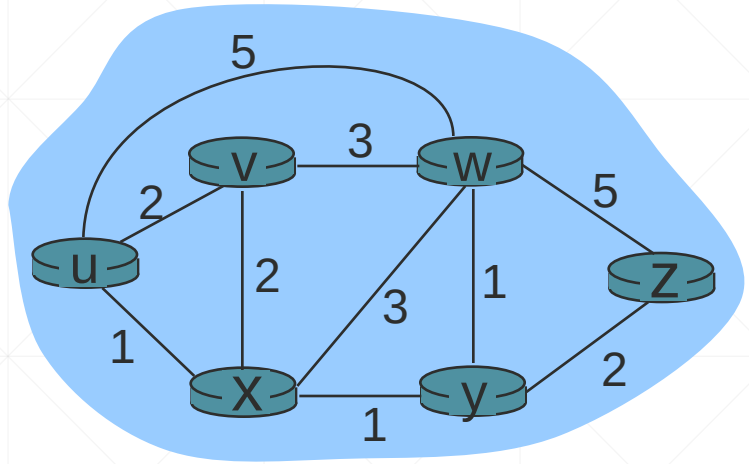
```
1  Initialization:
2    N' = {u}
3    for all nodes v
4      if v adjacent to u
5          then D(v) = c(u,v)
6      else D(v) = ∞
7
8   Loop
9     find w not in N' such that D(w) is a minimum
10    add w to N'
11    update D(v) for all v adjacent to w and not in N' :
12       D(v) = min( D(v), D(w)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

Dijkstra算法 – 例子1

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



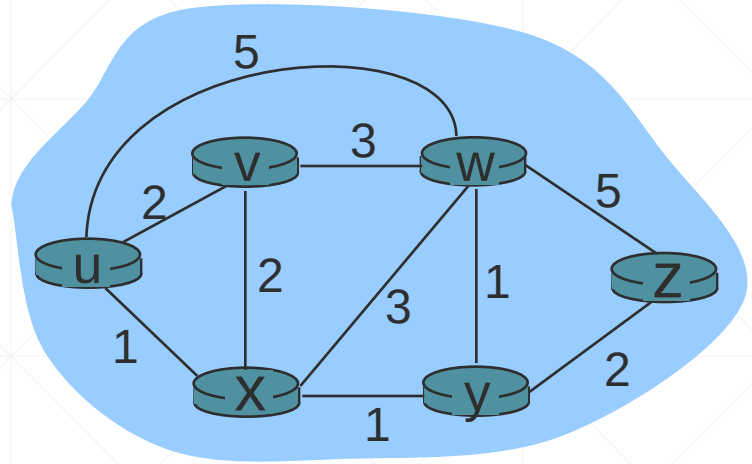
Dijkstra算法 – 例子2



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

Dijkstra算法 – 例子2

- 由此可以得到了u的转发表



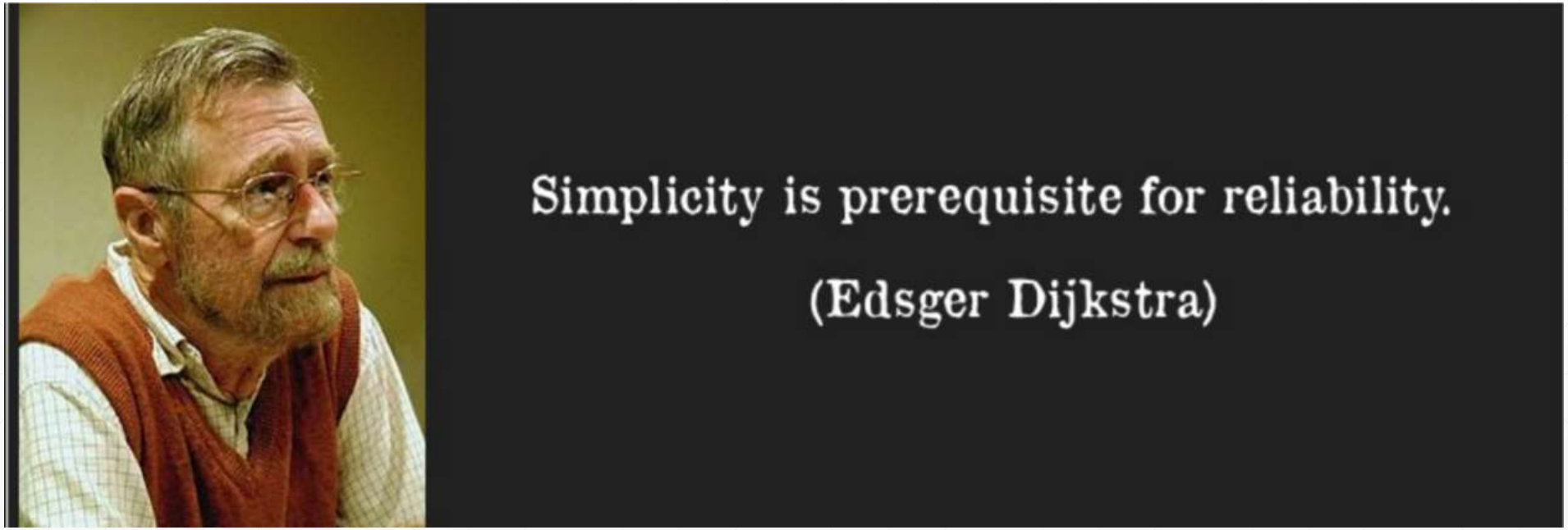
目的地址	出链路
V	(u,v)
X	(u,x)
Y	(u,x)
W	(u,x)
Z	(u,x)

Dijkstra算法 – 总结和讨论

- 实际上是计算“单源最短路径树”(Single Source Shortest Path Tree)
- 算法复杂度：n个节点的情形
  - 每次迭代，需要检查所有不在N’内的节点
  - 做$n(n+1)/2$次比较，复杂度O(n^2)
  - 更高效的实现方式，可以做到O(nlogn)
- 若边的权值（链路的成本）为负值，不能使用Dijkstra算法
  - 可以使用Bellmen-Ford算法
  - 从最短路径算法上来说，Dijkstra算法和Bellman-Ford算法的区别在于是否能在权值为负值的情况下得到最短路径

题外话 – Edsger Wybe Dijkstra

- 1930.05.11 – 2002.08.06
- 获得了1972年图灵奖

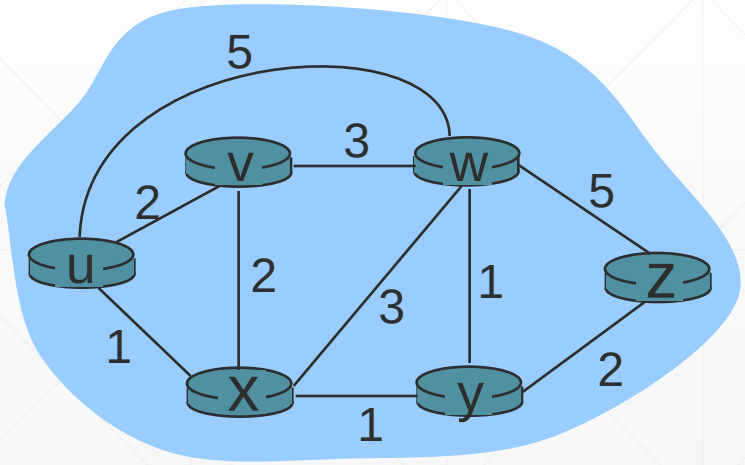


距离向量算法

- 每个路由器只知道和自己相邻的路由器的存在，以及和相邻路由器间的链路的成本信息
- Bellman-Ford等式（动态规划的思想）
  - 用$D_x(y)$表示“从x到y的最佳路径的成本”
  - 则有$D_x(y) = \min_v\{c(x,v) + D_v(y)\}$
  - “x到v的成本” + “从v到y的最佳路径的成本”

Bellman-Ford等式 – 例子

- 显然$D_v(z) = 5, D_x(z) = 3, D_w(z) = 3$
- 由B-F等式，有以下关系：
- $D_u(z) = \min\{c(u,v) + D_v(z), c(u,x) + D_x(z), c(u,w) + D_w(z)\}$
- $= \min\{2 + 5, 1 + 3, 5 + 3\} = 4$
- v、x、w，都是u的相邻节点
- 通过v、x、w的DV，可以得到u的DV。
- 取最小值的那个点，就是从u到z的“下一个节点”



距离向量算法

- $D_x(y)$实际上就是“从x到y的最优路径（最小成本）的成本的估计”
- 每个节点x都维护一组距离向量，用$\boldsymbol{D_x} = [D_x(y), y \in N]$表示

- 于是，在距离向量算法中，每个节点x还知道两组量：
  - 从x到每个相邻节点v的成本，用$c(x, v)$表示
  - 对于每个相邻节点v，维护节点v的距离向量，即$\boldsymbol{D_v} = [D_v(y), y \in N]$

距离向量算法

- **核心思路**
- 每个节点总是把自己的DV发送给所有邻居
- 当节点x收到来自一个邻居的新的DV时，基于B-F方程，更新自己的DV，即：
- $D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\}$，　（对于每一个N中的节点y）
- 在满足某些条件下，$D_x(y)$可以收敛到真正的“x到y的最佳路径的成本”
- 注意，DV只是估计，但是长时间后，估计可以收敛到真实值

距离向量算法 – 特点

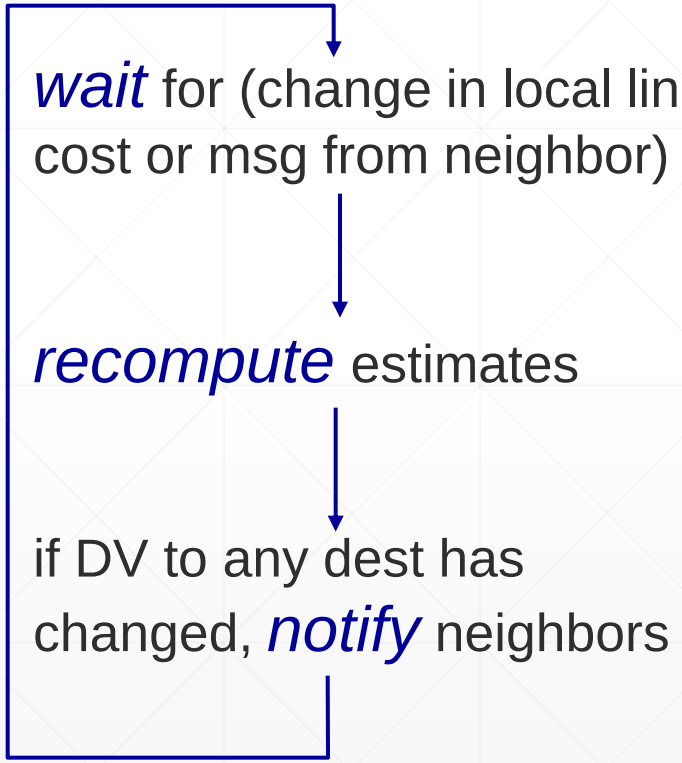
迭代、异步：

- 每一次本地的迭代，都是由以下之一的因素引起
- 本地链路成本发生改变
- 邻居更新了DV信息

分布式算法：

- 每个节点都在更新信息的时候仅仅通知自己的邻居
