National University of Computer & Emerging Sciences

CS 3001 – COMPUTER NETWORKS

Lecture 19
Chapter 5

26th October, 2023

Nauman Moazzam Hayat
nauman.moazzam@lhr.nu.edu.pk

Office Hours: 02:30 pm till 06:00 pm (Every Tuesday & Thursday)

Chapter 5
Network Layer: Control Plane

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.

For a revision history, see the slide note for this page.

Thanks and enjoy! JFK/KWR
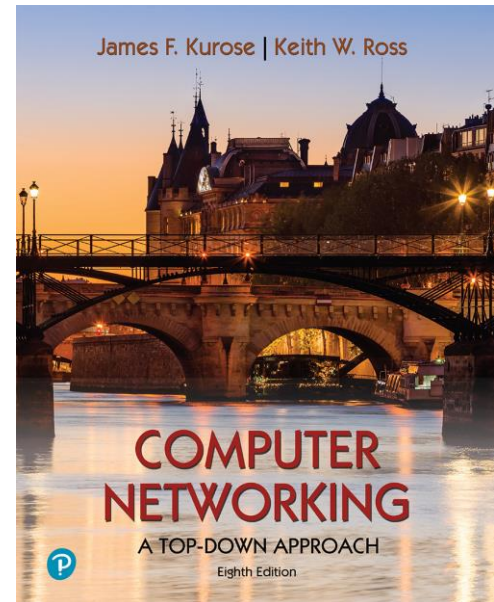
Computer Networking: A Top-Down Approach
8th edition
Jim Kurose, Keith Ross
Pearson, 2020

Network layer control plane: our goals

- understand principles behind network control plane:
  - traditional routing algorithms
  - SDN controllers
  - network management, configuration

- instantiation, implementation in the Internet:
  - OSPF, BGP
  - OpenFlow, ODL and ONOS controllers
  - Internet Control Message Protocol: ICMP
  - SNMP, YANG/NETCONF

Network layer: "control plane" roadmap

- introduction
- routing protocols
  - link state
  - distance vector
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol

- network management, configuration
  - SNMP
  - NETCONF/YANG

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

Two approaches to structuring network control plane:
- per-router control (traditional)
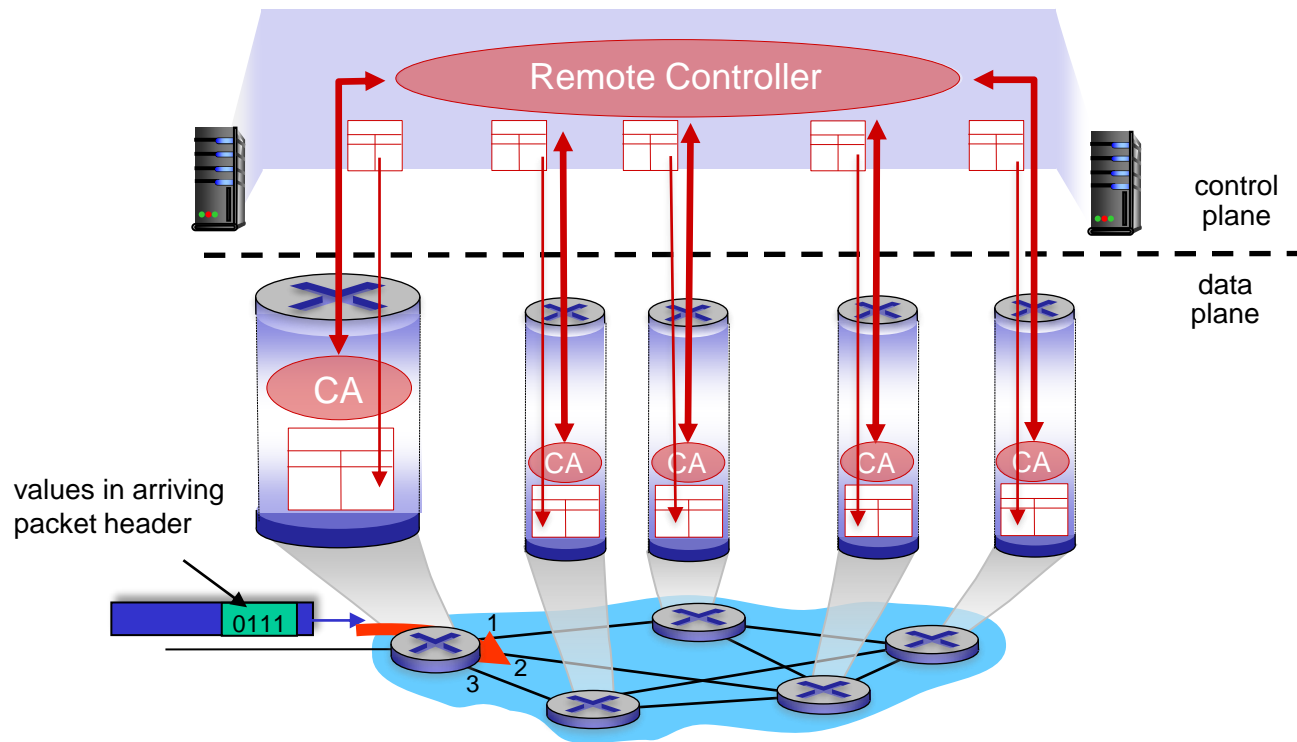- logically centralized control (software defined networking)

