



## 저작자표시-비영리-동일조건변경허락 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.
- 이차적 저작물을 작성할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



동일조건변경허락. 귀하가 이 저작물을 개작, 변형 또는 가공했을 경우에는, 이 저작물과 동일한 이용허락조건하에서만 배포할 수 있습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

공학석사 학위논문

# HTTP기반의 크로스레이어 적응형 스트리밍 프레임워크 제안

Cross-Layer Adaptive Streaming over HTTP

2016년 2월

서울대학교 대학원

컴퓨터공학부

권 윤

# HTTP 기반의 크로스레이어 적응형 스트리밍 프레임워크 제안

지도교수 권 태 경  
이 논문을 공학석사학위논문으로 제출함

2016년 1월

서울대학교 대학원  
컴퓨터 공학부  
권 윤

권 윤의 석사학위논문을 인준함  
2016년 1월

|      |       |     |
|------|-------|-----|
| 위원장  | 김 종 권 | (인) |
| 부위원장 | 전 화 숙 | (인) |
| 위원   | 권 태 경 | (인) |

## 초 록

최근 모바일 기기 보급의 증가와 인터넷 네트워크 인프라의 확대 및 글로벌 비디오 스트리밍 서비스의 등장으로 비디오 콘텐츠에 대한 수요가 폭발적으로 증가했다. 그 결과 현재 전체 인터넷 트래픽 중 스트리밍이 차지하는 비율이 70%이상을 차지하고 있으며 이러한 추세는 점차 강화되고 있다. 이러한 상황에서 몇 년 전부터 사용자의 네트워크 상황을 예측하여 이에 맞춰 최적의 화질을 제공하는 HTTP 기반의 적응형 스트리밍에 대한 연구가 활발히 이루어져 왔고, 비디오와 오디오 표준화를 위한 워킹 그룹인 MPEG(Moving Picture Expert Group)에서는 DASH (Dynamic Adaptive Streaming over HTTP) 기법을 표준화 하였다. 다양한 연구를 통해 클라이언트 측에서 알고리즘을 통해 좀 더 정확하게 네트워크 상황을 예측하려는 시도가 있어왔지만 근본적으로 DASH 가 TCP 기반의 HTTP 위에서 동작하기 때문에 그 성능이 TCP 의 특성에 크게 종속될 수 밖에 없다. 그러나 TCP 의 일차적인 목표는 신뢰성 있는 통신으로, 사용자 QoE(Quality of Experience)라는 DASH 의 최종 목표와는 상충되는 경우가 존재한다. 따라서 본 연구에서는 클라이언트의 TCP 레이어와 DASH 플레이어간 동작하는 어플리케이션 레이어 간의 크로스레이어 적응형 스트리밍 프레임워크인 CLASH 를 제안 하고자 한다. 이를 통해 패킷 손실과 같이 TCP 에 영향을 주는 문제가 발생했을 때에도 스트리밍의 품질에 영향을 최소화 할 수 있도록 하는 시스템을 구현하였고, 실험을 통해 이를 검증해 보았다.

주요어 : 비디오 스트리밍, HTTP 기반 적응형 스트리밍, DASH, TCP,  
크로스 레이어, QoE

학 번 : 2014-21781

# 목 차

|                                       |    |
|---------------------------------------|----|
| 초 록 .....                             | 1  |
| 목 차 .....                             | 3  |
| 제 1 장 서 론 .....                       | 4  |
| 제 1 절 연구의 배경 .....                    | 4  |
| 제 2 절 논문의 구성 .....                    | 6  |
| 제 2 장 기존 연구의 한계점 .....                | 7  |
| 제 1 절 전송 프로토콜 (TCP) .....             | 7  |
| 제 2 절 HTTP 적응형 스트리밍 프로토콜 (DASH) ..... | 8  |
| 제 3 장 CLASH 프레임워크 .....               | 11 |
| 제 1 절 사용자 상황을 고려한 크로스 레이어 스트리밍 .....  | 11 |
| 제 2 절 CLASH 의 구성요소 및 동작 원리 .....      | 14 |
| 제 3 절 시스템의 구현 .....                   | 17 |
| 1. 패킷 모니터링 시스템 .....                  | 17 |
| 2. 클라이언트 DASH 알고리즘 .....              | 19 |
| 3. 버퍼 모니터링 기법 .....                   | 19 |
| 4. Raw 소켓을 이용한 로스 하이드 기법 .....        | 20 |
| 제 4 장 시스템 분석 및 평가 .....               | 21 |
| 제 1 절 실험 환경 .....                     | 21 |
| 제 2 절 성능 평가 .....                     | 23 |
| 제 3 절 결과 논의 .....                     | 27 |
| 제 5 장 결 론 .....                       | 29 |
| 참고 문헌 .....                           | 31 |
| Abstract .....                        | 33 |

# 제 1 장 서 론

## 제 1 절 연구의 배경

오늘날, 네트워킹 기술이 발달함에 따라 온라인 상에서 다양한 방법으로 콘텐츠의 공급, 유통이 가능해졌으며, 시간 및 공간의 제약없이 인터넷 사용자들의 콘텐츠 소비가 가능해지게 되었다. 이러한 콘텐츠 공급 및 유통의 기술이 지속적으로 발전함에 따라 인터넷 사용자들은 사진, 비디오 등 다양한 멀티미디어 콘텐츠에 대한 소비 요구를 가지게 되었고, 그 중에서도 실시간 방송 등 비디오 스트리밍 서비스 (Streaming Service)에 대한 수요가 증가하게 되었다. 그 결과, 현재 비디오 및 스트리밍 관련된 트래픽이 전체 인터넷의 70% 이상을 차지하게 되었으며, 이 추세는 더욱 심화될 것으로 예상된다. [1]

이러한 수요의 증가에 따라, 비디오 스트리밍 서비스에서 사용자의 만족도 (Quality of Experience, QoE)를 최대화하는 등의 효율적 스트리밍에 대한 연구가 진행되어 왔다. 특히 최근에는 콘텐츠 전송 트래픽 최소화 등 효율적인 전송을 위한 기술 연구를 넘어, 사용자의 네트워크 상태, 서비스 비디오 품질 등 사용자 상황을 고려한 연구가 활발하게 진행되어 왔다. [2] 그 연구 결과 중 하나로 HTTP 기반 동적 적응형 스트리밍 (Dynamic Adaptive Streaming over HTTP, 이하 DASH) 기술이 개발되었으며 [3], MPEG 그룹에서 공식적으로 표준화 하는 등 관련 기술 연구 및 개발이 꾸준히 진행되고 있다. DASH 기술을 통하여 사용자는 현재의 네트워크 상태 및 요구 만족도에 따라 동적으로

비디오 콘텐츠의 품질을 요청하여 원하는 품질의 비디오를 시청함으로써 만족도를 최대화 할 수 있다.

하지만, 이러한 DASH 스트리밍 기법은 전송 제어 프로토콜 (Transmission Control Protocol, 이하 TCP) 및 HTTP 상에서 동작하기 때문에 TCP 의 신뢰성 있는 통신 (Reliable Transmission) 동작 과정의 범위 내에서만 사용자의 만족도를 극대화할 수 있다는 한계가 있다. 실제로, 소량의 패킷 손실 (Packet Loss)이 발생한 상황에서, 현재의 DASH 는 신뢰성 있는 통신을 위해 TCP 에서 재전송 (Retransmission) 등의 동작을 수행하고, 이는 패킷 처리량 (Throughput)의 감소로 이어진다. 결과적으로 패킷 처리량의 감소는 DASH 에서 저품질 비디오를 선택하게 하도록 유도함으로써 사용자의 서비스 만족도를 감소시키는 DASH 의 근본적인 구조적 문제가 발생한다.

