일반논문 (Regular Paper)
방송공학회논문지 제24권 제4호, 2019년 7월 (JBE Vol. 24, No. 4, July 2019)
https://doi.org/10.5909/JBE.2019.24.4.602
ISSN 2287-9137 (Online) ISSN 1226-7953 (Print)

실감미디어 전송을 위한 차세대 HTTP 기반 적응적 스트리밍 전송 프로토콜 연구

송 민 정[a], 유 성 근[a], 박 상 일[a]‡

A Study on Next Generation HTTP-based Adaptive Streaming Transmission Protocol for Realistic Media

Minjeong Song[a], Seong-geun Yoo[a], and Sang-il Park[a]‡

요 약

실감 미디어의 발전에 의해 시청자의 QoE를 보장하기 위해 다양한 스트리밍 기술에 대한 연구가 진행되고 있다. HTTP 적응적 스트리밍은 그 대표적인 예이며 HTTP/1.1과 TCP 기반으로 이루어져 있다. 해당 프로토콜들은 웹 페이지 대기 시간을 증가시키고 영상의 지연을 불어일으키는 원인의 하나로 대두되고 있다. 따라서 본 논문에서는 다양한 전송 프로토콜과 HTTP의 발전과정에 대해 분석한 후 UDP 기반의 전송프로토콜인 QUIC과 HTTP/2를 MPEG-DASH 시스템에 적용한 QUIC-DASH 시스템을 제안한다. 실험을 통해 해당 QUIC-DASH 시스템은 LTE 환경의 전송 속도 측면에서 기존의 시스템보다 최적의 성능 제공 가능성을 확인하였다. 또한, 더 나은 성능을 위해 다양한 차후 연구에 대해 제안한다.

Abstract

Various streaming technologies are being studied to guarantee the QoE of viewers due to the development of realistic media. HTTP adaptive streaming is a typical example, and it is based on HTTP / 1.1 and TCP. These protocols have become one of the causes of delaying the image delay and increasing the waiting time of web pages. Therefore, in this paper, we propose a QUIC-DASH system applying the UDP-based transmission protocols QUIC and HTTP / 2 to the MPEG-DASH system after analyzing various transmission protocols and development process of HTTP. Through experiments, the QUIC-DASH system confirmed the possibility of providing optimal performance in terms of transmission speed of LTE environment than existing system. We also suggest various future studies for better performance.

Keyword : HTTP/2, HTTP/3, HTTP Adaptive Streaming, QUIC, MPEG-DASH

a) 서울과학기술대학교 나노IT디자인융합대학원 정보통신미디어공학전공(Dept. of Information Technology & Media Engineering Graduate School of Nano IT Design Fusion Seoul National University of Science and Technology)
‡ Corresponding Author : 박상일 (Sangil Park)
E-mail: sangilparkmail@gmail.com
Tel: +82-2-970-6765
ORCID: https://orcid.org/0000-0001-8848-4975

※ This work was supported by an Institute for Information and communications Technology Promotion (IITP) grant funded by the Korea Government (MSIT) (No. 2016-0-00144, Moving Free-viewpoint 360VR Immersive Media System Design and Component Technologies)
· Manuscript received January 16, 2019; Revised April 26, 2019; Accepted June 25, 2019.

Ⅰ. 서 론

현재 스마트폰, 태블릿 PC와 같은 스마트 기기의 보급이 대중화되고 그에 맞는 네트워크의 기술이 발전함에 따라 스마트 기기로 멀티미디어 스트리밍 서비스를 즐길 수 있는 넷플릭스(Netflix), 유튜브(Youtube) 같은 OTT (Over the Top) 플랫폼이 등장하였다[15]. 글로벌 네트워크 장비 업체인 시스코(Cisco)는 VNI 보고서에서 2021년, 전 세계 인터넷 트래픽 중 동영상의 트래픽이 80% 이상일 것으로 예상하였다[16]. 이와 같이 동영상 스트리밍의 수요가 증가함에 따라 효율적인 전송을 가능하도록 하는 HTTP (Hyper-Text Transfer Protocol) 기반의 적응적 스트리밍 기술(HAS, HTTP Adaptive Streaming) 주목받고 있다. HAS 기술로는 HLS(HTTP Live Streaming), HSS(HTTP Smooth Streaming), HDS(HTTP Dynamic Streaming)가 있으며 이들의 표준으로 MPEG에서 개발한 MPEG-DASH(Dynamic Adaptive Streaming over HTTP)가 있다.

모바일 기반의 동영상 스트리밍 서비스는 자체의 객관적인 품질(QoS, Quality of Service)뿐만 아니라 사용자 경험 품질(QoE, Quality of Experience)도 중요하게 여겨지고 있다. 이러한 적응적 스트리밍 기반의 QoE 측정 기준에는 동영상의 지속적인 재생 끊김, 비디오 품질의 평균, 품질 변경 시 발생되는 품질 간의 차이 등이 존재하며 이와 같은 이슈를 해결하기 위한 여러 연구가 진행되고 있다[6]. 또한 불안정한 모바일 기반의 네트워크 환경에서는 클라이언트가 품질 변경을 연속적으로 반복하기 때문에 동영상 스트리밍 서비스의 QoE를 보장하지 못하는 문제가 발생한다[8].

특히, VR(Virtual Reality)과 AR(Argumented Reality) 등의 실감 미디어 기술이 발전함에 따라 해당 콘텐츠들을 모바일 상에서 즐길 수 있는 서비스들이 많이 생성되고 있다. 이러한 실감 미디어 콘텐츠들은 특성상 고 해상도 및 대용량의 특성을 가지고 있다. HTTP 적응적 스트리밍은 현재 TCP(Transmission Control Protocol)기반의 전송 시스템을 이루고 있기 때문에 대용량 컨텐츠 제공을 위한 요청-응답의 지속적 반복으로 전송 대역폭의 효율성을 낮춘다는 이슈가 있다.

따라서 본 논문에는 구글에서 개발하고 IETF(Internet Engineering Task Force)에서 표준화를 진행 중인 UDP (User Datagram Protocol) 기반의 프로토콜인 QUIC(Quick UDP Internet Connection)을 이용하여, 기존의 UDP 멀티미디어 전송 시스템에서 개선점으로 판단되는 혼잡 메커니즘의 부재와 보안 기능을 추가한 UDP 전송 시스템인 QUIC 서버를 구축하고 MPEG-DASH 시스템에 접목하여 QUIC- DASH 시스템을 제안 및 개발하였고 기존 TCP 기반의 DASH 시스템과 비교 분석 하였다.

