计算机网络

Computer Networks

网络互联：路由选择

提纲

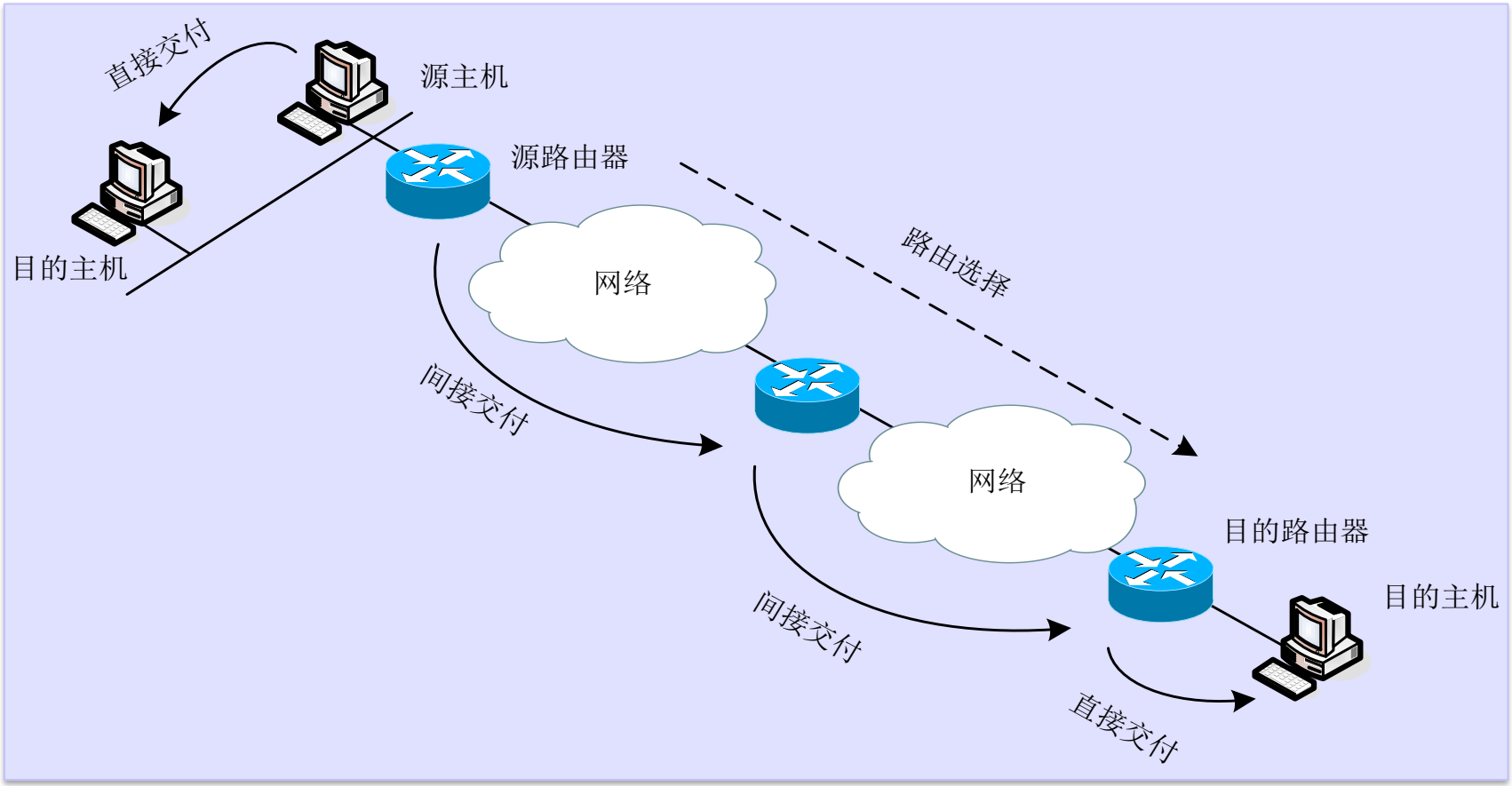
- 问题: 网络如何互联
  - 网络层和网络互联
  - IP协议
  - IP地址
  - IP附属协议
  - 分组交付与路由选择
    - 分组交付
    - 路由选择
    - 路由信息协议RIP
    - 最短路径优先协议OSPF
    - 边界网关协议BGP

提纲

- 问题: 网络如何互联
  - 网络层和网络互联
  - IP协议
  - IP地址
  - IP附属协议
  - 分组交付与路由选择
    - 分组交付
    - 路由选择
    - 路由信息协议RIP
    - 最短路径优先协议OSPF
    - 边界网关协议BGP

IP中的分组交付

- 直接交付与间接交付



IP中的分组交付

- 网络层的两大核心功能
  - 转发: 将路由器输入端口收到的分组从正确的输出端口发送出去
  - 路由选择: 决定分组从源节点到达目的节点的路径
    - 路由选择算法

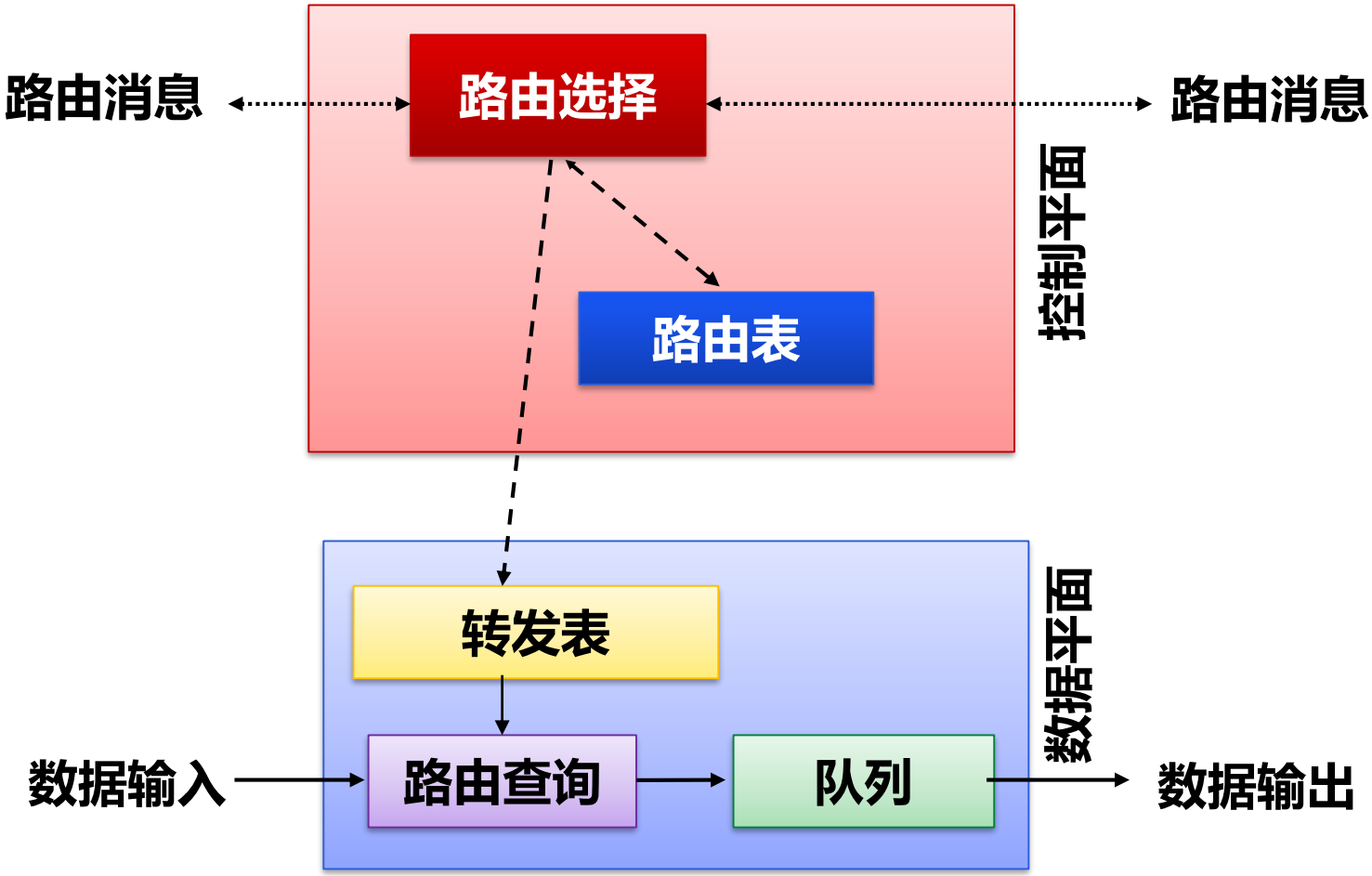
- 可以类比到实际生活中
  - 路由：策划一次从家出发的旅行
  - 转发：旅行途中每个中转站的换乘过程

转发 vs. 路由

- 转发(数据平面)
  - 过程: 获得一个分组, 查看其目的地址, 查询本地转发表, 将分组从输出端口发送出去
  - 节点(路由器/交换机)本地执行

- 路由选择(控制平面)
  - 转发表的构建过程
  - 通常由基于复杂分布式算法的路由协议完成

转发 vs. 路由



IP分组转发

- IP分组转发
  - 路由器将某一输入端口收到的数据从一个或多个接口输出的过程
  - 基于转发表
- 路由表
  - <NetNum, NextHop>
  - 如果不采用层次化寻址，路由表形式为：<IPAddr, NextHop>
- NextHop 信息通过路由算法获得

IP分组转发

- 主机节点

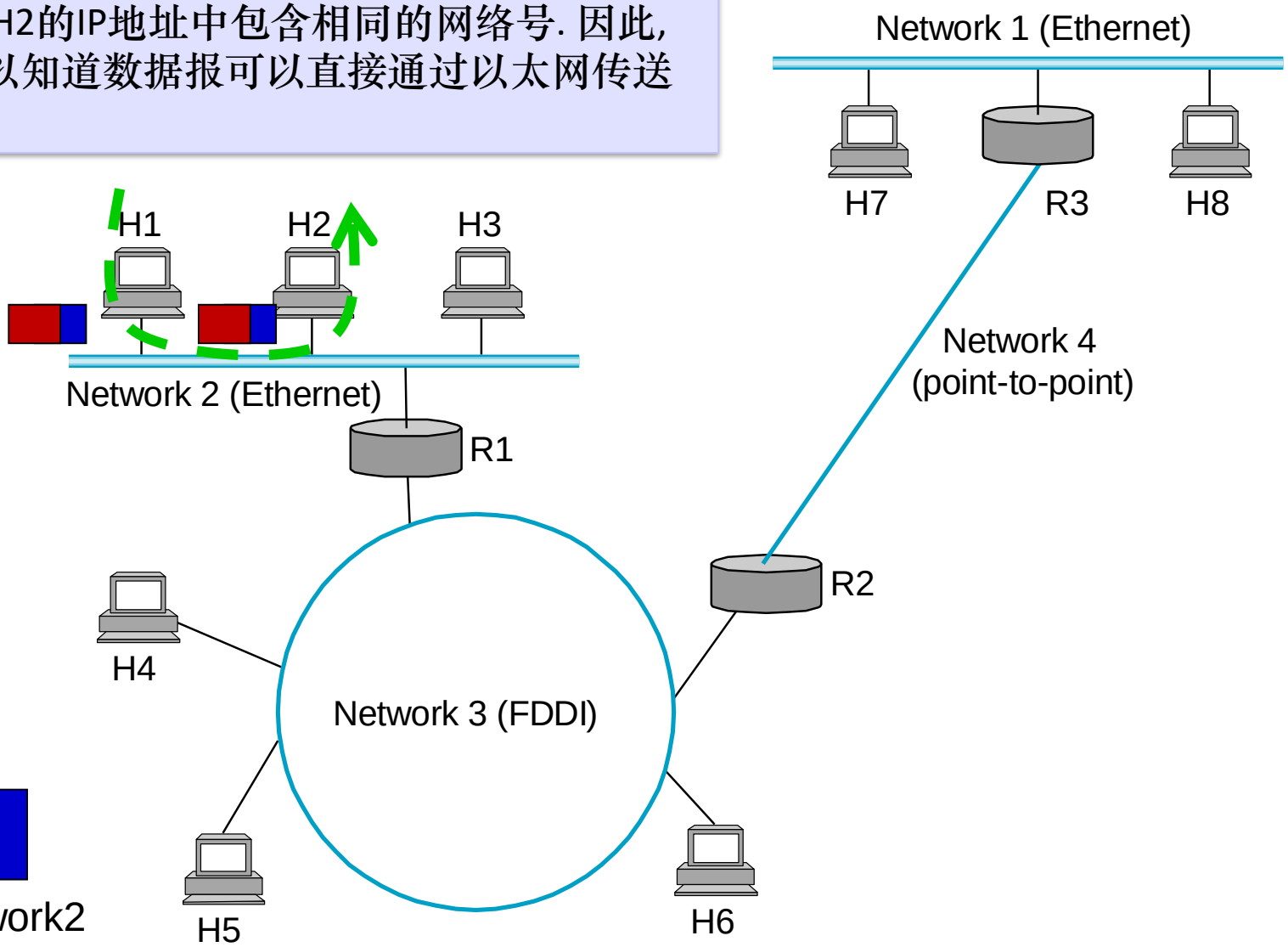
```
if (目的节点NetNum == 我的NetNum)
    直接传送分组至目的节点
else
    传送分组至缺省路由器
```

- 路由器节点

```
if (目的节点的NetNum == 本地其中一个接口的NetNum)
    经过该接口传送分组到目的节点
else
    if (目的节点的NetNum在本地的转发表中)
        传送分组至NextHop路由器
    else
        传送分组至缺省路由器
```

IP分组转发示例

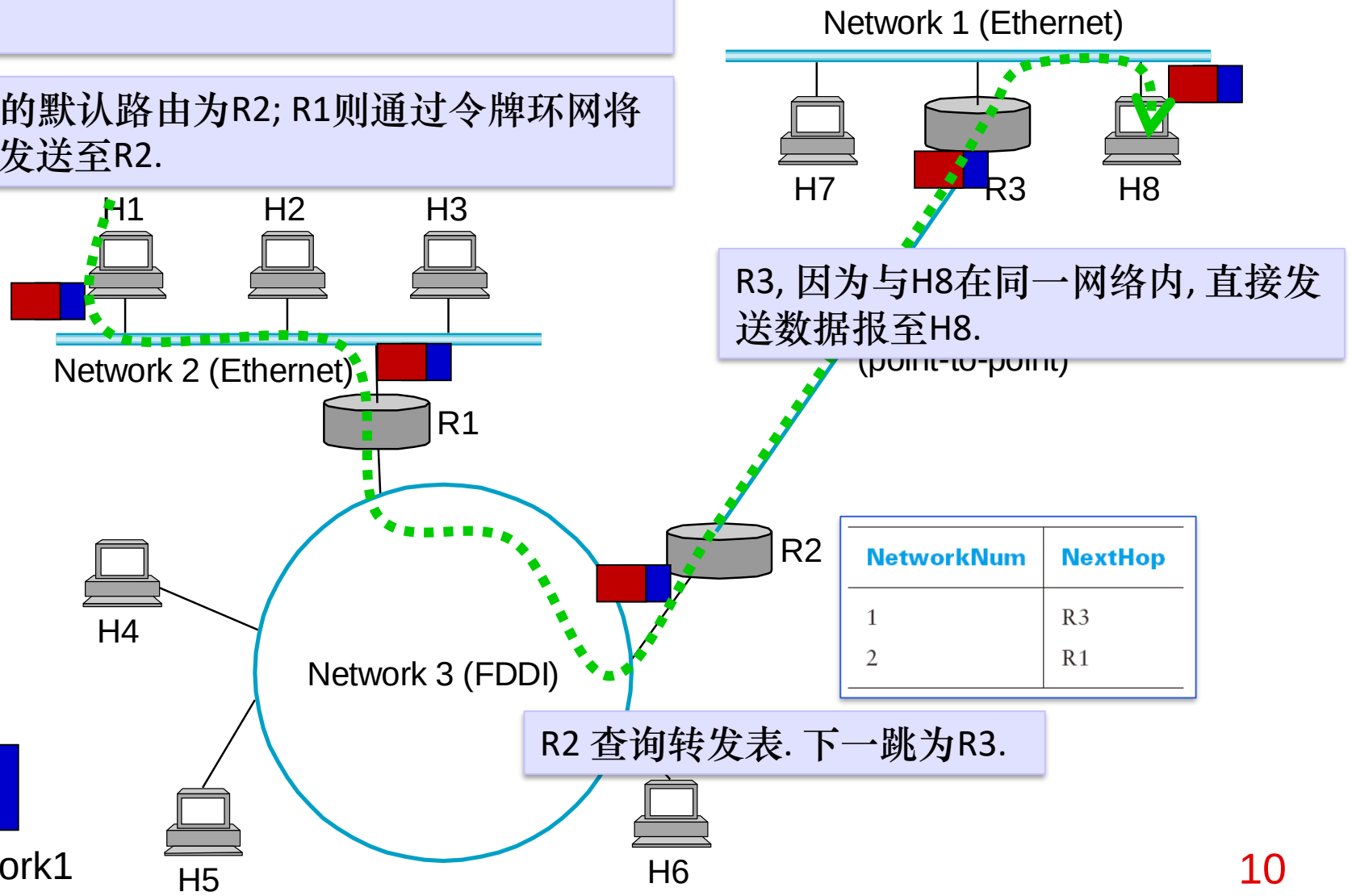
H1 和 H2的IP地址中包含相同的网络号. 因此, H1 可以知道数据报可以直接通过以太网传送至H2.



IP分组转发示例

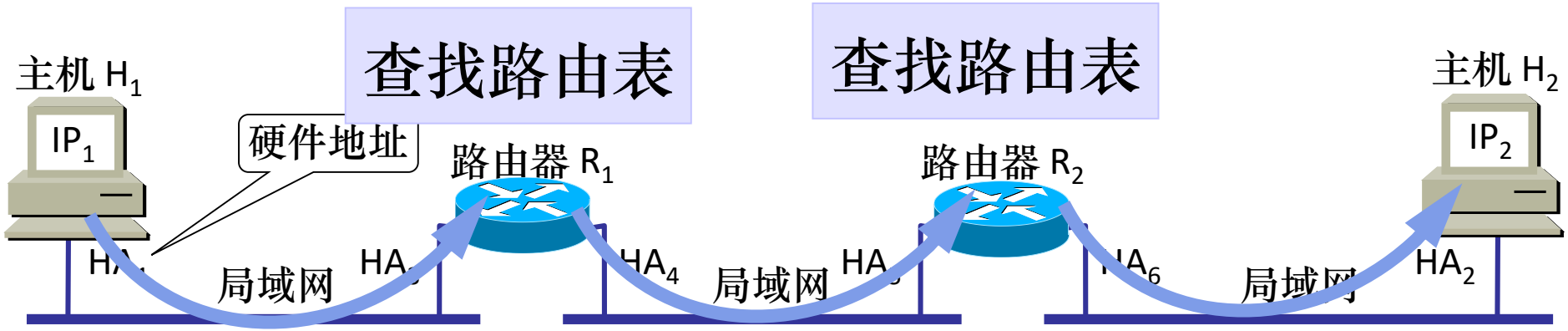
H1 可以知道数据报需要通过路由器转发, 则向R1传送

假设R1的默认路由为R2; R1则通过令牌环网将数据报发送至R2.