Per-router control plane

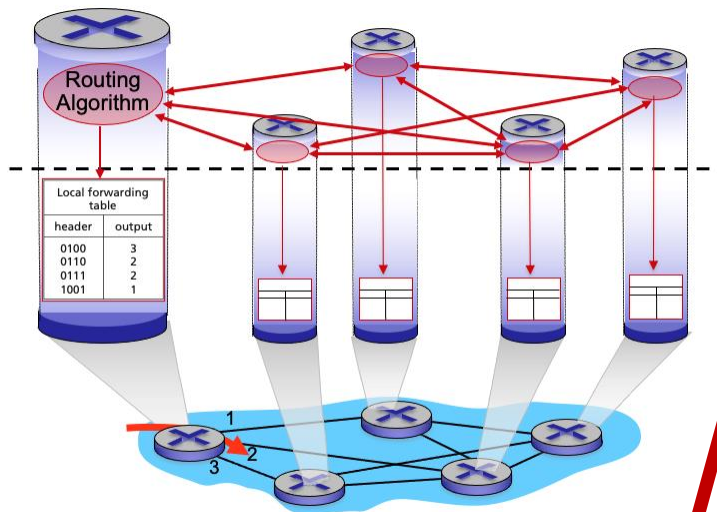Individual routing algorithm components *in each and every router* interact in the control plane
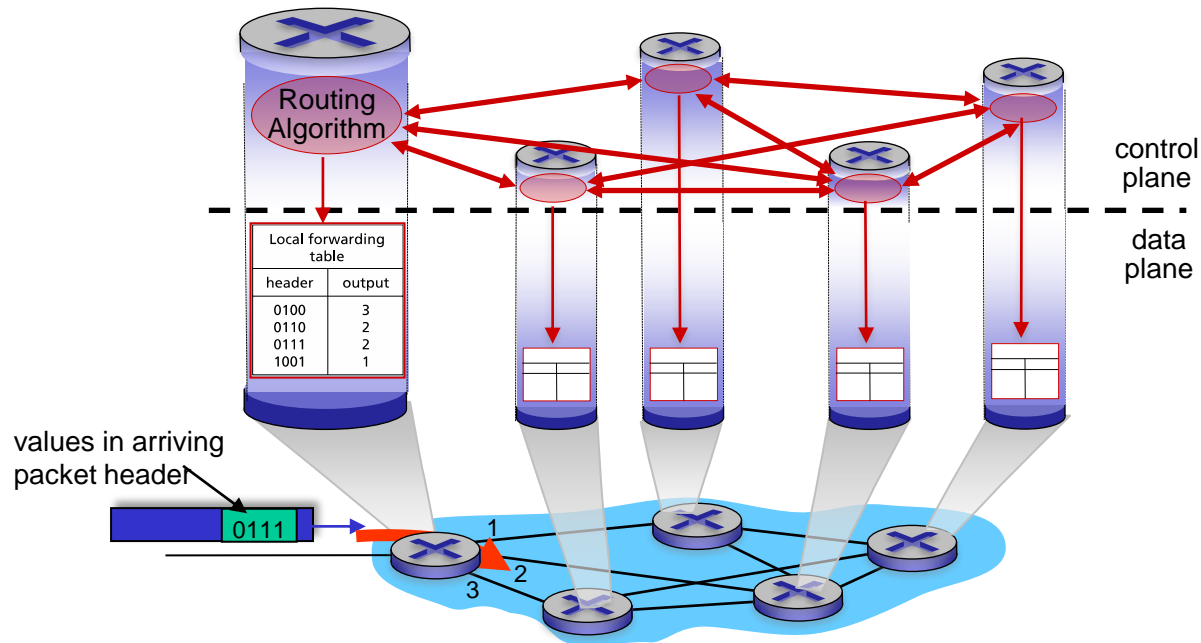
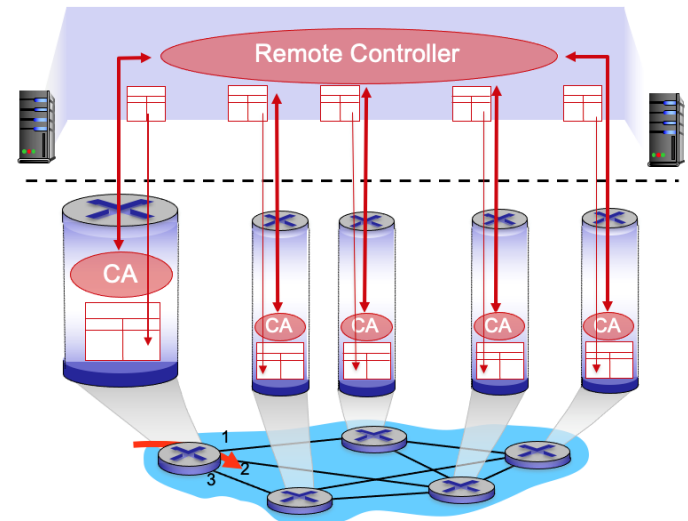# Software-Defined Networking (SDN) control plane

