NANYANG TECHNOLOGICAL UNIVERSITY

SEMESTER 1 EXAMINATION 2013-2014

CZ3006/CSC302 – NET-CENTRIC COMPUTING

Nov/Dec 2013 Time Allowed: 2 hours

INSTRUCTIONS

1. This paper contains 6 questions and comprises 7 pages.

2. Answer **ALL** questions.

3. This is a closed-book examination.

4. Questions **DO NOT** carry equal marks.

---

1. (a) List two ways in which the OSI reference model and the TCP/IP model are the same, and two ways in which they differ.

(4 marks)

(b) Give four example networks that exhibit (1) high bandwidth and high latency, (2) high bandwidth and low latency, (3) low bandwidth and high latency, and (4) low bandwidth and low latency, respectively. Explain your answers.

(8 marks)

(c) Consider the effect of using slow start on a line with a 100-msec round-trip time and no congestion. The receive window is 30 KB and the maximum segment size is 2KB. How long does it take before the first full window can be sent?

(8 marks)

2. Suppose machine A is communicating with machine B using the sliding window protocol in Assignment 1 (attached in the **Appendix** on pages 6 and 7). Answer the following questions:

    (a) When will a retransmission timer be started, stopped, and expire, respectively?

    (2 marks)

    (b) When will the acknowledgement timer be started, stopped, and expire, respectively?

    (2 marks)

    (c) Under what conditions will an arrived frame be rejected, saved in the buffer, and passed to the network layer, respectively?

    (2 marks)

    (d) Suppose B has received frames 0, 2, and 3 from A. Which frame(s) can B deliver to the network layer? Which frame(s) can B save in the buffer? Which sequence number can B (positively) acknowledge A? What sequence numbers are within the receiving window of B?

    (4 marks)

3. Frames of 2500 bytes are sent over a 1Gbps communication channel that has a 50-msec one-way delay. Suppose that acknowledgements are always piggybacked onto data frames, and the headers are very short (so the overhead can be ignored). If 10-bit sequence numbers are used in a sliding-window protocol using the Selective-repeat strategy, is it possible to achieve 100% channel utilization? Explain your answer.

    (10 marks)

4. Consider a TCP machine sending windows of 65,535 bytes over a 1Gbps channel that has a 200-msec two-way delay. Answer the questions and explain your answers: What is the maximum achievable channel utilization for this line? How long does it take to send 64MB over this line?

    (10 marks)

5. (a) Explain the terms: CSMA, CSMA/CD and CSMA/CA. Which one is used in 802.11 wireless multiple access? Why?

    (8 marks)

Note: Question No. 5 continues on Page 3

(b) If a laptop has a wireless connection to the Internet, does that laptop have to be mobile? Explain. Suppose that you are using your laptop which always accesses the Internet through the same AP. Are you mobile from a network standpoint? Explain. What if you walk around NTU and the laptop connects to different APs from time to time? Explain.

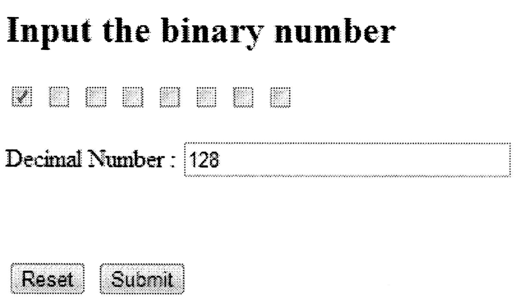
(6 marks)

6. Figure Q6a depicts the browser display of an HTML document, which collects an 8-bit input (by checkboxes) from the client. The input is treated as an **8-bit binary number** where each bit is interpreted as a '1' if the corresponding checkbox is ticked or a '0' if not. The leftmost checkbox represents the most significant digit of the binary number. Two functionalities are to be implemented: First, whenever the checkboxes are changed (ticked or unticked) by the client, the new binary number input is **translated into a decimal number** and displayed in the textbox "Decimal Number". The textbox is not intended for user input, so it will be **blurred** when it gets focus. Second, when the "Submit" button is clicked by the client, it will be examined whether the input number **can be divided by 6**. If yes, the values of the checkboxes will be sent to a remote PHP server for further processing. Otherwise the client will be alerted to re-input.

