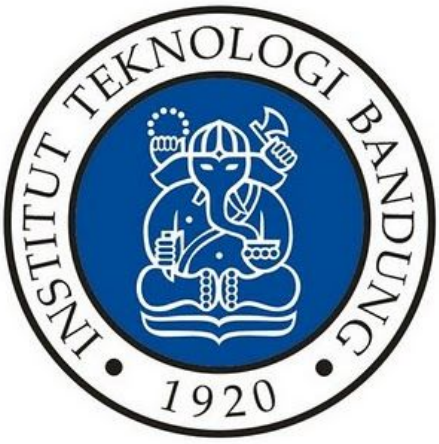
Tugas Kecil 1 - IF3130 Jaringan Komputer

Petunjuk Pengerjaan Tugas

Flow Control

Dipersiapkan oleh:

Asisten Lab Sistem Terdistribusi



START: Rabu, 2 November 2016

END: Kamis, 10 November 2016

A. Latar Belakang

Ditunda untuk tugas selanjutnya.

B. Dasar Teori

Mekanisme flow control atau speed matching diperlukan oleh alat-alat yang memiliki kecepatan transfer data yang lebih tinggi daripada kecepatan alat tersebut memroses data. Sebagai contoh, printer pada saat ini umumnya memiliki kecepatan transfer data yang jauh lebih cepat dibandingkan dengan kecepatan printer mencetak (biasa diukur dalam pages per minute, PPM). Data harus ditampung dalam buffer dalam printer agar data dapat dicetak sesuai dengan kecepatan printer. Namun, membuat buffer saja tidak dapat mengatasi masalah. Buffer hanya akan menunda terjadinya data overflow atau over run.

Sebuah mekanisme flow control diperlukan untuk membuka dan menutup jalannya input data. Metoda XON/XOFF adalah metoda yang paling umum digunakan untuk mengendalikan aliran data. Karakter ASCII yang biasa digunakan sebagai XON adalah DC1 dan XOFF adalah DC3. Program transmitter harus mengenali dua control character ini dan melakukan aksi ketika menerima salah satu karakter tersebut. Program receiver harus memiliki kecerdasan untuk mengenali kondisi buffer untuk mengirim XON/XOFF kepada transmitter.

C. Pengerjaan

Tugas ini melibatkan pembuatan dua buah program, yaitu transmitter dan receiver. Dalam tugas ini digunakan UDP. Bahasa yang dibolehkan dibatasi yaitu C dan C++.

Transmitter

1. Program transmitter menerima masukan melalui parameter program berupa alamat komputer tujuan (alamat IP atau *hostname* dan *port number*) dan nama file teks yang berisi data yang akan dikirim.

2. Pada saat pertama kali dijalankan, transmitter akan membuat *socket* untuk mengirimkan dan menerima data dari alamat receiver.

3. Selanjutnya, transmitter membuat sebuah “proses” anak (boleh menggunakan *fork* atau membuat thread baru) untuk menerima sinyal XON/XOFF dari receiver. Proses utama akan bekerja mengirimkan data.

4. Pada proses anak, transmitter melakukan looping untuk membaca data dari receiver dengan perintah `recvfrom()`. Data yang diterima disimpan di variabel global agar bisa diakses oleh proses utama. Logika prosedur ini adalah sebagai berikut:

```
repeat
      ch <- read a character from socket
      last received char <- ch
```

```
until connection is terminated
```

5. Pada proses utama, transmitter membuka file teks, melakukan *looping* hingga akhir file, sambil mengirimkan karakter demi karakter (1 karakter dalam 1 kali pengiriman) menggunakan perintah `sendto()`. Apabila sinyal terakhir yang diterima dari receiver adalah XOFF, transmitter akan menunggu untuk mengirimkan data hingga sinyal XON diterima.

```
while not end of file do
      if last received char is not XOFF then
            ch <- read a character from file
            send ch through socket
```

6. Proses yang dilakukan oleh transmitter harus tampak dengan jelas pada layar output. Berikut ini contoh keluaran pada layar output:

```
>> transmitter 192.168.0.8 20000 teks.txt
Membuat socket untuk koneksi ke 192.168.0.8:20000 ...
Mengirim byte ke-1: 'H'
Mengirim byte ke-2: 'e'
Mengirim byte ke-3: 'l'
Mengirim byte ke-4: 'l'
Mengirim byte ke-5: 'o'
Mengirim byte ke-6: ' '
Mengirim byte ke-7: 'w'
Mengirim byte ke-8: 'o'
XOFF diterima.
Menunggu XON...
Menunggu XON...
Menunggu XON...
XON diterima.
Mengirim byte ke-9: 'r'
Mengirim byte ke-10: 'l'
Mengirim byte ke-11: 'd'
Mengirim byte ke-12: '.'
Mengirim byte ke-13: '.'
Mengirim byte ke-14: '!'
```

Receiver

1. Program receiver menerima masukan melalui parameter program berupa port tempat receiver akan menerima koneksi.

2. Pertama kali, receiver akan membuat socket dan mem-bind nya ke port tertentu di localhost.

3. Sama seperti transmitter, pada program receiver juga dibuat proses anak yang bekerja mengkonsumsi karakter dari buffer menggunakan prosedur q_get. Sedangkan proses utama adalah melakukan semi-infinite repeat until loop yang memanggil prosedur rcvchar untuk

membaca sebuah karakter dari socket dan menyimpannya ke dalam buffer hingga karakter end-of-file diterima.

