모바일 P2P 멀티미디어 서비스를 위한 경량 스트리밍 서버 Cappuccino Bubble 설계 및 구현

전성규^{O}, 김태형
한양대학교 컴퓨터공학과
{skjun^{O}, tkim}@cse.hanyang.ac.kr

Design and Implementation of A Light-Weight Streaming Server Cappuccino Bubble for Mobile P2P Multimedia Service

Seong-Kyu Jeon^{O}, Tae-Hyung Kim
Dept. of Computer Science & Engineering, Hanyang University

요 약

일반적으로 모바일 단말기는 스트리밍 서비스의 클라이언트로서 그 역할을 담당하고 있으며 여기에 스트리밍 서버를 설치하는 것은 매우 도전적인 과제이다. 본 논문에서는 RTP 프로토콜을 이용하여 Jpeg 형식의 스트리밍 데이터를 송수신 할 수 있는 초경량 스트리밍 서버를 모바일 단말기에 구현함으로써 궁극적으로 모바일 단말기간의 P2P 기반 스트리밍 서비스 네트워크를 구축하는 것을 목표로 한다. 본 시스템에서는 현재 차세대 Mobile 인터넷 플랫폼 표준으로 채택된 WIPI를 기반 시스템으로 하였으며 향후 모바일 단말기간 본격적인 P2P 기반 멀티미디어 서비스 시스템으로 확장성을 고려한 개발을 진행중이다.

1. 서론

현재 급속도로 발전하고 있는 Mobile 네트워크 기술과 포스트 PC시대에 부응하는 고성능 휴대형 단말기들의 출현으로 과거 PC상에서나 가능했던 일들이 이제는 손바닥 위에서 구현되고 있으며 멀티미디어의 구현도 충분히 가능해지고 그 가운데 Mobile Multimedia Streaming Service에 대한 관심이 높아지고 있다.

본 논문에서는 Mobile Multimedia Streaming Service에서 모바일 단말기가 단순히 Client기능뿐만 아니라 Server의 역할까지도 수행할 수 있음을 보여주고자 한다. 일반적인 스트리밍 서버는 모바일 단말기에 적재될 수 있는 수준을 초과하므로 경량 시스템의 부족한 자원 환경에 최적화된 스트리밍 서버를 구축하여 단말기간 스트리밍 서비스를 제공하는 것에 대한 연구가 필요하다.

J2ME기반의 WIPI 에뮬레이터와 SK의 XCE에뮬레이터에서 하나의 Application을 구동하는데 전체 메모리 heap size가 2M를 넘지 않는다[1]. 그러므로 메모리 heap size를 최적화할 수 있는 스트리밍 서버를 구축하는 것이 본 논문의 핵심내용이다.

또한, 음성이나 영상 위주의 멀티미디어 데이터는 파일의 전송과는 달리 신뢰성보다 데이터의 실시간성이 보장되어야 한다. 왜냐하면 전송 중에 packet이 손실되더라도 그 손실이 일정량 이하면 소프트웨어적으로 보상이 가능하며, 보상이 안되더라도 전체적인 영상을 보는데 큰 지장이 없기 때문이다. 이러한 필요성 때문에 기존의 TCP/IP 프로토콜과는 별도의 RTP (Real-time Transport Protocol), RSVP (ReSerVation Protocol) streaming 프로토콜이 등장하게 되었다.

스트리밍 프로토콜은 Handshaking 기반의 전송 방식과 같이 재전송을 통한 오류 복구 능력은 없지만, 실시간 전송이 가능하기 때문에 오류가 발생하더라도 약간의 화질 열화만 나타나고 멀티미디어 데이터를 끊어짐 없이 계속적으로 전송할 수 있다. RTP는 기존의 IP를 이용하기 위해 접속 기반이 아닌 UDP를 주로 사용한다. 특히, RTP는 서비스 품질 정보를 교환하고 네트워크의 상태를 파악하기 위해 RTCP (Real Time Control Protocol)을 사용하기 때문에 현재의 네트워크 상태에 적합한 데이터 품질로 서비스를 할 수 있다.

