CS5222 Computer Networks and Internets

Tutorial 8 (Week 9)

Prof Weifa Liang

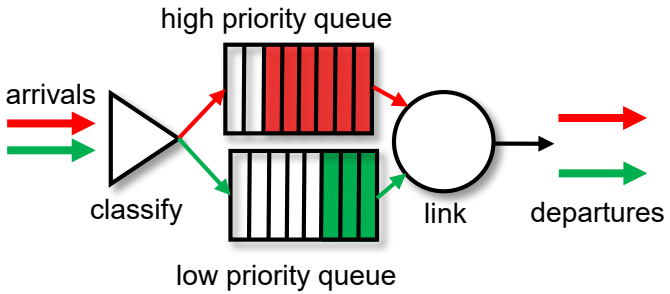
Weifa.liang@cityu-dg.edu.cn

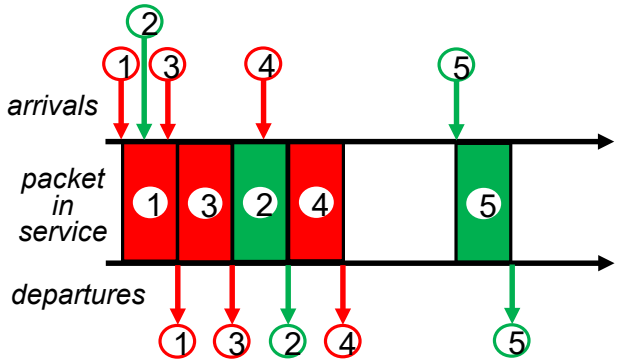
Slides based on book *Computer Networking: A Top-Down Approach.*

Revision

Scheduling policies: priority

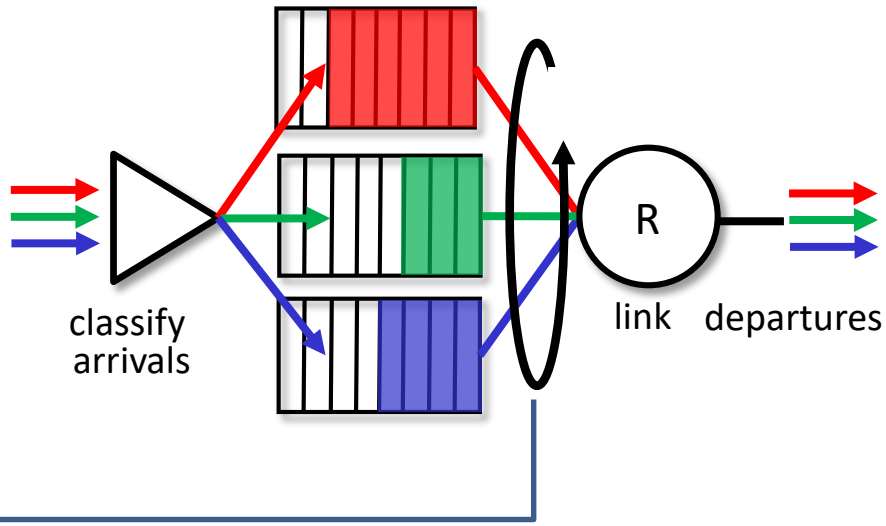
- arriving traffic classified, queued by class
  - any header fields can be used for classification
- send packet from highest priority queue that has buffered packets
  - FCFS within priority class



