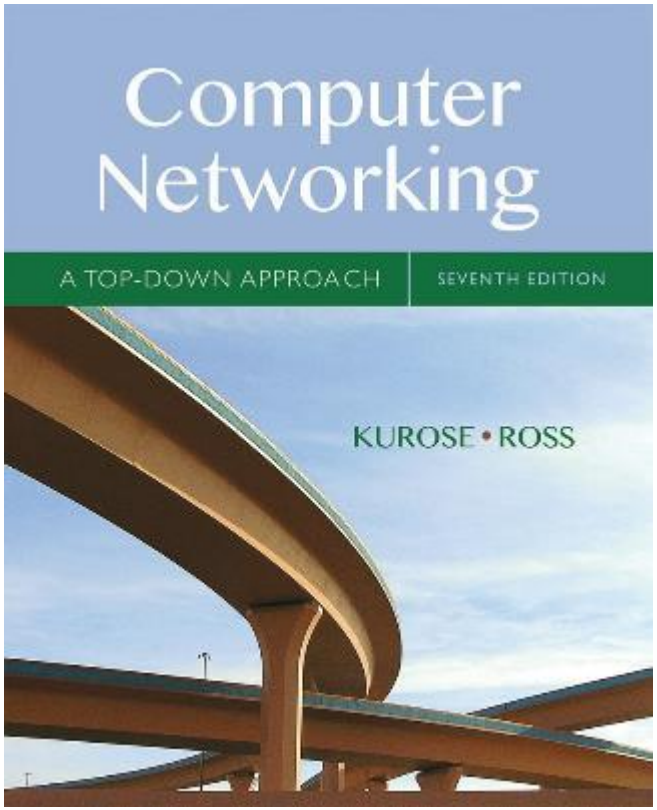
Chapter 5
Network Layer: The Control Plane (Part 1/3)



Computer Networking: A Top Down Approach

7th edition
Jim Kurose, Keith Ross
Pearson/Addison Wesley
April 2016

A note on the use of these Powerpoint slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 ICMP: The Internet Control Message Protocol

Network-layer functions

Recall: two network-layer functions:

