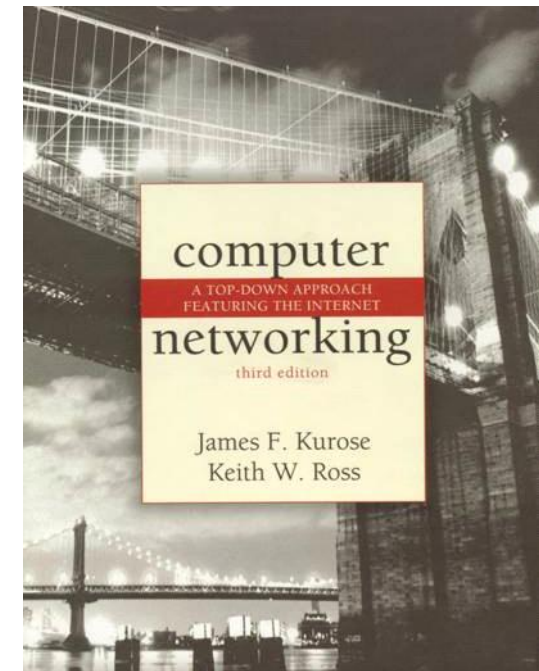


第五章 网络层

——控制平面



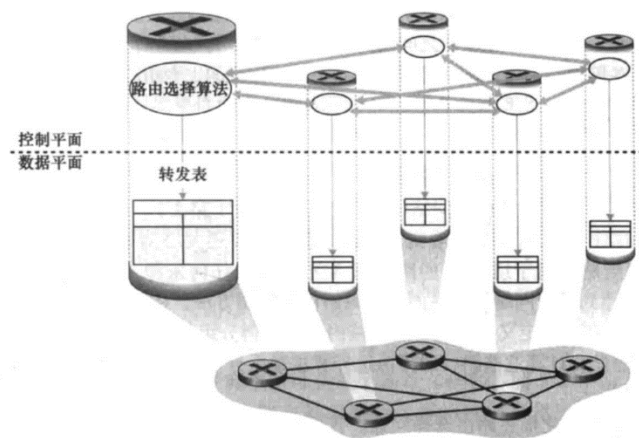
第五章：网络层——控制平面

- 5.1 概述
- 5.2 路由选择算法
 - 链路状态选路算法
 - 距离向量算法
- 5.3 因特网中的自治系统内部路由选择
- 5.4 ISP之间的路由选择： BGP
- 5.5 ICMP协议

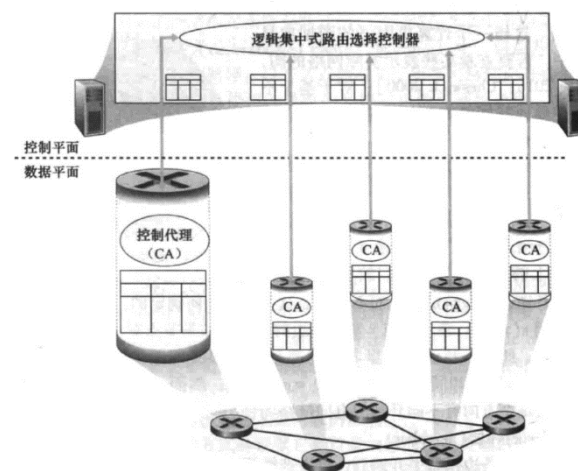
5.1 概述

- 控制平面：控制源主机到目的主机端到端路径上的路由器如何转发数据包

- 两种控制方式
- 每路由器控制：路由器包含转发和路由选择功能

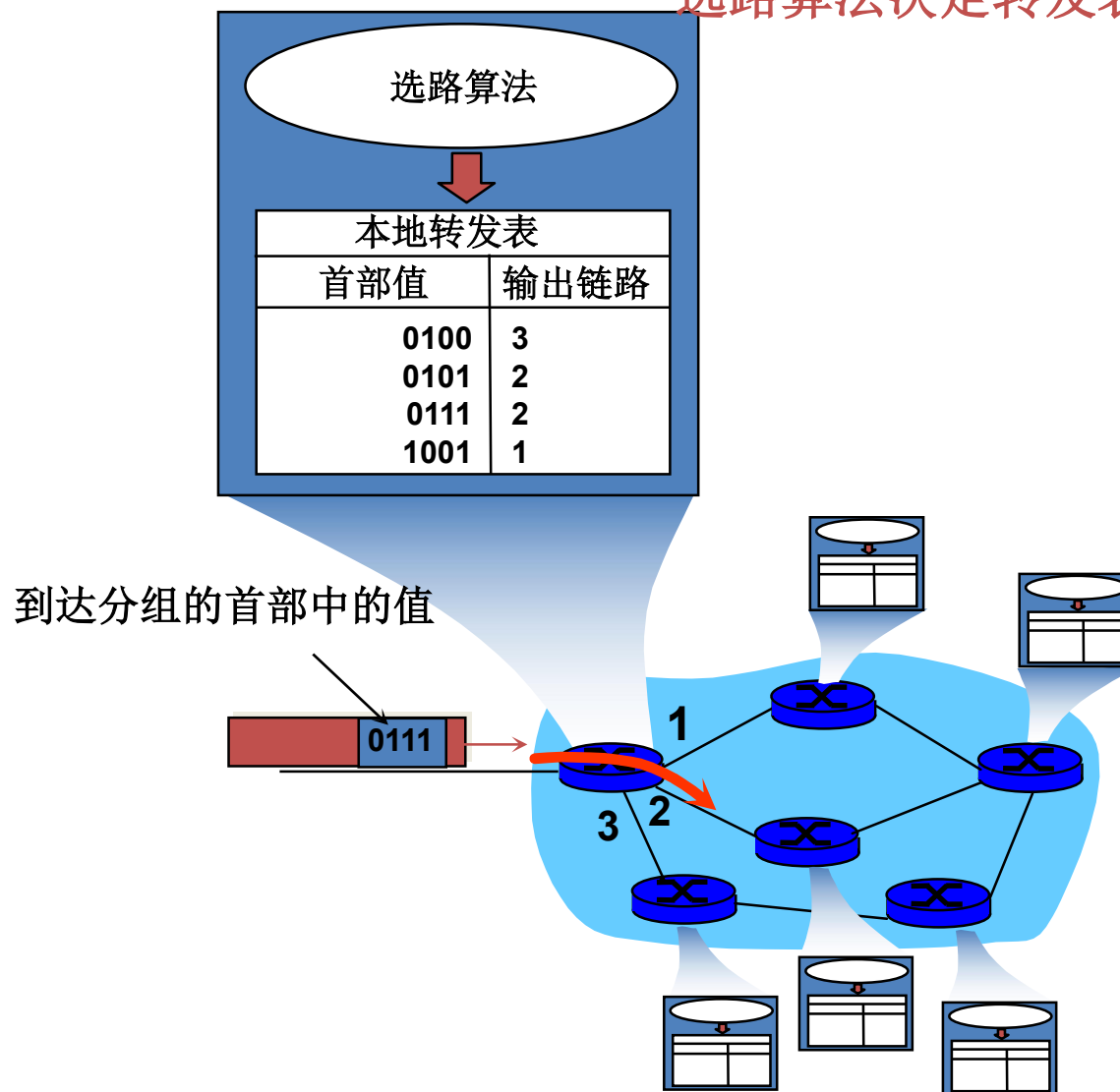


- 逻辑集中式控制：路由由控制器决定



转发和选路

选路算法决定转发表中的值



第五章：网络层——控制平面

- 5.1 概述
- 5.2 路由选择算法
 - 链路状态选路算法
 - 距离向量算法
- 5.3 因特网中的自治系统内部路由选择
- 5.4 ISP之间的路由选择： BGP
- 5.5 ICMP协议

图形抽象

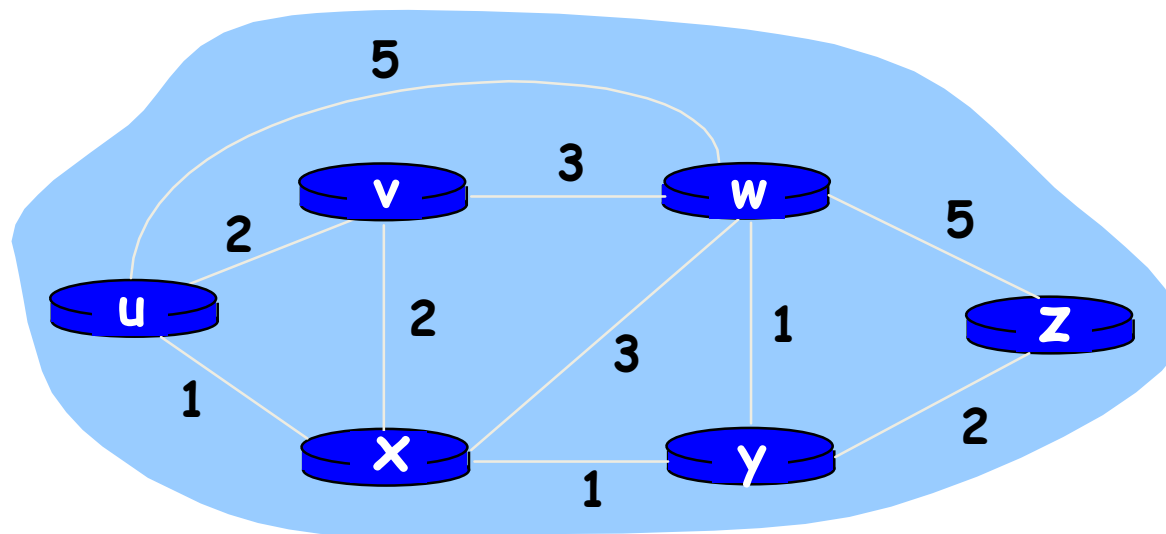


图: $G = (N, E)$

N = 所有路由器的集合 = $\{ u, v, w, x, y, z \}$

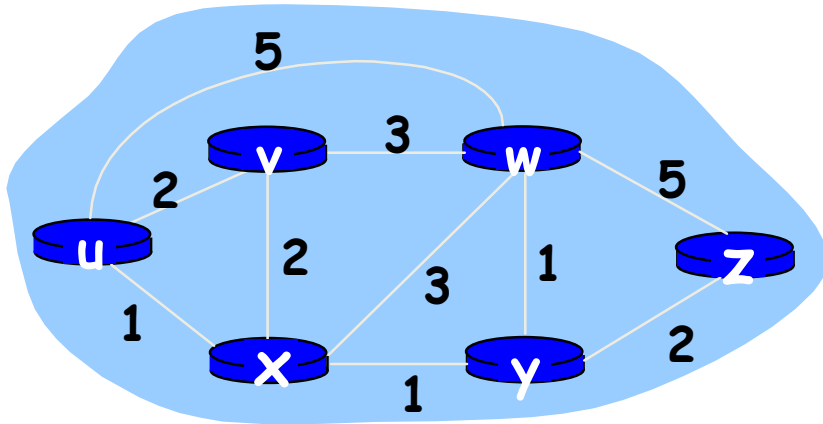
E = 所有链路的集合 = $\{ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) \}$
 (u,v) 表示 u, v 互为邻居, 可直接通信

两个节点之间的路径由多条链路连接而成。

如 u, y 之间的路径可以是: $(u,x), (x,y)$

也可以是: $(u,v), (v,x), (x,y)$

图形抽象: 开销



• $c(x, x')$ = 链路 (x, x') 的开销

例如, $c(w, z) = 5$

• 链路开销一般是1, 或者与带宽成反比, 或与拥塞程度成反比

路径 $(x_1, x_2, x_3, \dots, x_p)$ 的开销 = $c(x_1, x_2) + c(x_2, x_3) + \dots + c(x_{p-1}, x_p)$

Question: U和Z之间的最小开销是多少 ?