이와 같은 문제는 DASH 알고리즘과 TCP 동작이 스트리밍 서비스 과정에서 상호 영향이 있음에도 불구하고 독립적으로 동작해야 하는, 인터넷의 계층적 구조로 인한 문제라 할 수 있으며, 계층 구조를 없애는 크로스레이어 (Cross Layer) 방식을 통해 사용자의 만족도를 더욱 극대화 할 수 있다. 다시 말해서, 일시적인 패킷 손실 상황은 경우에 따라 사용자의 비디오 시청에 미치는 영향이 적을 수 있기 때문에, 이러한 경우 TCP 의 재전송 동작을 수행하지 않음으로써 DASH 동작에서의 비디오 품질을 유지하는 방법으로 사용자의 만족도를 더욱 높일 수 있다. 하지만 이와 같은 해결책이 존재할 수 있음에도 불구하고 계층 구조가 가지는 독립성의 이점 때문에, 많은 연구가 진행되고 있지 못하다. 따라서, 본 연구에서는 적응형 스트리밍을 이용하는 사용자의 만족도를 높이기 위하여 크로스 레이어

방식을 통한 TCP 최적화 프레임 워크를 제안하고, 실험을 통하여 기존의 DASH 가 가지고 있는 사용자 만족도 향상의 한계를 넘을 수 있는 가능성을 제시하고자 한다.

## 제 2 절 논문의 구성

본 논문의 구성은 다음과 같다. 2 장에서는 본 연구에 필수적 이해 요소인 DASH 와 TCP 의 스트리밍 서비스에서의 동작과정 및 한계점에 대해 간략히 설명하고, 3 장에서는 본 논문에서 제시한 크로스 레이어를 통한 TCP 최적화 프레임워크의 설계 및 구현에 대해 설명한다. 4 장에서는 실험을 통해 TCP 와 DASH 를 함께 고려하는 것이 사용자의 만족도를 높이는 것임을 검증한다. 5 장에서는 연구의 한계점과 향후 연구계획에 대해 논한다.

## 제 2 장 기존 연구의 한계점

DASH 프로토콜은 계층적 네트워크 구조에 따라 TCP 상에서 동작하며, 이에 따라 DASH 서비스 성능이 영향을 받는다. 그럼에도 불구하고, 계층적 구조의 특성상 DASH 프로토콜은 TCP 와 독립적으로 동작해야만 하기에, 사용자 만족도 향상에 구조적 한계점이 존재한다. 본 장에서는 스트리밍 서비스 시 DASH 및 TCP 의 동작과정에 대해 설명하고, 계층적 구조가 같은 한계점에 대해 설명한다.

### 제 1 절 전송 프로토콜 (TCP)

전송 프로토콜은 네트워크 상의 두 종단 간 통신 (End-to-End Communication)에 있어 패킷 순서 변경 (Reordering) 및 손실 상황에 대한 신뢰성 확보, 혼잡 제어, 흐름 제어기능을 수행하기 위한 목적으로 설계되었으며, 네트워크의 전송 계층 (Transport Layer)에 위치하고 있다. [4] 전송 프로토콜은 패킷을 단위로 하여 손실 상황에 대해 재전송 (Retransmission)을 수행하고, 혼잡 상황을 예측하여 전송하는 패킷의 양을 조절하는 방식으로 위의 기능을 수행한다. [5] 현재 대다수의 응용 계층 프로토콜이 전송 프로토콜을 이용하고 있으며, DASH 또한 위의 전송 프로토콜을 이용하는 HTTP 상에서 동작한다.

DASH 알고리즘을 수행할 때, TCP 는 다음과 같이 동작한다.

TCP 에는 네트워크상의 부하로 인해 패킷 손실이 발생하는 것을 줄이기 위하여 네트워크 혼잡 제어를 사용한다. [5] 혼잡을 조절하기 위해 TCP 에서는 각 연결에 대해 혼잡 윈도우를 관리하며, 혼잡 윈도우를 통하여 패킷을 보내는 측에서 안전하게 보낼 수 있는 패킷의

수를 알고 있고, 패킷이 잘 도착하면 ACK 패킷을 받는다. 즉 이전에 보낸 패킷이 잘 도착되었다는 것을 ACK 패킷을 받은 것으로 알 수 있고, ACK 패킷을 받으면 안전하게 새 패킷을 더 보낼 수 있기 때문에 TCP 의 혼잡 제어를 셀프클록 방식이라고 한다. TCP 혼잡 제어에서 전송된 패킷에 대한 ACK 패킷의 수신여부로 혼잡 윈도우의 크기가 결정되며, 만약 해당 패킷에 대한 ACK 을 받을 경우 혼잡 윈도우 크기를 1 씩 증가시켜가면서 전송하고, 패킷 전송을 실패하거나 일정한 시간을 넘으면 혼잡 윈도우 크기를 절반으로 줄이는 AIMD(Additive Increase/Multiplicative Decrease) 방식을 기본으로 한다. TCP 에서의 성능 평가는 단위시간 당 처리 패킷의 수 및 수신된 패킷간의 지연시간을 기준으로 이루어진다.

## 제 2 절 HTTP 적응형 스트리밍 프로토콜 (DASH)

HTTP 적응형 스트리밍 프로토콜 중 DASH 는 실시간으로 변하는 단말의 상황에 따라 서비스의 품질을 동적으로 변화하여 사용자의 만족도를 높이기 위해 설계되었으며, 동작과정은 서버의 역할과 클라이언트의 역할로 나누어 볼 수 있다. [3] 서버 측에서는 동일한 비디오 콘텐츠를 다양한 화질로 인코딩한 후 일정 크기로 나누어 HTTP URL 을 매핑시켜 업로드를 한 후, 각 나누어진 비디오 세그먼트 정보를 기술한 Media Presentation Description(이하 MPD)를 클라이언트에 제공한다. 클라이언트 측에서는 MPD 파일을 우선적으로 받은 다음 이것을 바탕으로 자신의 단말상황에 적합한 비디오 세그먼트를 해당 URL 로 HTTP 요청 메시지를 통해 받아온다. 이 때

요청하는 주기는 한 비디오 세그먼트를 받은 직후가 되며 받은 비디오 세그먼트를 순차대로 재생한다.

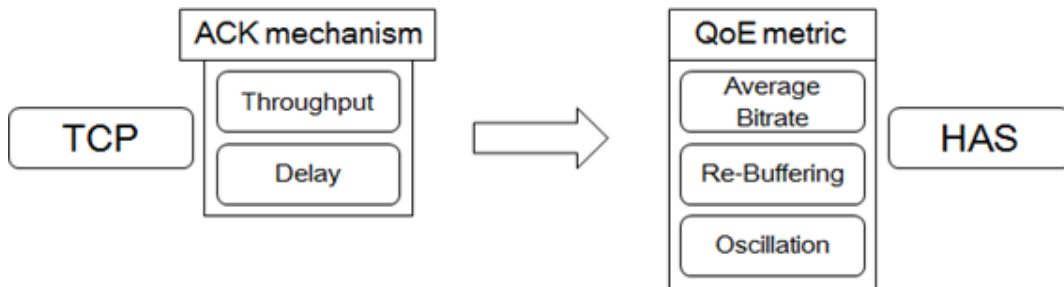


그림 1. TCP 와 HAS 의 성능평가지표 차이 및 평가지표간의 관계

비디오 스트리밍에서 서비스 성능평가를 위해 객관적이고 주관적인 요인들로 결정이 되며, [6] 객관적인 요인들은 네트워크 QoS (Quality of Service)와 어플리케이션 QoS 로 구성된다. [7] 네트워크 QoS 의 경우는 Round-trip time (RTT), 지터와 패킷 로스 rate 등으로 이루어지며, 어플리케이션 QoS 는 비디오 비트레이트, 비디오 화질 등으로 구성이 된다. 반면, 주관적인 요인들로는, 정량화하기 힘들기는 하나 화질의 변화횟수, 버퍼링 시간, 끊김 횟수, 스크린의 크기 등이 있다. [8, 9] 이 중 적응형 스트리밍에서 사용자 만족도를 위해 주로 사용되는 평가지표로 평균 비디오 화질, 평균 화질 변화, 버퍼링 시간을 사용하고 있다.[10] 적응형 스트리밍 서비스를 위해 표준화된 DASH 는 TCP 의 성능에 큰 영향을 받는 구조적인 네트워크 계층 관계가 있지만 위와 같이 상이한 목적으로 인해 독립적으로 동작하고 있어 더 높은 품질의 스트리밍 서비스를 제공하는데 있어 한계가 있다. 본 한계를 극복하기 위하여 두 프로토콜의 같은 목적을 공유하고 서로의 동작과정의 유기적 관계와 소통을 할 필요성이 있다. 따라서 적응형 스트리밍을 이용하는 사용자의 만족도를 높이기 위해서 두 프로토콜 네트워크 계층을 하나의

계층으로 보는 크로스 레이어 통한 접근을 통하여 현재 적응형 스트리밍의 한계를 극복하고자 한다.

