블루투스를 이용한 DirectShow기반의 H.264 동영상 플레이어의 설계 및 구현

Design and Implementation of a H.264 Video player based on DirectShow via Bluetooth

박태준* · 조태훈**

Taejun Park* and Tai-Hoon Cho**

* 한국기술교육대학교 전기전자공학과

** 한국기술교육대학교 정보기술공학부

요 약

블루투스는 저전력으로 근거리상의 디바이스들을 무선으로 연결하여 통신하는 기술이나 저속의 통신속도를 갖고 있다. 따라서 파일이나 네트워크상의 동영상 스트리밍을 위한 플레이어는 많지만 Bluetooth를 통한 동영상 재생기는 찾아보기 힘든 상황이다. 낮은 데이터 전송 속도를 갖는 Bluetooth를 통해 동영상을 전송시 인코더로는 높은 압축 효율을 보이는 MPEG-4 AVC/H.264가 적합하다. DirectShow는 Microsoft사가 개발한 소스필터 기반의 멀티미디어 프레임워크이자 API이다. DirectShow를 이용하여 다른 어플리케이션에서도 사용할 수 있는 필터를 쉽고 빠르게 개발할 수 있다. 본 논문에서는 H.264방식으로 인코딩된 동영상 데이터를 블루투스를 통해 전송받아 디코딩하여 재생시킬 수 있는 DirectShow 기반의 플레이어의 설계 및 구현 방법에 대해 기술한다. 다양한 비디오 샘플을 가지고 실험하여 개발된 프로그램의 타당성을 입증하였다.

키워드 : DirectShow, H.264, Source Filter, Multimedia, Bluetooth

Abstract

Bluetooth is a popular wireless data transmission method with low power consumption, but it has low data transmission rate. Thus, although many video stream players of a local or network file exist, there have been few players of video stream transmitted via Bluetooth. MPEG-4 AVC/H.264 codec is one of video codecs available with best compression rates for a certain quality, so a H.264 encoder seems to be adequate for video stream to be transmitted via Bluetooth. In this paper, we present a DirectShow filter based player of video stream encoded by H.264 codec, which is transmitted via Bluetooth. Details on the design and implementation of this program are described. Experimental results are shown to demonstrate the validity of the implemented program using various video samples.

Key Words : DirectShow, H.264, Source Filter, Multimedia, Bluetooth

1. 서 론

최근 인터넷 속도는 비약적으로 증가하였지만 Bluetooth[1]나 Zigbee같은 저전력 무선 통신장치의 전송속도는 그에 미치지 못하고 있다. 따라서 저전력 무선 통신장치의 전송속도에서도 실시간으로 동영상을 스트리밍할 수 있는 높은 압축률을 자랑하는 동영상 압축 기술의 필요성이 요구되었다. 동영상 압축 코덱으로는 Motion JPEG 2000[2], MPEG-2[3], MPEG-4 ASP(Advanced Simple Profile), MPEG-4 AVC(Advanced Video Coding)/H.264[4]등이 있다. 이중 Motion JPEG는 동영상의 각 프레임을 JPEG 영상

압축 코덱으로 압축하는 코덱으로 구현하기 쉬운 장점이 있지만 압축률이 다른 동영상 코덱들 보다 현저히 낮은 특징을 보인다. 또한 MPEG-2, 4 ASP는 버전이 올라감에 따라 압축률이 높아지지만 MPEG-4 AVC/H.264 코덱(이하 H.264) 보다 낫다. Bluetooth 2.0+EDR의 대역폭은 이론상으로 최대 3Mbit/s이고 실제 데이터 전송속도는 최대 2.1Mbit/s이다. 이러한 낮은 전송속도 때문에 병목현상이 발생되는데 이러한 병목현상을 최대한 줄이기 위해서 압축률이 높은 코덱의 선택이 필수적이다. 따라서 높은 압축률을 자랑하는 H.264코덱이 적합하다. H.264 코덱은 기존의 표준(MPEG-2, H.263, MPEG-4 ASP)과 비교했을 때, 낮은(절반이하의) 비트레이트(bitrate)에서 비슷하거나 더 좋은 화질을 얻을 수 있도록 개발되었다. 현재 파일이나 네트워크상의 동영상 스트리밍을 위한 플레이어는 많지만 Bluetooth같은 무선전송장치를 이용한 동영상 재생기는 찾아보기 힘든 상황이다. 본 논문에서는 저전력 무선 통신장치인 Bluetooth를 사용하여 전송받은

