CS118 Discussion Week 7: The Network Layer (Control Plane)

Questions?

- From this week or about the HW/Midterm

This week

- Review the two main types of traditional routing algorithms
  - Dijkastra's/Link-State
  - Bellman-Ford/Distance-Vector
- Vagrant/Mininet Tutorial

The Network Layer (The Control Plane)

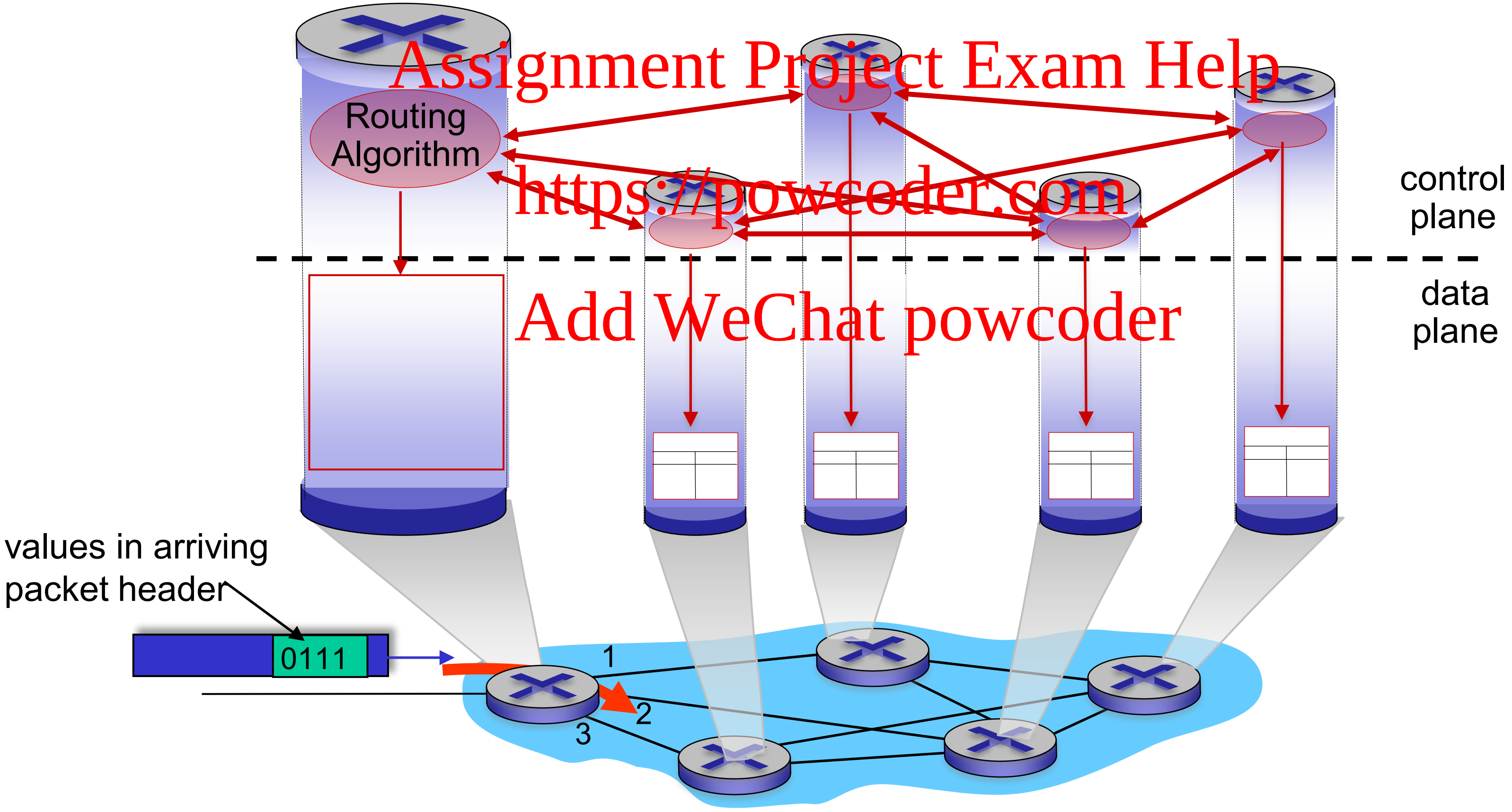
- Two Key Features to the Network Layer as a whole:
  - Forwarding (move packets from a router's input link to appropriate router output link)
  - Routing (determine route taken by packets from source to destination)

Two approaches to structuring network control plane:

- per-router control (traditional)
- logically centralized control (software defined networking)
- Can anyone give some guesses as to advantages/disadvantages?

