멀티미디어 통신환경에서 QoS 지원을 위한 동적 버퍼관리 기법

박진형, 이승룡
경희대학교 전자계산공학과

Dynamic Buffer Management Method to Support QoS in Multimedia Communication Environment

Jin-Hyung Park, Sungyoung Lee
Dept. of Computer Engineering, Kyung Hee University

요 약

멀티미디어 통신 환경에서 대용량의 멀티미디어 데이터를 처리하기 위하여 버퍼의 사용을 효율적으로 관리하고 최적의 성능을 내기 위한 버퍼 관리 기법이 요구된다. 지금까지 대부분의 버퍼 관리 기법에서는 버퍼의 크기를 고정시킴으로써 네트웍의 상황과 서버에서의 데이터 흐름에 유연하게 대처할 수 없었다. 본 논문에서는 멀티미디어 클라이언트 서버 통신 환경에서 클라이언트 측의 버퍼를 동적으로 관리함으로써 오버플로우를 방지하여 사용자에게 양질의 QoS를 제공할 수 있는 동적인 버퍼 관리 기법을 제안한다. 제안한 기법은 서버에서 데이터를 보내기 전에 서버가 Bitrate를 전송함으로써 클라이언트는 전송되는 데이터의 양을 예측하고, 예측된 결과를 바탕으로 클라이언트는 오버플로우가 예상되면 보조 버퍼를 생성한 뒤 그곳에 데이터를 할당함으로서 오버플로우를 방지할 수 있다.

1. 서론

인터넷을 이용한 컴퓨터와 통신기술의 발전으로 VOD(Video On Demand), AOD(Audio On Demand), 화상회의, 인터넷 방송 등과 같은 다양한 멀티미디어 데이터를 이용한 서비스가 제공되고 있다. 이러한 서비스들은 음성, 동영상과 같은 다양한 멀티미디어 데이터를 네트웍을 통하여 처리하기 위하여 고성능의 컴퓨터와 고속의 통신망 기술이 요구되어 진다. 또한 대용량의 멀티미디어 데이터를 처리하기 위한 매우 큰 크기의 버퍼가 요구되며, 버퍼에 저장된 데이터에 대한 I/O 횟수도 상당히 많다. 따라서 버퍼에 대한 I/O를 효율적으로 처리하고, 응용 서비스 및 미디어에 알맞은 버퍼 크기를 계산하며, 버퍼 언더플로우(underflow)와 버퍼 오버플로우(overflow)에 효과적으로 대처할 수 있는 유연한 버퍼 관리 기법이 요구된다.

지금까지 멀티미디어 통신 시스템과 관련된 버퍼 관리 기법에 관한 연구는 상당히 많이 있어왔으나 대부분이 네트웍의 상황에 따라 버퍼의 크기가 동적으로 변하는 시스템이 아닌 버퍼링 초기에 버퍼의 크기가 고정된 정적인 시스템이 대부분이었다[2][3]. 대역폭 요구(bandwidth requirement)를 조정하는 방법은 버퍼의 오버플로우를 예방하는데 있어서 좋은 결과를 기대할 수 있으나 사용자에게 좋은 QoS를 제공할 수는 없다. 이러한 정적인 버퍼 관리 기법은 버퍼의 오버플로우에 유연하게 대처 할 수 없어 사용자에게 양질의 QoS 제공을 기대할 수 없다.

따라서 본 논문에서는 다양한 멀티미디어 데이터에 대해서 버퍼 오버플로우를 효과적으로 대처하며, 사용자에게 양질의 QoS를 보장할 수 있는 효율적이고 유연한 동적인 버퍼 관리 기법을 제안한다. 제안된 버퍼 관리 기법은 서버 측에서의 데이터 전송 정보와 네트웍의 상황을 고려하여 클라이언트 측의 버퍼 크기를 동적으로 변환시킴으로써 사용자에게 보다 빠른 응답시간을 제공할 수 있다. 또한 클라이언트의 패킷 수신 버퍼와

미디어 재생 장치간의 데이터 전달을 Push와 Pull 방식 모두 지원함으로써 단일의 버퍼로 다양한 미디어 재생 장치를 지원할 수 있다는 장점이 있다.