- *forwarding:* move packets from router's input to appropriate router output

data plane

- *routing:* determine route taken by packets from source to destination

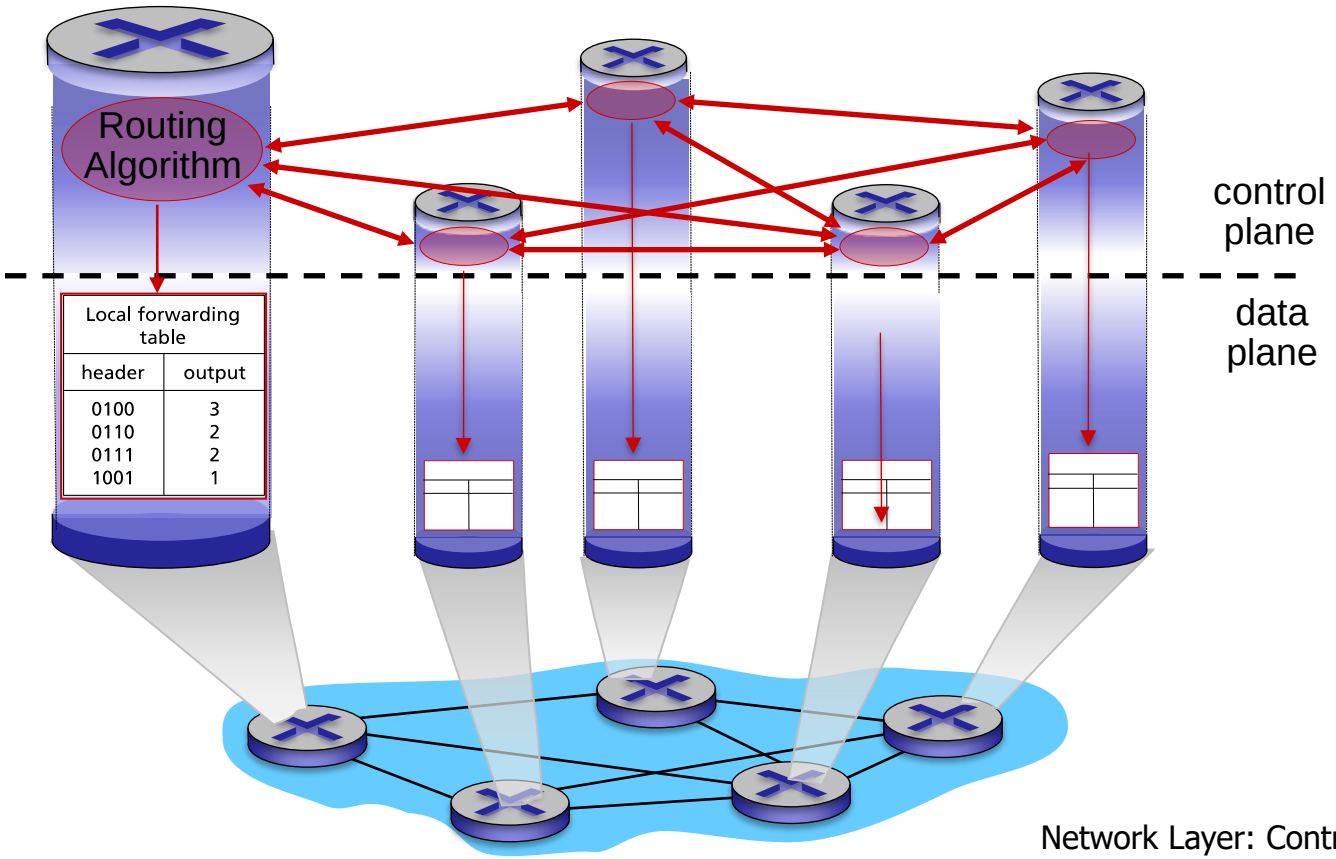
control plane

Two approaches to structuring network control plane:

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

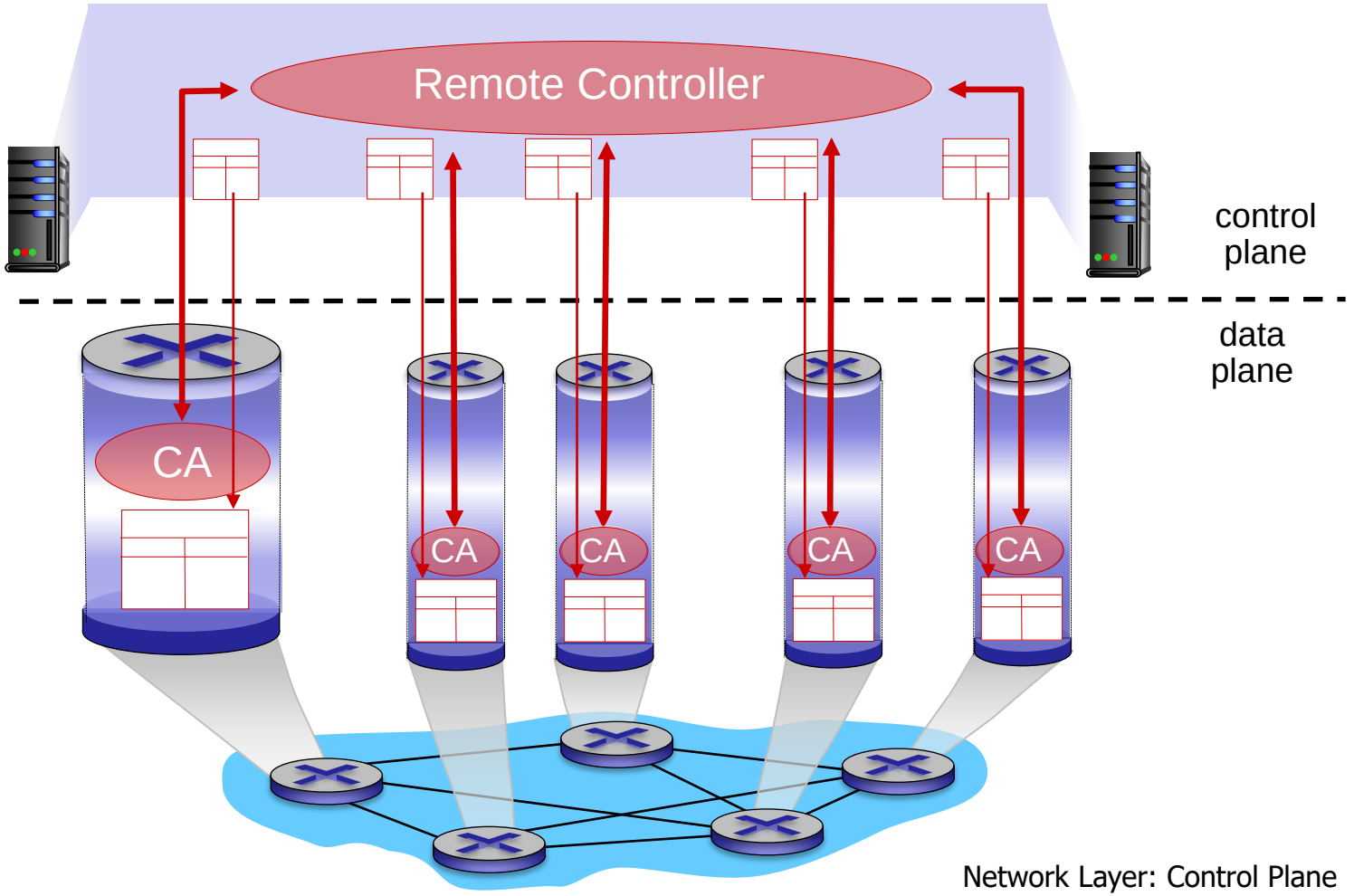
Per-router control plane

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



Logically centralized control plane

A distinct (typically remote) controller interacts with local control agents (CAs) in routers to compute forwarding tables



