스트리밍 서비스를 위한 RTP/RTCP 기반의 단방향 지연 측정

박진호° 김화성
광운대학교 전자통신공학과
cava9@kw.ac.kr°, hwkim@daisy.kw.ac.kr

One-way delay estimation based on RTP/RTCP for Streaming Service

Jinho Park°, Hwasung Kim
Dept. Electronic and Communications Engineering, KwangWoon Univ.

요 약

VoD(Video on Demand)나 IP 텔레폰, 화상채팅과 같은 멀티미디어 서비스는 일정한 대역폭의 자원을 요구하고 지터, 전송 지연에 매우 민감한 데이터이다. 멀티미디어 서비스의 QoS 보장과 TCP와 형평성을 위하여 TFRC, RAP, TLFC 와 같은 흐름제어 기법이 사용되고 있다. 하지만 흐름제어 기법에서 네트워크 판단을 위해서 사용하고 있는 RTT값은 서버와 클라이언트 사이의 왕복 전송 지연시간을 측정한 값으로써 클라이언트에서 서버로 전송되는 피드백 정보의 전송 지연시간에 따라서 오차가 발생한다. 본 논문에서는 서버에서 클라이언트로 단방향으로 데이터가 전송되는 특징을 가진 스트리밍 서비스를 위하여 RTP/RTCP를 이용한 단방향 전송 지연 측정을 제안한다.

1. 서 론

대부분의 멀티미디어 서비스는 UDP 프로토콜에 멀티미디어 서비스의 QoS를 제공하기 위해서 RTP,RTCP 프로토콜을 이용하여 전송한다. RTP 패킷을 수신한 호스트는 주기적으로 RTCP패킷을 송신자에게 전송한다. 보고된 RTCP 패킷을 통하여 RTT, 지터, 패킷 유실 등의 QoS를 위한 파라미터 값을 얻을 수 있다. 이 파라미터 값들은 TFRC(TCP-Friendly Rate Control), RAP(Rate Adaptive Protocol), TLFC(TCP Like Flow Control)과 같은 흐름 제어 기법에 의해 네트워크 상태를 판단하고 전송률을 조절한다[1][2][3].

하지만 RTCP 패킷을 이용하여 QoS 파라미터를 측정 시에 세 가지 문제점이 있다. 첫째는 RTT 값을 측정 시, 실제로 전송되고 데이터 패킷의 RTT 값이 아니라 주기적으로 교환하는 RTCP 패킷의 RTT 값을 구한다. 즉 정해진 간격으로 전송하는 RTCP 패킷의 RTT 값이므로 정확하게 실제적으로 주고받는 데이터의 전송 지연에 대한 것은 알 수 없다. 둘째 서버에서 클라이언트로 전송되는 방향이 순방향이고 클라이언트에서 서버로 전송되는 방향을 역방향이라고 한다면, RTT 값은 순방향과 역방향을 더한 왕복시간의 값이다. 만약 역방향에서 혼잡이 발생하거나 큐잉 delay 등이 발생하면 서버에서 측정되는 RTT 값은 증가하게 된다. 역방향에서의 경험한 지연 때문에 측정된 RTT값은 오차가 발생할 수 있다.

본 논문에서는 세 가지 문제점을 해결하기 위해 단방향 전송 지연 값을 구하기 위한 동기화 기법을 제안한다. 본 논문의 구성은 다음과 같다. 2장에서 표준 멀티미디어 스트리밍 프로토콜에 대해서 기술하고, 3장에서 단방향 전송 지연 측정 방법을 기술한다. 4장에서 모의실험을 통하여 성능평가를 하고, 마지막으로 5장에서 결론을 맺는다.