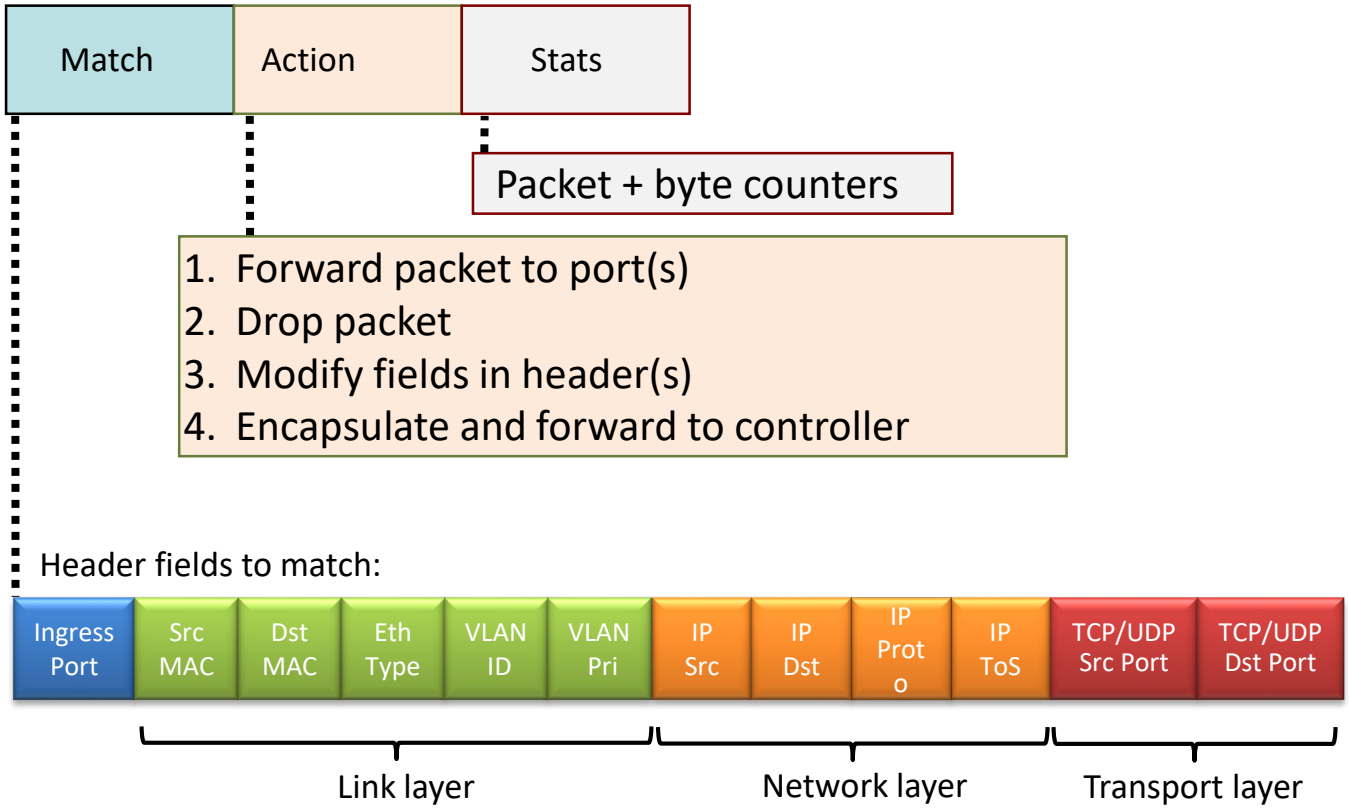


Scheduling policies: round robin (RR)

- arriving traffic classified, queued by class
  - any header fields can be used for classification
- Process class queues in cyclic fashion
- sending one (complete) packet from each class (if available) in turn



OpenFlow: flow table entries



Packets that do not match any rules are dropped in production systems!

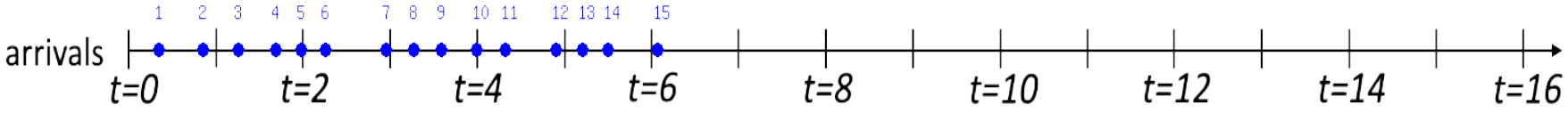
Dijkstra's link-state routing algorithm

```
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
14     v or known least-cost-path to w plus direct-cost from w to v */
15   until all nodes in N'
```

notation

- $c_{x,y}$: direct link cost from node x to y; $= \infty$ 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

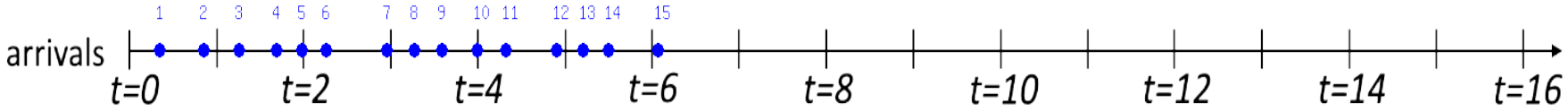
1. Consider the arrival of 15 packets to an output link at a router in some interval of time, as indicated by the figure below. We'll consider time to be "slotted", with a slot beginning at t = 0, 1, 2, 3, etc. At the beginning of each time slot, the packet scheduler will choose one packet, among those queued (if any), for transmission according to the packet scheduling discipline (that you will select below). Each packet requires exactly one slot time to transmit, and so a packet selected for transmission at time t, will complete its transmission at t+1, at which time another packet will be selected for transmission, among those queued. You can see the assignment of packets to classes 1,2, and 3 below.



