클라이언트 선 저장방식을 위한 멀티미디어 데이터의 P2P 전송.

신광식[0] 정진하 윤완오 최상방
인하대학교 컴퓨터 구조 및 네트워크 연구실
sangbang@inha.ac.kr

P2P Transfer for Multimedia Data with Client initiated with Prefetching.

kwang-sik Shin[0] jin-ha Jung wan-oh woon sang-bang Choi
Computer Architecture & Networks Lab.

요 약

본 논문에서 우리는 부분 스트림 데이터를 위해 피어 투 피어 방식을 도입하는 방식을 제안하고 평가 하였다. 기존의 클라이언트 측 미리 저장 방식과 비슷하게 앞선 요청자에게 지연 없이 두 클라이언트가 같은 비디오 요청에 대해 같은 채널을 공유하는 것을 허용한다. 그것은 앞선 요청에 대해 멀티캐스트 세션을 초기화 및 진행하고 뒤 늦게 도달한 클라이언트에게 그 세션을 공유하는 것을 허용함으로써 보장된다. 그러나 공유 스트림 전송과는 달리 부분 스트림 전송에 있어서는 기존의 방식과 차이점을 보인다. 늦게 도달한 클라이언트는 늦은 만큼의 스트림 조각이 부족하게 되는 데 이 부분을 서버가 아닌 선행 클라이언트로부터 받게끔 함으로써 서버 측의 자원을 절약하는 효과를 얻고 있다. 분석결과는 우리가 제안한 방법이 주는 서버 네트워크 자원 이용의 절감과 그를 위해 대용되는 클라이언트 자원을 보여준다. 또한 시뮬레이션을 통해 실제 얻을 수 있는 이득을 확인하였다.

1.서 론

ADSL이나 케이블 모뎀등의 광대역 접속기술이 발전함으로 인해 양질의 스트림 서비스에 대한 요구가 높아지고 있다. 클라이언트 측에서의 끊김 없는 스트림 재생을 보장하기 위해서 특히 서버 측 네트워크 자원을 충분히 확보해야만 한다. 그러므로, 서버 네트워크 I/O의 대역폭을 얼마나 효율적으로 이용하느냐가 미디어 서비스의 질을 결정하게 된다.

서버 자원을 효율적으로 이용하기위해 여러 사용자가 하나의 스트림을 공유할 수 있도록 멀티캐스트 세션을 이용하는데, 이는 server-push와 client-pull의 두 가지 패러다임으로 분류하고 [1], 이를 좀더 세분화하여 Server-initiated, SIWP, Client-initiated, CIWP 4가지타입을 제시한다 [2].

본 논문은 CIWP 알고리즘을 기본으로하여 개선하였다. 후발 유저를 위해 유니캐스트로 보내주던 부분 스트림을 앞선 유저로부터 받게끔 함으로써 기본적인 멀티캐스트 세션 외에 추가로 필요하던 부분을 클라이언트 측 자원이용으로 대체하였다. 이를 위해 서버는 각 클라이언트에게 비디오를 멀티캐스트해주는 미디어 서버기능에 추가적으로 "단일 검색 서버를 사용하는 피어-투-피어" 모델에서의 검색 서버 역할을 해야만 한다 [3]. 이를위해 피어에게 추가적인 쿼리를 요구하진 않고 서버는 처음 받은 요청 정보만을 적절하게 스케쥴링하여 각 피어에게 P2P연결에 필요한 정보를 보내주는 것이다.

2. 연구 내용 및 방법

임의의 피어로부터 요청(Vid)이 들어왔을 때, 스케쥴러는 Vid에 해당하는 비디오의 크기(Vsize)를 보내준다. 이때 다음 피어로부터 요청이 들어왔을 때 이를 이용하게 P2P연결을 위한 정보를 주기위해 각 피어의 IP 어드레스를 저장하여 둔다. 스케쥴러는 비디오 데이터를 받기위해 조인하게될 멀티캐스트 세션과 부분 스트림을 제공해줄 피어를 결정하여 message (MGc, Vlength, peerIP)의 형태로 보내준다. 그리고 나서 부분 스트림을 제공하게 될 피어에게도 messae (NULL, Vlength, peerIP)의 메시지를 보내주어 두 피어 사이의 P2P연결을 통한 부분 스트림 전송을 원활히 하도록 돕는다. 그림1은 이러한 전반적인 제어용 메시지와 데이터의 흐름을 보여주며, 아래는 계속해서 각각의 알고리즘을 나타낸다.

