Assignment 4

EECS325 Spring 2019

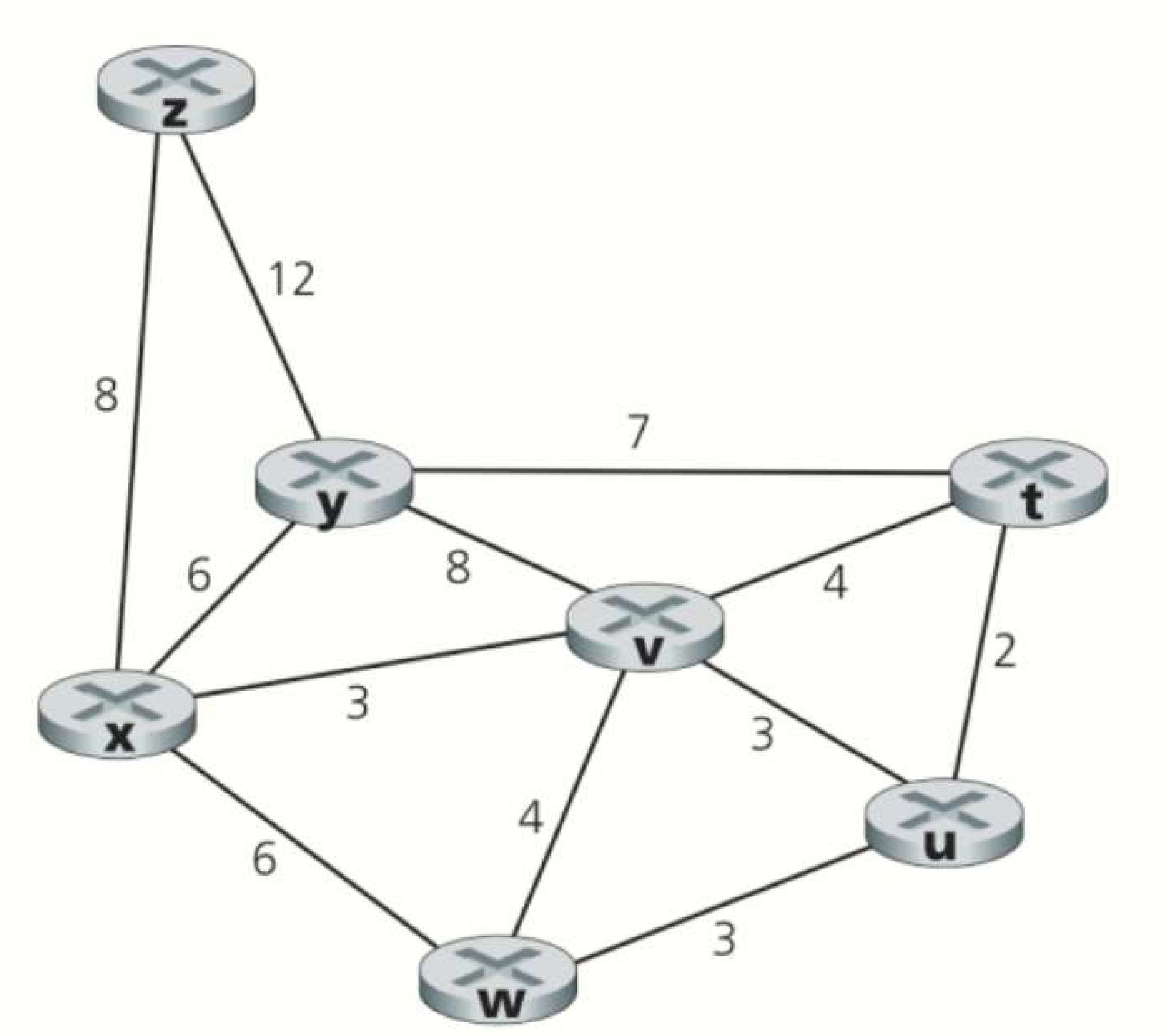
Yue Shu

Due: Tuesday, April 23, 2019

1. Compare and contrast link-state and distance-vector routing algorithms.

- **link-state**
  - global information:
    - all routers have complete topology, link cost info
    - all nodes have the same information
    - accomplished via link state broadcast
  - uses Dijkstra algorithm to calculate the shortest path
  - computes least cost paths from one node to all other nodes
  - iterative: after k iterations, know least cost path to k destinations
  - needs more CPU utilization and more memory space than distance-vector
  - with n nodes, E links, O(nE) messages sent in O(n^2) algorithm, can have oscillations
  - if router malfunctions, node can advertise incorrect link cost
  - each node computes only its own table
- **distance-vector**
  - decentralized information:
    - routers know about physically-connected neighbours, link cost
    - each node share different node tables
  - uses Bellman-Ford equation to calcuulate the cost lowest-cost path from source to destination
  - exchange between neighbours only, convergence time may vary
  - can have count-to-infinity problem
  - if router malfunctions, node can advertise incorrect path cost
  - each node's table is also used by the others, so errors can be propogated through the entire network

2. Consider the following network. With the indicated link costs, use Dijkstra’s shortest-path algorithm to compute the shortest path from x to all network nodes. Show the algorithm by computing a table similar to Slide 15 in Chapter 5.



Step	N'	D(v), p(v)	D(y), p(y)	D(w), p(w)	D(z), p(z)	D(u), p(u)	D(t), p(t)
0	x	3, x	6, x	6, x	8, x	∞	∞
1	xv		6, x	6, x	8, x	6, v	7, v
2	xvy			6, x	8, x	6, v	7, v
3	xvyw				8, x	6, v	7, v
4	xvywu				8, x		7, v
5	xvywut				8, x		
6	xvywutz						

3. Consider the network shown below, and assume that each node initially knows the costs to each of its neighbors. Consider the distance-vector algorithm and show the distance table entries at node z. (Show the distance table entries in each step)