Remote controller computes, installs forwarding tables in routers

values in arriving packet header

Per-router control plane

SDN control plane

Network layer: "control plane" roadmap

- introduction
- **routing protocols**
  - **link state**
  - **distance vector**
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol

- network management, configuration
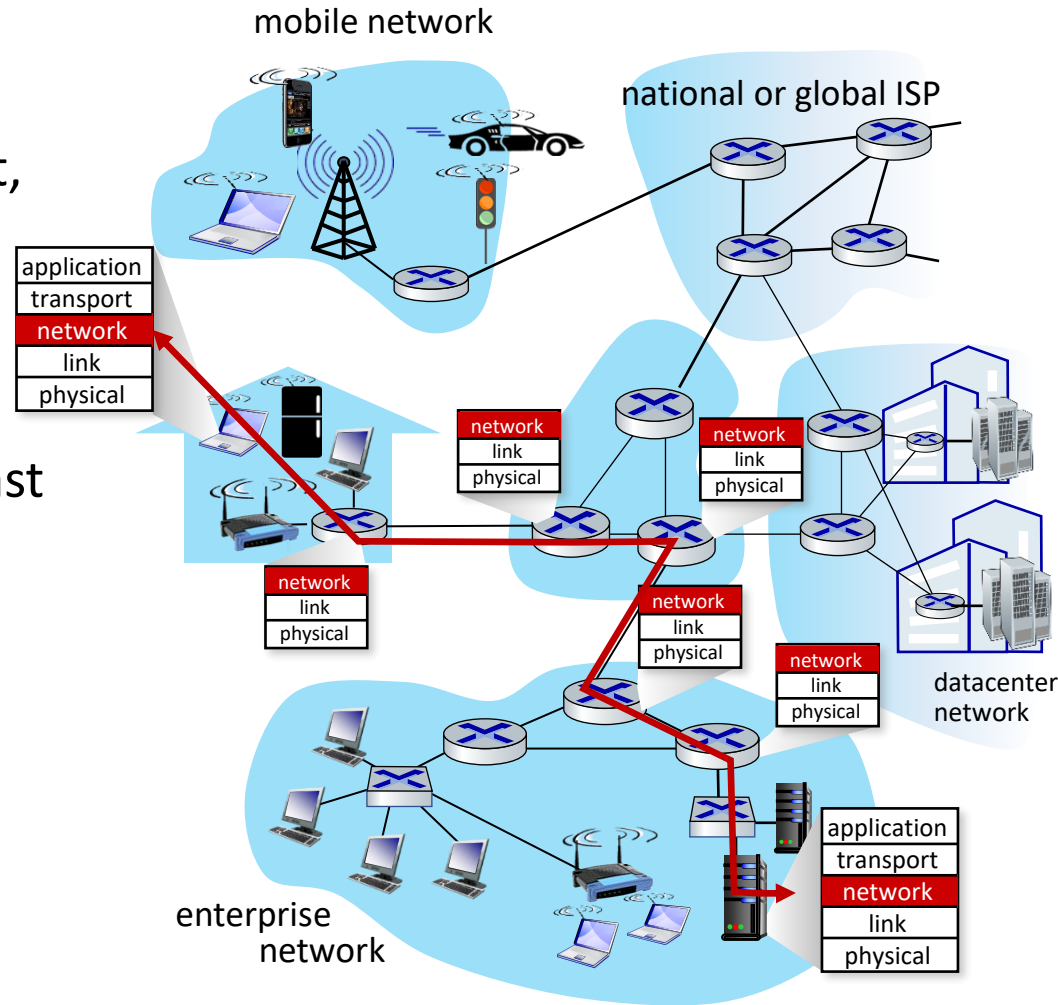 - SNMP
  - NETCONF/YANG

