网络层：路由协议

5.1.3 逐跳路由：扩散法和逆向学习法

5.4 路由协议

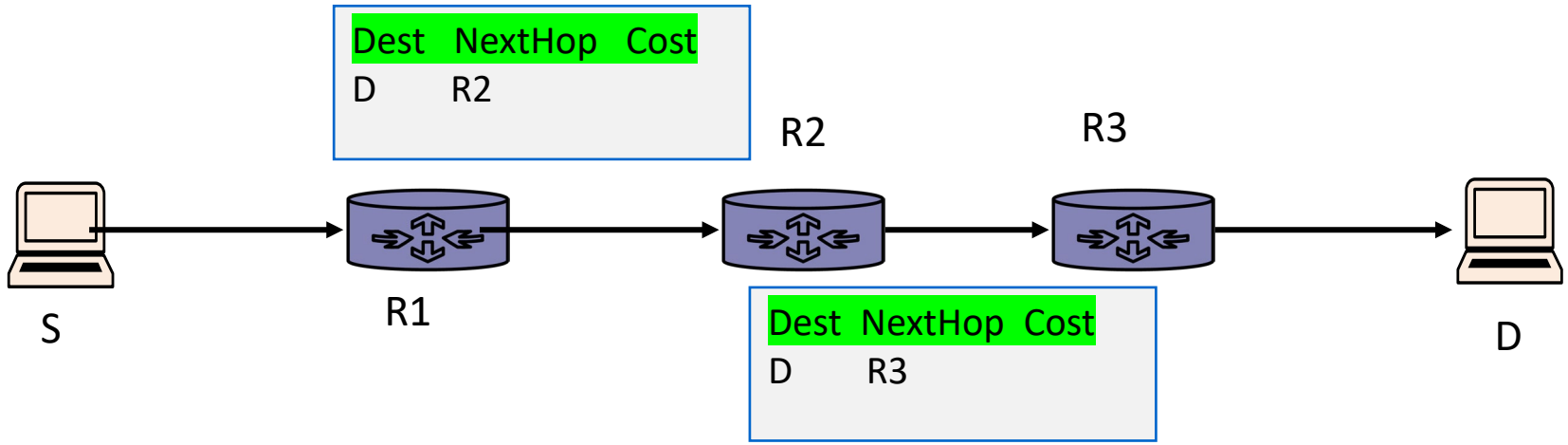
- 自治系统
- 链路状态路由
- 距离向量路由
- BGP路由协议

5.5.3 组播路由协议

逐跳路由(5.1.3)

逐跳路由：从源到目的地的路径由途中的节点共同维护

- 假设S到D的路径： S→R1→R2→R3...→Rn→D
- 每个节点只需要维护到目的地的下一跳节点信息即可
  - 可能短暂出现路由回路
  - 单播(unicast)路由：**单个目的节点**
  - 多播（组播, multicast）路由：一个或者多个源发送分组给**多个目的节点**，多个节点共同维护一棵组播树



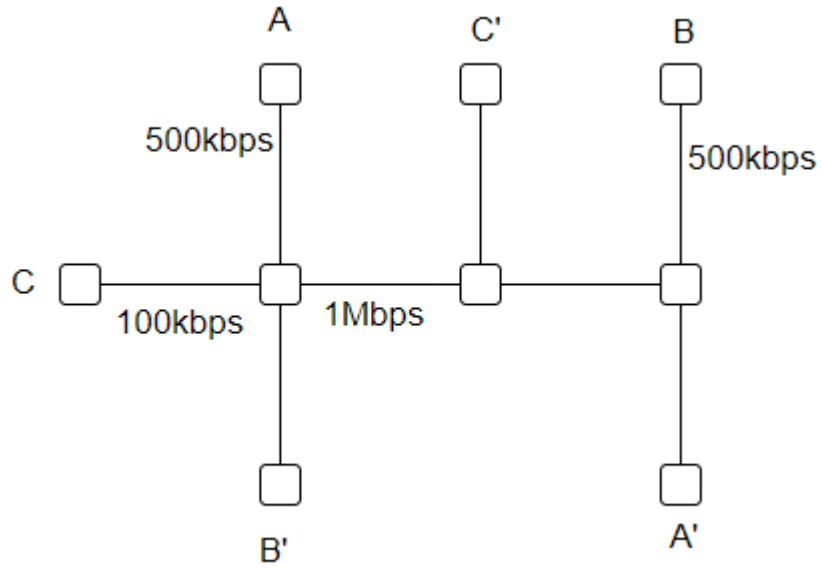
逐跳路由

- 性能度量：多条路由情况下选择哪条路由？
  - 最小花费路由：路径**花费(cost)最少**的路径
    - 可以是跳段数(hop count)
    - 可以考虑链路的数据速率、延迟、可靠性等
  - 策略路由：策略方面的因素，比如要求经过哪些或者绕过哪些区域，多种选择下基于某些路径属性进行选择等
- 如何了解路径信息：
  - 静态路由：按照预先定义好的静态信息来选择
    - 人工配置或者路由信息中心集中配置后分发
    - 拓扑变化无法及时响应
    - 仅适合于规模较小且拓扑稳定的网络
  - 动态路由：自适应路由
    - 根据网络的当前状态信息，运行相应的路由算法来决定
    - 可以是本地信息（扩散法和逆向学习法等），也可以是路由器之间交换的路由信息（定期交换或者有变化时交换、有数据传输时交换）
    - 在软件定义网络（Software-Defined Networks)中，路由器与(集中的）控制器之间交换信息

路由算法的要求

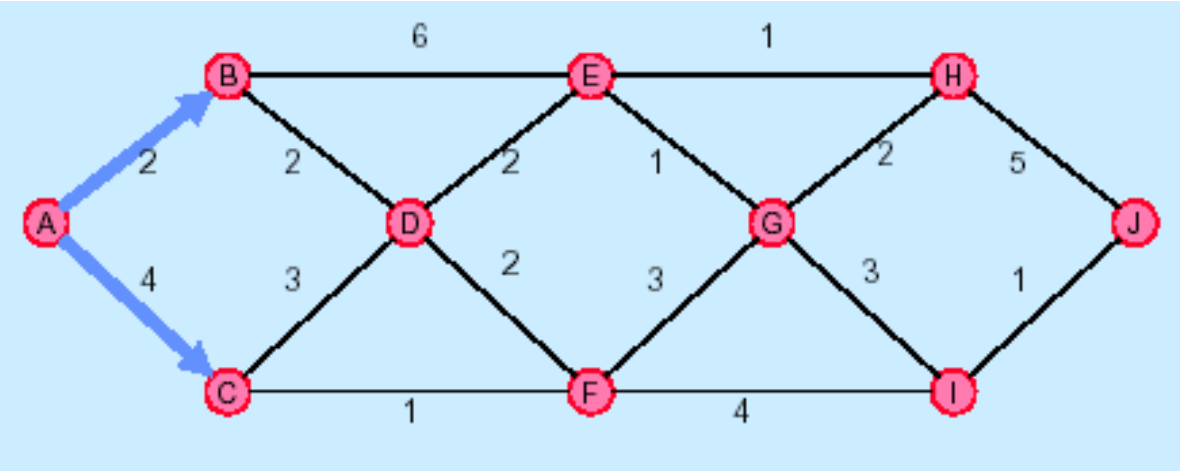
- 效率：有效利用网络资源
- 稳定：能够找到一条或多条稳定的路由，拓扑变化时能够较快地收敛
- 健壮：局部的故障不会影响整个网络，避免路由回路
- 公平：节点获得同等待遇，要在全局效率和局部公平间权衡
- 伸缩性：适应不同的网络规模

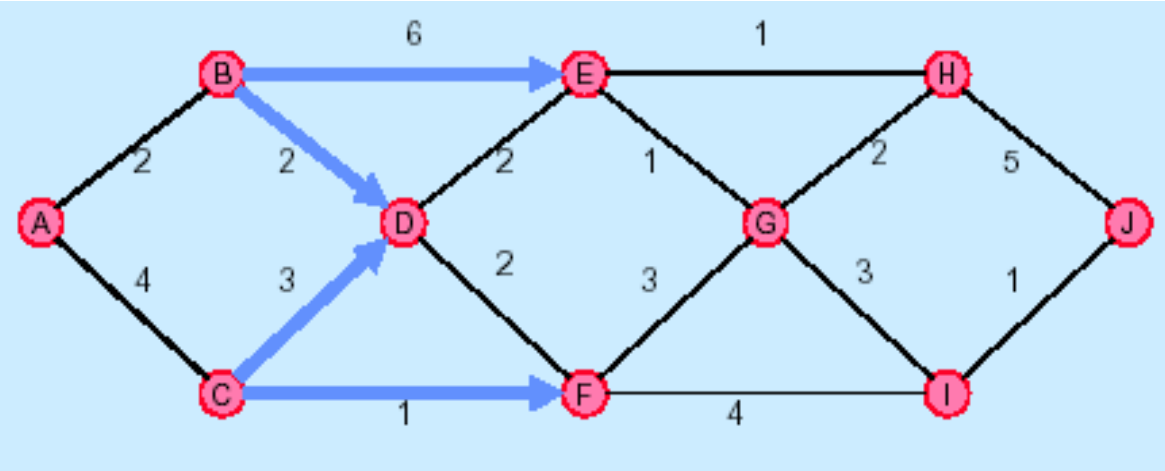
- A到A'、B到B'的负载数据速率为500kbps，C到C'的负载数据速率为100kbps
- 从效率的角度，C和C’之间的负载应该停止



扩散法（Flooding）

- 根据本地信息选择路由
- 将收到的分组向除到来链路外的所有其他链路转发
- 扩散法会将分组沿着各种可能的路径转发到所有节点
- 可用于：
  - 健壮性要求较高的场合，比如战地网络等
  - 判断可达性：只要连通则一定可达
  - 信息的分发：比如链路状态分组的扩散
  - 寻找节点间的最短路由：最早到达目的节点的路径





扩散法：抑制重复分组

- 节点计数器:
  - 源端在分组头部包含一个节点计数器
  - 每到达一个节点减1，为0时丢弃
  - 节点计数器初值可设置为网络的直径
- 记录分组扩散的路径
  1. 分组头部记录目前经过的节点列表。如果节点本身出现在列表中，则丢弃
  2. 分组头部包含源节点和顺序号，途中节点记录收到的分组的源节点和顺序号，如果分组与节点记录的分组接收信息匹配，则丢弃
- 选择扩散：一般要求有地理位置信息，选择可能向目的节点的方向

逆向学习法（backward learning）

- 仅仅根据本地信息选择
- 每个分组中包含源节点和经过节点的个数（或者**花费**）
- 节点接收到分组时，可了解到到源节点方向的路径花费，经过一段时间学习到最短路径
  - 基于分组中的源节点信息进行学习。考虑到源节点的路径：
    - 如果分组中的新路径比路由表中的路径更短，则更新
    - 如果路由表中没有相应的表项，则新增
  - 该分组如何进一步往前转发以到达目的节点？
    - 如果路由表中有到目的节点的路由，则转发给该路由的下一个节点
    - 否则类似于扩散法，将其转发给所有链路或者除了到来链路外的所有链路
- 节点仅记录较好路径的变化，原有最短路径上的节点或者链路崩溃时无法了解到，需要重新刷新节点对当前网络的了解
  - 路由表中的路由表项定期超时移走
  - 频繁刷新→学习最短路径需要一段时间，从而可能使用不确定的路径
  - 周期过长→网络变化的反应太慢

5.1.3 逐跳路由：扩散法和逆向学习法

5.4 路由协议

- **自治系统**
- 链路状态路由
- 距离向量路由
- BGP路由协议

Dijkstra算法

链路状态路由协议

OSPF

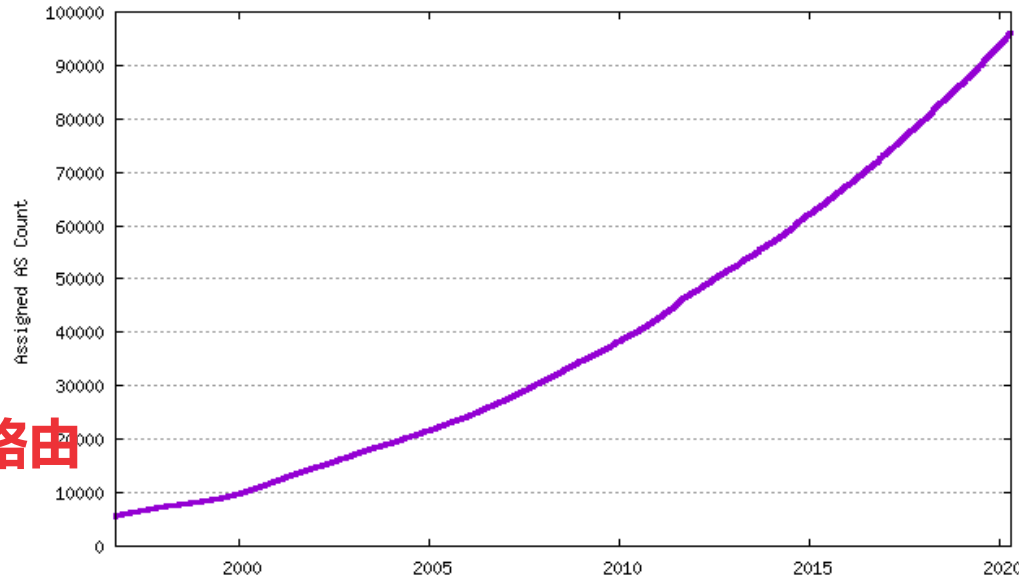
5.5.3 组播路由协议

自治系统

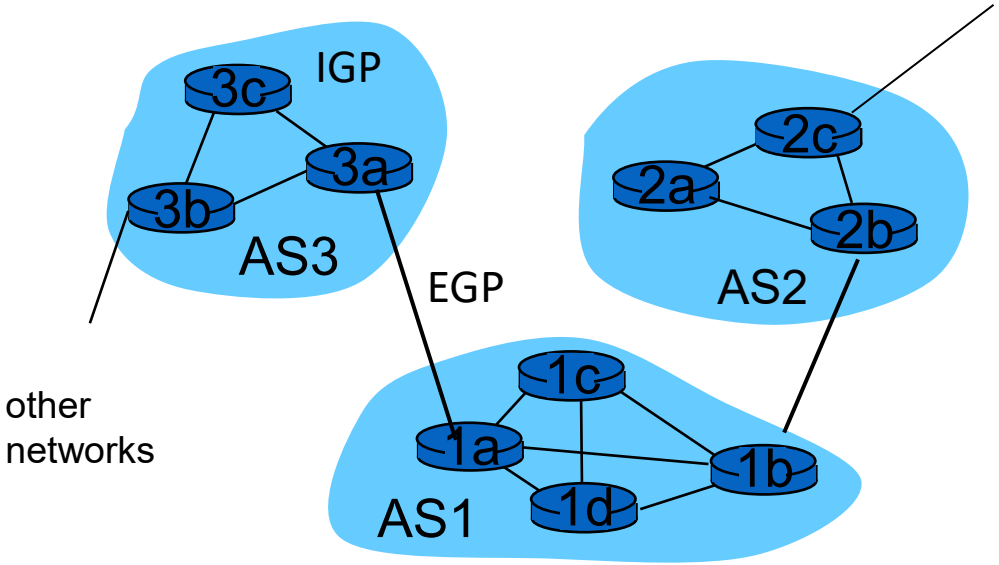
- Internet一般采用分布式动态路由协议，假设
  - 网络拓扑会动态变化。大多数情况下网络拓扑稳定
  - 节点存储和处理能力相对较强

Internet规模大，路由消息、路由表和收敛性都要求采用**层次路由**

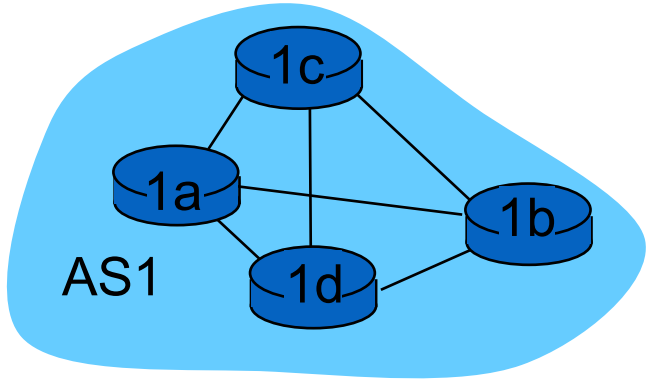
- 分成多个**自治系统AS**（Autonomous System）
  - 由一个或者多个独立的**管理实体**控制的采用**相同路由策略**的网络和路由器组成
  - 分配一个自治系统编号（ASN）。最初为16比特
    - 64512~65534为内部自治系统编号
    - 2007年RFC 4893扩充为32比特，X.Y来表示
- 自治系统内部选择各自的内部路由协议（IGP）
  - 可以采用不同的路由协议，可以采用不同的metric
- 自治系统之间运行外部路由协议（EGP）
  - 将AS内部的网络汇集在一起发布到其他AS
  - 策略路由