Chapter 5: outline

5.1 introduction

5.2 routing protocols

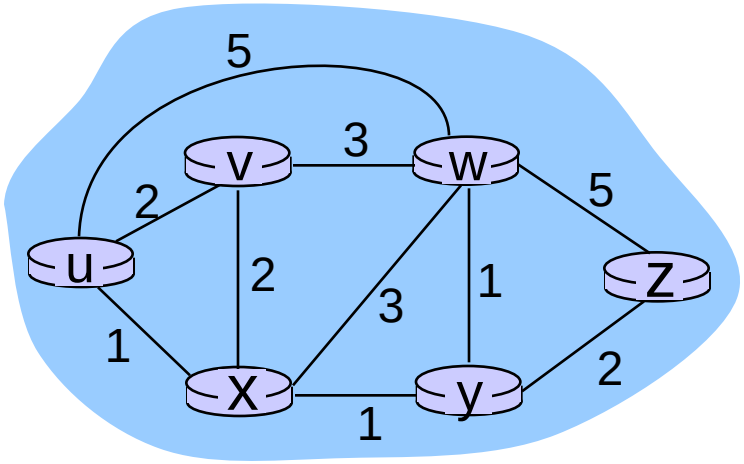
- link state
- distance vector

Routing protocols

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

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

Graph abstraction of the network

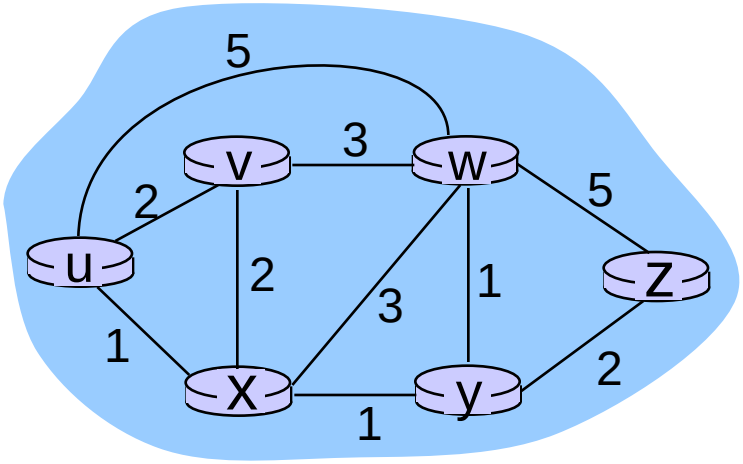


graph: G = (N,E)

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) }

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 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

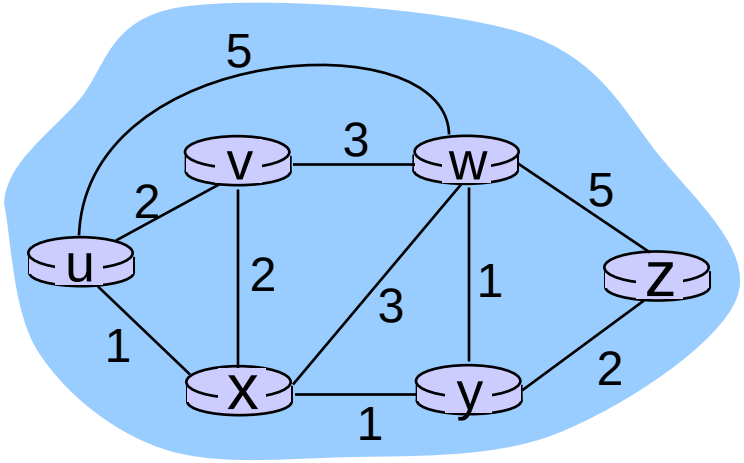
A link-state routing Protocol

Dijkstra's algorithm

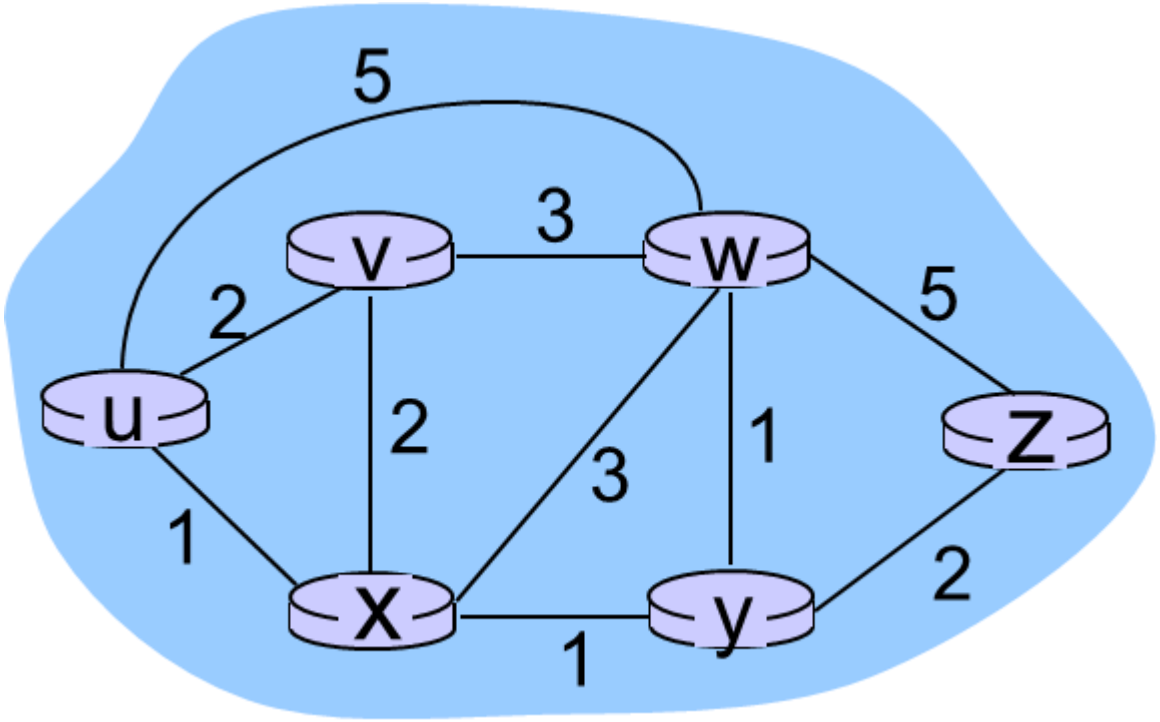
- network topology, link costs known to all nodes
  - accomplished via "link state broadcast"
  - all nodes have same info
- computes least cost paths from one node ("source") to all other nodes
  - gives *forwarding table* for that node
- iterative: after k iterations, know least cost path to k destinations