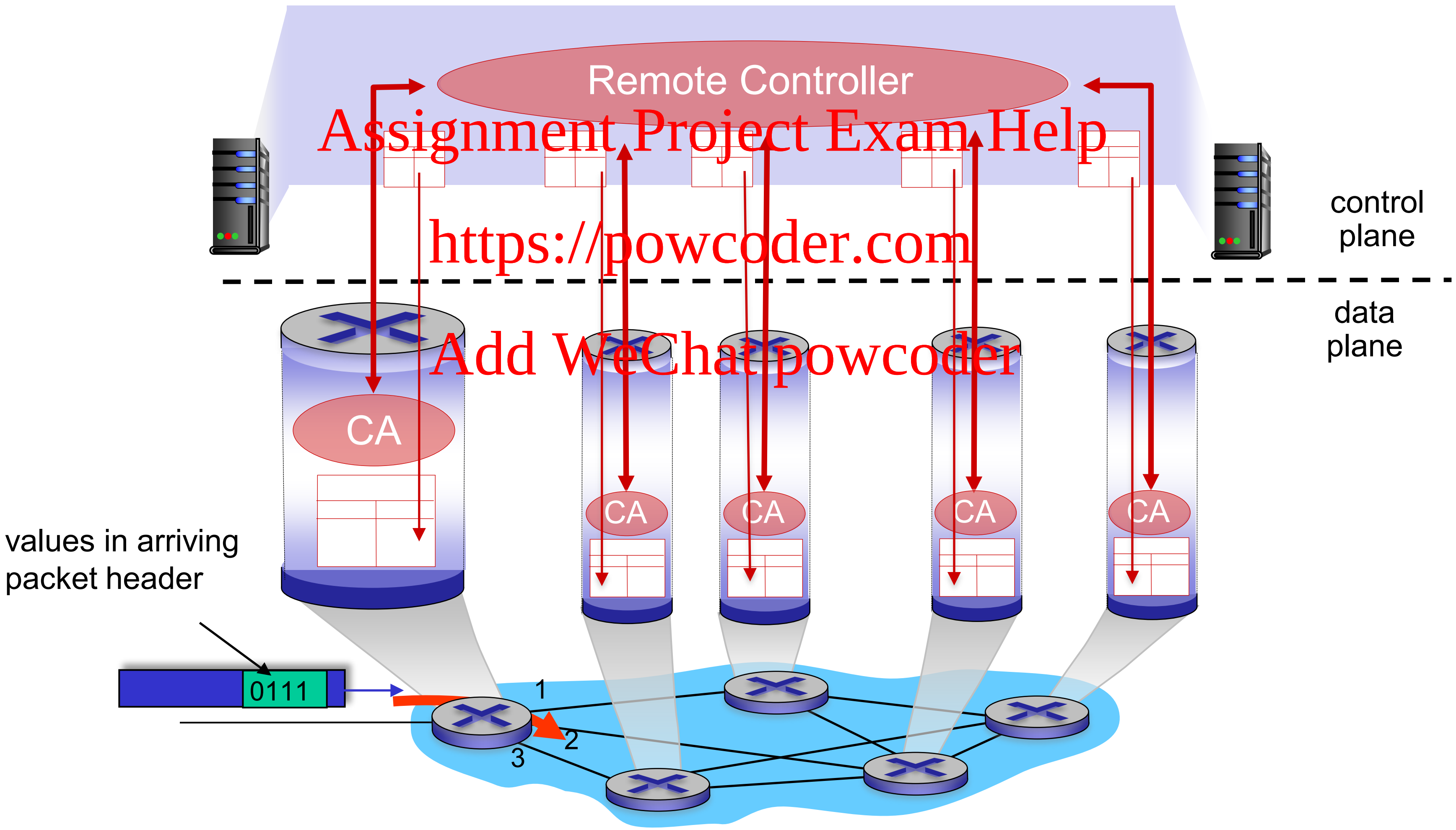
Per-router control plane

Individual routing algorithm components *in each and every router* interact in the control plane – aka each router runs something like Dijkstra's



Software-Defined Networking (SDN) control plane

Remote controller computes, installs forwarding tables in routers



What is a Routing Protocol?

- The purpose of a routing protocol is to find a ‘good’ (where good can mean least cost, fewest hops, least congested, etc) path from the sender to receiver, through the network.
- The path is the sequence of routers the packet traverses on this journey.
- Non-trivial problem!

Formalizing the Problem

- Definitions: A graph $G(V, E)$ is composed of a set V or vertices and a set E of edges between these vertices. Let $e(v_1, v_2)$ denote the presence of an edge between vertices v_1 and v_2. Each edge e in E has a weight $w(e)$. Let $dist_G(v_1, v_2)$ denote the length of the shortest path between vertices v_1 and v_2 in the graph
- The problem, then, is finding the path with value $dist_G(v_1, v_2)$ for each pair of nodes/vertices.
  - This is the simple version of the problem, where we don't worry about anything like 'this type of packet must go along this route'.

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?

