Computer Networking

谢 逸
中山大学 • 计算机学院
2024. Fall

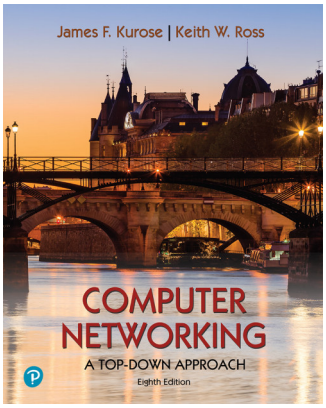
Chapter 5 Network Layer: Control Plane

A note on the use of these PowerPoint slides:
We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

For a revision history, see the slide note for this page.

Thanks and enjoy! JFK/KWR



Computer Networking: A Top-Down Approach
8th edition
Jim Kurose, Keith Ross
Pearson, 2020

Assignments :

- Ch5 (ver 8, cn): 3, 7, 11, 14, 16, 19
- A team project

- Keywords: Forwarding and Routing, Virtual Circuit and Datagram Networks, Addressing, DHCP, NAT, ICMP, Routing Algorithms (DV, LS), RIP, OSPF, BGP

Chapter 5: network layer control plane

chapter goals: understand principles behind network control plane

- traditional routing algorithms
- SDN controllers
- Internet Control Message Protocol
- network management

and their instantiation, implementation in the Internet:

- OSPF, BGP, OpenFlow, ODL and ONOS controllers, ICMP, SNMP

Chapter 5: outline

5.1 introduction

5.2 routing protocols
- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

Network-layer functions

Recall: two network-layer functions:

data plane

Given a forwarding table

- ***forwarding:*** **move packets from router's input to appropriate router output**

control plane

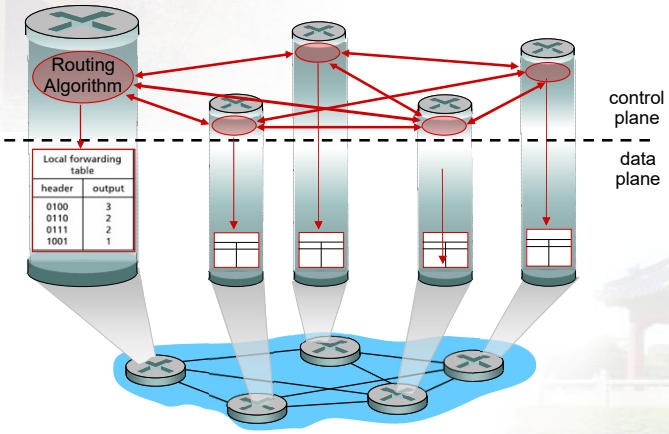
Create a routing table for forwarding

- ***routing***: determine route taken by packets from source to destination

Two approaches to structuring network control plane:
- per-router control (traditional)
- logically centralized control (software defined networking)

Per-router control plane

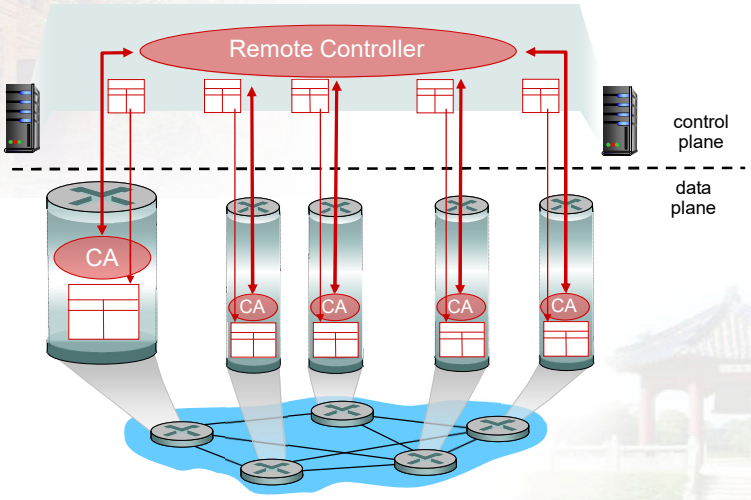
Individual routing algorithm components *in each and every router* interact with each other in control plane to compute forwarding tables



5-7

Logically centralized control plane

A distinct (typically remote) controller interacts with local control agents (CAs) in routers to compute forwarding tables



5-8

Chapter 5: outline

5.1 introduction

5.2 routing protocols
- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

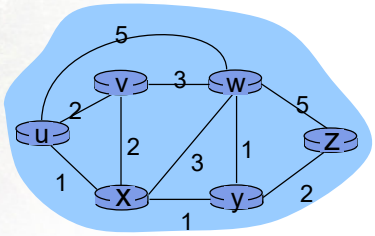
5.7 Network management and SNMP

Routing protocols

Routing protocol goal: determine "good" paths (equivalently, routes), from sending hosts to receiving host, through network of routers

- path: sequence of routers packets will traverse in going from given initial source host to given final destination host
- "good": least "cost", "fastest", "least congested"
- routing: a "top-10" networking challenge!

Graph abstraction



graph: G = (N,E)

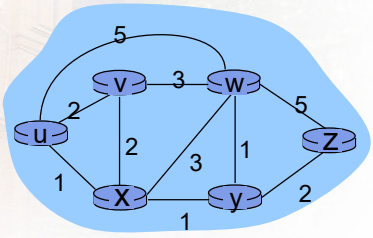
N = set of routers = { u, v, w, x, y, z }

E = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

aside: graph abstraction is useful in other network contexts, e.g., P2P, where *N* is set of peers and *E* is set of TCP connections

4-11

Graph abstraction: costs



c(x,x') = cost of link (x,x')
e.g., c(w,z) = 5

cost could always be 1, or inversely related to bandwidth, or inversely related to congestion

cost of path $(x_1, x_2, x_3, \ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

key question: what is the least-cost path between u and z ?
routing algorithm: algorithm that finds that least cost path

4-12

Routing algorithm classification

global: all routers have *complete* topology, link cost info
- "link state" algorithms

How fast do routes change?

static: routes change slowly over time

dynamic: routes change more quickly
- periodic updates or in response to link cost changes

global or decentralized information?

decentralized: iterative process of computation, exchange of info with neighbors
- routers initially only know link costs to attached neighbors
- "distance vector" algorithms

Chapter 5: outline

5.1 introduction

5.2 routing protocols
- **link state**
- **distance vector**

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

A Link-State Routing Algorithm

Dijkstra's algorithm

- net topology, link costs known to all nodes
  - accomplished via "link state broadcast"
  - all nodes have same info
- computes least cost paths from one node ('source") to all other nodes
  - gives *forwarding table* for that node
- iterative: after k iterations, know least cost path to k dest.'s

notation:

- **c(x,y):** link cost from node x to y; = ∞ if not direct neighbors
- **D(v):** current value of cost of path from source to dest. v
- **p(v):** predecessor node along path from source to v
- **N':** set of nodes whose least cost path definitively known

4-15

Dijsktra's Algorithm

```
1  Initialization:
2    N' = {u}
3    for all nodes v
4      if v adjacent to u
5          then D(v) = c(u,v)
6      else D(v) = ∞
7
8  Loop
9    find w not in N' such that D(w) is a minimum
10   add w to N'
11   update D(v) for all v adjacent to w and not in N' :
12      D(v) = min( D(v), D(w) + c(w,v) )
13   /* new cost to v is either old cost to v or known
14    shortest path cost to w plus cost from w to v */
15 until all nodes in N'
```

4-16

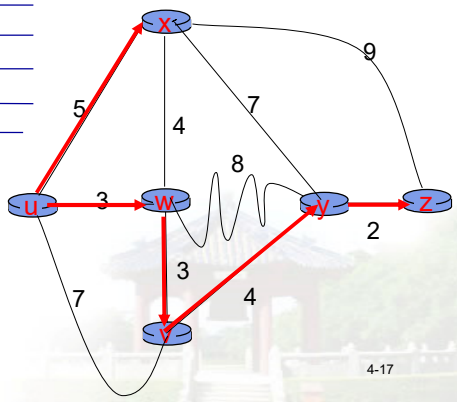
Dijkstra's algorithm: example

Find the least cost paths between u and other nodes

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					

notes:

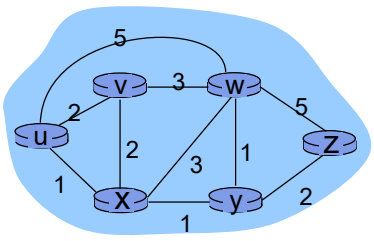
- construct shortest path tree by tracing predecessor nodes
- ties can exist (can be broken arbitrarily)

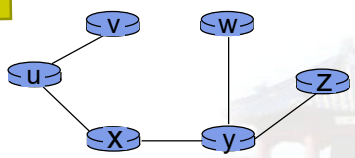


4-17

Dijkstra's algorithm: another example

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					



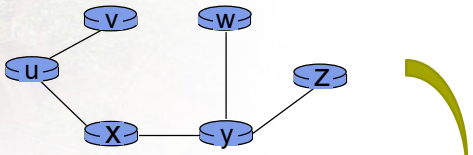


20241117

4-18

Dijkstra's algorithm: example (2)

resulting shortest-path tree from u:



resulting forwarding table in u:

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

4-19

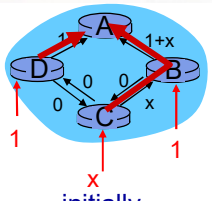
Dijkstra's algorithm, discussion

algorithm complexity: **n nodes**

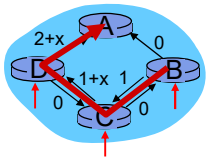
- **each iteration: need to check all nodes, w, not in N**
- **n(n+1)/2 comparisons: $O(n^2)$**
- **more efficient implementations possible: O(nlogn)**

oscillations possible:

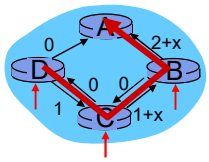
- **e.g., support link cost equals amount of carried traffic:**



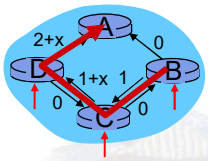
initially



given these costs, find new routing…. resulting in new costs



given these costs, find new routing…. resulting in new costs



given these costs, find new routing…. resulting in new costs

4-20

Chapter 5: outline

5.1 introduction

5.2 routing protocols
- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

Distance vector algorithm

- 邻居能去到的，自己一定能通过邻居到达；
- 多个邻居能到的，找一个总代价最小的邻居作为下一跳；

Bellman-Ford equation (dynamic programming)

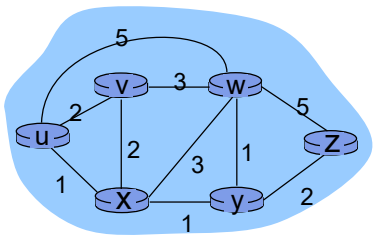
let $d_x(y)$:= cost of least-cost path from x to y

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

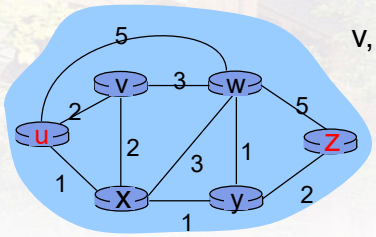
cost from neighbor v to destination y

cost to **neighbor** v

min taken over all neighbors v of x



Bellman-Ford example



v, x, w are u's neighbors, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

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

node achieving minimum is next hop in shortest path, used in forwarding table

4-23

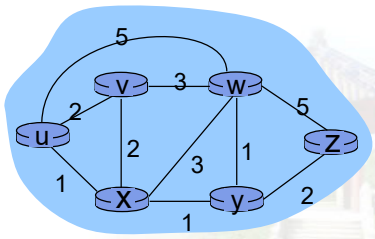
Distance vector algorithm

- $d_x(y)$ = estimate of least cost from x to y

  $d_x(y) = \min_v \{c(x,v) + d_v(y)\}$
  - x maintains distance vector $d_x = [d_x(y): y \in N]$
- node x:
  - knows cost to each neighbor v by detecting: c(x,v)
  - maintains its neighbors' distance vectors. For each neighbor v, x maintains

    $d_v = [d_v(y): y \in N]$

How to obtain $d_v(y)$, $y \in N$?



