미디어 전송을 위한 분산형 스트리밍 프로토콜 설계 및 구현

*정태준, *이홍래, *서광덕
*연세대학교 컴퓨터정보통신공학부
*jeung86@naver.com

Design and Implementation of a Distributed Streaming Protocol for Media Transmission

*Jung Tae Jun, *Hong-rae Lee, *Kwang-Deok Seo
*Computer and Telecommunications Engineering Division, Yonsei University

요약

스트리밍 전송을 위한 P2P 구조는 크게 메쉬구조와 트리 구조로 구분된다. 실시간 스트리밍에 유리한 트리 구조의 단점을 보완하기 위해 많은 연구가 진행되고 있다. 대표적으로 두개 이상의 트리 구조를 유지하는 다중 트리구조와 사용자들 중에서 안정적인 사용자를 트리의 중심에 위치시키는 백본 트리 구조가 있다. 다중 트리 구조는 단일트리의 단점을 극복하기 위해 트리와 메쉬 구조를 혼용한 구조로 안정적인 반면, 사용자들의 출입이 잦아지면 구조를 유지하기가 어려워지는 단점이 있다. 백본 트리 구조는 안정적인 사용자들로 트리의 중심 빼대를 구축한 후 나머지 사용자를 뼈대의 가지에 위치시키는 방법이다. 본 논문에서는 새롭게 설계한 분산형 스트리밍 전송기술을 위한 프로토콜의 구조에 대해서 설명하고, 제안된 프로토콜을 시뮬레이션을 통해 P2P 기반의 스트리밍 서비스의 가능성을 본다.

1. 서론

P2P 기술을 이용한 파일 다운로딩 서비스와 스트리밍 서비스는 가장 인기 있는 인터넷 어플리케이션이다. P2P 네트워크는 다른 인터넷 어플리케이션보다 훨씬 더 많은 트래픽을 발생시키고 어떤 백본에서의 모든 대역폭의 2/3를 차지한다. 그림 1 은 인터넷 어플리케이션에 의해 발생하는 트래픽의 연도별 추이를 나타낸다.

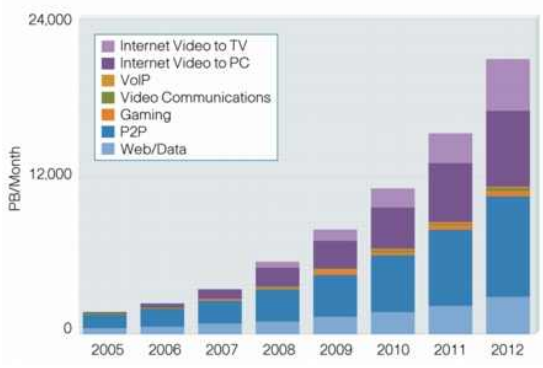

그림 1. 인터넷 어플리케이션에 의해 발생하는 트래픽의 연도별 추이

P2P의 장점은 모든 클라이언트들이 대역폭을 포함하여 시스템의 자원과 저장공간, 계산능력 등을 제공한다는 것이다. 따라서 노드가 추가될 때마다 시스템의 규모가 커지고, 또한 전체적인 시스템의 서비스 제공 능력도 증가하게 된다. P2P 분산 네트워크는 여러 피어들의 데이터를 복제하기 때문에 결함발생이 줄어든다. 그리고 사용자 상호간에 데이터를 공유하기 때문에 서버의 부하가 확실하게 줄어든다.

P2P 시스템은 많은 다른 종류의 서비스를 제공한다. P2P file downloading 서비스(BitTorrent, Emule), Voice Over Internet Protocol(Skype), P2P live streaming(Coolstreaming, PPLive streaming), P2P video-on-demand서비스(P2P-VOD)가 있다 [1][2].

본 논문에서는 P2P기반 실시간 스트리밍 프로토콜의 새로운 모델인 분산형 스트리밍 전송기술을 위한 프로토콜의 설계 하고 NS-2 시뮬레이션 프로그램을 통해서 P2P 스트리밍 서비스의 가능성을 확인한다.