본 논문의 구성은 다음과 같다. 2장에서는 기존에 연구되었던 버퍼 관리 기법에 대한 관련 연구를 살펴보고, 3장에서는 제안한 시스템의 구조에 대해서 알아보며, 4장에서는 동적인 버퍼 관리를 위한 알고리즘들에 대해서 설명하고, 5장에서 결론을 내린다.

2. 관련 연구

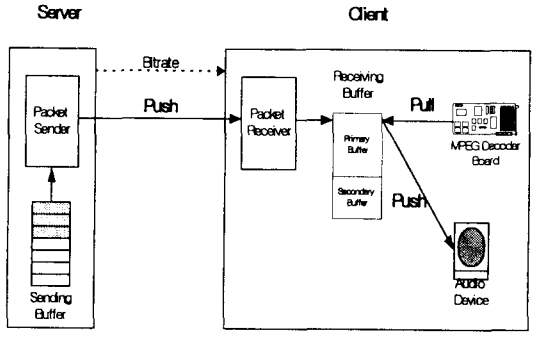
클라이언트에서의 버퍼 관리 기법과 관련된 연구는 매우 다양한 방면에서 많이 있었으나, 대부분이 고정된 크기의 버퍼를 사용하는 정적인 버퍼 관리 기법이다. 그리고, 네트웍의 상황을 고려하여 대역폭 요구를 조절하여 버퍼의 오버플로우를 방지하는 시스템에 대한 연구도 많이 이루어졌다.

Hong Kong University의 Jack Y. B. Lee는 Concurrent Push 방식을 사용한 병렬 비디오 서버 시스템에 관한 연구에서 서버 측에서의 버퍼 언더플로우와 클라이언트 측에서의 버퍼 오버플로우를 방지하기 위한 효율적인 버퍼 크기를 계산하는 버퍼 관리 기법을 제시하였다[3]. 그러나 이 연구는 서버와 클라이언트간의 네트웍 전송에서 기존의 단일 전송대신에 병렬방식의 전송 기법을 사용함으로써, 매우 좋은 성능을 얻는 반면에 단일 전송에 비해 더 많은 크기의 버퍼 용량을 요구한다는 단점이 있다.

또한 GMD의 Ingo Busse, Bernd Deffner, Henning Schulzrinne은 대역폭 요구를 동적으로 조정하는 방법을 제안하였다[2]. 이 연구는 서버에서 클라이언트로 데이터를 전달하는데 RTP를 사용하고 RTCP를 이용하여 네트웍 상태를 추정한 후 이를 분석한 결과를 바탕으로 bandwidth를 조정하는 방법을 사용하였다. 그러나 이 방법은 네트웍 상황만을 고려하고 클라이언트 측의 버퍼의 상황을 고려하지 않기 때문에 버퍼의 오버플로우에 효과적으로 대처할 수 없고 서버 측에서의 데이터 전송이 불규칙적일 경우 양질의 QoS를 제공할 수 없다는 단점이 있다.

3. 버퍼 관리 모델

이 장에서는 버퍼의 오버플로우를 방지하고 사용자에게 양질의 QoS를 제공할 수 있는 클라이언트에서의 동적인 버퍼 관리 모델을 제안한다. 제안한 모델의 구조는 [그림 1]과 같다.