현재 RTP는 IETF(Internet Engineering Task Force)의 RFC1889로 표준화된 상태이다 또한 RTP는 실시간 streaming을 위해 사용된다. 비디오 packet들은 실시간으로 전송되어서, 1분짜리 비디오는 1분 안에 실시간 전송된다. packet들에 타임-스탬프(time-stamp)가 찍히므로, packet들은 시간에 동기화되어 재생될 수 있다. packet들이 실시간으로 전송되기 때문에, RTP streaming은 저장된 컨텐츠(stored contents) 뿐만 아니라 라이브 컨텐츠(live contents)도 지원한다[2].

RTP는 통신하는데 일정 시간 내에 전송되어야 할 필요성이 있는 실시간 멀티미디어 정보를 효율적으로 보내기 위한 전송프로토콜이지만 소프트웨어적으로 구현되는 응용전송 프로토콜이다.

본 논문에서는 단말기내의 RTP/JPEG Packet에 대하여 Packetizing 과 Depacketizing에 대한 알고리즘을 설계함으로써 WIPI 플랫폼 상에서 Streaming Service를 제공할 수 있는 Cappuccino Bubble 시스템을 구현하였다. 인코드된 Jpeg packet은 전송되기 전에 이 RTP프로토콜에 맞

게 변경된다. 또한 전체적인 시스템의 Memory Size를 WIPI 에뮬레이터의 Memory heap size 1024K에서 실행 메모리 300K로 최적화 시켰다.

2. 관련연구

본 논문에서 구현한 Cappuccino Bubble에 해당 하는 기술인 J2ME의 MMAPI와 JMF에서 지원되는 멀티미디어 스트리밍 서비스를 비교해 볼 수 있다. J2ME의 MMAPI에서는 Mpeg1에 대한 Codec과 RTP프로토콜에 대하여 Client 기능에 한하여 지원되고 있다. JMF는 여러 미디어 포멧에 대한 Codec과 Streaming Protocol에 대한 추상화된 Class들을 API에서 제공하여 멀티미디어 Application 개발이 용이하다.

	J2ME	JMF	Cappuccino
지원되는 Codec	Mpeg1	Mpeg1 Jpeg AVI MOV	Jpeg
지원되는 Protocol	RTP	RTP RTSP RTCP	RTP

2.1 J2ME

J2ME의 경우 코덱과 프로토콜은 MIDP2.0 MMAPI에서 스펙으로 정의하고 있지만 추상화된 클래스로는 API에서 지원되지 않는다 [3]. 또한 현재 나온 LG ez-java, SK XCE 에뮬레이터에서는 MIDP 1.0만이 지원되며 차세대 무선 Mobile 플랫폼인 WIPI에서도 아직까지 RTP지원이 되지 않고 있다.

2.2 JMF

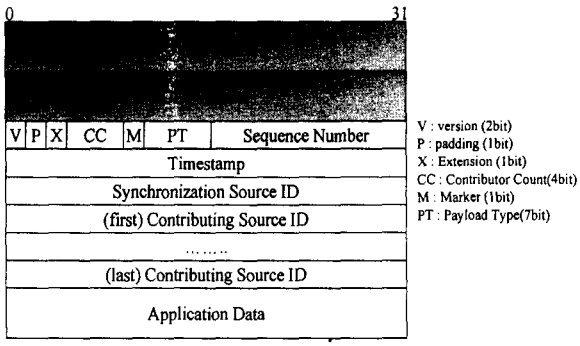
현재 JMF는 최신 버전이 2.1.1e까지 나와있다. 다양한 미디어 포멧을 지원하며 여러 프로토콜까지 지원하고 있다. 또한 오디오 캡쳐, 파일 스트리밍 전송, 캠 기능등 강력한 멀티미디어 기능을 갖추고 있다. JMF API를 이용하여 Mobile 스트리밍 서버 구현도 가능하다. 하지만 현재 지원되는 Mobile 에뮬레이터 메모리 heap size에 올리기에는 그 용량이 너무 크다. JMF 소스의 용량은 11.7M이다 [4]. 현재 XCE의 에뮬레이터 메모리 heap size가 2M이고, WIPI는 1024K이다. 그러므로 Mobile 위에서 스트리밍 서버를 구현하기 위해선 파일 포멧에 대한 경량화된 코덱과 RTP구현이 직접 이루어 져야 한다.

