WebRTC 기반 P2P 통신 병용 DASH 시스템을 위한 이력 기반 피어 선택 알고리듬

*최성현, *서주호, **김상진, **전재영, *김용한†
*서울시립대학교 전자전기컴퓨터공학부, **㈜SBS
†yhkim@uos.ac.kr

A history-based peer selection algorithm for WebRTC-based P2P-assisted DASH systems

*Seong Hyun Choi, *Ju Ho Seo, **Sang Jin Kim, **Jae Young Jeon, *Yong Han Kim

*School of Electrical and Computer Engineering, University of Seoul, **SBS Co., Ltd.

요 약

최근 인터넷 미디어 스트리밍 수요의 증가로 인해 CDN(Content Delivery Network) 서버 비용이 크게 증가하였으며 이를 절감하기 위한 방안의 필요성이 날로 증가하고 있다. 이러한 상황에 맞춰 최근 CDN 서버 비용을 절감할 수 있는 WebRTC(Web Real-Time Communication) 표준 기반의 P2P(Peer-to-Peer) 통신을 병용하는 DASH(Dynamic Adaptive Streaming over HTTP) 기술이 등장하였다. 본 논문에서는 이 기술의 CDN 서버 부하 절감 효과를 크게 개선할 수 있는 알고리듬을 제안한다. 또한 실제 모바일 네트워크 환경과 유사하게 실험 조건을 설정한 후, 이 알고리듬을 구현하여 그 성능을 측정한 결과, 기존과 비교하여 더 높은 절감 효과를 달성할 수 있음을 실험실 내 실험을 통해 보인다.

I. 서론

최근 몇 년 동안 인터넷 미디어 스트리밍은 급격히 증가하였으며 2012년에 시행된 연구에 따르면 인터넷 대역폭의 90%가 미디어일 것으로 예측되고 있다[1]. 이에 따라 인터넷 미디어 스트리밍 서비스를 제공하는 CDN 사업자의 입장에서는 CDN 서버를 증설하고 네트워크 사용료를 증액하여 지불해야 하는 등 미디어 배포 비용이 크게 증가하게 되어 비용 절감에 대한 필요성이 점차 증가하고 있다.

WebRTC는 웹 브라우저 간 실시간 통신을 가능케 하는 API(Application Programming Interface) 표준[2]으로서 이를 지원하는 웹 브라우저 간 실시간 영상 통신, 음성 통신, 채팅 등을 쉽게 구현할 수 있게 한다. 최근 WebRTC를 기반으로 P2P 통신을 병용함으로써 DASH의 효율을 높일 수 있는 기술이 연구되었다[3]. 해당 연구는 미디어를 스트리밍할 때 QoS(Quality of Service)를 낮추지 않으면서 CDN 서버의 부하를 현저하게 줄일 수 있음을 보였다. 또한, 기존 P2P 방식에 비해, WebRTC 표준을 지원하는 브라우저의 경우, 표준 API를 사용할 수 있으므로, 자바스크립트를 사용한 웹 프로그램 개발 비용이 절감되고, 특별한 플러그인(plug-in)이나 추가 애플리케이션을 설치할 필요가 없으므로 사용자의 접근성 및 편의성이 증대된다.

현재 이 방식[3]에 의하면 P2P 연결이 성공적으로 완료된 후 연결된 피어(peer)들은 자신이 가지고 있는 미디어 데이터 정보를 서로에게 알려준다. 피어들은 이 정보를 이용하여 특정 미디어 데이터가 필요할 때 해당 미디어 데이터를 가진 피어들 중 임의의 피어에게 미디어 데이터를 요청하게 된다. 만약 데이터 요청이 거절되거나 일정 시간을 초과하는 경우 CDN 미디어 서버로 요청을 보내 미디어 데이터를 받아오게 된다. 이 과정에서 저속 네트워크 환경에 있는 피어들에게 미디어 데이터를 요청할 경우 원활하지 못한 P2P 연결로 인해 CDN 미디어 서버에로의 요청이 증가하게 되어 "CDN 서버 부하 절감비"(P2P로 받은 데이터 양/총 받은 데이터 양)가 저하된다.

