Happy Thursday !! :)

CSC358 Week 8

Midterm

- Class average: 71%

- Highest mark: 68/70

- Solutions have been posted

- Remarking request form posted, please attach it to the test when submitting remarking request.

- Winning stickers: if you got >= 60 / 70

- This Friday's tutorial: midterm take-up

Summary: Network Layer Data Plane

4.1 Overview of Network layer: data plane and control plane

4.2 What's inside a router

4.3 IP: Internet Protocol

- datagram format
- fragmentation
- IPv4 addressing
- NAT
- IPv6

Question: how do forwarding tables (destination-based forwarding) or flow tables (generalized forwarding) computed?

Answer: by the control plane (next chapter)

Network layer **control** plane

goals: understand principles behind network control plane

- routing algorithms

- Internet Control Message Protocol

- network management

Outline

Network-layer functions

Recall: two network-layer functions:

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

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

data plane
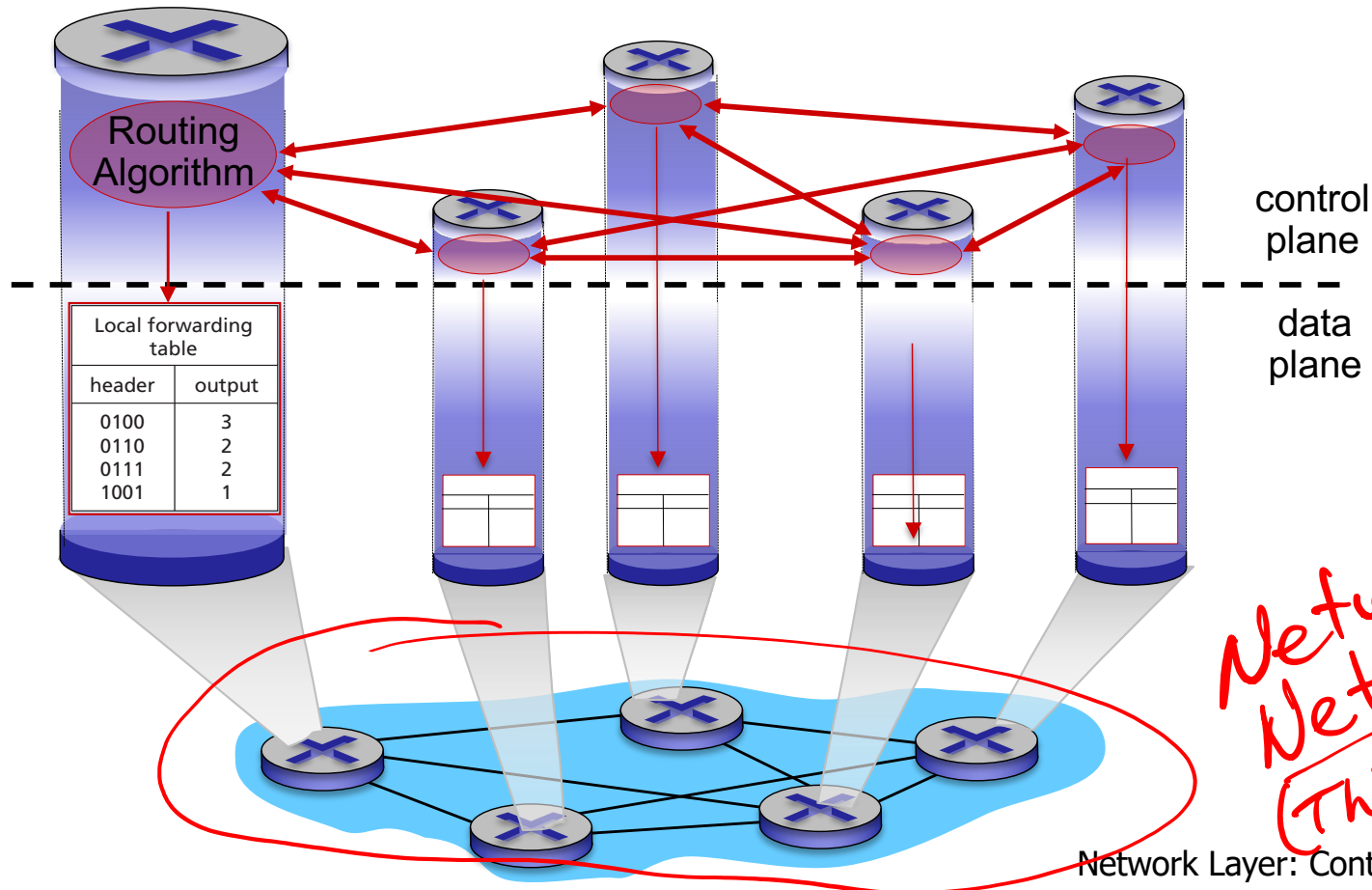
control plane

↳ functions, Compoers pieces,

↳ routetaking

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

Routing Algorithm

Local forwarding table	
header	output
0100	3
0110	2
0111	2
1001	1

control plane

data plane

Network of Networks (This is just one)

