BREW 동영상 스트리밍의 이슈

윤민홍° 정진환 유 혁
고려대학교 컴퓨터학과 운영체제 연구실
{mhyun, jhjeong, hxy}@os.korea.ac.kr

Issues at Video Streaming on BREW

Min-Hong Yun° Jin-Hwan Jeong Hyuck Yoo
Dept. of Computer Science and Engineering, Korea University

요 약

휴대폰과 PDA 와 같이 휴대하기 용이한 장치를 통해 많은 사람이 동영상을 보고있다. 하지만 짧은 시간동안 플레이 되는 컨텐츠를 보기위해 더 많은 시간을 투자해서 컨텐츠를 다운로드 해야하며 짧은 재생시간은 사용자에게 불만을 준다. 이 단점을 피하기 위해 스트리밍 서비스를 하는데, 휴대폰이라는 제한된 영역에서 그리고 다중 타이머가 존재하지 않는 등의 열악한 BREW 환경에서 스트리밍 서비스를 하기 위해서는 고려해야 할 사항들이 많다. 이 논문은 BREW 기반의 휴대폰에서 동영상 프로그래밍을 할 때 고려해야 할 사항들과 스트리밍 서비스를 하기위해 고려해야 할 사항들을 살펴보고 문제가 될 수 있는 부분에 대해서 대안을 제시한다.

1. 서론

휴대폰과 PDA 와 같은 휴대하기 편한 장치(device)가 널리 보급됨과 동시에, 이 장치들이 가지고있는 프로세서(processor)의 연산 능력(computation power)이 향상되었고, 이 장치들이 갖는 디스플레이(display)의 해상도(resolution)가 높아지고 응답시간(response time)이 짧아졌다. 이 결과 사용자들은 동영상 컨텐츠(contents)를 단말기(terminal)에서 플레이 할 수 있었다. 그러나 현재 대부분의 단말기는 PC 에 비해 열악한 무선 네트워크환경을 가지고 있기 때문에 스트리밍 보다는 컨텐츠를 다운로드한 후 플레이 한다. 그렇기 때문에 전체 컨텐츠의 크기는 각 단말기의 저장장치의 크기에 크게 영향을 받는다. 만약 단말기가 PDA 가 아닌 휴대전화라면 컨텐츠의 크기는 더더욱 작아진다. 또 열악한 무선 네트워크 환경을 가지고있기 때문에 컨텐츠의 크기가 커지면 사용자는 그 컨텐츠를 다운로드하기 위해 플레이 시간보다 더 긴 시간을 다운로드에 투자해야 한다.

이 논문에서는 BREW(Binary Runtime Environment for Wireless)[1] 플랫폼(platform) 상에서 IDCT 와 Motion Compensation 의 과정을 뺀 중간 데이터(Intermediate data)를 사용한 동영상 플레이어[2][3] 프로그래밍[4]을 설명하고 중간데이터를 사용해서 동영상 스트리밍 시에 고려해야 할 사항들과 효과적인 접근 방법에 대해 소개한다.

2. 실험 및 구현 환경

실험과 구현은 Qualcomm 사가 제공하는 BREW 1.1 library 를 사용해 에뮬레이터(emulator)에서 작업을 마친 뒤 실제 단말기에서 동작하도록 했다.

단말기는 LG 전자의 VX10 제품이다[5]. BREW 1.1 을 탑재하고있는 이 단말기는 ARM7TDMI 프로세서를 사용하며[6] BREW 애플리케이션(application)을 위해 200KB 의 힙(heap)을 제공하고 약 2KB 의 스택 (stack)을 제공한다. 이 단말기의 디스플레이는 4 그레이(4-level gray)이며, 해상도는 120 x 100 을 지원한다. 이 단말기는 CDMA2000 (ISO-95C)망을 지원하고 테스트는 이 망에 연동해서 진행했다.

3. 본론

3.2 단말기의 제한적인 환경

휴대폰 단말기는 탑재한 프로세서의 열악한 계산 능력, 적은 메모리 용량의 제한 점을 가지고 있으며, BREW 를 탑재한 단말기는 다중 타이머(timer)의 부재, 전역 변수(global variable) 사용의 불가능과 그로 인해 파생되는 여러 문제점들 때문에 비디오 디코더(decoder)에 매우 결정적인 단점들을 가지고 있다[4].