4-24

Distance vector algorithm

key idea:

- **from time-to-time, each node sends its own distance vector estimate to neighbors**
- **when x receives new DV estimate from neighbor, it updates its own DV using B-F equation:**

$$D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\} \quad \text{for each node } y \in N$$

- under minor, natural conditions, the estimate $D_x(y)$ *converge to the actual least cost* $d_x(y)$

Distance vector algorithm

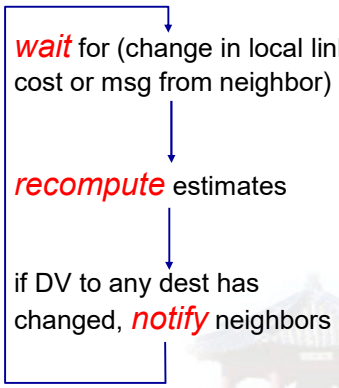
iterative, asynchronous: **each local iteration caused by:**

- **local link cost change**
- **DV update message from neighbor**

distributed:

- **each node notifies neighbors *only* when its DV changes**
  - **neighbors then notify their neighbors if necessary**

each node:

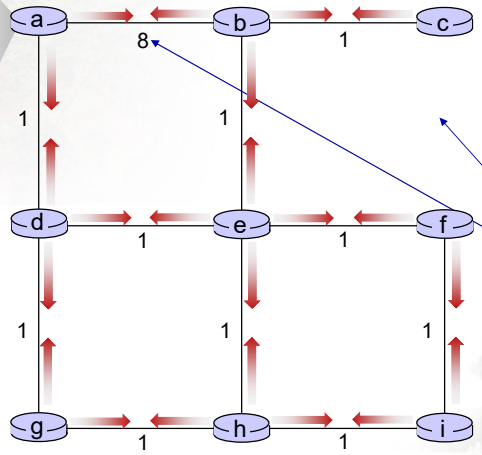


Distance vector: example

t=0

- All nodes have distance estimates to nearest neighbors (only)
- All nodes send their local distance vector to their neighbors

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

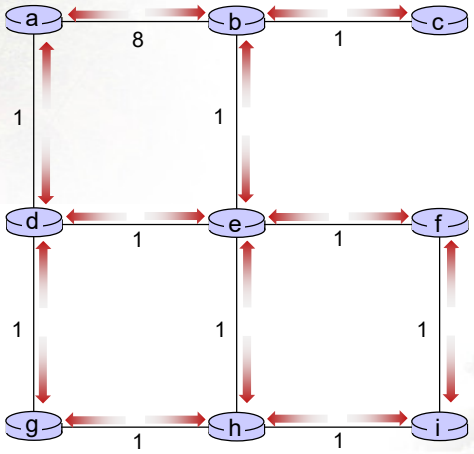


A few asymmetries:
- missing link
- larger cost

Distance vector example: iteration

t=1

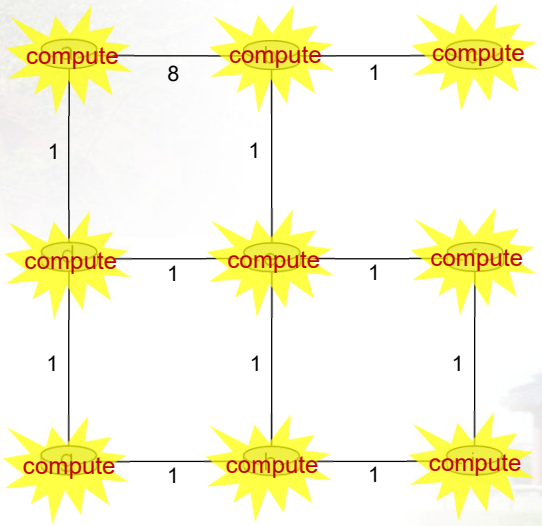
All nodes:
- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



Distance vector example: iteration

t=1

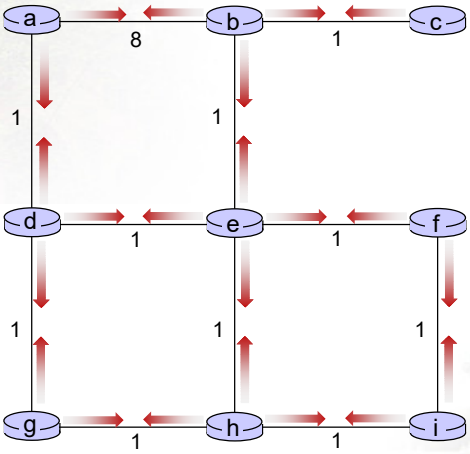
All nodes:
- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



Distance vector example: iteration

t=1

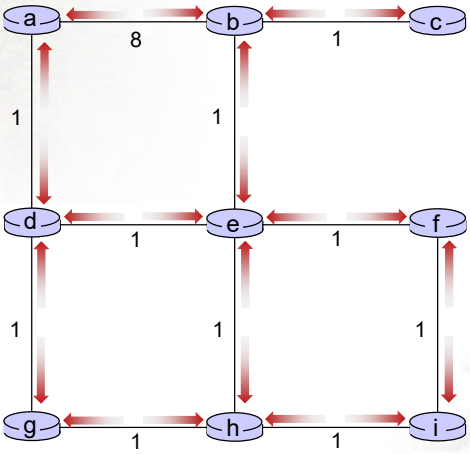
All nodes:
- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



Distance vector example: iteration

t=2

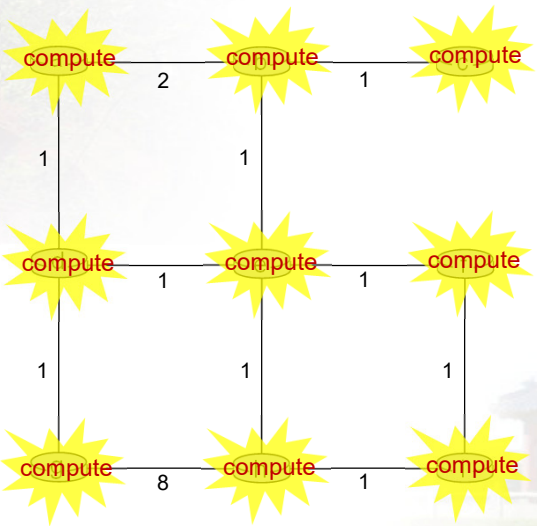
All nodes:
- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



Distance vector example: iteration

t=2

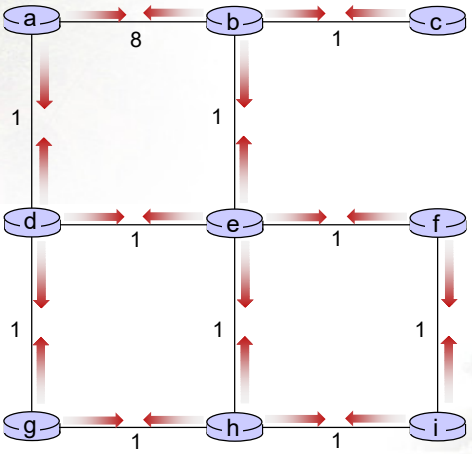
All nodes:
- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



Distance vector example: iteration

t=2

All nodes:
- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors

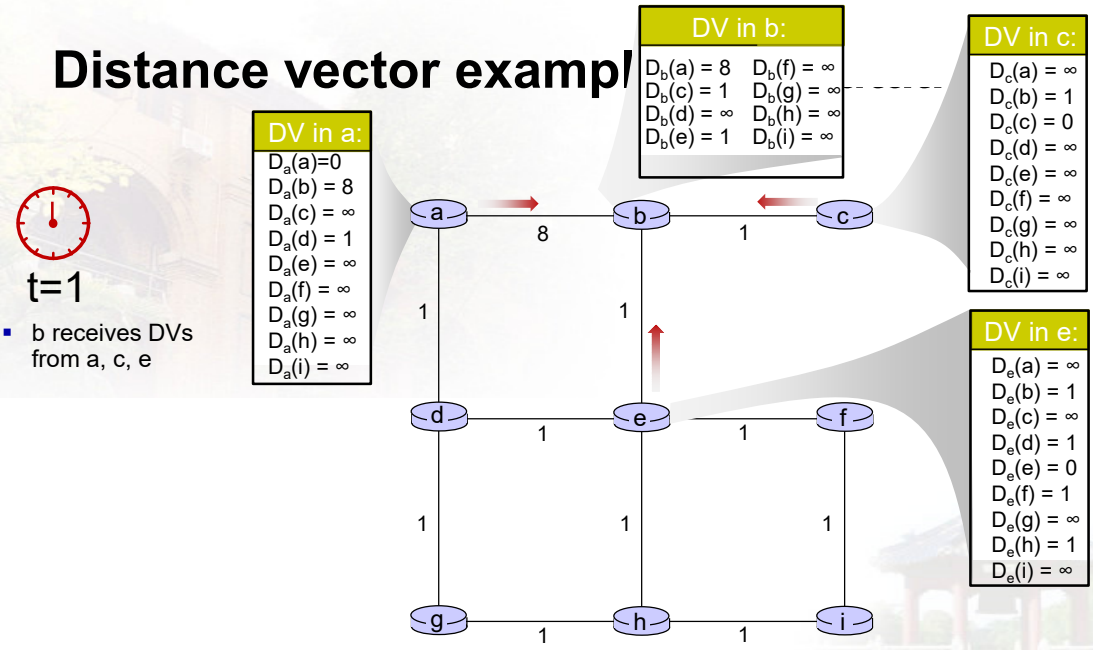


Distance vector example: iteration

…. and so on

Let's next take a look at the iterative *computations* at nodes

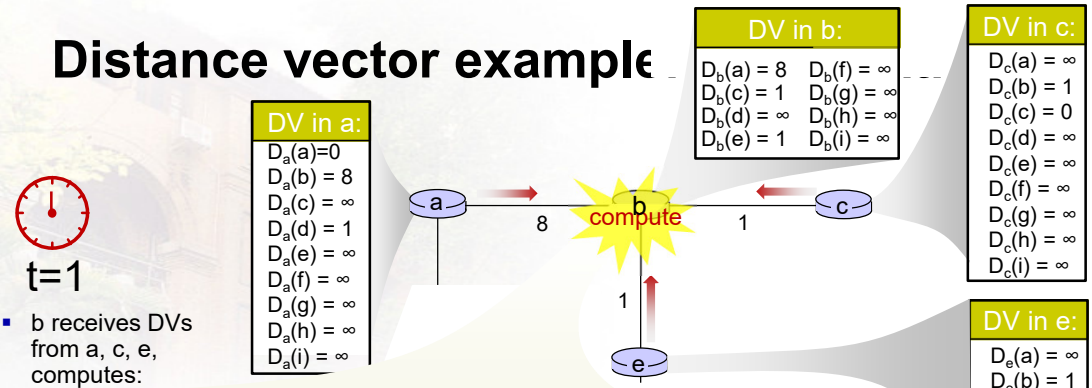
Distance vector exampl



t=1

- b receives DVs from a, c, e

Distance vector example



t=1

- b receives DVs from a, c, e, computes:

$D_b(a) = \min\{c_{b,a}+D_a(a), c_{b,c}+D_c(a), c_{b,e}+D_e(a)\} = \min\{8,\infty,\infty\} = 8$

$D_b(c) = \min\{c_{b,a}+D_a(c), c_{b,c}+D_c(c), c_{b,e}+D_e(c)\} = \min\{\infty,1,\infty\} = 1$

$D_b(d) = \min\{c_{b,a}+D_a(d), c_{b,c}+D_c(d), c_{b,e}+D_e(d)\} = \min\{9,2,\infty\} = 2$

$D_b(e) = \min\{c_{b,a}+D_a(e), c_{b,c}+D_c(e), c_{b,e}+D_e(e)\} = \min\{\infty,\infty,1\} = 1$

$D_b(f) = \min\{c_{b,a}+D_a(f), c_{b,c}+D_c(f), c_{b,e}+D_e(f)\} = \min\{\infty,\infty,2\} = 2$

$D_b(g) = \min\{c_{b,a}+D_a(g), c_{b,c}+D_c(g), c_{b,e}+D_e(g)\} = \min\{\infty,\infty,\infty\} = \infty$

$D_b(h) = \min\{c_{b,a}+D_a(h), c_{b,c}+D_c(h), c_{b,e}+D_e(h)\} = \min\{\infty,\infty,2\} = 2$

$D_b(i) = \min\{c_{b,a}+D_a(i), c_{b,c}+D_c(i), c_{b,e}+D_e(i)\} = \min\{\infty,\infty,\infty\} = \infty$

DV in b:

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

Distance vector example

t=1

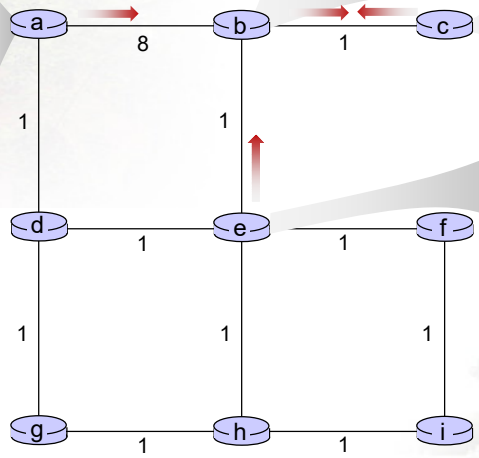
- c receives DVs from b

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



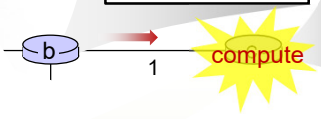
Distance vector example

t=1

- c receives DVs from b computes:

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

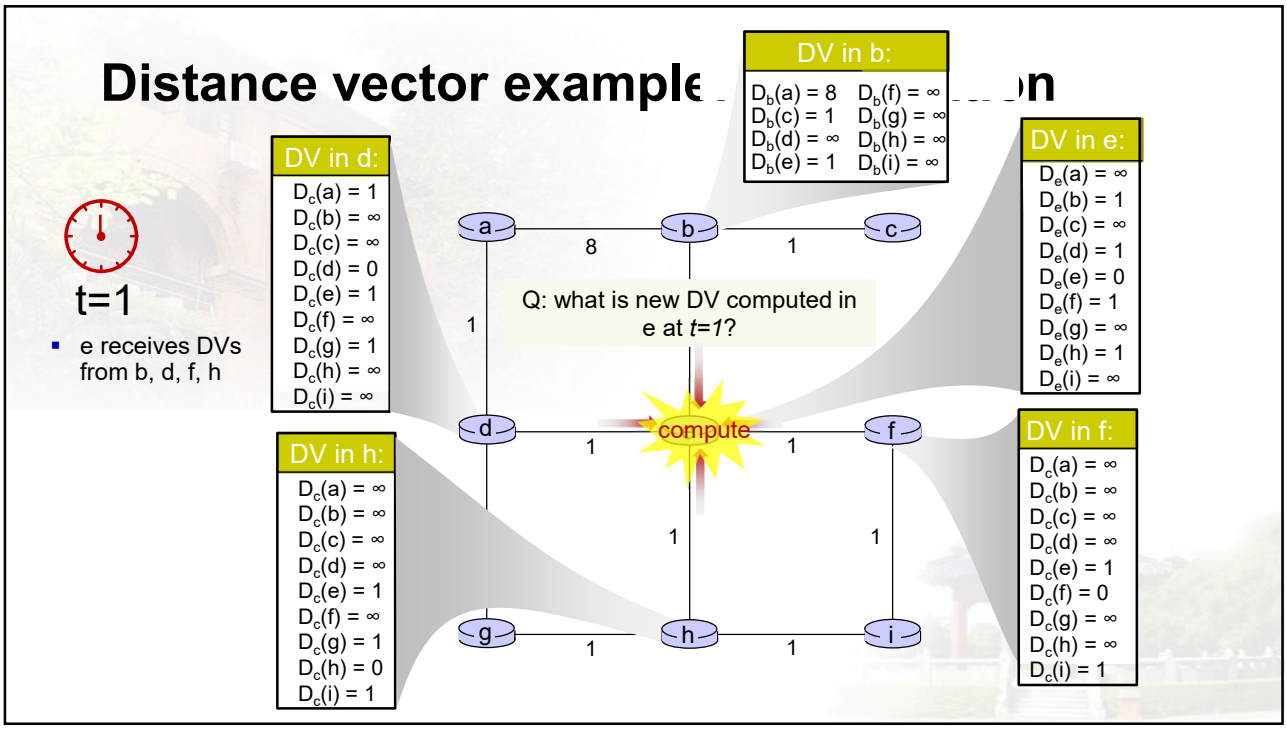


$D_c(a) = \min\{c_{c,b}+D_b(a)\} = 1 + 8 = 9$
$D_c(b) = \min\{c_{c,b}+D_b(b)\} = 1 + 0 = 1$
$D_c(d) = \min\{c_{c,b}+D_b(d)\} = 1+ \infty = \infty$
$D_c(e) = \min\{c_{c,b}+D_b(e)\} = 1 + 1 = 2$
$D_c(f) = \min\{c_{c,b}+D_b(f)\} = 1+ \infty = \infty$
$D_c(g) = \min\{c_{c,b}+D_b(g)\} = 1+ \infty =$
∞
$D_c(h)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

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/

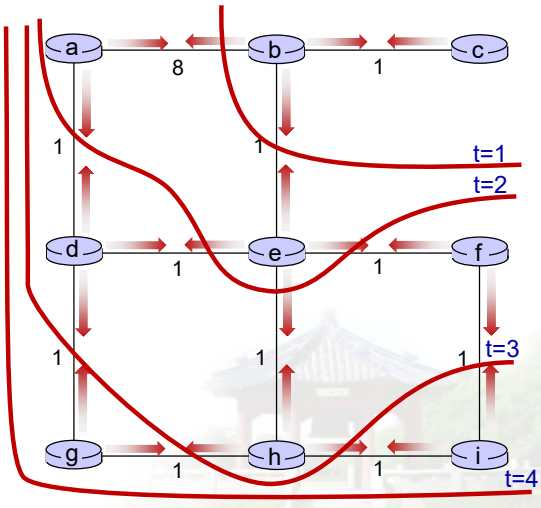
Distance vector example



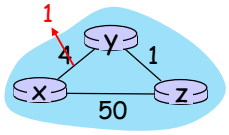
Distance vector: state information diffusion

Iterative communication, computation steps diffuses information through network:

- t=0 c's state at t=0 is at c only
- t=1 c's state at t=0 has propagated to b, and may influence distance vector computations up to **1** hop away, i.e., at b
- t=2 c's state at t=0 may now influence distance vector computations up to **2** hops away, i.e., at b and now at a, e as well
- t=3 c's state at t=0 may influence distance vector computations up to **3** hops away, i.e., at b,a,e and now at c,f,h as well
- t=4 c's state at t=0 may influence distance vector computations up to **4** hops away, i.e., at b,a,e, c, f, h and now at g,i as well



Distance vector: link cost changes



link cost changes:

- node detects local link cost change
- updates routing info, recalculates local DV
- if DV changes, notify neighbors

"good news travels fast"

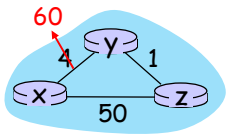
t_0 : *y* detects link-cost change, updates its DV, informs its neighbors.

t_1 : *z* receives update from *y*, updates its table, computes new least cost to *x* , sends its neighbors its DV.

t_2 : *y* receives *z*'s update, updates its distance table. *y*'s least costs do *not* change, so *y* does *not* send a message to *z*.

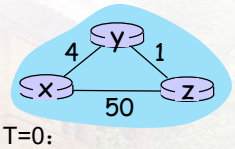
Distance vector: link cost changes

link cost changes:



- node detects local link cost change
- "bad news travels slow" – count-to-infinity problem:
  - *y* sees direct link to *x* has new cost 60, **but z has said it has a path at cost of 5**. So *y* computes "my new cost to x will be 6, via z); notifies *z* of new cost of 6 to *x*.
  - *z* learns that path to *x* via *y* has new cost 6, so *z* computes "my new cost to *x* will be 7 via y), notifies *y* of new cost of 7 to *x*.
  - *y* learns that path to *x* via *z* has new cost 7, so *y* computes "my new cost to *x* will be 8 via y), notifies *z* of new cost of 8 to *x*.
  - *z* learns that path to *x* via *y* has new cost 8, so *z* computes "my new cost to *x* will be 9 via y), notifies *y* of new cost of 9 to *x*.
  - …
- see text for solutions. *Distributed algorithms are tricky!*

解决方法：毒性反转



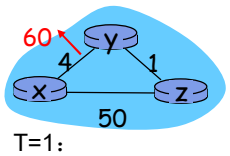
T=0：

X tells Y: $D_X(Z)=50$;
X tells Z: $D_X(Y)=4$;
Y tells X: $D_Y(Z)=1$;
Y tells Z: $D_Y(X)=4$;
Z tells X: $D_Z(Y)=1$;
Z tells Y: $D_Z(X)=50$;

Z updates:
$D_Y(X) = 5$
$D_Z(X) = 50$

Z propagates:
Z tells X: $D_Z(Y)=1$;
Z tells Y: $D_Z(X)=\infty$;

Due to the current shortest path passing through Y, blocking the loop path



T=1：

…

Y tells Z: $D_Y(X)=60$;

…

Z updates:
$D_Y(X) = 60$
$D_Z(X) = 50$

Z propagates:
Z tells X: $D_Z(Y)=1$;
Z tells Y: $D_Z(X)=50$;

45

Comparison of LS and DV algorithms

message complexity

- ***LS:*** **with n nodes, E links, O(nE) msgs sent**
- ***DV:*** **exchange between neighbors only**
  - **convergence time varies**

speed of convergence

- ***LS:*** **$O(n^2)$ algorithm requires O(nE) msgs**
  - **may have oscillations**
- ***DV:*** **convergence time varies**
  - **may be routing loops**
  - **count-to-infinity problem**

robustness: **what happens if router malfunctions?**

LS:

- **node can advertise incorrect *link* cost**
- **each node computes only its *own* table**

DV:

- **DV node can advertise incorrect *path* cost**
- **each node's table used by others**
  - **error propagate thru network**

4-49

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

Making routing scalable

our routing study thus far - idealized

- all routers identical
- network "**flat**"

… *not* true in practice

scale: **with billions of destinations:**

- **Can't store all destinations in routing tables!**
- **routing table exchange would swamp links!**

administrative autonomy

- **internet = network of networks**
- **each network admin may want to control routing in its own network**

Internet approach to scalable routing

aggregate routers into regions known as "autonomous systems" (AS) (a.k.a. "domains")

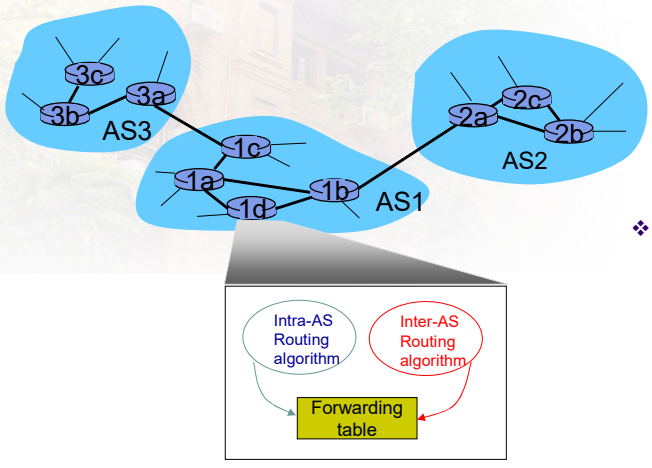
intra-AS routing

- routing among hosts, routers in **same** AS ("network")
- all routers in AS must run ***same*** intra-domain protocol
- routers in *different* AS can run *different* intra-domain routing protocol
- gateway router: at "edge" of its own AS, has link(s) to router(s) in other AS'es

inter-AS routing

- **routing among AS'es**
- **gateways perform inter-domain routing (as well as intra-domain routing)**

Interconnected ASes



- **forwarding table configured by both intra- and inter-AS routing algorithm**
  - **intra-AS sets entries for internal dests**
  - **inter-AS & intra-AS sets entries for external dests**

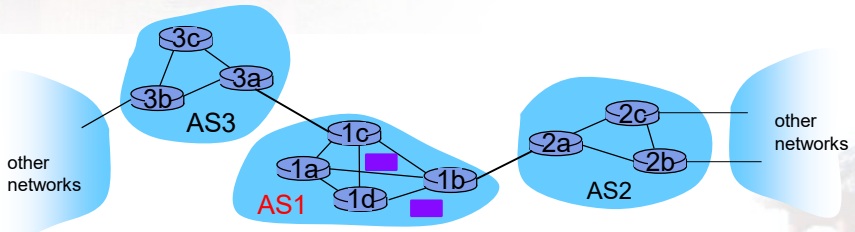
Inter-AS tasks

- suppose router in AS1 receives datagram destined outside of AS1:
  - router should forward packet to gateway router, but which one?

AS1 must:

1. learn which dests are reachable through AS2, which through AS3
2. propagate this reachability info to all routers in AS1

job of inter-AS routing!