Logically centralized control plane

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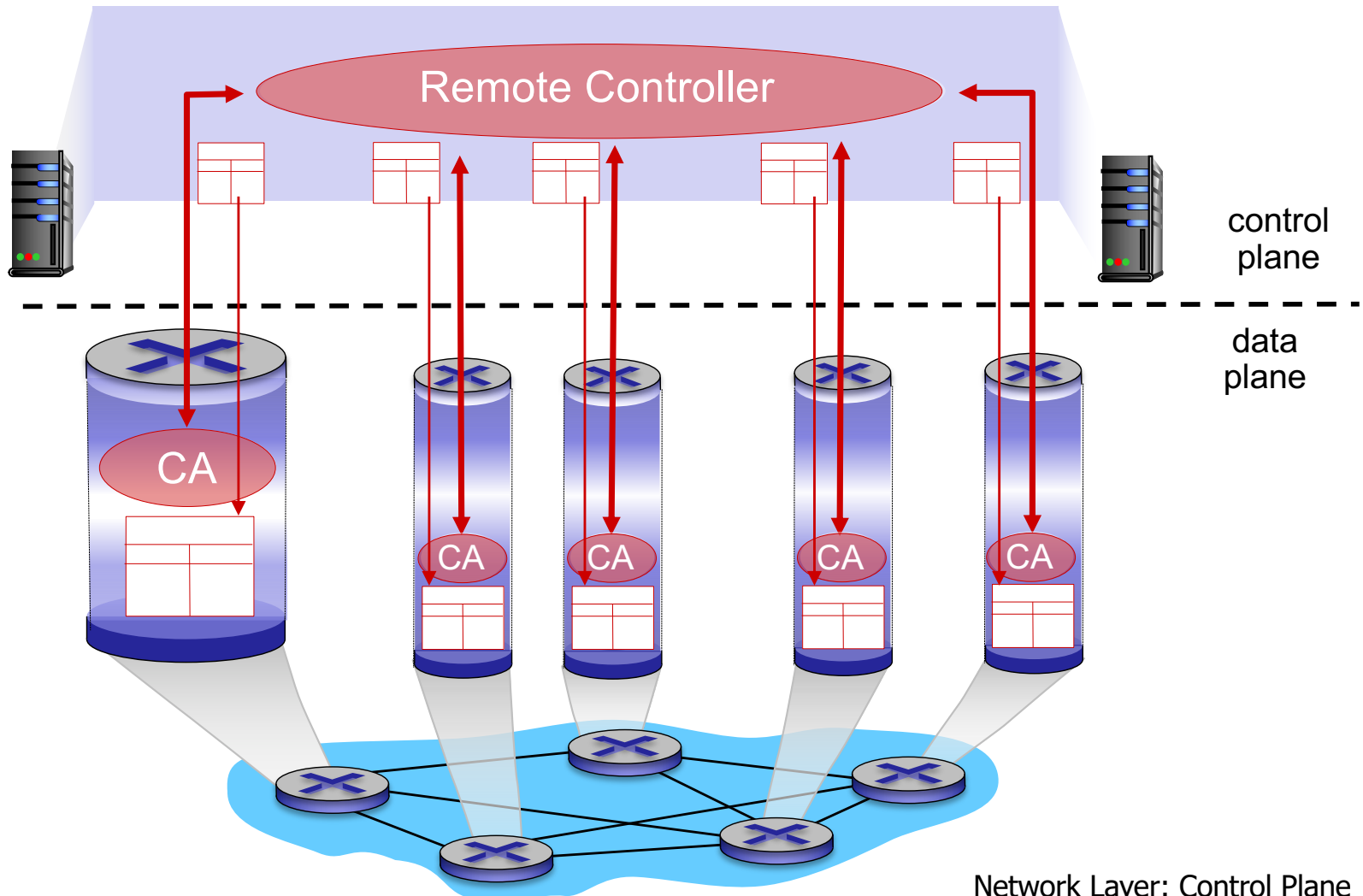
Remote Controller

