第四章 网络层控制平面

目录

网络层功能

两大网络层功能

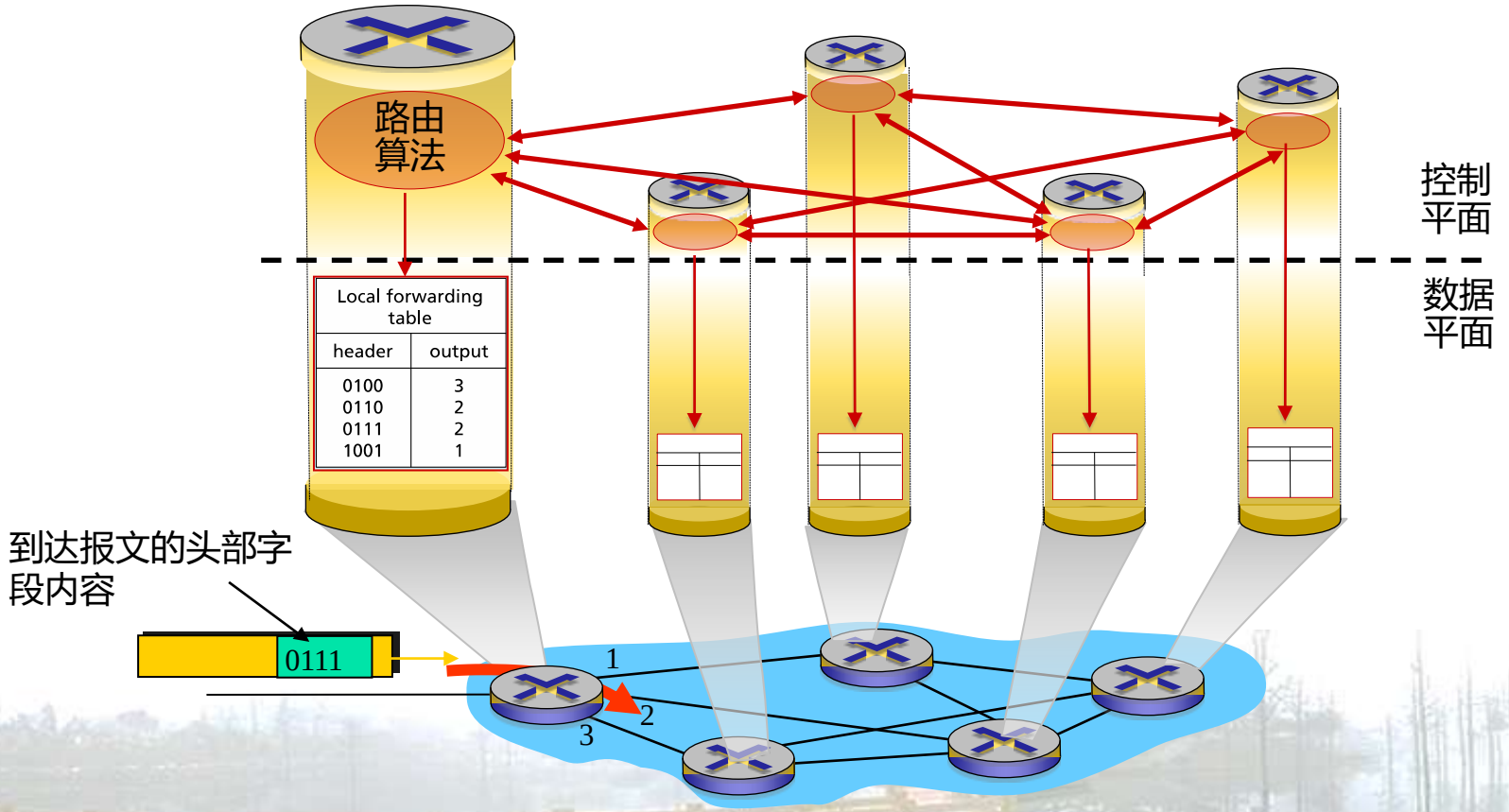
- 转发:将数据包从路由器的入端口转移到正确的出端口 — 数据平面
- 路由：决定数据包从源到目的地的转发路径 — 控制平面

两类控制平面实施方案:

- 传统的路由算法:在每个路由器上实现
- 软件定义网络 (SDN): 逻辑上集中实现

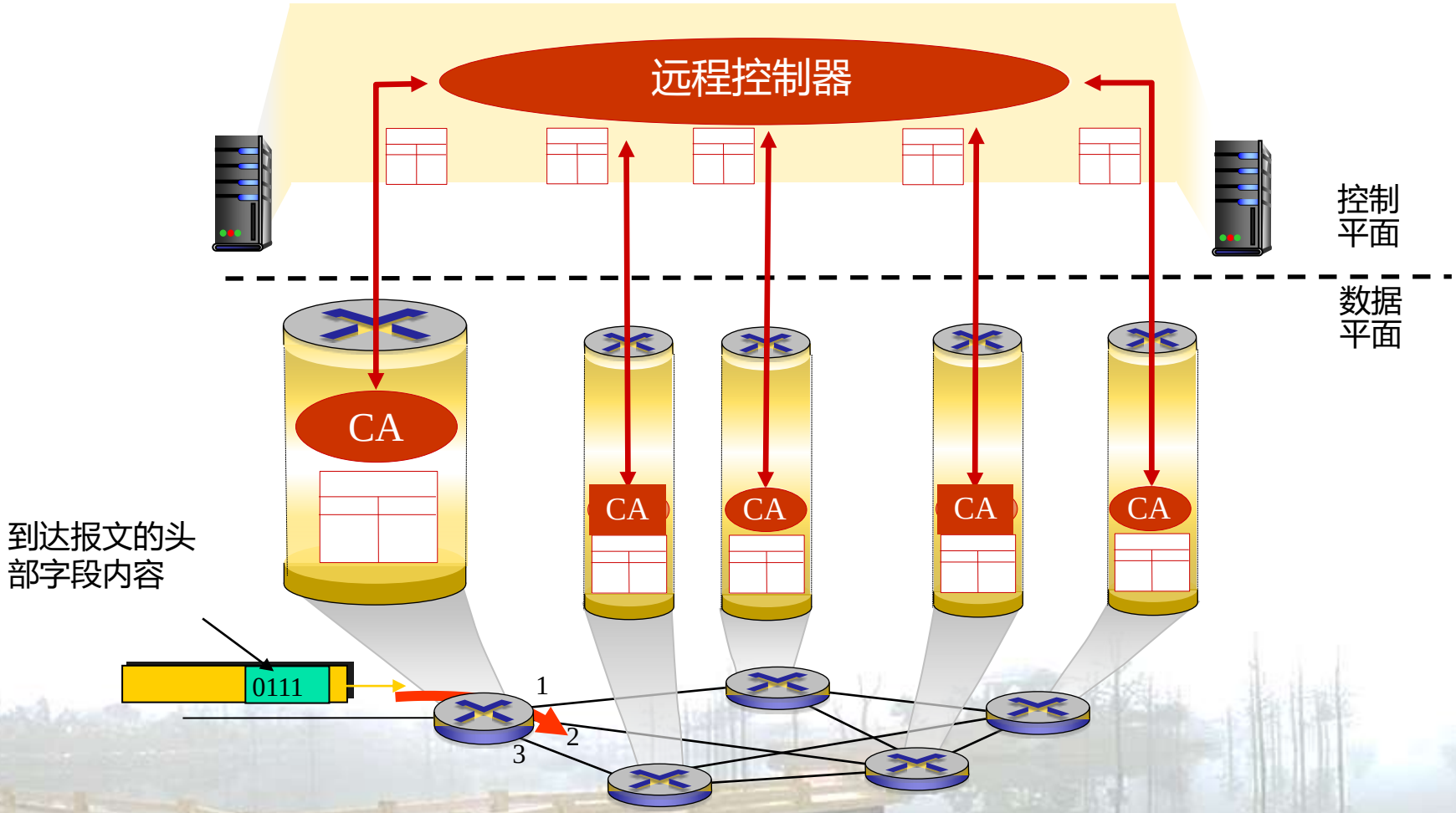
每个路由器参与的控制平面

每个路由器中都有单独的路由算法模块，它们相互交互，计算出各自的转发表，一起构成控制平面



逻辑上集中的控制平面

单独的（远程）控制器与路由器本地的控制代理（CA）交互



目录

路由协议

路由协议目标: 在由路由器构成的网络上，选择从发送端主机到接收端主机的“好”的路径

- 路径：数据包从发送端主机到接收端主机经过的路由器序列
- 什么是“好”：“代价”最小、“最快”、“最不拥塞”
- 路由：网络中top-10挑战!

网络的图抽象

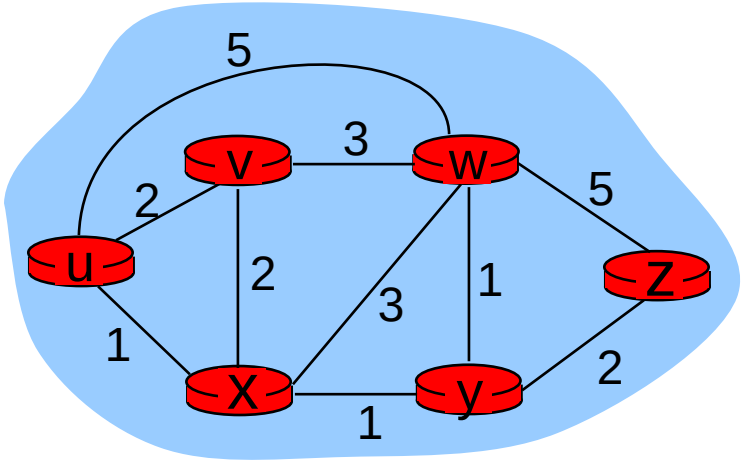


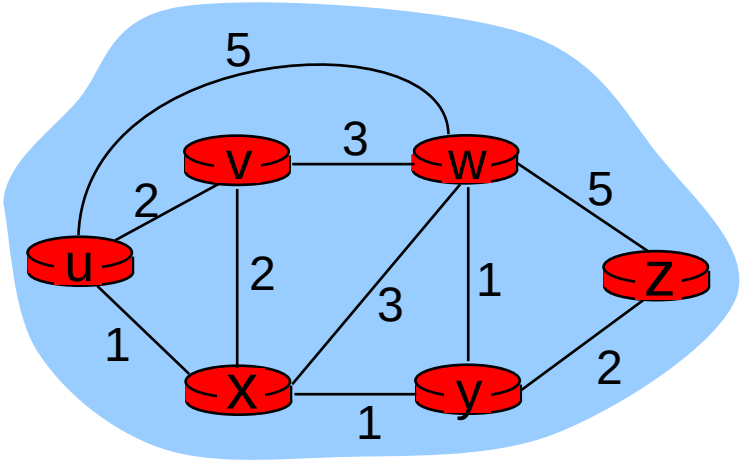
图: G = (N,E)

