CSC358 Tutorial 7

Julian Sequeira and KyoKeun Park

March 11, 2020

University of Toronto Mississauga

Question 1: Concept Review

(a) What is the difference between the data plane and the control of the network layer?

(b) Does a router have a IP address? If so, how many?

(c) We know that the complexity of Dijkstra's algorithm is $\mathcal{O}(N^2)$ where N is the number of nodes in the graph. We also know that the Internet has (at least) billions of nodes. $N62$ with N being billions is HUGE. How come the Dijkstra's algorithm is still being widely used and is, in fact, quite effective in the Internet?

For the answers, review the lectures, books, go to office hours, and use the discussion board!

Question 2: Forwarding Table

Consider a datagram network using 32-bit host addresses. Suppose a router four link interfaces, numbered 0 through 3, and packets are to be forwarded to the link interfaces as follows:

Destination Address Range	Link Interface
11100000 00000000 00000000 00000000 through *11100000 00111111 11111111 11111111*	*0*
11100000 01000000 00000000 00000000 through *11100000 01000000 11111111 11111111*	*1*
11100000 01000001 00000000 00000000 through *11100000 01111111 11111111 11111111*	*2*
otherwise	*3*

Provide a forwarding table that uses longest prefix matching, and forwards packets to the correct link interface. Your table should have as few entries as possible.

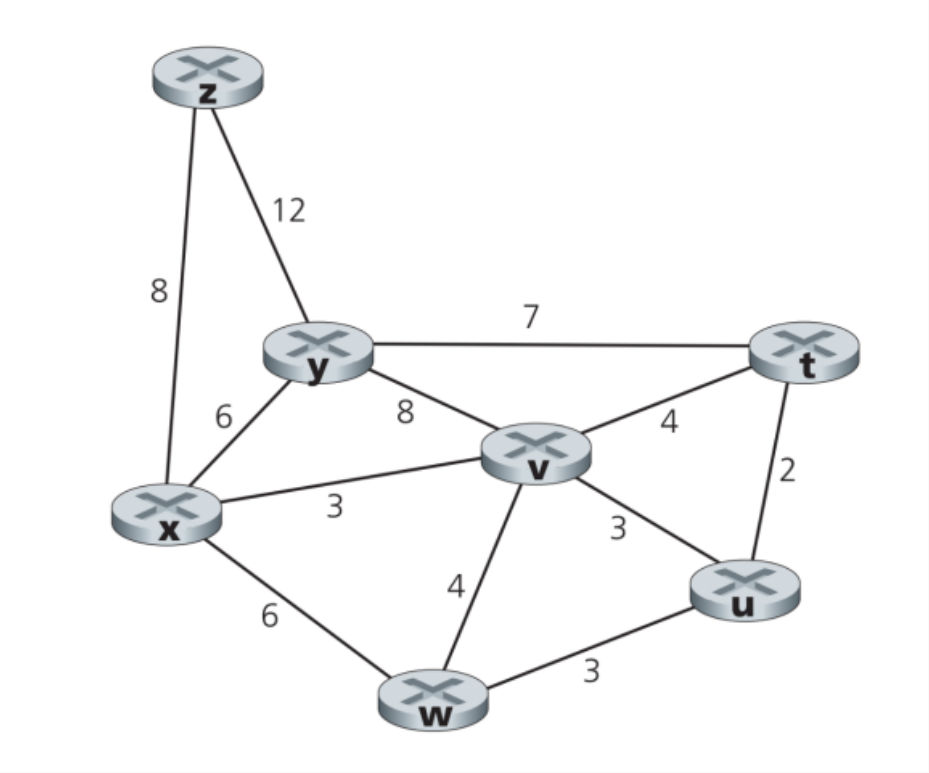
Question 2: Forwarding Table

Prefix Match	Link Interface
11100000 00	0
11100000 01000000	1
1110000	2
11100001 1	3
otherwise	3

The algorithm iterates through the IP string one bit at a time. If all other interfaces are disqualified, or you have the longest matching ”interface key”, you can match the IP to a link interface.

