Chapter 4
Network Layer

Reti degli Elaboratori
Canale ALProf.ssa Chiara
 Petrioli
a.a. 2018/2019
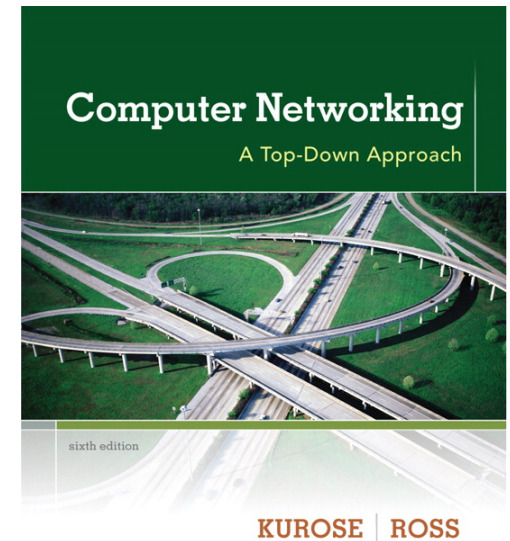
Computer Networking: A Top Down Approach
6th edition
Jim Kurose, Keith Ross
Addison-Wesley
March 2012

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

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

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
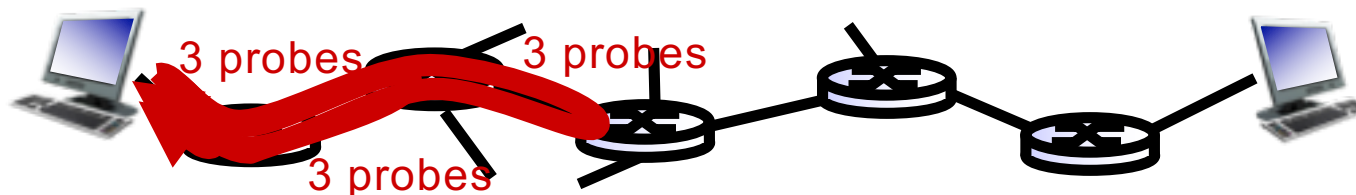
3 probes 3 probes