N = 路由器集合 = { u, v, w, x, y, z }

E = 链路集合 ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

除了用来描述路由器构成的网络，图还可以用于描述其它网络，
例如P2P网络，其中节点是peer，边是peer之间的TCP连接

网络的图抽象：代价



c(x,x′) = 链路(x,x′)的代价
例如, c(w,z) = 5

链路的代价可统一设为1，或者带宽分之一，或者拥塞的程度

路径 $(x_1, x_2, x_3, \ldots, x_p)$ 的代价 = $c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

核心问题: 节点u和z之间最小代价的路径是哪一条？
路由算法: 寻找最小代价（最短）路径的算法

路由算法分类

问: 依赖全局还是局部信息

全局:

- 所有路由器拥有全网拓扑结构和链路代价信息
- “链路状态” 算法

局部:

- 路由器只知道直接相连的邻居，以及到邻居的链路代价
- 通过相邻路由器之间反复迭代交换信息，计算路由
- “距离矢量” 算法

问：静态还是动态

静态:

- 路由变化缓慢

动态:

- 路由变化块
  - 周期更新
  - 链路代价变化触发更新

目录

一种链路状态路由算法

Dijkstra算法

- 所有节点知道拓扑、链路代价信息
  - 通过“链路状态通告”获得
  - 所有节点信息一致
- 每个节点以自己为源，计算到各节点的最小代价路径
  - 得到该节点的转发表
- 迭代：k轮迭代后，获得到k个目的地的最小代价路径

符号:

- c(x,y): 节点x到y链路代价；如果x和y不直连，= ∞
- D(v): 当前从源到节点v的最小代价路径
- p(v): 从源到节点v的最短路径上v前面的一个节点（前驱节点）
- N‘: 已完成最短路径计算的节点集合

Dijsktra算法

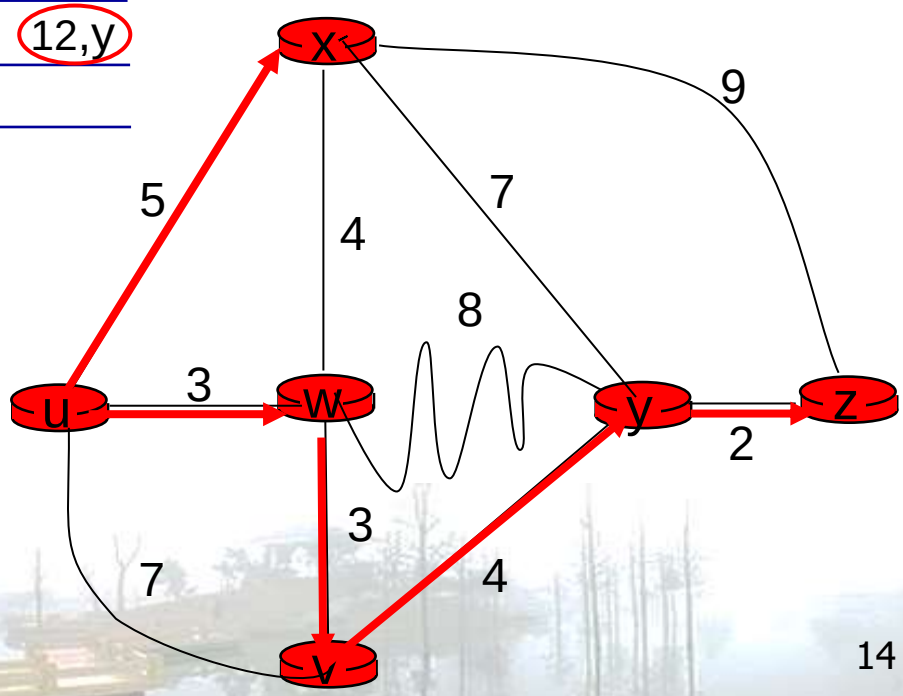
```
1  初始化:
2    N' = {u}
3    for all nodes v
4      if v adjacent to u
5          then D(v) = c(u,v)
6      else D(v) = ∞
7
8   Loop
9     find w not in N' such that D(w) is a minimum
10    add w to N'
11    update D(v) for all v adjacent to w and not in N' :
12       D(v) = min( D(v), D(w) + c(w,v) )
13    /* 到v的新的最短路径要么不变，要么是到w的最短
14     路径加上w到v的链路代价 */
15  until all nodes are in N'
```

Dijsktra算法举例一

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					

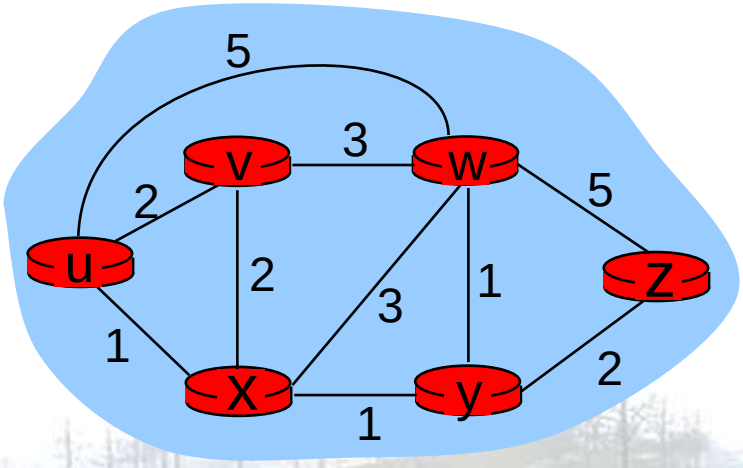
注意:

- 追踪前驱节点获得最短路径树
- 路径等长时可用任何标准选择一条



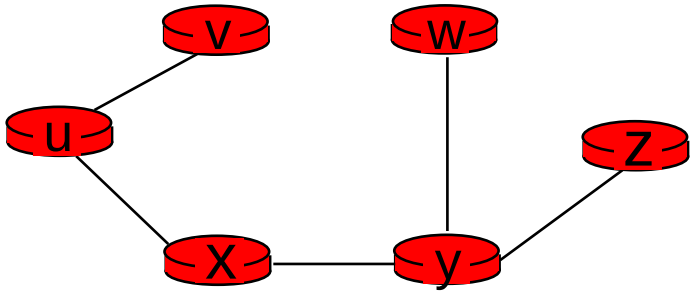
Dijsktra算法举例二

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					



Dijsktra算法举例二

获得的从u出发的最短路径树:



节点u的转发表:

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

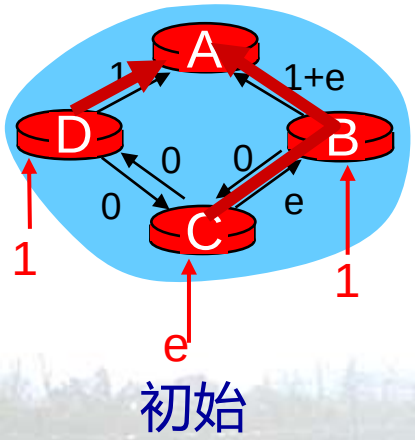
对Dijkstra算法的讨论

算法复杂度: n个节点

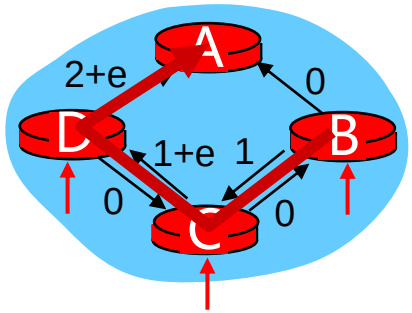
- 每轮迭代：检查所有不在N集合里的节点
- 进行n(n+1)/2次比较：$O(n^2)$
- 可以实现为 O(nlogn)

可能有路由震荡

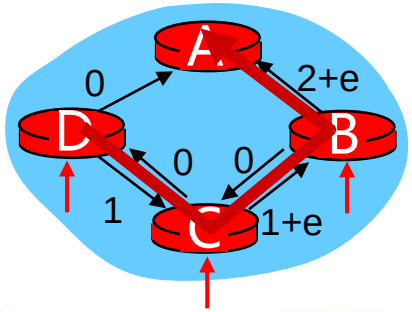
- 例如：假设链路代价等于其上的流量:



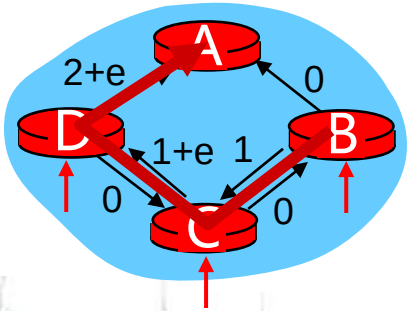
初始



根据代价计算路径



根据代价计算路径



根据代价计算路径

目录

距离矢量算法

Bellman-Ford 等式 (动态规划)

令

$d_x(y)$:= 从x 到 y最小代价路径的代价

则

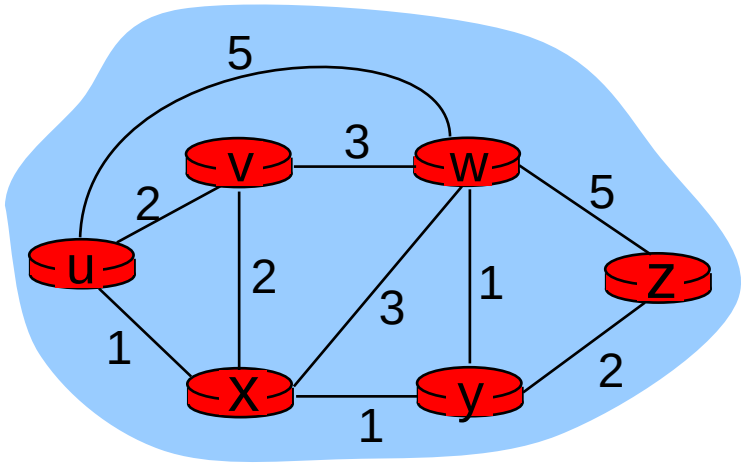
$d_x(y) = \min_v\{c(x,v) + d_v(y)\}$

从v到y的最小代价路径的代价

到邻节点v的代价

对x的所有邻节点v取min

Bellman-Ford算法举例



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

B-F 等式:

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

min计算选中的节点是最短路径上的下一个节点，用于配置u节点上的转发表

距离矢量算法

- $D_x(y)$ = 从x到y的最小路径代价
  - x 维护距离矢量 $\mathbf{D}_x = [D_x(y): y \in N]$
- 节点x:
  - 知道其到所有邻节点v的代价: c(x,v)
  - 有它所有邻节点的距离矢量。对邻节点v，x获取
    $\mathbf{D}_v = [D_v(y): y \in N]$

距离矢量算法

核心思想

- 每个节点不时地将自己的距离矢量发给邻节点
- 当节点x收到来自邻节点的新的距离矢量，使用B-F公式更新自己的距离矢量

对每个节点$y \in N$，$D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\}$

- 在拓扑结构和链路代价变化缓慢时，$D_x(y)$收敛为正确的最小代价 $d_x(y)$

距离矢量算法

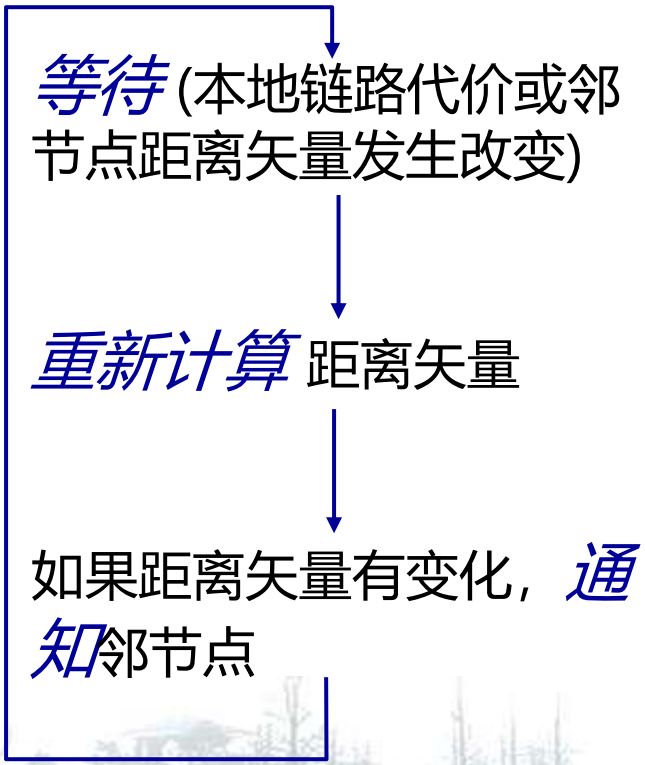
迭代、异步: 每次本地更新由以下事件触发:

- 到邻节点的链路代价变化
- 收到邻节点发来的距离矢量

分布式:

- 当距离矢量变化时，每个节点通知其邻节点
  - 邻节点更新后，如有变化，再通知它的邻节点

节点:



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

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

节点x table

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

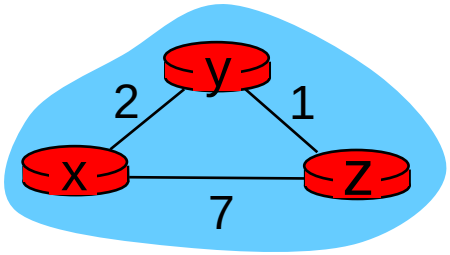
from	cost to x	cost to y	cost to z
x	0	2	3
y	2	0	1
z	7	1	0

节点y table

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

节点z table

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



时间

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

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

节点x table

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

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

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

节点y table

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

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

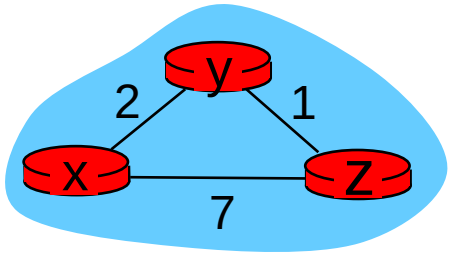
from	cost to x	cost to y	cost to z
x	0	2	3
y	2	0	1
z	3	1	0

节点z table

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

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

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

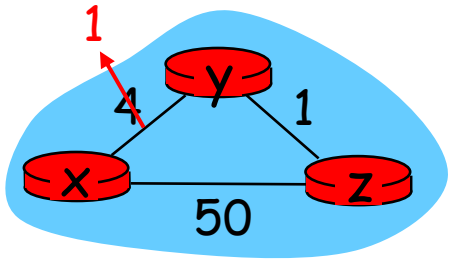


时间

距离矢量：链路代价变化

链路代价变化:

- 节点检测到本地链路代价变化
- 计算并更新距离矢量
- 如距离矢量变化，通知邻节点



好事传得快

t_0 : y检测到本地链路代价变化，更新其距离矢量，通知邻节点.

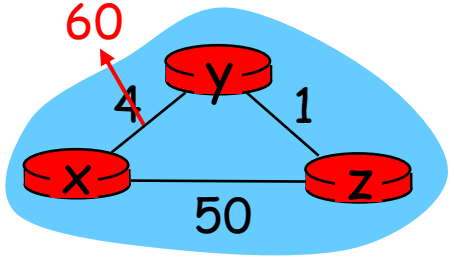
t_1 : z收到来自y的更新，更新本地的距离矢量表，重新计算到x的最小代价，把新的距离矢量发给邻节点.

t_2 : y收到z的更新，更新自己的距离矢量表，因为y到各节点的最小代价没有变化，y不发送更新消息到邻节点。

距离矢量：链路代价变化

链路代价变化:

- 节点检测到本地链路代价变化
- 坏事传得慢– 无穷迭代问题!
- 44次迭代才能稳定



t_0: y检测到本地链路代价变化，y计算到x的最短路径为

$$D_y(x) = \min\{c(y,x) + D_x(x),\ c(y,z) + D_z(x)\} = \min\{60 + 0,\ 1 + 5\} = 6$$

基于y相信z有一条到x的代价为5的路径

t_1: y 告知 z 它的新距离矢量

t_2: z 收到来自y的距离矢量, 更新它到x的最短距离为

$$D_z(x) = \min\{50 + 0, 1 + 6\} = 7$$

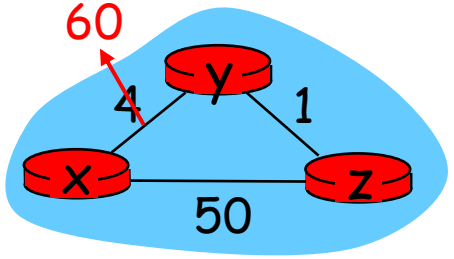
t_3: y 再次更新距离矢量

....

距离矢量：链路代价变化

反向下毒:

- 如果z通过y路由到x：
  - z告知y它到x的代价是无穷大
- 并不能完全解决问题



z告知y $D_z(x) = \infty$

t_0：y检测到本地链路代价变化，y计算到x的最小代价路径为$D_y(x) = 60$，并告知z

t_1：收到y的距离矢量后，z立刻改变到x的路由为直连链路，代价为$D_z(x)=50$

t_2：z告知y $D_z(x) = 50$.

t_3：收到z的距离矢量，y更新其到x的最小代价为$D_y(x) = 51$，路径改为经过z，同时，y告知z 它到x的最小代价为无穷大，即$D_y(x) = \infty$

比较链路状态和距离矢量算法

通信复杂度

- **链路状态:** n个节点，E条链路，需要发送O(nE)条消息
- **距离矢量:** 在邻节点之间交换消息

收敛速度

- **链路状态:** $O(n^2)$ 算法需要发送O(nE)条消息
  - 可能震荡
- **距离矢量:** 收敛时间不定
  - 可能有路由环路
  - 无穷迭代问题

健壮性: 路由器故障

LS:

- 节点可能发布不正确的链路代价
- 每个节点计算自己的转发表

DV:

- 节点发布不正确的距离矢量
- 每个节点的距离矢量被其它节点使用
  - 错误在网络中传播

目录

路由的可扩展性

理想情况

- 所有路由器一样
- 网络是“平”的

… 实际并非如此

可扩展性:因特网上有亿万级别数量的目的地:

- 无法在路由表中记录到所有目的地的路由信息
- 仅仅交换路由信息就可以瘫痪网络

管理自治

- 因特网 = 网路构成的网络
- 每个网络希望控制自己网络上的路由

因特网可扩展路由

同一机构管理的路由器构成的网络被称为“自治系统”（AS），或“域”