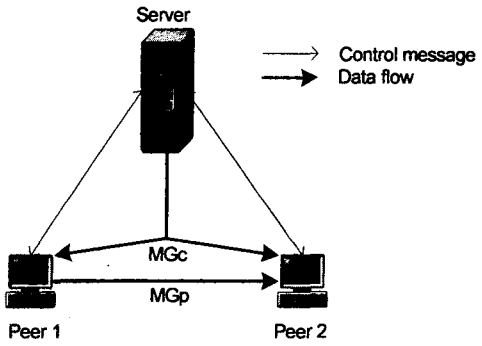

그림 1. 클라이언트와 서버간의 상호작용.

Vid는 클라이언트가 요청한 비디오 이름이며, Vsize는 서버에서 알려주는 요청된 비디오의 사이즈, MGc는 멀티캐스트 채널, MGp는 부분 스트림 채널 Vlength는 부분 스트림의 사이즈(즉, 멀티캐스트 채널이 시작되는 시퀀스 넘버)이고, pbuffer는 부분 스트림을 저장하기위한 버퍼, cbuffer는 멀티캐스트 채널을 통해 받은 스트림을 저장하기위한 버퍼, sPeer는 부분 스트림을 보내 줄 피어, rPeer는 부분 스트림을 받을 피어, sIP는 부분 스트림을 보내 줄 피어의 아이피 어드레스, rIP는 부분 스트림을 받을 피어의 아이피 어드레스를 나타낸다.

클라이언트 주 제어 쓰레드

1 Send a request message(Vid) to the server
2 Wait for the Vsize
3 If disk space is available
4 Send status "TRUE" to the server
5 Wait for the message(MGc,Vlength,sIP) from the server
6 Start both the complete and partial stream receiver thread
7 wait for the message(NULL,Vlength,rIP)
8 Start partial stream sender thread
9 Else
10 Send status "FALSE" to the server
11 Error Display
12 Wait for correct error and then only retry from start

멀티 캐스트 수신 쓰레드

1 Join channel MGc
2 Start to receive data for MGc
3 Start to save the received data to cbuffer after receiving a packet with a sequence number>Vlength
4 Leave multicast group MGc when the complete stream ends

부분 스트림 수신 쓰레드

1 If MGp is not NULL
2 Join channel MGp
3 Start to receive data from MGp
4 Start to save the received data to pbuffer
5 Playback the data saved in the pbuffer until no more data in the pbuffer
6 Leave multicast group MGp
7 Playback the data saved in the cbuffer until no more data in the cbuffer

부분 스트림 전송 쓰레드

1 Create channel MGp
2 Start to send data saved in the pbuffer to rIP until no more data in the pbuffer
3 Start to send data saved in the cbuffer to rIP until sequence number is Vlengths
4 Release the cbuffer

스케쥴러

1 When a client requesting video i arrive
2 Search Vid and then insert (Vid,Vsize) to video list
3 Send $Vsize_i$
4 Wait client disk space status
5 If status is TRUE(disk space available)
6 Check IP
7 If group channel i is NULL
8 Create channel MGi
9 Start to data delivery thread
10 Else
11 choose sPeer
12 Send message(MGi,Vlengthr,sIP) to rPeer
13 Send message(NULL,Vlengths,receiverIP) to sPeer
14 Else
15 Clear requested video list

데이터 전송 쓰레드

1 Repeat forever
2 While MG's video list is not empty
3 Remove the first pair(Vid,Vsize) from MG's video list
4 Multicast the first Vsize packets of video Vid to multicast group MG
5 Release the MG

3. 제안된 모델의 초기 효율성 분석

임의의 비디오 i 에 대한 요청만을 생각할 때 요청의 발생 분포는 평균 분당 요청율 λ_i 인 쁘아송 분포를 갖는다고 가정한다. 또한 서버자원은 무한대의 대역폭을 갖고있고, 요청이 들어왔을 땐 즉시 서비스 된다고 가정한다. 우리는 임의의 시간 동안의 요청에 대해 서버 측 자원과 클라이언트 측 자원의 이용 변화를 분석하여 우리가 제시하는 이론의 타당성을 검증하고자 한다. 또한 다음장에서 시뮬레이션을 통해 이를 확인 하였다.