选路算法: 发现最小开销路径的算法

路由选择算法分类

全局的还是分散的信息？

全局的：

- 所有的路由器都有完整的网络拓扑结构、链路开销信息
- “链路状态” 算法

分散的：

- 路由器知道物理相连的邻居，到邻居的链路开销。
- 邻居间反复进行计算处理，交换信息
- “距离矢量” 算法

静态的还是动态的？

静态的：

- 路由变化很慢，通常由人工进行设置

动态的：

- 路由变化很快
 - 周期性更新
 - 直接响应链路开销的变化

第五章：网络层——控制平面

- 5.1 概述
- 5.2 路由选择算法
 - 链路状态选路算法
 - 距离向量算法
- 5.3 因特网中的自治系统内部路由选择
- 5.4 ISP之间的路由选择： BGP
- 5.5 ICMP协议

5.2.1 链路状态选路算法

Dijkstra算法

- 所有节点知道网络拓扑结构和链路开销
 - 通过“链路状态广播”完成
 - 所有节点有同样的信息
- 计算从一个节点(源节点)到其他所有节点的最小开销路径
 - 得到这个节点的路由表
- 重复(迭代): K次迭代以后, 得知到其它K个目的地的最小开销路径

符号:

- $c(x,y)$: 从节点 x 到 y 的链路开销. 如果没有直接相邻, 开销是无穷大
- $D(v)$: 从源到目的节点 v 的路径开销的当前值
- $p(v)$: 沿着从源到节点 v 的路径上 v 的前一个节点 (v 的邻接点)
- N : 已经明确的具有最小开销路径的节点集

Dijkstra's 算法

1 初始化:

2 $N' = \{u\}$

3 对所有的节点 v

4 if v 与 u 相邻

5 then $D(v) = c(u, v)$

6 else $D(v) = \text{无限大}$

7

8 循环

9 找到 w 不属于 N' , 而 $D(w)$ 是最小的

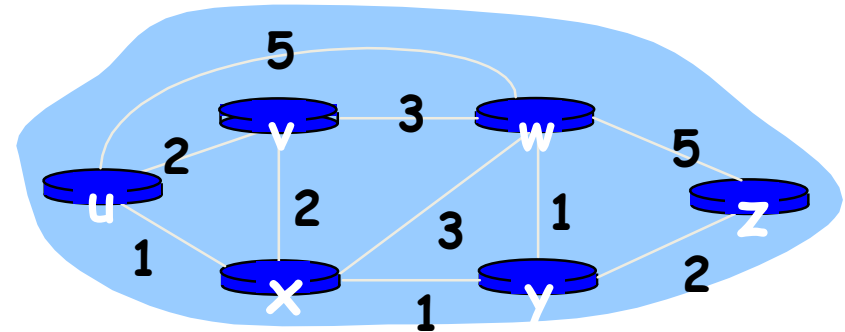
10 将 w 加入 N'

11 更新所有和 w 相邻, 没有在 N' 中的所有节点 v 的 $D(v)$:

12 $D(v) = \min(D(v), D(w) + c(w, v))$

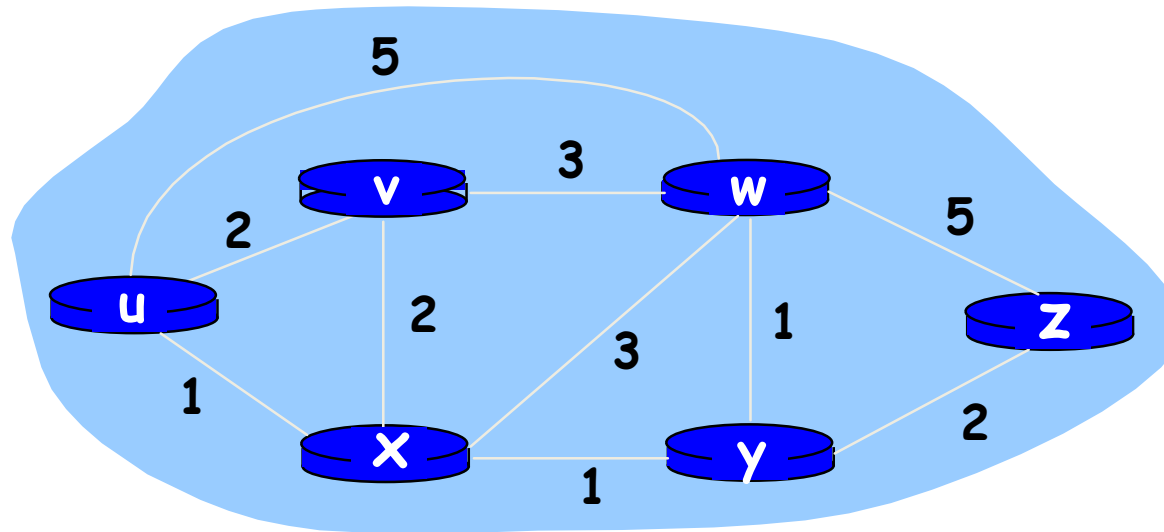
/* 到 v 的新的代价要么不变, 要么这时已经知道的到 w 的最短路径加上 w 到 v 的路径*/

15 直到所有的节点属于 N

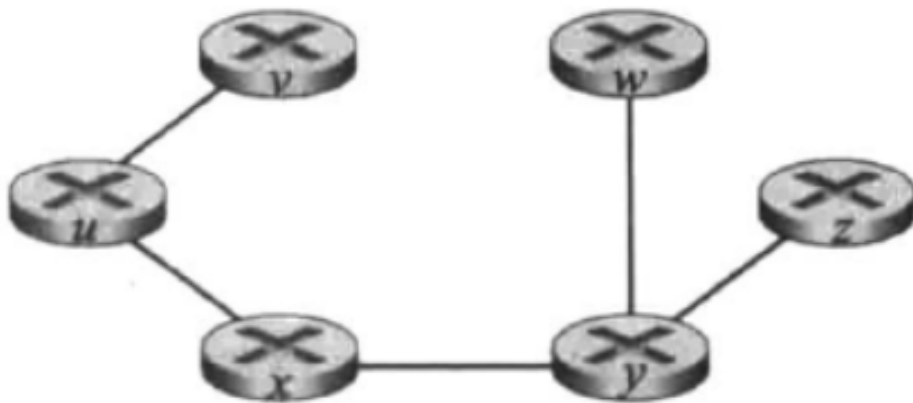


Dijkstra算法: 举例

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



节点U的最短路径和转发表



目的地	链路
v	(u, v)
w	(u, x)
x	(u, x)
y	(u, x)
z	(u, x)

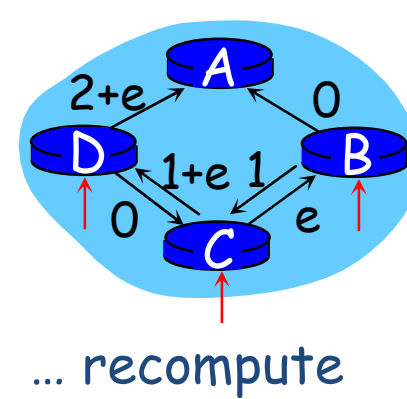
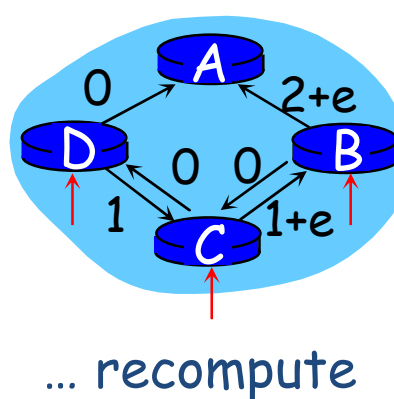
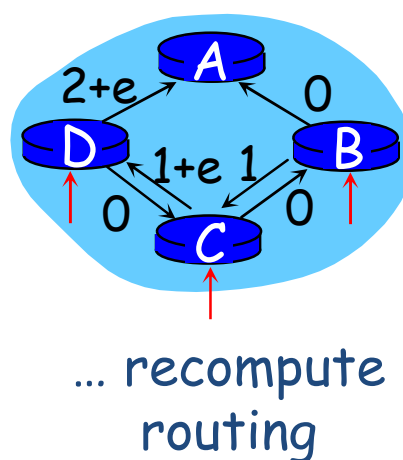
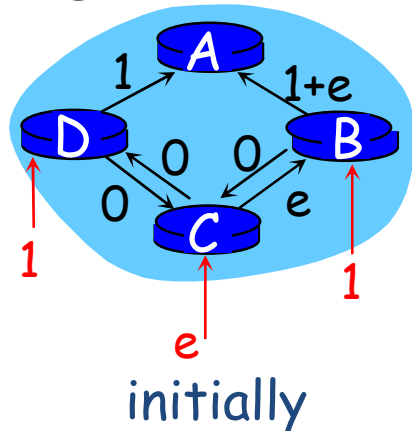
Dijkstra算法, 讨论

算法复杂性: n 节点

- 每次迭代: 需要检查所有不在 N' 中的节点, 找出 w 节点, 具有最低路径消耗
- $n(n+1)/2$ 次比较: 复杂性 $O(n^2)$
- 更有效的执行可能: $O(n \log n)$

可能产生振荡:

- e.g., 链路代价 = 链路承当的流量



第五章：网络层——控制平面

- 5.1 概述
- 5.2 路由选择算法
 - 链路状态选路算法
 - 距离向量算法
- 5.3 因特网中的自治系统内部路由选择
- 5.4 ISP之间的路由选择： BGP
- 5.5 ICMP协议

5.2.2 距离向量选路算法

Bellman-Ford 方程 (动态计算)

定义

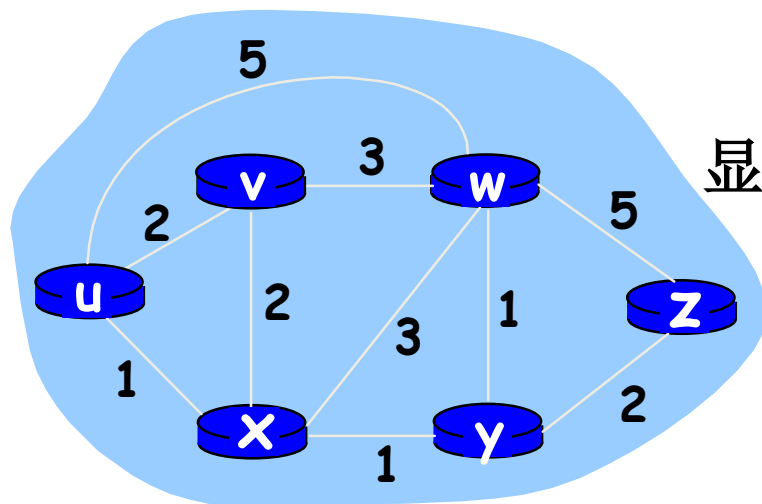
$d_x(y) :=$ 从 x 到 y 的具有最低开销路径的开销值

那么有

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

其中 v 是 x 的相邻节点, \min_v 是对于与 x 相连的所有邻居而言

Bellman-Ford 方程 举例



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

应用B-F 方程, 即:

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

最近的节点是最短路径中的下一跳节点（转发表中有）

距离向量算法

- $D_x(y)$ = 从 x 到 y 的最小开销估值
- 距离向量: $D_x = [D_x(y): y \in N]$
- 节点 x 到直接相连的邻居 v 的开销: $c(x, v)$
- 节点 x 维护自己到其他节点的距离向量

$$D_x = [D_x(y): y \in N]$$

- 节点 x 同时还维护相邻节点的距离向量
 - 对每个邻居 v , x 维护着 $D_v = [D_v(y): y \in N]$

距离向量算法

基本思想:

- 每个节点周期性的给相邻节点发送自己的距离向量估值
- 当节点 x 从它的任何一个邻居 v 收到一个新的距离向量估值,就使用**B-F** 方程更新自己的距离向量估值

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

在简单正常的情况下,距离向量估值 $D_x(y)$ 收敛到实际的最小路径开销 $d_x(y)$

距离向量算法

迭代的, 异步的:

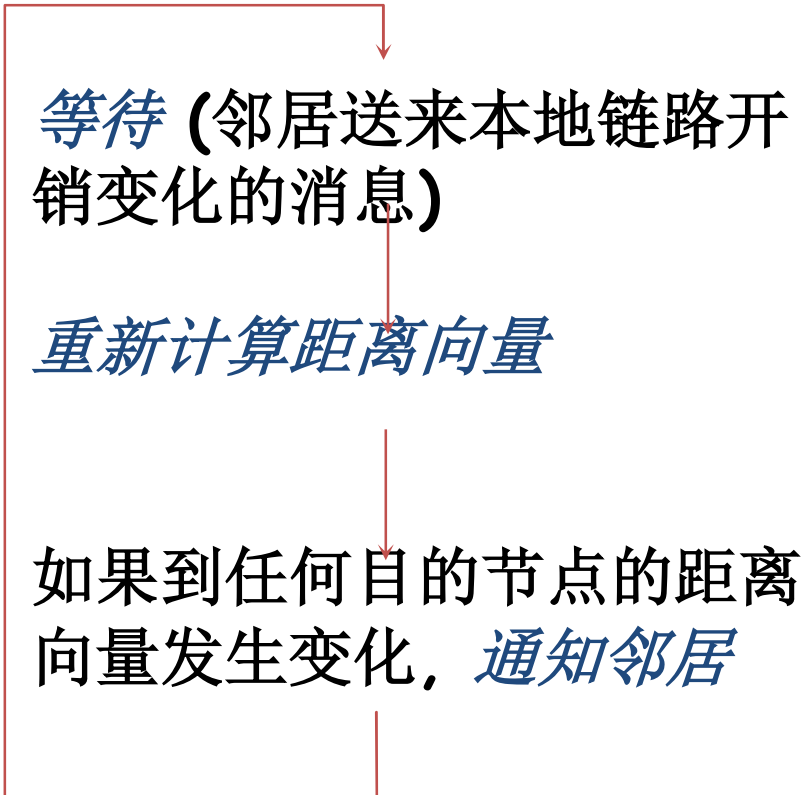
每次本地迭代的产生是由于:

- 本地链路开销变化
- 邻居来的信息: 从邻居来的最小开销路径变化

分布的:

- 每个节点只是在它的距离向量发送变化的时候通知相邻节点
 - 在必要时邻居再通知它们的邻居

每个节点



```
graph TD; A[等待 (邻居送来本地链路开销变化的消息)] --> B[重新计算距离向量]; B --> C[如果到任何目的节点的距离向量发生变化, 通知邻居]; C --> A;
```

等待 (邻居送来本地链路开销变化的消息)

重新计算距离向量

如果到任何目的节点的距离向量发生变化, 通知邻居

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

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

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

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

node x table

开销

	x	y	z
from x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

node y table

开销

	x	y	z
from x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

node z table

开销

	x	y	z
from x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

开销

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

开销

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

开销

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

开销

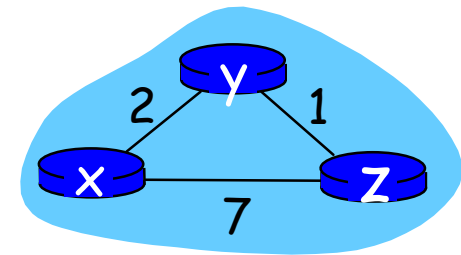
	x	y	z
from x	0	2	3
y	2	0	1
z	3	1	0

开销to

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

开销

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

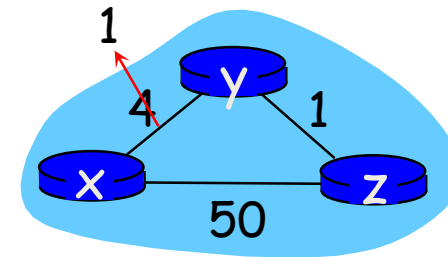


time

距离向量: 链路开销变化

链路开销变化:

节点探测到本地链路开销变化
更新路由选择信息, 重新计算距离向量
如果 DV 变了, 通知邻居

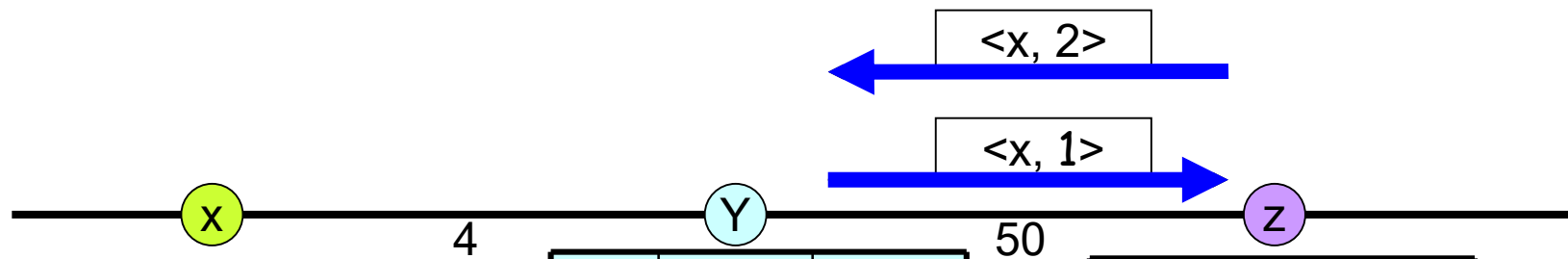


"good
news
travels
fast"

在 t_0 时刻, y 探测到链路开销变了, 就更新它的 DV, 并通知邻居节点.

在 t_1 时刻, z 接收到从 y 发来的更新信息, 并更新自己的 DV. 它计算得到一个到达 x 最短路径, 然后向它的邻居发送自己的 DV。

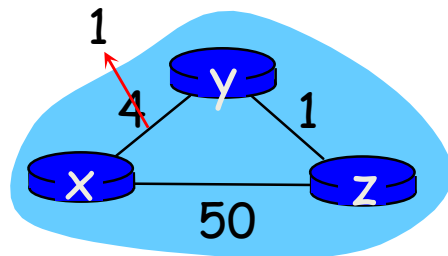
在 t_2 时刻, y 接收到 z 的更新消息然后更新自己的距离向量表, y 的最短路径没有变化, 因此 y 不发送任何消息给 z



X	Y	Z
4	0	1

X	Y	Z
2	1	0

X	Y	Z
2	0	1



距离向量: 链路开销变化

链路开销变化:

好消息传得快

坏消息传得慢 - “计数到无穷” 问题!

(右图中) 在[链路算法](#)稳定之前要进行40余次迭代

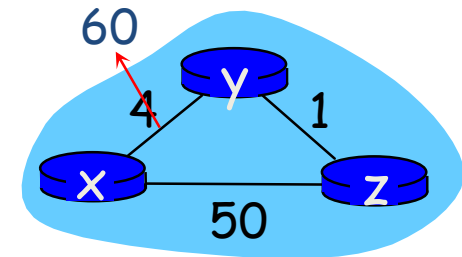
计数到无穷的解决办法: 毒性逆转

假如 Z通过 Y 到达 X:

Z告诉Y它到X的距离是无穷大, Y将不会再经过Z到X

这可以完全解决计数到无穷的问题吗?

