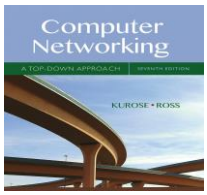
Chapter 5
网络层：控制层面
Network Layer: Control Plane



Computer Networking: A Top Down Approach

7th edition
Jim Kurose, Keith Ross
Pearson/Addison Wesley
April 2016

网络层控制层面：我们的目标

- 了解网络控制层面背后的原理：
  - 传统路由选择算法
  - SDN控制器
  - 网络管理，配置
- 实例化，在Internet中实现：
  - OSPF, BGP
  - OpenFlow, ODL和ONOS 控制器
  - 互联网控制消息协议：ICMP
  - SNMP, YANG/NETCONF

网络层：控制层面的"线路图"

- 介绍
- 路由选择协议Routing protocols
  - 链路状态Link State
  - 距离向量Distance-Vector
- ISP内部路由选择（routing）：OSPF
- ISP间的路由选择（routing）：BGP
- SDN 控制层
- 互联网控制消息协议ICMP

- 网络管理，配置
  - SNMP
  - NETCONF/YANG

网络层功能

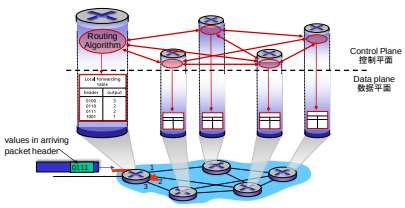
- 转发forwarding: 将分组从一个输入链路接口转移到适当的输出链路接口 — *data plane*
- 路由选择routing: 确定分组从源到目的地所采取的端到端路径的网络范围处理过程 — *control plane*

构造网络控制层的两种方法：

- 每路由控制per-router control (传统)
- 逻辑集中式控制logically centralized control (软件定义的网络)

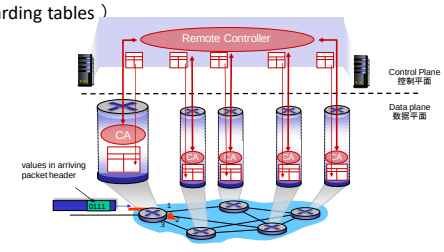
每路由器控制平面Per-router control plane

在控制平面中*每个路由器中的*各个路由选择算法组件相互作用



软件定义网络(SDN) control plane

远程控制器（Remote controller）计算，在路由器中安装转发表（forwarding tables）



网络层：控制层面的“线路图”

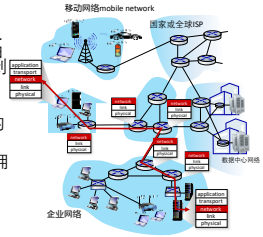
- 介绍
- **路由选择协议Routing protocols**
  - **链路状态Link State**
  - **距离向量Distance-Vector**
- ISP内部路由选择(routing)：OSPF
- ISP间的路由选择(routing)：BGP
- SDN 控制层
- 互联网控制消息协议ICMP

- 网络管理，配置
  - SNMP
  - NETCONF/YANG

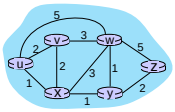
路由选择协议Routing protocols

Routing protocol目标: 通过一条定义良好“good”的路径（即，一连串路由器）将分组跨越网络从发送主机送到接收主机

- 路径path: 分组从给定的初始源主机传输到最终目标主机所遍历过的节点序列
- “good”: “成本”最少，“最快”，“最少拥堵”
- 路由选择: 十大网络挑战之一!



抽象图：链路开销link costs



$c_{a,b}$: *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：每天直接连连链路为1，或者与带宽bandwidth成反比，或者与拥塞congestion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)* }

路由算法分类

网络层：控制层面的“线路图”

- 介绍
- **路由选择协议Routing protocols**
  - **链路状态Link State**
  - **距离向量Distance-Vector**
- ISP内部路由选择(routing)：OSPF
- ISP间的路由选择(routing)：BGP
- SDN 控制层
- 互联网控制消息协议ICMP

- 网络管理，配置
  - SNMP
  - NETCONF/YANG

Dijkstra的链路状态路由选择算法

- 集中式centralized: 所有节点都已知网络拓扑、链路开销
  - 通过“链接状态广播”link state broadcast实现
  - 所有节点存储信息相同
- 计算从一个源节点(“source”)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forwarding table*
- 迭代iterative: 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

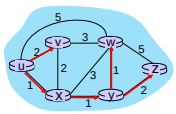
- $c_{x,y}$: 从节点*x* 到 *y的直接* 链路开销;如果不是相邻节点= ∞
- *D(v): 到算法的本次迭代*, 从源节点到目的节点*v*的最低开销路径
- *p(v):* 从源到*v*沿着当前最低开销路径的前一节点
- *N': 明确*知道其最低开销路径的节点集

Dijkstra的链路状态路由选择算法

```
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       /*但可能不是最低成本! */
6    else D(v) = ∞
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的直接
14   路径*/
15 直到所有节点都在数据集N′中
```


Dijkstra算法：样例

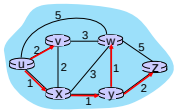
Step	N′	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



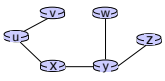
初始化(step 0): 对所有的a: 如果与 *a* 相邻，则 $D(a) = c_{u,a}$

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

Dijkstra算法：样例



从u出发得到的最低开销路径树:



在u中生成的转发表forwarding table :

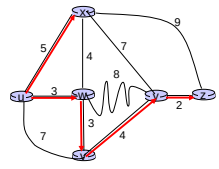
destination	outgoing link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
x	(u,x)

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

Dijkstra算法：样例

Step	N′	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					



注意:
- 通过跟踪前一个节点构造最小成本路径树
- 可以存在纽带ties(也可以任意打破)

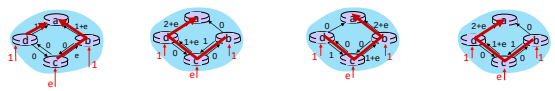
Dijkstra算法：讨论

算法复杂度: *n* 节点
- *n* 次迭代中每一次:需要搜索所有节点*w*,而不是*N*
- $n(n+1)/2$ 比较: $O(n^2)$
- 更高效的实现（堆）的复杂度是: $O(n\log n)$

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

Dijkstra算法：可能的振荡

- 当链路开销取决于流量时，可能会发生路由振荡route oscillations
- 示例场景:
  - 路由到目的地a, 流量以1, e (<1), 1的速度进入d, c, e
  - 链接成本是有方向性的，并取决于数量volume-dependent