접수일자 : 2009년 4월 6일

완료일자 : 2009년 6월 26일

“본 논문은 본 학회 2009년도 춘계 학술대회에서 선정된 우수논문입니다”

H.264 동영상 스트림을 복호화하여 화면에 출력하는 프로그램의 구현에 대하여 기술한다. 이러한 동영상 플레이어는 DirectShow[5]기술을 이용하여 구현을 하였다. DirectShow란 미국의 Microsoft사에서 개발한 멀티미디어 스트림을 제어하는 기술을 COM(Component Object Model)으로 구현할 수 있게 하는 프레임워크이며 필터기반으로 이루어져 있다. DirectShow를 이용하여 필터를 개발하게 되면 다양한 필터들의 혼합적용, 우수한 이식성, 필터개발의 용이성 등의 장점을 갖는다. 2장에서는 Bluetooth, MPEG-4 AVC/H.264, DirectShow에 대하여 알아보고 3장에서는 프로그램의 설계를 하며 4장에서는 설계한 프로그램을 구현에 대하여 기술한다. 5장에서는 구현한 프로그램을 사용하여 여러 가지 옵션으로 인코딩한 H.264 raw format파일을 이용하여 실험 및 결과를 보여준다. 마지막으로 6장에서는 본 논문의 결론을 맺는다.

2. 관련 기술

2.1 Bluetooth

IEEE 802.15.1규격을 사용하는 Bluetooth는 1994년 에릭슨스르초로 개발한 (PANs)Personal Area Networks의 산업 표준이다. 본 논문에서는 Bluetooth 2.0+EDR을 사용한다. Bluetooth 2.0+EDR은 전송속도는 서론에서 언급한 것서 같이 이론상 르스전송 속도는 3.0M.0M/s이고 실제 데이터 전송속도는 2.1M.0M/s이다. 이러한 전송속도를 고려하여 동영상을 전송하여야 한다. Bluetooth를 이용하여 통신을 하개발한 (는 prof0le을 이용하여야 한다. prof0le은 장치들발개의 Bluetooth interface 규격을 폼다.해 Ⅱ 산업것이다. Bluetooth는 총 28가지다.prof0le을 지원한다. 본 논문에서는 서론P해 Ⅱ 산업것이통신을 하였다. 서론P(서rial Port Prof0le)는 ET서IDR서 07.1쉴 서 RFCOMM 프로토콜D사용반으로 하고 있으며 Bluetooth를 이용하여 R서ot3t 기반의 시리쉴이통신을 마치 유선으로 연결한 것처럼 예물레이션 한다. 서론P해 Ⅱ 산업게 되면 마치 Bluetooth이용하여 데이터해 전송하고 받는 절차들에 대하여 서erial통신 하듯이 하개발때문에 프로그램을 구현할 때 Serial통신을 하도록 구현하면 된다.

2.2. MPEG-4 AVC/H.264

H.264는 매우 높은 데이터 압축률을 가지는 디지털 비디오 코덱의 표준으로 MPEG-4 파트 10 또는 AVC(Advanced Video Coding)이라 부른다. 본 논문에서는 H.264 Raw format을 Bluetooth를 이용하여 전송한다. Raw format은 NAL(Network Abstraction Layer)유닛으로 구성되어 있다.

그림 1에서 보는 것과 같이 NAL유닛은 NAL Header와 RBSP(Raw Byte Sequence Payload)로 구성되어 있다. RBSP는 슬라이스 데이터를 저장하고 있는데 슬라이스의 구성은 그림 2와 같다.

그림 1. NAL 유닛 구성
Fig 1. Structure of NAL Units

그림 2에서 MB는 Macro block을 의미하고 skip_run은 연속적으로 생략되어지는 매크로 블록의 개수를 의미한다. Macro block은 영상 압축을 할 때의 단위이다.

NAL유닛의 첫 부분에 있는 데이터가 startcode이다. 이것은 NAL의 처음을 알려주는 코드로 다음과 같다.

0000 0000 0000 0000 0000 0000 0000 0001 (1)

startcode는 NAL의 처음을 알려주기 때문에 startcode를 기준으로 나누면 NAL 단위로 나눌 수 있다. startcode 뒤에 있는 8비트는 NAL header이다.