不能, 三个以上节点的环路不能被毒性逆转技术检测到。



链路开销远大于60
无穷计数！

<x, 50>

<x, 50>

<x, 7>

<x, 6>



	X	Y	Z
Y	4	0	1
Z	5	1	0

X	Y	Z
5	1	0

X	Y	Z
6	0	1

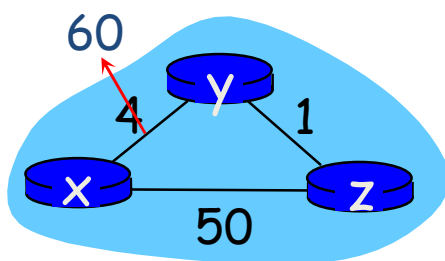
X	Y	Z
7	1	0

X	Y	Z
8	0	1

X	Y	Z
50	0	1

X	Y	Z
50	1	0

X	Y	Z
51	0	1



LS算法和DV算法的比较

报文复杂性

- LS: 具有 n 个节点, E 个链路情况, 每次发送 $O(nE)$ 个报文
- DV: 只是在邻居间交换信息
 - 收敛时间是变化的

收敛速度

- LS: $O(n^2)$ 算法要求 发送 $O(nE)$ 个消息
 - 可能导致振荡
- DV: 收敛时间变化
 - 可能产生循环路由
 - 计数到无限的问题

健壮性: 如果路由器出故障会怎么样?

LS:

- 节点会广告错误的链路开销
- 每个节点只计算自己的转发表 (提供了一定程度的健壮性)

DV:

- DV节点会通告错误的路径开销
- 每个节点的转发表可被其他节点使用
 - 错误会扩散到整个网络

第五章：网络层——控制平面

- 5.1 概述
- 5.2 路由选择算法
 - 链路状态选路算法
 - 距离向量算法
- 5.3 因特网中的自治系统内部路由选择
- 5.4 ISP之间的路由选择： BGP
- 5.5 ICMP协议

5.3 自治系统内部的路由选择

迄今为止，我们的路由研究都是理想化的
所有路由器一样的
网络是“平面的”
... 实际中并不是这样的

规模:具有数十亿个节点

- 路由表中不可能存储所有的节点!
- 路由表的信息交换将淹没数据链路!

管理自治

- internet = 众多网络组成的网络
- 每个网络管理者管理自己网络的路由选择

层次选路

- 一个区域内的路由器组成集合“自治系统”(AS)。
- 每个ISP的网络构成一个或多个AS，具有全局唯一的AS号(ASN)
- 同一个自治系统的路由器运行相同的路由协议
 - 区域内路由协议
 - 不同自治系统内的路由器可以运行不同的区域内路由协议

网关路由器

- 和其他自治系统内的路由器直接相连的路由器
 - 运行域间路由协议，与其他网关路由器交互
- 同自治系统内的所有其他路由器一样也运行域内路由协议

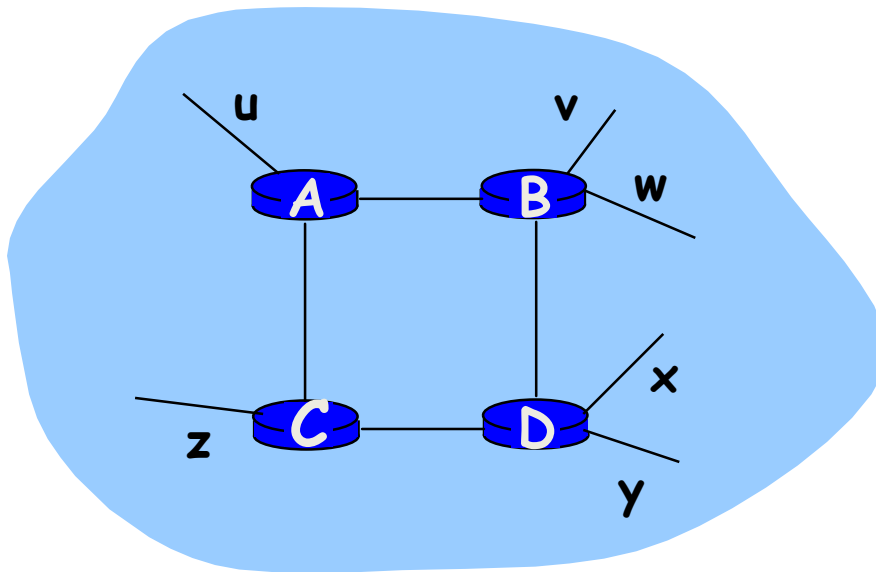
自治系统内的路由选择

- 也被称作**内部网关协议 (IGP)**
- 标准的域内路由协议：
 - **RIP**: 路由信息协议
 - **OSPF**: 开放最短路径优先
 - **IGRP**: 内部网关路由协议 (**Cisco** 所有)

5.3.1 RIP (Routing Information Protocol)

- 距离向量算法
- 包含在 软件1982年发布的BSD-UNIX 版本中。
- 距离衡量: 跳数 (max = 15 hops)

从A到所有叶子子网的跳数

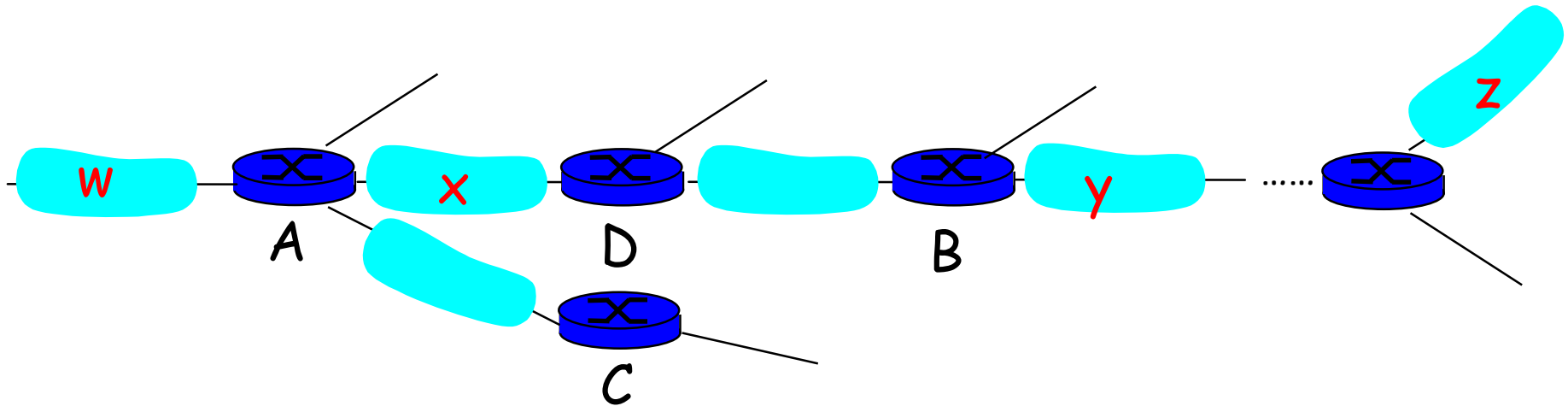


<u>destination</u>	<u>hops</u>
u	1
v	2
w	2
x	3
y	3
z	2

RIP 通告

- 距离向量: 每隔**30秒**,通过响应报文在邻居间进行交换(也被称为**RIP通告**, **advertisement**)
- 每个通告: 包含了多达**25个AS**内的目的子网的列表

RIP: Example



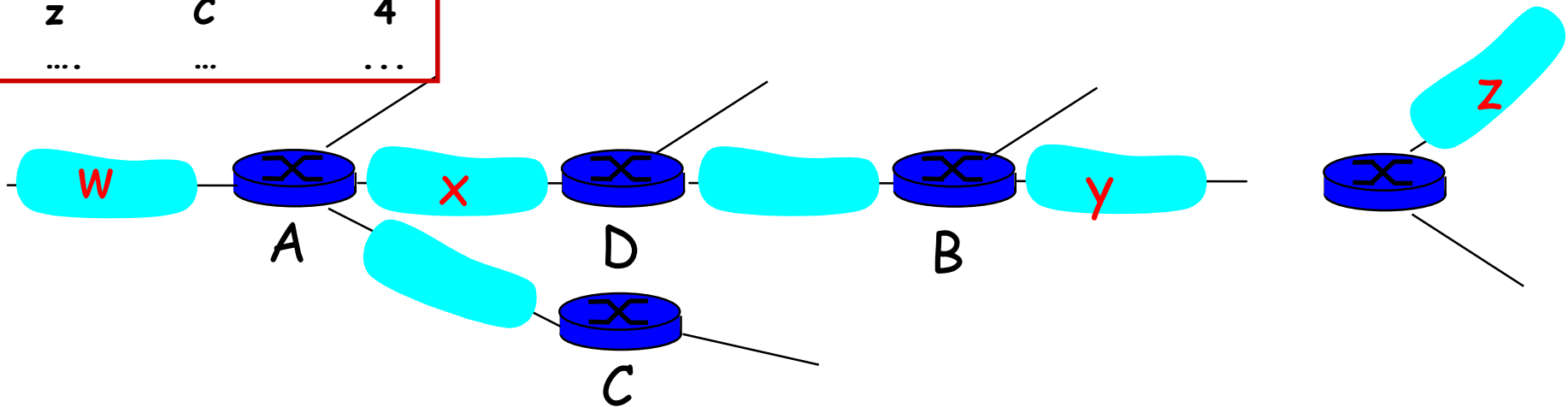
目的子网	下一跳路由器	到目的地的跳数
w	A	2
y	B	2
z	B	7
x	--	1
....

D的转发表

RIP: Example

目的	下一跳路由器	跳数
w	-	-
x	-	-
z	C	4
...