[그림 1] 시스템의 전체 구조

[그림 1]에서처럼 제안된 동적 버퍼관리 모델은 RTP를 사용하여 클라이언트/서버간에 PUSH 방식으로 데이터를 전달하고, 클라이언트의 패킷 수신 버퍼와 미디어 재생 장치간의 데이터 전달은 Push와 Pull 방식 모두 지원한다. 대부분의 버퍼 관리 기법들이 클라이언트의 패킷 수신 버퍼와 미디어 재생 장치간의 데이터 전달을 Push와 Pull 방식중의 하나만 제공하였기 때문에 다양한 미디어 재생장치를 지원할 수 없었고, Push와 Pull 방식을 모두 지원한다 하더라도 단일한 인터페이스로 제공하지 않아 다양한 미디어 재생 장치를 손쉽게 이용할 수 없다는 한계를 가지고 있었다. 본 논문에서는 클라이언트에서의 패킷 수신 버퍼가 Push 방식과 Pull 방식의 데이터 출력을 단일한 인터페이스로 제공하여 다양한 미디어 재생 장치를 쉽게 지원할 수 있다.

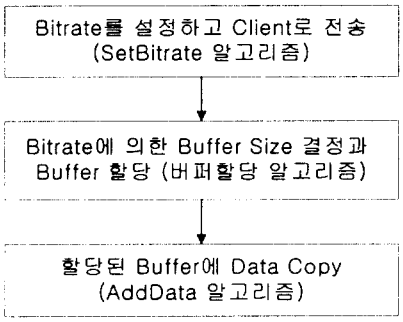
서버와 클라이언트간에는 네트웍의 대역폭에 따라 크기가 변하는 Bitrate라는 채널을 설정하여 서버에서 전송되어질 데이터의 Bitrate를 클라이언트에 미리 전송함으로써, 클라이언트는 서버에서 전송되는 Bitrate에 따라 버퍼에서 오버플로우의 발생 유무를 예측할 수 있다.

버퍼는 최소의 크기를 유지하는 주(primary) 버퍼와 오버플로우를 대비하여 준비하는 보조(secondary) 버퍼로 나누어진다. 주 버퍼는 시스템 구현 초기에 할당되고, 보조 버퍼는 시스템 작동 시 서버에서 전송되는 Bitrate에 근거하여 오버플로우가 예상되면 설정된다.

4. 동적인 버퍼 관리 알고리즘

클라이언트 측에서 버퍼의 크기를 동적으로 변화시키는 방법(보조버퍼 할당정책)은 아래의 [그림 2]와 같다. 서버에서 클라이언트로 데이터를 전송하기 위하여 Bitrate를 정할 때 이를 클라이언트

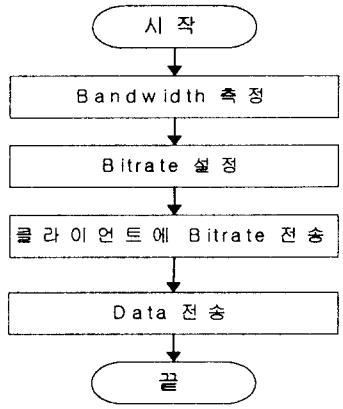
로 보내줌으로써 클라이언트 측에서는 서버에서 보내지는 데이터의 양을 알 수 있다. 서버에서 보내는 데이터의 양이 미디어에 의해서 플레이되어 소모되어지는 데이터의 양보다 적을 경우, 언더플로우가 발생하게 되며 최초의 주 버퍼만 유지된다. 그러나 소모되어지는 데이터의 양보다 서버에서 보내는 데이터의 양이 많을 경우, 버퍼의 오버플로우가 발생되어질 수 있는데 이를 방지하기 위하여 서버에서 보내는 데이터의 양에 비례해서 클라이언트에 보조 버퍼를 할당함으로써 버퍼의 오버플로우를 예방할 수 있다.



[그림 2] 제안한 시스템의 동작 구조

4.1 SetBitrate 알고리즘

서버에서 Bitrate를 결정하고 클라이언트로 보내는 SetBitrate 알고리즘은 아래의 [그림 3]과 같다.