On the server side, a "record.txt" file is maintained to record all client input numbers **in hexadecimal form**, as depicted in Figure Q6b. On receiving the form data from one client, the PHP program "NumProcessing.php" translates the input number into a hexadecimal representation in string and compares with the stored numbers in "record.txt". If the input number does not exist in the file, the string is appended to the end of the file as a new line. Otherwise, the input number is ignored.

Complete the source HTML document in Figure Q6c and the source PHP program in Figure Q6d by filling in the boxes **A** to **L**, each with a single statement or expression to implement the above functionalities (note: the same expression is to be used for all **F** boxes).

(36 marks)

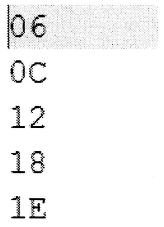


Input the binary number

☑ □ □ □ □ □ □ □

Decimal Number : 128

Figure Q6a



```
06
0C
12
18
1E
```

Figure Q6b

Note: Question No. 6 continues on Page 4

```
<html xmlns = "http://www.w3.org/1999/xhtml"><head>
<title> binaryNumCollection </title>
<script type="text/javascript">
<!--     function bin_to_deci(){
                 var deciNum=0;
                 var binaryForm = document.getElementById("binaryForm");
                 for (index =0; [   A   ] index++)
                         if (binaryForm.elements["binNum[]"][index].checked)
                                 deciNum+=Math.pow(2,(7-index));
                 return deciNum;
          }
         function convert_display(){
                 var binaryForm = document.getElementById("binaryForm");
                  [   B   ]
         }
         function chk_division(){
                 deciNum=bin_to_deci();
                 if ((deciNum%6)== 0){
                          [   C   ]
                 }
                 else {
                         alert("The input number cannot be divided by 6");
                          [   D   ]
                 }
         }
//--></script>
</head><body>
     <form [   E   ]>
           <h2> Input the binary number </h2>
           <p>  <input type = "checkbox" name = "binNum[]" value = "8" [   F   ]/>
                <input type = "checkbox" name = "binNum[]" value = "7" [   F   ]/>
                <input type = "checkbox" name = "binNum[]" value = "6" [   F   ]/>
                <input type = "checkbox" name = "binNum[]" value = "5" [   F   ]/>
                <input type = "checkbox" name = "binNum[]" value = "4" [   F   ]/>
                <input type = "checkbox" name = "binNum[]" value = "3" [   F   ]/>
                <input type = "checkbox" name = "binNum[]" value = "2" [   F   ]/>
                <input type = "checkbox" name = "binNum[]" value = "1" [   F   ]/></p>
           <p> Decimal Number :
           <input type = "text" name = "deciNum" size = "30" value="0" [   G   ]/>
           <br/><br/><input type = "reset" /><input type = "submit" /> </p>
     </form>
<script type = "text/javascript">
<!--
         [   H   ]
// --></script>
</body></html>
```

Figure Q6c

Note: Question No. 6 continues on Page 5

```
<html xmlns="http://www.w3.org/1999/xhtml"><head>
<title>NumProcessing</title>
</head><body><h2> NumPorcessing: </h2>
  <?php
                function deci_to_hexa($deciNum){
                        switch ($deciNum){
                                case 10:
                                        return "A";
                                case 11:
                                        return "B";
                                case 12:
                                        return "C";
                                case 13:
                                        return "D";
                                case 14:
                                        return "E";
                                case 15:
                                        return "F";
                                default:
                                        return       I
                        }
                }
                $binNum = $_POST["binNum"];  //to accept checkbox values as an array.
                $deciNumL = 0;
                $deciNumM = 0;
                if (!empty($binNum)){
                        foreach($binNum as $bNum){
                                if (intval($bNum) <= 4){
                                        $deciNumL += pow(2,(intval($bNum)-1));
                                }
                                else{
                                        $deciNumM += pow(2,       J       );
                                }
                        }
                }
                $hexaNum=deci_to_hexa($deciNumM).deci_to_hexa($deciNumL)."\n";
                $fileLines=file("record.txt");
                if (!empty($fileLines)){
                        foreach($fileLines as $line){
                                $strcmpResult=strcmp($hexaNum,$line);
                                if($strcmpResult==0){
                                        return;
                                }
                        }
                }
                $file = fopen("record.txt","a")
                        or exit("Unable to open file!");
                      K
                      L
        ?>
 </body></html>
```