## 제 3 장 CLASH 프레임워크

### 제 1 절 사용자 상황을 고려한 크로스 레이어 스트리밍

TCP 에서의 동작과정이 스트리밍 성능과 긴밀한 관계가 있지만 사용자의 스트리밍을 위한 단말의 상황은 어플리케이션 계층에서 확인이 가능하나 어플리케이션 계층과 전송 계층은 독립적으로 작동하기 때문에 기존 적응형 스트리밍 성능을 높이는 것에 있어 한계가 있다. 따라서 비디오 스트리밍 서비스를 이용하는 사용자의 만족도를 높이기 위해 네트워크 계층 중 어플리케이션 계층과 전송 계층을 동시에 상황을 파악하고 작동할 수 있는 크로스레이어 모델인 CLASH(Cross Layer Adaptive Streaming over HTTP)프레임 워크를 제안하고자 한다.

본 연구에서 제안하고자 하는 크로스레이어 모델은 어플리케이션 계층에 속하는 DASH 프로토콜과 전송 계층의 TCP 를 동시에 고려하고 있다. 각 프로토콜의 요구사항과 필요자원 등의 불일치 또는 부조화(Mismatch)로 인해 존재할 수 있는 성능의 한계가 존재하며 이것을 극복하기 위해 각 프로토콜간의 경계를 뛰어넘는 범 프로토콜 적 스트리밍 서비스를 목표로 하고 있다.

본 모델의 기본적인 동작과정은 서비스 받고 있는 단말의 스트리밍 상황에 따라 동적인 TCP 알고리즘으로 구성되어 있다. 이 중 단말의 스트리밍 상황으로 고려할 수 있는 것들로는 현재 남아있는 버퍼의

잔여량, 네트워크 대역폭 상황 등이 있다. 기존 TCP에서는 단말의 상황으로 패킷의 전송 실패 등을 고려하고 있다.

먼저 스트리밍 상황으로, 적응형 스트리밍에서는 끊임 없는 서비스를 위하여 버퍼 상황을 중요하게 고려한다. [11] 만약 사용자의 잔여 버퍼량이 많이 남을 경우 요금의 낭비 및 단말에서 할당받은 대역폭의 사용량이 부족하여 높은 화질의 서비스를 제공하는데 문제점이 있으며, 반대로 부족할 경우 스트리밍 서비스에 리버퍼링 현상이 발생할 수 있다. 사용자의 만족도에 리버퍼링 현상은 가장 사용자 만족도 평가에 저해되는 것으로 알려져 있다. 따라서 제안하고자 하는 모델에서도 스트리밍 시 단말의 상황을 잔여 버퍼량에 통하여 접근하고자 한다.

스트리밍 서비스 받는 단말의 잔여 버퍼상황을 긴급(Emergency) 상태와 안정(Stable) 상태로 나누어 다른 동작을 하고자 한다. 먼저 긴급 상태는 현재 단말이 확보하고 있는 잔여 버퍼량이 부족한 상황을 뜻한다. 이 때 부족한 단말의 버퍼 잔여량은 끊임 현상을 유발할 수 있으므로 우선적으로 높은 대역폭을 할당받을 필요가 있다.

TCP에서는 신뢰성 있는 통신을 목표로 하기 때문에 클라이언트 측 단말에서 수신실패가 성능에 가장 큰 영향을 준다. TCP에서는 패킷을 단위로 통신하고 있으며, 클라이언트에서 패킷의 수신 여부는 전송이 완료되는 패킷들의 시퀀스 번호를 통해 확인할 수 있다.

따라서 본 모델에서는 고려하는 상황으로 응용 계층과 전송 계층 모두를 사용하고자 하며, 응용 계층으로 DASH 프로토콜에서 단말의 상황의 지표인 잔여 버퍼량 그리고 전송 계층에서는 TCP에서의 패킷 수신 여부를 사용하고자 한다.

각 계층의 상황에 따라 다르게 사용되는 TCP 알고리즘은 일반 TCP 와 비재전송 알고리즘으로 나누어져 있다. 이 중 비재전송 알고리즘은 단말 측에서 일부 패킷이 수신되지 않았지만 재전송을 하지 않는 방식으로, TCP 의 혼잡 윈도우의 크기를 지속적으로 높여 높은 대역폭을 할당받을 수 있다. 하지만 수신되지 않는 패킷은 데이터 자체의 신뢰성에 영향을 준다는 문제점이 있다. 일반 어플리케이션과 달리 비디오 스트리밍에서는 물론 이것이 비디오 화질을 낮추는 원인이 되지만 FFMPEG 에서 제공하는 에러보간법이 있어 그 피해를 줄일 수 있으며 화질에 대한 사용자의 만족도 보다 리버퍼링에 대한 사용자의 만족도가 더 영향이 있으므로, 버퍼상황이 안정적이지 않을 경우 이 방법을 택하고자 한다. 버퍼 상황이 좋을 경우는 재전송을 하여 화질의 저하를 감소하고자 한다.

이 프레임워크는 선택적으로 높은 대역폭을 할당받는 방식이다. 기존 적응형 스트리밍 비트레이트 결정 알고리즘에서는 이전의 대역폭을 바탕으로 앞으로의 대역폭 상황을 이용하거나 단말에 수신된 버퍼 잔여량의 상황으로 화질을 선택한다. 따라서 선택적으로 높은 대역폭을 할당받아 앞으로의 대역폭을 높게 예측을 하여 적응형 스트리밍 비트레이트 결정 알고리즘은 높은 화질을 선택할 수 있으며, 버퍼량이 적을 때 우선적으로 버퍼 잔여량을 빠르게 채우기 때문에 역시 끊김이 적으면서 동시에 높은 대역폭을 단말의 상황에 따라 선택적으로 할당받아 높은 화질의 스트리밍 시청이 가능할 것으로 기대한다.

## 제 2 절 CLASH 의 구성요소 및 동작 원리

본 절에서는 CLASH 프레임워크를 구성하는 모듈들에 대해 제시하고 이 모듈들이 어떻게 동작하는지에 대해 설명한다. 그림 2 는 구성도를 통해 전체적인 모듈과 동작원리를 나타내고 있다.