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 traverse from given initial source host to final destination host

- "good": least "cost", "fastest", "least congested"

- routing: a "top-10" networking challenge!

Graph abstraction: link costs
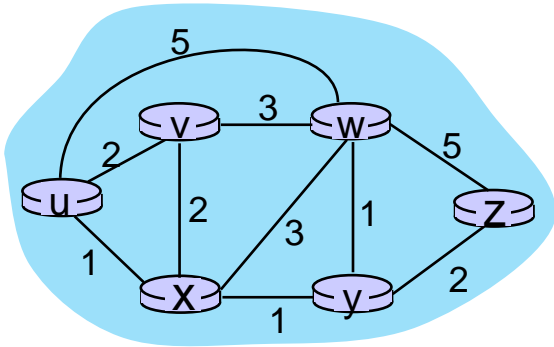
$c_{a,b}$: cost of *direct* link connecting a and b

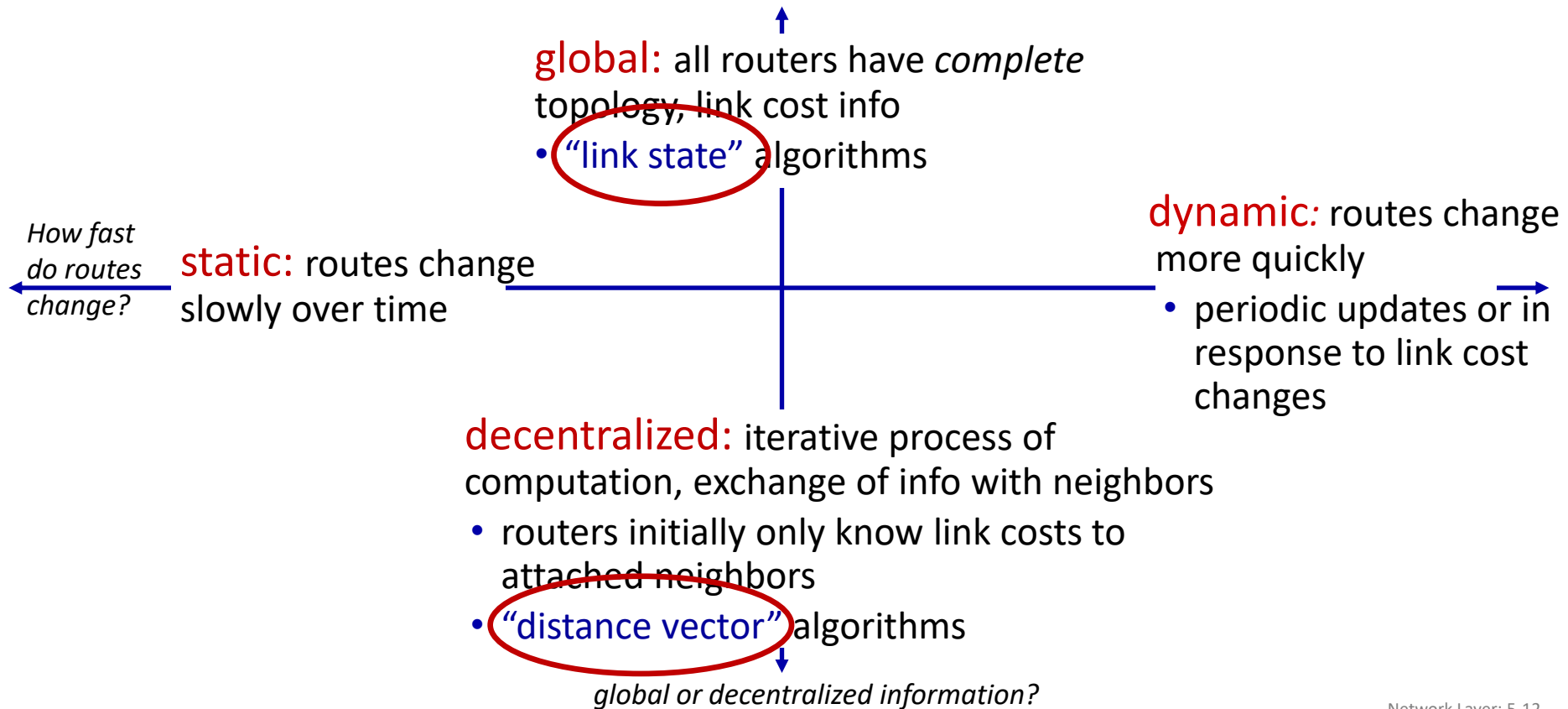e.g., $c_{w,z} = 5$, $c_{u,z} = \infty$

cost defined by network operator:
could always be 1, or inversely related
to bandwidth, or inversely related to
congestion

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)* }

Routing algorithm classification

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

How fast do routes change?

static: routes change slowly over time

dynamic: routes change more quickly
- periodic updates or in response to link cost changes

decentralized: iterative process of computation, exchange of info with neighbors
- routers initially only know link costs to attached neighbors
- "distance vector" algorithms

global or decentralized information?

Network layer: "control plane" roadmap

- introduction
- **routing protocols**
  - link state
  - distance vector
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol

- network management, configuration
  - SNMP
  - NETCONF/YANG

Dijkstra's link-state routing algorithm

