Chapter 5 Network Layer: Control Plane

Sunday, April 25, 2021 2:14 PM

5.1 Intro

- how forwarding and flow tables are computed, maintainted and installed
  - **Per-router control:** routing algorithm runs in each router, fwd and routing func contained in each router

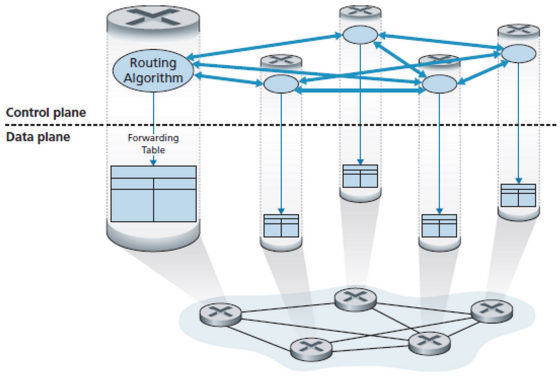
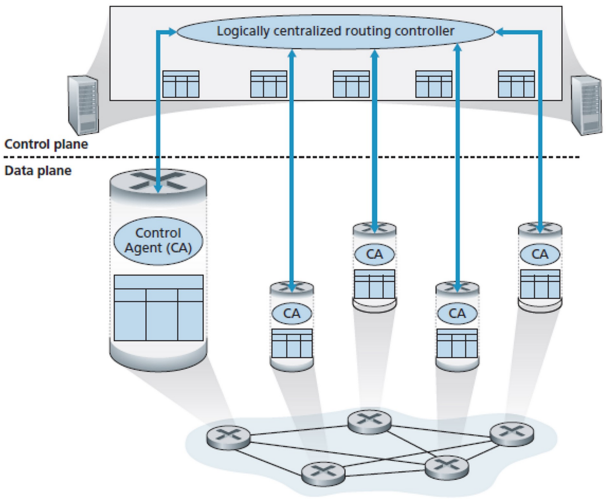
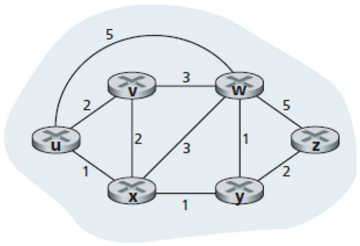


Figure 5.1 ♦ Per-router control: Individual routing algorithm components interact in the control plane

  - **Logically centralized control:** logically centrlalized controller computes and distributes the fwd tables to be used by each router



e 5.2 ♦ Logically centralized control: A distinct, typically remote, controller interacts with local control agents (CAs)



Abstract graph model of a computer network

5.2 Routing Algorithms

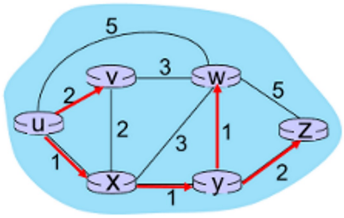
- **Routing protocols:**
  - determine good paths/routes from senders to receivers through network for routers
  - has least cost
- Use graph for routing problesm, G = (N,E)
  - N = set of routers {u, v, w, x, y, z}
  - E = set of links {(u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z)}
- **Classification**
  - **Centralized/Global**
    - All routers have complete topology, link cost info
  - **Decentralized**
    - iteratice process of computation, exchange info with neighbors
    - routers initially only know link costs to attached neighbors
    - distance vector algorithms
  - **Static**
    - routes change slowly over time
  - **Dynamic**

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

5.2.1 Link-State (LS) Routing Algorithm

- **Dijkstra's algorithm**
  - computes least-cost path from one node (source u) to all other nodes in network
  - iterative, after k iterations, know least cost path to k destinations
  - **Notation**
    - **D(v):** cost of least cost path from source node u to dest v during iteration
    - **p(v):** previous node along current least cost path from source to v
    - **N':** subset of nodes. v is in N' if least cost path from source to v is known

```
1 Initialization:
2   N' = {u}                      /* compute least cost path from u to all other nodes */
3   for all nodes v
4     if v adjacent to u          /* u initially knows direct-path-cost only to direct neighbors */
5       then D(v) = c_{u,v}       /* but may not be minimum cost! */
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 least-path-cost to v is either old least-cost-path to v or known
14    least-cost-path to w plus direct-cost from w to v */
15  until all nodes in N'
```