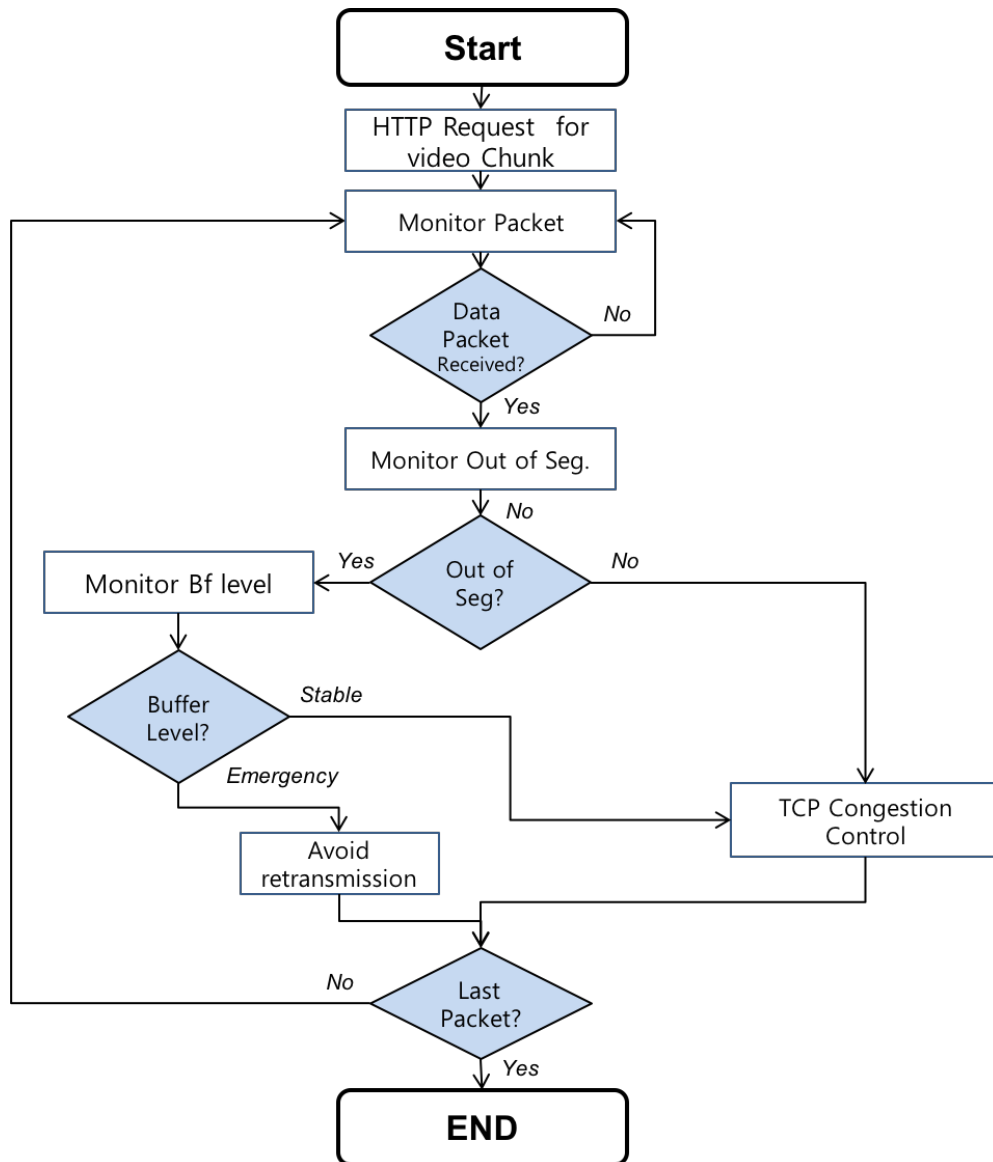


그림 2. CLASH 구성요소 및 동작원리

HTTP Request for video Chunk 는 단말이 비디오 MPD 를 바탕으로 적응형 스트리밍 비트레이트 결정 알고리즘을 통해 결정된 비디오

세그먼트를 요청하기 위해 비디오 서버로 HTTP 요청 메시지를 전송하는 모듈로, 매 비디오 세그먼트의 전송이 완료될 때 마다 실행이 된다.

Monitor Packet 은 패킷의 수신을 관찰하는 모듈로, 한 비디오 세그먼트를 구성하는 다수의 패킷이 서버로부터 도착하는데, 패킷이 도착할 때 마다 그 수신 여부를 알려준다. Monitor Out of Segment 는 TCP 측에서 패킷 수신시 전송에 있어 에러를 감지하며 수신에 실패한 패킷의 정보를 알려주는 모듈로, 그 방법은 수신되는 패킷들의 시퀀스 번호를 이용한다. 패킷들은 MTU(Maximum Transmission Unit)의 크기에 따라 시퀀스 넘버가 변화한다. 따라서 만약 수신된 패킷이 한 MTU 사이즈보다 클 경우 그 중간에 패킷 손실이 일어났다고 감지한다.

Monitor Bf level 은 현재 스트리밍 서비스를 받고 있는 단말의 버퍼 상황을 감지하는 모듈로 특정 임계점을 기준으로 더 많은 잔여 버퍼량이 있을 경우 안정(Stable)하다고 판단하고, 그렇지 않을 경우에는 긴급(Emergency)상태라고 판단한다. 이 모듈에서 임계점을 낮게 잡을수록 패킷 미수신에 대한 화질저하의 문제는 감소될 수 있으나 버퍼링 및 요청하는 비디오 세그먼트의 화질은 낮아질 수 있다. 반면 임계점을 높게 잡으면 높은 화질의 세그먼트를 요청하며 끊김 현상은 줄어들 수 있으나, 미수신 패킷에 의해 손상된 프레임에 적용되는 에러보간 연산이 필요하며 높은 혼잡 윈도우의 크기 때문에 더 큰 네트워크 혼잡을 불러일으킬 수 있다. 따라서 임계점을 적절히 정하는 것이 본 프레임워크의 성능 개선에 있어 중요하다.

Avoid Retransmission 은 특정 조건이 주어졌을 경우 전송에 실패한 패킷을 요청하지 않는 모듈로, 그 방법으로 마지막으로 수신된 패킷의

시퀀스 번호와 그 이전의 시퀀스 번호를 비교하여, MTU 크기를 이용하여 전송 실패한 패킷의 시퀀스 번호를 파악한 후 이 패킷에 맞는 시퀀스 넘버에 맞는 ACK(ACKnowledgement)를 생성하여 보내어 서버로부터의 재전송을 피한다. TCP Congestion Control 은 기존 운영체제 측이 택하는 TCP 를 뜻한다.

CLASH 의 각 모듈은 DASH 프로토콜에서 동작하는 스트리밍 상황의 정보와 TCP 에서의 패킷 전송 상황에 따라 다르게 동작을 한다.

먼저 클라이언트는 HTTP Request for video chunk 모듈을 이용하여 비트레이트 결정 알고리즘이 선택한 비디오 세그먼트를 HTTP 요청 메시지를 이용하여 받아온다. 요청 후 Monitor Packet 모듈에서 수신되는 패킷을 지속적으로 관찰하며, 패킷이 도착하였을 경우 수신된 패킷 중에 손실된 패킷이 있는지 여부와 어떤 시퀀스 번호를 가진 패킷이었는지를 Monitor out of Segment 모듈을 통해 파악하고, 만약 손실된 패킷이 없으면 운영체제에서 택한 TCP Congestion Control 을 실행하며, 손실된 패킷이 있을 경우 현재 단말의 버퍼상황을 Monitor Bf level 모듈을 통해 관찰한다. 현재 단말의 버퍼 상태가 안정적이면 일반 TCP Congestion Control 을 실행하며, 긴급 상태일 경우는 Avoid Retransmission 을 이용하여 할당되는 대역폭을 높여 버퍼를 우선적으로 빠르게 받아와서 끊김 현상을 피한다. 이 동작과정은 매 패킷에 대해 반복적으로 해당이 되며, 만약 한 비디오 세그먼트에 대한 패킷을 모두 수신하였을 경우 종료하고 다음 비디오 세그먼트에 대한 요청을 새롭게 한다.

## 제 3 절 시스템의 구현

본절에서는 제 2 절에서 정의한 모듈들을 바탕으로 실제 프레임워크를 구현하였는데, 기존 DASH 시스템에서 특별히 추가된 시스템들에 대해 설명하고자 한다. 그림 3 은 CLASH 프레임워크의 아키텍처로, 콘텐츠 서버로 보내는 리퀘스트와 그로부터 받아진 비디오 패킷이 오고 가는 과정에서 어떠한 시스템을 거쳐서 가는지를 보여주고 있다. CLASH 프레임워크의 핵심 모듈은 버퍼 모니터, 패킷 모니터, 패킷 핸들러의 세가지 부분으로 구성되어있다.

### 1. 패킷 모니터링 시스템

CLASH 프레임워크는 기본적으로 클라이언트 측에서 받아오고 내보내는 모든 패킷 데이터의 흐름을 실시간으로 감지하고 제어할 수 있어야 한다. 따라서 이를 위해 TCP 소켓을 통해 특정 IP 와 포트를 통해 통신하는 모든 패킷을 다루기 위해 linux 기반의 libnetfilter queue 라이브러리를 사용하였다. 이 라이브러리는 그림 4 와 같이 커널단에서 들어오는 모든 패킷에 대해 특정 룰을 적용해서 그 룰에 해당하는 패킷을 리다이렉트 하여 사용자 정의한 네트워크 어플리케이션을 통해 패킷을 받아 들어거나 드롭시키거나를 결정 할 수 있도록 해준다. 따라서 클라이언트에서는 관리자 권한으로 이러한 룰을 비디오가 실행하는 동안 설정하여, 기본적인 패킷의 루트를 CLASH 를 거치도록 하였다. 이때, 패킷의 시퀀스 번호와 ACK 번호, 패킷 사이즈를 기록 해서, 패킷 손실과 재전송, 그리고 로스 하이드 정보, 도착 시간의 간격, 패킷의 타입과 패킷 처리 히스토리등의 정보를

분류하고 저장하여, CLASH 의 알고리즘이 적용되어야 할 시점에 판단의 기준을 만든다.