2. 분산형 스트리밍 전송을 위한 프로토콜 동작 구조

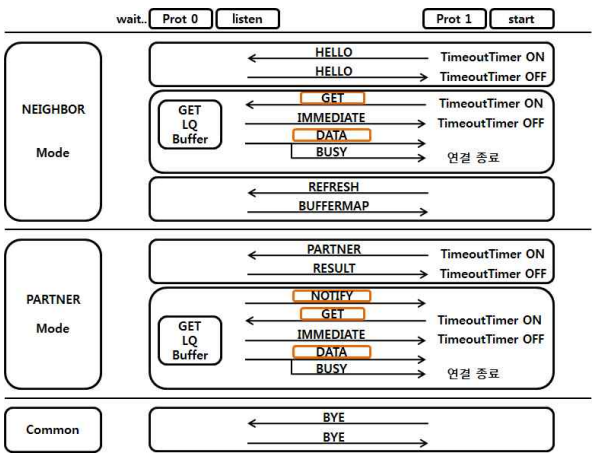

그림 2. 분산형 스트리밍 전송을 위한 프로토콜 동작 구조

그림 2 는 분산형 스트리밍 전송을 위한 프로토콜의 기본적인 동작 구조에 대해서 설명하고 있다. 임의의 두 프로토콜 Agent간 관계는 Neighbor 와 Partner로 구분되며, 관계에 따라서 프로토콜 Agent의 동작이 달라진다. Neighbor 관계의 두 Agent는 동등한 관계로 서로 필요한 데이터를 주고받을 수 있다. Partner 관계는 둘 중 하나의 Agent가 일방적으로 데이터를 보내주는 관계이다.

3. 분산형 스트리밍 프로토콜 설계

3.1 분산형 스트리밍 프로토콜 메시지 구조

메시지 헤더는 프로토콜 Agent간 교환하는 모든 메시지에 사용되고 구조는 다음 그림 3 과 같다.

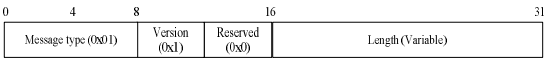

그림 3. 분산형 스트리밍 전송을 위한 프로토콜 헤더 구조

Message type 은 어떠한 메시지인지 메시지 종류에 대한 정보를 담고 있다. Version 은 버전정보를 포함하고 있고, Reserved 필드의 경우는 예약 필드로서 차후에 어떻게 사용될지 결정되지 않은 필드이다. Length 는 메시지 길이에 대한 정보를 담고 있다.

Buffer map 은 프로토콜 Agent 간에 데이터를 공유하기 위해서 각 Agent 가 보유하고 있는 데이터의 상황을 알리기 위해 필요한 메시지이다. Buffer map은 STB와 SHB로 나뉘어진다.

STB는 Stored buffer로 현재 송수신 중인 버퍼구간 (SHB)에 속하지는 않지만 이미 지나간 구간에 대한 공유를 위해서 보유하고 있는 구간을 말한다. SHB는 Sharing buffer로 현재 송수신 중인 구간으로 재생을 위한 구간이다 특히, 원활한 재생을 위해서 Playback index를 포함한 Urgent section은 먼저 수신한다. 그림 4 는 STB와 SHB 그리고 Urgent section을 보여주고 있다.

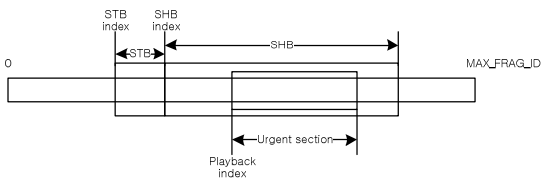

그림 4. Buffermap 구조

STB index는 STB의 시작 index를 뜻한다. STB length는 STB의 길이를 뜻하고 있고, SHB index는 SHB의 시작 index를 뜻한다. SHB length는 SHB 버퍼의 길이정보를 담고 있다. SHB map의 경우는 SHB에 대한 buffermap 정보를 담고 있고 필드길이는 buffermap 정보에 따라서 유동적으로 바뀌게 된다 [3].

3.2 분산형 스트리밍 프로토콜 타이머 작동 원리

현재 NS2 시뮬레이션 환경은 multi thread가 돌아가지 않기 때문에 Timer를 사용함으로써 thread 처럼 구동이 가능하게 할 수 있다. 타이머의 작동 원리는 그림 5와 같이 메시지가 전송되면서 Timer가 동작한다. 일정시간이 지나도 응답이 없으면 Expire() 함수를 호출하게 되고 Expire() 함수가 호출되고 나면 정의된 함수가 호출되게 되고 Timeout()함수가 호출된다. Timeout()함수 호출 후에는 다음 Timer 동작시간이 설정된다.