본 논문의 구성은 다음과 같다. 2장에서는 QUIC 프로토콜의 기본적인 개념과 더불어 기존 TCP, UDP로부터의 개선점, 해당 프로토콜이 적용되는 HTTP/3까지에 대해 논한다. 3장에서는 전반적인 QUIC-DASH 시스템에 대해 설명한다. 4장에서는 HTTP/1.1 기반의 TCP 프로토콜을 사용한 기존의 MPEG-DASH 시스템과 HTTP/2 기반 QUIC 프로토콜을 이용한 QUIC-DASH 시스템을 각각 실험하고 결과를 비교 분석한 후 해당 시스템을 응용한 향후 연구 계획에 대해 설명한다. 마지막으로 5장에서는 본 논문의 결론에 대해 논한다.

Ⅱ. QUIC 프로토콜

1. QUIC

QUIC은 구글에서 웹 페이지 대기시간을 감소시키기 위해 개발한 UDP 기반의 전송 프로토콜로, UDP의 신뢰성 문제를 보완한 새로운 인터넷 전송 프로토콜이다[11]. QUIC은 기존의 전송 프로토콜보다 HTTP 트래픽을 가속화하고 HTTP/2와 같은 다중화 및 흐름 제어, TLS(Transport Layer Security)와 동등한 보안, TCP와 동일한 안정성 및 혼잡 제어를 제공한다[1]. QUIC 프로토콜은 2012년 Jim Roskind가 디자인하여 확장된 실험이 2013년에 공개적으로 발표되었다. 이 후 구글은 크롬 브라우저와 구글 검색, 유튜브 등의 서비스 측에 적용하였고 QUIC 프로토콜이 재버퍼링(rebuffer)을 30% 감소시키는 것을 증명하였다[2,12].

QUIC 프로토콜은 기본적으로 TCP+TLS+HTTP/2와 유사하다. 하지만 TCP는 운영 체제 커널 및 Middlebox 펌웨어에 구현되므로 TCP 자체를 크게 변경하는 것은 불가능하다. 하지만 QUIC은 UDP와 함께 사용되기 때문에 TCP

에서 발생하는 제한이 없다[13].

1.1 기존 프로토콜과 QUIC의 비교 분석

TCP는 세 방향 핸드셰이크(3-way Handshake)라는 특징을 지니는데 이러한 핸드셰이크 후에만 데이터를 교환할 수 있다. 핸드셰이크 기반의 전송 방식은 클라이언트에서 서버로 데이터를 주고받는 과정에서 사용자가 즉시 콘텐츠를 원할 때 데이터를 제공할 수 없는 지연을 일으키는 주된 원인 중 하나이다. 또한, TCP 기술의 특징 상, HTTP/2와 같이 여러 스트림이 다중화 되어있는 상황에서 서로 다른 스트림으로부터 트래픽이 송신된다 하더라도 하나의 스트림이 이 전에 전송되었던 스트림으로 이해 발생하는 데이터 혼잡 상황에 놓여 진다면 이 후 스트림의 트래픽 또한 전송되지 않는 HoL(Head of Line) 문제가 발생한다[9].

하지만 QUIC은 코어(Core)에서 0 왕복 시간(RTT, Round Trip Time)을 목표로 개발되었다. 클라이언트에 데이터를 전송하는 동안 커넥션을 개방하고 모든 매개 변수를 동시에 처리할 수 있다. QUIC은 UDP를 전송 프로토콜로 사용

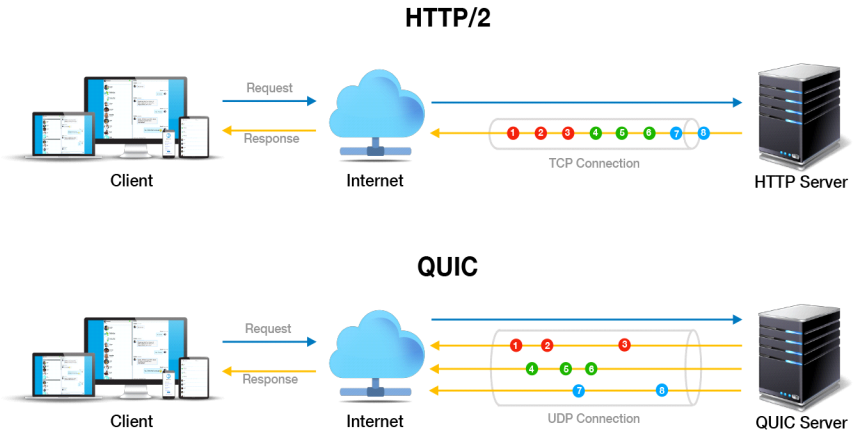

그림 1 . HTTP/2 및 QUIC의 요청-응답 경로
Fig. 1. Request-response path for HTTP / 2 and QUIC

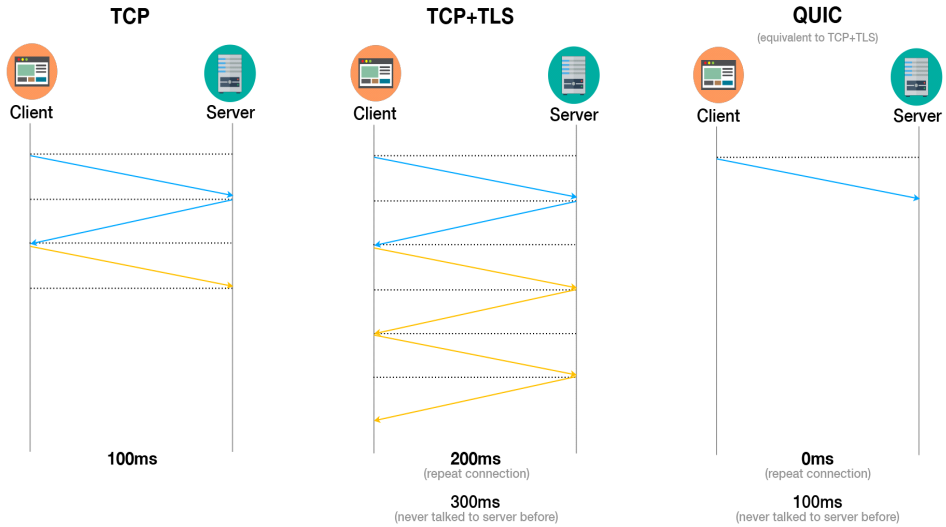

그림 2. TCP, TCP+TLS, QUIC의 연결 설정 과정
Fig. 2. TCP, TCP + TLS, QUIC connection establishment process

하기 때문에 클라이언트 및 서버가 데이터가 목적지에 도착했는지 확인하는 과정이 생략된다.

