



## 第5章 网络层：控制平面

5

## 5.1 路由算法

链接状态

距离矢量

分层路由

6

## 5.2 因特网路由

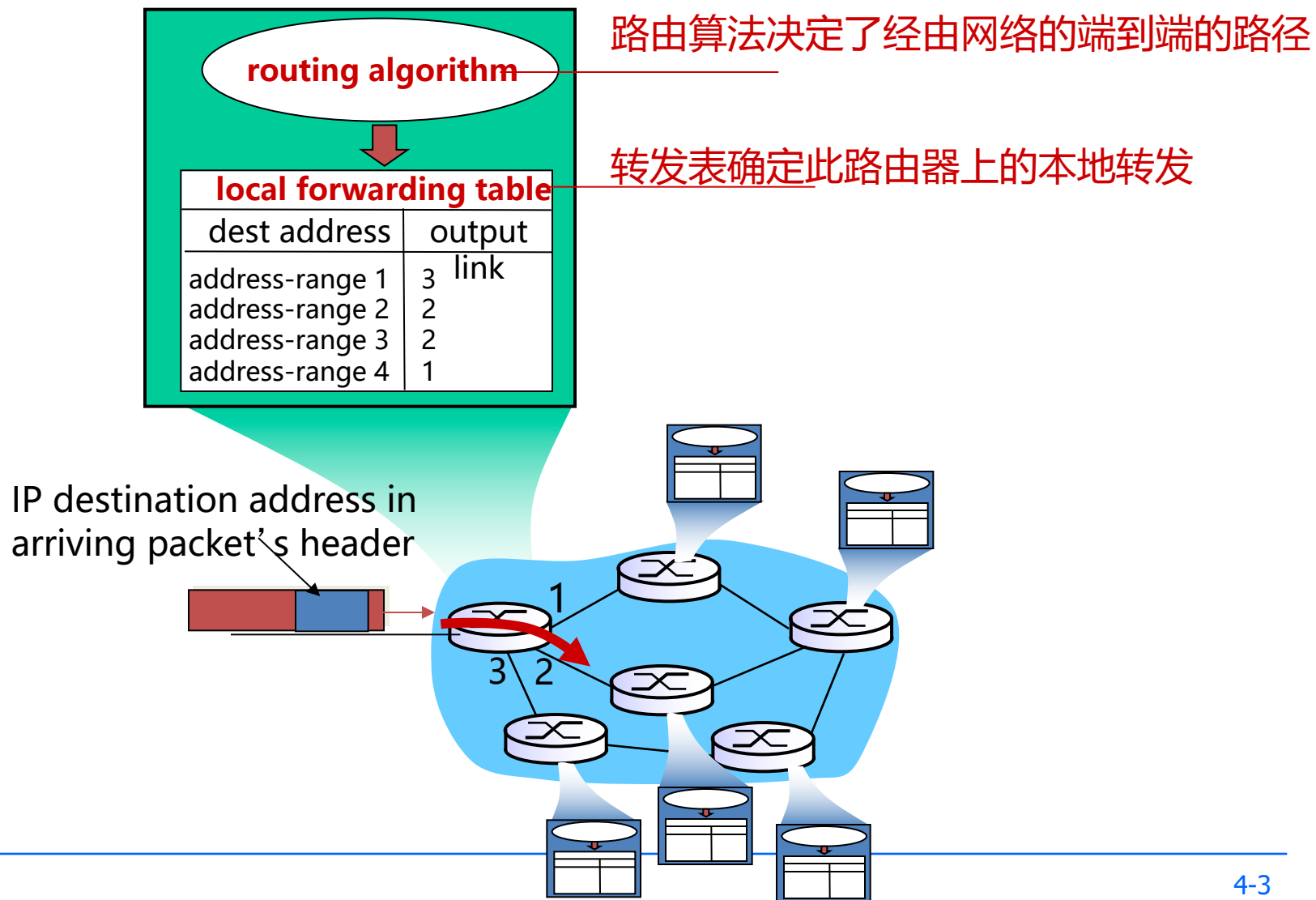
RIP

OSPF

BGP

7

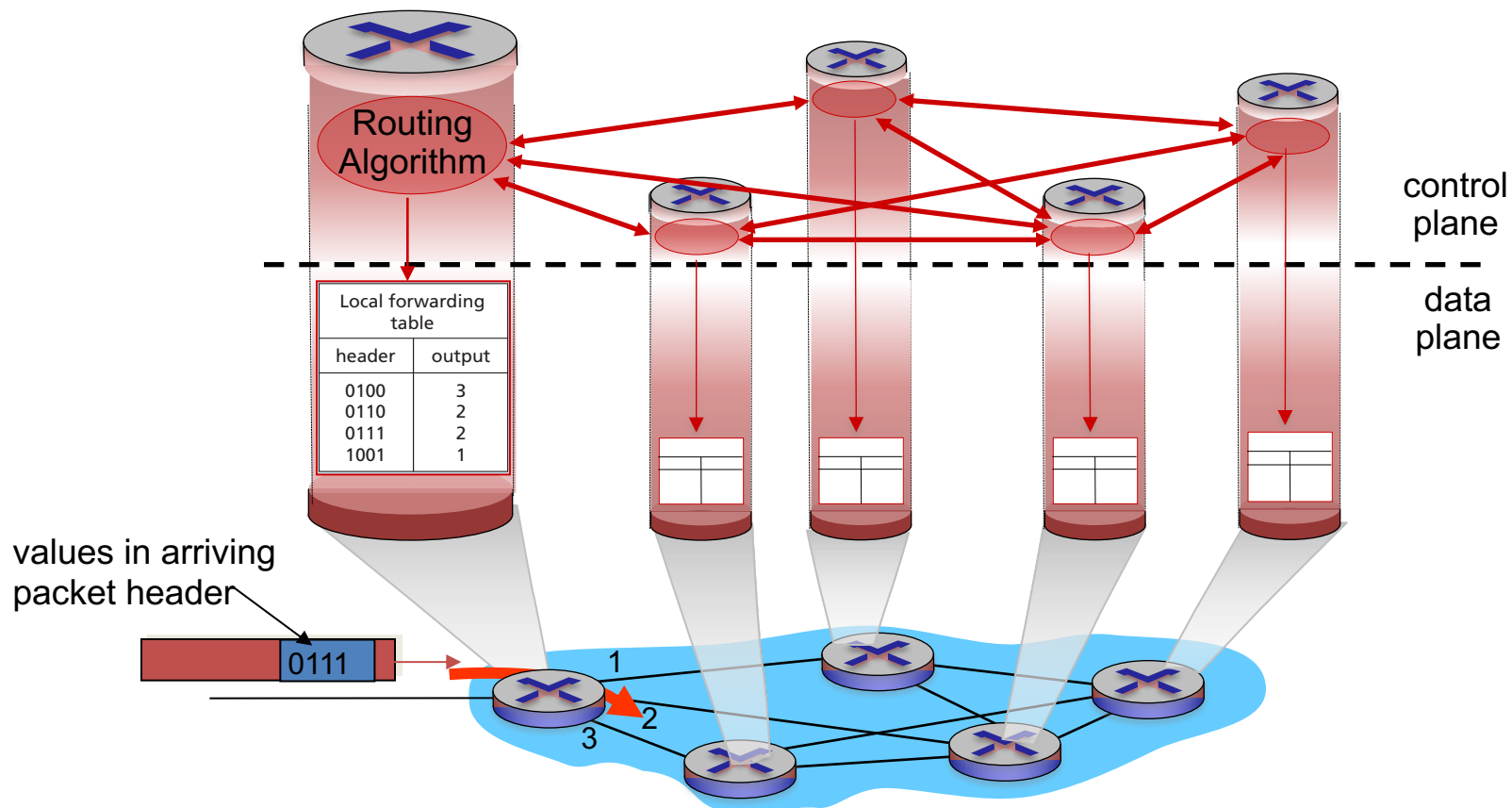
## 5.3 ICMP and SNMP



# 网络层： 数据平面 vs 控制平面

基于单独的路由器的路由算法

*in each and every router* interact in the control plane



# 什么是路由?



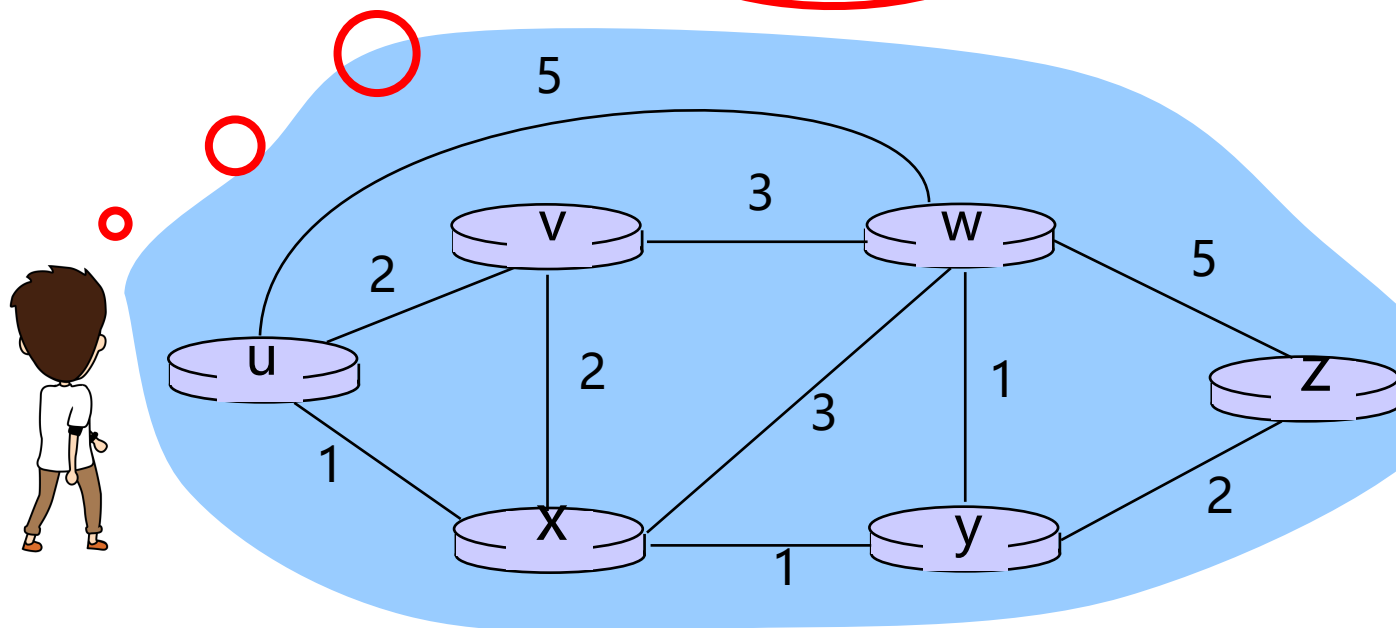
您将走哪条路?

你为什么选择它?

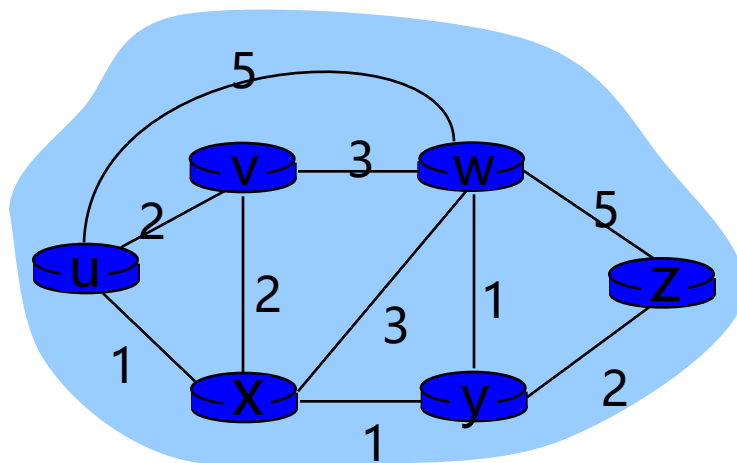
# 什么是路由?