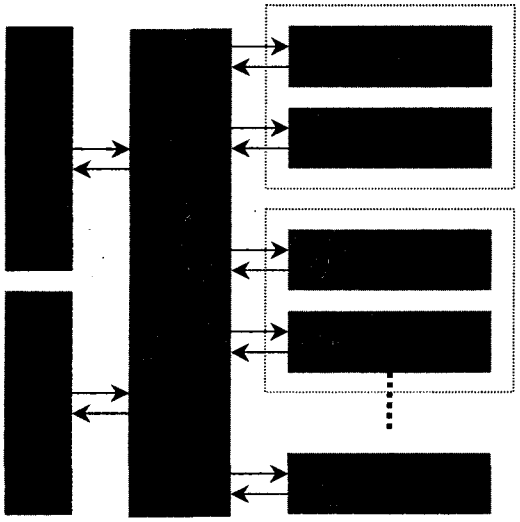
3.3 단말기의 제한적인 환경을 극복하는 프로그래밍 방법

휴대폰 단말기, 더욱이 BREW 를 탑재한 휴대폰 단말기에서의 프로그래밍을 효과적으로 하기위해서는 다음의 방법을 따라야 한다[4].

- 큰 간격(interval)의 타이머를 작은 간격의 타

이머를 여러 번 호출하는 것으로 대체
- 적당한 크기의 지역변수(local variable)를 사용함으로써 스택 오버플로우(stack overflow)의 방지
- 자주 사용되는 변수를 포인터(pointer) 연산을 피하게 해서 메모리 접근을 최소화하는 방법
- 초기에 CONVERTBMP() 함수(function)을 한번 호출해서 그 결과 만들어진 헤더(header)를 재사용함으로써 연산시간이 오래 걸리는 CONVERTBMP() 함수의 호출을 최소화 하는 방법

디코더 응용프로그램을 최적화 하기 위해서 전체적인 디코더의 디자인을 효과적으로 해야하는데, 그것을 위해서는 [그림 1]의 방법을 따라야 한다.

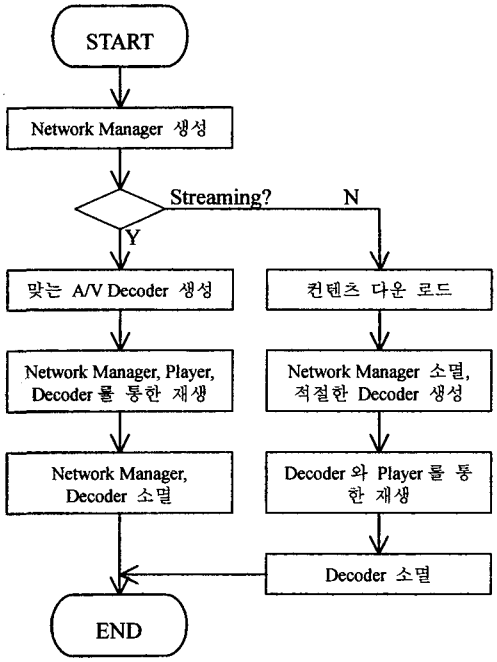


[그림 1] 플레이어 모듈의 디자인

사용자는 User Interface 를 통해 응용 프로그램을 제어하며 Player 는 UI(User Interface)의 입력을 분석해 필요에 따라 다른 모듈들을 생성하고 소멸시킨다. Network Manager 는 다운로드 플레이와 스트리밍 서비스의 네트워크 부분을 담당하는 모듈이고, Video Decoder 와 Audio Decoder 및 Text Decoder 는 코덱(codec) 부분으로 여러 코덱이 존재할 수 있다.

동영상 플레이어와 같은 규모가 큰 응용프로그램을 만들 때 가장 쉽게 부딪히는 문제가 메모리 부족 현상과 스택 오버플로우 문제이다. 이를 피하기 위해 응용프로그램을 [그림 1]과 같은 방법으로 모듈별로 디자인 한 다음 각 모듈은 자신의 모듈에서 사용하는 모든 변수의 관리하게 해야 한다. 전체적인 흐름은 다음 [그림 2]와 같다.