QUIC은 TCP로 인해 일어나는 세 방향 핸드셰이크 지연을 제거하는 것 외에도 커넥션이 설정되면 클라이언트와 서버에 데이터를 효율적으로 전달할 수 있게 한다. 그림 1에서 볼 수 있듯이, QUIC은 하나의 커넥션 내에서 여러 개의 데이터 스트림을 사용하여 데이터를 서로 연결할 수 있도록 한다. 다시 말해, 한 스트림에서 데이터가 지연되거나 손실 되더라도 다른 스트림에서 데이터가 전송되는 것에 영향을 미치지 않는다.

하지만, UDP는 전송한 메시지가 손실되거나 훼손되어도 송신 측에서는 확인하지 못하기 때문에 신뢰성 및 보안성이 TCP와 비교하여 떨어진다는 단점이 있다[10]. 이러한 점을 해결하기 위해 QUIC은 프로토콜 내에 TLS를 추가하여 인증 및 암호화와 같은 보안 기능과 신뢰성을 제공하고 있다. QUIC은 TLS 1.3의 메커니즘을 응용하여 응용 프로그램의 커넥션이 기밀, 인증 및 무결성 보호되도록 암호화 스키마를 통합한다. HTTPS의 데이터를 보호하는 것과 동일한 기술이지만 QUIC에서는 데이터와 전송 프로토콜 자체를 보호한다[3]. 그림 2는 TCP, TCP+TLS, QUIC의 왕복 단계에 따른 소비 시간을 비교한 그림이다. 현재 사용되는 HTTPS는 TCP+TLS의 구조로 이루어져 있다. 이러한 구조는 하나의 HTTP 요청 시 DNS 조회, TCP 연결을 위한 핸드셰이크, TLS 연결을 위한 핸드셰이크 등의 복잡한 구조로 이루어져 있어 네트워크의 지연을 일으킨다. QUIC은 TCP를 제외하고 TLS 보안 기능의 일부를 사용하기 때문에 TCP 연결 단계 없이 데이터 전송을 더 일찍 시작할 수 있도록 하고 기존 시스템에서 존재했던 재협상(Reneging) 단계를 삭제함으로써 최종적으로 0 RTT를 이루고 있다.

2. HTTP/3

HTTP/2는 다중화 및 서버푸시 등의 다양한 솔루션을 적용하여 기존의 TCP 기반 전송 프로토콜을 수정하지 않고 웹 페이지 대기시간을 향상 시켰다[4]. 하지만 TCP는 요청과 응답이 순차적으로 진행이 되는 연결 지향 프로토콜이기 때문에 개선된 기능의 성능을 최대화 시키지 못하는 상황이다[5]. 따라서 IETF는 인터넷 전송 프로토콜을 TCP에서 QUIC으로 변경하는 새로운 HTTP/3의 초안을 2019년 1월에 발표하였다[7]. UDP 기반의 QUIC을 전송 프로토콜로 대체하는 HTTP/3은 웹 서비스의 사용자에게 혁신적인 속도를 제공하는 것을 목표로 하고 있다[12]. 그림 3에서 볼 수 있듯이, QUIC 프로토콜을 접목한 HTTP/3는 기존 HTTP/2와 유사하게 스트림 다중화 및 스트림 단위 흐름 제어를 통합한다. 스트림 레벨에서 안정성을 제공하고 전체 연결에서 정체 제어를 제공함으로써 TCP 매핑과 비교하여 HTTP의 성능을 향상시킬 수 있다.

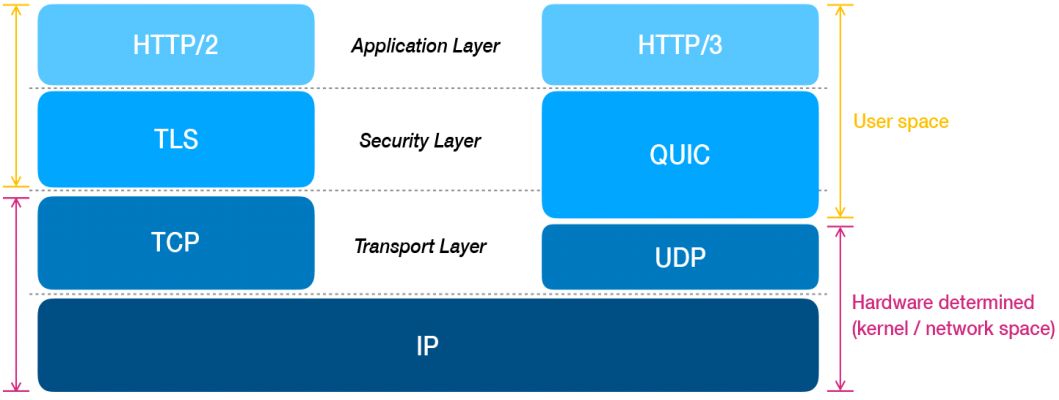

그림 3. HTTP/2 및 HTTP/3의 프로토콜 스택
Fig. 3. HTTP/2 and HTTP/3 protocol stacks

Ⅲ. 제안 시스템

1. QUIC-DASH

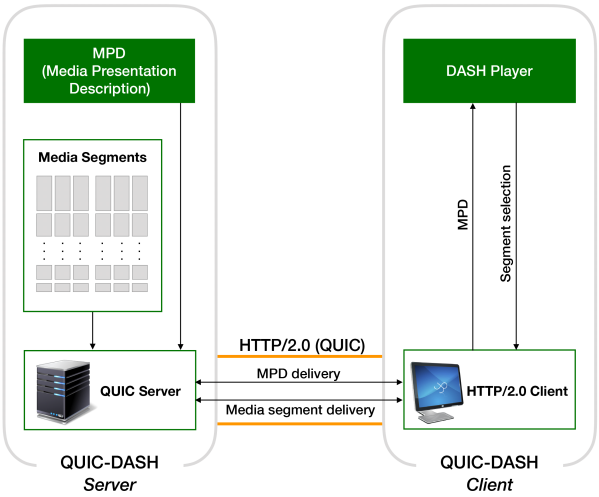

그림 4. QUIC-DASH 시스템 구성도
Fig. 4. QUIC-DASH system configuration diagram

현재 MPEG에서는 기존의 HTTP/1.1 환경에서 일어나는 문제를 해결하기 위해 HTTP/2.0과 웹 소켓(Web Socket)을 적용한 시스템을 개발하고 있다[14]. 본 논문에서는 MPEG의 연구와 동일하게 HTTP/2 기반의 DASH 시스템을 제안하며 TCP 전송 방식 대신 UDP기반의 gQUIC 프로토콜을 이용하여 QUIC-DASH 시스템을 개발하였다. 해당 시스템은 기존의 TCP 전송 방식 및 HTTP/1.1 기반의 웹 MPEG-DASH의 시스템의 문제점을 개선시키기 위해 설계되었다.

