VOD 시스템에서 클라이언트 버퍼 및 네트워크 상태를 고려한 흐름제어

김종수 김정원 정기동
부 산 대 학 교 전 자 계 산 학 과
{jskim, jwkim}@melon.cs.pusan.ac.kr, kdchung@hyowon.cc.pusan.ac.kr

A Flow Control using Client Buffer and Network State in VOD Systems

Jong-Su Kim Jeong-Won Kim Ki-Dong Chung
Dept. of Computer Science, Pusan National University

요 약

본 논문은 VOD 시스템에서 QoS를 보장해 주기 위한 흐름 제어 기법을 제안하고 실제 구현된 VOD 시스템에서 제안된 알고리즘을 적용하고 성능을 측정하였다. 기존의 버퍼를 관리하는 Leaky Bucket 알고리즘과 패킷의 손실률에 기반한 네트워크의 상태를 동시에 고려해서 클라이언트와 서버간에 전송 속도를 제어한다. 즉, 시간에 따라 변화하는 네트워크의 상태를 클라이언트 버퍼 관리에 반영해줌으로써 클라이언트 버퍼의 오버플로우 및 언더플로우를 방지하고 QoS를 보장한다.

1. 서론

오늘날 컴퓨터와 통신망 기술이 급속도로 발전함에 따라 텍스트, 이미지, 오디오, 비디오 등과 같은 다양한 매체들을 VOD 시스템과 같이 하나의 종합된 멀티미디어 응용 형태로 서비스하기 위해 많은 연구가 활발히 진행되고 있다.

이러한 멀티미디어 데이터를 실시간으로 서비스하기 위해서는 서버의 전송률과 버퍼 공간 및 네트워크의 가용 능력이 고려되어야 할 뿐만 아니라, 멀티미디어 데이터의 특성인 대용량 처리와 연속성을 보장해야 한다[1].

VOD는 통신망 및 서버/클라이언트 시스템의 상태 변화에 의해 스트림의 전송 및 수신이 가변적이다. 따라서 클라이언트 버퍼의 오버플로우 또는 언더플로우가 발생하여 단절감 없는 비디오의 재생과 QoS 보장이 어렵다.

이러한 버퍼의 오버플로우와 언더플로우를 방지하기 위해서는 버퍼가 정해진 임계 구역을 벗어날 때 서버가 새로운 전송률로 클라이언트에게 데이터를 전송하는 메카니즘이 필요하다.

본 논문은 클라이언트의 버퍼량과 네트워크의 상태를 동시에 고려하여 서버와 클라이언트간에 전송 속도를 제어하는 알고리즘을 제안하고 있으며 실제 구현을 통해서 그 성능을 측정한다.

2장에서는 패킷 손실률에 기반한 기존의 흐름제어 기법을 살펴보고 3장에서는 본 논문에서 제안하는 흐름제어 기법을 설명한다. 4장에서는 실험을 통해서 제안된 알고리즘의 성능을 평가하며 5장에서는 결론 및 향후 연구과제를 제시한다.

2. 패킷 손실률에 기반한 흐름제어 알고리즘

패킷 손실에는 두 가지 이유가 있다. 첫째는 버퍼 오버플로우이고 두 번째는 비트 에러 때문이다. 대부분의 네트워크에서 비트 에러의 가능성은 매우 낮으며 대부분의 패킷 손실은 라우터에서 버퍼의 오버플로우에 기인한다. 따라서 패킷 손실률을 통해서 네트워크의 상태를 파악할 수 있다[2,3].

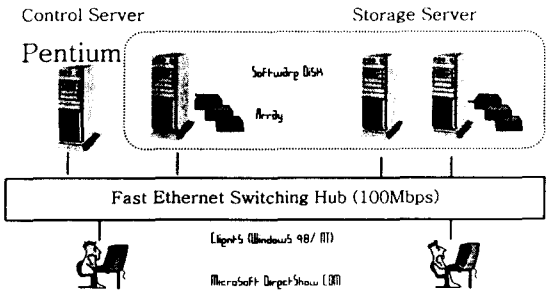
패킷 손실률에 기반한 흐름 제어 알고리즘은 다음과 같다.

```
if(packLoss > threshold)
        then currRate = max(α * currRate, minRate)
else
        then currRate = min(β * currRate, maxRate)
                        ( 0 < α < 1, β >= 1)
```

만약 packLoss가 임계치(threshold)를 초과하면 네트워크가 과부하임을 나타내므로 currRate를 감소시켜 더 적은 양의 패킷을 보내야 하며, 반대로 packLoss가 임계치보다 적으면 네트워크가 unloaded임을 나타내므로 currRate를 증가시켜 더 많은 양의 패킷을 보내야 한다[2]. 그리고 임계치가 높을수록 currRate는 증가하게 된다. 따라서 만약 currRate가 떨어져서 한동안 minRate와 같게 된다면, 네트워크가 과부하임을 나타내므로 임계치를 더 높여주어야 한다[4].