Question 3: Link-State Routing Algorithm

Consider the following network. With the indicated link costs, use Dijkstra's algorithm to compute the shortest path from *x* to all network nodes. Show your work



Question 3: Link-State Routing Algorithm

Basic Idea:

(a) Choose the unvisited vertex with the current shortest distance from x, let this vertex be a

(b) Mark a as visited, record the current distance $d(x, a)$ as the final shortest distance from x to a

(c) Look at all the neighbours n of d, if $d(x, a) + d(a, n)$ is less than the current value of $d(x, n)$, then set $d(x, n) = d(x, a) + d(a, n)$

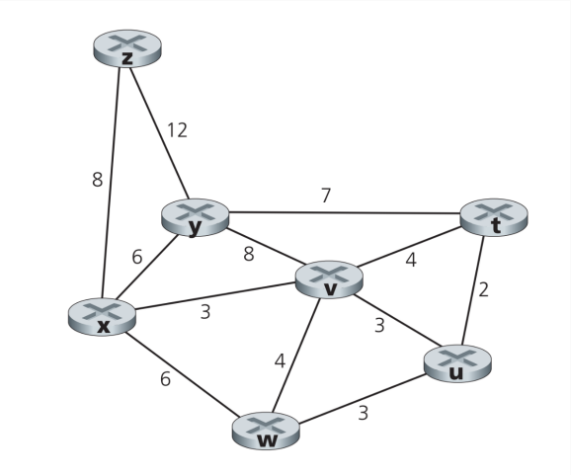
(d) Repeat until you have no unvisited vertices

Question 3: Link-Statement Routing Algorithm

Let N' be the nodes travelled, $D(n)$ be the distance to the given vertex, and $p(n)$ be the second-last vertex visited in that path. Then we get the following graph:

$Step = 0, N' = x$

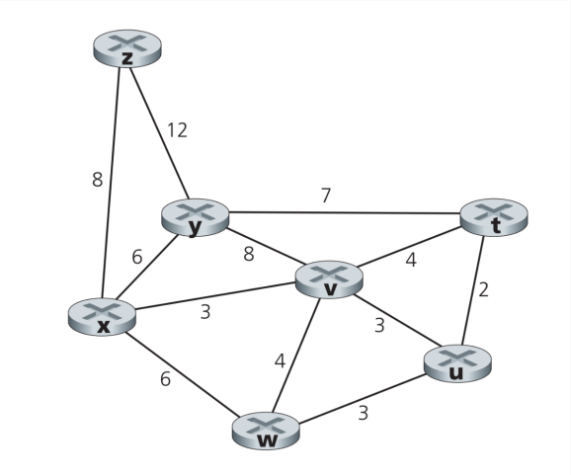
n	$D(n)$	$p(n)$
t	∞	∞
u	∞	∞
v	3	x
w	6	x
y	6	x
z	8	x



Question 3: Link-Statement Routing Algorithm

$Step = 1, N' = xv$

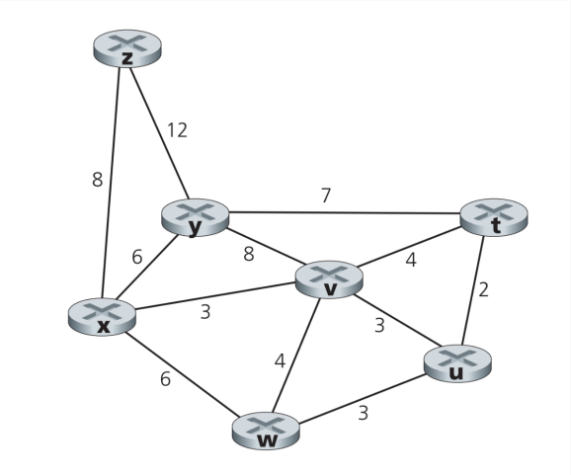
n	$D(n)$	$p(n)$
t	7	v
u	6	v
~~v~~	3	x
w	6	x
y	6	x
z	8	x



Question 3: Link-Statement Routing Algorithm

$Step = 2, N' = xvu$

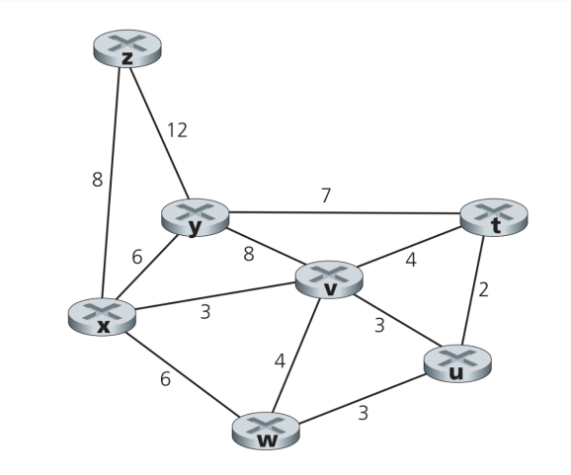
n	$D(n)$	$p(n)$
t	7	v
~~u~~	6	v
~~v~~	3	x
w	6	x
y	6	x
z	8	x