control plane

data plane

CA

Outline

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
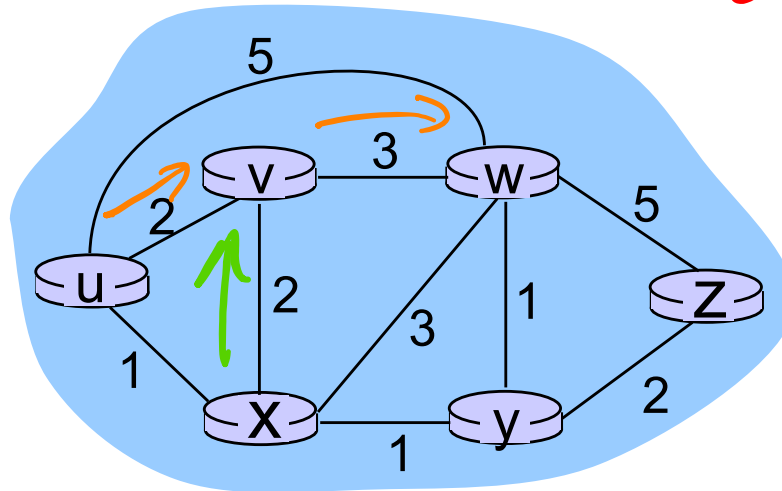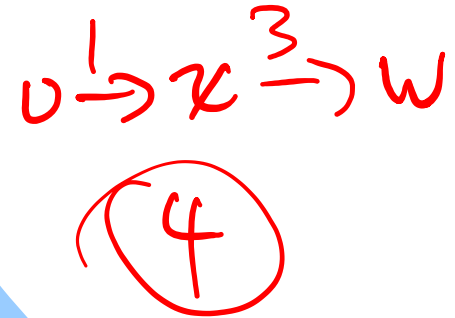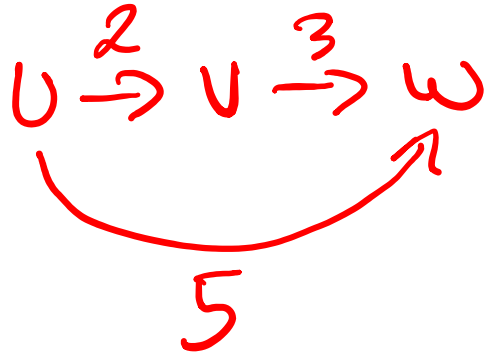
graph: G = (N,E)

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
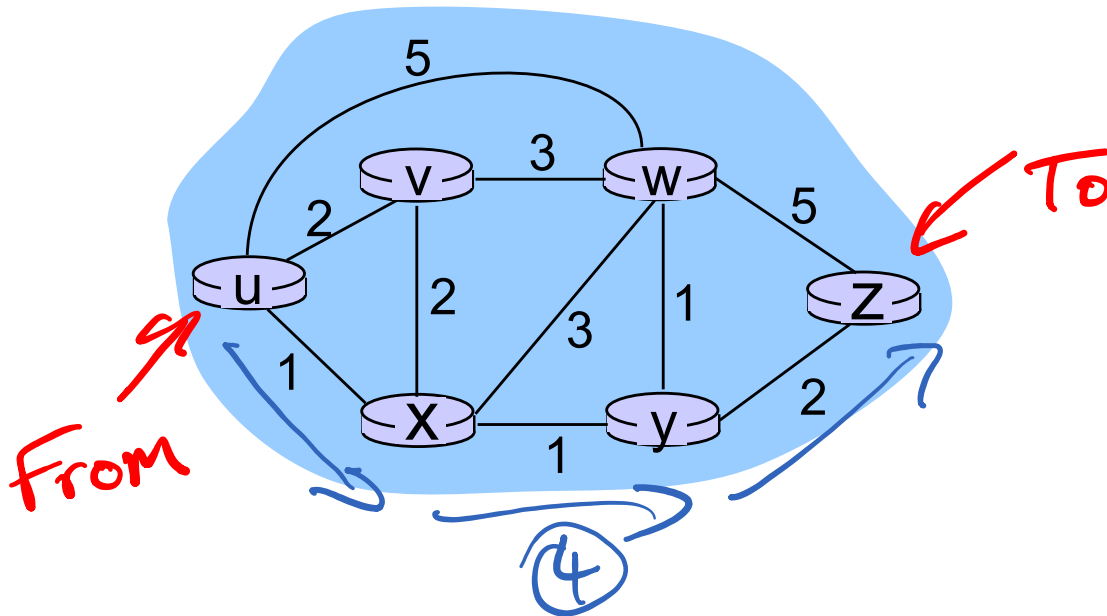
c(x,x') = cost of link (x,x')

    e.g., c(w,z) = 5

cost could be 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

Outline

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 dests

notation: *Cost*

- c(x,y): link cost from node x to y; = ∞ if not direct neighbors — *value*
- D(v): current value of cost of path from source to dest v — *last node*
- p(v): predecessor node along path from source to v — *All nodes*
- 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