- 邻居可能再通知它们的邻居
- 迭代发生

每个节点



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

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

node x table

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

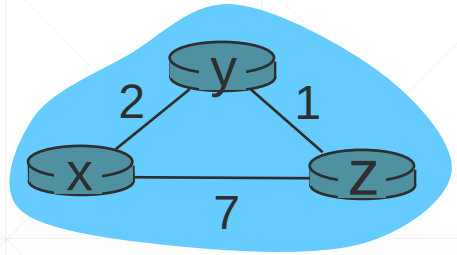
node z table

from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

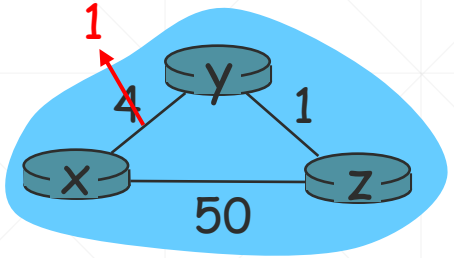
from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

time

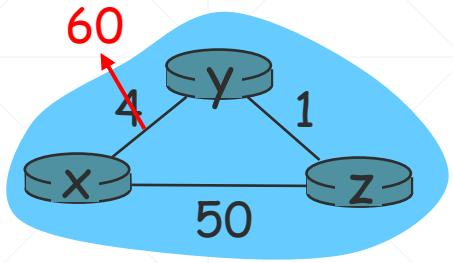


距离向量算法 – 链路成本发生变化



- 回忆：当链路成本发生变化时：
  - 节点发现链路成本发生变化→重新计算DV→如果DV发生变化，则通知邻居
- **“好消息传得快（good news travels fast）”**：
  - 什么是好消息 → 链路成本变小
  - t_0时刻：y发现链路成本的变化，更新自身的DV（$D_y(x)$从4变成了1），并将更新发送给邻居（即x和z）
  - t_1时刻：z收到来自y的更新，于是更新自己的DV表格，计算前往x的新的最小成本，并且发送自己的DV给z的邻居
  - t_2时刻：y又收到了刚刚z发出的更新，但是这一次y的DV没有更新，所以y不会再发送更新给z。至此，因为链路成本变小的变化全部完成。

距离向量算法 – 链路成本发生变化



- “**坏消息传得慢（bad news travels slow）**”：
  - 什么是坏消息 → 链路成本变大、链路断了等等
  - 链路成本变化前：$D_y(x) = 4, D_y(z) = 1, D_z(y) = 1, D_z(x) = 5$。
  - t_0时刻：y检测到链路成本发生变化，y计算它到x的新的最低成本路径的成本，得到 $D_y(x) = \min\{c(y,x) + D_x(x), c(y,z) + D_z(x)\} = \min\{60 + 0, 1 + 5\} = 6$
  - 全局来看，这个结果其实是错的。但是这是y能得到的所有信息，y希望通过z能到达x。
  - t_1时刻，z收到y的最新DV，它将计算得到$D_z(x) = \min\{50 + 0, 1 + 6\} = 7$，即z通过y到达x。Z的DV发生变化，它又将通知y。发生了**路由环路**！
  - 以此类推，循环往复，共44次迭代，直到算出来z经过y到达x的成本大于50为止。
  - 无穷计数问题（Count-to-infinity）

距离向量算法 – 毒性逆转（Poisoned Reverse）

- 避免路由环路的方法之一
- 基本思想：如果z→y→x，则z向y通告$D_z(x) = \infty$（即使z知道$D_z(x)$是某个数值）。这样可以防止y重新经过z到达x。
- 这只是解决方法之一。并且，实际上，有一些路由环路的产生是无法通过毒性逆转来避免的。

链路状态算法 v.s. 距离向量算法

消息的复杂度

- ***链路状态算法：*** n个节点，E条链路，则需要发送O(nE)条消息
- ***距离向量算法：*** 消息只在相邻节点中传播，数量不定

收敛速度

- ***链路状态算法：*** O(n^2)算法，复杂度确定
- ***距离向量算法：*** 收敛时间不定，可能有路由环路、无穷计数问题等

健壮性/鲁棒性：如果路由器出错

- ***链路状态算法：***
  - 节点可能会广播错误的**链路**成本
  - 每个节点只计算自己的转发表
- ***距离向量算法：***
  - 节点可能会广播错误的**路径**成本
  - 每个节点的转发表（及DV）都会被其他节点所使用，因此错误可能会在网络中逐渐传播。

内部网关协议 Part 1
RIP协议

Routing Information Protocol

具体的网络层路由协议

- 我们已经从通用层面上，讨论了两类路由算法（链路状态算法和距离向量算法）
- 下面，我们进入对路由协议的具体讨论，包括
  - 路由信息协议 RIP (Routing Information Protocol) – 内部网关协议
  - 开放最短路径优先 OSPF (Open Shortest Path First) – 内部网关协议
  - 边界网关协议 BGP (Boarder Gateway Protocol) – 外部网关协议

内部网关协议 RIP

1. 工作原理

- 路由信息协议 RIP (Routing Information Protocol) 是内部网关协议 IGP 中最先得到广泛使用的协议。
