RTSP 기반 스트리밍 서버의 성능 측정 기술

이용주° 민옥기 김학영 김명준
한국전자통신연구원, 디지털홈연구단, 인터넷서버그룹
{yongju°, ogmin, h0kim, joonkim}@etri.re.kr

Performance Evaluation Technique of the RTSP based Streaming Server

YongJu Lee° OkGee Min HagYoung Kim MyungJoon Kim
Internet Server Group, Digital Home Research Division, ETRI

요 약

There have been many streaming servers that provide a large number of contents for a user' s preference. General purpose streaming server makes use of a RTSP protocol for streaming controls such as message passing with client players. To date, there has been minimal research regarding streaming server' s performance test tools. For measuring streaming server' s performance, performance evaluation technique is needed and also achieved by RTSP based controls, a server' s performance result and its miscellaneous test tools such the PseudoPlayer for pumping data to a specified port and the PseudoMonitor for gathering information. In this paper, We implement a test toolkit for evaluating a streaming server' s performance and show the case of its application

1. 서 론

인터넷이 보편화 되어 감에 따라 다량의 콘텐트를 사용자의 구미에 맞게 보여줄 수 있는 스트리밍 서버에 대한 연구가 활발히 진행되고 있다. 일반적으로 스트리밍 서버는 RTSP 라고 불리는 표준 통신 프로토콜을 기반으로 서버가 동작할 수 있으며 성능을 측정하게 하기 위해서는 이 프로토콜에 맞게 서버의 요청을 해서 임의의 특정주소를 스트리밍 데이터를 보내게 할 수 있어야 한다. 이에 본 논문에서는 성능 측정을 위해 스트리밍 서버측면에서 요구되는 RTSP 표준에 따라 메시지 송수신 방법, 서버의 성능데이터 기록 방법과, 가상의 클라이언트 플레이어를 생성과 서버가 기록한 스트리밍 성능 데이터를 획득하는 방법에 대해 논한다.

2. 스트리밍 서버 요구사항

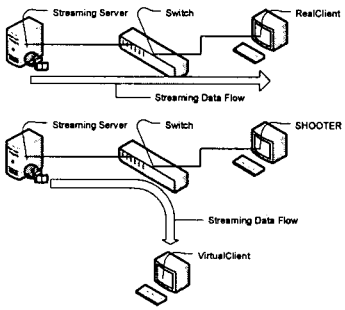
스트리밍 서버에서는 표준 RTSP에 기반한 메시지를 주고 받을 수 있어야 하며, 메시지 형태는 크게 DESCRIBE/SETUP/PLAY/PAUSE/TEARDOWN을 들 수 있다[1]. 서버 측면에서는 가상의 클라이언트로 플레이 되기 위해서는 서버에서는 초기 DESCRIBE단계/PLAY단계/PAUSE 단계/TEARDOWN단계는 실제 플레이어와 동일하게 수행되면 되지만 SETUP단계에서 클라이언트 주소를 임의의 주소를 대치하기 위해 SETUP에 Response로 정의된 Transport헤더의 수정이 필요하다. Transport헤더에는 스트리밍 프로토콜 형식/소스주소/클라이언트포트/서버포트 등이 존재하며, destination필드를 통해 클라이언트 IP가 아닌 특정 IP로 실제 데이터를 전송하게 해야만 한다. 아울러 가상의 클라이언트 플레이어 입장에서는 스트리밍 서버의 실제 데이터를 받아 보지 못하기 때문에 나중에 세션이 종료되면 서버에 TEARDOWN메시지를 보내야 한다. 하지만 클라이언트 플레이어가 데이터를 받아보기 못하기 때문에 서버의 DESCRIBE Response에 실려오는 SDP(Session Description Protocol)에서 NPT시간 정보를 획득하여 PLAY가 시작된 후 NPT 시간 동안 SLEEP한 후 TEARDOWN 메시지를 전송하게 된다

스트리밍 서버입장에서는 자체적으로 스트리밍 데이터와 기타 정보를 정의해서 나름대로 기록하는 방법이 존재할 것이다. 하지만 임의의 서버와의 호환성을 위해 스트리밍 서버의 기록을 특정 공유메모리에 적도록

함으로써 본 논문에서 이루고자 하는 실제 성능 데이터를 얻는다고 가정하면, 스트리밍 서버가 기록해 주어야만 하는 데이터는 스트림수/콘텐트 종류별 스트림수으로 할 수 있을 것이다. 이를 통해 네트워크로 전송되는 양으로부터 실제 스트림수에 따라 패킷 양이 정확한지 측정할 수 있는 것이다.

