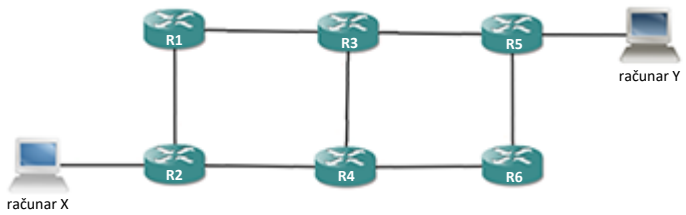
Računarske mreže i mrežne tehnologije
I kolokvijum, školska 2015/16.

Prirodno-matematički fakultet Kragujevac
Institut za matematiku i informatiku
15. April 2016. god.

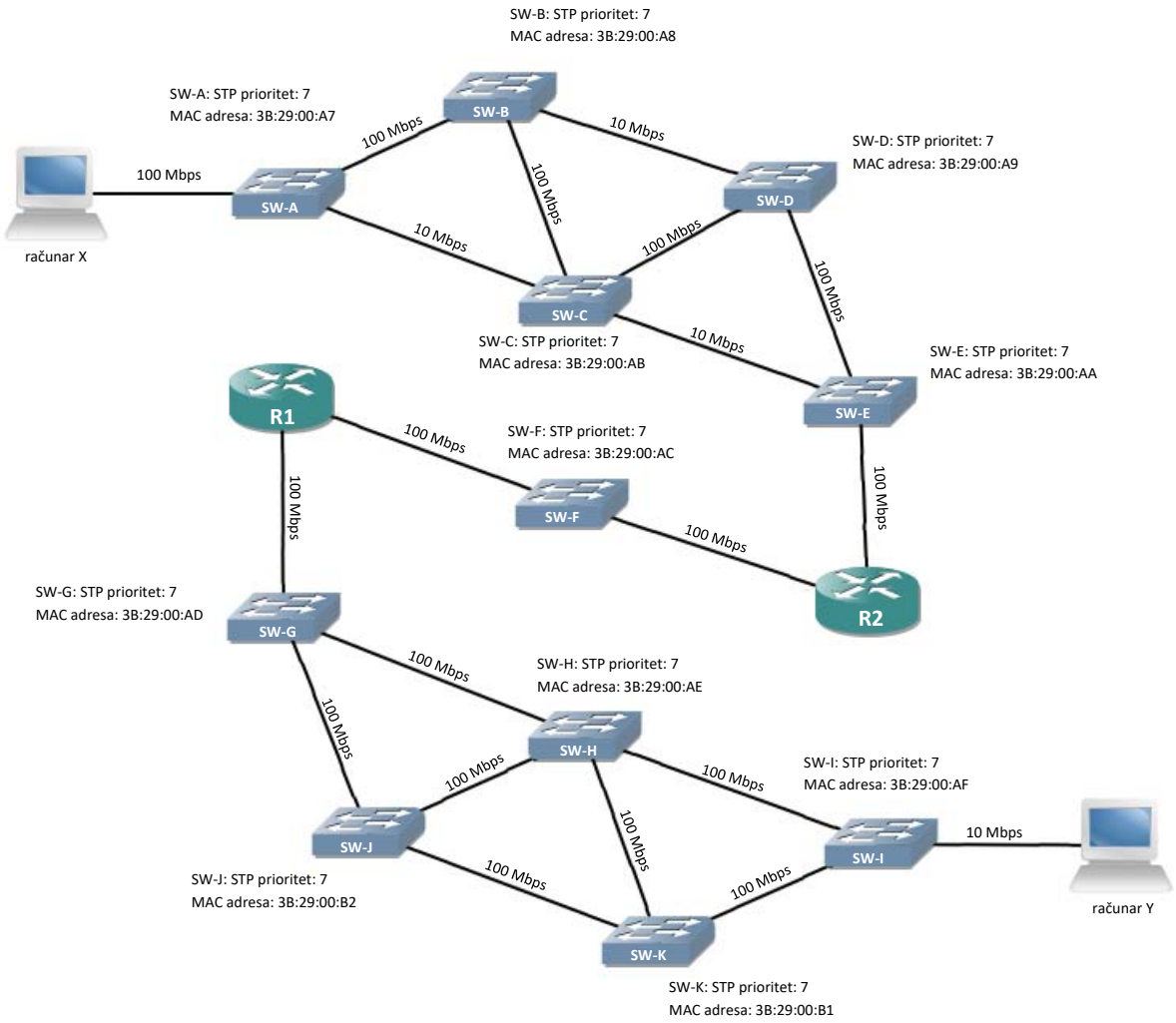
1. Koliko mora biti Hamingovo rastojanje koda da bi se ispravilo *d* grešaka?
2. Koje vrednosti polja *Protocol* se mogu pojaviti kod *IP datagrama*?
3. Na **Slici 1**, računar *X* šalje paket računaru *Y*. Računar *X* pripada mreži 123.0.0.0 dok računar *Y* pripada mreži 190.10.0.0. Paket putuje preko rutera R2 -> R4 -> R3 -> R5. Dodeliti proizvoljne IP adrese ruterima tako da svaki ruter pripada zasebnoj mreži i u zavisnosti od dodeljenih IP adresa formirati *Next Hop* tabele za rutere R2, R4, R3 i R5.