사용자는 UI 를 통해 스트리밍 서비스 또는 다운로드 플레이를 할 것인지 명령한다. UI 는 Player 객체(Object)를 생성한 다음 사용자가 명령한 내용을 Player 에게 전달한다. Player 는 Network Manager 를 생성한다.



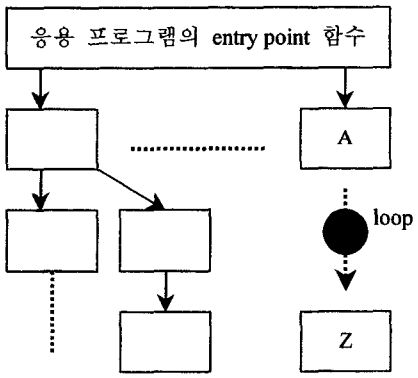
[그림 2] 응용 프로그램의 흐름도

a. 사용자가 스트리밍 서비스를 요청했을 경우(Y) Player 는 Network Manager 를 통해 서버에서 제공한 컨텐츠의 종류를 구별하고, 적절한 디코더를 생성한다. Player 는 Network 을 통해 들어오는 비디오 데이터를 디코더에 전달해 디코딩한 다음 UI 를 통해 화면에 보여주고 소리로 들려준다. 이러한 스트리밍 재생이 끝나면 Player 는 더 이상 필요 없는 Network Manager 와 Decoder 를 소멸시킨다.

b. 사용자가 다운로드 플레이를 요청한 경우(N) Network Manager 를 통해 컨텐츠를 다운로드 한다. 다운로드가 완료되면 더 이상 필요 없는 Network Manager 를 소멸시키고, 다운로드된 컨텐츠를 플레이하기 위해 적절한 디코더를 선택하여 생성한다. Player 는 다운로드된 컨텐츠를 생성한 디코더를 사용해서 디코딩하고, UI 를 통해 화면에 보여주고 소리로 들려준다. 재생이 완료되면 Player 는 더 이상 필요 없는 디코더를 소멸시킨다.

이 방법으로 얻을 수 있는 장점은 두 가지가 있다.

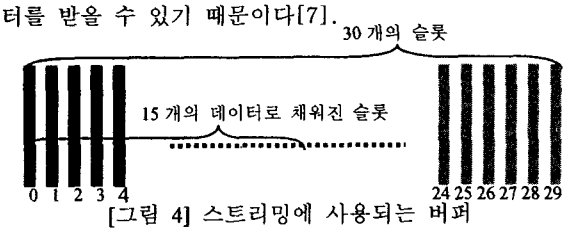
모듈을 필요에 따라 생성 또는 삭제한다는 것은 응용 프로그램이 사용하는 실행코드를 메모리에 적재한다는 것이 아니고, 모듈의 실행 코드는 이미 메모리에 적재된 상태에서 모듈이 필요로 하는 변수들에게 메모리 영역을 할당해준다는 것을 의미한다. 따라서 첫번째 장점은 실험에 사용한 VX10 는 200KB 만의 힙을 제공하기 때문에 이 제한적인 힙을 필요한 변수들에게만 효율적으로 할당하게 된다는 것이다. 두 번째 장점은 BREW 에서는 전역 변수를 사용할 수 없기 때문에 시작 지점(entry point)가 되는 함수에서 힙을 할당한 다음 그것에 대한 포인터를 하위 함수에 파라메터(parameter)로 전달을 한다. 그런데 자주 사용되는 함수에서 이런 파라메터 전달 방법을 사용하면 성능의 저하가 일어나는데[4]([그림 3]), [그림 1]과 같은 디자인을 적용하면 힙을 할당하는 위치가 응용 프로그램의 시작 지점이 아닌 모듈을 생성하는 위치가 되기 때문에 이러한 성능 저하를 줄일 수 있다.