2.3 Cappuccino Bubble

Cappuccino Bubble은 JMF가 지원하는 JPEG 에 대한 RTPDePacketizer를 직접 구현함으로써 메모리 크기에 따른 제한을 기술적으로 해결하였다. 또한 JMF의 API를 사용하지 않고 직접 Protocol구현하여 포팅 함으로써 WIPI 에뮬레이터의 메모리 heap size에 최적화 시켰다.

3. Cappuccino Depacketizer Algorithm Design

Cappuccino Bubble의 RTP/JPEG 스트림에 대한 RTP Protocol 과 Depacketizing 알고리즘 구현 내용은 다음과 같다.



[그림1] RTP fixed header fields

위 그림에서 보는 바와 같이 RTP header의 정보를 사용하여 프레임을 구성하게 된다. Depacketizing에서 주로 사용하는 header 필드의 정보는 MARKER, Sequence Number, Timestamp값들이다. 또한 Jpeg encoding된 데이터를 Jpeg 디코더로 넘겨주기 전에 JPEG File Interchange Format(JFIF)으로 변환시켜야 한다[5]. 그러기 위해선 Jpeg header의 정보를 이용하여 JFIF를 구성해야 한다. Depacketizer Algorithm은 우선 RTP 패킷이 정상으로 들어오는지에 대한 Validation Check를 한 후에 JPEGFrame이라는 버퍼를 생성하여 먼저 JPEGFrame에 들어온 패킷의 타임스탬프 값과 현재 RTP패킷의 타임스탬프 값을 비교하여 전자가 클 경우 현재 패킷은 버려버리고 후자가 클 경우 JPEGFrame 전체를 비운 후 다시 채우기를 시작한다. 이 과정을 요약하면 다음과 같다.

```
JPEGFrame = 하나의 JPEGFrame을 이루는 packet들
            의 저장소
startSeqNum = 시작 시퀀스 넘버
EndSeqNum = 마지막 시퀀스 넘버

If (JPEGFrame == null){
 startSeqNum, EndSeqNum, Received 값들 초기화시
킴.
 }else{
 If (JPEGFrame의 타임스탬프 값이 현재 RTP 패킷의
타임스탬프 값보다 크면){
 // 현재 들어오는 RTP 패킷을 버려버린다.
 }Else if (JPEGFrame의 타임스탬프 값이 현재 RTP
패킷의 타임스탬프 값보다 작으면){
 // 현재 JPEGFrame을 버리고 새로운 JPEGFrame을
 생성// 한다.
 }
```

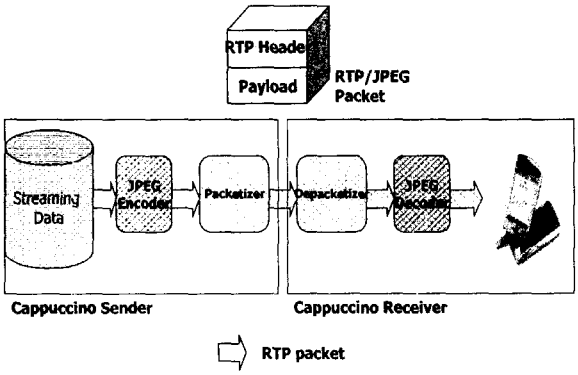
```
If (JPEG헤더의 FragmentOffset값이 0이면){
StartSeqNum은 현재 패킷의 시퀀스넘버이다.
}
If(현재 패킷이 MakerBit를 가지고 있으면){
EndSeqNum은 현재 패킷의 시퀀스넘버이다.
}
Received++;
```

[그림2] Depacketizer Algorithm

4 " Cappuccino Bubble" 시스템 Architecture 및 구현

4.1 Cappuccino Bubble 시스템 구조

Cappuccino Bubble이 스트리밍 서버의 기능을 갖추기 위해선 Sender의 기능과 Receiver 두 가지 기능을 다 갖고 있어야 한다. 또한 JPEG Encoder와 Decoder가 부속 모듈로써 존재해야 한다. 앞에서는 Depacketizing에 대한 알고리즘을 언급하였고 Jpeg Decoder만이 Cappuccino Bubble안에 존재하며 이에 대응하는 프레임에 대한 Packetizing 역시 유사한 방법으로 구현할 수 있다.