- RIP 是一种**分布式的、基于距离向量的路由选择协议。**
- **RIP 协议要求**网络中的每一个路由器都要维护从它自己到其他每一个目的网络的距离记录。

“距离”的定义

- 从一个路由器到**直接连接**的网络的距离定义为 1。
- 从一个路由器到非直接连接的网络的距离定义为所经过的路由器数加 1。
- RIP 协议中的“距离”也称为“**跳数**”(hop count)，因为每经过一个路由器，跳数就加 1。
- 这里的“距离”实际上指的是“**最短距离**”。

“距离”的定义

- RIP 认为一个好的路由就是它通过的路由器的数目少，即“距离短”。
- RIP 允许一条路径最多只能包含 15 个路由器。
- “距离”的最大值为 16 时即相当于不可达。可见 RIP 只适用于小型互联网。
- RIP 不能在两个网络之间同时使用多条路由。RIP 选择一个具有最少路由器的路由（即最短路由），哪怕还存在另一条高速(低时延)但路由器较多的路由。

RIP 协议的三个特点

1. 仅和相邻路由器交换信息。
2. 交换的信息是当前本路由器所知道的全部信息，即自己的路由表。
3. 按固定的时间间隔交换路由信息，例如，每隔 30 秒。当网络拓扑发生变化时，路由器也及时向相邻路由器通告拓扑变化后的路由信息。

路由表的建立

- 路由器在**刚刚开始工作**时，只知道到直接连接的网络的距离（此距离定义为 1）。它的**路由表是空的**。
- 以后，每一个路由器也只和数目非常有限的相邻路由器交换并更新路由信息。
- 经过若干次更新后，所有的路由器最终都会知道到达本自治系统中任何一个网络的最短距离和下一跳路由器的地址。
- RIP 协议的**收敛** (convergence) 过程较快。“收敛”就是在自治系统中所有的结点都得到正确的路由选择信息的过程。

2. 距离向量算法

路由器收到相邻路由器（其地址为 X）的一个 RIP 报文：

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

(2) 对修改后的 RIP 报文中的每一个项目，重复以下步骤：

若项目中的目的网络不在路由表中，则把该项目加到路由表中。

否则

若下一跳字段给出的路由器地址是同样的，则把收到的项目替换原路由表中的项目。

否则

若收到项目中的距离小于路由表中的距离，则进行更新，

否则，什么也不做。

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

(4) 返回。

路由器之间交换信息与路由表更新

- RIP 协议让互联网中的所有路由器都和自己的相邻路由器不断交换路由信息，并不断更新其路由表，使得从每一个路由器到每一个目的网络的路由都是最短的（即跳数最少）。
- 虽然所有的路由器最终都拥有了整个自治系统的全局路由信息，但由于每一个路由器的位置不同，它们的路由表当然也应当是不同的。

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

表 4-9(a) 路由器 R_6 的路由表

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

表 4-9(b) R4 发来的路由更新信息

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

计算更新

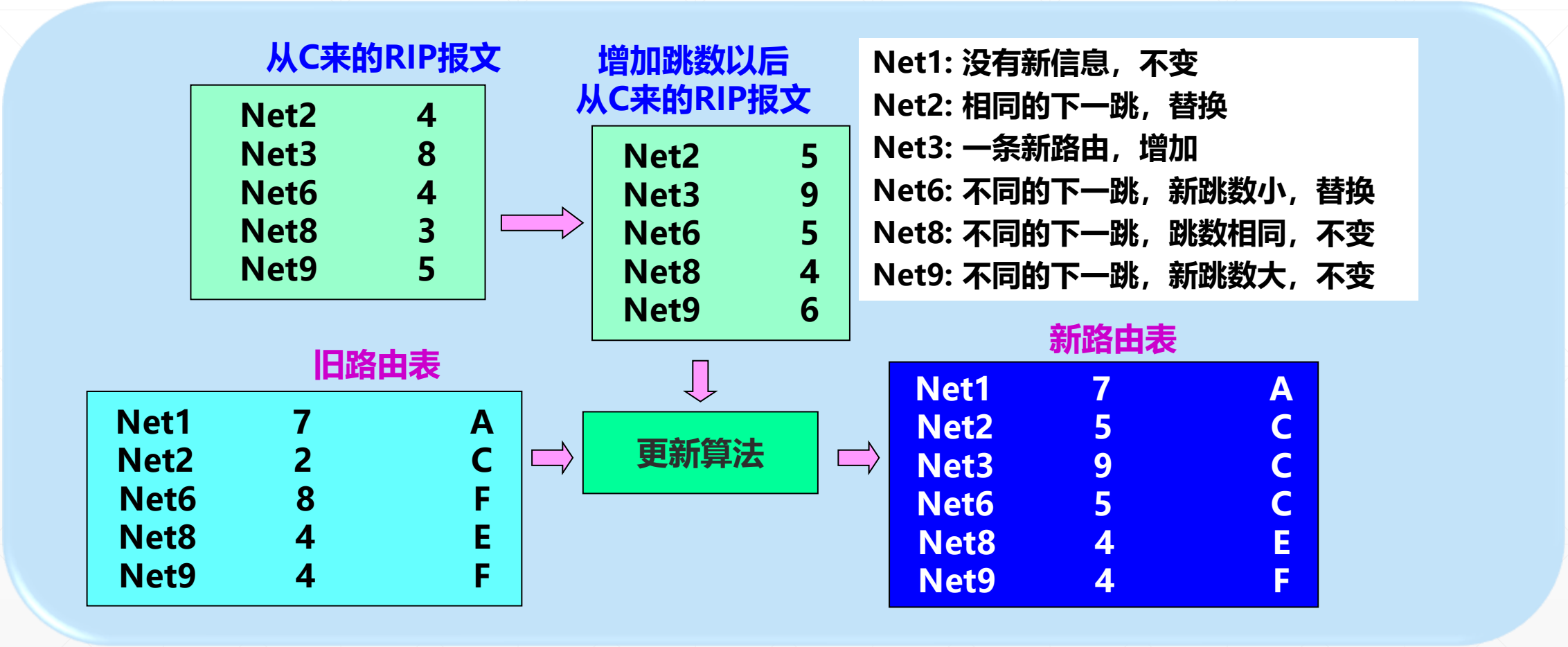
表 4-9(c) 修改后的表 4-9(b)

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

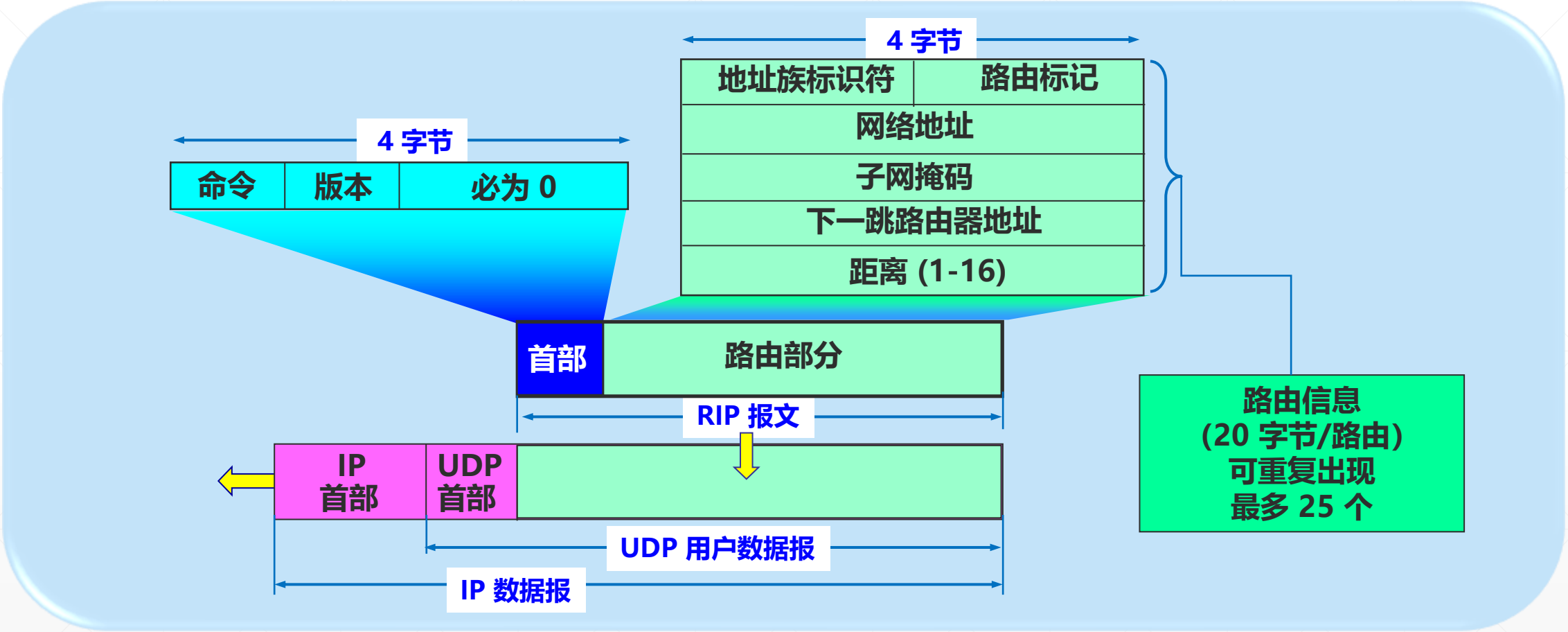
表 4-9(d) 路由器 R_6 更新后的路由表

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

【例】路由表更新


从C来的RIP报文
Net2 4
Net3 8
Net6 4
Net8 3
Net9 5
增加跳数以后
从C来的RIP报文
Net2 5
Net3 9
Net6 5
Net8 4
Net9 6
Net1: 没有新信息，不变
Net2: 相同的下一跳，替换
Net3: 一条新路由，增加
Net6: 不同的下一跳，新跳数小，替换
Net8: 不同的下一跳，跳数相同，不变
Net9: 不同的下一跳，新跳数大，不变
旧路由表
Net1 7 A
Net2 2 C
Net6 8 F
Net8 4 E
Net9 4 F
更新算法
新路由表
Net1 7 A
Net2 5 C
Net3 9 C
Net6 5 C
Net8 4 E
Net9 4 F

3. RIP2 协议的报文格式



RIP2 报文

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