- centralized: 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

- $c_{x,y}$: <u>direct</u> link cost from node *x* to *y*; = ∞ if not direct neighbors
- *D(v): current* estimate of cost of least-cost-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

Dijkstra's link-state routing algorithm

```
1  Initialization:
2    N' = {u}                    /* compute least cost path from u to all other nodes */
3    for all nodes v
4      if v adjacent to u         /* u initially knows direct-path-cost only to  direct neighbors   */
5        then D(v) = c_u,v        /* but may not be minimum cost!                                   */
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 least-path-cost to v is either old least-cost-path to v or known
14   least-cost-path to w plus direct-cost from w to v */
15 until all nodes in N'
```
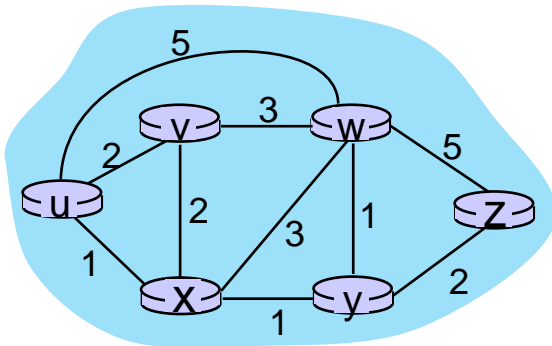
Dijkstra's algorithm: an example

Step	N'	v D(v),p(v)	w D(w),p(w)	x D(x),p(x)	y D(y),p(y)	z D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1						
2						
3						
4						
5						

Initialization (step 0):
For all a: if a adjacent to u then $D(a) = c_{u,a}$

Dijkstra's algorithm: an example

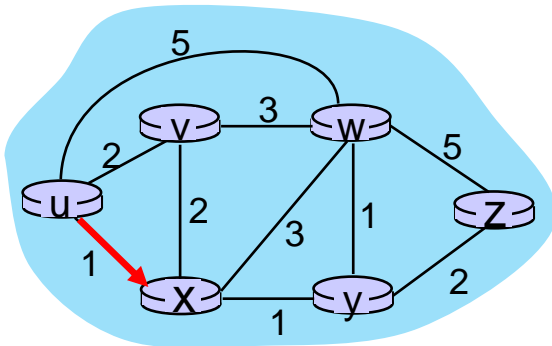|		v	w	x	y	z
Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux					
2						
3						
4						
5						