$\{s(t):t>0\}$ 즉, $S(t)$ 를 0부터 시간 t까지 필요한 총 서버의 대역폭이라고 할 때, $\{N(t):t>0\}$ 즉, $N(t)$ 를 시간 t 까지의 서비스 받은 총 클라이언트 수를 나타낸다. 특히 본 연구에서 우리가 관심을 갖는 평균 서버 대역폭은 $\overline{C}=\lim_{t\to\infty} S(t)/t$ 로 나타내어지며 이는 다음과 같이 표현된다.

$$\overline{C}=\lambda_i \frac{E[S]}{E[N]}$$

그리고, $E[N]=1+\lambda_i T_i$ 이다.

먼저 클라이언트 측 디스크는 충분하다고 가정하고, 임의의 시간 T동안의 요청에 대해 필요한 서버자원의 평균값을 구하기 위해 우선 그기간 동안 k번의 요청이 이루어질 확률을 먼저 구한다.

$$P[K=k]=(\lambda_i T)^k e^{-\lambda_i T}/k!.$$

그리고 나서 k번 요청이 들어왔을 때 이를 즉시 서비스하는 과정에서 필요한 자원은 다음과 같다.

3.1 평균 서버 측 네트워크 I/O 대역폭($\overline{C_s}$)

1) CIWP

$E[S|K=k]=(L_i+\frac{(k+1)T}{2})b$ 에서

$$E[S]=\sum_{k=1}^{\infty}\frac{(\lambda_i T)^k e^{-\lambda_i T}}{k!}E[S|K=k]$$

$$=bL_i+\frac{b}{2}\sum_{k=1}^{\infty}\frac{(\lambda_i T)^k e^{-\lambda_i T}}{k!}(k+1)T$$

$$=bL_i+\frac{bT(\lambda_i T+1)}{2}$$

그러므로,

$$\overline{C_s}=\lambda_i\frac{2L_i+(\lambda_i T+1)T}{2(\lambda_i T+1)}b$$

2) CIWP with P2P for partial stream

$$E[S|K=k]=L_i b$$

$$E[S]=L_i b$$

$$\overline{C_s}=\frac{\lambda_i L_i b}{(1+\lambda_i T)}$$

여기서 b는 실시간 재생을 위해 보내지는 비디오 데이터 패킷의 크기이고, Li는 비디오의 전체 데이터 패킷의 길이이다.

3.2 평균 클라이언트 네트워크 I/O 대역폭($\overline{C_c}$)

1) CIWP

$$T[C|K=k]=(k+1)L_i b$$

$$T[C]=(\lambda_i T+1)L_i b$$

$$\overline{C_c}=\lambda_i L_i b$$

2) CIWP with P2P for partial stream

$$T[C \mid K=k]=(k+1)L_t b+\frac{(k+1)T}{2}b$$

$$T[C]=(\lambda_t T+1)L_t b+\frac{(\lambda_t T+1)T}{2}b$$

$$\overline{C_c}=\lambda_t L_t b+\frac{\lambda_t T b}{2}$$

피어 간에 전송하는 부분 스트림 만큼의 대역폭이 더 필요하다.

$T[C]$는 전체 클라이언트 네트워크의 대역폭.

3.3 평균 클라이언트 디스크 공간($\overline{C_b}$)

1) CIWP

$$T[B \mid K=k]=\frac{(k+1)T}{2}b$$

$$T[B]=\frac{(\lambda_t T+1)T}{2}b$$

$$\overline{C_b}=\frac{\lambda_t T}{2}b$$

2) CIWP with P2P for partial stream

$$T[B \mid K=k]=\frac{(k+3)T}{2}b$$

$$T[B]=\frac{(\lambda_t T+3)T}{2}b$$

$$\overline{C_b}=\frac{(\lambda_t T+3)T}{(\lambda_t T+1)2}b$$

$T[B]$ 는 전체 클라이언트 디스크 공간.

