UNIVERSITETI POLITEKNIK I TIRANËS
FAKULTETI I TEKNOLOGJISË SË INFORMACIONIT
Departamenti i Inxhinierisë Informatike

Detyrë Kursi

Lënda: Rrjeta Telematike

Tema: Protokolli 5 dhe 6 në Data Link

Grupi:B

Punoi: Piro Gjikdhima

Pranoi: Prof. Dr. Indrit Enesi

Bachelor “ Inxhinieri Informatike”

Tiranë

Maj 2024

PROTOKOLLI 5

Ky protokoll ne momentin qe marresi merr nje pakete te gabuar, paketave te tjera te dërguar ai ben discard derisa te vije paketa e sakte.

Pra marrresi “refuzon” paketat e tjera deri ne momentin qe merr paketen e demtuar, kjo kur shkalla e gabimit eshte e larte mund te sjell nje ulje te larte te bandwidth.

Deri tani, kemi supozuar se koha e transmetimit që i duhet një frame të arrijë te marrësi plus koha e marrjes së ACK-së është e papërfillshme, gjë që nuk është e vërtetë. Kjo vjen nga fakti se deri tani kemi pritur një ACK përpara se të dërgojmë framen tjetër. Protokolli 5 dhe më tej e rregullon këtë problem duke dërguar më shumë se një frame përpara se të bllokohet (gjatë një kohe të caktuar).

Protokolli Go-Back-N dërgon frame me radhë (0, 1, 2, ...). Kur frame 1 vjen në rregull, dërgohet ACK. Nëse frame 2 dëmtohet, dërguesi vazhdon të dërgojë frame derisa timeri i frame 2 përfundon. Në këtë moment, dërguesi ridergon frame që nga frame 2, pasi frame duhet të vijnë në mënyrë kronologjike.

Protokolli nr. 5 (Go-Back-N) përdor një mekanizëm me dritare rëshqitëse. Transmetuesi mund të transmetojë deri në një numër MAX_SEQ framesh pa pritur për një ACK. Ndryshe nga protokollet e tjera, network layer nuk pret një paketë të re në çdo moment, por vetëm atëherë kur ajo shkakton një "network layer ready" event, pra është gati për të pritur një paketë. Dritarja do të zhvendoset kur ACK-të e frame-ve të mëparshme merren.

KODI:

```
/* Protocol 5 (go back n) lejon te dergohen shume frame nga derguesi. Derguesi mund te
transmetoje deri ne MAX_SEQ frame pa qene nevoje te prese per ack. Ndryshe nga
protokollet e tjera, network layer nuk eshte e thene te marre nje pakete te re ne çdo moment.
Ne te kundert, network layer nxit nje network_layer_ready event kur ka nje pakete per te
derguar. */

#define MAX_SEQ 7 /* duhet te jete 2^n - 1 */
typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready} event_type;

#include "protocol.h"

static boolean between(seq_nr a, seq_nr b, seq_nr c) {
   /* kthen true nese a <= b < c; false ne te kundert */
   return ((a <= b) && (b < c)) || ((c < a) && (a <= b)) || ((b < c) && (c < a));
}

static void send_data(seq_nr frame_nr, seq_nr frame_expected, packet buffer[]) {
   /* nderton dhe dergon nje frame te dhenash */
   frame s; /* deklarimi variables te tipit frame */
   s.info = buffer[frame_nr]; /* inserton paketen ne frame */
   s.seq = frame_nr; /* inserton nr e sekuences ne frame */
   s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1); /* piggyback ack */