NetworkNum	NextHop
1	R3
2	R1

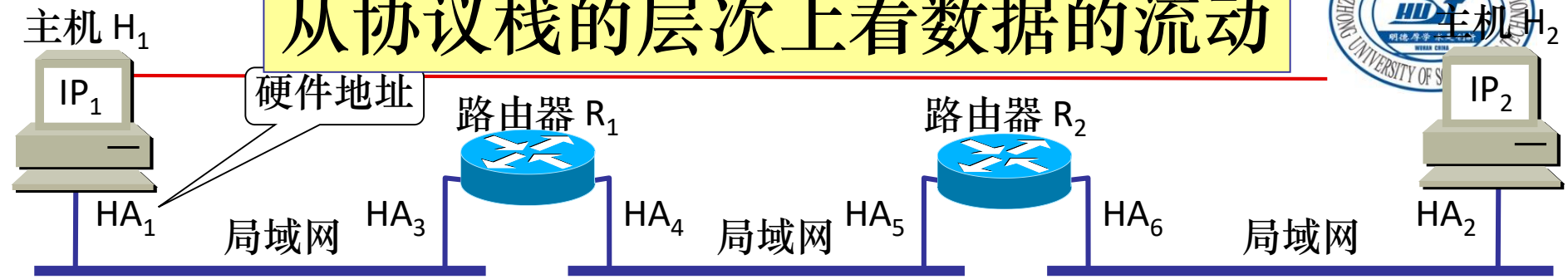
IP地址与数据链路层地址

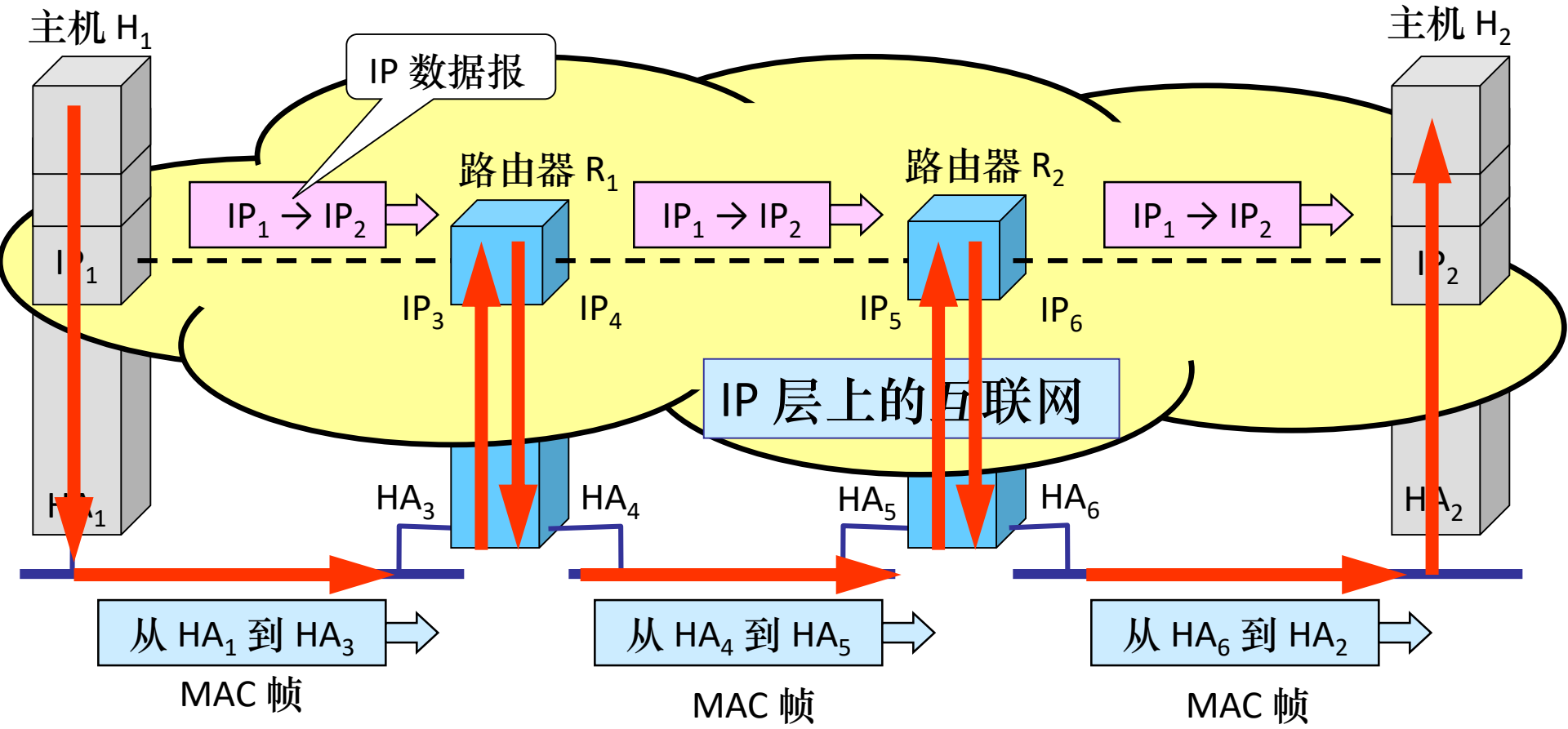


通信的路径

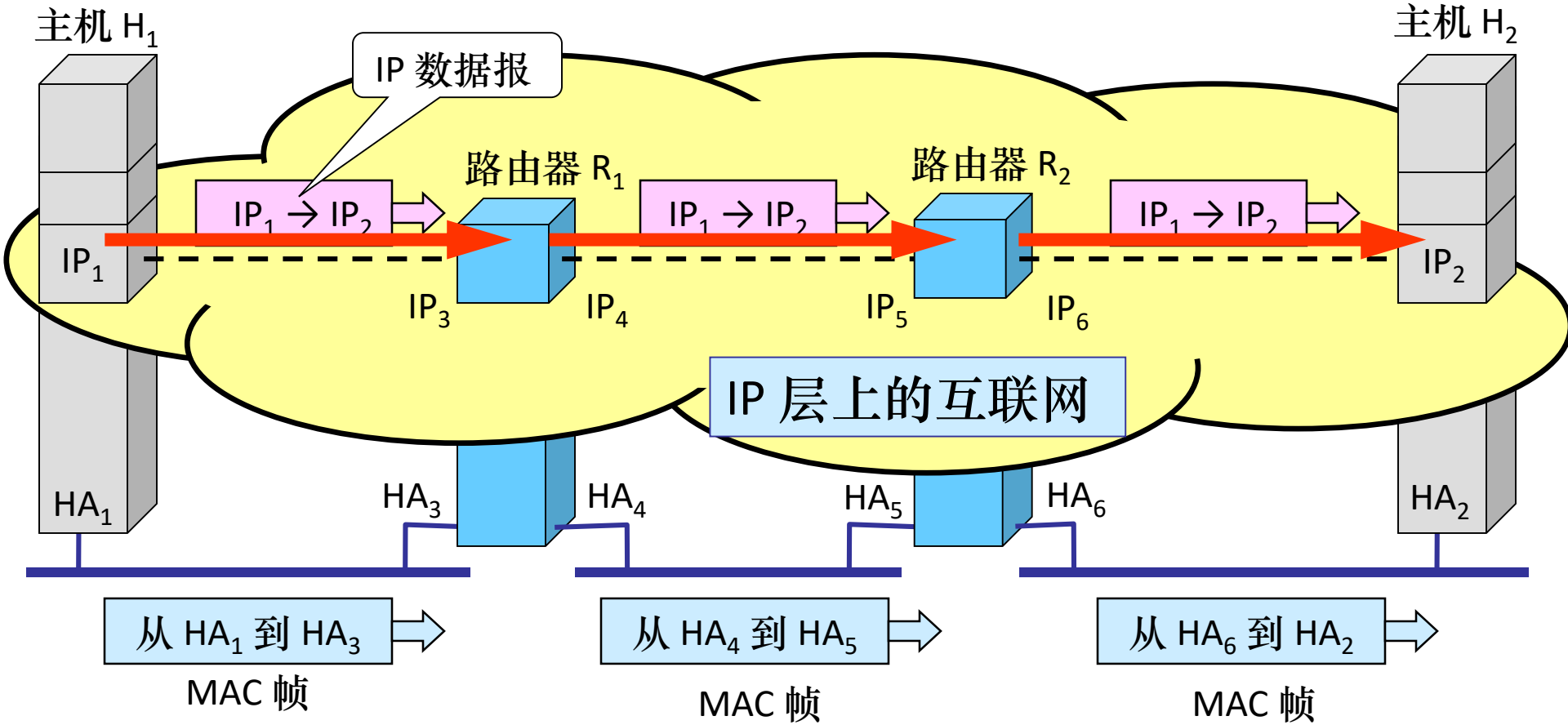
H_1→经过 R_1 转发→再经过 R_2 转发→H_2

从协议栈的层次上看数据的流动

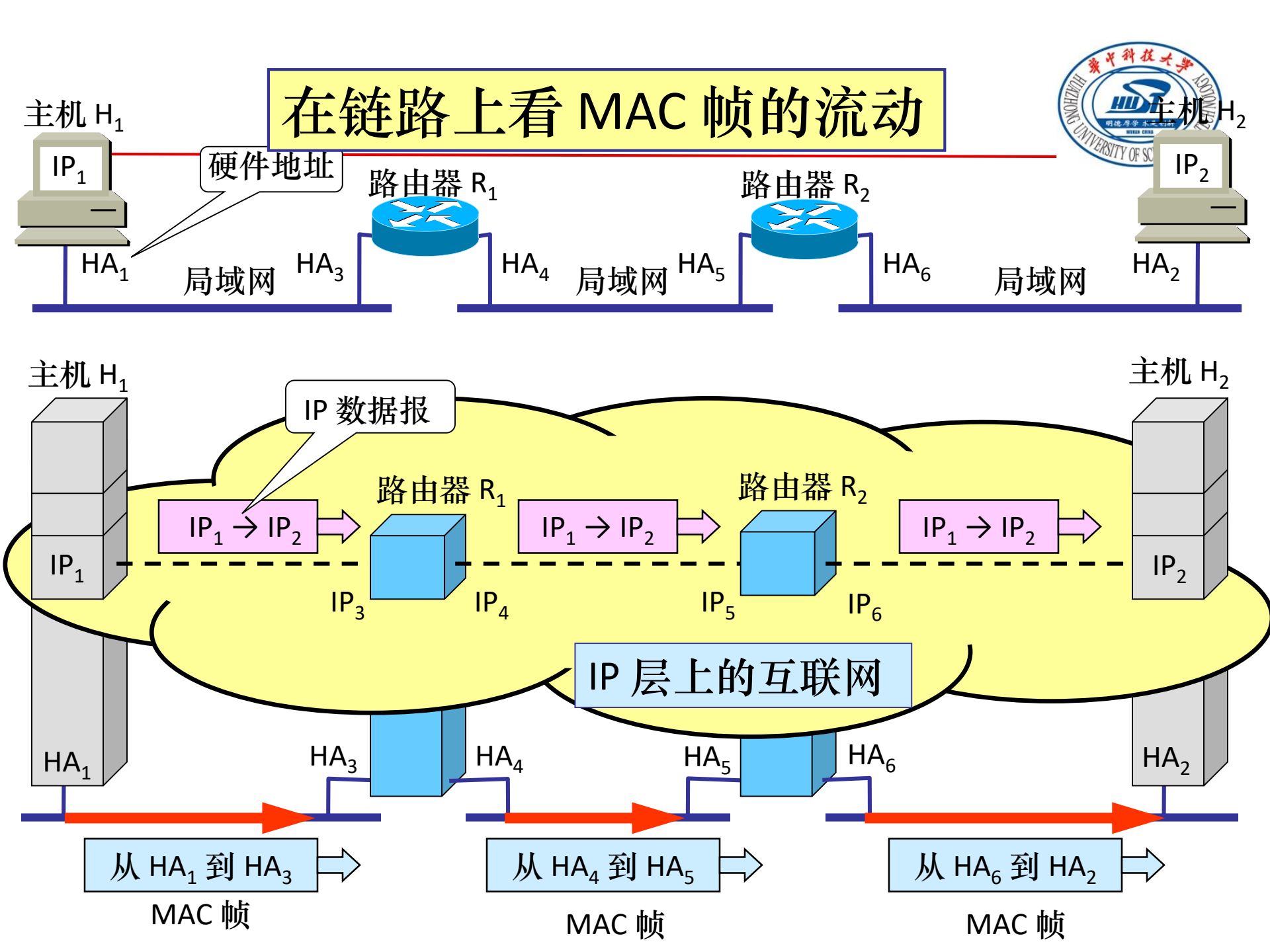




从虚拟的 IP 层上看 IP 数据报的流动



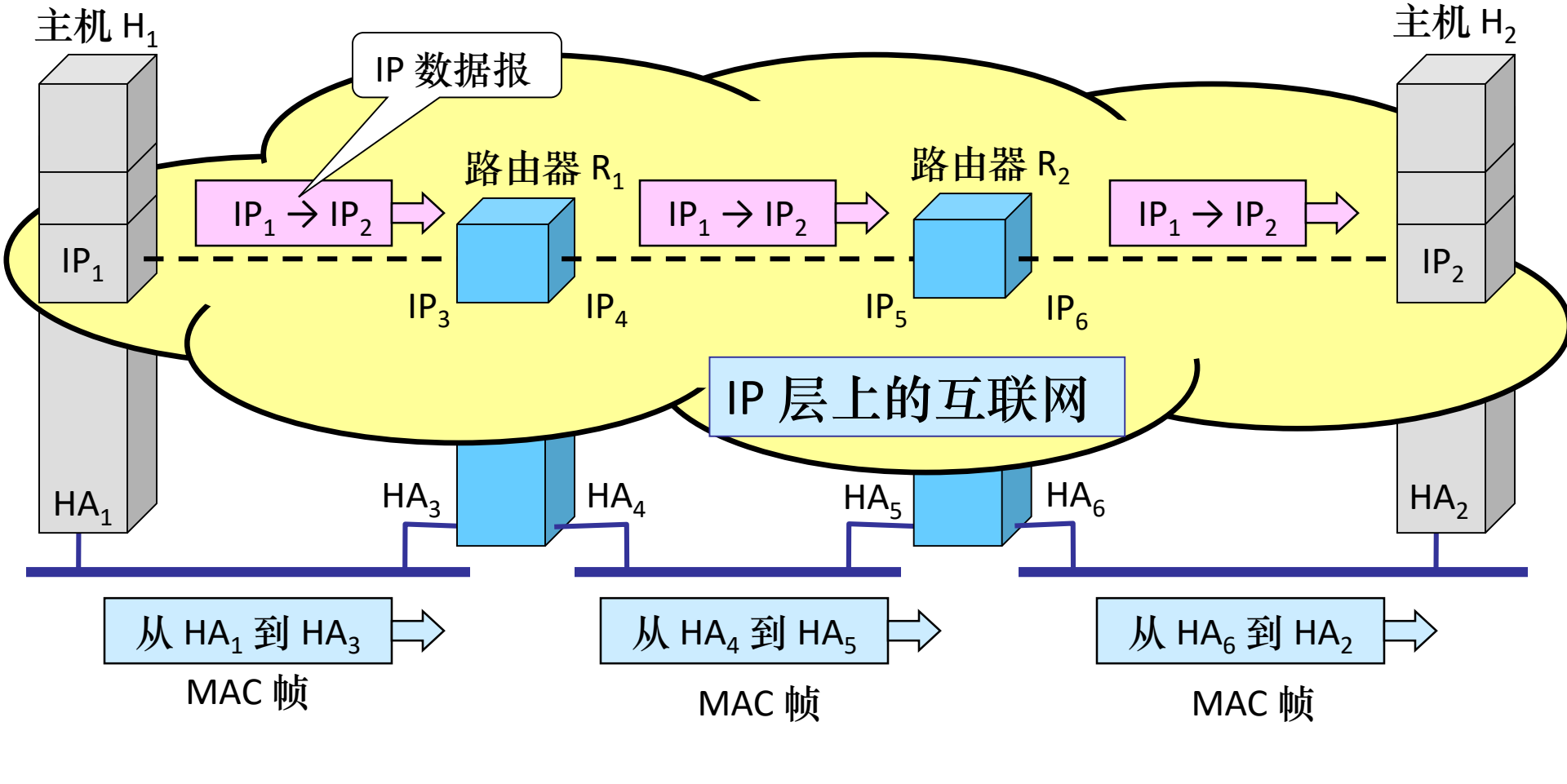
在链路上看 MAC 帧的流动



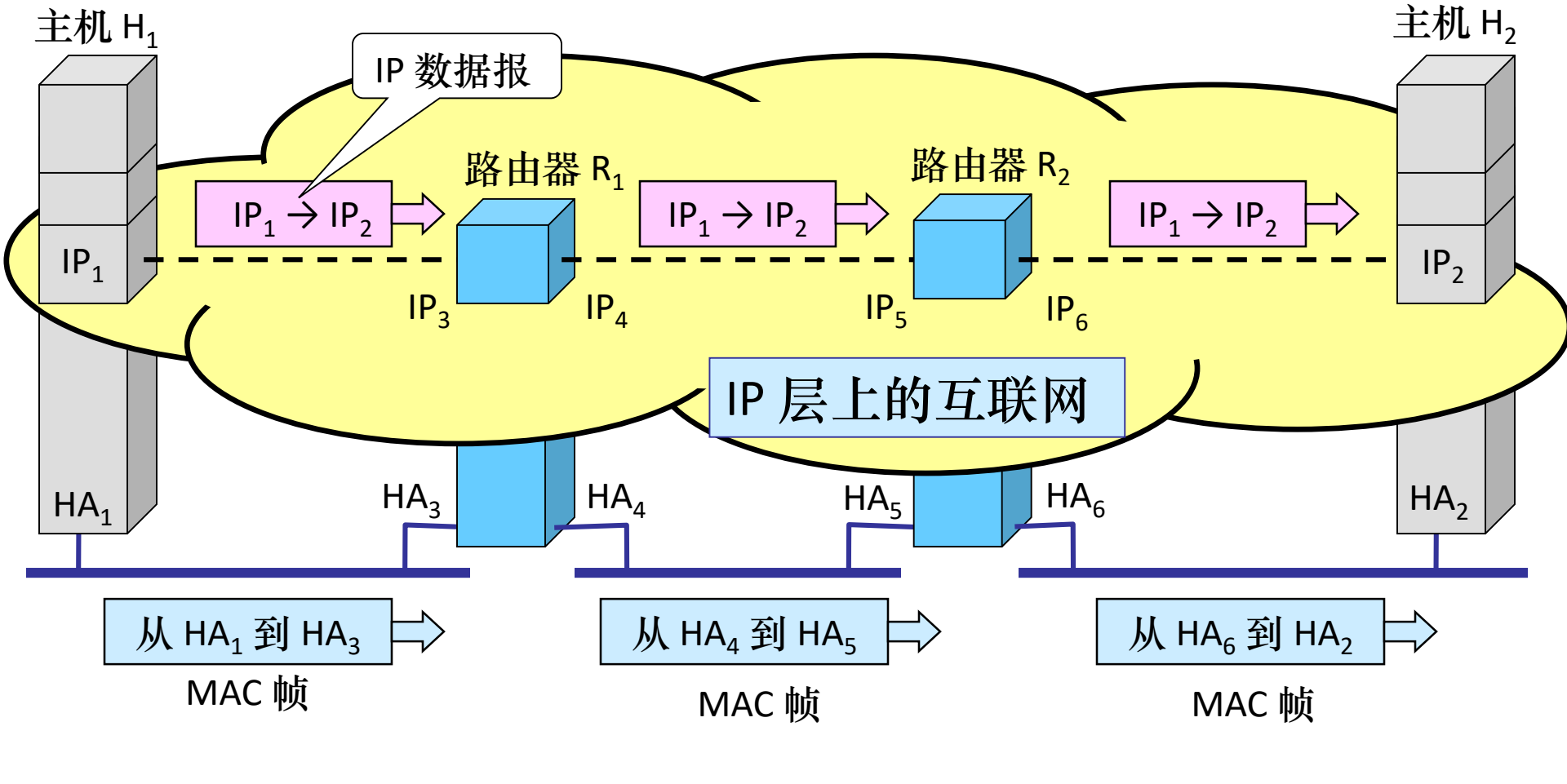
在 IP 层抽象的互联网上只能看到 IP 数据报

图中的 $IP_1 \rightarrow IP_2$ 表示从源地址 IP_1 到目的地址 IP_2

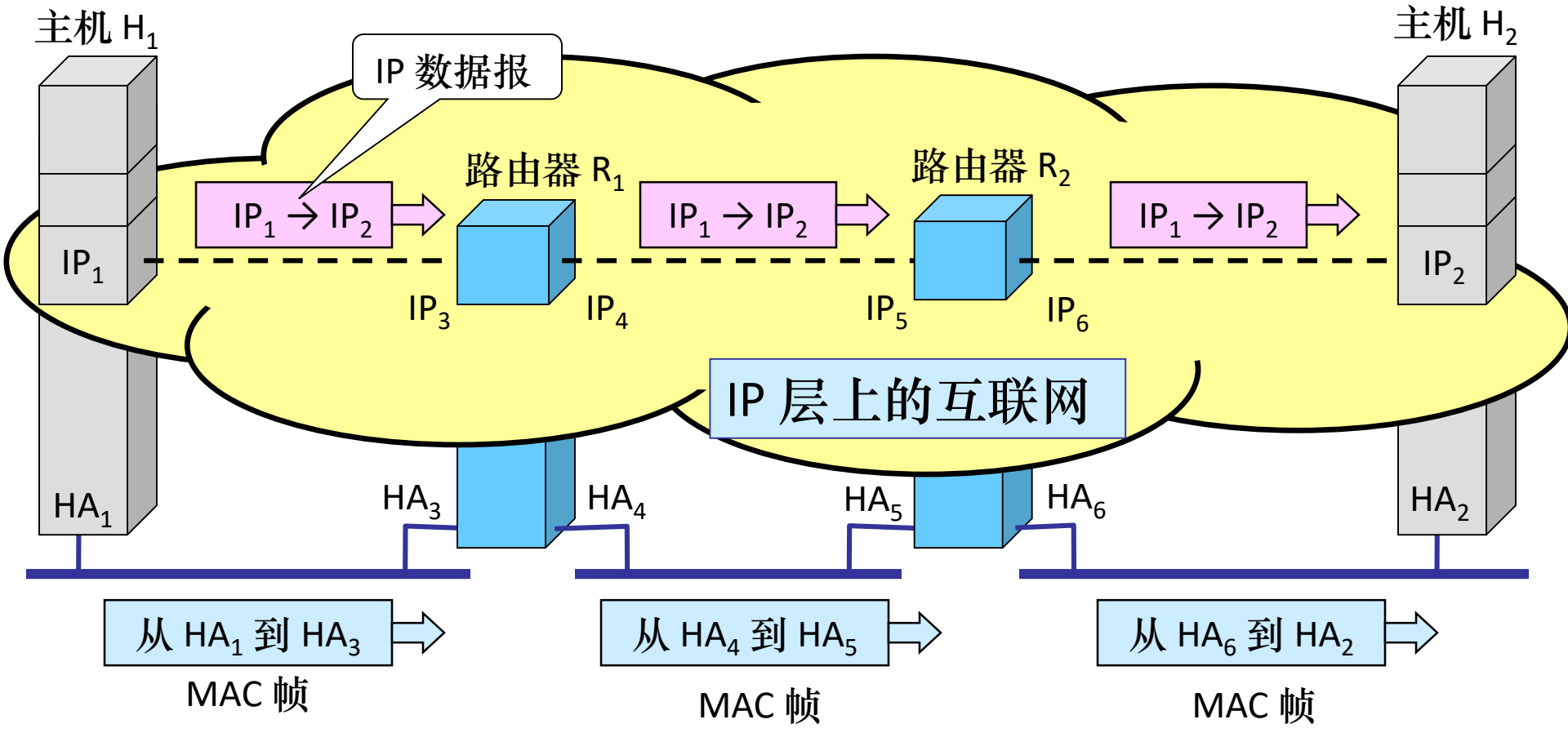
两个路由器的 IP 地址并不出现在 IP 数据报的首部中



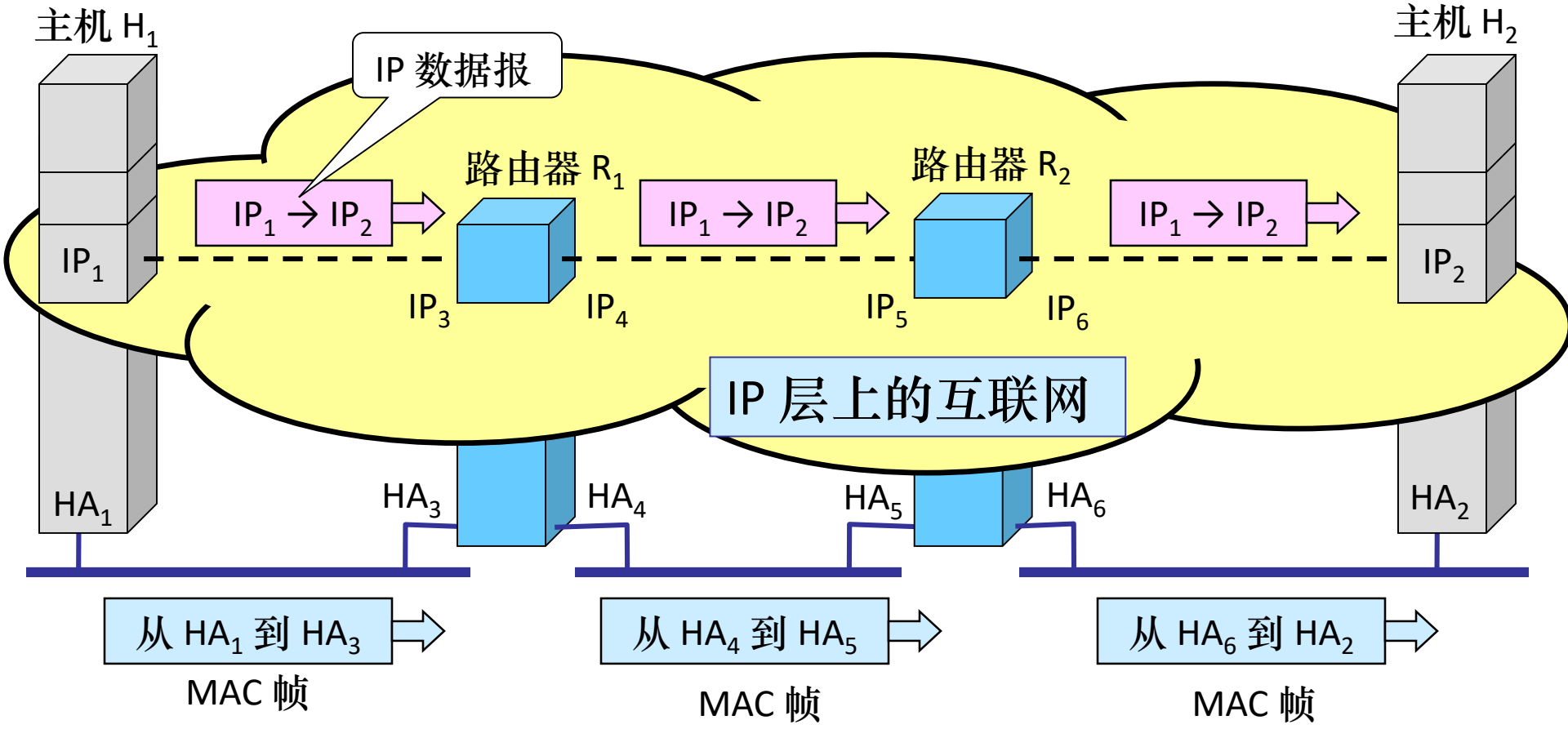
路由器只根据目的主机 IP 地址的网络号进行路由选择



在具体的物理网络的链路层
只能看见 MAC 帧而看不见 IP 数据报



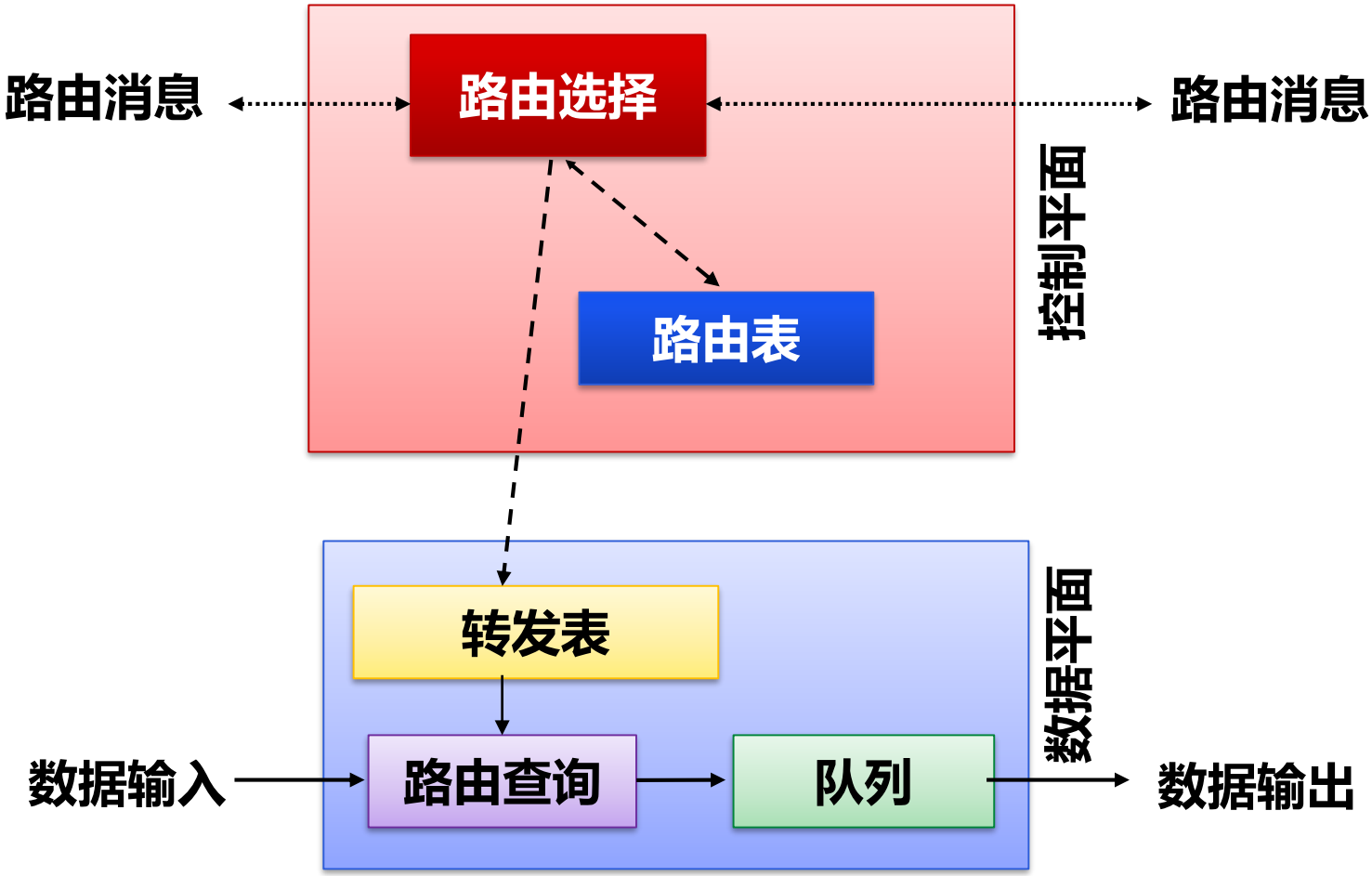