Slika 1

4. Pošiljalac treba da pošalje niz bitova vrednosti 0x0F0F.
    a) Za dati niz bitova odrediti koeficijente Furijeove funkcije a_n, b_n i c.
    b) Koristeći Hamingov kod, napisati tok bitova koji se stvarno šalje. Ako je 4 bit sa desne stane invertovan, dokazati da primalac detektuje grešku.
    c) Napisati tok bitova koji se stvarno šalje, ako se za prenos koristi standardna CRC metoda sa generatorski polinomom $G(x) = x^5 + x^2 + x^1$.
5. Modifikovati kod *simplex* protokola za slanje podataka bučnim kanalom tako da *sender* prihvata pet uzastopnih paketa od mrežnog sloja, pa tek onda šalje svih pet od jednom. *Sender* uzima novih pet paketa od mrežnog sloja tek onda kada je stigla potvrda za svih pet prethodno poslatih paketa. Paketi koji ne stignu na odredište ili im se desi greška prilikom prenosa, se šalju ponovo.
6. Na **Slici 2** je data mreža, koja se sastoji od 11 svičeva i 2 rutera. Za svaki svič je dat simbolički naziv, prioritet za *spanning-tree* protokol i MAC adresa sviča. Odrediti:

    a. Putanju kojom idu okviri između Računara X i Računara Y.
    b. Maksimalni kapacitet prenosa po ovoj putanji.
    c. Na svim vezama označiti statuse svih portova (statuse označiti sa RP, DP, BP), i označiti uspostavljeno stablo prenosa okvira.


SW-B: STP prioritet: 7
MAC adresa: 3B:29:00:A8
SW-A: STP prioritet: 7
MAC adresa: 3B:29:00:A7
SW-B
SW-A
SW-D
SW-C
SW-E
SW-F
SW-G
SW-H
SW-I
SW-J
SW-K
R1
R2
100 Mbps
10 Mbps
računar X
računar Y
SW-D: STP prioritet: 7
MAC adresa: 3B:29:00:A9
SW-C: STP prioritet: 7
MAC adresa: 3B:29:00:AB
SW-E: STP prioritet: 7
MAC adresa: 3B:29:00:AA
SW-F: STP prioritet: 7
MAC adresa: 3B:29:00:AC
SW-G: STP prioritet: 7
MAC adresa: 3B:29:00:AD
SW-H: STP prioritet: 7
MAC adresa: 3B:29:00:AE
SW-I: STP prioritet: 7
MAC adresa: 3B:29:00:AF
SW-J: STP prioritet: 7
MAC adresa: 3B:29:00:B2
SW-K: STP prioritet: 7
MAC adresa: 3B:29:00:B1

Slika 2

```
/* 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 */
      }
  }
}
```