   to_physical_layer(&s); /* transmeton framen */
   start_timer(frame_nr); /* fillon numerimi i timerit */
}

void protocol5(void) {
   seq_nr next_frame_to_send; /* MAX_SEQ > 1; perdoret per streamin e derguar */
   seq_nr ack_expected; /* frami me i vjeter i pa konfirmuar akoma */
```

```
seq_nr frame_expected; /* frame tjeter i cili pritet te vije ne stream */
frame r;
packet buffer[MAX_SEQ + 1]; /* bufferat per streamin qe del */
seq_nr nbuffered; /* numri i buferave ne dalje ne perdorim */
seq_nr i; /* perdoret per te indeksuar brenda vektorit te bufferit */
event_type event;

enable_network_layer(); /* lejon eventet network_layer_ready */
ack_expected = 0; /* ack tjeter qe pritet te vije */
next_frame_to_send = 0; /* frame tjeter qe dergohet */
frame_expected = 0; /* numri i frameve qe pritet ne hyrje */
nbuffered = 0; /* ne fillim asnje pakete nuk eshte bufferuar */

while (True) {
   wait_for_event(&event); /* kater mundesi */

   switch(event) {
      case network_layer_ready: /* shtresa network layer ka nje pakete per te derguar */
         /* pranon, ruan, dhe transmeton nje frame te ri */
         from_network_layer(&buffer[next_frame_to_send]); /* ngarkon nje pakete te re */
         nbuffered = nbuffered + 1; /* zgjeron dritaren e derguesit */
         send_data(next_frame_to_send, frame_expected, buffer); /* transmeton framen */
         inc(next_frame_to_send); /* rrit kufirin e siperm te dritares se derguesit */
         break;

      case frame_arrival: /* nje frame te dhenash ose kontrolli ka mberritur */
         from_physical_layer(&r); /* merr framen e ardhur nga shtresa physical layer */
         if (r.seq == frame_expected) {
            /* framet pranohen vetem sipas rradhes */
            to_network_layer(&r.info); /* kalon paketen ne shtresen network */
```

```
            inc(frame_expected); /* rrit kufirin e poshtem te dritares se marresit */
        }

        while (between(ack_expected, r.ack, next_frame_to_send)) {
            /* dorezon piggybacked ack */
            nbuffered = nbuffered - 1; /* nje frame me pak e bufferuar */
            stop_timer(ack_expected); /* frame arrin ne rregull, ndalon timerin */
            inc(ack_expected);
        }
        break;

      case cksum_err:
        break; /* injoron framet e jo te mira */

      case timeout: /* problem; ritransmeto te gjitha framet ne dalje */
        next_frame_to_send = ack_expected; /* fillon ritransmetimi ketu */
        for (i = 1; i <= nbuffered; i++) {
            send_data(next_frame_to_send, frame_expected, buffer); /* ridergon framen */
            inc(next_frame_to_send); /* pergatitet te dergoje tjetren */
        }
        break;
    }

    if (nbuffered < MAX_SEQ) {
      enable_network_layer();
    } else {
      disable_network_layer();
    }
  }
}
```