본 논문에서는, 클라이언트 피어가 피어 목록에 있는 피어들의 통신 이력을 축적하여 통계 값으로 가지고 있다가 P2P 통신을 통해 미디어 데이터를 요청할 때 이를 활용하여 P2P 성공률이 가장 높을 것으로 기대되는 피어에게 우선적으로 미디어 데이터를 요청하는 새로운 알고리듬을 제안하고 이를 구현하였다. 그 결과, 실험실 내 실험을 통해 저속 네트워크 피어들이 많은 환경에서도 CDN 서버 부하를 크게 줄일 수 있음을 보였다.

본 논문의 구성은 다음과 같다. 2절에서는 WebRTC 기반 P2P 통신 병용 DASH 기술에 대해 설명하고, 3절에서는 이 기술의 문제점에 대해 살펴 본다. 4절에서는 본 논문에서 제안하는 알고리듬에 대해 설명하고, 5절에서는 실험을 통해 제안한 알고리듬의 성능을 확인한다. 마지막으로 6절에서는 본 논문에 대한 결론을 맺는다.

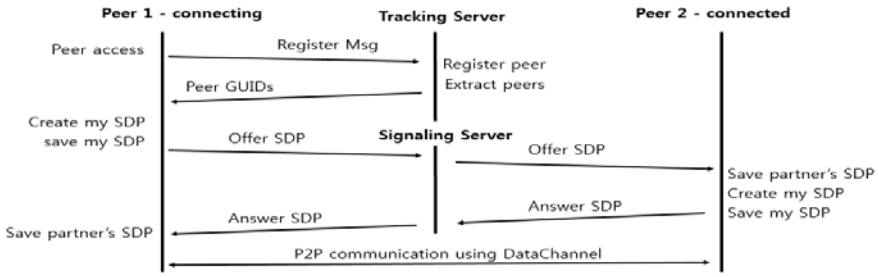

그림 1. P2P 통신 병용 DASH를 위한 P2P 연결 과정
Fig 1. P2P connection process for P2P-assisted DASH

II. WebRTC 기반 P2P 통신 병용 DASH 기술

WebRTC 기반의 P2P 통신 병용 DASH 시스템을 구현하기 위해서는 트래킹(tracking) 서버, 시그널링(signaling) 서버, 미디어 서버 등 총 3개의 서버가 필요하다.

트래킹 서버는 P2P 연결을 위한 피어 목록을 관리하는 역할을 한다. 여기서 피어는 동일한 미디어를 스트리밍 받고자 하는 클라이언트들 중 하나를 말한다. 새로운 피어가 나타나면 해당 피어를 기존에 구성된 피어 목록에 저장하고, 접속 종료 시 목록에서 이를 삭제하며 새 피어가 기존 피어들과 P2P 연결을 설정할 수 있도록 피어 목록에 있는 피어들의 GUID(Globally Unique Identifier)를 제공한다.

시그널링 서버는 피어들 간에 임의의 데이터를 보낼 수 있는 WebRTC DataChannel을 생성하기 위해 필요한 메타데이터와 후보 IP 주소들을 양쪽 피어들에게 전달하기 위한 중계 서버이다. P2P 통신에서 각 피어는 사설 IP를 사용할 수 있기 때문에 이러한 중계 서버의 도움 없이는 연결을 설정할 수 없다. 이 과정은 매우 복잡하므로 여기서는 상세 설명을 생략하고자 한다.

미디어 서버는 스트리밍을 위한 미디어 데이터 즉 DASH 세그먼트(segment)들을 제공하며 이로부터의 전송은 CDN 비용을 증가시킨다. 본 논문에서는 P2P 통신의 상대 피어와 구별하기 위해 미디어 서버를 소스(source)라 부르기로 한다.