notation:

- c(x,y): link cost from node x to y if x , y are neighbors else = ∞ if not direct neighbors
- D(v): current value of cost of 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



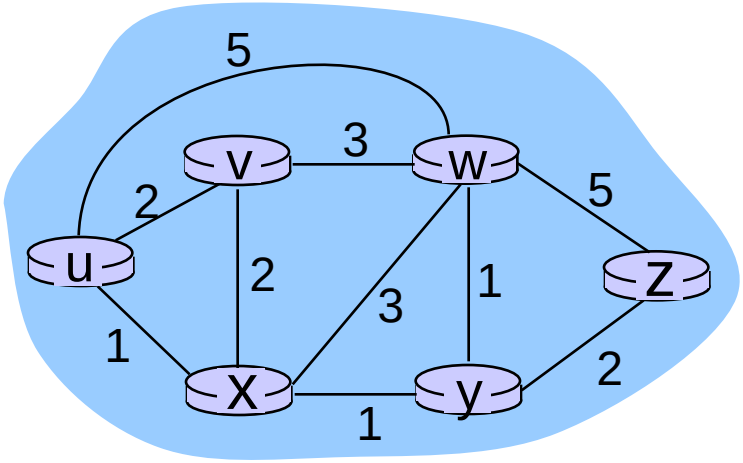
Find the shortest Path using Dijkstra algorithm



Dijkstra's algorithm: another example

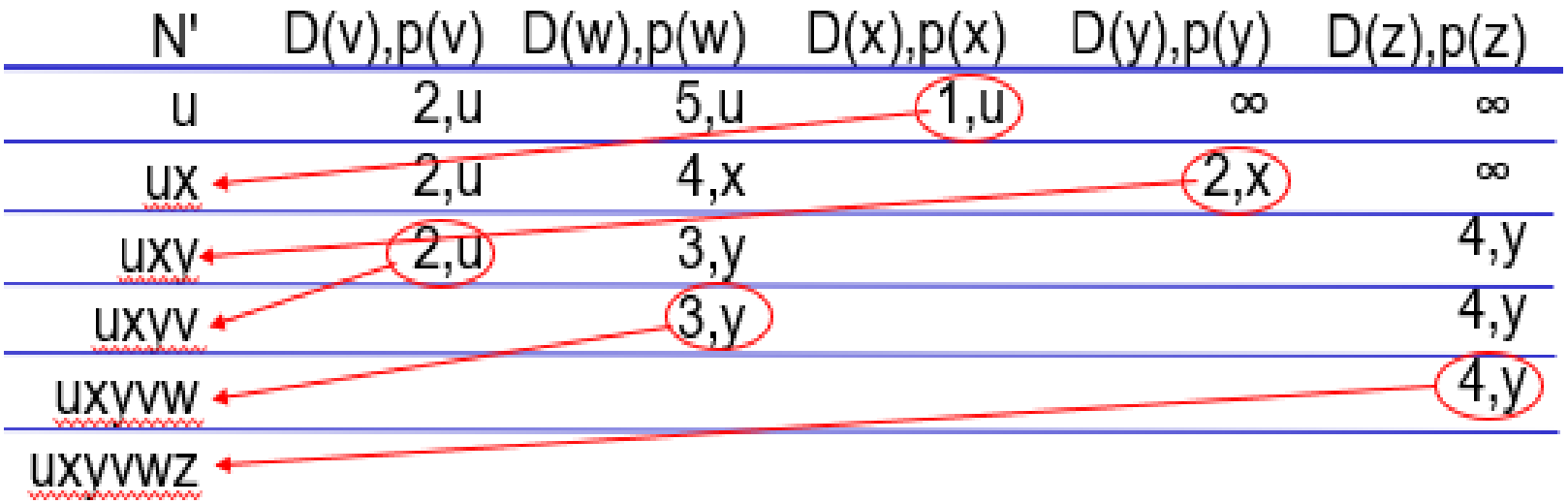
From Source node 'u'

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					



D(v) = min(D(v), D(w) + c(w,v))

Dijkstra's algorithm: example



N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
u	2,u	5,u	1,u	∞	∞
ux	2,u	4,x		2,x	∞
uxv	2,u	3,y			4,y
uxvv		3,y			4,y
uxvvw					4,y
uxvvwz					

resulting shortest-path tree from u?