IP层抽象的互联网屏蔽了下层很复杂的细节
在抽象的网络层上讨论问题，就能够使用
统一的、抽象的 IP 地址
研究主机和主机或主机和路由器之间的通信



提纲

- 问题: 网络如何互联
  - 网络层和网络互联
  - IP协议
  - IP地址
  - IP附属协议
  - 分组交付与路由选择
    - 分组交付
    - 路由选择
    - 路由信息协议RIP
    - 最短路径优先协议OSPF
    - 边界网关协议BGP

转发 vs. 路由



路由表和转发表

❑ 路由表

- 由路由算法建立，进而生成转发表
- 包含从网络号到下一跳的映射

Network Number	NextHop
10	171.69.245.10

❑ 转发表

- 分组转发时使用转发表，表中需要包含足够的信息完成转发功能
- 包括从网络号到输出接口的映射以及一些MAC信息，例如下一跳的以太网地址

Network Number	Interface	MAC Address
10	if0	8:0:2b:e4:b:1:2

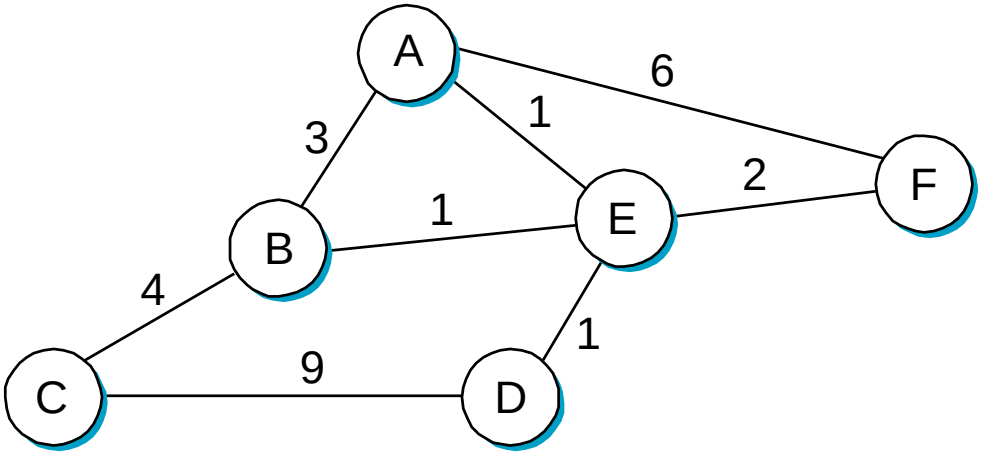
本节说明

- 本节讨论小到中型网络的路由选择问题, 而非整个Internet的.

- 本节讨论的路由选择
  - 解决方案可扩展? NO
  - 为中等规模的网络而设计(数百个结点的规模)
  - 本节讨论的协议通常称为域内路由协议, 或内部网关协议 (IGPs)

用图表示网络

- 网络可以用图表示
  - 节点: 路由器
  - 边: 链路
    - 边的代价: 成本, 时延, ...