8 *Loop*
9   find *a* not in *N'* such that *D(a)* is a minimum
10   add *a* to *N'*

Dijkstra's algorithm: an 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						
3						
4						
5						

(column headers above table in red: v, w, x, y, z)

8   *Loop*
9   find a not in N' such that $D(a)$ is a minimum
10   add a to N'
11   update $D(b)$ for all b adjacent to a and not in N' :
           $D(b) = \min (D(b), D(a) + c_{a,b})$

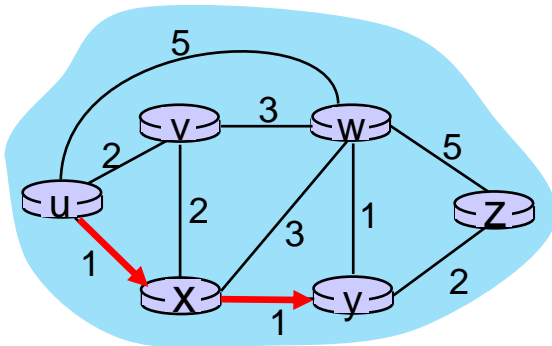$D(v) = \min (D(v), D(x) + c_{x,v}) = \min(2, 1+2) = 2$
$D(w) = \min (D(w), D(x) + c_{x,w}) = \min (5, 1+3) = 4$ NEW!
$D(y) = \min (D(y), D(x) + c_{x,y}) = \min(\inf, 1+1) = 2$ NEW!

Dijkstra's algorithm: an example

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

8   *Loop*
9   find *a* not in *N'* such that *D(a)* is a minimum
10   add *a* to *N'*

Dijkstra's algorithm: an example

		v	w	x	y	z
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						
4						
5						

8 *Loop*
9   find *a* not in *N'* such that *D(a)* is a minimum
10   add *a* to *N'*
11   update *D(b)* for all *b* adjacent to *a* and not in *N'* :
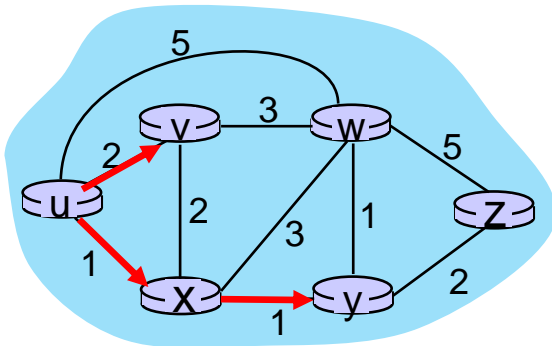   **$D(b) = min (D(b), D(a) + c_{a,b})$**

$D(w) = min (D(w), D(y) + c_{y,w}) = min (4, 2+1) = 3$ NEW!
$D(z) = min (D(z), D(y) + c_{y,z}) = min(inf, 2+2) = 4$ NEW!

Dijkstra's algorithm: an example

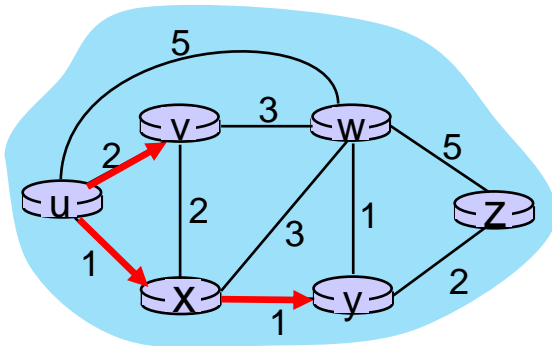|		v	w	x	y	z
Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv					
4						
5						

8 *Loop*
9   find *a* not in *N'* such that *D(a)* is a minimum
10   add *a* to *N'*

Dijkstra's algorithm: an example

Step	N'	v D(v),p(v)	w D(w),p(w)	x D(x),p(x)	y D(y),p(y)	z 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						
5						