2. 관련연구

2.1 RTP : Real-Time Transport Protocol

RTP는 멀티캐스트 또는 유니캐스트 상에서 오디오, 비디오 및 시뮬레이션 데이터 와 같은 실시간 데이터를 전송하는 응용에 적합한 단대 단 트랜스포트 기능을 제공한다. 그러나 RTP는 자원 예약에 대한 내용은 다루지는 않으며, 특히 적시 데이터 전송 (Timely Delivery), QoS보장, 순차 전달과 같은 기능을 제공하지 않는다. RTP와 RTCP는 하위의 전송 및 네트웍 계층에 무관하게 설계되었다. RTP는 별개의 독립 계층으로 구현되기 보다는 특정 응용에서 요구되는 정보를 제공하여 프로토콜의 처리가 응용의 처리 과정으로 통합될 수 있도록 설계되었다. 따라서 기존의 프로토콜들과는 달리 RTP는 응용의 필요에 따라 헤더를 변경하거나 추가하여 응용에 맞는 프로토콜이 될 수 있도록 하는 일종의 맞춤형 프로토콜이다. 따라서 트랜스포트의 의미는 실시간 데이터의 특성에 중점을 두어 제정한 표준이라고 할 수 있다.

2.2 RTCP: Real-Time Control Protocol

RTCP는 컨트롤 프로토콜로써 세션에 참가한 모든 참가자에게 피드백을 주기적으로 전송한다. RTCP는 데이터 송수신 간의 분실된 패킷 수, 지터 간격, 앞의 패킷과의 지연시간 등의 QoS 정보를 교환하여 응용이 적당한 QoS를 평가하여 adaptive encoding을 제공하도록 한다. 또한 RTCP는 많은 참여자들의 스케일을 위해서 패킷 송신율을 계산하고 사용자 인터페이스의 참여자 ID를 지칭하는 최소한의 세션 제어 정보를 나른다.

RTP는 수명에서 수 천명의 참가자를 하나의 세션에

참가시킬 수 있도록 설계되었다. 오디오 회의의 경우에 데이터 패킷은 참가자의 수에 상관없이 비교적 일정한 비트율을 가지지만(언제나 발언하는 사람은 한 두 사람이므로) 제어 패킷의 경우에는 참가자의 수에 비례하여 비트율이 증가(각 참가자가 나머지 모두에게 일정한 간격으로 제어 패킷을 전송하기 때문에)하게 된다. 따라서 제어 패킷의 전송 간격은 제어되어야 한다. RTCP에 할당되는 대역폭은 세션 대역폭의 5%로 고정되는 것이 바람직하다.

3. 단방향 전송 지연 측정 방법

단방향 전송 지연 값을 구하기 위해선 서버와 클라이언트 시스템의 시간 동기화 작업이 필요하며 가장 어려운 문제이다. 이러한 문제를 해결하기 위해 TCP의 서비스를 받는 환경에서 RTT를 이용한 네트워크 모니터링 기법과 GPS(Global Positioning System)를 이용한 시스템 동기화 기법이 제안되었다[4]. 하지만 RTT를 이용한 기법은 서버와 클라이언트에서 모든 데이터에 피드백을 전송해야 한다는 오버헤드가 존재하고, GPS를 이용한 기법은 모든 클라이언트, 서버 시스템에 GPS가 장착되어있어야 한다는 경제적 부담이 있다.

본 논문에서 제안하는 알고리즘은 세션이 설정된 후 RTT를 이용하여 서버와 클라이언트의 시간을 동기화 시킴으로써 상대적인 단방향 전송 지연시간을 구하고 네트워크 상태를 판단함으로써 효율적이고 경제적인 알고리즘을 제안한다.

그림 1는 단방향 전송 지연을 구하기 위한 동기화 순서를 도식화 한 것이다. 서버를 동기화 하기 위해 RTT를 구한다. RTT를 값을 구하기 위해 Server시간을 샘플링한 시점의 시간을 ServerTime이라고 하고 클라이언트의 시간을 ClientTime이라고 한다. ClientTime은 현재 알 수 없는 미지수 이다.