그림 2. 슬라이스 구성
Fig 2. Structure of Slice

이 NAL header를 복호화하여 NAL Unit Type을 결정한다. 그림 3과 같이 하위 5개 비트를 기반으로 NAL 유닛타입을 결정한다. 각 타입에 따른 값은 표 1과 같이 할당된다. 이 타입은 실제 H.264 raw format에서 SPS와 PPS는 각각 7과 8을 의미한다. 그리고 I-Frame은 5이고 P-Frame은 1이다. 실제로 NAL Unit Type은 표1과 같이 정의되어 있지만 직접 H.264 raw format을 분석해보면 표1에 쓰여 있는 것으로는 직관적으로 알기 힘든 것을 알 수 있다.

그림 3. NAL Unit Type 복호
Fig 3. Decryption of NAL Unit Type

표 1. NAL 유닛 타입
Table 1. NAL Unit Type

타입	숫자
SLICE(P-Frame)	1
DPA	2
DPB	3
DPC	4
IDR(I-Frame)	5
SEI	6
SPS	7
PPS	8
AUD	9
EOSEQ	10
EOSTREAM	11
FILL	12

NAL Unit Type은 크게 보면 총 4가지의 타입이 있다. SPS(Sequence Parameter Set), PPS(Picture Parameter Set), I-Frame 그리고 P-Frame이다. SPS는 프로파일 ID 나 레벨, 참조 프레임 개수, 화면의 크기 등의 정보를 가지고 있다. PPS는 엔트로피 부호화 방법, 슬라이스 그룹 개수, 가중치 예측 방법, 디블록킹 필터 사용 여부, 8x8 변환 모드 사용 여부 등의 정보를 가지고 있다. 이러한 SPS와 PPS는 H.264의 헤더라고 볼 수 있다. I-Frame은 정지영상을 압축하는 것과 동일한 방법을 사용한 한 화면이다. 그리고 P-Frame은 이전에 재생된 영상을 기준으로 삼아 기존 영상과의 차이점만을 보충하여 재생하는 화면을 의미하며 다음에 재생 될 P-Frame의 기준이 된다.

H.264를 인코딩 할 때에는 두 가지 방식이 존재한다. 첫 번째로 CBR(Constant Bitrate)이다. CBR은 인코딩 데이터의 크기가 일정하도록 하는 기술이다. 두 번째로 VBR(Variable Bitrate)이다. VBR은 CBR과는 다르게 일정한 인코딩 크기를 유지하는 것이 아니라 데이터가 많이 필요로 하는 부분에는 더 많은 bitrate를 할당하고 데이터가 필요없는 부분에는 적은 bitrate를 할당하는 기술이다. 그래서 VBR로 인코딩한 결과물은 CBR로 인코딩한 결과물보다 상대적으로 더 품질이 좋다.

2.3. DirectShow

DirectShow는 미국의 Microsoft사에서 개발한 필터(Filter)기반의 멀티미디어 프레임워크(Framework)이다. DirectShow를 이용하면 멀티미디어 어플리케이션을 하드웨어 가속 지원을 받으면서 보다 쉽고 빠르게 개발할 수 있다. DirectShow는 COM(Component Object Model) 프레임워크 기반으로 만들어져 있으며 미디어(Media)를 위한 인터페이스를 제공한다. DirectShow는 필터를 만들거나 기존의 필터들을 조합하여 사용한다. DirectShow는 필터(Filter)라고 불리는 부품을 이어 맞추어 필터 그래프(filter graph)를 작성해 미디어를 조작하는 구조이다. 필터는 크게 3가지로 구분할 수 있다. 첫 번째로 Source Filter이다. Source Filter는 저장장치나 네트워크 혹은 캡처카드 같은 미디어의 소스로부터 미디어를 얻어오는 역할을 한다. 두 번째로 Transform Filter이다. Transform Filter는 다른 필터에서 받은 데이터를 변환(Transform)하는 필터이다. 예를 들면 다른 필터로부터 받은 압축된 데이터의 디코딩이나 영상에 자막을 넣는다던지 하는 필터들이 Transform Filter이다. 세 번째로 Renderer Filter이다. Renderer Filter는 다른 필터에서 받은 데이터를 렌더링(Rendering) 한다. 일반적으로 화면에 영상을 그리거나 사운드카드를 이용해서 소리를 들려주는 필터들이다.