[그림 3] SetBitrate 알고리즘

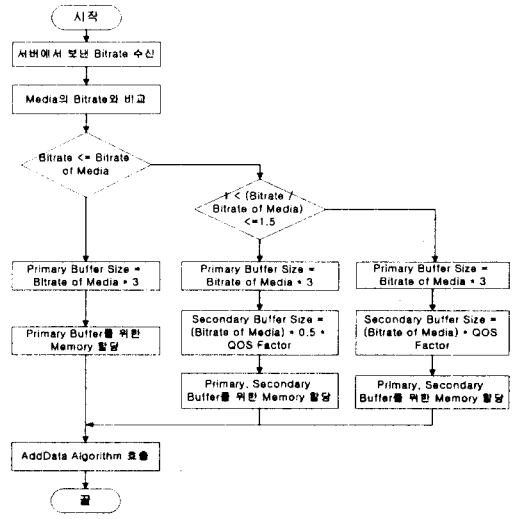
서버에서 클라이언트로 전송하는 데이터의 양을 나타내는 Bitrate는 네트워의 대역폭에 따라 변하게 된다. 네트워의 대역폭를 체크하여 그에 따라 서버에서 클라이언트로 보내는 데이터의 Bitrate를 설정하고, 설정된 Bitrate를 클라이언트로 보낸 후 서버에서 클라이언트로 데이터를 전송한다.

서버와 클라이언트 사이의 대역폭을 측정하는 방법으로는, 서버와 클라이언트 사이에 네트워의 상황을 모니터링 할 수 있는 제어 라인을 두어 이에 따라 대역폭을 할당하는 방법[1][2]과 트래픽을 예측하여 전송 시 필요한 대역폭을 동적으로 할당하는 방법[4][5][6]등이 있다.

4.2 버퍼할당 알고리즘

서버에서 보낸 Bitrate에 따라 보조버퍼의 크기를 결정하고 메모리에 버퍼를 할당하는 버퍼할당 알고리즘은 아래의 [그림 4]와 같다.

Bitrate의 잦은 변화로 버퍼의 크기를 계속 변화시키는 것은 시스템에 큰 오버헤드가 될 수 있기 때문에 최초의 Bitrate를 받은 이후 그에 따라 Low, Medium, High의 세 가지 상태로 나눈다. 이를 결정하는 방법은, *Bitrate < Bitrate of Given Media*일 경우에는 Low, *1 <= Bitrate/(Bitrate of Given Media) < 1.5* 일 경우에는 Medium, *Bitrate/(Bitrate of Given Media) >= 1.5*일 경우를 High로 정한다. Low인 경우에는 보조버퍼를 할당하지 않고 주버퍼만을 유지하며, Medium 과 High 인 경우에만 보조버퍼를 할당하게된다.



[그림 4] 버퍼할당 알고리즘

Medium일 경우 보조버퍼의 크기를 결정하는 방법은, *Secondary_Buffer_Size = (Bitrate of Given*

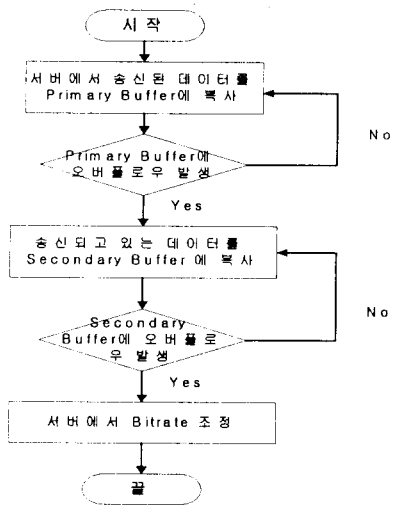
*Media) * 0.5 * QoS Factor* 이고, High인 경우 *Secondary_Buffer_Size = (Bitrate of Given Media) * QoS Factor*이다.