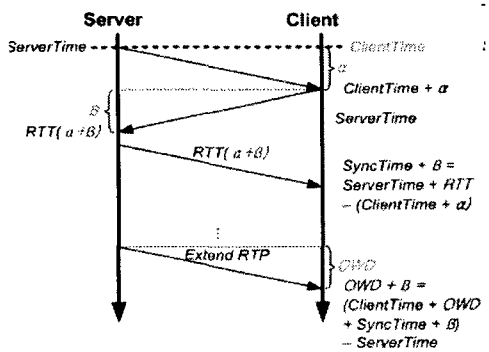

그림 1. 서버/클라이언트 동기화 순서

서버에서 클라이언트까지 걸리는 시간은 α이며 클라이언트에서 서버까지 걸리는 시간을 β라고 한다. 서버에서 ServerTime을 클라이언트로 보냈을 때 패킷이 도착하는 시점의 클라이언트 시간은 ClientTime + α 이며 ServerTime의 시간을 저장한 후 서버로 되돌려준다. 클라이언트에서 패킷을 수신한 서버는 왕복지연시간 RTT를 구할 수 있으며 RTT = α+ β 이다. RTT값을 클라이언트로 전송하면 클라이언트는 ServerTime과 RTT(α + β)값을 알 수 있다.

$$SyncTime = ServerTime - ClientTime \quad ①$$
$$(ClientTime + SyncTime) - ServerTime = 0 \quad ②$$
$$(ClientTime + OWD + SyncTime) - ServerTime = OWD \quad ③$$

위의 식과 같이 이상적인 경우 ServerTime 에서 ClientTime을 뺀 시간을 SyncTime이라 했을 때 , 단방향 전송 지연 값을 구하기 위한 공식은 식 ③과 같다. 여기서 OWD는 단방향 전송 지연(One-Way Delay)를 나타낸다. α는 OWD와 같은 값이며 Client + OWD 값은 ServerTime이 도착했을 때 시간이다. 이런 이상적인 경우가 아닌 실제 상황에선 ClientTime을 알아낼 수 있는 방법이 없다. 그렇기 때문에 정확한 SyncTime을 구할 수 없다.

$$SyncTime = ServerTime + \alpha - (ClientTime + \alpha) \quad ④$$
$$SyncTime + \beta = ServerTime + RTT - (ClientTime + \alpha) \quad ⑤$$
$$(ClientTime + SyncTime + \beta) - ServerTime = \beta \quad ⑥$$
$$(ClientTime + OWD + SyncTime + \beta) - ServerTime = OWD + \beta \quad ⑦$$

본 논문에서 제안하는 알고리즘은 다음과 같다.

식 ④는 식 ①에 α값을 더하고 뺌으로 성립함을 알 수 있다. 하지만 위 식은 식 ①과 같이 동시에 성립할 수 없으므로 양변에 β값을 더해줌으로써 식 ⑤를 구 할 수 있다. 식 ⑤의 우측에 있는 식은 우리가 알 수 있는 값이다. 이를 통해 SyncTime + β를 구할 수 있다. 식 ⑥은 식②의 양변에 β값을 더해 줌으로써 성립이 되며 이를 통해 식 ⑦이 성립함을 알 수 있다. 위 식을 사용하였을 경우 항상 OWD값 + β의 값을 구할 수 있으며 β값은 고정된 상수값이고 β값 만큼 오차가 발생한다.

5. 모의 실험

본 장에서는 단방향 전송 지연의 모의 실험과 성능 분석을 한다. 성능평가를 위해 LBNL(Lawrence Berkely National Laboratory)의 NS-2(Network Simulator)를 사용한다[5].

단방향 전송 모의실험은 그림 2과 같은 실험 환경

을 구성하여 성능 실험을 수행하였다. S1과 C1은 측정하려는 프로토콜이고 S2과 S3는 서버에서 클라이언트 쪽으로 트래픽을 발생시키는 에이전트이고 C4와 C5는 클라이언트 쪽에서 서버쪽으로 트래픽을 발생시키는 에이전트이다. 혼잡을 발생시키기 위해 가기 다른 시간에 트래픽을 발생시키며 7초간 성능을 측정하였다.