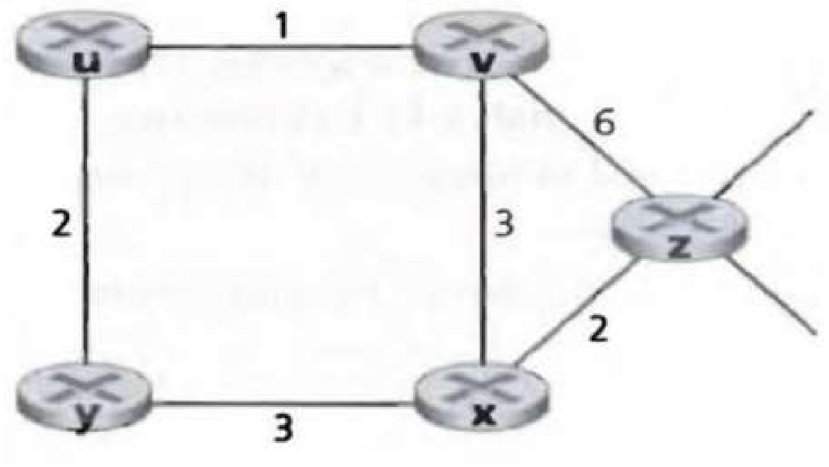


Table of Node z	z	x	v	y	u
z	0	2	6	∞	∞
x	∞	∞	∞	∞	∞
v	∞	∞	∞	∞	∞

Table of Node z	z	x	v	y	u
z	0	2	5	5	7
x	2	0	3	3	∞
v	6	3	0	∞	1

Table of Node z	z	x	v	y	u
z	0	2	5	5	6
x	2	0	3	3	4
v	5	3	0	3	1

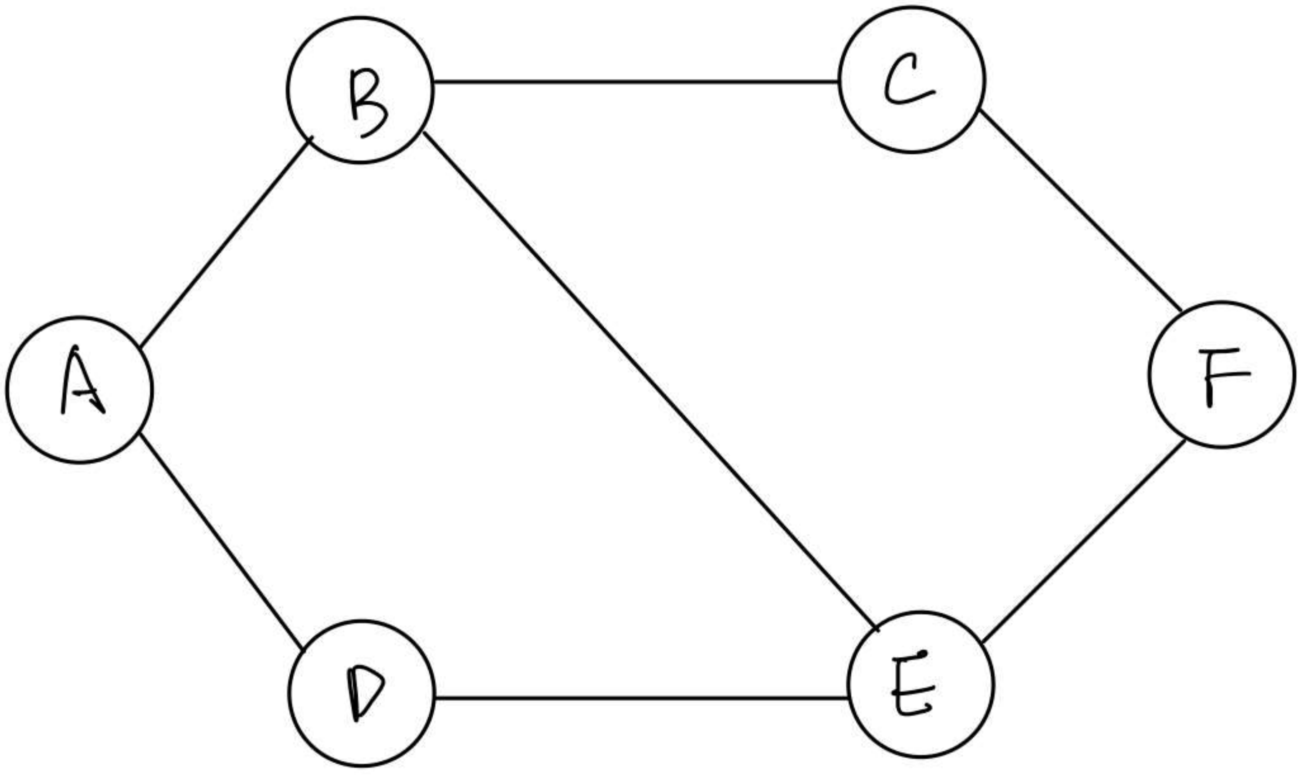
Table of Node z	z	x	v	y	u
z	0	2	5	5	6
x	2	0	3	3	4
v	5	3	0	3	1

↑ ends at the fourth chart, which is the same as the third chart

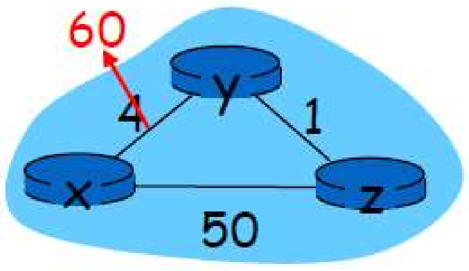
4. Suppose we have the forwarding tables shown in the following table for nodes A and F, in a network where all links have cost 1. Give a diagram of the smallest network consistent with these tables.

	A	
Node	Cost	Nexthop
B	1	B
C	2	B
D	1	D
E	2	B
F	3	D

	F	
Node	Cost	Nexthop
A	3	E
B	2	C
C	1	C
D	2	E
E	1	E



5. Consider the network below. When the link cost c(x, y) increases from 4 to 60. Explain why it takes 44 iterations for the algorithm to stabilize. (Hint: show the distance table entries in each iteration)



According to the distance-vector algorithm, upon each iteration, we have:

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

Then according to this equation, suppose before the first iteration the algorithm has already stabilized, and we only focus on the table of node x, then our first iteration would be:

Table of Node x	x	y	z
x	0	6	5
y	4	0	1

Table of Node x	x	y	z
z	5	1	0

If we continue to follow the same algorithm below, we have our second iteration as:

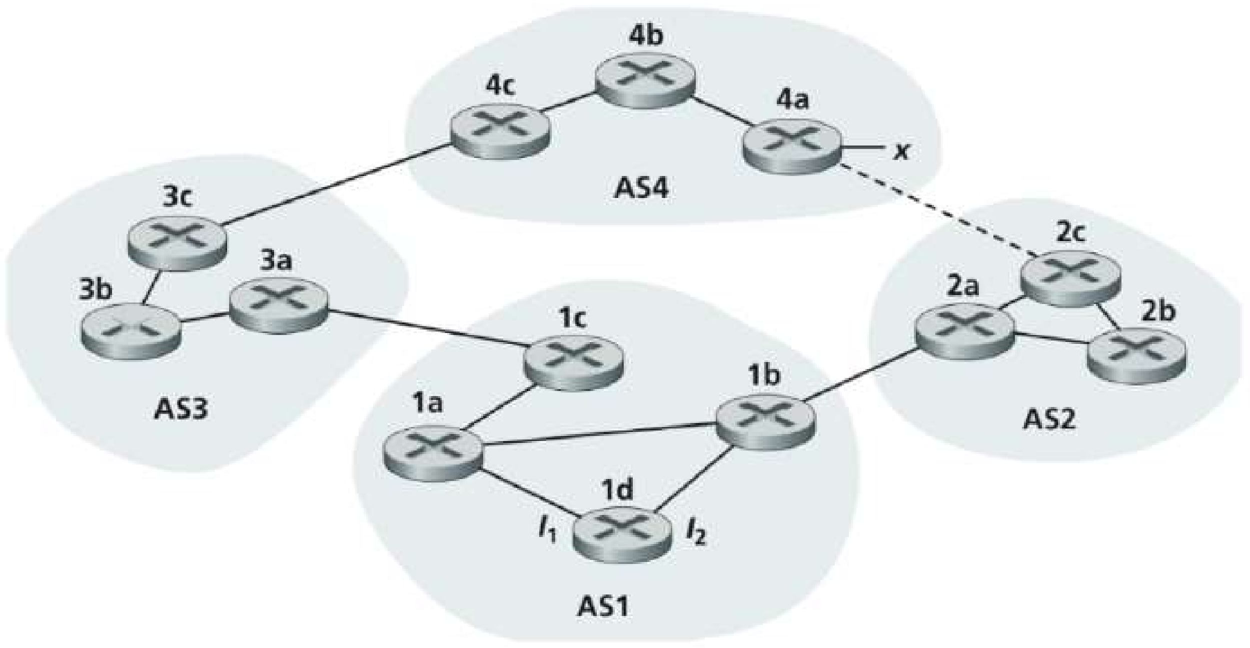
Table of Node x	x	y	z
x	0	6	7
y	6	0	1
z	5	1	0

And our third iteration:

Table of Node x	x	y	z
x	0	8	7
y	6	0	1
z	7	1	0

So for every other iteration, we have $D_x(y)$ or $D_x(z)$ increase by 2 due to the algorithm we have above. We also know that the updating only stop when the distance vector between `x` and `z` reaches 50. This is because the BF equation only cares about the distance cost, instead of the path, which in this case is misled by the preferable "lower" yet unrealistic cost of `xy` and `xz` given the original unupdated `xy` distance. Therefore, the algorithm demands $50 - 7 = 43$ increments in order to reach stablization. Combining the first link change iteration, we have $43 + 1 = 44$ iterations in total.

6. Consider the network shown below. Suppose AS3 and AS2 are running OSPF for their intra-AS routing protocol. Suppose AS1 and AS4 are running RIP for their intra-AS routing protocol. Suppose eBGP and iBGP are used for the inter-AS routing protocol. Initially suppose there is no physical link between AS2 and AS4.



a. Router 3c learns about prefix x from which routing protocol: OSPF, RIP, eBGP, or iBGP?

Reachability information from neighbouring AS `AS4`, thus through eBGP.

b. Router 3a learns about prefix x from which routing protocol?

Reachability information through AS-internal router `3b`, thus through iBGP.

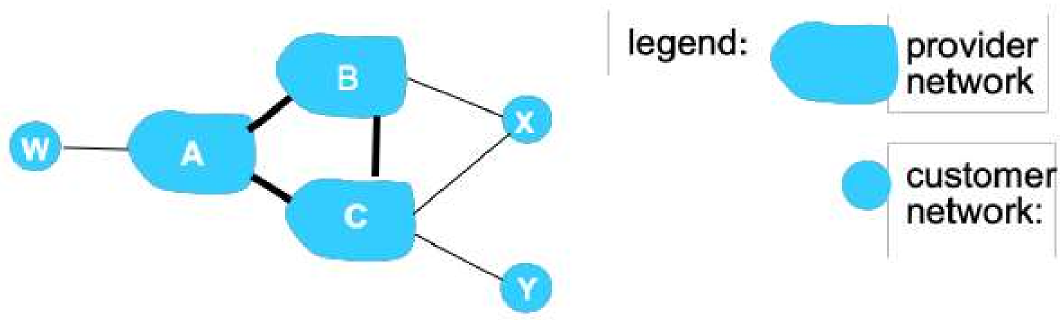
c. Router 1c learns about x from which routing protocol?