AS编号增长



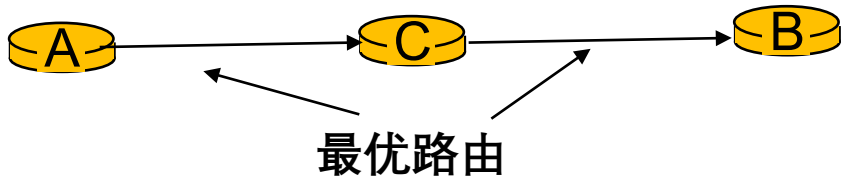
网络拓扑与图的对应关系



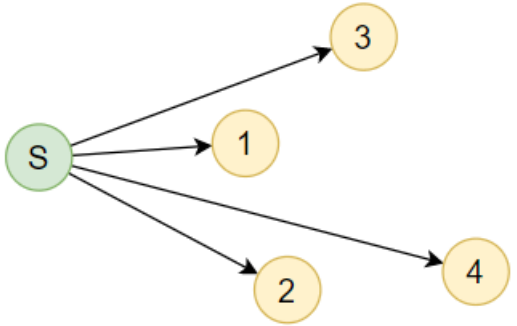
网络拓扑用图表示：

- 节点对应路由器
- 边对应路由器之间的链路
- 边的权重对应链路的花费
- 花费可能是距离、信道带宽、平均通信量、通信开销、队列平均长度、测量到的时延和其它一些因素的综合
- 源到目的地之间的路径由一系列链路组成：S-R1-R2-R3…D
- 最小花费路径：源到目的地之间所有可能的路径的链路花费总和最小。（若所有的链路花费相同，则最小花费路径即最短路径）

Dijkstra算法：基本思想



- 基于最优原理：
  - 如果节点A到节点B的最优路由经过了节点C (A→C→B)，则该路由上的A到C和C到B分别也是节点A和节点C、节点C和节点B的最优路由
- 由于所有**链路花费非负**，可以经过迭代的方法，按照**到源节点的最短路径花费**的**递增的顺序**，逐步寻找那些离源节点越来越远的节点
  - 第1轮，从源的直接邻居中寻找到离源节点最近的节点
  - 第2轮，从**标记节点**(源和第1轮找到的节点)的邻居中寻找到离源节点次近的节点
  - 第3轮，从**标记节点**(源和第1/2轮找到的节点)的邻居中寻找到离源节点第三近的节点
  - …
  - 重复直到所有节点都已标记为止
  - **标记节点集合**：已知到源的最短路径的节点的集合，初始仅仅包括源，每轮选择的节点加入到该标记节点集合



Dijkstra算法：基本步骤

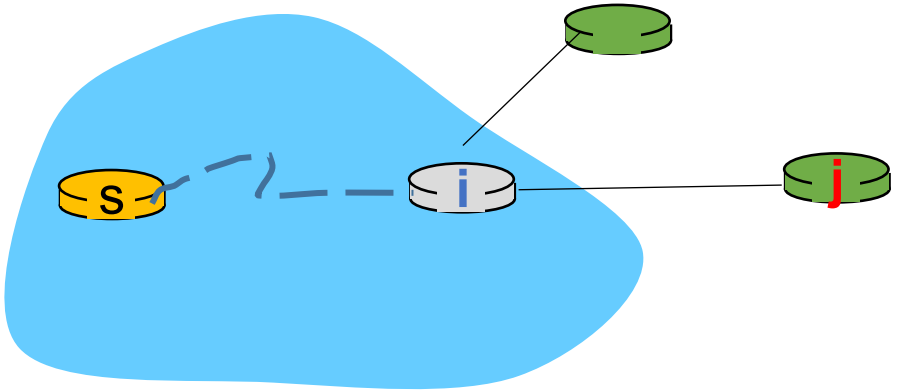
N = 网络中所有节点的集合

s = 源节点

C_{ij} = 节点i与j之间链路花费；若两个节点间没有直接连接则为∞

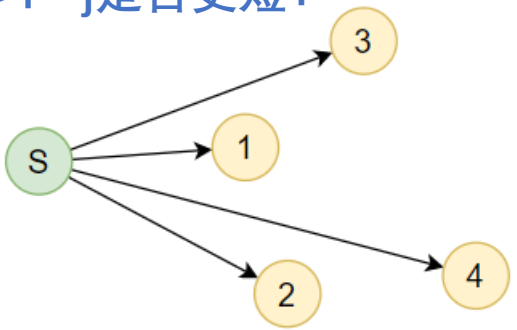
M = 已标记节点的集合

D_n = 算法求得的当前从源s到节点n的最短路径的花费



新路径 S--....->i--j是否更短?

$\min(D_j, D_i+C_{ij})$



(1) 初始化: **标记节点集合**与当前最短路径花费

$M = \{s\}$

$D_s = 0$

$D_j = C_{sj}$ for $j \neq s$

重复(2)(3)直到 M=N

(2)**选择新的标记节点i**加入到M中

寻找节点$i \in M$，使得$D_i = \min_{j \in M} D_j$

把i加入到M中

选择不在标记节点集合中那个离源最近的节点（如果有多种选择时可以任选其中一个）

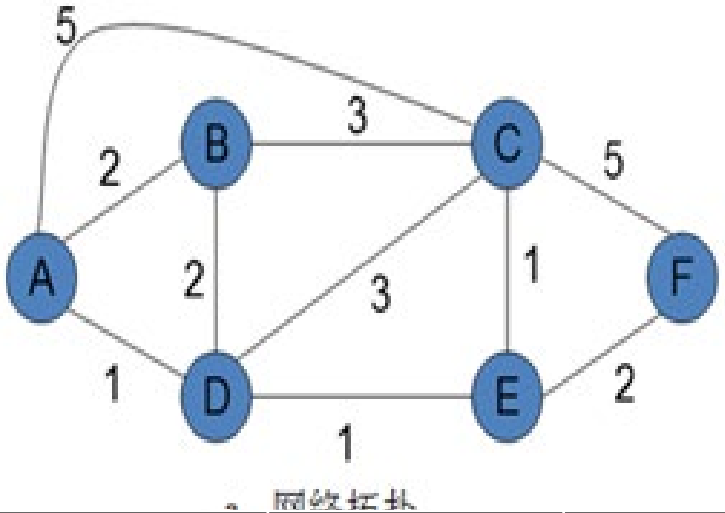
(3) **更新最短路径花费**

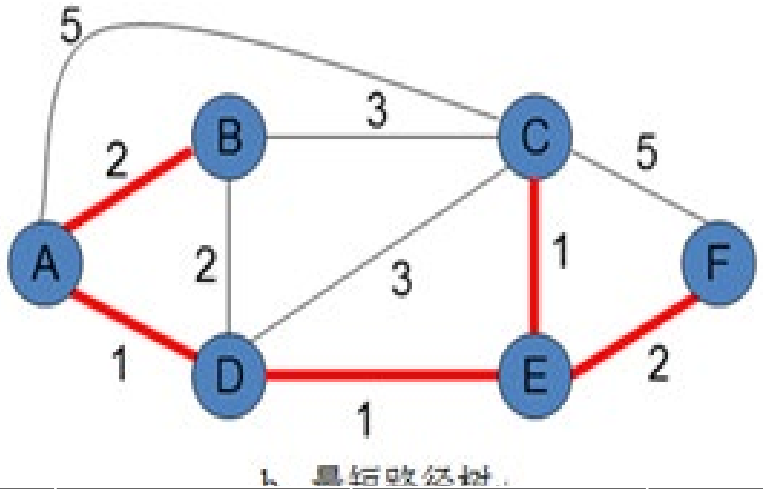
对所有i的邻居节点$j \in M$ 更新$D_j = \min[D_j, D_i+C_{ij}]$

如果后一项为最小值，则从S到j的路径变为从**S到i**的路径再加上从**i到j**的链路

▲节点j只需要记录前一节点i

Dijkstra算法：示例



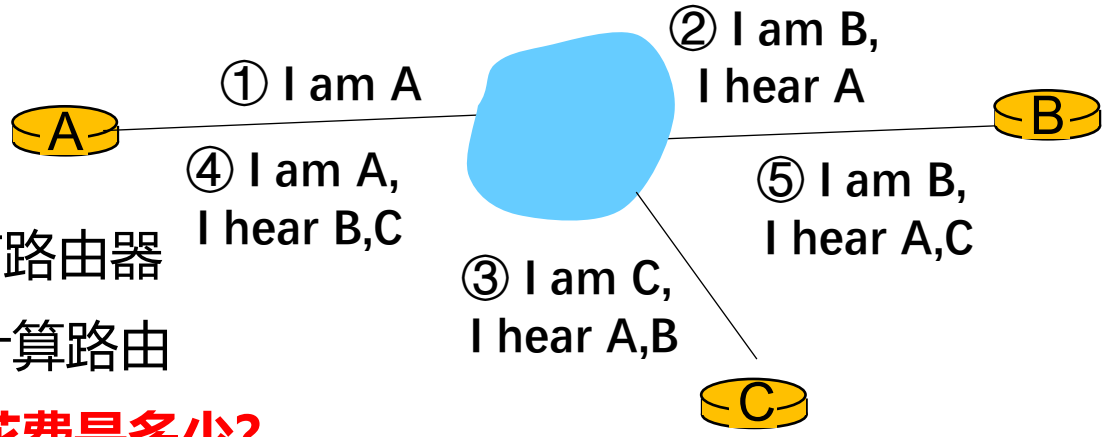


D(N): 当前求得的**节点N到源节点**的最短路径花费

P(N)：最短路径上的前一节点

步骤	标记节点	D(B),P(B)	D(C),P(C)	D(D),P(D)	D(E),P(E)	D(F),P(F)
0	A	2,A	5,A	1,A	-	-
1	AD	2,A	4,D		2,D	-
2	ADE	2,A	3,E			4,E
3	ADEB		3,E			4,E
4	ADEBC					4,E
5	ADEBCF					

链路状态路由协议：概述



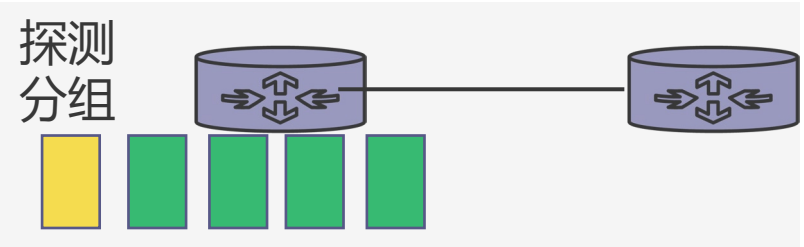
- 路由器维护本地的**链路状态信息**
- 每个节点将其链路状态信息通过**扩散的方法**传播给所有路由器
- 每个节点根据了解到的当前**全局拓扑采用Dijkstra算法**计算路由

如何了解路由器的链路状态信息？ 邻居是谁？到邻居的花费是多少？

- 路由器定期(比如10秒)在链路上发送Hello分组来**发现和维护邻居（邻居发现和邻居可达性）**
  - Hello分组给出最近（dead间隔）收听到的发送过Hello分组的**邻居路由器列表**
  - 如果路由器发现收到的Hello分组包含自身路由器的ID，其到邻居路由器的**双向链路**是相通的
  - 考虑到分组可能丢失，采用k-out-of-n机制，(k=1, n = 4) 即hello间隔为10秒，dead间隔为40秒
    - dead间隔期间(n次hello间隔) 只要有k(k一般为1)次收到，则认为链路相通
    - 如果dead间隔期间(n次hello间隔)都没有收到R发的Hello分组，则认为R已经断开

R2

- **到邻居节点的花费**可通过**手工配置**或通过**测量延迟**的方法来获得
  - 根据链路带宽设置，带宽越高，花费越小
  - 发一个探测分组，邻居节点响应，分组来回的时间/2为链路延迟
    - 测量延迟时是否考虑负载因素：排队还是队头（不考虑负载）时采样？
    - 考虑负载因素可以进行负载均衡，但可能产生不稳定路由



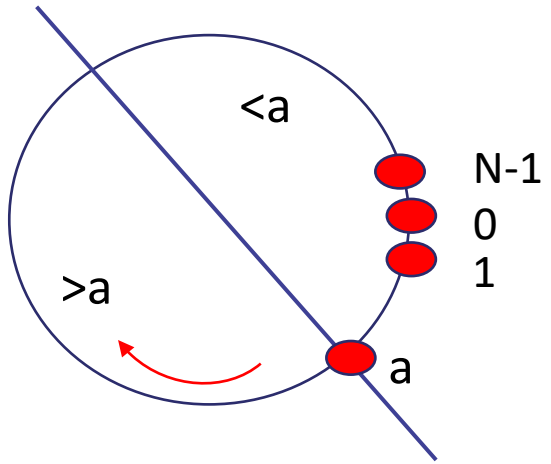
链路状态路由协议：扩散

- 链路状态分组LSP(Link State Packet)：
  - 每个路由器周围的链路情况
  - 需要传播给所有路由器

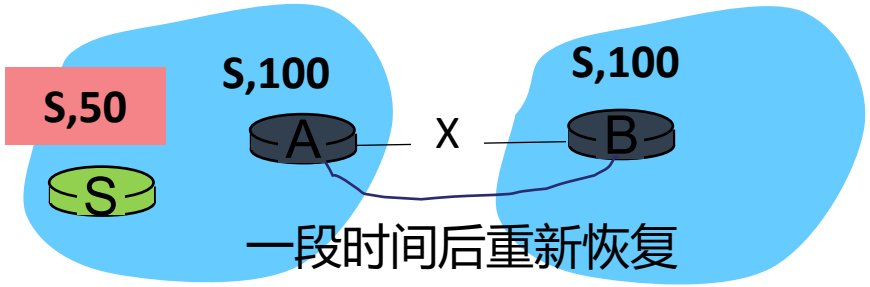
扩散法：记录分组扩散的路径

1. **分组头部记录**目前经过的节点列表。如果节点本身出现在列表中，则丢弃
2. 分组头部包含源节点和顺序号，**途中节点记录**收到的分组的源节点和顺序号字段

- 链路状态分组LSP的传输不能依赖路由表: chick and egg problem
- 采用扩散法，如何避免重复的分组?
  - 链路状态分组：路由器ID+顺序号。顺序号更大，链路状态更新
  - 链路状态数据库LSDB：每个路由器记录收到的各个路由器的LSP，每个路由器仅仅保留其最大顺序号的LSP
  - 过时或重复的信息（比较顺序号）丢弃，新的信息被扩散
  - 顺序号重复使用→32比特的顺序号
    - 假设顺序号增长的幅度不会太大，在LSP被移走之前顺序号增长的幅度小于整个顺序号空间N的一半
    - 对于每个顺序号a，一半顺序号大于a，另一半小于a
      |a—b|<N/2且a<b，或|a—b|>N/2且a>b，则认为a小于b



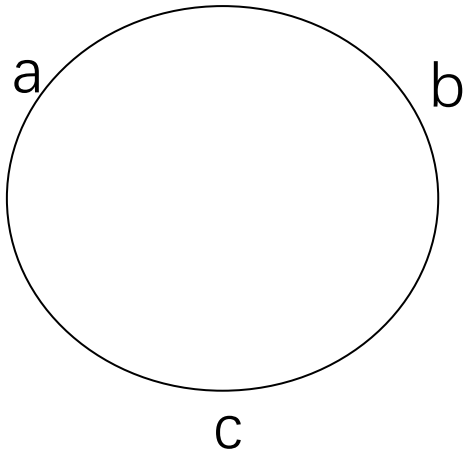
链路状态路由协议：扩散，顺序号+年龄(age)



- 在扩散过程中LSP可能会丢失
  - 采用软状态机制：定期发送一个新的LSP（顺序号加1）
  - 采用可靠扩散的机制：重传确认
- 路由器可能
  - 崩溃：顺序号从0开始，新的LSP比崩溃前的LSP顺序号要小
  - 链路断开：分割成两个不连通的部分，过一段时间重新恢复后(一半还是老的顺序号，一半是新的顺序号，由于可能回绕)无法比较大小
- 解决方法：
  - 增加**年龄（age)字段**，以合适的速度减少age
  - age=0表示LSP超时，需要丢弃，不用于路由计算
    - 考虑到不同路由器的时钟可能会不同，LSP超时时，扩散顺序号相同但age=0的LSA
    - 如果收到LSP的age为0，且顺序号与保存的LSP相同时，将保存的LSP的age设置为0(表示需要舍弃)，然后扩散给其他邻居
    - 如果收到新的LSP的age不为0，且保存的LSP的age为0时(顺序号不比较)，替换为新的LSP
  - 崩溃重启时等待足够长的时间(超过最大age)，确保崩溃前发送的LSP在其他节点处超时

ARPANET采用的链路状态路由协议实现的问题

- ARPANET中最初的实现
  - 考虑到扩散时分组可能丢失，路由器可能动态加入和退出， LSP每隔60秒定期发送
  - LSP包括顺序号+age，最大age =7*8= 56秒
  - 路由器中保存的LSP定期(每隔8秒)减少age
  - 重启的路由器等待90秒后再发送LSP
- 某一天ARPANET陷入“瘫痪”
  - 一个路由器S出现硬件（内存）故障，发送了三个LSP
    - 顺序号为a, b, c
    - 满足a < b < c < a
  - 路由器R:
    - 收到顺序号为a的LSP，扩散
    - 收到顺序号为b的LSP，b > a，扩散
    - 收到顺序号为c的LSP，c > b，扩散
    - 收到顺序号为a的LSP（之前扩散出去的分组被扩散回来）， a > c，扩散
    - 在这个过程中LSP的age并没有减少
  - 每个路由器采用Dijkstra算法计算路由表，其队列中充满来自于S的LSP