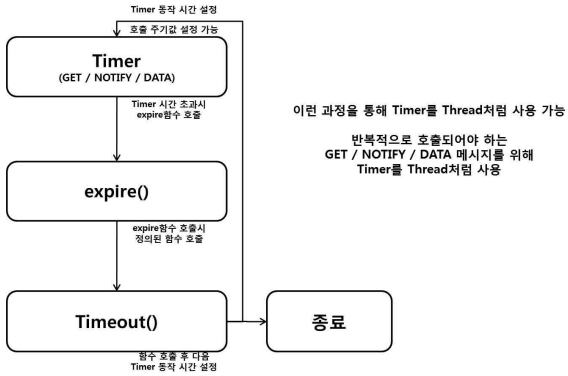

그림 5. Timer 함수의 작동원리

현재 타이머는 GET, NOTIFY, DATA, Playback Timer 이렇게 4가지 타이머가 돌아가고 있다. 각 타이머 별로 살펴보면 GET Timer의 경우에는 Timer를 Thread 처럼 사용해서 일정주기마다 호출하게 설정하였다. 그리고 Get Timer는 Neighbor mode 일때만 동작하도록 설정되어있다.

3.3 분산형 스트리밍 프로토콜 Media file handle/ file 저장방법

프로토콜을 실행했을시 초기 file 관련 동작은 Media file이 열리면서 동시에 file size를 계산하며 File point를 연결한다. File pointer에 연결후에 buffermap 크기만큼 파일에 기록하게되고 file size를 계산한 후에는 fragment 개수를 계산한다.

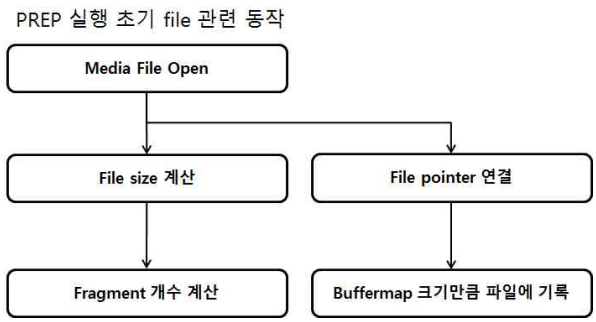

그림 10. 분산형 스트리밍 프로토콜 실행초기 file 관련 동작

이후 DATA메시지 송신시에는 전송할 fragment index와 block offset 정보를 획득한후에 Media 파일의 해당위치로 File pointer를 이동시키고 그 이동된 file pointer에서부터 fragment를 읽어오게된다. 그후 상대 peer에게 해당하는 fragment를 전송한다.

DATA메시지를 수신시에는 상대 Peer로부터 fragment 를 수신한 후 수신된 fragment가 buffermap 에 존재하는지 확인한다. Fragment index와 block offset을 이용하여 저장된 위치로 filer pointer를 이동한 후 해당하는 위치에 fragment를 저장하는 원리로 동작한다.

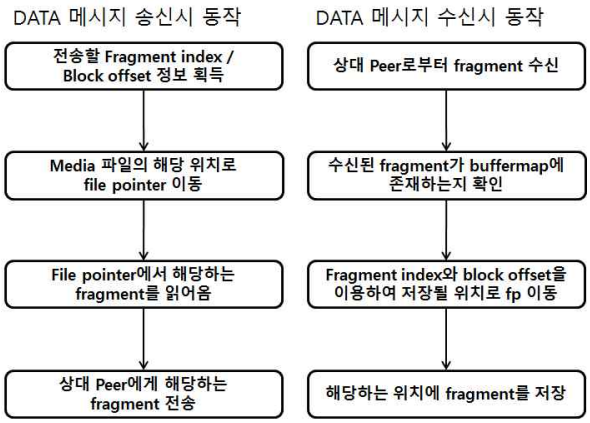

그림 11. DATA 메시지 송/수신시 file 관련 동작

4. 분산형 스트리밍 프로토콜 구현 및 시뮬레이션

PREP 프로토콜을 SVC 코덱으로 인코딩을 한 영상파일을 전송하는 시뮬레이션을 했다 [4]. 시뮬레이션 결과 성공적으로 전송되었음을 확인할 수 있었다. 시뮬레이션에 사용된 파일은 kara_video.mp4 파일이고 시뮬레이션 결과 아래의 그림과 같이 정상적으로 전송되었음을 확인할 수 있다. 또한 PREP프로토콜이 정상적으로 동작하는지는 Trace파일을 통해서 확인할 수 있다. Neighbor mode 에서 Partner mode로 전환되는 부분을 살펴보면 다음 그림 12에서 살펴볼 수 있으며 데이터 송신피어와 수신피어에 대한정보를 확인할 수 있고, 또한 메시지 타입도 살펴볼 수 있다.