Current value
lastnode

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					

$u \xrightarrow{5} w$

$u \xrightarrow{2} v$

$u \xrightarrow{1} x$
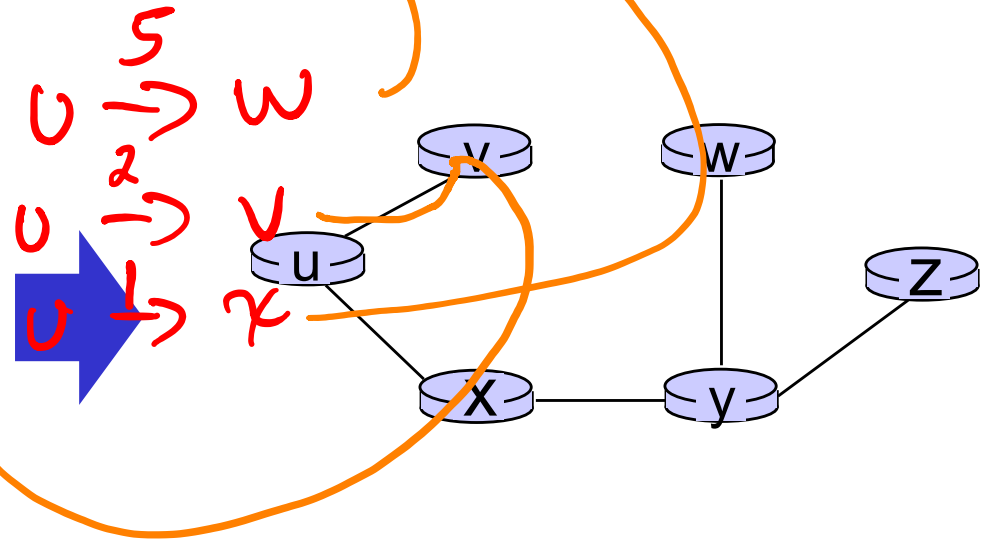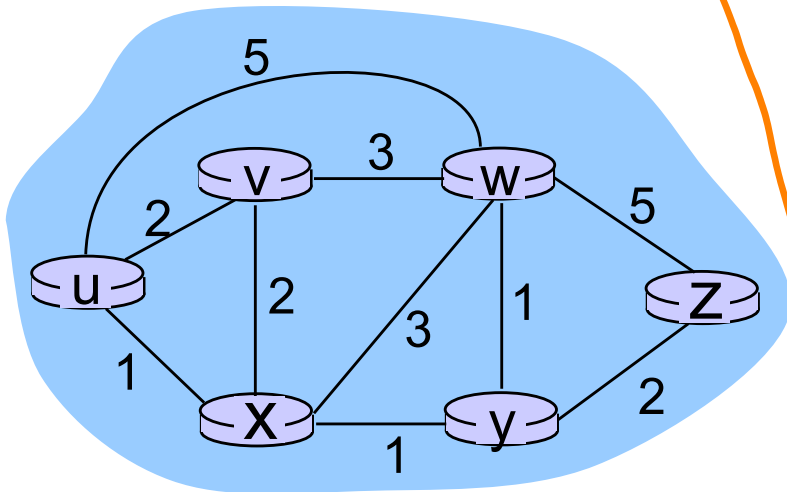
Dijkstra's algorithm: example (2)

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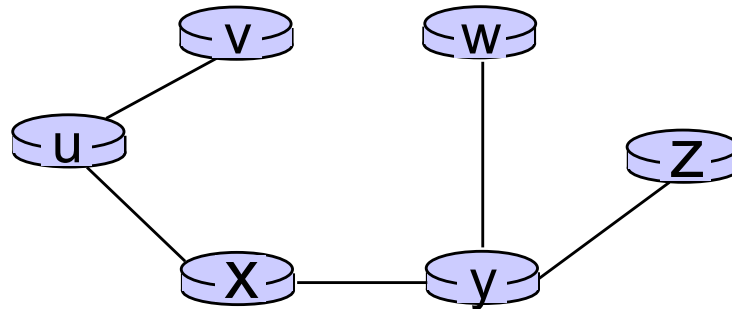
resulting forwarding table in u:

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

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: $O(n\log n)$

Each node knows the entire graph's topology.

Outline

Distance vector algorithm

Bellman-Ford equation (dynamic programming)

