National University of Singapore
School of Computing

CS2105 **Tutorial 3** Answer paper

---

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 network communication.
   Copy-and-paste the following URL in the address bar of your browser:
   http://tiny.cc/atupaz

   Enter your choice and press the "Submit" button.

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

   POST

   b) Briefly explain when HTTP POST and GET methods are used.

   (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.

To tutors:

Please demo it in class.

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.

   You may use 'dig' program to query the local DNS server. For example,

   ```
   dig -t a www.abc.com
   ```

   If IP address of this Web page has been queried by another computer seconds ago, your local DNS server should keep this knowledge in local DNS 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: **TCPEchoServer.py**, **TCPEchoClient.py**, **UDPEchoServer.py** and **UDPEchoClient.py**.

   a) Suppose you run **TCPEchoClient** before you run **TCPEchoServer**. What happens? Why?

   When creating a local socket, client attempts to make a TCP connection to a non-existent server process. Exception will be thrown.

   b) Suppose you run **UDPEchoClient** before you run **UDPEchoServer**. What happens? Why?

   UDP client doesn't establish connection to server when creating local socket. Thus it works fine if you start client program first and then server program (but data sent to server are all lost).

To tutors:

Please show the demo by running the programs on the instructor's PC.

4. **[KR, Chapter 3, R7]** Suppose a process in Host C has a UDP socket with port number 6,789. Suppose both Host A and Host B each sends a UDP segment to Host C with destination port number 6,789. 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?

   Yes, both segments will be directed to the same socket. Sock APIs are available for us to learn the IP address of the origin of a packet (e.g. in Java that is `getAddress()` method).

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

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

   Sum: 11000001, checksum: 00111110

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

   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).

6. **[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 Q5 as an example to explain.

   If sender transmits the following two bytes: `01011100` and `01100101`, and the two bits highlighted in red flip, then checksum remains unchanged and receiver will fail to detect this error.

7. **[KR, Chapter 3, R9]** In our `rdt` protocols, why did we need to introduce sequence numbers?

   Sequence numbers are required for a receiver to find out whether an arriving packet contains new data or is a retransmission.