자바에 기반을 둔 비디오 스트리밍 시스템의 설계 및 구현

박정애 °, 정진환, 한상범, 허재석, 유혁
{japark, jhjeong, sbhan, jsheo}@os.korea.ac.kr, hxy@joy.korea.ac.kr
고려대학교 컴퓨터학과

Design and Implementation of Java-based Streaming System

Jeong-ae Park °, Jin-hwan Jeong, Sang-beom Han, Jae-seok Heo, Chuck Yoo
Dept. of Computer Science and Engineering, Korea University

요 약

본 논문에서는 자바 기반의 비디오 스트리밍을 위한 시스템의 설계와 구현에 대해 논한다. 웹 브라우저를 통해 수행되는 애플릿(applet) 형식의 클라이언트는 지역 디스크에 존재할 필요가 없고 플랫폼에 대한 독립성(independence)과 이식성(portability)을 가질 수 있다. 본 논문에서는 실험을 통해서 이러한 비디오 스트리밍 시스템의 유용성을 검증했다.

1. 서론

인터넷이 널리 보급되면서, 인터넷에 대한 일반인들의 관심이 증가하고 있다. 그리고 Fast Ethernet, Gigabit Ethernet, ATM 등의 보급을 통해서 그 동안 문제가 되었던 네트워크의 속도가 고속화되고 대역폭이 빠르게 증가함에 따라 컴퓨팅 환경은 과거에 비해서 놀라울 정도로 향상되었다.

네트워크의 성능이 향상되면서 일반 텍스트(text)나 정지화상(JPEG) 데이터뿐만 아니라 대용량의 실시간 동영상 데이터 처리도 가능하게 되었다. 최근에는 동영상 데이터 파일을 디스크에 미리 전송 받을 필요 없이 실시간으로 네트워크에서 전송 받아 출력시키는 스트리밍 기술이 널리 보급되고 있다.

이러한 실시간 스트리밍 기술에 대한 대표적인 예로서 리얼 네트워크사의 리얼 플레이어 시스템(Real Player System) [1]이나 마이크로소프트사의 넷쇼 서비스(NetShow Service) [2]등 을 들 수 있다. 그러나 이런 시스템은 구동되기 위해서 클라이언트 프로그램이 지역 디스크에 존재할 필요가 있다. 뿐만 아니라 플랫폼에 의존적(dependent)이기 때문에 이질적인 플랫폼에는 쉽게 이식 할 수 없다는 단점이 있다.

이러한 단점을 극복하기 위한 방법으로 Java[3]에서 제공하는 애플릿이 있다. 애플릿은 네트워크를 통해 서버로부터 지역 메모리(local memory)로 전송되어진 후, 넷스케이프나 익스플로러 같은 웹 브라우저(web browser)가 설치된 호스트 상에서 수행된다. 이처럼 애플릿은 다른 애플리케이션과 달리 지역 디스크에 존재하지 않고 전송되어 웹 브라우저의 지역 문맥(context) 내에서 실행된다. 애플릿을 이용할 경우 네트워크가 가능한 환경이 마련되고, 웹 브라우저가 설치되어 있는 호스트만 있다면, 별도의 애플리케이션을 갖추지 않고도 네트워크로부터 전송 받은 비디오 스트림을 처리 할 수 있는 클라이언트의 구현이 가능하다.

그러므로 본 연구진은 시스템의 클라이언트 부분을 애플릿을 사용하여 구현하였다. 우리의 클라이언트는 지역 디스크에 존재할 필요가 없을 뿐만 아니라, 자바 가상 머신(java virtual machine)상에서 실행되기 때문에 플랫폼에 독립적인 장점을 가지게 된다.

본 논문에서는 서버에서 비디오 스트림을 전송하고, 네트워크에서 들어오는 비디오 스트림을 애플릿을 통해 스크린에 출력하는 비디오 스트리밍 시스템(video streaming system)을 설계하고 구현하였다. 또한 실험을 통해 시스템을 분석해 봄으로써 시스템의 유용성을 알아보았다.

2. 시스템 개요

본 논문에서 구현한 비디오 스트리밍 시스템의 전체적인 구조는 다음 그림과 같다.

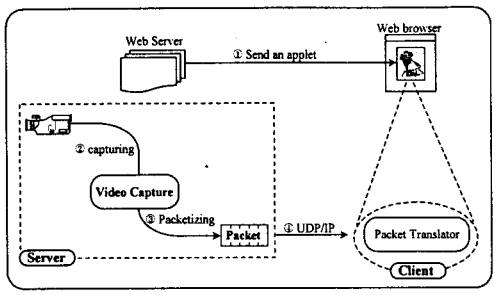

그림 1. 시스템 구조

비디오 스트리밍 시스템을 구현하기 위해서 서버는 카메라를 통해서 나오는 비디오 스트림의 각 프레임을 캡쳐하고, 이 비디오 데이타를 변환하여 패킷으로 만든 후에 네트워크로 보내게 된다. 클라이언트는 네트워크로부터 들어오는 패킷을 분석해서 데이터 값을 얻은 뒤 이를 적절한 포맷으로 변환하여 화면에 출력하게 된다.