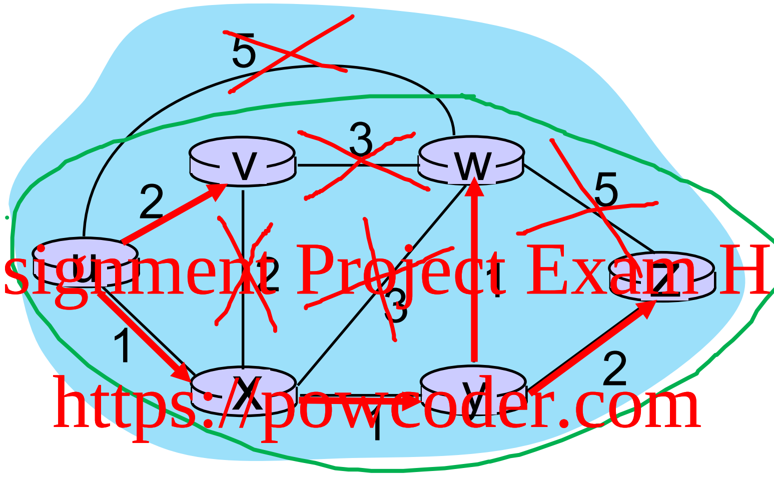
Dijkstra's link-state routing algorithm

- 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

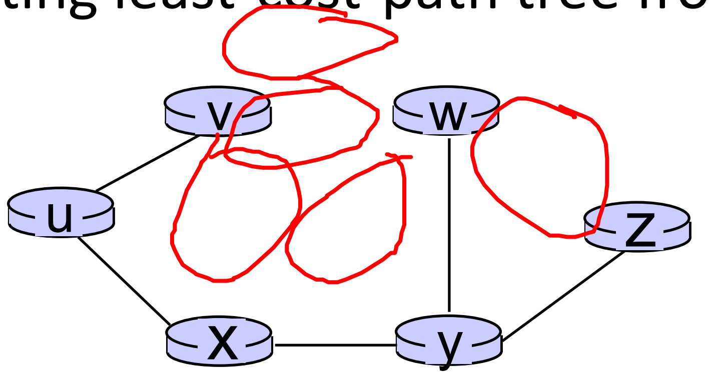
notation

- $c_{x,y}$: direct 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 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: discussion

Alg Complexity: *n* nodes

- each of *n* iteration: need to check all nodes not already in the completed set
- $n(n+1)/2$ comparisons: $O(n^2)$ complexity (where n is the number of nodes)
- 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: $O(n^2)$

Distance Vector Routing

- Key idea is that each node communicates with each other, sending their own distance vector estimates to neighbors

- when x receives new DV estimate from any neighbor, it updates its own DV using Bellman-Ford equation:

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

min taken over all neighbors *v* of *x*

direct cost of link from *x* to *v*

v's estimated least-cost-path cost to *y*

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

Distance vector algorithm:

each node:

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

recompute DV estimates using DV received from neighbor

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

iterative, asynchronous: each local iteration caused by:

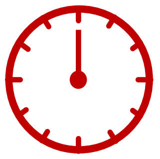
- local link cost change
- DV update message from neighbor

distributed, self-stopping: each node notifies neighbors *only* when its DV changes

- neighbors then notify their neighbors – *only if necessary*
- no notification received, no actions taken!

Distance Vector Discussion

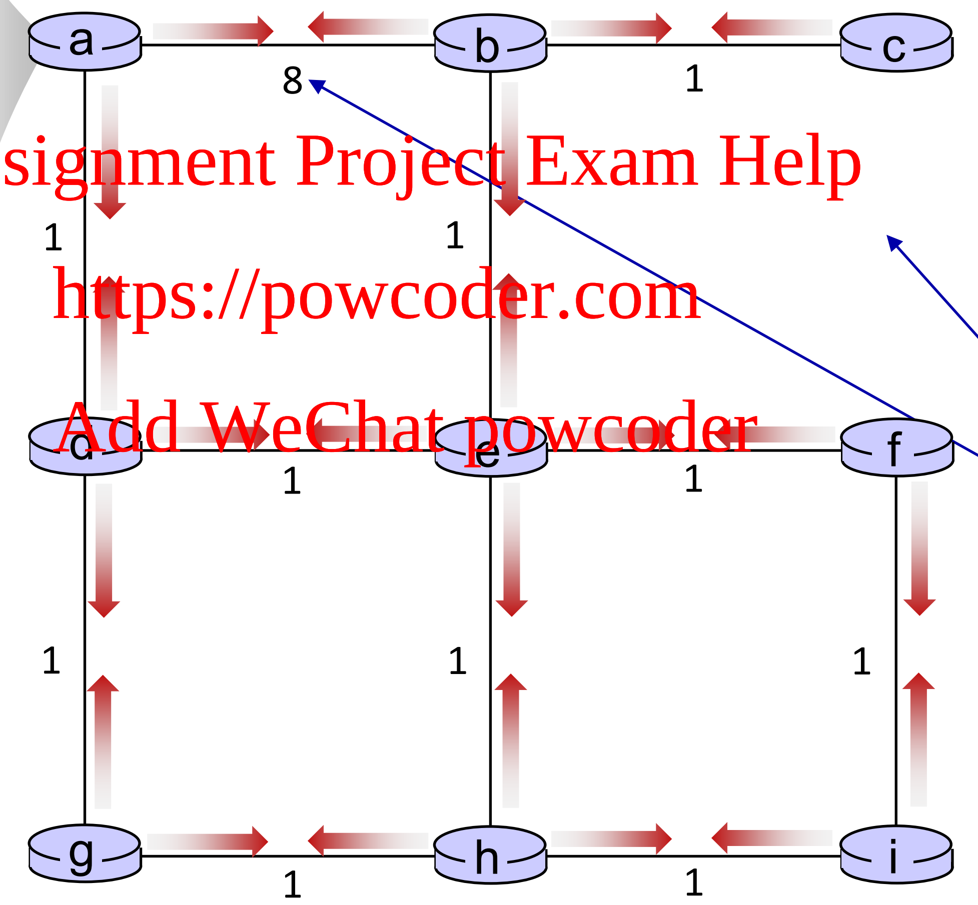
Distance vector: example



t=0

- All nodes have distance estimates to nearest neighbors (only)
- All nodes send their local distance vector to their neighbors

