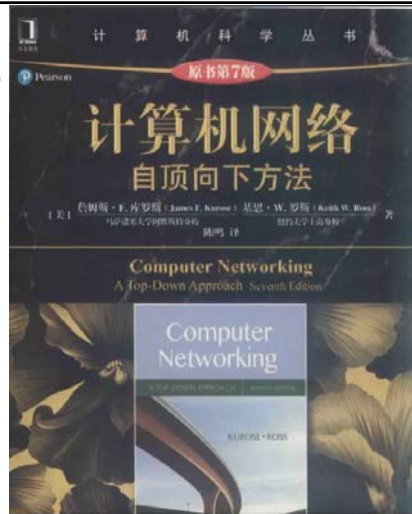


## Chapter 5 网络层：控制平面 Network Layer: Control Plane



## 第五章：网络层控制平面

### 章节目标:

- 了解网络控制平面背后的原理：
  - 传统路由选择算法
  - SDN控制平面
  - 网络管理，配置
- 实例化，在Internet中实现：
  - OSPF, BGP
  - OpenFlow, ODL和ONOS 控制器
  - 因特网控制报文协议: ICMP
  - SNMP, YANG/NETCONF

Network Layer: Data Plane 4-2

## 网络层：“控制平面”的路线图

- 概述
- 路由选择算法
  - 链路状态Link State
  - 距离向量Distance Vector
- ISP内部路由选择协议: OSPF
- ISP间的路由选择协议: BGP
- SDN 控制平面
- 因特网控制报文协议ICMP



- 网络管理，配置
  - SNMP
  - NETCONF/YANG

Network Layer: 5-3

## 网络层功能

- 转发forwarding: 将分组从一个输入端口转移到适当的输出端口
- 路由选择routing: 确定分组从源到目的地所采用的路由或路径

数据平面

控制平面

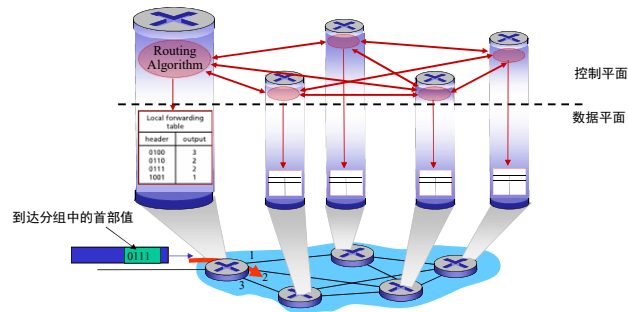
### 构造网络控制平面的两种方法:

- 每路由器控制 (传统)
- 逻辑集中式控制 (软件定义网络)

Network Layer: 5-4

## 每路由器控制平面

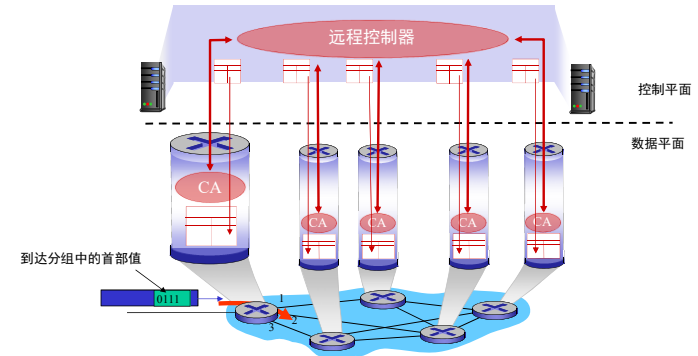
每台路由器中的路由选择算法组件在控制平面中的相互通信



Network Layer: 5-5

## 软件定义网络(SDN) 的控制平面

远程控制器计算并分发转发表以供每台路由器使用



Network Layer: 5-6

## 网络层：“控制平面”的路线图

- 概述
- 路由选择算法
  - 链路状态Link State
  - 距离向量Distance Vector
- ISP内部路由选择协议: OSPF
- ISP间的路由选择协议: BGP
- SDN 控制平面
- 因特网控制报文协议ICMP



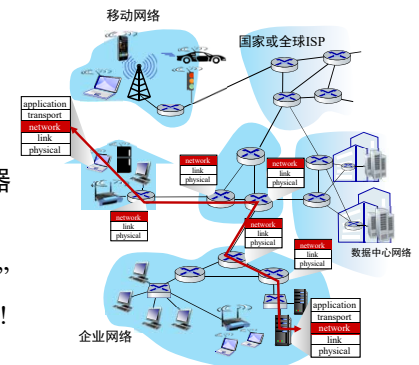
- 网络管理，配置
  - SNMP
  - NETCONF/YANG

Network Layer: 5-7

## 路由选择算法

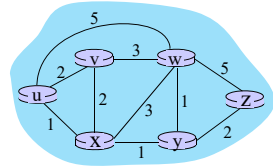
**目标:**从发送方到接收方的过程中确定一条通过路由器网络的好的路径（等价于路由）

- **路径:** 分组从给定的初始源主机传输到最终目标主机所经过的路由器序列
- **“好的”路径:** 有最低开销的路径  
如“成本”最少,“最快”,“最少拥堵”
- **路由选择算法:** **十大网络挑战之一!**



Network Layer: 5-8

## 一个计算机网络的抽象图：链路开销



$c_{a,b}$ : 节点 $a$ 和 $b$ 间边的链路开销  
e.g.,  $c_{w,z} = 5$ ,  $c_{u,z} = \infty$

网络运营商定义的开销：所有直连链路为1，或者与带宽成反比

图:  $G = (N, E)$

$N$ : 路由器集合 =  $\{u, v, w, x, y, z\}$

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

Network Layer: 5-9

## 路由选择算法分类



Network Layer: 5-10

## 网络层：“控制平面”的路线图

- 概述
- 路由选择算法
  - 链路状态Link State
  - 距离向量Distance Vector
- ISP内部路由选择协议: OSPF
- ISP间的路由选择协议: BGP
- SDN 控制平面
- 因特网控制报文协议ICMP



- 网络管理, 配置
  - SNMP
  - NETCONF/YANG

Network Layer: 5-11

## 链路状态路由选择算法: Dijkstra (迪杰斯特拉) 算法

- **集中式**: 所有节点都已知网络拓扑和链路开销
  - 通过“链路状态广播”实现
  - 所有节点有相同的信息
- 计算从源节点 (该节点) 到网络中所有其他节点的最低开销路径
  - 给出该节点的转发表
- **迭代**: 经过 $k$ 次迭代, 知道到达 $k$ 个目的节点的最低开销路径

### 符号

- $c_{x,y}$ : 从节点 $x$ 到 $y$ 的直接链路开销; 如果不是相邻节点 =  $\infty$
- $D(v)$ : 到算法的本次迭代, 从源节点到目的节点 $v$ 的最低开销路径
- $p(v)$ : 从源到 $v$ 沿着当前最低开销路径的前一节点 ( $v$ 的邻居)
- $N'$ : 节点子集, 明确知道从源到 $v$ 的最低开销路径的节点集

Network Layer: 5-12

## 链路状态路由选择算法：Dijkstra（迪杰斯特拉）算法

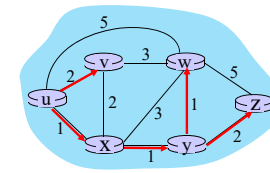
```

1 初始化Initialization:
2   $N' = \{u\}$  /* 计算u到所有其他节点的最低开销路径path */
3  for all nodes v
4    if v adjacent to u /* u初始仅知道相邻节点的直接路径 */
5      then  $D(v) = c_{u,v}$  /* 但可能不是最低开销 */
6    else  $D(v) = \infty$ 
7
8 循环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的最低开销路径是原本的值或已知的到w的最低开销路径加上w到v的直连
14 路径 */
15 直到所有节点都在数据集 $N'$ 中
  
```

Network Layer: 5-13

## 链路状态路由选择算法： Dijkstra（迪杰斯特拉）算法的例子

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	$\infty$	$\infty$
1	ux	2,u	4,x	$\infty$	2,x	$\infty$
2	uxy	2,u	3,y	$\infty$	4,y	$\infty$
3	uxyv	$\infty$	3,y	$\infty$	4,y	$\infty$
4	uxyvw	$\infty$	$\infty$	$\infty$	4,y	$\infty$
5	uxyvwz	$\infty$	$\infty$	$\infty$	$\infty$	4,y



初始化(step 0): 对所有的a: 如果与a相邻, 则  $D(a) = c_{u,a}$

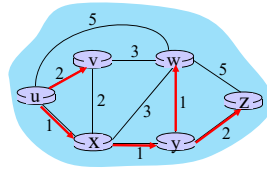
找出 $D(a)$ 为最小值但不在 $N'$ 里的a  
将a加入 $N'$

更新与a相邻但不在 $N'$ 里的所有b的 $D(b)$ 值:

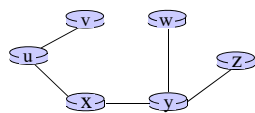
$$D(b) = \min(D(b), D(a) + c_{a,b})$$

Network Layer: 5-14

## 链路状态路由选择算法： Dijkstra（迪杰斯特拉）算法的例子



从u出发得到的最低开销路径树:



在u中生成的转发表:

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

直接从u到v的路由  
从u到所有其他目的地的路由都经过x

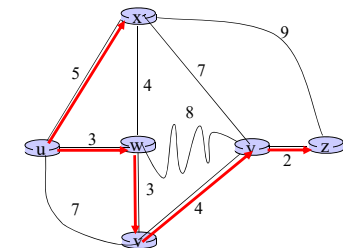
Network Layer: 5-15

## 链路状态路由选择算法： Dijkstra（迪杰斯特拉）算法的另一个例子

Step	$N'$	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	u	7,u	3,u	5,u	$\infty$	$\infty$
1	uw	6,w	$\infty$	5,u	11,w	$\infty$
2	uwx	6,w	$\infty$	$\infty$	11,w	14,x
3	uwxy	$\infty$	$\infty$	10,v	14,x	$\infty$
4	uwxyv	$\infty$	$\infty$	$\infty$	12,y	$\infty$
5	uwxyvz	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$

注意:

- 通过跟踪前一个节点构造最低开销路径树
- 可以存在纽带 (也可以任意打破)



问题: 请计算上图中节点u到其他节点的最低开销, 并给出最低开销树

Network Layer: 5-16

## 链路状态路由选择算法：讨论

**算法复杂度：**  $n$  个节点（不算源节点）

- $n(n+1)/2$  次搜索,  $O(n^2)$  复杂度
  - 每一次迭代均需要重新计算不在  $N'$  中的所有节点  $w$ : 第一次迭代, 需要检查  $n$  个节点以确定最低开销; 第二次迭代, 检查  $n-1$  个节点, 依次类推
- 更高效的实现（使用堆的数据结构）的复杂度是:  $O(n \log n)$