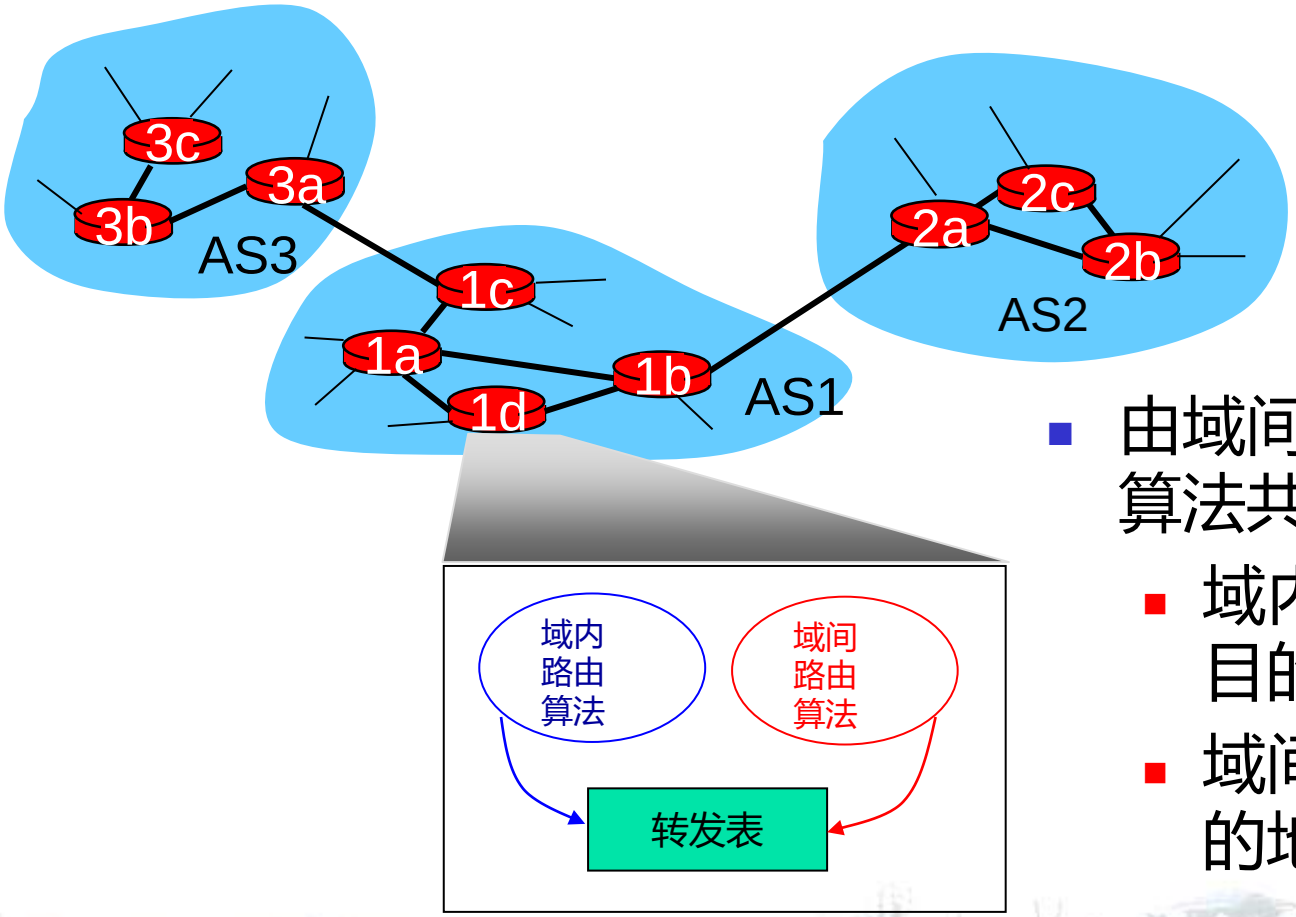
域内(intra-AS)路由

- 同一AS内部主机和路由器的路由
- AS中所有路由器必须运行同一种域内路由协议
- 不同的AS可以运行不同的域内路由协议
- 网关路由器：位于AS边界，通过链路与其它AS的网关路由器直连

域间（inter-AS）路由

- AS之间的路由
- 由网关路由器执行（它们同时也执行域内路由）

连接众多AS



- 由域间路由和域内路由算法共同生成转发表
  - 域内路由决定AS内部目的地的转发表项
  - 域间路由决定外部目的地的转发表项

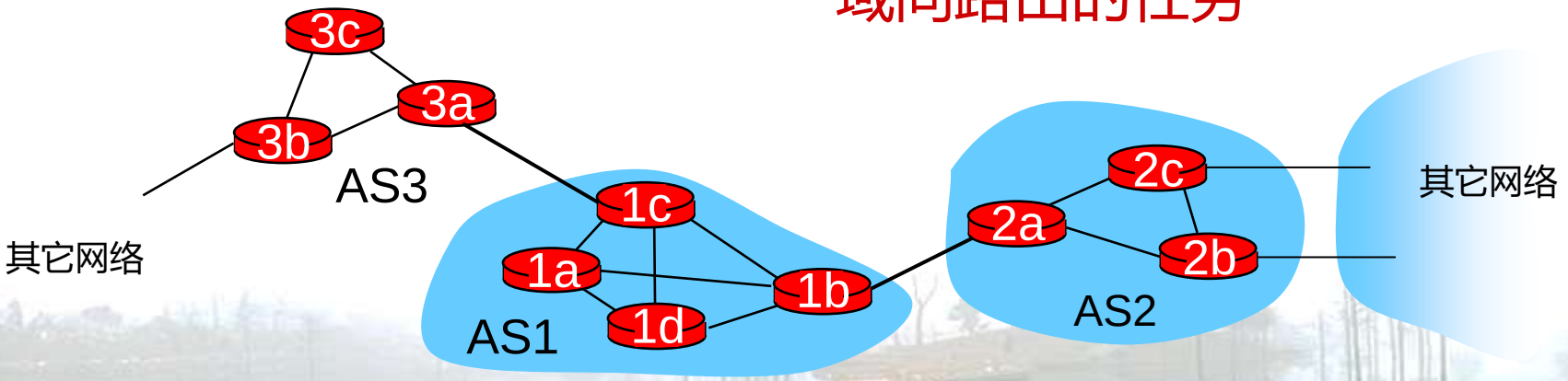
域间路由的任务

- 假设AS1中路由器收到目的地在AS1外的报文
  - 该路由器应该把报文转发给AS1的网关路由器，但是，具体给哪一个？

AS1必须:

1. 知道经过AS2可以到达哪些目的地址，经过AS3可以到达哪些?
2. 在AS1内传播可达性信息，让所有路由器都知道

域间路由的任务

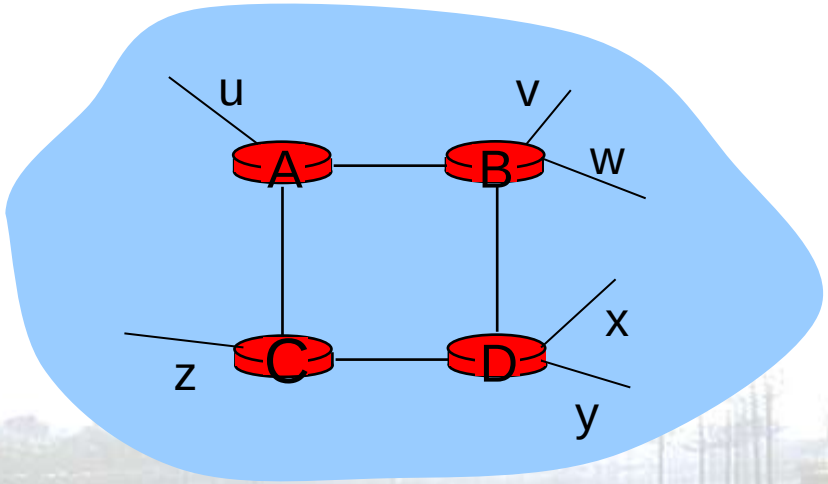


域内路由

- 又被称为内部网关协议 (IGP)
- 最常见的域内路由协议：
  - RIP: 路由信息协议
  - OSPF: 开放最短路径优先协议(IS-IS和OSPF协议基本一样)
  - IGRP: 内部网关协议 (2016年以前是Cisco私有协议)

RIP协议

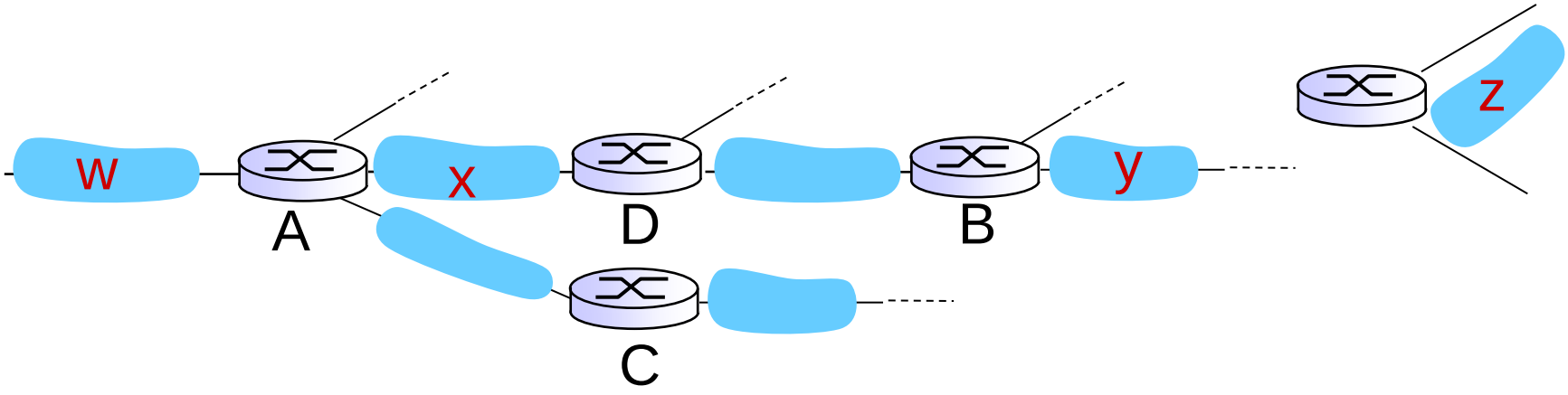
- 1982年开始被集成在BSD-UNIX中
- 距离矢量算法
  - 用跳数度量距离 (max = 15 跳)，相当于每条链路代价1
  - AS直径应小于15跳
  - 相邻节点每30秒通过RIP响应消息（又称通告）交换距离矢量
  - 每个通告包含到最多25个域内子网的距离



从路由器A 到目的地子网:

子网	跳数
u	1
v	2
w	2
x	3
y	3
z	2

RIP举例



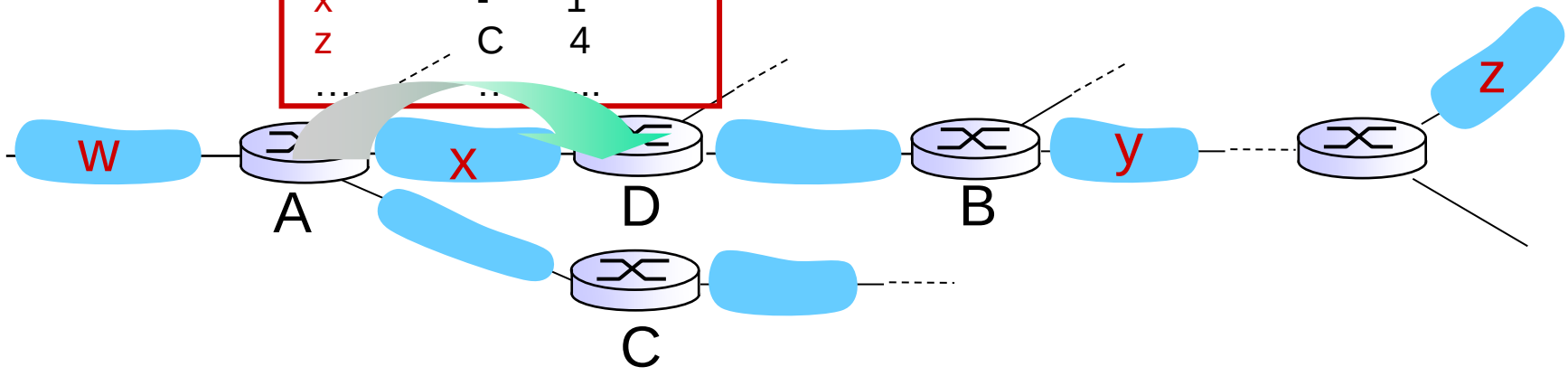
路由器D上的路由表

目的地子网	下一跳路由器	跳数
w	A	2
y	B	2
z	B	7
x	--	1
….	….	….