RIP2 报文

- 一个 RIP 报文最多可包括 25 个路由，因而 RIP 报文的最大长度是 4+20 x25=504 字节。如超过，必须再用一个 RIP 报文来传送。
- **RIP2 具有简单的鉴别功能。**
  1. 若使用鉴别功能，则将原来写入第一个路由信息（20 个字节）的位置用作鉴别。
  2. 在鉴别数据之后才写入路由信息，但这时最多只能再放入 24 个路由信息。

好消息传播得快，坏消息传播得慢

- **RIP 协议特点**：好消息传播得快，坏消息传播得慢。
- **RIP 存在的一个问题**：当网络出现故障时，要经过比较长的时间 (例如数分钟) 才能将此信息传送到所有的路由器。

RIP 协议的优缺点

- 优点：
  1. 实现简单，开销较小。
- 缺点：
  1. RIP 限制了网络的规模，它能使用的最大距离为 15（16 表示不可达）。
  2. 路由器之间交换的路由信息是路由器中的完整路由表，因而随着网络规模的扩大，开销也就增加。
  3. “坏消息传播得慢”，使更新过程的收敛时间过长。

内部网关协议 Part 2
OSPF协议

Open Shortest Path First

内部网关协议 OSPF

- 开放最短路径优先 OSPF (Open Shortest Path First)是为克服 RIP 的缺点在 1989 年开发出来的。
- **OSPF 的原理很简单，但实现起来却较复杂。**

1．OSPF 协议的基本特点

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

三个要点

- 向本自治系统中所有路由器发送信息，这里使用的方法是洪泛法。
- 发送的信息就是与本路由器相邻的所有路由器的链路状态，但这只是路由器所知道的部分信息。
  - “链路状态”就是说明本路由器都和哪些路由器相邻，以及该链路的“度量”(metric)。
- 只有当链路状态发生变化时，路由器才用洪泛法向所有路由器发送此信息。

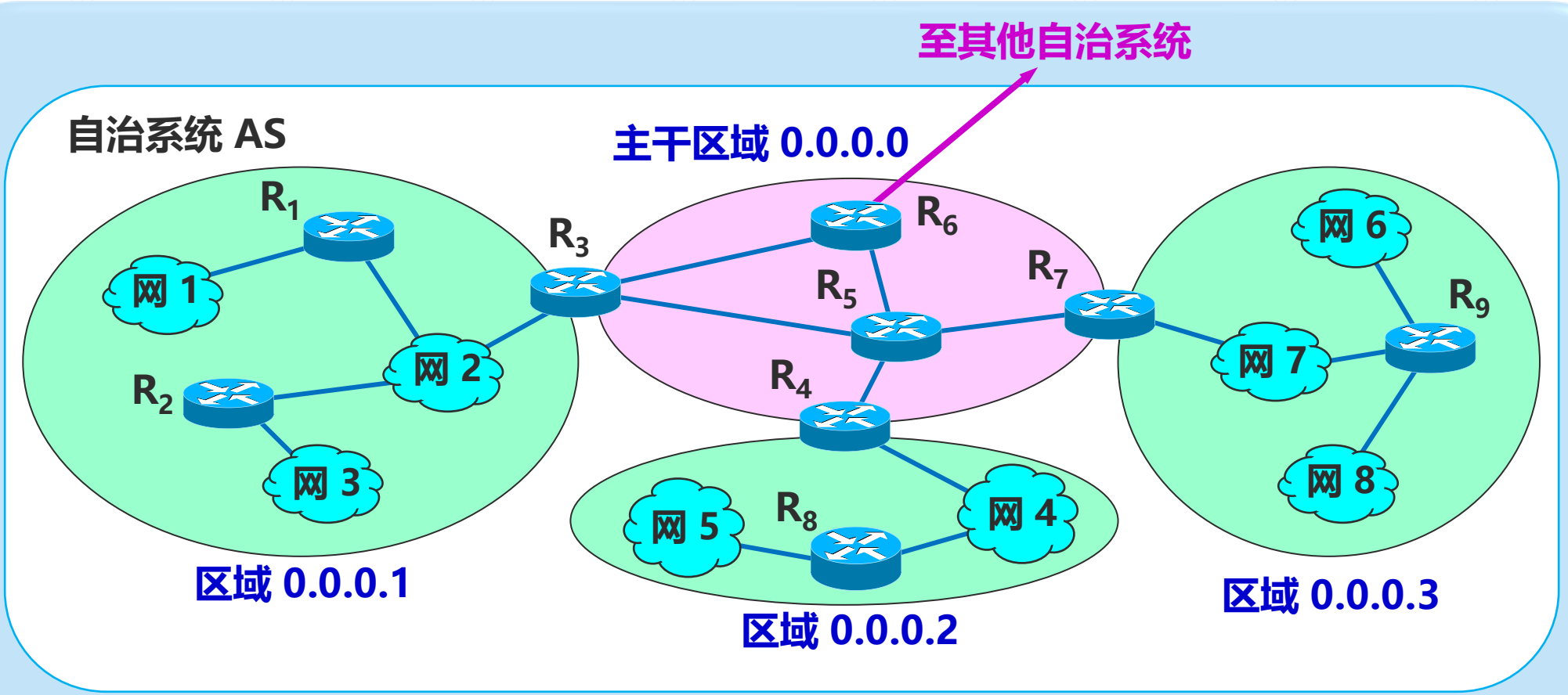
链路状态数据库 (link-state database)

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

OSPF 的区域 (area)

- 为了使 OSPF 能够用于规模很大的网络，OSPF 将一个自治系统再划分为若干个更小的范围，叫做**区域**。
- 每一个区域都有一个 32 位的区域标识符（用点分十进制表示）。
- 区域也不能太大，在一个区域内的路由器最好不超过 200 个。

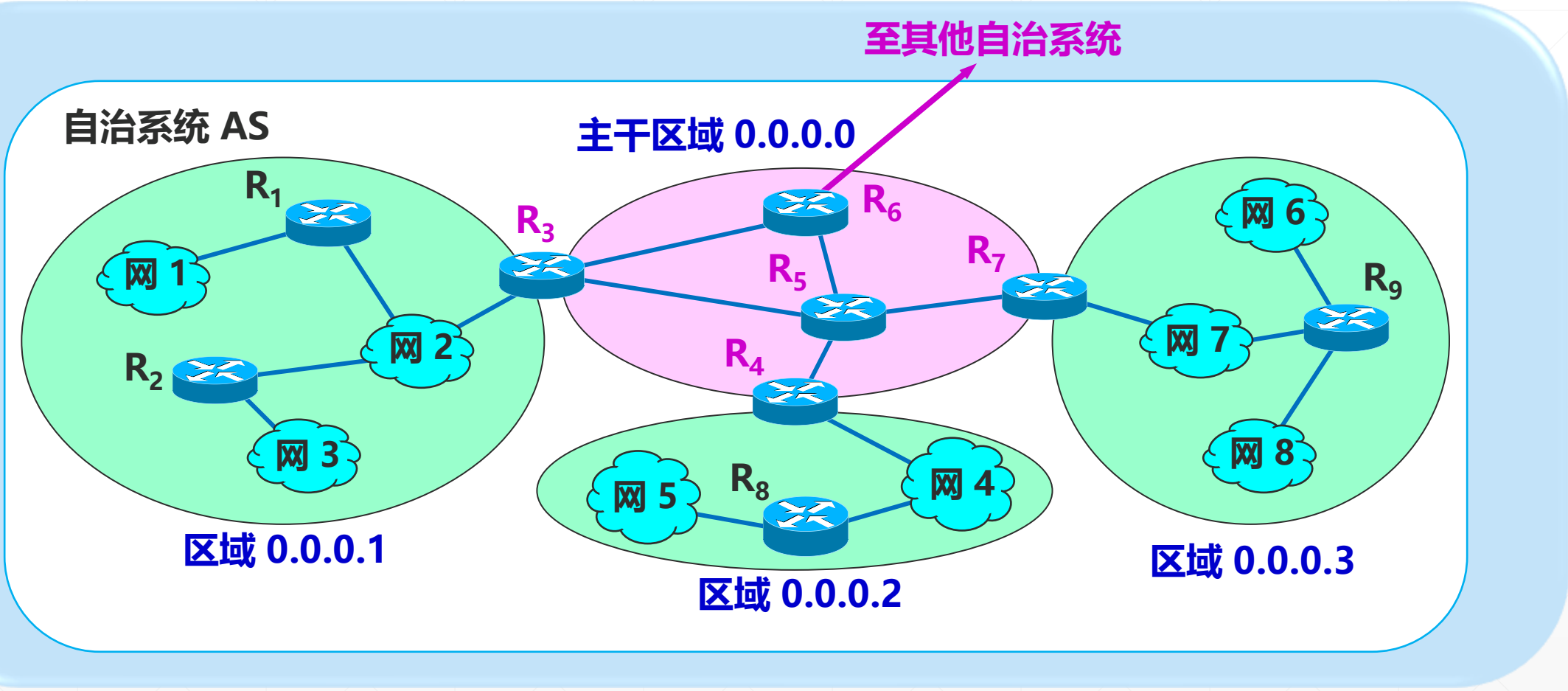
OSPF 划分为两种不同的区域



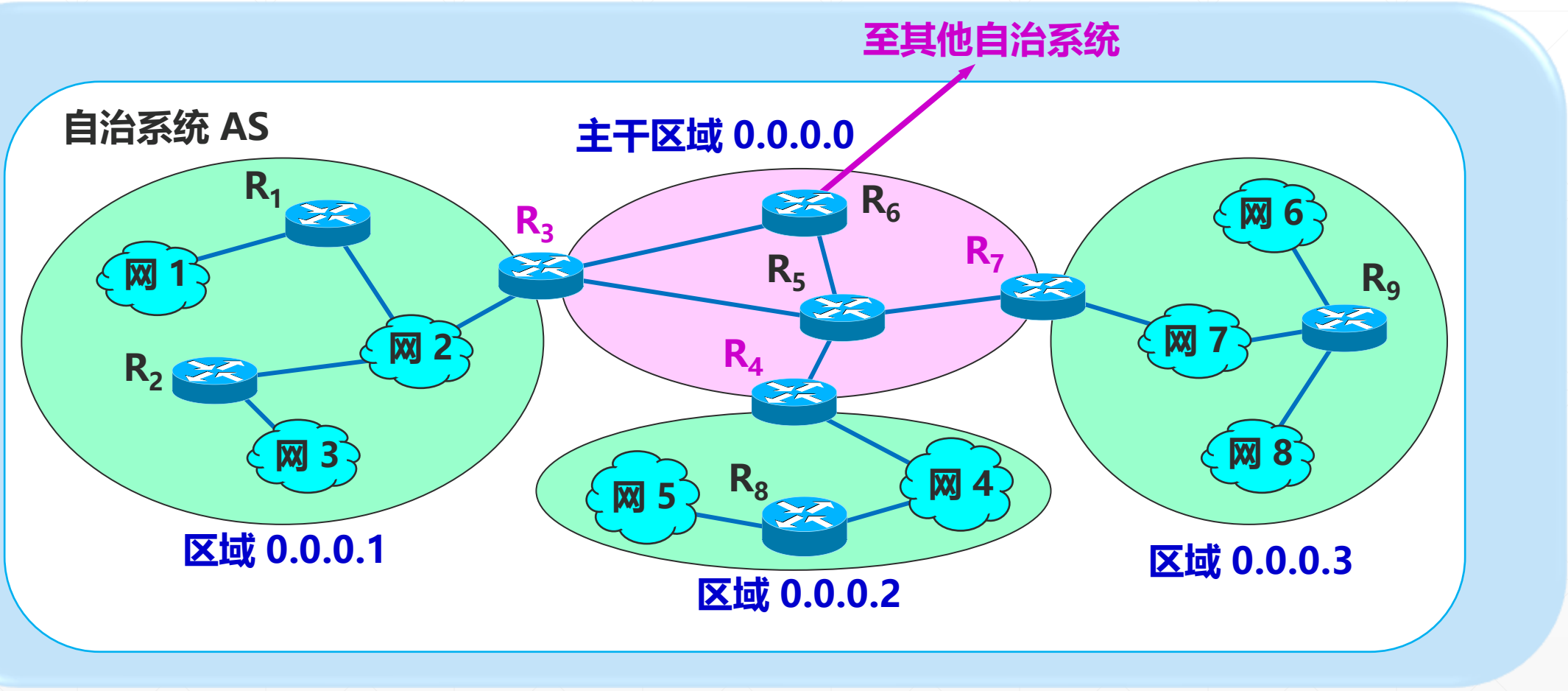
划分区域

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

主干路由器


至其他自治系统
自治系统 AS
主干区域 0.0.0.0
R1
网 1
网 2
R2
网 3
区域 0.0.0.1
R3
R6
R5
R7
R4
网 5
R8
网 4
区域 0.0.0.2
网 6
R9
网 7
网 8
区域 0.0.0.3

区域边界路由器


至其他自治系统
自治系统 AS
主干区域 0.0.0.0
R1
网 1
网 2
R2
网 3
区域 0.0.0.1
R3
R6
R5
R7
R4
网 5
R8
网 4
区域 0.0.0.2
网 6
R9
网 7
网 8
区域 0.0.0.3

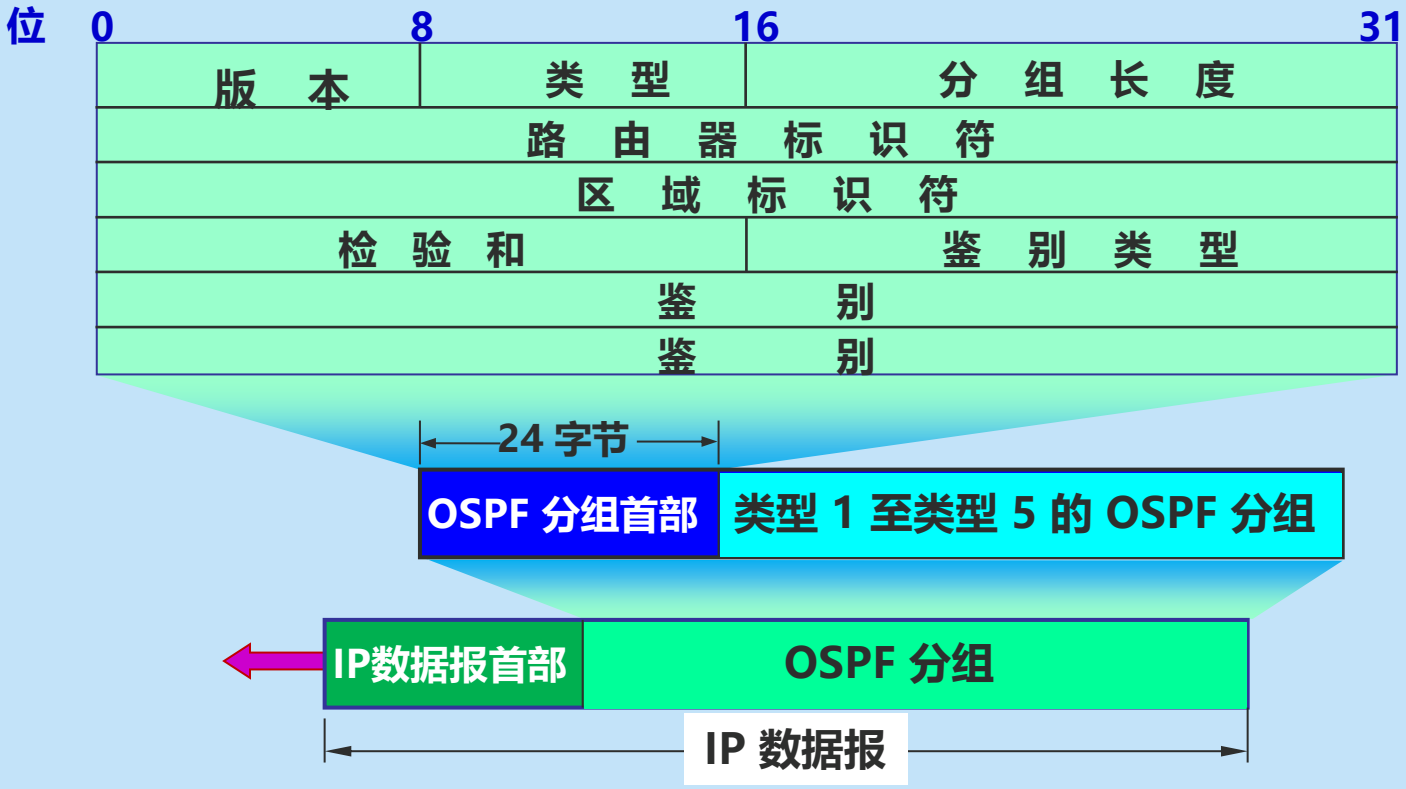
OSPF 直接用 IP 数据报传送

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

OSPF 的其他特点

- OSPF 对不同的链路可根据 IP 分组的不同服务类型 TOS 而设置成不同的代价。因此，**OSPF 对于不同类型的业务可计算出不同的路由**。
- 如果到同一个目的网络有多条相同代价的路径，那么可以将通信量分配给这几条路径。这叫做**多路径间的负载平衡**。
- 所有在 OSPF 路由器之间交换的分组都具有**鉴别**的功能。
- **支持可变长度的子网划分和无分类编址 CIDR**。
- 每一个链路状态都带上一个 32 位的序号，序号越大状态就越新。

OSPF 分组



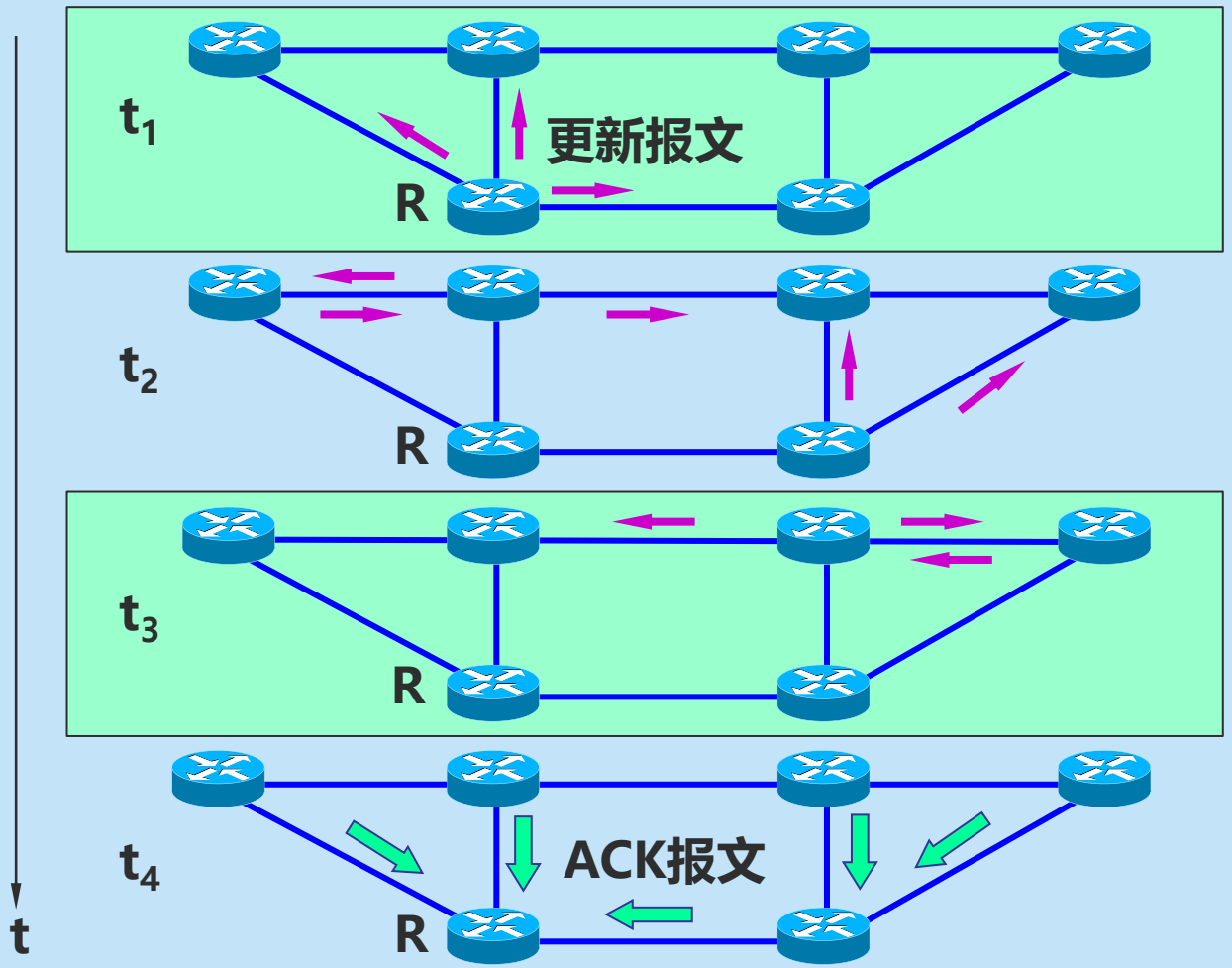
OSPF 分组用 IP 数据报传送

2. OSPF 的五种分组类型

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

OSPF 的基本操作


确定可达性
问候
问候
达到数据库的同步
数据库描述
数据库描述
数据库描述
数据库描述
新情况下的同步
链路状态请求
链路状态更新
链路状态确认