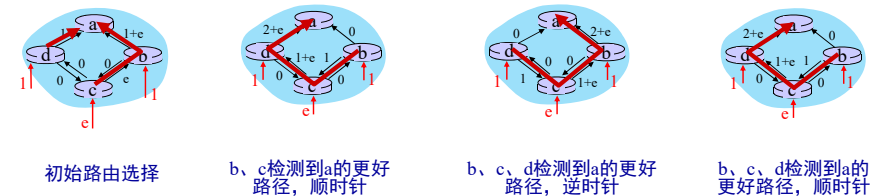
**报文复杂度：**

- 每台路由器必须向其他  $n$  台路由器 **广播** 其链路状态信息
- 高效 (且有趣!) 的广播算法:  $O(n)$  链路交叉来传播来自一个源的广播消息
- 每台路由器的消息都通过  $O(n)$  链路: 总体报文复杂度:  $O(n^2)$

Network Layer: 5-17

## 链路状态路由选择算法：路由选择的振荡

- 当链路开销等于链路上承载的流量, 例如, 反映经历的时延 (拥塞)
- 示例场景: 拥塞敏感的路由选择的振荡
  - 路由到目的地  $a$ , 流量以  $1, e (< 1), 1$  的速度分别进入  $d, c, b$
  - 链接开销是有方向性的, 对应承载的流量



**问：如何避免这种路由选择振荡？**

答: 1) 强制链路开销不依赖所承载的流量-违背了路由选择的避免拥塞链路的目标  
2) 确保并非所有路由器都同时运行LS - 发布链路通告的时间随机化

Network Layer: 5-18

## 网络层：“控制平面”的路线图

- 概述
- 路由选择算法
  - 链路状态Link State
  - 距离向量Distance Vector
- ISP内部路由选择协议: OSPF
- ISP间的路由选择协议: BGP
- SDN 控制平面
- 因特网控制报文协议ICMP



- 网络管理, 配置
  - SNMP
  - NETCONF/YANG

Network Layer: 5-19

## 距离向量路由选择算法

基于 **Bellman-Ford (BF)** 方程 (动态规划):

**Bellman-Ford 方程**

设  $D_x(y)$  为从节点  $x$  到  $y$  的距离向量, 即从节点  $x$  到  $y$  的最低开销路径的开销的估计值, 则节点  $x$  使用如下BF方程更新自己的距离向量:

$$D_x(y) = \min_v \{ c_{x,v} + D_v(y) \}$$

取  $x$  的所有邻居节点  $v$  的  $\min$

$v$  到  $y$  的最低开销路径的开销

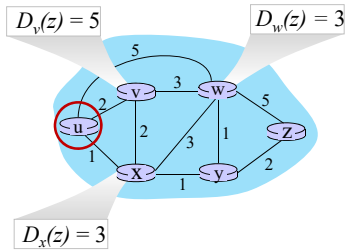
从  $x$  到  $v$  的链路开销,  $v$  为  $x$  的邻居

- 如果节点  $x$  的距离向量因这个更新步骤而改变, 节点  $x$  接下来将向它的每个邻居发送更新后的距离向量, 这继而让所有邻居更新它们自己的距离向量
- 所有节点继续以这种异步方式交换它们的距离向量, 最后开销估计  $D_x(y)$  收敛到从  $x$  到  $y$  的实际最低开销路径的开销。

Network Layer: 5-20

## Bellman-Ford 方程的例子

假设 $u$ 的邻居节点 $x, v, w$ 均知道到达目的节点 $z$ 的最低开销路径的开销:



Bellman-Ford 方程:

$$\begin{aligned} D_u(z) &= \min \{ c_{u,v} + D_v(z), \\ &\quad c_{u,x} + D_x(z), \\ &\quad c_{u,w} + D_w(z) \} \\ &= \min \{ 2 + 5, \\ &\quad 1 + 3, \\ &\quad 5 + 3 \} = 4 \end{aligned}$$

到达目的节点(z)的最低开销  
路径的下一跳是节点(x)

Network Layer: 5-21

## 距离向量路由选择算法

要点:

- 每个节点不时地将自己的距离向量估计值发送给它的邻居
- 当  $x$  从任何一个邻居接收到新的 DV 估计值时，它使用 BF 方程更新自身到任意节点  $y \in N$  的 DV 估计值：

$$D_x(y) \leftarrow \min_v \{c_{x,v} + D_v(y)\}$$

- 只要所有的节点继续以异步方式交换它们的距离向量，每个开销估计值  $D_x(y)$  收敛到实际最低开销路径的开销【Bersekas 1991】

Network Layer: 5-22

## 距离向量路由选择算法

每个节点:

等待自己的直连链路开销变化或来自邻居的DV变化通知

使用从邻居收到的DV值或者变化的直连链路开销值重新计算自己的DV估计值

如果到某一目的地的 DV 发生变化, 则通知邻居

**迭代的、异步的:** 引发每次本地迭代:

- 本地直连链路开销发生变化
- 邻居的DV更新报文

**分散式的、自停的:** 每个节点仅在其DV值变化时才通知它的邻居

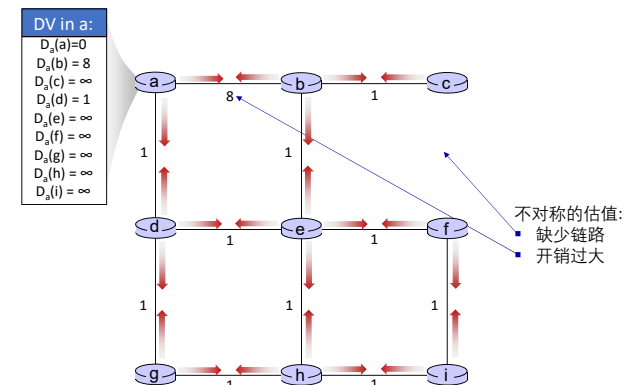
- 然后邻居再通知其邻居- 若必要
- 若未收到通知，不采取任何行动！

Network Layer: 5-23

## 距离向量路由选择算法-例子

 $t=0$ 

- 所有节点只有其邻接邻居的距离估计值
- 所有节点均发送其本地距离向量给它的邻居



Network Layer: 5-24

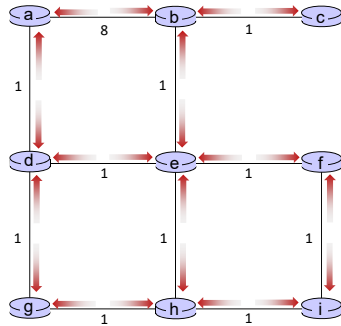
## 距离向量路由选择算法例子: 迭代



t=1

所有节点:

- 接收其邻居发送的距离向量
- 完成本地距离向量更新
- 将新的本地距离向量发送给它的邻居



Network Layer: 5-25

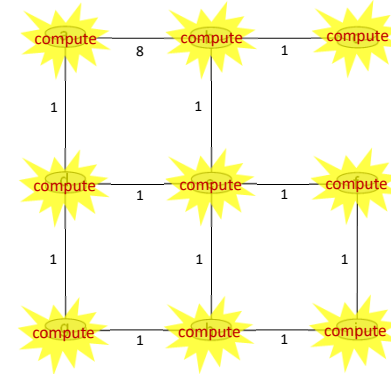
## 距离向量路由选择算法例子: 迭代



t=1

所有节点:

- 接收其邻居发送的距离向量
- 完成本地距离向量更新
- 将新的本地距离向量发送给它的邻居



Network Layer: 5-26

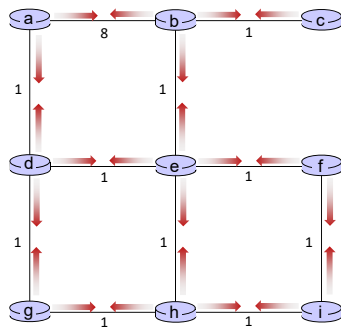
## 距离向量路由选择算法例子: 迭代



t=1

所有节点:

- 接收其邻居发送的距离向量
- 完成本地距离向量更新
- 将新的本地距离向量发送给它的邻居



Network Layer: 5-27

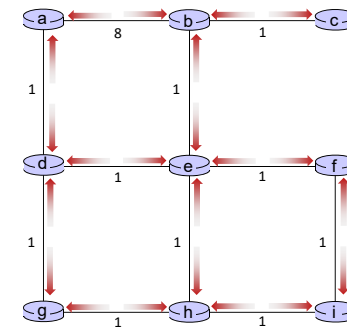
## 距离向量路由选择算法例子: 迭代



t=2

所有节点:

- 接收其邻居发送的距离向量
- 完成本地距离向量更新
- 将新的本地距离向量发送给它的邻居



Network Layer: 5-28



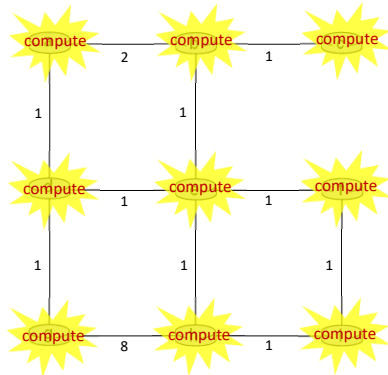
## 距离向量路由选择算法例子: 迭代



t=2

所有节点:

- 接收其邻居发送的距离向量
- 完成本地距离向量更新
- 将新的本地距离向量发送给它的邻居



Network Layer: 5-29

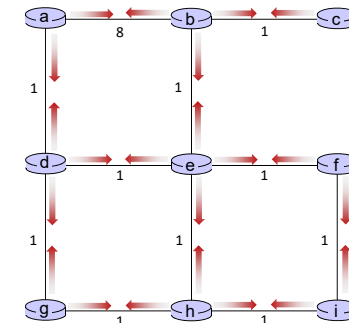
## 距离向量路由选择算法例子: 迭代



t=2

所有节点:

- 接收邻居发送的距离向量
- 完成本地距离向量更新
- 将新的本地距离向量发送给它的邻居



Network Layer: 5-30

## 距离向量路由选择算法例子: 迭代

... 以此类推  
让我们看一下每个节点上的迭代计算



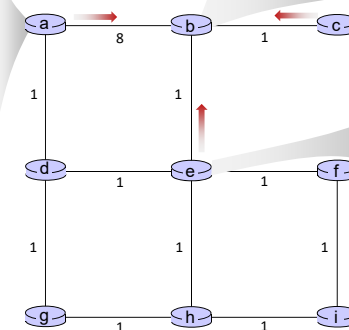
t=1

- b 接收来自 a, c, e 的 DV

DV in a:
$D_a(a)=0$
$D_a(b)=8$
$D_a(c)=\infty$
$D_a(d)=1$
$D_a(e)=\infty$
$D_a(f)=\infty$
$D_a(g)=\infty$
$D_a(h)=\infty$
$D_a(i)=\infty$