그림 4. 필터 그래프

Fig 4. Filter Graph

3. 프로그램 설계

전장에서 언급한 DirectShow기반으로 프로그램을 만들기 때문에 동영상을 보여주기 위해서 필요한 Filter중 만들

어야 하는 것과 기존의 것을 사용해도 되는 것을 결정해야 한다. 본 논문에서는 Bluetooth의 SPP 프로파일을 이용하기 때문에 Serial port에서 H.264 Raw format으로 전송되는 데이터를 받기 위해 Source Filter를 만들기로 하였다. Source Filter로 받은 H.264 데이터를 디코딩을 하기 위한 Transform Filter가 필요하기 때문에 기존에 구현되어 있는 ffdshow video decoder[11]를 사용하였다. ffdshow는 ffmpeg[12]를 DirectShow와 프FW(Video For Window) 기반으로 영상과 음성의 여주기와 디코딩을 할 수 있는 프로그램으로 ffdshowow)제공하는 Directshow H.264 디코딩 필터가 ffdshow video decoder이다. 마지막으로 화면에 디코딩 된 동영상을 출력해줄 renderer filter가 필요한데 이 renderer filterrect본적으로 제공되는 Video Renderer를 사용하였다. 해서적으로 본 논문에서는 Source Filter를 구현하는데 초점이 맞추어rect본관림 4ow)보는 것과 같이 필터 그래프를 작성하였다.

Source Filter를 구현하기 위해서는 DirectShow에서 요구하는 요구사항에 맞춰 만들어야 한다. Source Filter는 DirectShow SDK에서 제공하는 BaseClasses 프로젝트에 있는 CSource 클래스를 상속 받아야 하고 Source Filter의 interface인 SourceStream 클래스는 CSourceStream 클래스를 상속 받아야 한다. 또한 DirectShow는 COM기반이기 때문에 COM을 구현하기 위해 필요한 요구사항도 맞춰야 한다. 본 논문에서 구현한 Source Filter는 총 2개의 클래스로 구성되어 있다. 첫 번째는 Source Filter의 본체인 SerialSourceFilter 클래스이다. SerialSourceFilter는 Source Filter의 Pin을 관리하는 기능을 한다. 두 번째로 Source Filter의 Interface를 담당하는 Pin인 SerialSourceStream 클래스가 있다. SerialSourceStream 클래스는 실질적인 Source Filter의 거의 모든 역할을 하는 클래스로 다른 필터와의 통신을 담당하고 있다. Source Filter나 Source Stream 클래스는 각각 상속받는 클래스에 꼭 구현해야 하는 순수가상함수들이 정의되어 있다. 그리고 Serial port에서 받은 데이터들을 NAL단위로 저장하기 위해서 Slice 구조체도 정의하였다. 그림 5를 보면 전체적인 프로그램의 구조를 볼 수 있고 표 2를 보면 각 클래스별로 구현해야 할 함수들을 볼 수 있다.

그림 5. 간략화된 프로그램의 구조

Fig 5. Simplified Structure of the Program

표 2. 구현해야 할 함수

Table 2. Functions to Implement

CTS_SerialSourceFilter	
함수명	기능
CreateInstance	Pin 인스턴스 생성

CTS_SerialSourceStream	
함수명	기능
OpenPort	Serialport 열기
DecideBufferSize	버퍼 협상
GetMediaType	미디어 협상
FillBuffer	버퍼를 채우기
read_th_func	Bluetooth에서 데이터를 받아 저장

4. 프로그램 구현

3장에서 설계하고 정의한 함수들을 구현하면 Source Filter가 완성된다. 다음은 각 클래스에서 구현이 필요한 함수들에 대한 설명이다.

4.1 CTS_SerialSourceFilter

CTS_SerialSourceFilter는 소스 필터의 몸체에 해당하는 부분으로 Source Filter의 인스턴스(Instance)를 생성할 때 만들어지는 클래스이다. 이 클래스에서는 Source Filter의 interface인 SourceStream 클래스의 인스턴스 생성과 관리를 맡는다.