3 probes

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol
- datagram format
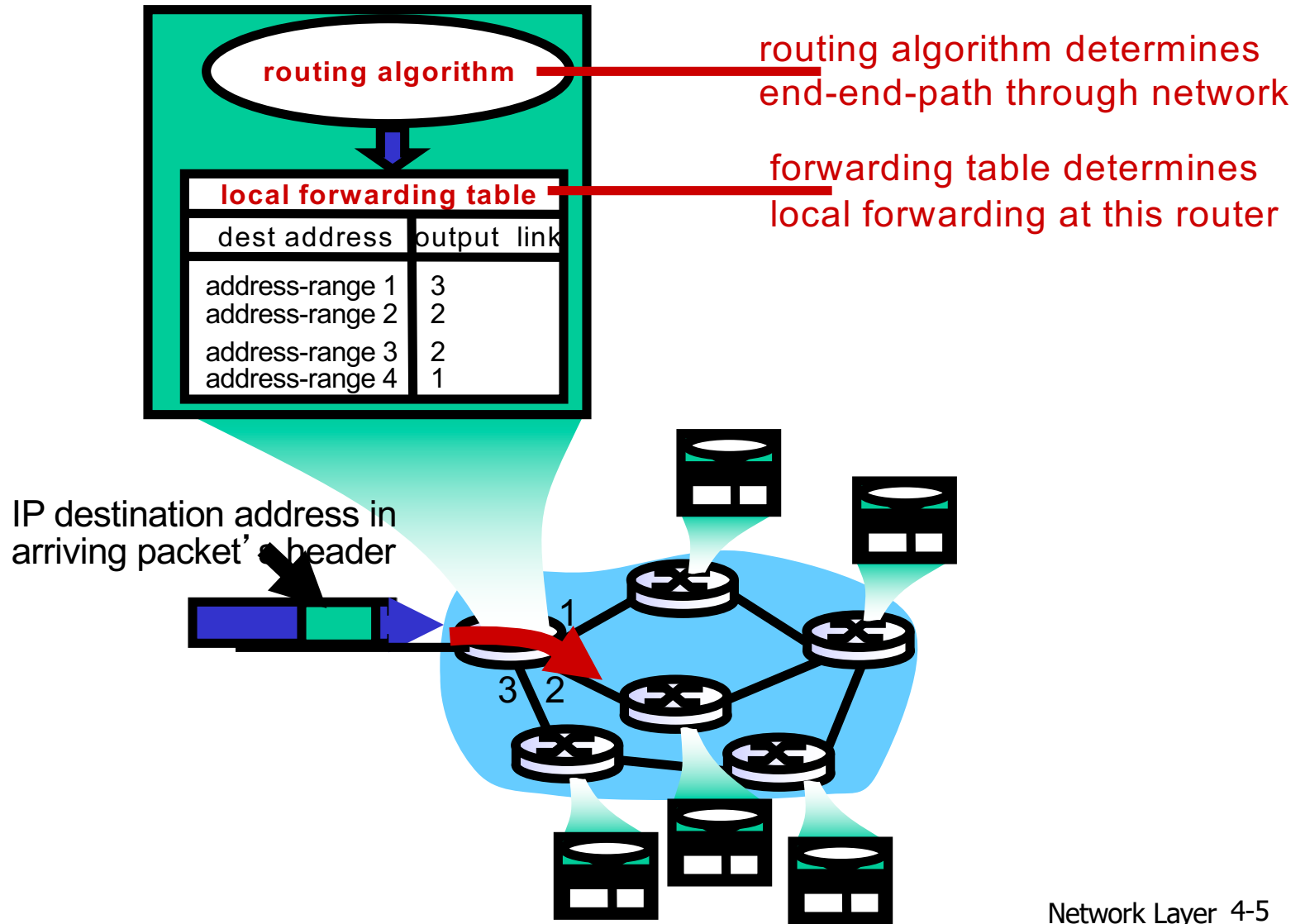- IPv4 addressing
- ICMP
- IPv6

4.5 routing algorithms
- link state
- distance vector
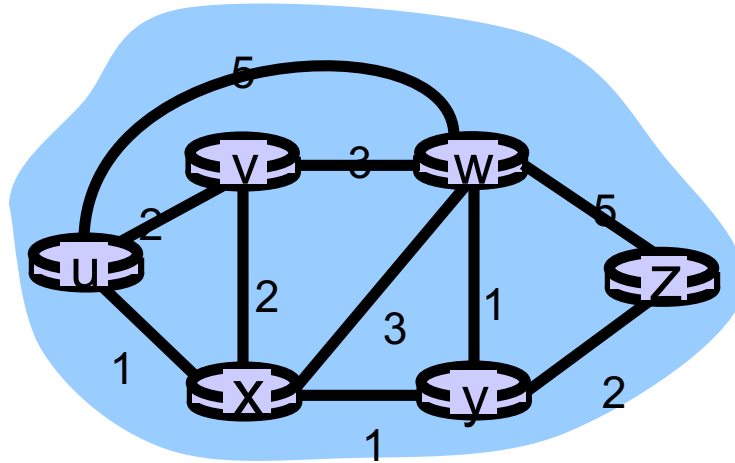- hierarchical routing

4.6 routing in the Internet
- RIP
- OSPF
- BGP

4.7 broadcast and multicast routing

Interplay between routing, forwarding

routing algorithm

local forwarding table

dest address	output link
address-range 1	3
address-range 2	2
address-range 3	2
address-range 4	1