DV in b:
$D_b(a)=8$
$D_b(c)=1$
$D_b(d)=\infty$
$D_b(e)=1$
$D_b(f)=\infty$
$D_b(g)=\infty$
$D_b(h)=\infty$
$D_b(i)=\infty$

DV in c:
$D_c(a)=\infty$
$D_c(b)=1$
$D_c(c)=0$
$D_c(d)=\infty$
$D_c(e)=\infty$
$D_c(f)=\infty$
$D_c(g)=\infty$
$D_c(h)=\infty$
$D_c(i)=\infty$



DV in e:
$D_e(a)=\infty$
$D_e(b)=1$
$D_e(c)=\infty$
$D_e(d)=1$
$D_e(e)=0$
$D_e(f)=1$
$D_e(g)=\infty$
$D_e(h)=1$
$D_e(i)=\infty$

Network Layer: 5-31

Network Layer: 5-32



## 距离向量路由选择算法例子



t=1

- b 接收来自 a, c, e 的 DV, 计算:

$$\begin{aligned}
 D_b(a) &= \min\{c_{b,a} + D_a(a), c_{b,c} + D_c(a), c_{b,e} + D_e(a)\} = \min\{8, \infty, \infty\} = 8 \\
 D_b(c) &= \min\{c_{b,a} + D_a(c), c_{b,c} + D_c(c), c_{b,e} + D_e(c)\} = \min\{\infty, 1, \infty\} = 1 \\
 D_b(d) &= \min\{c_{b,a} + D_a(d), c_{b,c} + D_c(d), c_{b,e} + D_e(d)\} = \min\{9, 2, \infty\} = 2 \\
 D_b(e) &= \min\{c_{b,a} + D_a(e), c_{b,c} + D_c(e), c_{b,e} + D_e(e)\} = \min\{\infty, \infty, 1\} = 1 \\
 D_b(f) &= \min\{c_{b,a} + D_a(f), c_{b,c} + D_c(f), c_{b,e} + D_e(f)\} = \min\{\infty, \infty, 2\} = 2 \\
 D_b(g) &= \min\{c_{b,a} + D_a(g), c_{b,c} + D_c(g), c_{b,e} + D_e(g)\} = \min\{\infty, \infty, \infty\} = \infty \\
 D_b(h) &= \min\{c_{b,a} + D_a(h), c_{b,c} + D_c(h), c_{b,e} + D_e(h)\} = \min\{\infty, \infty, 2\} = 2 \\
 D_b(i) &= \min\{c_{b,a} + D_a(i), c_{b,c} + D_c(i), c_{b,e} + D_e(i)\} = \min\{\infty, \infty, \infty\} = \infty
 \end{aligned}$$

DV in b:

$D_b(a) = 8$	$D_b(f) = \infty$
$D_b(c) = 1$	$D_b(g) = \infty$
$D_b(d) = \infty$	$D_b(h) = \infty$
$D_b(e) = 1$	$D_b(i) = \infty$

DV in c:

$D_c(a) = \infty$
$D_c(b) = 1$
$D_c(c) = 0$
$D_c(d) = \infty$
$D_c(e) = \infty$
$D_c(f) = \infty$
$D_c(g) = \infty$
$D_c(h) = \infty$
$D_c(i) = \infty$

DV in b:

$D_b(a) = 8$	$D_b(f) = 2$
$D_b(c) = 1$	$D_b(g) = \infty$
$D_b(d) = 2$	$D_b(h) = 2$
$D_b(e) = 1$	$D_b(i) = \infty$

Network Layer: 5-33

## 距离向量路由选择算法例子



t=1

- c 接收来自 b 的 DV

DV in a:

$D_a(a) = 0$
$D_a(b) = 8$
$D_a(c) = \infty$
$D_a(d) = 1$
$D_a(e) = \infty$
$D_a(f) = \infty$
$D_a(g) = \infty$
$D_a(h) = \infty$
$D_a(i) = \infty$

DV in b:

$D_b(a) = 8$	$D_b(f) = \infty$
$D_b(c) = 1$	$D_b(g) = \infty$
$D_b(d) = \infty$	$D_b(h) = \infty$
$D_b(e) = 1$	$D_b(i) = \infty$

DV in c:

$D_c(a) = \infty$
$D_c(b) = 1$
$D_c(c) = 0$
$D_c(d) = \infty$
$D_c(e) = \infty$
$D_c(f) = \infty$
$D_c(g) = \infty$
$D_c(h) = \infty$
$D_c(i) = \infty$

DV in e:

$D_e(a) = \infty$
$D_e(b) = 1$
$D_e(c) = \infty$
$D_e(d) = 1$
$D_e(e) = 0$
$D_e(f) = 1$
$D_e(g) = \infty$
$D_e(h) = 1$
$D_e(i) = \infty$

Network Layer: 5-34

## 距离向量路由选择算法例子



t=1

- c 接收来自 b 的 DV, 计算:

$$\begin{aligned}
 D_c(a) &= \min\{c_{c,b} + D_b(a)\} = 1 + 8 = 9 \\
 D_c(b) &= \min\{c_{c,b} + D_b(b)\} = 1 + 0 = 1 \\
 D_c(d) &= \min\{c_{c,b} + D_b(d)\} = 1 + \infty = \infty \\
 D_c(e) &= \min\{c_{c,b} + D_b(e)\} = 1 + 1 = 2 \\
 D_c(f) &= \min\{c_{c,b} + D_b(f)\} = 1 + \infty = \infty \\
 D_c(g) &= \min\{c_{c,b} + D_b(g)\} = 1 + \infty = \infty \\
 D_c(h) &= \min\{c_{c,b} + D_b(h)\} = 1 + \infty = \infty \\
 D_c(i) &= \min\{c_{c,b} + D_b(i)\} = 1 + \infty = \infty
 \end{aligned}$$

DV in b:

$D_b(a) = 8$	$D_b(f) = \infty$
$D_b(c) = 1$	$D_b(g) = \infty$
$D_b(d) = \infty$	$D_b(h) = \infty$
$D_b(e) = 1$	$D_b(i) = \infty$

DV in c:

$D_c(a) = \infty$
$D_c(b) = 1$
$D_c(c) = 0$
$D_c(d) = \infty$
$D_c(e) = \infty$
$D_c(f) = \infty$
$D_c(g) = \infty$
$D_c(h) = \infty$
$D_c(i) = \infty$

DV in c:

$D_c(a) = 9$
$D_c(b) = 1$
$D_c(c) = 0$
$D_c(d) = 2$
$D_c(e) = \infty$
$D_c(f) = \infty$
$D_c(g) = \infty$
$D_c(h) = \infty$
$D_c(i) = \infty$

\* Check out the online interactive exercises for more examples:  
[http://gaia.cs.umass.edu/kurose\\_ross/interactive/](http://gaia.cs.umass.edu/kurose_ross/interactive/)

Network Layer: 5-35

## 距离向量路由选择算法例子



t=1

- e 接收来自 b, d, f, h 的 DV

DV in d:

$D_d(a) = 1$
$D_d(b) = \infty$
$D_d(c) = \infty$
$D_d(d) = 0$
$D_d(e) = 1$
$D_d(f) = \infty$
$D_d(g) = 1$
$D_d(h) = \infty$
$D_d(i) = \infty$

DV in b:

$D_b(a) = 8$	$D_b(f) = \infty$
$D_b(c) = 1$	$D_b(g) = \infty$
$D_b(d) = \infty$	$D_b(h) = \infty$
$D_b(e) = 1$	$D_b(i) = \infty$

DV in e:

$D_e(a) = \infty$
$D_e(b) = 1$
$D_e(c) = \infty$
$D_e(d) = 1$
$D_e(e) = 0$
$D_e(f) = 1$
$D_e(g) = \infty$
$D_e(h) = 1$
$D_e(i) = \infty$

DV in h:

$D_h(a) = \infty$
$D_h(b) = \infty$
$D_h(c) = \infty$
$D_h(d) = \infty$
$D_h(e) = 1$
$D_h(f) = \infty$
$D_h(g) = 1$
$D_h(h) = 0$
$D_h(i) = 1$

DV in f:

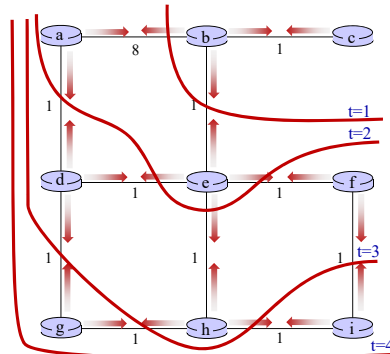
$D_f(a) = \infty$
$D_f(b) = \infty$
$D_f(c) = \infty$
$D_f(d) = \infty$
$D_f(e) = 1$
$D_f(f) = 0$
$D_f(g) = \infty$
$D_f(h) = \infty$
$D_f(i) = 1$

Network Layer: 5-36

## 距离向量路由选择算法: 状态信息的扩散

迭代通信, 计算步骤沿着网络扩散开来:

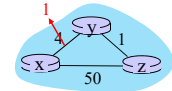
- ①  $t=0$   $t=0$ 时c的状态仅在c处
- ②  $t=1$  c在 $t=0$ 时的状态已传播到b处, 并且影响了1跳处(即b)的距离向量的计算
- ③  $t=2$  c在 $t=0$ 时的状态已传播到并影响了2跳处的距离向量的计算, 传播到b处和现在的a、e处
- ④  $t=3$  c在 $t=0$ 时的状态已传播到并影响了3跳处的距离向量的计算, 传播到b、a、e处和现在的d、f、h处
- ⑤  $t=4$  c在 $t=0$ 时的状态已传播到并影响了3跳处的距离向量的计算, 传播到b、a、e、d、f、h处和现在的g、i处



## 距离向量路由选择算法: 链路开销的变化

链路开销的变化:

- 节点检测链路开销变化
- 更新路由选择信息, 重新计算本地DV
- 如果DV发生变化, 通知邻居



$t_0$ : y节点检测链路开销变化, 更新它的DV, 并通知邻居.

“好消息传得快”

$t_1$ : z收到y更新, 更新它的距离表, 计算它到x新的链路开销, 发送新的DV给它的邻居.

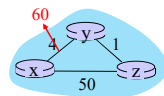
$t_2$ : y收到z的更新, 更新它的距离表. y的最小链路开销没有变化, 所以y不发送报文给z.

