IP 카메라를 위한 서버 및 클라이언트 구현

임성락*, 이우영*
*호서대학교 컴퓨터공학과
e-mail : srrim@hoseo.edu

An Implementation of Server & Client for IP Camera

Seong-Rak Rim*, Woo-Young Lee*
* Dept. of Computer Engineering, Hoseo University

요 약

인터넷을 통하여 동영상을 실시간으로 감시하고 사용자의 요청에 따라 저장 및 재생할 수 있는 IP 카메라를 위한 서버 및 클라이언트를 리눅스와 윈도우즈 환경에서 각각 구현하여 CMOS 카메라가 연결된 EZ-S3C2440 임베디드 보드에서 실험하였다.

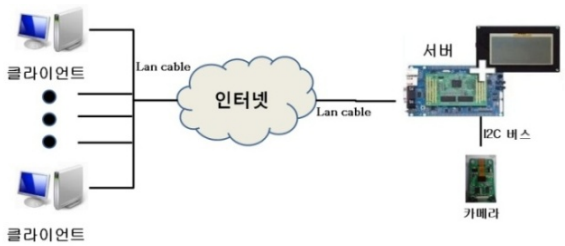
1. 서론

오늘날 감시 카메라의 설치가 증가하고 있으며 주로 CCTV(Closed Circuit Television) 시스템[1]과 DVR(Digital Video Recorder) 시스템[2]을 많이 이용하고 있다. 그러나 이러한 시스템들은 회사나 백화점처럼 규모가 큰 곳을 감시할 목적으로 구축되기 때문에 다양한 기능과 고성능의 시스템을 요구하고 있다. 따라서 이러한 감시 카메라를 개발하기 위해선 많은 비용과 시간이 소요되며 유지 보수에도 어려움이 있다.

본 논문에서는 인터넷으로 연결된 IP 카메라의 동영상을 실시간으로 원격 감시하고, 사용자 요청에 따라 동영상을 저장한 후 다시 볼 수 있는 서버 및 클라이언트를 구현하고 그 타당성을 검토하기 위하여 리눅스와 윈도우즈 환경에서 각각 구현하여 EZ-S3C2440 임베디드 보드[3]에서 실험하였다.

2. 시스템 설계

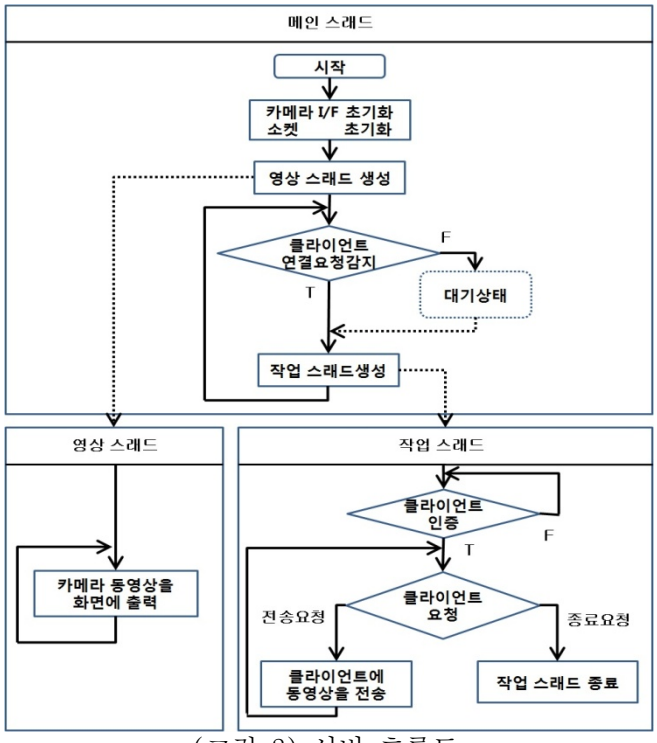
IP 카메라를 이용한 원격 감시 시스템의 전체 구성도는 (그림 1)과 같다. IP 카메라가 연결된 임베디드 서버는 카메라의 동영상을 자신의 화면에 출력하면서 동시에 인터넷을 통해 연결된 클라이언트들에게 동영상을 전송한다. 한편 서버에 접속된 클라이언트들은 서버로부터 수신된 동영상을 화면에 출력하면서 사용자의 요청에 따라 동영상을 저장 후 다시 볼 수 있다.