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

clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

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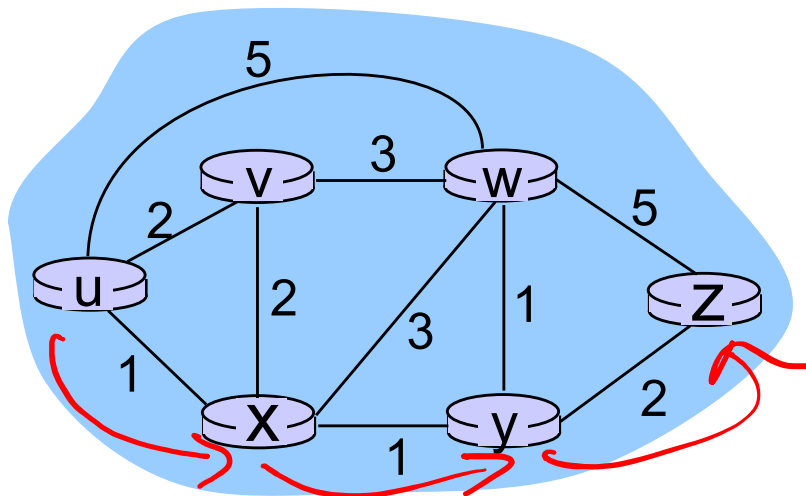
$\qquad = \min \{2 + 5,$
$\qquad\qquad 1 + 3,$
$\qquad\qquad 5 + 3\} = 4$

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

Distance vector algorithm

- $D_x(y)$ = estimate of least cost from x to y
  - x maintains distance vector $D_x = [D_x(y): y \in N]$
- node x:
  - knows cost to each neighbor v: c(x,v)
  - maintains its neighbors' distance vectors. For each neighbor v, x maintains
    $D_v = [D_v(y): y \in N]$

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:

$$D_x(y) \leftarrow min_v\{c(x,v) + D_v(y)\} \text{ for each node } y \in N$$

- ❖ under minor, natural conditions, the estimate $D_x(y)$ *converge to the actual least cost* $d_x(y)$

C→B→A ⑥   C→D→A ㊶

Distance vector algorithm

50
A   40
D

B   1

C
1

C→B
→D
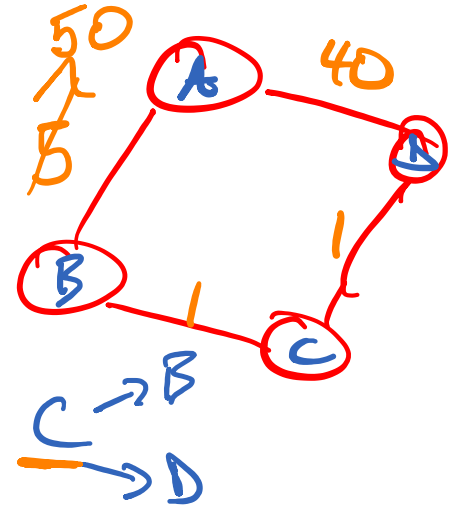
iterative, asynchronous:

each local iteration caused by:

- local link cost change
- DV update message from neighbor

distributed:

- each node notifies neighbors *only* when its DV changes
  - neighbors then notify their neighbors if necessary

each node:

wait for (change in local link cost or msg from neighbor)

↓

recompute estimates

↓

if DV to any dest has changed, *notify* neighbors

Each node only needs to know their neighbours.

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

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

node x table

cost to

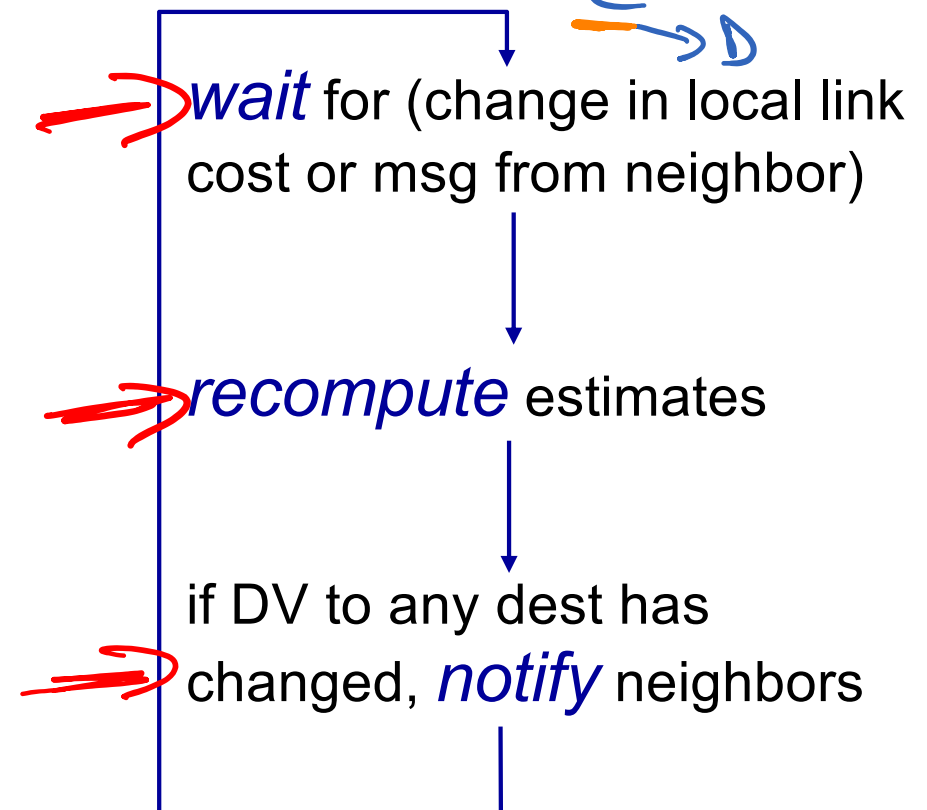|	x	y	z
from x	0	2	7
from y	∞	∞	∞
from z	∞	∞	∞