初始状态

由于这些成本，寻找新的路由…导致产生新的成本

由于这些成本，寻找新的路由…导致产生新的成本

由于这些成本，寻找新的路由…导致产生新的成本

网络层：控制层面的“线路图”

- 介绍
- 路由选择协议Routing protocols
  - 链路状态Link State
  - 距离向量Distance-Vector
- ISP内部路由选择(routing)：OSPF
- ISP间的路由选择(routing)：BGP
- SDN 控制层
- 互联网控制消息协议ICMP

- 网络管理，配置
  - SNMP
  - NETCONF/YANG

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 $D_x(y)$: 从 x 的 y的最低开销路径的开销.
则:

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

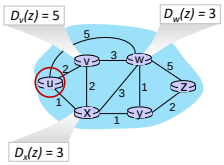
v 到 y 的最低开销路径

取x的所有邻居节点v的min

从 x 到 v的链路开销

Bellman-Ford 样例

假设u 的相邻节点x,v,w 已知目的节点z:



Bellman-Ford 方程:

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

达到最小值的节点(x) 是到达目的节点(z)的最低开销路径的下一跳

距离向量路由选择算法

关键思路:

- 每个节点都会将自己的距离向量估计值发送给邻居neighbor
- 当 x从任何一个邻居接收到新的 DV估计值时，它将使用B-F 方程更新自身的DV :

$$D_x(y) \leftarrow \min_v\{c_{x,v} + D_v(y)\} \text{ 对任意节点 } y \in N$$

- 在较小的自然条件下，估算值$D_x(y)$*收敛到实际的最低开销*$d_x(y)$

距离向量路由选择算法

每个节点:

等待(本地链路开销变化或邻居发送报文)

用从邻居收到的DV *重新计算* DV估计值

如果到任意目的地的 DV 都发生变化, *通知* 邻居

迭代, 异步: 引起局部迭代的因素:

- 本地链路开销发送变化
- 来自邻居的DV更新报文

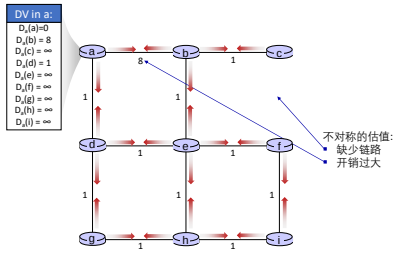
分布式, 自停的: 每个节点 *仅*在其DV更改时通知邻居

- 邻居通知其邻居– 若必要
- 若未收到通知，不采取任何行动!

距离向量: 样例

t=0

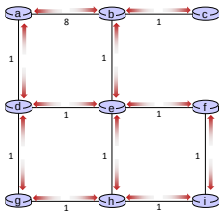
- 所有节点均有其邻接邻居的距离估计值 (only)
- 所有节点均发送其本地距离向量给邻居



距离向量样例: 迭代

t=1

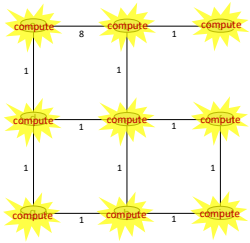
所有节点:
- 接收邻居发送的距离向量
- 完成本地距离向量更新
- 将新的本地距离向量发送给邻居



距离向量样例: 迭代

t=1

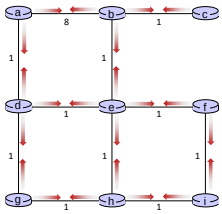
所有节点:
- 接收邻居发送的距离向量
- 完成本地距离向量更新
- 将新的本地距离向量发送给邻居



距离向量样例: 迭代

t=1

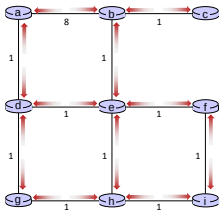
所有节点:
- 接收邻居发送的距离向量
- 完成本地距离向量更新
- 将新的本地距离向量发送给邻居



距离向量样例: 迭代

t=2

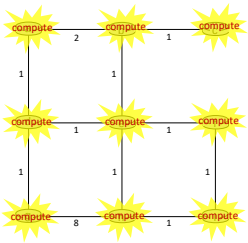
所有节点:
- 接收邻居发送的距离向量
- 完成本地距离向量更新
- 将新的本地距离向量发送给邻居



距离向量样例: 迭代

t=2

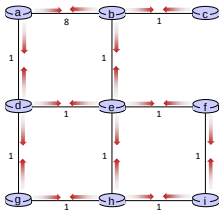
所有节点:
- 接收邻居发送的距离向量
- 完成本地距离向量更新
- 将新的本地距离向量发送给邻居



距离向量样例: 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邻居



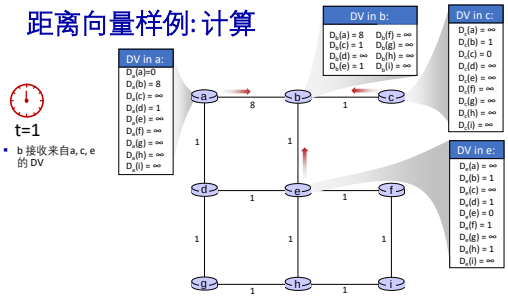
距离向量样例: 迭代

.... 以此类推
接下来让我们看一下节点上的迭代计算

距离向量样例: 计算

t=1

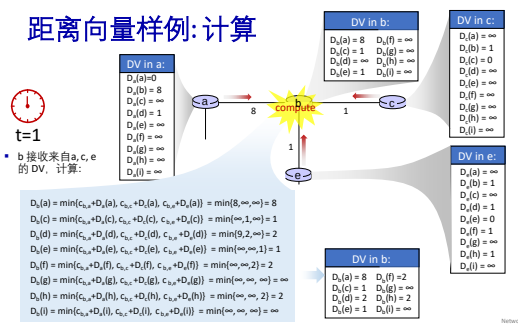
- b 接收来自a, c, e 的 DV



距离向量样例: 计算

t=1

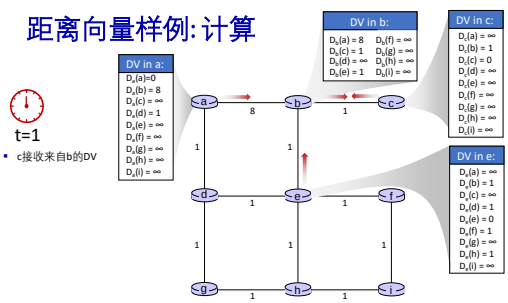
- b 接收来自a, c, e 的 DV，计算:


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:
$D_e(a) = \infty$
$D_e(b) = 1$
$D_e(c) = \infty$
$D_e(d) = 1$
$D_e(e) = 0$
$D_e(f) = 1$
$D_e(g) = \infty$
$D_e(h) = 1$
$D_e(i) = \infty$
compute
a
b
c
d
e
8
1
1
1
$D_b(a) = \min\{c_{b,a}+D_a(a), c_{b,c}+D_c(a), c_{b,e}+D_e(a)\} = \min\{8,\infty,\infty\} = 8$
$D_b(c) = \min\{c_{b,a}+D_a(c), c_{b,c}+D_c(c), c_{b,e}+D_e(c)\} = \min\{\infty,1,\infty\} = 1$
$D_b(d) = \min\{c_{b,a}+D_a(d), c_{b,c}+D_c(d), c_{b,e}+D_e(d)\} = \min\{9,2,\infty\} = 2$
$D_b(e) = \min\{c_{b,a}+D_a(e), c_{b,c}+D_c(e), c_{b,e}+D_e(e)\} = \min\{\infty,\infty,1\} = 1$
$D_b(f) = \min\{c_{b,a}+D_a(f), c_{b,c}+D_c(f), c_{b,e}+D_e(f)\} = \min\{\infty,\infty,2\} = 2$
$D_b(g) = \min\{c_{b,a}+D_a(g), c_{b,c}+D_c(g), c_{b,e}+D_e(g)\} = \min\{\infty,\infty,\infty\} = \infty$
$D_b(h) = \min\{c_{b,a}+D_a(h), c_{b,c}+D_c(h), c_{b,e}+D_e(h)\} = \min\{\infty,\infty,2\} = 2$
$D_b(i) = \min\{c_{b,a}+D_a(i), c_{b,c}+D_c(i), c_{b,e}+D_e(i)\} = \min\{\infty,\infty,\infty\} = \infty$

距离向量样例: 计算

t=1

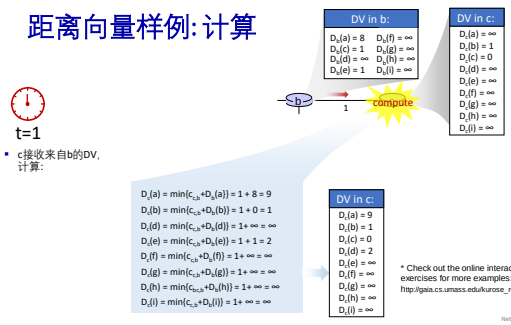
- c接收来自b的DV



距离向量样例: 计算

t=1

- c接收来自b的DV，计算:

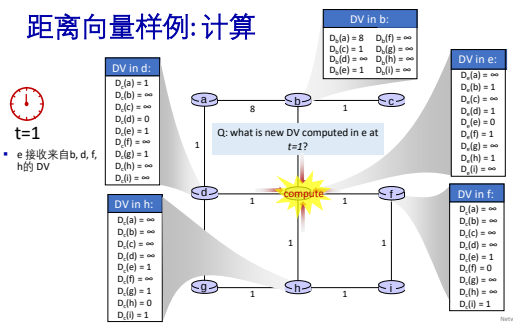


* Check out the online interactive exercises for more examples:
http://gaia.cs.umass.edu/kurose_ross/interactive/

距离向量样例: 计算

t=1

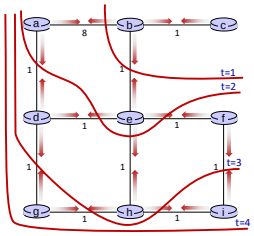
- e 接收来自b, d, f, h的 DV



距离向量: 状态信息扩散

迭代通信，计算步骤通过网络传播信息：

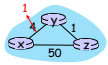
- t=0　t = 0时c的状态仅在c处
- t=1　c在t = 0时的状态已经传播到b，并且可能会影响距离向量计算（最多**1**跳远），即b
- t=2　c在t = 0时的状态已经传播到b，并且可能会影响距离向量计算（最多**2**跳远），即b以及后续的a, e
- t=3　c在t = 0时的状态已经传播到b，并且可能会影响距离向量计算（最多**3**跳远），即b,a,e以及后续的c,f,h
- t=4　c在t = 0时的状态已经传播到b，并且可能会影响距离向量计算（最多**4**跳远），即b,a,e, c, f, h以及后续的g,i



距离向量样例: 链路开销变化

链路开销变化:

- 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



“好消息传得快”

t_0: y节点检测链路开销变化, 更新它的DV,通知邻居.

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, 发送它的 DV给邻居.

t_2: y 收到 z 的更新,更新它的距离表. y的最新的链路开销*没有*变化, 所有 y *没有* 发送信息给z.

距离向量样例: 链路开销变化

链路开销变化:

- 节点检测链路开销变化
- “坏消息传得慢” –无穷计数count-to-infinity问题:
  - y看到与x的直接链路有60的新开销，但是z表示有开销仅5的路径。所以将计算“我通过z到x的新开销是6；通知z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“我通过y到达x的新成本将是7，将z到x的新开销通知y。
  - y得知通过z到达x的路径具有新开销7，因此y计算“我通过y到达x的新成本将是8，将y到x的新开销通知z。
  - z得知通过y到达x的路径具有新开销8，因此z计算“我通过y到达x的新成本将是9，将z到x的新开销通知y。
  - …
- 请参阅文本以获取解决方案。分布式算法很棘手*!*

比较LS(link state)和DV(distance-vector)算法

报文复杂度message complexity

LS: n 路由器, $O(n^2)$ 报文发送

DV: 邻居之间的交互； 收敛时间各不相同

收敛速度speed of convergence

LS: $O(n^2)$ 算法, $O(n^2)$ 报文

- 可能会有振荡

DV: 收敛时间各不相同

- 可能有路由选择环路（routing loops）
- 无穷计数（count-to-infinity） 问题

健壮性robustness: 如果路由器发生故障或受到损害，该怎么办?