PROTOKOLLI 6

Ky protokoll eshte nje alternative e mire sepse ne nje kanal transmetimi me nje shkalle gabimi te larte ndryshe nga protokolli 5 I cli pas nje frame te derguar gabim gjithe frame e tjera I bente discard,ne kete rast protokolli 6 I ruan framet pasardhes ne buffer-at e sender-it pa qene nevoja e ridergimit serish.Ne kete protokoll dritarja e derguesit ka nje madhesi nga 0 ne MAX_SEQ, kurse dratarja e marresit eshte nje madhesi e fiksuar ne MAX_SEQ.

KODI:

```
#define MAX_SEQ 7
// Numri maksimal i sekuencave është 7, kështu që do të kalojnë nga 0 deri në 7.
// Definimi i numrit maksimal të sekuencave.

#define NR_BUFS ((MAX_SEQ + 1) / 2)
// Numri i buffer-ave ku do të ruhen framat e dërguara deri në momentin që do të vijë konfirmimi.
// Përcakton numrin e buffer-ave.

typedef enum {
    frame_arrival,    // Ngjarja kur vjen një frama.
    cksum_err,        // Ngjarja kur ka një gabim në checksum.
    timeout,          // Ngjarja e timeout-it.
    network_layer_ready,  // Ngjarja kur shtresa e rrjetit është gati.
    ack_timeout       // Ngjarja e timeout-it të konfirmimit.
} event_type;
// Enumerimi i llojeve të ndryshme të ngjarjeve që lidhen me ardhjen e framas, gabimin në checksum,
eventet e timeout-it, gatishmërinë e shtresës së rrjetit dhe timeout-in e konfirmimit.
```

```
#include "protocol.h"
boolean_no_nak = true;
// Tregon se nuk kemi dërguar asnjë NAK. S'ka ndodhur humbje frame.
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));
}
// Funksioni between do të na bëjë kontrollin e numrave sekuencial të eventeve të shpjeguara më poshtë.
// Ky funksion do të percaktojë dritaren e dhënësit dhe marrësit.

static void send_frame(frame_kind fk, seq_nr frame_nr, seq_nr frame_expected, packet buffer[]) {
   // Ky funksion merr si prototip llojin e framave, numrin sekuencial të framës së dërguar,
   // numrin sekuencial të framës që pritet dhe buferin e inicializojme si vector.

   frame s; // s është frama që do të dërgohet
   s.kind = fk; // kind është e tipit data, ack ose nak

   if (fk == data) {
      // Për sa kohë që frama është e tipit data, informacioni do të vendoset në buffer ndërkohë që
      // bufferi pret akoma informacion.
      s.info = buffer[frame_nr % NR_BUFS];
   }

   s.seq = frame_nr; // numri sekuencial i framës që dërgohet
   s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1); // PIGGY BACK

   if (fk == nak) {
      no_nak = false;
      // Nëse ka ardhur një NAK, nuk pranojmë më një tjetër.
      // Frama dërgohet në pjesën e shtresës fizike, dhe menjëherë ndezim timer-in nëse frama
```

```
        // e dërguar është e tipit data dhe në buffer ka akoma vend të lirë.
        // Gjithashtu, timer-i i PIGGY BACK-ut ndalon pritjen dhe niset me framen e re data, ose
        // vetem nëse kemi timeout te ack.
    }
    to_physical_layer(&s);
    if (fk==data)
            start_timer(frame_nr % NR_BUFS);
   stop_ack_timer();
}
void protocol_6(void) {
    seq_nr ack_expected; // Shtresa më e ulët e dritares të dërguesit
    seq_nr next_frame_to_send; // Shtresa më e lartë e dritares të dërguesit
    seq_nr frame_expected; // Shtresa më e ulët e dritares të marrësit
    seq_nr too_far; // Shtresa më e lartë e dritares të marrësit
    int i; // Indeksi i bufferit
    frame r; // Variabël e framas që do të dërgohet
    packet_out_buf[NR_BUFS]; // Bufferat e jashtëm
    packet_in_buf[NR_BUFS]; // Bufferat e brendshëm
    boolean arrived[NR_BUFS];
    seq_nr nbuffered; // Sa bufera të jashtëm janë përdorur
    event_type event;

    enable_network_layer(); // Shtresa e rrjetit vendoset në gjendje gati

    ack_expected = 0; // Gjendja e inicializimit për çdo gjendje
    next_frame_to_send = 0;
    frame_expected = 0;
    too_far = NR_BUFS;
    nbuffered = 0; // Fillimisht s'ka bufera të zënë
```

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

while (true) {
   wait_for_event(&event); // Sistemi pret një ngjarje

   switch (event) {
      case network_layer_ready:
         nbuffered++;
         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);
         // Në rastin kur shtresa e rrjetit është gati për të dhënë informacione,
         // vihet në punë dritarja dhe fillon të numërojmë buferat.
         // Kur informacioni vjen nga shtresa e rrjetit dhe buferat për jashtë
         // nuk janë mbushur akoma, atëherë mund të dërgohet frama data pa gabime.
         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)) {
```

/Ndërkohë që frama e ardhur është një NAK, domethënë na ka ndodhur një gabim dhe jemi brenda dritares së vendosur në dispozicion, atëherë buffer-i do të lirohet në mënyrë që të presi framën tjetër, do ndalohet timeri qe pret konfirmimin dhe do të kalohet në pritje të framës pasardhëse/

```
            send_frame(data, (r.ack + 1) % (MAX_SEQ + 1), frame_expected, out_buf);
            while (between(ack_expected, r.ack, next_frame_to_send)) {
                nbuffered--;
                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;
```

/Në rast se na vjen një framë gabim dhe nuk ka ardhur një NAK, atëherë do të dërgohet NAK-u /

```
    case timeout:
        send_frame(data, oldest_frame, frame_expected, out_buf);
        break;
```

/Në rast se kemi timeout, do të ridërgohet frama e mëparshme/

```
        case ack_timeout:
            send_frame(ack, 0, frame_expected, out_buf);
```

/Nëse PIGGY BACK-u ka kaluar në timeout(gjendje e cila ndodh kur marrësi nuk ka frame për të dërguar) atëherë frama e konfirmimit dërgohet menjëherë pa shkuar akoma në timeout të timerit të frames/