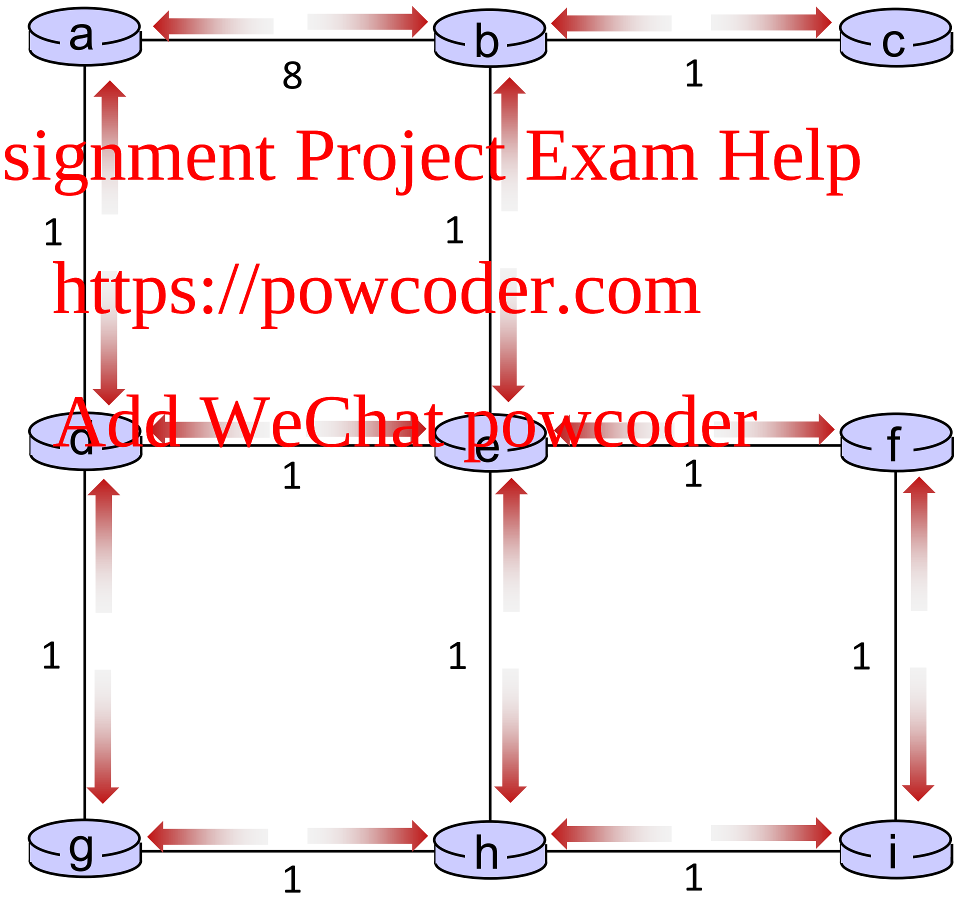
DV in a:
$D_a(a)=0$
$D_a(b) = 8$
$D_a(c) = \infty$
$D_a(d) = 1$
$D_a(e) = \infty$
$D_a(f) = \infty$
$D_a(g) = \infty$
$D_a(h) = \infty$
$D_a(i) = \infty$