OSPF 使用可靠的洪泛法发送更新分组

OSPF 的其他特点

- OSPF 还规定每隔一段时间，如 30 分钟，要刷新一次数据库中的链路状态。
- 由于一个路由器的链路状态只涉及到与相邻路由器的连通状态，因而与整个互联网的规模并无直接关系。因此**当互联网规模很大时，OSPF 协议要比距离向量协议 RIP 好得多。**
- **OSPF 没有“坏消息传播得慢”的问题，**据统计，其响应网络变化的时间小于 100 ms。

指定的路由器

- 多点接入的局域网采用了**指定的路由器** (designated router) 的方法，**使广播的信息量大大减少**。
- 指定的路由器**代表**该局域网上所有的链路向连接到该网络上的各路由器发送状态信息。

RIP v.s. OSPF

属性	RIP	OSPF
算法类型	距离向量	链路状态
具体算法	Bellman-Ford	Dijkstra
收敛速度	慢	快
适用网络大小	小规模或中等规模	小规模/大规模均可
设备资源占用	CPU/内存占用较小	较大
网络资源占用	占用较大带宽（发送DV、转发表等）	较小（只需发送较小的update信息）
度量	基于跳数	基于带宽
网络设计	平面网络（Flat）	支持层次结构（Hierarchical network）
基于协议	UDP（端口号20）	IP（端口号89）

外部网关协议 – BGP协议

Boarder Gateway Protocol

外部网关协议 BGP

- BGP 是不同自治系统的路由器之间交换路由信息的协议。
- BGP 较新版本是 2006 年 1 月发表的 BGP-4（BGP 第 4 个版本），即 RFC 4271 ~ 4278。
- 可以将 BGP-4 简写为 BGP。

BGP 使用环境不同

- 互联网的规模太大，使得自治系统之间路由选择非常困难。对于自治系统之间的路由选择，要寻找最佳路由是很不现实的。
  1. 当一条路径通过几个不同 AS 时，要想对这样的路径计算出有意义的代价是不太可能的。
  2. **比较合理的做法是在 AS 之间交换“可达性”信息。**
- 自治系统之间的路由选择必须考虑有关**策略**。
- 因此，边界网关协议 BGP 只能是力求寻找一条能够到达目的网络且**比较好的路由**（不能兜圈子），而**并非要寻找一条最佳路由**。

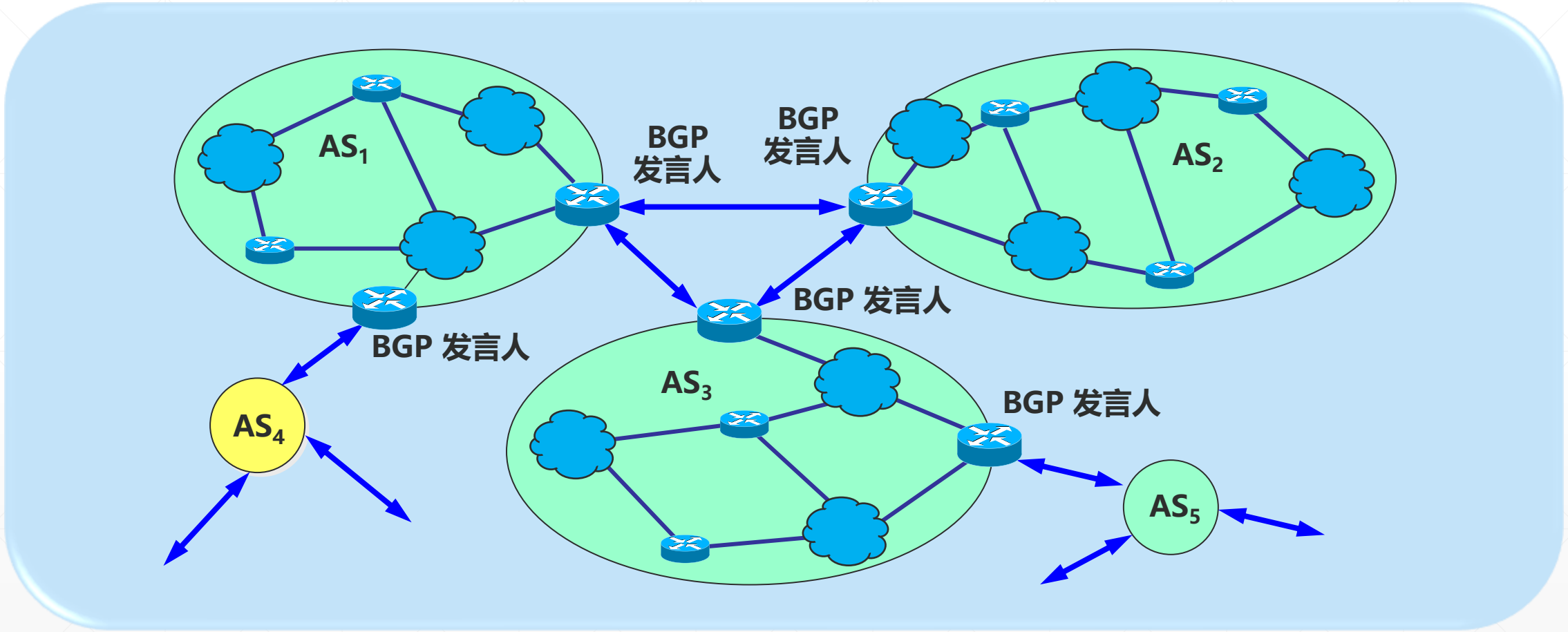
BGP 发言人

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

BGP 交换路由信息

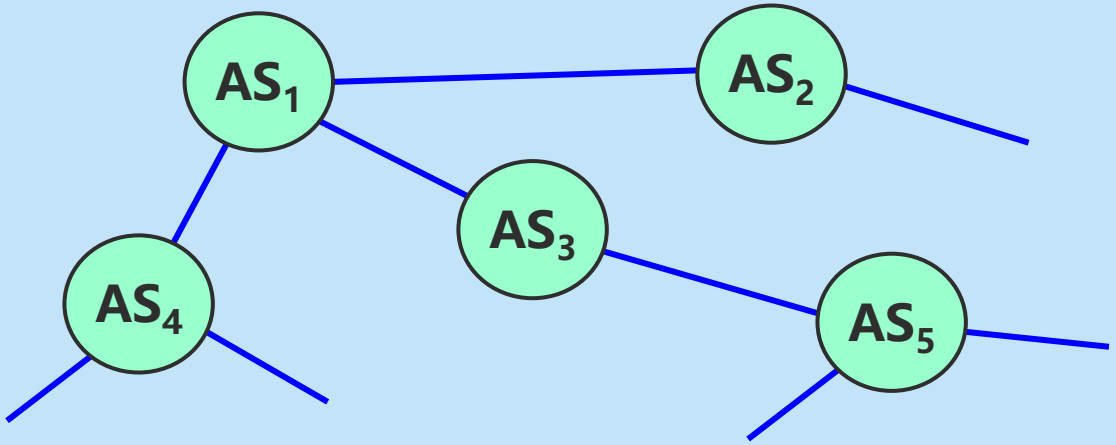
- 一个 BGP 发言人与其他自治系统中的 BGP 发言人要交换路由信息，就要先建立 **TCP 连接**，然后在此连接上交换 BGP 报文以建立 BGP **会话**(session)，利用 BGP 会话交换路由信息。
- 使用 TCP 连接能提供可靠的服务，也简化了路由选择协议。
- 使用 TCP 连接交换路由信息的两个 BGP 发言人，彼此成为对方的**邻站**(neighbor)或**对等站**(peer) 。

BGP 发言人和自治系统 AS 的关系

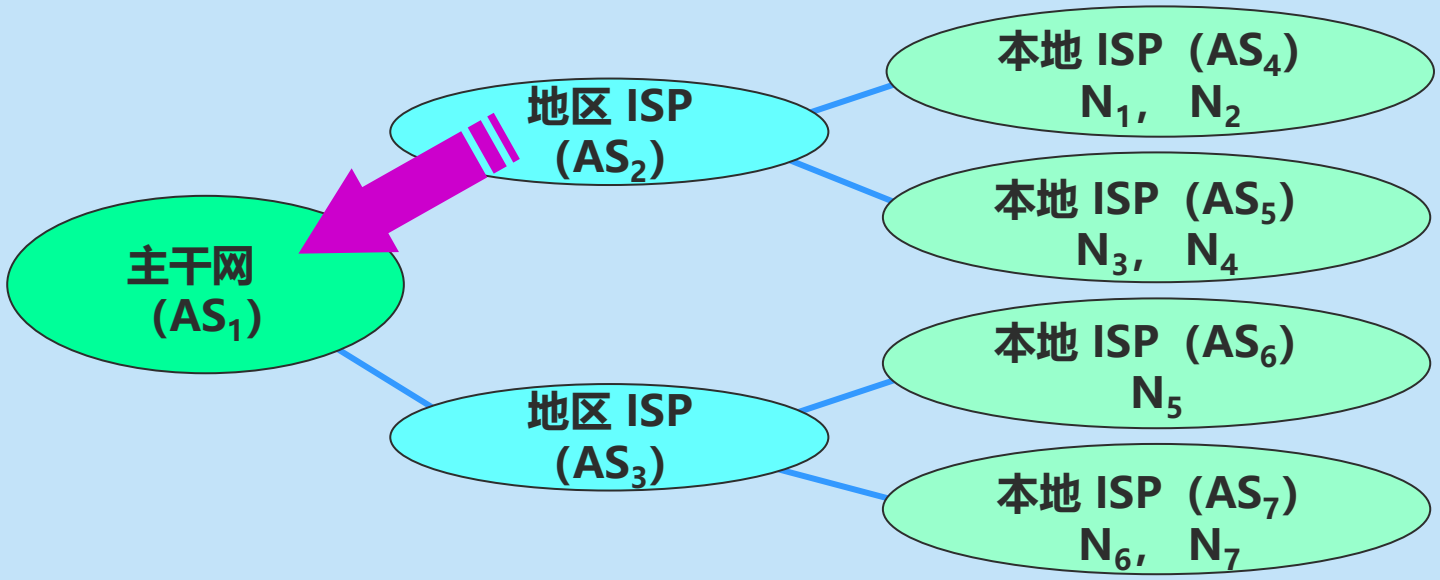


AS 的连通图举例

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



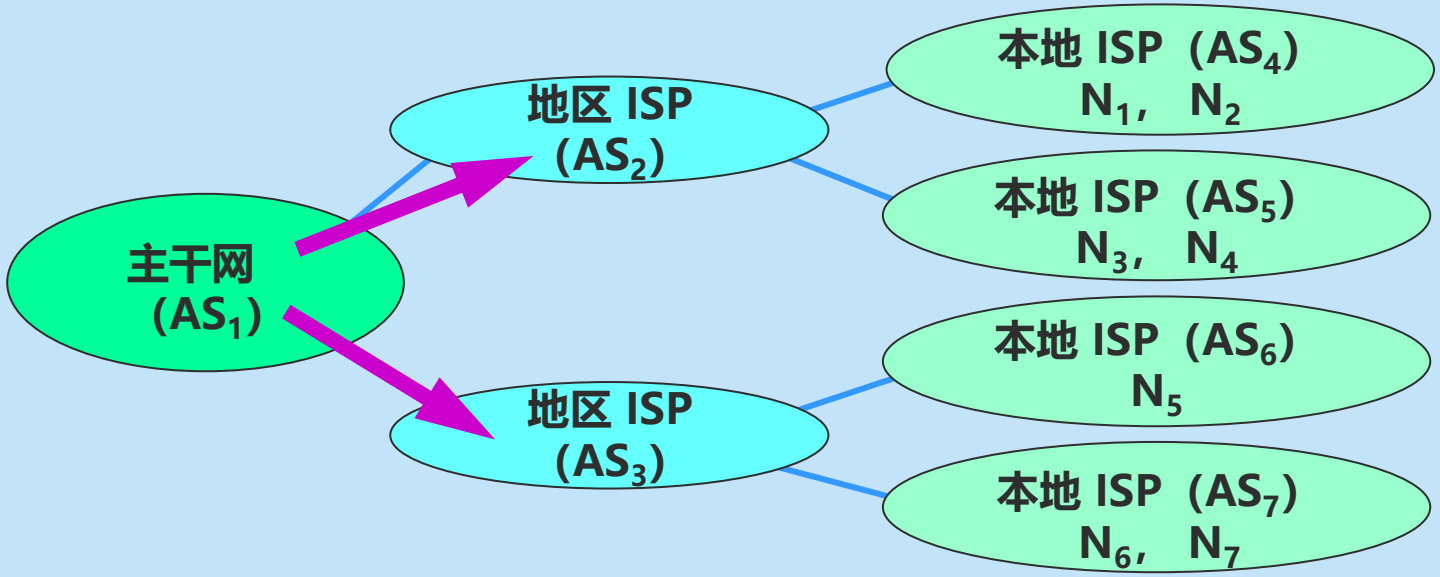
BGP 发言人交换路径向量



自治系统 AS_2 的 BGP 发言人通知主干网 AS_1 的 BGP 发言人：

“要到达网络 N_1、 N_2、N_3 和 N_4 可经过 AS_2。”

BGP 发言人交换路径向量



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

BGP 协议的特点

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

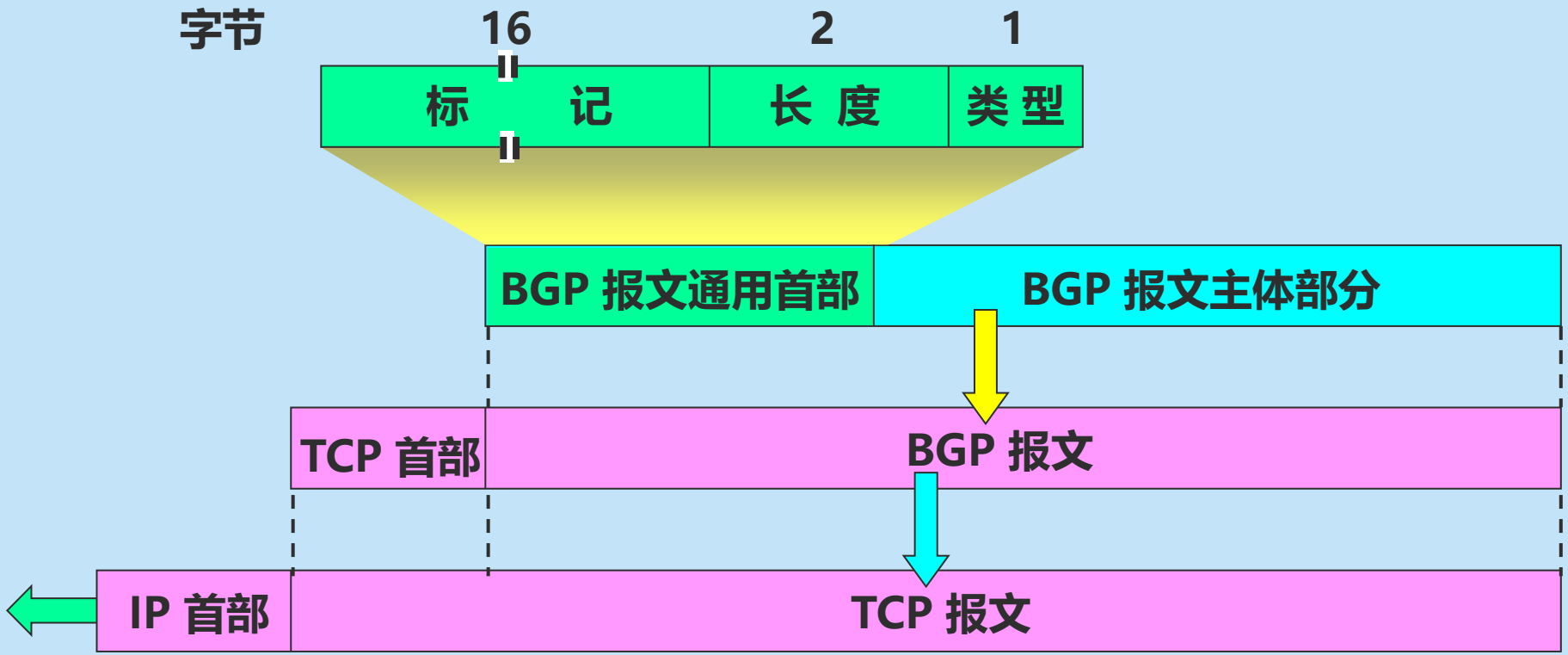
BGP 协议的特点

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

BGP-4 共使用四种报文

1. 打开 (OPEN) 报文，用来与相邻的另一个BGP发言人建立关系。
2. 更新 (UPDATE) 报文，用来发送某一路由的信息，以及列出要撤消的多条路由。
3. 保活 (KEEPALIVE) 报文，用来确认打开报文和周期性地证实邻站关系。
4. 通知 (NOTIFICATION) 报文，用来发送检测到的差错。

BGP 报文具有通用首部



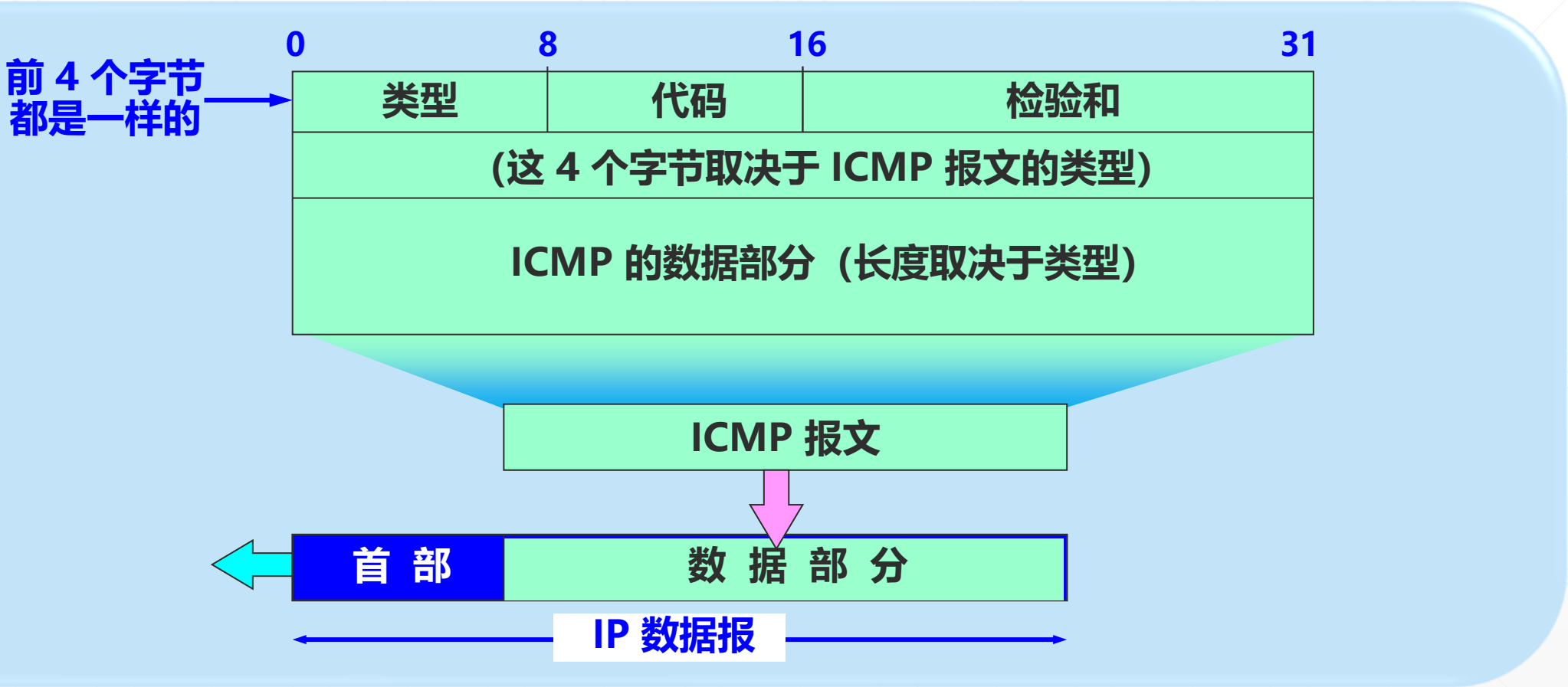
ICMP协议

Internet Control Message Protocol