LS:

- 路由器可以广播不正确的*链路link* 开销
- 每个路由器仅计算*自己的*表

DV:

- DV路由器可能会宣告不正确的路径*path* 开销 (“我到任何地方的开销都非常低”): 黑洞black-holing
- 每个路由器的表都被其他路由器使用 :错误会通过网络传播

网络层：控制层面的“线路图”

- 介绍
- 路由选择协议Routing protocols
- ISP内部路由选择(routing)：OSPF
- ISP间的路由选择(routing)：BGP
- SDN 控制层
- 互联网控制消息协议ICMP

- 网络管理，配置
  - SNMP
  - NETCONF/YANG

可扩展路由选择

到目前为止，我们的路由研究-已理想化

- 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

互联网的可扩展路由方法

将路由器聚集到自治系统“autonomous systems” (AS) (又称为“域”“domains”)

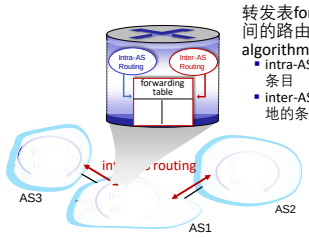
自治系统内intra-AS (也称 域内“intra-domain”): *同一个AS (“network”)* 内的路由

- AS中的所有路由器必须运行相同的域内协议
- 不同AS中的路由器可以运行不同的域内路由协议
- 网关路由器gateway router: 在自身AS的边缘 “edge”, 具有到其他AS路由器router(s) 的链路 link(s)

跨自治系统inter-AS (也称 跨域“inter-domain”): AS*之间*的路由

- 网关gateways 执行域间 inter-domain 路由选择 (以及域内intra-domain 路由选择)

互连的自治系统



转发表forwarding table由AS内与AS间的路由选择算法routing algorithms配置

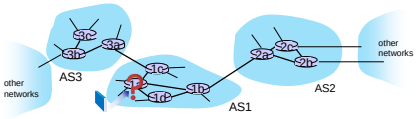
- intra-AS 路由选择确定AS内的目的地条目
- inter-AS 和 intra-AS 共同确定外部目的地的条目

AS间路由选择: 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:
  - 路由器应该将数据包转发到AS1中的网关路由器，但是应该给哪一个呢？？

AS1 域间路由选择必须:

1. 了解通过AS2可以到达哪些目的地，通过AS3可以到达哪些目的地……
2. 将此可达信息告知AS1中的所有路由器



AS间路由选择: routing within an AS

大部分常见自治系统内部路由选择协议intra-AS routing protocols:

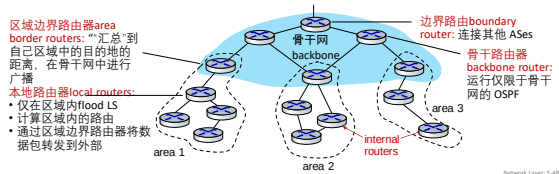
- RIP:路由信息协议[RFC 1723]
  - 经典DV：DV每30秒交换一次
  - 不再广泛使用
- EIGRP:增强的内部网关路由协议
  - 基于DV
  - 以前为思科专有，已有数十年的历史（于2013年开放[RFC 7868])
- OSPF:开放最短路径优先[RFC 2328]
  - link-state routing链接状态路由选择
  - IS-IS 协议 (ISO 标准, 不是 RFC 标准)与OSPF基本相同

OSPF (Open Shortest Path First) routing

- 开放“open”: 公开可用
- classic link-state经典链接状态
  - 每个路由器将OSPF链接状态通告（直接通过IP而不是使用TCP / UDP）泛洪到整个AS中的所有其他路由器
  - 可能有多个链路成本指标：带宽，延迟
  - 每个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*: 所有 OSPF 报文均已通过身份验证（以防止恶意入侵）

分层 OSPF

- 两级层次结构two-level hierarchy: 本地，骨干网.
  - 链路状态link-state 广播仅在区域或骨干网中泛滥
  - 每个节点都有详细的区域拓扑；只知道到达其他目的地的方向



网络层：控制层面的“线路图”

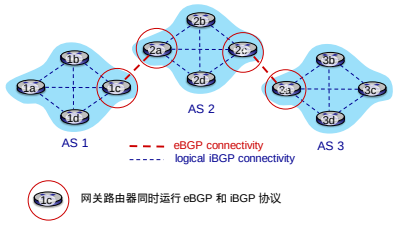
- 介绍
- 路由选择协议Routing protocols
- ISP内部路由选择(routing)：OSPF
- ISP间的路由选择(routing)：BGP
- SDN 控制层
- 互联网控制消息协议ICMP

- 网络管理，配置
  - SNMP
  - NETCONF/YANG

Internet 跨AS 路由选择: BGP

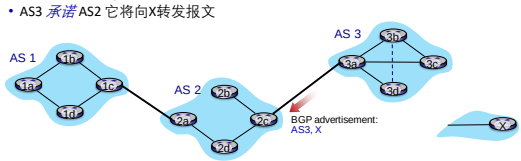
- BGP (边界网关协议Border Gateway Protocol): 实际上的域间路由协议
  - “将互联网连接在一起的胶水”
- 允许子网向Internet的其余部分宣传其存在以及它可以到达的目的地: *“我在这里，这是我可以到达的位置，以及到达的方式”*
- BGP 为每个AS提供了一种手段来:
  - 外部BGP（eBGP）: 从相邻AS获取子网可达性信息
  - 内部BGP（iBGP）:将可达性信息传播到所有AS内部路由器。
  - 根据可达性信息和*策略*确定到其他网络的“良好”路由

eBGP, iBGP 连接



BGP 基础

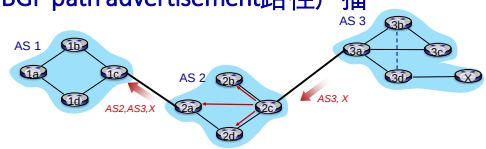
- BGP 会话: 两个BGP路由器（“对等” peers ）通过半永久TCP连接交换BGP消息:
  - 广播路径 *paths* 到不同的目标网络前缀（BGP是“路径向量”协议）
- 当AS3网关3a将path（路径）AS3,X 到 AS2 告诉网关 2c:
  - AS3 *承诺* AS2 它将向X转发报文



路径属性and BGP 路由

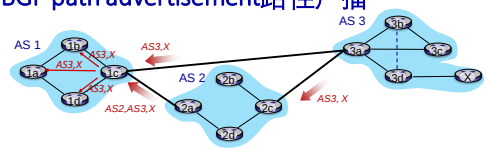
- BGP 发布的路由：前缀prefix + 属性attributes
  - prefix: 广播目的地
  - 两个重要属性:
    - AS-PATH: 前缀广播已通过的AS列表
    - NEXT-HOP: 表示到下一跳自治系统的特定内部自治系统路由器
- policy-based routing基于策略的路由:
  - 接收路由广播的网关使用*导入策略import policy* 接受/拒绝路径（例如，从不通过AS Y路由）
  - 策略还确定是否向其他相邻AS*通告*路径