routing algorithm determines
end-end-path through network

forwarding table determines
local forwarding at this router

IP destination address in
arriving packet's header

1

3 2

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

Graph abstraction: costs
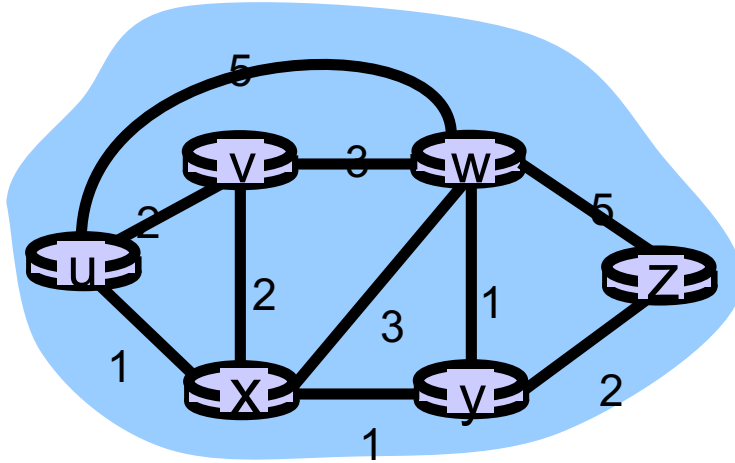
$c(x,x') =$ cost of link (x,x')
e.g., $c(w,z) = 5$

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

Routing algorithm classification

Q: global or decentralized information?

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

decentralized:
- ❖ router knows physically-connected neighbors, link costs to neighbors
- ❖ iterative process of computation, exchange of info with neighbors
- ❖ "distance vector" algorithms

Q: static or dynamic?

static:
- ❖ routes change slowly over time

dynamic:
- ❖ routes change more quickly
  - ▪ periodic update
  - ▪ in response to link cost changes

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol
- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 routing algorithms
- link state
- distance vector
- hierarchical routing