4.2 CTS_SerialSourceStream

실질적으로 Source Filter의 역할을 하는 부분이 SerialSourceStream이다. SerialSourceStream의 구조는 그림 5와 같다. Head와 Tail 변수는 Slice 구조체의 포인터 변수로써 Linked List 자료구조를 이용하여 Queue를 구현하기 위해 사용하는 변수이다. LinkedList를 이용하여 Queue를 구현하는 이유는 동영상 스트리밍 데이터가 실시간으로 재생시 필요한 데이터 전송속도보다 더 빠른 속도로 전송이 된다면 LinkedList가 아닌 고정 길이 배열 같은 자료구조를 사용할 경우 오버플로우(overflow)가 일어나 오작동을 할 가능성이 있기 때문에 LinkedList를 이용한 Queue를 사용한다. Queue를 사용하는 함수로는 FillBuffer함수와 read_th_func함수가 있다.

4.2.1 read_th_func

read_th_func함수는 Bluetooth에서 전송한 데이터를 NAL단위로 나누어 Queue에 넣는 역할을 하는 함수이며 SourceStream 클래스 생성자(Constructor)에서 쓰레드(thread)로 생성되어 독립적으로 동작하게 된다. 이 함수에서 Queue에 NAL Unit을 넣게 된다.

4.2.2 OpenPort

이 함수는 Serial port를 이용하기 위하여 여는 함수로 read_th_func함수가 Bluetooth를 통하여 데이터를 받으려면 반드시 먼저 호출되어야 하는 함수이다. OpenPort 함수는 win32 API(Application Programming Interface)를 이

용하여 구현하였다. 이 함수는 SourceStream 클래스의 생성자에서 호출된다.

그림 6. SourceStream 클래스의 클래스 다이어그램

Fig 6. Class diagram of the SourceStream

4.2.3 GetMediaType

이 함수는 다른 필터와 연결이 가능한지를 알아보기 위하여 MediaType을 비교하게 되는데 이때 MediaType을 얻기 위해 호출하는 함수가 GetMediaType 함수이다.

표 3. VIDEOINFO 구조체의 구조

Table 3. Structure of VIDEOINFO

```
typedef struct tagVIDEOINFO {
    RECT          rcSource;
    RECT          rcTarget;
    DWORD         dwBitRate;
    DWORD         dwBitErrorRate;
    REFERENCE_TIME AvgTimePerFrame;
    BITMAPINFOHEADER bmiHeader;
    union {
        RGBQUAD bmiColors[256];
        DWORD dwBitMasks[3];
        TRUECOLORINFO TrueColorInfo;
    };
} VIDEOINFO;
```

표 4. BITMAPHEADER의 구조

Table 4. Structure of BITMAPHEADER

```
typedef struct tagBITMAPINFOHEADER {
    DWORD biSize;
    LONG biWidth;
    LONG biHeight;
    WORD biPlanes;
    WORD biBitCount;
    DWORD biCompression;
    DWORD biSizeImage;
    LONG biXPelsPerMeter;
    LONG biYPelsPerMeter;
    DWORD biClrUsed;
    DWORD biClrImportant;
} BITMAPINFOHEADER;
```


이 함수는 CMediaType 포인터 pMediaType를 매개변수로 받아 pMediaType에 동영상 정보를 넣어준다. 동영상 정보는 pMediaType->AllocFormatBuffer함수를 이용하여 VIDEOINFO 포인터를 리턴 받아 정보를 넣게 된다. 표 3을 보면 VIDEOINFO의 구조를 볼 수 있다. VIDEOINFO 구조체에서 볼 수 있는 중요한 값으로는 AvgTimePerFrame가 있다. 이 값은 동영상의 FPS(Frame Per Second)를 결정하는 값으로 한 프레임의 시간을 100ns 단위로 설정 할 수 있다. VIDEOINFO를 보면 FPS관련 변수를 제외하면 동영상의 정보가 거의 없는 것을 볼 수 있는데 이는 VIDEOINFO 구조체 안의 bmiHeader 변수에 대부분의 정보가 있기 때문이다. 표 4를 보면 bmiHeader의 자료형인 BITMAPHEADER의 구조를 볼 수 있다. 표 4에서 정의되어 있는 BITMAPHEADER의 구조를 보면 동영상의 여러 가지 정보를 넣을 수 있는데 중요한 값으로 biCompression이 있다. biCompression은 압축된 동영상의 경우 FOURCC(Four-Character Codes) 코드값을 넣도록 되어 있다. FOURCC는 영상 코덱을 구분할 때 사용하는 코드로 본 논문에서는 H.264를 사용하기 때문에 H.264 코드를 넣는다. 그밖에 biWidth나 biHeight는 동영상의 크기를 나타내는 변수로 동영상의 크기를 넣으면 된다.