QUIC-DASH 시스템은 그림 4와 같다. 기어 360(Gear 360)으로 촬영한 360VR 영상을 각 화질 별로 세그먼트화 하여 gQUIC 서버에 저장한다. 이 QUIC 서버는 HTTP/2와 UDP를 통해 HTTP/2 클라이언트와 통신한다. 이 단계에서 각 영상의 세그먼트들과 MPD 파일이 전송되며 해당 파일을 지닌 클라이언트는 웹 페이지 내의 VR 플레이어에 MPD 및 throughput estimation에 대해 전송한다. 웹 페이지는 또한 시청자의 네트워크 상황을 고려하여 360 영상의 화질을 선택적으로 고려하고 상황에 알맞은 세그먼트들을 선택하고 해당 영상의 타이밍을 클라이언트 측에 정보를 전달한다.

QUIC-DASH 시스템의 목표는 HTTP/2.0의 다중화 기능과 QUIC의 0 RTT 기능을 적용시켜 고 해상도, 대용량의 동영상 전송 시 전송 지연을 방지하고 시청자에게 보다 나은 QoE를 보장하는 것이다.

Ⅳ. 실험 결과

1. 실험 환경

360VR 영상의 HTTP/1.1+TCP와 HTTP/2+QUIC의 전송 속도 및 대역폭 측정 테스트를 진행하기 위해 해당 영상을 HTTP/1.1 서버와 QUIC 서버에 동일하게 업로드하여 총 8개의 MPEG-DASH 세그먼트가 로드될 때까지 문서 오브젝트 모델 콘텐츠 로드(DOM Contents Loaded)에 걸리는 시간과 대역폭을 크롬의 개발자 도구를 통해 측정하였다. 또한 패킷 분석 도구인 Wireshark를 이용하여 서버와 클라이언트를 오고가는 패킷을 캡쳐하여 각 전송방식의 특성을 분석하였다.

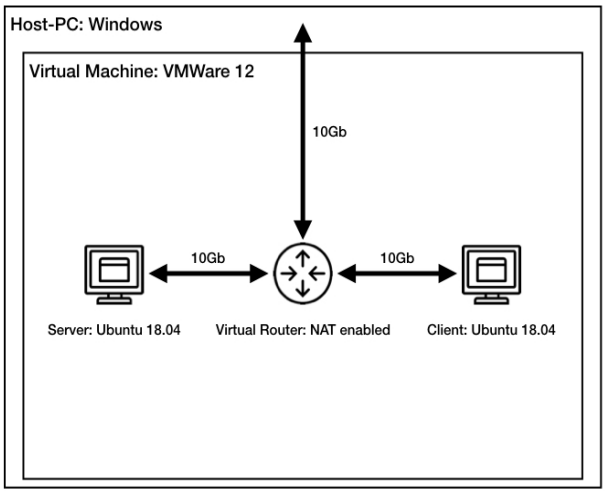

그림 5. 실험을 위한 전체 구성도
Fig. 5. Configuration diagram of experiments

기존의 HTTP/1.1을 사용하는 웹 서버와 QUIC을 사용하는 웹 서버의 테스트를 위하여 VMWare 12를 사용하여 Windows OS가 설치된 호스트 컴퓨터에 두 개의 가상머신을 만들고 Ubuntu 18.04를 설치하였다. 호스트 컴퓨터는

Intel I5 4460 CPU와 16GB 메모리가 장착되어 있다. 두 가상머신은 호스트 컴퓨터에 가상 네트워크 인터페이스로 연결되어 있으며 NAT(Network Address Translation)을 사용하여 각각 사설 IP주소를 할당하였다. 두 가상머신 중 하나는 서버로 사용할 것이고 다른 하나는 클라이언트로 사용할 것이다. 실험을 위한 전체 구성은 그림 5와 같다.

1.1 서버

서버로 사용할 가상머신에 gQUIC을 지원하는 웹 서버를 구성하기 위해 구글에서 제공하는 오픈소스 서버 소프트웨어인 proto-quic을 빌드하였다. 이에 사용된 QUIC 프로토콜은 QUIC-Version-39이다. 또한 HTTP/1.1과의 비교 실험을 위해 Node.js v.4.8.6을 이용하여 해당 웹 서버를 구현하였다. 실험에 사용된 4K 360VR 비디오를 x264를 이용해 인코딩 하였다. 생성된 인코딩 파일을 MP4Box를 이용해 4000ms 청크(Chunk)로 분리하여 각 1080p, 720p, 480p 화질마다의 비디오 세그먼트를 생성한 뒤 MPD 파일을 작성하였다. 현재 많은 사용자들이 비디오를 시청하는 환경인 모바일에서 콘텐츠 로드를 시뮬레이션하기 위하여 LTE 네트워크 평균속도에 가까운 3,200Kbps로 대역폭을 제한하여 실험하였다. 각 웹서버에는 DASH영상을 로드할 수 있도록 Javascript로 구현된 레퍼런스 소프트웨어인 Dash.js와 프론트엔드 HTML파일, MPD와 영상세그먼트 등을 웹서버의 루트 디렉토리에 각각 저장하였다.

1.2 클라이언트

클라이언트로 사용할 가상머신에는 문서 오브젝트 모델 콘텐츠 로드속도와 대역폭을 측정 할 수 있는 도구가 포함되어 있는 구글의 Chromium 웹 브라우저를 설치하였다. QUIC-Version-39로 실험을 진행하기 위하여 브라우저를 실행할 때 명령 줄 변수에 QUIC버전을 명시하여 전달하였다. gQUIC서버에 6121번 포트를 통하여 접근하였고, HTTP/1.1서버에 8081번 포트를 통하여 접근하였다.