RIP举例

目的地	下一跳	跳数
w	-	1
x	-	1
z	C	4
…	…	…

A-到-D 通告



路由器D上的路由表

目的地子网	下一跳路由器	跳数
w	A	2
y	B	2
z	B → A	7 → 5
x	--	1
….	….	….

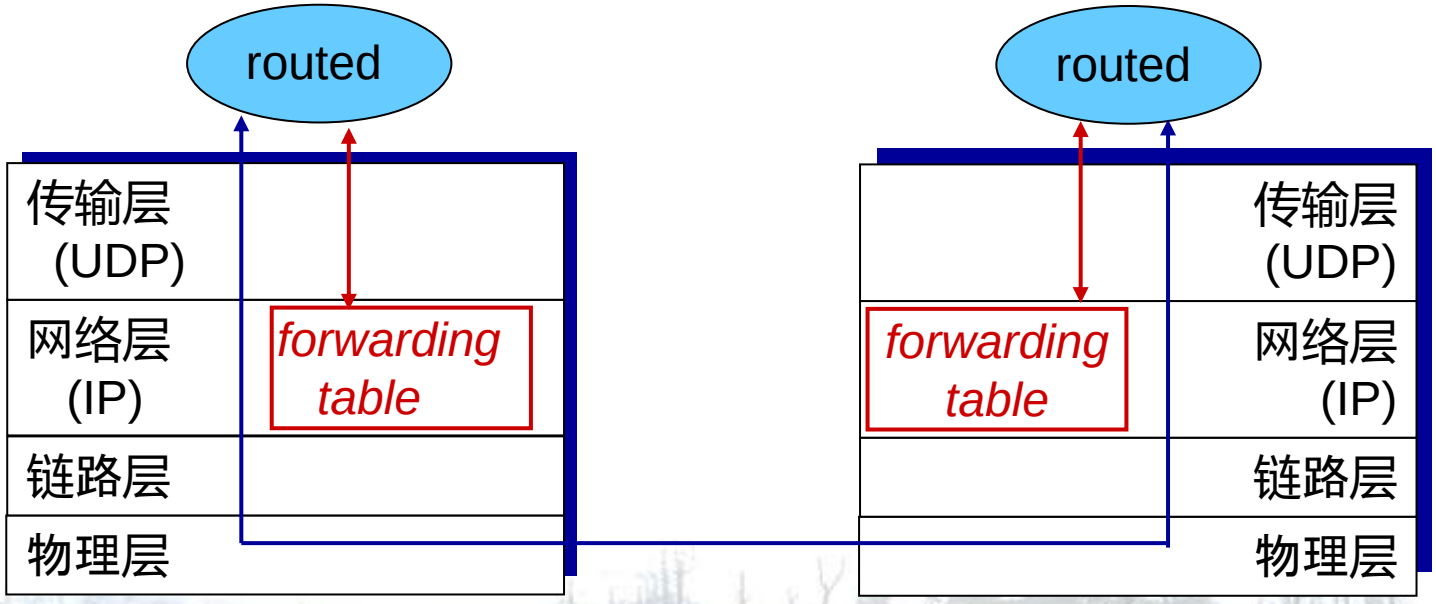
RIP：链路故障和恢复

如果180秒未收到通过→判定邻居路由器或链路故障

- 作废下一跳为该邻居路由器的路由
- 向所有邻居发送新的通告
- 如果邻居路由表变化，发回新的通告
- 网络规模小，故障信息快速传播到全网
- 反向下毒，阻止无穷迭代

RIP路由表进程

- UNIX的route-d守护进程维护RIP路由表
- 通过UDP在520端口上收发通告



目录

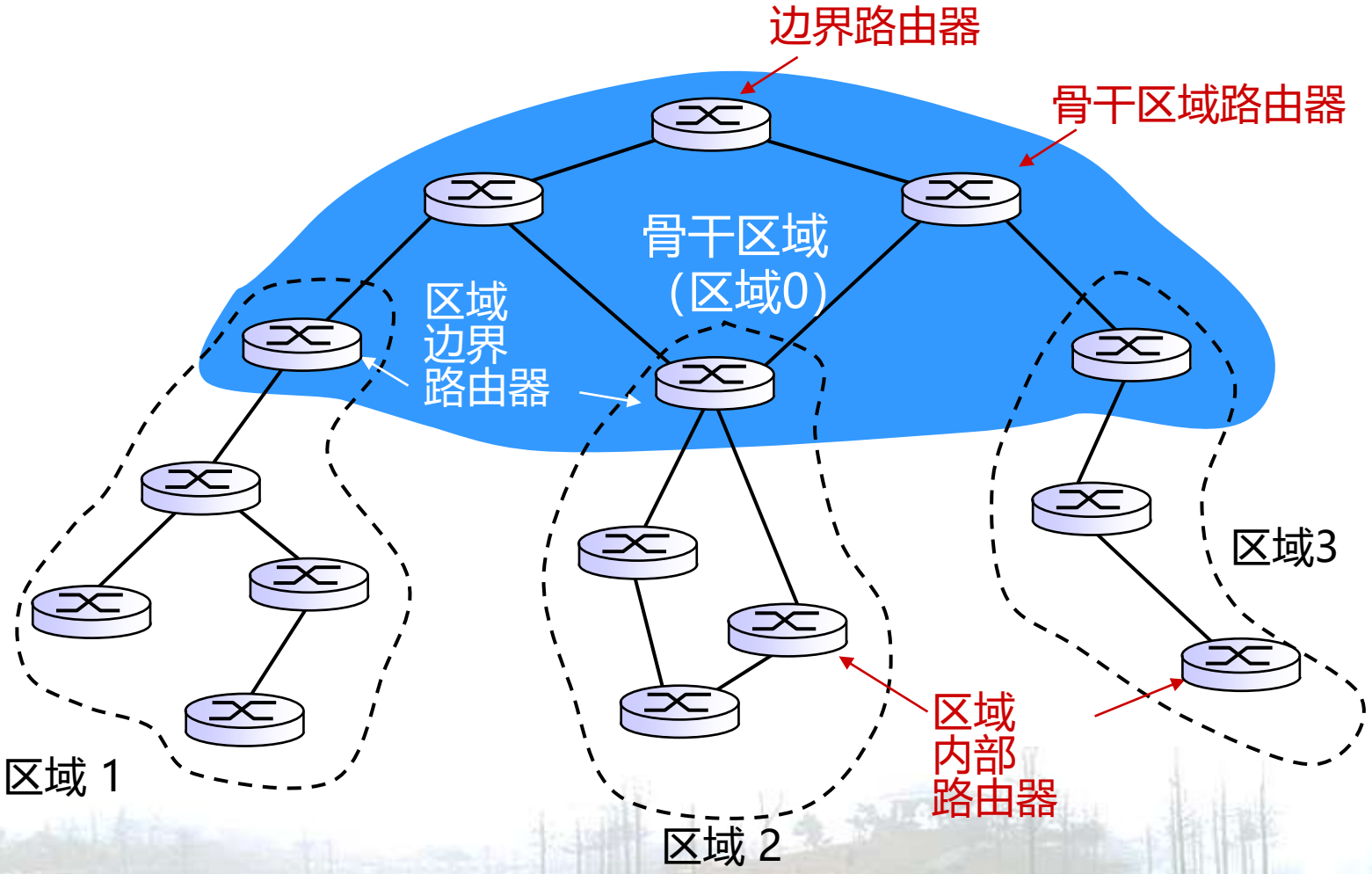
OSPF协议

- 公开协议：rfc 2328
- 使用链路状态路由算法
  - 发布链路状态数据包
  - 每个节点掌握网络拓扑
  - 使用Dijkstra算法进行路由计算
- 路由器在整个AS中向其它所有路由器泛洪OSPF链路状态通告
  - OSPF协议消息直接由IP报文携带（不使用TCP或UDP）
  - 链路状态：连接到路由器上每条链路的代价

OSPF高级特性

- 安全: 所有OSPF消息经过认证(避免恶意入侵者发布虚假链路状态)
- 允许多条等代价路径同时存在 (RIP协议仅计算一条)
- 每条链路上可以有多个代价指标，对应不同的服务类型 (带宽、时延等。例如，卫星链路在尽力而为的服务里代价高，在实时业务中代价低)
- 支持单播和多播:
  - 多播OSPF (MOSPF)使用和OSPF一样的网络拓扑信息
- 在大的网络域中分层

分层OSPF



分层OSPF

- 两层结构: 本地区域、骨干区域
  - 在区域内部泛洪链路状态通告
  - 每个节点有详细的区域内拓扑；仅知道通向其它区域的（到本区域边界路由器的）最短路径.
- 区域边界路由器: 综合本区域的距离信息，通告给其它区域边界路由器
- 骨干区域路由器: 在骨干区域内部运行OSPF.
- 边界路由器: 连接其它AS