链路状态路由协议

- OSPF协议, 顺序号为32比特的有符号整数 [-N+1, N-1]，$N=2^{31}$
- OSPF协议中**age初始为0**，在传输和保存途中age会逐步增加，到MaxAge时链路状态过期

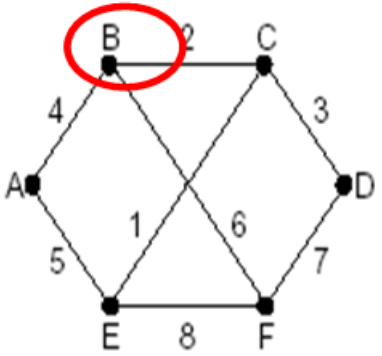
- 解决方法：**顺序号永远不回绕**
- LSP(OSPF术语为LSA，Link State Advertisement ）的传播采用**可靠扩散**，即扩散给邻居时要求**确认**
- **最大age为1个小时**以提高效率
  - 每隔30分钟**定期产生一个新的LSA**，**有变化时**产生新的LSA, 顺序号回绕需要极长的时间
  - LSA**初始age为0**，在转发时也会增加**age(至少加1),** 在LSDB中保存时也会增加，直到等于MaxAge。此时扩散该LSA将其从其他路由器中也清除(考虑时钟不同步)
- 当路由器S的LSA顺序号到达最大值时, 应等老的链路状态“过期”，才能发送新的最小顺序号的LSA
  - 可通过扩散顺序号为最大值，但是**age=MaxAge**的LSA来清除老的LSA
- 拓展：LSA包含了一个checksum，采用Fletcher算法，在LSDB中的LSA要求每5分钟验证checksum，防止硬件错误
- 拓展：**新启动**的路由器不需要等待1个小时，可以**立即发送**LSA
  - 如果一个**路由器收到LSA的源为自身**，而且LSA顺序号比自身的更加新时，两种策略：
    (1) 发送新的LSA, 提升顺序号为收到的LSA顺序号+1
    (2) 提早老化该LSA， 扩散age=MaxAge的LSA

OSPF
Link State Update

Header
LSA
LSA
LSA

链路状态路由协议：可靠扩散优化

- 节点间的路由信息交换采用确认机制: 扩散LSP时要求ACK
- 收到路由分组后不是马上扩散，而是等待一段时间，丢弃这段时间来的重复和过时的分组，减少负载开销
  - 路由器有k个邻居，对于要扩散的每个LSP，对于每个邻居，维护2个标记（共2k个标记），分别对应是否要SEND到该邻居，是否要发送ACK给该邻居
  - 路由器从X收到S的LSP时
    - 如果顺序号更新，则接受，保存LSP，设置ACK X，设置SEND给其它路由器
    - 如果更老，则忽略
    - 如果相等，则设置ACK X，清除SEND X
  - 轮流扫描ACK和SEND标志，发送实际的报文

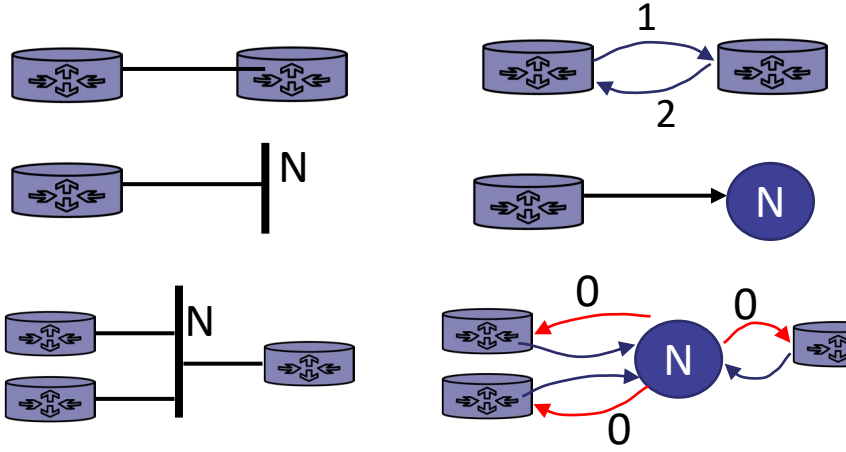


源	顺序号	Age	发送标志 A	发送标志 C	发送标志 F	ACK标志 A	ACK标志 C	ACK标志 F	LSP
A	21	60	0	1	1	1	0	0	A 扩散过来
F	21	60	1	1	0	0	0	1	F 扩散过来
E	21	59	0	1	0	1	0	1	A 和 F 扩散过来
C	20	60	1	0	1	0	1	0	C 扩散过来
D	21	59	1	0	0	0	1	1	C 和 F 扩散过来

开放最短路径优先协议OSPF（Open Shortest Path First）

- IS-IS(Intermediate System-to-Intermediate System): 为OSI参考模型中的CLNP(Connection-less Network Protocol)设计的链路状态路由协议
- OSPF v2，在RFC 2328定义, 运行在IP之上，协议号89
- OSPF v3，在RFC2740定义，支持IPv6
- 采用链路状态路由算法，收敛速度快
- 支持变长子网（路由消息包含了网络掩码），支持多种Metric，而不是RIP的Hop Count
- **支持各种类型的链路**
- 路由器和路由器之间采用**组播**来传递消息
  - 224.0.0.5 = AllSPFRouters
  - 224.0.0.6 = AllDRouters
- **支持区域，引入了层次路由**
- **支持各种路由，允许交换通过其他途径了解到的路由信息，比如到外部的路由**
- 支持多条最短路由间的负载均衡：等价多路径路由ECMP（Equal Cost Multipath Routing）
- 路由消息认证

OSPF： 多种类型链路

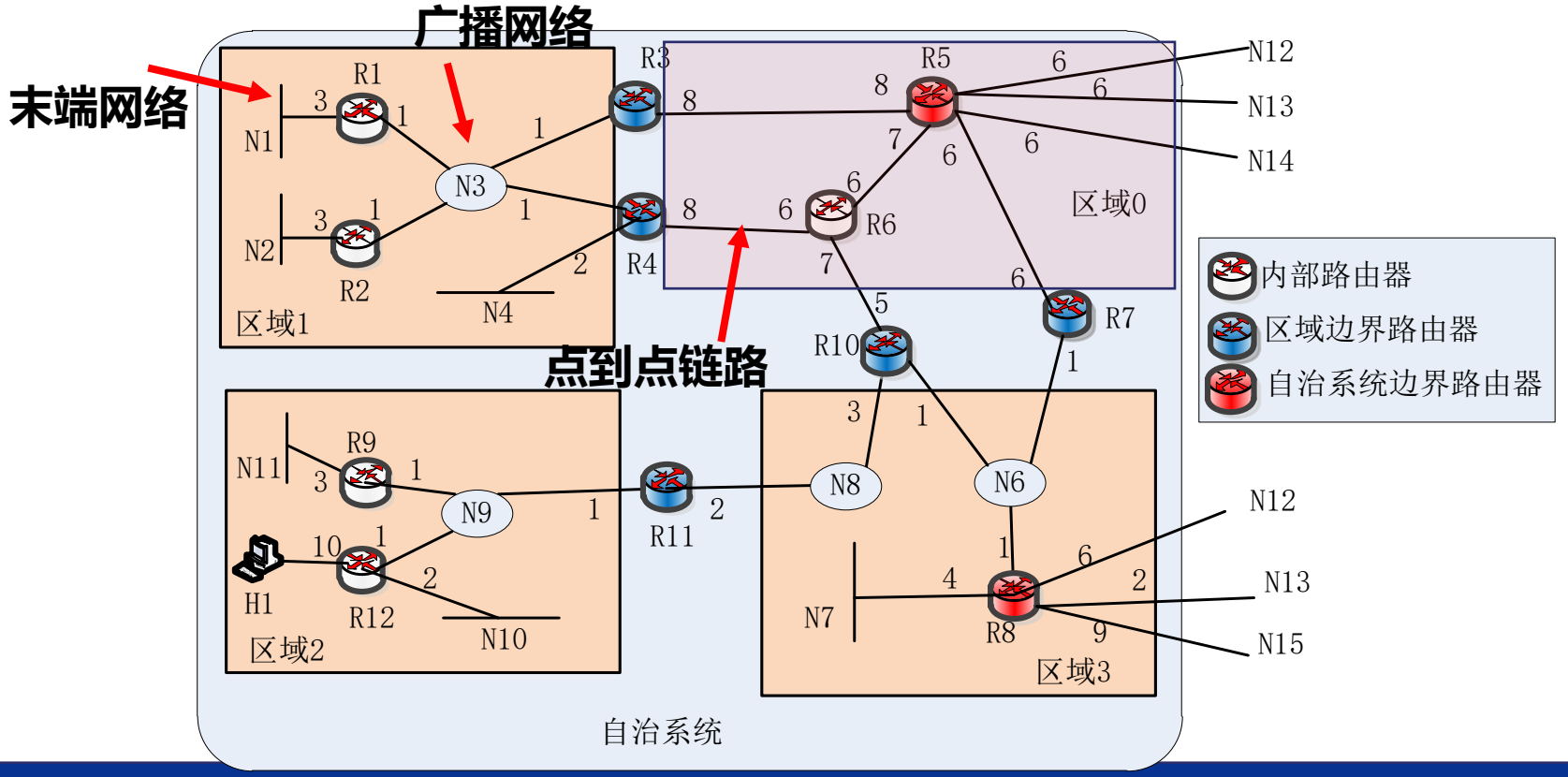


- **点到点链路**：连接两个路由器节点的双向边来表示
- **末端网络**（Stub network）：路由器节点到网络节点的单向边
- 多个路由器连接到**广播网络**
  - 一种选择是路由器之间的双向边
  - 引入网络节点，路由器节点通过双向边连接到网络节点，网络节点到路由器节点的边花费为0

拓展：NBMA（Non-broadcast multiple access）

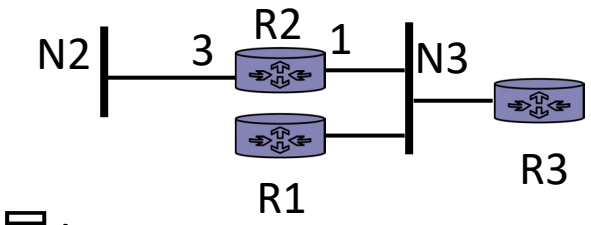
- 可以有多个路由器连接在该网络
- 但不是广播网络，有些路由器之间有逻辑通道(虚电路)
- 通过路由器之间的双向边描述
- 也可引入网络节点，但是需要人工配置

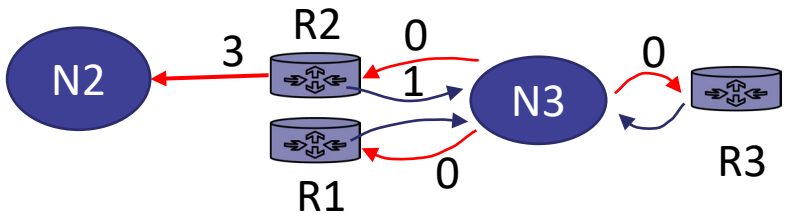
虚拟链路： 将一个区域通过另一个区域连接到主干区域



OSPF：链路状态通告LSA（Link State Advertisement）

- 链路状态通告(Link State Advertisement)： 路由器周围的链路状况，路由器ID+顺序号+age + 链路信息
- **路由器LSA**（第1类 LSA）描述路由器周围情况
- **网络LSA**（第2类 LSA）描述广播网络的链路状况



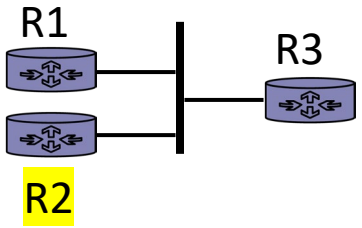


R2的路由器LSA	
目的	花费
N2	3
N3	1

N3的网络LSA	
目的	花费
R1	
R2	
R3	

- 邻居：
  - 连接到同一个链路上的路由器之间的关系
  - hello协议用于发现邻居和判断邻居可达性
- 对于广播链路而言，有多个路由器连接
  - 网络LSA由谁来发送?
  - 连接在广播链路上的路由器互为邻居。如果这些邻居之间互相扩散LSA，链路上会出现$O(N^2)$个相同的LSA

OSPF：广播链路



在广播链路上，R2为DR
R1 --> 224.0.0.6: LSA
R2 --> 224.0.0.5: LSA

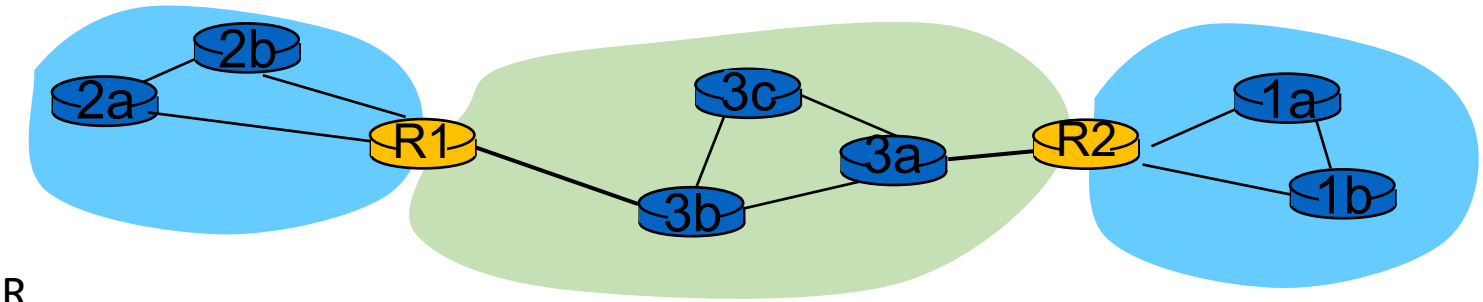
引入邻接(adjacency)关系的概念

- 点到点链路上的邻居关系等同于邻接关系
- 广播链路上选取**选取路由器DR**（Designated Router），考虑DR可能出现故障，还会同时选取备份选取路由BDR
  - **只有DR和BDR与广播链路上的其他路由器有邻接关系**
- 链路状态信息只有在**邻接关系的路由器间**交换
  - 组播地址224.0.0.5表示所有OSPF路由器，224.0.0.6表示DR和BDR路由器
  - 普通的节点在广播链路扩散LSA时首先传输给DR，然后由DR传输给所有OSPF路由器
- DR还负责扩散网络LSA
- DR和BDR的选取通过Hello协议来完成
  - 每隔10秒(可配置)发送Hello分组(目的为224.0.0.5)，该分组包括自身的ID、**优先级**以及最近收到过Hello分组的邻居ID列表，选取的DR和BDR，接口的网络掩码以及Hello和Dead间隔(要求链路上的路由器参数一致)
  - 比较所有节点的优先级，越高优先为DR，优先级相同时选择路由器ID更高的作为DR

OSPF：区域

OSPF支持**层次路由**，进一步划分成多个**区域(area)**：

- 每个区域通过一个32比特的区域ID来标识，也可用点十进制描述
- 主干区域的区域ID为0（0.0.0.0），其他区域为本地区域
- 每个区域内部都运行一个基本链路状态路由算法
- 区域内的路由器并不需要知道区域外的详细的拓扑情况
  - 区域内（intra-area）路由只需根据区域内部的路由信息来选择路由
  - 区域间（inter-area）路由**必须通过主干区域**，限制路由的规模，**避免路由回路**
  - 外部(external)路由通过自治系统边界路由器(**ASBR**)到外部网络
- **路由器的各个链路分别配置属于哪个区域**
- **区域边界路由器ABR**连接到多个区域
  - 运行多个OSPF实例，每个区域一个
  - ABR必须要连接到主干区域
- 主干路由器：主干区域内的路由器，包括ABR
- 内部路由器：不连接到主干区域的路由器
- ASBR：连接其他AS的路由器，可在本地或者主干区域
- ABR和ASBR会在相应区域扩散路由器LSA(Type 1 LSA)时，**标识其是ABR或者ASBR**