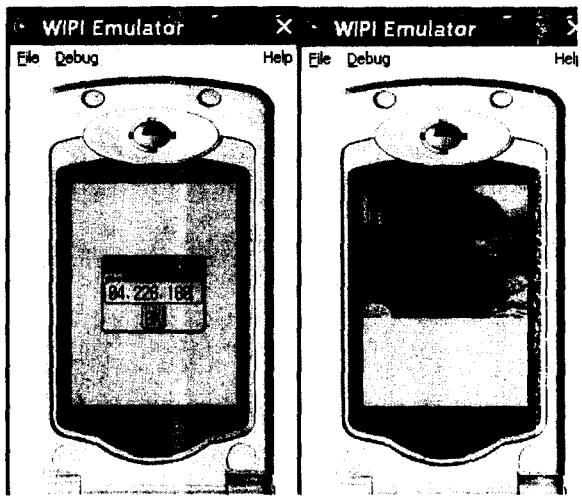
[그림3] Cappuccino Bubble의 시스템 Architecture

위의 [그림3]에서 볼 수 있는 것처럼 Cappuccino Bubble의 시스템 구조는 크게 Sender부분과 Receiver부분으로 나뉘게 된다. Sender부분에서는 스트리밍 데이터 Jpeg encoder로 encoding된 frame을 RTP크기에 맞도록 Packetizing하는 과정을 거치게 된다. 그 다음엔 RTP/JPEG Packet들이 수신부로 전해지게 된다. 수신부로 전해진 Pakcet들은 Depacketizing과정을 거친 후 JPEG Decoder로 넘겨지고 Decoding된 영상이 WIPI Emulator에서 출력된다. WIPI card위에 출력되는 이미지의 폭과 넓이는 128, 96으로 설정하여 나타나게 하였다.

4.2 Cappuccino Bubble 구현

시스템의 구현 환경은 다음과 같다. OS는 Windows XP를, 사용언어는 java로 오픈 개발 프로젝트인 Eclipse를 기반으로 하였고 코덱은 C로 구현된 JPEG Decoder를 사용하였다. 아직까지 Mobile P2P 멀티미디어 서비스를 위한 Streaming 서버는 구현하지 못하였고 클라이어트 기능만 구현된 상태이다.

테스트는 서버로 JMF를 사용하여 캠으로부터 영상을 받아와 JPEG 포멧으로 변환한 RTP 스트림을 UDP 소켓을 열어 WIPI 에뮬레이터에서 작동하게 해 보았다. 아래 [그림5]는 JMF서버가 설치된 PC에서 캠으로부터 영상을 받아와 출력하는 실행화면이다.



[그림4] Cappuccino Bubble실행화면

5. 결론 및 향후 과제

본 논문에서는 모바일 단말기에서 P2P 기반의 스트리밍 서비스를 제공하기 위한 첫단계로서 경량 스트리밍 서버인 Cappuccino Bubble의 구현 내용을 소개하였다. 메모리 요구사항을 최소화하기 위해 J2ME 또는 JMF에서 제공되는 서비스를 사용하지 않고 직접 RTP 프로토콜을 Jpeg 형식의 자료를 packetizing/depacketizing하여 전송하는 방법을 사용하였다. 기본적인 경량화된 스트리밍 서버를 구축하였으므로 향후에는 이러한 모바일 단말기들이 P2P 네트워크를 구성하여 복수 단말기간의 멀티미디어 서비스를 제공할 수 있는 기반을 구축하여 전체 셀룰러 통신망에 부하를 주지 않는 시스템을 구축할 수 있을 것이다.

6. 참고 문헌

[1] http://www.kwisa.org/
[2] http://www.ietf.org/rfc/rfc1889.txt
[3] http://www.java.sun.com/j2me
[4] http://java.sun.com/products/java-media/jmf/
[5] http://www.faqs.org/rfcs/rfc2035.html