网际控制报文协议 ICMP

- 为了更有效地转发 IP 数据报和提高交付成功的机会，在网际层使用了网际控制报文协议 ICMP (Internet Control Message Protocol)。
- ICMP 是互联网的标准协议。
- ICMP 允许主机或路由器报告差错情况和提供有关异常情况的报告。
- 但 ICMP 不是高层协议（看起来好像是高层协议，因为 ICMP 报文是装在 IP 数据报中，作为其中的数据部分），而是 IP 层的协议。

ICMP 报文的格式


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

ICMP

- 具体来说，ICMP允许主机和路由器交换网络层信息，常用的有如下两类：
  - 报告差错情况（Error reporting）：主机/网络/端口/协议不可达
  - 回送请求/回答（Echo request/reply）：主要由PING采用

Type	Code	description
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

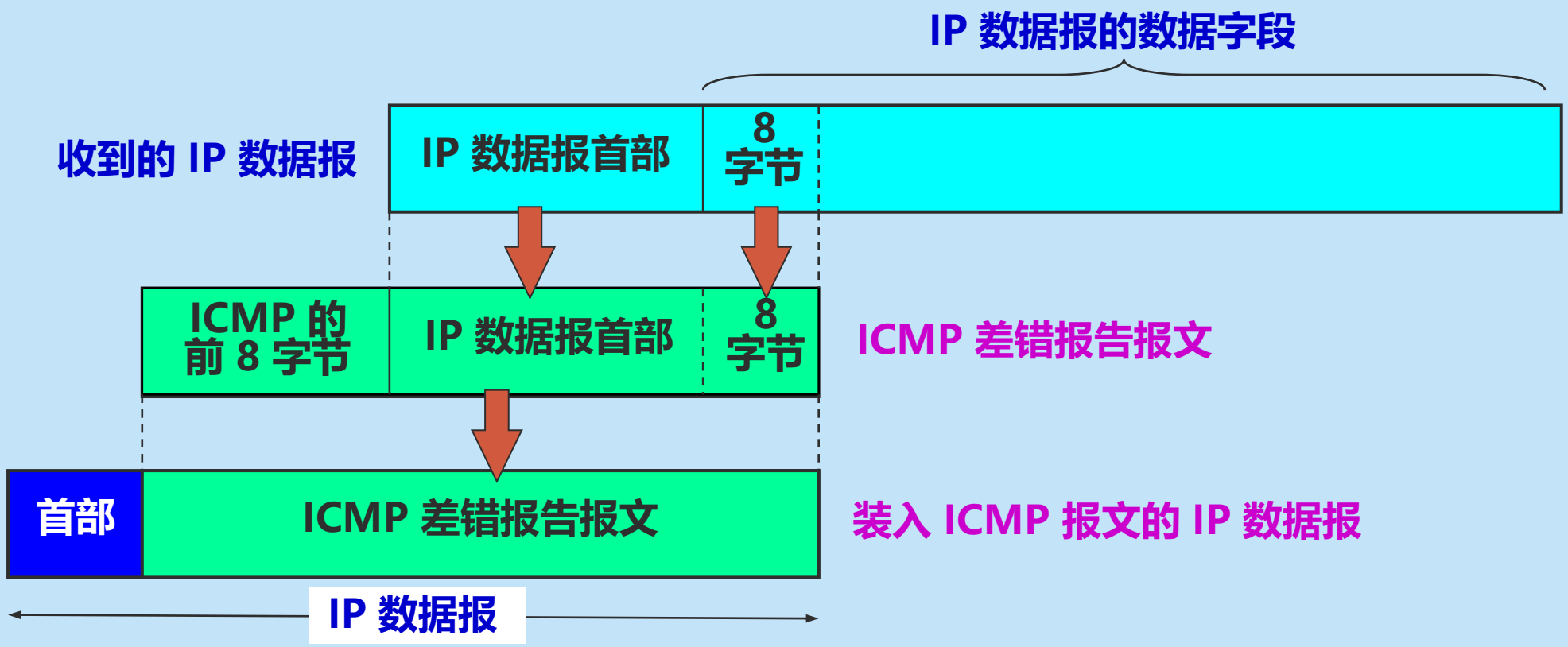
ICMP 报文的种类

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

ICMP 差错报告报文共有 4 种

- 终点不可达
- 时间超过
- 参数问题
- 改变路由（重定向）(Redirect)

ICMP 差错报告报文的数据字段的内容



不应发送 ICMP 差错报告报文的几种情况

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

ICMP 询问报文有两种

- 回送请求和回答报文
- 时间戳请求和回答报文

下面的几种 ICMP 报文不再使用：

- 信息请求与回答报文
- 掩码地址请求和回答报文
- 路由器询问和通告报文
- 源点抑制报文

ICMP 的应用举例

PING (Packet InterNet Groper)

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

PING 的应用举例

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

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

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

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

用 PING 测试主机的连通性

ICMP 的应用举例

Traceroute 的应用举例

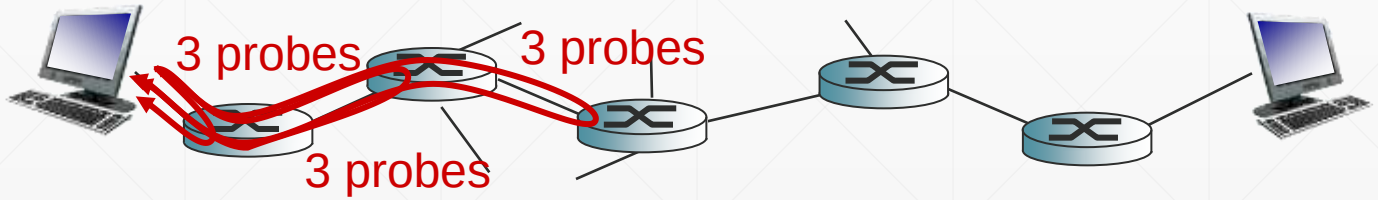
- 在 Windows 操作系统中这个命令是 tracert。
- **用来跟踪一个分组从源点到终点的路径。**
- 它利用 IP 数据报中的 TTL 字段和 ICMP 时间超过差错报告报文实现对从源点到终点的路径的跟踪。

- source sends series of UDP segments to destination
  - first set has TTL =1
  - second set has TTL=2, etc.
  - unlikely port number
- when datagram in *n*th set arrives to nth router:
  - router discards datagram and sends source ICMP message (type 11, code 0)
  - ICMP message include name of router & IP address

- when ICMP message arrives, source records RTTs

stopping criteria:

- UDP segment eventually arrives at destination host
- destination returns ICMP "port unreachable" message (type 3, code 3)
- source stops



ICMP 的应用举例

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

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

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

Trace complete.
```