서버에서 클라이언트로 패킷을 전송할 때 사용되는 네트워크 프로토콜[4에는 TCP(Transmission Control Protocol)와 UDP(Unreliable Control Protocol)가 있다. TCP는 신뢰성 있는 데이터 전송을 목적으로 하는 프로토콜로서 네트워크 혼잡(congestion) 등으로 인해서 패킷이 정상적으로 클라이언트에게 전송되지 못했을 때, 패킷의 재전송을 통해서 데이터의 신뢰성을 높이게 된다. 이러한 재전송 메카니즘은 오버헤드가 매우 크며, 멀티미디어 데이터의 주기적 특성을 보장하지 못하고 스트리밍의 실시간적 특성에도 부적합하다. 반면에 UDP는 패킷의 일부가 유실 될 수 있는 가능성을 가지고 있지만, 전송 자체의 오버헤드가 비교적 적고 실시간 전송에 필요한 여러 가지 메커니즘을 쉽게 추가할 수 있어서 실시간 비디오 스트리밍에서 유용하다. 따라서 시스템에서는 네트워크 전송 프로토콜로서 UDP를 사용하며, LAN(Local Area Network) 환경에서 구현되었다.

3. 시스템 구현

3.1. 서버

서버는 카메라를 통해서 캡쳐된 비디오 스트림 데이터를 실시간으로 전송하는 기능을 수행한다. 비디오 프레임의 캡쳐를 위해서 마이크로소프트사에서 제공하는 C++ 클래스 라이브러리(예 : CBaseOutputPin class)를 사용한다. 비디어 데이터는 변환 필터를 통해서 RGB 값으로 변환된 값을 얻을 수 있다.(그림 2) 그리고 네트워크로 전송하기 전에 데이터를 스트리밍에 적합하게 변환하고 비디오 데이터에 대한 헤더 정보를 추가한 후, 패킷으로 만들어서 네트워크로 전송하게 된다. 이 때 전송되는 데이터의 프레임의 수는 카메라의 캡쳐 비율인 초당 30프레임으로 하였다.

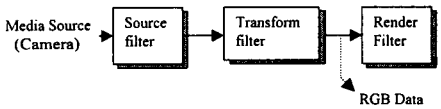

그림 2. 서버의 데이터 흐름 구조

3.2. 클라이언트

클라이언트는 자바를 사용해 애플릿으로 구현되며 사용자에게 전송되어 온 비디오 스트림을 재생하는 기능을 웹 브라우저 상에서 수행하게 된다.

클라이언트는 바이트 코드 형태로 컴파일 되어서 웹 서버에 저장된다. 사용자는 웹 브라우저를 통해 웹 서버에서 애플릿 형식의 클라이언트를 전송 받아서 이것을 수행한다. 애플릿은 자바 가상 머신에서 수행되기 때문에 플랫폼 독립성과 이식성을 얻을 수 있다.

클라이언트는 네트워크로부터 전송되어 온 패킷을 분석하는 모듈, 패킷에서 데이터를 추출하여 출력 가능한 형태로 변환하는 모듈 그리고 화면에 출력하는 모듈로 구성된다. 네트워크 모듈에서는 UDP로 전송되어온 패킷을 처리한다. 그리고 변환 모듈에서는 패킷에서 추출한 비디오 데이터 값을 적절한 데이터 형태로 변환하여 출력 가능한 상태로 만든다. 최종적으로 출력 모듈에서는 이미지를 구성하고 비디오 정보를 대입한 후 화면에 출력하는 역할을 수행한다. 이 때 사용되는 자바의 API가 플랫폼에 독립적인 특성을 제공한다.

본 연구진이 구현한 클라이언트가 자바 가상 머신 상에서 수행될 때 발생하는 부하로 인해서 비디오 스트리밍에 미치는 영향을 다음의 실험을 통해서 분석하고 살펴보자.

4. 실험 및 평가

본 연구진은 실제 네트워크 전송 과정의 지연(delay)이나 혼잡(congestion) 등이 발생하지 않는다고 보고, 이에 대한 추가적인 제어 메커니즘을 시스템에 고려하지 않았다. 또한 클라이언트에서 패킷 처리 시에 발생할 수 있는 패킷 손실(drop)을 다루는 패킷 흐름 제어 메커니즘을 고려하지 않았다. 따라서 본 논문의 실험은 네트워크의 대역폭이 충분히 제공된다는 가정 하에 이루어진다. 실험에 사용된 시스템을 살펴보면, 아래와 같다.