Reachability information from neighbouring AS `AS3`, thus through eBGP.

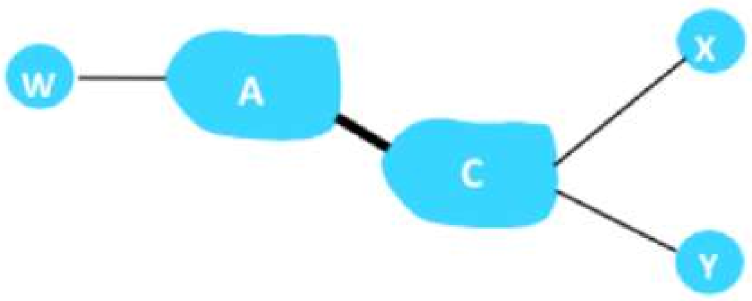
d. Router 1d learns about x from which routing protocol?

Reachability information through AS-internal router `1a` or `1b`, thus through iBGP.

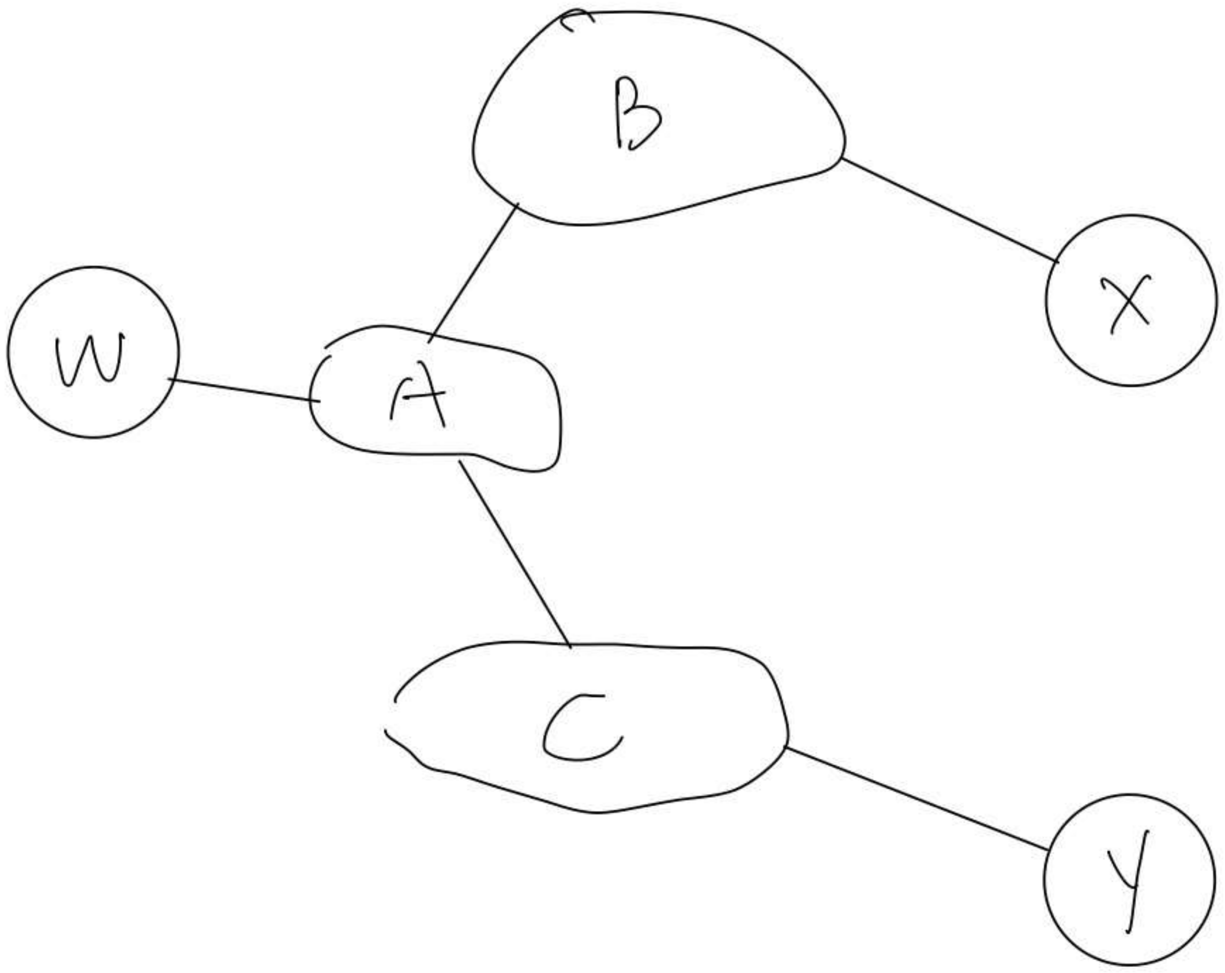
7. In the figure below, X, Y and Z are access ISPs and A, B and C are backbone provider networks. Suppose an ISP only wants to route traffic to/from its customer networks (does not want to carry transit traffic between other ISPs). Another BGP policy X wants to enforce is that X does not want to route from B to C via X.