위의 3개 서버를 이용한 WebRTC 기반 P2P 통신 병용 DASH를 위한 P2P 연결 과정[4]은 그림 1과 같다. 우선 새로운 피어가 접속하면, 트래킹 서버는 P2P 연결 생성을 위해 피어의 GUID들을 새로운 피어에게 제공한다. 이를 받은 후 새 피어는 WebRTC API를 이용하여 SDP(Session Description Protocol)[5] 형식의 메타데이터를 생성하는데 여기에 접근 가능한 후보 IP 주소들을 찾아 SDP 메타데이터에 기입한다. 이후 새 피어는 시그널링 서버를 통해 연결하고자 하는 상대 피어와 Offer/Answer 메시지를 통해 SDP 메타데이터를 교환한다. 이 과정을 트래킹 서버로부터 받은 피어의 GUID 개수만큼 반복하여 진행한다. 위의 과정이 성공적으로 수행되면 WebRTC DataChannel이 생성되며, 이후 P2P 통신은 WebRTC DataChannel을 통해 이루어진다. 이처럼 자신과 WebRTC DataChannel이 성공적으로 생성된 피어들을 이웃(neighbor)이라고 부른다. P2P 연결 직후 이웃으로부터 그들이 가지고 있는 모든 세그먼트들의 ID를 전달받고, 이후 이웃이 새로운 세그먼트를 수신할 때 추가적으로 해당 세그먼트 ID를 전달 받는다. 전달 받은 세그먼트 ID와 이 세그먼트를 갖는 이웃의 GUID를 자신의 인덱스(index)에 저장해 두었다가 자신이 필요한 세그먼트가 인덱스에 있으면 해당 이웃에게, 그렇지 않으면 소스 측에 세그먼트를 요청한다. 이로써 소스 측과 P2P 통신을 통해 이웃 측으로 세그먼트 요청하고 수신하는 것이 가능하게 된다.

WebRTC 기반의 P2P 통신 병용 DASH 시스템의 구조와 프로토콜 스택(stack) 등 보다 더 상세한 내용에 대해서는 [3]을 참고하기 바란다.

III. 기존의 피어 선택 방법

기존의 방식[3]에서는 그림 2의 과정에 따라 세그먼트를 요청한다. 우선 자신이 관리하고 있는 인덱스 내에 원하는 세그먼트 ID가 있는지 조사한다. 만약 없다면 이웃으로부터 전송 받는 것은 불가능하므로 소스 측으로부터 세그먼트를 받는다. 만약 있다면 해당 세그먼트를 가진 이웃 중 연결 상태가 정상인 것들을 추린 후 그 중 임의로 하나의 피어를 선택하여 해당 세그먼트를 전송해 주도록 요청한다. 이렇게 P2P 통신을 이용하여 세그먼트를 받게 되면 소스로부터 받는 데이터 양이 줄어 들기 때문에 CDN 비용이 감소한다. P2P 측이든 소스 측이든 요청이 정상적으로 완료되어 세그먼트를 받게 되면 이를 DASH 미디어 플레이어 측으로 전달함으로써 일련의 과정이 종료된다.

이렇게 임의로 요청할 피어를 선택하는 “랜덤(random)” 방법은 모든 이웃이 P2P 통신을 위한 충분한 속도의 네트워크 환경을 가질 때에는 문제가 없으나, 그렇지 않을 경우 저속 피어가 선택되면 P2P 통신을 통한 세그먼트 전달이 원활하지 못하여 결국 소스 측으로 해당 세그먼트를 요청하는 일이 발생한다. 이로 인해 CDN 서버 부하 절감 효과가 감소하게 된다. 이러한 문제점을 개선하기 위해 다음 장에서 저속 이웃을 선택할 가능성이 낮은 알고리듬을 새롭게 제안한다.

IV. 제안하는 피어 선택 방법

본 논문에서 새롭게 제안하는 방법에서는 이웃의 네트워크 상태 즉, RTT(Round-Trip Time)와 전송 속도에 대한 최근 이력 정보를 이용하여 보다 나은 피어 측으로 세그먼트 요청을 보낸다. 제안하는 알고리듬을 그림 3에 보였다. 이웃의 보유 세그먼트 ID들을 인덱스에 기입하는 등 전반부 과정은 기존 방법과 같으나 피어의 우선 순위(priority)를 이용하여 세그먼트 요청을 보낼 피어를 선택하는 후반부 과정은 기존 방법과 다르다. 이 방법에서는, 최초 P2P 연결 시, 연결된 피어들의 우선 순위를 디폴트(default) 값인 3으로 설정한다. 이후 P2P 요청을 보낼 피어를 선택하는 과정에서는 각 피어의