谨记:

1. 工具和分类。
2. 两个关键算法。
3. 问题和讨论。



# 网络拓扑结构可抽象为加权图



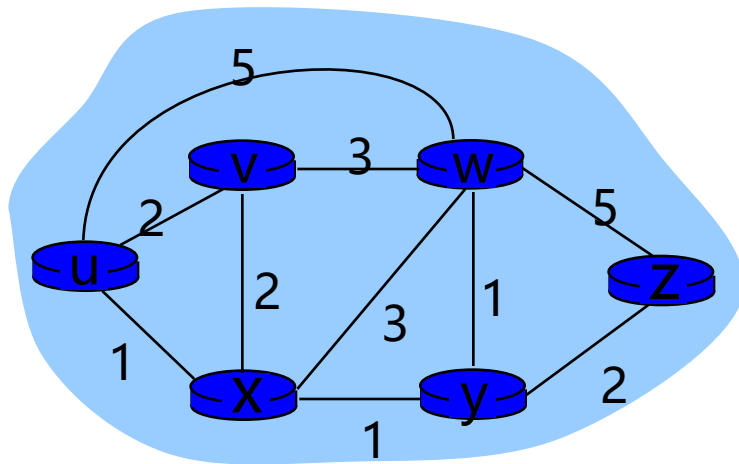
graph:  $G = (N, E)$

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

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

图抽象在其他网络上下文中很有用，例如，P2P，其中  $N$  是一组Peers， $E$  是一组 TCP 连接

# 图抽象：开销



$c(x, x') = \text{cost of link } (x, x')$

e.g.,  $c(w, z) = 5$

链路开销可以均为1,

也可以与带宽、拥塞等因素相关

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

**关键问题：**你和 Z 之间的最低成本路径是什么？

**路由算法：**找到最低成本路径的算法



*Q:全局还是非集中式的信息?*

*全局:*

- 所有路由器有完整的网络拓扑结构, 链路开销信息
- 链路状态算法

*非集中:*

- 路由器只知道与其物理邻接的路由器及其到这些邻接路由器的链路开销
- 周期性的执行路由算法, 并与邻接路由器交换信息
- 距离矢量路由算法

*Q:静态还是动态?*

*静态:*

- ❖ 路由长时间不会改变

*动态:*

- ❖ 路由变化较快
  - 周期性的更新
  - 当链路开销改变时, 可以及时响应

*Q:负载敏感与否?*

5

## 5.1 路由算法

链接状态

距离矢量

分层路由

6

## 5.2 因特网路由

RIP

OSPF

BGP

7

## 5.3 ICMP and SNMP

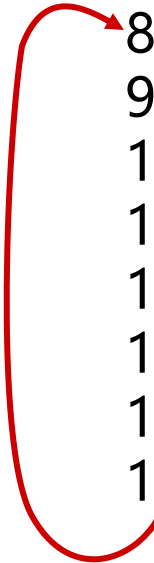
## *Dijkstra算法*

- 所有节点知道网络拓扑结构，链路开销信息。通过链路状态广播实现所有节点拥有相同的信息
- 计算从某一节点到其他所有的节点的最小开销路径，将结果存储于转发表中
- 迭代：经过k次迭代，就知道了到达k跳目的的最佳路径

## *notation:*

- $c(X, Y)$ : 从节点X到Y的链路开销；若非直连，则为无穷大
- $D(v)$ : 当前从起始端到达节点v的开销值
- $p(v)$ : 从起始端到达节点v的路径的前置节点
- $N'$ : 最佳路径应经确定的节点的集合

```
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  $D(v) = \infty$ 
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  $N'$ 
```

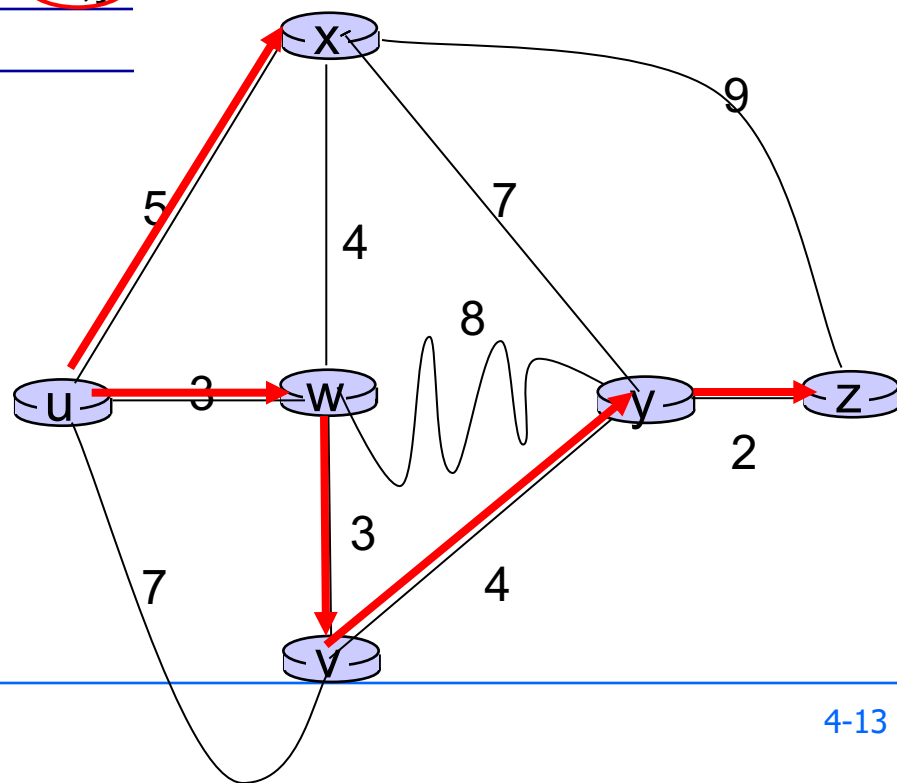


# Dijkstra算法：示例

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	$\infty$	$\infty$
1	uw	6,w		5,u	11,w	$\infty$
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					

## notes:

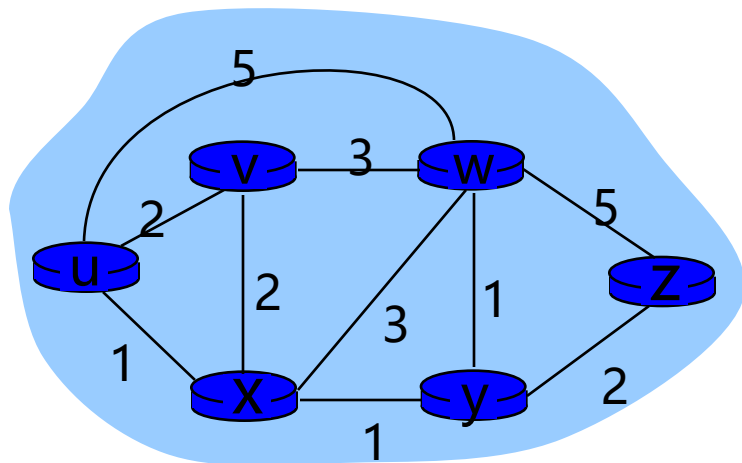
- ❖ 通过跟踪前置节点构造最短路径树。
- ❖ 联系可以存在（可以任意打破）



# Dijkstra算法：另一个例子

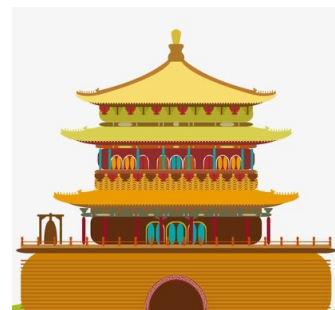
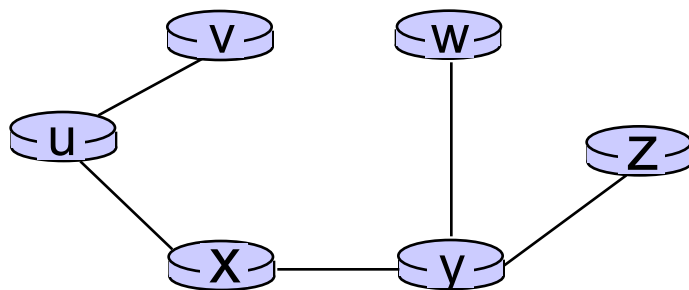
软件学院·计算机网络

Step	$N'$	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
------	------	--------------	--------------	--------------	--------------	--------------



# Dijkstra算法：示例 (2)

从 U 生成的最短路径树：



在 U 中生成的转发表：

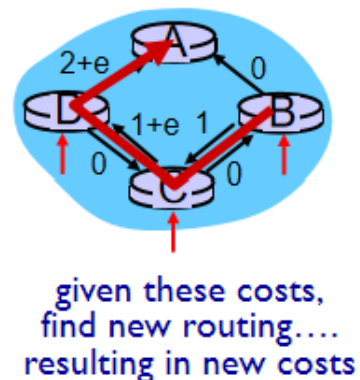
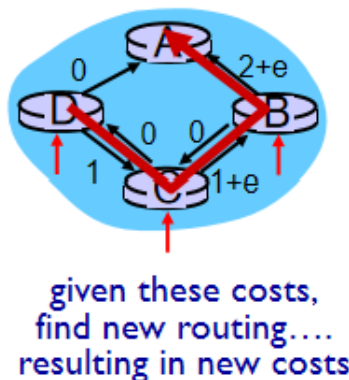
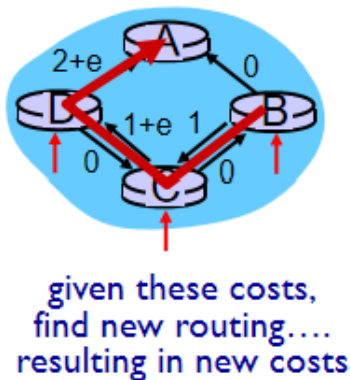
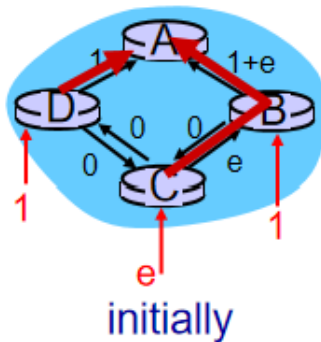
destination	link
v	(u,v)
w	(u,x)
x	(u,x)
y	(u,x)
z	(u,x)

算法复杂度:  $n$ 节点

- ❖ 每次迭代: 需要检查每个不在 $N$ 集合里的节点 $w$
- ❖  $n(n+1)/2$ :  $O(n^2)$
- ❖ 更优化的实现:  $O(n \log n)$

震荡:

- ❖ 例如: 链路开销等于链路上承载的负载





5

## 5.1 路由算法

链接状态

距离矢量

分层路由

6

## 5.2 因特网路由

RIP

OSPF

BGP

7

## 5.3 ICMP and SNMP

贝尔曼-福特方程 (动态规划)

*Let*

$d_x(y)$  := 从 X 到 Y 的最小成本路径的成本

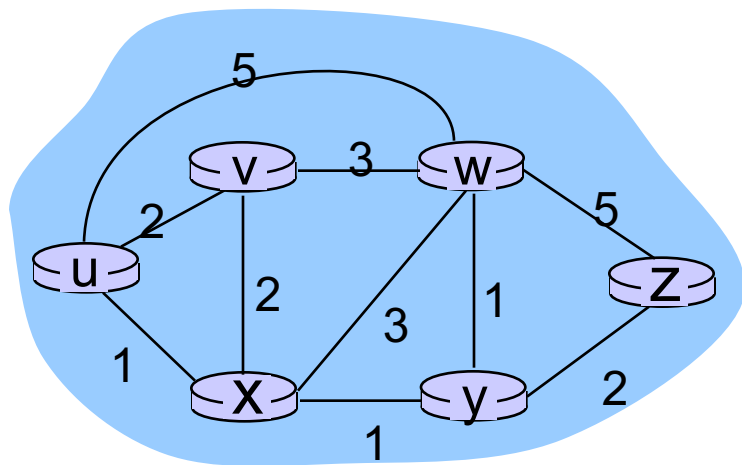
然后

$$d_x(y) = \min \{ c(x,v) + d_v(y) \}$$

从邻居 V 到目标Y 的成本

与邻居v之间的成本

在x的所有邻居v中取最小值



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

B-F equation says:

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

**在实践中很重要:**

- ❖ 达到最小值的节点是下一跳节点，直接用于转发表
- ❖ 自然适合于分布式系统，即邻居之间的通信

## ■ $D_x(y)$ = Bellman-Ford方程

- x 保持距离向量  $\mathbf{D}_x = [D_x(y): y \in N]$

## ■ 节点x:

- 知道每个邻居v的成本 :  $c(x, v)$
- 维护其邻居的距离矢量。对于每个邻居 v, x 保持

$$\mathbf{D}_v = [D_v(y): y \in N]$$

## 关键思想:

- ❖ 每个节点不时的向它的每个邻居发送它的距离向量副本
- ❖ 当节点x从它的某个邻居接收到一个新的距离向量时，它会使用Bellman-Ford方程更新自己的距离向量
- ❖ 若节点x的距离向量因更新发生改变，则它会向每个邻居发送更新后的距离向量，最终， $D_x(y)$  会收敛到从节点x到节点y的实际最小开销值

$$D_x(y) \leftarrow \min_v \{c(x,v) + D_v(y)\} \text{ for each node } y \in N$$

## 更新, 异步:

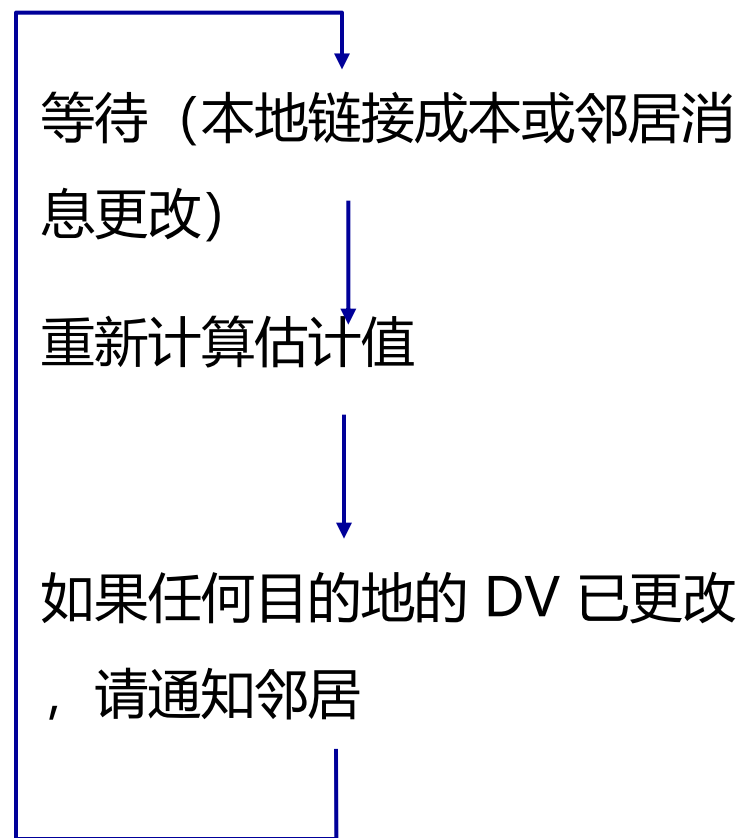
每个本地的更新是由于:

- 本地链路开销变化
- 从邻居得到距离向量的更新信息

## 分布式:

- 每个节点仅当自己的距离向量发生变化时才通知其邻居节点
- 若果必要, 邻居们紧接着再通知它们各自的邻居节点, 以此类推

## 每个节点:



**node x  
table**

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

**node y  
table**

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

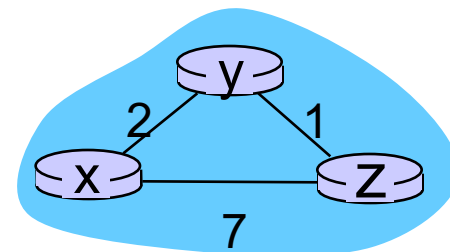
**node z  
table**

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

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

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

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



time

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

$$= \min\{2+0, 7+1\} = 2$$

**node x  
table**

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

**node y  
table**

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

**node z  
table**

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

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

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

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

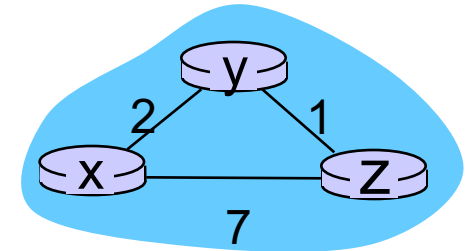
		cost to		
		x	y	z
from	x	0	2	3
	y	2	0	1
	z	3	1	0

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

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

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

$$= \min\{2+1, 7+0\} = 3$$

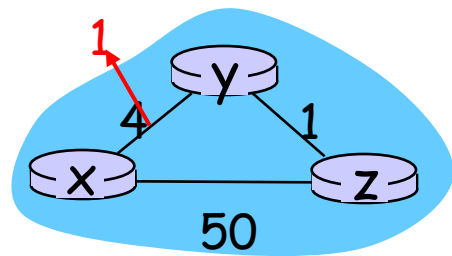


time



## 链接开销改变:

- ❖ 节点检测到本地链路开销改变
- ❖ 更新路由信息，重新计算距离向量
- ❖ 如果距离向量发生变化，则通知邻居节点



t1: y 检测链路成本变化，更新其 DV，通知其邻居。

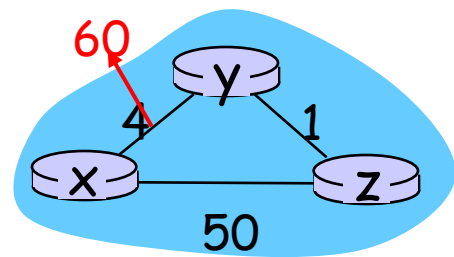
“good  
news  
travels  
fast”

t2 : z 接收来自 y 的更新，更新其表，计算新的最小成本到 x，向其邻居发送其 DV。

t3 : y 接收 z 的更新，更新其距离表。Y 的最小成本不会改变，因此 y 不会向 Z 发送消息。

## 链路开销改变:

- ❖ 节点检测本地链路成本变化。
- ❖ 坏消息传播缓慢 - “数到无穷大” 问题!
- ❖ 算法稳定之前的 44 次迭代：见文本



## 毒性逆转:

- ❖ 如果 Z 通过 Y 到达 X:
  - Z 告诉 Y 它 (Z) 到 X 的距离是无限的 (所以 Y 不会通过 Z 路由到 X)
- ❖ 这会完全解决计数到无穷大的问题吗?

# 链路状态与距离矢量算法的比较

## 消息复杂性

- **LS:**  $n$ 个节点,  $E$ 条链路,  $O(nE)$ 个报文发送

- **DV:** 仅在邻居之间交换报文, 收敛时间视情况而定

## 收敛速度

- **LS:** 需要 $O(nE)$ 个报文的复杂度为 $O(n^2)$ 算法, 可能会出现震荡

- **DV:** 收敛时间不定, 可能会遇上路由环路, 以及无穷计数的问题

**鲁棒性:** 如果路由器出现故障会怎样?

**LS:** 节点会通告一个错误的链路开销, 每个节点计算各自的路由表

■ **DV:** 节点通告错误的路径开销, 每个节点的路由表被其他节点使用, 错误通过网络蔓延开

5

## 5.1 路由算法

链接状态

距离矢量

分层路由

6

## 5.2 因特网路由

RIP

OSPF

BGP

7

## 5.3 ICMP and SNMP

# 层次路由

至此，我们所学习的路由知识都过于理想化

- ❖ 所有的路由器都是一样的
- ❖ 网络的“平”的
- ❖ 现实并非如此

规模：6亿

- 无法将所有路由信息存储在路由表
- 路由表交换将瘫痪所有链路

管理自治

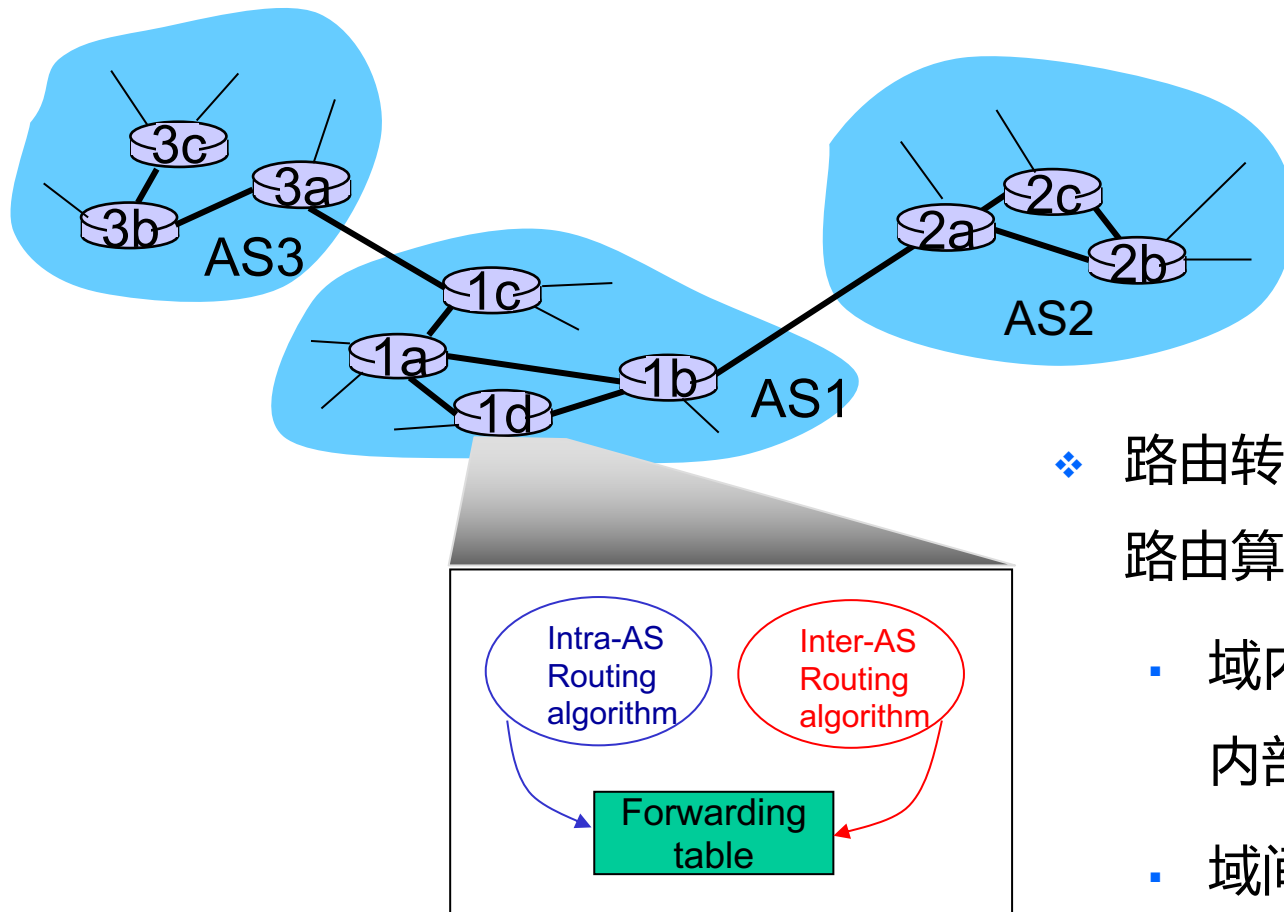
- ❖ internet=网络之网络
- ❖ 每个网络都希望管理各自的路由信息

# 层次路由

- 将路由器组织到自治域系统 (AS) 中
- 在同一AS中的路由器运行相同的路由协议
  - intra-AS路由协议
  - 在不同AS里的路由器可能运行不同的intra-AS路由协议