Network Layer: 5-38

## 距离向量路由选择算法: 链路开销的变化

链路开销的变化:

- 节点检测链路开销的变化
- “坏消息传得慢” – 无穷计数问题:



- 链路开销变化之前,  $D_y(x)=4$ ,  $D_z(z)=1$ ,  $D_z(y)=1$ ,  $D_z(x)=5$
- y看到与x的直连链路的新开销为60, 但是z上次已告知y它到x的开销仅为5, 所以y将计算得到“我通过z到x的新开销是6”; 并通知z, 其到x的新开销是6。
- z得知通过y到达x的路径具有新开销6, 因此z计算得到“我通过y到达x的新开销是7”, 并通知y, 其到x的新开销是7。
- y得知通过z到达x的路径具有新开销7, 因此y计算得到“我通过z到达x的新开销是8”, 并通知z, 其到x的新开销是8。
- z得知通过y到达x的路径具有新开销8, 因此z计算得到“我通过y到达x的新开销是9, 并通知y, 其到x的新开销是9。

... 问: 这一过程要持续多久?

答: 直到z最终算出它经由y的路径开销大于50为止

## LS (链路状态) 和DV (距离向量) 路由选择算法的比较

报文复杂度

LS:  $n$  路由器, 发送报文 $O(n^2)$

DV: 邻居之间交互; 收敛时间各不相同

收敛速度

LS: 算法复杂度 $O(n^2)$

- 可能会有振荡

DV: 收敛时间各不相同

- 可能有路由选择环路
- 无穷计数问题

健壮性: 路由器在发生故障或受到攻击时, 是否路由选择仍然会正常工作?

LS:

- 路由器广播 **不正确的链路开销**
- 每台路由器 **仅计算自己的表**

DV:

- DV路由器可能会宣告 **不正确链路开销** (如“我到任何地方的开销都非常低”): **造成黑洞**
- 每台路由器的转发表都被其他路由器使用: **错误会通过网络传播**

Network Layer: 5-40

## 网络层：“控制平面”的路线图

- 概述
- 路由选择算法
- ISP内部路由选择协议: OSPF
- ISP间的路由选择协议: BGP
- SDN 控制平面
- 因特网控制报文协议ICMP



- 网络管理，配置
  - SNMP
  - NETCONF/YANG

Network Layer: 5-41

## 实际的路由选择协议 - 令路由选择可扩展

到目前为止，我们的路由选择算法的研究是理想化的，即假定

- 所有路由器相同
- 网络“扁平化”

... 但现实并非如此

**规模:** 数十亿个目的地:

- 无法将所有目的地存储在路由表中!
- 路由表交换报文就会淹没链路!

**管理自治:**

- Internet: 网络的网络
- 每个网络管理员都希望能控制其自己网络中的路由，对外部隐藏其网络的内部组织面貌。

Network Layer: 5-42

## 互联网中实际应用的路由选择协议

将路由器组织进自治系统“Autonomous Systems”(AS) (又称为“域” domains), 每个AS由其全局唯一的ASN标识 (ASN即AS号, 由ICANN区域注册机构分配)

**自治系统内intra-AS (也称域内 “intra-domain”):** 一个AS内 (“network”) 的路由

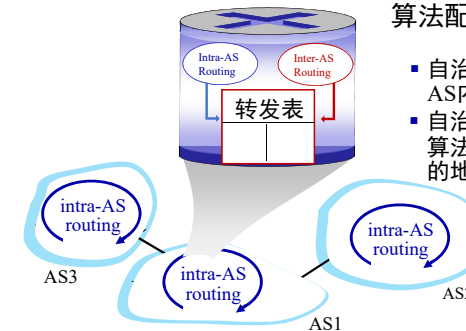
- 同一个AS中的所有路由器必须运行相同的**自治系统内部路由选择协议**
- 不同AS中的路由器可以运行不同的域内路由选择协议
- **网关路由器:** 位于AS边缘的路由器, 直接连接到其他AS中的一台或多台路由器

**自治系统间inter-AS (也称域间inter-domain):** AS间的路由

- 网关执行域间路由选择协议, 以及域内路由选择协议

Network Layer: 5-43

## 互连的自治系统

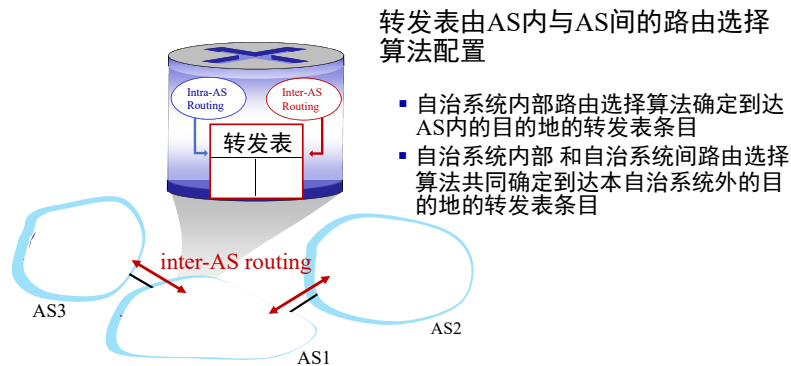


转发表由AS内与AS间的路由选择算法配置

- 自治系统内部路由选择算法确定到达AS内的目的地的转发表条目
- 自治系统内部 和自治系统间路由选择算法共同确定到达本自治系统外的目的地的转发表条目

Network Layer: 5-44

## 互连的自治系统



Network Layer: 5-45

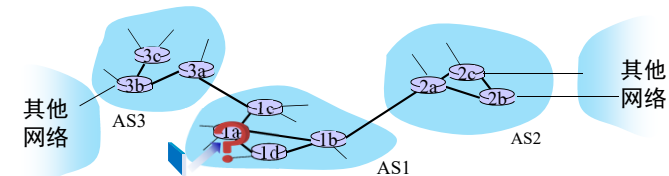
## 自治系统间路由选择协议：在域内转发中的作用

- 假设AS1中的路由器接收发往AS1外部的报文：

? 路由器应将数据包转发到AS1中的哪个网关路由器？

**AS1 域间路由选择协议必须：**

1. 学习到通过AS2可以到达哪些目的地，通过AS3可以到达哪些目的地……
2. 传播此可达信息到AS1中的所有路由器



Network Layer: 5-46

## 路由协议的分类：IGP与EGP

根据范围进行分类

根据AS自治系统的范围：AS内和AS间

ASN: 0-65535。其中，公网唯一ASN 1-64511；64512-65535为私有。ASN拓展为32位，以满足实际需要，目前绝大多数网络设备支持拓展ASN。

- **IGP: Interior Gateway Protocol 内部网关协议**
  - 指常用于企业内部及中小型网络中的路由协议统称
  - 包括: RIP、EIGRP、OSPF、IS-IS
- **EGP: External Gateway Protocol 外部网关协议**
  - 指常用于企业之间及大型网络中使用的路由协议的统称
  - 包括: BGP

Network Layer: 5-47

## 自治系统内部路由选择协议

大部分常见的自治系统内部路由选择协议包括：

- **RIP: 路由信息协议[RFC 1723]**
  - 经典DV算法：DV每30秒交换一次
  - 不再广泛使用
- **EIGRP:增强的内部网关路由协议**
  - 基于DV算法
  - 以前为思科专有并实际使用了约20年的时间，于2013年才成为开放的[RFC 7868])
- **OSPF:开放最短路径优先[RFC 2328]**
  - 经典链接状态路由选择算法
  - IS-IS 协议与OSPF基本相同；**IS-IS (IntermediateSystem-to-Intermediate System, 中间系统 - 中间系统) 路由选择协议**，是ISO 标准, 但不是 RFC 标准

Network Layer: 5-48

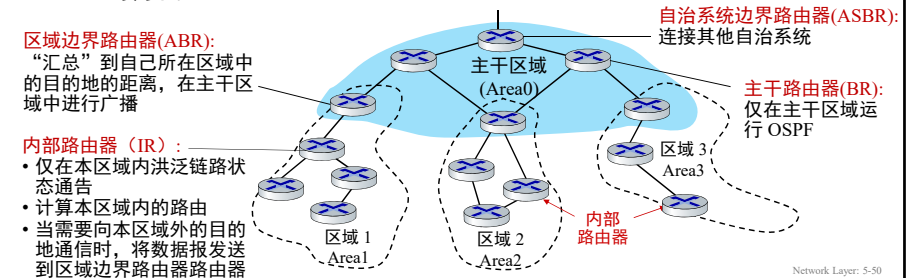
## OSPF (Open Shortest Path First, 开放最短路优先) 路由选择协议[RFC 2328]

- 协议名称中的开放 (“open”) 是指：该协议规范是公众可用的
- 经典的链路状态算法
  - 使用洪泛链路状态信息和Dijkstra最低开销路径算法
  - 各条链路开销由网络管理员配置，可以有多个链路成本指标：带宽，延迟，或者将所有链路开销设为1
  - 每台路由器向自治系统内的所有其他路由器广播路由选择信息：链路状态发生变化时，就会广播；若链路状态未变化，也要周期性地（至少每隔30分钟一次）广播链路状态。
  - 链路状态通告包含在OSPF报文中，该报文由IP承载，其IP报头协议号为89。OSPF协议须自己实现诸如可靠报文传输、链路状态广播等功能。OSPF链接状态通告泛洪到整个AS中的所有其他路由器
  - 每台路由器都具有完整的拓扑，使用Dijkstra的算法来计算转发表
- 安全：可以配置两类鉴别：简单的和MD5。仅有受信任的路由器能参与一个AS内的OSPF协议，以防止恶意入侵，防止将不正确的信息注入路由器转发表内

Network Layer: 5-49

## OSPF支持在单个AS中的层次结构

- 一个OSPF自治系统可以配置两级层次结构：区域和主干区域。
  - 链路状态通告广播仅在本区域内和主干区域中洪泛
  - 每个节点都有其所在本地区域的详细拓扑；只知道到达其他目的地的方向



Network Layer: 5-50

## 网络层：“控制平面”的路线图

- 概述
- 路由选择算法
- ISP内部路由选择协议: OSPF
- ISP间的路由选择协议: BGP
- SDN 控制平面
- 因特网控制报文协议ICMP



- 网络管理，配置
  - SNMP
  - NETCONF/YANG

Network Layer: 5-51

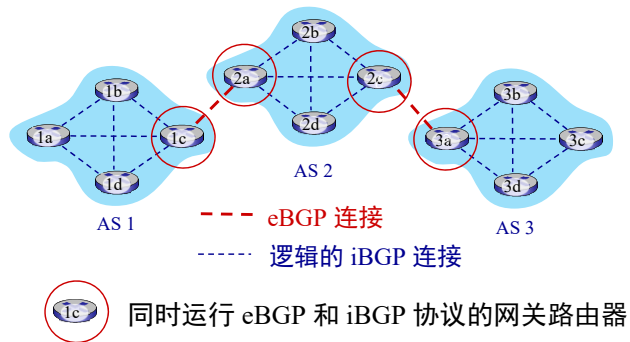
## ISP间的路由选择协议： BGP

- BGP (Border Gateway Protocol, 边界网关协议 [RFC4271]): 互联网事实上的域间路由选择标准，是一种基于策略的路由选择协议
  - “将互联网连接在一起的胶水”
- 允许子网（用CIDR化的前缀表示）向互联网的其余部分通告其存在以及如何到达它那里：
 