cost to

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

node y table

cost to

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

node z table

cost to

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

time

2+1 = 3

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

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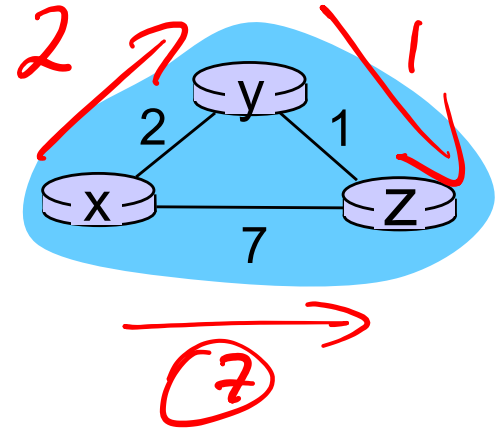
node x table

cost to

from	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

cost to

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

cost to

from	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node y table

cost to

from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

cost to

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

cost to

from	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node z table

cost to

from	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

cost to

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

cost to

from	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

time

Distance vector: link cost changes

Dy(x)	Dz(x)
4	5
1	5
1	2
1	2

link cost changes:

- ❖ node detects local link cost change
- ❖ updates routing info, recalculates distance vector
- ❖ if DV changes, notify neighbors
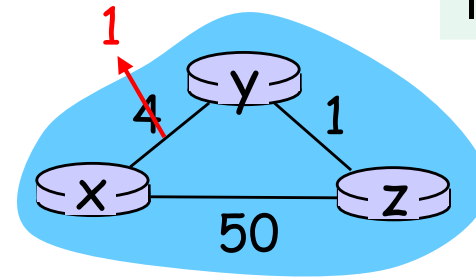
"good news travels fast"

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.

D^x	x	y	z
x	0	4	15
y	4	0	1
z	5	1	0

D^y	x	y	z
x	0	4	1
y	4	0	1
z	5	1	0

1

4

y

1

x

z

50
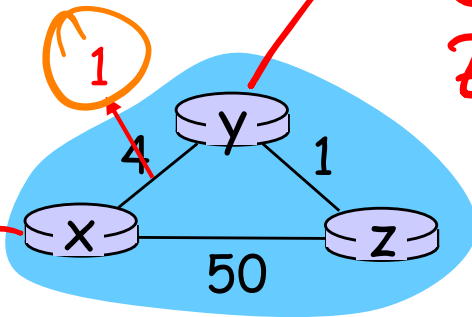
Message will pass to ⓩ "node" & table will recompute

Distance vector: link cost changes

Dy(x)	Dz(x)
4	5
6	5
6	7
…	…
48	49
50	49
50	50
51	50
51	50

This table is just meant to give a rough idea of the change of the values. To be more precise, you should put a distance table next to each node in the graph and update/trace them.

link cost changes:

❖ node detects local link cost change

❖ *bad news travels slow* - "count to infinity" problem!

❖ **44** iterations before algorithm stabilizes: see text

60

4 — Y — 1

X —— 50 —— Z

solution: 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)

❖ will this completely solve count to infinity problem?

Dy(x)	Dz(x)
4	5 (inf)
60	5 (inf)
60	50
51	50
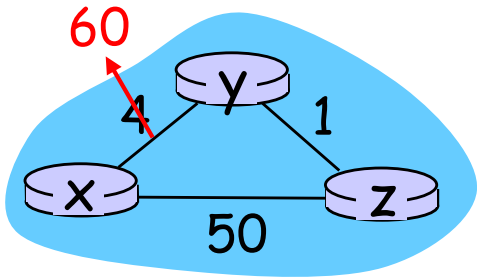
Comparison of LS and DV algorithms

message complexity

- **LS:** with n nodes, E links, O(nE) msgs sent
- **DV:** exchange between neighbors only
  - convergence time varies

speed of convergence

- **LS:** O(n²) algorithm requires O(nE) msgs

- **DV:** convergence time varies
  - may be routing loops
  - count-to-infinity problem

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

No clear obvious winner. Both are used in the Internet.

