**Final Exam**

Computer Networks Fall 2021

Prof. Cheng-Fu Chou

**Question 1: SDN (15%)**

1. (5%) Please explain the components of the SDN architectures.   
   Figure 5.14: Components of the SDN architectures: SDN-controlled switches, the SDN controller, network-control applications.

1. (5%) Please give two of the most well-known protocols used by SDN Controllers to communicate with the switches/routers.

OpenFlow, OVSDB, (Netconf, SNMP)

1. (5%) Please give two of the most well-known SDN controllers   
   ODL (OpenDaylight), ONOS controller

**Question 2: High Speed TCP (15%)**

Consider TCP over long fat pipes.

1. (5%) If we want to achieve average 10 Gbps throughput by using 1000-byte segment over a 100ms RTT connection, what is the average size of W (the average congestion window size)?

By using the TCP throughput eqn = 0.75\*W/RTT, So W~= 166666, the average size of W = 3/4\*166666 ~= 125000

1. (5%) According to the TCP throughput formula as a function of the loss rate (L), the round-trip time (RTT) and the maximum segment size (MSS), what is the segment loss probability that Today’s TCP congestion-control algorithm could tolerate?

By using the TCP throughput eqn = 1.22\*MSS/ (RTT\*sqrt(L)), So L ~= 9.5 \*10^(-11)

1. (5%) How do we design for a high-speed TCP for such long and fat network environment?  
   - Fast adjusting the number of segments send by TCP for each RTT (iteration) (not using slow start , congestion avoidance …. by using the above TCP eqn., TCP friendly)

* Others (if the approach is reasonable)

**Question 3: Reliable Broadcast Channel (20%)**

Consider a scenario in which a host, A, wants to simultaneously send messages to hosts B and C. A is connected to B and C via a broadcast channel --- a packet sent by A is carried by the channel to both B and C Suppose that broadcast channel connecting A, B, and C can independently lose and corrupt messages (and so, for example , a message sent from A might be correctly received by B, but not by C.) Design a stop-and-wait-like error-control protocol for reliably transferring a packet from A to B and C, such that A will not get new data from the upper layer until it knows that both B and C have correctly received the current packet. Give FSM descriptions of A and B. (Hint: The FSM for C should be essentially the same as for B.)

This problem is a variation on the simple stop and wait protocol (rdt3.0). Because the channel may lose messages and because the sender may resend a message that one of the receivers has already received (either because of a premature timeout or because the other

receiver has yet to receive the data correctly), sequence numbers are needed. As in rdt3.0, a 0-bit sequence number will suffice here.

The sender and receiver FSM are shown in Figure 3. In this problem, the sender state indicates whether the sender has received an ACK from B (only), from C (only) or from neither C nor B. The receiver state indicates which sequence number the receiver is waiting for.

hw12

Figure 3. Sender and receiver FSMs.

**Question 4: The Switching Fabric (20%)**

Consider an 8x8 Banyan switch as well as an 8x8 Batcher switch.

1. (10%) Label the input and output ports from 0 to 7 ( upper most as port 0) Four packets arrive to the input side: packet 1 at input port 1, destined to output port 3; packet 2 at input port 3, destined to output port 4; packet 3 at input port 5, destined to output port 2; packet 4 at input port 6, destined to output port 1. Please show the internal paths taken by these four packets. If contention occurred at one switch module, the packet from the upper input port of the switch module will win.
2. (10%) Now assume it is a Batcher-Banyan switch. Consider four incoming packets described in the above question. Which packets will be switched to the output side in the current operation round? Justify your answers.



* + - 1. Banyan switch



* + - 1. Batcher switch



* + - 1. Batcher-Banyan Switch

**Question 5: Routing Algorithms (20%)**

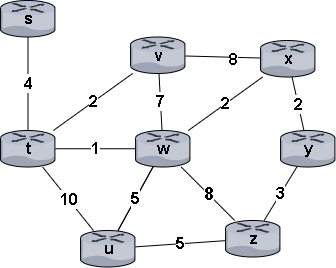
1. (10%) Please give an example to explain what the oscillations is for the link-state routing algorithm.

Figure 5.5: Oscillations with congestion-sensitive routing.

1. (10%) What could be done to prevent such oscillations.
2. One solution is to mandate that link costs not depends on the amount of traffic carried.
3. Another solution is to ensure that not all routers run the LS algorithm at the same time.

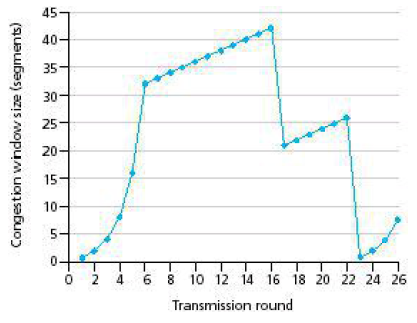
**Question 6: Routing Algorithms (10%)**

Consider the following network. With the indicated link costs, use Dijkstra’s shortest-path algorithm to compute the shortest path from **t** to all network nodes. Show how the algorithm works by computing a table similar to the table in lecture slide 4-81 (Where **D(x)** current value of cost of path from source to destination x, and **p(x)** is predecessor node along path from source to x).



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **step** | **N’** | **D(s),p(s)** | **D(u),p(u)** | **D(v),p(v)** | **D(w),p(w)** | **D(x),p(x)** | **D(y),p(y)** | **D(z),p(z)** |
| **0** | t | 4, t | 10, t | 2, t | 1, t | ∞ | ∞ | ∞ |
| **1** | tw | 4, t | 6, w | 2, t | 1, t | 3, w | ∞ | 9, w |
| **2** | twv | 4, t | 6, w | 2, t | 1, t | 3, w | ∞ | 9, w |
| **3** | twvx | 4, t | 6, w | 2, t | 1, t | 3, w | 5, x | 9, w |
| **4** | twvxs | 4, t | 6, w | 2, t | 1, t | 3, w | 5, x | 9, w |
| **5** | twvxsy | 4, t | 6, w | 2, t | 1, t | 3, w | 5, x | 8, y |
| **6** | twvxsyu | 4, t | 6, w | 2, t | 1, t | 3, w | 5, x | 8, y |
| **7** | twvsyuz | 4, t | 6, w | 2, t | 1, t | 3, w | 5, x | 8, y |

**Question 7: TCP congestion control (homework review) (10%)**



Assuming TCP Reno is the protocol experiencing the behavior shown above, answer the following questions. In all cases, you should provide a short discussion justifying your answer.

1. (3%) After the 16th transmission round, is segment loss detected by a triple duplicate ACK or by a timeout? How about 22nd transmission round?
2. Triple duplicate ACK. Because the congestion window size only becomes half the value after 16th round.
3. Timeout. Because the congestion window size has dropped to 1 after 22th round.
4. (3%) What is the value of *ssthresh* at the 24th transmission round?

13.

The threshold is set to half the value of the congestion window when packet loss is detected. When loss is detected during 22nd transmission round, the congestion windows size is 26. Hence the threshold is 13 during the 24th transmission round.

1. (4%) Suppose TCP Tahoe is used (instead of TCP Reno), and assume that triple duplicate ACKs are received at the 16th round. What is the value of *ssthresh* and how many packets sent out at the 20th round?

21, 8.

Threshold is set to 42/2 = 21. The packet is first reset to 1, and then grows to 8 in 20th round.