(그림 1) 시스템 전체 구성도

2.1. 서버

본 논문에서는 (그림 2)와 같이 다중 스래딩 기법을 이용하여 클라이언트의 다중 접속을 지원하도록 서버를 설계한다.

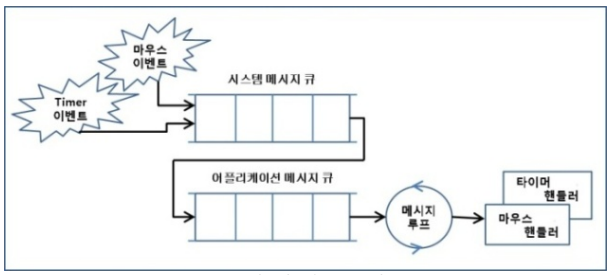


(그림 2) 서버 흐름도

(그림 1)에서 영상 스래드는 IP 카메라의 동영상을 임베디드 서버의 화면에 실시간으로 출력한다. 작업 스래드는 클라이언트로부터 수신된 사용자 ID 및 암호를 인증하고, 클라이언트로부터 수신된 요청(전송/종료)에 따라 처리한다. 전송 요청일 경우, 클라이언트들에게 동영상을 전송하고, 종료 요청일 경우, 작업 스래드를 종료한다.

2.2. 클라이언트

클라이언트는 서버로부터 수신된 동영상을 화면에 출력하면서 저장 및 재생을 지원 하도록 윈도우 환경에서 이벤트 기반(event-driven) 방식으로 설계한다. 윈도우 환경에서 이벤트가 발생할 경우 (그림 3)과 같은 메시지 큐를 이용하여 처리한다[4].



(그림 3) 클라이언트 이벤트 흐름

(그림 3)에서 윈도우 환경에서 발생하는 모든 이벤트는 일단 시스템 메시지 큐에 저장되고, 각각의 어플리케이션에 해당하는 이벤트들은 다시 해당 어플리케이션 메시지 큐에 저장한다. 메시지 루프는 어플리케이션 메시지 큐를 감시하여 해당 이벤트를 처리하는 핸들러로 이벤트를 보낸다. 본 논문에서는 마우스 이벤트와 타이머 이벤트를 이용하여 사용자의 요청과 동영상 출력을 처리하도록 한다.

마우스 이벤트 핸들러는 다음과 같은 사용자 요청을 처리한다.

① **연결** 혹은 **해지**: 입력된 IP 주소의 서버에 연결 혹은 해지를 시도한다. 연결 요청이 성공되면 사용자 ID 및 암호를 서버에 전송하여 서버로부터 수신된 동영상을 화면에 출력하고 동작상태를 출력상태로 설정한다. 해지 요청은 서버와의 연결을 해지한다.

② **저장** 혹은 **정지**: 저장 요청은 화면에 출력되고 있는 동영상을 저장장치에 저장하고 동작상태를 저장상태로 설정한다. 정지 요청은 저장작업을 정지하고 동작상태를 출력상태로 설정한다

③ **재생** 혹은 **정지**: 재생 요청은 저장장치에 저장된 동영상을 화면에 출력하고 동작상태를 재생상태로 설정한다. 정지 요청은 동영상 출력을 정지하고 동작상태를 출력상태로 설정한다.

타이머 이벤트 핸들러는 클라이언트의 동작 상태(출력/저장/재생)에 따라 동영상을 다음과 같이 처리한다.

① **출력** 상태: 서버로부터 수신한 동영상을 화면에 출력한다.

② **저장** 상태: 서버로부터 수신한 동영상을 저장장치에 저장한다.

③ **재생** 상태: 저장 장치로부터 동영상을 읽어 화면에 출력한다.