比较RIP和OSPF

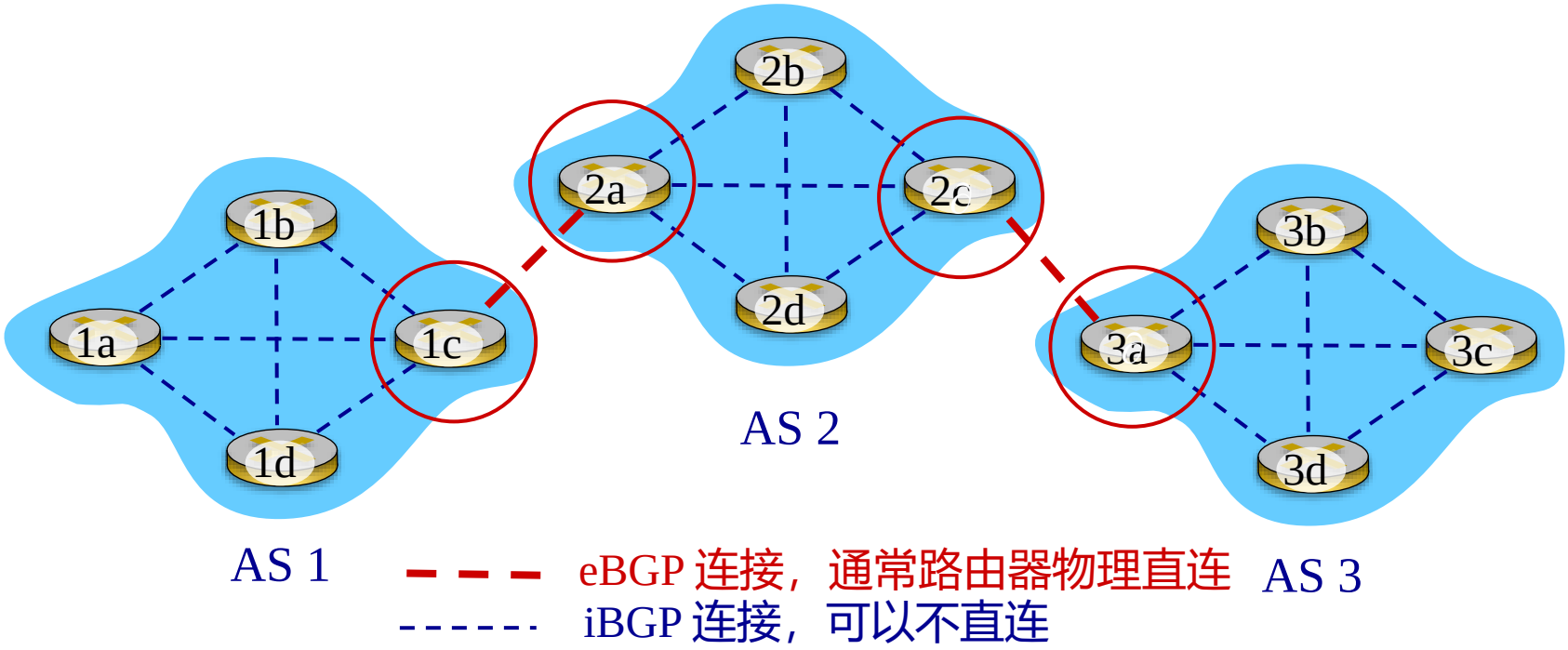
- RIP缺点
  - 不分区，管理网络规模小（直径不超过15跳）
  - 只能用跳数作为路径代价
  - 不支持多条等代价路径
  - 更新频繁
- OSPF缺点
  - 路由通告泛洪，代价高（链路状态路由协议固有问题）

目录

因特网域间路由：BGP

- BGP (边界网关协议): 事实上唯一的域间路由协议
  - 确保因特网是一个整体
- 路由器在179端口上建立半永久TCP连接，交换路由信息
- 每个AS的路由器可建立两类连接，执行两种任务
  - eBGP连接：从相邻AS获取子网可达性信息
  - iBGP连接：向AS内所有路由器传播外部的可达性信息
  - 通过可达性信息和策略决定通向外部目的地子网的“好”的路径
- 子网通过eBGP和iBGP向整个因特网通告自己的存在：“I am here”

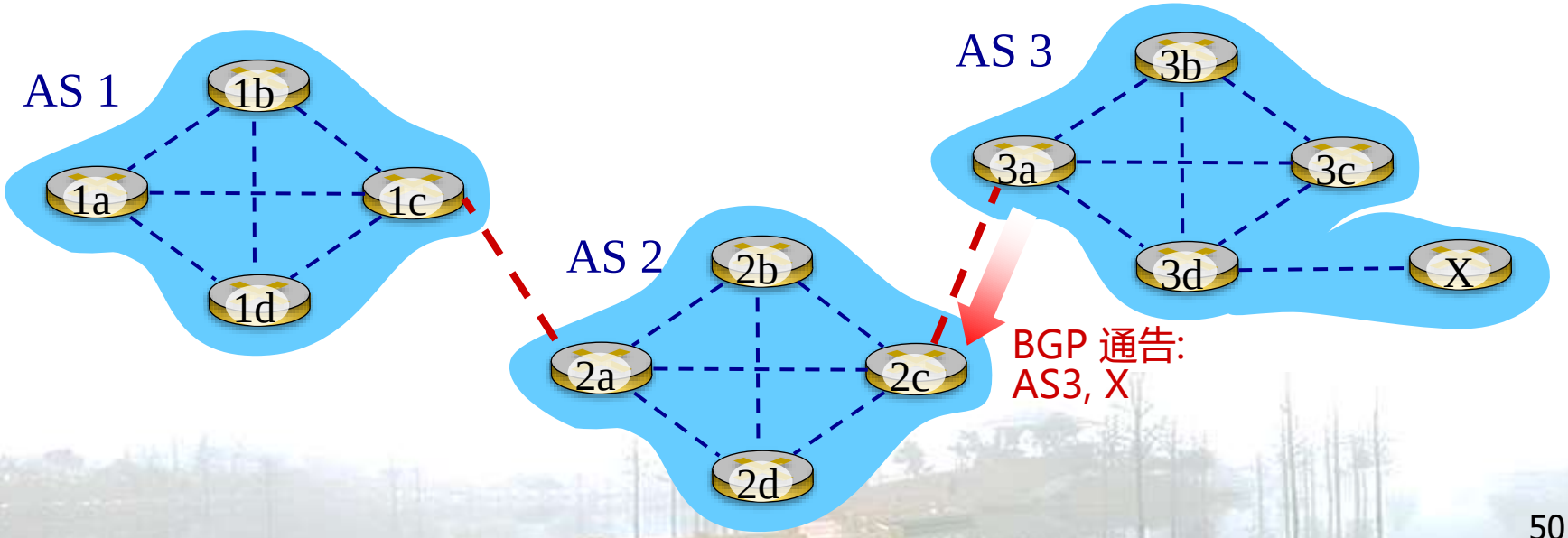
eBGP和iBGP连接



该网关路由器同时运行eBGP和iBGP

BGP基础

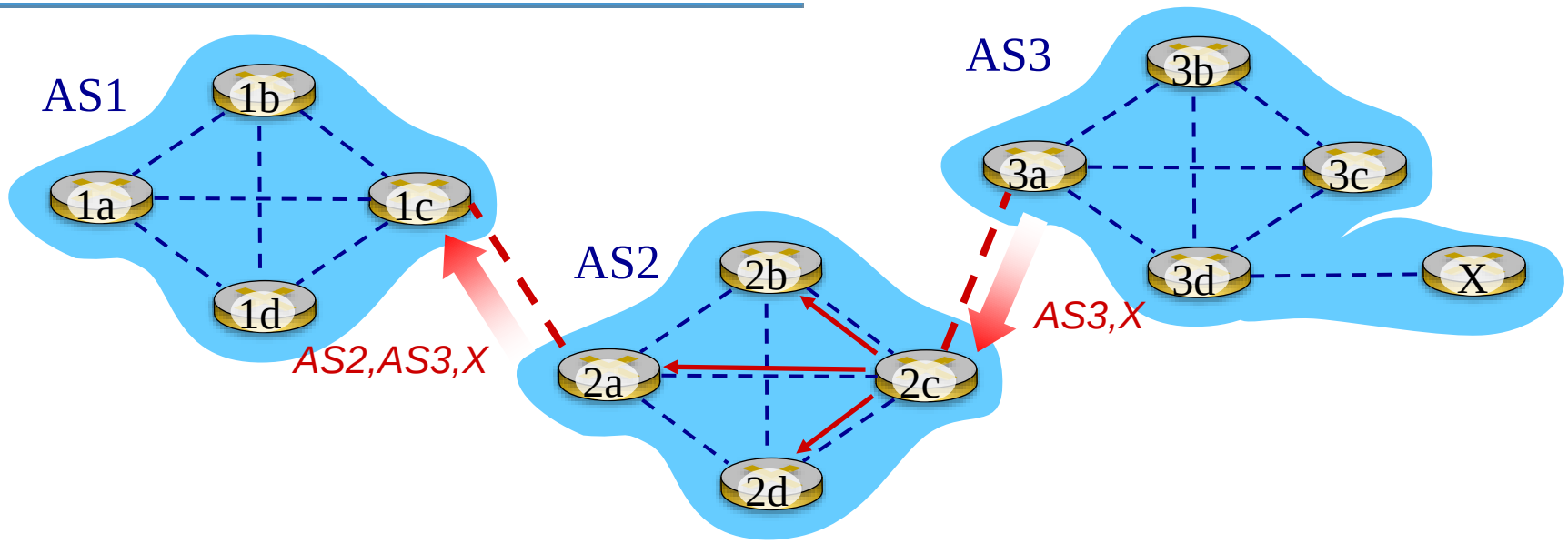
- BGP 会话：两个BGP路由器通过半永久的TCP连接交换BGP消息:
  - 通告通向各网段地址前缀的路径信息 (BGP是路径矢量协议)
- 当AS3的网关路由器 3a 向AS2的网关路由器2c 通告路径 AS3,X:
  - AS3向AS2承诺它可以转发目的地地址在X子网前缀内的报文