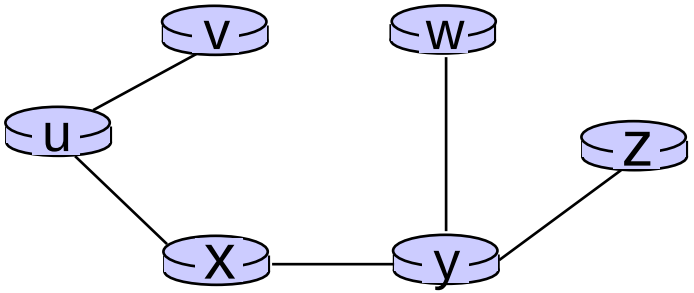
resulting forwarding table in u ?

notes:

- construct shortest path tree by tracing predecessor nodes

Dijkstra's algorithm: example

resulting shortest-path tree from u:



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

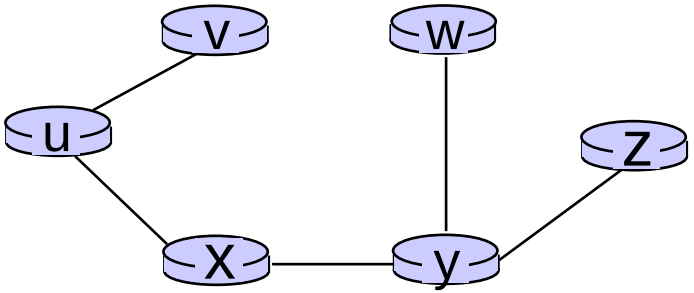
resulting forwarding table in u ?

notes:

- construct shortest path tree by tracing predecessor nodes

Dijkstra's algorithm: example

resulting shortest-path tree from u:



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

resulting forwarding table in u:

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

notes:

- construct shortest path tree by tracing predecessor nodes

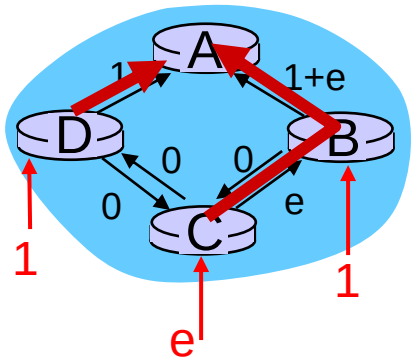
Dijkstra's algorithm, discussion

oscillations possible:

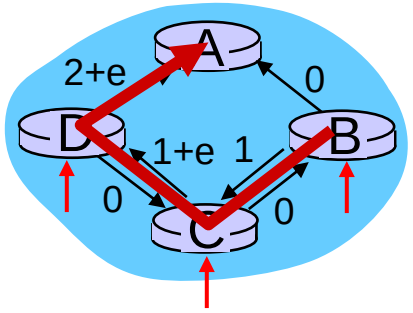
- e.g., 'link cost' = amount of carried traffic:

Possible solution:

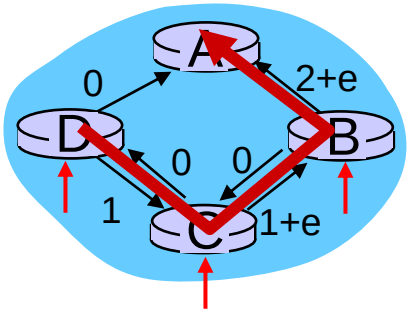
- Make link cost independent of traffic load
- Randomize time of link state advertisement
  - prevent the routers from running the routing algorithm simultaneously. If this is not done carefully, the routers will tend to self-synchronize.