BGP path advertisement路径广播



- AS2路由器2c从AS3路由器3a接收路径广播AS3，X（通过eBGP）
- 根据AS2策略，AS2路由器2c接受路径AS3，X，（通过iBGP）传播到所有AS2路由器
- 根据AS2策略，AS2路由器2a（通过eBGP）将路径AS2，AS3，X广播给AS1路由器1c

BGP path advertisement路径广播



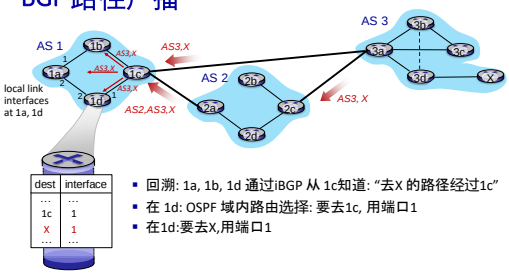
网关路由器可能会了解到目标的多个路径:

- AS1 网关路由器gateway router 1c从2a学习路径*AS2,AS3,X*
- AS1网关路由器gateway router 1c 从学习路径*AS3,X*
- 根据 *策略policy,* AS1网关路由器1c选择路径*AS3,X* ，并通过iBGP在AS1内发布路径

BGP 报文

- 过TCP连接在对等体之间交换BGP消息
- BGP 报文messages:
  - OPEN: 打开与远程BGP对等体的TCP连接并验证发送BGP对等体peer
  - UPDATE: 广播新路径（或撤消旧路径）
  - KEEPALIVE: 在没有更新的情况下使连接保持活动状态； 也确认打开请求
  - NOTIFICATION: 报告以前的报文msg中的错误； 也用于关闭连接

BGP 路径广播



- 回溯: 1a, 1b, 1d 通过iBGP 从 1c知道: “去X 的路径经过1c”
- 在 1d: OSPF 域内路由选择: 要去1c, 用端口1
- 在1d:要去X,用端口1

BGP 路径广播

- 回溯: 1a, 1b, 1d 通过iBGP 从 1c知道: “去X 的路径经过1c”
- 在 1d: OSPF 域内路由选择: 要去1c, 用端口1
- 在1d:要去X,用端口1
- 在1a: OSPF 域内路由选择: 要去1c, 用端口2
- 在1a:要去X,用端口2

为什么域内域间AS路由选择不同?

Policy策略:

- 跨AS：管理员希望控制其流量的路由方式，谁通过其网络进行路由
- 自治系统内：单一管理员，因此策略上没有什么问题

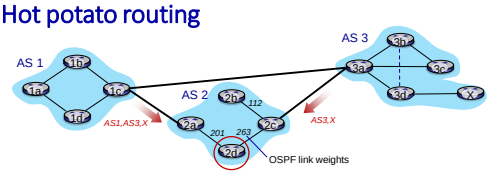
Scale规模:

- 分层路由节省了表的大小，减少了更新流量

Performance表现:

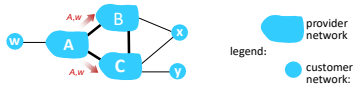
- 内部AS：注重性能
- 跨AS：策略（优先）高于性能

Hot potato routing



- 2d learns (via iBGP) 它可以通过2a或2c路由到X
- hot potato routing: 选择域内成本最低的本地网关（例如，即使有更多AS跳到X，2d选择2a）: 不必担心域间成本！

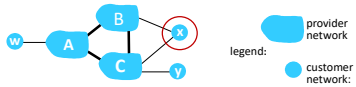
BGP:通过广播实现策略



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–一种典型的"现实世界"策略）

- A 广播路径 Aw 到 B 和C
- B *选择不广播*C BAw!
  - B 没有从路由CBAw获得"收入revenue", 因为C, A, w 都不是 B的客户
  - C 不知道路径CBAw
- C 将路由CAw (不使用B) 来到达w

BGP:通过广播实现策略



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–一种典型的"现实世界"策略）

- A,B,C 是 provider networks提供商网络
- x,w,y 是(提供商网络的) customer客户
- x 是 dual-homed双宿主的: 连接到两个网络
- *policy to enforce强制执行的策略:* 不想通过x从B路由到C
  - .. 因此x不会向B广播通往C的路线

BGP 路由选择

- 路由器可能会了解到到目的地AS的多条路由，并根据以下条件选择路由：
  1. 本地偏好值属性：策略决策
  2. 最短的AS-PATH
  3. 最近的NEXT-HOP路由器: hot potato routing
  4. 附加标准

网络层：控制层面的"线路图"

- 介绍
- 路由选择协议Routing protocols
- ISP内部路由选择（routing）: OSPF
- ISP间的路由选择（routing）: BGP
- SDN 控制层
- 互联网控制消息协议ICMP

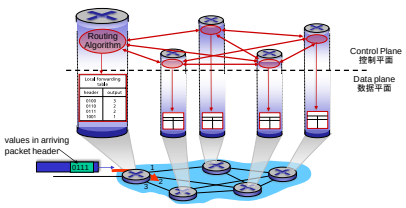
- 网络管理，配置
  - SNMP
  - NETCONF/YANG

软件定义网络(SDN)

- 互联网网络层：历史上是通过分布式的，每路由器的控制方法来实现的：
  - *monolithic* 单片路由器包含交换硬件，在专有路由器OS（例如Cisco IOS）中运行Internet标准协议（IP，RIP，IS-IS，OSPF，BGP）的专有实现
  - 不同的 "middleboxes中间箱"用于不同网络层功能: 防火墙, load balancers负载均衡, NAT boxes, ..
- ~2005:重新思考网络控制层面

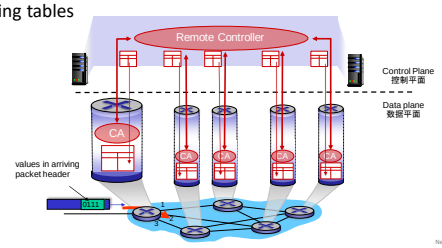
每路由器控制平面Per-router control plane

在控制平面中 *每个路由器中的*各个路由选择算法组件相互作用



软件定义网络(SDN) control plane

远程控制器Remote controller计算，在路由器中安装转发表forwarding tables

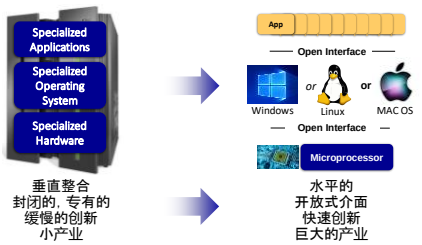


软件定义网络(SDN)