BGP路径和属性

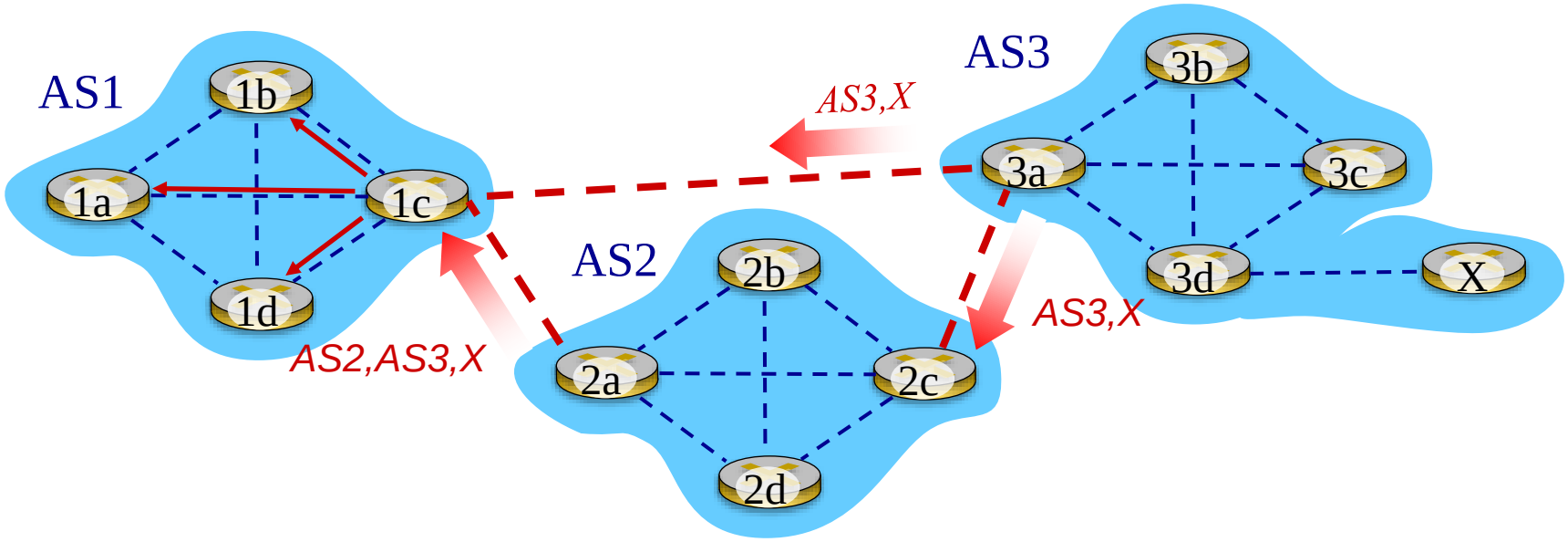
- 通告的路径信息
  - 子网前缀 + 属性 = “路由”
- 两个重要属性:
  - AS路径: 列出所有到达目的地子网需要经过的AS
  - 下一跳: 经过哪个AS内部路由器到达下一跳AS
- 策略路由:
  - 网关路由器使用策略选择接受或者拒绝路径 (例如，永远不通过AS Y路由).
  - AS的策略还决定是否向邻居AS通告路径的存在

BGP路径通告



- AS2路由器2c通过eBGP会话收到来在AS3路由器3a的路径通告AS3,X
- 基于AS2的策略，AS2路由器2c接受该路径AS3,X，通过iBGP会话将该路径传播到所有AS2的路由器
- 基于AS2的策略，AS2路由器2a通过eBGP会话向AS1的路由器1c通告 路径AS2, AS3, X

BGP路径通告



网关路由器可能获得到同一目的子网的多条路径信息:

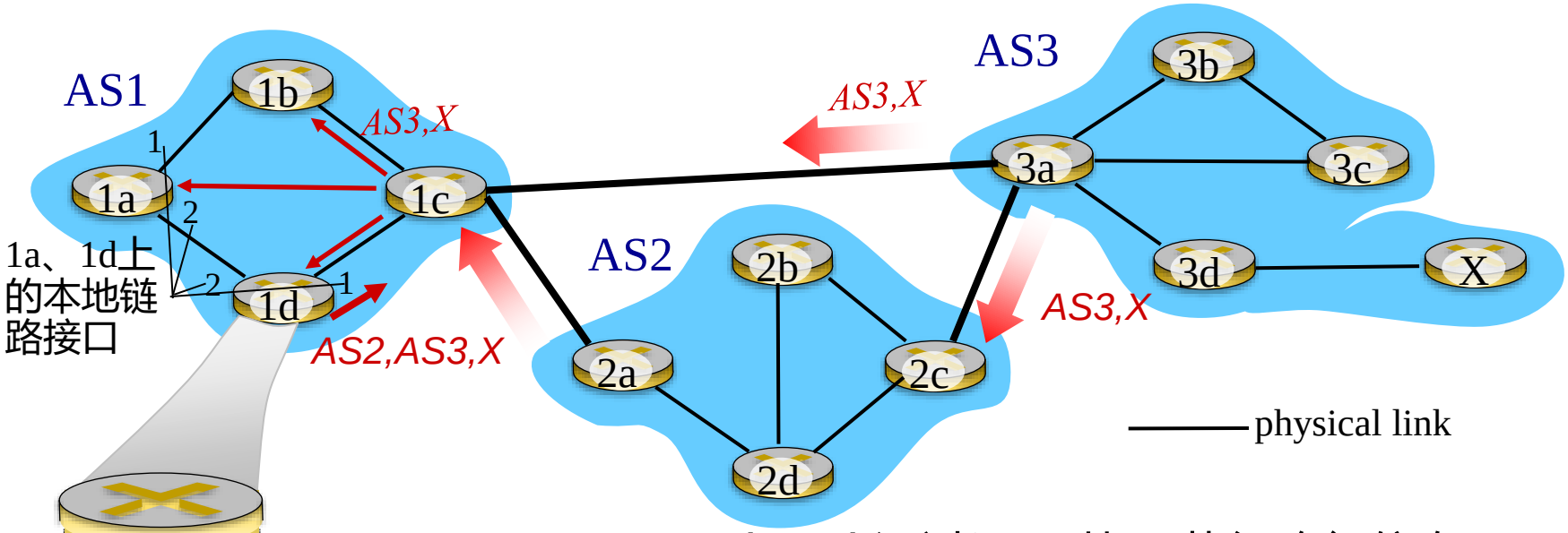
- AS1网关路由器1c从2a获得路径AS2,AS3,X
- AS1网关路由器1c从3a获得路径AS3,X
- 基于策略，AS1网关路由器1c选择路径AS3,X，并且通过iBGP在AS1内部通告

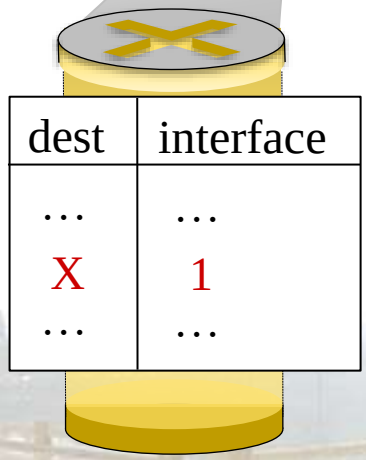
BGP消息

- 通过TCP连接交换
- BGP消息:
  - OPEN:与远端的BGP路由器建立TCP连接并认证对方
  - UPDATE: 通告一条新路径 (或撤回一条路径)
  - KEEPALIVE: 在没有交换UPDATES时保持连接；应答OPEN请求
  - NOTIFICATION: 通知前一条消息中的错误；关闭连接

BGP、OSPF和转发表项

问：路由器如何设置目的地为远端子网前缀的转发表项?

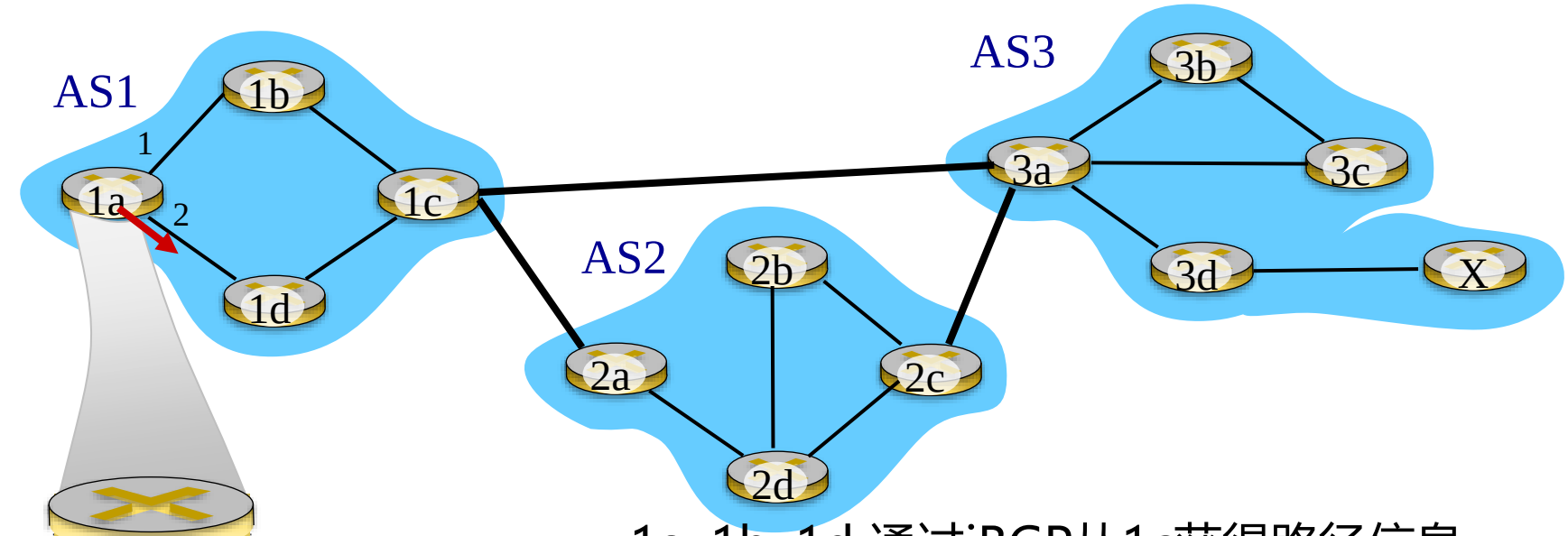


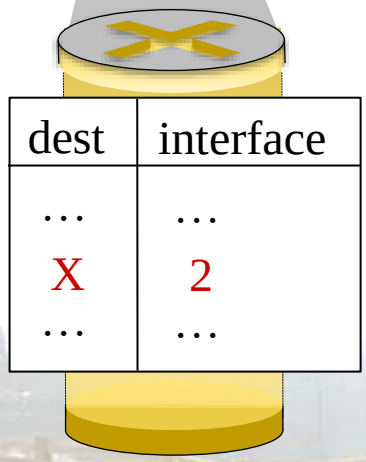


- 1a, 1b, 1d 通过iBGP从1c获得路径信息 “经过1c可到达X”
- 1d: 使用OSFP路由到1c, 从本地接口1转发数据包

BGP、OSPF和转发表项

问：路由器如何设置目的地为远端子网前缀的转发表项?