4. Karakter yang dibaca dari socket disimpan dalam sebuah buffer. Disarankan untuk membuat buffer ini sebagai circular buffer.

5. Prosedur rcvchar akan membaca karakter dari socket dan menempatkannya di circular buffer. Prosedur ini harus mendeteksi overflow dari buffer. Overflow terjadi ketika jumlah karakter di buffer lebih besar dari minimum upperlimit. Minimum upperlimit harus lebih kecil dari jumlah karakter yang bisa ditampung dalam buffer. Ketika terjadi overflow, karakter XOFF dikirimkan ke transmitter. Logika prosedur ini adalah sebagai berikut:

```
read the character into the buffer and
      increment the buffer pointers properly;
if the number of characters in the
      buffer > minimum upperlimit then
            send XOFF to the transmitter;
```

6. Prosedur q_get mengkonsumsi tepat satu karakter dari buffer. Namun, sebelum mengkonsumsi sebuah karakter, harus dilakukan pengecekan apakah ada karakter pada buffer yang siap dikonsumsi dan apakah karakter tersebut valid. Ketika jumlah karakter yang masih tersisa di buffer lebih kecil dari batas bawah maksimum, maximum lowerlimit, tertentu, karakter XON dikirimkan ke transmitter menandakan bahwa receiver memiliki cukup ruang kosong untuk menerima karakter dari transmitter. Logika prosedur ini adalah sebagai berikut:

```
repeat
      if buffer count > 0 then
            read a character from the buffer;
            decrement the buffer count;
            is it a valid character? (>32, CR, LF, end-of-file)
until valid character;
if buffer count < maximum lowerlimit then
      send XON to the transmitter;
```

7. Timing pada pemanggilan prosedur rcvchar dan q_get sangat krusial. Jika q_get lebih cepat dalam mengkonsumsi karakter dibandingkan rcvchar membacanya dari socket, maka tidak akan pernah terjadi pengiriman sinyal XOFF karena buffer tidak pernah overflow. Cobalah untuk menambahkan delay pada prosedur ini dan mengamati apa yang terjadi.

8. Program receiver harus dijalankan terlebih dahulu sebelum program transmitter.

9. Proses yang dilakukan oleh receiver harus tampak dengan jelas pada layar output. Berikut ini contoh keluaran pada layar output:

```
>> receiver 20000
Binding pada 192.168.0.8:20000 ...
Menerima byte ke-1.
```

```
Menerima byte ke-2.
Mengkonsumsi byte ke-1: 'H'
Menerima byte ke-3.
Menerima byte ke-4.
Mengkonsumsi byte ke-2: 'e'
Menerima byte ke-5.
Menerima byte ke-6.
Mengkonsumsi byte ke-3: 'l'
Menerima byte ke-7.
Buffer > minimum upperlimit.
Mengirim XOFF.
Menerima byte ke-8.
Mengkonsumsi byte ke-4: 'l'
Mengkonsumsi byte ke-5: 'o'
Mengkonsumsi byte ke-6: ' '
Mengkonsumsi byte ke-7: 'w'
Buffer < maximum lowerlimit.
Mengirim XON.
Menerima byte ke-9.
Menerima byte ke-10.
Mengkonsumsi byte ke-8: 'o'
Menerima byte ke-11.
Menerima byte ke-12.
Mengkonsumsi byte ke-9: 'r'
Menerima byte ke-13.
Menerima byte ke-14.
Mengkonsumsi byte ke-10: 'l'
Mengkonsumsi byte ke-11: 'd'
Mengkonsumsi byte ke-12: '.'
Mengkonsumsi byte ke-13: '.'
Mengkonsumsi byte ke-14: '!'
```

Catatan: untuk praktikum ini disediakan contoh header program untuk receiver. Header ini boleh disalin dan digunakan dalam program