4.6 routing in the Internet
- RIP
- OSPF
- BGP

4.7 broadcast and multicast routing

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

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'
```
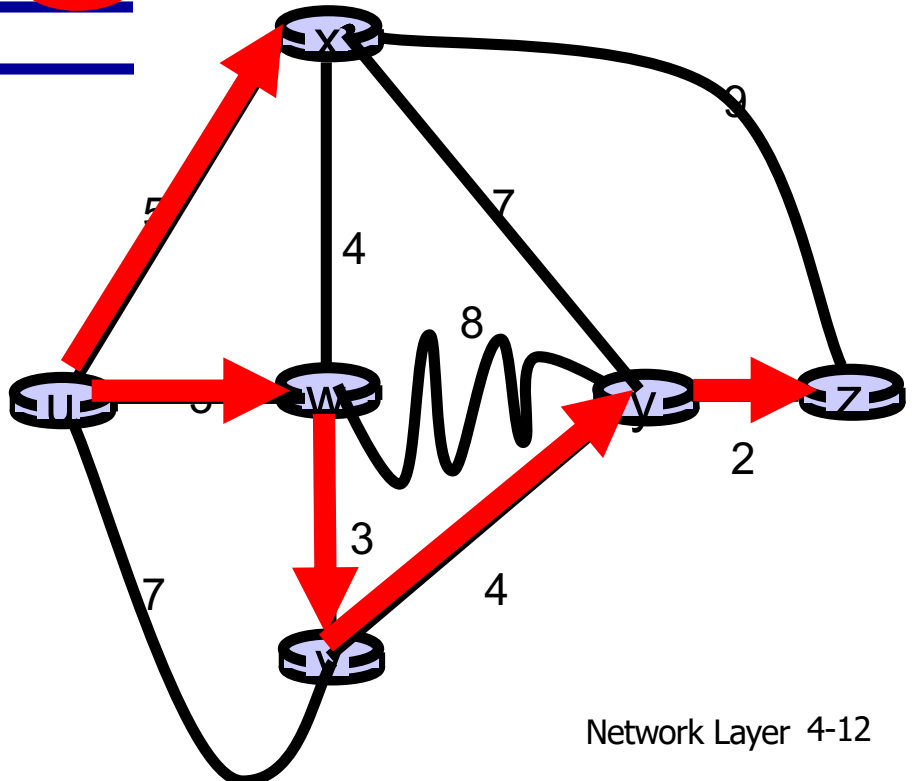
Dijkstra's algorithm: example

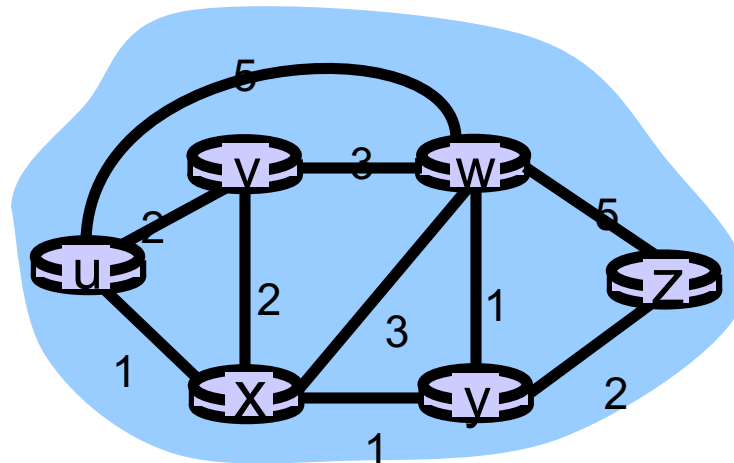| Step	N'	D(v) p(v)	D(w) p(w)	D(x) p(x)	D(y) p(y)	D(z) p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					

notes:

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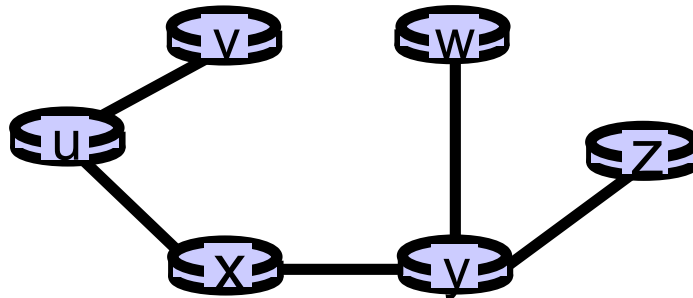
❖ ties can exist (can be broken arbitrarily)

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	uxv	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					

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)

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'
```
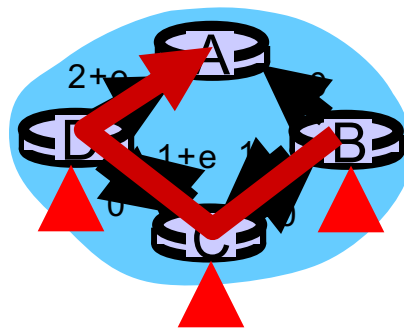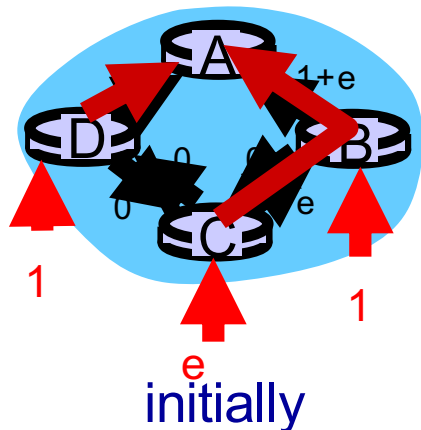
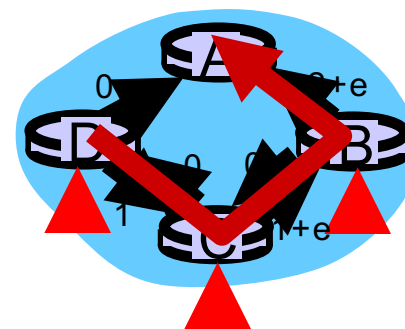# Dijkstra's algorithm, discussion

algorithm complexity: n nodes

❖ each iteration: need to check all nodes, w, not in N

❖ n(n+1)/2 comparisons: $O(n^2)$

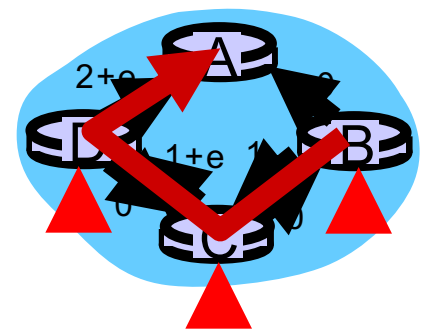❖ more efficient implementations possible: $O(n \log n)$

oscillations possible:

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

1

e
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