4.2.4 DecideBufferSize

DecideBufferSize 함수는 연결된 다른 필터와 공유하는 버퍼의 크기를 결정하는 함수이다. 이 함수가 호출되면 실제로 공유할 버퍼를 할당하고 사용하게 된다. 본 논문에서는 표 4의 biSizeImage의 크기를 buffer의 크기로 설정한다.

4.2.5 FillBuffer

FillBuffer함수는 연결된 다른 필터에서 동영상 스트림을 요구할 때 호출되는 함수이다. FillBuffer함수는 한번 호출될 때 마다 read_th_func함수에서 Queue에 누적된 NAL 유닛을 Queue로부터 순차적으로 읽어서 버퍼에 넣고 리턴을 한다. 만약 NAL 유닛 타입이 SPS나 PPS이면 I-Frame나 P-Frame이 나올 때까지 버퍼에 누적하여 리턴 한다. Queue에 NAL 유닛이 없으면 read_th_func함수가 Bluetooth에서 NAL 유닛을 받아 Queue에 받을때까지 대기하도록 구현하였다.

Server는 동영상을 실시간으로 보낼 수 있도록 노트북에서 Visual C++ 6.0을 이용하여 프로그램을 제작하여 실험하였다. Client는 동영상 플레이어이다. 실험 영상으로는 320x240(WxH) 해상도의 H.264 Raw format의 파일로 총 300프레임이며 VBR(Variable Bitrate)와 CBR(Constant Bitrate)로 인코딩된 영상 45개를 사용하였다. 이러한 환경에서 실험 영상을 전송하여 재생을 해본 결과 일부 동영상에서 재생 초기에 버퍼링이 발생하는 것을 알 수 있었고 이 현상은 초기의 I-Frame이 다 전송이 되어야 화면에 보여줄 수 있기 때문에 I-Frame이 전송되는 동안 시간지연이 생기는 것을 볼 수 있었다. 이러한 현상은 CBR로 인코딩된 동영상에서만 발견 할 수 있었으며 CBR과 VBR로 인코딩된 동영상을 비교해 본 결과 CBR은 I-Frame의 크기가 P-Frame에 비해 상대적으로 컸다. 반면에 VBR은 I-Frame과 P-Frame간의 크기 차이가 적었다. 이러한 차이점은 전송속도가 높은 장치에서 전송할 때에는 느낄 수 없었던 시간지연 현상을 발생하게 했다

시간지연 현상은 CBR의 크기가 큰 I-Frame이 Bluetooth로 다 전송될 때까지 대기함으로서 발생하게 되어 실제로 Bluetooth를 이용하여 동영상을 전송할 때에는 CBR이 적합하지 않고 VBR이 적합하다는 사실을 알게 되었다. Bluetooth 2.0+EDR은 이론상 전송속도가 아닌 실제 전송속도가 2.1Mbit/s 이다. 이러한 속도는 15프레임을 기준으로 3.3KB크기의 프레임을 전송할 수 있는 전송속도로 Bluetooth로 통신이 원활하지 않을 경우에는 더 느려지기 때문에 실시간 스트리밍을 하기 위해서는 주위환경을 고려하여 프레임 사이즈를 조절할 필요가 있다.

일반적인 동영상 스트리밍의 경우 UDP같이 패킷이 손실되더라도 재전송을 하지 않아 실시간이라는 특성을 잘 살리는 프로토콜을 사용하지만 본 논문에서는 Bluetooth의 SPP를 사용했기 때문에 Bluetooth의 특성에 따라 주위환경의 문제로 전송 패킷(Packet)이 손실되면 Bluetooth는 재전송을 하게 된다. 심한 경우에는 장애가 사라질 때까지 버퍼에 담아두고 있다가 송신을 한다. 이러한 Bluetooth의 특성은 실시간 동영상 스트리밍을 하기에 부적합한 특성을 보인다. 이러한 특성을 제거한 Profile이 나온다면 주위 환경에 영향이 덜 받을 수 있을 것이다. 마지막으로 Motion JPEG과 H.264를 비교해보면 같은 품질의 영상을 전송하기 위해서 필요한 대역폭이 월등하게 H.264가 더 낮음을 볼 수 있었다.