D. Sampel Header

```
/*
* File : T1_rx.cpp
*/
#include "dcomm.h"

/* Delay to adjust speed of consuming buffer, in milliseconds */
#define DELAY 500

/* Define receive buffer size */
#define RXQSIZE 8

Byte rxbuf[RXQSIZE];
QTYPE rcvq = { 0, 0, 0, RXQSIZE, rxbuf };
```

```
QTYPE *rxq = &rcvq;
Byte sent_xonxoff = XON;
Boolean send_xon = false, send_xoff = false;

/* Socket */
int sockfd; // listen on sock_fd

/* Functions declaration */
static Byte *rcvchar(int sockfd, QTYPE *queue);
static Byte *q_get(QTYPE *, Byte *);

int main(int argc, char *argv[])
{
        Byte c;

        /*
        Insert code here to bind socket to the port number given in argv[1].
        */

        /* Initialize XON/XOFF flags */

        /* Create child process */

        /*** IF PARENT PROCESS ***/
                while (true) {
                        c = *(rcvchar(sockfd, rxq));

                        /* Quit on end of file */
                        if (c == Endfile) {
                                exit(0);
                        }
                }
        /*** ELSE IF CHILD PROCESS ***/
                while (true) {
                        /* Call q_get */
                        /* Can introduce some delay here. */
                }
}

static Byte *rcvchar(int sockfd, QTYPE *queue)
{
        /*
                Insert code here.
                Read a character from socket and put it to the receive
                buffer.
                If the number of characters in the receive buffer is above
                certain level, then send XOFF and set a flag (why?).
                Return a pointer to the buffer where data is put.
        */
}

/* q_get returns a pointer to the buffer where data is read or NULL if
* buffer is empty.
*/
static Byte *q_get(QTYPE *queue, Byte *data)
{
        Byte *current;
        /* Nothing in the queue */
        if (!queue->count) return (NULL);
```

```
        /*
                Insert code here.
                Retrieve data from buffer, save it to "current" and "data"
                If the number of characters in the receive buffer is below
                certain level, then send XON.
                Increment front index and check for wraparound.
        */
}
```

```
/*
 * File : dcomm.h
 */
#ifndef _DCOMM_H_
#define _DCOMM_H_

/* ASCII Const */
#define SOH       1  /* Start of Header Character */
#define STX       2  /* Start of Text Character */
#define ETX       3  /* End of Text Character */
#define ENQ       5  /* Enquiry Character */
#define ACK       6  /* Acknowledgement */
#define BEL       7  /* Message Error Warning */
#define CR        13 /* Carriage Return */
#define LF        10 /* Line Feed */
#define NAK       21 /* Negative Acknowledgement */
#define Endfile   26 /* End of file character */
#define ESC       27 /* ESC key */

/* XON/XOFF protocol */
#define XON       (0x11)
#define XOFF      (0x13)

/* Const */
#define BYTESIZE   256     /* The maximum value of a byte */
#define MAXLEN     1024    /* Maximum messages length */

typedef enum { false=0, true } Boolean;

typedef unsigned char Byte;

typedef struct QTYPE
{
         unsigned int count;
         unsigned int front;
         unsigned int rear;
         unsigned int maxsize;
         Byte *data;
} QTYPE;

typedef struct MESGB
{
         unsigned int soh;
         unsigned int stx;
         unsigned int etx;
         Byte checksum;
         Byte msgno;
         Byte *data;
} MESGB;
```

```
#endif
```

E. Deliverables

Berkas yang dikumpulkan beserta struktur peletakan adalah sebagai berikut:

- Folder src berisi;
  - *Source code* dengan komentar yang jelas
- Folder bin berisi;
  - *Executable* hasil kompilasi *source code*
- `Makefile`
- Laporan berekstensi .pdf berisi;
  - Pembahasan pertanyaan-pertanyaan berikut
    - Mengapa dalam tugas kecil ini digunakan UDP, bukan TCP?
    - Jelaskan perbedaan TCP dan UDP
    - Mengapa *minimum upperlimit* / batas atas minimum harus lebih kecil dari jumlah karakter yang bisa ditampung dalam *buffer*?
  - Petunjuk kompilasi program
  - Petunjuk penggunaan program
  - Pranala sumber referensi yang digunakan dalam pengerjaan tugas, dan bagian kode yang digunakan dari referensi tersebut apabila ada.
  - Tidak perlu menyertakan soal pada laporan, apalagi melakukan pengkopian langsung dari berkas soal.
  - Pembagian kerja dalam kelompok

F. Teknis Pengumpulan

Tugas ini dikerjakan dengan kelompok **maksimal** 3 orang, **tidak boleh** lintas kelas. Kelompok harap didaftarkan pada form berikut http://bit.ly/KelompokFlowControl paling lambat H-2 batas pengumpulan (8 November 2016, 23.59). Kelompok yang tidak terdaftar sebelum tanggal tersebut tidak akan dianggap sebagai kelompok dan pengumpul tugas yang valid.

Tenggat waktu pengumpulan tugas ini adalah 10 November 2016, pk. 23.59, dengan teknis pengumpulan tugas diberitahu maksimal H-2 melalui milis mata kuliah IF3130 Jaringan Komputer, dengan seluruh file yang dikumpulkan dalam sebuah file kompresi (.zip) dengan penamaan IF3130_Flow_Control_KX-GYY, dengan X adalah nomor kelas [1, 2], dan YY adalah nomor grup sesuai pengisian formulir [01..30].

Hal-hal relevan yang kurang jelas ataupun tidak dicantumkan pada berkas soal ini dapat ditanyakan melalui milis. Peserta dianjurkan untuk menggunakan milis agar seluruh informasi dapat tersebar dengan merata ke seluruh pihak yang terlibat.

N.B. Segala tindak kecurangan yang diketahui oleh asisten akan ditindaklanjuti, dan akan diikuti konsekuensi apabila perlu.