위의 식에서 QoS Factor는 보조 버퍼의 크기를 결정하는 중요한 요소로써 사용자의 요구와 시스템의 성능에 따라 결정할 수 있다. 보조 버퍼를 크게 잡아 메모리 효율이 떨어질 지라도 사용자가 양질의 QoS를 원하는 경우 또는 시스템에 여유 메모리가 클 경우에는 QoS Factor를 크게 한다.

양질의 QoS를 원하는 경우 QoS Factor를 높여 버퍼의 크기를 키우고, 저수준의 QoS를 원하는 경우 QoS Factor를 낮추어 적은 사이즈로 버퍼의 크기를 키운다. QoS와 메모리 사용 효율간에는 Trade-Off 관계가 있어, 양질 QoS를 원한다고 QoS Factor를 높게 키울 경우 메모리 효율에 매우 안 좋은 영향을 끼칠 수 있다.

4.3 AddData 알고리즘

서버에서 송신된 데이터를 클라이언트의 버퍼에 추가하는 AddData 알고리즘은 아래의 [그림 5]와 같다.



[그림 5] AddData 알고리즘

버퍼에 데이터를 저장하는 방법은, 주 버퍼에 여유가 있을 경우 먼저 주 버퍼로 데이터를 저장시키고, 주 버퍼에 오버플로우 발생이 예측되면 새롭게 생성한 보조버퍼에 데이터를 저장한다. 오버플로우를 방지하기 위하여 만든 보조버퍼에서 오버플로우가 예상되면 서버에서의 데이터 전송 Bitrate를 조절하여 클라이언트에서 버퍼의 오버플로우를 예방한다.

버퍼 내에 더 이상 데이터가 존재하지 않을 경우 보조버퍼에 할당된 메모리를 해제한다.

5. 결론

본 논문에서는 사용자에게 양질의 QoS를 제공할 수 있는 클라이언트 측에서의 버퍼 관리 기법을 제안하였다. 제안된 기법은 클라이언트의 패킷 수신 버퍼와 미디어 재생 장치간의 데이터 전달을 Push와 Pull 방식 모두 지원함으로써 단일의 버퍼로 다양한 미디어 재생 장치를 지원할 수 있다. 또한 서버에서 Bitrate를 클라이언트에 미리 전송함으로써 클라이언트에서는 버퍼의 오버플로우를 예측하여 버퍼의 크기를 동적으로 변화시켜 오버플로우를 방지하고 양질의 QoS를 제공할 수 있다.

오버플로우 방지와 양질의 QoS 제공을 위한 보조 버퍼의 할당은 메모리 효율과 Trade-Off 관계에 있기 때문에, 양질의 QoS를 보장하며 시스템에 적은 부담을 줄 수 있는 QoS Factor의 적절한 선택이 요구되어진다.

[참고문헌]

[1]. Pawan Goyal, Harrick M. Vin, Chia Shen and Prashant J. Shenoy, "A Reliable, Adaptive Network Protocol for Video Transport", In Proceedings of IEEE INFOCOM, 1996

[2]. Ingo Busse, Bernd Deffner, Henning Shulzrinne, "Dynamic QoS Control of Multimedia Application based on RTP", Computer Communications, Vol. 19, pp. 49-58, January, 1996

[3]. Jack Y. B. Lee, "Concurrent Push-A Scheduling Algorithm for Push-Based Parallel Video Servers", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 9, No. 3, pp. 467-477, April, 1999

[4]. A. Garcia-Martinez, J. Fernandez-Conde, A. Vina, "Efficient memory management in video on demand servers", Computer Communications 23 (2000), pp. 253-266

[5]. Song Chong, San-qi Li, and Joydeep Ghosh, "Predictive Dynamic Bandwidth Allocation for Efficient Transport of Real-Time VBR Video over ATM", IEEE Journal on selected areas In Communications, Vol. 13, No. 1, January, 1995

[6] G.Chiruvolu, R. Sankar, and N. Ranganathan, "VBR video traffic management using a predictor-based architecture", Computer Communications 23(2000), pp. 62-70, July, 1999