4-54

Intra-AS Routing

- also known as *interior gateway protocols (IGP)*
- most common intra-AS routing protocols:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - IGRP: Interior Gateway Routing Protocol (Cisco proprietary for decades, until 2016)

4-55

RIP (Routing Information Protocol)

- RIP 是内部网关协议 IGP 中最先得到广泛使用的协议。
- RIP 是一种分布式的、基于距离向量的路由选择协议。
- RIP 协议要求网络中的每一个路由器都要维护从它自己到其他每一个目的网络的距离记录。
- RIP 认为一个好的路由就是它通过的路由器的数目少，即“距离短”。
- RIP 允许一条路径最多只能包含 15 个路由器。
- “距离”的最大值为 16 时即相当于不可达。可见 RIP 只适用于小型互联网。
- RIP 不能在两个网络之间同时使用多条路由。RIP 选择一个具有最少路由器的路由（即最短路由），哪怕还存在另一条高速(低时延)但路由器较多的路由。

RIP 协议的三个特点

1. 仅和相邻路由器交换信息。
2. 交换的信息是当前本路由器所知道的全部信息，即自己的路由表。
3. 按固定的时间间隔交换路由信息，例如，每隔 30 秒。当网络拓扑发生变化时，路由器也及时向相邻路由器通告拓扑变化后的路由信息。

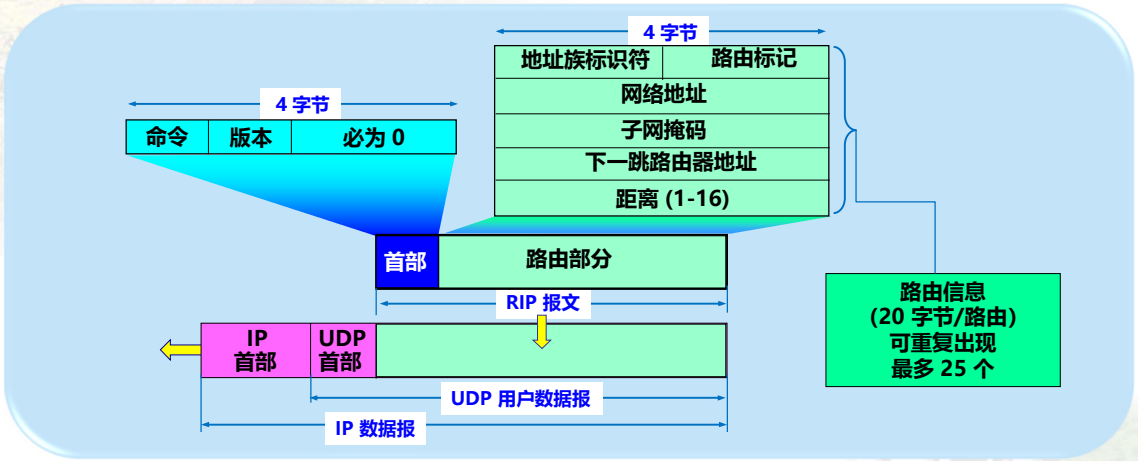
56

路由表的建立

- 路由器在刚开始工作时，只知道到直接连接的网络的距离（此距离定义为 1）。它的路由表是空的。
- 以后，每一个路由器也只和数目非常有限的相邻路由器交换并更新路由信息。
- 经过若干次更新后，所有的路由器最终都会知道到达本自治系统中任何一个网络的最短距离和下一跳路由器的地址。
- RIP 协议的收敛 (convergence) 过程较快。“收敛”就是在自治系统中所有的结点都得到正确的路由选择信息的过程。

57

RIP2 协议的报文格式



RIP2 报文

- RIP2 报文由首部和路由部分组成。
- RIP2 报文中的路由部分由若干个路由信息组成。每个路由信息需要用 20 个字节。地址族标识符（又称为地址类别）字段用来标志所使用的地址协议。
- 路由标记填入自治系统的号码，这是考虑使 RIP 有可能收到本自治系统以外的路由选择信息。
- 再后面指出某个网络地址、该网络的子网掩码、下一跳路由器地址以及到此网络的距离。

RIP2 报文

- 一个 RIP 报文最多可包括 25 个路由，因而 RIP 报文的最大长度是 4+20 x25=504 字节。如超过，必须再用一个 RIP 报文来传送。
- RIP2 具有简单的鉴别功能。
  1. 若使用鉴别功能，则将原来写入第一个路由信息（20 个字节）的位置用作鉴别。
  2. 在鉴别数据之后才写入路由信息，但这时最多只能再放入 24 个路由信息。

好消息传播得快，坏消息传播得慢

- RIP 协议特点：好消息传播得快，坏消息传播得慢。
- RIP 存在的一个问题：当网络出现故障时，要经过比较长的时间 (例如数分钟) 才能将此信息传送到所有的路由器。

RIP 协议的优缺点

- 优点：
  1. 实现简单，开销较小。
- 缺点：
  1. RIP 限制了网络的规模，它能使用的最大距离为 15（16 表示不可达）。
  2. 路由器之间交换的路由信息是路由器中的完整路由表，因而随着网络规模的扩大，开销也就增加。
  3. “坏消息传播得慢”，使更新过程的收敛时间过长。

OSPF (Open Shortest Path First)

- “open”: publicly available
- uses link state algorithm
  - LS packet dissemination
  - topology map at each node
  - route computation using Dijkstra’s algorithm
- OSPF advertisement carries one entry per neighbor
- advertisements flooded to *entire* AS
  - carried in OSPF messages directly over IP (rather than TCP or UDP)
- *IS-IS routing* protocol: nearly identical to OSPF

4-63

OSPF "advanced" features (not in RIP)

- ***security:*** **all OSPF messages authenticated (to prevent malicious intrusion)**
- **multiple same-cost paths allowed (only one path in RIP)**
- **for each link, multiple cost metrics for different TOS (e.g., satellite link cost set "low" for best effort ToS; high for real time ToS)**
- **integrated uni- and multicast support:**
  - **Multicast OSPF (MOSPF) uses same topology data base as OSPF**
- **hierarchical OSPF in large domains.**

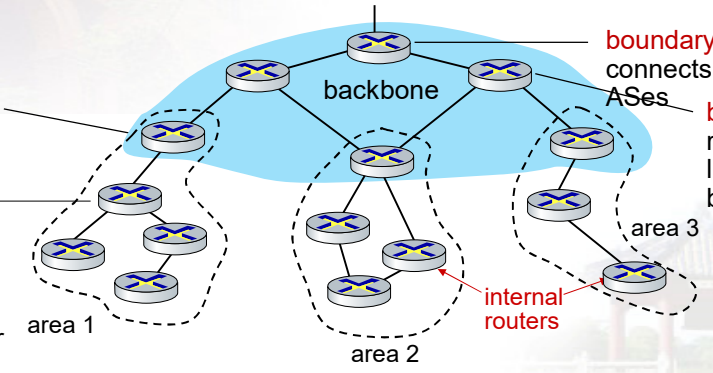
4-64

Hierarchical OSPF

- two-level hierarchy: local area, backbone.
  - link-state advertisements flooded only in area, or backbone
  - each node has detailed area topology; only knows direction to reach other destinations

area border routers: "summarize" distances to destinations in own area, advertise in backbone

local routers:
- flood LS in area only
- compute routing within area
- forward packets to outside via area border router



Network Layer: 5-65

Hierarchical OSPF

- *two-level hierarchy:* local area, backbone.
  - link-state advertisements only in area
  - each nodes has detailed area topology; only know direction (shortest path) to nets in other areas.
- *area border routers:* "summarize" distances to nets in own area, advertise to other Area Border routers.
- *backbone routers:* run OSPF routing limited to backbone.
- *boundary routers:* connect to other AS's.

Chapter 5: outline

5.1 introduction

5.2 routing protocols
- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

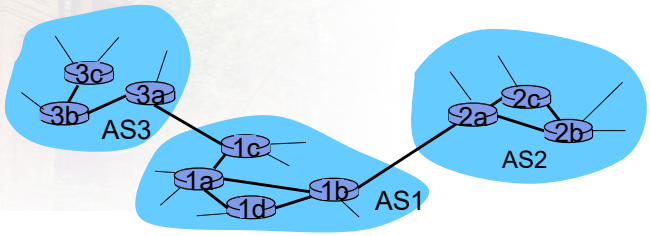
5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

Internet inter-AS routing: BGP



BGP： Implementing communication between ASes

20241118

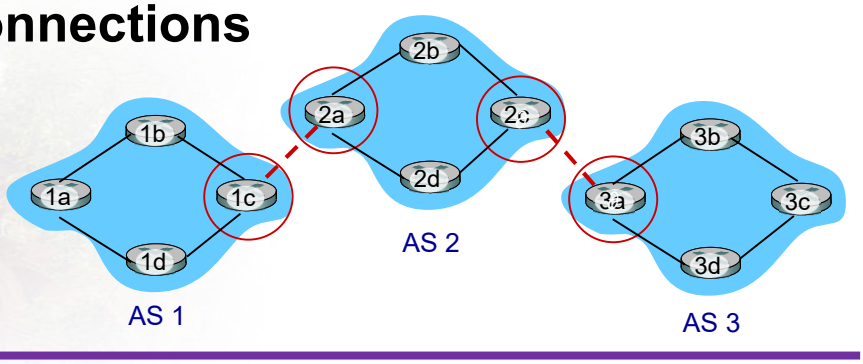
4-68

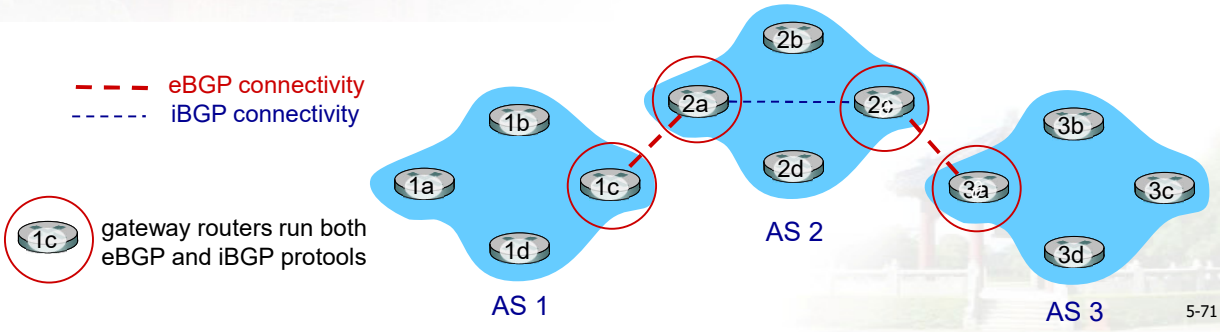
Internet inter-AS routing: BGP

- **BGP (Border Gateway Protocol): *the* de facto inter-domain routing protocol**
  - **"glue that holds the Internet together"**
- **BGP provides each AS a means to:**
  - **eBGP: obtain subnet reachability information from neighboring ASs.**
  - **iBGP: propagate AS reachability information to all AS-internal routers.**
  - **determine "good" routes to other networks based on reachability information and policy.**
- **allows subnet to advertise its existence to rest of Internet: *"I am here"***

4-69

eBGP, iBGP connections





5-71

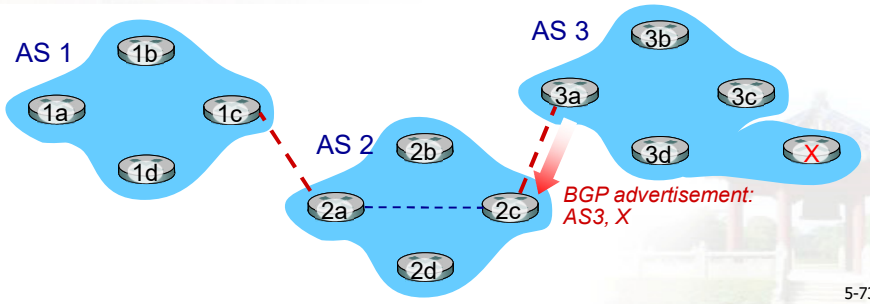
Theoretically, a border router can transmit the AS accessibility message to all internal routers through iBGP, including all non-border routers. However, this will sharply increase the burden on the internal router, because the routing table of any core router is very large and changes frequently.

Therefore, it is not a good idea to propagate the AS accessibility message to all internal routers.