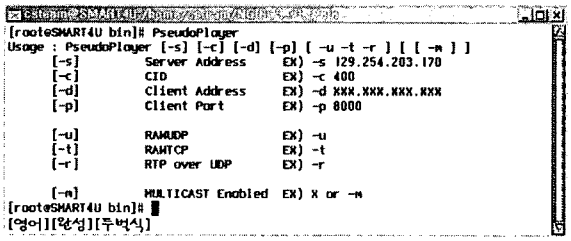
3. 구현 및 적용 사례

성능 측정을 위한 스트리밍 서버와 가상의 클라이언트 노드에서 수행하는 클라이언트 SHOOTER와의 관계는 [그림 1]과 같다.



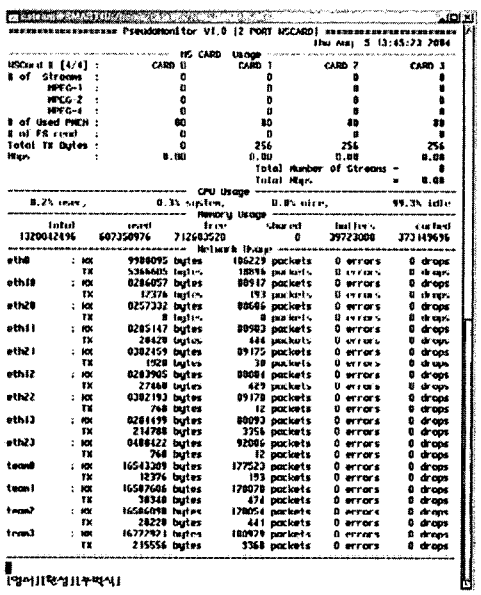
[그림 1] 전체 구조

[그림 1]에서 일반적인 스트리밍 프로토콜의 통해 이루어지는 스트리밍 방법은 스위치에 연결된 스트리밍 서버와 클라이언트가 RTSP컨트롤을 통해 스트리밍 서비스를 실제 받는 것이며, 아래 그림은 본 논문에서 성능 측정을 위해 이루어지는 방법에 대한 것을 보이고 있다. 성능 측정을 위해 가상의 클라이언트 수천 개를 생성할 수 있는 노드를 여러 개 둘 수 있으며 가상의 클라이언트 플레이어가 스트리밍 서버에 RTSP컨트롤을 통해 실제 서비스를 수행하게 된다. 하지만 서비스 된 데이터는 가상의 클라이언트들에게 보내진다고 가정하고 스위치 레벨에서 Drop시킨다. [그림 2]는 실제 구현한 PseudoPlayer에 대한 사용법이 도시된 그림이다. 가상 클라이언트 플레이어 프로그램인 PseudoPlayer에서는 가장 먼저 사용자가 주는 서버주소와 콘텐트 ID인 CID를 가지고 스트리밍 서버에 DESCRIBE를 전송한다. DESCRIBE의 Response로 받은 SDP정보를 가지고 콘텐트에 대한 NPT정보를 획득하고, 사용자에게 받은 특정 클라이언트 주소/포트를 가지고 서버에 SETUP을 전송한다. 서버에서 받은 SETUP Response를 가지고 콘텐트의 종류를 판단하여 MPEG-2와 MPEG-4에 대한 루틴을 수행하게 된다. PLAY를 전송하여 스트리밍 서버에 실제 데이터를 펌핑할수 있게 해주며, 받은 NPT동안 SLEEP을 수행한 후 TEARDOWN을 전송하여 플레이를 종료하게 된다.



[그림 2] PseudoPlayer 구현

[그림 3]에서는 특정 스트리밍 서버로 받은 스트림수와 서버의 자원 정보를 얻어서 도시한 PseudoMonitor에 한 예이다. 예에서 스트리밍을 수행하는 카드가 4개 존재하며 각각의 카드에서의 스트림수/ 메모리사용량/ 전송바이트 수/ Bandwidth가 나와 있으며 그 다음으로 CPU, Memory와 각각의 Network Device 개별 송/수신 바이트 수/패킷 수/에러 수/Drop 수가 표시된다.



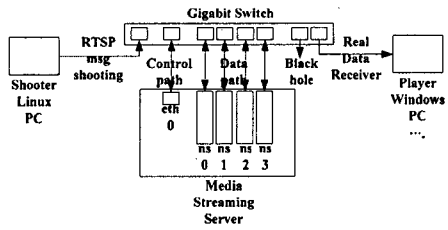
[그림 3] PseudoMonitor 구현

테스트를 위해서 특정 스트리밍 서버에 맞게 스트림수/스트림 형태에 따른 정보를 일정 간격으로 읽어서

정보를 도시해야만 한다. 시스템 자원 정보는 크게 CPU/Memory/Network정보로 대표되며 스트리밍 서버관련 정보와는 달리 자동적으로 획득되는 정보이다. PseudoMonitor에서 기록한 데이터를 롱런 테스트의 같이 장시간의 테스트 슈트로 사용하기 위해서는 PseudoMonitor에 스트리밍 서버로부터 획득한 정보와 시스템 자원 정보 모두를 하나의 파일 형태로 기록하게 된다. 아래는 그 한 예이다. 총 스트림수가 4인 경우와 3181인 경우를 나타낸 것이다.