8   *Loop*
9   find *a* not in *N'* such that *D(a)* is a minimum
10   add *a* to *N'*
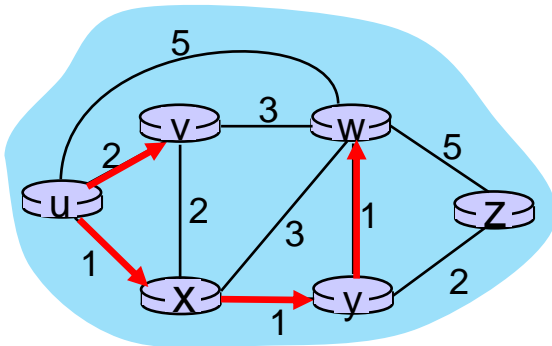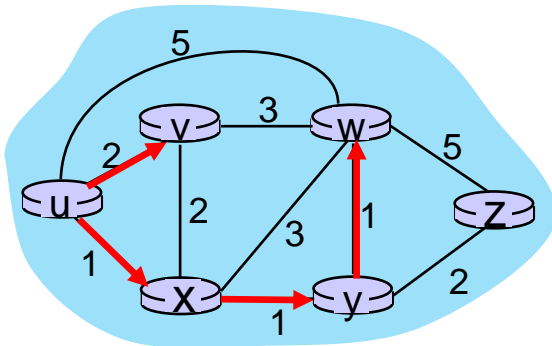11   update *D(b)* for all *b* adjacent to *a* and not in *N'* :
        D(b) = min (D(b), D(a) + c$_{a,b}$)

D(w) = min (D(w), D(v) + c$_{v,w}$) = min (3, 2+3) = 3

Dijkstra's algorithm: an example

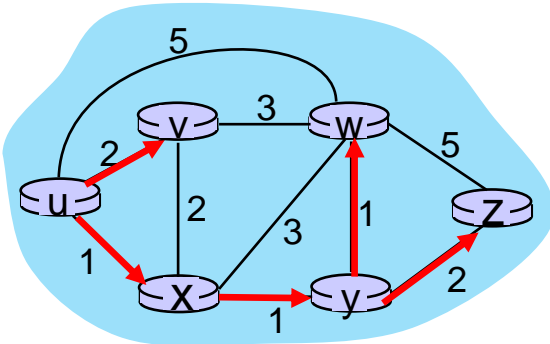| Step	N'	v D(v),p(v)	w D(w),p(w)	x D(x),p(x)	y D(y),p(y)	z D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					
5						

8 *Loop*
9   find *a* not in *N'* such that *D(a)* is a minimum
10   add *a* to *N'*

Dijkstra's algorithm: an example

| Step	N'	v D(v),p(v)	w D(w),p(w)	x D(x),p(x)	y D(y),p(y)	z 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						

8 *Loop*
9   find *a* not in *N'* such that *D(a)* is a minimum
10   add *a* to *N'*
11   update *D(b)* for all *b* adjacent to *a* and not in *N'* :
        $D(b) = min (D(b), D(a) + c_{a,b})$

$D(z) = min (D(z), D(w) + c_{w,z}) = min (4, 3+5) = 4$

Dijkstra's algorithm: an example

Step	N'	v D(v),p(v)	w D(w),p(w)	x D(x),p(x)	y D(y),p(y)	z 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					

8 *Loop*
9   find *a* not in *N'* such that *D(a)* is a minimum
10   add *a* to *N'*

Dijkstra's algorithm: an example

		v	w	x	y	z
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					

8   *Loop*

9   find *a* not in *N'* such that *D(a)* is a minimum

10   add *a* to *N'*

11   update *D(b)* for all *b* adjacent to *a* and not in *N'* :
$$D(b) = \min (D(b), D(a) + c_{a,b})$$