## 网关路由器：

- 位于AS的边缘
- 连接到其他AS的路由器

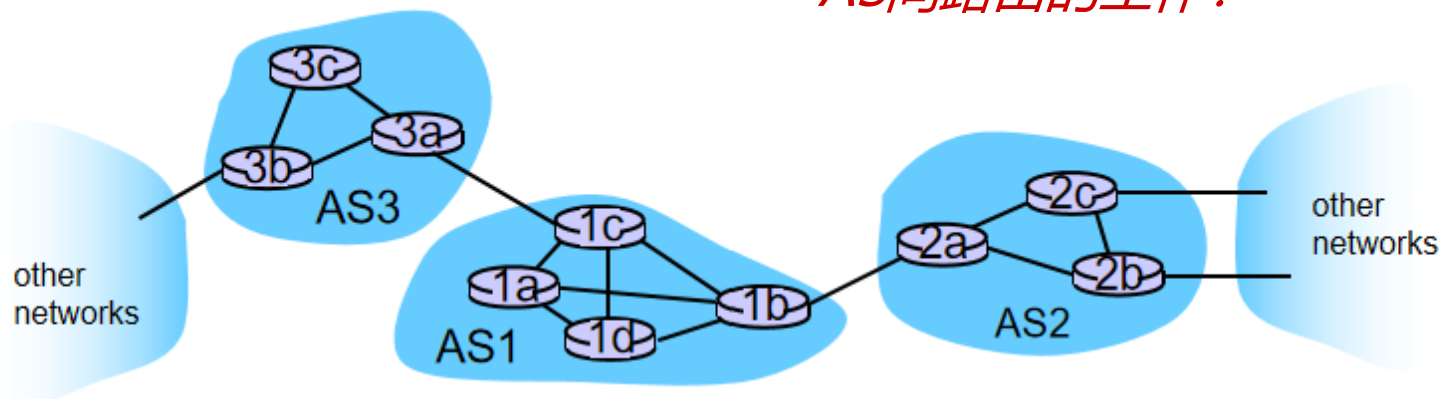


- ❖ 假设AS1的路由器收到目的地址在AS1之外的数据报:
  - 该路由器需要把分组转发给网  
关路由器，哪一个呢？

*AS1必须:*

- 1、学习通过AS2可以到达那些目的地  
，通过AS2可以到达那些目的地
- 2、将这些可达信息通告给AS1中的所有路由器

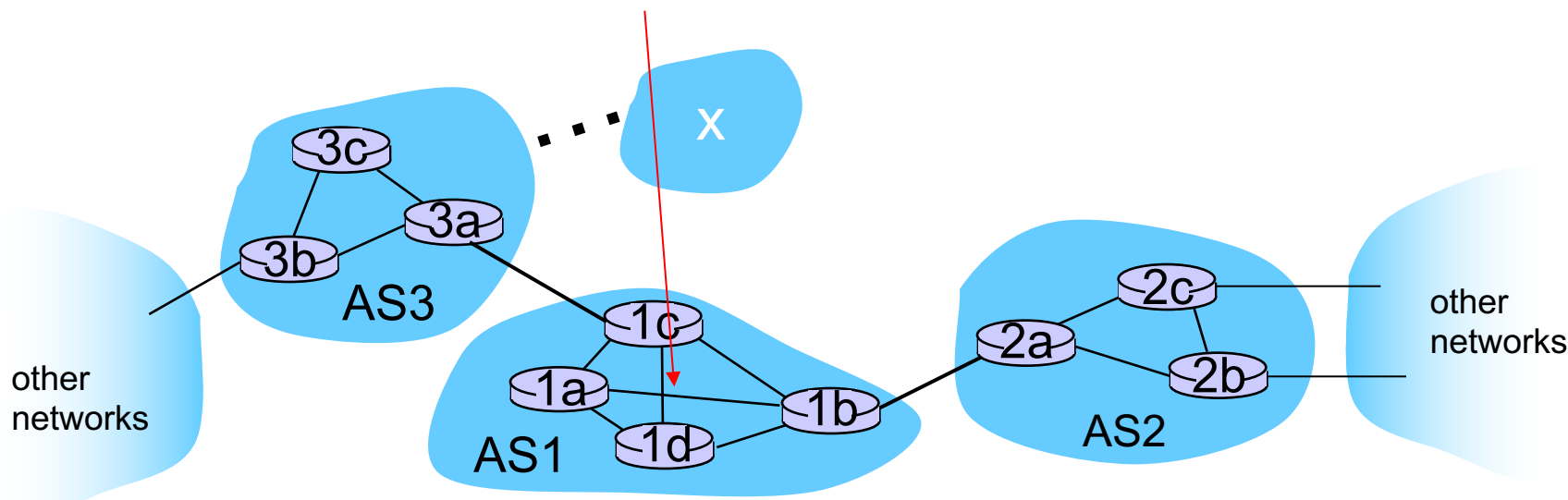
*AS间路由的工作!*





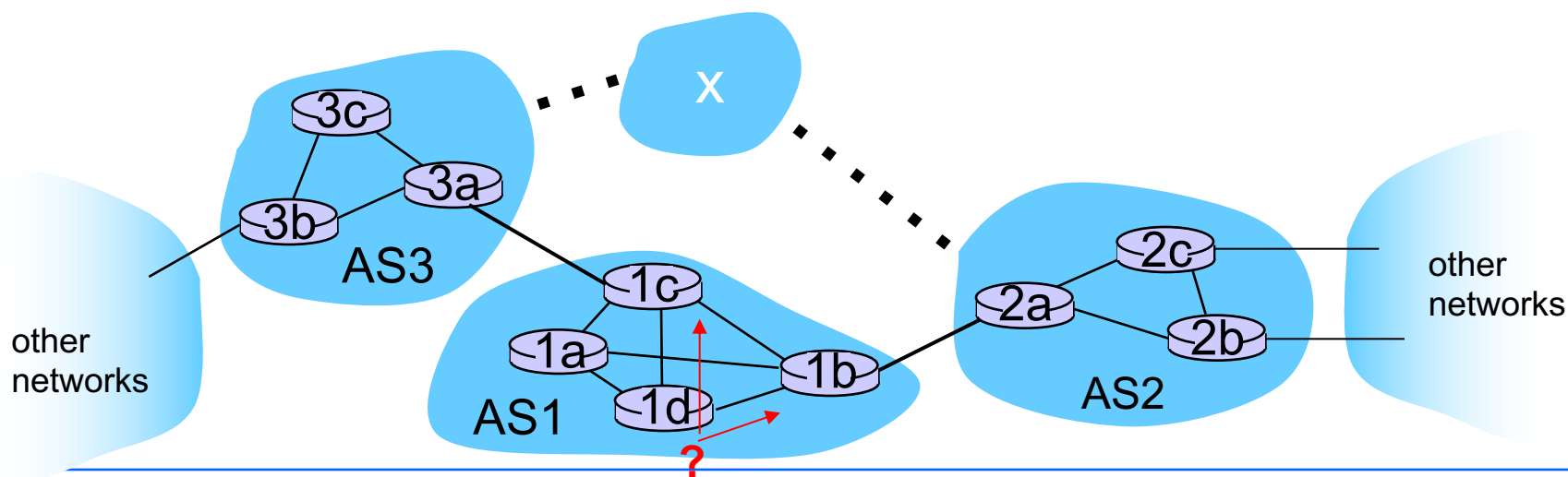
# 示例：为router1d设置转发表

- 假设AS1通过域间路由协议学习到通过AS3（经网关1c）可以到达子网x
  - 域间路由协议会将可达信息通告所有域内路由器
- Router1通过域内路由信息确定到达1c的最短路径所使用的本地出口为interface 1
  - 建立路由转发条目  $(x, 1)$

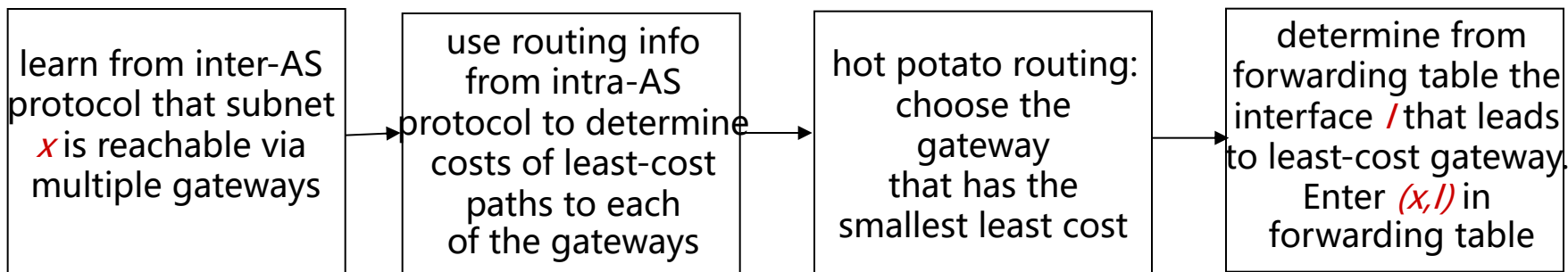


# 示例：多域选择

- 假设AS1学习到通过AS2和AS3均可到达子网x
- 为了设置路由转发表，router 1d必须确定到达目的地x的分组应该转发给哪个网关
  - 这也是域间路由协议的任务



- 假设AS1学习到通过AS2和AS3均可到达子网x
  - 为了设置路由转发表，router 1d必须确定到达目的地x的分组应该转发给哪个网关
- 这也是域间路由协议的任务
- “热土豆”式路由选择：分组在两个最近路由器之间转发



- ❖ 亦称之为 “内部网关协议” (IGP)
- ❖ 常用域内路由协议:
  - RIP:路由信息协议
  - OSPF:首先打开最短路径
  - IGRP:内部网关路由协议

5

## 5.1 路由算法

链接状态

距离矢量

分层路由

6

## 5.2 因特网路由

RIP

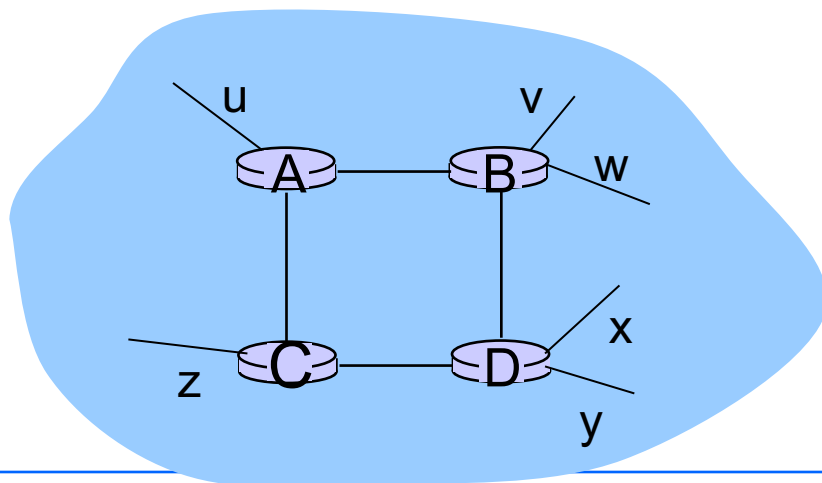
OSPF

BGP

7

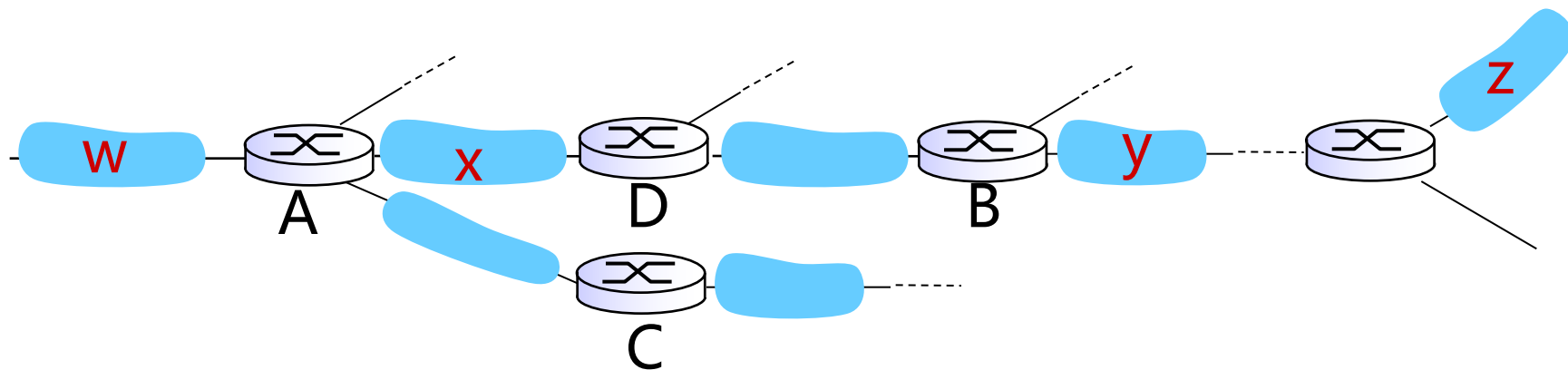
## 5.3 ICMP and SNMP

- 与1982年包含于BSD-UNIX发布版中
- 距离矢量算法
  - 距离度量值：跳数（最大15跳），每个链路的开销均为1
  - 邻居之间每30s通过响应报文（亦称之为“通告”）交换一次距离向量
  - 每次通告：至多25条到达目的子网的列表



from router A to destination *subnets*:

<u>subnet</u>	<u>hops</u>
u	1
v	2
w	2
x	3
y	3
z	2



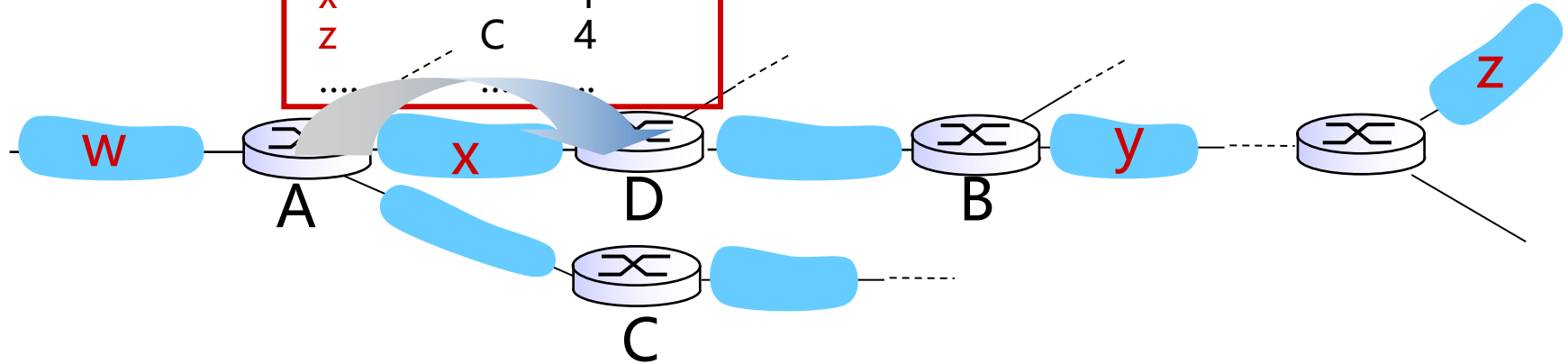
routing table in router D

destination subnet	next router	# hops to dest
w	A	2
y	B	2
z	B	7
x	--	1
....	....	....

# RIP:示例

A-to-D advertisement

dest	next	hops
W	-	1
X	-	1
Z	C	4
...	...	...



