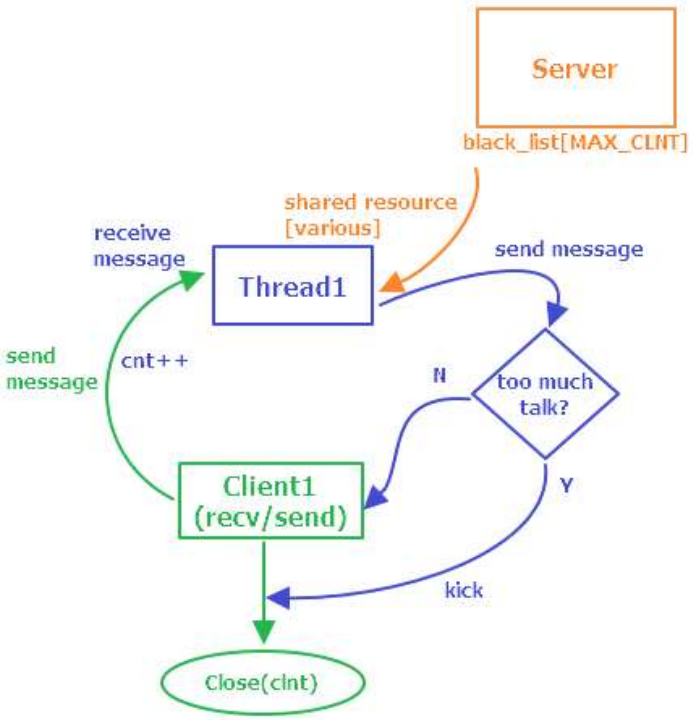
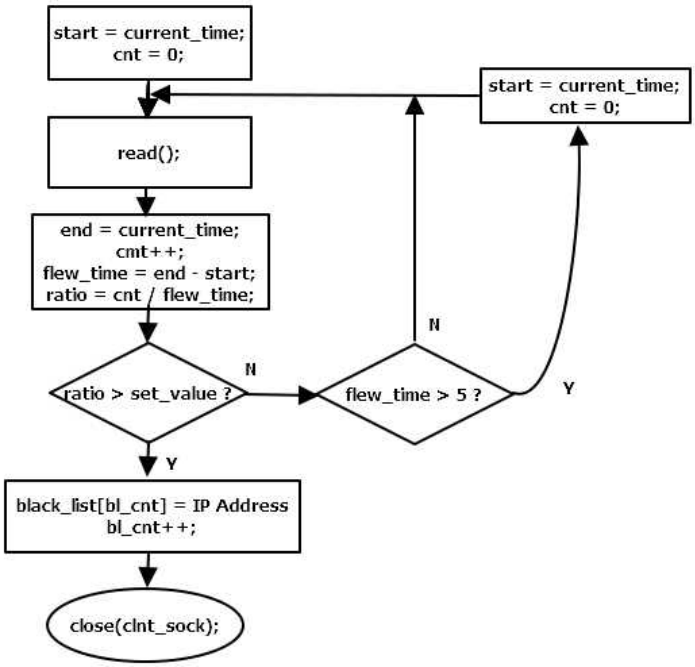
# TI DSP, MCU, Xilinx Zynq FPGA Based Programming Expert Program

**Instructor – Innova Lee (Sanghoon Lee)**
gcccompil3r@gmail.com
**Student – Howard Kim (Hyungju Kim)**
mihaelkel@naver.com

# Chatting Program - Add blocking a client




How to distinguish who's talking too much?
The answer is, similar to calculate differential coefficient.
You can get a percentage by dividing count by flown time.
Above flow chart describes an informal method.
If you want to make accurately, can use average.
More parameters, more accuracy.

chat_serv.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <pthread.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <sys/epoll.h>
#include <sys/time.h>
#include <stdbool.h>

#define BUF_SIZE     128
#define MAX_CLNT     256

typedef struct sockaddr_in si;
typedef struct sockaddr *  sp;
typedef struct timeval tv;

typedef struct __clnt{
    int clnt_sock;
    si clnt_addr;
}clnt;

int clnt_cnt = 0;
int clnt_socks[MAX_CLNT];
si ban_clnt_addr[MAX_CLNT];
int ban_cnt = 0;
pthread_mutex_t mtx;

double get_runtime(tv start, tv end);
bool chk_too_much(int cnt, double flew_time);
void kick_message(si clnt_addr,char* msg);
bool chk_ban_list(si clnt_addr);
void err_handler(char* msg){
    fputs(msg, stderr);
    fputc('\n', stderr);
    exit(1);
}

void send_msg(char* msg,int len){
    int i;

    pthread_mutex_lock(&mtx);

    for(i=0;i<clnt_cnt;i++)
        write(clnt_socks[i], msg, len);
    pthread_mutex_unlock(&mtx);
}

