Chapter 5
Network Layer: Control Plane

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

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

optimal path

connection between ISP infrastructure, determine path from 1 isp infras to another infras

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Network-layer functions

- *f* forwarding: move packets from router's input to appropriate router output — *data plane*
- *f* routing: determine route taken by packets from source to destination — *control plane*

Two approaches to structuring network control plane:

- *f* per-router control (traditional) router control
- *f* logically centralized control (software defined networking)

controller control (centrialize controler)

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Per-router control plane

router implement both forwarding and routing

Individual routing algorithm components *in each and every router* interact in the control plane

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Software-Defined Networking (SDN) control plane

Remote controller computes, installs forwarding tables in routers

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

E š Á } Œ l o Ç Œ W ^ } v š Œ } o ‰

- introduction
- routing protocols
  - link state
  - distance vector
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol
- network management, configuration
  - SNMP
  - NETCONF/YANG

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Routing protocols

routing algo is the algo which input are all information..., using routing protocols , output is the path, the router need the fowarding table, so the output must be transfer to the forwrding table

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

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

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Graph abstraction: link costs

good path : it can be the good interm of delay, distance or bandwidth

$c_{a,b}$: cost of *direct* link connecting *a* and *b*

e.g., $c_{w,z} = 5$, $c_{u,z} = \infty$

cost defined by network operator: could always be 1, or inversely related to bandwidth, or inversely related to congestion

graph: $G = (N,E)$

nowadays, the link is very fast,\ -> we just "hop count": count number of the router

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)* }

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Routing algorithm classification

global: all routers have *complete* topology, link cost info

- "link state" algorithms

each time there is a change in state, it will recompute

How fast do routes change?

static: routes change slowly over time

fix the path from 1 router to another router
the determination is very fast,
can be use for some network like
homenetwork: we have just 1
router at home , and this router has only 1 link to
the isp -> routing is very expensive
-> we should use static

dynamic: routes change more quickly

- periodic updates or in response to link cost changes

decentralized: iterative process of computation, exchange of info with neighbors

- routers initially only know link costs to attached neighbors
- "distance vector" algorithms

global or decentralized information?

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

E š Á } Œ l o Ç Œ W ^ } v š Œ } o ‰

- introduction
- routing protocols
  - link state
  - distance vector
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol
- network management, configuration
  - SNMP
  - NETCONF/YANG

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Dijkstra's link-state routing algorithm

find path with least hop in the graph

- *centralized:* network 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 destinations*

notation

- $c_{x,y}$: direct link cost from node *x* to *y*; = ∞ if not direct neighbors -> cost
- *D(v): current* estimate of cost of least-cost-path from source to destination *v*
- *p(v):* predecessor node along path from source to *v*
- *N':* set of nodes whose least-cost-path *definitively* known

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Dijkstra's link-state routing algorithm

```
Initialization:
  N' = {u}                          /* compute least cost path from u to all other nodes */
  for all nodes v
    if v adjacent to u              /* u initially knows direct-path-cost only to direct neighbors */
        then D(v) = c_{u,v}         /* but may not be minimum cost!                                */
    else D(v) = ∞

Loop
    find w not in N' such that D(w) is a minimum
    add w to N'
    update D(v) for all v adjacent to w and not in N' :
        D(v) = min ( D(v),  D(w) + c_{w,v} )
    /* new least-path-cost to v is either old least-cost-path to v or known
    least-cost-path to w plus direct-cost from w to v */
until all nodes in N'
```

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Dijkstra's algorithm: an 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					

Initialization (step 0): For all a: if *a* adjacent to u then $D(a) = c_{u,a}$

find *a* not in *N'* such that *D(a)* is a minimum
add *a* to *N'*
update *D(b)* for all *b* adjacent to *a* and not in *N'* :

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

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Dijkstra's algorithm: an example

resulting *least-cost-path tree* from u:

resulting *forwarding table* in u:

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

route from *u* to *v* directly

route from u to all other destinations via *x*

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Dijkstra's algorithm: another example

Step	N'	v D(v), p(v)	w D(w), p(w)	x D(x), p(x)	y D(y), p(y)	z 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:

- *f* construct *least-cost-path tree* by tracing predecessor nodes
- *f* *ties* can exist (can be broken arbitrarily)

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Dijkstra's algorithm: discussion

algorithm complexity: n nodes

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

message complexity:

- each router must *broadcast* its link state information to other n routers
- efficient (and interesting!) broadcast algorithms: $O(n)$ link crossings to disseminate a broadcast message from one source
- each router's message crosses $O(n)$ links: overall message complexity: $O(n^2)$

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Dijkstra's algorithm: oscillations possible

- when link costs depend on *traffic volume*, *route oscillations* possible
  the metrics such as delay .... change over time
- sample scenario:
  - routing to destination a, traffic entering at d, c, b with rates 1, e (<1), 1
  - link costs are *directional*, and *volume-dependent*

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

When the metrics is traffic -> the path change too fast, the table change more frequently

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

E š Á } Œ l o Ç Œ W ^ } v š Œ } o ‰

- introduction
- routing protocols
  - link state
  - distance vector
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol
- network management, configuration
  - SNMP
  - NETCONF/YANG

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Distance vector algorithm

Based on *Bellman-Ford* (BF) equation (dynamic programming):

Bellman-Ford equation

Let $D_x(y)$: cost of *least-cost path* from x to y.
Then:

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

min taken over all neighbors v of x

direct cost of link from x to v

v's estimated least-cost-path cost to y

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Bellman-Ford Example

Suppose that *u*'s neighboring nodes, *x*,*v*,*w*, know that for destination *z*:

Bellman-Ford 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 (x) is next hop on estimated least-cost path to destination (z)

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

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 any 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* $d_x(y)$

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Distance vector algorithm:

each node:

wait for (change in local link cost or msg from neighbor)

recompute DV estimates using DV received from neighbor

if DV to any destination has changed, ***notify*** neighbors

iterative, asynchronous: each local iteration caused by:

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

distributed, self-stopping: each node *notifies neighbors only* when its *DV changes*

- neighbors then notify their neighbors – *only if necessary*
- no notification received; no actions taken!

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Distance vector: example

t=0

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

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

A few asymmetries:
- missing link
- larger cost

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Distance vector example: iteration

t=1

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

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

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

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

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

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Distance vector example: iteration

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

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

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

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Distance vector example: iteration

…. and so on

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

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Distance vector example:

t=1

- b receives DVs from a, c, e

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:

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

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Distance vector example:

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:

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

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:

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

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Distance vector example:

t=1

f 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:

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

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Distance vector example: c

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:

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

t=1

- c receives DVs from b computes:

$D_c(a) = \min\{c_{c,b}+D_b(a\}\} = 1 + 8 = 9$

$D_c(b) = \min\{c_{c,b}+D_b(b)\} = 1 + 0 = 1$

$D_c(d) = \min\{c_{c,b}+D_b(d)\} = 1+ \infty = \infty$

$D_c(e) = \min\{c_{c,b}+D_b(e)\} = 1 + 1 = 2$

$D_c(f) = \min\{c_{c,b}+D_b(f)\} = 1+ \infty = \infty$

$D_c(g) = \min\{c_{c,b}+D_b(g)\} = 1+ \infty = \infty$

$D_c(h) = \min\{c_{bc,b}+D_b(h)\} = 1+ \infty = \infty$

$D_c(i) = \min\{c_{c,b}+D_b(i)\} = 1+ \infty = \infty$

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/

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Distance vector example: c

t=1

f e receives DVs from b, d, f, h

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 d:

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

DV in e:

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

Q: what is new DV computed in e at *t=1*?

DV in h:

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

DV in f:

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

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

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 **d**,**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**, **d**, **f**, **h** and now at **g**,**i** as well

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Distance vector: link cost changes

link cost changes:

- *f* node detects *local link cost change*
- *f* updates routing info, *recalculates local DV*
- *f* 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*.