A few asymmetries:
- missing link
- larger cost

Distance vector example: iteration

All nodes:
- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors

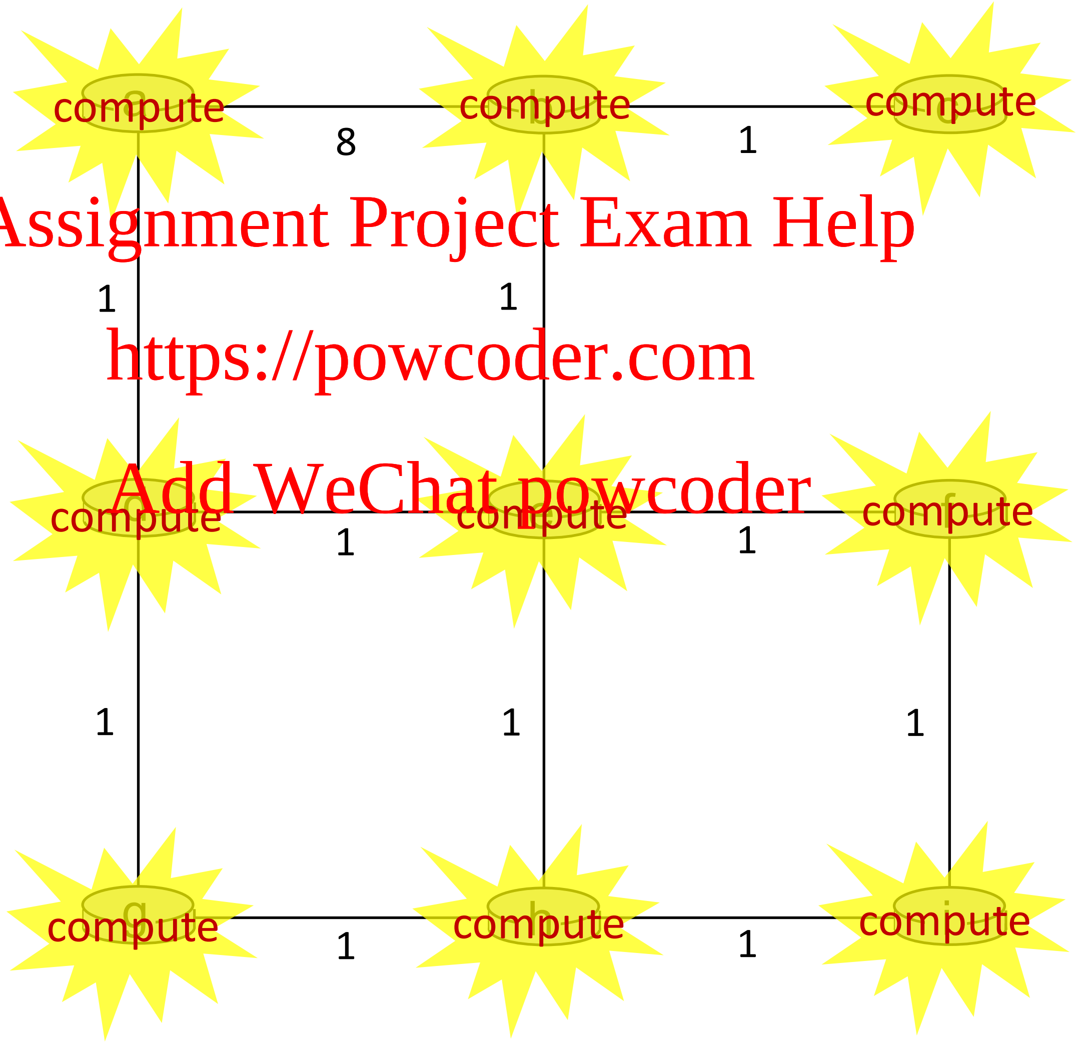


Distance vector example: iteration

t=1

All nodes:

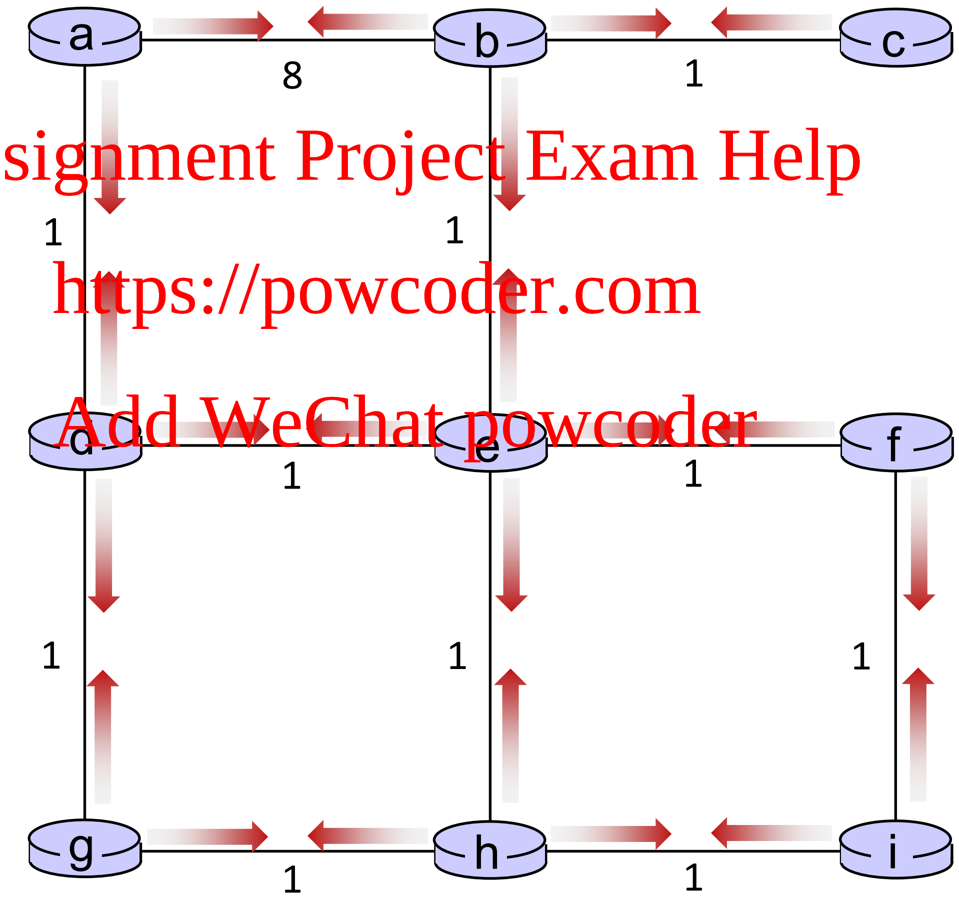
- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