[그림 3] 콜 스택과 전역변수

[그림 3]에서 시작 함수는 이미 힙을 할당해서 할당된 메모리가 필요한 함수들에게는 파라메터 형식으로 전달한다. 시작 함수가 함수 A 를 호출하고 함수 A 는 다시 몇 개의 함수를 호출할 것이다. 호출된 함수들 중 loop 이 존재하고, 이 loop 에서는 다시 어떤 함수를 호출하고 호출된 함수가 Z 를 단 한번 호출하고 Z 는 시작함수에서 할당한 힙의 1byte 가 필요하다고 가정하자. 그러면 힙에대한 포인터를 단 한번 사용하기 위해 거대한 콜 스택에 이 포인터가 불필요하게 푸쉬(push)되고 팝(pop)되는데 이것이 디코더의 성능에 큰 영향을 미친다.

스트리밍 서비스를 하기 위해서는 다운로드 플레이와는 다르게 버퍼관리가 추가된다. 실험에서 구현한 스트리밍 서비스는 UDP 를 사용하며 각 데이터그램(datagram)의 최대 크기는 1500Byte 이다. 구현에서는 링 버퍼를 사용해 버퍼 슬롯(slot)을 30 개를 유지하며 반 이상의 슬롯이 채워지면 플레이를 시작하도록 했다([그림 4]). 이렇게 한 이유는 CDMA 2000 망에서 50,000 bps 에서 80,000bps 사이에 맞추어 서버가 동영상 데이터를 전송해야 90%이상의 높은 정확도로 데이터를 받을 수 있기 때문이다[7].



[그림 4] 스트리밍에 사용되는 버퍼

4. 결론

휴대폰과 같은 적은 메모리를 갖고있는 단말기에서는 다운로드 플레이 만으로는 사용자를 만족 시킬 수 없다. 스트리밍 서비스를 하기위한 BREW 용 응용 프로그램을 만들 때 전화기 자체의 제약사항과, BREW 의 제약사항을 고려해야 한다. CPU 의 낮은 계산 능력과 적은 메모리, 그리고 전역 변수를 사용할 수 없다는 결점과 그에 따르는 불필요한 연산의 증가는 디코더와같이 반복적이고 많은 연산을 하는 응용 프로그램[8]에는 큰 제약사항이 될 수 있다. 이처럼 한정된 자원 때문에 타이머 멀티플렉싱을 하고 메모리 버스를 쓰지 않기 위해 메모리 접근을 최소화하며 연산 시간이 많이 들어가는 함수의 호출을 최소화 해야 한다. 뿐만 아니라 큰 응용 프로그램에서 효과적으로 메모리를 관리하기 위해 적절한 모듈 디자인이 필요하며 CDMA2000 네트워크의 특성을 고려해서 버퍼를 관리해야 한다.

참고 문헌

[1] Qualcomm, " BREW SDK User's Guide", " http://www.qualcomm.com/brew/" , July 2001

[2] Jin Hwan Jeong, Chuck Yoo, " A Server-centric Streaming Model" , NOSSDAV, pp.25-34

[3] 한종민,한상범,정진환, 유 혁, " BREW 기반의 동영상 플레이어 구현" , 제 29 회 정보과학회 추계학술발표회 논문집, Vol. III, 691, Oct. 2001.

[4] 윤민홍, 류은석, 유혁, " 동영상 플레이어를 위한 BREW (Binary Runtime Environment for Wireless) 프로그래밍의 최적화 기법, 제 17 회 정보처리학회 춘계학술발표회 논문집, Vol. 9, 1187-1190, Apr 2002

[5] Qualcomm," BREW-Enabled Device Data Sheet for Application Developers" , " https://www.qualcomm.com" , Jan 2002

[6] ARM, " ARM7TDMI (Rev 4) Technical Reference Manual" , " http://www.arm.com/arm/documentation?OpenDocument" , 2001

[7] 김일진,정진환,유혁," 효율적인 멀티미디어 데이터 전송을 위한 CDMA-2000 망의 특성 분석 연구" , 제 29 회 정보과학회 추계학술발표회 논문집, Vol.III, 751, Oct. 2001

[8] 한상범, " 계산 자원을 최소화하는 화상회의 기법 연구" , 고려대학교 석사논문, Aug. 2000.