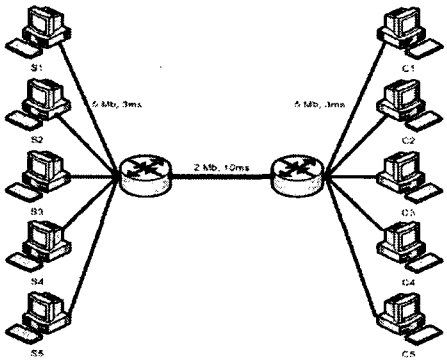

그림 2. 모의실험 환경

그림 3는 GPS를 이용하였을 경우 전송 지연 값과 단방향 전송 지연과 RTT를 이용하였을 경우의 전송지연 값을 보여준다. 그림에서 알 수 있듯이 단방향 전송 지연 값과 GPS를 이용한 모니터링 기법은 동일한 패턴의 전송 지연 값을 갖는 것을 확일 할 수 있다.

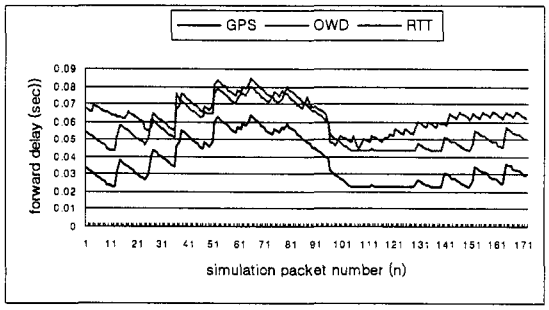

그림 3. 전송 지연 비교 분석

그림 4은 "단방향 전송 지연 값 - GPS" 와 " RTT - GPS " 값을 보여주는 그래프로 단방향 전송 지연 값은 항상 일정한 상수 β값 만큼 더 큰 값을 갖는 다는 것을 알 수 있다. 그림 3와 4을 통해 RTT 값은 클라이언트에서 서버로 전송되는 트래픽에 영향을 받아 전송 지연 값이 변화됨으로써 서버에서 클라이언트로 전송되는 스트리밍 서비스에서 시용하였을 경우 네트워크 상태를 판단함에 있어 오차가 발생 할 수 있음을 확인 할 수 있다.

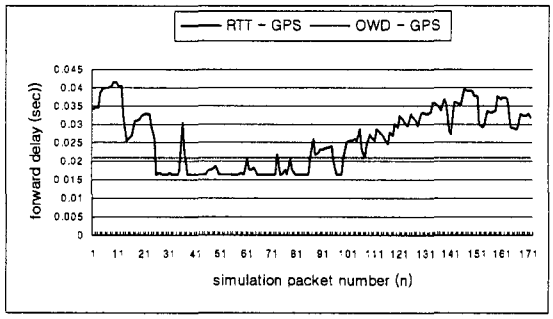

그림 4. 전송 지연

V. 결과

본 논문에서는 IETF의 RTP/RTCP프로토콜을 사용하는 멀티미디어 스트리밍 서비스 환경에서 네트워크 상태판단의 오차율을 줄이기 위한 단방향 전송 지연 측정법을 제안하였다. 단방향 전송을 이용한 모니터링 기법은 상대적인 단방향 전송 지연 값을 구하고 네트워크 상태판단에 이용함으로써 실제 단방향 전송 지연 값과 동일한 패턴의 지연 값을 얻을 수 있었고 이를 이용하여 네트워크 상태판단 오차를 줄였다. 모의실험에서 단방향 전송 지연을 이용한 모니터링 기법은 오차율을 줄이기 위한 기법을 적용하였을 경우 GPS를 이용하여 단방향 전송 지연 값을 구한 것과 동일한 성능을 나타냈으며, RTT 값 이용시 보다 더 정확한 네트워크 상태를 판단 할 수 있다.

참고문헌

[1] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications,"in Proc.ACMSIGCOMM, Stockholm, Sweden, August 2000

[2] R. Rejaie, M. Handley, and D. Estrin, "Rap: An end-to-end ratebased congestion control mechanism for realtime streams in the internet," Proc. IEEE Infocom, March 1999.

[3] Seung-Gu Na and Jong-Suk Ahn, "TCP-like flow control algorithm for real-time applications" IEEE International Conference, September 2000.

[4] G. Almes, S. Kalidindi and M. Zekauskas "A One-way Delay Metric for IPPM", IETF, RFC 2679, September 1999.

[5] URL:http://www.isi.edu/nsnam/ns/