“我存在，我在这里，以及如何到达我”
- BGP 为每个自治系统、每台路由器提供了完成以下任务的手段：
  - 从邻居AS获得前缀（子网）的可达性信息
    - 每个前缀标识一个子网或者一个子网的集合，如138.16.68/22
    - 外部BGP (eBGP) 连接: 从邻居AS获取前缀的可达性信息
    - 内部BGP (iBGP) 连接: 向AS内部的所有路由器传播子网的可达性信息
  - 确定到该前缀的“最好的”路由
    - 一台路由器运行一个BGP路由选择过程
    - 根据策略和可达性信息确定到该前缀的“最好的”路由

Network Layer: 5-52

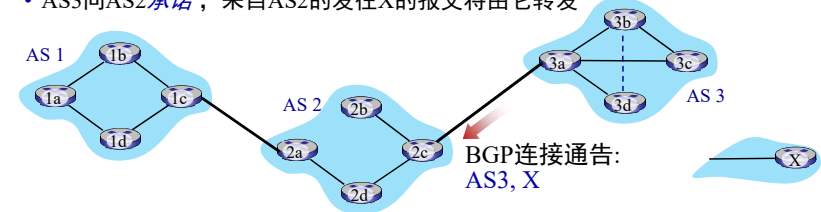
## eBGP, iBGP 连接



Network Layer: 5-53

## BGP 基础

- **BGP 会话:** 两个BGP路由器通过使用179端口的半永久TCP连接（连接建立后可能长时间不拆除）来交换BGP报文
  - 通告去往不同前缀的路径（*paths*）（BGP是“路径向量”协议，从设计上避免了环路的发生-可以明确地从AS路径列表中知道路由信息是否源于自己）
- 当AS3网关路由器3a将BGP报文 即**路径 AS3,X** 发送给AS2的网关路由器 2c时:
  - AS3向AS2**承诺**，来自AS2的发给X的报文将由它转发



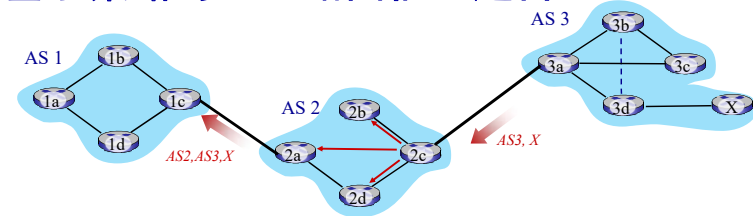
Network Layer: 5-54

## BGP 路由：前缀及其属性

- BGP 路由：前缀 + 属性（带有属性的前缀称为一条路由）
  - 前缀：子网或者子网的集合
  - 两个较为重要的属性：
    - **AS-PATH:** 包含了通告已经通过的AS的列表，当一个前缀通过某AS时，该AS将其ASN加入 AS-PATH中的现有列表。路由器使用该属性来检测和防止循环通告。如果一台路由器看到它的AS 被包括在该路径列表中，它将拒绝该通告。
    - **NEXT-HOP:** 是AS-PATH起始的路由器接口的IP地址
- **基于AS策略的路由选择:**
  - 接收路由连接通告的网关路由器根据**入口策略import policy** 接受/或者拒绝该条路径
    - 例如，若有不允许经过AS1的入口策略，则该网关将拒绝接收“AS1,y”这条路径
  - AS策略也决定是否向其他邻居AS通告该路径

Network Layer: 5-55

## 基于策略的BGP路由信息通告

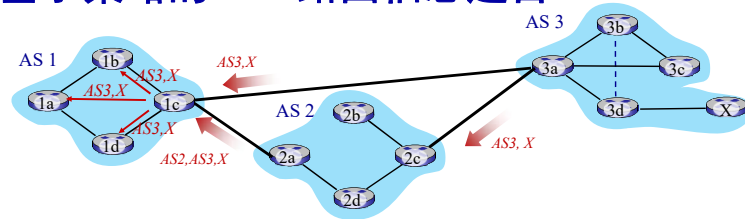


- AS2网关路由器2c收到来自AS3路由器3a的eBGP报文 “AS3, X”
- 基于AS2的策略，AS2网关路由器2c接受路径 “AS3, X”，并向AS2中的所有其他路由器（包括网关路由器2a）发送iBGP报文 “AS3, X”
- 基于AS2策略，AS2网关路由器2a向AS1网关路由器1c 发送eBGP报文 “AS2, AS3, X”

Network Layer: 5-56



## 基于策略的BGP路由信息通告

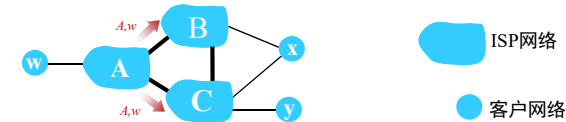


网关路由器到一个目的子网可能有多条路径  
AS1的网关路由器1c到前缀X有两条路径：

- 路径  $AS2, AS3, X$
- 路径  $AS3, X$

根据策略，AS1网关路由器1c选择路径  $AS3, X$ ，并通过iBGP报文通告AS1内的所有路由器

## 基于策略的BGP的部署 – 例子



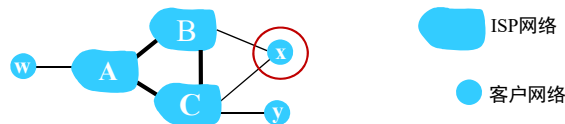
一种典型的“现实世界”策略：

ISP只希望将流量路由到其客户网络或从其客户网络路由（不希望在其他ISP之间传输中转流量）

- A 向B和C通告路径  $A, w$
- B 根据策略选择不通告路径  $B, A, w$  给C!
  - B 没有从路由  $C, B, A, w$  获得“收入”，因为C, A, w 都不是 B的客户
  - C 不知道路径  $C, B, A, w$
- C 到达w将使用路由  $C, A, w$  (不使用B)

Network Layer: 5-58

## 基于策略的BGP的部署 – 例子



一种典型的“现实世界”策略：

ISP只希望将流量路由到其客户网络或从其客户网络路由（不希望在其他ISP之间传输中转流量）

- A,B,C 是 服务提供商网络
- x,w,y 是ISP的客户网络
- x 双宿主的: 同时是两个ISP的客户，连接到两个网络
- 执行的策略: X不想承担从B到C的路由
  - .. 则x就不会向B通告：我有一条路由可以到达C

Network Layer: 5-59

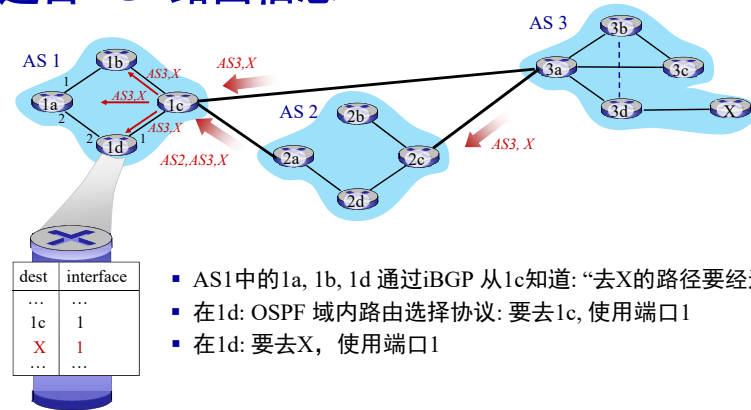
## BGP 报文

- 在TCP连接的对等体之间交换BGP报文
- BGP 报文:
  - OPEN: 打开与远程BGP对等体的TCP连接、并验证BGP发送方；包括BGP版本号、自己所属的AS号、路由器ID、Hold Time值、认证信息等
  - UPDATE: 通告新路由（或撤消旧路由）
  - KEEPALIVE: 在没有更新报文的情况下保持连接状态；确认OPEN请求；19字节，只有首部，包括标记、长度、类型等，没有数据
  - NOTIFICATION: 如果发现对方发过来的消息有错误或者主动断开BGP链接，就发送该消息，并关闭链接；如收到该消息，也会将链接变为idle（空闲）状态。
  - ROUTE-REFRESH: 请求对等体重新发送路由信息

Network Layer: 5-60



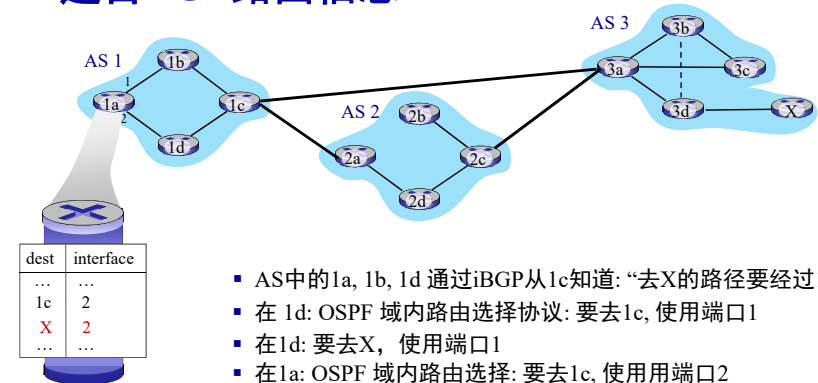
## 通告BGP路由信息



- AS1中的1a, 1b, 1d 通过iBGP 从1c知道: “去X的路径要经过1c”
- 在1d: OSPF 域内路由选择协议: 要去1c, 使用端口1
- 在1d: 要去X, 使用端口1

Network Layer: 5-61

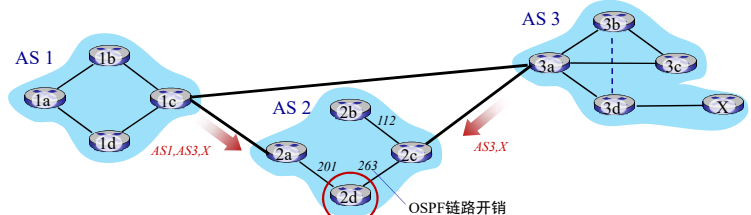
## 通告BGP路由信息



- AS中的1a, 1b, 1d 通过iBGP 从1c知道: “去X的路径要经过1c”
- 在 1d: OSPF 域内路由选择协议: 要去1c, 使用端口1
- 在1d: 要去X, 使用端口1
- 在1a: OSPF 域内路由选择: 要去1c, 使用端口2
- 在1a: 要去X, 使用端口2

