National University of Singapore
School of Computing
CS2105: Introduction to Computer Networks
Semester 1, 2018/2019

Tutorial 3

These questions will be discussed during the next week's discussion group meetings. Please be prepared to answer these questions during the session in class. Some of the questions are taken from the textbook, so please bring it along for reference.

1. Launch your browser and open its network diagnostic tool (e.g. press F12 if you use Chrome on Windows, or Cmd + Opt + I for Mac). Then click the "Network" tab to observe the network communication while you load the following URL in your browser: `http://tiny.cc/nc4xiy`

   Enter your choice and press the "Submit" button.

   (a) Look at the entry named "formResponse". What is the HTTP request method issued?
   Ans: POST

   (b) Briefly explain when HTTP POST and GET methods are used.
   Ans: (From Wikipedia) The POST request method requests that a web server accepts and stores the data enclosed in the body of the request message. It is often used when uploading a file or submitting a completed web form. In contrast, the HTTP GET request method is designed to retrieve information from the server.

2. **[KR, Chapter 2, P21]** Suppose that your department has a local DNS server for all computers in the department. You are an ordinary user (i.e., not a network/system administrator). Can you determine if an external web site was likely accessed from a computer in your department a couple of seconds ago? Explain.

   Ans: You can query the local DNS server to see if it has cached the record for the external web site, e.g., `dig -t a www.abc.com`.

   If the IP address of this web site has been queried by another computer a few seconds ago, your local DNS server have this record in its local cache and is able to answer your query quickly. Otherwise, the query time will be long.

3. **[Modified from KR, Chapter 2, P31]** You are given 4 programs: `TCPEchoClient`, `UDPEchoClient`, `TCPEchoServer` and `UDPEchoServer`. Compile/run them on `sunfire`.

   (a) Suppose you run `TCPEchoClient` before you run `TCPEchoServer`. What happens and why?
   Ans: When the local socket is created, the client will attempt to make a TCP connection to a nonexistent server process. An exception will be thrown.

   (b) Suppose you run `UDPEchoClient` before you run `UDPEchoServer`. What happens and why?
   Ans: A UDP socket does not need to establish any connection when it is created. Thus, it works fine even if you start client program first and then server program.

4. **[KR, Chapter 3, R7]** Suppose a process in Host C has a UDP socket with port number 6789. Suppose both Host A and Host B each sends a UDP segment to Host C with destination port number 6789. Will both of these segments be directed to the same socket at Host C? If

so, how will the process at Host C know that these two segments originated from two different hosts?

Ans: Yes, both segments will be directed to the same socket. The source IP address can be obtained from the socket interface. In Java, the `DatagramPacket` class provides a method `getAddress()` for to obtain the IP address of the source of a packet.

5. **[KR, Chapter 3, R8]** Suppose that a Web server runs in Host C on port 80. Suppose this Web server uses persistent connections, and is currently receiving requests from two different Hosts, A and B. Are all of the requests being sent through the same socket at Host C? If they are being passed through different sockets, do both of these sockets have port 80? Discuss and explain.

   Ans: For each persistent connection, the Web server creates a separate "connection socket". Each connection socket is identified with a four-tuple: (source IP address, source port number, destination IP address, destination port number). When host C receives and IP datagram, it examines these four fields in the datagram/segment to determine to which socket it should pass the payload of the TCP segment. Thus, the requests from A and B pass through different sockets. The identifier for both of these sockets has 80 for the destination port; however, the identifiers for these sockets have different values for source IP addresses. Unlike UDP, when the transport layer passes a TCP segment's payload to the application process, it does not specify the source IP address, as this is implicitly specified by the socket identifier.

6. **[Modified from KR, Chapter 3, P4]**

   (a) Suppose you have the following 2 bytes: `01011100` and `01100101`. What is the 1s complement of the sum of these 2 bytes?

   Ans: Sum: `11000001`, checksum: `00111110`

   (b) Suppose you have the following 2 bytes: 11011010 and 01100101. What is the 1s complement of the sum of these 2 bytes?

   Ans: Sum: `01000000`, checksum: `10111111`

   (Note: UDP and TCP use 16-bit words in computing their checksums. For simplicity you are asked to consider 8-bit checksums in this problem).

7. **[Modified from KR, Chapter 3, P5]** Suppose that UDP receiver computes the checksum for the received UDP segment and finds that it matches the value carried in the checksum field. Can the receiver be absolutely certain that no bit errors have occurred? You may use Q6 as an example to explain.

   Ans: The checksum might not be able to detect if there are more than one bit error. For example, if sender transmits the following two bytes: $01\underline{0}11100$ and $01\underline{1}00101$, and the two underlined bits happens to flip, then checksum remains unchanged and receiver will fail to detect this error.

8. Why is it that UDP takes 1s complement of the sum as checksum, i.e., why not just use sum? (Hint: How can the receiver detect erros with the 1s complement scheme?)

   Ans: This is to simplify the computation at receiver side. To detect errors, receiver adds all the 16-bit words (including the checksum). If the sum contains a zero, then receiver knows there has been an error. All one-bit errors will be detected, but some two-bit errors can remain undetected.