Figure Q6d

Appendix: A Sliding-Window Protocol (for Question No. 2)

```
#define MAX_SEQ 7
#define NR_BUFS ((MAX_SEQ + 1)/2)
typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready, ack_timeout}
event_type;
#include "protocol.h"
boolean no_nak = true;
seq_nr oldest_frame = MAX_SEQ + 1;

static boolean between(seq_nr a, seq_nr b, seq_nr c)
{   return ((a <= b) && (b < c)) || ((c < a) && (a <= b)) ||  ((b < c) && (c < a)); }

static void send_frame(frame_kind fk, seq_nr frame_nr, seq_nr frame_expected, packet
buffer[])
{
        frame s;  s.kind = fk;
        if (fk == data) s.info = buffer[frame_nr % NR_BUFS];
        s.seq = frame_nr;
        s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1);
        if (fk == nak) no_nak = false;
        to_physical_layer(&s);
        if (fk == data) start_timer(frame_nr % NR_BUFS);
        stop_ack_timer();
}

void Sliding-Window-Protocol(void)
{
        seq_nr ack_expected, next_frame_to_send, frame_expected, too_far;
        int i;
        frame r;
        packet out_buf[NR_BUFS];
        packet in_buf[NR_BUFS];
        boolean arrived[NR_BUFS];
        event_type event;

        enable_network_layer(?);
        ack_expected = 0;
        next_frame_to_send = 0;
        frame_expected = 0;
        too_far = NR_BUFS;

        for (i = 0; i < NR_BUFS; i++) arrived[i] = false;
```

Note: Appendix continues on Page 7

```
while (true) {
  wait_for_event(&event);
  switch(event) {
    case network_layer_ready:
        from_network_layer(&out_buf[next_frame_to_send % NR_BUFS]);
        send_frame(data, next_frame_to_send, frame_expected, out_buf);
        inc(next_frame_to_send);
        break;
     case frame_arrival:
       from_physical_layer(&r);
       if (r.kind == data) {
          if ((r.seq != frame_expected) && no_nak)
             send_frame(nak, 0, frame_expected, out_buf); else
        start_ack_timer();
          if (between(frame_expected, r.seq, too_far) &&
                      (arrived[r.seq % NR_BUFS]==false)) {
              arrived[r.seq % NR_BUFS] = true;
              in_buf[r.seq % NR_BUFS] = r.info;
              while (arrived[frame_expected % NR_BUFS]) {
                 to_network_layer(&in_buf[frame_expected % NR_BUFS]);
                 no_nak = true;
                 arrived[frame_expected % NR_BUFS] = false;
                 inc(frame_expected);
                 inc(too_far);
                 start_ack_timer();
              } } }
         if((r.kind==nak) && between(ack_expected, (r.ack+1)%(MAX_SEQ+1),
                                       next_frame_to_send))
            send_frame(data, (r.ack+1) % (MAX_SEQ+1), frame_expected,
     out_buf);
          while (between(ack_expected, r.ack, next_frame_to_send)) {
              enable_network_layer(?);
              stop_timer(ack_expected % NR_BUFS);
              inc(ack_expected);
          }
          break;
    case cksum_err:
         if (no_nak) send_frame(nak, 0, frame_expected, out_buf);
         break;
    case timeout:
         send_frame(data, oldest_frame, frame_expected, out_buf);
         break;
    case ack_timeout:
        send_frame(ack, 0, frame_expected, out_buf);
    } } }
```

END OF PAPER

CZ3006 NET CENTRIC COMPUTING
CSC302 NET CENTRIC COMPUTING

Please read the following instructions carefully:

1. **Please do not turn over the question paper until you are told to do so. Disciplinary action may be taken against you if you do so.**

2. You are not allowed to leave the examination hall unless accompanied by an invigilator. You may raise your hand if you need to communicate with the invigilator.

3. Please write your Matriculation Number on the front of the answer book.

4. Please indicate clearly in the answer book (at the appropriate place) if you are continuing the answer to a question elsewhere in the book.