用 tracert 命令获得到目的主机的路由信息

MPLS

Multi-Protocol Label Switching

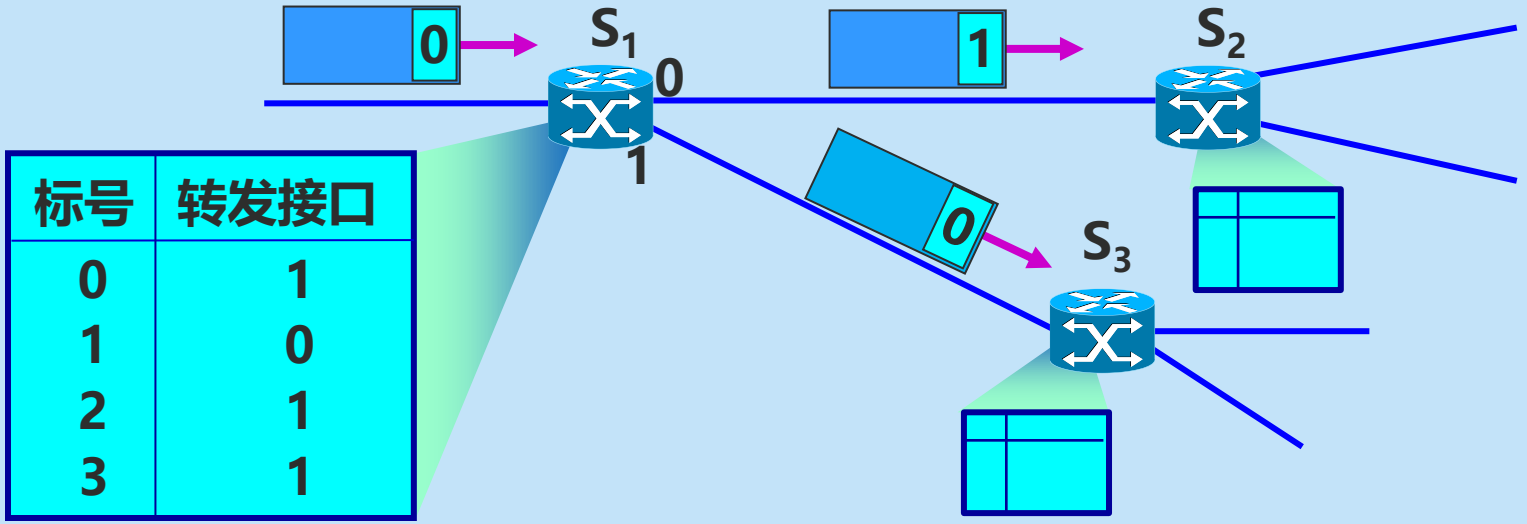
MPLS概述

- 事实上，MPLS常常被称为2.5层协议（Layer 2.5）
- 处于网络层和数据链路层之间
- 作为补充知识了解
  - 事实上和网络层数据平面/控制平面没什么关系

多协议标记交换 MPLS

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

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



标号	转发接口
0	1
1	0
2	1
3	1

S_1 的转发表

MPLS 特点

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

MPLS 的工作原理

1. 基本工作过程

- **IP 分组的转发**

(1) 在传统的 IP 网络中，分组每到达一个路由器后，都**必须提取出其目的地址，按目的地址查找路由表**，并按照“**最长前缀匹配**”的原则找到下一跳的 IP 地址（请注意，前缀的长度是不确定的）。

(2) 当网络很大时，查找含有大量项目的路由表要花费很多的时间。

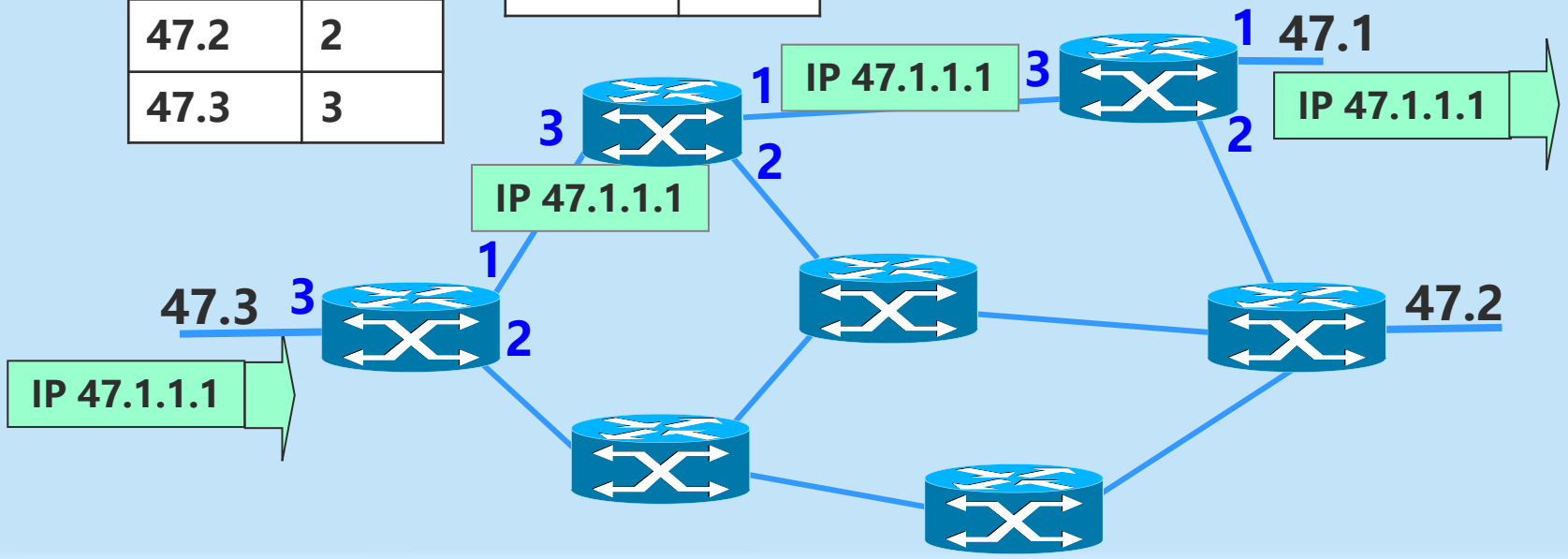
(3) 在出现突发性的通信量时，往往还会使缓存溢出，这就会引起分组丢失、传输时延增大和服务质量下降。