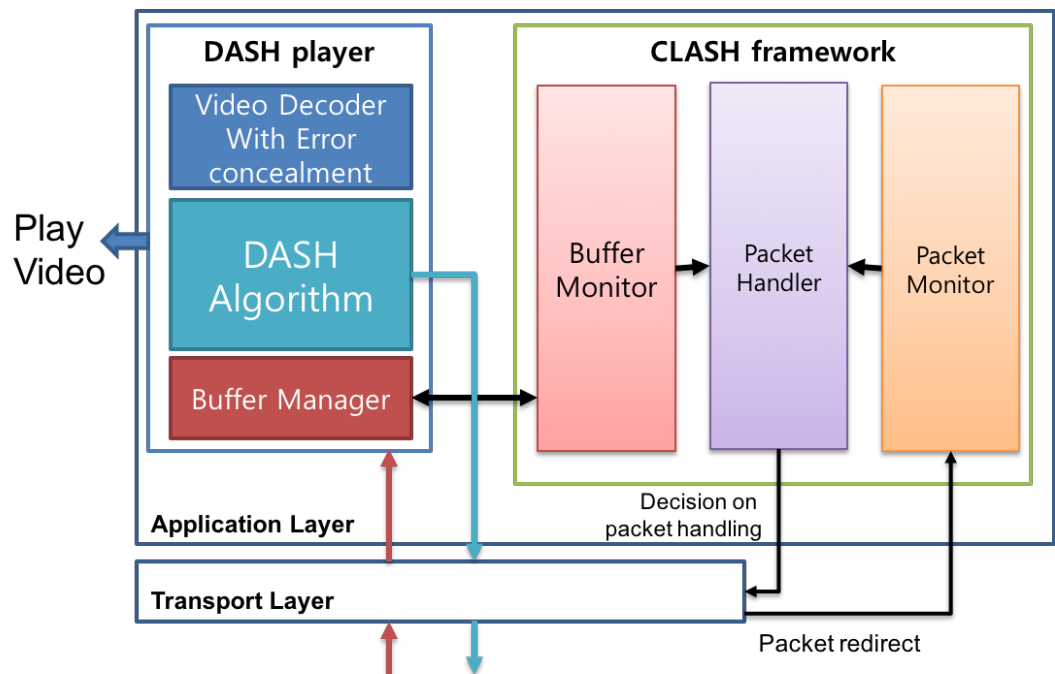


그림 3 CLASH 프레임워크 아키텍처

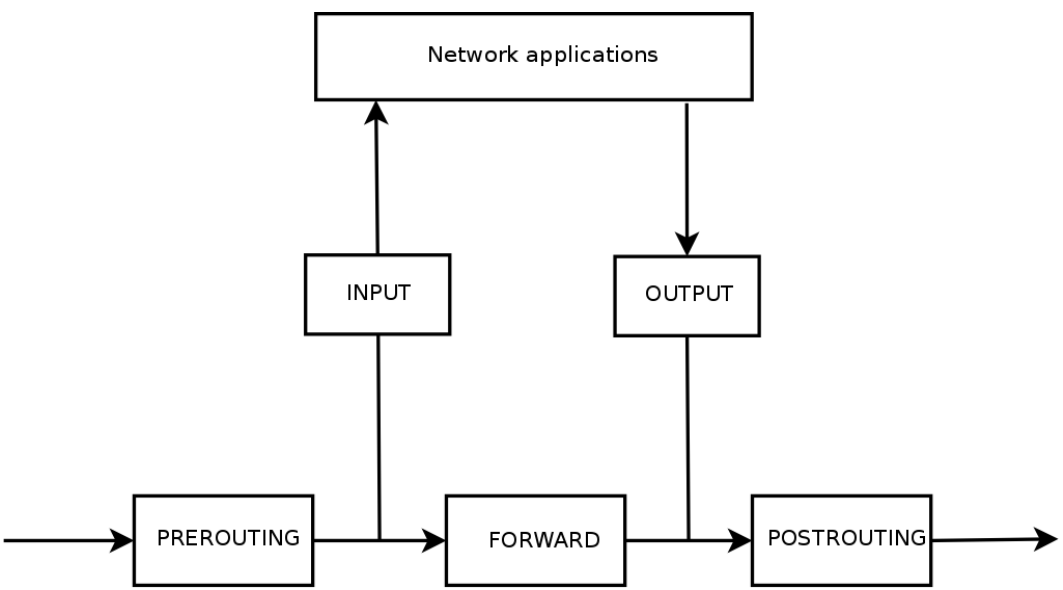


그림 4 Netfilter 체인의 호출 구조

## 2. 클라이언트 DASH 알고리즘

기존 연구에서 클라이언트 측 DASH 알고리즘은 아주 많이 제시되었지만 2012 년 CoNEXT 에서 발표된 FESTIVE 알고리즘[2]은 공정성(Fairness), 효율성(Efficiency), 안정성(Stability) 측면에서 상당히 뛰어난 알고리즘으로, 네트워크 밴드위트 예측값을 변수로 다음 요청시 비트레이트를 결정하는데, 결정된 비디오의 비트레이트가 높아질수록 다음 비트레이트를 더 높이는 데에는 보수적으로 결정을 하는 특성이 있다. 본 연구에서도 클라이언트 측 DASH 알고리즘으로 FESTIVE 를 선택 하였는데, 알고리즘이 버퍼상태를 고려하지 않고, 상대적으로 보수적인 결정을 통해 비트레이트를 결정하기 때문에, 다른 알고리즘에 비해서 다른 변수의 영향을 통제하고 TCP 상황에 따른 변화값을 확인할 수 있었기 때문이다.

## 3. 버퍼 모니터링 기법

앞서 패킷 모니터링 시스템에서 패킷 흐름에 대한 정보들을 수집했다면, 클라이언트의 웹브라우저에서 동작하는 플레이어에는 플레이 버퍼 상태를 가지고 있다. 1 초 단위로 비디오 화면으로 소모되며, 패킷 데이터를 받으면 이를 바이트 스트림으로 연결해 디코딩하여 재생하게 된다. 본 연구에서는 이 버퍼 레벨이 특정 기준 이하에 다다르게 되면 리버퍼의 위험이 큰 상황으로 특별히 긴급 상태(Emergency)로 구분하여 해당 버퍼레벨의 상태를 버퍼 레벨이 변화할때마다 실시간으로 패킷 모니터링 시스템에 웹소켓을 통하여 공지하도록 하였다. 패킷 모니터링 시스템에서는 이 정보를 토대로

긴급상태일때에는 다음 절에서 설명할 로스 하이드 기법을 이용하여 손실 상태를 숨김으로서 현재 스루풋이나 버퍼가 긴급 상태에서부터 빠르게 벗어나도록 한다.

#### 4. Raw 소켓을 이용한 로스 하이드 기법

CLASH 에서는 앞서 설명한 바와 같이 버퍼레벨이 긴급상태에 도달하였을때는, 가장 높은 우선순위는 해당 상태에서 벗어나는 것이다. 따라서 패킷 모니터가 버퍼 모니터로부터 버퍼 레벨이 긴급 상태임을 확인하면 그 때 발생한 패킷 손실에 대해 해당 데이터를 더미 데이터를 채워 Raw 소켓을 이용해 보내게 된다. 이때 더미 데이터는 패킷간 이를 통해 재전송이 발생하기 전에 ack 패킷이 전송됨으로써 서버측에서 혼잡윈도우의 크기를 줄이지 않게 되어 비디오 스트리밍에서 네트워크 측면의 성능저하를 피할 수 있게 되어 리버퍼 상황과 스루풋 저하를 피할 수 있게 된다. 이때 Raw 소켓을 사용한 이유는 모든 리눅스 시스템에서 커널레벨을 고치지 않고도 해당 프레임워크를 사용할 수 있도록 배포가능성을 고려한 결정이었다.

## 제 4 장 시스템 분석 및 평가

본 장에서는 DASH 시스템을 통해 스트리밍을 사용할 때 패킷 손실에 따라 클라이언트의 스루풋 예측과 버퍼 상황이 어떻게 변화하는지 알아보고, 이 때 CLAH 프레임워크가 적용되었을 때, 두 항목에 어떤 효과가 발생하는지 살펴보고 시스템의 성능을 검증하였다.