2. 실험 결과

2.1 페이지 로드 속도

그림 6(좌)에서 볼 수 있듯이, HTTP/1.1로 구현된 서버는 DOM Contents Loaded에 총 1.86초가 소요되었다. 문서에 추가적인 엘레먼트(Element)가 없기 때문에 문서 로드에도 같은 시간이 걸렸다. HTTP/1.1 클라이언트는 8개의 영상 세그먼트를 로드하는 것에 평균적으로 약 39초가 소요되었고 총 23개의 리퀘스트를 통하여 14.1MB의 데이터가 전송

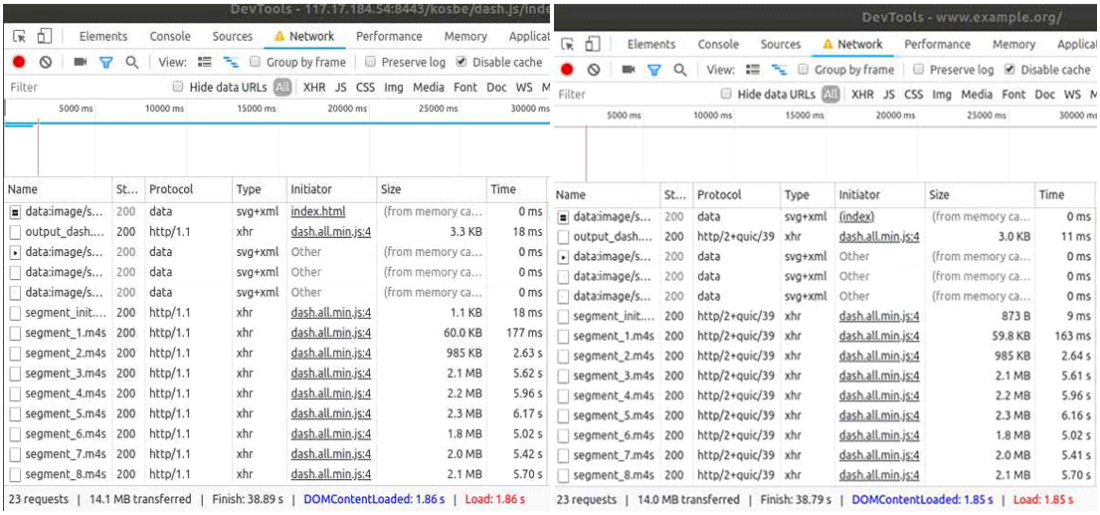

그림 6. HTTP/1.1+TCP 실험 결과 (좌), HTTP/2+QUIC 실험 결과(우)
Fig. 6. HTTP / 1.1 + TCP experiment result (left), HTTP / 2 + QUIC experiment result (right)

되었다.

그림 6(우) 같이 gQUIC 서버는 DOM Contents Loaded에 총 1.85초가 소요되었다. 문서 로드에도 같은 시간이 걸렸다. HTTP/2 클라이언트는 8개의 영상 세그먼트를 로드하는 것에 평균적으로 약 38초가 소요되었고 총 23개의 리퀘스트를 통하여 14MB의 데이터가 전송되었다.

2.2 Wireshark를 이용한 패킷 분석

HTTP/1.1 서버와 gQUIC서버의 패킷을 캡쳐하기 위해서 호스트 컴퓨터에 Wireshark를 설치하고 서버로 사용하는 가상머신의 가상 네트워크 인터페이스의 패킷을 획득하였다. 먼저 클라이언트에서 HTTP/1.1서버로 8081번 포트를 통해 DASH 웹페이지를 요청하면서 Wireshark의 Statistics 메뉴의 I/O Graph를 통해 100ms단위로 전달되는 패킷의 바이트 수 통계를 분석하였다.

HTTP/1.1로 전달되는 패킷만 필터링 하기 위해 TCP 8081번 포트만 그래프에 나타내도록 필터를 설정하였다. 그림 7은 대역폭 제한을 걸지 않은 I/O 그래프이다. 일반적인 MPEG-DASH의 네트워크 I/O 그래프와 같이 각 세그먼트를 요청할 때 대역폭이 급격히 상승하였다가 바로 0에 가까워지는 것을 볼 수 있다. 다음은 일반적인 LTE 모바일 네트워크의 평균속도인 3,200Kbps로 대역폭 제한을 걸어놓은 그래프이다. 대역폭 제한은 클라이언트의 Chromium 브라우저의 대역폭 제한 기능을 사용하여 구현하였다.

그림 8과 같이 세그먼트를 요청할 때마다 대역폭 제한을 설정한 속도까지 대역폭이 상승했다가 곧바로 속도가 떨어지면서 제한속도보다 훨씬 느린 속도로 세그먼트가 전송되는 것을 볼 수 있다.

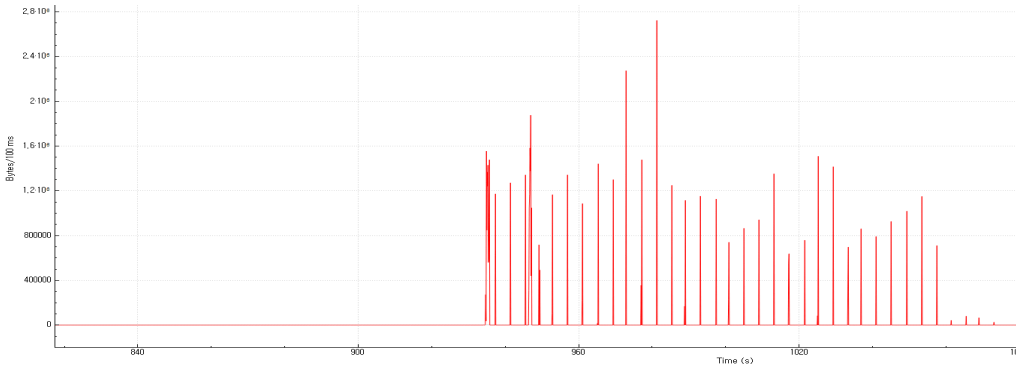

그림 7. 대역폭 제한을 걸지 않은 HTTP/1.1 서버의 DASH 세그먼트 I/O 그래프
Fig 7. DASH segment I/O graph of HTTP / 1.1 server without bandwidth limit

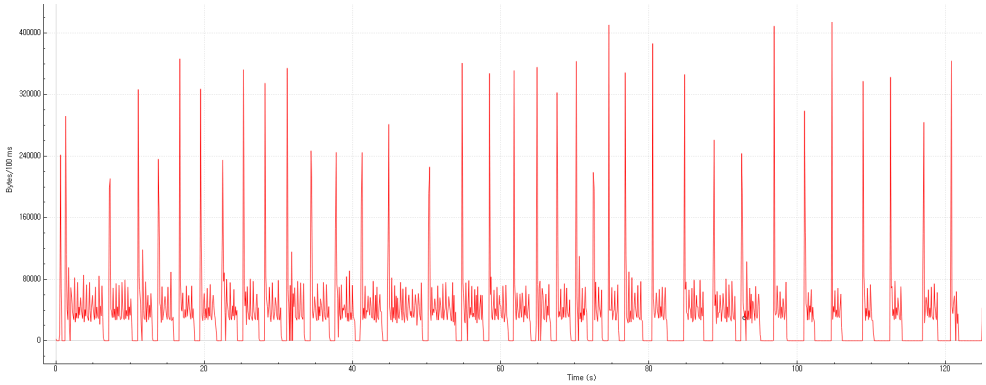

그림 8. 대역폭 제한을 건(3,200Kbps) HTTP/1.1 서버의 DASH 세그먼트 I/O 그래프
Fig. 8. A DASH segment I/O graph of HTTP/1.1 server with bandwidth limit(3,200Kbps)