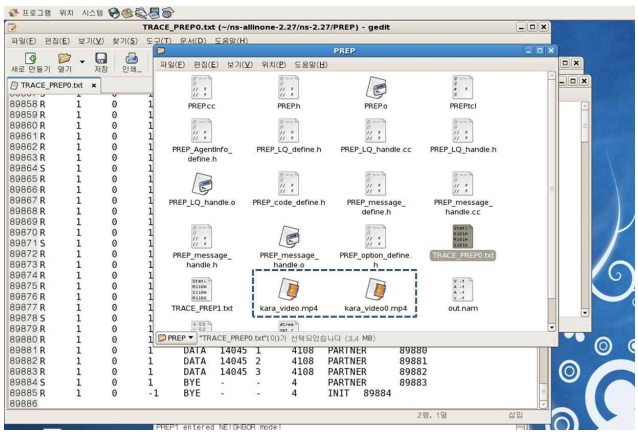

그림 12. mp4 미디어 파일 시뮬레이션 결과

Fragment 사이즈를 4096으로 정의하고 있는데 Trace파일상의 Fragment 사이즈는 4108로 나타나는데 이것은 헤더가 포함되서 전송이 되기 때문에 이와 같이 나타난다. 정상적으로 전송이 되었는지 확인하기 위해서는 총 전송된 File size를 합친 후 헤더에 사용된 byte를 뺏을 때 원본 파일과 같은지 확인할 수 있다. 그림 13 의 1번은 데이터를 성공적으로 전송된 후 BYE메시지를 교환하는 부분이고 2번은 마지막 Data의 Segment 번호이다. 그림 14 의 3번을 통해서 정상적인 메시지 구동이 되는지 확인할 수 있다. 4번은 Neighbor mode에서 Partner mode로 넘어가는 부분이다.

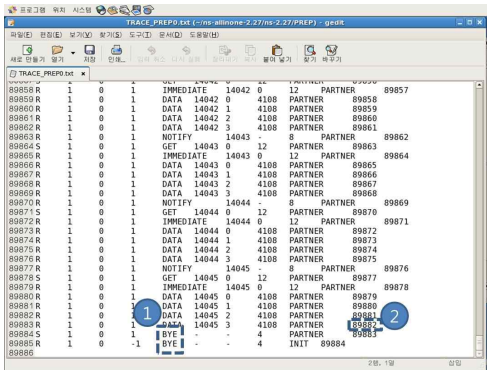

그림 13. 데이터 전송후 BYE 메시지 전송 및 마지막 segment 번호

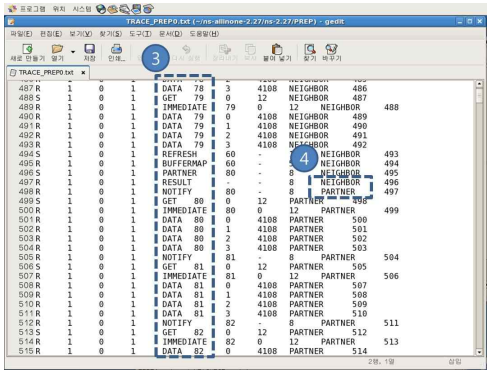

그림 14. 프로토콜 메시지 작동 확인 및 Neighbor mode / Partner mode 전환

5. 결론

기존의 P2P기반 스트리밍 서비스의 결점을 도출해서 이를 통해 새로운 분산형 스트리밍 전송 기술을 위한 프로토콜을 설계하고 시뮬레이션을 통해 성능을 확인해보았다. P2P의 고질적인 문제인 유동적인 피어 참여와 이탈 및 이기적인 사용자들로 인해서 지속적이고 안정적인 피어가 유지되기가 힘들었고 이를 보완할 필요가 있었다. 하지만 지속적이고 안정적인 피어 유지가 될 경우 분산형 스트리밍 서비스를 통해 차세대 네트워크의 핵심적인 역할을 담당할 것이다.

감사의 글

"본 연구는 미래창조과학부 및 정보통신산업진흥원의 대학ICT연구센터 육성·지원사업의 연구결과로 수행되었음" (NIPA-2014-H0301-14-1012).

6. 참고 문헌

[1] P. Frey "Master Thesis Swistry : P2P Live Streaming"January-July, 2006

[2] D. Ren, Y.-T. H. Li, S.-H. G. Chan "On reducing mesh delay for peer-to-peer live streaming," in Proc. *IEEE INFOCOM*, pp. 1058-1066, 2008.

[3] J. Liu, S. G. Rao, B. Li, and H. Zhang "Opportunities and challenges of peer-to-peer internet video broadcast," *Proc. of the IEEE*, vol. 96, no. 1, pp. 11-24, Jan. 2008.

[4] K. Fall, K. Varadhan "The NS manual (formerly ns notes and documentation)" *The VINT Project-A collaboration between researchers at UC Berkeley, LBL, USC/ISI, and Xerox PARC.* May 2010