

Računarske mreže i mrežne tehnologije

I kolokvijum, školska 2014/15.

Prirodno-matematički fakultet Kragujevac
Institut za matematiku i informatiku
29. April 2015. god.

1. Koja je osnovna uloga sloja veze podataka (data link layer).
2. Po čemu se povremeni CSMA (non-persistent CSMA) protokol razlikuje od 1-trajnog CSMA protokola.
3. Čemu služi preambula u DIX okviru, koliko tipova postoji i u čemu je razlika.
4. Niz bitova 011100100011100101 grafički predstaviti diferencijalnim Mančester kodiranjem.
5. Računar A šalje okvire računaru B i radi uokviravanje sa indikatorskim bajtom Y. Kontrolni karakter je predstavljen simbolom H. Niz bajtova koji dobija računar B je

YGOHYEIDYJDKKEDYJLLHHEJMYDOHWPEYKFJHDKEDY

Napisati kako izgledaju frejmovi posle obrade na računaru B.

6. Modifikovati kod simplex protokola za slanje podataka bučnim kanalom tako da *sender* prihvaćeni paket od mrežnog sloja deli na dva paketa funkcijom *split* i tako ih šalje *receiveru*.

```
void split(packet * from_network_layer, packet * first, packet * second);
```

7. Za niz bitova 0x7EF odrediti koeficijente Furijeove funkcije a_n , b_n i c .
8. Pošiljalac treba da pošalje niz bitova vrednosti 100101011000001010 koristeći Hamingov kod.
 - a) Napisati tok bitova koji se stvarno šalje.
 - b) Ako je 7. bit sa desne strane greškom invertovan dokazati da primalac detektuje ovu grešku.

```

/* 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 */
        }
    }
}

```