안정적인 재생을 위한 Local Streaming Server 개발

Naver Campus HackDay
강우진

목차

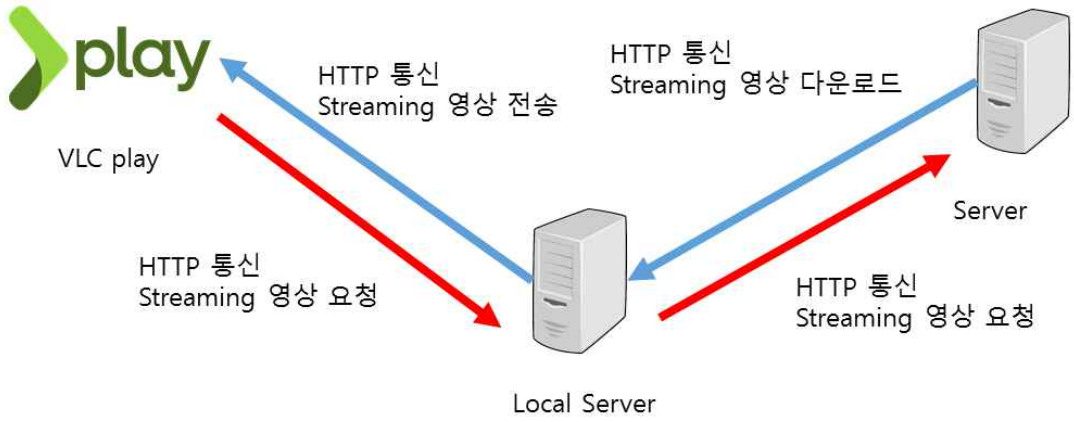
1. 개발목표

- 모바일 네트워크는 다양한 속도 품질 및 잦은 네트워크 오류로 불안정하다. 이러한 네트워크 환경에서 최대한 안정적인 동영상 재생을 지원하고자 한다.
- 플레이어 <-> Local Streaming Server <-> CDN(동영상 파일) 구조를 갖는 스트리밍 서버에서 제공하는 다양한 기능을 통해 안정적인 재생을 지원한다.

2. 요구사항

- Local Streaming Server는 기본적인 Proxy 기능을 제공해야 한다.
- Local Streaming Server는 안정적인 재생을 위한 다양한 기능을 제공해야 한다.
- Pre-fetch, Cache, Network(timeout, retry, fail-over)
- 플레이어에서 Local Streaming Server를 통해 재생 후 initial time, seeking time, buffering count등 재생 통계 정보를 제공해야 한다.

2.1 시스템 구조도



2.2 기능 명세

기능명	상세 내용	비고
파일 다운로드	Server에서 Local Server로 사용자가 요청한 영상을 다운로드 하여 메모리에 보관한다.	
파일 전송	사용자가 요청한 영상을 VLC Play로 전송한다.	
버퍼 관리	다양한 영상의 크기로부터 버퍼를 처리 해야한다.	
캐시 관리	Procy를 통해 Server로부터 현재 영상을 받아왔는지 확인한다.	
Seek 이동	사용자는 원하는 영상의 위치로 이동하여 영상을 제공받을 수 있다.	

3. 설계

3.1 함수명세

함수명	상세 내용	비고
LPWSTR ptrCharToLPWSTR(char* pszUrl)	char*를 LPWSTR형으로 변환한다.	
int getFileFromHttp(char* pszUrl, char* pszFile)	Server로에서 Local Server로 파일을 얻어온다.	
void sendFile()	사용자로부터 요청을 받아 헤더를 분석한 뒤 Local Server에서 play로 파일을 전송한다.	
string* StringSplit(string strTarget, string strTok, int& nIndex)	http protocol Header를 분리한다.	
void run()	Socket init후 receive상태	
void init()	많은 사용자로부터 멀티스레드 환경의 Socket 통신을 하기 위해 socket 생성, bind, listen, accept 한다.	
void getClientReceive()	사용자로부터 응답을 받아온다.	
bool isFile(string fileName)	현재 메모리에 파일이 존재하는지 확인한다.	
int fileSendOk()	버퍼에 일정량의 데이터가 쌓여 play에게 데이터를 전송할 수 있는 상태인지 확인한다.	
void memoryManager()	Local Server에 File 메모리를 관리한다.	