Network Layer: 5-62

## 如何确定最好的路由-热土豆路由选择算法



- AS2中的路由器2d通过iBGP知道, 它可以通过2a或2c路由到X
- 热土豆路由选择算法:** 选择域内成本最低的本地网关、不必担心域间成本!
  - 尽可能快地(确切地说, 用可能的最低开销)将分组送出其AS, 而不担心其AS外部到目的地的余下部分的开销
  - 减小它自己AS中的开销, 忽略在其他AS之外的端到端开销的其他部分
  - 如2d选择2a, 即使通过2a的路由会通过2个自治系统AS1和AS3、更多跳

Network Layer: 5-63

## 如何确定最好的路由-路由器选择算法

- 路由器可能知道到达一条前缀的多条路由, 在这种情况下, 路由器必须在可能的路由中选择一条放入转发表中
- 路由器会顺序地调用下列规则, 直到留下一条路由:
  - 本地偏好值**属性: 策略决策, 取决于该AS的网络管理员。由该路由器设置或由在相同AS中的另一台路由器学习到, 具有最高本地偏好值的路由将被选择。
  - 在余下的路由中, 具有**最短AS-PATH**的路由将被选择
  - 在余下的路由中, 具有**最近NEXT-HOP路由器**的路由将被选择: 即热土豆路由
  - 附加规则, 如使用BGP标识符来选择路由等

Network Layer: 5-64

## 为什么采用不同的AS内和AS间路由协议？

### 策略

- 在AS之间，策略问题起主导作用。一个给定的AS也许不能穿过另一个特定的AS，可能非常重要，这不是一个简单的路由选择算法可以解决的问题
- 在AS之内，策略问题微不足道，一切都是在相同的管理控制下进行

### 规模

- 在AS之间，扩展性是AS间路由选择的一个关键问题
- 在AS之内，可扩展性不是关注的焦点，因为如果单个管理域变得太大时，总是能将其分成两个AS（如OSPF层次划分）

### 性能

- 在AS之间，由于AS间路由选择是面向策略的，因此所用路由的性能通常是次要关心的问题（即一条费用更高但能满足某些策略要求的路由也许被采用），甚至没有与路由相关的费用概念（除了AS跳计数外）
- 在AS之内，不关心策略，更多地关注性能

Network Layer: 5-65

## Internet中主流的路由协议：RIP、OSPF、BGP

是路由器用来计算、维护网络路由信息的协议

–RIP基于UDP, 端口号520

–OSPF基于IP, 端口号89

–BGP基于TCP, 端口号179

BGP	RIP	OSPF
TCP	UDP	
IP		
链路层		
物理层		

Network Layer: 5-66

## 网络层：“控制平面”的路线图

### 概述

### 路由选择算法

### ISP内部路由选择协议: OSPF

### ISP间的路由选择协议: BGP

### SDN 控制平面

### 因特网控制报文协议ICMP



### 网络管理，配置

- SNMP
- NETCONF/YANG

Network Layer: 5-67

## 软件定义网络(SDN)

### 互联网的网络层控制平面：历史上是分布式的、每路由器控制

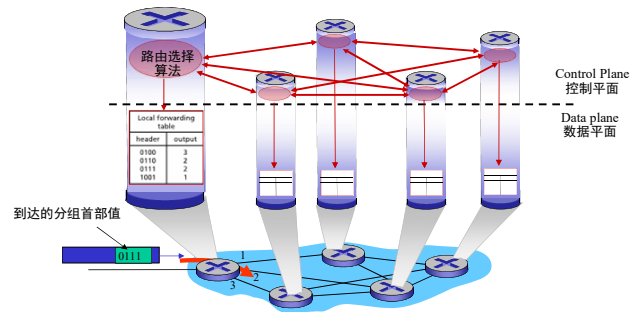
- 每台路由器包含交换硬件，专有路由器操作系统（例如Cisco IOS）运行互联网的标准协议（IP，RIP，IS-IS，OSPF，BGP）
- 不同的“中间盒子”实现不同的网络层功能：防火墙、负载均衡、网络地址转换..

### ~2005:重新思考网络控制平面

Network Layer: 5-68

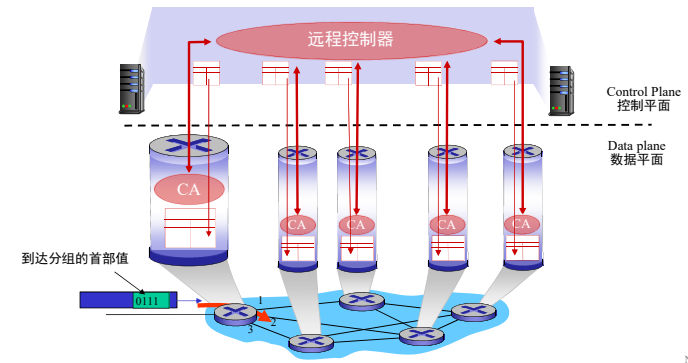
## 每路由器控制的控制平面

每台路由器有一个路由选择组件，用于与其他路由器中的路由选择组件通信，以计算路由器转发表的值。



## 软件定义网络(SDN) 的控制平面

远程控制器计算并分发转发表，以供每台路由器使用



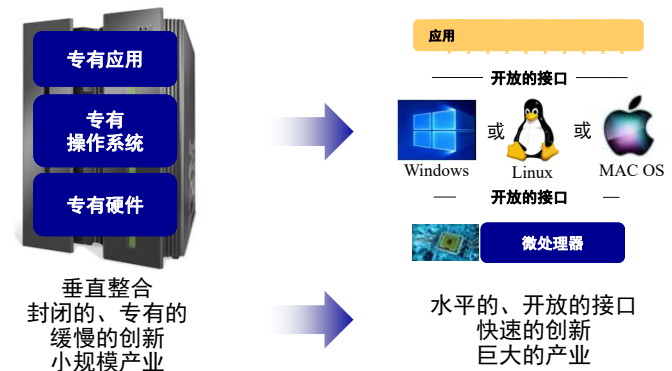
## 软件定义网络(SDN)

为什么是逻辑集中式控制的控制平面？

- 简化网络管理：避免路由器配置错误，更加灵活地处理网络流
- 可“编程”的网络，基于表的转发允许编程路由器
  - 集中式“编程”更加容易：集中计算表并分发
  - 分散式“编程”更加困难：需要每台路由器实现分散式的算法
- 控制平面的实现是开放式的
  - 非专有、任何公司或团体都可以部署和实现
  - 促进创新：百花盛开

Network Layer: 5-71

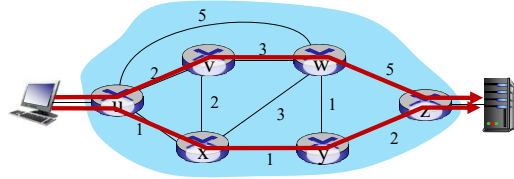
## SDN的类比：大型机到PC的变革



\* Slide courtesy: N. McKeown

Network Layer: 5-72

## 网络流量工程：传统路由选择的难题



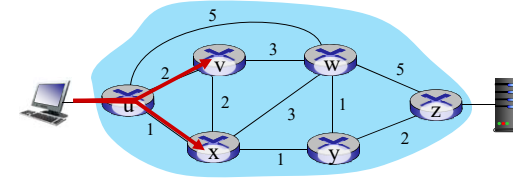
问：如果网络运营商希望u到z的网络流量沿uvwz而不是uxyz流动，该怎么办？

答：需要重新定义链路权重，使得路由选择算法可以相应地计算出这种希望的流路由选择（或需要新的路由算法）！

**链路权重仅是控制“旋钮”：没有太多控制权！**

Network Layer: 5-73

## 网络流量工程：传统路由选择的难题

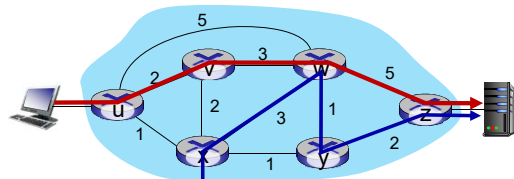


问：如果网络运营商希望u到z的网络流量平均分配到uvwz和uxyz两条路径（负载均衡），该怎么办？

答：使用传统路由选择算法是无法做到的（或需要新的路由算法）

Network Layer: 5-74

## 网络流量工程：传统路由选择的难题



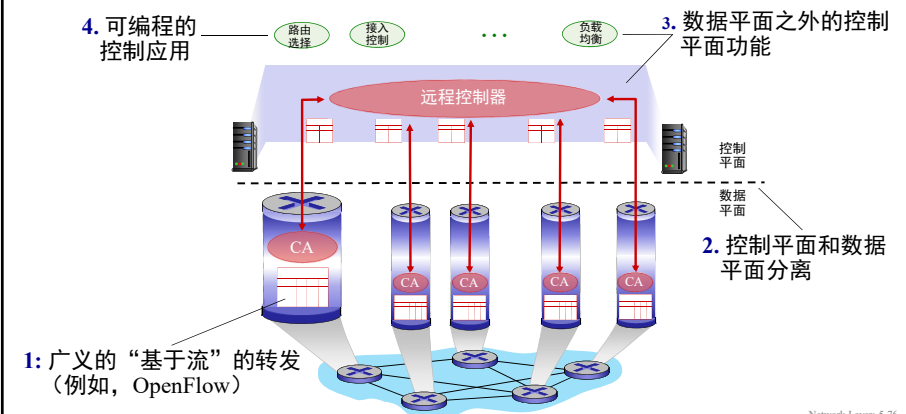
问：如果w作为中间路由器，对到达自己处的流量进行转发时区分红色和蓝色的流量，即红色流量从w直接到z，而蓝色流量要经过y再到z，该怎么办？

答：传统该路由选择做不到（使用的是基于目的地址的转发、LS，DV路由选择算法）

**使用通用转发和SDN可实现所需的路由**

Network Layer: 5-75

## 软件定义网络(SDN)

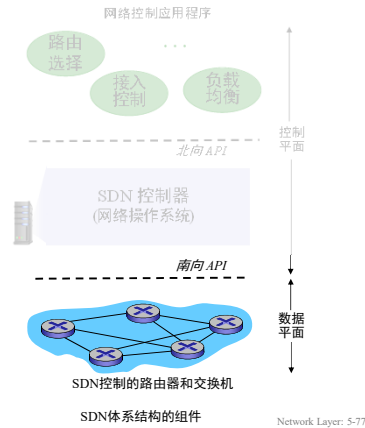


Network Layer: 5-76

## 软件定义网络(SDN)

### 与数据平面的交互:

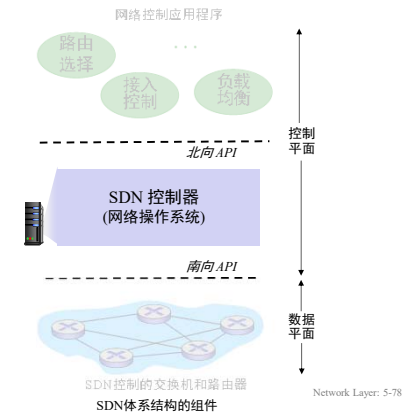
- 通用的数据平面的转发由快速的、简单的、商用交换机硬件实现（第4.4节）
- 远程控制器负责计算并分发流表（转发表）到路由器中
- SDN提供了用于表交换的API
  - 定义什么是可被控制的、可编程的
- SDN定义了与控制器通信的协议（如OpenFlow）