Dijkstra's algorithm: an example
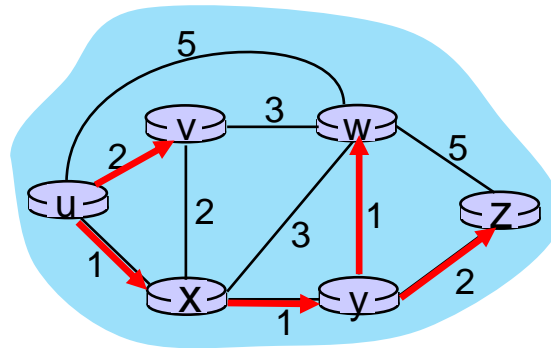
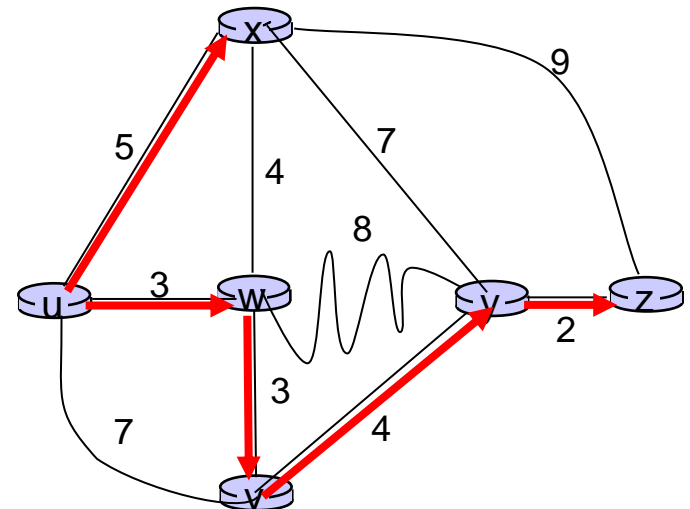resulting least-cost-path tree from u:
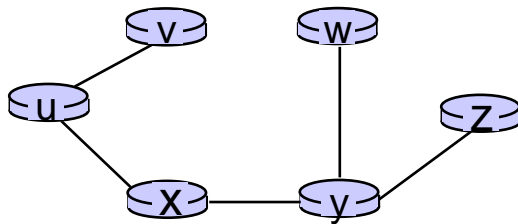
resulting forwarding table in u:

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

route from *u* to *v* directly

route from u to all other destinations via *x*

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	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 least-cost-path tree by tracing predecessor nodes
- ties can exist (can be broken arbitrarily)

# Dijkstra's algorithm: discussion

algorithm complexity: n nodes

- each of n iteration: need to check all nodes, w, not in N
- $n(n+1)/2$ comparisons: $O(n^2)$ complexity
- more efficient implementations possible: $O(n\log n)$

message complexity:

- each router must *broadcast* its link state information to other n routers
- efficient (and interesting!) broadcast algorithms: $O(n)$ link crossings to disseminate a broadcast message from one source
- each router's message crosses $O(n)$ links: overall message complexity: $O(n^2)$

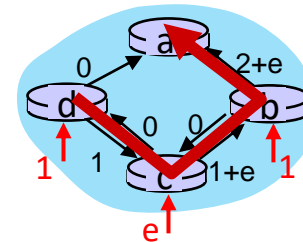# Dijkstra's algorithm: oscillations possible

- when link costs depend on traffic volume, route oscillations possible
- sample scenario:
  - routing to destination a, traffic entering at d, c, e with rates 1, e (<1), 1
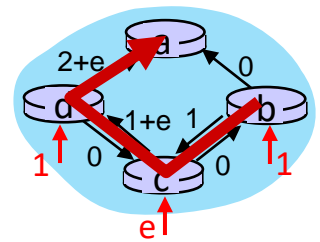 - link costs are directional, and volume-dependent
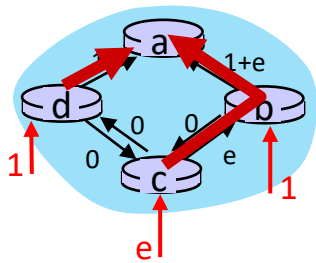
initially
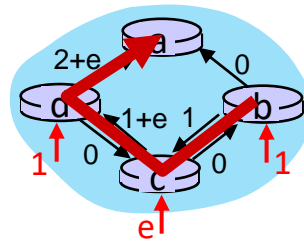
given these costs,
find new routing….
resulting in new costs

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

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

Assignement # 4 (Chapter – 4)

- *4th Assignment will be uploaded on Google Classroom on Thursday, 26th October, 2023, in the Stream - Announcement Section*

- *Due Date: Thursday, 2nd November, 2023 (Handwritten solutions to be submitted during the lecture)*

- *Please read all the instructions carefully in the uploaded Assignment document, follow & submit accordingly*

Quiz # 4 (Chapter – 4)

- *On: Thursday, 2nd November, 2023 (During the lecture)*

- *Quiz to be taken during own section class only*