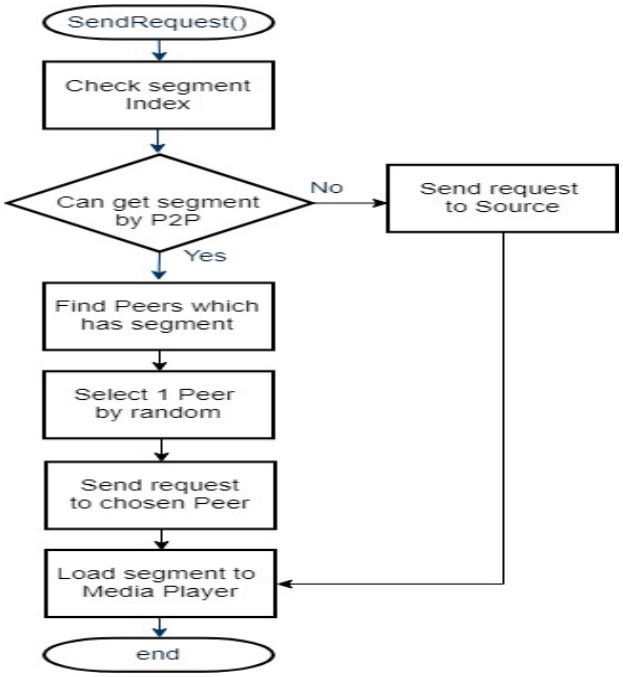

그림 2. 랜덤 방법을 이용한 피어 선택 알고리듬
Fig 2. Peer selection algorithm using random method

네트워크 환경에 대한 최근 이력에 따라 책정되는 우선 순위를 활용하여 가장 높은 순위의 피어를 선택한다. 우선 순위가 가장 높은 피어가 하나라면 해당 피어로 요청을 보내지만, 2개 이상이라면 그 중에서 주기적으로 측정해 두었던 핑(ping) 메시지의 평균 RTT를 구하여 가장 짧은 시간을 기록한 피어를 선택하여 요청을 보낸다. 세그먼트를 전송 받은 후 전송 속도 측정치(받은 데이터 양/요청 완료에 걸린 시간)를 이용하여 요청한 피어의 우선 순위를 조정한다. 만일 요청 과정에서 오류가 발생한다면 우선 순위를 2만큼 감소시키고, 1,200KB/s 미만의 속도라면 1만큼 감소시키며, 2,400KB/s 이상의 속도라면 1만큼 증가시킨다. 여기서, 문턱값 1,200KB/s와 2,400KB/s는 국내 모바일 네트워크의 현황을 고려하여 설정된 값으로서, 전자는 CDN 서버 부하 절감비가 100%에 근접하도록 하기 위한 최소 전송 속도이고 후자는 통상적 LTE 상황에서의 평균 전송 속도이다. 이 때 우선 순위는, 최근 네트워크 상태만을 이용하기 위해, 그 상한 값을 5로 제한한다. 이는 꾸준히 높은 전송속도를 유지하던 피어라 하더라도, 그 네트워크 환경이 악화되었을 때, 신속히 선택에서 배제하기 위함이다. 이러한 방법을 통해 기존 랜덤 피어 선택 방법에 비해 보다 좋은 네트워크 환경을 가진 피어를 선택할 수 있다. 이에 따라 CDN 서버 부하 절감 효과가 크게 증가한다.

V. 실험 및 결과 분석

1. 실험 환경 및 방법

실험을 위해 사용한 웹 브라우저는 현재 WebRTC 표준이 가장 잘 지원되고 있는 크롬 브라우저(69.0.3497.81 버전)이다. 실험에 사용된 DASH 콘텐츠는 Envivio사에서 제공하는 재생시간이 260s인 콘텐츠[6]이다. 또 피어들의 네트워크 환경을 에뮬레이션하기 위해 NetLimiter(4.0.31.0 버전)[7]를 사용하였다.