路由选择本质上是图论中寻找两个节点之间较低成本的路径的问题

图的抽象

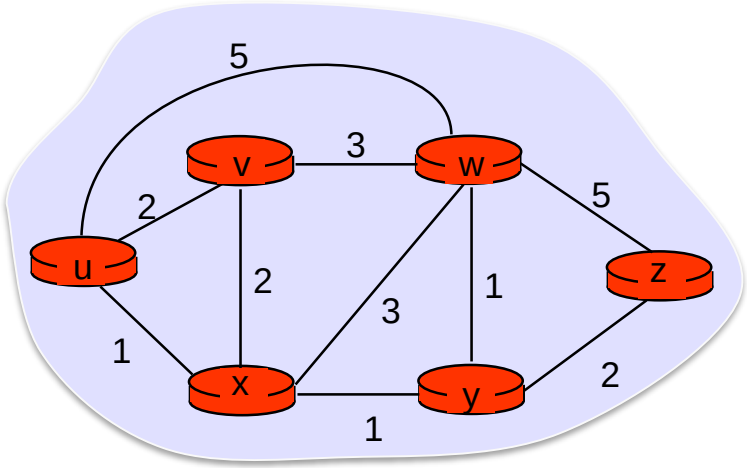
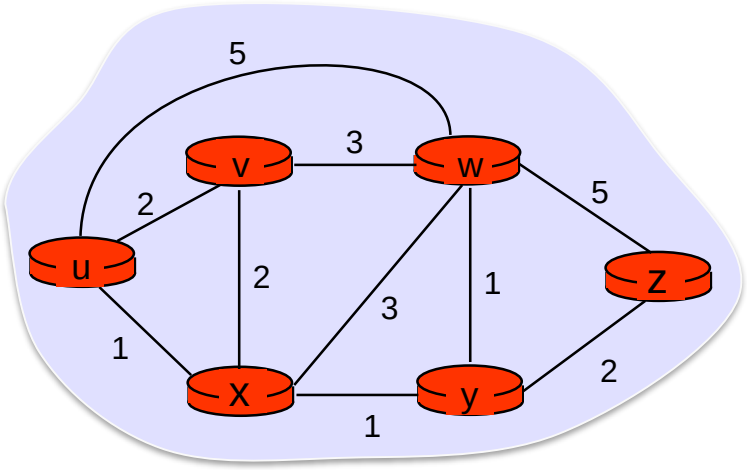


图: G = (N,E)

N = 路由器的集合 = { u, v, w, x, y, z }

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

图的抽象: 代价



- c(x1,x2) = 链路的代价(x1,x2)
  - 例如, c(w,z) = 5
- 代价通常取值为1, 或者与带宽成反比, 或者与拥塞成反比

路由的代价(x1, x2, x3,..., xp) = c(x1,x2) + c(x2,x3) + ... + c(xp-1,xp)

问题: 哪一条路径是节点u和z之间的最小代价路径？

解决方案: 路由选择算法, 寻找代价最小的路径

最短路由

- 用广义的距离(或代价)概念代表边的权值
- 图 $G = (V,E)$, 其权值函数为 $W: E \rightarrow R$ (为每一条边指定一个实数值)
- 路径的权值 $p = \rightarrow v_2 \rightarrow \cdots \rightarrow v_k$ is

$$w(p) = \sum_{i=1}^{k-1} w(v_i, v_{i+1})$$

- 最短路由 = 拥有最小权值的路径
- 应用
  - 静态/动态网络路由选择
  - 机器人动作规划
  - 交通地图/路径生成

最短路径问题

- 图论中的最短路径问题
  - 双向, 用实数值作为权值
  - 更加普遍的问题
- 两种算法
  - Bellman-Ford算法
    - 顶点对之间交换信息
  - Dijkstra算法
    - 全局交换信息
- 实际上, 网络互联中使用的协议更加简单

路由协议

- 运行于路由器
  - 基于路由选择算法
  - 挑战: 分布式, 动态, …
- 包括两个部分
  - 拓扑信息分发
  - 路由计算
- 类型
  - 距离向量（域内，RIP）
  - 链路状态（域内，OSPF）
  - 路径向量（域间，BGP）

拓扑信息分发

- 如何获得拓扑信息
  - 当一个节点启动或与其他节点通过一条新的链路互联时，节点通告其连接信息
  - 节点发现其邻节点或链路
    - 通过控制分组持续探测
    - 意识到最近的若干个更新周期内均未收到预期的定期更新拓扑信息
- 什么时候分发拓扑信息
  - 周期性分发
  - 触发更新

提纲

- 问题: 网络如何互联
  - 网络层和网络互联
  - IP协议
  - IP地址
  - IP附属协议
  - 分组交付与路由选择
    - 分组交付
    - 路由选择
    - 路由信息协议RIP
    - 最短路径优先协议OSPF
    - 边界网关协议BGP

距离向量路由选择

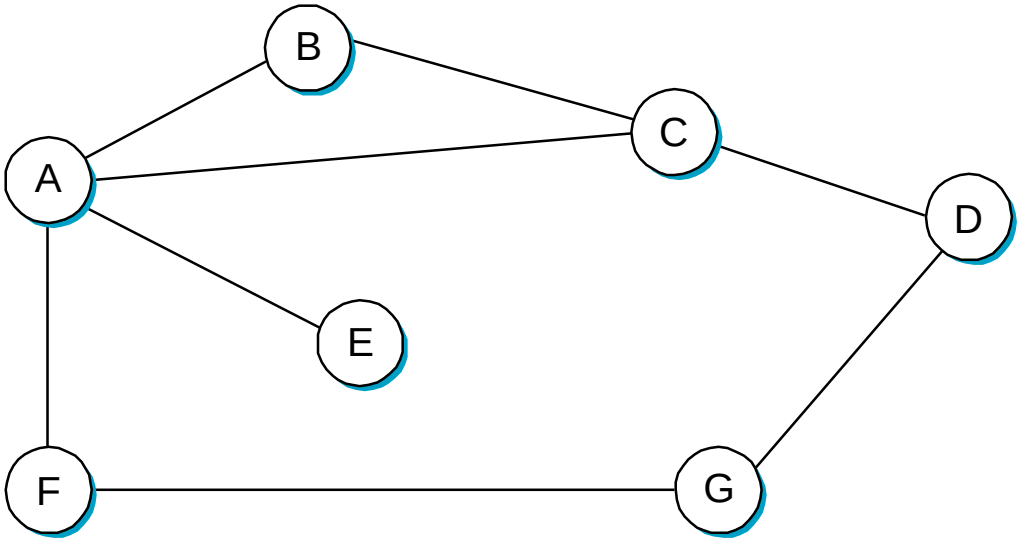
基本思想

- 每个节点构造一个包含到其他所有节点的“距离”(代价)的一维数组(向量)
- 将该向量分发给其邻节点
- 节点根据接收到的距离向量计算到达其他所有节点的最短路径
- 经过距离向量的迭代交换和计算过程最终构造完整的路由表

距离向量路由选择

- 步骤1: 最初节点将向量中可直接到达的邻节点的代价设置为**1**, 到所有其他节点的代价赋值为∞
- 步骤2: 节点将距离向量发送至直接可达的邻节点
- 步骤3: 根据从邻节点Y 收到的距离向量, 节点X:
  - 计算本节点到达所有其他节点的距离: 将来自Y 的向量中到达所有其他节点的距离(例如 Z)加上本节点到Y 的距离(即**1**)
  - 比较计算结果与本地向量中对应结果的大小: 如果计算结果更小, 则用该值替换本地距离向量的对应结果, 并记录NextHop为Y

距离向量路由选择: 示例



dst.	cost	NextHop
A	1	A
C	1	C
D	∞	–
E	∞	–
F	∞	–
G	∞	–

节点B中的初始路由表

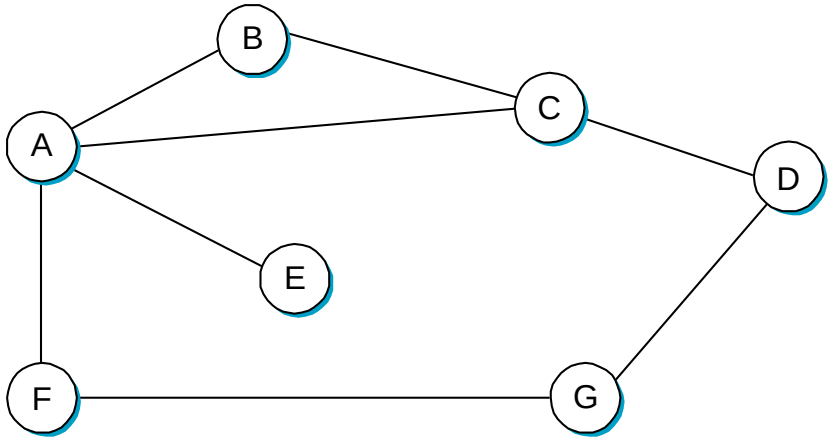
- 初始距离向量

A: (0, 1, 1, ∞, 1, 1, ∞)　　B: (1, 0, 1, ∞, ∞, ∞, ∞)

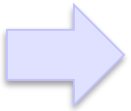
C: (1, 1, 0, 1, ∞, ∞, ∞)　　E: (1, ∞, ∞, ∞, 0, ∞, ∞)

F: (1, ∞, ∞, ∞, ∞, 0, 1)

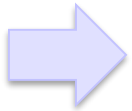
距离向量路由选择: 示例



Dest	Cost	Next Hop
A	1	A
C	1	C
D	∞	–
E	∞	–
F	∞	–
G	∞	–



Dest	Cost	Next Hop
A	1	A
C	1	C
D	2	C
E	2	A
F	2	A
G	∞	–



Dest	Cost	Next Hop
A	1	A
C	1	C
D	2	C
E	2	A
F	2	A
G	3	A

节点B中的初始路由表

节点B中最终的路由表

距离向量 – 本地视图

Dest	Cost	Next Hop
B	1	B
C	1	C
D	∞	–
E	1	E
F	1	F
G	∞	–

节点A的路由表

Dest	Cost	Next Hop
A	1	A
C	1	C
D	∞	–
E	∞	–
F	∞	–
G	∞	–

节点B的路由表

Dest	Cost	Next Hop
A	1	A
B	1	B
D	1	D
E	∞	–
F	∞	–
G	∞	–

节点C的路由表

Dest	Cost	Next Hop
A	∞	–
B	∞	–
C	1	C
E	∞	–
F	∞	–
G	1	G

节点D的路由表

Dest	Cost	Next Hop
A	1	A
B	∞	–
C	∞	–
D	∞	–
F	∞	–
G	∞	–

节点E的路由表

Dest	Cost	Next Hop
A	1	A
B	∞	–
C	∞	–
D	∞	–
E	∞	–
G	1	G

节点F的路由表

Dest	Cost	Next Hop
A	∞	–
B	∞	–
C	∞	–
D	1	D
E	∞	–
F	1	F

节点G的路由表

距离向量 – 第一次更新

Dest	Cost	Next Hop
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	2	F

节点A的路由表

Dest	Cost	Next Hop
A	1	A
C	1	C
D	2	C
E	2	A
F	2	A
G	∞	–

节点B的路由表

Dest	Cost	Next Hop
A	1	A
B	1	B
D	1	D
E	2	A
F	2	A
G	2	G

节点C的路由表

Dest	Cost	Next Hop
A	2	C
B	2	C
C	1	C
E	∞	–
F	2	G
G	1	G

节点D的路由表

Dest	Cost	Next Hop
A	1	A
B	2	A
C	2	A
D	∞	–
F	2	A
G	∞	–

节点E的路由表

Dest	Cost	Next Hop
A	1	A
B	2	A
C	2	A
D	2	G
E	2	E
G	1	G

节点F的路由表

Dest	Cost	Next Hop
A	2	F
B	∞	–
C	2	D
D	1	D
E	∞	–
F	1	F

节点G的路由表

距离向量 – 第二次更新

Dest	Cost	Next Hop
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	2	F

节点A的路由表

Dest	Cost	Next Hop
A	1	A
C	1	C
D	2	C
E	2	A
F	2	A
G	3	A

节点B的路由表

Dest	Cost	Next Hop
A	1	A
B	1	B
D	1	D
E	2	A
F	2	A
G	2	G

节点C的路由表

Dest	Cost	Next Hop
A	2	C
B	2	C
C	1	C
E	3	C
F	2	G
G	1	G

节点D的路由表

Dest	Cost	Next Hop
A	1	A
B	2	A
C	2	A
D	3	A
F	2	A
G	3	A

节点E的路由表

Dest	Cost	Next Hop
A	1	A
B	2	A
C	2	A
D	2	G
E	2	E
G	1	G

节点F的路由表

Dest	Cost	Next Hop
A	2	F
B	3	D
C	2	D
D	1	D
E	3	F
F	1	F

节点G的路由表

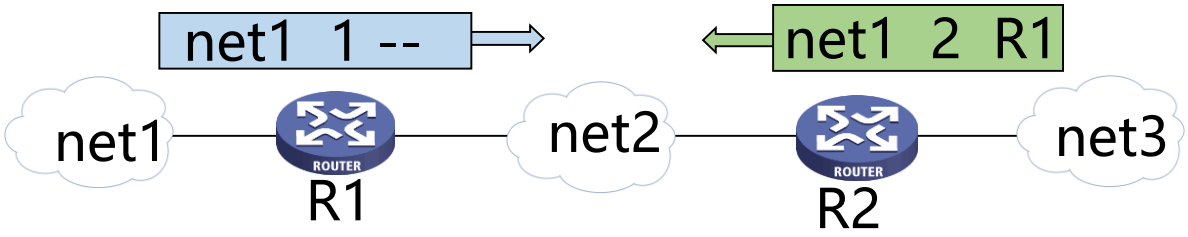
实际问题

❑ 如何分发路由更新信息?

- ➢ 解决方案1: 周期性更新
  - 及时没有任何变化, 每一个节点总是定期自动发送更新报文
  - 这使得其他节点知道它仍在正常运行
  - 可以确保当现有路由不看可用时, 他们仍然可以一直得到所需的信息
  - 更新频率: 几秒到几分钟
- ➢ 解决方案2: 触发更新
  - 每当一个节点从它的邻节点收到导致更新的报文时
  - 当一个节点的路由表发生变化时将向邻节点发送一条更新报文
  - 可能引起邻节点的路由表变化, 从而使得这些邻节点又向他们的邻节点发送更新报文

无穷计算问题

正常情况

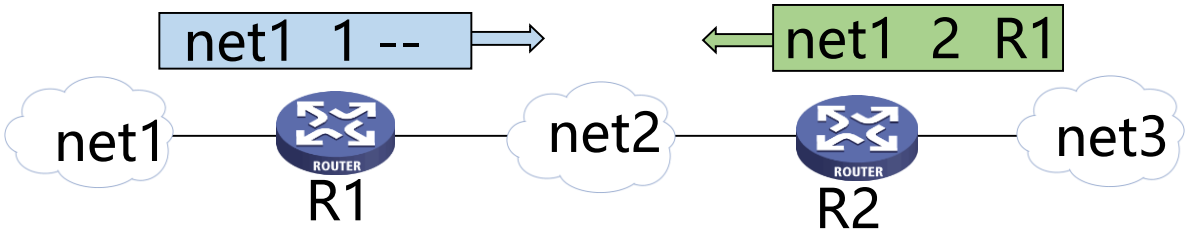


R1 说：“我到 net1 的距离是 1，是直接交付。”

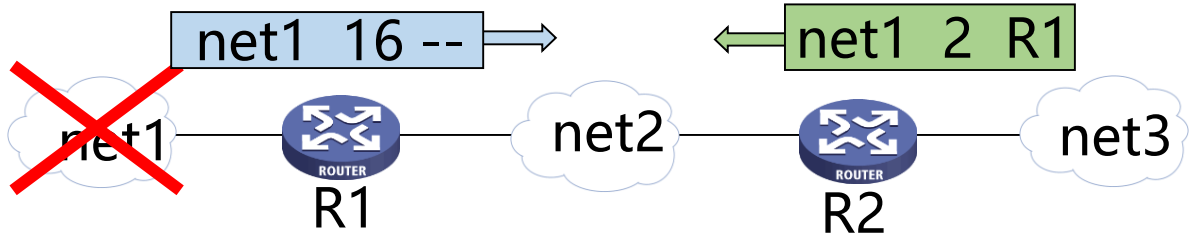
R2 说：“我到 net1 的距离是 2，是经过 R1。”

无穷计算问题

正常情况



net1出了故障

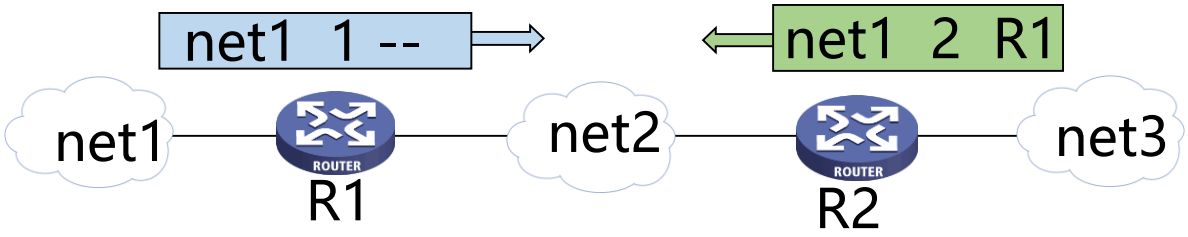


R1 说：“我到 net1 的距离是 16（表示无法到达），是直接交付。”

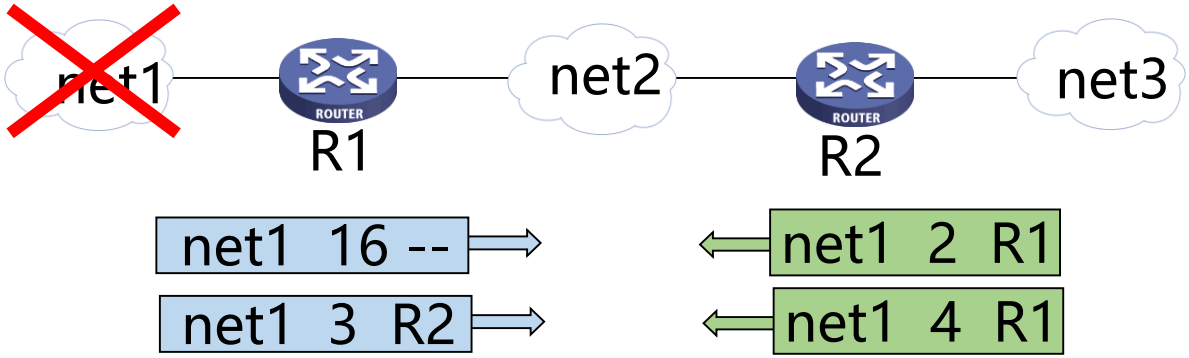
但 R2 在收到 R1 的更新报文之前，还发送原来的报文，因为这时 R2 并不知道 R1 出了故障。

无穷计算问题

正常情况



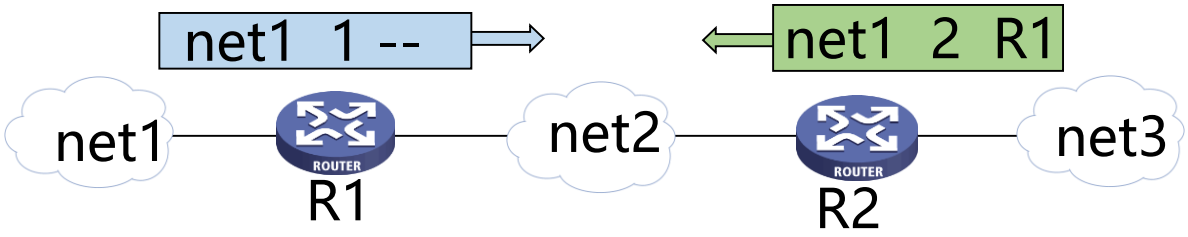
net1出了故障



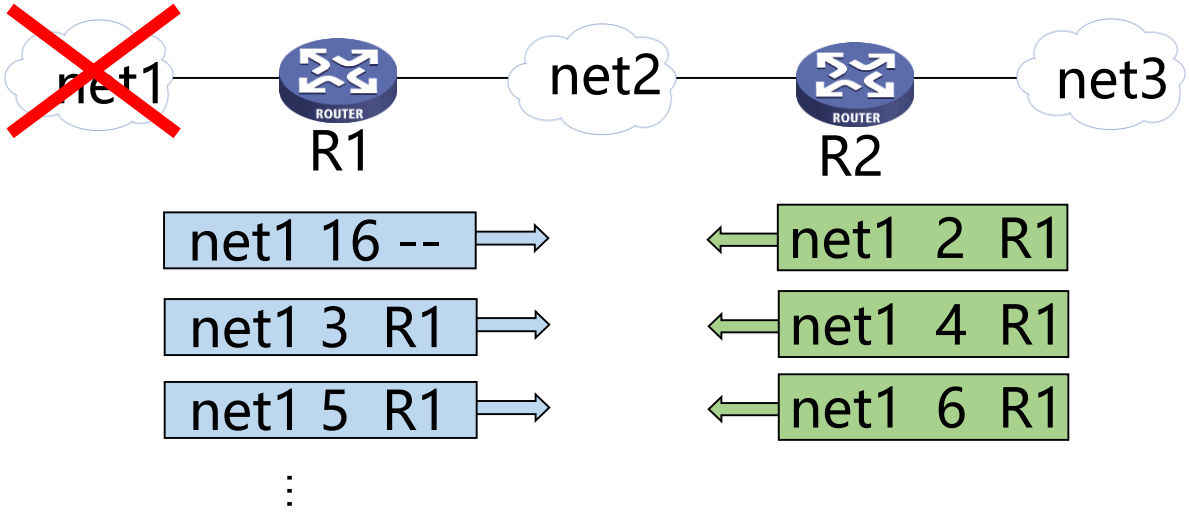
R2 以后又更新自己的路由表为“net1, 4, R1”，表明 “我到 net 1 距离是 4，下一跳经过 R1”。

