COSC 4377 – Networking - Kevin B Long

# interlocking-uh-m-186.eps

Homework #4

Due 11:59am, Friday, 1 March, 2019

We will work these problems in class on Wednesday.

Multiple submissions accepted.

Name:\_\_James Richardson\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Peoplesoft ID: \_\_\_\_\_\_\_1555520\_\_\_\_\_\_\_\_\_\_\_

1. (6 pts) Why is UDP over TCP preferred for streaming content, like radio?

Why? There are two critical reasons and bunch of others that aren’t that important. Try and find the two most important ones.

1. Lower overhead (less transmission delay: send → receive)
2. Doesn’t resend anything that's lost, because applications that use it don’t need the packets re transmitted.
3. (19 pts) Port numbering
   1. (6 pts) What are the three ranges of port numbers set aside by iana? Give their names and number ranges?

1st Range: Official Internet Services number range: 0-255

2nd Range: Other Well Known Service number range: 256-1023

3rd Range: Ephemeral Ports number range: 1024 - 65535

* 1. (3 pts) How many total port numbers are there?

65536

1. (3 pts) How many bits is the field in the UDP and TCP header to storing the port numbers?

Answer: 16

1. (3 pts) What is the type of port number called that is usually assigned to software you and I run as regular users?

Ephemeral Ports

1. (4 pts) If a developer must use UDP but still requires reliability in her protocol, what can she do? What’s the only viable option?

The application has to implement that reliability.

1. (12 pts) The type of checksum TCP and UDP use is called a “1s complement”. Consider these three 8-bit bytes: 01010011, 01100110, 01110100 (a little easier than the usual 16-bit numbers TCP and UDP use).
2. Calculate the 1s complement? (normally we’d use 16-bit numbers.) Show all work.

01010011

01100110

01110100

100101101

00101101

1

00101110

11010001

1. Why do we take the 1s complement of the sum instead of just using the sum?

Because on the other end (receiver) we just sum everything up without the ones compliment. Then that side can just add in the ones compliment to the calculated sum and the result will be all 1s unless there is an error.

1. How will the receiver detect errors in this scheme?

As stated before, if the sum is added to the ones complement and the result is not all 1’s, then there was an error.

1. Can a 1-bit error ever go undetected? What about a 2-bit error?

1 bit: ☐Yes or XNo. 2-bit XYes or ☐No.

1. Calculate the 1s complement of the sum of 11011010 and 01100101.

11011010

01100101

100111111

00111111

1

01000000

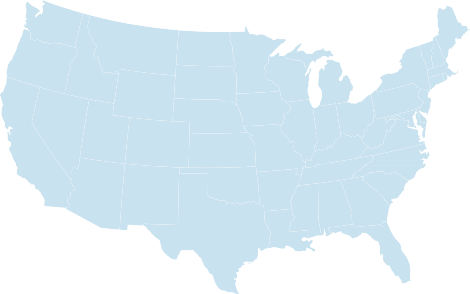
10111111

1. Give an example where a bit is flipped in each of the 2 bytes in the previous problem that does not cause the 1s complement to change.

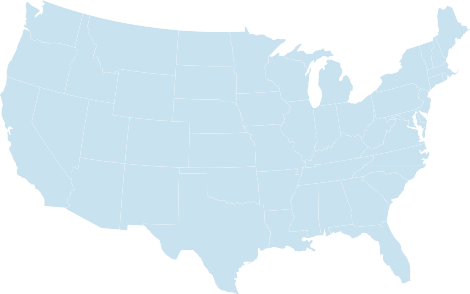
11011010 → 11011011

01100101 → 01100100

1. (5 pts) In section 3.4, the book showed a cross-country network which has a very low efficiency of .00027%. How large does the window size need to be increased to achieve a utilization ≥98%? You need to show the formula you’re using.