Outline

5.1 introduction

5.2 routing protocols
- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

Making routing scalable

our routing study thus far - idealized

- all routers identical
- network "flat"

… *not* true in practice

scale: with billions of destinations:

- can't store all destinations in routing tables!
- routing table exchange would swamp links!

administrative autonomy

- internet = network of networks
- each network admin may want to control routing in its own network

Internet approach to scalable routing

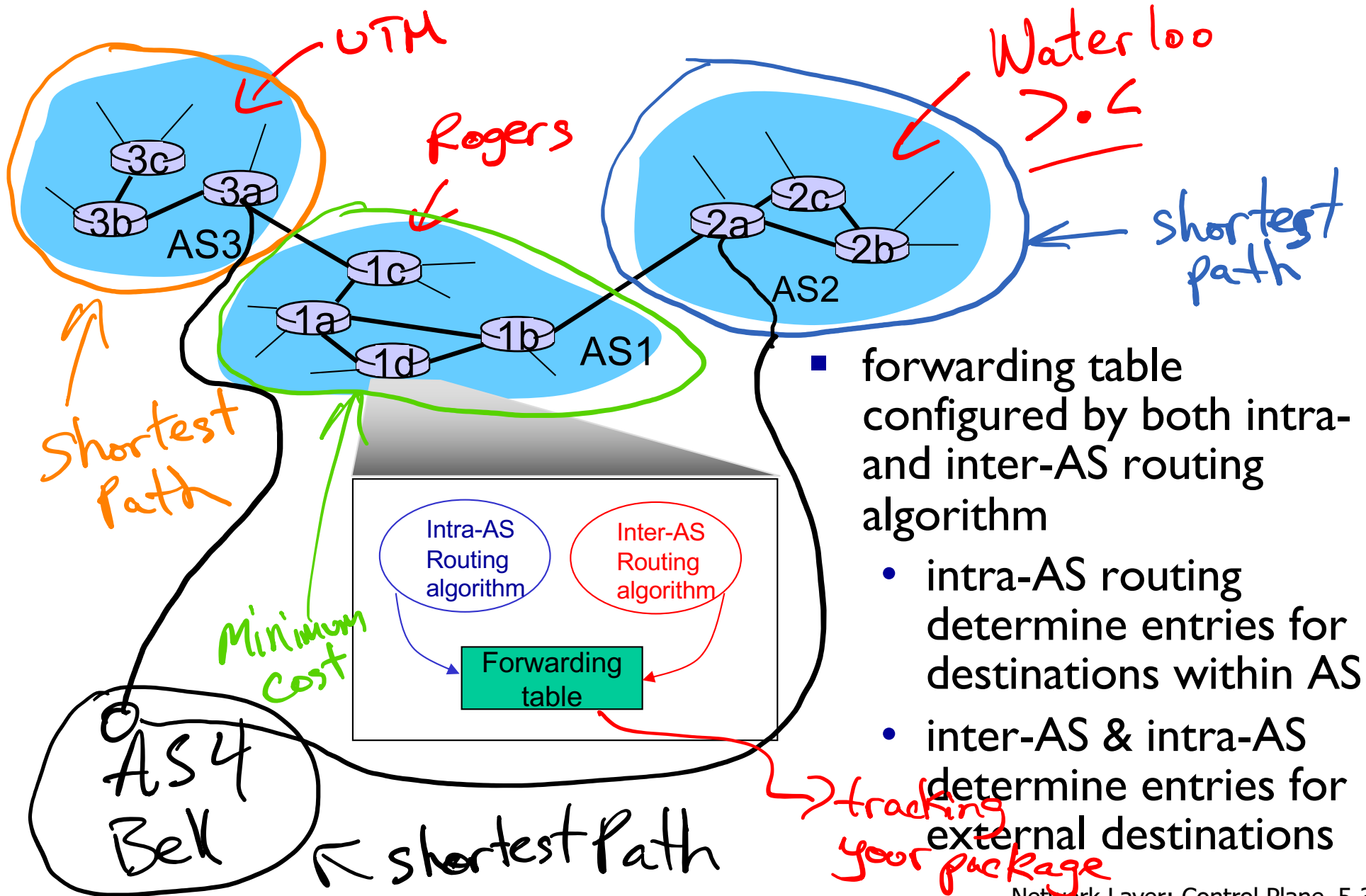aggregate routers into regions known as "autonomous systems" (AS) (a.k.a. "domains")

intra-AS routing

- routing among hosts, routers in same AS ("network")
- all routers in AS must run *same* intra-domain protocol
- routers in *different* AS can run *different* intra-domain routing protocol
- gateway router: at "edge" of its own AS, has link(s) to router(s) in other AS'es

inter-AS routing

- routing among AS'es
- gateways perform inter-domain routing (as well as intra-domain routing)

