Računarske mreže i mrežne tehnologije
I kolokvijum, školska 2013/14.

Prirodno-matematički fakultet Kragujevac
Institut za matematiku i informatiku
16. April 2014. god.

1. Objasniti koncept slojeva u mrežnom softveru.
2. Objasniti razliku između *switch*-a i *hub*-a.
3. Jedinstven kanal podeljen je na 5 nezavisnih FDM potkanala. Koliko je prosečno kašnjenje ako je brzina kanala 200 Mb/s, srednja dužina okvira 50000 bitova i brzina pristizanja 1000 okvira/s ?
4. Dva računara A i B koriste Simplex protokol za slanje podataka bučnim kanalom. Prepraviti dati programski kôd tako da pošaljilac uvek šalje po tri sukcesivna frejma i blokira se dok ne dobije potvrdu da su sva tri primljena. Računari A i B prate redne brojeve frejmova i ne ponavljaju ih prilikom slanja novih.
5. Koliko računara je moguće spojiti sa ethernet *crossover* kablom ? Detaljno objasniti razliku u odnosu na standardni ethernet kabl koji vezuje računar i switch.
6. Za niz bitova 0xAA odrediti koeficijente Furijeove funkcije a_n, b_n i c_n.
7. Računar A šalje okvire računaru B i radi uokviravanje sa indikatorskim bajtom uz umetanje znakova. Dva paketa na računaru A su pre slanja predstavljeni sledećim nizom bajtova FFVEERTC i PHYOFEQW. Kao Flag bajt se koristi karakter F, a kao kontrolni karakter (ESC) se koristi E. Napisati kako izgleda pristgli niz bajtova na računaru B pre obrade.
8. Tok bitova 1111000011110000 prenosi se standardnom CRC metodom sa generatorskim polinomom $G(x) = x^6 + x$.
    a. Napisati tok bitova koji se stvarno šalje.
    b. Ako je 8. bit sa desne strane greškom invertovan dokazati da primalac detektuje ovu grešku.
9. Pošaljilac treba da pošalje niz bitova vrednosti 0xE49C koristeći Hamingov kod.
    a. Napisati tok bitova koji se stvarno šalje.
    b. Ako je 10. bit sa desne strane greškom invertovan dokazati da primalac detektuje ovu grešku.

Kôd uz zadatak 4.

```
/* Protocol 3 (par) allows unidirectional data flow over an unreliable channel. */

#define MAX_SEQ 1                                   /* must be 1 for protocol 3 */
typedef enum  {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"

void sender3(void)
{
 seq_nr next_frame_to_send;                         /* seq number of next outgoing frame */
 frame s;                                           /* scratch variable */
 packet buffer;                                     /* buffer for an outbound packet */
 event_type event;

 next_frame_to_send = 0;                            /* initialize outbound sequence numbers */
 from_network_layer(&buffer);                       /* fetch first packet */
 while (true) {
     s.info = buffer;                               /* construct a frame for transmission */
     s.seq = next_frame_to_send;                    /* insert sequence number in frame */
     to_physical_layer(&s);                         /* send it on its way */
     start_timer(s.seq);                            /* if answer takes too long, time out */
     wait_for_event(&event);                        /* frame_arrival, cksum_err, timeout */
     if (event == frame_arrival) {
          from_physical_layer(&s);                  /* get the acknowledgement */
          if (s.ack == next_frame_to_send) {
               stop_timer(s.ack);                   /* turn the timer off */
               from_network_layer(&buffer);         /* get the next one to send */
               inc(next_frame_to_send);             /* invert next_frame_to_send */
          }
     }
 }
}

void receiver3(void)
{
 seq_nr frame_expected;
 frame r, s;
 event_type event;

 frame_expected = 0;
 while (true) {
     wait_for_event(&event);                        /* possibilities: frame_arrival, cksum_err */
     if (event == frame_arrival) {                  /* a valid frame has arrived. */
          from_physical_layer(&r);                  /* go get the newly arrived frame */
          if (r.seq == frame_expected) {            /* this is what we have been waiting for. */
               to_network_layer(&r.info);           /* pass the data to the network layer */
               inc(frame_expected);                 /* next time expect the other sequence nr */
          }
          s.ack = 1 – frame_expected;               /* tell which frame is being acked */
          to_physical_layer(&s);                    /* send acknowledgement */
     }
 }
}
```