IP 分组的转发

Dest	Out
47.1	1
47.2	2
47.3	3

Dest	Out
47.1	1
47.2	2
47.3	3

Dest	Out
47.1	1
47.2	2
47.3	3



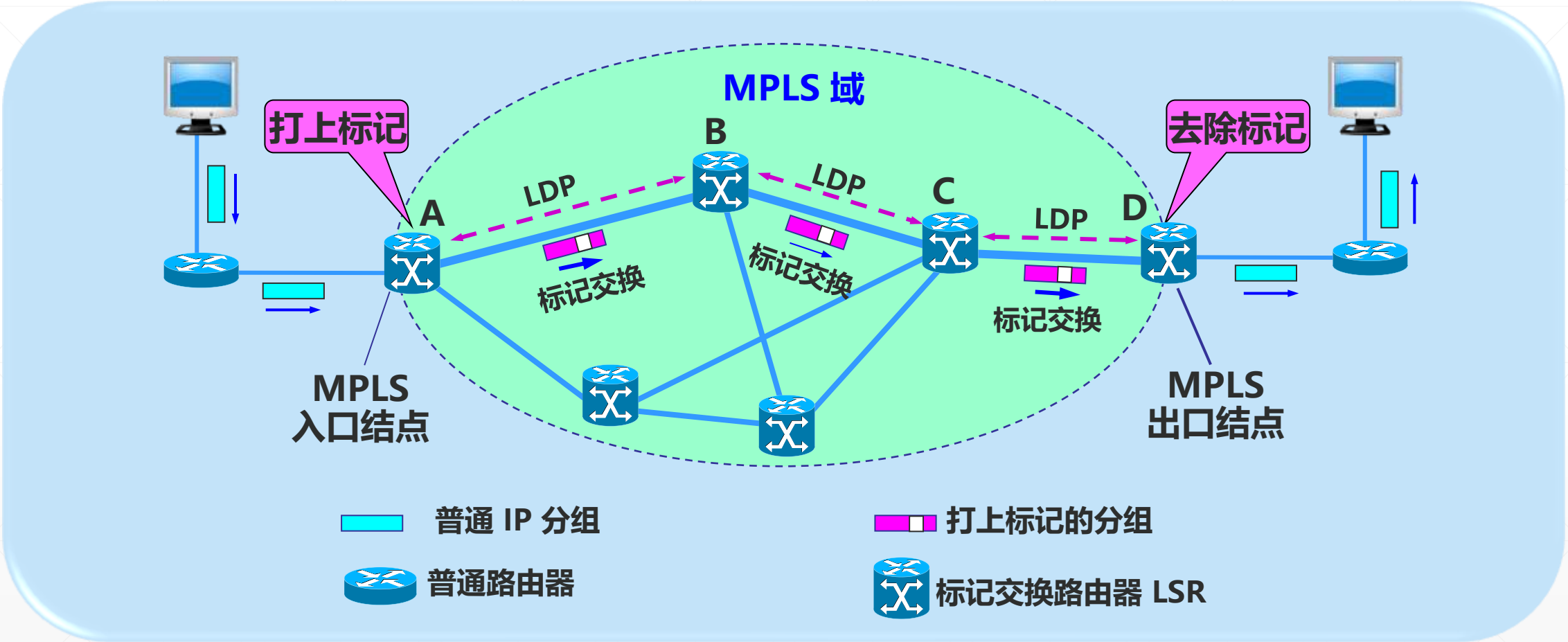
MPLS 协议的基本原理

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

MPLS 协议的基本原理

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

MPLS 协议的基本原理



MPLS 的基本工作过程

(1) MPLS 域中的各 LSR 使用专门的**标记分配协议 LDP** 交换报文，并找出**标记交换路径 LSP**。各 LSR 根据这些路径**构造出分组转发表**。

(2) 分组进入到 MPLS 域时， MPLS **入口结点把分组打上标记**，并按照转发表将分组转发给下一个 LSR。给 IP 数据报打标记的过程叫做**分类** (classification)。

MPLS 的基本工作过程

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

转发表

入接口	入标记	出接口	出标记
0	3	1	1

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

MPLS 的基本工作过程

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

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

2. 转发等价类 FEC

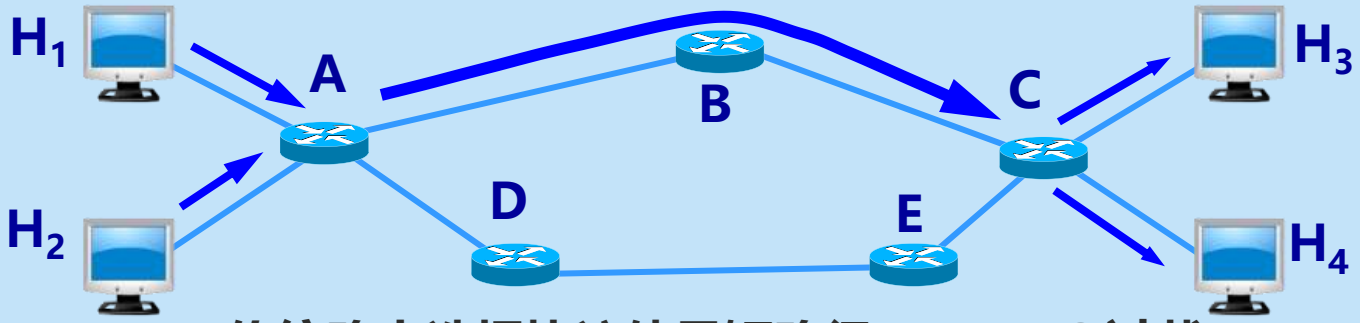
- MPLS 有个很重要的概念就是转发等价类 FEC (Forwarding Equivalence Class)。
- “**转发等价类**”就是路由器**按照同样方式对待**的分组的集合。

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

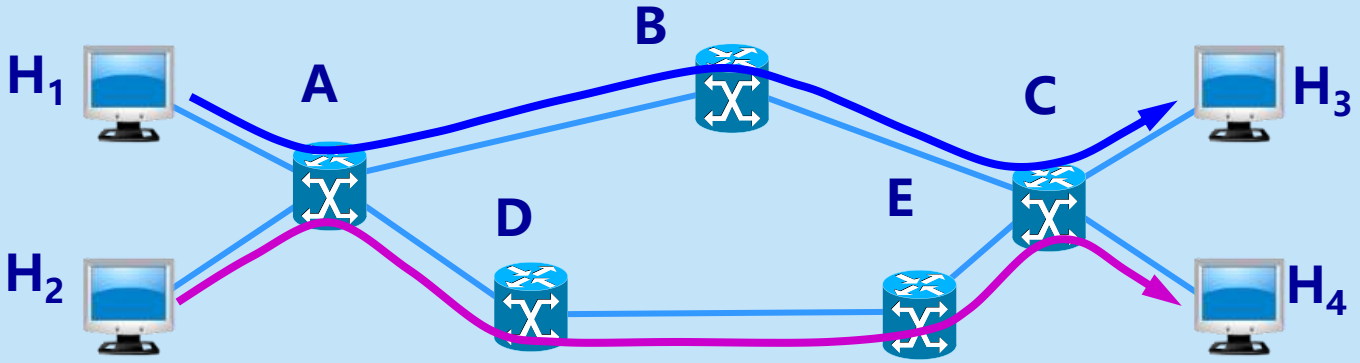
2. 转发等价类 FEC

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

FEC 用于负载平衡



(a) 传统路由选择协议使最短路径 A→B→C 过载



(b) 设置两种 FEC，利用 FEC 使通信量分散

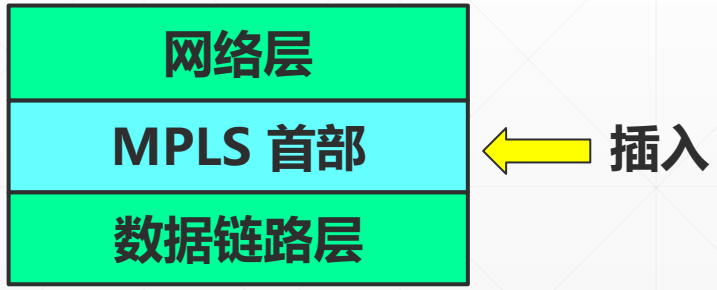
流量工程

(1) 网络管理员采用自定义的 FEC 就可以更好地管理网络的资源。

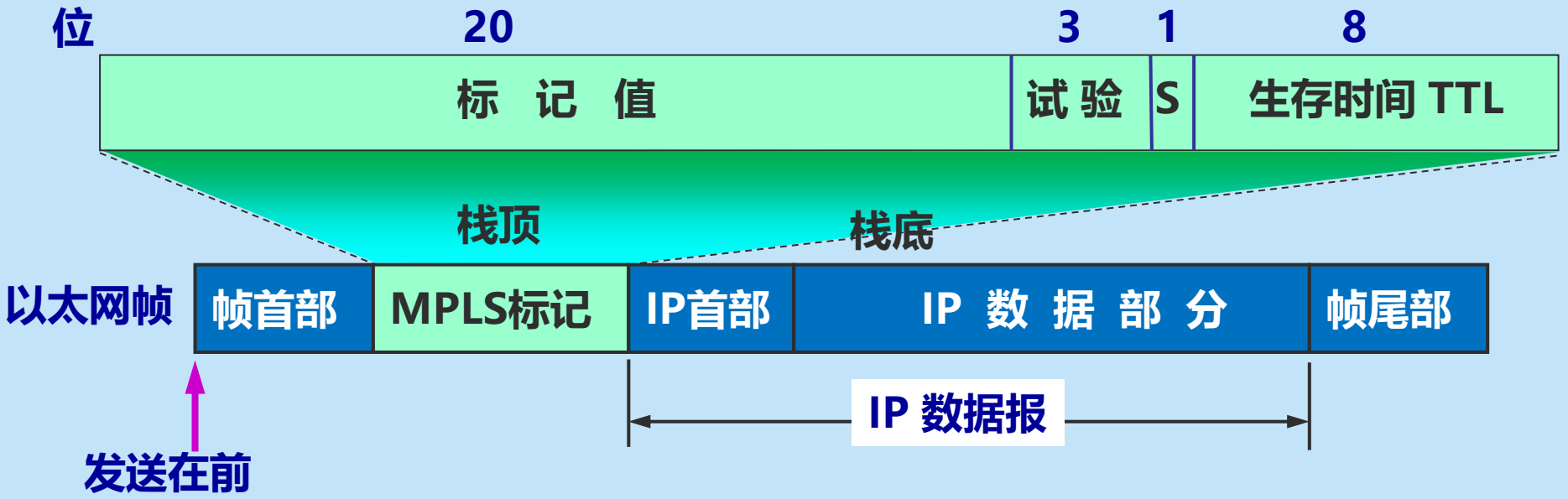
(2) 这种均衡网络负载的做法也称为**流量工程 TE** (Traffic Engineering) 或通信量工程。

MPLS 首部的位置与格式

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



MPLS 首部的格式



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

MPLS 首部的格式

- MPLS 首部共包括以下**四个字段**：

  (1) 标记值（占 20 位）。可以同时容纳高达 2^20 个流（即 1048576 个流）。实际上几乎没有哪个 MPLS 实例会使用很大数目的流，因为通常需要管理员人工管理和设置每条交换路径。

  (2) 试验（占 3 位）。目前保留用作试验。

  (3) 栈S（占 1 位）。在有“标记栈”时使用。

  (4) 生存时间TTL（占 8 位）。用来防止 MPLS 分组在 MPLS 域中兜圈子。