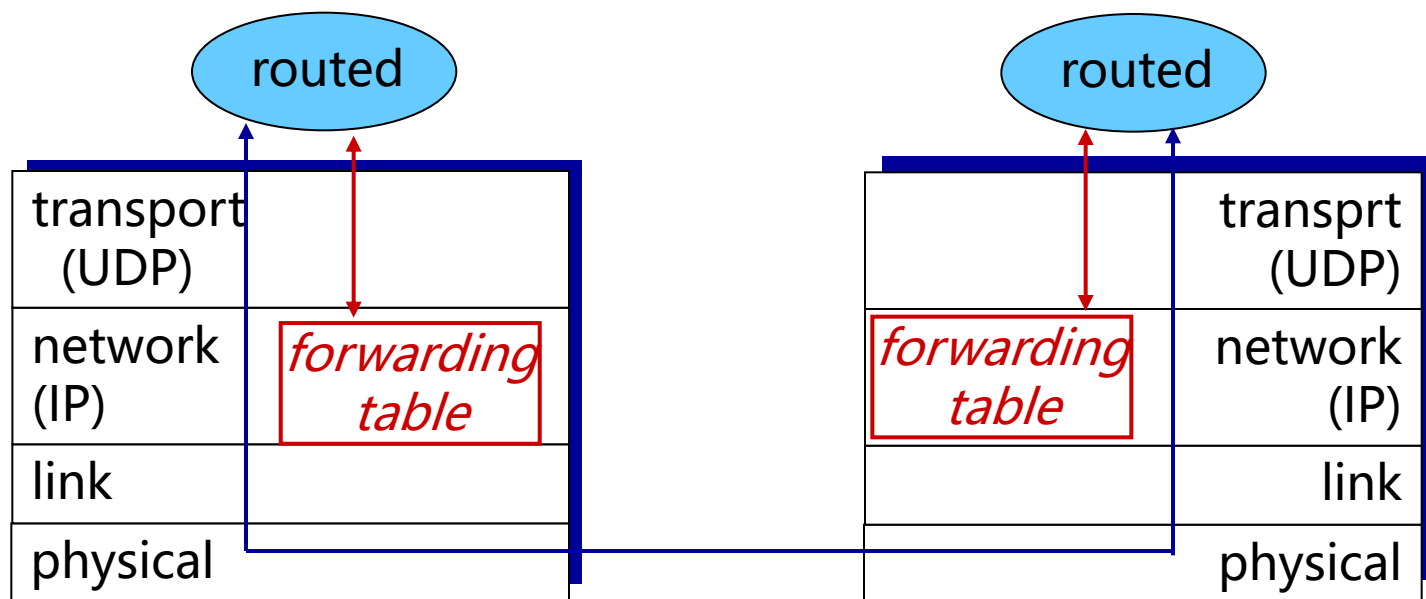
routing table in router D

destination subnet	next router	# hops to dest
W	A	2
y	B	2
Z	<del>B</del> → A	<del>7</del> → 5
X	--	1
....	....	....



- 如果180s没有收到某邻居的通过--》该邻居/链路已失效
  - 所有通过该邻居的路由均失效
  - 通告其他邻居路由器
  - 邻居接收并转发新的通告给自己的邻居（如果路由表改变）
  - 链路失效的信息快速的传播到整个网络中
  - 毒性逆转，阻止路由环路，最大跳数16

- ❖ RIP路由表由应用层进程 (route-d) 管理
- ❖ 通过UDP协议周期性的向邻居发送通告报文



5

## 5.1 路由算法

链接状态

距离矢量

分层路由

6

## 5.2 因特网路由

RIP

OSPF

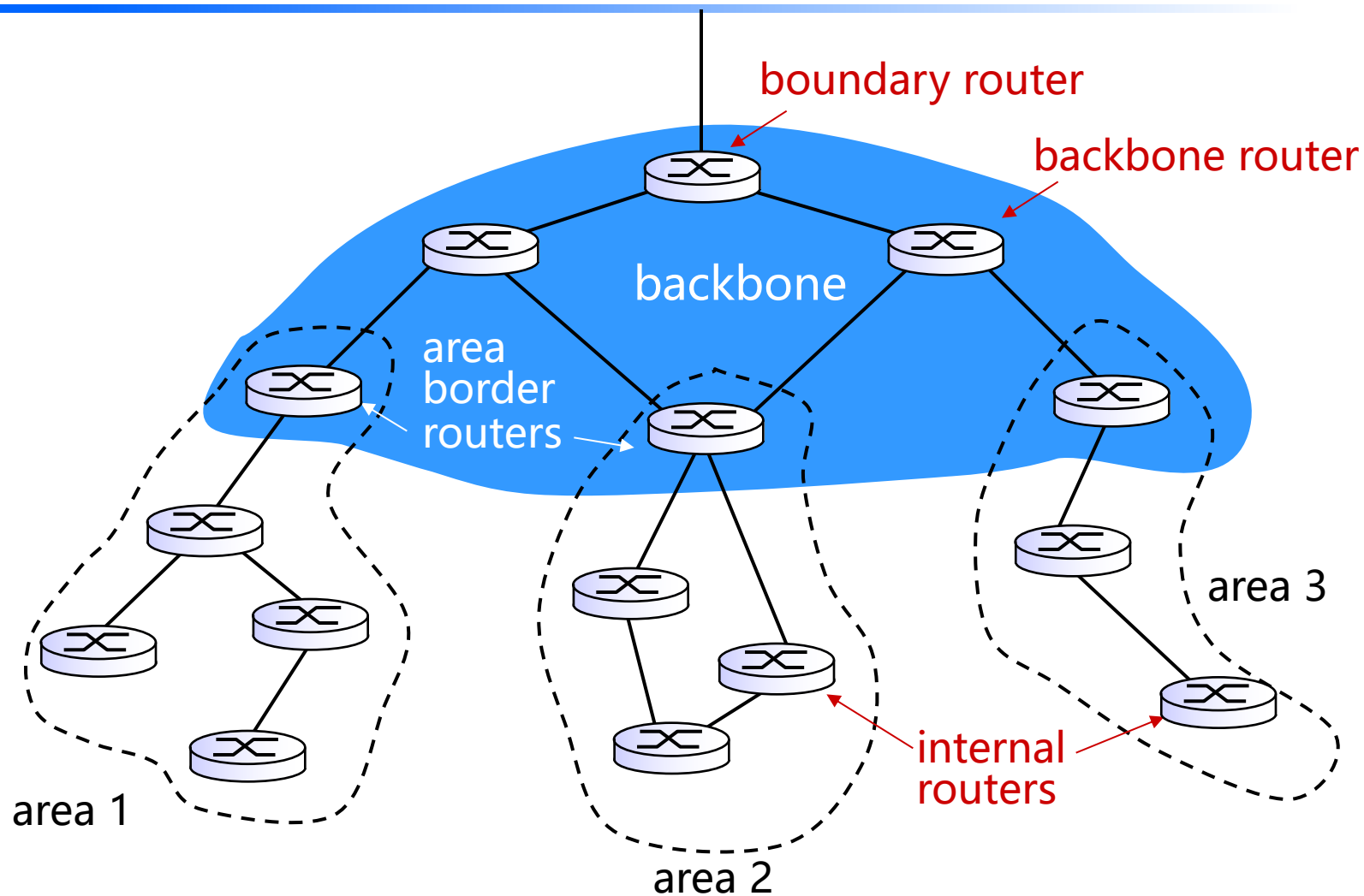
BGP

7

## 5.3 ICMP and SNMP

- “open”：开放标准
- 采用链路状态算法
  - 洪泛链路状态信息
  - 每个节点保存完整的网络拓扑结构
  - 采用Dijkstra算法计算路由
- OSPF向每个邻居发送链路状态信息的通告
- 通告洪泛到整个AS
  - OSPF报文直接使用IP协议，而非TCP或UDP
- IS-IS路由协议：与OSPF类似

- **安全性**：能够鉴别所有的OSPF报文，防止恶意入侵
- 允许多条相同开销的路径
- 同时支持单播和多播路由选择（知道即可，多播路由不做要求）
- 支持在一个较大的自治域内的层次结构



- **两层结构**：本地区域，主干区域
  - 链路状态信息的通告仅在本区域内
  - 每个节点拥有本区域的详细拓扑结构；仅知道直接到达其他区域的（最短）路径
- **区域边界路由器**：汇总本区域网络，并通告给其他区域边界路由器
- **主干区域路由器**：仅在主干区域运行OSPF路由选择
- **边界路由器**：连接其他AS

5

## 5.1 路由算法

链接状态

距离矢量

分层路由

6

## 5.2 因特网路由

RIP

OSPF

BGP

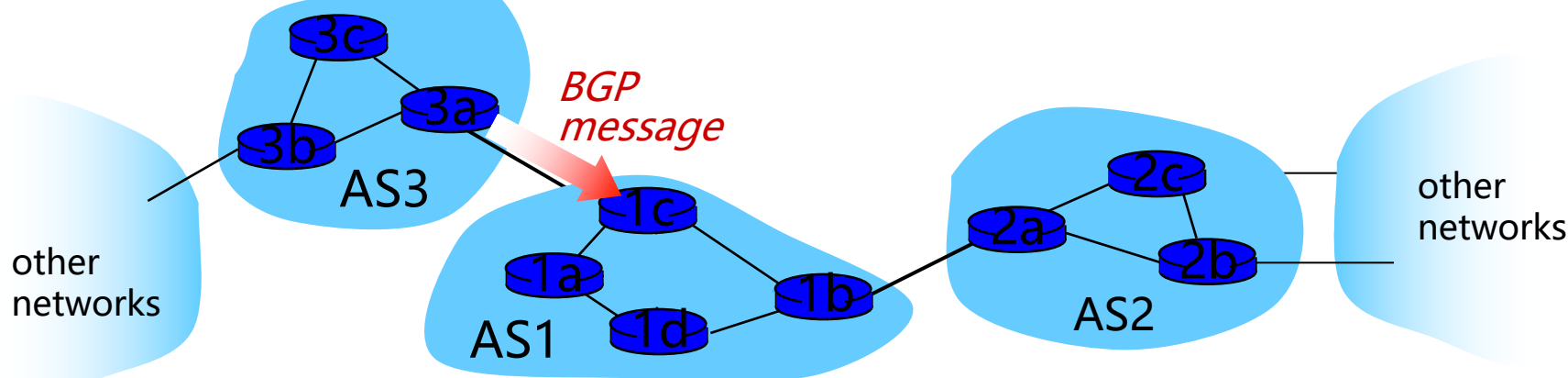
7

## 5.3 ICMP and SNMP

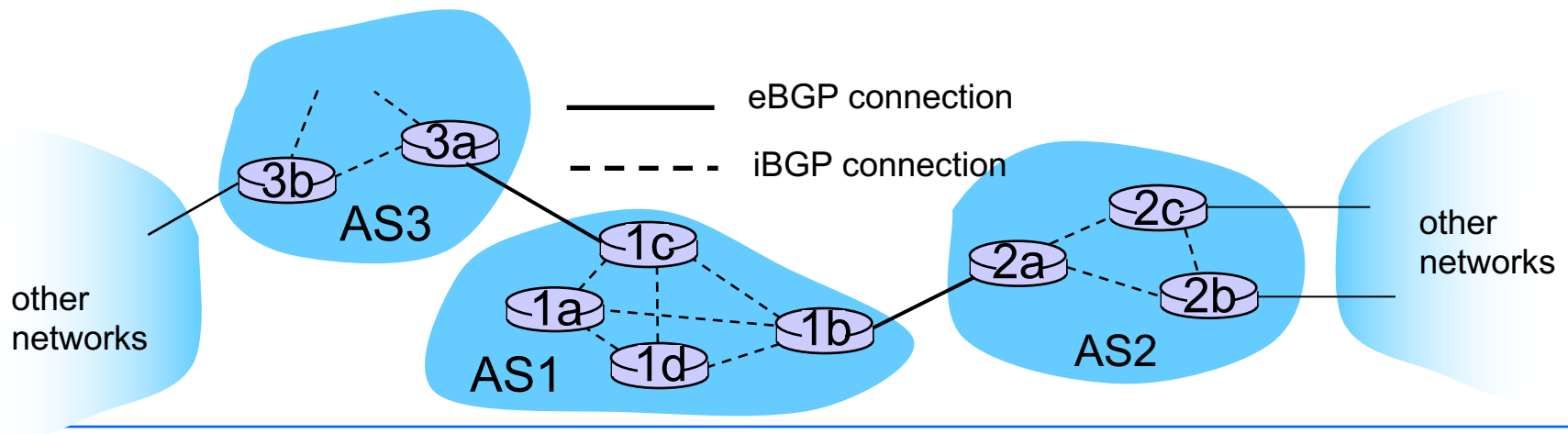


- **BGP（边界网关协议）**：域间路由协议事实上的标准
  - 将Internet粘连在一起
- BGP为每个AS提供了进行以下工作的手段：
  - **外部BGP**：从相邻AS获得子网可达性信息
  - **内部BGP**：向本AS内部的所有路由器传播这些可达性信息
  - 基于可达性信息和AS策略，判定到达子网的最优路由
- 允许每个子网向Internet的其余网络通告自己的存在