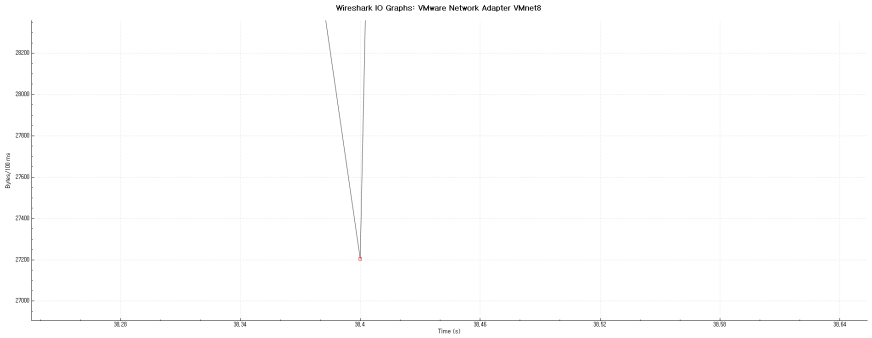

그림 9. 대역폭이 급격히 하락한 지점의 I/O그래프
Fig. 9. I/O graph at the point where the bandwidth has fallen sharply

No.	Time	Source	Protocol	Destination	Length	Info
12493	38.345266	192.168.62.132	TCP	192.168.62.135	1514	8081 → 53858 [ACK] Seq=15586224 Ack=6734 Win=47232 Len=14
12494	38.345312	192.168.62.132	TCP	192.168.62.135	1514	8081 → 53858 [ACK] Seq=15587672 Ack=6734 Win=47232 Len=14
12495	38.345392	192.168.62.132	TCP	192.168.62.135	1514	8081 → 53858 [ACK] Seq=15589120 Ack=6734 Win=47232 Len=14
12496	38.345421	192.168.62.132	TCP	192.168.62.135	1514	8081 → 53858 [ACK] Seq=15590568 Ack=6734 Win=47232 Len=14
12497	38.345573	192.168.62.132	TCP	192.168.62.135	1514	8081 → 53858 [ACK] Seq=15592016 Ack=6734 Win=47232 Len=14
12498	38.345581	192.168.62.132	TCP	192.168.62.135	1514	8081 → 53858 [ACK] Seq=15593464 Ack=6734 Win=47232 Len=14
12499	38.345645	192.168.62.132	TCP	192.168.62.135	1514	8081 → 53858 [ACK] Seq=15594912 Ack=6734 Win=47232 Len=14
12500	38.345653	192.168.62.132	TCP	192.168.62.135	378	[TCP Window Full] 8081 → 53858 [ACK] Seq=15596360 Ack=673
12501	38.346072	192.168.62.135	TCP	192.168.62.132	78	[TCP Dup ACK 12476#1] 53858 → 8081 [ACK] Seq=6734 Ack=155
12502	38.400857	192.168.62.135	TCP	192.168.62.132	66	[TCP ZeroWindow] 53858 → 8081 [ACK] Seq=6734 Ack=1559667
12503	38.413387	192.168.62.135	TCP	192.168.62.132	66	[TCP Window Update] 53858 → 8081 [ACK] Seq=6734 Ack=1559
12504	38.427273	192.168.62.132	TCP	192.168.62.135	1514	8081 → 53858 [ACK] Seq=15596672 Ack=6734 Win=47232 Len=14
12505	38.428908	192.168.62.132	TCP	192.168.62.135	1202	8081 → 53858 [PSH, ACK] Seq=15598120 Ack=6734 Win=47232 L
12506	38.429591	192.168.62.132	TCP	192.168.62.135	1514	8081 → 53858 [ACK] Seq=15599256 Ack=6734 Win=47232 Len=14

그림 10. 대역폭이 급격히 하락한 지점의 패킷 메시지
Fig. 10. Packet messages at the point where the bandwidth has fallen sharply

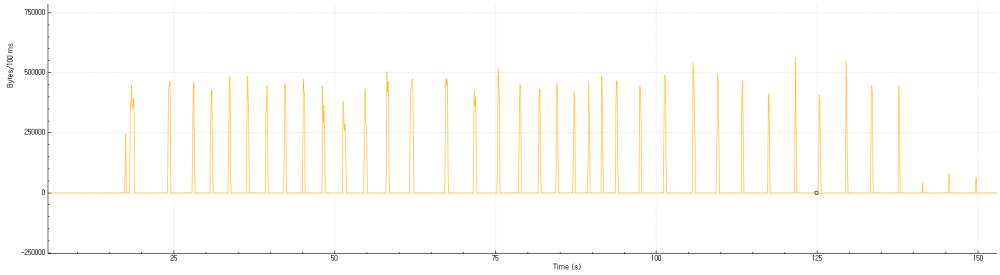

그림 11. 대역폭 제한을 건(3,200Kbps) gQUIC 서버의 DASH 세그먼트 I/O 그래프
Fig. 11. A DASH segment I/O graph of gQUIC server without bandwidth limit

그림 9는 대역폭이 급격히 하락하는 지점의 I/O그래프를 확대한 것이다. 패킷을 캡쳐한지 38.4초가 지났을 때 대역폭이 급격히 감소하였으며, 그 구간의 패킷을 검사해 보면 그림 10와 같이 TCP Window Full이 발생하는 것을 볼 수 있다. 따라서 대역폭이 급격히 느려지는 원인은 세그먼트 요청 이후 TCP Window Full이 지속적으로 발생하여 TCP 윈도우를 비우게 되기 때문이다. 또한 바로 대역폭이 회복되지 않는 이유는 TCP는 기본적으로 혼잡제어를 위하여 현재 연결에 알맞은 혼잡 윈도우의 크기를 결정하기 위해 느린 시작(Slow start)이라는 방법을 사용하기 때문이다[5]. 이와 같이 모바일 네트워크와 같이 대역폭이 제한되는 경우나 변동되는 경우에 TCP를 사용하는 HTTP/1.1은 대역폭 할당이 제대로 이루어지지 않으며 속도 회복이 어려운 Stall구간에 빠지게 되어 대역폭 최적화가 어려운 근본적인 문제점을 가지게 된다.

gQUIC서버의 I/O그래프를 분석하기 위해 클라이언트에

서 gQUIC서버로 6121번 포트를 통해 DASH 웹페이지를 요청하면서 Wireshark의 Statistics 메뉴의 I/O Graph를 통해 100ms단위로 전달되는 패킷의 바이트 수 통계를 분석하였다. 그림 11.은 3200Kbps로 제한을 건 I/O 그래프이다. HTTP/1.1과 달리 Stall 구간이 없이 제한이 설정된 최대 대역폭(3,200Kbps)으로 세그먼트를 수신하는 것을 볼 수 있다.