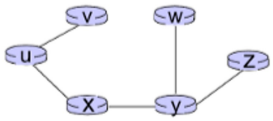
step	N'	D(v), p(v)	D(w), p(w)	D(x), p(x)	D(y), p(y)	D(z), p(z)
0	u	2, u	5, u	1,u	∞	∞
1	ux	2, u	4, x		2, x	∞
2	uxy	2, u	3, y			4, y
3	uxyv		3, y			4, y
4	uxyvw					4, y
5	uxyvwz					

Initialization (step 0): For all a: if *a* adjacent to 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} )
```

Table 5.1 ♦ Running the link-state algorithm on the network in Figure 5.3

resulting least-cost-path tree from u:



resulting forwarding table in u:

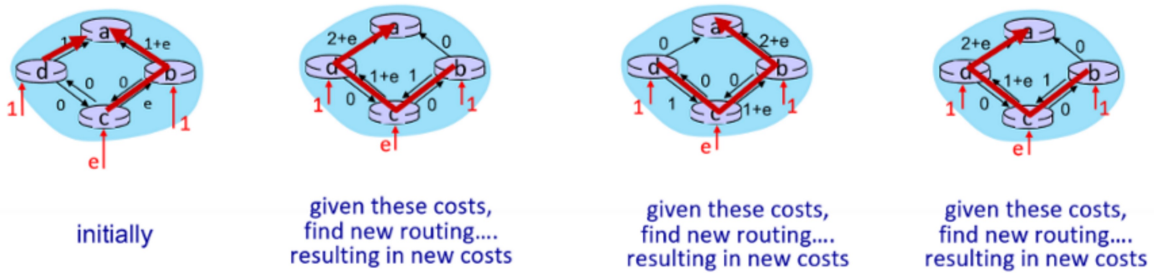
destination	outgoing link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
x	(u,x)

route from *u* to *v* directly

route from u to all other destinations via *x*

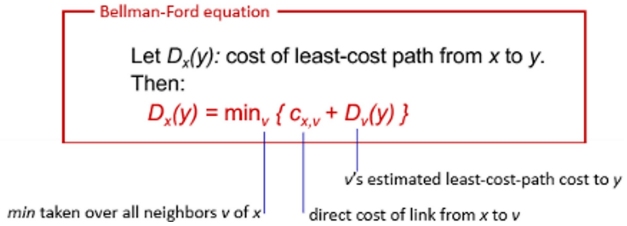
- **0 step initialization**
  - least cost paths (direct one hop) from u to neighbors v,x,w are 2,1,5
  - y and z are inf since not directly connected
  - least cost is x at 1, add to N' set
  - Line 12 of LS algorithm update D(v)
- **1 step**
  - cost of path to v unchanged (least cost is still 2 from u, rather than 3 u,x)
  - cost of path to w is 4 (was 5, but least cost is 4 through x)
  - cost of path to y is 2
  - least cost is y at 2, add to N' set
- **2 step**
  - nodes v and y have least cost path, break tie arbitrarily, add y to set
  - cost to remaining nodes not yet in N' are v,w,z
- **Complexity**
  - **Algorithm:** n nodes, O(n^2) complexity, with n(n+1)/2 comparisons
  - **Message:** each router must broadcast link info to other n routers, crosses O(n) links, overall msg complexity = O(n^2)

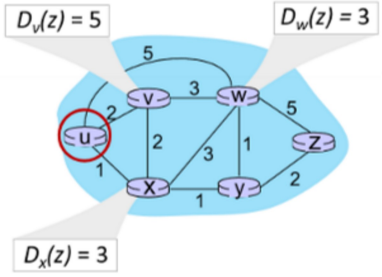
- **Oscillations**
  - when link consts depend on traffic volume, route oscillations possible
  - routing to dest a, traffic entering at d,c,e rates 1, e(<1), 1
  - **Prevent Oscillations:**
    - mandate that link costs not depend on amount of traffic carried (unnacceptable since need to avoid congestion links)
    - ensure that not all routers run the LS algorithm at same time



5.2.2 The Distance-Vector (DV) Routing Algorithm

- iterative async and distributed
  - each node receives some info from one or 2 of neighbors, calculates, then distributes results
  - process continues until no more info exchanged
  - does not require all of nodes to operate in lockstep with each other
- Least cost d_x(y) related by **Bellman ford equation**



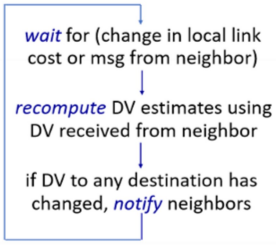


Bellman-Ford equation says:

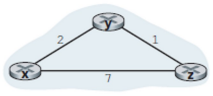
$$D_u(z) = \min \{ c_{u,v} + D_v(z), c_{u,x} + D_x(z), c_{u,w} + D_w(z) \}$$
$$= \min \{2 + 5, 1 + 3, 5 + 3\} = 4$$

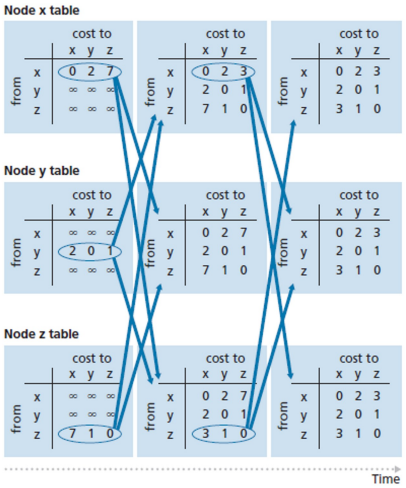
  - provides entries in node x's forwarding table
  - node achieving min (x) is next hop on estimated least cost path to dest (z)
- each node sends its own distacne vector estimate to neighbors
- when x receives new DV estimate from neighbors, updates its own DV using BF eqn

each node:




```
1  Initialization:
2    for all destinations y in N:
3       D_x(y) = c(x,y) /* if y is not a neighbor then c(x,y) = ∞ */
4    for each neighbor w
5       D_w(y) = ? for all destinations y in N
6    for each neighbor w
7       send distance vector  D_x = [D_x(y): y in N] to w
8
9  loop
10     wait   (until I see a link cost change to some neighbor w or
11              until I receive a distance vector from some neighbor w)
12
13     for each y in N:
14         D_x(y) = min_v{c(x,v) + D_v(y)}
15
16 if D_x(y) changed for any destination y
17       send distance vector D_x = [D_x(y): y in N] to all neighbors
18
19 forever
```





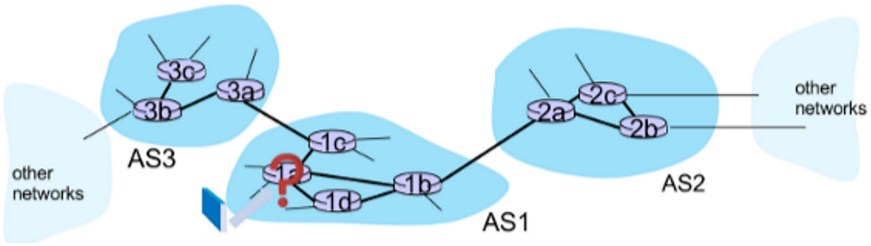
5.6 ♦ Distance-vector (DV) algorithm in operation

- **LS vs DV algorithms**
  - message complexity
    - LS: n routers, O(n^2) msgs sent
    - DV: exchange bw neighbors, convergence varies
  - speed of convergence
    - LS: O(n^2) algorithm, O(n^2) msg, may have oscillations
    - DV: convergence time vaires, may have routing loops
  - robustness
    - LS: router can advertise incorrect link cost, router computes own table
    - DV: can advertise incorrect path cost - black holing

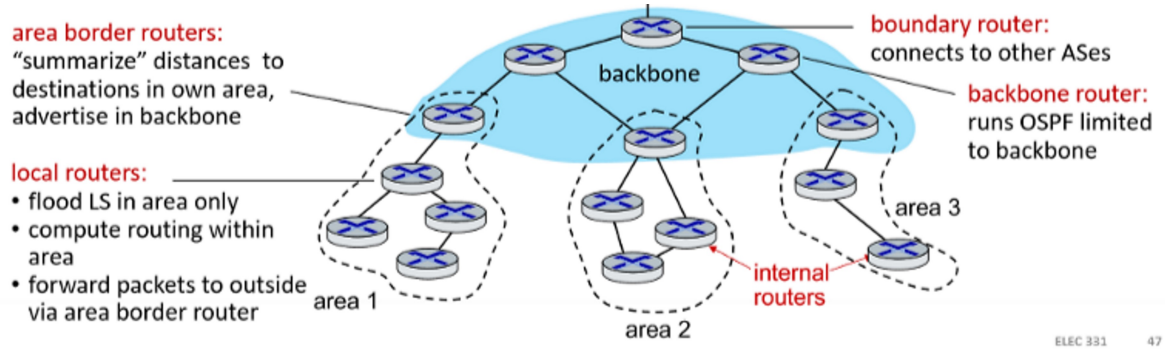
- DV: can advertise incorrect path cost - black holing, each routers table used by others, error propagates

5.2 Intra-AS Routing in the Internet: OSPF

- aggregate routers into regions - Autonomous systems, domains
- **Scale**
  - all routers identical, network flat, but not true in practice
  - cant store all dest in routing tables, exchanges would swamp links
- **administrative autonomy**
  - each network admin may want to control routing in its own network
- **intra-AS**
  - routing among withing same AS
  - all routers must run same intra domain protocol
  - gateway router: at edge of its own AS, has links to routers in other AS
- **inter-AS**
  - routing among AS
  - gatewarys perform inter domain routing