```
    }
        if (nbuffered < NR_BUFS)
            enable_network_layer();
        else
            disable_network_layer();
        break;
    }
}
```

/Gjithcka ndodh për sa kohë që numri i buferave që numërohen vazhdimit është më I vogël se sa numri i bufferave të diposnueshëm dhe kjo bën që shtresa e networkut të jetë në gjendje enable, në rast të kundërt shtresa e networkut nuk do të jetë në gjendje për të dërguar apo për të marrë informacion nga data linku/

Krahasimi midis 2 protokolleve, e permbledhur ne nje tabele:

Karakteristikat	Protokolli 5	Protokolli 6
Përkufizimi	Në Go-Back-N nëse një Frame e dergruar gjendet e ndryshuar ose e demtuar te gjitha frame te tjera ritransmetohet deri sa te korrigjohet frama me problem	Ne Selective Repeat vetem framat qe jane te demtuara ose te humbura ritransmetohet
Permasat e dritares	Permasat e dritares se derguesit jane N Permasat e dritares se marresit jane 1	Permasat e dritares se derguesit jane N Permasat e dritares se marresit inicializohet me N
	Eshte me pak kompleks	Ka nje numer me te madh hapash per tu kryer
Out of Order Packet	Out of order packets nuk pranohen dhe ritransmetohet e gjithe dritarja	Out of order packets pranohen
Eficenca	Eficienca= N/(1+2a)	Eficienca= N/(1+2a)

1. Ne protokollin G-Back-N, parametrat e komunikimit jane:

Bandwidth: 1 Mbps

Round-trip Time (RTT): 50 ms

Packet Loss Rate: 10%

Packet size: 1000 bit

Acknowledgement size: 40 bit

Timeout: 100 ms (for simplicity, a fixed timeout for all packets)

Përcaktoni throughput-in e kanalit.

Fillimisht gjejmë t_p dhe t_{totale}

$$t_p = \frac{\text{Packet Size}}{\text{Bandwidth}} = \frac{1000 \text{ bit}}{1\,000\,000\ bit/sec} = 0.001\ sec$$

$$t_{p_{Ack}} = \frac{\text{Acknowledgement size}}{\text{Bandwidth}} = \frac{40 \text{ bit}}{1\,000\,000\ bit/sec} = 0.00004\ sec$$

$$W = \frac{Bandwidth \times RTT}{Packet\ size} = \frac{1\,000\,000\ bit/sec \times 0.05\ sec}{1000\ bit} = 50\ paketa$$

$$Utilization = \frac{W \times t_p}{t_p + RTT + t_{p_{Ack}}}$$

$$Utilization = \frac{50 \times 0.001\ sec}{0.001\ sec + 0.00004\ \text{sec} + 0.05\ \text{sec}} = 0.979624 = 97.9624\ \%$$

Kemi një Shfrytëzim të lartë meqë po përdorim Protokollin 5

Gjejmë Throughput-in

$$Throughput = Utilization \times Bandwidth$$
$$Throughput = 0.979624 \times 1\text{Mbps} = 0.979624\ \text{Mbps}$$

$$Throughput_{effective} = Throughput \times (1 - Packet\ Loss\ Rate) = 0.979624\ Mbps \times (1 - 0.1)$$

$$= 0.8816616\ Mbps = 881.6616\ \text{kbps}$$

2. Ne protokollin Selective Repeat, parametrat e komunikimit jane si ne vijim:

Bandwidth: 2 Mbps

Round-trip Time (RTT): 80 ms

Packet Loss Rate: 5%

Packet size: 1000 bit

Acknowledgement size: 30 bit

Timeout: 200 ms (for simplicity, a fixed timeout for all packets)

Percaktoni permasen e dritares (Window Size), madhesine e memories (Buffer size) si dhe Throughput-in e kanalit.

Përcaktoni throughput-in e kanalit.

Fillimisht gjejmë t_p ,t_{totale} dhe Window Size

$$W = \frac{Bandwidth \times RTT}{Packet\ size} = \frac{2\,000\,000\ bit/sec \times 0.08\ sec}{1000\ bit} = 160\ paketa$$

Window Size = Buffer Size = 160 paketa

$$t_p = \frac{\text{Packet Size}}{\text{Bandwidth}} = \frac{1000\ \text{bit}}{2\,000\,000\ bit/sec} = 0.0005\ sec$$

$$t_{p_{Ack}} = \frac{\text{Acknowledgement size}}{\text{Bandwidth}} = \frac{30\ \text{bit}}{2\,000\,000\ bit/sec} = 0.000015\ sec$$

$$Utilization = \frac{W \times t_p}{t_p + RTT + t_{p_{Ack}}}$$

$$Utilization = \frac{160 \times 0.0005\ sec}{0.0005\ sec + 0.000015\ \text{sec} + 0.08\ \text{sec}} = 0.993604 = 99.3604\ \%$$

Kemi një Shfrytëzim të lartë meqë po përdorim Protokollin 6

Gjejmë Throughput-in

$$Throughput = Utilization \times Bandwidth$$

$$Throughput = 0.993604 \times 2\text{Mbps} = 1.987208\ \text{Mbps}$$

$$Throughput_{effective} = Throughput \times (1 - Packet\ Loss\ Rate) = 1.987208\ Mbps \times (1 - 0.05)$$

$$= 1.8878476\ Mbps = 1887.8476\ \text{kbps}$$