## 软件定义网络(SDN)

### SDN 控制器 (网络操作系统):

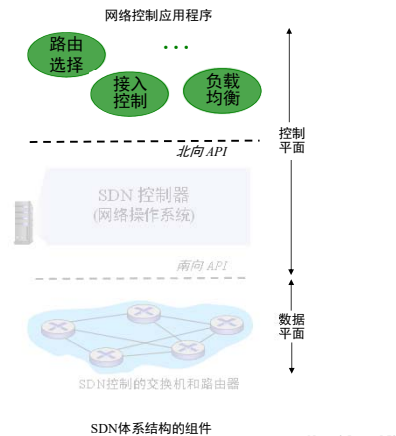
- 维护网络状态信息
- 通过北向API与“上方”的网络控制应用程序交互
- 通过南向API与“下方”的受控设备交互
- 在实践中采用分布式服务器集合来实现，以实现故障容忍、高可用性、可伸缩性和高性能



## 软件定义网络(SDN)

### 网络控制应用程序:

- 控制的“大脑”：使用SDN提供的低层的服务和API来实现控制功能
- 非捆绑的：
  - 网络控制应用程序可由非路由器供应商、非SDN控制器的第三方提供

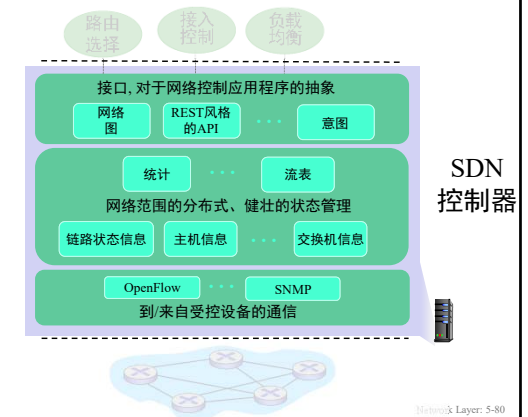


## SDN控制器的组件

**对于网络控制应用程序的接口:** 该API允许网络控制应用程序在状态管理层之间读/写网络状态和流表

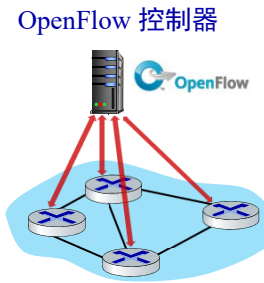
**网络范围状态管理层:** 网络链路、交换机、和其他SDN控制设备的最新状态信息的管理: **一个分布式数据库**

**通信层:** SDN控制器与受控交换机，与受控设备之间的通信



## OpenFlow 协议

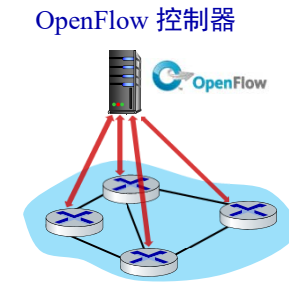
- 运行在SDN控制器和SDN控制的交换机或其他实现OpenFlowAPI的设备之间
- 运行在TCP之上，使用6653的默认端口号
  - 可选加密通信
- 三类OpenFlow 报文:
  - Controller-to-Switch (控制器到交换机): 消息由控制器发出，主要用于管理和获取switch状态
  - Asynchronous(交换机到控制器): 消息由交换机发出，用于网络事件和交换机状态变化更新发送到控制器
  - symmetric (其他): 消息可以由交换机或控制器发出。
- 与 API不同
  - API 用于制定广义的转发操作



Network Layer: 5-81

## OpenFlow: 控制器到受控交换机的主要报文

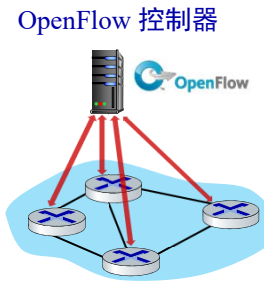
- 读状态:** 从交换机的流表和端口收集统计数据和计数器值
- 配置:** 控制器查询并设置交换机配置参数
- 修改状态:** 增加、删除或修改交换机流表中的表项，并且设置交换机端口特性
- 发送分组:** 在受控交换机从特定的端口发送出一个特定的报文



Network Layer: 5-82

## OpenFlow: 受控交换机到控制器的主要报文

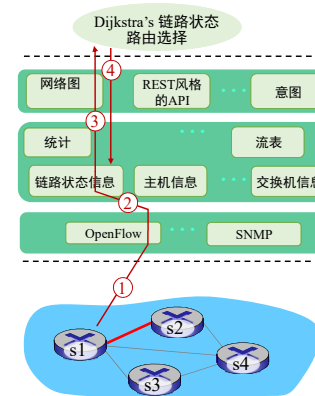
- 分组入:** 一个分组到达交换机端口，并且不能与任何流表表项匹配，则被发送给控制器进行额外处理。匹配的分组也被发送给控制器，作为匹配是所采取的一个动作。该报文用于将分组发给控制器
- 流删除:** 通知控制器已删除一个流表项
- 端口状态:** 向控制器通知端口状态的变化



幸运的是，网络运营商不会通过创建/发送OpenFlow报文来对交换机直接进行“编程”，而是使用控制器上的API

Network Layer: 5-83

## SDN: 控制/数据平面相互作用示例

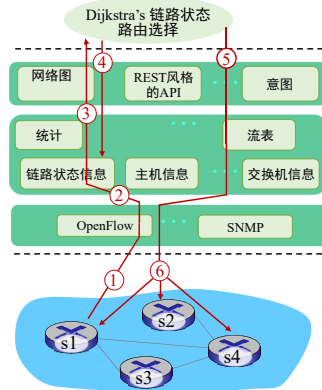


- ① S1遇到链路故障，使用OpenFlow端口状态报文通知控制器
- ② SDN控制器收到OpenFlow报文，更新链路状态信息
- ③ Dijkstra路由选择算法应用被调用，该应用先前已注册为每次链路状态发生变化时都会被调用。
- ④ Dijkstra路由选择算法访问网络图信息、控制器中的链路状态信息，并计算新路由

Network Layer: 5-84



## SDN: 控制/数据平面相互作用示例

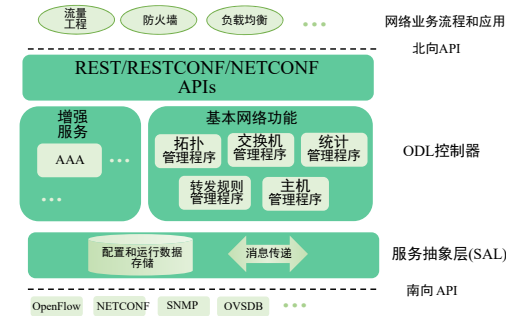


⑤ 链路状态路由选择应用与SDN控制器中的流表计算组件进行交互，该组件计算出新流表

⑥ 控制器使用OpenFlow在需要更新的交换机中安装新的路由表

Network Layer: 5-85

## OpenDaylight (ODL) 控制器



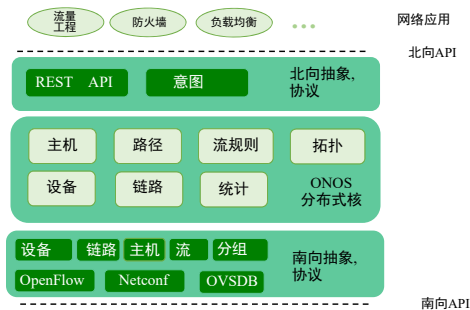
服务抽象层:

- 互连内部、外部应用程序和服务

OpenDaylight (ODL) 是Linux基金会负责管理的**开源SDN控制器**，是一款使用JAVA开发的控制器，提供一套基于SDN开发的**模块化**、**可扩展**、**可升级**、**支持多协议**的控制器框架，目的是推动SDN技术的创新实施和透明化。

Network Layer: 5-86

## ONOS 控制器



ONOS是一个采用OSGI技术来管理子项目的**开源SDN控制器**。其设计的目标是：**代码模块化**:支持新的功能作为新的独立单元引入；**特性可配置**:可动态加载和卸载特性；**支持协议无关**:应用不需要和具体的协议库和实现绑定。

Network Layer: 5-87

## SDN的优势

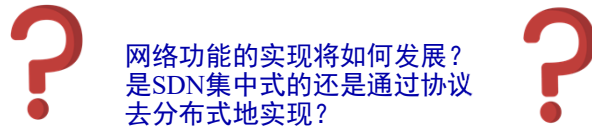
- 强化控制平面：令控制平面更可信、可靠、性能可扩展、实现一个安全的分布式系统
  - 对故障的鲁棒性：将可靠的分布式系统的强大理论应用于控制平面
  - 可信性和安全性：在设计时就“融入”了这些概念？
- 网络和协议的设计目标
  - 实时性，可靠性，安全性
- 规模可扩展：不限于单个AS
- SDN在5G移动通信网络中有很重要的应用**

Network Layer: 5-88



## SDN和传统网络协议的未来

- SDN计算的转发表和每路由器计算的转发表
  - 仅是逻辑集中计算与协议计算的一个例子
- 我们甚至可以想象，比如SDN计算的拥塞控制：
  - 控制器根据路由器报告的（发送给控制器的）拥塞级别设置发送速率



Network Layer: 5-89

## 网络层：“控制平面”的路线图

- 概述
- 路由选择算法
- ISP内部路由选择协议: OSPF
- ISP间的路由选择协议: BGP
- SDN 控制平面
- 因特网控制报文协议ICMP
  - 网络管理，配置
    - SNMP
    - NETCONF/YANG



Network Layer: 5-90

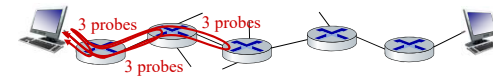
## ICMP:因特网控制报文协议

- 被主机和路由器用来彼此沟通网络层的信息
  - 差错报告: 目的网络、主机、协议、端口不可达
  - 回显请求: 对ping的回答
- ICMP通常被认为是IP的一部分，但从体系结构上讲它位于IP之上：
  - 承载在IP分组的载荷中
- **ICMP 报文:**
  - 类型字段
  - 编码字段
  - 引起该ICMP报文首次生成的IP数据报的首部和前8个字节（以便发送方能确定引发该差错的数据报）
  - 如 ping—发送ICMP类型8、编码0的报文到指定主机、Echo请求报文；