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Distance vector: link cost changes

link cost changes:

need more time to converge
the algo can fo to inf

- *f*node detects *local link cost change*
- *f*"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***.
  - …
- *f*see text for solutions. *Distributed algorithms are tricky!*

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Comparison of LS and DV algorithms

message complexity

- LS: n routers, $O(n^2)$ messages sent
- DV: exchange between neighbors; *convergence time varies*

speed of convergence

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

robustness: what happens if router *malfunctions*, or is *compromised*?

LS:

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

DV:

- DV router can advertise *incorrect path cost* ("I have a *really* low-cost path to everywhere"): black-holing
- each router's table used by others: *error propagate through network*

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

E š Á } Œ l o Ç Œ W ^ } v š Œ } o ‰

- introduction
- routing protocols
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol
- network management, configuration
  - SNMP
  - NETCONF/YANG

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Making routing scalable

our routing study thus far - idealized

- all routers identical
- network "flat"

… *not true in practice*

scale: billions of destinations:

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

divide internet into different domains

administrative autonomy:

- **Internet**: a network of networks
- each network admin may want to *control routing* in its own network

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Internet approach to scalable routing

in each AS, implement different routing protocols

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

intra-AS (aka "intra-domain"): routing among *within same AS ("network")*

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

need to run multiple protocols

inter-AS (aka "inter-domain"): routing *among AS'es*

- *gateways* perform inter-domain routing (as well as intra-domain routing)

Copyright© 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Interconnected ASes

forwarding table configured by *intra-* and *inter-AS routing* algorithms

- *intra-AS routing* determine entries for destinations within AS
- *inter-AS & intra-AS* determine entries for external destinations

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Inter-AS routing: a role in intradomain forwarding

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

AS1 inter-domain routing must:

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

Copyright © 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Inter-AS routing: routing within an AS

most common *intra-AS routing protocols*:

- RIP: Routing Information Protocol [RFC 1723]
  - classic **DV**: DVs exchanged every *30 secs*
  - *no longer widely used*
- EIGRP: Enhanced Interior Gateway Routing Protocol
  - DV based
  - formerly *Cisco-proprietary* for decades (became *open* in 2013 [RFC 7868])
- OSPF: Open Shortest Path First [RFC 2328]
  - *link-state routing*
  - IS-IS protocol (ISO standard, not RFC standard) essentially same as OSPF

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

OSPF (Open Shortest Path First) routing

- *"open"*: publicly available
- classic *link-state*
  - each router floods **OSPF** *link-state advertisements* (directly *over IP* rather than using TCP/UDP) to all other routers in entire AS
  - multiple *link costs metrics* possible: *bandwidth*, *delay*
  - each router has *full topology*, uses *Dijkstra's algorithm* to compute forwarding table
- *security:* all OSPF messages *authenticated* (to prevent malicious intrusion)

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

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

boundary router: connects to other ASes

backbone router: runs OSPF limited to backbone

Copyright © 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

E šÁ}Œl o Ç ŒW ^ }všŒ}o ‰

- introduction
- routing protocols
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol
- network management, configuration
  - SNMP
  - NETCONF/YANG

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Internet inter-AS routing: BGP

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

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

eBGP, iBGP connections

gateway routers run both eBGP and iBGP protocols

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

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*)
- e.g., when **AS3** gateway **3a** advertises **path AS3,X** to **AS2** gateway **2c**:
  - **AS3** *promises* to **AS2** it will forward datagrams towards **X**

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Path attributes and BGP routes

- BGP advertised **route = prefix + attributes**
  - *prefix*: destination being advertised
  - 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 neighboring ASes

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

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

Copyright © 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

BGP path advertisement (more)

It can base on attribute, 1c can decide which root
to use, if it choose direct -> AS3, X
otherwise AS3, AS2, X
attribute is involved in the 1c

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

Copyright © 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

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

Copyright© 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

BGP path advertisement

dest	interface
«	«
1c	1
X	1
«	«

- 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**

TO reach X, 1d will use interface 1 (output port number 1)

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

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**
- at **1a**: OSPF intra-domain routing: to get to **1c**, use interface **2**
- at **1a**: to get to **X**, use interface **2**

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Why different Intra-, Inter-AS routing ?

policy:

- *finter-AS*: admin wants control over how its traffic routed, who routes through its network
- *fintra-AS*: single admin, so policy less of an issue

scale:

- *fhierarchical routing* saves table size, reduced update traffic

performance:

- *fintra-AS*: can focus on *performance*
- *finter-AS*: *policy* dominates over performance

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

rthis path is very long, so 2d addopt for next hop attribute, it will decide to use the root with shortest path within AD
but if use AS1, AS3, X -> the root can be very long (through 2 ASes)
--> choose the shortest one

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

BGP: achieving policy via advertisements

has not much device with high capacity, the path will not be x -> B -> C (because B -> C can cause traffic) --> prevent becomming the transit

ISP only wants to route traffic to/from its customer networks (does not want to carry *transit traffic* between other ISPs – a typical "real world" policy)

- **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**

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

BGP: achieving policy via advertisements (more)

ISP only wants to route traffic to/from its customer networks (does not want to carry *transit traffic* between other ISPs – a typical "real world" policy)

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

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

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*

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

E šÁ}Œl o Ç ŒW ^ }všŒ}o ‰

- introduction
- routing protocols
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol
- network management, configuration
  - SNMP
  - NETCONF/YANG

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Software defined networking (SDN)

- *Internet network layer*: historically, implemented via *distributed, per-router control* 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*, ..

the router has 2 function: forwading and routing, to add more function, we need middle box

- ~2005: renewed interest in rethinking network control plane

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Per-router control plane

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

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Software-Defined Networking (SDN) control plane

Remote controller computes, installs forwarding tables in routers

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

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
  - foster innovation: let 1000 flowers bloom

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

SDN analogy: mainframe to PC revolution

Vertically integrated
Closed, proprietary
Slow innovation
Small industry

Horizontal
Open interfaces
Rapid innovation
Huge industry

* Slide courtesy: N. McKeown

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Traffic engineering: difficult with traditional routing

Q: what if network operator wants *u-to-z traffic* to *flow* along ***uvwz***, rather than ***uxyz***?

A: need to re-define *link weights* so traffic routing algorithm computes routes accordingly (or need a new routing algorithm)!

link weights are only control "knobs": not much control!

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Traffic engineering: difficult with traditional routing

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)

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Traffic engineering: difficult with traditional routing

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

A: can't do it (with destination-based forwarding, and LS, DV routing)

We learned in Chapter 4 that *generalized forwarding* and ***SDN*** can be used to *achieve any routing desired*

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Software defined networking (SDN)

***4.** programmable control applications*

***3.** control plane functions external to data-plane switches*

***2.** control, data plane separation*

CA will report some inpformation about the infrastructe (any link failed v.v) -> so that the controller will have the global view, compute shortest path from 1 port to another port ..., it will translate information into flow table for every router in the infrastructure, it can push the flow table to the router. Now the router use the flow table as forwarding table

***1:** generalized "flow-based" forwarding (e.g., **OpenFlow**)*

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Software defined networking (SDN)

Data-plane switches:

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

framework can be used to implement
for communication between switches and routers...

SDN-controlled switches

Copyright © 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Software defined networking (SDN)

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

Copyright© 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Software defined networking (SDN)

network-control apps:

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

Copyright © 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Components of SDN controller

interface layer to network control apps: abstractions API

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

communication: communicate between SDN controller and controlled switches

SDN controller

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

OpenFlow protocol

OPENFLOW: for how to intergrate a switch to the network, how to build the controller, how to use protocol to communicate from the switch to the controler, specify on using TCP pro

- operates between *controller*, *switch*
- **TCP** used to exchange messages
  - optional *encryption*
- three classes of *OpenFlow messages*:
  - *controller-to-switch*
  - *asynchronous* (*switch to controller*)
  - *symmetric* (misc.)
- distinct from **OpenFlow API**
  - API used to *specify generalized forwarding actions*

OpenFlow Controller

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

OpenFlow: controller-to-switch messages

Key <u>controller-to-switch messages</u>

- *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

OpenFlow Controller

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

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.