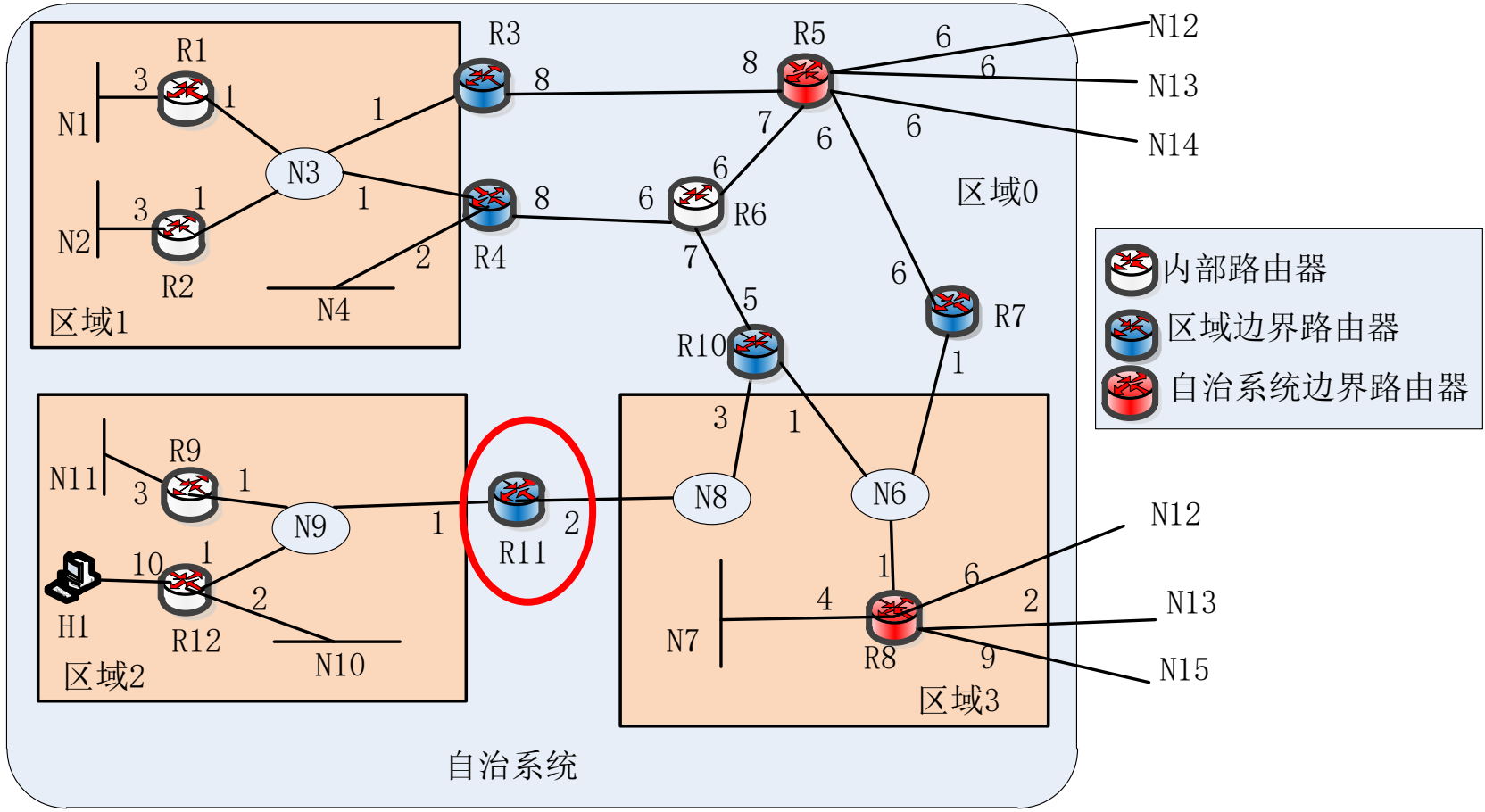


OSPF：虚拟链路

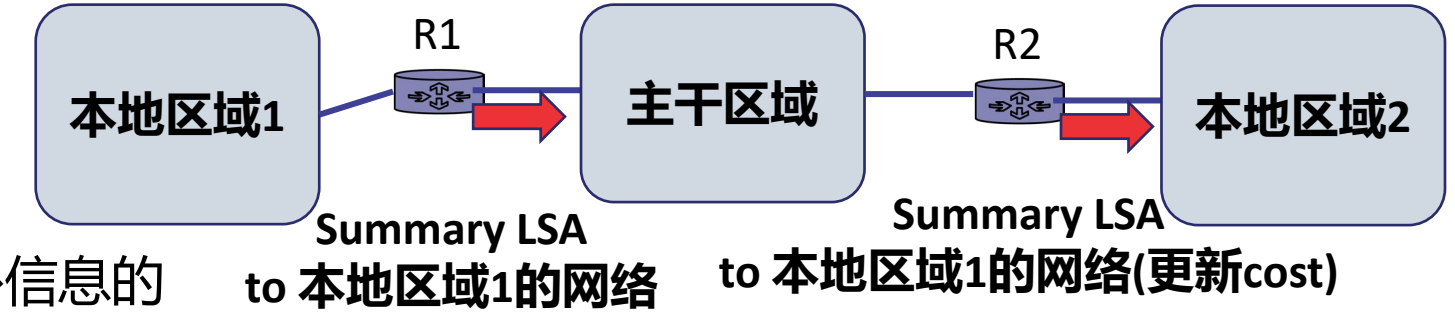
两个ABR在主干区域不连通时可以在这两个ABR之间建立一条**虚拟链路**

- 两个ABR应该连接到同一个本地区域，虚拟链路的路由通过该区域内的路由完成
- 虚拟链路属于主干区域
- 虚拟链路仅仅作为临时的措施

区域2的ABR R11需要和其他ABR（比如区域3的R10或者R7)建立虚拟链路



OSPF: Summary LSA



汇集LSA（第3类 LSA）：

- 由ABR产生，汇总所在区域的拓扑信息的链路状态，发布到其他区域
  - ABR可以到达哪个网络(network/mask)
  - 到该网络的花费是多少
- ABR将本地区域的汇集LSA扩散到主干区域
- ABR在主干区域会收到主干区域的路由器LSA和网络LSA，也会收到其他ABR发布的汇集LSA
- ABR汇总主干区域的拓扑信息，产生汇集LSA，将其扩散到本地区域
  - 到其他区域的网络N可以通过本ABR到达，其花费=主干区域中到目的区域某个ABR的花费 + 该ABR的汇集LSA中的花费

拓展：一个区域也可以配置将多个网络汇集成一个路由前缀，此时的cost为到这些网络的cost的最大值

!!! 教材图5.28(b)和(c)中到N1,N2,N3,N4的花费有误

R2的路由LSA

目的	花费
N2	3
N3	1

N3的网络LSA

目的
R1
R2
R3
R4

(a) 路由LSA和网络LSA

图(b)中给出了4个汇集LSA。

R4的区域1的汇集LSA

目的	花费
N1	4
N2	4
N3	1
N4	2

(b) 汇集LSA

R4的路由表

目的	花费	下一跳	路由类型
N1	4	R1	区域内
N2	4	R2	
N3	1	直连	
N4	2	直连	
N6	16	R6	区域间
N7	20	R6	
N8	18	R6	
N9	19	R6	
N10	21	R6	
N11	22	R6	
H1	29	R6	
N12	6	R6	类型2外部
N13	18	R6	类型1外部
N14	20	R6	类型1外部
N15	25	R6	类型1外部

(c) OSPF路由

OSPF: AS-External/ASBR Summary LSA

AS外部LSA（第5类LSA）

- ASBR汇总从其他自治系统收到的外部路由信息
  - ASBR连接到哪个(汇集后的)网络，**花费是多少**
- ASBR（可在本地或主干区域）将其扩散至自治系统，即首先扩散到其所在的区域，进一步由ABR扩散到其他区域
- 到外部路由的花费的定义
  - 类型1：外部路由花费和内部路由花费的和
  - **类型2**：仅包含外部路由花费，忽略内部路由花费。缺省
    - 适用于外部路由花费的数量级远大于内部路由花费的情况
- AS外部LSA在各个区域扩散时不更新花费
- 需要回答：有多个ASBR时选择哪个出口?
  - 有哪些ASBR?
  - 热土豆路由：采用类型2外部花费时，选择最近的ASBR
- **ASBR汇集LSA**（第4类 LSA），类似第3类汇集LSA
  - 由ASBR所在的区域的**ABR负责构造**并将其扩散到其他区域
  - ASBR的路由器LSA中包含相应的比特，给出其是否为ASBR
  - 到哪个ASBR的花费是多少?
  - ABR从主干区域收到ASBR汇集LSA后，进一步扩散到其他区域

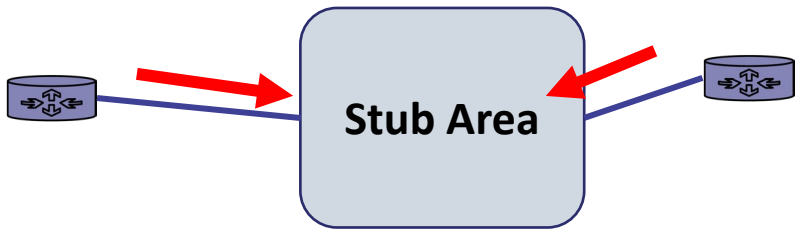
R4的路由表

目的	花费	下一跳	路由类型
N1	4	R1	区域内
N2	4	R2	
N3	1	直连	
N4	2	直连	
N6	16	R6	区域间
N7	20	R6	
N8	18	R6	
N9	19	R6	
N10	21	R6	
N11	22	R6	
H1	29	R6	
N12	6	R6	类型2外部
N13	18	R6	类型1外部
N14	20	R6	类型1外部
N15	25	R6	类型1外部

(c) OSPF路由

OSPF：各种LSA

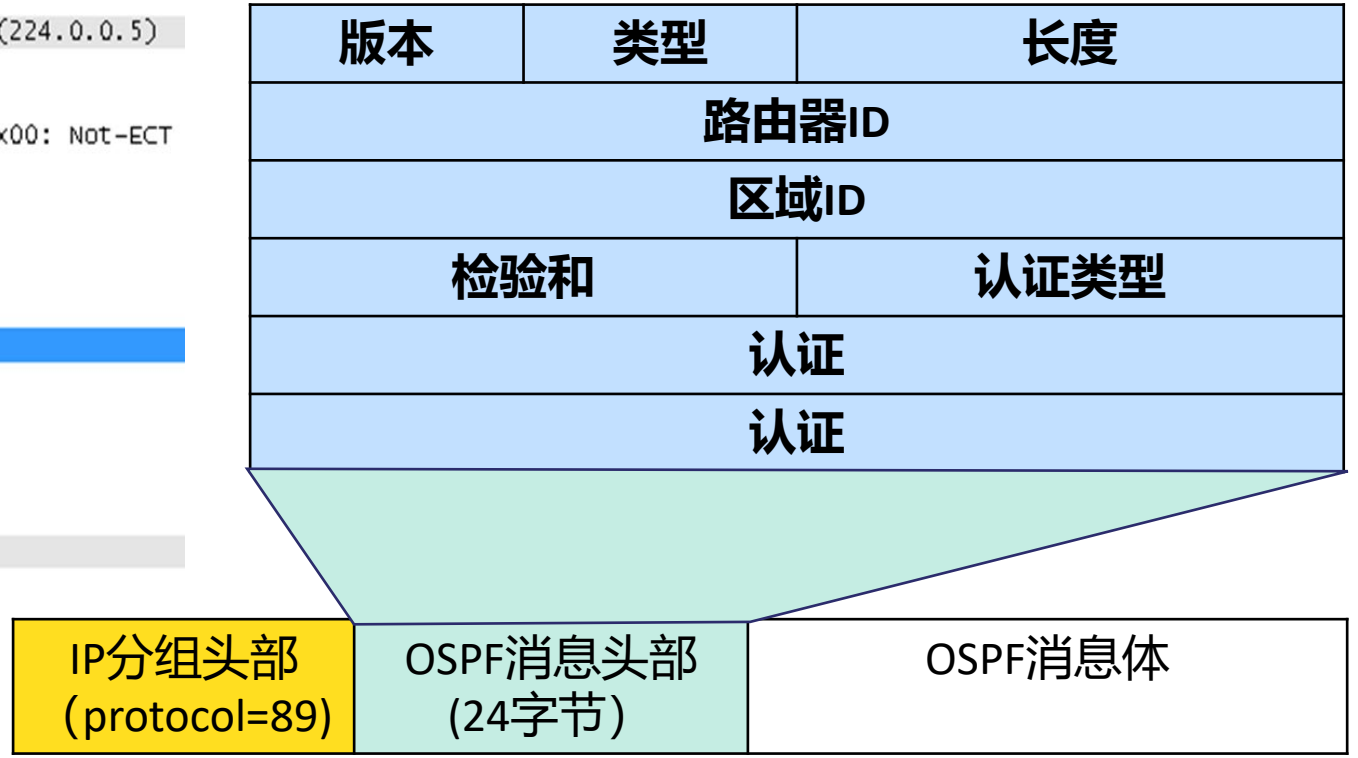
- 路由器LSA和网络LSA会在其所在的区域中扩散
- 汇集LSA由ABR生成扩散到主干区域，再经过其他区域的ABR扩散到其他本地区域
- AS外部LSA由ASBR生成扩散到其所在的区域，再经由ABR扩散到主干和其他本地区域
- ASBR汇集LSA由其所在区域的ABR生成，扩散到主干区域，再扩散到其他本地区域
- 末端区域(Stub Area): 没有ASBR的本地区域
  - 不需要往该区域扩散AS外部LSA和ASBR汇集LSA
  - ABR在该区域扩散一条缺省路由（Summary-LSA)即可
  - 在该区域中往AS外部的路由都基于发布的缺省路由决定，如果有多个ABR，采用热土豆路由
- **（补充内容）完全末端区域（Totally Stub Area）**：该区域只有**1个ABR，且该区域没有ASBR**
  - 并不需要往该区域扩散**汇集LSA**、AS外部LSA和ASBR汇集LSA
  - ABR在该区域扩散一条缺省路由即可



OSPF消息

所有消息都会包括产生该消息的路由器ID以及该消息所属的区域ID，同时还包含认证字段以确认该消息的确是一个合法的OSPF路由器产生的

```
Internet Protocol Version 4, Src: 12.12.12.1 (12.12.12.1), Dst: 224.0.0.5 (224.0.0.5)
    Version: 4
    Header Length: 20 bytes
  ⊞ Differentiated Services Field: 0xc0 (DSCP 0x30: Class Selector 6; ECN: 0x00: Not-ECT
    Total Length: 76
    Identification: 0x0002 (2)
  ⊞ Flags: 0x00
    Fragment offset: 0
    Time to live: 1
    Protocol: OSPF IGP (89)
  ⊞ Header checksum: 0xc085 [validation disabled]
    Source: 12.12.12.1 (12.12.12.1)
    Destination: 224.0.0.5 (224.0.0.5)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
Open Shortest Path First
  OSPF Header
      Version: 2
      Message Type: Hello Packet (1)
      Packet Length: 44
      Source OSPF Router: 1.1.1.1 (1.1.1.1)
      Area ID: 0.0.0.0 (0.0.0.0) (Backbone)
      Checksum: 0xea9c [correct]
      Auth Type: Null (0)
      Auth Data (none): 0000000000000000
```



- Hello
- Database Description
- LS Request
- LS Update
- LS ACK

OSPF消息

所有消息都会包括产生该消息的路由器ID以及该消息所属的区域ID，同时还包含认证字段以确认该消息的确是一个合法的OSPF路由器产生的

- Hello分组：链路上定期（缺省10秒）发送来发现和维持邻居，在广播链路上选取DR和BDR
- 数据库描述（Database Description）
  - 链路开启或恢复时在链路上发送以同步链路状态数据库
  - 描述各自拥有的LSA的当前顺序号，从而决定谁的LSA更新
  - 拓展：
    - 首先交换空的DD消息，路由器ID更高的成为Leader，控制DD的交换过程
    - 采用停等协议+捎带确认：
      - Leader发送携带顺序号的DD，Follower发送相同顺序号的DD以确认
      - 持续直到双方发送了More=0的DD为止
- 链路状态请求（Link State Request）
  - 要求邻接路由器传递特定的LSA
- 链路状态更新（Link State Update）
  - 定期（缺省30分钟）或者有变化时扩散
  - 一个链路状态更新消息可包括多个LSA
- 链路状态确认（Link State ACK): 链路状态采用可靠扩散，在收到一个Update消息时发送链路状态确认消息

开放最短路径优先协议OSPF（Open Shortest Path First）

- OSPF v2，在RFC 2328定义, 运行在IP之上，协议号89
- OSPF v3，在RFC2740定义，支持IPv6
- 采用链路状态路由算法，收敛速度快
- 支持变长子网（路由消息包含了网络掩码），支持多种Metric，而不是RIP的Hop Count
- **支持各种类型的链路**
- 路由器和路由器之间采用**组播**来传递消息
  - 224.0.0.5 = AllSPFRouters
  - 224.0.0.6 = AllDRouters
- **支持区域，引入了层次路由**
- **支持各种路由，允许交换通过其他途径了解到的路由信息，比如到外部的路由**
- 支持多条最短路由间的负载均衡：等价多路径路由ECMP（Equal Cost Multipath Routing）
- 路由消息认证

5.1.3 逐跳路由：扩散法和逆向学习法

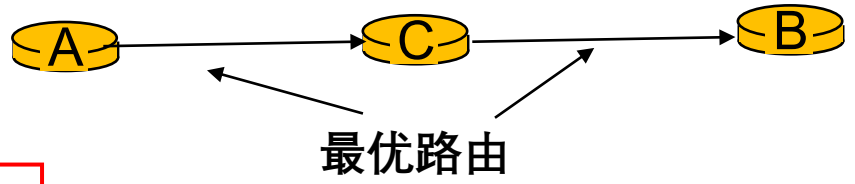
5.4 路由协议

- 自治系统
- 链路状态路由
- **距离向量路由**
  - 距离向量路由算法
  - 无穷计数问题
  - 距离向量路由协议RIP
- BGP路由协议

5.5.3 组播路由协议

距离向量路由（Distance Vector Routing）

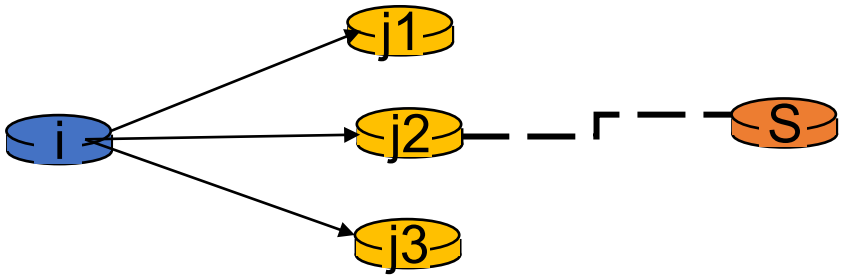
- 最优原理：如果节点A到节点B的最优路由经过了节点C (A→C→B)，则该路由上的A到C和C到B分别也是节点A和节点C、节点C和节点B的最优路由