3. 결과 분석

실험을 통해 QUIC 프로토콜은 기존의 HTTP/1.1+TCP에 비해 전송 속도가 약 1초 정도 빠르고 대역폭의 감소 또한 약 800Kb 이루어졌다. 기존의 시스템에 비해 크게 향상하지는 않았지만 총 리퀘스트의 수가 23개로 많지 않은 상황에서 전송 속도와 대역폭 모두 소폭 감소되었다는 결과를 얻었다. 이와 같은 결과가 도출된 원인은 QUIC 프로토콜이 HPACK을 이용한 자체 헤더 압축과 UDP 기반의 전송을 사용하기 때문으로 추정된다. 또한 실험에 사용된 QUIC-Version-39는 과거 gQUIC의 한 종류로써 현재 다양한 성능 테스트에 이용되는 표준화 QUIC에 비해 많은 기능을 최적화 시키지 못한 프로토콜이기 때문에 성능에 대해 많은 개선을 보이지 못했다고 판단한다. 다음으로 패킷을 분석하기 위하여 Wireshark를 통해 HTTP/1.1+TCP와 gQUIC의 I/O그래프를 확인한 실험에서는 대역폭 제한을 걸지 않은 경우에서는 두 프로토콜 모두 비슷한 결과를 확인할 수 있었다. 그러나 LTE네트워크를 시뮬레이션 하기 위해 3,200Kbps로 대역폭 제한을 걸고 실험하였을 때에는 gQUIC서버는 제한된 대역폭 내에서 최대한의 성능을 나타내었다. 하지만 HTTP/1.1+TCP는 TCP의 느린 시작 알고리즘이 가지는 근본적인 문제점 때문에 Stall구간이 발생하여 제한된 대역폭 내에서 최적의 성능을 보이지 않았다. Stall구간이 발생하게 되면 LTE와 같은 모바일 네트워크에서 하나의 기지국에 여러 대의 클라이언트가 동시에 MPEG- DASH로 영상을 수신할 때 전체 네트워크의 시간당 전송율을 떨어뜨리게 되므로 전송 효율을 낮추게 된다. 그러나 gQUIC을 사용하게 되면 UDP를 사용하고 자체적인 혼잡 제어 알고리즘을 사용하기 때문에 모바일 네트워크에서도 성능 저하를 일으키지 않는 것을 알 수 있다. 기존의 시스템과 이와 같은 결과는 영상의 길이 또는 DASH 세그먼트의 길이가 긴 영상에서 더 좋은 결과를 보여줄 것으로 예상된다.

4. 향후 연구 제안

4.1 HTTP/3 기반 미디어 전송 테스트벤치 구축

현재 IETF에 의해 표준화가 진행중인 HTTP/3는 현재 전송계층 프로토콜인 IETF-QUIC을 지원하는 ngtcp2와 같은 여러 클라이언트-서버 구현체들이 존재한다. 그러나 아직 HTTP/3를 지원하는 웹서버가 존재하지 않고 이를 지원하는 구글 크롬과 같은 웹 브라우저에서 이를 지원하지 않으므로 향후 기존의 TCP기반의 HTTP와 HTTP/3의 성능을 비교할 수 있는 테스트 툴이 요구된다. 따라서 IETF-QUIC의 구현체들을 이용하여 성능을 비교하고 분석할 수 있는 HTTP/3기반 미디어 전송 테스트 벤치를 구축하는 것이 필요하다.

4.2 HTTP/3 기반 적응적 스트리밍 프리패치(Prefetch) 연구

기존의 TCP기반 HTTP프로토콜에서 적응적 스트리밍을 가능하게 하기 위하여 고안된 MPEG-DASH는 HTTP/1이 가지고 있는 HOL 차단 등의 문제를 피해가기 위하여 다중 세그먼트 전송이나 프리패치와 같은 기법이 적용되지 않았다. 따라서 기존의 HTTP/1에서 영상이 끊기거나 화질이 저하되는 문제가 존재하였다. 그러나 UDP를 사용하며 다중화 전송을 지원하는 HTTP/3에서는 기본적으로 미디어 스트리밍의 성능이 HTTP/1보다 우수할 것으로 생각된다. 이를 최적화하기 위하여 기존의 MPEG-DASH와 같은 적응적 스트리밍 기술을 HTTP/3에 맞게 수정하는 작업이 필요하며, 특히 다상과 같은 실감 미디어 전송 최적화를 위해서 사용자가 보고자 하는 세그먼트를 시공간적으로 미리 받아 끊김이나 화질 저하 등을 줄이는 것이 필요하다.

V. 결 론

실감미디어의 발전에 따라 시청자의 QoE를 보장하기 위한 다양한 스트리밍 기술이 개발되고 있다. 이 기술의 중심을 이루는 HTTP 적응형 스트리밍은 넷플릭스, 유튜브 등 다양한 서비스에 응용이 되고 있으며 표준 기술인 MPEG-

DASH는 HTTP/2와의 접목을 꾀하고 있다. 하지만 이러한 HTTP/2는 TCP 기반의 전송 방식을 지니고 있기 때문에 연결 지향 및 순차적 프로토콜인 TCP 자체에서 발생하는 HOL 차단 문제 등을 해결하지 못하였다. 이러한 점에서 IETF는 HTTP/2와 UDP위에서 구현하는 QUIC 프로토콜을 접목한 시스템을 HTTP/3라 칭하고 개발 중에 있다.

본 논문에서는 QUIC 프로토콜을 MPEG-DASH에 적용시킨 새로운 HTTP/2 기반의 QUIC-DASH 시스템을 제안하고 개발하였다. 기존의 MPEG-DASH와 같이 HTTP/1.1+TCP의 구조가 아닌 HTTP/2+gQUIC의 구조의 서버-클라이언트를 구현하였고 이를 통해 360VR 영상을 표출하였다.

기존 HTTP/1.1 기반의 MPEG-DASH와 본 논문에서 제안한 HTTP/2.0 기반의 QUIC-DASH를 비교 분석한 결과, 23 request를 수신 받았을 때, 약 1초의 전송 속도의 개선과 800Kb의 대역폭 데이터가 절약되는 것을 확인 하였다. 이러한 실험 결과를 통해 같은 데이터를 전송하더라도 대역폭 및 전송 속도를 감소시킬 수 있다는 것을 확인하였고, 용량이 크거나 세그먼트의 길이가 긴 360VR을 전송하는 경우 더 좋은 결과를 보여줄 것으로 판단하였다.