无穷计算问题

正常情况



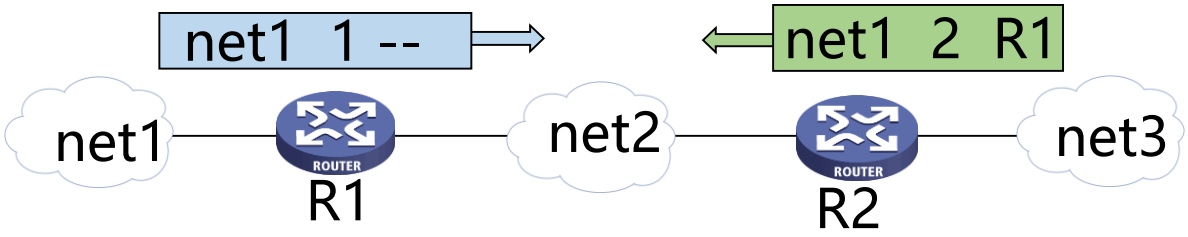
net1出了故障



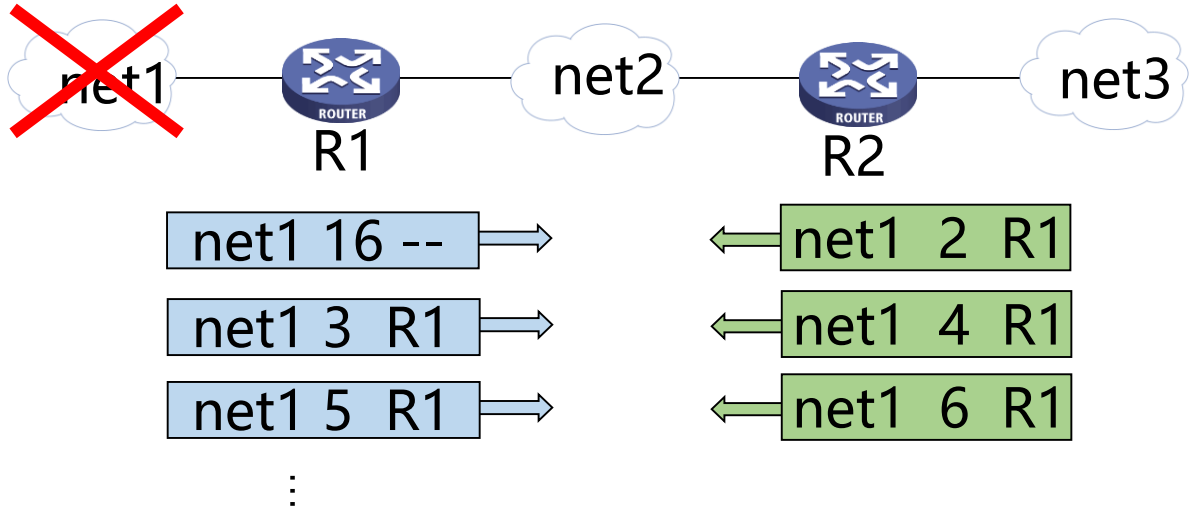
这样不断更新下去，直到 R1 和 R2 到 net 1 的距离都增大到 16 时， R1 和 R2 才知道 net 1 是不可达的。

无穷计算问题

正常情况



net1出了故障



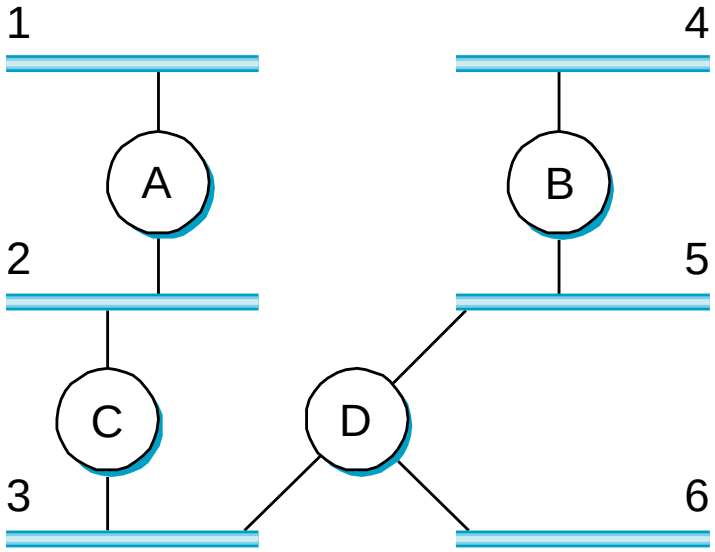
好消息传播快，坏消息传播慢，是距离向量路由的一个主要缺点！

无穷计算问题的解决方案

- 方案A: 采用最大跳数取代无穷大
- 方案B: 水平分割
  - 不向提供路由的邻节点发送更新报文
  - 例如: R2不向R1发送有关net1的更新信息, 只向其他邻居发送更新信息
- 方案C: 带反向抑制的水平分割
  - 向提供路由的邻节点发送最大跳数信息
  - 例如： R2向R1发送有关net1的更新信息为16, 而向其他邻居发送正常的更新信息
- 上述解决方案在路由循环超过3节点的网络会失效

路由选择信息协议(RIP)

- 基于距离向量算法的路由选择协议
  - 距离向量算法
  - 1982年跟随BSD-UNIX发布版本一同发布
  - 距离评价指标: # 跳数(最大值 = 15 跳)



路由器通告到达网络的代价, 而不是到达其他路由器的代价

RIP协议的特点

- 版本: v1(有类路由)、v2(无类路由)、ng(支持IPv6)
- 支持多协议族 (不仅仅是IP)
- 拓扑信息分发
  - 周期性更新：每隔30秒
  - 触发更新：无论何时收到来自其他路由器的引起路由表改变的信息
- 链路的代价(评价指标)
  - RIP 中的“距离”也称为“跳数”，所有链路的代价为1
  - RIP 认为一个好的路由就是它通过的路由器的数目少
  - 有效距离取值范围[1, 15], 16意味着无穷大

小结：距离向量路由

- 距离向量算法
  - 距离向量: 每个节点仅与直接相连的节点通信，但是包含到所有节点的距离

提纲

- 问题: 网络如何互联
  - 网络层和网络互联
  - IP协议
  - IP地址
  - IP附属协议
  - 分组交付与路由选择
    - 分组交付
    - 路由选择
    - 路由信息协议RIP
    - 最短路径优先协议OSPF
    - 边界网关协议BGP

链路状态路由选择

基本思想

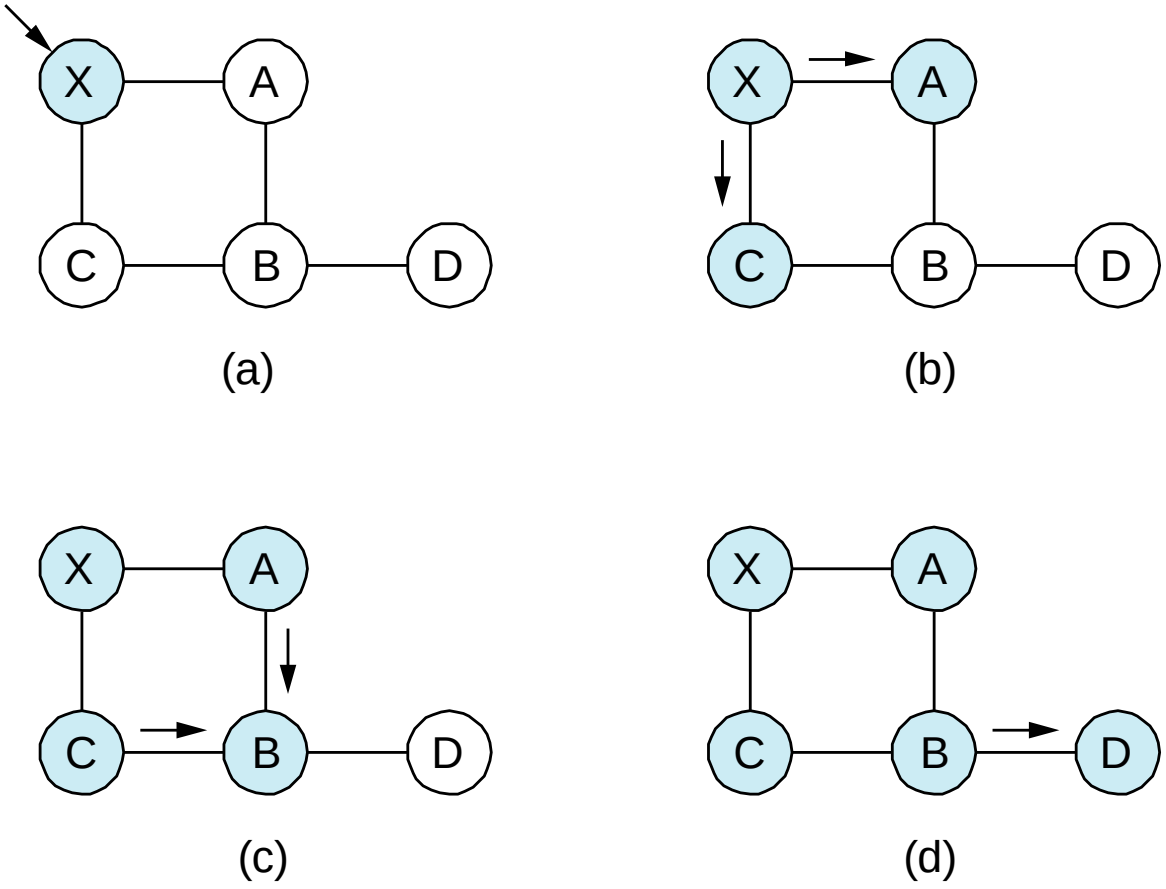
- 每个节点都知道怎样到达它的邻节点(link state), 并通告给其他所有节点
  - “链路状态”就是说明本路由器都和哪些路由器相邻，以及该链路的“度量”(metric)
- 每个节点都可以获得完整的链路状态数据库(全网的拓扑结构图)来建立其路由表

依赖于两个机制

- 链路状态的可靠洪泛
- 根据所有积累的链路状态知识的总和进行的路由计算

洪泛

- 洪泛: 通告报文达到所有节点



路由计算：基于Dijkstra 算法

- 每个路由器维护两个列表
  - 试探表(Tentative)
  - 证实表(Confirmed)
- 每张表中有多条记录
  - 记录包含(Destination, Cost, NextHop)
- 最终证实表即期望的结果 – 形成节点的路由表
- 步骤(详见下一页)

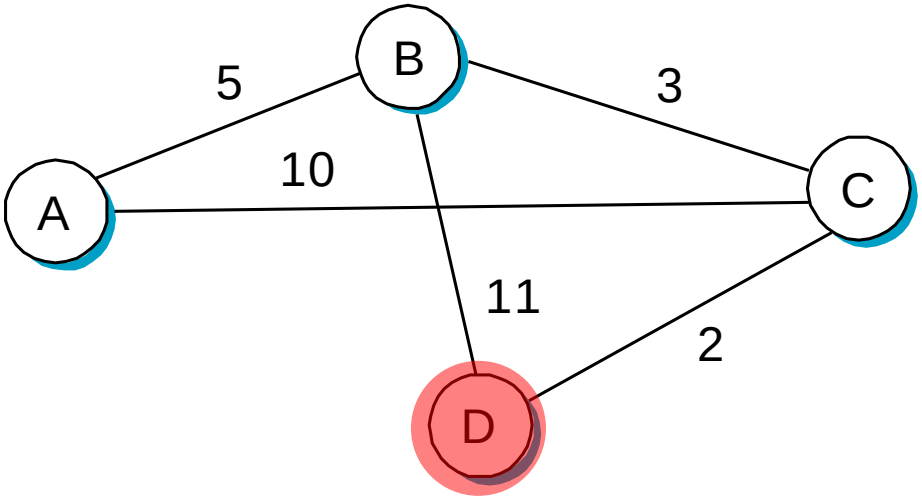
Dijkstra算法的实现: 前向搜索

1. 用记录(Myself, 0, -)初始化证实表
2. 上一步加入证实表的节点称为Next节点, 选择其LSP
3. For (Next节点的每一个邻节点)
   到达该邻节点的代价 = 从自身到Next节点的代价+ 从Next节点到邻节点的代价

   - If 邻节点当前既不在证实表中也不在试探表中, then 将记录(Neighbor, Cost, NextHop) 加入试探表中, 其中 NextHop 为自身到达Next节点所经的节点

   - If 邻节点在试探表中, 且代价小于当前表中记录的代价, then 用记录(Neighbor, Cost, NextHop)替换当前记录, 其中 NextHop 为自身到达Next节点所经的节点
4. If 试探表为空, 则停止; otherwise, 从试探表中挑选最小代价的记录, 将其移入确认表, 并转回执行步骤2

Dijkstra算法实现：示例



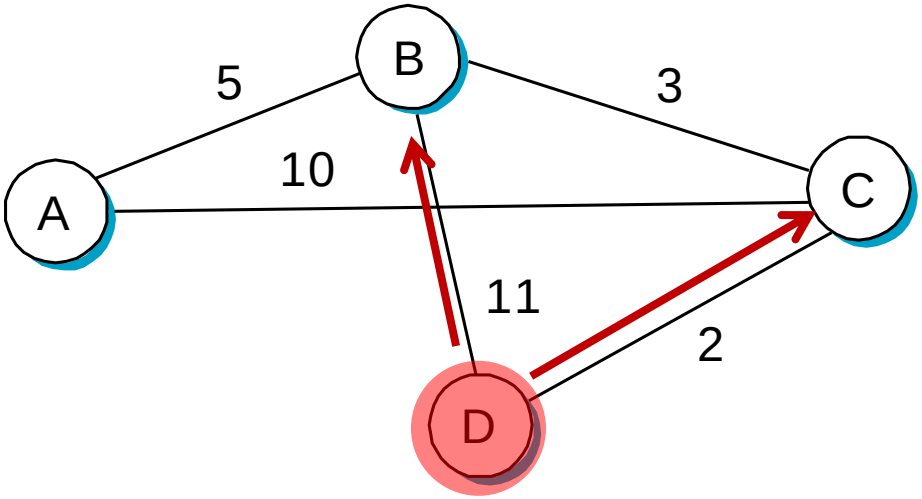
路由器D

证实节点

搜索方向

Step	1	2	3	4	5	6	7
证实表	(D,0,-)						
试探表							

Dijkstra算法实现：示例



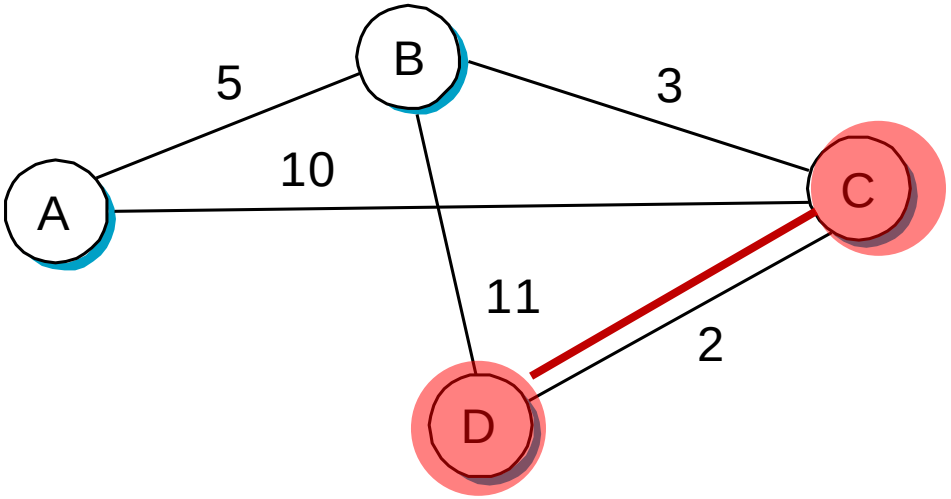
路由器D

证实节点

搜索方向

Step	1	2	3	4	5	6	7
证实表	(D,0,-)	(D, 0,-)					
试探表		(B,11,B) (C,2,C)					

Dijkstra算法实现：示例



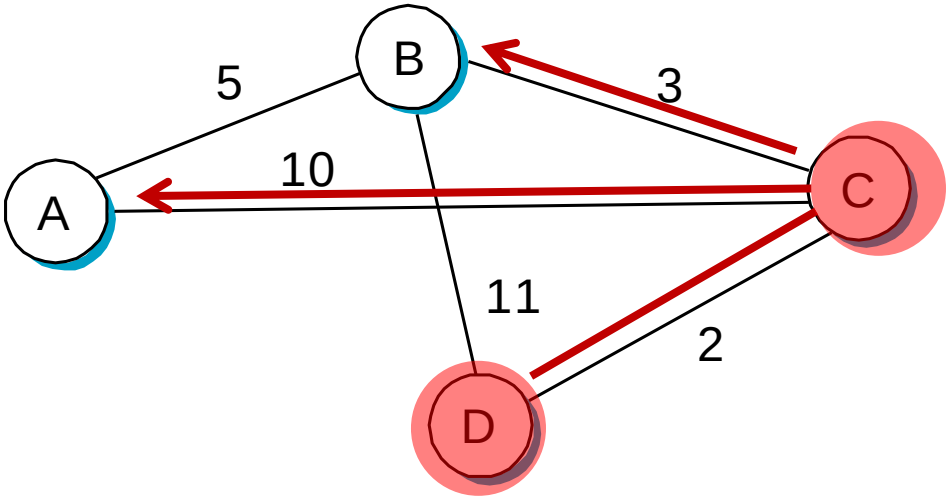
路由器D

证实节点

搜索方向

Step	1	2	3	4	5	6	7
证实表	(D,0,-)	(D, 0,-)	(D,0,-) (C,2,C)				
试探表		(B,11,B) (C,2,C)	(B,11,B)				