*为什么 是*一个 *logically centralized逻辑上集中的*控制层？

- 简化网络管理：避免路由器配置错误，流量流flow有更大的灵活性
- 基于表的转发(调用 OpenFlow API)允许对路由器进行“编程”
  - 集中的“编程”更加容易：集中计算表并分发
  - 分布式“编程”更加困难：计算表是在每个路由器中实施的分布式
- 开放（非专有）控制平面实施
  - 促进创新：百花盛开

SDN analogy:大型机引领PC革命

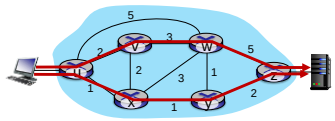


垂直整合
封闭的，专有的
缓慢的创新
小产业

水平的
开放式介面
快速创新
巨大的产业

* Slide courtesy: N. McKeown

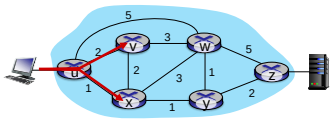
流量工程：传统路由选择困难



问：如果网络运营商希望u-z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，以便流量路由算法相应地计算路由（或需要新的路由算法）！

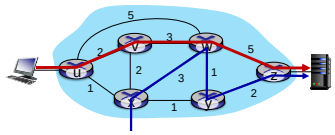
链路权重仅是控制“旋钮”：没有太多控制权！

流量工程：传统路由选择困难



问：如果网络运营商想要沿uvwz和uxyz（负载平衡）分配u-z流量怎么办？
答：无法做到（或需要新的路由算法）

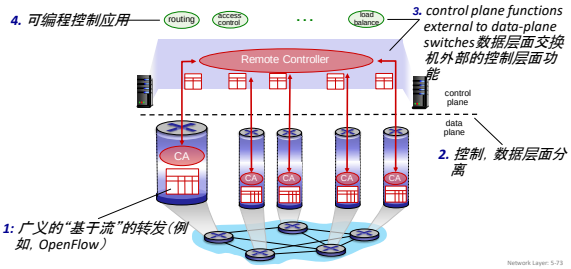
流量工程：传统路由选择困难



问：如果w希望将蓝色和红色流量从w路由到z，该怎么办？
答：做不到（使用基于目标的转发以及LS，DV路由）

我们在第4章了解到，通用转发和SDN可用于实现所需的任何路由

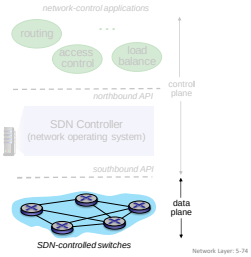
软件定义网络(SDN)



软件定义网络(SDN)

数据平面交换:

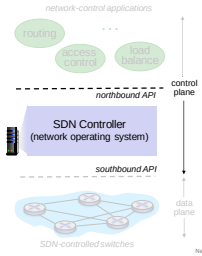
- 快速，简单的商用交换机，在硬件中实现了通用数据平面转发（第4.4节）
- 计算流量（转发）表，在控制器的监督下安装
- 用于基于表的开关控制（例如OpenFlow）的API
  - 定义什么是可控制的，什么不是可控制的与控制器通信的
- 协议（例如，OpenFlow））



软件定义网络(SDN)

SDN 控制器 (网络操作系统):

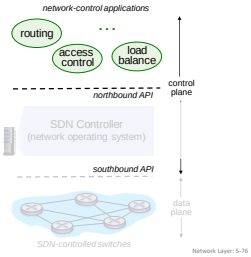
- 维护网络状态信息
- 通过北面的API于"上方"的网络控制应用程序进行交互
- 通过南面的API于"下方"的网络控制应用程序进行交互
- 采用分布式系统，以实现性能，可伸缩性，容错性和鲁棒性



软件定义网络(SDN)

网络控制（network-contro）l应用程序:

- 控制的"大脑"：使用SDN控制器提供的较低层的服、API来实现控制功能
- *取消捆绑unbundled:* 可以由第三方提供：与路由供应商或SDN控制器不同

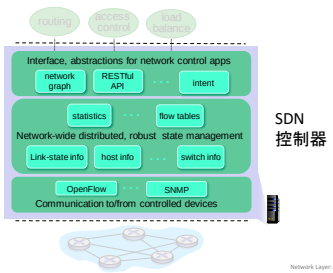


SDN控制器的组件

网络控制应用程序的接口层: 抽象 API

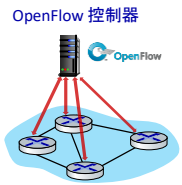
全网（network-wide）状态 管理：网络链路，交换机，服务的状态: *分布式数据库 distributed database*

通信communication: SDN控制器与受控交换机之间的通信



OpenFlow 协议

- 在控制器（controller），交换机（switch）之间操作，
- TCP 用于交换报文
  - 可选加密
- 三类OpenFlow 报文:
  - 控制器到交换机controller-to-switch
  - 异步asynchronous (交换机到控制器 switch to controller)
  - 对称的symmetric (杂项)
- 与 OpenFlow API不同
  - API 用于指定广义转发操作

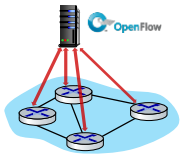


OpenFlow:控制器到交换机的报文

关键 controller-to-switch 报文

- *功能features:* 控制器查询交换机功能，交换机回复
- *配置configure:* 控制器查询/设置交换机配置参数
- *修正状态modify-state:* OpenFlow表中添加，删除，修改流条目
- *数据包输出packet-out:* 控制器可以将此数据包从特定的交换机端口发送出去

OpenFlow 控制器

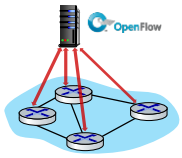


OpenFlow:控制器到交换机的报文

关键 controller-to-switch 报文

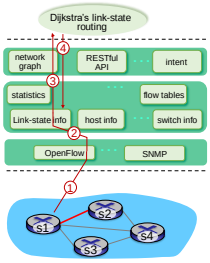
- *数据包输入packet-in:* 将数据包（及其控制）传输到控制器。查看来自控制器的数据包输出报文
- *已删除的流flow-removed:* flow表的条目已在交换机上删除
- *端口状态port status:* 通知控制器端口更改