void* clnt_handler(void* arg){

    clnt st_clnt = *((clnt*)arg);
    int clnt_sock = st_clnt.clnt_sock;
    si clnt_addr = st_clnt.clnt_addr;

    int str_len = 0, i;
    char msg[BUF_SIZE];
    int cnt = 0;
```

```
59      tv start, end;
60      double flew_time = 0;
61      int sd_flag = -1;
62      gettimeofday(&start, NULL);
63
64      if(chk_ban_list(clnt_addr))
65          goto ban;
66
67      while((str_len = read(clnt_sock, msg, sizeof(msg))) != 0){
68  #if 1
69          if(chk_too_much(cnt,flew_time) && cnt > 5){
70              cnt = 0;
71              gettimeofday(&start, NULL);
72              sd_flag = shutdown(clnt_sock, SHUT_RD);
73              ban_clnt_addr[ban_cnt++] = clnt_addr;
74
75              printf("add ban list : %s\n",inet_ntoa(clnt_addr.sin_addr));
76          }
77          else if(flew_time > 10){
78              cnt = 0;
79              gettimeofday(&start, NULL);
80          }
81  #endif
82          send_msg(msg, str_len);
83          cnt++;
84          gettimeofday(&end, NULL);
85          flew_time = get_runtime(start,end);
86      }
87
88      if(sd_flag == 0)
89          kick_message(clnt_addr, msg);
90
91
92
93      pthread_mutex_lock(&mtx);
94
95      for(i=0;i<clnt_cnt;i++){
96          if(clnt_sock == clnt_socks[i]){
97              while(i++ < clnt_cnt - 1)
98                  clnt_socks[i] = clnt_socks[i+1];
99              break;
100         }
101     }
102     clnt_cnt--;
103     pthread_mutex_unlock(&mtx);
104 ban:
105     close(clnt_sock);
106     return NULL;
107 }
108
109 int main(int argc, char** argv){
110     int serv_sock, clnt_sock;
111     si serv_addr, clnt_addr;
112     socklen_t addr_size;
113     pthread_t t_id;
114
115     if(argc != 2){
116         printf("Usage : %s <port>\n",argv[0]);
117         exit(1);
```

```
118         }
119
120         pthread_mutex_init(&mtx, NULL);
121
122         serv_sock = socket(PF_INET, SOCK_STREAM, 0);
123
124         if(serv_sock == -1)
125             err_handler("socket() error");
126
127         memset(&serv_addr, 0, sizeof(serv_addr));
128         serv_addr.sin_family = AF_INET;
129         serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
130         serv_addr.sin_port = htons(atoi(argv[1]));
131
132         if(bind(serv_sock, (sp)&serv_addr, sizeof(serv_addr)) == -1)
133             err_handler("bind() error");
134
135         if(listen(serv_sock, 5) == -1)
136             err_handler("listen() error");
137
138         for(;;){
139             addr_size = sizeof(clnt_addr);
140
141             clnt_sock = accept(serv_sock, (sp)&clnt_addr, &addr_size);
142             if(chk_ban_list(clnt_addr)){
143                 close(clnt_sock);
144                 continue;
145             }
146
147             pthread_mutex_lock(&mtx);
148             clnt_socks[clnt_cnt++] = clnt_sock;
149             pthread_mutex_unlock(&mtx);
150
151             clnt st_clnt;
152             st_clnt.clnt_sock = clnt_sock;
153             st_clnt.clnt_addr = clnt_addr;
154
155             pthread_create(&t_id, NULL, clnt_handler, (void*)&st_clnt);
156             pthread_detach(t_id);
157             printf("Connected Client IP : %s\n",inet_ntoa(clnt_addr.sin_addr));
158         }
159         close(serv_sock);
160         return 0;
161     }
162
163     double get_runtime(tv start, tv end){
164         end.tv_usec = end.tv_usec - start.tv_usec;
165         end.tv_sec = end.tv_sec - start.tv_sec;
166         end.tv_usec += end.tv_sec * 1000000;
167         return end.tv_usec / 1000000.0;
168     }
169
170     bool chk_too_much(int cnt,double flew_time){
171         if(cnt > (int)flew_time*5)
172             return true;
173         return false;
174     }
175
176     void kick_message(si clnt_addr,char* msg){
```

```
177         char kick_msg[512];
178         sprintf(kick_msg,"kick [%s] : %s",inet_ntoa(clnt_addr.sin_addr),msg);
179         send_msg(kick_msg, strlen(kick_msg));
180     }
181
182     bool chk_ban_list(si clnt_addr){
183         int i;
184         for(i=0;i<ban_cnt;i++)
185             if(inet_ntoa(clnt_addr.sin_addr) == inet_ntoa(ban_clnt_addr[i].sin_addr))
186                 return true;
187         return false;
188     }
189
```