- Bellman算法

$$D^{i}(S) = \min_{i\text{的邻居}j}[i\text{通过}j\text{到}S\text{的最短路径}] = \min_{i\text{的邻居}j}[C(i,j) + D^{j}(S)]$$

$$= \min_{i\text{的邻居}j} D^{i}(S,j)$$



i到S的最短距离

迭代实现

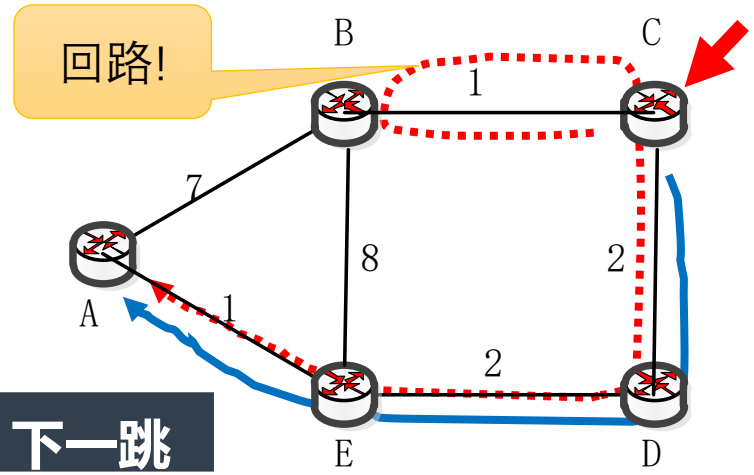
- 初始：每个节点知道其到直接邻居的花费
- 多次迭代计算：
  - 每个节点根据上一轮计算出来的邻居节点到目的地的花费，确定其到目的地的最短花费
  - 由于节点到S的最短距离为非递增的，k轮计算的距离为最多经过k跳的最短路径
- 当最短距离不变的情况下迭代结束
- 最多迭代次数为节点的个数

二维距离表: 节点i通过邻居j到S的最小花费

目的地	j1	j2	j3
S1	3	5	4
S2	9	8	12

目的地	花费	下一跳
S1	3	j1
S2	8	j2

距离向量路由：距离表和路由表



(a) 拓扑结构

C
D (Dest,Next Hop)

目的	B（通过B到）	D（通过D到）
A	7 loop!!	5
B	1	5
D	4	2
E	6	4

C
D (Dest)

目的	花费	下一跳
A	5	D
B	1	B
D	2	D
E	4	D

距离表中的第一行，到目的地A可通过B或者D到达

$D^C(A,D) = C(C,D) + D^D(A) = 2 + 3 = 5$　　C→D→E→A

$D^C(A,B) = C(C,B) + D^B(A) = 1 + 6 = 7$　　C→B→C→D→E→A　有回路

距离向量路由：分布式的Bellman-Ford算法

分布式实现

- 初始路由表中仅仅包含
  - 包含到直接邻居的路由，距离为定义的链路花费
- 分布式：每个节点只需把路由表(到目的地的距离是多少)传递给直接邻居
- 异步：每个节点独立**定期**发送路由表，或者在拓扑有变化时**触发更新**，节点间路由交换步骤无需同步
- 迭代：
  - 每个节点收到邻居节点来的路由表后重新计算路由表（寻找通过直接邻居到达目的地中的最短路径）
  - 迭代计算过程中无需额外的信号来通知结束，迭代过程在没有新的信息更新时自动收敛结束

迭代实现

- 初始：每个节点知道其到直接邻居的花费
- 多次迭代计算：
  - 每个节点根据上一轮计算出来的邻居节点到目的地的花费，确定其到目的地的最短花费
  - 由于节点到S的最短距离为非递增的，k轮计算的距离为最多经过k跳的最短路径
- 当最短距离不变的情况下迭代结束
- 最多迭代次数为节点的个数

$$D^{i}(S) = \min_{i\text{的邻居}j} [i\text{通过}j\text{到}S\text{的最短路径}] = \min_{i\text{的邻居}j} [C(i,j) + D^{j}(S)]$$

距离向量路由：算法描述

假设节点为x，距离向量d(x,y)记录了x到y的距离

初始化：更新距离向量：对于所有邻居y，距离向量d(x,y)=c(x,y)，下一跳为y

定期或者触发更新发送距离向量给所有邻居节点

迭代计算:

Repeat {

当节点x到邻居节点的链路花费改变或者收到邻居节点的距离向量时：

For 所有邻居 y

For 所有目的地 z {

d ← c(x,y) + d(y,z)

if (d < d(x,z)) { // 通过y到达目的地z的路径更短

距离向量d(x,z) ← 距离为d,下一跳为y

}

}

如果d(x,z)改变(触发更新）或者定期更新计时器超时，发送距离向量给所有邻居

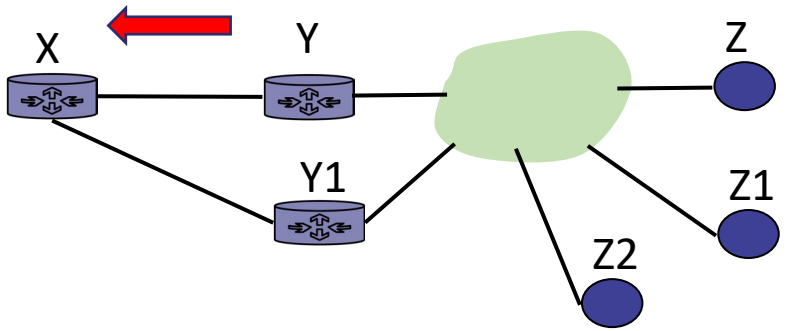
}

- ✓ 节点x记录所有邻居y最近发送的距离向量$d(y,z) = D^y(z,*)$
- ✓ 节点基于保存的来自于邻居的距离向量以及其到邻居的花费，计算出路由表

距离向量路由：

- 上述的分布式Bellman算法要求每个节点记录所有邻居y最近发送的距离向量d(y,z)，在节点数相对较多时会带来较多的存储开销
- 实践中一般只要求每个节点记录来自于邻居的距离向量中最好的一个，也就是说只需要记录最短路径（路由表 d(x,z)）：并没有改变算法的收敛特性，而且收敛的速度也不会有大的影响
- 相应的算法，改为：x如果从邻居y收到路由信息，看看通过邻居y到某个目的地z的路由是否更好，并更新相应的路由表项：
  - 以前无到该目的地的路由，新增：到目的地z的路由经过y转发，距离为c(x,y)+d(y,z)，下一跳为y
  - 以前有到目的地的路由，并且下一跳段也正好是该邻居，则更新路径花费，距离为c(x,y)+d(y,z)
  - 否则比较是否新的路径要更短，如果是，即c(x,y) + d(y,z) < d(x,z)则更新表项，距离为c(x,y)+d(y,z)，下一跳为y

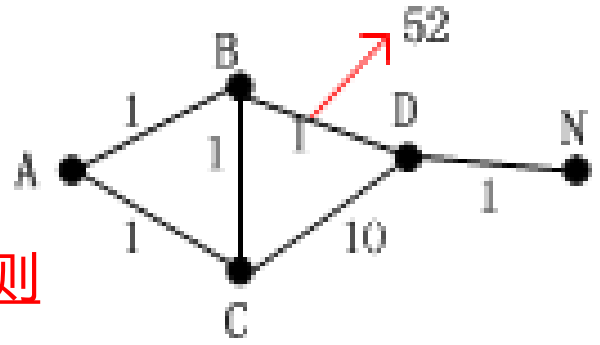
Y发布到Z/Z1/Z2的距离向量



目的地	花费	下一跳
z1	c(x, z1) = c(x,y) + d(y, z1)	y
z2	c(x, z2) = c(x,y1) + d(y1, z2)	y1

	目的地	花费	下一跳
更新距离	z1	c(x, z1) = c(x,y) + d(y, z1)	y
更新	z2	c(x, z2) = c(x,y) + d(y, z2)	y
新增	z	c(x, z) = c(x,y) + d(y, z)	y

距离向量路由：无穷计数(count to infinity)问题



假设：不采用触发更新（即有变化时不会立即发送）

- 计算和传输同步，即收到所有距离向量后计算
- 在计算时考虑稳定性，如果有多条最短路径，且上轮的最短路径仍然存在，则继续选择上次选择的路径；如果不存在，选择其中邻居ID最大的最短路径

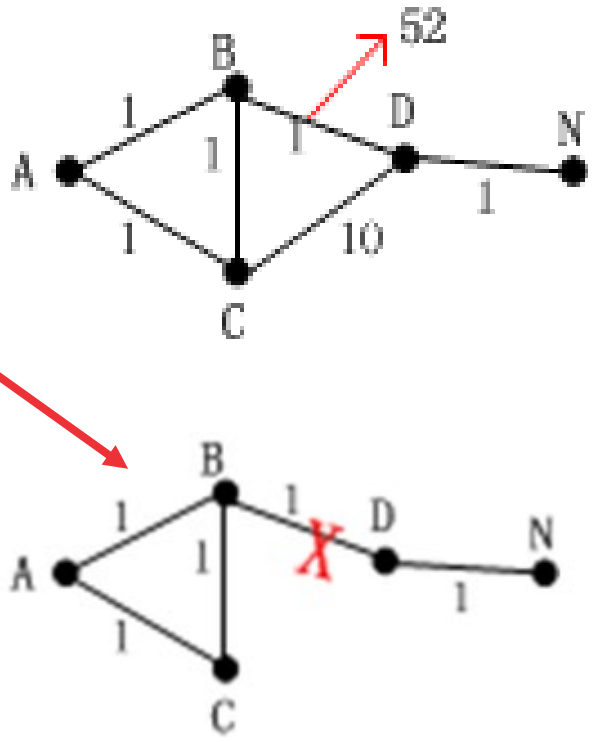
时刻	A		B			C		
	B(AB=1)	C(AC=1)	A(BA=1)	C(BC=1)	D(BD=1→52)	A(CA=1)	B(CB=1)	D(CD=10)
稳定	2+1=3	3+1=4	3+1=4	3+1=4	1+1=2	3+1=4	2+1=3	1+10=11
BD→52	2+1=3	3+1=4	3+1=4	3+1=4	1+52=53	3+1=4	2+1=3	1+10=11
第一轮	4+1=5	3+1=4	3+1=4	3+1=4	1+52=53	3+1=4	4+1=5	1+10=11
第二轮	4+1=5	4+1=5	4+1=5	4+1=5	1+52=53	4+1=5	4+1=5	1+10=11
第三轮	5+1=6	5+1=6	5+1=6	5+1=6	1+52=53	5+1=6	5+1=6	1+10=11
…								
第八轮	10+1=11	10+1=11	10+1=11	10+1=11	1+52=53	10+1=11	10+1=11	1+10=11
第九轮	11+1=12	11+1=12	11+1=12	11+1=12	1+52=53	11+1=12	11+1=12	1+10=11
第十轮	12+1=13	11+1=12	12+1=13	11+1=12	1+52=53	12+1=13	12+1=13	1+10=11
BD→1	12+1=13	11+1=12	12+1=13	11+1=12	1+1=2	12+1=13	12+1=13	1+10=11
新一轮	2+1=3	11+1=12	12+1=13	11+1=12	1+1=2	12+1=13	2+1=3	1+10=11
新二轮	2+1=3	3+1=4	3+1=4	3+1=4	1+1=2	3+1=4	2+1=3	1+10=11

A→C →A 出现了路由回路，距离缓慢增加，直到C发现通过D到N的距离更短

距离向量路由：无穷计数(count to infinity)问题

- 在同步的版本中，在新(坏)消息出现后，由于保存了距离表，实际上采用的是老的消息，也就是说第一步就出现了老消息和新消息混合的情况，在接下来的一轮中，从**邻居节点收到的也是老消息**，自己传播出去的是新老混合的消息
- 如果是保留路由表的版本，则新(坏)消息出现后，节点使用最新的消息，下一轮传播给邻居，但是**下一轮来自于邻居的可能是老消息**

- **无穷计数问题：坏消息的传播非常缓慢**
  - 假设BD链路断开时，距离缓慢增加，直到最后为无穷大，才知道不可达
  - 无穷大可设置为网络的直径+1
  - 坏消息意味着原来的最短路径不可用，可是新老信息混合在一起，之前的老信息仍然在传播，导致了路由回路的出现
- 好消息的传播却非常迅速
  - BD链路花费从52变为1时，传播非常迅速
  - 好消息意味着一条更好的路径，很快被相邻路由器知道，并且更新相应路由表，再通知给其它路由器



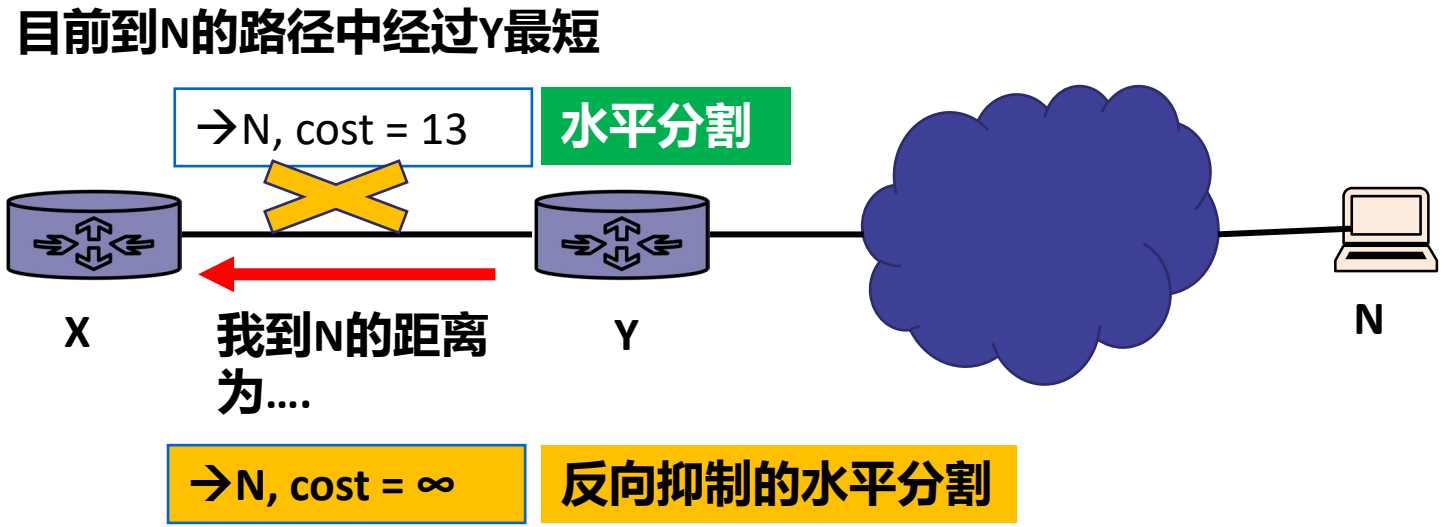
无穷计数问题：触发更新和抑制机制

- **触发更新(triggered update)：**不仅仅定期发送距离向量，在距离向量改变时立刻通知直接邻居，邻居再进一步传播开去
- **抑制（hold-down）机制**：
  - 路由器了解到某个网络不可达（距离为无穷大）时开启抑制计时器（抑制期）
  - 抑制期间前，收到路由时：
    - 如果有一条比**抑制计时器开启前的路径更好**的路由，关闭计时器
    - 否则忽略收到的路由信息，即便其通知有一条到目的网络路由
  - 抑制期一般设为60秒，使得网络的不可达状态能传播到所有节点
  - 所有的路由器必须使用相同的抑制期，否则有可能发生路由回路
  - 抑制机制避免了可能的路由回路，但是以增加收敛时间为代价

无穷计数问题：水平分割（Split Horizon）

- **水平分割：**通过一个特定的网络接口X发送距离向量时，绝对不要包括通过那个网络接口学习到的路由信息（即下一跳为接口X上的邻居路由器的距离向量）
- **带反向抑制的水平分割**（split horizon with poisoned reverse）
  - 给邻居路由器发送路由信息时将那些从该邻居了解到的路由信息的距离设为无穷大
  - 相比水平分割路由消息更大一些
  - 尽管有水平分割，仍然可能会出现两个节点之间的路由回路
  - 通过反向抑制可以更快地消除两个节点之间的路由回路，而不是等待超时而消除

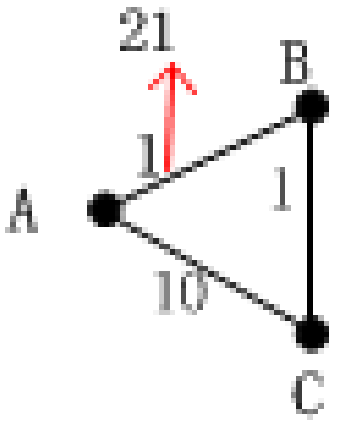
目的地	花费	下一跳
N	13	Y
…	…	…



带反向抑制的水平分割：示例

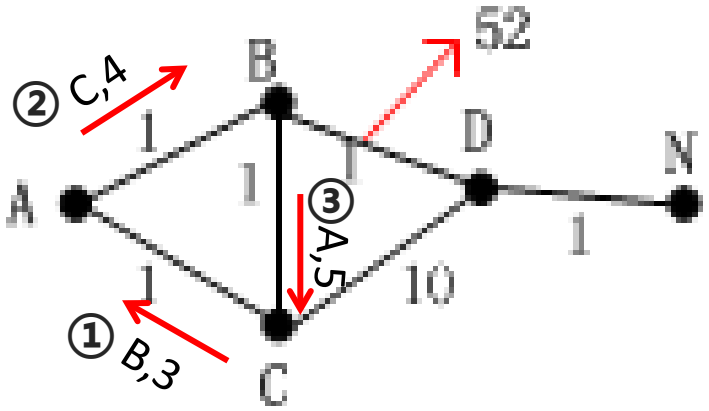
考察到A的路由

时刻	B		C	
	A(AB=1→21)	C(BC=1)	A(CA=10)	B(CB=1)
稳定	1+0=1	∞	10+0=10	1+1=2
AB→21	21+0=21	∞	10+0=10	1+1=2
第一轮	21+0=21	∞	10+0=10	21+1=22
第二轮	21+0=21	10+1=11	10+0=10	21+1=22
第三轮	21+0=21	10+1=11	10+0=10	∞



(c) 两个节点间的路由回路

- 水平分割用于解决两个节点间的路由回路，但可能出现3个甚至更多节点之间的路由回路
  - 可能出现的时序：对于路由器A，B发送的触发更新消息到来之前，C定期发送路由时刻已经到来



教材P217页第4行(B,5)改为(A,5)

(B,3)表示到N的距离为3，下一跳为B(下一跳信息并不会发布给邻居，而是用于说明发布的是哪条路径)

Loop:
B → A → C →B → … N!!!