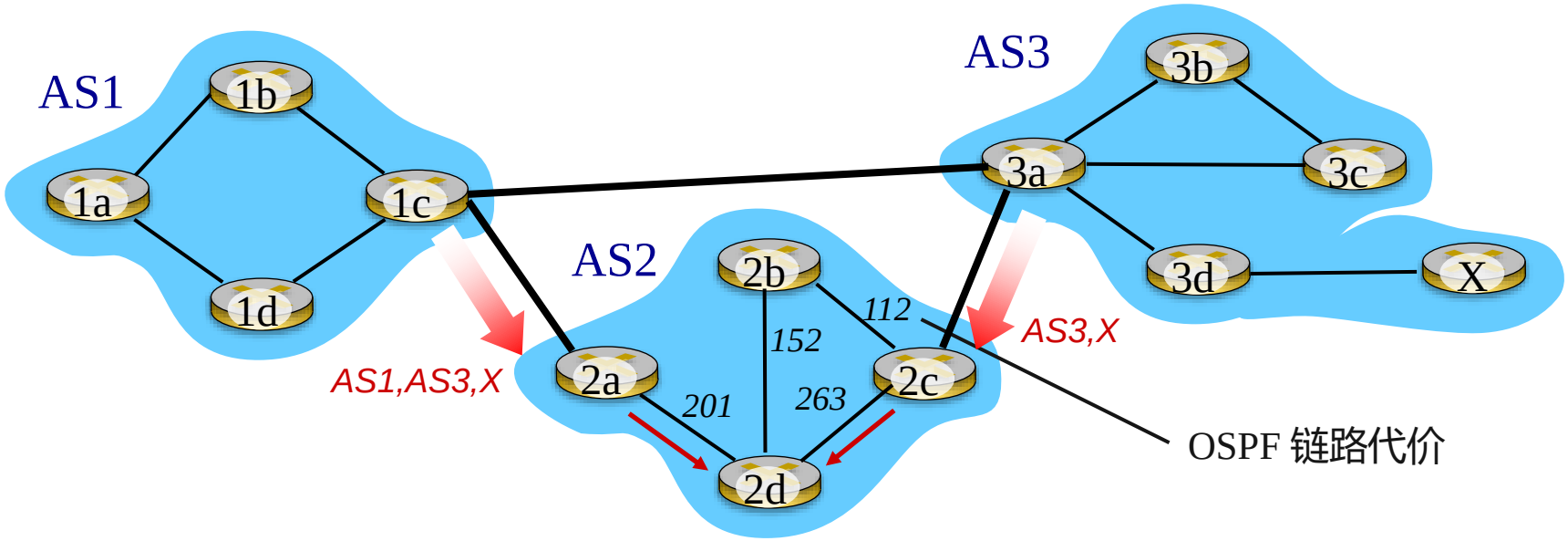


- 1a, 1b, 1d 通过iBGP从1c获得路径信息 “经过1c可到达X”
- 1d: 使用OSFP路由到1c, 从本地接口1转发数据包
- 1a:使用OSFP路由到1c, 从本地接口2转发数据包

BGP路由选择

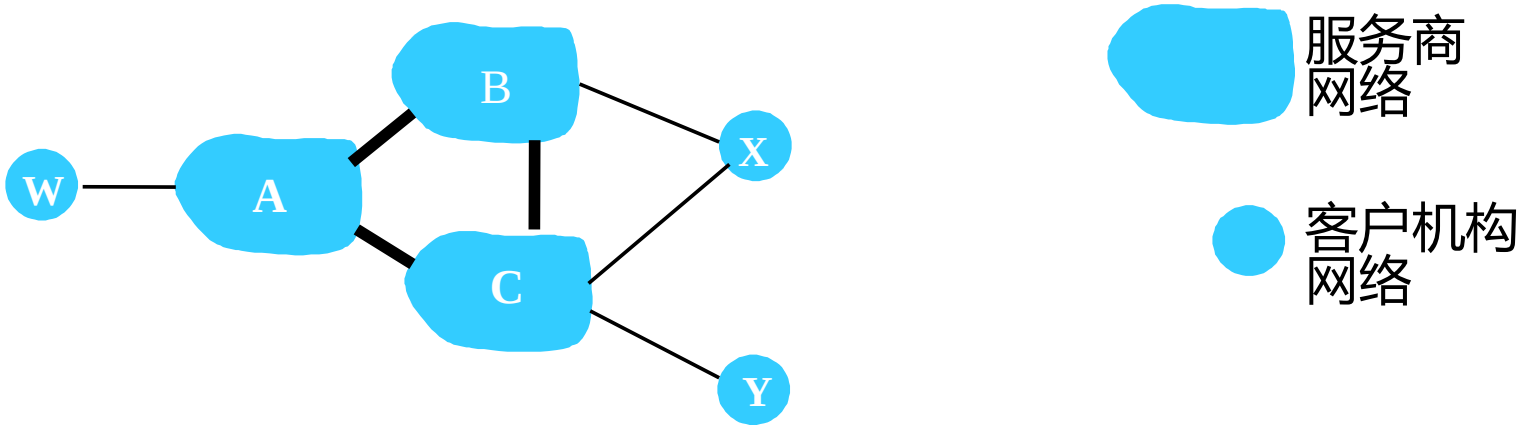
- 当存在到目的地的多条路径时，基于以下原则选择路径:
  1. 本地策略选择路由
  2. 选择AS-PATH最短的
  3. 运用热土豆策略选择到下一跳路由器代价最小的
  4. 其它

热土豆路由



- 2d通过iBGP从2a和2c获知经过2a和2c的到达X的两条路径
- 热土豆路由: 选择域内路由代价最小的本地网关路由器 (这里2d 选择 2a，即使该路由的AS跳数更多)：此时不考虑域间路由的代价

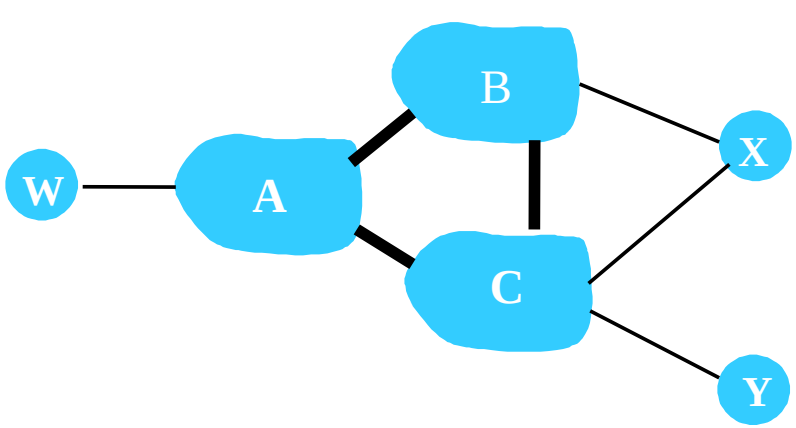
BGP：通过选择性通告实现策略



运营商只愿意承担顾客网络的流量，不希望承担过境流量

- A 向B和C通告路径 Aw
- B 选择不向C通告路径 BAw:
  - B无法从CBAw路径上赚钱，因为C和A都不是B的顾客
  - C不知道路径CBAw存在
- C 通过路径CAw路由到w

BGP：通过选择性通告实现策略



服务商网络

客户机构网络:

假设运营商只愿意承担顾客网络的流量，不希望承担过境流量

- X 连接两个服务商网络，B和C
- X不希望B通过X路由到C
  - X选择不向B通告路径XC

域间路由和域内路由的区别

策略:

- 域间路由：管理员希望通过策略控制哪些流量可以经过自己的网络转发
- 域内路由: 都是内部流量，无需策略

性能:

- 域内路由：关注路由性能（最短路径）
- 域间路由：策略优先（可能导致次优路径，例如，热土豆路由）

目录

ICMP: 因特网控制消息协议

- 用于主机和路由器交流网络层面的信息
  - 报错: 主机、网络、协议、端口不可达
  - Echo请求/响应 (ping)
- IP之上的网络层协议:
  - ICMP消息由IP报文携带
- ICMP消息: 类型、代码、和导致错误的IP报文的前8个byte

类型	代码	描述
0	0	echo响应(ping)
3	0	目的地网络不可达
3	1	目的地主机不可达
3	2	目的协议不可达
3	3	目的端口不可达
3	6	目的地网络未知
3	7	目的地主机未知
4	0	源抑制(未启用)
8	0	echo请求 (ping)
9	0	路由通知
10	0	路由器发现
11	0	TTL过期
12	0	损坏的IP头部

ICMP和traceroute

- 源主机向目的地IP发送一系列UDP分段
  - 第一个分段设置TTL =1
  - 第一个分段设置TTL =2，…
  - 任意不常用的端口号
- 当第n个分段到达路径上第n个路由器:
  - 路由器丢弃报文，向源主机返回ICMP消息 (类型 11, 代码 0)
  - ICMP消息包含路由器的名字和IP地址

何时停止?

- UDP分段到达目的地主机
- 目的地返回 “端口不可达” 的ICMP消息 (类型 3, 代码 3)
- 源停止发包