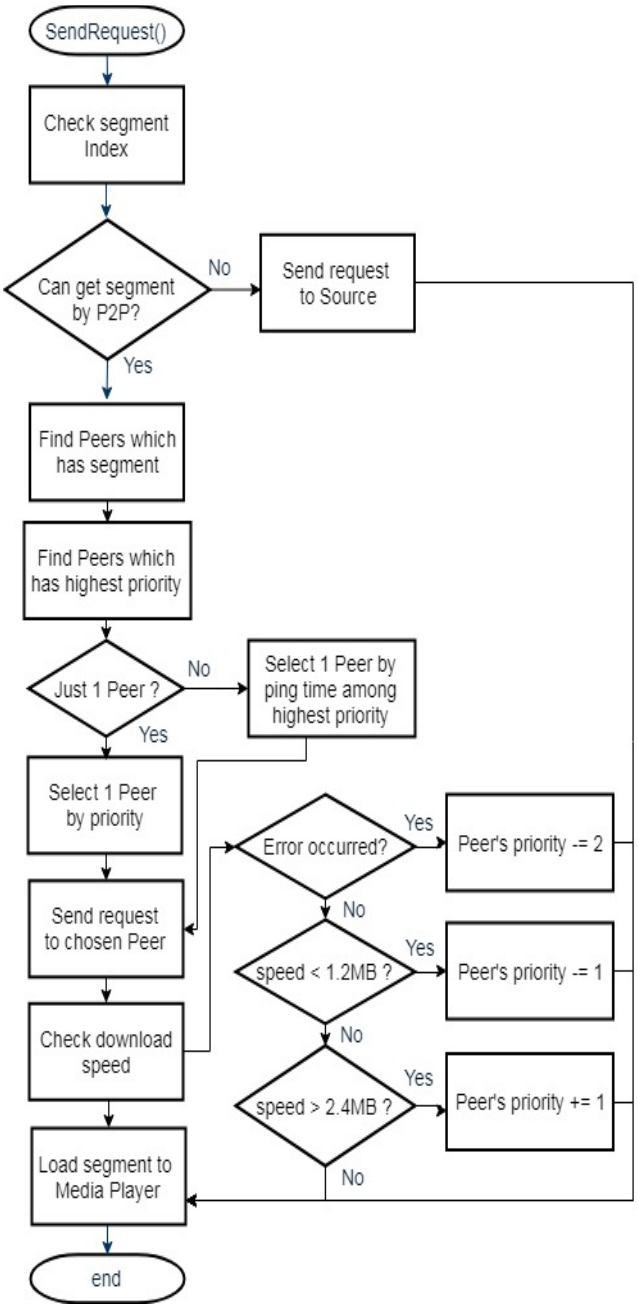

그림 3. 우선 순위에 의한 피어 선택 알고리듬
Fig 3. Peer selection algorithm using priority

미디어 HTML 페이지의 자바스크립트, 트래킹 서버 프로그램, 시그널링 서버 프로그램 등은 오픈 소스[8]의 내용을 수정하여 사용하였다.

하나의 클라이언트 피어는 9개의 이웃을 가지며, 각 이웃은 미리 미디어 사이트에 접속하여 콘텐츠를 플레이함으로써 해당 세그먼트들을 갖고 있도록 하였다. 그 후 클라이언트 피어가 미디어 사이트에 접속하여 콘텐츠를 재생한 후 클라이언트 피어 측의 "CDN 서버 부하 절감비"(P2P로 받은 데이터 양/총 받은 데이터 양)를 측정하였다. 9개의 이웃 중 각기 0, 2, 4, 6, 8개가 저속 피어가 되도록 네트워크 에뮬레이터를 설정한 5가지 실험을 시행하되 각 실험은 3회 시행하였다. 우선 순위를 이용한 피어 선택 방법에 한하여 9개 모두 저속 이웃인

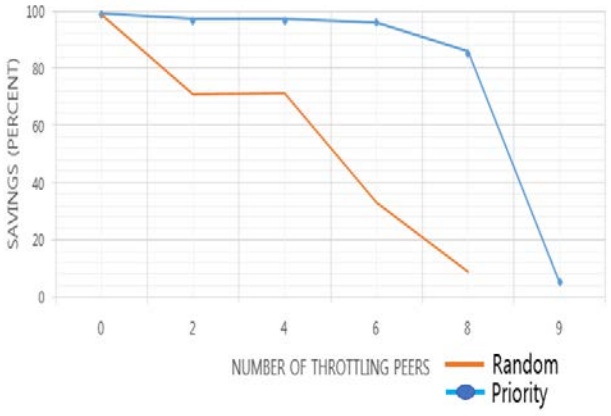

그림 4. 두 알고리듬의 성능 비교
Fig 4. Performance comparison of the two algorithms

경우를 추가 실험하였다. 이 때 네트워크 에뮬레이터는 정상 속도 피어의 경우, 2017년 기준[9]의 LTE 평균 속도 즉, 다운로드 속도 15.85MB/s와 업로드 속도 4.05MB/s로 설정하였고, 저속 피어의 경우, 실험을 위해 임의의 속도를 정하였는데, 다운로드 및 업로드 속도 모두 200KB/s로 설정하였다.