距离向量路由协议：RIP（Routing Information Protocol）

- RFC 1058定义RIP-1，采用距离向量路由算法，仅记录最优路由在路由表中
  - UDP，端口520
  - RIP消息以本地广播（255.255.255.255）方式交换，距离向量： Network + Distance
  - 定期（30秒）交换和触发更新，6X30=180秒超时
  - 超时**或者计算为不可达**的路由表项等待120秒后移走(flush)
  - 距离为节点计数，直接连接的网络距离为1。允许设置更大的距离度量值。无穷大取值为16
  - 支持水平分割以及反向抑制的水平分割
  - 有些实现支持抑制（缺省60秒）机制
  - 支持主动和被动方式。被动方式仅仅接收路由表，用在主机中
- RFC 2453定义RIP-2.　　RFC 2080定义了RIPng for IPv6
  - 采用**IP组播（224.0.0.9，RIP-2路由器）**，而不是采用本地广播来交换路由消息
  - 距离向量添加了**网络掩码**，从而支持子网路由，距离向量： Network/Mask + Distance
  - 支持认证，第一个距离向量用于传递认证信息
  - 支持通过别的外部协议（如BGP-4）了解到的路由，引入了路由标签route tag

RIP-2消息

教材图5.31不是很准确

- RIP消息通过UDP携带，端口号520
- UDP消息最长512字节，最多25个距离向量。路由表大时通过多个RIP消息传递
- 认证是可选的，如果有，第1个距离向量的地址家族为0xFFFF时，表示该距离向量携带的是认证信息
- 命令：
  - 请求：请求链路上的路由器立即发送距离向量
  - 响应消息：定期30秒发送，或者收到请求时发送
- 路由标签、路由域和下一跳都是用在一些特殊的情形（有外部路由等）
- RIP-1的距离向量仅仅包括：network/distance，其他字段为0

命令	版本	route domain
地址家族=0xFFFF(认证)		auth protocol
auth data		
auth data		
auth data		
auth data		
地址家族=0x2(IP)		route tag
network		
mask		
next hop		
distance		
...		

8字节UDP头部 + 4 + 25 * 20 = 512字节

IP头部	UDP (端口520)	RIP-2消息 最多25 * 20个距离向量，认证部分可选)

RIP-2消息

No.	Time	Source	Destination	Protocol	Length	Info
36	481.203000	10.0.12.2	224.0.0.9	RIPv2	86	Response
37	490.125000	10.0.12.1	224.0.0.9	RIPv2	86	Response
38	510.375000	10.0.12.2	224.0.0.9	RIPv2	86	Response
39	525.343000	10.0.12.1	224.0.0.9	RIPv2	86	Response
40	540.671000	10.0.12.2	224.0.0.9	RIPv2	86	Response
41	559.609000	10.0.12.1	224.0.0.9	RIPv2	86	Response
42	572.921000	10.0.12.2	224.0.0.9	RIPv2	86	Response
43	589.875000	10.0.12.1	224.0.0.9	RIPv2	86	Response
44	607.203000	10.0.12.2	224.0.0.9	RIPv2	86	Response
45	625.203000	10.0.12.1	224.0.0.9	RIPv2	86	Response
46	639.359000	10.0.12.2	224.0.0.9	RIPv2	86	Response
47	657.484000	10.0.12.1	224.0.0.9	RIPv2	86	Response
48	674.718000	10.0.12.2	224.0.0.9	RIPv2	86	Response
49	684.671000	10.0.12.1	224.0.0.9	RIPv2	86	Response
50	704.984000	10.0.12.2	224.0.0.9	RIPv2	86	Response
51	709.921000	10.0.12.1	224.0.0.9	RIPv2	86	Response

```
> Frame 39: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface -, id 0
> Ethernet II, Src: HuaweiTe_db:36:bb (54:89:98:db:36:bb), Dst: IPv4mcast_09 (01:00:5e:00:00:09)
> Internet Protocol Version 4, Src: 10.0.12.1, Dst: 224.0.0.9
> User Datagram Protocol, Src Port: 520, Dst Port: 520
v Routing Information Protocol
    Command: Response (2)
    Version: RIPv2 (2)
  v IP Address: 10.0.1.0, Metric: 1
      Address Family: IP (2)
      Route Tag: 0
      IP Address: 10.0.1.0
      Netmask: 255.255.255.0
      Next Hop: 0.0.0.0
      Metric: 1
  > IP Address: 10.0.13.0, Metric: 1
```

```
0000  01 00 5e 00 00 09 54 89  98 db 36 bb 08 00 45 c0   ··^···T· ··6···E·
0010  00 48 00 27 00 00 0e 11  b5 b4 0a 00 0c 01 e0 00   ·H·'···· ········
0020  00 09 02 08 02 08 00 34  e3 61 02 02 00 00 00 02   ·······4 ·a······
0030  00 00 0a 00 01 00 ff ff  ff 00 00 00 00 00 00 00   ········ ········
0040  00 01 00 02 00 00 0a 00  0d 00 ff ff ff 00 00 00   ········ ········
0050  00 00 00 00 00 01                                 ......
```

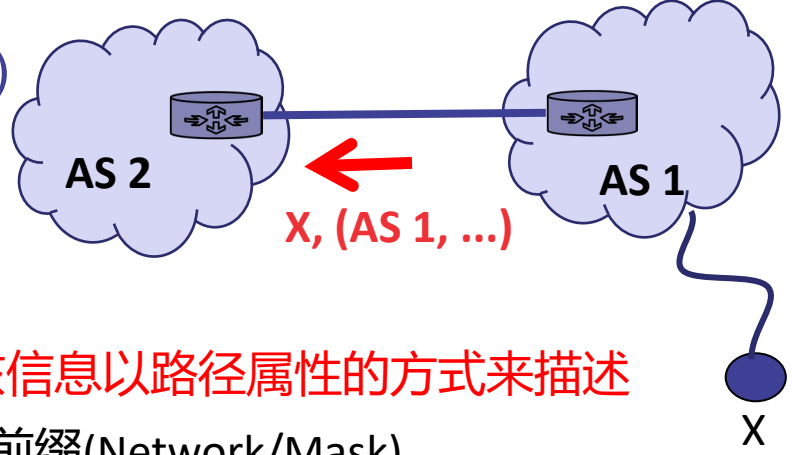
5.1.3 逐跳路由：扩散法和逆向学习法

5.4 路由协议

- 自治系统
- 链路状态路由
- 距离向量路由
- **BGP路由协议**

5.5.3 组播路由协议

边界网关协议BGP（border gateway protocol）



RFC 4271定义了BGP-4，自治系统之间的路由协议

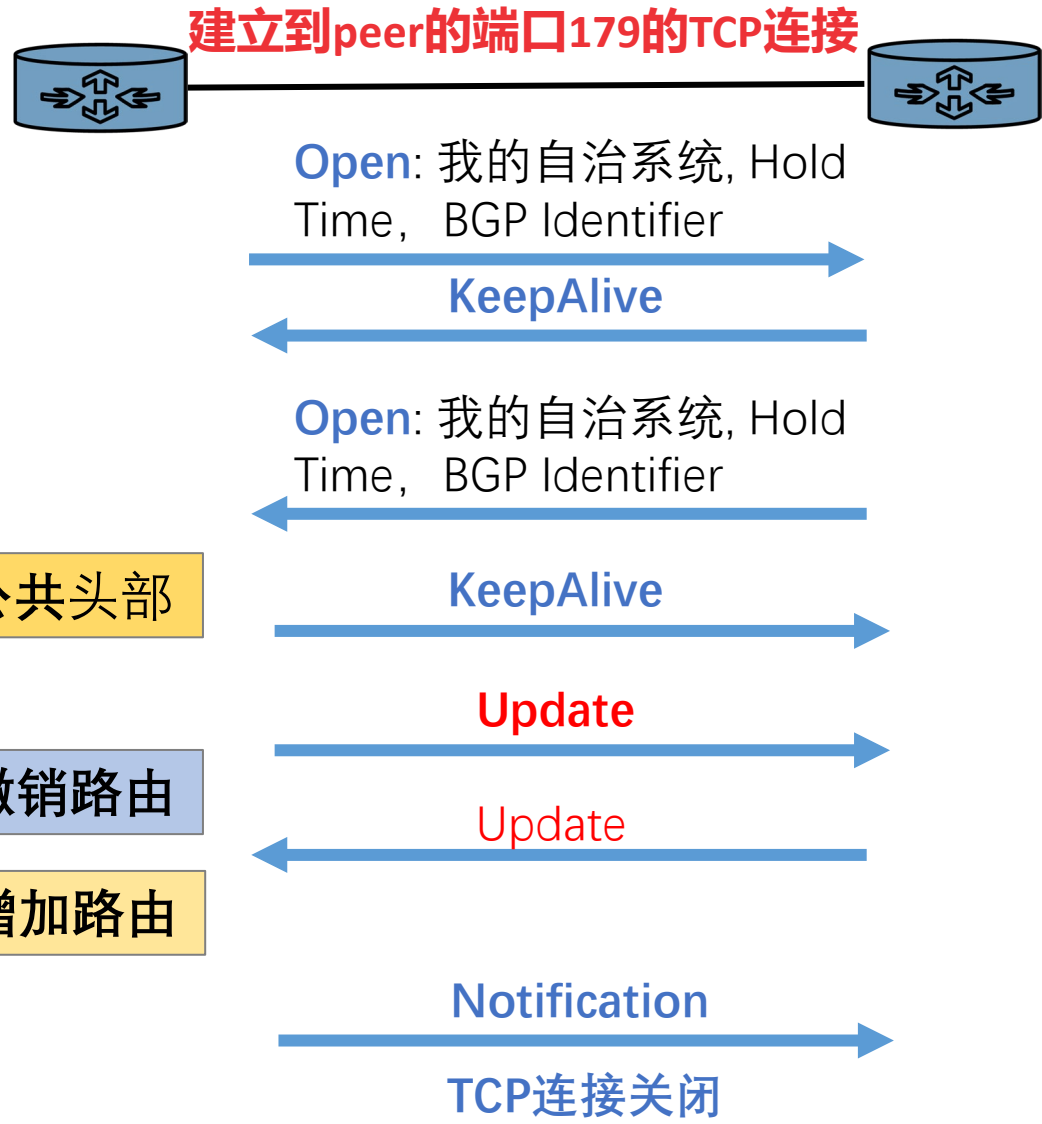
- 考虑到路由消息交换、路由表、收敛需要采用层次路由→自治系统AS
- 考虑到路径花费定义不同，需要采用策略路由，传递网络可达性的信息，该信息以路径属性的方式来描述
  - 网络可达性信息NLRI(Network Layer Reachability Information)：多个路由前缀(Network/Mask)
  - 多个路径属性
    - AS-PATH(路径向量）：经过的自治系统列表，可以检测自治系统回路
      - 路由发布到另一个自治系统的邻居时附加自身的AS编号
      - 如果从另外一个自治系统的路由器收到一条路由，发现当前自治系统编号已经出现在AS-PATH，说明有回路，丢弃该路由
  - RIP/OSPF不适合扩展来传递自治系统之间的可达性信息

	RIP	OSPF	BGP
邻居发现	定期交换路由表	定期交换Hello分组	管理人员配置
路由信息	邻居间交换距离向量	扩散链路状态分组	交换路径向量
传输方式	UDP，软状态的定期交换	IP，可靠扩散	TCP，增量更新

BGP消息

- 管理员决定哪两个路由器之间运行BGP协议，首先建立一条TCP连接，端口号为179

- Open消息：建立到端口179的TCP连接后发送，协商认证机制，路由器回应以KeepAlive
- KeepAlive:定期（没有Update时每Hold Time，缺省180秒）发送保持邻居活跃
- Notification:出错或特殊情况(比如达到允许接收的路由限制）下发送，连接关闭，相应的路由被移走，重新计算路由
- Update消息：增加、更新或撤销路由



标记（16字节，全1，兼容考虑）		
长度（2字节）	类型（1字节）	不可用路由长度
不可用路由长度	撤销路由（可变长度）	
路径属性长度	路径属性（可变长度）	
网络可达性信息（可变长）		

公共头部

撤销路由

增加路由

BGP消息

OPEN消息示例

```
> Internet Protocol Version 4, Src: 192.168.0.15, Dst: 192.168.0.33
> Transmission Control Protocol, Src Port: 2124, Dst Port: 179, Seq: 1, Ack: 1, Len: 29
v Border Gateway Protocol - OPEN Message
    Marker: ffffffffffffffffffffffffffffffff
    Length: 29
    Type: OPEN Message (1)
    Version: 4
    My AS: 65033
    Hold Time: 180
    BGP Identifier: 192.168.0.15
    Optional Parameters Length: 0
```

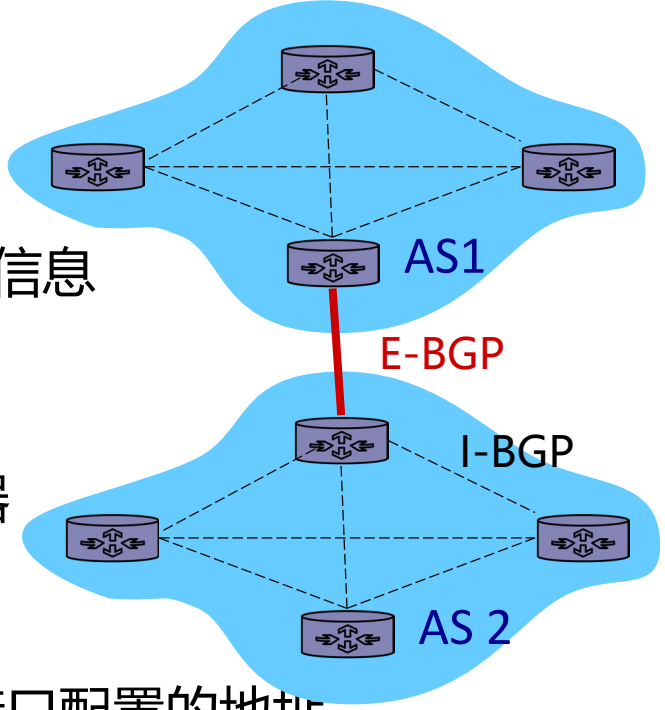
KEEPALIVE消息示例

```
> Internet Protocol Version 4, Src: 192.168.0.15, Dst: 192.168.0.33
> Transmission Control Protocol, Src Port: 2124, Dst Port: 179, Seq
v Border Gateway Protocol - KEEPALIVE Message
    Marker: ffffffffffffffffffffffffffffffff
    Length: 19
    Type: KEEPALIVE Message (4)
```

UPDATE消息示例

```
v Border Gateway Protocol - UPDATE Message
    Marker: ffffffffffffffffffffffffffffffff
    Length: 75
    Type: UPDATE Message (2)
    Withdrawn Routes Length: 0
    Total Path Attribute Length: 27
  v Path attributes
    > Path Attribute - ORIGIN: IGP
    > Path Attribute - AS_PATH: 200
    > Path Attribute - NEXT_HOP: 10.1.10.10
    > Path Attribute - MULTI_EXIT_DISC: 0
  v Network Layer Reachability Information (NLRI)
    > 10.0.0.10/32
    > 203.0.113.64/26
    > 203.0.113.128/26
    > 203.0.113.192/26
    > 203.0.113.0/26
```