- **Interconnected AS**
  - fwd table configured by intra and inter AS routing algorithms
  - intra AS determine entries for dest within AS
  - inter AS and intra AS determine entries for external dest

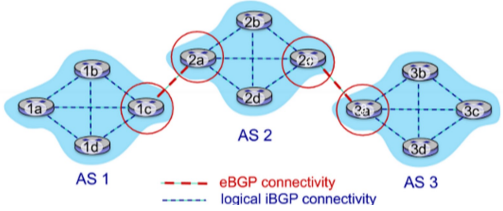


- **Open Shortest Path First (OSPF)**
  - classic link state routing
    - each router floods OSPF link state advertisements
    - each router has full topology, uses Dijkstra's algorithm to compute fwd table
    - all msgs are authenticated
  - two-level hierarchy
    - link state advertisements flooded
    - each node has detailed area topology, only knows direction to reach other dest

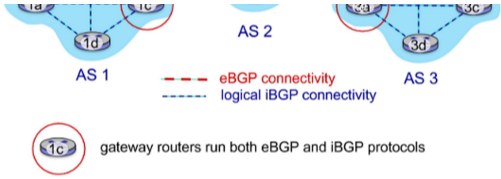


5.4 Routing Among the ISPs: BGP

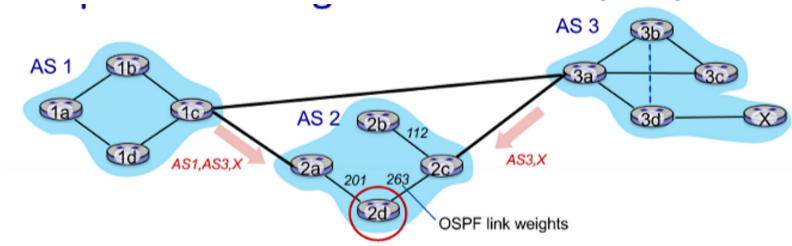
- allows subnet to advertise its existance, and destination it can reach
- BGP Border Gateway protocol
- Provides each AS a means to
  - eBGP: obtain subnet reachability information from neighboring Ases
  - iBGP: propagate reachability info to all AS internal routers
  - determine good routes



- eBGP: obtain subnet reachability information from neighboring Ases
- iBGP: propagate reachability info to all AS internal routers
- determine good routes



- **Hot potato routing**
  - choose local gateway with least intra domain cost
    - ex. 2d chooses 2a even though more AS hops to X
  - tries to reduce cost in its own AS while ignoring other components of the end-to-end costs outside its AS



5.4.5 Routing Policy

- when a router selects a route to a destination, AS routing policy can trump all other considerations, such as shortest AS path or hot potato

- **BGP route selection**
  - router may learn about more than 1 route to destination
  - selects based on:
    - local preference value attribute: policy decision
    - shortest AS-PATH
    - closest NEXT-HOP router: hot potato routing
    - additional criteria

5.5 SDN Control Plane

-