Question 3: Link-Statement Routing Algorithm

$Step = 3, N' = xvuw$

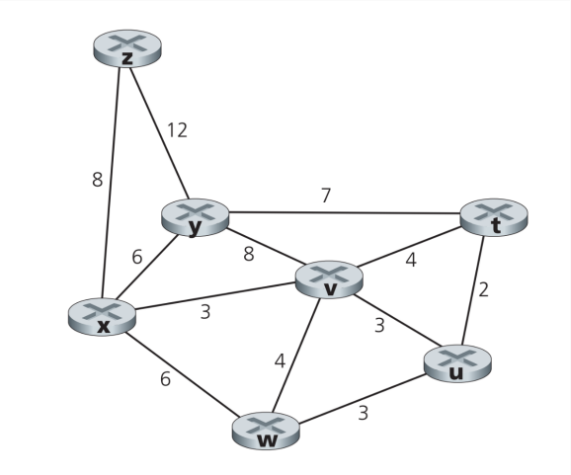
n	$D(n)$	$p(n)$
t	7	v
~~u~~	6	v
~~v~~	3	x
~~w~~	6	x
y	6	x
z	8	x



Question 3: Link-Statement Routing Algorithm

$Step = 4, N' = xvuwy$

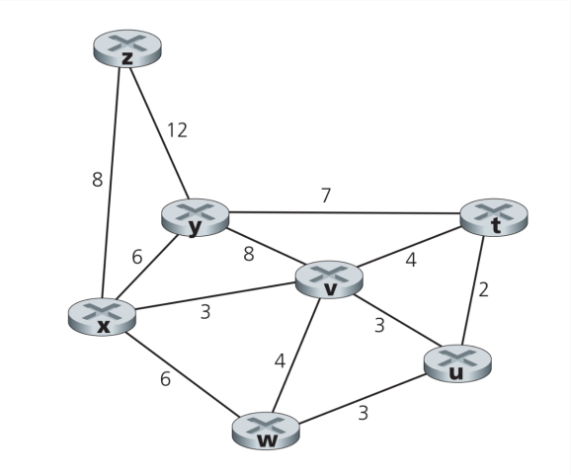
n	$D(n)$	$p(n)$
t	7	v
~~u~~	6	v
~~v~~	3	x
~~w~~	6	x
~~y~~	6	x
z	8	x



Question 3: Link-Statement Routing Algorithm

$Step = 5, N' = xvuwyt$

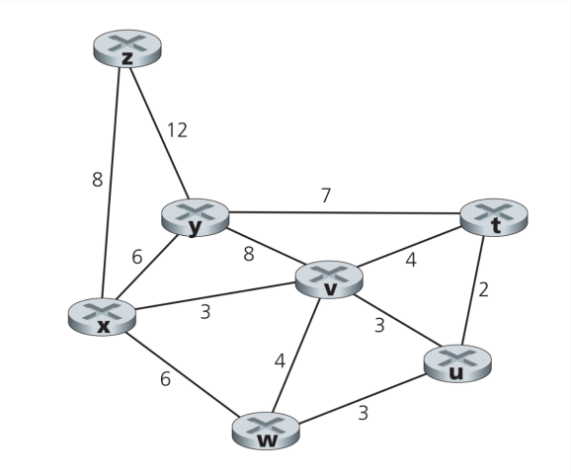
n	$D(n)$	$p(n)$
~~t~~	7	v
~~u~~	6	v
~~v~~	3	x
~~w~~	6	x
~~y~~	6	x
z	8	x