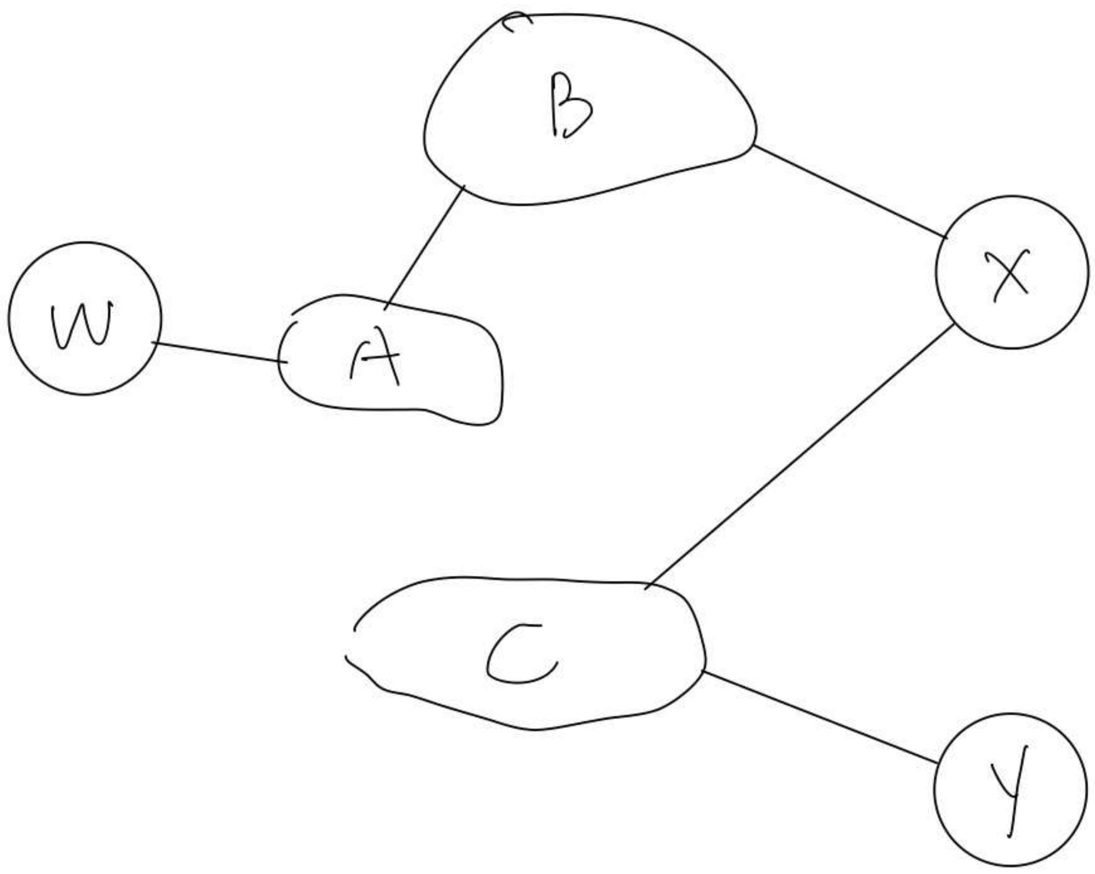
Following is the topology view at Y. Draw the topology views of W and X.



For node `W`:



For node `x`:


B
W
A
X
C
Y