2000년 한국정보처리학회 추계 학술발표논문집 제7권 제2호

스트리밍 동영상을 위한 자동속도설정에 관한 연구

이종찬*, 김응곤**
*순천청암대학 컴퓨터정보과학부
**순천대학교 컴퓨터과학과
e-mail : jclee@scjc.ac.kr

A Study on Automatic Speed Configuration for Streaming Movie

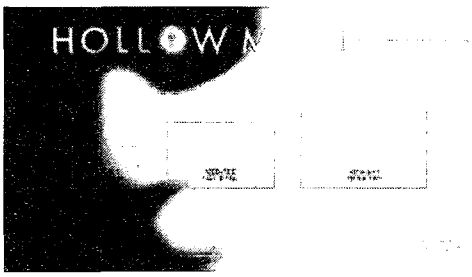
Jong-Chan Lee*, Eung-Kon Kim**
*School of Computer Infomation Science, SunCheon CheongAm College
**Department of Computer Science, SunChon National University

요 약

본 논문에서는 최근 활성화되고 있는 인터넷방송의 스트리밍 영상물 제공 방법을 개선하여, 자동으로 사용자의 연결속도에 적합한 스트리밍 영상물을 선택 제공하는 방법을 구현하고자 한다. 대부분의 스트리밍 영상물을 제공하는 방법은 각 속도에 적합한 영상물을 각각 모두 나열하여 사용자가 선택, 제공받는 방법을 택하고 있지만 이는 사용자의 선택이 잘못된 경우 연결속도에 따른 최적의 영상물을 제공받을 수 없는 단점을 지니고 있다. 따라서, 본 논문에서는 사용자의 연결속도에 따라서 자동적으로 최적의 스트리밍 영상물을 제공할 수 있는 방법을 제시하고자 한다.

1. 서론

대부분의 인터넷방송 사용자는 사용하는 시스템과 사용하는 인터넷 네트워크 환경에 따라서 인터넷 방송을 시청하기 위한 스트리밍 서버와 사용자의 컴퓨터간의 네트워크 연결 속도를 사용자 스스로가 파악하여야만 원하는 영상물을 시청할 수 있다. 이러한 방법은 그림1에서와 같이 사용자가 원하는 영상물을 인터넷방송국에서 미리 사용자의 속도를 예상하여 이에 적합한 각각의 영상물을 제작하여, 웹 페이지상에 접속 가능한 link를 나열하여 사용자가 선택하도록 유도하고 있다. 일반적으로, 14.4K, 28.8K 56K, 128K, 256K, 512K 등으로 접속 가능 속도를 설정한 후에 이에 적합한 동영상을 각각 제작하게 된다. 하지만 이러한 방식은 사용자가 자신이 연결된 네트워크에 대한 정보의 부재로 잘못 선택한 경우 네트워크 연결 속도에 알맞은 최적의 영상물을 시청하지 못하는 단점이 있다. 이러한 단점을 제거하여 사용자가 28.8K 모뎀 사용자일 경우는 이에 알맞은 저화질의 가장 작은 크기의 영상물을 자동적으로 전송할 수 있으며, T1 전용선 사용자일 경우 고속 연결에 알맞은 보다 좋은 화질과 큰 크기의 영상물을 시청할 수 있도록 자동 선택적인 영상물 전송을 이룰 수 있다면 이는 사용자에게 보다 편리한 인터넷방송 시청 환경을 제공할 수 있을 것이다



<그림 1> 3개의 link로 복잡한 인터페이스

출처: http://www.apple.com/trailers/columbia/hollow_man/

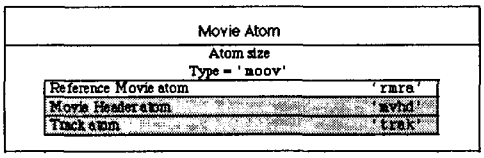
2. 인터넷방송 현황

현재 인터넷방송에서 사용중인 스트리밍 기술은 대부분 RealServer, Window MediaServer, Stream

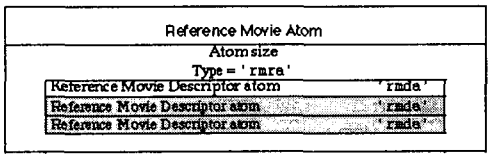
Works과 QuickTime Streaming Server등이 있으며, 현재는 Window의 보급이 확산되면서 RealSystem에서 WindowMediaServer와 Apple사의 QuickTime 사용자가 늘어가고 있는 추세이다. 따라서, 본 연구는 고급의 화질을 전송하며 사용자가 많고 Windows와 Macintosh상에서 공동으로 지원이 가능한 QuickTime을 기반으로 구현하고자 한다. 또한, Apple사는 인터넷상에 QuickTime API를 개발자에게 제공하고 있어 다른 고가의 스트리밍 미디어와 다르게 손쉽게 접근할 수 있는 장점도 있다.