OpenFlow Controller



幸运的是，网络运营商不会通过直接创建/发送OpenFlow报文来对交换机进行“编程”，而是使用控制器上更高级别的抽象来代替

SDN: 控制/数据层面相互作用示例



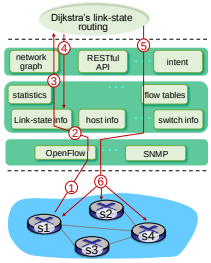
① S1遇到链路故障，使用OpenFlow端口状态报文通知控制器

② SDN控制器收到OpenFlow报文，更新链路状态信息

③ Dijkstra路由选择算法应用程序先前已注册为每次链路状态发生变化时都会被调用。该算法应用程序被调用。

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

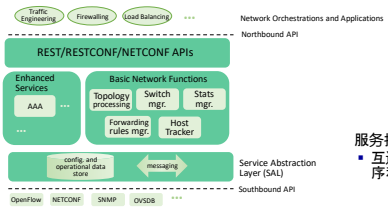
SDN: 控制/数据层面相互作用示例



⑤ 链路状态路由应用程序与SDN控制器中的流表计算（flow-table-computation）组件进行交互，该组件计算所需的新流表

⑥ 控制器使用OpenFlow在需要更新的交换机中安装新表

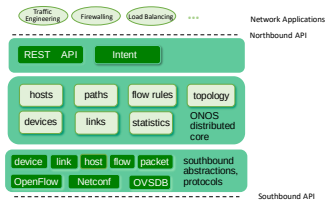
OpenDaylight (ODL) 控制器



服务抽象层:

- 互连内部，外部应用程序和服务

ONOS 控制器



- 控制应用程序与控制器分开
- 意图框架：服务的高级规范：什么而不是如何
- 相当重视分布式核心：服务可靠性，复制性能扩展

SDN: 选择的挑战

- 强化控制平面：可靠，可靠，性能可扩展，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用于控制平面
  - 可靠性和安全性：从第一天开始"融入"?
- 网络，符合任务特定要求的协议
  - 例如，实时，超可靠，超安全
- 互联网扩展：超越单个AS
- SDN在5G蜂窝网络中至关重要

SDN和传统网络协议的未来

- SDN-computed versus router-computer forwarding tables:
  - 只是logically-centralized-computed 逻辑集中计算与协议计算的一个示例
- 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

网络功能（SDN与协议）的实现将如何发展?

网络层：控制层面的"线路图"

- 介绍
- 路由选择协议Routing protocols
- ISP内部路由选择(routing)：OSPF
- ISP间的路由选择(routing)：BGP
- SDN 控制层
- 互联网控制消息协议ICMP

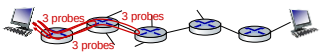
- 网络管理，配置
  - SNMP
  - NETCONF/YANG

ICMP:互联网控制消息协议

- 由主机和路由器用于通信网络级信息
  - 错误报告：无法访问的主机，网络，端口，协议
  - 回显请求/回复（用于ping）
- 网络层"之上"IP：
  - 报文中携带的ICMP消息
- *ICMP 报文*: 类型，代码以及导致错误的IP数据报的前8个字节

Type	Code	description
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

Traceroute and ICMP



- 源将UDP分网段发送到目标
  - 1st 设置为TTL =1, 2nd设置为TTL=2, etc.
- 第n个集合中的数据报到达第n个路由器：
  - 路由器丢弃数据报并发送源ICMP消息（类型11，代码0）
  - ICMP报文可能包含路由器名称和IP地址
- 当ICMP消息到达源时：记录RTT

停止条件:
- UDP段最终到达目标主机
- 目的地返回ICMP"端口不可达"消息（类型3，代码3）
- 源停止

网络层：控制层面的"线路图"

- 介绍
- 路由选择协议Routing protocols
- ISP内部路由选择(routing)：OSPF
- ISP间的路由选择(routing)：BGP
- SDN 控制层
- 互联网控制消息协议ICMP

- 网络管理，配置
  - SNMP
  - NETCONF/YANG

什么是网络管理?

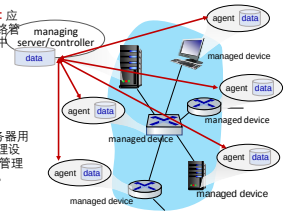
- 自治系统autonomous systems (aka “network”): 数千个交互的硬件/软件组件
- 其他需要监视，配置，控制的复杂系统:
  - 喷气飞机，核电站，其他?

"Network management包括硬件，软件和人工元素的部署，集成和协调，以监视，测试，轮询，配置，分析，评估和控制网络和元素资源，以满足实时，运营性能和服务质量 要求，且费用合理。”

网络管理的组成部分

Managing server: 应用程序，通常是网络管理员（人）在回路中

Network management protocol: 管理服务器用来查询，配置，管理设备；设备用于通知管理服务器数据，事件。



Managed device: 具有可管理，可配置硬件，软件组件的设备

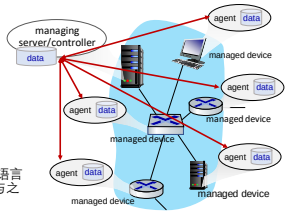
Data: 设备“状态”配置数据，操作数据，设备统计信息

网络运营商的管理方法

CLI (Command Line Interface命令行界面)
- 操作员发出（类型，脚本）直接针对各个设备（例如vis ssh）

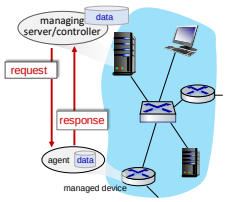
SNMP/MIB
- 操作员使用简单网络管理协议（SNMP）查询/设置设备数据（MIB）

NETCONF/YANG
- 更抽象，网络范围内的整体
- 强调多设备配置管理。
- YANG: data modeling language数据建模语言
- NETCONF:向/从远程设备/向远程设备/与之通信YANG兼容的动作/数据