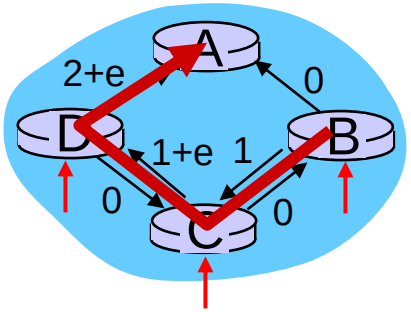
initially



given these costs, find new routing…. resulting in new costs



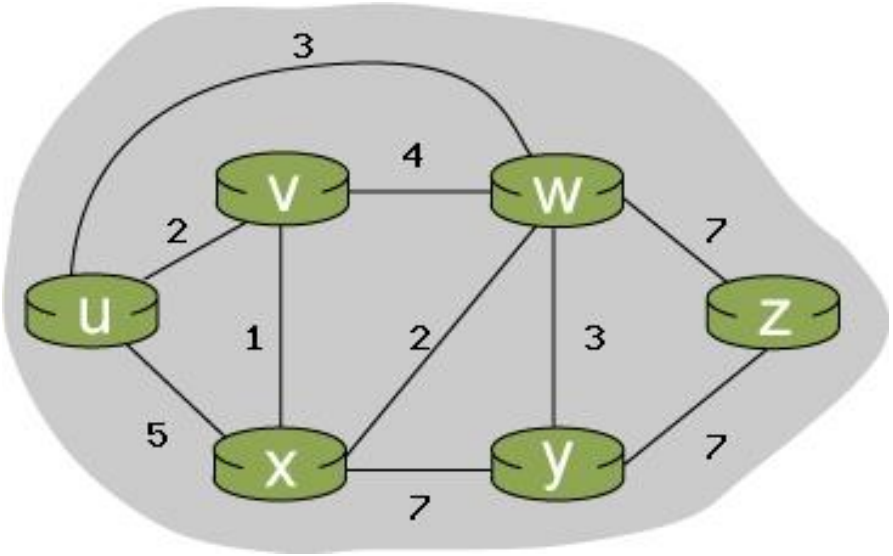
given these costs, find new routing…. resulting in new costs



given these costs, find new routing…. resulting in new costs

Class exercise

Using Dijkstra's algorithm, find the least cost path from source node **u** to all other destinations. Show your work in tabular format

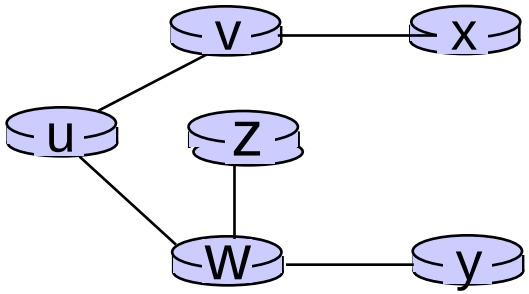


Class exercise (solution)

Using Dijkstra's algorithm, find the least cost path from source node **u** to all other destinations. Show your work in tabular format

D(u),p(u)	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
-	2,u	3,u	5,u	infty	infty
-	-	3,u	3,v	infty	infty
-	-	-	3,v	6,w	10,w
-	-	-	-	6,w	10,w
-	-	-	-	-	10,w
-	-	-	-	-	-

resulting shortest-path tree from u and resulting forwarding table from u:



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

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 ICMP: The Internet Control Message Protocol

Distance vector algorithm

Bellman-Ford equation

let

$d_x(y)$:= cost of least-cost path from x to y

then

$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

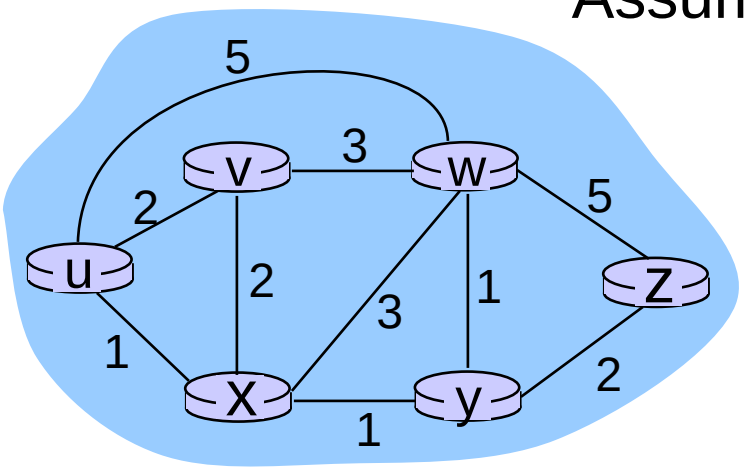
cost from neighbor v to destination y

cost to neighbor v

min taken over all neighbors v of x

Bellman-Ford example

Assume we know $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$



B-F equation says:

$d_x(y) = min_v \{c(x,v) + d_v(y)\}$

To find cost from source u to destination z using B-F equation:

$$d_u(z) = \min \{ c(u,v) + d_v(z),$$
$$c(u,x) + d_x(z),$$
$$c(u,w) + d_w(z) \}$$
$$= \min \{2 + 5,$$
$$1 + 3,$$
$$5 + 3\} = 4$$

node achieving minimum is next hop in shortest path, used in forwarding table

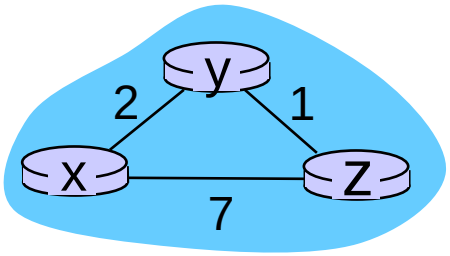
Distance vector algorithm

key idea:

- from time-to-time, each node sends its own distance vector estimate to *neighbors*
- when x receives new DV estimate from *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* $d_x(y)$



node x table

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

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

node y table

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

node z table

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

time

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

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

node x table

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

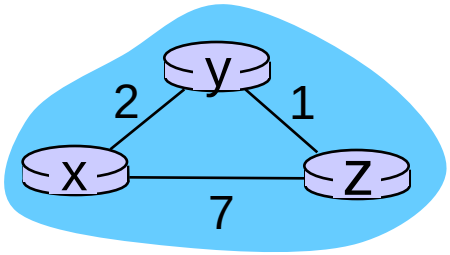
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

node y table

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

node z table

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



time

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

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

node x table

from	cost to x	cost to y	cost to z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