从A到D的一个通告



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

D收到上述通告后的转发表

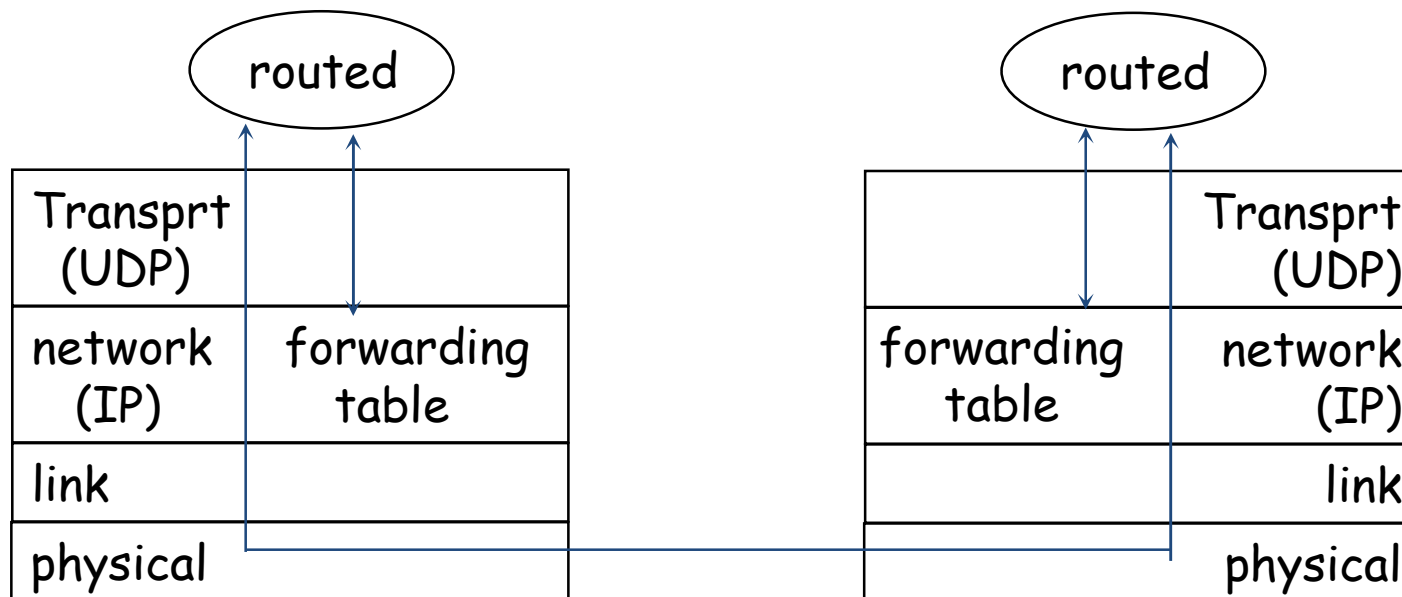
RIP: 链路失败及恢复

若**180**秒后没有收到通告，则认为邻居死机或链路中断。

- 通过故障邻居的路由失败
- 新的公告发送给其他邻居
- 邻居然后再发送新的公告 (如果转发表发生变化)
- 链路故障信息快速传播到整个网络
- 毒性逆转用于防止**乒乓循环** (无限距离 = 16 跳)

RIP 转发表处理

- **RIP转发表**（路由表）是由应用级的进程称为 **route-d** (后台程序)管理的。
- 通告通过 **UDP** 报文(端口**520**)进行发送, 周期性重复



5.3.2 OSPF (Open Shortest Path First)

- “open”: “开放” : 公用的
- 用链路状态算法
 - 分发LS 分组
 - 每个节点具有拓扑图
 - 路由计算使用 **Dijkstra**算法
- 每个**router**都广播**OSPF**通告， **OSPF**通告里为每个邻居路由器设一个表项（记录每个邻居的链路特征和费用）。
- 通告会散布到 **整个** 自治系统 (通过洪泛法)
 - **OSPF**信息直接通过 **IP**传输 (不是 **TCP** 或 **UDP**) , **IP**首部中上层协议字段为**89**

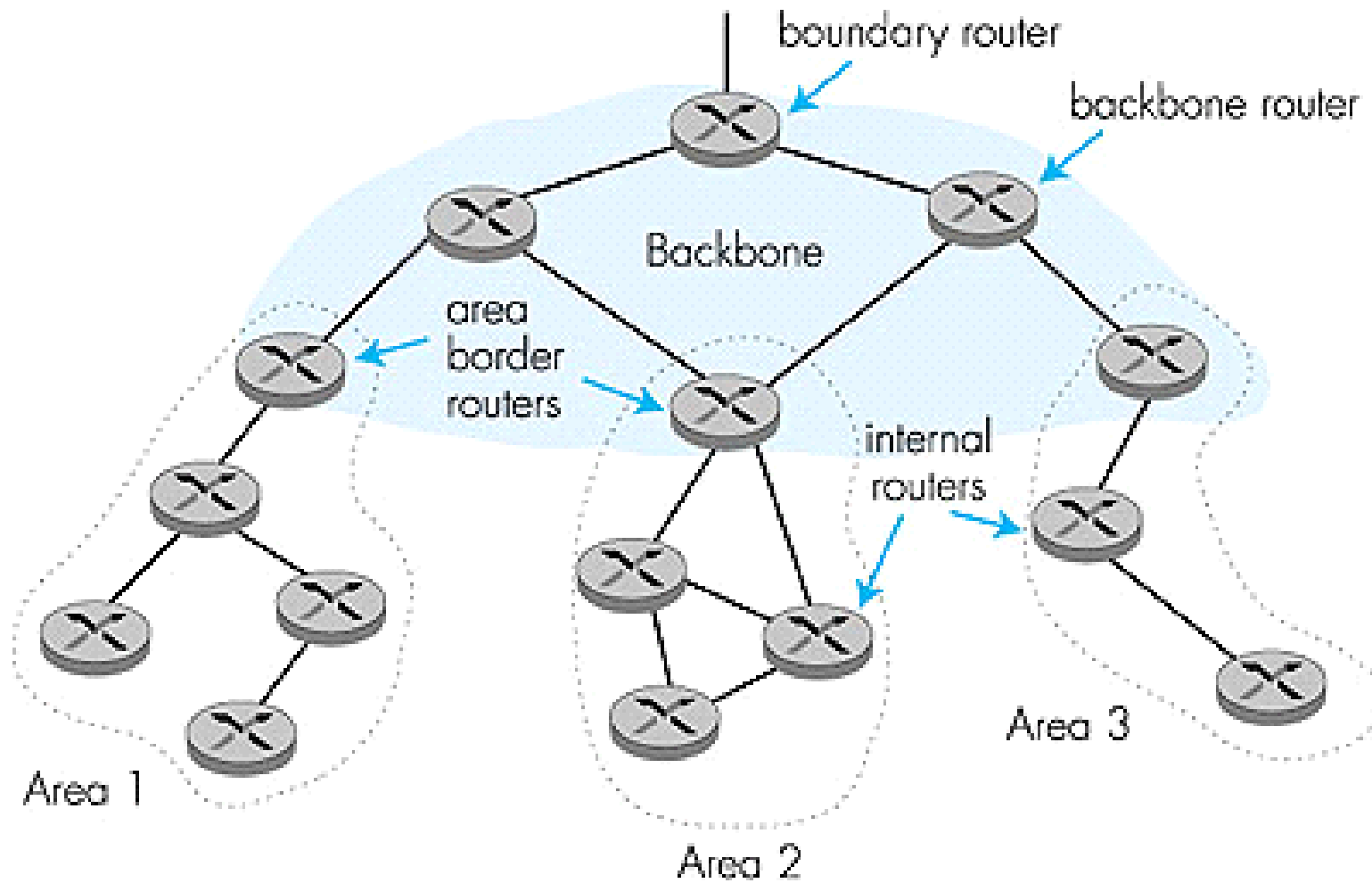
OSPF 优点 (RIP所没有的)

- **安全:** 所有OSPF 消息需要认证 (防止恶意入侵)
- 允许**多个**相同开销的**路径** (在 RIP中只有一条路径)
- 对于每个链路, 有多个消费尺度用于不同的**服务类型TOS** (例如在尽力转发时卫星链路代价设置为“低”, 而对实时应用设置为高)
- 单播和多播综合支持:
 - 多播 **OSPF (MOSPF)** 使用和 **OSPF**同样的链路数据库
- 在大的区域中使用**层次 OSPF**.

层次 OSPF

- 两级层次: 本地区域, 主干区域. (这些区域都是在同一个自治系统内)
 - 只在区域内发送链路状态通告
 - 每个节点有详细的区域拓扑; 仅知道到达其他区域内网络的方向 (即最短路径)
- 区域边界路由器 (同时属于本地区域和主干区域): “汇总” 了到本区域内部网络的路径, 并通告给其他区域边界路由器.
- 主干路由器: 限于在主干区域内运行OSPF路由协议. (本身不是区域边界路由器)
- 边界路由器: 连接到其他自治系统.

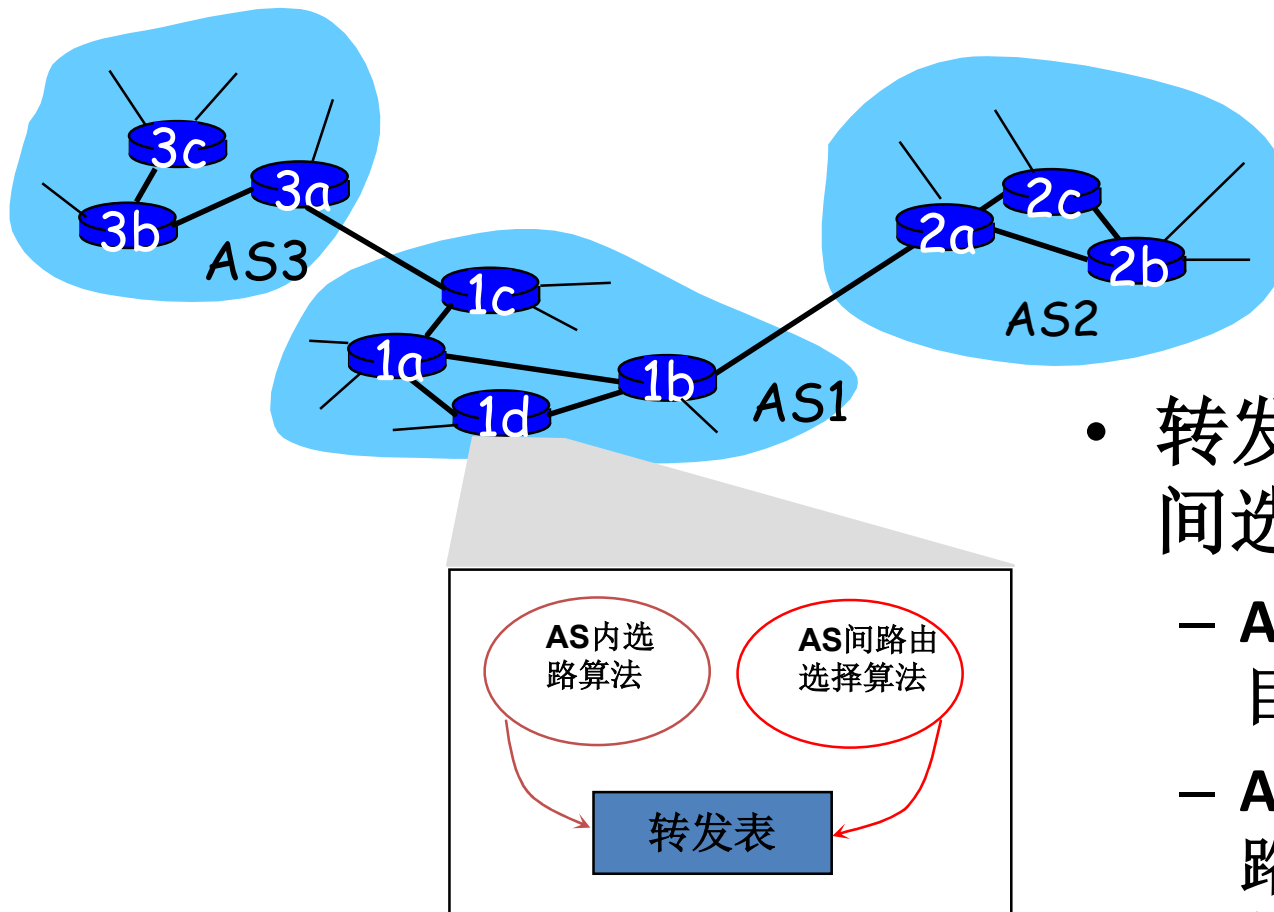
层次 OSPF



第五章：网络层——控制平面

- 5.1 概述
- 5.2 路由选择算法
 - 链路状态选路算法
 - 距离向量算法
- 5.3 因特网中的自治系统内部路由选择
- 5.4 ISP之间的路由选择：BGP
- 5.5 ICMP协议