72

BGP basics

- BGP session: two BGP routers ("peers") exchange BGP messages over semi-permanent TCP connection:
  - advertising *paths* to different destination network prefixes (BGP is a "path vector" protocol)
- **when AS3 gateway router 3a advertises path AS3,X to AS2 gateway router 2c:**
  - **AS3 *promises* to AS2 it will forward datagrams towards X**



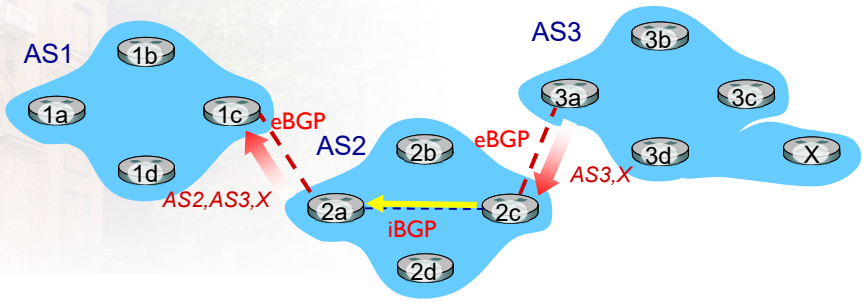
5-73

Path attributes and BGP routes

- **advertised prefix includes BGP attributes**
  - **prefix + attributes = "route"**
- **two important attributes:**
  - **AS-PATH: list of ASes through which prefix advertisement has passed**
  - **NEXT-HOP: indicates specific internal-AS router to next-hop AS**
- ***Policy-based routing:***
  - **gateway receiving route advertisement uses *import policy* to accept/decline path (e.g., never route through AS Y).**
  - **AS policy also determines whether to *advertise* path to other other neighboring ASes**

5-74

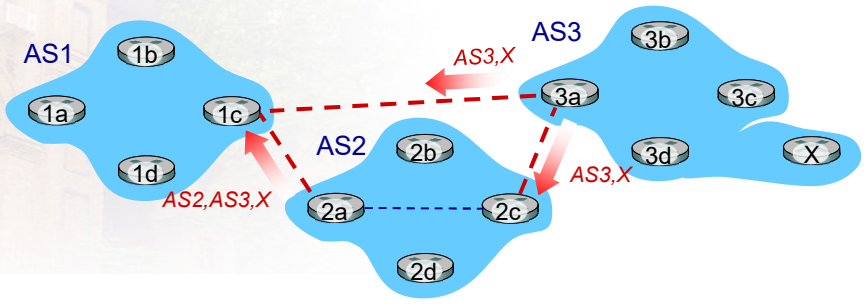
BGP path advertisement



- AS2 router 2c receives path advertisement AS3,X (via eBGP) from AS3 router 3a
- **Based on AS2 policy, AS2 router 2c accepts path AS3,X, propagates (via iBGP) to all AS2 routers**
- Based on AS2 policy, AS2 router 2a advertises (via eBGP) path AS2,AS3,X to AS1 router 1c

5-75

BGP path advertisement



gateway router may learn about multiple paths to destination:

- **AS1 gateway router 1c learns path *AS2,AS3,X* from 2a**
- AS1 gateway router 1c learns path *AS3,X* from 3a
- Based on policy, AS1 gateway router 1c chooses path *AS3,X, and advertises path within AS1 via iBGP*

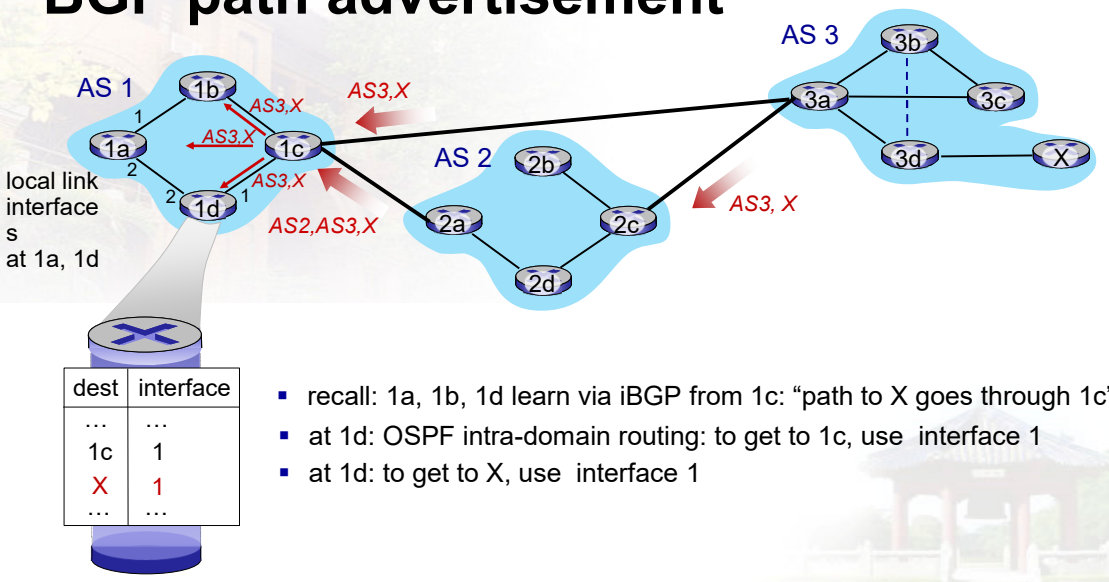
5-76

BGP messages

- BGP messages exchanged between peers over TCP connection
- BGP messages:
  - **OPEN:** opens TCP connection to remote BGP peer and authenticates sending BGP peer
  - **UPDATE:** advertises **new path** (or withdraws old)
  - **KEEPALIVE:** keeps connection alive in absence of UPDATES; also ACKs OPEN request
  - **NOTIFICATION:** reports **errors** in previous msg; also used to close connection

5-77

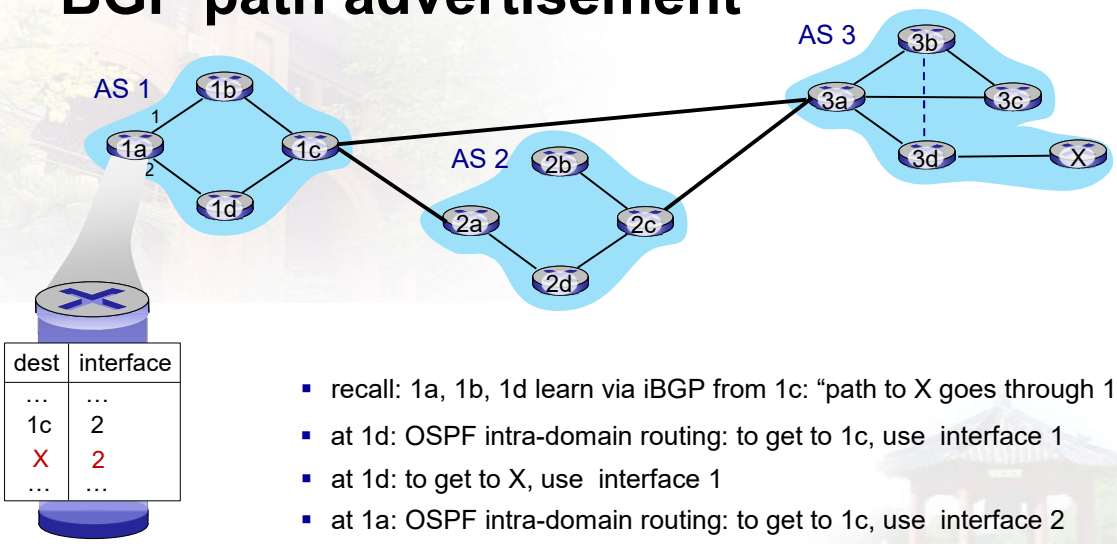
BGP path advertisement



- recall: 1a, 1b, 1d learn via iBGP from 1c: "path to X goes through 1c"
- at 1d: OSPF intra-domain routing: to get to 1c, use interface 1
- at 1d: to get to X, use interface 1

Network Layer: 5-78

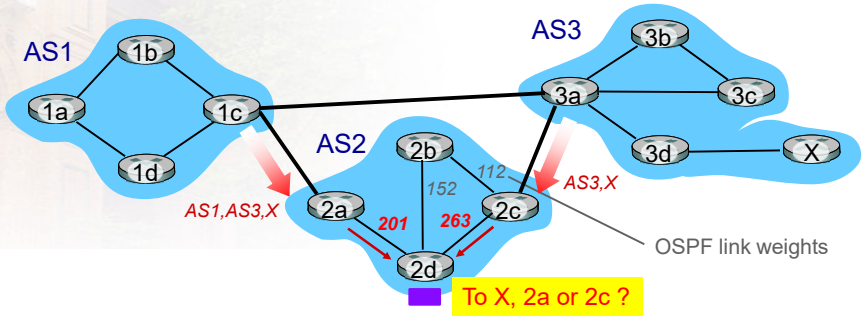
BGP path advertisement



dest	interface
…	…
1c	2
X	2
…	…

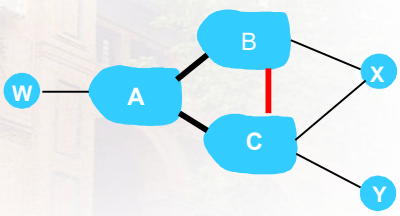
- recall: 1a, 1b, 1d learn via iBGP from 1c: "path to X goes through 1c"
- at 1d: OSPF intra-domain routing: to get to 1c, use interface 1
- at 1d: to get to X, use interface 1
- at 1a: OSPF intra-domain routing: to get to 1c, use interface 2
- at 1a: to get to X, use interface 2

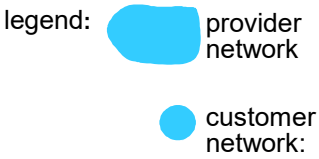
Hot Potato Routing



- **2d learns (via iBGP) it can route to X via 2a or 2c**
- ***hot potato routing:*** **choose local gateway that has least intra-domain cost (e.g., 2d chooses 2a, even though more AS hops to *X*): don't worry about inter-domain cost!**

BGP: achieving policy via advertisements



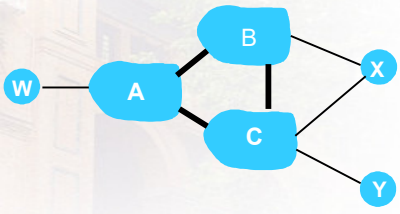


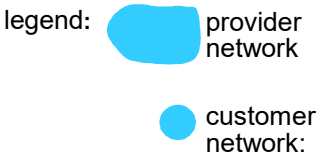
Suppose an ISP only wants to route traffic to/from its customer networks (does not want to carry transit traffic between other ISPs)

- A advertises path Aw to B and to C
- B *chooses not to advertise* BAw to C:
  - B gets no "revenue" for routing CBAw, since none of C, A, w are B's customers
  - C does not learn about CBAw path
- C will route CAw (not using B) to get to w

5-81

BGP: achieving policy via advertisements





Suppose an ISP only wants to route traffic to/from its customer networks (does not want to carry transit traffic between other ISPs)

- A,B,C are *provider networks*
- X,W,Y are customer (of provider networks)
- X is *dual-homed:* attached to two networks
- *policy to enforce:* X does not want to route from B to C via X
  - .. so X will not advertise to B a route to C

5-82

Why different Intra-, Inter-AS routing ?

policy:

- inter-AS: admin wants control over how its traffic routed, who routes through its net. uncontrollable
- intra-AS: single admin, so no policy decisions needed

scale:

- hierarchical routing saves table size, reduced update traffic

performance:

- intra-AS: can focus on performance
- inter-AS: policy may dominate over performance

4-83

BGP route selection

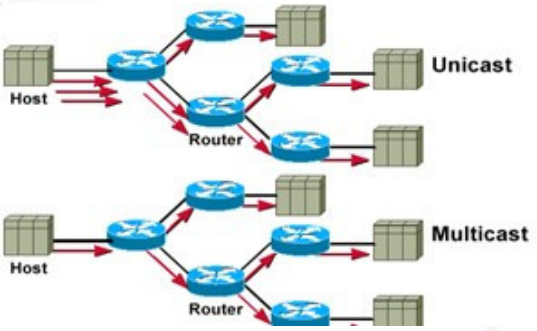
- router may learn about more than one route to destination AS, selects route based on:
  1. local preference value attribute: policy decision
  2. shortest AS-PATH
  3. closest NEXT-HOP router: hot potato routing
  4. additional criteria

Network Layer: 5-84

Unicast & Multicast

单播（Unicast）是在一个单个的发送者和一个接受者之间通过网络进行的通信。

IP多播是实现数据一对多通信的模式。从一个源点传送到多个目的地，数据仅仅拷贝一份。这里说的数据仅仅拷贝一份，是指在每一条须要它的两个点之间，数据仅仅有一份。



85

IP-Anycast

- **replicating the same content on different servers in many different dispersed geographical locations**

- having each user access the content from the server that is closest

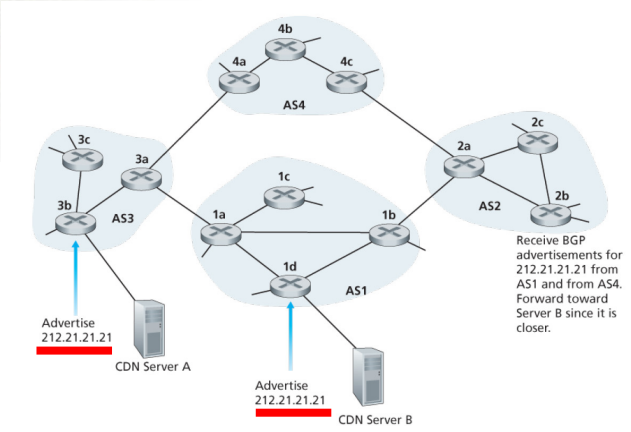


Figure 5.12 Using IP-anycast to bring users to the closest CDN server

86

IP-Anycast

- **in practice CDNs generally choose NOT to use IP-anycast because BGP routing changes can result in different packets of the same TCP connection arriving at different instances of the Web server**
- **IP-anycast is extensively used by the DNS system to direct DNS queries to the closest root DNS server.**
  - **There are currently 13 IP addresses for root DNS servers. But corresponding to each of these addresses, there are multiple DNS root servers, with some of these addresses having over 100 DNS root servers scattered over all corners of the world.**

87

5.4.6 Putting the Pieces Together: Obtaining Internet Presence

- **Please read the Section 5.4.6 carefully**
  - IP addressing, DNS, BGP and intra-AS routing