서버/클라이언트	CPU	Pentium II 350MHz
	Memory	128MB
	OS	Windows NT 4.0
네트워크 환경	Fast Ethernet	
컴파일러 환경	JDK 1.2 (Java 2 Platform)	

표 1. 시스템 사양

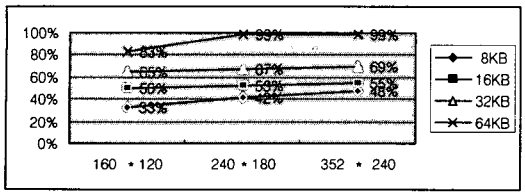

그림 3. 클라이언트의 CPU 이용률

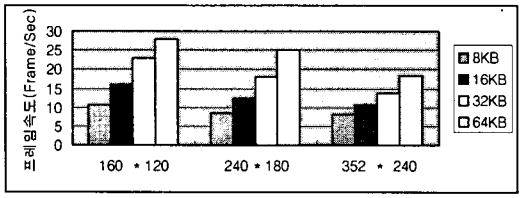

그림 4. 클라이언트의 프레임 출력 속도

서버는 프레임의 크기가 160×120, 240×180, 352×240으로 설정된 패킷을 초당 30프레임의 비율로 클라이언트에게 전송한다. 이 때 클라이언트가 패킷을 처리하는데 소요되는 CPU 이용률을 계산하였다.(그림 3) 실험 결과를 살펴보면 프레임의 크기가 증가할수록 CPU 이용률이 증가하고 있는데 이는 프레임의 크기가 증가함에 따라 CPU가 처리해야 하는 데이터 량이 증가했기 때문이다.

그리고 커널 내의 소켓 버퍼의 크기를 8KB, 16KB, 32KB, 64KB로 증가시킴에 따라 CPU 이용률이 증가하고 있다. 그 이유는 버퍼링을 통해서 처리할 수 있는 데이터 량이 증가했기 때문이다. 따라서 같은 크기의 프레임에 대하여 클라이언트의 소켓 버퍼를 증가시킴에 따라 프레임 출력 속도가 증가함을 알 수 있다.(그림 4)

현재 구현된 서버는 패킷을 일정한 비율로 전송한다. 따라서 클라이언트의 CPU 처리 속도가 패킷이 서버로부터 네트워크를 통해 전송되는 속도보다 느릴 경우, 커널 내의 소켓 버퍼에서 발생되는 오버플로우(overflow)로 인해서 전송되어 온 패킷을 잃어버리는 현상이 발생할 수 있다. 그러므로 버퍼 크기가 작을 수록 손실되는 패킷의 수가 증가하므로 클라이언트에서 출력되는 프레임 수가 줄어드는 경향을 보인다.

그림 3과 4에서 클라이언트의 버퍼 크기가 8KB인 경우를 살펴보면, CPU가 여유가 있음에도 불구하고 버퍼 크기가 작아 패킷 손실이 발생하기 때문에 프레임 속도와 CPU 이용률이 모든 경우에 큰 차이를 보이지 않고 있다. 그러나 버퍼 크기가 64KB인 경우를 살펴보면, 충분한 버퍼링에도 불구하고 비디오 데이터에 대한 연산 부하로 인해서 프레임 크기에 따라 프레임 속도가 비교적 크게 차이를 보이게 된다.

이러한 CPU 연산 부하를 측정하기 위해서 본 실험에서는 Java Profiler Performance Tool[5]를 사용하였다. 측정 결과

구분	기 능	CPU 이용률(%)
①	Receive From Socket Buffer	2.16
②	Data Reconstruction	28.23
③	Set Data Value	45.51
④	Draw Image	21.49
⑤	Misc.	2.61

표 2. 클라이언트의 CPU 사용 분석

(표 2)를 살펴보면, 소켓 버퍼로부터 데이터를 받는데는 전체 CPU 사용 시간 중 2.16%이 소요되고 있다.(①) 반면 비디오 데이터를 적절한 타입으로 변환한 후, 실제 이미지에 데이터 값을 대입하고 출력하는 과정에서 67% 정도의 CPU 시간이 소요되고 있음을 알 수 있다.(②③④) 본 시스템에서 사용하는 자바의 AWT 패키지는 플랫폼에 독립적인 API를 제공하기 때문에 이미지 출력 함수가 비디오 메모리에 데이터를 쓰기 위해서는 데이터 복사가 중복해서 일어나게 된다. 따라서 이미지를 출력하는 과정에서 높은 CPU 이용률을 나타내게 된다.

5. 결론 및 향후 계획

비디오 스트리밍은 VOD(Video on Demand) 서비스, 화상 회의(Video Conference) 및 원격 교육, 원격 진료 등 여러 분야에 활용되어 질 수 있는 중요한 기술이다. 본 논문에서는 기존의 실시간 스트리밍 시스템의 문제점을 해결하기 위해서 카메라를 통해 나온 비디오 데이터를 전송 받은 후 실시간으로 재생하는 클라이언트를 애플릿으로 설계하고 구현하였다.

각 프레임 크기별로 비디오 데이터를 전송한 후 클라이언트에서 실험을 통해 CPU 이용률과 프레임 출력 속도 등을 알아보았다. 앞으로는 클라이언트에서 출력하는 과정에서 발생하는 부하를 줄이는 방안을 연구함으로써 본 연구진이 구현한 시스템의 구조를 최적화 할 예정이다.

[참고문헌]

[1] http://www.real.com/
[2] http://www.microsoft.com/
[3] http://java.sun.com/
[4] W. Richard Stevens, Gary R. Wright, TCP/IP Illustrated, Volume 1, Addison-Wesley Publishing Company, 1994
[5] http://www.oprimizeit.com/