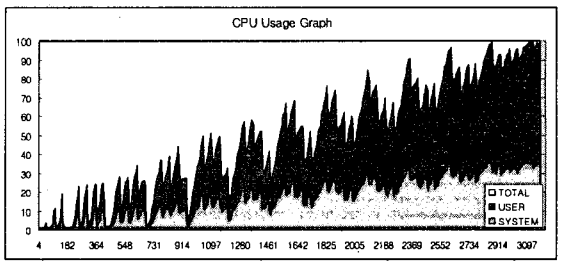
```
== PseudoMonitor V1.1 [ HW: 2 NSCard, CS: 4 NSProcess, 10 #OfNP in EachNS ]==
TotalStreams= 4 EachCARD= 1 1 Mbps= 0.47 0.29 CPU=USER 0.0 %,SYSTEM= 0.2 %,NICE= 0.0 %,IDLE= 99.7 %MEM=TOTAL 2114195456 USED 1920856064 FREE 193339392 SHARED 0 BUFFERS 74305536 CACHED 1141878784
...
TotalStreams= 3181 EachCARD= 796 796 Mbps= 1125.91 1127.30 CPU=USER 34.4 %,SYSTEM= 65.0 %,NICE= 0.0 %,IDLE= 0.4 %MEM=TOTAL 2114195456 USED 2077851648 FREE 36343808 SHARED 0 BUFFERS 74682368 CACHED 1128095744
```

본 논문에서 구현한 성능 측정 도구들의 실제 Kasenna MediaBase XMP video Server와 한국전자통신연구원 디지털홈연구단의 NSCard를 통한 차세대 인터넷 서버의 스트리밍 서버에 적용되었다[2][3]. 네트워크와 스토리지 기능을 가진 NSCard가 장착된 스트리밍 서버는 한 노드에 여러 개의 네트워크 포트를 가지고 있으며 아울러 eth0로 전송된 DESCRIBE메시지를 스트리밍 서버가 Redirect를 해서 NS카드의 IP를 통해 스트리밍을 수행할 수 있도록 한다. 이에 성능측정 도구들의 수정을 통해 해결한다. [그림 4]는 실제 테스트 환경을 나타내는 그림이다. 스위치에 마지막 포트를 Hole로 설정하고 스트리밍 서버와 연결된 스위치에 가상의 SHOOTER를 위한 머신을 두어 PseudoPlayer를 구동시킨다. 특히 스트리밍 자체에서 지원하는 스트림수가 많기 때문에 한 SHOOTER 머신 에서는 자원 부족으로 수행이 불가능하여 여러 SHOOTER머신 에서 나누어 수행하는 구조이며 실제 패킷이 전송되는 모습은 PseudoMonitor를 통한 소프트웨어적으로 측정할 수 있으며, 네트워크 Analyzer를 통한 하드웨어적으로 한 포트를 가로채서 패킷양을 볼 수 있다. 실제 전송되는 스트림을 몇 개만 보기 위해 PC클라이언트나/IP STB/Cable STB를 연결하여 여러 콘텐트를 Look and Feel형태로도 스트림의 끊김 현상을 볼 수 있다.



[그림 4] 테스트 환경

테스트 결과의 한 예로 한국전자통신연구원에서 개발한 차세대 인터넷서버의 실제 성능 측정의 한 예는 아래와 같다. 차세대 인터넷서버에 NS카드 2장을 장착하고, H.264 600K 콘텐트는 [그림 4]의 환경과 같이 구성하여 3.3절에서 언급한 PseudoMonitor를 통해 모니터링 로그를 기록한 파일에서 얻은 데이터의 그래프는 아래와 같다.



[그림 5] 실 테스트의 CPU그래프

4. 결론

본 논문에서는 고성능 스트리밍 서버의 성능을 측정할 수 있는 도구들인 가상 플레이어인 PseudoPlayer와 서버의 성능데이터를 획득하는 PsuedoMonitor 에 대해 구현 방법 및 실제 적용 사례를 기술하였다.

참고문헌

[1] H. Schulzrinne, R. Lanphier, and A. Rao, "Real time streaming protocol (RTSP)," RFC 2326, IETF, Apr. 1998

[2] http://www.kasenna.com/products/index.jsp

[3] Y.J. Lee, O.G. Min, S.J. Mun, H.Y. Kim, "Enabling High Performance Media Streaming Server on Network Storage Card", Internet and Multimedia Systems and Applications, IASTED, Aug, 2004