88

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

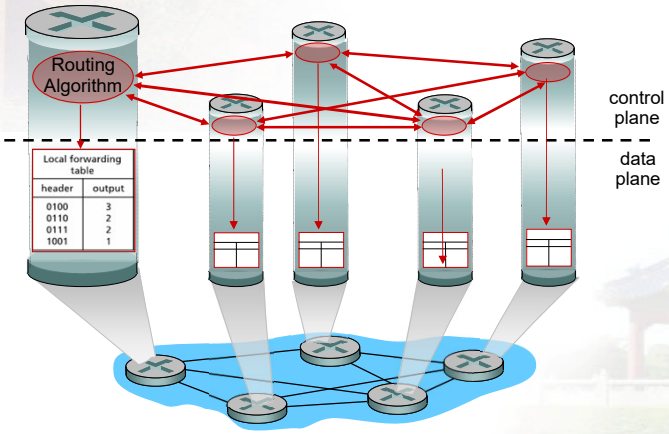
5.7 Network management and SNMP

Software defined networking (SDN)

- Internet network layer: historically has been implemented via distributed, per-router approach
  - *monolithic* router contains switching hardware, runs proprietary implementation of Internet standard protocols (IP, RIP, IS-IS, OSPF, BGP) in proprietary router OS (e.g., Cisco IOS)
  - different "middleboxes" for different network layer functions: firewalls, load balancers, NAT boxes, …
- ~2005: renewed interest in rethinking network control plane

Recall: per-router control plane

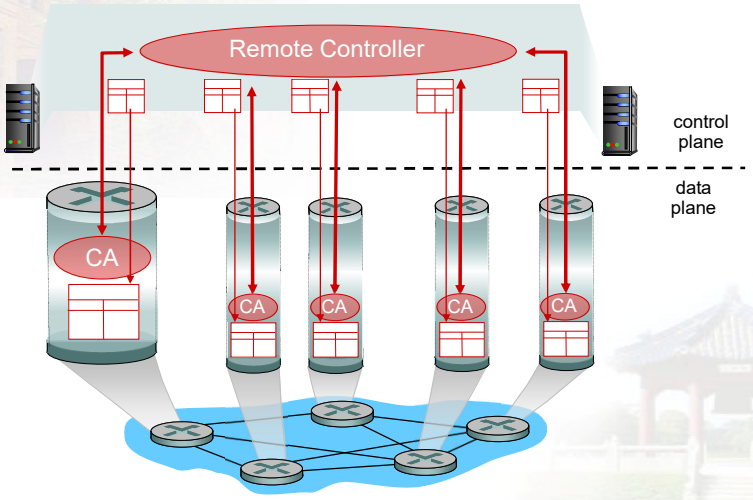
Individual routing algorithm components *in each and every router* interact with each other in control plane to compute forwarding tables



5-91

Recall: logically centralized control plane

A distinct (typically remote) controller interacts with local control agents (CAs) in routers to compute forwarding tables



2023.11.6

5-92

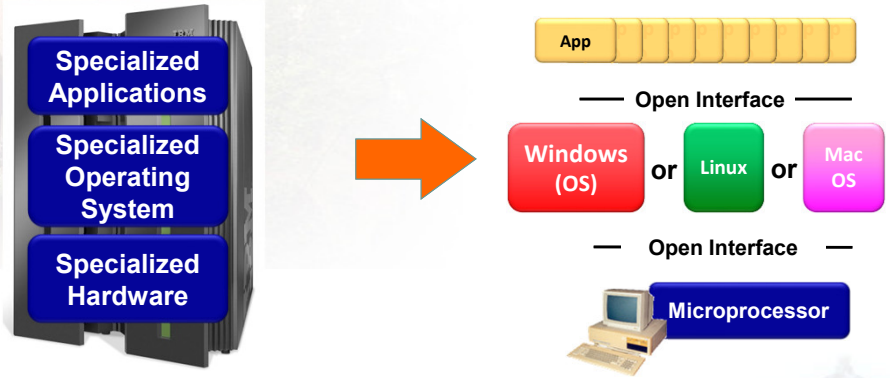
Software defined networking (SDN)

Why a _logically centralized_ control plane?

- easier network management: avoid router misconfigurations, greater flexibility of traffic flows
- table-based forwarding (recall OpenFlow API) allows "programming" routers
  - centralized "programming" easier: compute tables centrally and distribute
  - distributed "programming: more difficult: compute tables as result of distributed algorithm (protocol) implemented in each and every router
- open (non-proprietary) implementation of control plane

5-93

Analogy: mainframe to PC evolution*



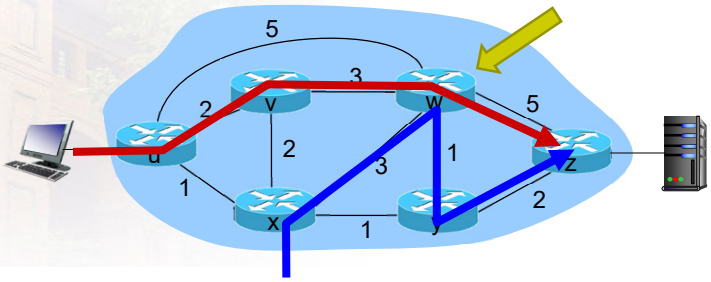
Vertically integrated
Closed, proprietary
Slow innovation
Small industry

Horizontal
Open interfaces
Rapid innovation
Huge industry

* Slide courtesy: N. McKeown

5-94

Traffic engineering: difficult

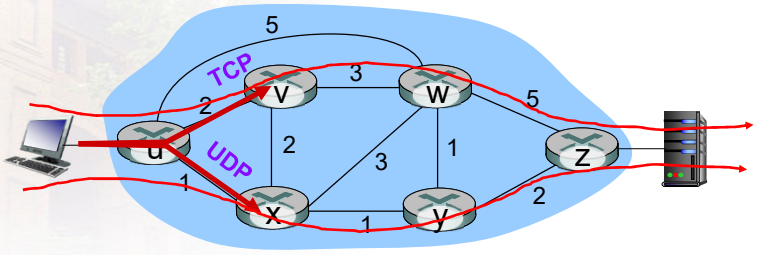


Q: what if w wants to route blue and red traffic differently?

A: can't do it (with destination based forwarding, and LS, DV routing) (or need a new routing algorithm)

5-95

Traffic engineering: difficult

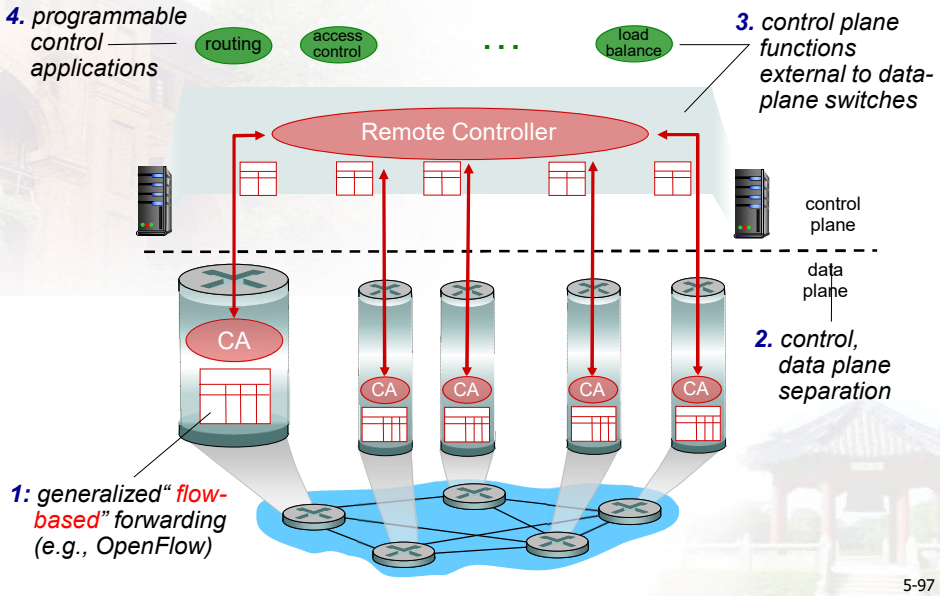


Q: what if network operator wants to split u-to-z traffic along uvwz *and* uxyz (load balancing)?

A: can't do it (or need a new routing algorithm)

5-96

Software defined networking (SDN)

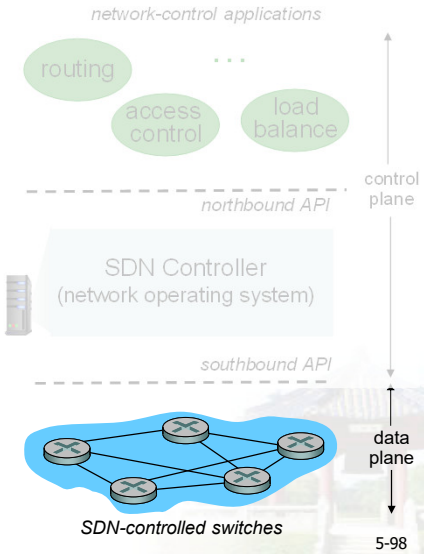


5-97

SDN perspective: data plane switches

Data plane switches

- **fast, simple, commodity switches implementing generalized data-plane forwarding (Section 4.4) in hardware**
- **switch flow table computed, installed by controller**
- **API for table-based switch control (e.g., OpenFlow)**
  - **defines what is controllable and what is not**
- **protocol for communicating with controller (e.g., OpenFlow)**



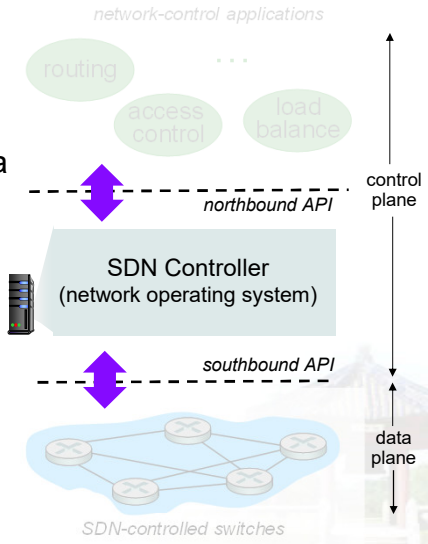
5-98

SDN perspective: SDN controller

SDN controller (network OS):

- maintain network state information
- interacts with network control applications "above" via northbound API
- interacts with network switches "below" via southbound API
- implemented as distributed system for performance, scalability, fault-tolerance, robustness

Transform the control policy into a forwarding table



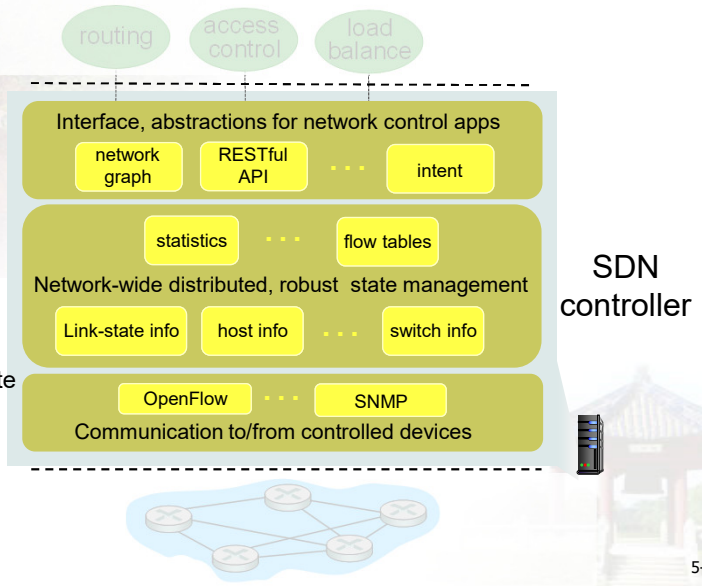
5-99

Components of SDN controller

Interface layer to network control apps: abstractions API

Network-wide state management layer: state of networks links, switches, services: a *distributed database*

communication layer: communicate between SDN controller and controlled switches



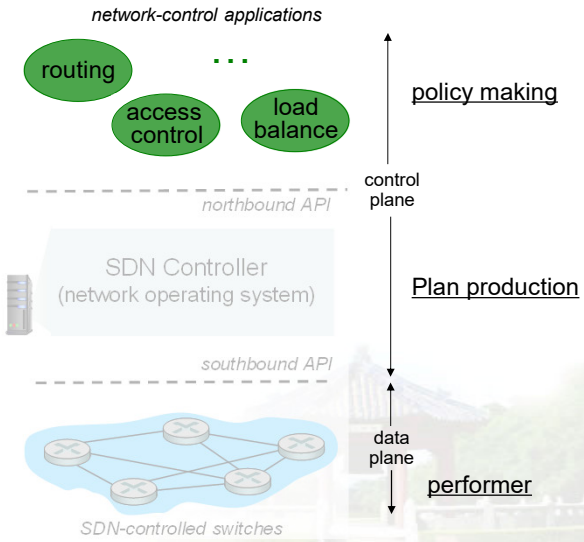
5-100

SDN perspective: control applications

network-control apps:

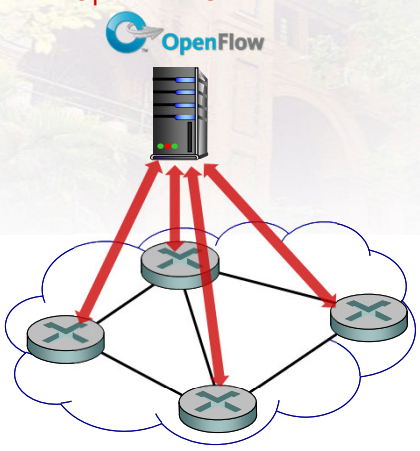
- "brains" of control: implement control functions using lower-level services, API provided by SND controller
- *unbundled:* can be provided by 3rd party: distinct from routing vendor, or SDN controller

Decide the control policy



OpenFlow protocol

OpenFlow Controller



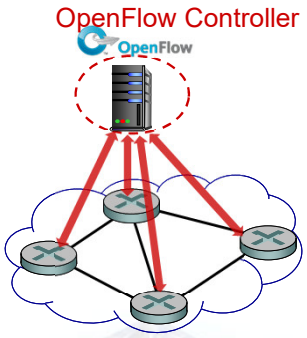
- operates between controller, switch
- TCP used to exchange messages
  - optional encryption
- OpenFlow messages:
  - controller-to-switch
  - switch to controller

5-102

OpenFlow: controller-to-switch messages

Key controller-to-switch messages

- ***features:*** **controller queries switch features, switch replies**
- ***configure:*** **controller queries/sets switch configuration parameters**
- ***modify-state:*** **add, delete, modify flow entries in the OpenFlow tables**
- ***packet-out:*** **controller can send this packet out of specific switch port**

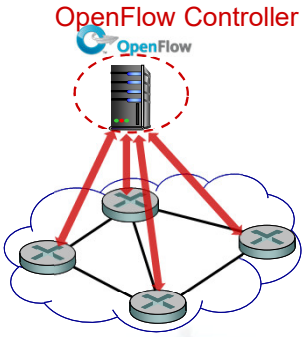


5-103

OpenFlow: switch-to-controller messages

Key switch-to-controller messages

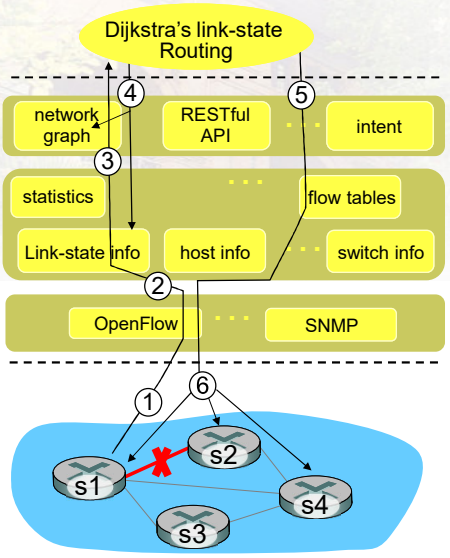
- ***packet-in:*** **transfer packet (and its control) to controller. See packet-out message from controller**
- ***flow-removed:*** **flow table entry deleted at switch**
- ***port status:*** **inform controller of a change on a port.**



Fortunately, network operators don't "program" switches by creating/sending OpenFlow messages directly. Instead use higher-level abstraction at controller

5-104

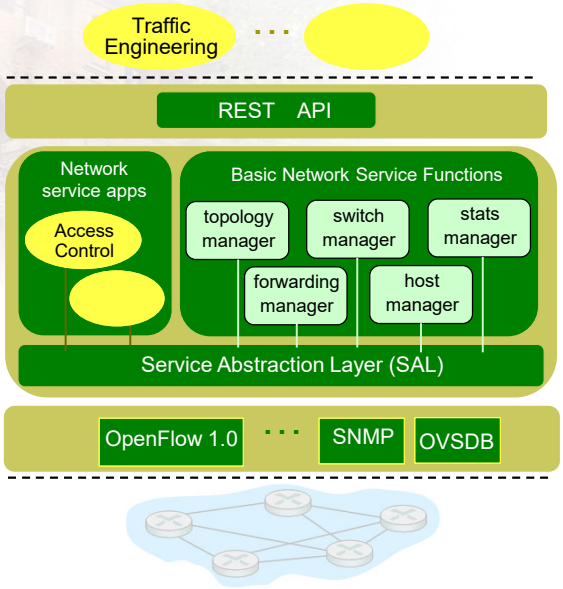
SDN: control/data plane interaction example



1. S1, experiencing link failure using OpenFlow port status message to notify controller
2. SDN controller receives OpenFlow message, updates link status info
3. Dijkstra's routing algorithm application has previously registered to be called when ever link status changes. It is called.
4. Dijkstra's routing algorithm access network graph info, link state info in controller, computes new routes
5. link state routing app interacts with flow-table-computation component in SDN controller, which computes new flow tables needed
6. Controller uses OpenFlow to install new tables in switches that need updating

5-105

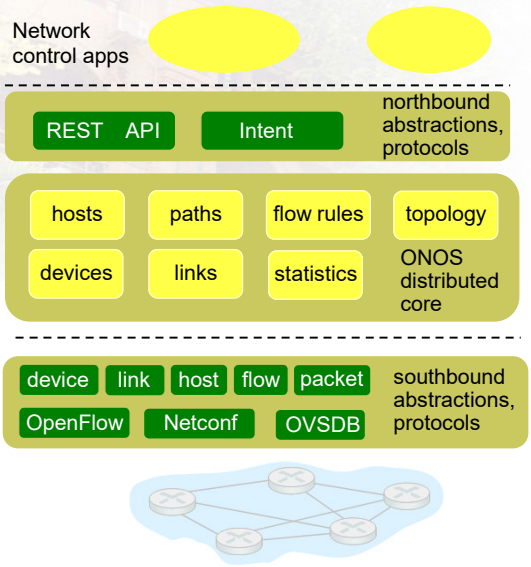
OpenDaylight (ODL) controller



- ODL Lithium controller
- network apps may be contained within, or be external to SDN controller
- Service Abstraction Layer: interconnects internal, external applications and services

5-106

OpenNetworkOS controller



- control apps separate from controller
- intent framework: high-level specification of service: what rather than how
- considerable emphasis on distributed core: service reliability, replication performance scaling

5-107

SDN: selected challenges

- **hardening the control plane: dependable, reliable, performance-scalable, secure distributed system**
  - **robustness to failures: leverage strong theory of reliable distributed system for control plane**
  - **dependability, security: "baked in" from day one?**
- **networks, protocols meeting mission-specific requirements**
  - **e.g., real-time, ultra-reliable, ultra-secure**
- **Internet-scaling**

5-108

SDN and the future of traditional network protocols

- SDN-computed versus router-computer forwarding tables:
  - just one example of logically-centralized-computed versus protocol computed (clouding)
- one could imagine SDN-computed congestion control:
  - controller sets sender rates based on router-reported (to controller) congestion levels

How will implementation of network functionality (SDN versus protocols) evolve?

Chapter 5: outline

5.1 introduction

5.2 routing protocols
- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

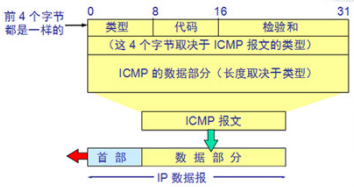
5.7 Network management and SNMP

ICMP: internet control message protocol

- used by hosts & routers to communicate network-level information
  - error reporting: unreachable host, network, port, protocol
  - echo request/reply (used by ping)
- network-layer "above" IP:
  - ICMP msgs carried in IP datagrams
- ICMP message: type, code plus first 8 bytes of IP datagram causing error

ICMP Protocol number：1

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



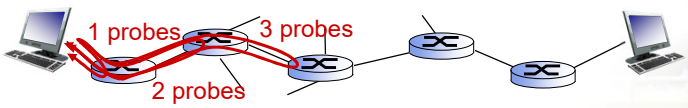
4-111

Traceroute and ICMP

- source sends series of UDP segments to dest
  - first set has TTL =1
  - second set has TTL=2, etc.
  - unlikely port number
- when *n*th set of datagrams arrives to nth router:
  - router discards datagrams
  - and sends source ICMP messages (type 11, code 0)
  - ICMP messages includes name of router & IP address
- when ICMP messages arrives, source records RTTs

stopping criteria:

- UDP segment eventually arrives at destination host
- destination returns ICMP "**port unreachable**" message (type 3, code 3)
- source stops



4-116

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

What is network management?

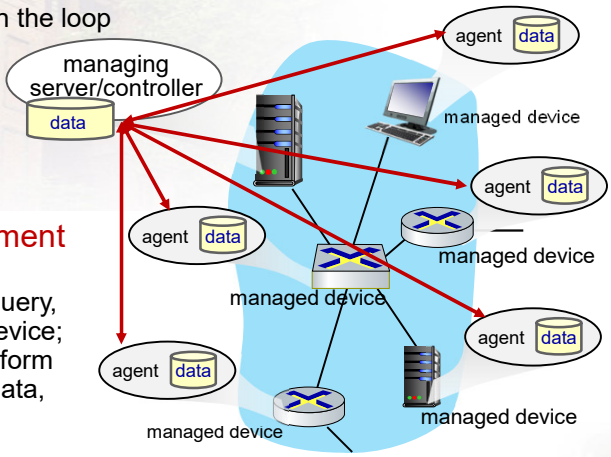
- autonomous systems (aka "network"): 1000s of interacting hardware/software components
- other complex systems requiring monitoring, control:
  - jet airplane
  - nuclear power plant
  - others?

"Network management includes the deployment, integration and coordination of the hardware, software, and human elements to monitor, test, poll, configure, analyze, evaluate, and control the network and element resources to meet the real-time, operational performance, and Quality of Service requirements at a reasonable cost."

Components of network management

Managing server: application, typically with network managers (humans) in the loop

Network management protocol: used by managing server to query, configure, manage device; used by devices to inform managing server of data, events.



Managed device: equipment with manageable, configurable hardware, software components

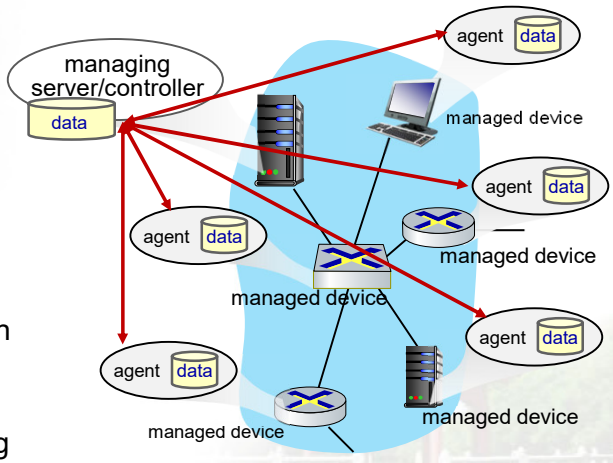
Data: device "state" configuration data, operational data, device statistics

Network operator approaches to management

CLI (Command Line Interface)
- operator issues (types, scripts) direct to individual devices (e.g., vis ssh)

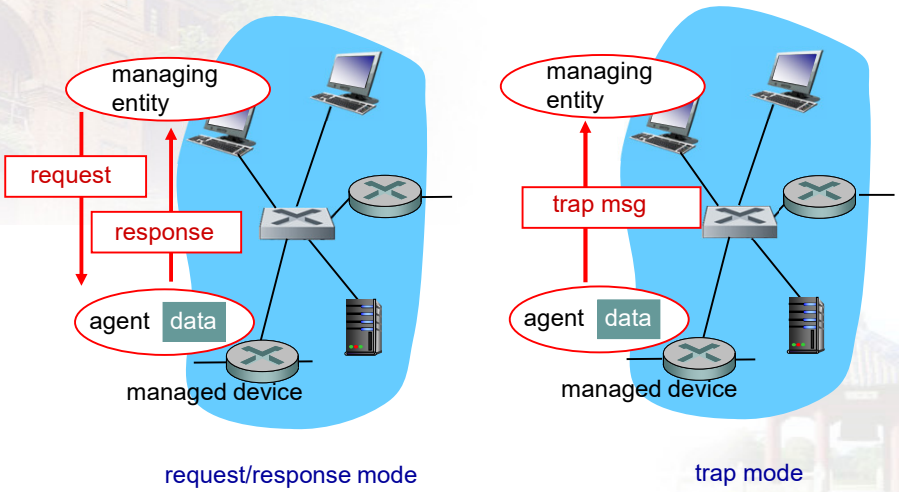
SNMP/MIB
- operator queries/sets devices data (MIB) using Simple Network Management Protocol (SNMP)

NETCONF/YANG
- more abstract, network-wide, holistic
- emphasis on multi-device configuration management.
- YANG: data modeling language
- NETCONF: communicate YANG-compatible actions/data to/from/among remote devices



SNMP protocol

Two ways to convey MIB info, commands:



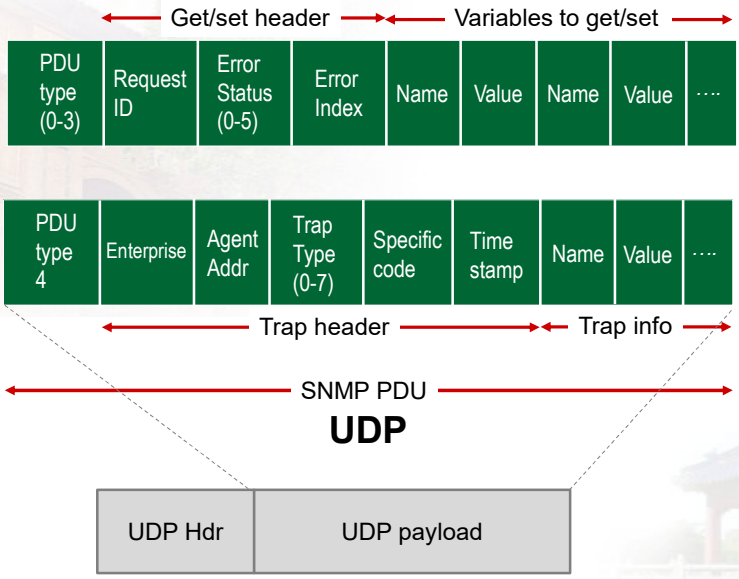
5-121

SNMP protocol: message types

Message type	Function
GetRequest GetNextRequest GetBulkRequest	manager-to-agent: "get me data" (data instance, next data in list, block of data).
SetRequest	manager-to-agent: set MIB value
Response	Agent-to-manager: value, response to Request
Trap	Agent-to-manager: inform manager of exceptional event

Network Layer: 5-122

SNMP protocol: message formats



More on network management: see earlier editions of text!