3. 구현 및 실험

설계한 서버 및 클라이언트를 리눅스와 윈도우즈 환경에서 각각 구현하여 EZ-S3C2440 임베디드 보드에서 실험한다.

3.1. 서버

메인 스래드는 (그림 4)와 같이 카메라 영상 프레임 버퍼의 크기를 320*240으로 설정 한 후 open()함수를 이용해 카메라 장치와 프레임 버퍼에 해당하는 파일 디스크립터를 얻어온다. 다중 스래드를 지원하기 위하여 pthread_create()를 사용하여 IP 카메라의 동영상을 임베디드 서버의 화면에 실시간으로 출력하기 위한 영상 스래드를 생성하고, 클라이언트로부터 수신된 사용자 ID 및 암호를 인증하고, 클라이언트로부터 수신된 요청을 처리하기 위한 작업 스래드를 생성한 후 다른 클라이언트의 접근을 감지한다.

```
int main(int argc, char** argv[]) {
        .
        .
   camif_cfg.dst_x = 320;
   camif_cfg.dst_y = 240;
   camif_cfg.bpp = 16;
   if((dev_cam = cam_open()) < 0) goto err;
   if((dev_fb = fbopen()) < 0) goto err;
   if(ioctl(dev_cam, 초_CAMERA_INIT, &camif_cfg)) goto err;
        .
        .
   pthread_create(&v_thread,NULL,&view_thread,&camif_cfg);
        .
        .
   while(1) {
      clnt_sock = accept(seev_sock,&clnt_addr,&clnt_addr_size);
      pthread_create(&w_thread,NULL,&work_thread,&clnt_sock);
   }
        .
        .
}
```

(그림 4) 메인 스래드

영상 스래드는 (그림 5)와 같이 카메라 파일 디스크립터 번호를 사용하여 영상을 읽어 카메라 영상을 memcpy() 함수를 사용하여 프레임 버퍼에 복사하여 화면에 출력한다.

```
void * view_Thread(void * arg) {
        .
        .
   read(dev_cam, rgb, arg->dst_x, arg->dst_y*2);
        .
        .
   draw(fb, &rgb[0], arg->dst_x, arg->dst_y, arg->bpp) {
        .
        .
      for(y=0; y<end_y; y++) {
         memcpy(dest+(((y+yray)*FB_WIDTH*2)+xray), src+width*2, end_x*2);
      }
   }
        .
        .
}
```

(그림 5) 영상 스래드

```
void * work_Thread( ) {
        .
        .
    for( i=0; i<s_num; i++) {
        if(strcmp(c_id, s_id[i]) == 0) {
            if(strcmp(c_pw, s_pw[i] == 0) {
                send(*clnt_sock,"1",1,0);
            }
            else {
                send(*clnt_sock,"2",1,0);
            }
        }
    }
    send_app(void) {
        send(*ip,s_size, 5, 0);
        send(*ip,ipg_temp, s_size, 0);
    }
}
```

(그림 6) 작업 스레드

작업 스레드는 (그림 6)과 같이 클라이언트에서 보내온 시용자 ID와 암호를 검사하여 접근을 허락하고 클라이언트에게 동영상을 전송한다.

3.2. 클라이언트

클라이언트는(그림 7)과 같이 소켓 및 클라이언트의 컨트롤러들을 초기화하고 사용자의 요청을 받아들이기 위한 메뉴를 다이얼로그 방식으로 생성한 후 생성된 다이얼로그의 창을 화면에 출력한다.

```
BOOL CClientApp::InitInstance() {
    if (!AfxSocketInit()) {
            AfxMessageBox(IDP_SOCKETS_INIT_FAILED);
            return FALSE;
    }
    AfxEnableControlContainer();
    .
    .
    CClientDlg dlg;
    m_pMainWnd = &dlg;
    int nResponse = dlg.DoModal();
    .
    .
    return FALSE;
}
```

