Computer Networks

Routing Algorithms

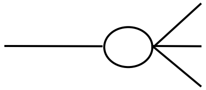
Jianping Pan
Fall 2022

Review

- IP
  - addressing and ***routing***
    - address classes, classless, NAT
  - fragmentation and reassembly
    - identification
    - total length, IP header length, fragment offset
- ICMP
  - also used in ping and traceroute

Forwarding and routing

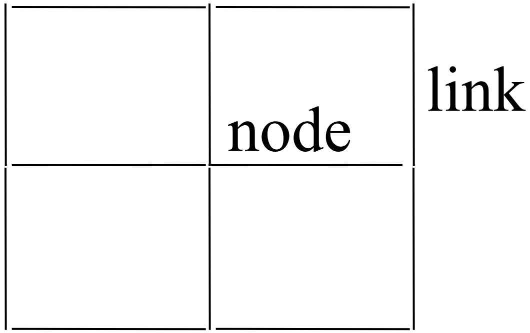
- Internet
  - store-and-forward packet switching
- Forwarding
  - table lookup
    - e.g., destination, next-hop
  - to determine outgoing interface
- Routing
  - to build the table
  - static and dynamic routing



default gateway in end-host

Routing

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



G(V,E): nodes, links

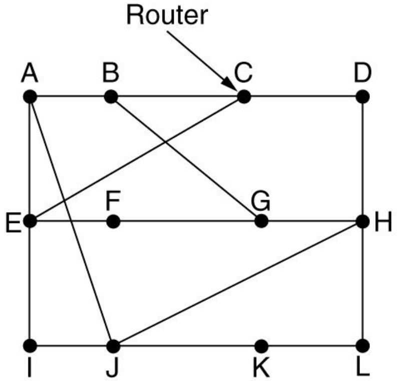
Distance vector routing

- 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: $\min_B$\{Path (A,X), Link (A,B) + Path (B,X)\}

Bellman-Ford algorithm

```
Initialization:
  for all adjacent nodes v:
    D  (*,v) = infinity        /* the * operator means "for all rows" */
    D  (v,v) = c(X,v)          /* direct neighbors */
  for all destinations, y
    send min  D  (y,w) to each neighbor  /* w over all X's neighbors */

loop
  wait (until I receive update from neighbor V)

  if (update received from V wrt destination Y)
    /* shortest path from V to some Y has changed  */
    /* V has sent a new value for its  min   DV(Y,w) */
    /* call this received new value is "newval"     */
    for the single destination y: D  (Y,V) = c(X,V) + newval

  if we have a new min   D  (Y,w) for any destination Y
     send new value of min   D  (Y,w) to all neighbors

forever
```

Bellman-Ford algorithm: example



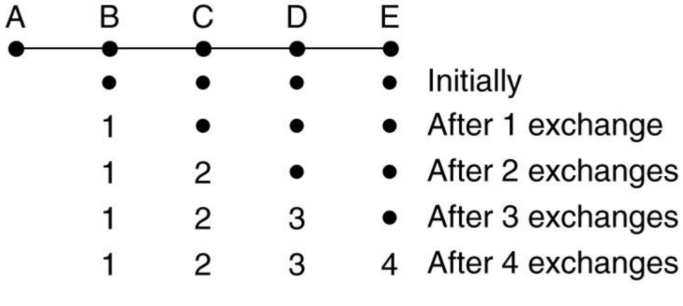
(a)

To	A	I	H	K	New estimated delay from J	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

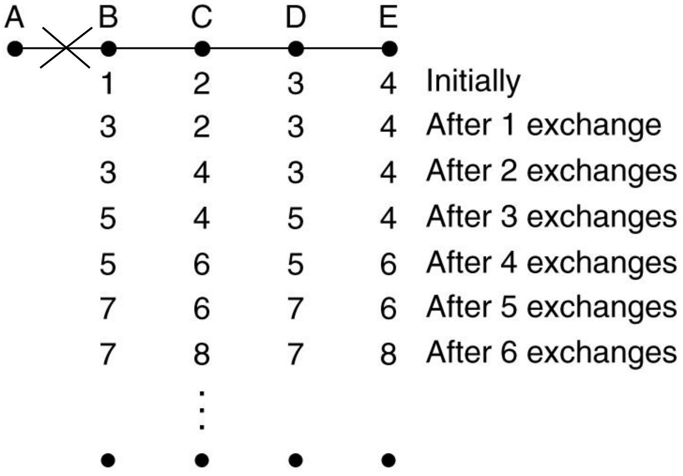
(b)