OpenFlow Controller

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

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

SDN: control/data plane interaction example

① **S1**, experiencing link failure uses 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* accesses network graph info, link state info in controller, *computes new routes*

Copyright © 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

SDN: control/data plane interaction example

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

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

OpenDaylight (ODL) controller

Service Abstraction Layer:

- *f* interconnects internal, external applications and services

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

ONOS 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

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

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*: beyond a single AS
- SDN *critical* in *5G cellular networks*

Copyright© 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

SDN and the future of traditional network protocols

- *SDN-computed* versus *router-computed* forwarding tables:
  - just one example of *logically-centralized-computed* versus *protocol-computed*
- e.g., 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?

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

E šÁ}Œl o Ç ŒW ^ }všŒ}o ‰

- introduction
- routing protocols
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol
- network management, configuration
  - SNMP
  - NETCONF/YANG

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

ICMP: internet control message protocol

- used by *hosts* and *routers* to communicate *network-level information*
  - *error reporting*: unreachable host, network, port, protocol
  - *echo request/reply* (used by *ping*)
- *network-layer "above" IP*:
  - ICMP messages carried in *IP datagrams*

    in network layer , but IP datagrams will carry ICMP message
- ***ICMP message***: type, code plus *first 8 bytes of IP datagram* causing error

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

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Traceroute and ICMP

stopping criteria:

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

- when ICMP message arrives at source: *record RTTs*

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

E šÁ}Œl o Ç ŒW ^ }všŒ}o ‰

- introduction
- routing protocols
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol

- network management, configuration
  - SNMP
  - NETCONF/YANG

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

What is network management?

- autonomous systems (aka "network"): 1000s of interacting hardware/software components
- other complex systems requiring monitoring, configuration, 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."

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Components of network management

use to collect data from manage devices

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

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Network operator approaches to management

CLI (Command Line Interface)

- operator issues (types, scripts) direct to individual devices (e.g., via ssh)

SNMP/MIB

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

use in almost software in network connection

NETCONF/YANG

language for data modeling, how to configure multiple device at the same time

- 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

Copyright © 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

SNMP protocol

manager can send request ti devuce, request for some information, and the inforation in MIP database, the device will provide information

Two ways to convey MIB info, commands:

request/response mode

trap mode

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

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

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

SNMP protocol: message formats

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

- 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

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

YANG

tamodeling language use to specify the syntax, sematic of network manager, folow the xml doc (human readable)

- *data modeling language* used to specify *structure*, *syntax*, *semantics* of NETCONF network management data
  - built-in data types, like SMI
- *XML document* describing device, capabilities can be generated from YANG description
- can express *constraints* among data that must be satisfied by a valid NETCONF configuration
  - ensure NETCONF configurations satisfy correctness, consistency constraints

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

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 (ICMP)
- network management (SNMP/SMI, NETCONF/YANG)

next stop: link layer!

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Network layer, control plane: Done!

- introduction
- routing protocols
  - link state
  - distance vector
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol
- network management, configuration
  - SNMP
  - NETCONF/YANG

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Additional Chapter 5 slides

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Distance vector: another example

$D_x()$

from \ cost to	x	y	z
x	0	2	7
y	,	,	,
z	,	,	,

$D_y()$

from \ cost to	x	y	z
x	,	,	,
y	2	0	1
z	,	,	,

$D_z()$

from \ cost to	x	y	z
x	,	,	,
y	,	,	,
z	7	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

$D_x(z) = \min\{c_{x,y} + D_y(z),\ c_{x,z} + D_z(z)\}$
$= \min\{2+1\ ,\ 7+0\} = 3$

$D_x(y) = \min\{c_{x,y} + D_y(y),\ c_{x,z} + D_z(y)\}$
$= \min\{2+0\ ,\ 7+1\} = 2$

time

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved

Distance vector: another example

$D_x()$

from \ cost to	x	y	z
x	0	2	7
y	,	,	,
z	,	,	,

$D_y()$

from \ cost to	x	y	z
x	,	,	,
y	2	0	1
z	,	,	,

$D_z()$

from \ cost to	x	y	z
x	,	,	,
y	,	,	,
z	7	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

time

Copyright® 1996-2020, J.F Kurose and K.W. Ross, All Rights Reserved