E-BGP和I-BGP

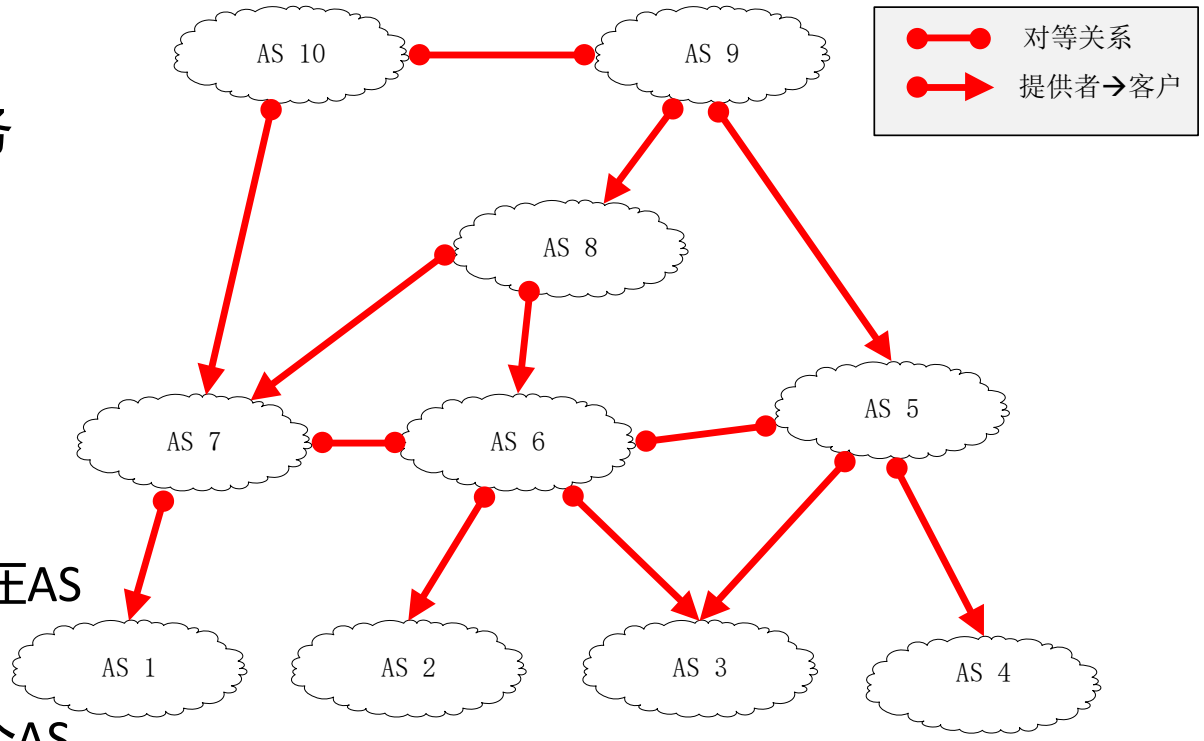


- 自治系统可能有多个BGP路由器
- E-BGP:不同自治系统的BGP路由器间的BGP协议，从相邻的AS获得网络可达信息
  - 路由发布到E-BGP邻居时添加自身的AS编号
- I-BGP:同一个自治系统的BGP路由器间的协议
  - 从E-BGP邻居了解到的网络可达信息要传播给AS内部的其他I-BGP路由器
  - RIP和OSPF无法完成：路径属性不支持，路由表项太多
  - I-BGP邻居可以不在同一个物理链路上
    - I-BGP一般不使用其某个接口的IP地址，而是采用虚拟的Loopback接口配置的地址
    - 只要连通，I-BGP会话不会断开
  - I-BGP和E-BGP采用相同消息格式和相同状态机，**区别在路由发布的限制**：
    - 通过E-BGP邻居了解到的路由再通过I-BGP发布给自治系统内部的其他I-BGP邻居
    - 从I-BGP邻居了解到的路由通过E-BGP发布给其他自治系统的BGP路由器
    - 从I-BGP邻居了解到的路由不允许发布给其他I-BGP邻居
      - 因为AS内部的I-BGP邻居之间无法通过路径向量来检测回路
      - 意味着自治系统的BGP路由器两两建立I-BGP会话，即全连接的方式

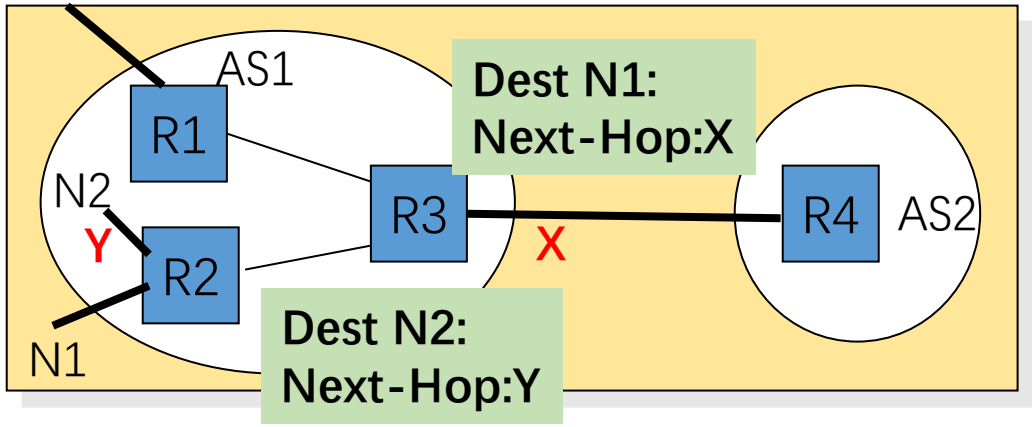
AS之间的关系和类型

AS之间:

- Customer-Provider: 客户支付费用使用提供者的服务
- Peer：对等关系提供AS之间的捷径
  - 一般共享，无需付费
  - 没有传递性
- 负载类型
  - 本地(Local)负载：IP分组的源或者目的地址在AS
  - 过境(Transit)负载：IP分组的源和目的地址都不在AS
- AS类型
  - 末端（Stub）AS：仅本地负载，且只连接到一个AS
  - 多穴（Multi-Homed）AS：仅本地负载，连接多个AS
  - 过境（Transit）AS：本地和过境负载
    - 多个AS之间可通过Internet交换中心IXP（Internet Exchange Point）连接
- 不是所有的AS都需要使用BGP
  - Stub AS: 边缘路由器通过IGP发布一个缺省路由即可
  - Multi-Homed AS: 如果有多个边缘路由器，也可通过IGP发布缺省路由



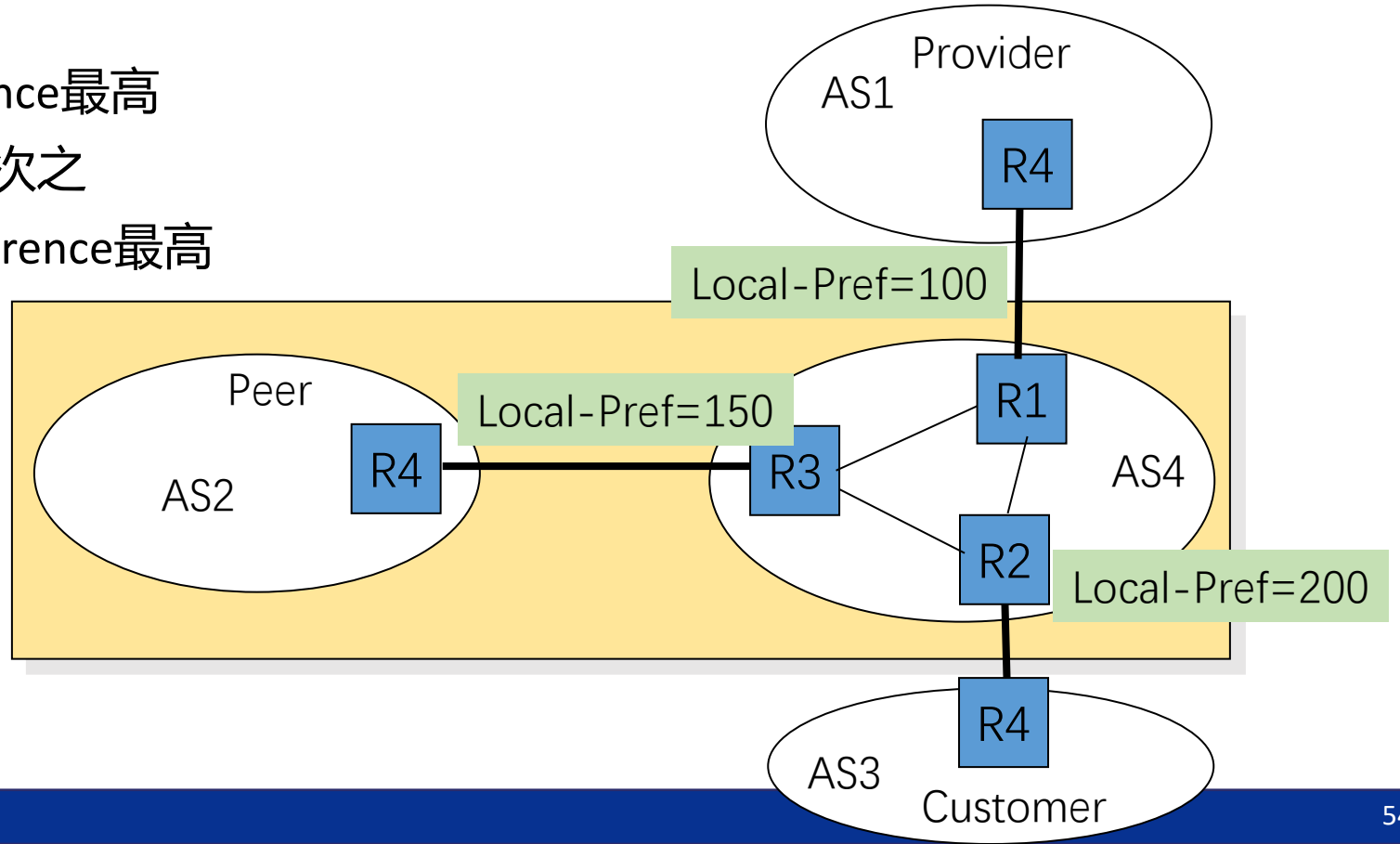
不介绍：BGP路径属性：必选属性



- 来源（Origin）：路由信息的来源，<u>在传播时不会被更改</u>
  - IGP：该路由最初来自于某个自治系统的IGP路由
  - EGP：早期的EGP路由协议(RFC 904)，已不再使用
  - INCOMPLETE（其他）：其他方式了解到的路由
- AS路径（AS-Path）：经过的自治系统编号，以最近的AS开始，以发起者(目的网络)的AS结束
  - 路由发布到E-BGP邻居时添加自身的AS编号，允许连续添加多个相同的AS编号
- 下一跳（Next-Hop）：到目的网络要经过该路由器转发，一般设置为发布该路由进入该AS的路由器的IP地址
  - AS-PATH仅仅传递了要经过哪个自治系统
  - 收到该路由的接收者知道到目的地途中要经过Next-Hop，但并不要求一定为直接连接的下一跳，采用**递归路由**的查找方法
    - 到目的网络的路由首先查找BGP路由表获得Next-Hop
    - 然后根据Next-Hop查IGP路由表决定直接连接的下一跳
  - E-BGP：R3发布路由到R4时设置为R3连接到R4的接口的IP地址
  - 通过E-BGP了解到的路由再发布到I-BGP Peer时Next-Hop字段一般不变
  - I-BGP：如果该路由从AS内部了解到时，Next-Hop为发布该路由的BGP路由器的地址，比如R2连接到N2的地址

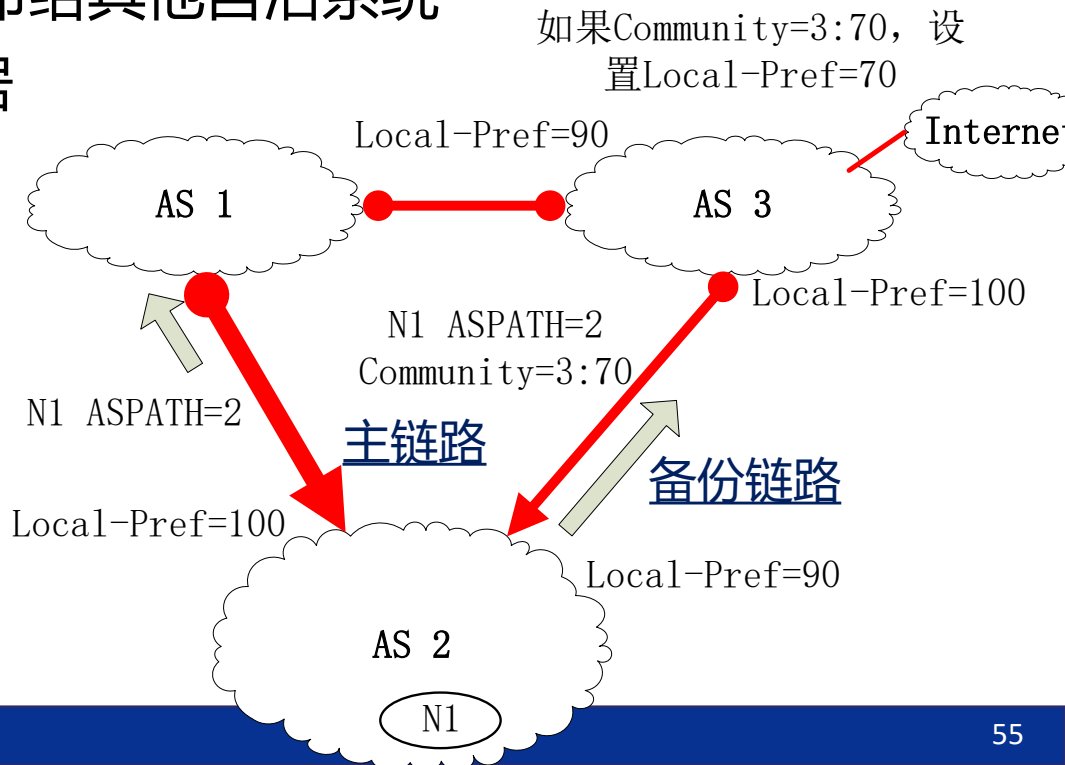
不介绍： BGP路径属性： 可选属性

- Atomic_Aggregate和Aggregator属性用于实现路由聚合
- Local_Pref: 可控制到自治系统之外的目的网络(Internet)的路由如何选择
  - 有多个出口路由器选择时选择Local_Pref属性最高的出口路由器
  - 仅用于自治系统内部，在I-BGP邻居之间交换
  - 一般的设置：
    - 到其Client AS的Local Preference最高
    - 到peer AS的Local Preference次之
    - 到其Provider AS的Local Preference最高



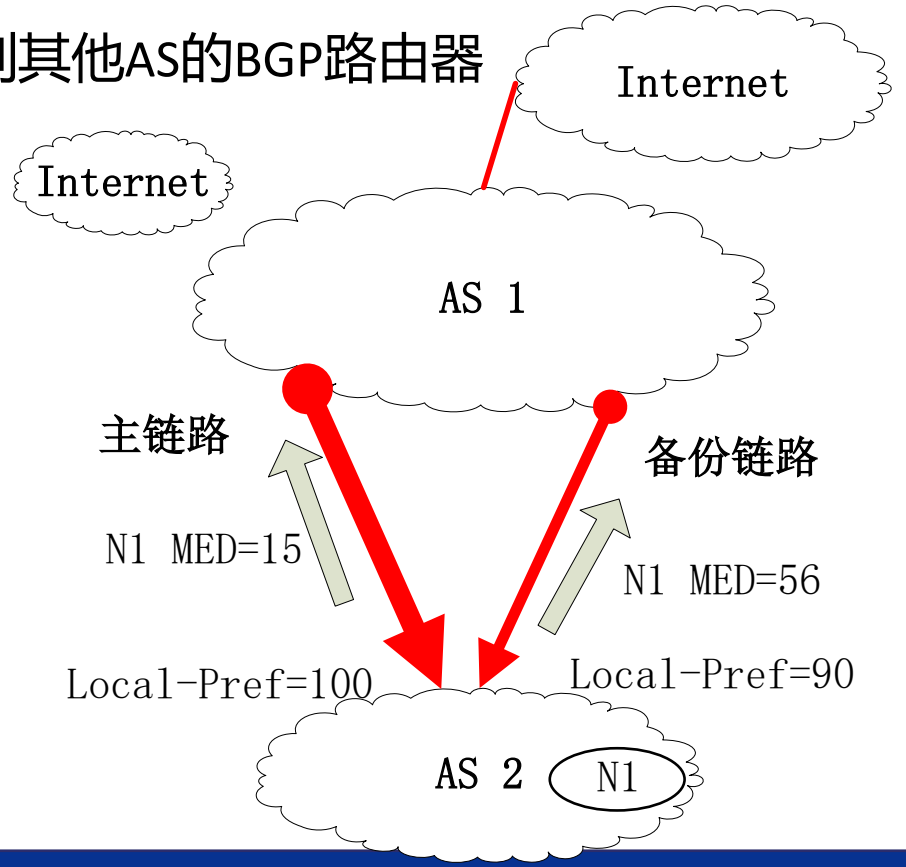
不介绍： BGP可选路径属性： Community

- **简化策略控制，给发布的路由打上一个标签(4个字节的整数)**
  - 前面两个字节为自治系统编号，后面两个字节为该community编号。经常以AS编号:community编号格式描述
- 自治系统的管理员可以针对同一个community定义规则（策略）
- 预先定义好的community：
  - NO-EXPORT（0xFFFFFF01）：不许(通过E-BGP)发布给其他自治系统
  - NO-ADVERTISE(0xFFFFFF02)：不许发布给其他邻居
- AS 2通过Local-pref配置，到Internet都通过AS1出去。希望反向的负载也是采用同样的路径
- 但是对AS 3而言，到AS 2的分组缺省情况下会采用AS 3到AS 2的链路转发
- AS 2发布到AS 3的路由通过community加上标签，这样AS 3可以基于标签来定义相应的规则