### 제 1 절 실험 환경

실험환경은 크게 클라이언트와 콘텐츠 서버 두 부분으로 구성되는데, 먼저 클라이언트 측 요구 환경은 GUI 를 지원하는 리눅스 OS, 구글 크롬 브라우저, C++11 을 사용할 수 있는 GCC 컴파일러, 패킷 필터링을 위한 라이브러리인 libnetfilter\_queue 와 웹소켓 라이브러리인 libwebsockets 를 지원해야 한다.

콘텐츠 서버의 경우는 일반적인 웹 어플리케이션 서버로, 실험에서는 물리적인 위치가 클라이언트와 멀리 떨어진 실제 서비스 상황을 가정하기 위해서 IBM SoftLayer 클라우드 서비스를 이용해 미국 캘리포니아 주 산호세 데이터 센터에 위치한 가상 머신을 임대하여 Apache 웹 서버를 구축하여 서비스하였다. 실험에 사용된 비디오 콘텐츠는 Google MPEG-DASH 오픈 소스 라이브러리[12]에서 제공하는 샘플 미디어 중 무인자동차 광고 콘텐츠인 재생시간 183 초짜리 car 비디오를 사용하였다. 실제 실험에 사용된 상세 조건은 표 1 과 같다.

표 1 실험에 사용된 환경

|       |        |                             |
|-------|--------|-----------------------------|
| 클라이언트 | 디바이스   | Samsung NT900X3G-K58 laptop |
|       | OS     | Lubuntu 15.04               |
|       | 커널 버전  | 4.2.0-18-generic            |
|       | 컴파일러   | G++ 5.2.1-22ubuntu2         |
|       | 브라우저   | Chrome 46.0.2490.86         |
|       | 패킷 필터  | libnetfilter-queue          |
|       | 웹소켓    | libwebsockets-dev           |
| 서버    | 클라우드   | IBM softLayer               |
|       | 머신     | 2.0GHz CPU, 1.0G RAM        |
|       | 데이터 센터 | 미국 캘리포니아 산호세                |
|       | OS     | Ubuntu 14.04.3 LTS          |
|       | 웹서버    | Apache/2.4.7                |

실험을 위해 구성된 전체 구성도는 그림 5 와 같다. 1 대의 콘텐츠 서버와 1 대의 CLASH 시스템이 구현된 1 대의 랩톱 디바이스를 통해 DASH 스트리밍을 수행하여 그 결과를 수집하였다.

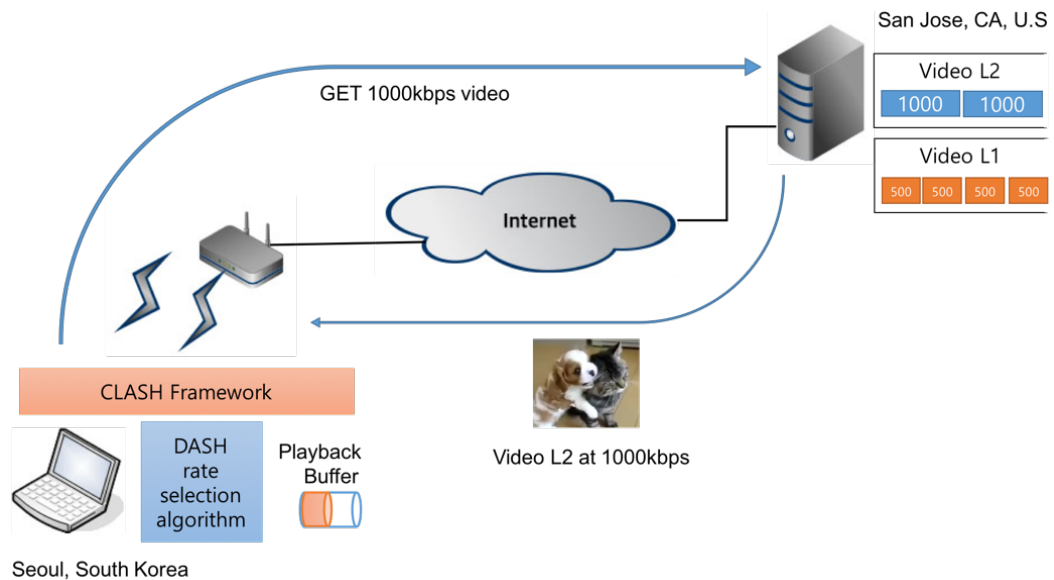


그림 5 실험 환경 구성도

## 제 2 절 성능 평가

CLASH 성능 검증을 위해 다음의 네가지 시나리오에 대해 평균 비트레이트, 선택된 레이트 변화, 버퍼상태 변화를 살펴 보았다.

- 1) 특정 제약 없이 실험환경의 네트워크 상황에서 DASH 스트리밍 서비스를 수행
- 2) 5%확률로 랜덤 순서의 패킷을 인위적으로 중간에서 드롭하여 재전송을 발생
- 3) 2)의 시나리오에서 Out-of-segment 검출 시 무조건 CLASH 의 패킷 로스 하이드 기법을 수행
- 4) 2)의 시나리오에서 Out-of-segment 검출 시에, 버퍼상황이 긴급 상황 (Emergency)에 해당하는 10 초 미만인 경우 로스 하이드 기법을 수행

이때 Out-of-segment 란, 패킷이 수신 되었을 때 sequence 번호 순서에서 몇 개가 빠진 상태로 받아진 경우를 의미한다.

시나리오별 평균 비트레이트는 그림 5 에 나타난 것과 같은데 범례의 순서대로 normal 은 1 번 시나리오, retrans 는 2 번, hideall 은 3 번, hide10s 는 4 번 시나리오를 나타낸다. 이때 각 시나리오에 따른 총 패킷과 드롭된 패킷수 및 패킷 하이딩수, 재전송 수는 표 2 에, 선택된 비디오 비트레이트의 변화는 그림 4 에 나타나있다. 결과는 아무 조건 없이 콘텐츠 서버로부터 비디오 스트리밍을 수행했을때, 평균 비트레이트가 약 635Kbps 으로 재전송은 발생하지 않았고, 이 결과가 실험 환경에서 설정한 네트워크 환경에서 재전송 발생하지 않았을때, FESTIVE 알고리즘을 통해 선택된 비트레이트로 비디오콘텐츠가 다운로드 받아질 때의 결과를 나타낸다.

이를 기준으로 시나리오 2에서는 4.99%의 패킷을 손실시켰으며 손실된 패킷은 모두 재전송되었다. 이때 평균 비트레이트는 약 263Kbps로 보통 상황에 비해 60%이상 성능 저하가 발생함을 확인 할 수 있었다. 그림 4의 비트레이트 선택을 살펴보면 거의 최저 수준의 비트레이트의 비디오를 선택함으로써 최저의 화질을 서비스하게 됨을 알 수 있다. 반면 시나리오 3에서는 4.82%의 드롭된 패킷을 모두 로스하이드를 하게 되면서, 재전송으로 인한 비트레이트 코스트를 전혀 가지고 가지 않으면서 보통상황에 거의 근접한 비트레이트를 유지할 수 있음을 확인할 수 있었다. 시나리오 4에서는 버퍼에서 10 초의 스트레스홀드를 가지고 10 초 미만의 상황에서 패킷로스의 경우만 로스하이드를 하도록 하였는데, 결과 3.7%에 해당하는 250 개의 패킷은 10 초 미만에서 발생하였으며 그 외에 1.3%에 해당하는 88 개의 패킷은 재전송을 받게 되었다. 그 결과 평균 비트레이트는 498Kbps로 약 20%의 품질 저하가 발생함을 확인할 수 있었다.