类型	编码	描述
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

Network Layer: 4-91

## Traceroute和ICMP



- Traceroute是用ICMP报文来实现的
- 源主机的Traceroute向目的主机发送一系列普通的IP数据报，这些数据报的每个携带了一个具有**不可达UDP端口号的UDP报文段**
  - 1<sup>st</sup> 数据报TTL设置为1, 2<sup>nd</sup>数据报的TTL设置为2, etc.
- 源主机为每个数据报启动定时器
- 当第n个数据报到达第n台路由器时：
  - TTL正好过期
  - 根据IP协议规则，路由器丢弃该数据报并发送ICMP告警报文给源主机（**类型11、编码0、TTL expired**）
  - 该ICMP告警报文包含该路由器的名称和它的IP地址
- 当该ICMP报文到达源主机时：源主机从定时器得到往返时延，从ICMP报文得到第n台路由器的名字与IP地址

**Traceroute源主机怎样知道何时停止发送UDP报文段呢？**

- 源主机为它发送的每个报文段的TTL字段加1
- 最终会有一个报文段到达目的主机
- 由于该数据包包含了一个具有不可达端口号的UDP报文，该目的主机将向源发送一个端口不可达的ICMP报文（**类型3、编码3、dest port unreachable**）
- 源停止

Network Layer: 4-92

## 网络层：“控制平面”的路线图

- 概述
- 路由选择算法
- ISP内部路由选择协议: OSPF
- ISP间的路由选择协议: BGP
- SDN 控制平面
- 因特网控制报文协议ICMP



- 网络管理，配置
  - SNMP
  - NETCONF/YANG

Network Layer: 5-93

## 什么是网络管理？

- 每个自治系统（或者称为网络）：数千个交互的硬件和软件的组件
- 复杂系统需要监视、配置和控制：
  - 如喷气飞机、核电站
  - 人肉的方式，一台台修改，运维工程人员比较累（还可能出错）



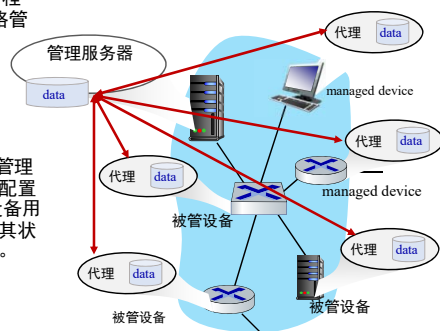
**网络管理：**包括了硬件、软件和人类元素的设置、综合和协调，以监视，测试，轮询，配置，分析，评价和控制网络及网元资源，用合理的成本满足实时性、运营性能和服务质量的要求

Network Layer: 5-94

## 网络管理的组成部分

**管理服务器：**应用程序，通常有人（网络管理员）的参与

**网络管理协议：**管理服务器用来查询和配置被管设备；被管设备用于通知管理服务器其状态数据以及事件等。



**被管设备：**具有可管理，可配置的硬件和软件组件

**管理信息库数据：**设备“状态”的配置数据、操作数据、设备统计信息

Network Layer: 5-95

## 网络管理的方法

**CLI (Command Line Interface 命令行界面)**

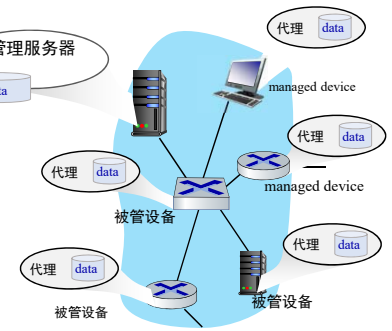
- 直接对每个设备发命令或者脚本（例如通过 ssh）

**SNMP/MIB**

- 使用简单网络管理协议（SNMP）查询/设置被管设备数据（MIB）

**NETCONF/YANG**

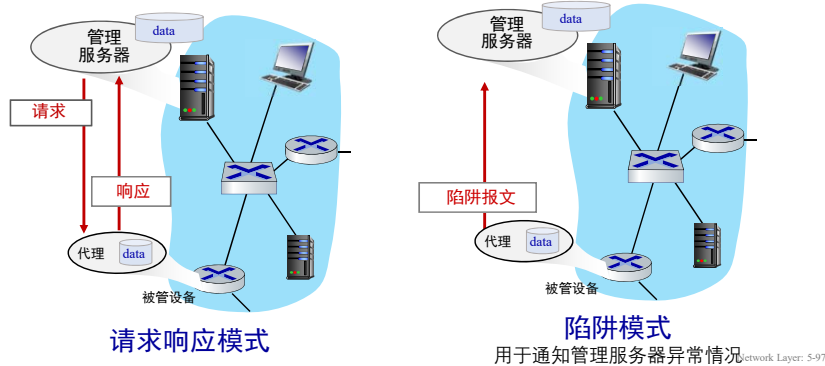
- 更抽象的，网络范围内、全面的
- 强调对多设备的配置管理
- YANG是一种数据建模语言
- NETCONF:与远端设备或者在远端设备之间交互兼容YANG的动作和数据



Network Layer: 5-96

## SNMP 协议

有两种模式来传递网络管理控制和信息报文：

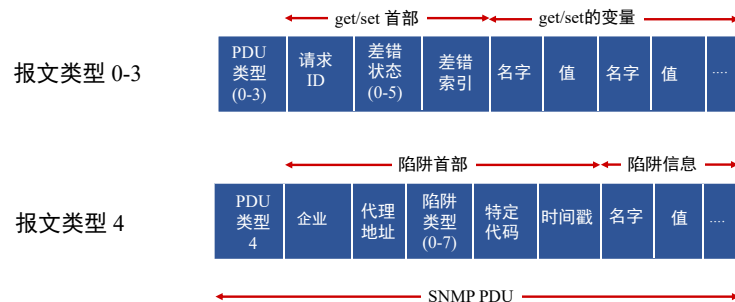


## SNMP 协议: 报文类型

报文类型	功能
GetRequest	管理器到代理 取得一个或多个MIB对象实例值 取得列表或表格中的下一个MIB对象的实例值
GetNextRequest	
GetBulkRequest	
SetRequest	管理器到代理 设置一个或多个MIB对象实例的值
Response	代理到管理器或 管理器到管理器 对GetRequest, GetNextRequest, GetBulkRequest, SetRequest产生的响应
Trap	代理到管理器 向管理器通知一个异常事件


Network Layer: 5-98

## SNMP 协议: 报文格式



## SNMP:管理信息库(MIB)

- 被管设备的操作和配置数据
- MIB 模块**
  - RFC定义了400个MIB模块;不同供应商也有各自特定的MIBs
- 管理信息结构(SMI): 数据定义语言**
- UDP 协议所使用的MIB 变量的例子:



Object ID	Name	Type	Comments
1.3.6.1.2.1.7.1	UDPInDatagrams	32-bit counter	total # datagrams delivered
1.3.6.1.2.1.7.2	UDPNoPorts	32-bit counter	# undeliverable datagrams (no application at port)
1.3.6.1.2.1.7.3	UDInErrors	32-bit counter	# undeliverable datagrams (all other reasons)
1.3.6.1.2.1.7.4	UDPOutDatagrams	32-bit counter	total # datagrams sent
1.3.6.1.2.1.7.5	udpTable	SEQUENCE	one entry for each port currently in use

Network Layer: 5-100

## NETCONF 概述

- **目标:** 在整个网络范围内主动管理和配置设备
- 在管理服务器和被管设备之间运行的协议
  - 操作：检索、设置、激活、增加、修改，删除设备的配置
  - 原子提交操作在多个设备上执行
  - 查询运行数据和统计数据
  - 订阅来自设备的通知
- NETCONF是一种协议，使用远程过程调用（RPC）方式进行通信
  - NETCONF 协议报文以XML编码
  - 通过安全，可靠的传输协议（例如TLS）进行交换

Network Layer: 5-101

## NETCONF初始化，交换，关闭



Network Layer: 5-102

## NETCONF的主要操作

NETCONF	操作描述
<get-config> <get>	取得给定的配置的全部或部分；一个设备可能具有多种配置取得全部或部分配置和运行状态数据。
<edit-config>	更改被管设备上的指定（可能正在运行）配置；被管设备的<rpc-reply> 包含<ok> 或带有回滚的 <rpcerror>
<lock>, <unlock>	锁定（解锁）被管设备上的配置数据
<create-subscription>	订阅被管设备的通知
<notification>	通知事件

Network Layer: 5-103

## NETCONF RPC 消息样例

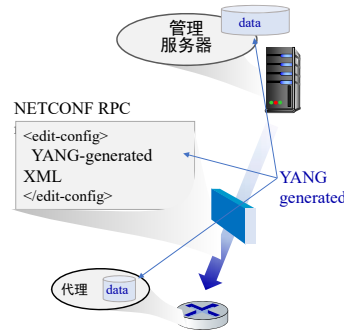
```

01 <?xml version="1.0" encoding="UTF-8"?>
02 <rpc message-id="101" 标识 message id
03   xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
04   <edit-config> 更改配置
05     <target>
06       <running/> 更改运行配置
07     </target>
08     <config>
09       <top xmlns="http://example.com/schema/
10         1.2/config">
11         <interface>
12           <name>Ethernet0/0</name> 更改 Ethernet 0/0 接口的MTU 为1500
13           <mtu>1500</mtu>
14         </interface>
15       </top>
16     </config>
17   </edit-config>
18 </rpc>
  
```

Network Layer: 5-104

## YANG (Yet Another Next Generation)

- 实现NETCONF协议的建模语言有多种，目前最广泛使用的是YANG语言
- YANG是数据建模语言，用于规范NETCONF网络管理数据的结构，语法，语义
  - 使用XML，内置了数据模型
  - 不同组织定义的YANG存在差异，业界有很多种YANG模型
- YANG可以有效地实现NETCONF对被管设备的管理和配置
  - 确保NETCONF配置满足正确性和一致性；可以看做是一个数据模板，运维人员只需要做一个完型填空即可
  - 企业只要基于Yang 文件编写控制器接口，就能通过下发一套Yang报文实现对各个厂商设备的控制，实现配置的兼容性



Network Layer: 5-105

## 第五章：网络层控制平面总结

- 网络控制平面的方法
  - per-router control (传统)
  - logically centralized control (SDN)
- 传统路由选择算法
  - 在Internet中实现: OSPF, BGP
- SDN 控制平面
  - 实现: ODL, ONOS
- ICMP (Internet Control Message Protocol)
- 网络管理

Network Layer: 5-106

## 第5章：网络层：“控制平面”完成！

- 网络层控制平面的方法
  - 每路由器控制
  - 逻辑集中式控制: SDN
- 路由选择算法
  - 链路状态Link State
  - 距离向量Distance Vector
- 实例化，在Internet中实现:
  - ISP内部路由选择协议: OSPF
  - ISP间的路由选择协议: BGP
  - SDN 控制平面
  - 因特网控制报文协议ICMP
- 网络管理，配置
  - SNMP
  - NETCONF/YANG



Network Layer: 5-107