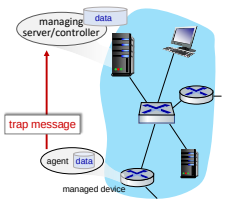


SNMP 协议

传递MIB信息，命令的两种方法:



request/response mode

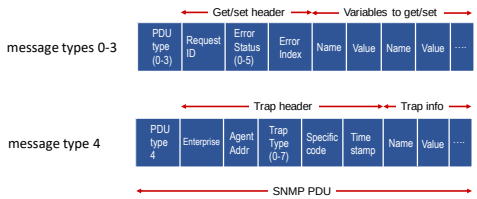


trap mode

SNMP 协议: 报文类型

Message type	Function
GetRequest GetNextRequest GetBulkRequest	manager-to-agent: “get me data” (数据实例, 列表中的下一个数据, 数据块).
SetRequest	manager-to-agent: 设置 MIB值
Response	Agent-to-manager: 值, 响应请求
Trap	Agent-to-manager: 通知异常事件

SNMP 协议: 报文格式



SNMP:管理信息库(MIB)

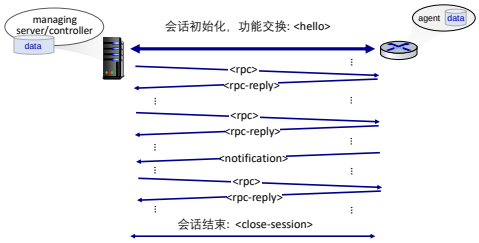
- 托管设备的操作（和某些配置）数据
- 收集到设备MIB 模块
  - RFC定义了400 MIB 模块;更多供应商特定的MIBs
- 管理信息结构(SMI): data definition language数据定义语言
- UDP 协议的示例MIB 变量:

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

NETCONF 总述

- 目标: 在整个网络范围内主动管理/配置设备
- 在管理服务器和托管网络设备之间运行
  - 操作：检索，设置，修改，激活配置
  - atomic-commit原子提交操作在多个设备上执行
  - 查询运营数据和统计数据
  - 订阅来自设备的通知
- 远程过程调用（RPC）范例
  - 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

NETCONF初始化，交换，关闭



选择 NETCONF 操作

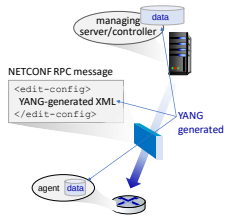
NETCONF	Operation Description
<get-config> <get>	检索给定配置的全部或部分。 一个设备可能具有多种配置。检索全部或部分配置状态和操作状态数据。
<edit-config>	在托管设备上更改指定的（可能正在运行）配置。 托管设备的<rpc-reply> 包含带有回滚的 <ok> or <rpcerror> <lock>, <unlock> 锁定（解锁）托管设备上的配置数据存储（以锁定其他来源的NETCONF, SNMP, or CLIs 的命令）
<create-subscription>,	从托管设备启用事件通知订阅<notification>

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/
         1.2/config">
10         <interface>
11           <name>Ethernet0/0</name>  更改 Ethernet 0/0 接口的MTU 为1500
12           <mtu>1500</mtu>
13         </interface>
14       </top>
15     </config>
16   </edit-config>
17 </rpc>
```


YANG

- 数据建模语言，用于指定NETCONF网络管理数据的结构，语法，语义
  - 内置数据类型built-in data types, 例如 SMI
- 文档描述设备，可以从YANG描述中生成功能
- 可以表达有效NETCONF配置必须满足的数据之间的约束
  - 确保NETCONF配置满足正确性，一致性约束



网络层: 总结

我们学到了很多!

- 网络控制层面的方法
  - 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 控制器
  - 实现: ODL, ONOS
- Internet Control Message Protocol
- 网络管理

下一站: 链路层!

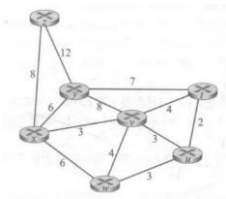
网络层控制层面:完成！

- 介绍
- 路由选择协议Routing protocols
  - 链路状态Link State
  - 距离向量Distance-Vector
- ISP内部路由选择(routing): OSPF
- ISP间的路由选择(routing) : BGP
- SDN 控制层
- 互联网控制消息协议ICMP

- 网络管理，配置
  - SNMP
  - NETCONF/YANG

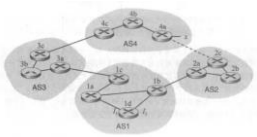
作业

1. 考虑下面的网络。对于标明的链路开销，用Dijkstra的最短路算法计算出从x到所有网络节点的最短路径。通过计算一个类似于表5-1的表，说明该算法是如何工作的。



作业

2. 考虑下图所示的网络。假定AS3和AS2正在运行OSPF作为其AS内部路由选择协议。假定AS1和AS4正在运行RIP作为其AS内部路由选择协议。假定AS间路由选择协议使用的是eBGP和iBGP。假定最初在AS2和AS4之间不存在物理链路。
   a. 路由器3c从下列哪个路由选择协议学习到了前缀x: OSPF、RIP, eBGP或iBGP?
   b. 路由器3a从哪个路由选择协议学习到了前缀x?
   c. 路由器1c从哪个路由选择协议学习到了前缀x?
   d. 路由器1d从哪个路由选择协议学习到了前缀x?



作业

3. 在图5-13中，假定有另一个桩网络V，它为ISP A的客户。假设B和C具有对等关系，并且A是B和C的客户。假设A希望让发向W的流量仅来自B，并且发向V的流量来自B或C。A如何向B和C通告其路由？C收到什么样的AS路由？

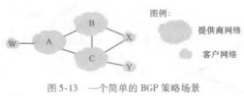


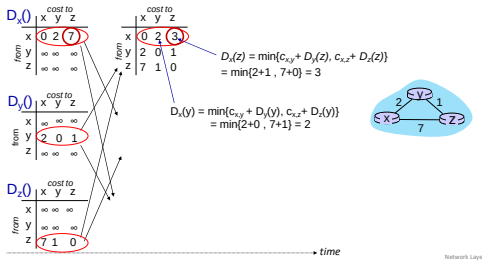
图 5-13 一个简单的 BGP 策略场景

作业

4. 比较和对照集中式和分布式路由选择算法的性质。给出一个路由选择协议的例子，该路由选择协议采用分布式方法和集中式方法。
5. 为什么在因特网中用到了不同的AS间与AS内部协议?
6. 假定你要在SDN控制平面中实现一个新型路由选择协议。你将在哪个层次中实现该协议？解释理由。
7. 在发送主机执行Traceroute程序，收到哪两种类型的ICMP报文?

Chapter 5 相关材料

距离向量: 另一个样例



距离向量: 另一个样例