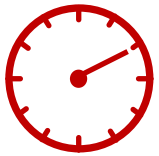
Distance vector example: iteration

t=1

All nodes:
- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors

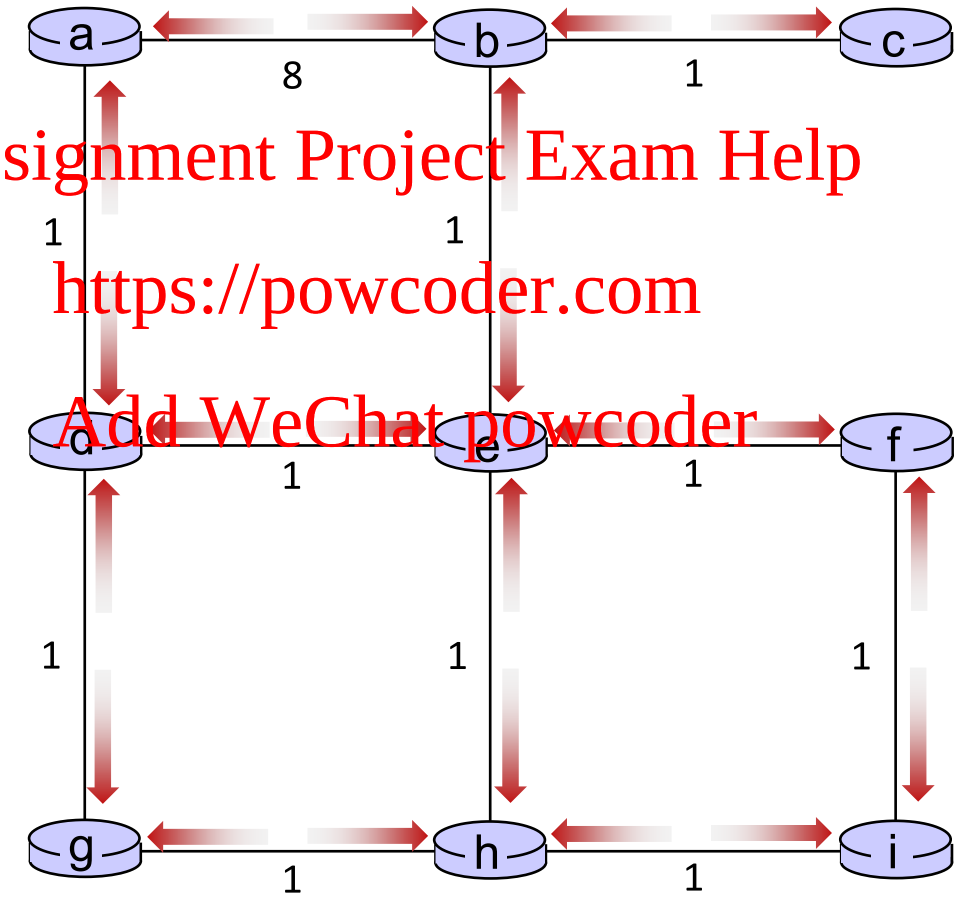


Distance vector example: iteration

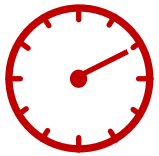


t=2

All nodes:
- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



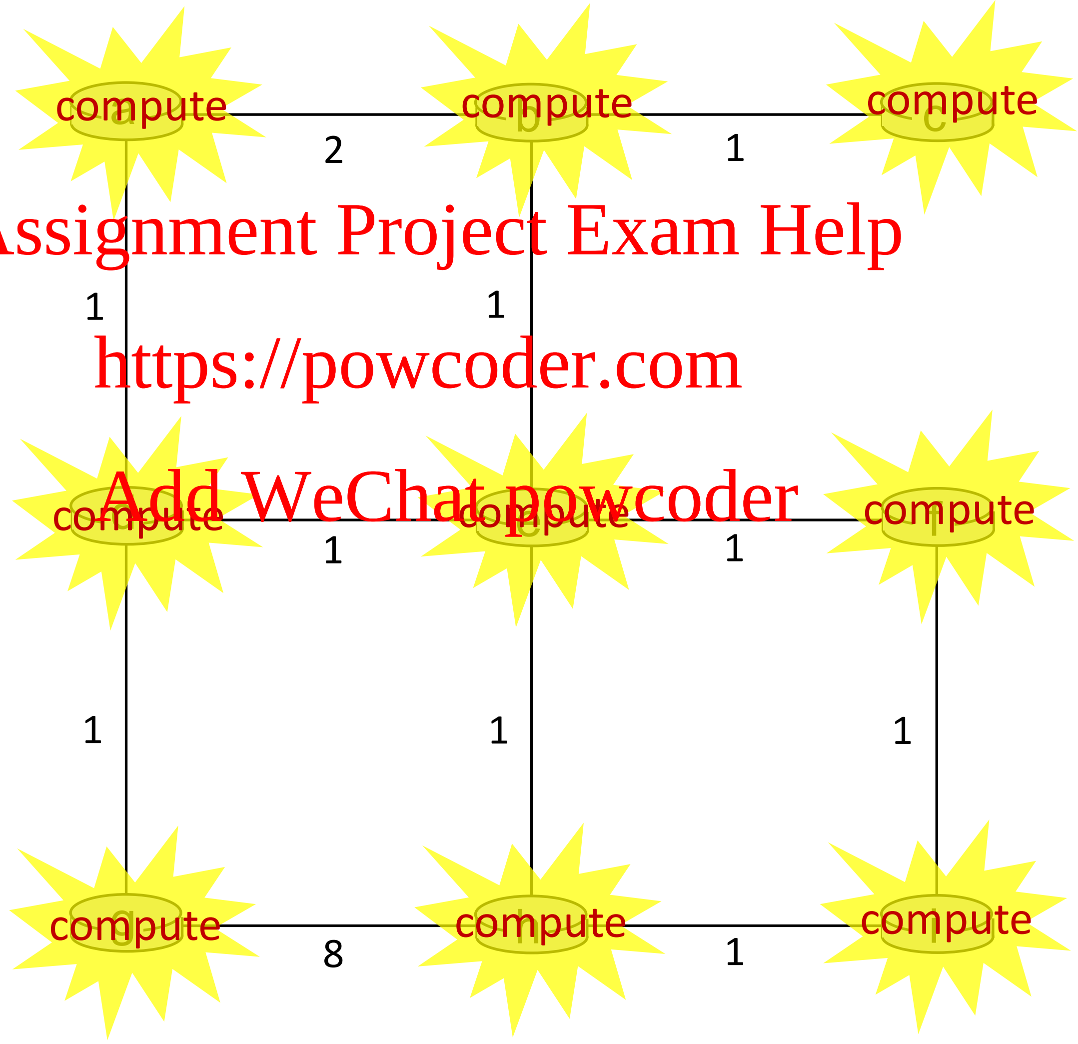
Distance vector example: iteration



t=2

All nodes:

- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors

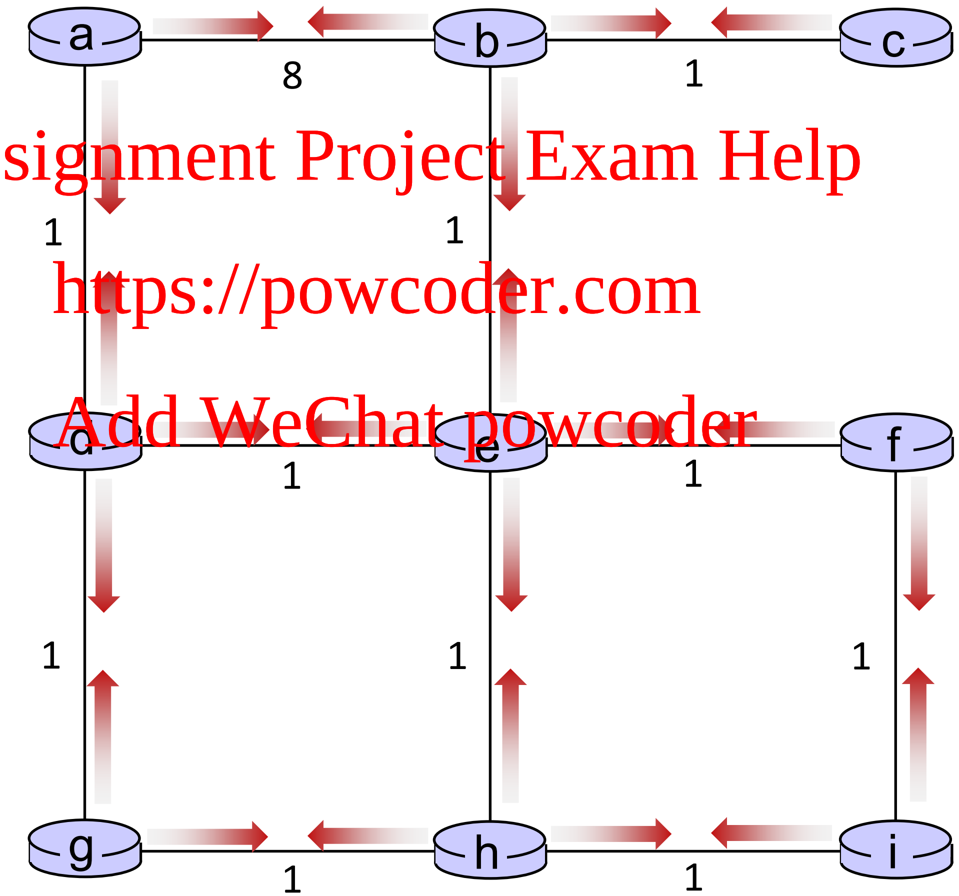


Distance vector example: iteration

t=2

All nodes:

- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



Distance vector example: iteration

…. and so on

Let's next take a look at the iterative *computations* at nodes

Distance vector example:

t=1

- b receives DVs from a, c, e

DV in a:

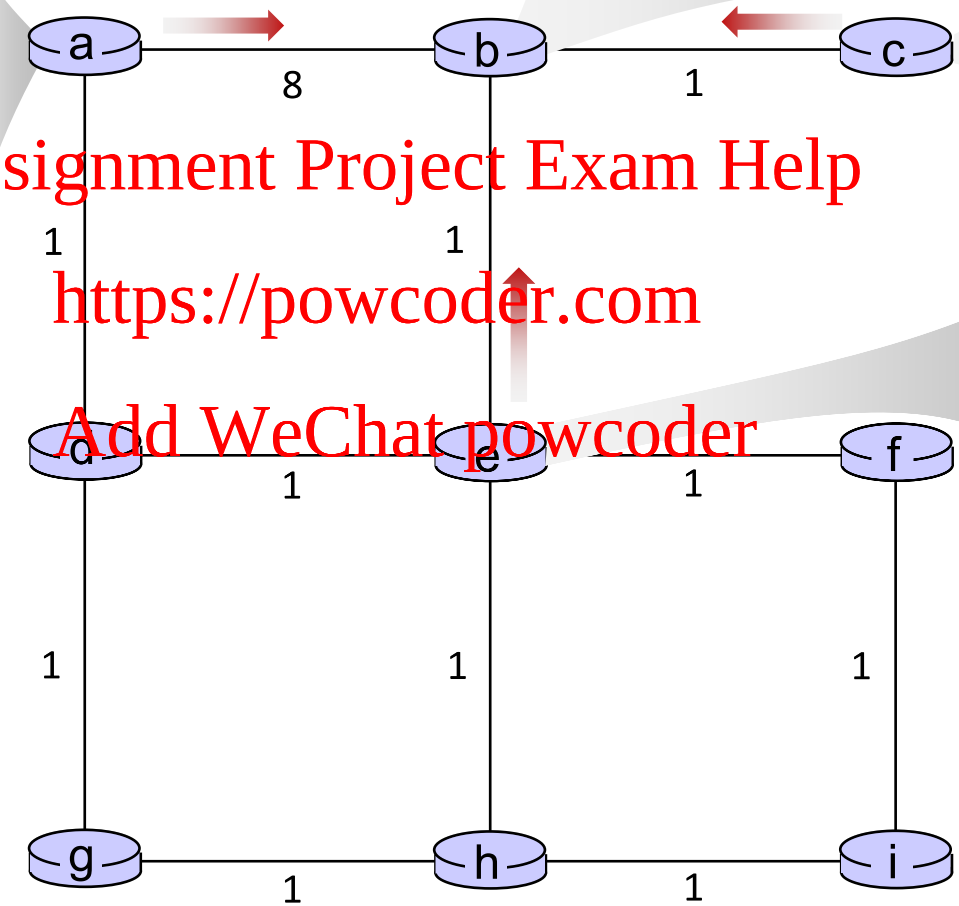
$D_a(a)=0$
$D_a(b) = 8$
$D_a(c) = \infty$
$D_a(d) = 1$
$D_a(e) = \infty$
$D_a(f) = \infty$
$D_a(g) = \infty$
$D_a(h) = \infty$
$D_a(i) = \infty$

DV in b:

$D_b(a) = 8$	$D_b(f) = \infty$
$D_b(c) = 1$	$D_b(g) = \infty$
$D_b(d) = \infty$	$D_b(h) = \infty$
$D_b(e) = 1$	$D_b(i) = \infty$

DV in c:

$D_c(a) = \infty$
$D_c(b) = 1$
$D_c(c) = 0$
$D_c(d) = \infty$
$D_c(e) = \infty$
$D_c(f) = \infty$
$D_c(g) = \infty$
$D_c(h) = \infty$
$D_c(i) = \infty$



DV in e:

$D_e(a) = \infty$
$D_e(b) = 1$
$D_e(c) = \infty$
$D_e(d) = 1$
$D_e(e) = 0$
$D_e(f) = 1$
$D_e(g) = \infty$
$D_e(h) = 1$
$D_e(i) = \infty$

Comparison of LS and DV algorithms

message complexity

LS: n routers, $O(n^2)$ messages sent

DV: exchange between neighbors; convergence time varies

speed of convergence

LS: $O(n^2)$ algorithm, $O(n^2)$ messages
- may have oscillations

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

robustness: what happens if router malfunctions, or is compromised?

LS:
- router can advertise incorrect *link* cost
- each router computes only its *own* table

DV:
- DV router can advertise incorrect *path* cost ("I have a *really* low cost path to everywhere"): black-holing
- each router's table used by others: error propagate thru network

Mininet and Vagrant