2. 실험 결과 및 분석

실험 결과는 그림 4와 같다. 랜덤 피어 선택 방법을 사용한 경우, 저속 이웃이 많을수록 저속 피어를 선택하여 세그먼트를 요청할 확률이 높아진다. 저속 피어에게 세그먼트를 요청하면 느린 전송 속도로 인해 기준 시간 내에 전송을 마치지 못하게 되는 일이 발생할 수 있다. 이 경우 클라이언트 피어는 소스 측으로 해당 세그먼트에 대한 요청을 다시 보내게 되는데 이로 인해 CDN 서버 부하 절감 효과(그림 4에서 "Saving"으로 표기)가 감소된다. 실험 결과에서도 저속 피어 수가 많을수록 이 효과가 급격히 감소됨을 알 수 있다.

우선 순위를 이용한 피어 선택 방법의 실험 결과에서도 저속 피어 수가 증가함에 따라 약간의 성능 저하를 관찰할 수 있다. 이는 스트리밍 초반에서의 성능 저하로 인한 것이다. 왜냐하면, 처음 클라이언트 피어가 9개의 이웃과 연결될 때 모든 이웃의 우선 순위를 디폴트 값인 3으로 동일하게 설정하므로 핑 메시지의 평균 RTT만으로는 정상 속도의 피어와 저속 피어를 정확히 구분해내지는 못하기 때문이다. 이에 따라 스트리밍 초반에는 저속 피어의 수가 많을수록 이러한 경향이 더 심한데 그 영향이 그림 4의 결과에도 보인다. 그럼에도 불구하고 전체 실험 결과에서 피어의 우선 순위를 이용하는 방법은 기존의 랜덤 피어 선택 방법에 비해 매우 높은 성능을 보였다. 가장 큰 차이를 보이는 실험 결과는 저속 피어 수가 8개일 때인데, 랜덤 피어 선택 방법은 약 9%, 우선 순위에 의한 피어 선택 방법은 약 85%의 "Saving"을 기록하였다. 이 결과는 새로운 방식이 연결 초기 이후 정상 속도를 가진 피어를 잘 찾아내었고, 해당 피어로 세그먼트를 요청하였음을 알 수 있다. 가장 빠른 피어에게 과부화가 걸릴 수 있는 문제 또한 자연스럽게 해결되었음을 알 수 있다. 이는 하나의 피어에게 요청이 집중될 시, 자연스럽게 전송 속도가 느려지고 이에 따라 해당 피어의 우선 순위가 낮아지므로 다른 피어들에게 요청이 분산되기 때문이다.

VI. 결론

본 논문에서는 WebRTC 기반의 P2P 통신 병용 DASH 시스템에서 CDN 서버 부하 절감 효과을 크게 개선할 수 있는 새로운 알고리듬을 제안하고 이를 구현하였으며 네트워크 에뮬레이터를 활용한 실험을 통해 그 성능을 검증하였다. 향후 실제 모바일 네트워크 환경에서의 실험을 통해 성능을 검증하는 것이 필요하다.

감사의 글

이 논문은 2018년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (2017-0-00176, 지상파 UHD 방송 기반 융합플랫폼 및 서비스 기술 개발)

참고 문헌

[1] H. Tsukayama, Youtube: The future of entertainment is on the web, http://www.washingtonpost.com/business/technology/youtube-the-future-of-entertainment-is-on-the-web/2012/01/12/gIQADpdBuP\story.html, Jan. 2012.

[2] W3C, "WebRTC 1.0: Real-time Communication between Browsers," https://w3c.github.io/webrtc-pc/, Sept. 2018.

[3] R. Roverso and M. H¨ogqvist, "Hive.js: Browser-Based Distributed Caching for Adaptive Video Streaming" in *2014 IEEE International Symposium on Multimedia*, pp. 143-146, 2014.

[4] B. Sredojev, D. Samardzija, and D. Posarac. "WebRTC technology overview and signaling solution design and implementation" in *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 1006-1009, May 2015.

[5] IETF - SDP, https://tools.ietf.org/html/rfc4566.

[6] http://dash.edgesuite.net/envivio/dashpr/clear

[7] https://www.netlimiter.com/.

[8] https://github.com/Peerialism/hive.js.

[9] 과학기술정보통신부,
2017년 통신서비스 품질평가 결과 발표,
https://www.msit.go.kr/web/msipContents/contentsView.do?cateId=mssw311&artId=1371275.