위에서 보는 바와 같이 CIWP와 비교할 때 $\overline{C_s}$ 가 $\lambda_t Tb/2$ 만큼 줄어든 대신 $\overline{C_c}$ 가 $\lambda_t Tb/2$ 만큼 그리고, $\overline{C_b}$ 가 $3Tb/2(\lambda_t T+1)$ 만큼 늘었다. 이는 클라이언트 측 자원을 이용하여 서버 측 자원을 대체하는 것을 보여준다. 특히 서버 측 자원이 클라이언트 측 자원보다 훨씬 비싸다는 걸 감안하면 우리가 제시한 이론이 상당히 효율적이리라는 걸 예측할수 있다.

4. 시뮬레이션

분석 과정에서는 하나의 비디오에 대해 초점을 맞췄지만, 일반적인 비디오 서버인 경우 다양한 선호도를 가진 여러 종류의 비디오가 서비스된다. 우리는 CIWP와 CIWP with P2P for partial stream의 요청수당 필요한 채널수, 고정된 개수의 채널에서의 지연시간 및 시간 지연이 발생할 확률을 비교하였다.

시뮬레이션에서 요청은 1/λ의 간격으로 발생하는 뽀아송 분포를 따르며, 비디오 선호도는 Zipf-like 분포를 따른다고 가정한다. 또한 클라이언트는 충분한 디스크 공간을 가지고 있으며, 데이터 전송 중에는 연결을 끊지 않고, 각 클라이언트는 고속 통신망을 이용한다고 가정한다. 마지막으로 원활한 제어를 위해 각 클라이언트는 자신의 아이피 어드레스를 숨기지 않으며, 각 피어 관리를 위해 필요한 오버헤드는 시간당 데이터 전송량에 비하여 매우 작으므로 결과를 비교하는 데 있어서는 무시하였다. 시뮬레이션 과정에서 사용된 파라미터 값은 표 1과 같다.

표 1 시뮬레이션에서 사용된 파라미터 값

	평균	범위
비디오 수	100	N/A
분당 요청률	50	40-100
비디오 길이(분)	90	70-110
Skew Factor	0.271	N/A

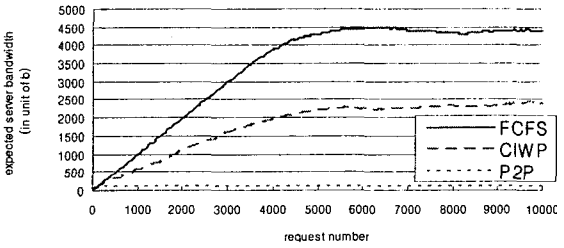

그림 2 클라이언트 요청에 따른 서버자원의 이용량

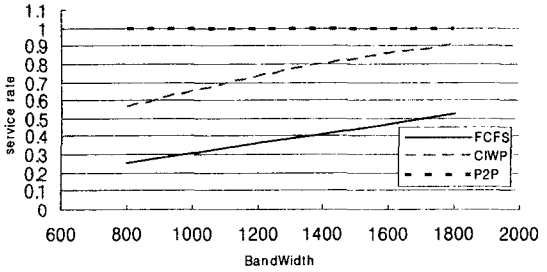

그림 3. 10000번의 요청 동안 각 대역폭에 따른 클라이언트가 서비스 받은 평균값

5. 결론

본 연구는 여분의 클라이언트 자원을 이용하여 서버 자원을 절감하는 CIWP 모델을 기본으로 서버의 부하를 최소화할 수 있는 P2P 모델을 접목시켜 클라이언트 자원이용의 비율을 늘림으로써 효율성을 극대화 시킬 수 있는 획기적인 모델을 제시하고 있다. 제안한 방법을 통해 얻을 수 있는 서버 측 네트워크 자원 이용의 절감 효과를 분석을 통해 살펴보았고, 또한 시뮬레이션을 통해 실제 얼마만큼의 이득을 얻을 수 있는지를 확인 하였다.

6. 참고문헌

[1] P. J. Shenoy, P.Goyal, and H. M. Vin, "Issues in Multimedia Server Design," ACM Computing Surveys, Vol. 27, No. 4, Dec. 1995.

[2] Lixin Gao and Don Towsley, "Threshold-Based Multicast for Continuous Media Delivery," IEEE Transactions on Multimedia, Vol. 3, No. 4, Dec. 2001.

[3] Lance Olson, "MSDN Magazine," Feb. 2001.