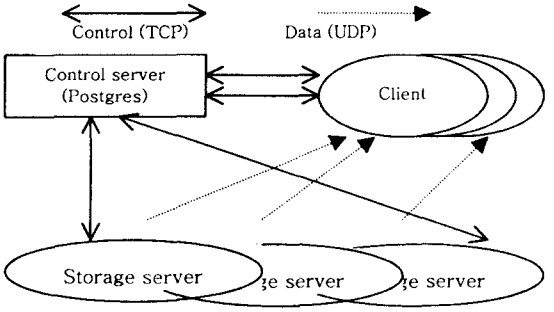
3. 흐름 제어 알고리즘 구현

3.1 VOD 시스템의 구조



[그림 1] 시스템 환경

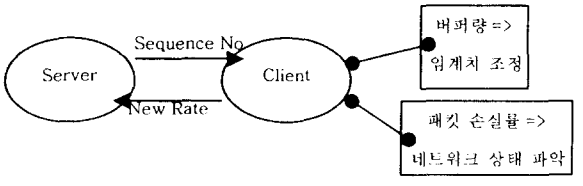
[그림 1]은 구축된 VOD 시스템의 구조를 나타내고 있다[5]. 명령 메시지를 처리하는 하나의 제어 서버(Control Server), 실제 스트림을 전송하는 다수의 스토리지 서버(Storage Server), 그리고 이들을 상호 연결하는 고속의 내부 네트워크로 구성된다.



[그림2] 시스템의 논리적 연결 구조

[그림 2]는 서버와 클라이언트와의 논리적인 연결구조를 표현한다. 제어 서버가 TCP/IP를 통해서 클라이언트의 모든 요구를 처리하고, 스토리지 서버는 UDP/IP를 통해서 연속적인 데이터 전송을 책임진다.

3.2 제안 알고리즘



[그림 3] 알고리즘 개요도

VOD 시스템은 대용량의 데이터를 실시간으로 서비스해야 하기 때문에 버퍼와 네트워크를 동시에 고려해야 한다.

[그림 3]은 흐름 제어를 위한 개괄적인 알고리즘을 나타내고 있다. 서버(storage server)는 전송되는 데이터의 헤드에 시퀀스 번호를 패킷화해서 전송한다. 클라이언트는 전송 받은 패킷의 시퀀스 번호를 이용해서 패킷 손실률을 구할 수 있다[6].

이 패킷 손실률에 의해서 파악된 네트워크의 상태와 Leaky Bucket 알고리즘[7]으로 관리되는 버퍼를 동시에 고려하여 서버의 새로운 전송률을 계산한다.

```
curr_Rate = (File_size / Playback_time) / Packet_size;
Send_currRate_To_Server;
Function Flow_control( ); // 주기적으로 호출되는 Function

BEGIN
count number of lost packets;
packLoss = number of lost packets /
       (number of lost packets + number of received packets);
threshold = Control_Buffer();  // 남아있는 버퍼량을 체크

number of lost packets  = 0;
number of received packets = 0;

//network congestion 상태
if(packLoss > threshold){   // rate 감소
    currRate = max(α * currRate, minRate);
    Send_currRate_To_Server;
}

// network unloaded 상태
else if(packLoss < threshold){  // rate 증가
      currRate = min(β * currRate, maxRate);
      Send_currRate_To_Server;
 }
 else                  // network loaded 상태
      return;

 save last packet number to be received ;
 END
                ( 0 < α < 1, β > 1 )
```

[알고리즘 1] 네트워크 상태를 고려한 흐름제어

[알고리즘 1]은 패킷 손실률에 기반해서 네트워크의 상태를 고려해서 서버의 새로운 전송률을 구해서 서버에게 전송해 준다. 그리고 임계치를 조정하기 위해 Contro_Buffer 루틴을 호출한다.