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

node y table

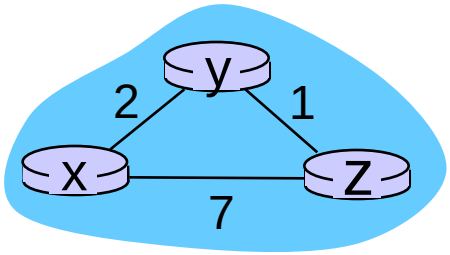
from	cost to x	cost to y	cost to z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

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

node z table

from	cost to x	cost to y	cost to z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

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



time

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

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

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

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

node x table

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

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	3
y	2	0	1
z	3	1	0

node y table

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

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

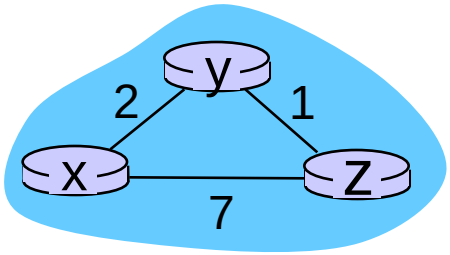
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node z table

from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
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



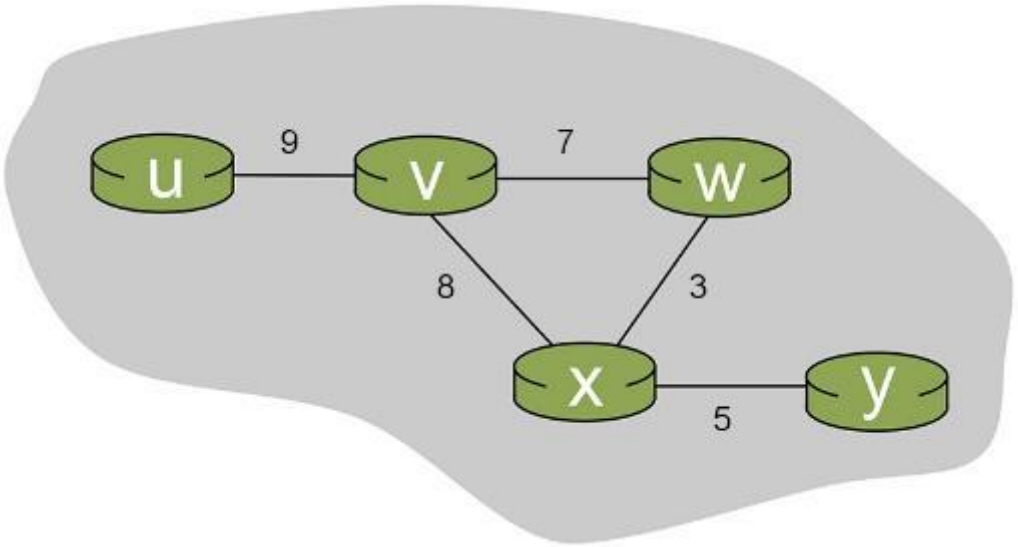
time

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

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

Class Exercise: Bellman Ford Equation

- When the algorithm converges, what are the distance vectors in each of the routers.



Class Exercise: Bellman Ford Equation

- When the algorithm converges, what are the distance vectors in each of the routers.

- The initial distance vectors at each router will be:

  Du = [0,9,∞,∞,∞]
  Dv = [9,0,8,7,∞]
  Dx = [∞,8,0,3,5]
  Dw = [∞,7,3,0,∞]
  Dy = [∞,∞,5,∞,0]

- The distance vector of the nodes after first iteration:

  Du = [0,9,17,∞,∞]
  Dv = [9,0,8,7,13]
  Dx = [17,8,0,3,5]
  Dw = [∞,7,3,0,8]
  Dy = [∞,13,5,8,0]

  The distance vector of the nodes after second iteration:
  Du = [0,9,17,16,∞]
  Dv = [9,0,8,7,13]
  Dx = [17,8,0,3,5]
  Dw = [16,7,3,0,8]
  Dy = [∞,13,5,8,0]

Class Exercise: Bellman Ford Equation

- When the algorithm converges, what are the distance vectors in each of the routers.