Dijkstra算法实现：示例



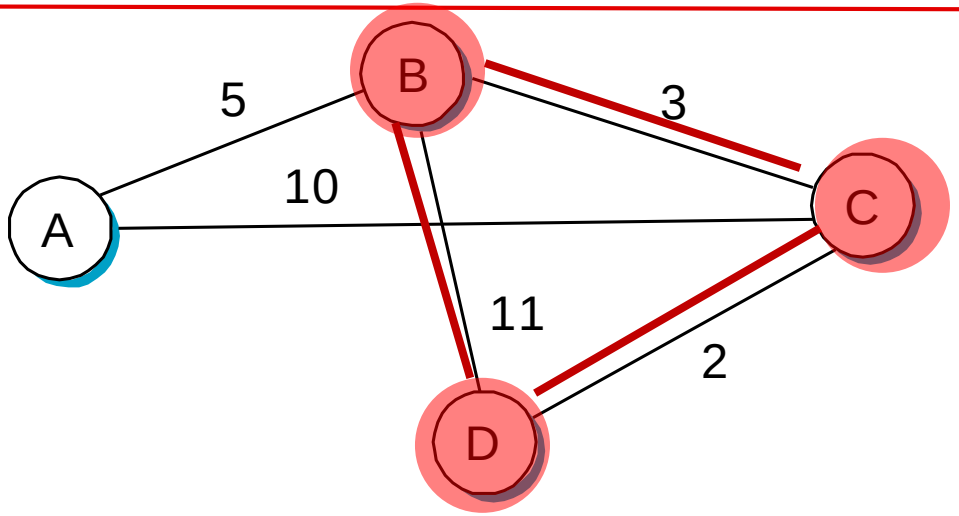
路由器D

证实节点

搜索方向

Step	1	2	3	4	5	6	7
证实表	(D,0,-)	(D, 0,-)	(D,0,-) (C,2,C)	(D,0,-) (C,2,C)			
试探表		(B,11,B) (C,2,C)	(B,11,B)	~~(B,11,B)~~ (B,5,C) (A,12,C)			

Dijkstra算法实现：示例



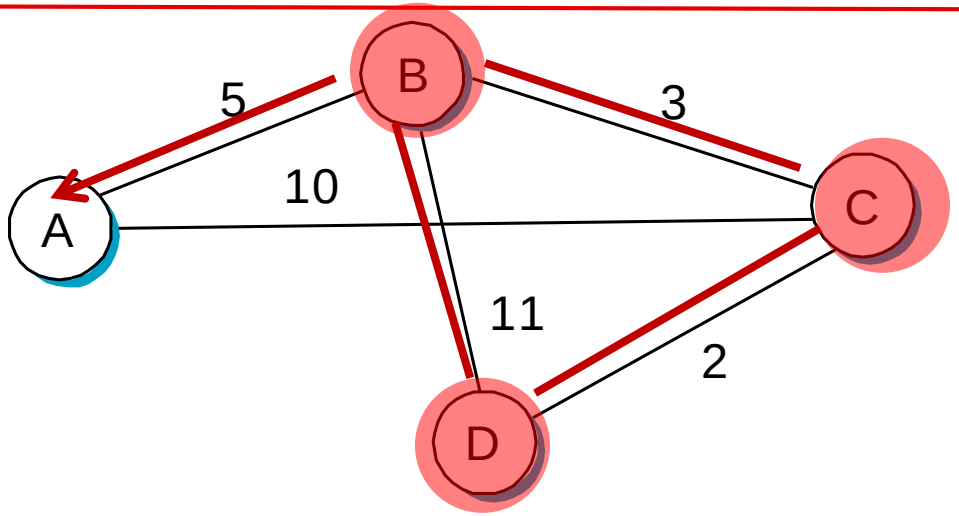
证实节点

搜索方向

路由器D

Step	1	2	3	4	5	6	7
证实表	(D,0,-)	(D, 0,-)	(D,0,-) (C,2,C)	(D,0,-) (C,2,C)	(D,0,-) (C,2,C) (B,5,C)		
试探表		(B,11,B) (C,2,C)	(B,11,B)	(B,5,C) (A,12,C)	(A,12,C)		

Dijkstra算法实现：示例



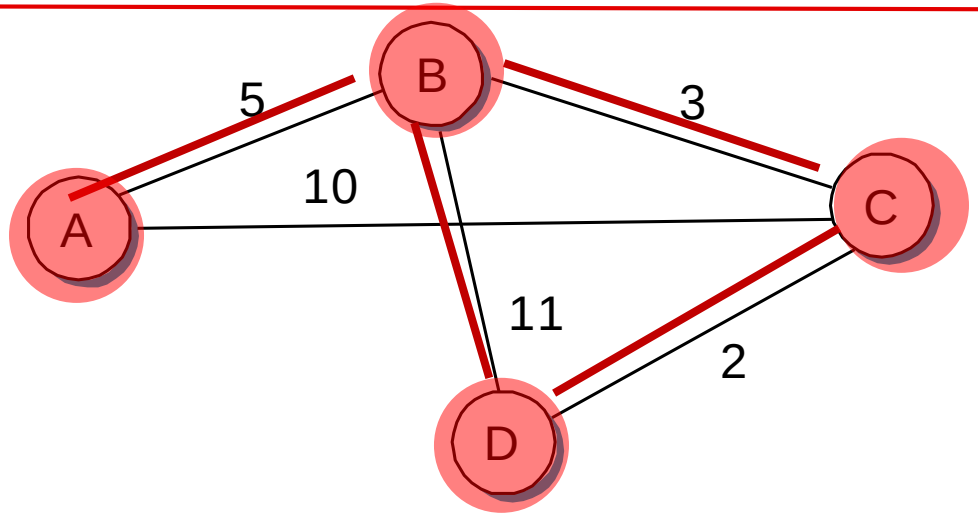
路由器D

证实节点

搜索方向

Step	1	2	3	4	5	6	7
证实表	(D,0,-)	(D, 0,-)	(D,0,-) (C,2,C)	(D,0,-) (C,2,C)	(D,0,-) (C,2,C) (B,5,C)	(D,0,-) (C,2,C) (B,5,C)	
试探表		(B,11,B) (C,2,C)	(B,11,B)	(B,5,C) (A,12,C)	(A,12,C)	~~(A,12,C)~~ (A,10,C)	

Dijkstra算法实现：示例



路由器D

证实节点

搜索方向

Step	1	2	3	4	5	6	7
证实表	(D,0,-)	(D, 0,-)	(D,0,-) (C,2,C)	(D,0,-) (C,2,C)	(D,0,-) (C,2,C) (B,5,C)	(D,0,-) (C,2,C) (B,5,C)	(D,0,-) (C,2,C) (B,5,C) (A,10,C)
试探表		(B,11,B) (C,2,C)	(B,11,B)	(B,5,C) (A,12,C)	(A,12,C)	(A,10,C)	

Done!

可靠洪泛

- 链路状态分组LSP
  - 创建LSP的节点ID
  - 与该节点直接相邻的节点信息列表 <AdjacentNode, Cost>, 其中包括到这些邻节点的链路代价
  - 一个序号
  - LSP的生命周期
- 如何保证链路状态分组的洪泛
  - 采用确认和重传机制
  - 通过序号
  - 不发回发送LSP的节点

可靠洪泛

- 新的链路状态分组快速洪泛，旧的分组快速被删除
  - 采用序列号和TTL
    - 长序列号
    - 逐跳递减TTL
- 最小化洪泛过程中的链路状态分组的数量
  - 使用很长时间(通常在几个小时)的定时器周期性地生成
  - 按需触发

链路状态洪泛的触发器

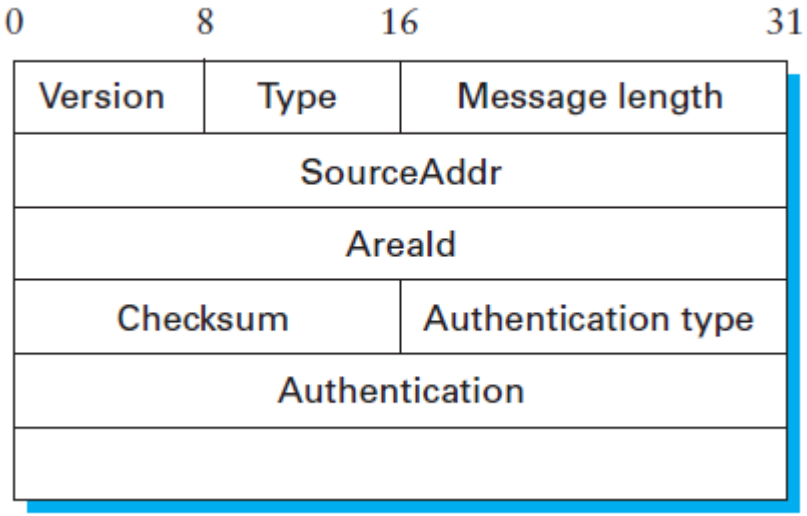
- 与RIP协议类似，两种情况下产生新的LSP
  - 周期性计时器超时
  - 拓扑结构变化
- 拓扑变化检测
  - 新的节点/链路：通过链路状态分组通告
  - 节点/链路失效
    - 通过链路层协议
    - 周期性交换“hello”报文

OSPF(开放最短路径优先)协议

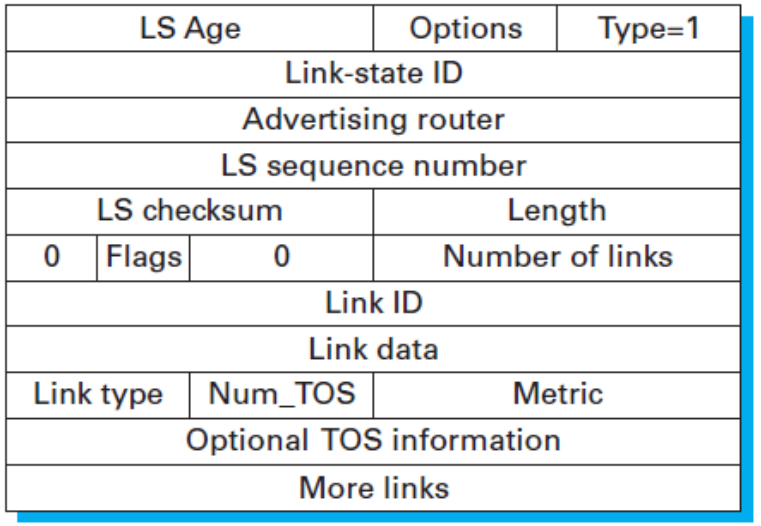
- “开放”: 公开的, 非专有的, 由IETF主持创建
- 采用链路状态算法
  - LS分发(可靠洪泛)
  - 每个节点构建网络拓扑(全局信息)
  - 采用Dijkstra算法进行路由计算
    - 逐步构建 “试探表-证实表”
- 在OSPF通告中, 每一个邻接路由器对应一条记录

OSPF

- 报文类型
  - hello, request, send 以及acknowledge
- 链路状态通告 (LSA)
  - 链路状态分组的基本构件
  - 一个链路状态分组可能包含多个LSAs



OSPF首部格式



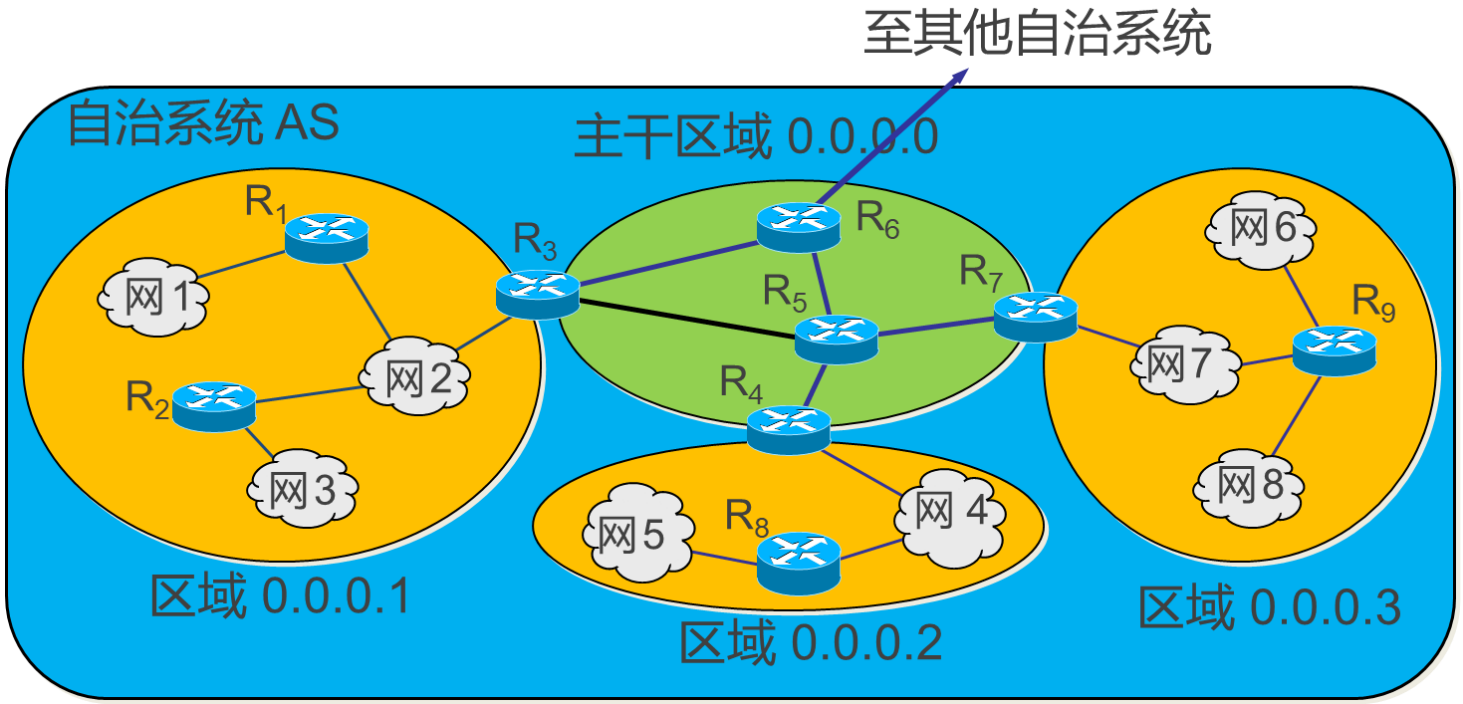
OSPF 链路状态通告

OSPF协议的特点(RIP协议不具备的)

- 除了基本的链路状态算法外, OSPF还支持
  - 安全机制：所有OSPF报文需要认证 (避免恶意入侵)
  - 分层路由：一个OSPF域可以划分为多个区域，在一个较大的域内可以采用层次化OSPF
  - 负载均衡: 允许多条相同代价的路径存在 (RIP中仅允许一条路径)，一个流可以均匀的分布于多条路径之上
  - 支持单播和多播: 多播OSPF(MOSPF)使用与OSPF相同的拓扑数据

OSPF的层次结构

- OSPF 将一个自治系统再划分为若干个更小的区域
  - 每一个区域都有一个32位的区域标识符
  - 在上层的区域叫作主干区域，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



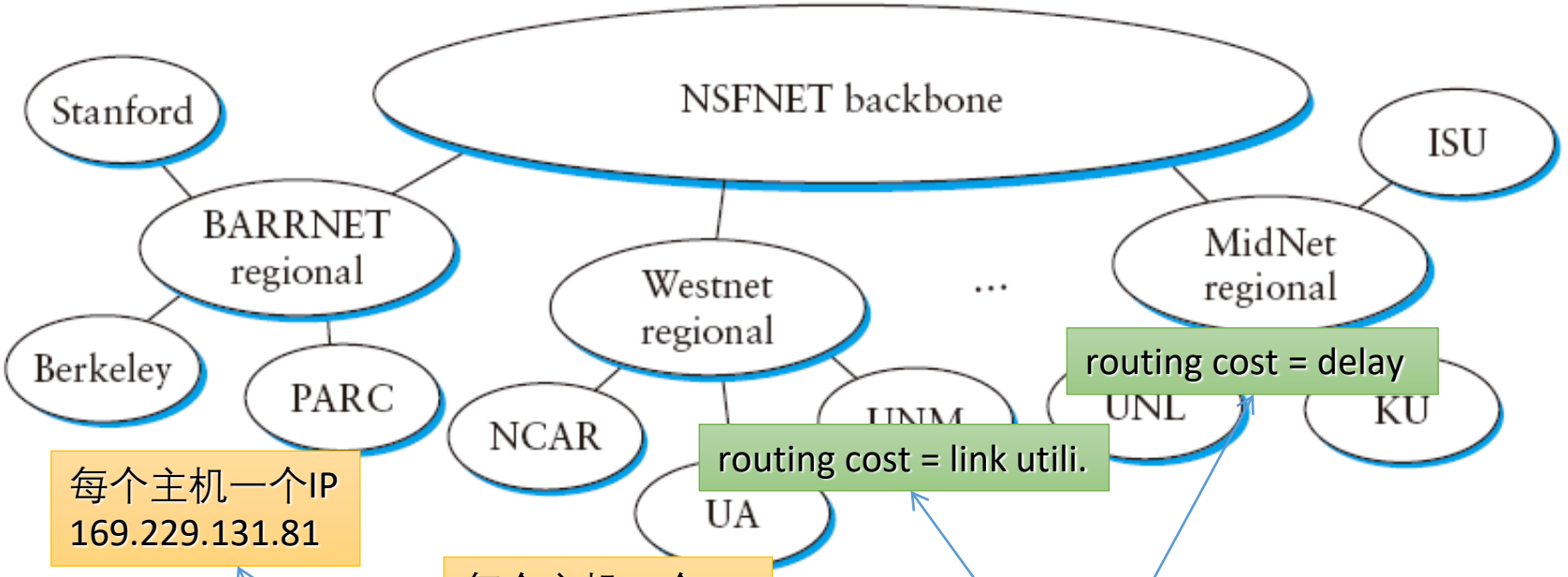
小结：链路状态路由

- 链路状态路由
  - 链路状态: 每个节点仅告诉所有其他节点与之直接相连的链路状态
  - 距离向量: 每个节点仅与直接相连的节点通信但是包含到达所有节点的距离

提纲

- 问题: 网络如何互联
  - 网络层和网络互联
  - IP协议
  - IP地址
  - IP附属协议
  - 分组交付与路由选择
    - 分组交付
    - 路由选择
    - 路由信息协议RIP
    - 最短路径优先协议OSPF
    - 边界网关协议BGP