AS互连



- 转发表根据**AS内**和**AS间**选路算法而配置。
 - **AS域内**的选路项用于目的端在域内的选路。
 - **AS域内**和**AS域间**的选路项用于目的端在域外的选路。

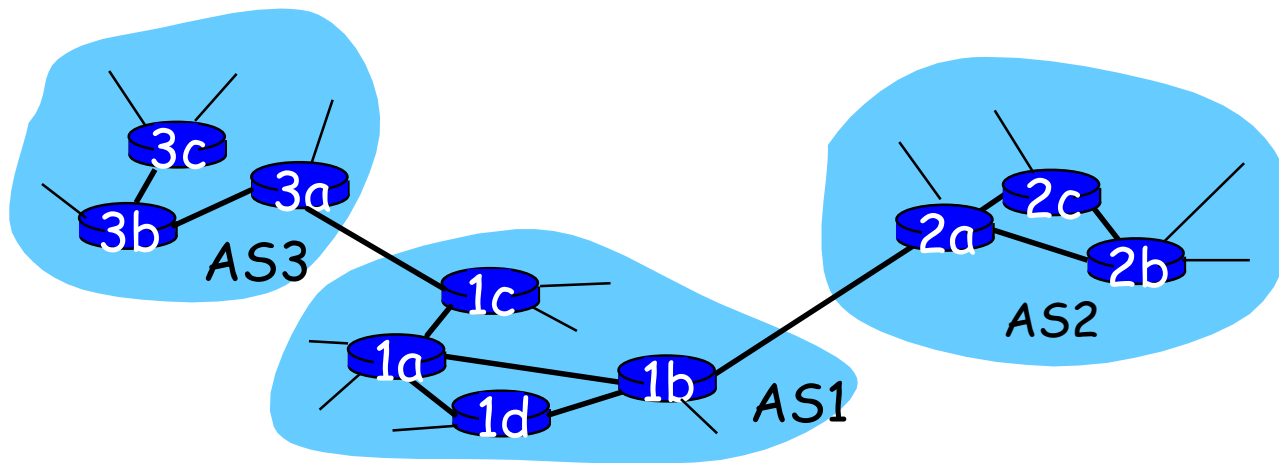
AS域间任务

- 假设**AS1**中的路由器接收到了目的端是**AS1**外的分组
 - 路由器将把这个分组转发到网关路由器，但是是哪个网关路由器呢？

AS1 需要知道:

- 通过**AS2**和**AS3**可以到达哪些目的端
- 将这些可达信息传播给**AS1**内的所有路由器

这就是域间选路的任务！

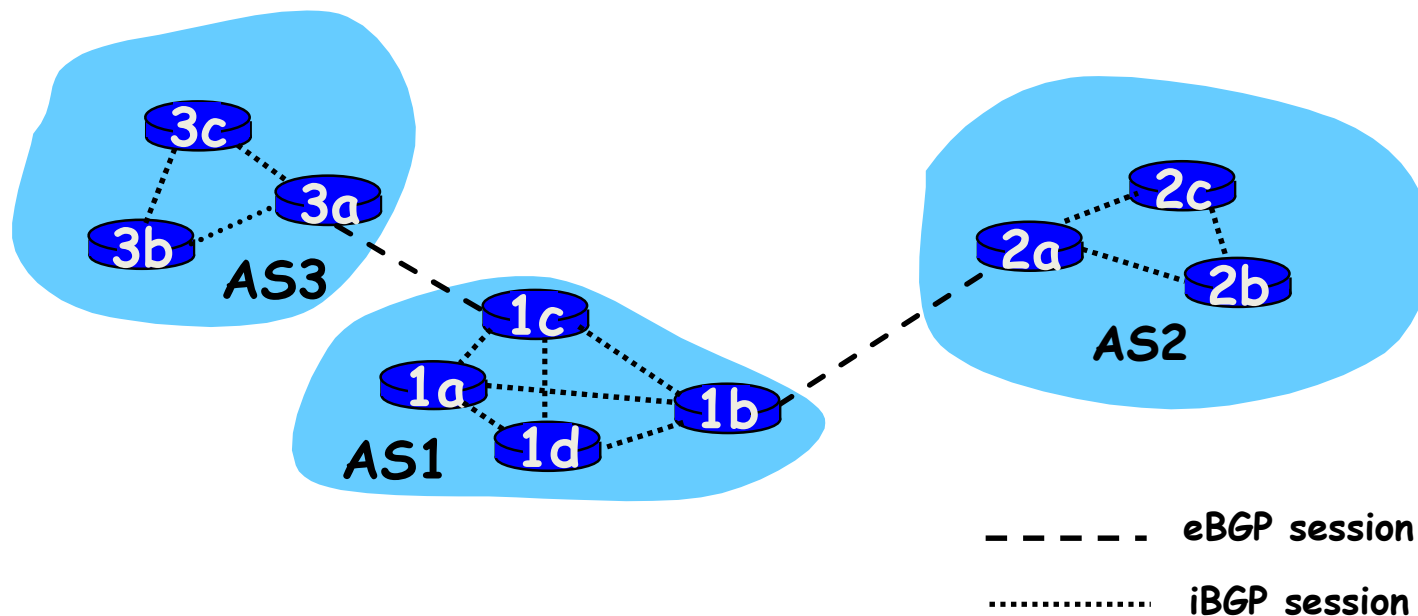


5.4.1 Internet 域间选路: BGP

- **BGP (Border Gateway Protocol):** 事实上的标准
- **BGP** 为每个 **AS** 提供了一种手段:
 - 从相邻**AS**获取子网可达信息
 - 向该**AS**内部的所有路由器传播这些可达性信息
 - 基于该可达信息和**AS**策略, 决定到达子网的“好”路由
- 允许一个子网向Internet的其他部分通告它的存在 *“I am here”*

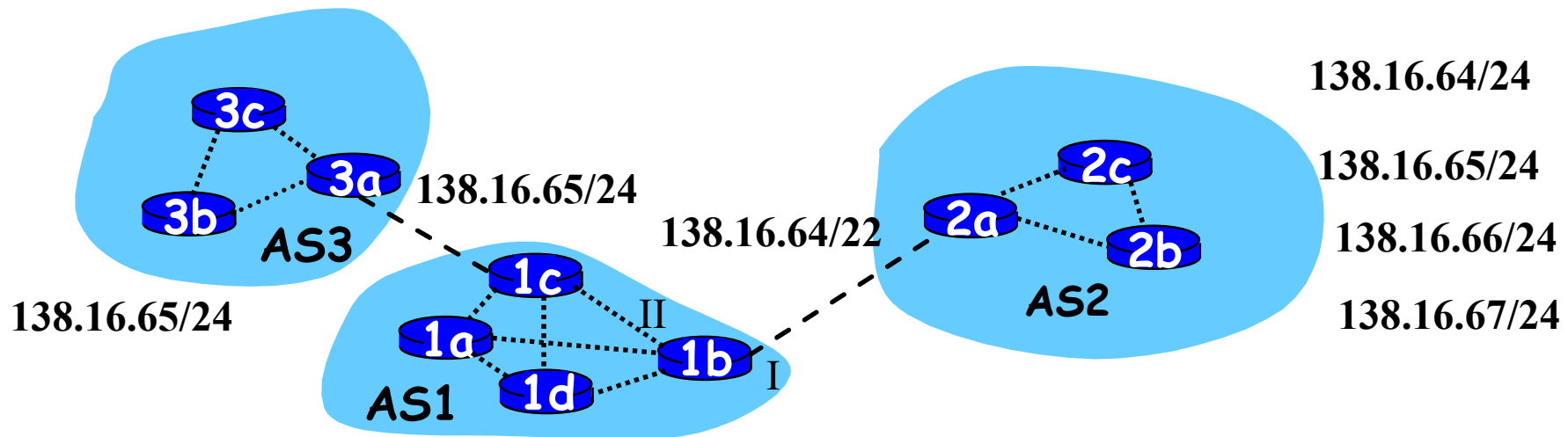
5.4.2 通告BGP路由信息

- 路由器对（BGP对等方）通过半永久TCP连接(端口179)来交换选路信息：**BGP 会话（eBGP和iBGP）**
- 当AS2通告一个前缀给AS1,说明AS2能够转发目的地址前缀是这个**通告前缀**的所有分组。
 - AS2能够在它的通告中汇总了这些前缀。



传播可达信息

- 在3a和1c的eBGP会话中，AS3向AS1通告一个前缀可达信息。
- 1c通过iBGP会话向AS1中的所有路由器发布这个新的前缀可达信息。
- 1b 又将这个可达信息通过1b和2a之间的eBGP会话通告给AS2。
- 当路由器得知一个新的前缀时，就在它的转发表中为该前缀创建一个项。