**표 2 패킷 모니터링 결과**

| 시나리오    | 총패킷수 | 패킷드롭        | 하이드            | 재전송            |
|---------|------|-------------|----------------|----------------|
| Normal  | 7297 | 0 (0.00%)   | 0<br>(0.00%)   | 0<br>(0.00%)   |
| Retrans | 4371 | 218 (4.99%) | 0<br>(0.00%)   | 218<br>(4.99%) |
| Hideall | 6800 | 328 (4.82%) | 328<br>(4.82%) | 0<br>(0.00%)   |
| Hide10s | 6774 | 328 (4.84%) | 250<br>(3.7%)  | 88<br>(1.3%)   |

그림 6는 각 시나리오에서의 버퍼 상태의 변화를 나타내는데, 1 초 단위로 기록되어있다. 2 번 시나리오에서 재전송의 영향으로 두번 정도 비디오 재생이 중단되고 버퍼가 차기를 기다리는 리버퍼 상황이

나타남을 확인할 수 있었다. 시나리오 3, 4 의 경우에는 10 초 미만에서 버퍼가 떨어지는 상황에서 상대적으로 빠른 속도로 10 초 이상으로 회복 되어 리버퍼 상황이 회피되는 것을 관찰 할 수 있었다.

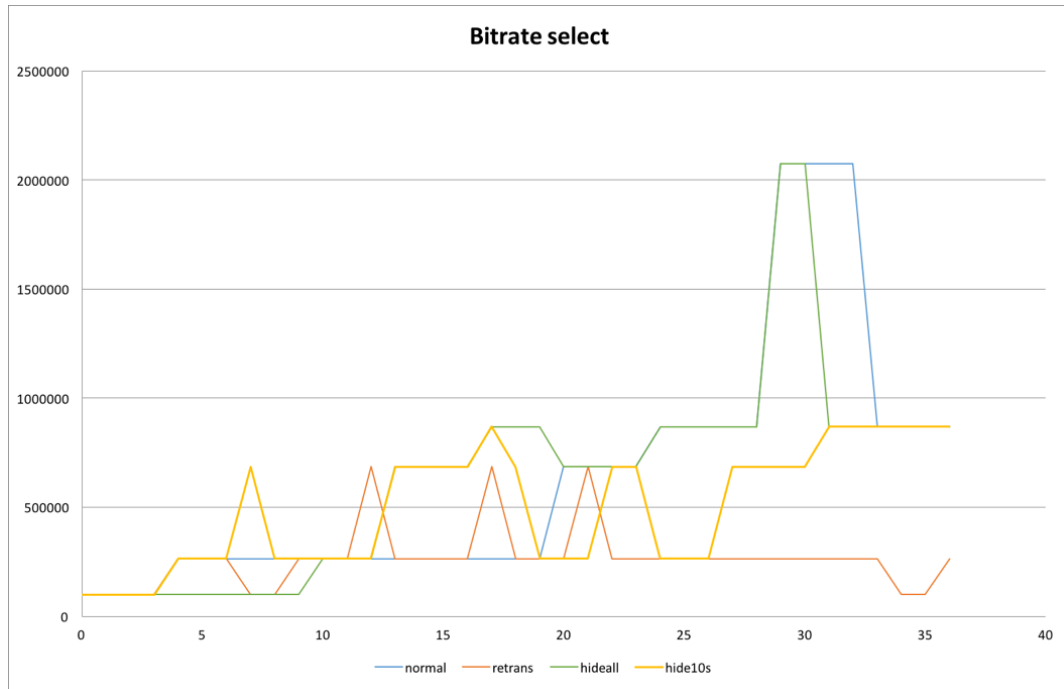


그림 6 선택된 비트레이트의 변화

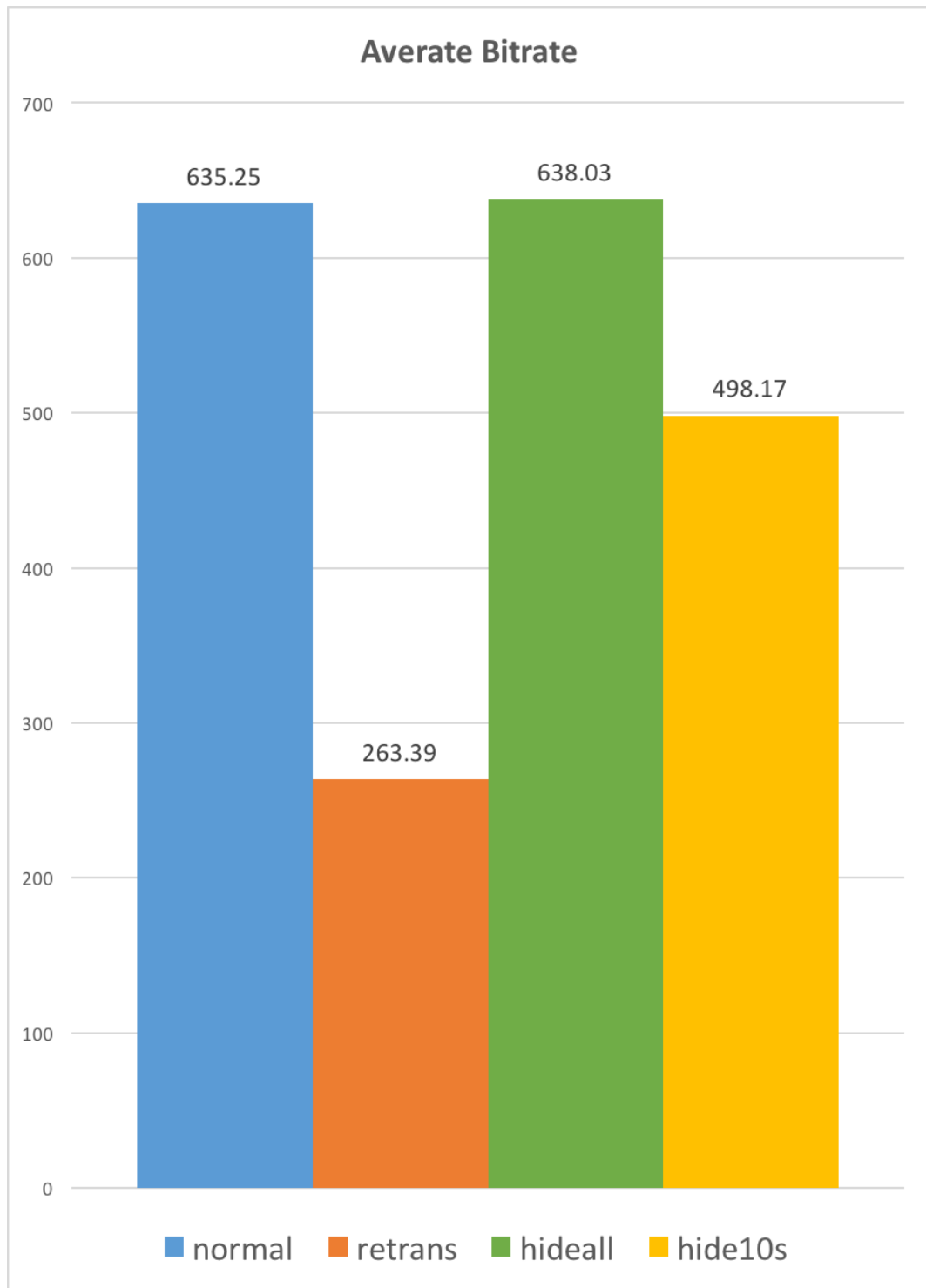


그림 7 시나리오별 평균 비트레이트 비교

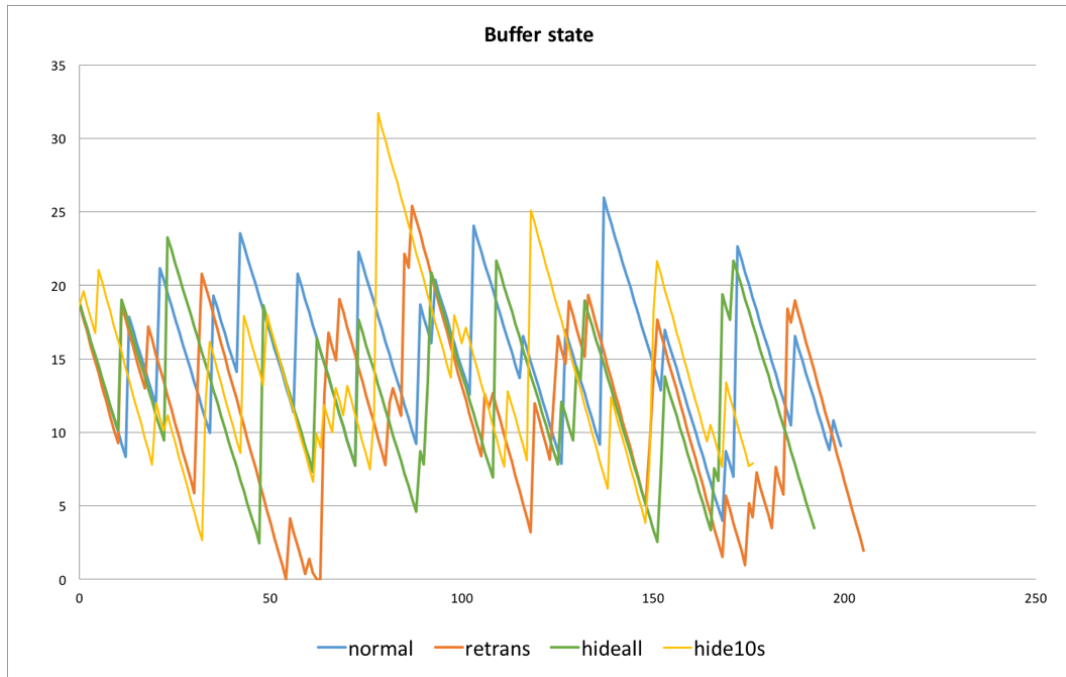


그림 8 버퍼 상태의 변화

### 제 3 절 결과 논의

실험 결과를 살펴보면 불과 5 퍼센트 미만의 재전송에도 TCP 의 영향으로 전체 스루풋에 큰 영향을 줄 수 있다. 따라서 CLASH 에서 사용하는 로스 하이딩 방식이 이 경우에 비디오 서비스에 긍정적인 영향을 줄 수 있음을 알 수 있다. 물론 이것이 무조건적으로 가능한 것은 아니다. 몇가지 제약사항이 있는데, 첫번째로 몇몇 특정 패킷에 대해서는 비디오 플레이를 위해 로스 하이드를 수행할 수 없다. 대표적으로 비디오 헤더를 포함하고 있는 패킷이나 청크의 패킷 정보를 포함하고 있는 요청된 청크의 마지막 패킷, 비디오 슬라이스 헤더 정보를 포함한 패킷등은 만일 손실이 발생한 경우에는 재전송을 반드시 받아야 하는 패킷들이다. 또, 본 연구의 실험에서는 손실이 발생된 패킷의 직전 패킷 정보를 대신 채워 넣는 방식으로 더미 데이터를 전송하도록 하였는데, 디코드에서 이러한 에러를 처리하기 위한

한계점이 있어서 그 이상의 오류가 발생하는 경우 디코드에 실패하는 경우가 발생하게 된다. 따라서 이러한 부분에 대한 고려도 필요하다.

그럼에도 불구하고 DASH 서비스의 QoE(Quality of Experience) 측면에서 가장 중요한 부분을 차지하는 리버퍼를 줄일 수 있다는 점에서도 CLASH 가 충분히 중요한 역할을 할 수 있다는 것을 알 수 있다.

## 제 5 장 결 론

본 논문에서는 기존의 DASH 알고리즘들이 다른 영역인 어플리케이션 레이어에서의 네트워크 밴드위트 예측 알고리즘을 정교화하는 방식에서 벗어나 TCP 에서의 상황과 비디오 플레이어의 상황을 동시에 고려하여 크로스레이어 적응형 스트리밍 프레임워크인 CLASH 를 제안했다. 이를 위해 클라이언트 측에서 패킷의 흐름을 제어하여 패킷 손실에 의해 발생하는 스트리밍의 품질 저하를 최소화 하기 위해 패킷 손실을 모니터링하고 손실이 발생하였을 때 TCP 에서 혼잡 윈도우의 크기를 유지할 수 있도록, raw 소켓을 활용한 로스하이드 기법을 이용하여 패킷 손실을 숨김으로써 적은 패킷 손실 비율에도 재전송에 의한 영향으로 전체 스루풋과 버퍼상황이 급격하게 악화되는 현상을 막도록 하였다. 이때, 발생한 손실에 대해서는 플레이어에서 사용하는 에러 보정 보간 기술을 활용하여 화질 저하를 어느 정도 극복하도록 하였다.

제안한 기법을 활용하여, 불규칙적인 패킷 로스가 발생할 수 있는 네트워크 상황에서도 안정적인 스트리밍을 통해 사용자 QoE 를 저하시키지 않을 수 있는 방안이 될 수 있을 것이다.

본 논문에서는 TCP 에서 패킷 손실을 숨김으로서 발생하는 경쟁 노드간 공정성(Fairness) 이슈에 대해서는 다루지 않았는데, 이 부분에 대해서 향후 이득을 얻은 만큼 버퍼 상황이 좋아 졌을 때, 해당 이득만큼의 손해를 발생시켜 궁극적으로 재생 시간 안에서는 비트레이트 측면에서 공정성을 해치지 않도록하는 연구를 진행할 예정이다. 또한 비디오의 I 프레임이 인코딩 된 위치에 대한 정보를 알

수 있다면 화질 저하라는 측면에서 보다 효과적인 해결책을 제시할 수 있을 것으로 예상된다.

## 참고 문헌

- [1] Cisco Forecast (<http://www.ciscoknowledgenetwork.com>).
- [2] Junchen Jiang, Vyas Sekar, Hui Zhang. Improving Fairness, Efficiency, and Stability in HTTP-based Adaptive Video Streaming with FESTIVE. In CoNEXT, 2012..
- [3] T. Stockhammer. Dynamic adaptive streaming over http--: standards and design principles. In ACM MMSys, 2011..