互联网面临的扩展性挑战



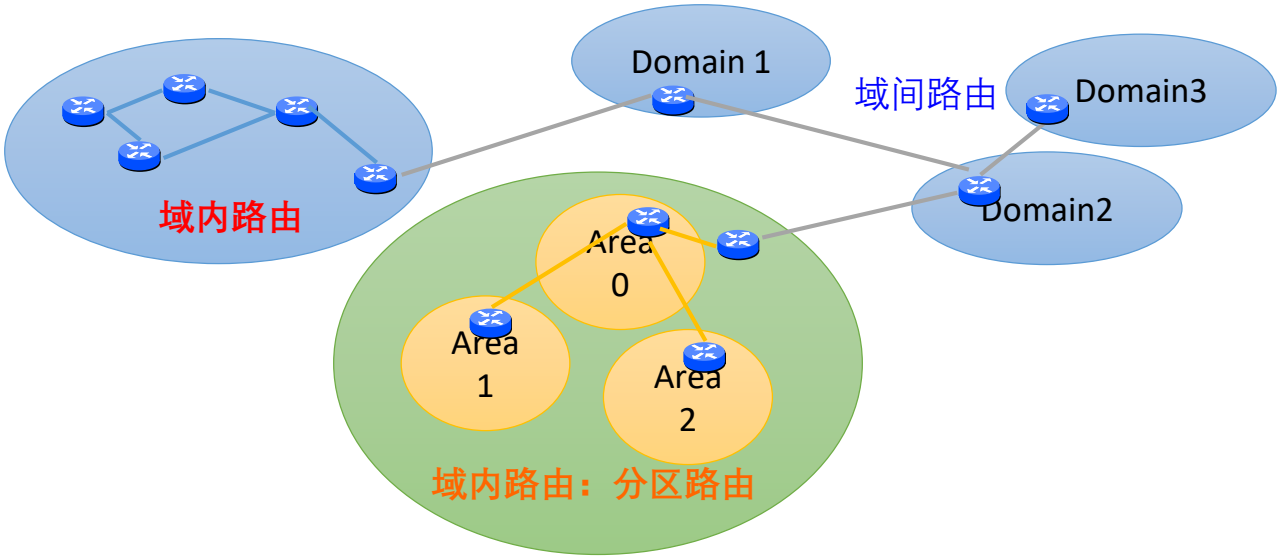
挑战之一：IP地址利用问题
如何确保IP地址不被迅速消耗掉

挑战之二：路由的扩展性问题
ISPs 对网络中所使用的最佳路由协议以及如何定义链路评价指标都有不同的看法

互联网面临的扩展性挑战

❑ 挑战之一：路由的扩展性问题

- 构造路由的层次化结构，应对扩展性
- 按照自治系统(AS)区分路由的域(Domain)
- AS内部的路由即域内路由，AS之间的路由即域间路由
- 域内路由可以进一步划分为路由的区域(Area)



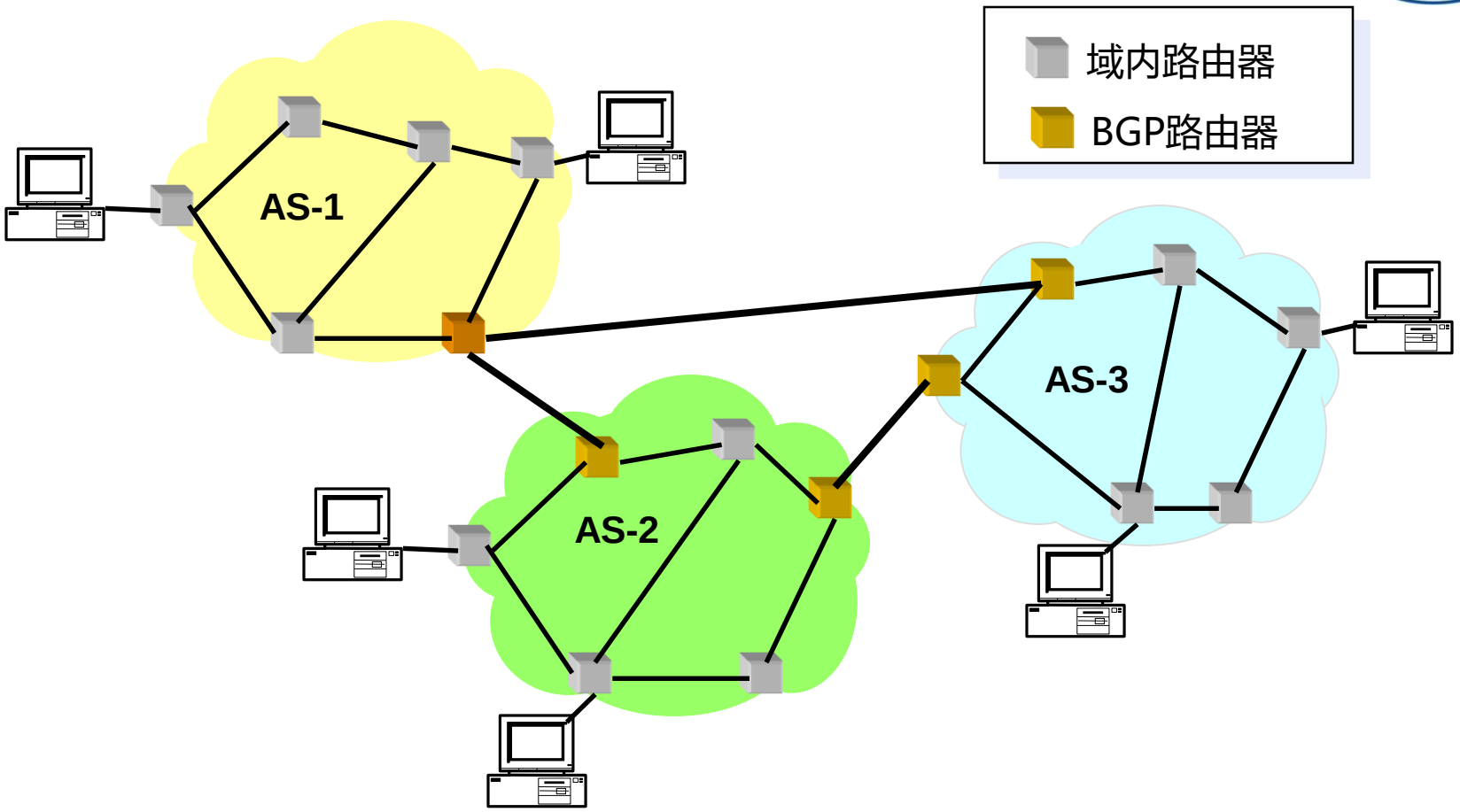
Internet和自治系统

- Internet 按照自治系统(也称为路由选择域)进行组织
- 每一个自治系统在一个独立的管理实体的控制之下
  - 示例: 校园网络, 公司网络
- 为什么提出自治系统?
  - 从管理和安全的角度考虑
  - 扩展性: 将大型互联网中路由选择信息进行层次汇聚的一种补充

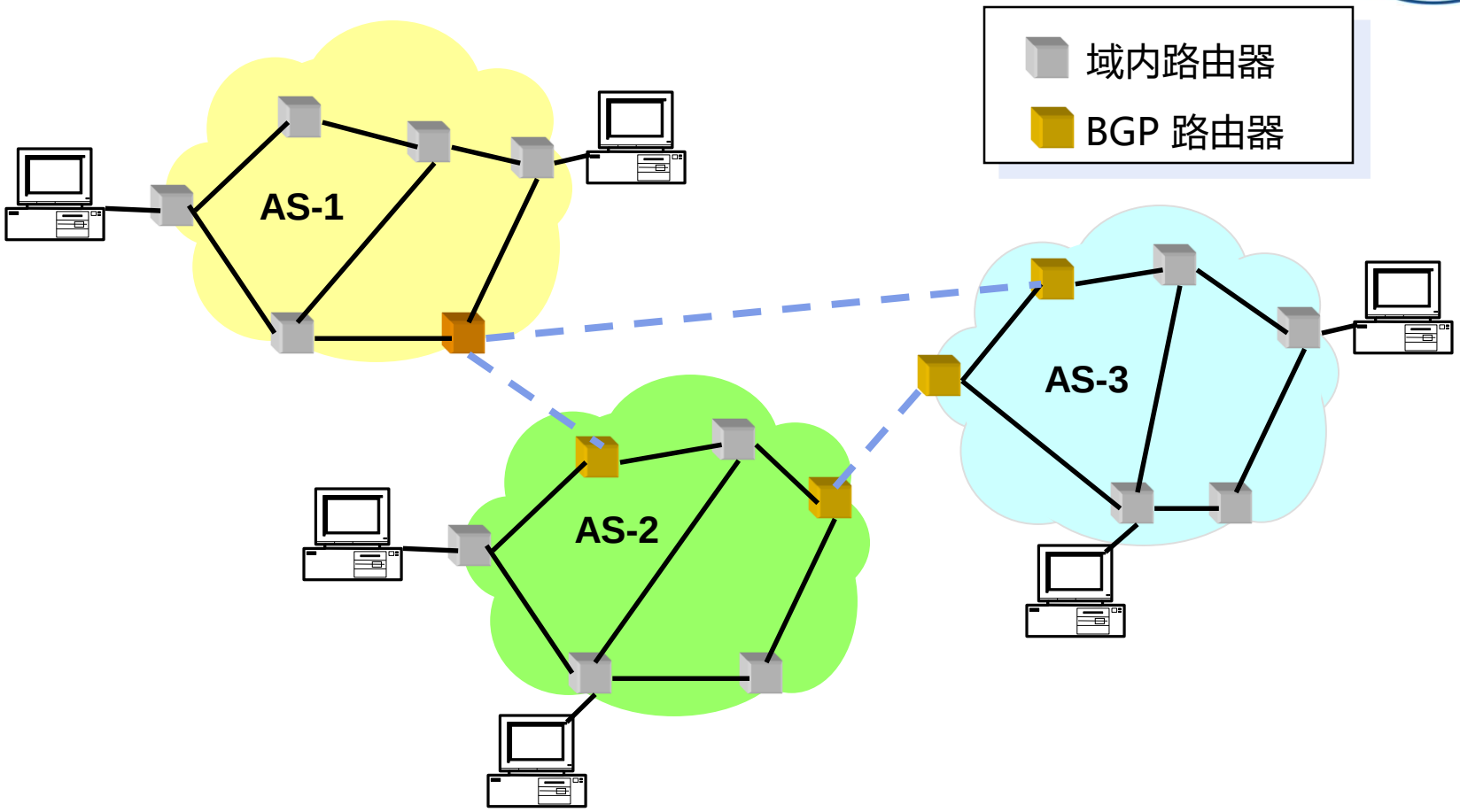
Internet路由选择架构-两级路由

- 域内路由选择
  - 运行于一个特定的网络, 即一个自治系统内
  - 网络内两个节点之间的最优路由
  - 内部网关协议(IGP)
    - 基于评价指标
    - 示例: OSPF, RIP, IS-IS
- 域间路由选择
  - 运行于多个网络之间, 即自治系统之间 (ASes)
  - 提供整个Internet的全连接
  - 外部网关协议(EGP)
    - 基于策略
    - 示例: EGP(外部网关协议), BGP(边界网关协议)

示例

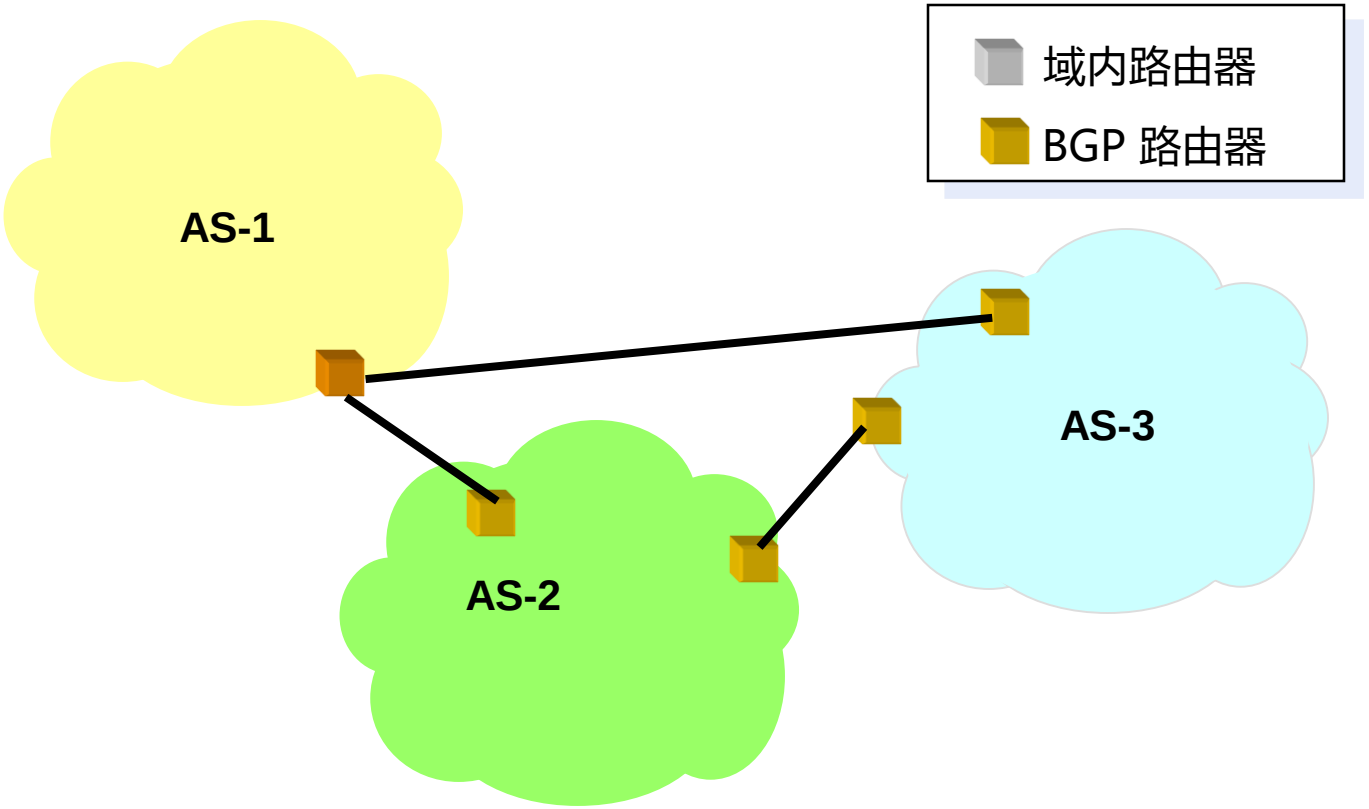


域内



域内路由选择协议又称为内部网关协议 (IGP), 例如 OSPF, RIP

域间



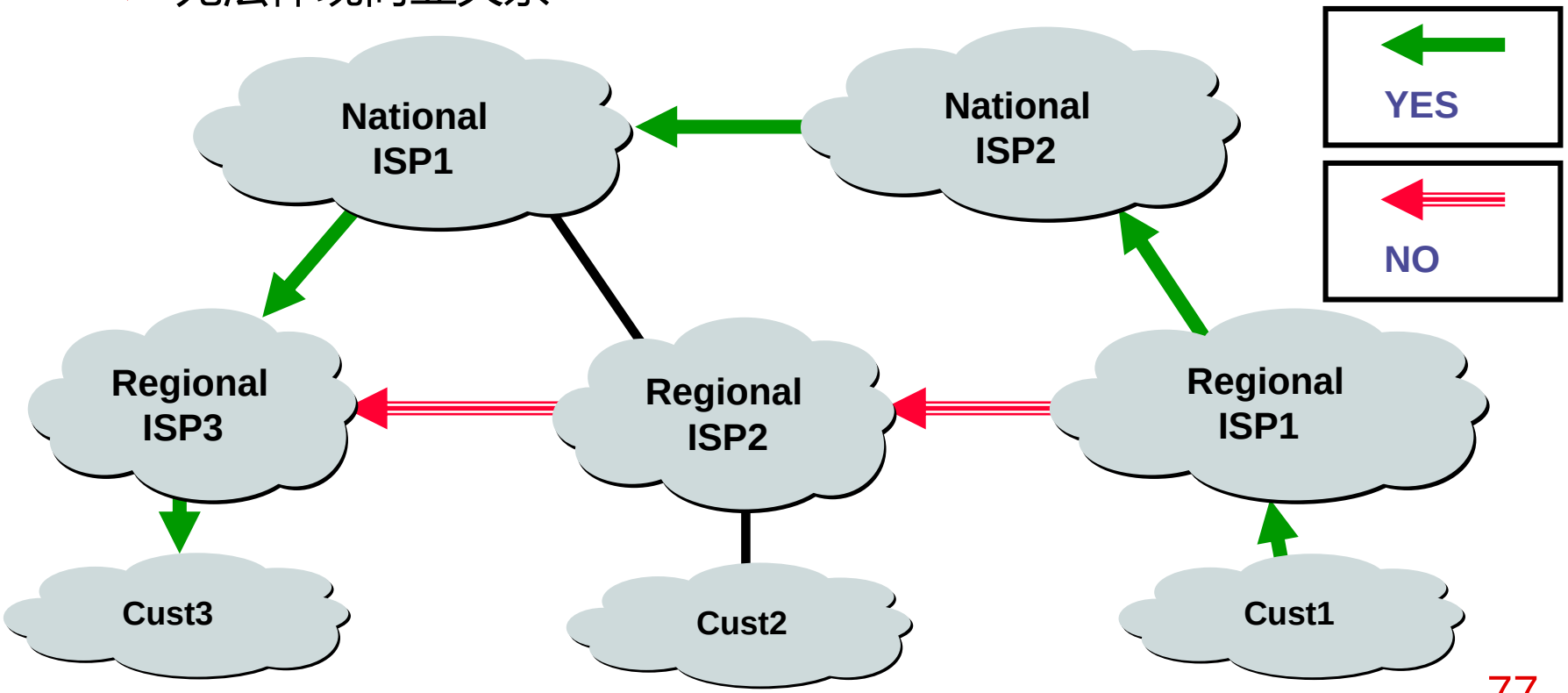
域间路由协议又称为外部网关协议 (EGP), 例如BGP

域间路由选择面临的挑战

- 规模
  - 前缀: 200,000, 仍在不断增长
  - AS编号: 已分配40K, 其中20,000+在使用中
  - 路由器: 数量至少上百万…
- 隐私
  - AS的管理者或ISP不希望泄露其拓扑信息以及与邻网络之间的商业关系
- 策略
  - 不存在全Internet通用的链路代价评价指标
  - 需要控制从哪里传送流量
    - 哪些路由器服务于域内流量
    - 哪些路由器服务于域间中转流量