Packets (#: time): 1: 0.35, 2: 0.85, 3: 1.25, 4: 1.68, 5: 1.97, 6: 2.24, 7: 2.93, 8: 3.24, 9: 3.55, 10: 3.95, 11: 4.28, 12: 4.86, 13: 5.16, 14: 5.44, 15: 6

Packets (#: class): 1: 2 , 2: 2 , 3: 2 , 4: 2 , 5: 2 , 6: 2 , 7: 2 , 8: 1 , 9: 1 , 10: 1 , 11: 2 , 12: 2 , 13: 2 , 14: 2 , 15: 3

a. Schedule the transmissions using a **priority** policy, where class 1 has the highest priority and class 3 the lowest priority.



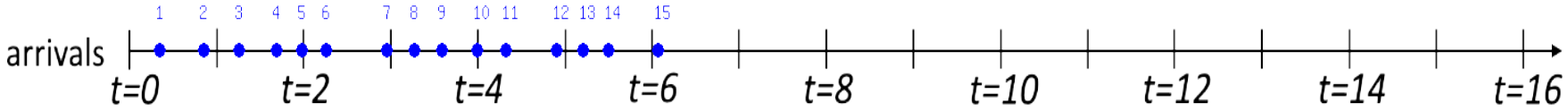
Packets (#: time): 1: 0.35, 2: 0.85, 3: 1.25, 4: 1.68, 5: 1.97, 6: 2.24, 7: 2.93, 8: 3.24, 9: 3.55, 10: 3.95, 11: 4.28, 12: 4.86, 13: 5.16, 14: 5.44, 15: 6

Packets (#: class): 1: 2 , 2: 2 , 3: 2 , 4: 2 , 5: 2 , 6: 2 , 7: 2 , 8: 1 , 9: 1 , 10: 1 , 11: 2 , 12: 2 , 13: 2 , 14: 2 , 15: 3

Answer: Packets are sent out as follows:

time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
pkt	1	2	3	8	9	10	4	5	6	7	11	12	13	14	15

b. Schedule the transmissions using a **round-robin** policy.



Packets (#: time): 1: 0.35, 2: 0.85, 3: 1.25, 4: 1.68, 5: 1.97, 6: 2.24, 7: 2.93, 8: 3.24, 9: 3.55, 10: 3.95, 11: 4.28, 12: 4.86, 13: 5.16, 14: 5.44, 15: 6

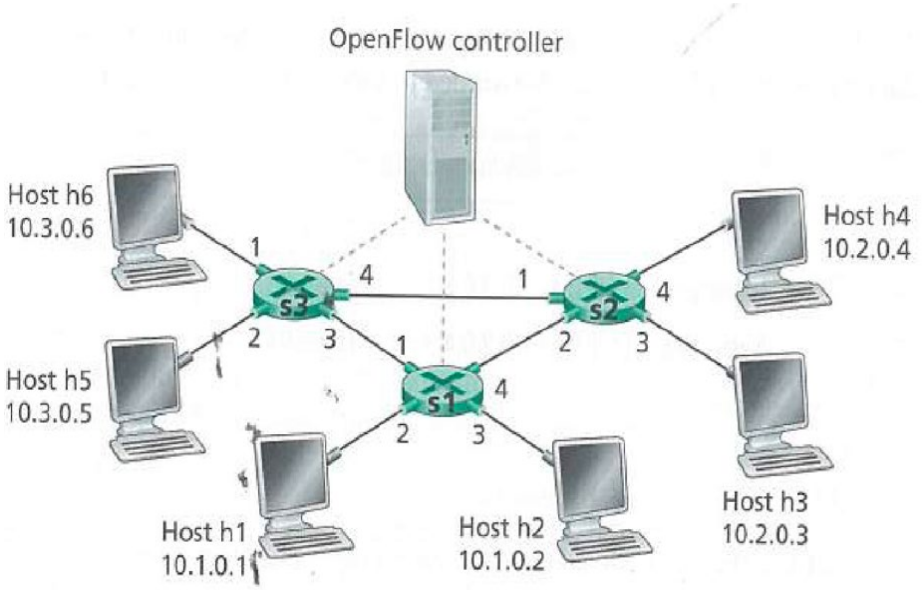
Packets (#: class): 1: 2 , 2: 2 , 3: 2 , 4: 2 , 5: 2 , 6: 2 , 7: 2 , 8: 1 , 9: 1 , 10: 1 , 11: 2 , 12: 2 , 13: 2 , 14: 2 , 15: 3