(그림 7) 메뉴 생성

사용자 요청을 받아들이기 위한 마우스 이벤트 핸들러는 (그림 8)과 같이 영상보기 클래스, 영상저장 클래스, 영상재생 클래스로 구현한다. 영상보기 클래스는 서버에서 인증검사를 통과한 경우 동작 상태를 출력 상태로 설정한다. 영상저장 클래스은 현재 클라이언트의 동작 상태가 출력 상태인 경우에만 저장 상태로 설정한다. 영상재생 클래스는 동작 상태를 재생 상태로 설정한다.

```
void CClientDlg::OnViewer()
{
    if(ID_check()) {
        stats = "view_stats";
    }
}
void CClientDlg::OnRecode()
{
    if(stats  == "view_stats") {
        fp = fopen(rec_name,"wb");
        stats = "save_stats";
    }
}
void CClientDlg::OnPlay()
{
    fp = fopen(Play_name,"rb");
    stats = "review_stats";
}
```

(그림 8) 마우스 이벤트 핸들러

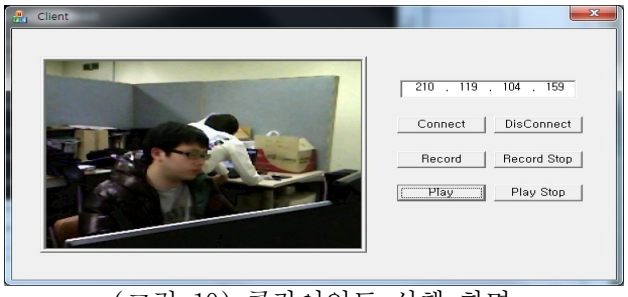
타이머 이벤트 핸들러는 (그림 9)같이 클라이언트의 동작 상태에 따라 동작 하도록 구현 한다. 출력 상태일 경우 서버에서 동영상을 수신 받아 화면에 출력한다. 저장 상태일 경우 서버에서 동영상을 수신 받아 저장장치에 저장하고 화면에 출력한다. 재생 상태일 경우 저장된 동영상을 화면에 출력한다.

```
void CClientDlg::Timer()
{
    switch(stats)
    {
    case "save_stats":
        m_socket->Send(Sendmeg, Sendmeg.GetLength());
        m_socket->Receive(buf, p_size, 0);
        break;
    case "view_stats":
        m_socket->Send(Sendmeg, Sendmeg.GetLength());
        m_socket->Receive(buf, p_size, 0);
        fwrite(buf,sizeof(char),size,fp);
        break;
    case "play_stats"
        fwrite(fp,sizeof(char),size,fp);
        break;
    }
    if( stats == "save_stats" || stats == "view_stats" ||
            stats == "view_stats" )  {
        fwrite(buf,sizeof(char),size,fb);
    }
}
```

(그림 9) 타이머 이벤트 핸들러

3.3. 실험

서버의 IP 주소와 사용 요청(연결/해지, 저장/정지, 재생/정지) 버튼을 구현하고, 연결 버튼으로 서버에서 인증 확인 후 수신된 동영상의 실시간 화면 출력을 확인한다. 마지막으로 저장 버튼을 통해 저장장치에 저장을 한 후, 재생 버튼으로 저장된 동영상 화면 출력을 확인한다.



(그림 10) 클라이언트 실행 화면

4. 결론

본 논문은 인터넷으로 연결된 IP 카메라의 동영상을 실시간으로 원격 감시하고, 사용자 요청에 따라 동영상을 저장한 후 다시 볼 수 있는 서버 및 클라이언트 구현하고 그 타당성을 검토하기 위하여 리눅스와 윈도우즈 환경에서 각각 구현하여 EZ-S3C2440 임베디드 보드에서 실험하였다.

참고문헌

[1] 최응렬, 김연수, “방범용 CCTV의 범죄 예방 효과에 관한 연구”, 한국동안행정학회 제 26권 2007.5

[2] 전황수, “DVR 시장 동향 및 국내외 개발 현황”, 전자통신동향분석 24권 3호, 2009.6

[3] FALinux 포럼, http://forum.falinux.com

[4] 홍차연, “멀티미디어 응용을 위한 Multithreaded 내장형 윈도우 시스템”, 서울대학교, 2003.2