3.2 함수 상세내용

함수명	LPWSTR ptrCharToLPWSTR(char* pszUrl)

```
LPWSTR ptrCharToLPWSTR(char* pszUrl)
{
        /*char* to LPWSTR */
        wchar_t wtext[1024];
        mbstowcs(wtext, pszUrl, strlen(pszUrl) + 1);//Plus null
        LPWSTR url = wtext;

        return url;

}
```

함수명	int fileSendOk()

```
int fileSendOk(int clientNum)
{
  //버퍼 관리

  //간단한 Procy 패턴을 적용하기 위해 File을 메모리에 저장한다.
  if ( isFile(clientNum) )
    return 1 ; // 이미 파일을 Local Server에 갖고 있기에 사용자에게 바로 전송
  if ( fileSize == mp4의 헤더와 트레일러부분이 되면)
    return 1;
  else if ( buff == 약 10초 재생의 크기만큼 ) // 약 10초 재생의 크기 만큼
    return 1 ;

}
```

함수명	void memoryManager()

```
int memoryManager()
{
  1. File의 배열이 max에 도달하였을 때 발동하는 함수
  2. 현재 File 기존 파일 배열 중 덮어쓸 배열을 선택한다.
  3. 영상을 시청하지 않는 FileManager의 file 영상 요청이 오래된 순 같을 경우 파일
     Size가 작은 경우 순으로 File 배열 교체
  4. File 배열의 전체가 영상 중인 경우 제한 -> singleton 패턴 적용
  // 소켓의 멀티스레드를 생성할 때 개수 제한

  //File 배열 관리
  //사용자로부터 파일을 요청 받아 Local Server에 저장하였을 때
  //LRU 페이지 교체  (Least-lecently-used Page Algorithm)에 따른 페이지 교체
  // Map<FileManager>  // Socket Number를 통해 File의 상세정보를 저장
  //struct FileManager
  {
    사용자의 영상 요청 시간정보 ;
    FileSize크기 ;
    Fileindex ;
    isPlay ;
    sort() // 1순위 file영상 요청 시간정보이 가장 나중인 순으로 오름차순
           // 2순위 파일 Size가 작은 순으로 오름차순

  }
  //로 구성된다.
}
```

함수명	bool isFile(int socketNum)

bool isFile(int socketNum)

{

// 영상은 데이터가 큰 파일이기 때문에 파일을 한번 받아 저장해두는게 효율이 좋음

// Procy를 패턴을 적용하기 위해 File *files[m]개를 미리 만들어

// 간단한 Procy Server를 구현한다.

1. FileManager를 저장한 Map에 socketNum을 통해 fileSize, fileIndex를 가져온다.
2. fileIndex를 통해 해당 File의 filesize가 null인지 체크한다.
3. null인 경우 false, null이 아닌 경우 true 리턴

}

함수명	void sendFile()

```
void sendFile()
{
/*Receive 부분*/
//사용자로부터 데이터를 받아 http header를 split한다.
//



/*영상 파일이 존재하지 않는 경우 */
1. 영상 Server로부터 데이터를 받아 buffer에 쌓아둔 뒤 전송해 안정적인 재생을 제공한다.
2. 어느정도의 영상이 메모리에 존재한다면 사용자에게 전송하면서 영상 Server에서 파일을 가져온다.
/*Seeking 부분*/
//header에 range부분에 숫자부분을 파싱한다.
//영상 Server에서 range부분에서부터 파일을 읽어온다.

/*파일이 존재하는 경우 - Seeking 부분*/
//사용자로부터 받은 http header중 range부분에 숫자부분을 파싱한다.
//파싱한 숫자는 해당 파일 seek_cur을 start로 설정한다.
// end = start + buffsize -1 => 1024씩 보냄 배열은 0부터기 때문에 -1을 해줌
//파일을 path경로로부터 file을 읽을 때 start부터 읽어와 버퍼에 저장한다.

char *sendData = "HTTP/1.1 206 Partial Content\r\n
Last-Modified: Wed, 15 Nov 1995 04:58:08 GMT\r\n
Content-Range: bytes start -end / fileSize \r\n
Content-Length: end - start +1\r\n
Content-Type: video/mp4\r\n\r\n\r\n
Accept-Ranges: bytes ";

// 206 Partial은 파일을 이어보낼 때 사용
// Last-Modified를 기준으로 뒤에 이어 붙임
// Content-Range의 크기를 보낼때마다 크기를 변경해줘야 한다.
// Content-Length는 보내는 양
// Accept-Ranges가 존재하면 브라우저는 처음부터 다시 다운로드를 시작하지 않고, 중단된 다운로드를 재개하려고 한다.



//
for ( I = start  ; I < end; I+=buffSize, fread(경로,buff,buffsize))
{

  iSendResult = send(ClientSocket, strcat(sendData,buff), strlen(buff),0);

}
if ( file의 마지막인 경우 )
{
```

4. 구현

기능명	상세 내용	구현
파일 다운로드	Server에서 Local Server로 사용자가 요청한 영상을 다운로드 하여 메모리에 보관한다.	
파일 전송	사용자가 요청한 영상을 VLC Play로 전송한다.	
버퍼 관리	다양한 영상의 크기로부터 버퍼를 처리 해야한다.	
캐시 관리	Procy를 통해 Server로부터 현재 영상을 받아왔는지 확인한다.	
Seek 이동	사용자는 원하는 영상의 위치로 이동하여 영상을 제공받을 수 있다.	