Answer: Packets are sent out as follows:

time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
pkt	1	2	3	8	4	9	5	15	10	6	7	11	12	13	14

2. Consider the **SDN OpenFlow** network shown below.

a. Give the flow tables at packet switch s2 to achieve the following behavior:



i. Any datagrams arriving on port 1 from hosts h5 or h6 that are destined to hosts h1 or h2 should be forwarded over output port 2.

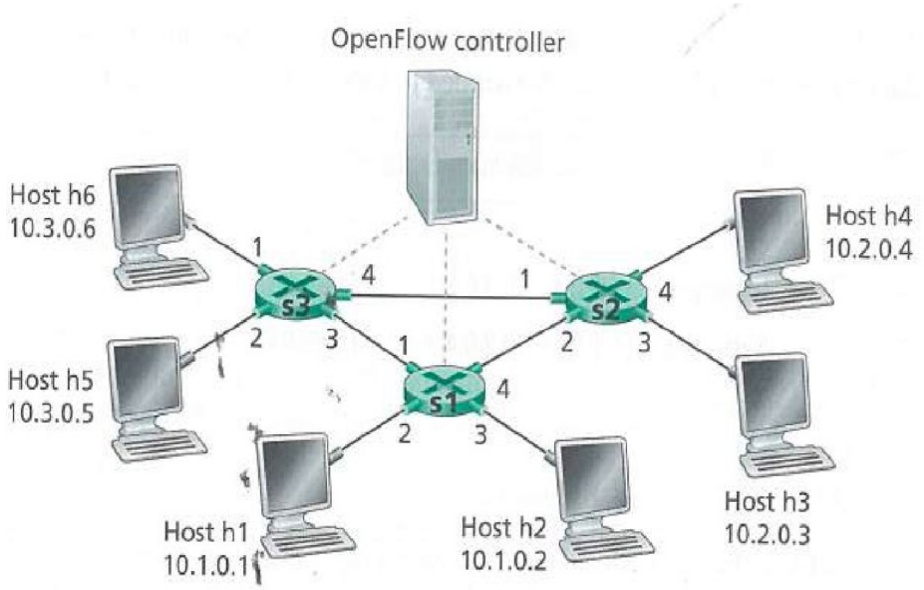
ii. Any datagrams arriving on port 2 from hosts h1 or h2 that are destined to hosts h5 or h6 should be forwarded over output port 1.

iii. Hosts h3 and h4 should be able to send datagrams to each other.

S2 Flow Table	
Match	**Action**
Ingress Port = 1; IP Src = 10.3.*.*; IP Dst = 10.1.*.*	Forward (2)
Ingress Port = 2; IP Src = 10.1.*.*; IP Dst = 10.3.*.*	Forward (1)
Ingress Port = 4	Forward (3)
Ingress Port = 3	Forward (4)

2. Consider the **SDN OpenFlow** network shown below.

a. Give the flow tables at packet switch s2 to achieve the following behavior:

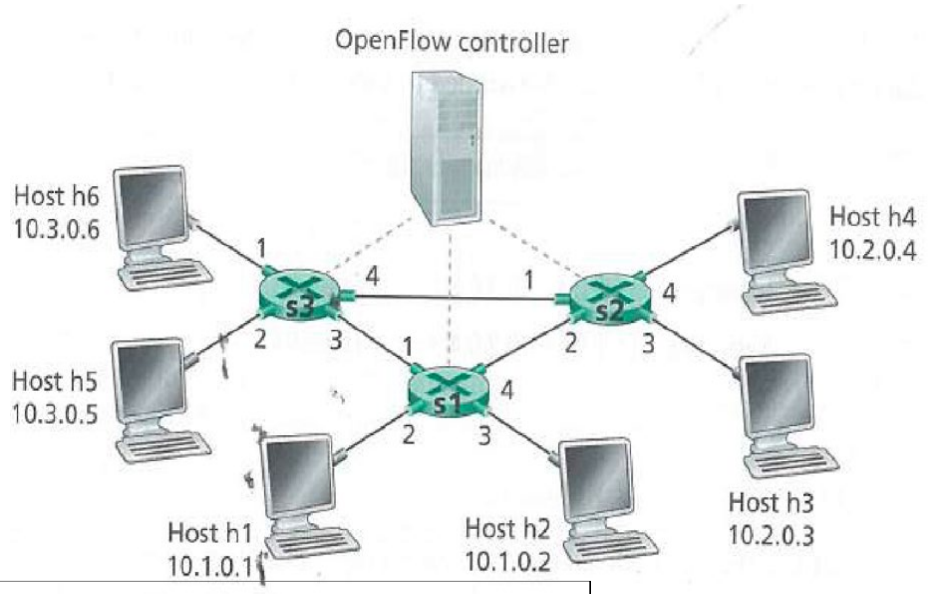


iv. Any datagrams arriving from host h3 and destined for h1, h2, h5, or h6 should be forwarded in clockwise direction in the network.

Match	Action
Ingress Port = 3; IP Dst = 10.1.*.*	Forward (2)
Ingress Port = 3; IP Dst = 10.3.*.*	Forward (2)

2. Consider the **SDN OpenFlow** network shown below.

b. Give the flow table entries at s1 and s3 such that any arriving datagrams with a source address of h3 or h4 are routed to the destination hosts specified in the destination address field in the IP datagram.



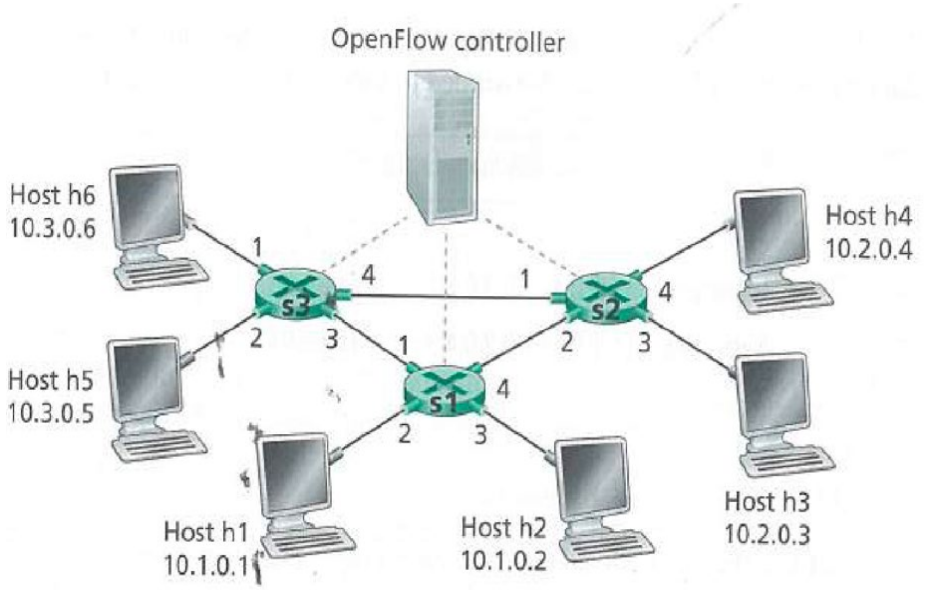
S1 Flow Table	
Match	**Action**
IP Src = 10.2.*.*; IP Dst = 10.1.0.1	Forward (2)
IP Src = 10.2.*.*; IP Dst = 10.1.0.2	Forward (3)
IP Src = 10.2.*.*; IP Dst = 10.3.*.*	Forward (1)

S3 Flow Table	
Match	**Action**
IP Src = 10.2.*.*; IP Dst = 10.3.0.6	Forward (1)
IP Src = 10.2.*.*; IP Dst = 10.3.0.5	Forward (2)
IP Src = 10.2.*.*; IP Dst = 10.1.*.*	Forward (3)

2. Consider the **SDN OpenFlow** network shown below.

c. Give a flow table for s2 that implements the following firewall behavior:

i. Only traffic arriving from hosts h1 and h6 should be delivered to hosts h3 or h4. Arriving traffic from hosts h2 and h5 is blocked.