Count-to-infinity problems



B	C	D	E	
•	•	•	•	Initially
1	•	•	•	After 1 exchange
1	2	•	•	After 2 exchanges
1	2	3	•	After 3 exchanges
1	2	3	4	After 4 exchanges

(a)

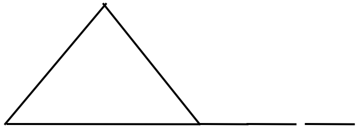


B	C	D	E	
1	2	3	4	Initially
3	2	3	4	After 1 exchange
3	4	3	4	After 2 exchanges
5	4	5	4	After 3 exchanges
5	6	5	6	After 4 exchanges
7	6	7	6	After 5 exchanges
7	8	7	8	After 6 exchanges
	⋮			
•	•	•	•	

(b)

Deal with CTI problems

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



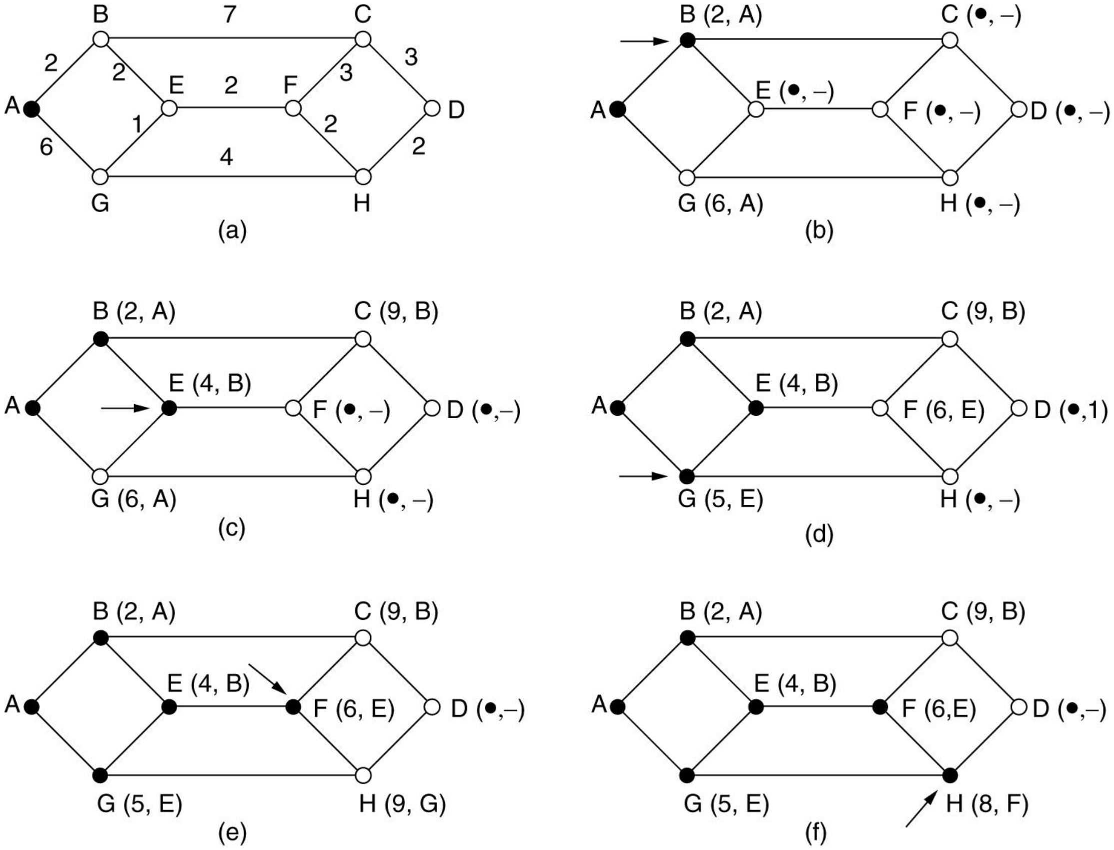
Link state routing

- 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

```
Initialization:
  N' = {u}
  for all nodes v
    if v adjacent to u
        then D(v) = c(u,v)
    else D(v) = $\infty$

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 cost to v is either old cost to v or known
   shortest path cost to w plus cost from w to v */
until all nodes in N'
```

Dijkstra’s algorithm: example



non-negative link cost

DV vs LS routing

- Information exchange
  - DV: just between neighbors
  - LS: among all nodes
- Shortest-path calculation
  - DV: distributed Bellman-Ford
  - LS: Dijkstra
- Pros and cons
  - discussion...