前缀	下一跳	出口
138.16.64/22	2a	I
138.16.65/24	1c	II

----- eBGP session

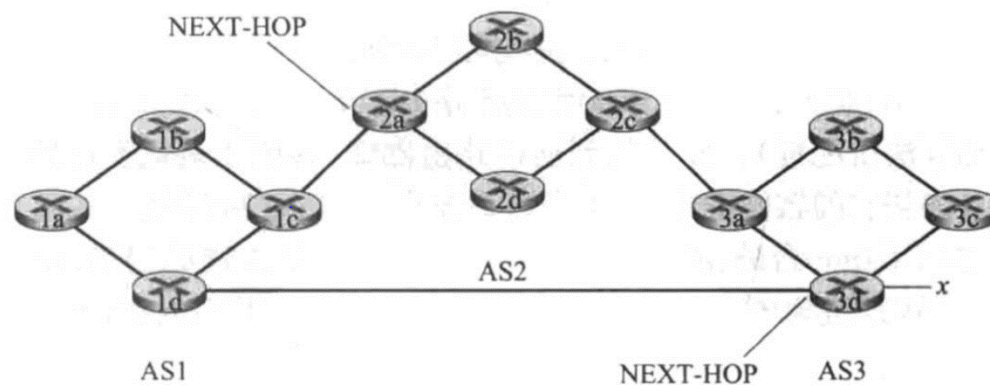
..... iBGP session

5.4.3 确定最好的路由

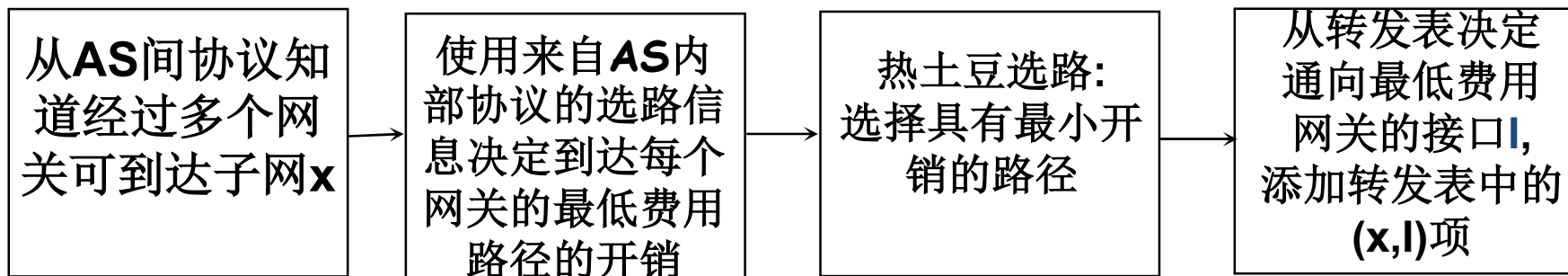
- 自治系统有全局唯一的自治系统号**ASN**标识
 - 桩**AS**没有**ASN**，仅承载源或目的为本**AS**的流量
- 当通告前缀时，通告包含了**BGP**属性。
 - 前缀+属性=“路由”
- 两个重要的属性：
 - **AS-PATH**
 - **NEXT-HOP**

路径属性 和 BGP 路由

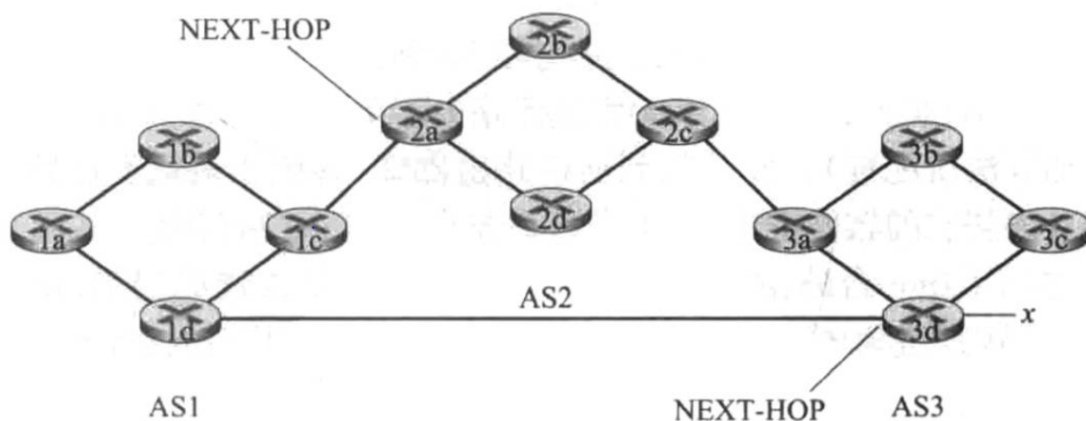
- **AS-PATH:** 包含了前缀的通告已经通过的那些AS
 - 如AS2的网关路由器2c将收到3a的通告AS3,x, AS1的网关路由器1c将收到来自2a的BGP通告AS2 AS3, x, 1d将收到来自3d的BGP通告AS3, x
 - 可以检测和防止通告环路
- **NEXT-HOP:** 指出到达下一个AS的具体AS间边界路由器的IP地址
 - 对AS1而言, AS2的2a路由器左侧出口IP地址和AS3的3d左侧IP地址
- AS1中, 路由器将知道到达网络x的两条BGP路由 (NEXT-HOP,AS-PATH,前缀)
 - 2a左侧IP地址, AS2 AS3, x
 - 3d左侧IP地址, AS3, x



热土豆路由

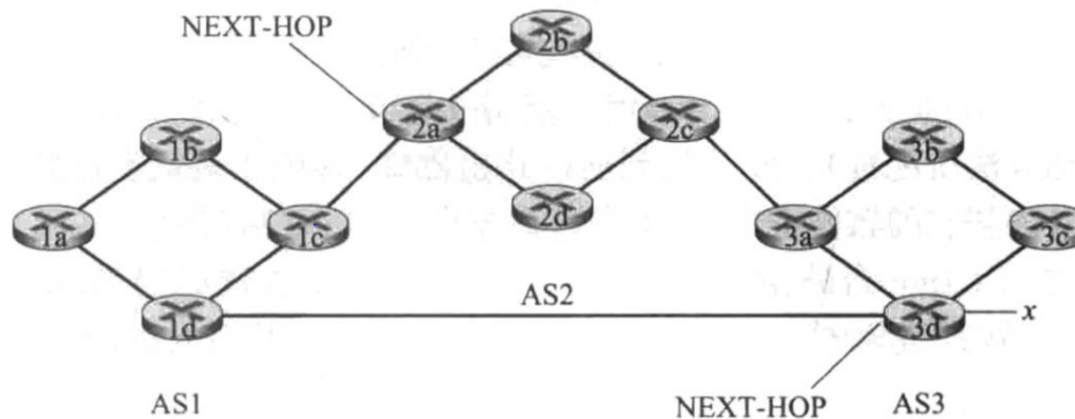


- 尽可能快的将分组转发到其他**AS**中
- 路由器**1b**会得到的两条到前缀**x**的BGP路由
 - **2a**左侧IP地址，**AS2 AS3**， **x**
 - **3d**左侧IP地址，**AS3**， **x**



BGP 路由选择

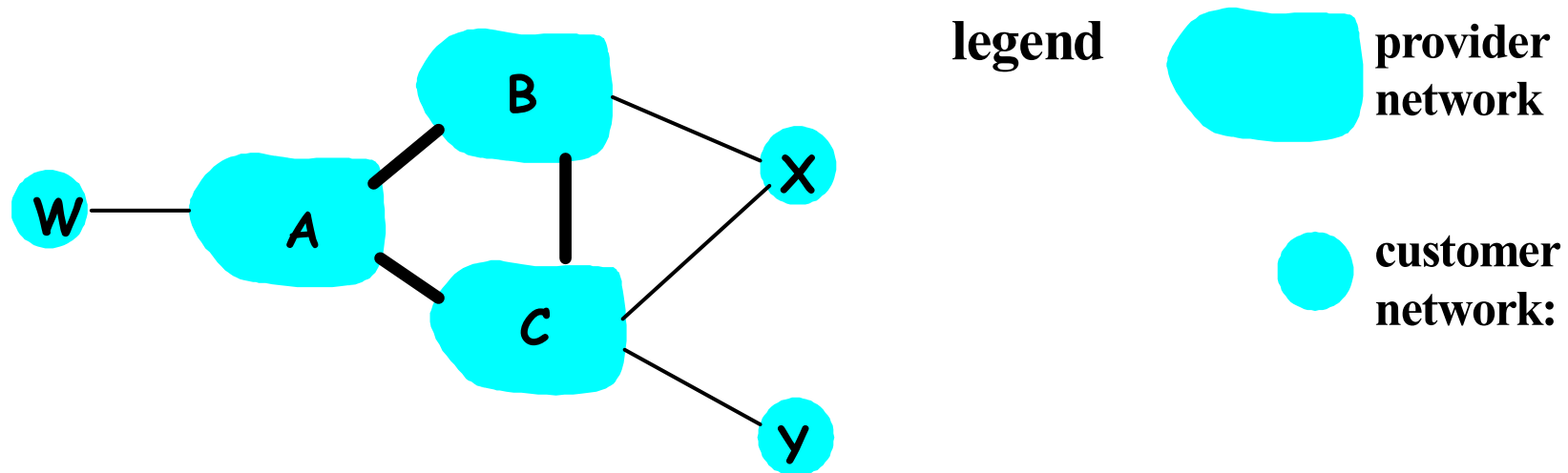
- 当网关路由器接收到路由通告时，使用**输入策略**来决定接收/舍弃该通告
 - 不希望为**AS-PATH**中的某个**AS**发送流量
 - 已经知道一条到相同前缀的路由
- 路由器可能知道到相同前缀的多条路由，路由器必须从中选择.
- 排除规则（应用排除规则直到有一条留下）
 - 本地偏好值属性: 具有最高偏好值的路由被选择
 - 最短**AS-PATH**的路由
 - 最靠近 **NEXT-HOP**路由器的路由：热土豆路由
 - 其他标准