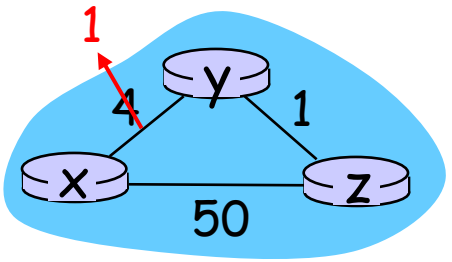
```
The distance vector of the nodes after third iteration:
Du = [0,9,17,16,22]
Dv = [9,0,8,7,13]
Dx = [17,8,0,3,5]
Dw = [16,7,3,0,8]
Dy = [22,13,5,8,0]

The distance vector of the nodes after fourth iteration:
Du = [0,9,17,16,22]
Dv = [9,0,8,7,13]
Dx = [17,8,0,3,5]
Dw = [16,7,3,0,8]
Dy = [22,13,5,8,0]
```

Distance vector: link cost changes

link cost changes:

- node detects local link cost change
- updates routing info, recalculates distance vector
- if DV changes, notify neighbors



"good news travels fast"

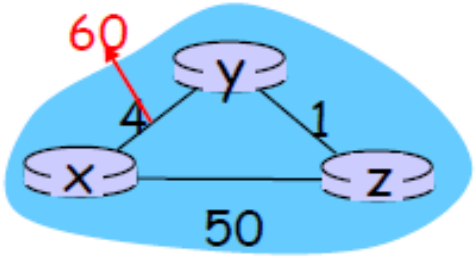
t_0 : *y* detects link-cost change, updates its DV, informs its neighbors.

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*.

Distance vector: link cost changes

Count-to-infinity Problem



Link cost changes:

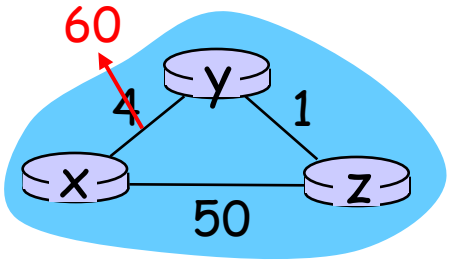
- Before change, $D_y(x)=4$, $D_y(z)=1$, $D_z(y)=1$, $D_z(x)=5$
- New min. cost from y to x, $D_y(x) = \min(60, 1+5) = 6$
  ➔ WRONG!!! and routing loop (ping-pong between y and z) until z receives y's advertisement
- After y's advertisement ($D_y(x)=6$), z updates path to x, $D_z(x) = \min(c(z,x), c(z,y)+D_y(x)) = 7$
- After z's advertisement ($D_z(x)=7$), y updates path to x, $D_y(x) = \min(c(y,x), c(y,z)+D_z(x)) = 8$
- .....
- After 44 iterations, routing stabilizes $D_y(x)=51$

Good news travels fast, but bad news travels slow!!!

Distance vector: link cost changes

link cost changes:

- node detects local link cost change
- *bad news travels slow* - "count to infinity" problem!
- 44 iterations before algorithm stabilizes: see text



t_0 : *y* detects link-cost change, (changed from 4 to 60). y computes the new minimum-cost path to x to have a cost of ***6*** via z. updates its DV, informs its neighbors.

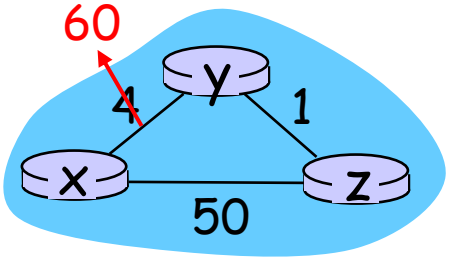
t_1 : there is a routing loop – to get to x router y routes through *z and z routes through y.*

t_2 : *y* computed new minimum cost to x and informs z.
z receives y's new DV indicating y's minimum cost to x is 6.
z knows it can get to y with a cost of 1 and computes least cost to x as 7.

poisoned reverse:

- If Z routes through Y to get to X :
  - Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)

Distance vector: Poisoned Reverse



How to prevent routing loop and slow convergence
→ Poisoned reverse

- If Z receives packets from Y to X, but it routes through Y to get to X :
  - Z tells Y as $D_Z(x) = \infty$ (little white lie)
  - Y will not route to X via Z
- Y updates $D_y(x)=60$
- Z updates $D_Z(x)=\min(c(z,x), c(z,y)+D_y(x))=\min(50, 61) = 50$
- Y updates $D_y(x)=\min(c(y,x), c(y,z)+D_z(x))=\min(60, 51) = 51$

no route change → stabilized

Comparison of LS and DV algorithms

- ***LS:***
  - suffers from 'oscillation problem'
  - A lot of advertisements have to be flooded to the network
- ***DV:***
  - suffers from count-to-infinity problem
  - Takes time in convergence

robustness: what happens if router malfunctions?

LS:

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

DV:

- DV node can advertise incorrect *path* cost
- each node's table used by others
  - error propagate thru network

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 ICMP: The Internet Control Message Protocol

ICMP: internet control message protocol

- ICMP used by hosts & routers to communicate network-level information
  - error reporting: unreachable host, network, port
  - echo request/reply (used by ping)
- network-layer "above" IP:
  - ICMP msgs carried in IP datagrams
- ICMP message: type, code, IP header and 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

- Sender sends series of UDP segments to destination. TTL field is incremented for each set of datagrams. For example,
  - first set has TTL =1
  - second set has TTL=2, etc.
- In addition, unlikely port number is set in the udp segment.
- when datagram in n^{th} set arrives to n^{th} router:
  - router discards datagram and sends source ICMP message with type 11 and code 0, which means "TTL expired"
  - ICMP message include name of router & IP address
- when ICMP message arrives, sender records RTTs

stopping criteria:

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