CSc 450/550
Computer Networks
Routing algorithms

Jianping Pan

Summer 2006

Review

- IP
  - addressing and *routing*
    - address class, classless, NAT
  - fragmentation and reassembly

Routing

- Routing algorithms
  - flooding
    - receive from one interface and send to other ifs
    - reduce duplicate packets
      - TTL
      - if received before, drop
      - shortest reverse path
  - link state
  - distance vector

Link state routing

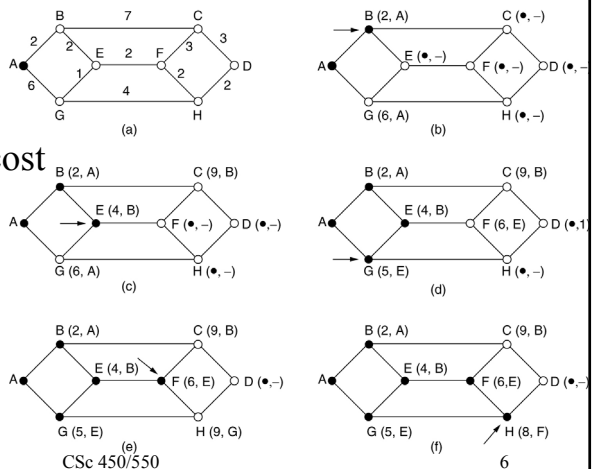
- Neighbor discovery
  - "hello-hello" between directly connected nodes
- Link-state broadcast
  - link state: cost, delay, or other metrics
- Topology generation
  - node/link graph
- Shortest-path calculation
  - from one node to all other nodes

Dijkstra 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'
```


Dijkstra's algorithm: example

- Condition
  - nonnegative link cost



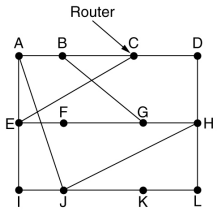
Distance vector routing

- Neighbor discovery
  - "hello-hello" between directly connected nodes
- Route exchange
  - A: "I can reach X at cost Path (A,X)."
  - B: "I can reach X at cost Path (B,X)."
  - A: "I am Link (A,B) away from B."
- Shortest-path calculation
  - A: min{Path (A,X), Link (A,B) + Path (B,X)}

Bellman-Ford algorithm

```
1  Initialization:
2   for all adjacent nodes v:
3      D  (*,v) = infinity        /* the * operator means "for all rows" */
4      D  (v,v) = c(X,v)          /* direct neighbors */
5   for all destinations, y
6      send min  D  (y,w) to each neighbor  /* w over all X's neighbors */
7
8  loop
9    wait (until I receive update from neighbor V)
10
11  if (update received from V wrt destination Y)
12     /* shortest path from V to some Y has changed  */
13     /* V has sent a new value for its  min   DV(Y,w) */
14     /* call this received new value is "newval"     */
15     for the single destination y: D  (Y,V) = c(X,V) + newval
16
17   if we have a new min   D  (Y,w)for any destination Y
18      send new value of min   D  (Y,w) to all neighbors
19
20  forever
```

Bellman-Ford algorithm: example



(a)

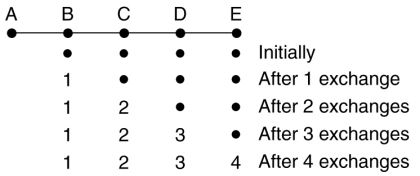
New estimated delay from J

To	A	I	H	K	Line
A	0	24	20	21	8 A
B	12	36	31	28	20 A
C	25	18	19	36	28 I
D	40	27	8	24	20 H
E	14	7	30	22	17 I
F	23	20	19	40	30 I
G	18	31	6	31	18 H
H	17	20	0	19	12 H
I	21	0	14	22	10 I
J	9	11	7	10	0 –
K	24	22	22	0	6 K
L	29	33	9	9	15 K
	JA delay is 8	JI delay is 10	JH delay is 12	JK delay is 6	New routing table for J

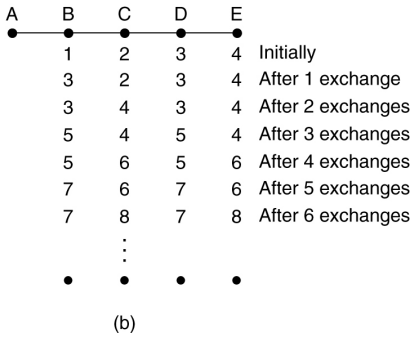
Vectors received from J's four neighbors

(b)

Count-to-infinity problem



(a)



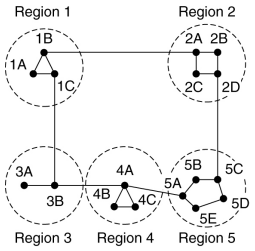
(b)

- Choose a small "infinity"
- Poisoned reverse
  - A: I can reach X through B for cost T
  - A tells B
    - I can reach X for infinity cost, since I reach X through you!
- When "poisoned reverse" fails

Hierarchical routing

- Why hierarchical
  - scalability
- Internet
  - autonomous system (AS)
  - Inter-domain routing
    - distance vector
  - Intra-domain routing
    - distance vector or link state

Hierarchical routing: example



(a)

Full table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

(b)

Hierarchical table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

(c)

This lecture

- Routing algorithms
  - Dijkstra algorithm
  - Bellman-Ford algorithm
- Explore further
  - /bin/netstat -r
  - More on routing
    - multicast (CSc 461), mobile & ad hoc (Topics), peer-to-peer (Topics), etc

Next lecture

- Internet routing
  - read CN Section 5.6