5. 실험 및 결과

실험은 표 5와 같은 실험환경에서 하였다.

표 5. 실험 환경
Table 5. An Experimental Environment

Server	Client
Core Duo L2400 1.5GB RAM	AMD Phenom 9550 4GB RAM Microsoft DirectX 9.0 SDK update (summer 2003)
Visual C++ 6.0 Bluetooth 2.0+EDR	Visual C++ 6.0 Bluetooth 2.0 + EDR

그림 7. Bluetooth로 전송되는 동영상의 캡처 화면
Fig 7. Caputre shots of video transmitted via Bluetooth

6. 결 론

본 논문에서는 Bluetooth의 SPP 프로파일을 이용하여

전송받은 H.264 비디오 데이터를 DirectShow를 이용하여 동영상 플레이어에 설계, 구현하였다. DirectShow 기반의 플레이어를 구현하려면, 소스 필터, 트랜스폼 필터, 랜더러 필터가 필요한데, 이 중 시리얼포트로부터 비디오 스트림을 입력받는 소스필터를 직접 제작하고 나머지 필터는 기존에 개발되어 있는 필터를 사용하였다. 다양한 비디오 샘플을 가지고 실험하여 개발된 프로그램의 타당성을 입증하였다. 하지만 현재 동영상이 재생 중에 화면에 보여줘야 할 프레임이 다 받아지지 않으면 화면이 멈추는 현상이 있다. 이러한 현상을 없애는 것이 앞으로의 연구 과제이다.

참 고 문 헌

- [1] Bluetooth SIG, Specification of the Bluetooth System, <http://www.bluetooth.com>, 2003.
- [2] M. W. Marcellin, M. Cornish, A. Bilgin, M. Boliek, "An Overview of JPEG 2000," *Proc. IEEE Data Compression Conference*, Snowbird, Utah, March 2000.
- [3] B. G. Haskell, A. Puri, and A. N. Netravali, *Digital Video: An Introduction to MPEG-2*, New York: Chapman & Hall, Sept. 1996.
- [4] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra: "Overview of the H.264/AVC Video Coding Standard", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, pp. 560-575, July 2003.
- [5] Web, "DirectShow SDK Documentation Page", <http://msdn.microsoft.com/en-us/library/>.
- [6] 신화선, *DirectShow 멀티미디어 프로그래밍*, 한빛 미디어, 서울, 2002.
- [7] Stephan Wenger, "H.264/AVC over IP", *IEEE Transactions on Circuits and Systems for Video Technology*, vol.13, no.7, pp 645-656, July 2003.
- [8] Thomas Stockhammer, Miska M, Hannuksela, and Thomas Wiegand, "H.264/AVC in Wireless Environments", *IEEE Transactions on Circuits and Systems for Video Technology*, vol.13, no.7, pp. 657-673, July 2003.

- [9] E. Richardson, *H.264 and MPEG-4 Video Compression*, John Wiley, 2003.
- [10] 호요성, 김승환, *H.264 AVC 알고리즘 이해와 프로그램 분석*, 두양사, 2008.
- [11] Web, "ffdshow official homepage", <http://sourceforge.net/projects/ffdshow>.
- [12] Web, "ffmpeg official homepage", <http://ffmpeg.org/>.

저 자 소 개

박태준(Taejun Park)

2008년 : 한국기술교육대학교 컴퓨터공학과 학사

2008년~현재 : 한국기술교육대학교 전기 전자공학과 석사과정

Phone : (010)3487-5700

E-mail : crazyvirus@gmail.com

조태훈(Tai-Hoon Cho)

1981년 : 서울대학교 전자공학과 학사

1983년 : 한국과학기술원 전기 및 전자공학과 석사

1991년 : Virginia Polytechnic Institute & State University 박사

1992년~1998년 : LG산전 연구소 책임/수석 연구원

1998년~현재 : 한국기술교육대학교 정보기술공학부 조교수/부교수/교수

Phone : (041) 560-1351

Fax : (041) 564-3261

E-mail : thcho@kut.ac.kr