3. 구현 및 결과

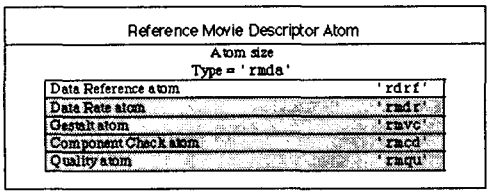
자동속도설정 스트리밍 영상물은 QuickTime에서 정의된 QuickTime Movie File Format을 이용하여 새로운 atom을 형성하여 구현하였다. 최상위의 Movie atom은 상황에 따라 선택할 수 있는 정보를 담은 Reference Movie atom에 대한 정보를 보유하게 된다.



<그림2> Movie Atom의 구조



<그림3> Reference Movie Atom의 구조



<그림4> Reference Movie Descriptor Atom의 구조

또한 Reference Movie atom은 간단하게 개별적 Refernce Movie Descriptors에 대한 정보를 보유하게 되며, Reference Movie Descriptor atom은 적어도 하나의 Data Reference atom의 정보를 가지게 되거나, 다른 설정에 필요한 atom들의 정보를 추가로 보유할 수 있게 설계하였다.

앞에서 설명되었듯이 사용자의 Reference Movie Descriptor atom은 최소한 하나의 Data Refernce atom을 보유하여야만 하는데, 이를 처리하기 위한 방법을 다음과 구현하였다.

```
enum {
    ReferenceMovieDataRefAID =
FOUR_CHAR_CODE('rdrf'),
};

struct ReferenceMovieDataRefRecord {
    long            flags;
    OSType          dataRefType;
    long            dataRefSize;
    char            dataRef[ 1] ;
};
typedef struct
ReferenceMovieDataRefRecord
ReferenceMovieDataRefRecord;

enum {
    kDataRefIsSelfContained = (1 << 0)
};
```

또한 대부분의 일반적 연결 속도에서 선택되어질 선택항목을 처리하기 위한 방법을 다음과 구현하였다.

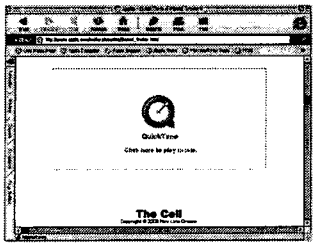
```
enum {
    ReferenceMovieDataRateAID   =
FOUR_CHAR_CODE('rmdr'),
};

enum {
    kDataRate144ModemRate     = 1400,
    kDataRate288ModemRate     = 2800,
    kDataRateISDNRate         = 5600,
    kDataRateDualISDNRate     = 11200,
    kDataRateT1Rate           = 150000L,
    kDataRateInfiniteRate     = 0x7FFFFFFF
};

struct QTAltDataRateRecord {
    long                    flags;
/* currently always 0 */
    long                    dataRate;
};
typedef struct QTAltDataRateRecord
QTAltDataRateRecord;
```

결과적으로 자동속도설정이 가능한 스트리밍 영상물을 제작은 다음과 같은 방법으로 기존의 영상물을 가공하여 웹 페이지에 연결함으로써 자동속도설정이 가능하게 구현하였다.

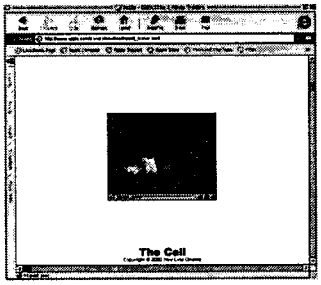
```
short outFref;
long atom[ 2] ;
Ptr foo;
long count, refMovieAtomSize;

// create and open output file
err = FSpCreate(&outputFSSpec, 'TVOD',
MovieFileType, 0);
if (err) goto bail;
err = FSpOpenDF(&outputFSSpec, fsRdWrPerm,
&outFref);
refMovieAtomSize = count =
GetHandleSize(refMovieAtomH);
foo = NewPtrClear(refMovieAtomSize);
if ((err = MemError()) != noErr) goto
bail;
*(long *)foo = refMovieAtomSize;
 *(long *)(foo + sizeof(long)) =
FreeAtomType;
FSWrite(outFref, &count, foo);
DisposePtr(foo);
FSClose(outFref);

// Flatten movie to output file
 newMovie = FlattenMovieData(origMovie,
          flattenAddMovieToDataFork |
flattenForceMovieResourceBeforeMovieData,
          &outputFSSpec, 'TVOD', -1, 0);
err = GetMoviesError();
if (err) goto bail;

// Open output file again
// Read the Movie atom
err = FSpOpenDF(&outputFSSpec, fsRdWrPerm,
&outFref);
if (err) goto bail;
SetFPos(outFref, fsFromStart,
refMovieAtomSize);
```

```
count = 8;
err = FSRead(outFref, &count, &(atom[ 0] ));
if (err) goto bail;
if (atom[ 1] != MovieAID) {
    err = paramErr;
    goto bail;
}
foo = NewPtr(refMovieAtomSize + atom[ 0] );
if ( (err = MemError()) != noErr) goto
bail;

// Merge the Movie atom
*(long *)foo = refMovieAtomSize + atom[ 0] ;
*(long *)(foo + sizeof(long)) = MovieAID;

// copy Reference Movie atom
BlockMoveData(*refMovieAtomH, foo + 2 *
sizeof(long), refMovieAtomSize);

// read original Movie atom
count = atom[ 0] - (sizeof(long) * 2);
err = FSRead(outFref, &count, foo + 2 *
sizeof(long) + refMovieAtomSize);
if (err) goto bail;

// Write final Movie atom to disk
SetFPos(outFref, fsFromStart, 0);
count = refMovieAtomSize + atom[ 0] ;
FSWrite(outFref, &count, foo);
DisposePtr(foo);
FSClose(outFref);
```