[알고리즘 2]에서는 남아있는 버퍼량을 통해서 임계치를 동적으로 할당해 줌으로써 버퍼와 네트워크의 상태를 동시에 고려해 주고 있다. 즉, 남아있는 버퍼량이 오버플로우 경고 구간(Upper_Bound)에 있을 경우, 임계치를 남아있는 버퍼량(ϕ)에 비례하여 낮추어 준다. 그

래서 서버의 전송률을 낮추어줌으로써 클라이언트 버퍼의 오버플로우를 대비한다. 또한 남아있는 버퍼량이 언더플로우 경고 구간(Lower_Bound)에 경우에는 임계치를 남아있는 버퍼량에 비례하여 높여줌으로써, 서버의 전송률을 높여주어 언더플로우를 막아주게 된다.

```
Control_Buffer( float curr_threshold ){
  φ = ΣInput_Data - ΣPlayout_Data; // 남아있는 버퍼량

  // φ가 오버플로우 경고 구간에 있을 경우
  if( φ > Upper_Bound)
    next_threshold = curr_threshold  - α * φ; // α는 임의의 상수

  // φ가 언더플로우 경고 구간에 있을 경우
  else if( φ < Lower_Bound)
    next_threshold = curr_threshold       // QUEUE_SIZE
              +      α * (QUEUE_SIZE - φ); // : 총 버퍼 사이즈

  else
    next_threshold = curr_threshold;

  return next_threshold;
}
```

[알고리즘 2] 제안된 임계치 결정 알고리즘

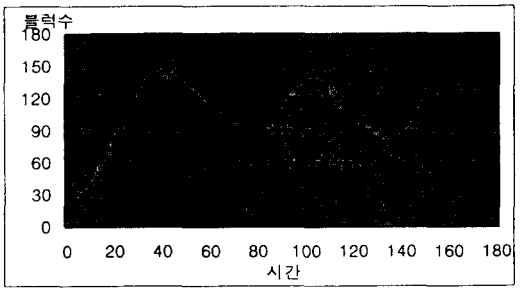
4. 실험 및 성능 분석

캠퍼스 LAN 환경에서 2개의 호스트(리눅스 기반)에 각각 storage 서버와 control 서버를 설치하고, Microsoft DirectShow 인터페이스 이용하여 구현된 클라이언트를 사용한다.

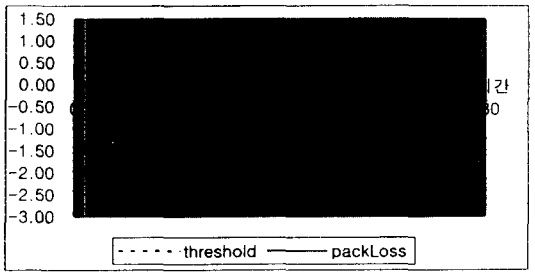
버퍼는 32KB 단위의 블록들을 200개 사용하였다. 그리고 36,113KB의 MPEG-1 데이터를 사용하였다.

성능 분석을 위해 시간에 따른 버퍼량과 임계치, 패킷 손실률을 구한다. 그리고 임계치와 패킷 손실률에 따른 버퍼량의 변화를 보여준다.

[그림4]와 [그림5]는 남아있는 버퍼량에 따라 임계치를 동적으로 조정해서 버퍼량을 일정한 범위 내(90 ~ 140 블록)에서 변화시킨다. 따라서 클라이언트 버퍼의 오버플로우와 언더플로우를 최대한 방지한다.



[그림4] 시간에 따른 버퍼량



[그림5] 시간에 따른 임계치와 패킷 손실률

5. 결론 및 향후 연구과제

본 논문은 클라이언트 버퍼의 오버플로와 언더플로를 최대한 막고 서비스의 질을 향상시킬 수 있도록 새로운 흐름 제어 기법을 제안하고 있다.

네트워크 상태와 버퍼를 동시에 고려해줌으로써 대용량의 데이터를 실시간으로 서비스해야 하는 VOD 시스템에 맞도록 제안되었다.

향후 RTP/RTCP 와 실시간 전송 프로토콜을 구현함으써 좀더 실시간성을 개선시키고 아울러 버퍼의 사용량도 최소화시켜야 할 것이다.

6. 참고 문헌

[1] 김완규, 박규석, "VOD 시스템에서 클라이언트 버퍼를 위한 전송률 제어 알고리즘의 설계 및 분석", 멀티미디어학회 논문지 제1권 제1호, 1998.6

[2] Ingo Busse, Bernd Deffner, Henning Schulrinne, "Dynamic QoS Control of Multimedia Applications based on RTP", 1995

[3] R. El-Marakby, D. Hutchison, "Integrating RTP into the World Wide Web", Proc. of W3C Workshop, Sophia-Antipolis, France, October 1996

[4] R. El-Marakby, D. Hutchison, "Delivery of Real-time Continuous Media over the Internet", Proc. of ISCC '97

[5] 김정원, 김인환, 정기동, "Linux 상에서 확장 가능한 VOD 시스템의 설계 및 구현", 한국정보처리학회 99춘계학술발표 논문집 제 6권 1호, 199.4

[6] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-time Applications", RFC 1889, January 1996.

[7] Andrew S.Tanenbaum, "Computer Networks", pp.380-381, Prentice Hall, 1996