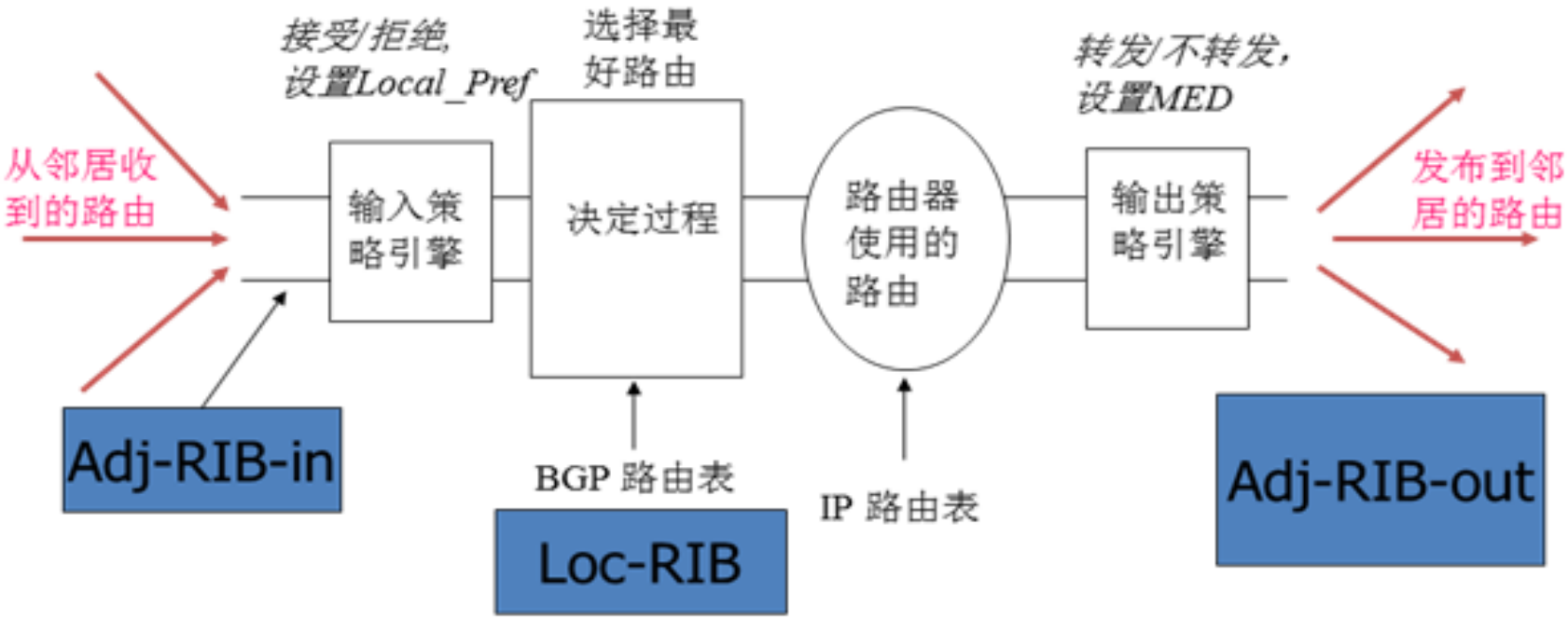
不介绍： BGP可选路径属性：MED （Multi_Exit_Disc）

- 两个自治系统之间有多条链路时来决定通过哪条链路进入该自治系统。比如AS 1→AS 2
- AS 2发布到AS 1的路由中包含MED选项，MED选项的取值一般反映了该BGP路由器到自治系统内部的目的网络的路径花费
- MED越低，越优先选取
- MED可以传播给自治系统的其他BGP路由器，但是不允许传播到其他AS的BGP路由器



BGP路由决策过程

- Loc-RIB保存了经过一个决定过程后从Adj-RIBs-In所选择的最好路由
- 输入策略控制那些外出（outbound)的负载，即怎么到达该目的网络
  - 设置Local Preference控制优先选择哪个出口路由器
  - **拒绝某条路由相当于不采用该路由到达对应的前缀网络**
- 输出策略控制用于控制那些到来(inbound)的负载，即是否允许其他节点通过该链路到达目的网络
  - 发布给E-BGP邻居时添加（可添加多个）自己的AS编号到AS-PATH以及设置MED等
  - **不发布某条路由给邻居路由器，意味着邻居不会转发相应的分组过来**



BGP路由决策过程

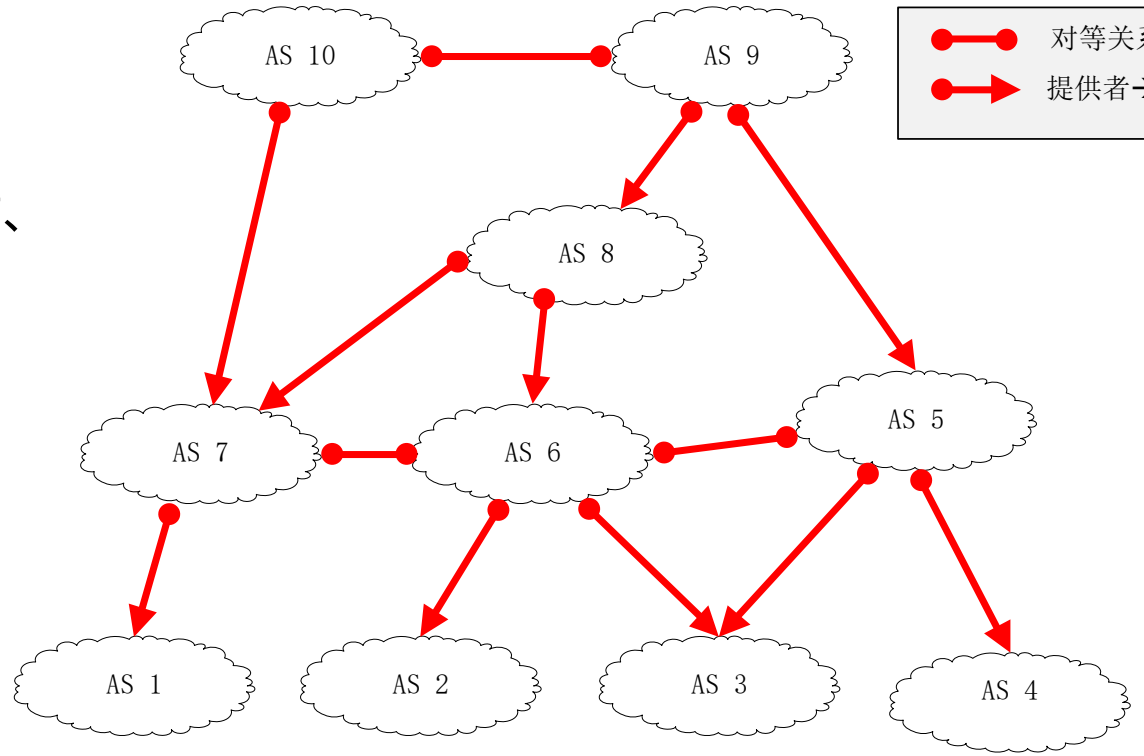
- 分组转发时采用最长前缀匹配的原则，那些尽管匹配但是前缀较短的路由不被采用

BGP路由器可能收到多个peer发布的到某个目的地路由，但是保存在路由表或者发布到其他peer时仅选择其中最好的路由：

- 如果有多条路由，则选择其中具有最高Local-Pref属性的路由；
- 如果仍然有多条路由，则选择AS-PATH最短的路由；
- 如果仍然有多条路由，则根据ORIGIN属性类型的大小顺序选择最小的路由，即通过IGP了解到的路由<通过EGP了解到的路由<通过其他方法了解到的路由；
- 如果仍然有多条路由，则选择具有最低MED属性的路由，当然这些路由应该是到同一个邻居自治系统中的；
- 如果仍然有多条分别通过I-BGP和E-BGP了解到的路由，则选择通过E-BGP了解到的路由；
- 如果仍然有多条路由，则根据到Next-Hop属性给出的路由器的IGP度量选择距离最近的路由； →热土豆路由
- 如果仍然有多条路由，则选择具有最低BGP路由器ID的路由

不介绍： BGP：AS系统常用的策略

- 提供者：为客户提供Internet接入服务
  - 将了解到的所有路由（来自于对等AS、提供者、其他客户以及本AS内部）发布给客户
  - 接受来自于客户本身或者客户后代自治系统的路由。
- 客户：
  - 接受来自于提供者的所有路由
  - 发布客户本身或者客户后代自治系统的路由
- 对等AS：按照对等关系的预先约定，为两个AS的的客户之间提供捷径
  - 将本AS以及后代AS的路由发布到对等AS
  - 仅接受从对等AS发布过来的属于该AS以及其后代AS的路由



如果一条路径满足**无谷底**（Valley-free）的特性，则一定不会出现路由回路
即一旦一个AS路径经过一个提供者→客户或者对等到对等的边，则在此之后不再允许出现客户→提供者或者其他对等到对等的边

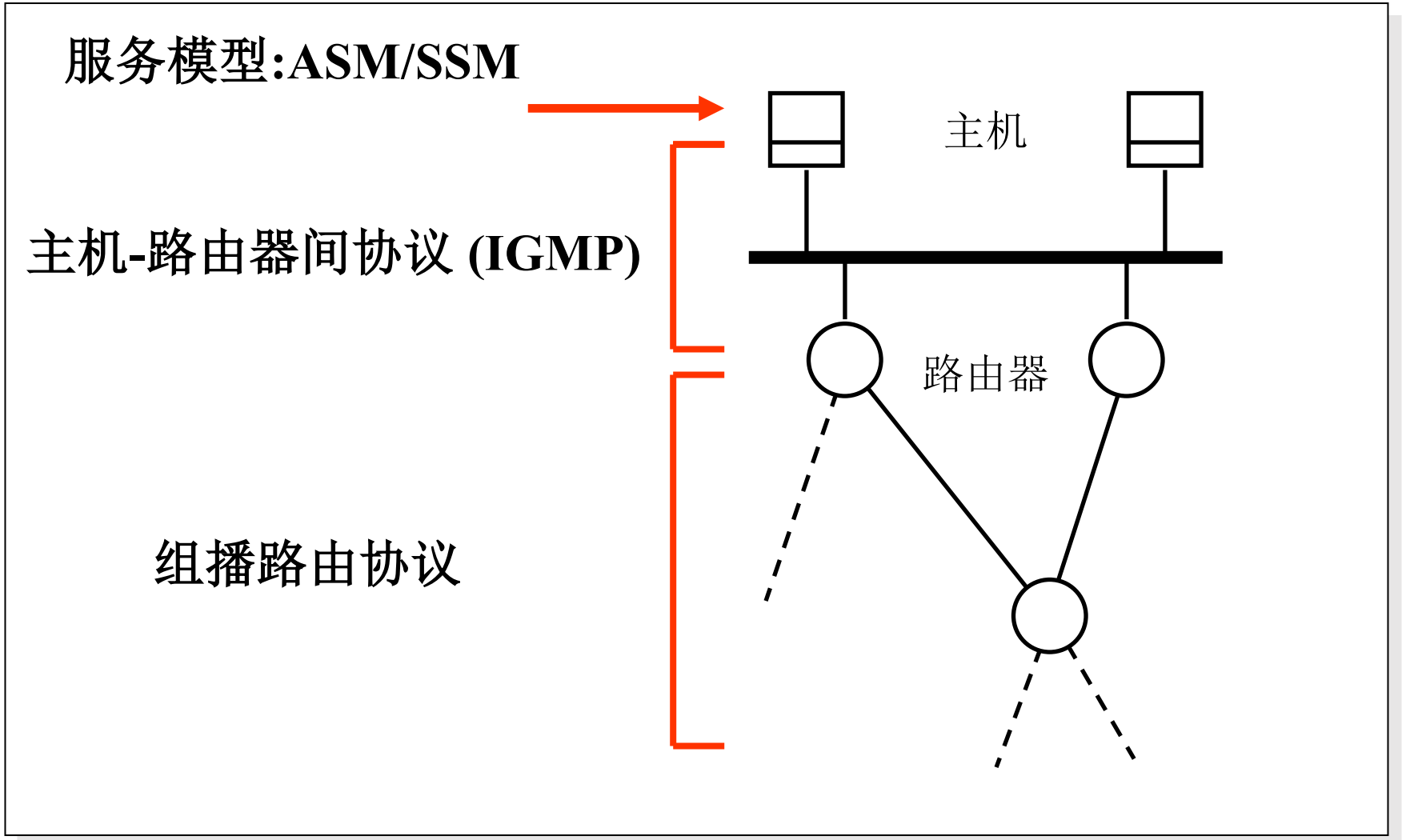
5.1.3 逐跳路由：扩散法和逆向学习法

5.4 路由协议

- 自治系统
- 链路状态路由
- 距离向量路由
- BGP路由协议

5.5.3 组播路由协议

IP组播体系结构

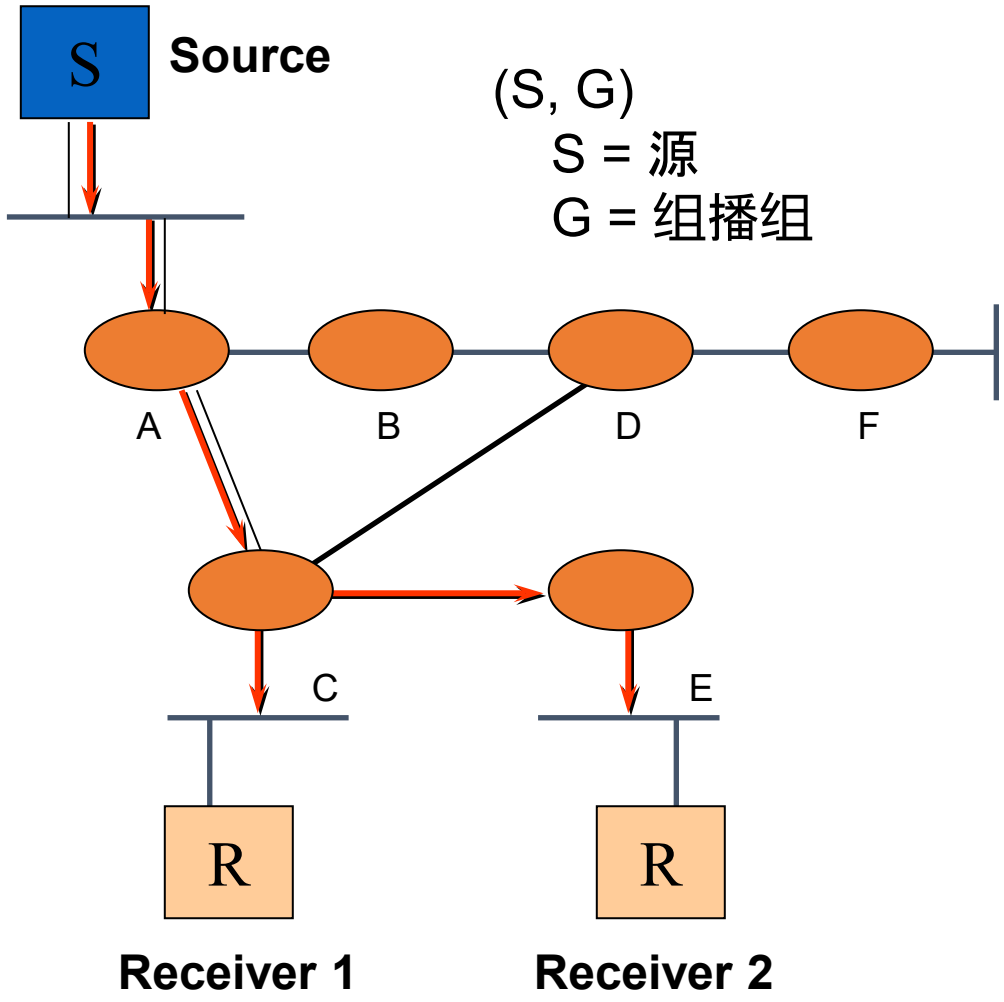


组播路由概述：源分发树

- 组播路由器之间交换信息构造一颗组播分发树，组播分组沿着该树转发直到到达所有的接收者
- 组播分发树必须动态构建：成员会动态加入和退出

组播分发树的根如何确定?

- 源分发树：(S, G)，主要协议包括DVMRP, MOSPF, PIM-DM等
  - 组播源（发送者）为根，组播路由器为每棵树(S.G)维护状态信息
  - DVMRP采用反向最短路径树：
    - 组播分组来自于到S的最短路径（根据单播路由协议了解到）的上一跳时才会进一步转发
    - 可采用先扩散，然后没有成员的方向将该该分支修剪掉的方法构建
  - 适合于密集模式: 发送者较少，成员密集分布



组播路由概述：共享树

共享树：所有成员共享一棵树

- 基于核心或RP(Rendezvous Point)的树，如CBT, PIM-SM
  - 多个源发送的组播分组首先发送给Core，然后沿着共享树分发给所有成员
  - 路由器只需要为共享树(*,G)维护状态
  - 成员加入到组播组时往Core发送join请求，从而构造共享树
  - 组播分组不是沿着最优路径到达，在源到core以及core到成员的路径有重复时会在该链路发送两次
- 适合于稀疏模式：成员零散分布