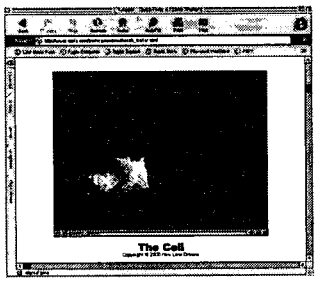
구현된 방법으로 자동속도설정이 가능한 스트리밍 동영상을 제공하는 웹 페이지로 그 결과를 다음과 같이 확인할 수 있다. 그림1에서 3개의 link로 설정되어져 사용자의 혼란스러웠던 점과는 다르게 그림5는 오직 하나만의 link로 간편해진 사용자의 접근 환경을 제공하고 있으며, 이 영상물을 각기 다른 환경의 접속 속도를 가진 사용자가 요청을 하였을 때, 각자에게 알맞은 최적의 크기와 화질을 제공함을 다음과 같이 확인할 수 있다.



<그림5> 1개의 link로 간편한 인터페이스
출처 : http://www.apple.com/trailers
/newline/thecell_trailer.html



<그림6> 56K모뎀 접속 화면
출처 : http://www.apple.com/trailers
/newline/thecell_trailer.html



<그림7> T1 전용선 접속 화면
출처 : http://www.apple.com/trailers
/newline/thecell_trailer.html

4. 결론

본 논문에서는 최근 활성화되고 있는 인터넷방송의 스트리밍 영상물 제공방법을 개선하여, 사용자의 연결속도에 적합한 스트리밍 영상물을 자동으로 선택 제공하는 방법을 구현하였다.

다양한 문화와 개성의 집합체인 인터넷 환경에서 사용자의 편의성을 제공하는 많은 연구가 진행되어져야만 인터넷을 항해하는 많은 사용자들이 컴퓨터와 네트워크에 대한 지식의 부재로 인한 최상의 서비스를 받지 못하는 단점을 극복할 수 있을 것이다. 본 논문의 지속적인 연구는 QuickTime에 Component Check, Gestalt, Quality atom을 추가로 활용할 수 있도록 설계되어 있어, QuickTime VR과 같이 QuickTime이 지원하는 다른 종류의 미디어를 지원할 수 있으며, Intel Indeo와 같이 다른 종류의 codec을 확인하여 지원할 수 있는 확장성을 가지고 있어 추가적인 개발에 활용 가능한 점을 가지고 있다. 향후 이에 대한 연구가 계속 진행된다면 보다 사용자에게 친절한 접근 방법과 사용자의 연결 속도에 따른 자동속도 설정만이 아닌 사용자의 컴퓨터의 종류, 접속하려는 codec 또는 미디어의 종류 등에 따라서 자동으로 최적의 영상물을 제공 가능하게 될 것이다.

감사의 글

본 연구는 광주·전남 테크노파크 기술고도화사업의 지원에 의한 것입니다.

참고문헌

[1] George Towner, "Discovering QuickTime", Morgan Kaufmann, 1999.

[2] Tom Maremaa and William Stewart, "QuickTime for Java", Morgan Kaufmann, 1999.

[3] Robert Hone, "QuickTime", 2nd Ed., Prima Publishing, 1995.

[4] Judith L. Stern and Robert A. Lettieri, "QuickTime", Hayden Books, 1994.

[5] "http://www.apple.com/trailers/newline/thecell_trailer.html"

[6] "http://www.apple.com/trailers/columbia/hollow_man"

[7] "http://hrstv.com/tvmain.htm"

[8] "http://www.castservice.com/about_webcast3.html"