Data packet



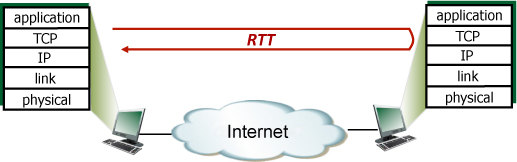
Data packets ACK packets

**A stop-and-wait protocol in operation b. A pipelined protocol in operation**

**Figure 3.17** ♦Stop-and-wait versus pipelined protocol

n = .98 \* 30.012 ms / .012 ms = 2450.98 packets => 2451 packets

1. (18 pts) You will find the following diagram at the TCP RTT calculator at <http://gaia.cs.umass.edu/kurose_ross/interactive/TCP_RTT.php>,

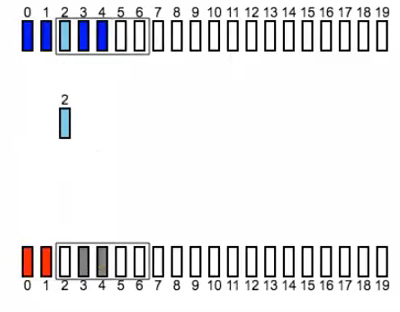
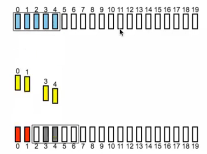
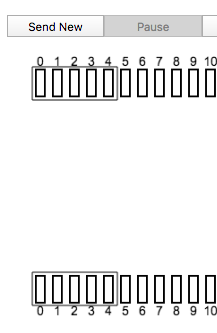


Suppose that TCP's current estimated values for the round trip time (estimatedRTT) and deviation in the RTT (DevRTT) are 210 msec and 36 msec, respectively (see Section 3.5.3 for a discussion of these variables). Suppose that the next three measured values of the RTT are 240, 360, and 230 respectively.

Compute TCP's new value of estimatedRTT, DevRTT, and the TCP timeout value after each of these three measured RTT values is obtained. Use the values of α = 0.125 and β = 0.25.

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable** | **240ms** | **360ms** | **230ms** |
| **Estimated RTT** | 213.75 | 228.75 | 212.5 |
| **DevRTT** | 33.52 | 169.5 | 162.0 |
| **TCP timeout value** | 347.83 | 906.75 | 860.5 |

1. (20 pts) Refer to the Selective Repeat animation tool at <https://media.pearsoncmg.com/aw/ecs_kurose_compnetwork_7/cw/content/interactiveanimations/selective-repeat-protocol/index.html>. Consider the following two scenarios:
2. Five packets 0-4 are sent. Packet #2 is lost. What will the sender do? Include a small snapshot showing how the sender responds that includes the sender and receiver windows from the animation. Something like this, but with the process under way:



The receiver will ack back for each of the packets received, but obviously not for the one that it didn’t get. The sender will only send back packet #2. The sender is able to send the next two packets. But must wait for the timer of packet 2 to expire before sending it again.

1. Five packets 0-4 are sent. ACK #2 is lost. What will the sender do? Again, include a snapshot of how it responds.

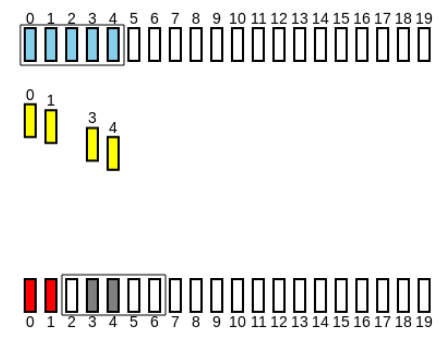
The sender is able to increment the window over, and will be able to send two more aditional packets. But again must wait for packet 2’s timer to expire before it can resend it.

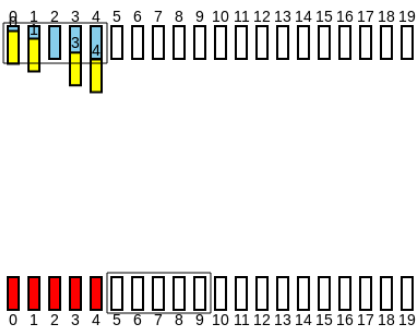
1. Will the sender do anything different for these two cases? Why or why not?

No. From the sender’s point of view its exactly the same, the sender can’t know whats going on with the receiver.

1. Will the receiver window end in a different state immediately after the two losses in the two scenarios? Include snapshots of the two scenarios and a description of what the difference is and why:

Yes, the receiver will see something different. In the first case, the receiver actually doesn’t get packet 2, but in the second case it does, the sender is just not informed of the receiver receiving.





1. In the end, after the receiver and sender reach their final steady state, is there a difference between the two scenarios in the states of the sender and receiver windows?

Yes, the reciever and sender windows will be the same for the first case, but different for the second case. In the second case the receiver will have already shifted the window to the next 5 packets, but the sender will have only shifted his window over 2 positions.