- ❖ **BGP会话**：两台BGP路由器（对对等方）交换BGP报文信息：
  - 通告到达不同目的网络前缀的**路径**（子网聚合）
  - 基于半永久性的TCP连接交换报文
- ❖ 当AS3向AS1通告一个前缀：
  - AS3**允诺**它将转发到达该前缀的数据报
  - AS3在通告中可以聚合前缀



- ❖ 3a与1c之间使用eBGP会话，AS3向AS1发送前缀可达信息。
  - 接着，1c使用iBGP向AS1中的所有路由器发布这一前缀信息
  - 随后，1b通过1b-到-2a的eBGP会话向AS2通过这一新的可达性信息
- ❖ 当路由器学习到这一新的前缀信息，就在自己的路由转发表里新建一条到达该前缀的路由条目。



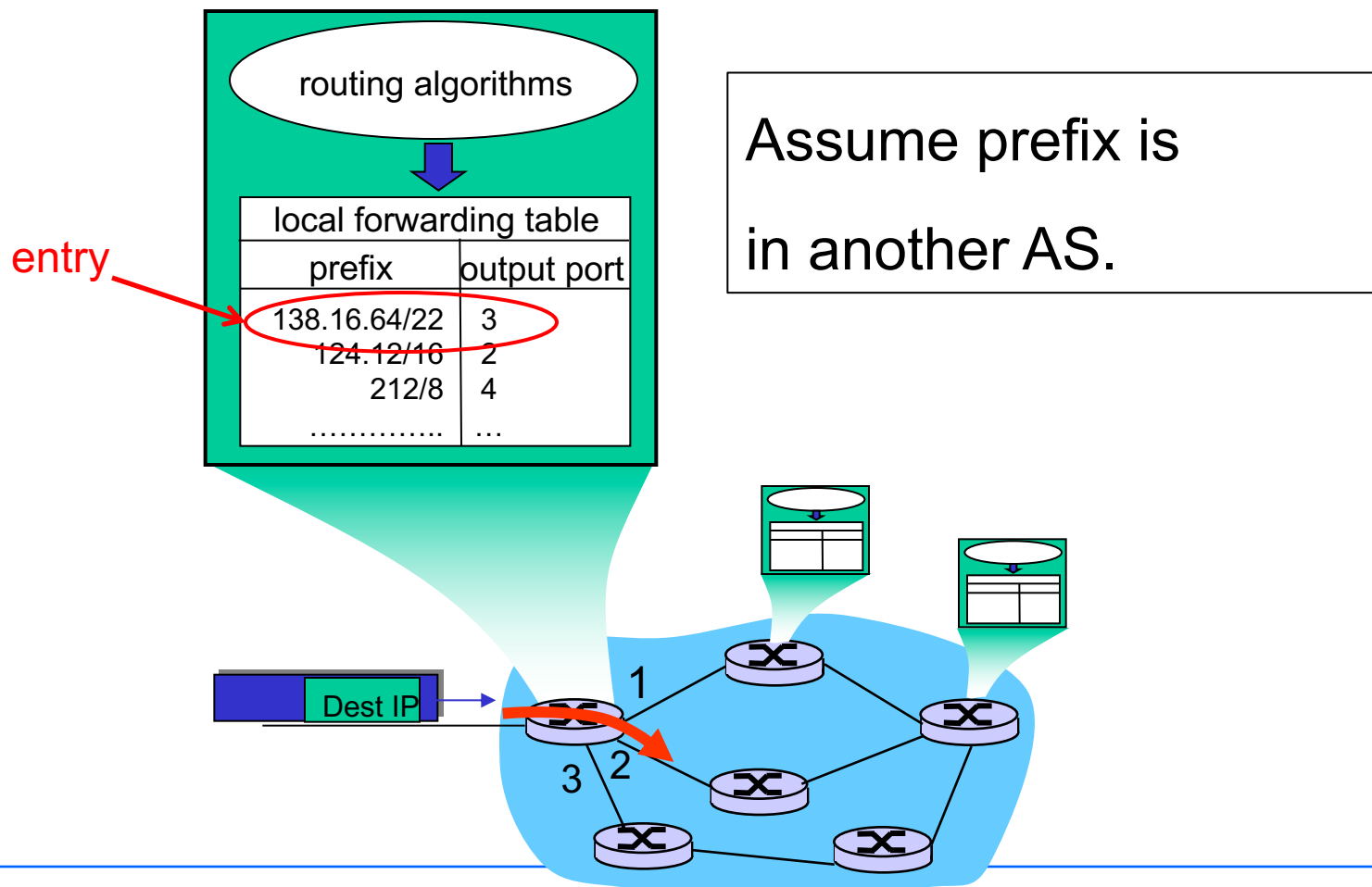
# 路径属性与BGP路由

- 通告前缀的信息中还包含了BGP属性
  - 前缀+属性=路由
- 两个重要属性:
  - **AS-PATH**: 该属性包含了前缀的通告已经通过了那些AS
  - **NEXT-HOP**: 指示了哪个域内路由器是下一跳AS的出口
- 当网关路由器接收到路由通告时, 使用输入策略来决定是否接受或拒绝该路由
  - 例如, 永远不要通过 AS x 路由
  - **基于策略**的路由

- ❖ 路由器可能学习到到达目的AS的多个路由信息，选择路由基于：
  - 1、有策略所决定的本地偏好值
  - 2、最短AS-PATH
  - 3、最靠近NEXT-HOP路由器
  - 4、附加规则（如BGP标识符）

- 基于TCP连接在对等方之间交换BGP报文
- BGP报文:
  - OPEN: 开发TCP连接
  - UPDATE: 通告新的路径
  - KEEPALIVE: 保持连接
  - NOTIFICATION: 报告之前报文的错误; 关闭连接

# 条目如何进入转发表?



# 总而言之：条目如何进入路由器的转发表？

- 答案很复杂！
- 将分层路由（第 4.5.3 节）与 BGP（4.6.3）和 OSPF（4.6.2）绑定在一起。
- 提供了BGP的良好概述！



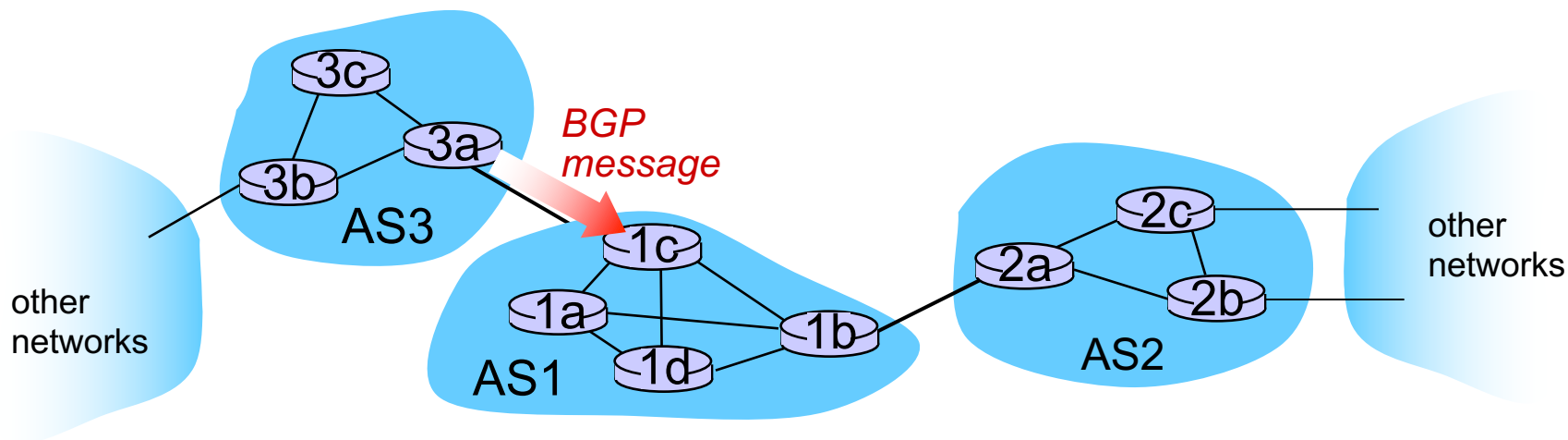
# 条目如何进入转发表?

## 高级概述

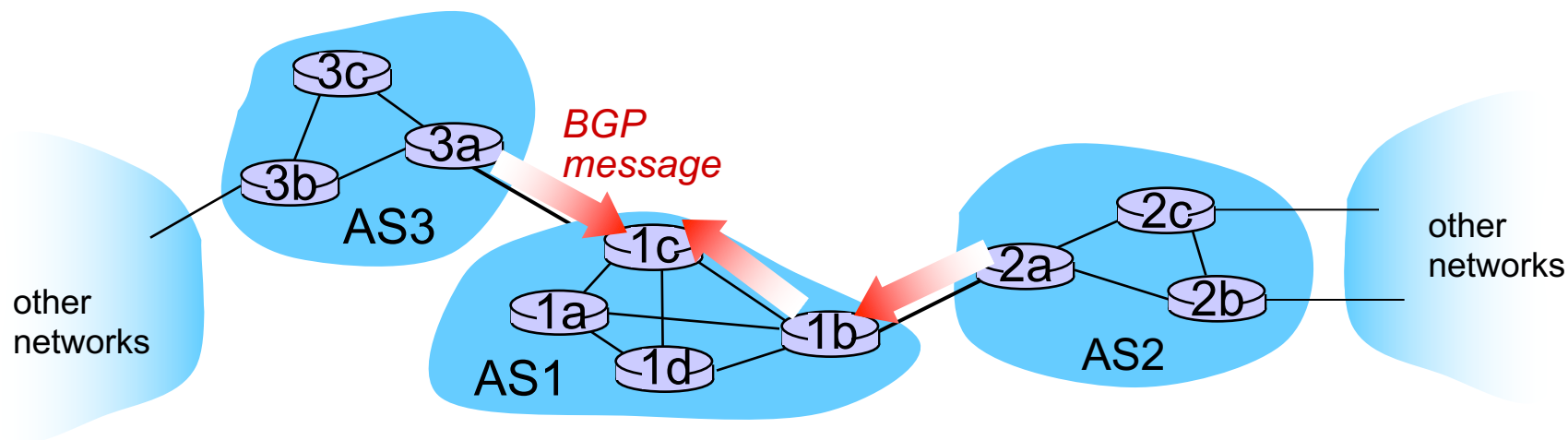
路由器开始知道前缀

路由器确定前缀的输出端口

路由器在转发表中输入前缀端口



- ❖ BGP 消息包含 “路由”
- ❖ “route” 是一个前缀和属性：AS-PATH, NEXT-HOP,...
- ❖ 示例：路由：
  - ❖ 前缀：138.16.64/22 ; 路径： AS3 AS131 ; 下一跃点：  
201.44.13.125



- ❖ 路由器可能会收到同一前缀的多个路由
- ❖ 必须选择一条路线

- ❖ 路由器根据最短的 AS-PATH 选择路由

- ❖ 例:

- ❖ AS2 AS17 to 138.16.64/22

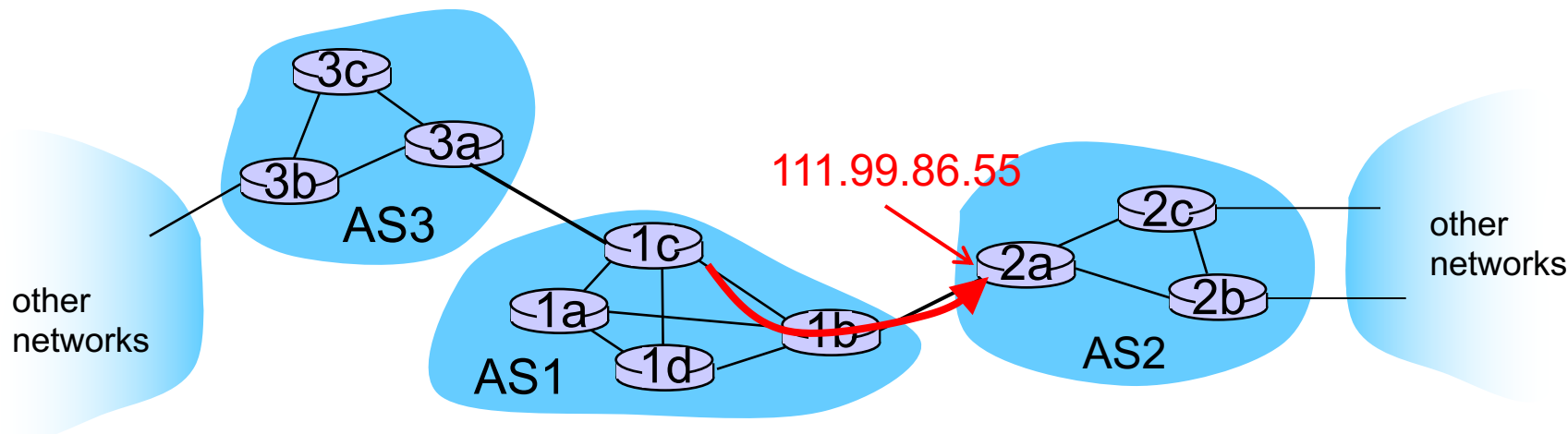
select

- ❖ AS3 AS131 AS201 to 138.16.64/22

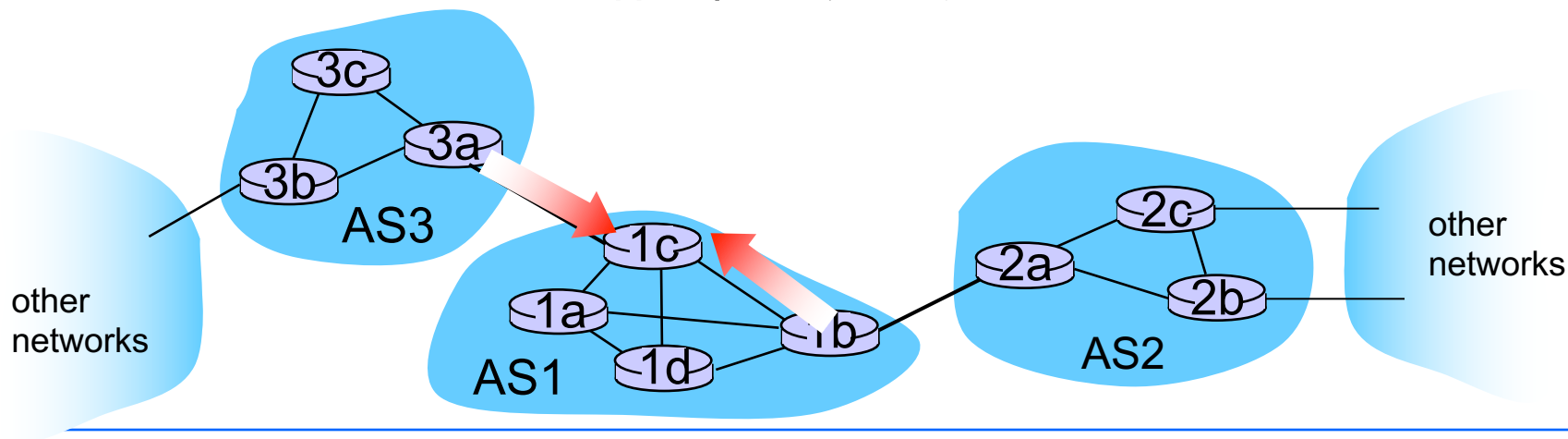
- ❖ 如果有平局怎么办？我们会回到那个！

# 查找到BGP路由的最佳内部路由

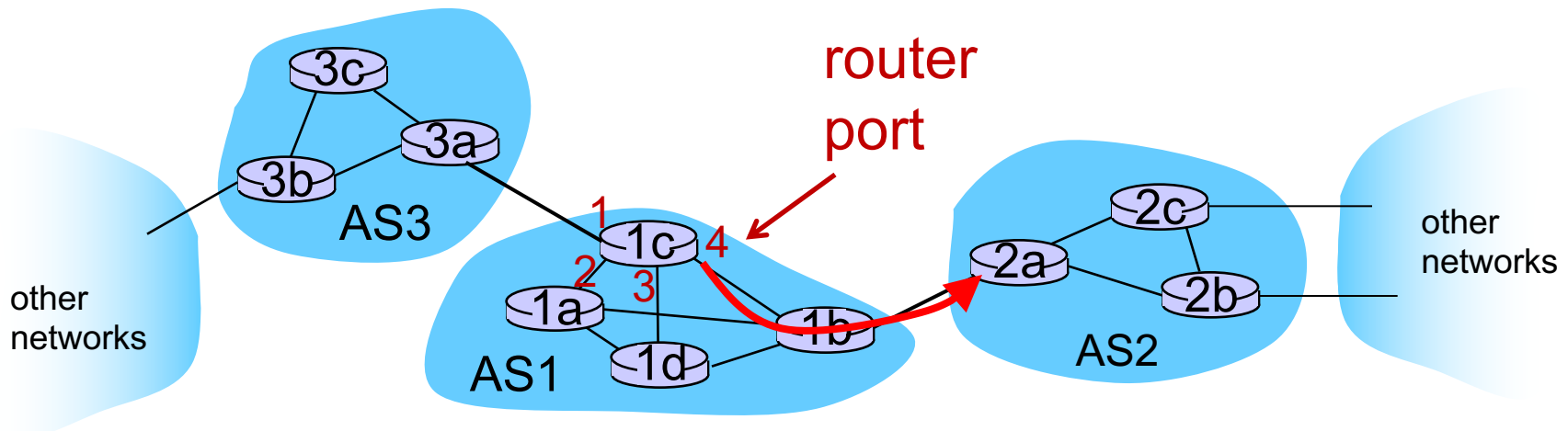
- 使用所选路由的下一跳属性
  - 路由的下一跳属性是作为AS路径开始的路由器接口的IP地址。
- 示例：AS-PATH:AS2 AS17; 下一站:111.99.86.55
- 路由器使用OSPF找到从1c到111.99.86.55的最短路径



- ❖ 假设有两条或多条最佳内部路由。
- ❖ 然后选择下一跳最近的路由
  - 使用OSPF来确定哪个网关最近
  - 问:从1c, 选择AS3 AS131还是AS2 AS17?
  - a:走AS3 AS131路线, 因为它更近



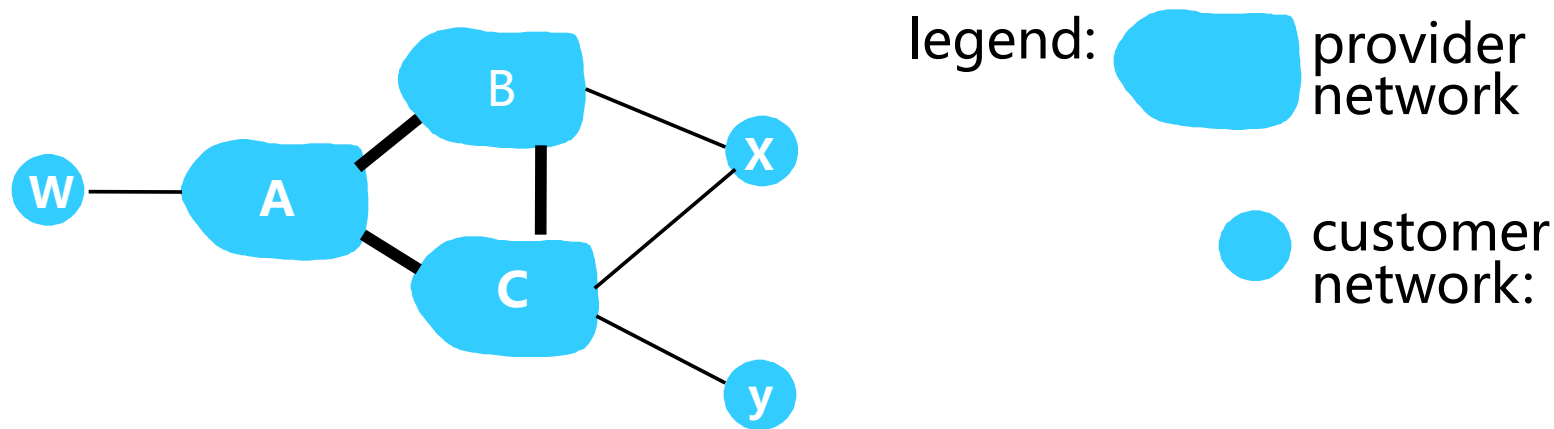
- ❖ 标识OSPF最短路径上的端口
- ❖ 将前缀端口条目添加到它的转发表：  
(138.16.64/22, 端口4)



## 摘要

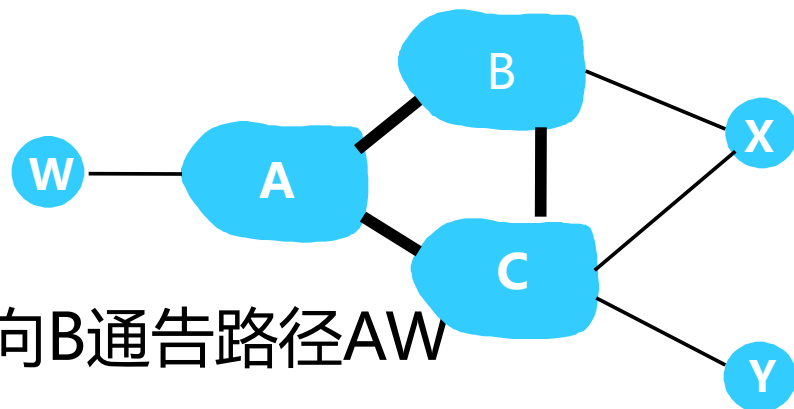
1. 路由器获知前缀
    - 经由来自其它路由器的BGP路由通告
  2. 确定前缀的路由器输出端口
    - 使用BGP路由选择找到最佳AS间路由
    - 使用OSPF找到通向最佳AS间路由的最佳AS内路由
    - 路由器识别最佳路由的路由器端口
  3. 在转发表中输入前缀端口条目
-







- ❖ A、B、C是**提供商网络**
- ❖ x、w、y是(提供商网络的)客户端
- ❖ x是**双重连接**:连接到两个网络
  - X不想从B经由X路由到C
  - .....因此X不会向B通告到C的路由

## BGP路由策略(2)



legend:  provider network  
 customer network:

- ❖ a向B通告路径AW
- ❖ b向X通告路径BAW
- ❖ B应该通告从BAW到C的路径吗?
  - 不会吧! 因为W和C都不是B的客户, 所以B没有得到路由CBAW的“收入”
  - b想迫使C通过A路由到w
  - b希望只向自己客户的发送路由!

# 为何有不同的域间和域内路由协议

## 策略:

- 域间：管理员希望可以控制路由流量，以及谁可以通过自己实现路由
- 域内：单一管理，无需策略抉择

## 规模:

- 层次结构路由减少了路由转发表的规模，减少了更新流量.

## 性能:

- 域内：着重关注性能
- 域间：策略优先于性能

5

## 5.1 路由算法

链接状态

距离矢量

分层路由

6

## 5.2 因特网路由

RIP

OSPF

BGP

7

## 5.3 ICMP and SNMP

# ICMP: internet报文控制协议

- 用于主机与路由器之间实现网络一级信息的通信
  - 错误报告：主机不可达，网络不可达，端口不可达，协议不可达
  - echo请求/应答（用于ping命令）
- 位于网络层IP协议之上：
  - ICMP报文承载于IP数据报中
- ICMP 报文:类型、代码加上导致错误的 IP 数据报的前 8 个字节

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

- ❖ 源端发送一系列的UDP报文段至目的地址

- 第一组, TTL=1
- 第二组, TTL= 2, ...
- 最后一组, 一个不可达的端口号

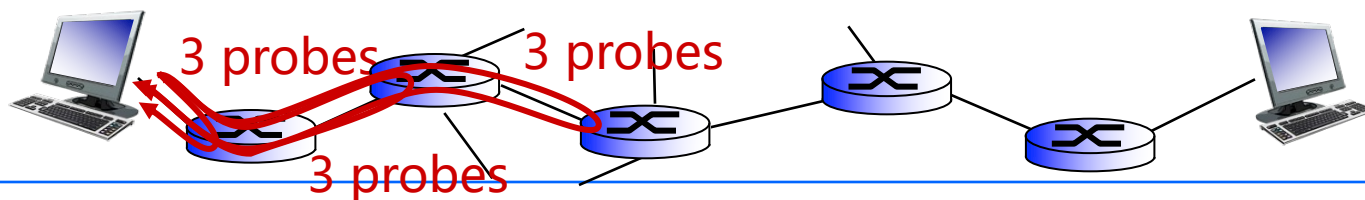
- ❖ 当第n组数据报到达第n个路由器时:

- 路由器丢弃数据报
- 然后向源端发送 (type 11, code 0) 的 ICMP报文
- ICMP报文包含了路由器的名称和IP地址

- ❖ 当ICMP返回源端时, 记录RTT时间

## 迭代终止准则:

- ❖ UDP报文段最终到达目的主机
- ❖ 目的主机返回端口不可达的ICMP报文 (type 3, code 3)
- ❖ 源端终止traceroute程序



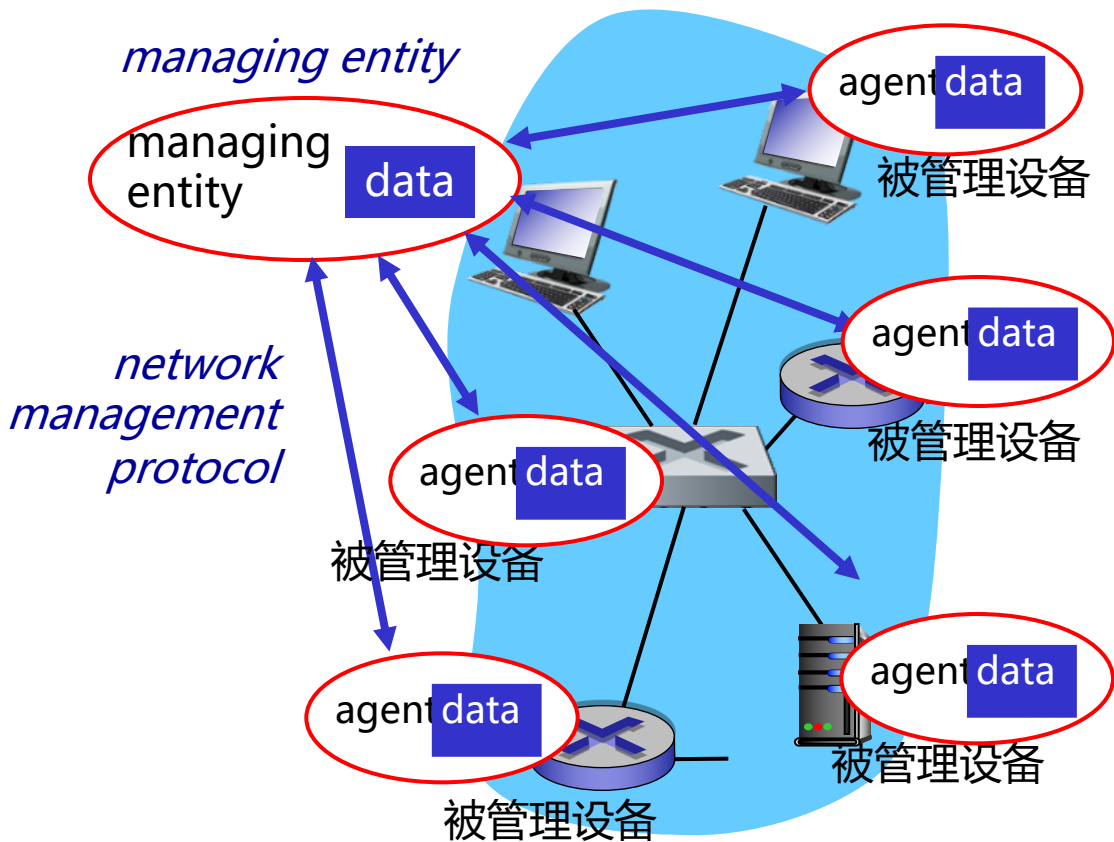
- ❖ 自治系统(又名“网络”):数以千计的交互硬件/软件组件
- ❖ 其他需要监测、控制的复杂系统:
  - 喷气式飞机
  - 核电站
  - 其他?



“网络管理包括部署、集成以及硬件、软件和人力的协调监控、测试、轮询、配置、分析、评估，并控制网络和元件资源以满足实时、运营性能和服务质量以合理的成本满足需求。

”

定义:

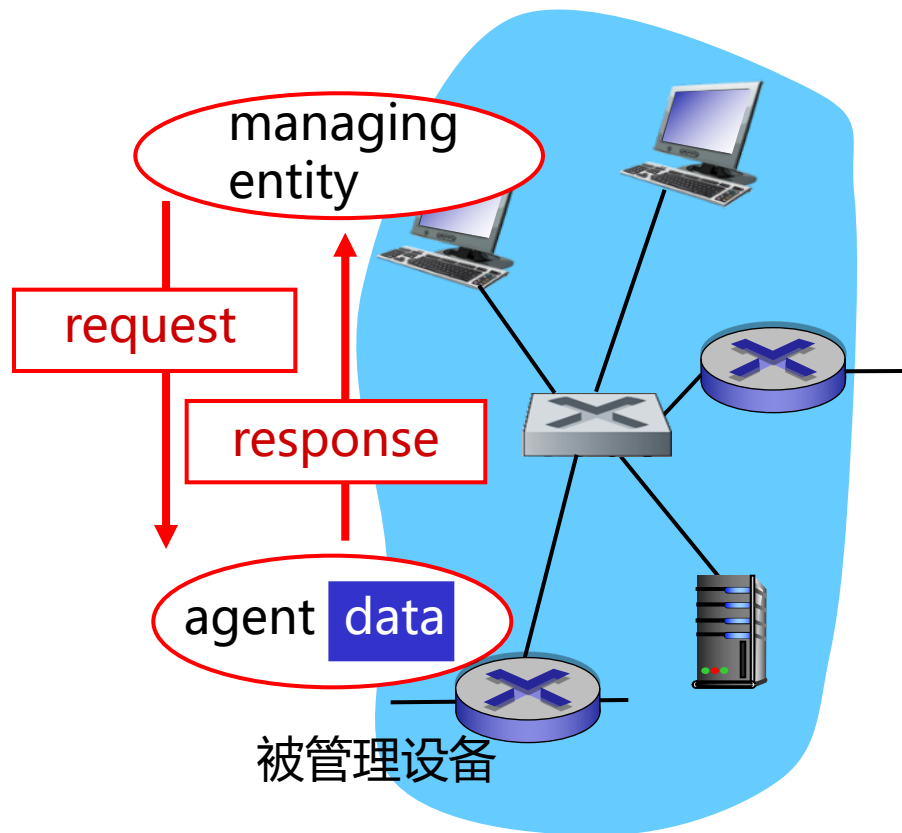


被管理设备包含被管理对象，其数据被收集到管理信息库(MIB)中

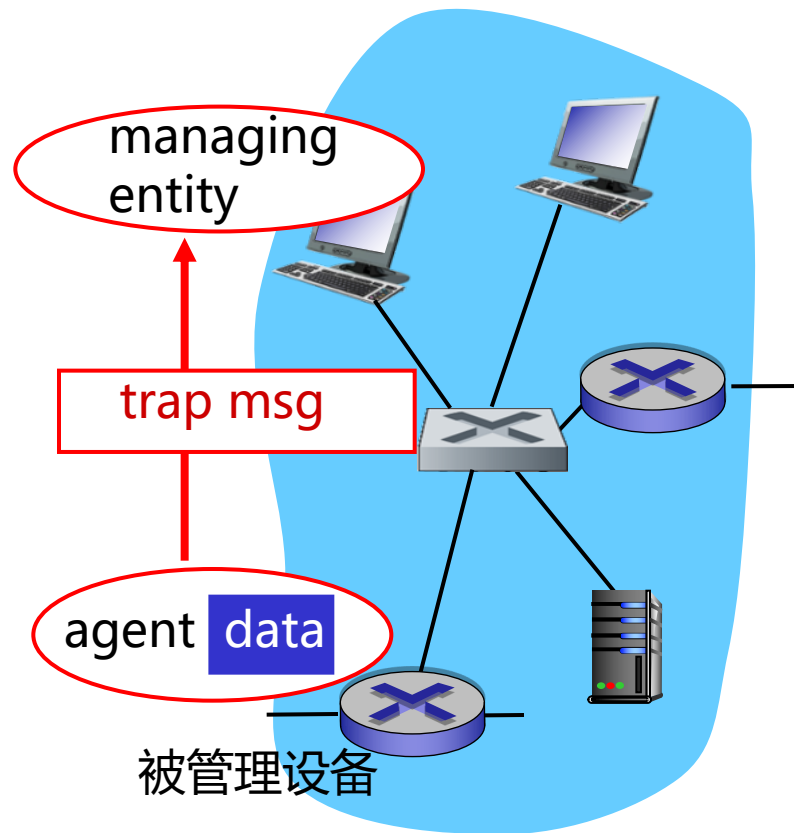


- ❖ 管理信息库(MIB):
  - 网络管理数据的分布式信息存储
- ❖ 管理信息的结构(SMI):
  - MIB对象的数据定义语言
- ❖ SNMP协议
  - 传达管理器<->被管理对象信息, 命令
- ❖ 安全、管理功能
  - SNMPv3的主要新增功能

传达MIB信息和命令的两种方式:

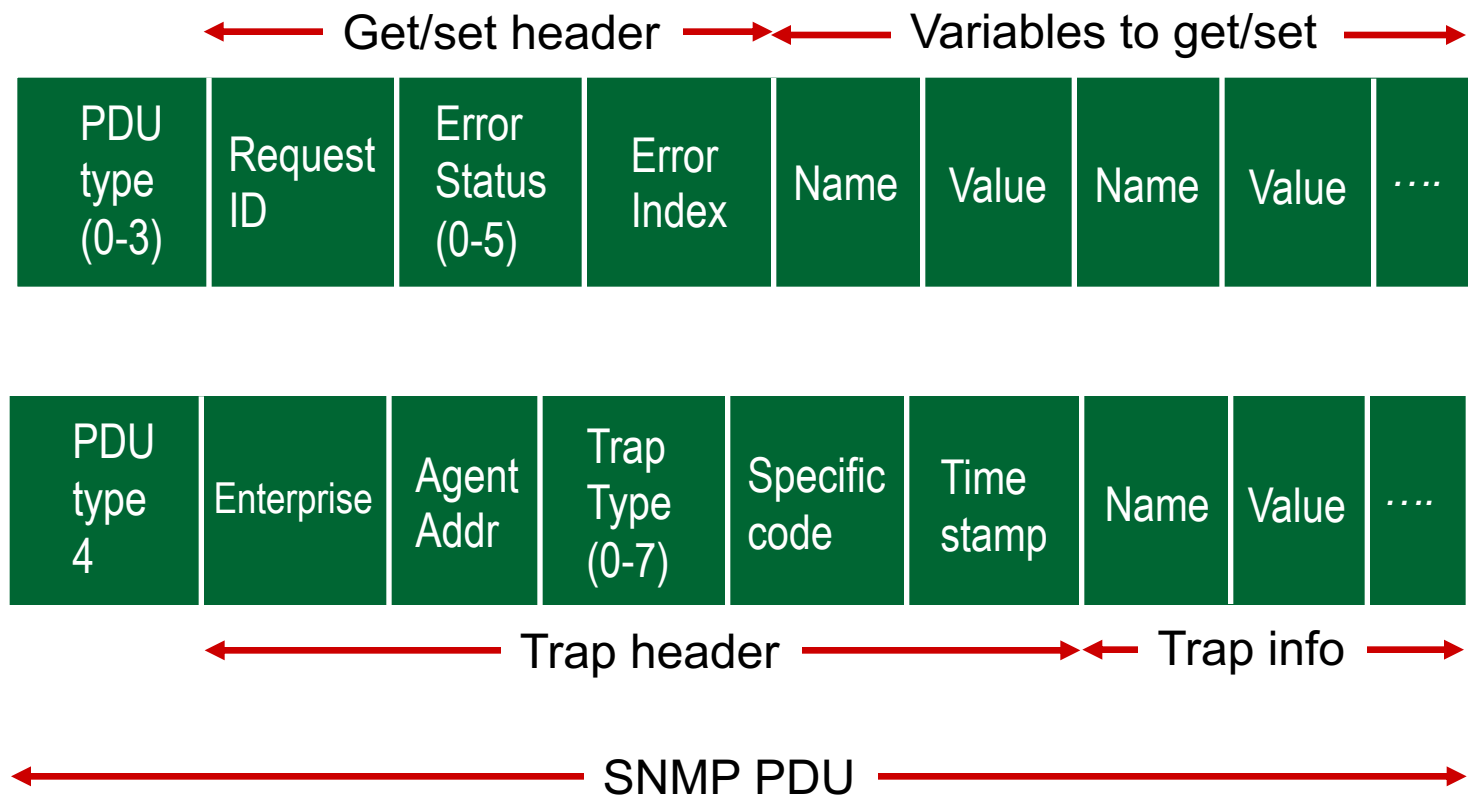


请求/响应模式



陷阱模式

消息类型	功能
获取请求 获取下一个请求 获取批量请求	经理对代理: “给我数据” (数据实例、列表中下一个数据、数据块)
信息请求	经理对经理:这是MIB值
SetRequest	经理到代理:设置MIB值
反应	代理对经理:价值, 响应 请求
圈套	代理对经理:通知经理 特殊事件



关于网络管理的更多信息:请参阅早期版本的文本!

- ❖ 加密:DES-加密SNMP消息
- ❖ 身份验证:计算, 发送MIC(m, k):通过消息(m)计算哈希(MIC), 秘密共享密钥(k)
- ❖ 防止回放:使用随机数
- ❖ 基于视图的访问控制:
  - SNMP实体维护各种用户的访问权限、策略的数据库
  - 数据库本身可作为管理对象访问!

## 第五章：小结

- ❖ 路由算法：链路状态 vs 距离向量
- ❖ 路由算法的优缺点：
- ❖ 路由协议：RIP OSPF BGP
- ❖ AS 与 层次路由；
- ❖ ICMP & SNMP

### 作业：

- R4
- P3、P4、P14、P15