또한 3,200Kbps로 대역폭 제한을 걸고 LTE네트워크의 평균 대역폭으로 실험 하였을 때에는 HTTP/1.1+TCP는 최적의 성능을 보이지 않았다. 이는 TCP의 근본적인 한계 때문에 Stall구간이 발생하기 때문이다. 그러나 gQUIC프로토콜의 경우는 제한된 대역폭 내에서도 최대한의 성능을 나타내었다. TCP의 느린시작 알고리즘의 영향을 받지 않기 때문에 영상의 길이가 길거나 DASH세그먼트의 길이가 긴 경우에 gQUIC의 경우에 더 우수한 결과를 보여줄 것으로 예상된다.

현재의 MPEG-DASH 시스템은 하나의 세그먼트의 전송이 완료된 후 다음 세그먼트를 전송하는 특성을 지니고 있다. 따라서 동시에 여러 스트림을 보낼 수 있는 QUIC의 효율이 두드러지지 않았지만 차후 DASH 세그먼트간의 프리패치 간격 최적화와 더불어 HTTP/3 기반의 테스트벤치 구축을 통해 보다 나은 성능의 시스템으로 발전시킬 수 있을 것이다.

참 고 문 헌 (References)

[1] Biswal, Prasenjeet, and Omprakash Gnawali. "Does quic make the web faster?." Proceeding of 2016 IEEE Global Communications Conference (GLOBECOM). Washington, DC USA, pp. 1-6, 2016, https://doi.org/10.1109/GLOCOM.2016.7841749

[2] Google Wants To Speed Up The Web With Its QUIC Protocol, https://techcrunch.com/2015/04/18/google-wants-to-speed-up-the-web-with-its-quic-protocol/ (accessed Nov 22, 2015)

[3] Why Fastly loves QUIC and HTTP/3, https://www.fastly.com/blog/why-fastly-loves-quic-http3 (accessed Mar 21, 2019)

[4] HTTP/2, https://hpbn.co/http2/ (accessed Jul 10, 2018)

[5] Stephen Ludin, Javier Garza, Learning HTTP/2, (Translated by Jaejoon Gang), O'Reilly Media, USA, pp. 36-131, 2018

[6] Yunho Kim, Heekwang Kim and Kwangsue Chung, 2018, "Video Quality Maintenance Scheme for Improve QoE of HTTP Adaptive Streaming Service," Journal of KIISE, Vol. 45, No. 2, pp. 187-194, February 2018, https://doi.org/10.5626/JOK.2018.45.2.187

[7] M. Bishop, Hypertext Transfer Protocol Version 3 (HTTP/3), https://quicwg.org/base-drafts/draft-ietf-quic-http.html (accessed Jan, 2019)

[8] Dooyeol Yun and Kwangsue Chung, 2016, "Segment Scheduling Scheme to Support Seamless DASH-based Live Streaming Service," KIISE Transactions on Computing Practices, Vol. 22, No. 7, pp. 310-314, July 2016, https://doi.org/10.5626/KTCP.2016.22.7.310

[9] Behrouz A. Forouzan, TCP/IP PROTOCOL Suite, 4TH EDITION, (Translated by ByungChul Kim and five others), USA, pp.435-521, 2009

[10] User datagram protocol, https://tools.ietf.org/html/rfc768, (accessed Aug.28, 1980)

[11] Hypertext Transfer Protocol (HTTP) over QUIC draft-ietf-quic-http-04, https://tools.ietf.org/html/draft-ietf-quic-http-04, (accessed Jun. 13, 2017)

[12] HTTP/3 explained, https://legacy.gitbook.com/book/ bagder/http3-explained/details, (accessed Jan, 2019)

[13] Minjeong Song, Sunggeun Yoo and Sangil Park, "A Study on Transmission Technology Trend of Web Based Realistic Media (VR / AR) Platform", The Journal of The Korean Institute of Communication Sciences, Vol. 35, No. 9, pp. 38-45, August 2018, http://www.dbpia.co.kr/journal/articleDetail?nodeId=NODE07544687

[14] MPEG-DASH with Server Push and WebSockets, https://mpeg.chiariglione.org/standards/mpeg-dash/dash-server-push-and-websockets, (accessed Oct, 2017)

[15] Yungyoun Kim and Kwangsue Chung, "A Video Quality Control Scheme Based on Content Characteristics for Improving QoE in DASH Environments," Journal of KIISE, Vol. 42, No. 8, pp. 1039-1048, Aug 2015, https://doi.org/10.5626/JOK.2015.42.8.1039

[16] Cisco Predicts More IP Traffic in the Next Five Years Than in the History of the Internet, https://newsroom.cisco.com/press-release-content?type=webcontent&articleId=1955935, (accessed Nov 27, 2018)

저 자 소 개

송 민 정

- 2017년 : 서울과학기술대학교 전자IT미디어공학과 공학사
- 2017년 ~ 현재 : 서울과학기술대학교 나노IT디자인융합대학원 정보통신미디어공학전공 석사과정
- ORCID : https://orcid.org/0000-0002-3532-131X
- 주관심분야 : 실감미디어, HTTP Adaptive Streaming, 웹 플랫폼, 차세대 방송

유 성 근

- 2013년 : 서울과학기술대학교 전자IT미디어공학과 공학사
- 2015년 : 서울과학기술대학교 미디어IT공학과 공학석사
- 2015년 ~ 현재 : 나노IT디자인융합대학원 정보통신미디어공학전공 박사수료
- ORCID : https://orcid.org/0000-0003-2817-454X
- 주관심분야 : 실감미디어, HTTP Adaptive Streaming, 웹 플랫폼, 차세대 방송

박 상 일

- 1977년 : 연세대학교 전자공학과 공학사
- 1983년 : Kansas State University 전기전자공학과 공학석사
- 1987년 : University of New Mexico 전기전자공학과 공학박사
- 1987년 ~ 1988년 : University of Pittsburgh 전자공학과 조교수
- 1988년 ~ 1995년 : Motorola DSP Semiconductor Design Manager
- 1995년 ~ 2006년 : 삼성전자 임원(반도체, 비서실, 본사기획실 등)
- 2006년 ~ 2012년 : 스카이레이크 인큐베스트 투자팀 부사장
- 2009년 ~ 2012년 : 방송통신위원회 차세대 방송 PM
- 2012년 ~ 현재 : 서울과학기술대학교 나노IT디자인융합대학원 정보통신미디어공학전공 교수
- ORCID : https://orcid.org/0000-0001-8848-4975
- 주관심분야 : 실감미디어, HTTP Adaptive Streaming, 웹 플랫폼, 차세대 방송