5-123

SNMP: Management Information Base (MIB)

- **managed device's operational (and some configuration) data**
- **gathered into device MIB module**
  - **400 MIB modules defined in RFC's; many more vendor-specific MIBs**
  - Structure of Management Information (SMI): data definition language
  - example MIB variables for UDP protocol:

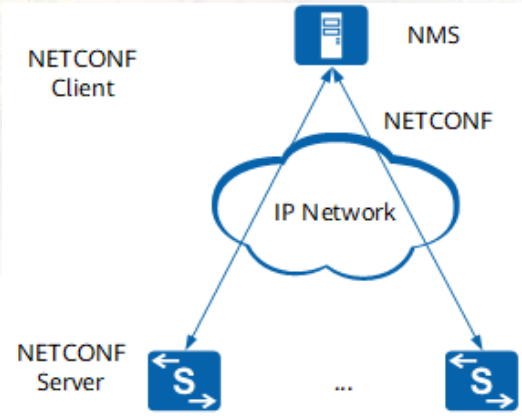
Object ID	Name	Type	Comments
1.3.6.1.2.1.7.1	UDPInDatagrams	32-bit counter	total # datagrams delivered
1.3.6.1.2.1.7.2	UDPNoPorts	32-bit counter	# undeliverable datagrams (no application at port)
1.3.6.1.2.1.7.3	UDInErrors	32-bit counter	# undeliverable datagrams (all other reasons)
1.3.6.1.2.1.7.4	UDPOutDatagrams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

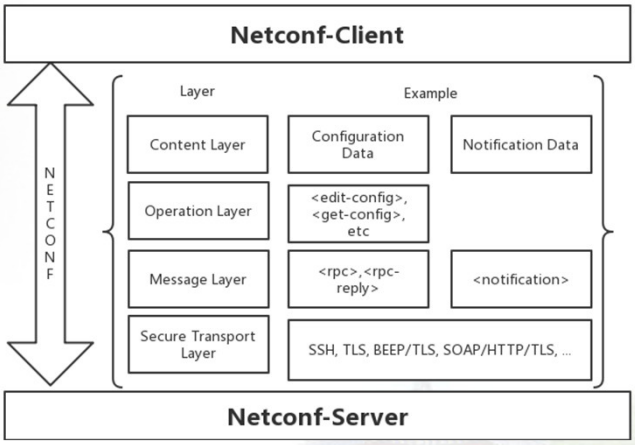
Network Layer: 5-124

NETCONF (Network Configuration Protocol) overview

- goal: actively manage/configure devices network-wide
- operates between managing server and managed network devices
  - actions: retrieve, set, modify, activate configurations
  - atomic-commit actions over multiple devices
  - query operational data and statistics
  - subscribe to notifications from devices
- remote procedure call (RPC) paradigm
  - NETCONF protocol messages encoded in XML
  - exchanged over secure, reliable transport (e.g., TLS) protocol

NETCONF overview



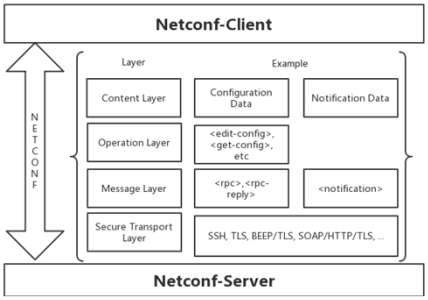


NETCONF overview

- NETCONF认为网络的模型数据可以分为两大类: 状态数据和配置数据。
  - 状态数据一般指server（设备）的固有属性数据和当前运行的状态数据等，这类数据仅能查询。
  - 配置数据则是指由用户（以某种方式）配置到server上的数据。而配置数据本身又可以存在多个数据库，标准中提到了<running/>库用于保存当前已经生效的配置；<candidate/>用于保存可以提交为生效的数据；以及<startup/>用于保存启动时的配置数据。

NETCONF overview

- 安全传输层
  - NETCONF是一个应用层协议, 承载在安全传输层之上, NETCONF协议中并没有规定建链和保活相关的内容。
- 消息层
  - NETCONF中定义了三种消息类型，分别是hello， rpc和rpc-reply, notification。
- 操作层
  - 操作层仅承载在<rpc>和<rpc-reply>消息上，<hello>和<notification>消息无操作层。
  - NETCONF协议规定了简单的rpc操作，同时也支持用户自定义rpc操作。

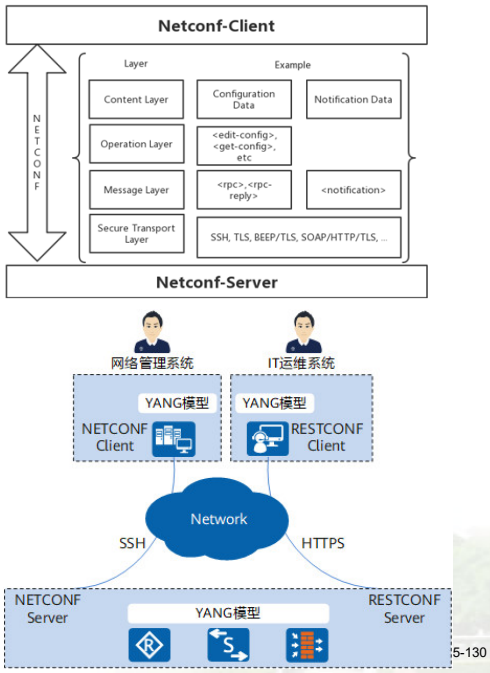


Selected NETCONF Operations

NETCONF	Operation Description
<get-config>	Retrieve all or part of a given configuration. A device may have multiple configurations.
<get>	Retrieve all or part of both configuration state and operational state data.
<edit-config>	Change specified (possibly running) configuration at managed device. Managed device <rpc-reply> contains <ok> or <rpcerror> with rollback.
<copy-config>	Copy a database
<delete-config>	Delete a database
<lock>, <unlock>	Lock (unlock) configuration datastore at managed device (to lock out NETCONF, SNMP, or CLIs commands from other sources).
<create-subscription>, <notification> <close-session> <kill-session>	Enable event notification subscription from managed device

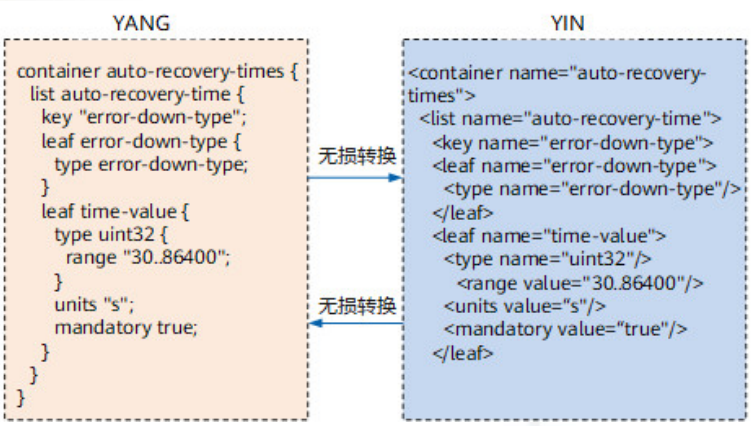
NETCONF overview

- 内容层
  - 在netconf出现之前，我们所熟知且常用的协议，均采用在协议中规定报文的结构体，并按字节流读取并解析的架构。(例如: TLV)
    - 缺乏扩展性
  - netconf协议本身没有对内容层的数据结构做任何的限定。而其规范则体现在其内容层需要使用Yang （Yet Another Next Generation）语言对其数据进行建模。



NETCONF overview

- YANG相对于SNMP的模型MIB，更有层次化，能够区分配置和状态，可扩展性强。
- 设备解析模型时用YIN（YANG Independent Notation）模型文件。YIN是XML表达方式的YANG，YIN与YANG之间使用不同的表达方法但包含等价的信息。

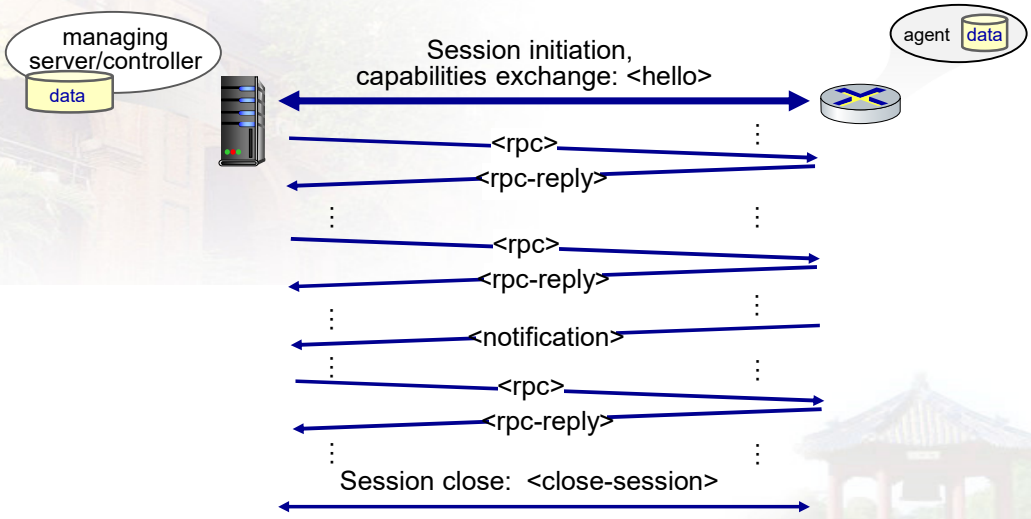


用YIN是为了利用各编程语言中现有的XML解析器等工具。这些工具可用来进行数据过滤和验证，自动生成代码和文件或者其他任务，这样可以提升设备解析YANG模型的效率。

Sample NETCONF RPC message

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <rpc message-id="101"  note message id
03   xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
04   <edit-config>     change a configuration
05     <target>
06       <running/> change the running configuration
07     </target>
08     <config>
09       <top xmlns="http://example.com/schema/
         1.2/config">
10         <interface>
11           <name>Ethernet0/0</name>  change MTU of Ethernet 0/0 interface to 1500
12           <mtu>1500</mtu>
13         </interface>
14       </top>
15     </config>
16   </edit-config>
17 </rpc>
```

NETCONF initialization, exchange, close



Network layer: Summary

we've learned a lot!

- approaches to network control plane
  - per-router control (traditional)
  - logically centralized control (software defined networking)
- traditional routing algorithms
  - implementation in Internet: OSPF , BGP
- SDN controllers
  - implementation in practice: ODL, ONOS
- Internet Control Message Protocol
- network management

next stop: link layer!

Chapter 5: summary

we've learned a lot!

- approaches to network control plane
  - per-router control (traditional)
  - logically centralized control (software defined networking)
- traditional routing algorithms
  - implementation in Internet: OSPF, BGP
- SDN controllers
  - implementation in practice: ODL, ONOS
- Internet Control Message Protocol
- network management

next stop: link layer!

Thanks

Q & A

Email: xieyi5@mail.sysu.edu.cn
https://cse.sysu.edu.cn/content/2462