路由器1b的两条BGP路由

- 2a左侧IP地址, AS2 AS3, x
- 3d左侧IP地址, AS3, x

5.4.5 BGP 路由选择策略



A,B,C 是提供商的网络

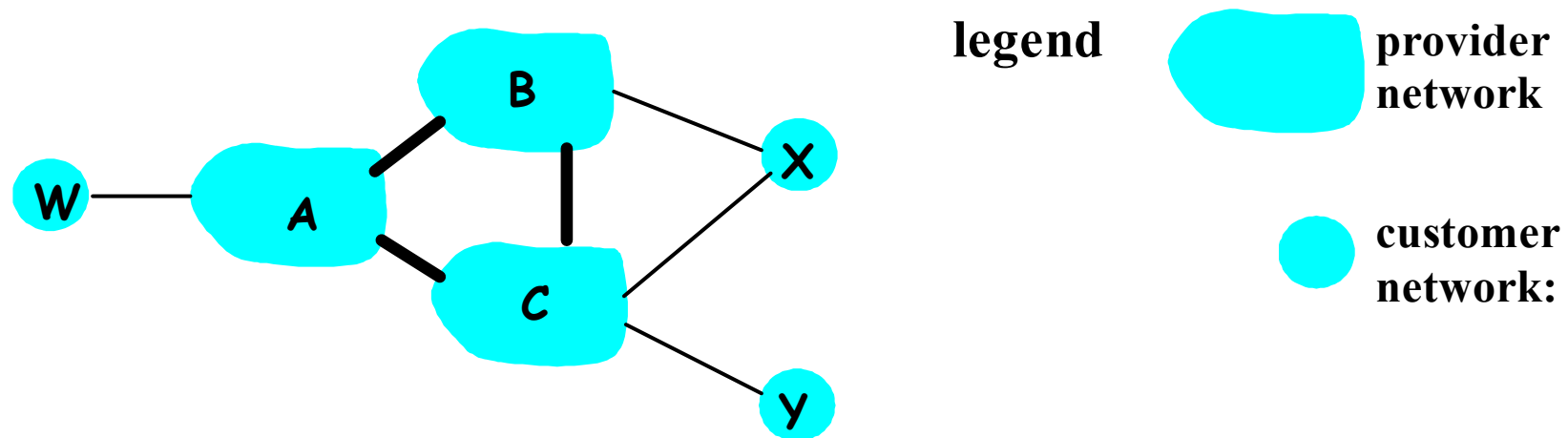
X,W,Y是提供商的客户，桩网络

X是双重的：连接到两个网络

X 不希望 B 通过 X 到 C 的路由BXC

.. 所以 X 不会向B公告到C的路由XC

BGP选路策略



提供商网络向它的客户转发BGP通告，提供商网络之间如何转发BGP通告没有统一规定

A 向B通告路径 AW

B 向X通告路由BAW

B 应该向C通告路由BAW?

决不! B 路由 CBAW 没有什么“好处”，因为 W 和 C 都不是 B 的客户

B 希望强迫 C 通过A路由到W--CAW。

B 只想路由它的客户!

•对于ISP：任何穿越该ISP主干网的流量必须是其源或目的位于该ISP的客户网络中

为什么AS内选路和AS间选路采用不同的协议？

策略:

- **AS间:** 管理员想控制本AS内产生的通信流怎样选路, 以及什么通信流穿过自己的网络
- **AS内:** 单个管理者, 因此不需要策略

规模:

- 层次路由节省了转发表的大小空间, 减少了路由更新的流量

性能:

- **AS内:** 集中在性能上
- **AS间:** 策略可能比性能更加重要

第五章：网络层——控制平面

- 5.1 概述
- 5.2 路由选择算法
 - 链路状态选路算法
 - 距离向量算法
- 5.3 因特网中的自治系统内部路由选择
- 5.4 ISP之间的路由选择： BGP
- 5.5 ICMP协议

5.5 ICMP: Internet Control Message Protocol

因特网控制报文协议

- 用于主机路由器之间彼此交流网络层信息
 - 差错报告: 不可到达的主机, 网络, 端口, 协议
 - 请求/应答 (用于 ping, traceroute)
- 位于IP之上
 - 因为ICMP消息是装载在IP分组里的
- **ICMP 报文结构:** 类型字段, 编码字段 以及引起该ICMP报文的IP分组的前8字节

类型	代码	描述
0	0	回应应答 (ping)
3	0	目的网络不可到达
3	1	目的主机不可到达
3	2	目的协议不可到达
3	3	目的端口不可达到
3	6	不知道的目的地网络
3	7	不知道的目的地主机
4	0	源端抑制 (拥塞控制 - 不用)
8	0	回应请求 (ping)
9	0	路由器公告
10	0	路由器发现
11	0	TTL 过期
12	0	IP首部损坏

Traceroute 和 ICMP

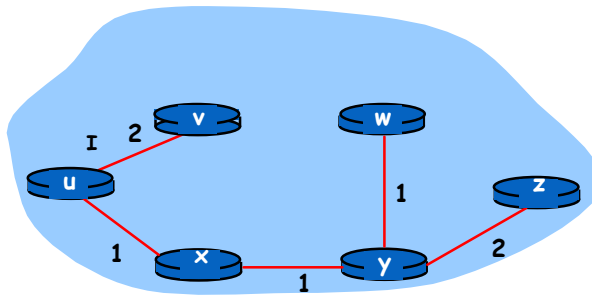
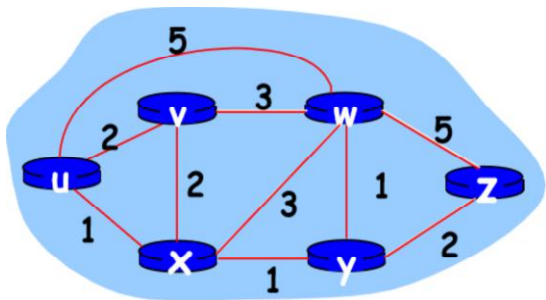
- 源端发送一系列的IP分组给目的端
 - 第一个分组 TTL = 1
 - 第二个 TTL=2, 等.
 - 当第n个分组到达第n个路由器时
 - 路由器丢弃该分组
 - 并给源端发送一个**ICMP报文** (type 11, code 0)
 - 这个报文包含了路由器的名称和IP地址
 - 当源端收到ICMP报文时, 计算传输往返时间RTT
 - 对每个TTL作三次
- 停止发送的根据
- IP报文最终到达目的端
 - 目的端返回回应应答的 **ICMP 报文 (type 3, code 3)**
 - 源端收到此报文, 停止发送

网络层：复习大纲

- 选路算法
 - 链路状态选路算法
 - 距离向量算法
 - 层次选路
- 了解因特网中的选路协议
 - 内部网关协议：RIP、OSPF、IGRP
 - 外部网关协议：BGP

选路算法

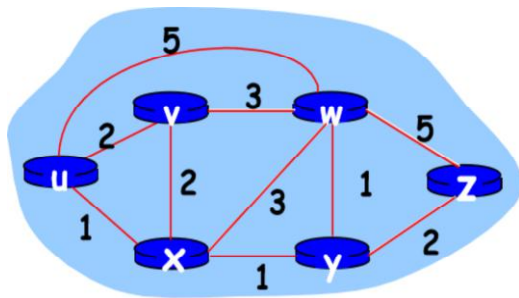
- 分类
 - 全局算法——链路状态算法；分布式算法——距离矢量算法
 - 静态算法和动态算法
- 链路状态算法*
 - 网络中所有节点都知道整个网络的拓扑结构和链路开销
 - 通过链路状态广播获得网络拓扑
 - Dijkstra算法：计算源节点到网络中其他所有节点的最短路径树
 - 根据最短路径树填写路由表



目的地x	下一跳	输出接口
v	-	$l(u,v)$
x	-	$l_l(u,x)$
y	x	$l_l(u,x)$
w	x	$l_l(u,x)$
z	x	$l_l(u,x)$

选路算法

- 距离向量算法*
 - 节点只知道邻居节点的信息，仅与邻居节点交换消息
 - 节点维护自己和邻居节点的距离向量
 - 节点收到任一邻居的新的距离向量，使用B-F方程更新自己的距离向量，并将新的距离向量发送给其邻居，直至收敛
 - 链路开销变化
 - 好消息传的快；
 - 坏消息传的慢，“无穷计数问题”。毒性逆转



U	V	W	X	Y	Z
0	2	5	1	-	-

U	V	W	X	Y	Z
0	2	4	1	2	10

U	V	W	X	Y	Z
0	2	3	1	2	4

层次选路

- 众多的自治系统
 - 自治系统内部路由器使用相同的域内路由协议（链路状态或距离向量）
 - 自治系统通过网关路由器相连，网关路由器运行域内和域间路由协议
- 域间选路任务
 - 获取相邻自治系统的子网可达信息
 - 将这些可达信息在本自治系统中传播

RIP协议

- 距离向量算法，距离衡量用跳数hops（最大15跳）
- 周期性的每30s向邻居发送RIP通告（包含AS内部多达25个子网距离向量），根据B-F方程更新
- 180s未收到通告，则认为节点或链路失效，传播故障信息
- RIP通告通过应用级进程route-d周期性发送，使用UDP协议，端口520*

IP首部			UDP首部			
...	上层协议 =17	...	源端口 =520	目的端口 =520	...	RIP通告

OSPF协议

- 链路状态算法,链路状态信息包含在OSPF通告中
- OSPF通告通过泛洪方式在整个网络中传播，每个节点将获得整个网络拓扑结构
- 层次OSPF
 - 两级层次——本地区域、主干区域，均运行OSPF协议
- OSPF运行在IP之上，即OSPF通告通过IP分组传输*

IP首部			
...	上层协议=89	...	OSPF通告

BGP协议

- 从相邻AS获取子网可达信息
- 在AS内部传播这些可达信息
- 基于可达信息和AS策略，确定可达子网的路由（添加路由表项）
- BGP通过半永久TCP连接交换信息（端口号179）
 - 外部BGP会话eBGP：网关路由器之间交换信息
 - 内部BGP会话iBGP：AS内路由器之间传播网关路由器由eBGP中获得的信息

IP首部			TCP首部			
...	上层协议=6	目的端口=179	...	BGP报文

- BGP路由=前缀+属性（AS-PATH、NEXT-HOP）
- 收到多条相同前缀的BGP路由选择：本地偏好值->最短AS-PATH->最靠近NEXT-HOP->其他标准
- BGP路由策略：提供商网络向它的客户转发BGP通告，提供商网络之间如何转发BGP通告没有统一规定。一般地，任何穿越提供商主干网的流量必须是其源或目的位于该提供商的客户网络中

ICMP协议

- ICMP协议：主机路由器之间交流网络层信息的协议
 - 位于IP之上，用IP分组承载ICMP消息，IP分组首部的上层协议字段为0x01
 - Traceroute利用了ICMP的TTL过期报文类型（11号）；
 - ping请求是ICMP8号类型报文，ping应答是ICMP 0号类型报文

IP首部			
...	上层协议=1	...	ICMP报文

本章作业

- 习题：P4， P5， P14

涉及计算-Dijkstra算法

- 初始化：邻居节点直接获得链路开销，非邻居节点为无穷大
- 每一轮次选出具有最小开销的节点添加到最短路径树节点集合中
- 新节点添加后，更新到其他节点的链路开销，重复上述操作
- 根据最短路径树填写路由表

涉及计算-DV算法

- 初始化：本节点距离向量-邻居节点直接获得距离信息，非邻居节点为无穷大

网络中其他节点距离向量为无穷大

- 与邻居节点交换距离向量信息，根据Bellman-Ford方程更新本节点的距离向量
- 再次和邻居交换距离向量，直到距离向量不再变换，路由收敛