- [4] Vinton G. Cerf, Robert E. Kahn, "A Protocol for Packet Network Intercommunication". IEEE Transactions on Communications, pp 637–648, 1974
- [5] Jacobson, Van; Karels, Michael "Congestion Avoidance and Control". ACM SIGCOMM Computer Communication Review, pp.157–187, 1988.
- [6] Wang, He, et al. "No need to war-drive: unsupervised indoor localization." Proceedings of the 10th international conference on Mobile systems, applications, and services. ACM, 2012.
- [7] A. Perkis, S. Munkcby, and O. Hillestad. A model for measruring quality of experience. In Proc. NORSIG, 2006.
- [8] N. Cranley, P. Perry, and L. Murphy. User perception of adapting video quality. International Journal of Human-Computer Studies, 2006.
- [9] R. K. P. Mok, E. W. W. Chan, X. Luo, and R. K. C. Chang. Inferring the QoE of HTTP Video Streaming from User-Viewing Activities. In SIGCOMM W-MUST, 2011.
- [10] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. A Control-Theoretic Approach for Dynamic Video Streaming, In Proc. SIGCOMM 2015.

- [11] Te-Yuan Huang., et al. "A buffer-based approach to rate adaptation: evidence from a large video streaming service", in Proc. SIGCOMM 2014.
- [12] Google MPEG-DASH/Media Source demo  
(<http://dash-mse-test.appspot.com>)

# **Abstract**

## **Cross-Layer Adaptive Streaming over HTTP**

Yoon Kwon

School of Computer Science & Engineering

The Graduate School

Seoul National University

Recently, global demands on video streaming has been widespread because of the explosive increasing mobile devices and expanded infrastructure. As a result, video constitute more than 70% of total internet traffic and this tendency seems growing rapidly. In this circumstance, over the past few years, many researches on HTTP based adaptive streaming over TCP has been widely studied and MPEG (Moving Picture Expert Group) standardized DASH (Dynamic Adaptive Streaming over HTTP) for it. Although there have been many efforts for improving client-side algorithm for DASH which makes the network throughput prediction more precisely, the performance of DASH is bind to the characteristics of TCP because fundamentally DASH works over TCP. But the main goals of TCP and DASH are different; the main goal for TCP is reliable end-to-end communication but that of DASH is high QoE(quality of experience) of video streaming.

So in this study, I suggest a framework, named CLASH (Cross Layer Adaptive Streaming over HTTP), which is a client-side system that makes TCP layer and Application layer work for optimized DASH video streaming

For this, I implemented a system for endure the case like packet loss which TCP performance affects critically and I evaluate by real world experiment

**Keywords :** Video Streaming, Adaptive Streaming, DASH, TCP, Cross-Layer Approach, QoE

**Student Number :** 2014-21781