Question 3: Link-Statement Routing Algorithm

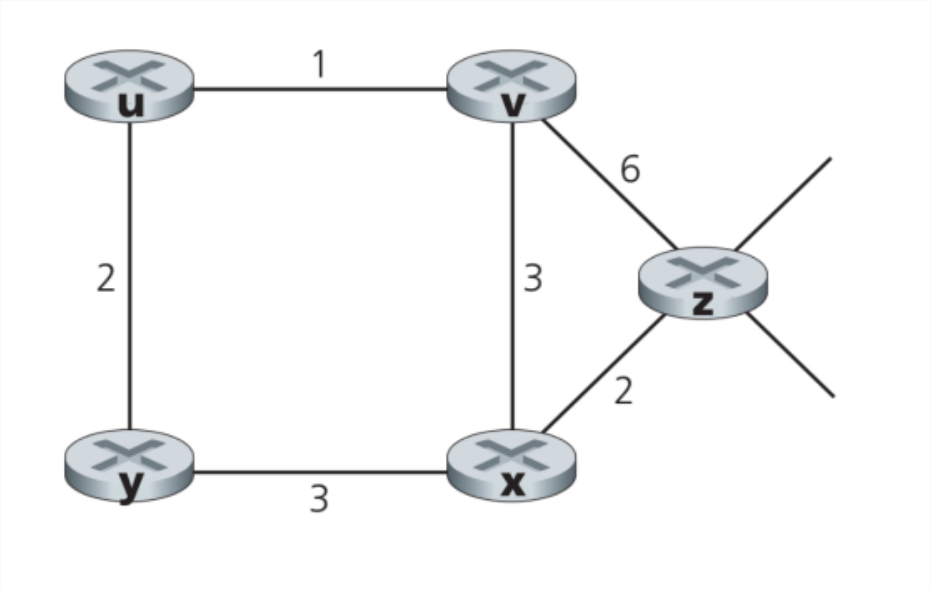
$Step = 6, N' = xvuwytz$

n	$D(n)$	$p(n)$
~~t~~	7	v
~~u~~	6	v
~~v~~	3	x
~~w~~	6	x
~~y~~	6	x
~~z~~	8	x



Question 4: Distance-Vector Routing Algorithm

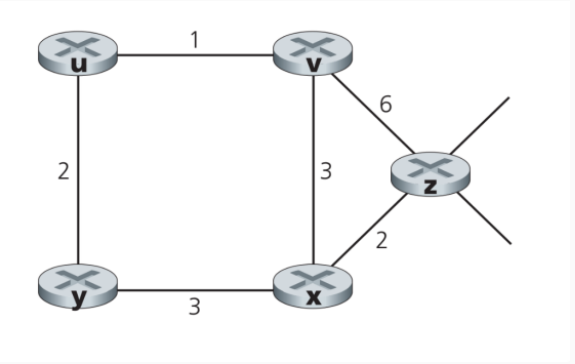
Consider the network shown below, and assume that each node initially knows the cost to each of its neighbours. Consider the distance-vector algorithm and show the change of the distance table entries at node *z*.



Question 4: Distance-Vector Routing Algorithm

The node *z* will only know the cost to each of its neighbors (*v* and *x*) initially:

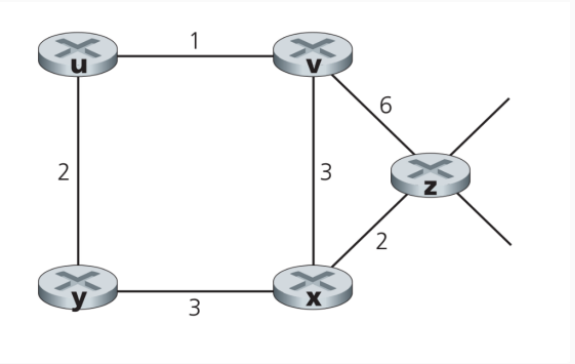
		Cost to				
		u	v	x	y	z
From	v	∞	∞	∞	∞	∞
	x	∞	∞	∞	∞	∞
	z	∞	6	2	∞	0



Question 4: Distance-Vector Routing Algorithm

We can now check for other neighbors using *x* and *v*, as well as look for cheaper costs for existing routes. Through these nodes, we can also find routes to *u* and *y*. Red-coloured values are for new routes found, and blue-coloured values are for cheaper costs of existing routes:

		Cost to				
		u	v	x	y	z
From	v	1	0	3	∞	6
	x	∞	3	0	3	2
	z	7	5	2	5	0



Question 4: Distance-Vector Routing Algorithm

Now that we have all of the nodes, we can find the all optimal routes:

		Cost to				
		u	v	x	y	z
From	v	1	0	3	3	5
	x	4	3	0	3	2
	z	6	5	2	5	0