Interconnected ASes

UTM

Rogers

Waterloo >.<

shortest path

AS3

AS1

AS2

shortest Path

Minimum Cost

Intra-AS Routing algorithm

Inter-AS Routing algorithm

Forwarding table

AS4 Bell

← shortest Path

→ tracking your package

- forwarding table configured by both intra- and inter-AS routing algorithm

  - intra-AS routing determine entries for destinations within AS

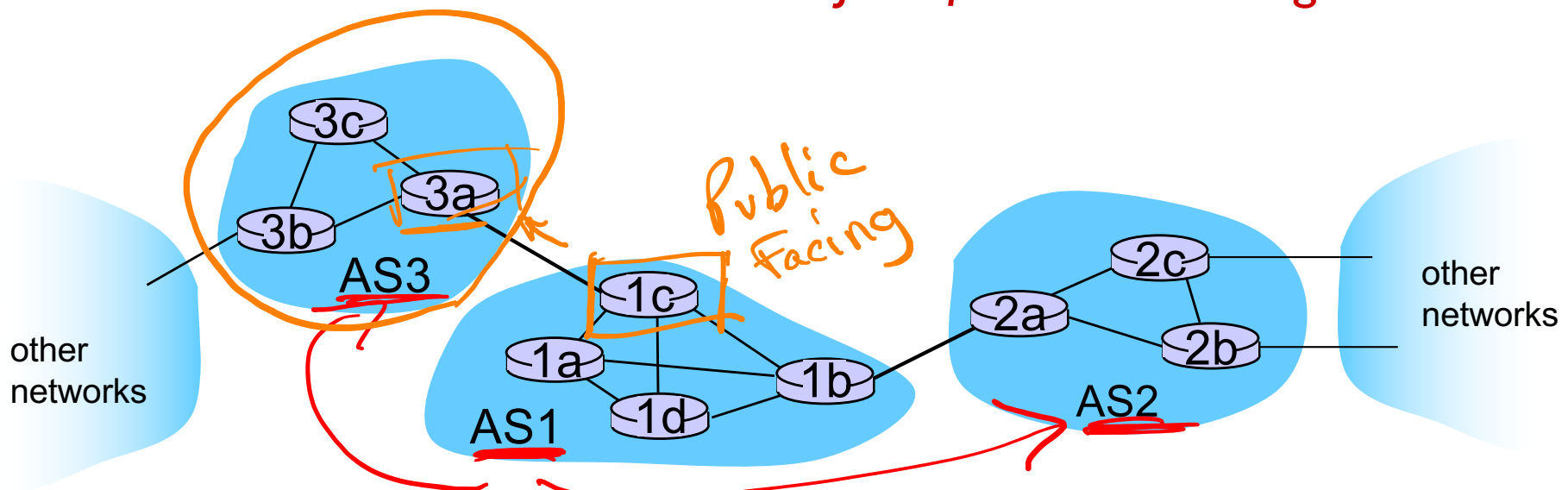 - inter-AS & intra-AS determine entries for external destinations

Inter-AS tasks

- suppose router in AS1 receives datagram destined outside of AS1:
  - router should forward packet to gateway router, but which one?

AS1 must:

1. learn which dests are reachable through AS2, which through AS3
2. propagate this reachability info to all routers in AS1

job of inter-AS routing!

Intra-AS Routing

- also known as *interior gateway protocols (IGP)*

- most common intra-AS routing protocols:

  - RIP: Routing Information Protocol

  - OSPF: Open Shortest Path First (IS-IS protocol essentially same as OSPF)

  - IGRP: Interior Gateway Routing Protocol (Cisco proprietary for decades, until 2016)

OSPF (Open Shortest Path First)

- "open": publicly available

- uses link-state algorithm
  - link state packet dissemination
  - topology map at each node
  - route computation using Dijkstra's algorithm

- router floods OSPF link-state advertisements to all other routers in *entire* AS
  - carried in OSPF messages directly over IP (rather than TCP or UDP
  - link state: for each attached link

OSPF "advanced" features

- *security:* all OSPF messages authenticated (to prevent malicious intrusion)
- multiple same-cost paths allowed (only one path in RIP)
- for each link, multiple cost metrics for different ToS (e.g., satellite link cost set low for best effort ToS; high for real-time ToS)
- integrated uni- and multi-cast support:
  - Multicast OSPF (MOSPF) uses same topology data base as OSPF
- hierarchical OSPF in large domains.

Outline

Internet inter-AS routing: BGP

- **BGP (Border Gateway Protocol):** *the* de facto inter-domain routing protocol
  - "glue that holds the Internet together"
- BGP provides each AS a means to:
  - **eBGP:** obtain subnet reachability information from neighboring ASes
  - **iBGP:** propagate reachability information to all AS-internal routers.
  - determine "good" routes to other networks based on reachability information and *policy*
- allows subnet to advertise its existence to rest of Internet: *"I am here"*