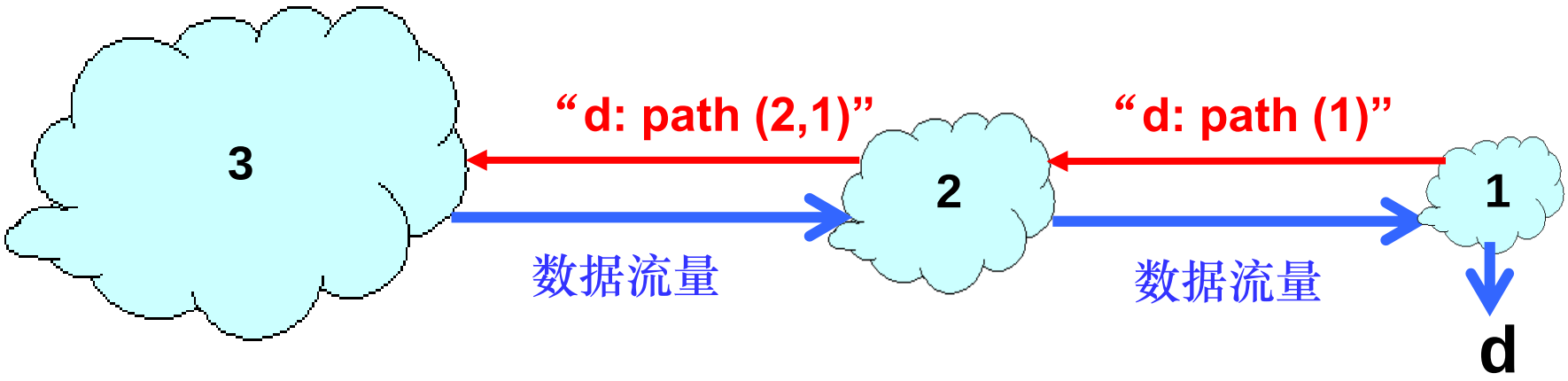
域间路由选择备选技术方案

- 最短路由选择的约束可以用吗？NO
  - 所有的流量必须通过最短路由传送
  - 所有节点需要拥有统一的链路代价标识
  - 无法体现商业关系



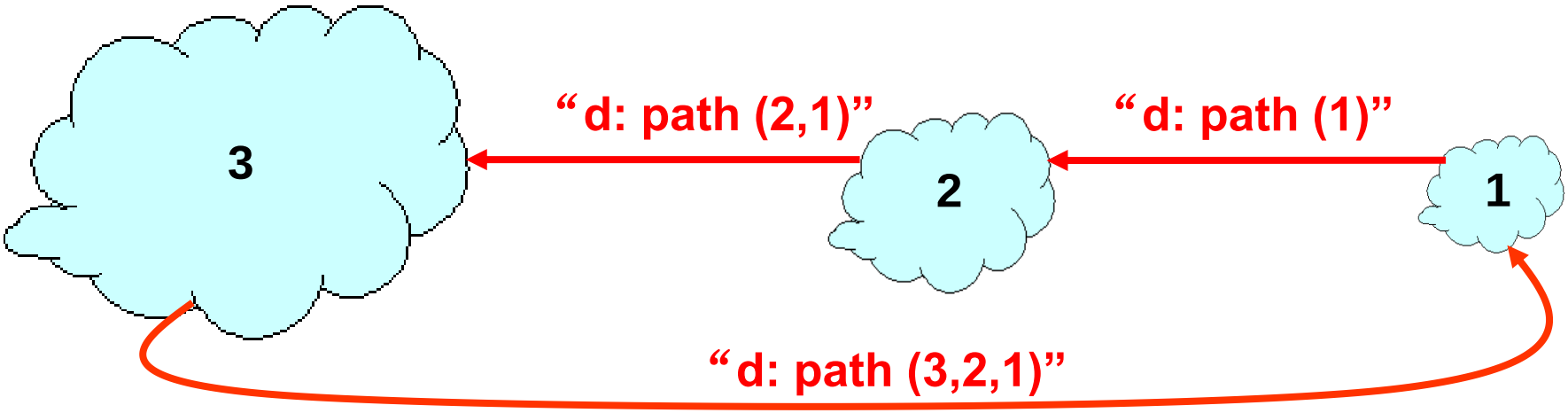
路径向量路由选择

- 是距离向量路由选择的扩展
  - 支持灵活的路由策略
  - 避免无穷计算问题
- 核心思想: 通告整个路径而非单个距离
  - 距离向量: 发送到每一个目的主机的距离向量
  - 路径向量: 发送到每一个目的主机的路径向量



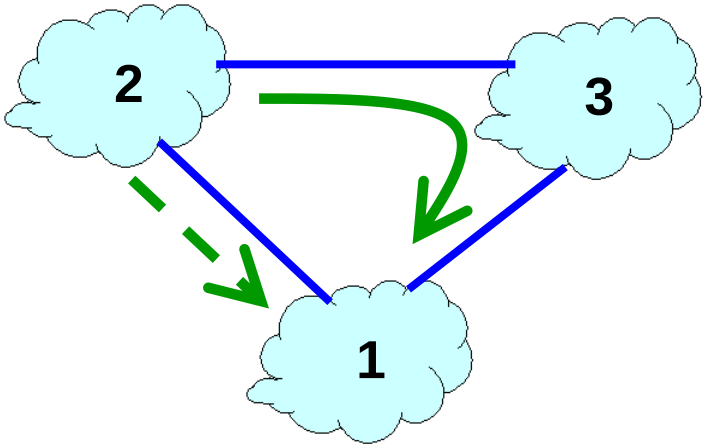
路径向量路由选择

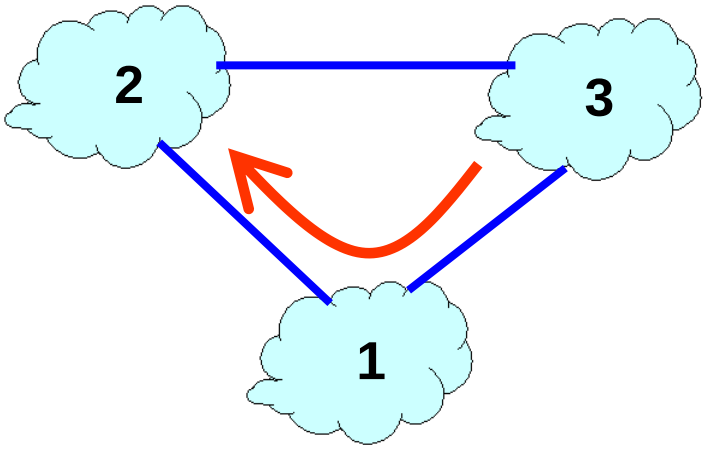
- 节点可以很容的检测环路路径
  - 在路径中查询自己的节点标识
  - 例如, 节点1发现自己的标识存在于路径“3, 2, 1”中
- 节点丢弃环路路径
  - 例如, 节点1丢弃该路径向量



路径向量路由选择

- 每一个节点可以采取灵活的本地策略
  - 路径选择: 采用哪一条路径?
  - 路径通告: 通告哪一条路径?
  - 例如:
    - 网络2更倾向于选择“2, 3, 1”而非“2, 1”
    - 网络1不允许其他网络知道路径“1, 2”的存在





边界网关协议BGP

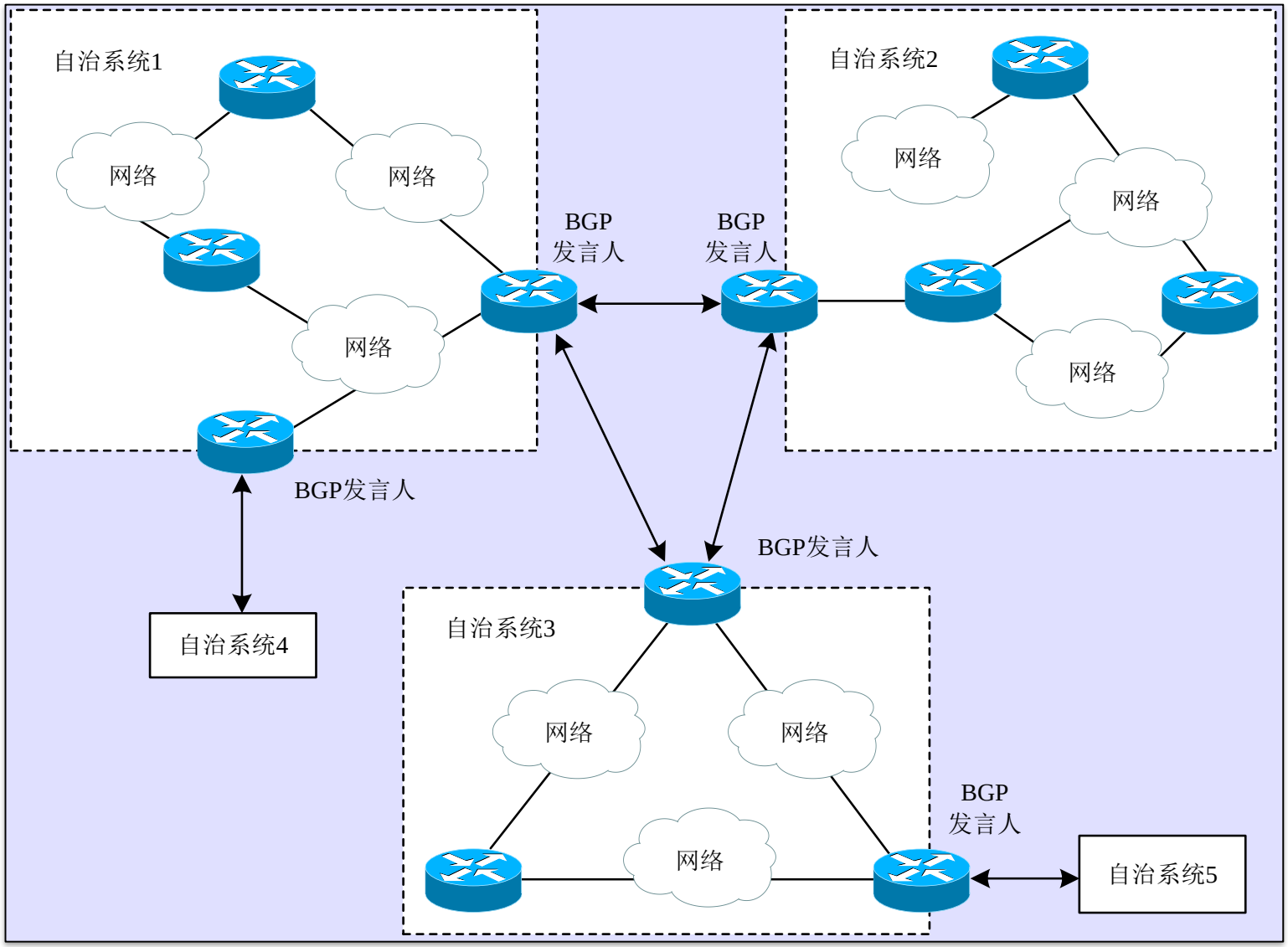
- Internet的域间路由选择协议
  - 基于前缀的路径向量路由选择协议
  - 基于策略进行路由选择构建AS路径
  - 过去的20年不断改进

- 1989 : BGP-1 [RFC 1105], 替代了最早的 EGP
- 1990 : BGP-2 [RFC 1163]
- 1991 : BGP-3 [RFC 1267]
- 1995 : BGP-4 [RFC 1771], 支持 CIDR
- 2006 : BGP-4 [RFC 4271], 修正

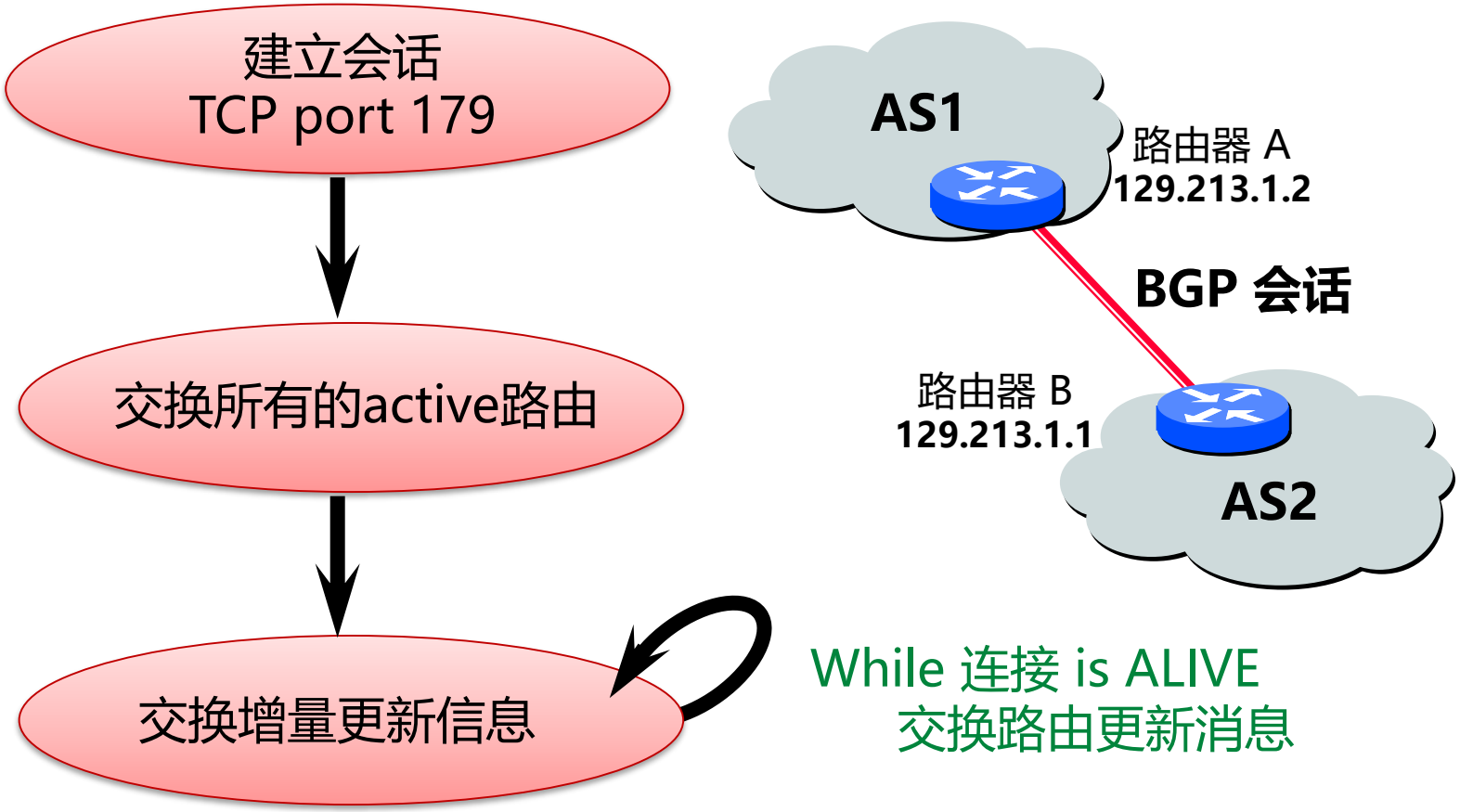
BGP的特点

- 允许ASes向其他 ASes “通告” 其有中转责任和如何到达
  - 每个AS至少有一个BGP “发言人” (Speaker)
    - 在AS的边界负责向其它AS转发路径的路由器
  - BGP发言人之间进行通信
  - 采用 “路由通告” 或 “承诺”
    - 也称为 “网络层可达信息”
  - 路径向量路由选择协议
- 基于策略: 允许ISP表达其路由策略
  - 包括选择outbound路径和通告internal路由
- 非常 “简单” 的协议, 但是配置相当复杂

BGP Speaker


自治系统1
网络
网络
网络
BGP
发言人
BGP
发言人
自治系统2
网络
网络
网络
网络
BGP发言人
自治系统4
BGP发言人
自治系统3
网络
网络
网络
BGP
发言人
自治系统5

BGP运行过程


建立会话
TCP port 179
交换所有的active路由
交换增量更新信息
AS1
路由器 A
129.213.1.2
BGP 会话
路由器 B
129.213.1.1
AS2
While 连接 is ALIVE
交换路由更新消息

增量协议

- 节点知道多条到达目的地的路径
  - 在路由表中存储所有的路由
  - 采用策略选择一条最好的路由
- 增量更新
  - 通告
    - 一旦选择一条新的路由，则将节点id加入路径向量
    - … 并(有选择性的)通告其他邻居
  - 撤销
    - 如果路由不再有效
    - … 发送撤销路由消息

小结

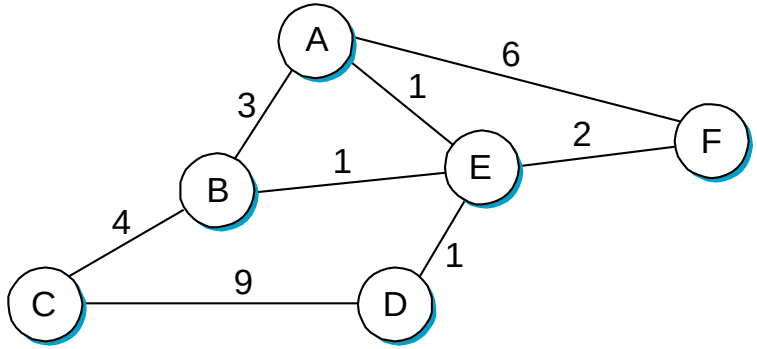
- 互联网路由的层次结构
  - Inter-Domain：AS之间的域间路由，外部网关协议BGP
  - Intra-Domain：AS之内的域内路由，内部网关协议RIP/OSPF
  - Routing Area：AS之内的分区域的路由，OSPF
- 域间路由
  - 自治系统(ASes)
  - 边界网关路由协议 BGP
    - 基于策略的路径向量路由选择协议

考纲要求

- 掌握：路由与转发的概念
- 了解：互联网路由协议指标设定的原理
- 掌握：路由表与转发表的区别
- 掌握：基于距离向量和基于链路状态的路由算法的原理，具备进行网络拓扑计算最短路径的能力
- 理解：RIP和OSPF的实现要点，及两种协议在信息收集、拓扑计算等方面的差异
- 理解：域内路由和域间路由的概念
- 了解：路径向量路由算法的原理
- 了解：自治系统的概念
- 理解：BGP协议的实现要点

要点回顾: 用图表示网络

- 采用图论表示网络
  - 节点: 路由器
  - 边: 链路
- 最短路径问题
  - Bellman-Ford 算法
    - 在节点对之间交换信息
    - 迭代, 异步, 分布式计算
    - 被距离向量路由选择算法采用
  - Dijkstra's 算法
    - 全局交换信息
    - 递归计算
    - 被链路状态路由选择算法采用



要点回顾: 距离向量路由

- 如何构造本地拓扑信息?
  - 构造描述到其它节点的距离向量, 如 (0, 1, 1, ∞, 1, 1, ∞)
  - 例如: RIP 路由报文信息，最大支持16跳
- 如何实现全局交换本地信息?
  - 在两两邻居节点之间递归交换距离向量
- 如何计算最短路径?
  - Bellman-Ford algorithm
  - 计算到其它所有节点的最短路径
- 如何在拓扑改变时更新最短路径?
  - 周期性交换信息、按需触发
  - 例如: RIP 每隔30秒更新

要点回顾: 链路状态路由

- 如何构造本地拓扑信息?
  - 构造到邻居的链路状态信息
  - 例如: OSPF路由的LSA报文，包括一系列的<AdjacentNode, Cost>
- 如何实现全局交换本地信息?
  - 向所有节点进行可靠的洪泛
  - 采用了可靠传输的机制： ACK, re-trans, Seq Num, TTL, …
- 如何计算最短路径?
  - Dijkstra algorithm
  - 计算到其它所有节点的最短路径
- 如何在拓扑改变时更新最短路径?
  - 周期性交换信息、按需触发

要点回顾：距离向量 vs. 链路状态

- 所交换的信息是全局的还是局部的?
  - 距离向量：每个节点告诉直接邻居所知道的所有信息
  - 链路状态：每个节点告诉所有节点所知道的邻域信息
- 信息更新的方式
  - 距离向量
    - 信息两两交换
    - 好消息传播快，坏消息传播慢
  - 链路状态
    - 信息在全网可靠洪泛
    - 可以通过SEQ和TTL的设置加速新消息的传播速度

THANKS

谢 谢 聆 听

Email: chenwang@hust.edu.cn

Website: http://www.chenwang.net.cn