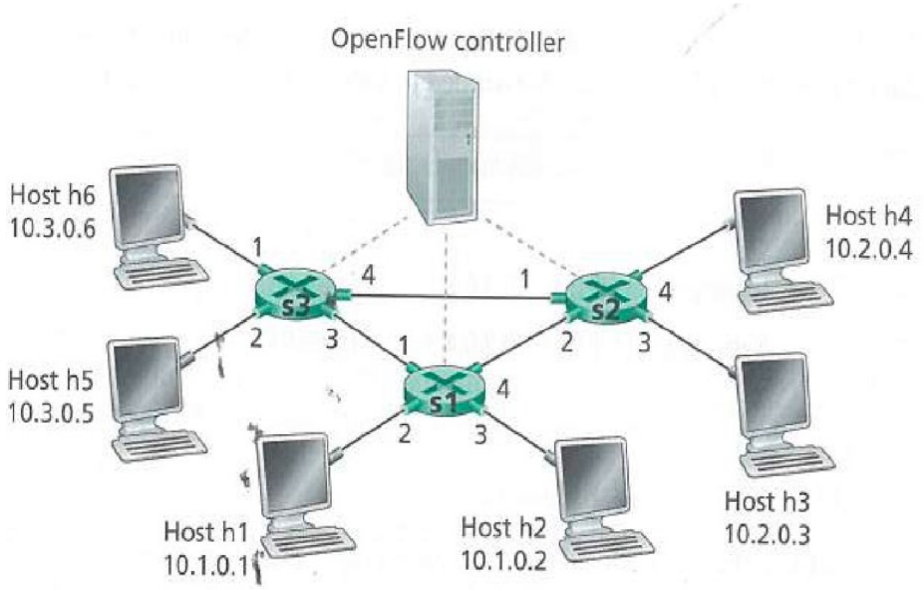
S2 Flow Table	
Match	**Action**
IP Src = 10.1.0.1; IP Dst = 10.2.0.3	Forward (3)
IP Src = 10.1.0.1; IP Dst = 10.2.0.4	Forward (4)
IP Src = 10.3.0.6; IP Dst = 10.2.0.3	Forward (3)
IP Src = 10.3.0.6; IP Dst = 10.2.0.4	Forward (4)

Packets that do not match any rules are dropped in production systems!

2. Consider the **SDN OpenFlow** network shown below.

c. Give a flow table for s2 that implements the following firewall behavior:

ii. Only TCP traffic is allowed to be delivered to hosts h3 or h4. (All UDP traffic is blocked.)

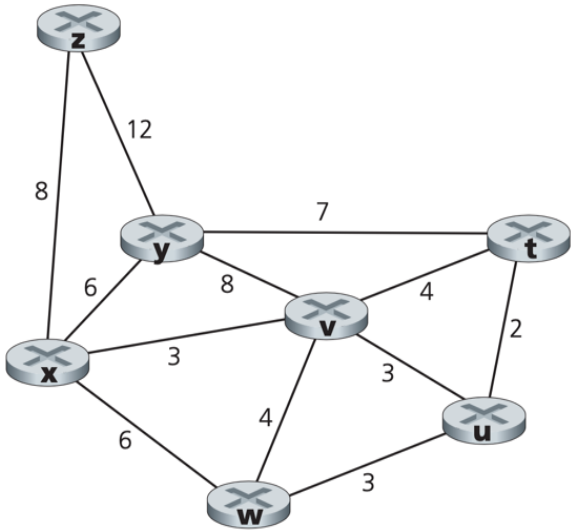


S2 Flow Table	
Match	**Action**
IP Src =.*.*.*.*; IP Dst = 10.2.0.3; IP proto = TCP	Forward (3)
IP Src =.*.*.*.*; IP Dst = 10.2.0.4; IP proto = TCP	Forward (4)

Or

IP Src=.*.*.*.*; IP Dst = 10.2.*.*; IP proto = UDP	Drop

3. Consider the following network of routers. With the indicated link costs, use the link state algorithm (Dijkstra's algorithm) to compute the least cost path from x to all other nodes. Format your answer as a table.



Step	*N'*	*D(t),p(t)*	*D(u),p(u)*	*D(v),p(v)*	*D(w),p(w)*	*D(y),p(y)*	*D(z),p(z)*
0	x	∞	∞	3,x	6,x	6,x	8,x
1	xv	7,v	6,v	3,x	6,x	6,x	8,x
2	xvu	7,v	6,v	3,x	6,x	6,x	8,x
3	xvuw	7,v	6,v	3,x	6,x	6,x	8,x
4	xvuwy	7,v	6,v	3,x	6,x	6,x	8,x
5	xvuwyt	7,v	6,v	3,x	6,x	6,x	8,x
6	xvuwytz	7,v	6,v	3,x	6,x	6,x	8,x