특집논문 (Special Paper)
방송공학회논문지 제24권 제2호, 2019년 3월 (JBE Vol. 24, No. 2, March 2019)
https://doi.org/10.5909/JBE.2019.24.2.251
ISSN 2287-9137 (Online) ISSN 1226-7953 (Print)

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

서 주 호[a], 최 성 현[a], 김 상 진[b], 전 재 영[b], 김 용 한[a)‡]

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

Ju Ho Seo[a], Seong Hyun Choi[a], Sang Jin Kim[b], Jae Young Jeon[b], and Yong Han Kim[a)‡]

요 약

최근 인터넷 미디어 스트리밍 수요의 증가로 인해 CDN(Content Delivery Network) 비용이 크게 증가하였으며 이를 절감하기 위한 방안의 필요성이 날로 증가하고 있다. 이러한 상황에 맞춰 최근 CDN 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, 전달 이력에 기반하여 피어(peer)를 선택하게 함으로써, 이 기술의 CDN 비용 절감 효과를 크게 개선할 수 있는 새로운 알고리듬을 제안하였다. 또한 실험실 내에서 실제 이 알고리듬을 구현하고 모바일 네트워크 환경과 유사한 환경 조건에서 실험을 시행하여 그 성능을 측정한 결과, 무작위로 피어를 선택하는 기존의 시스템과 비교하여 더 높은 CDN 비용 절감 효과를 달성할 수 있음을 보였다.

Abstract

Recently the huge demand for Internet media streaming has dramatically increased the cost of the CDN (Content Delivery Network) and the need for a means to reduce it is increasing day by day. In this situation, a P2P-assisted DASH technology has recently emerged, which uses P2P (Peer-to-Peer) communications based on WebRTC (Web Real-Time Communication) standards to reduce the CDN cost. This paper proposes an algorithm that can significantly improve CDN cost savings in this technology by selecting peers based on the transport history. Also we implemented this algorithm in an experimental system and, after setting experimental conditions that emulate the actual mobile network environment, we measured the performance of the experimental system. As a result, we demonstrated that the proposed algorithm can achieve higher CDN cost savings compared to the conventional algorithm where peers are selected at random.

Keywords: P2P-assisted streaming, DASH, WebRTC, CDN cost savings

a) 서울시립대학교 일반대학원 전자전기컴퓨터공학과(Department of Electrical and Computer Engineering, Graduate School, University of Seoul)
b) ㈜SBS(SBS Co. Ltd.)
‡ Corresponding Author : 김용한(Yong Han Kim)
E-mail: yhkim@uos.ac.kr
Tel: +82-2-6490-2330
ORCID: https://orcid.org/0000-0001-9470-6060
※ This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT). (2017-0-00176, Hybrid Platform and Service Technology Development Based on Terrestrial UHD Broadcast)
※ 이 논문의 연구 결과 중 일부는 "2018년도 한국방송·미디어공학회 추계학술대회"에서 발표한 바 있음.
· Manuscript received January 15, 2019; Revised March 11, 2019; Accepted March 11, 2019.

Ⅰ. 서 론

최근 몇 년 동안 인터넷 미디어 스트리밍은 급격히 증가하였으며 2012년에 시행된 연구에 따르면 인터넷 대역폭의 90%가 미디어일 것으로 예측되고 있다[1]. 현재 인터넷 미디어 스트리밍은 대부분 HTTP를 기반으로 하는 적응형 미디어 스트리밍 기술에 의해 시행되고 있다. 이 기술에서는 클라이언트로 하여금 가능한 한 CDN(Content Delivery Network) 서버가 아니라 HTTP 캐시(cache)로부터 미디어 데이터를 가져오도록 함으로써, CDN 서버의 부하를 줄이고 원활한 스트리밍을 달성할 수 있도록 하고 있다. 이로 인해, 인터넷 미디어 스트리밍 서비스가 폭증함에 따라, ISP(Internet Service Provider)들은 HTTP 캐시를 증설해야 했고, 마침내 이러한 증설 비용 부담을 줄이기 위해 CDN 서버로부터의 트래픽뿐만 아니라 HTTP 캐시로부터의 트래픽에 대해서도 CDN 사업자에게 과금을 하기에 이르렀다. 따라서 인터넷 미디어 스트리밍 서비스를 제공하는 CDN 사업자의 입장에서는 미디어 배포량이 증가함에 따라 CDN 서버 증설 비용뿐만 아니라 HTTP 캐시 트래픽 비용까지 증가하게 되어 비용 절감에 대한 필요성이 점증하고 있다.

WebRTC는 웹 브라우저 간 실시간 통신을 가능케 하는 API(Application Programming Interface) 표준[2]으로서 WebRTC를 지원하는 웹 브라우저 간 실시간 영상 통신, 음성 통신, 채팅 등을 쉽게 구현할 수 있게 한다. 최근 WebRTC를 기반으로 P2P 통신을 병용함으로써 HTTP 기반 적응형 미디어 스트리밍 시스템의 효율을 높일 수 있는 기술이 연구되었다[3]. 이 기술은 Roverso 등[4]에 의해 개발된 SmoothCache 시스템과 유사한 아이디어를 기반으로 하는 시스템이지만, WebRTC 표준을 사용한 시스템이다. [3]에서 제안한 시스템은 'Hive' 시스템이라 불리며, HTTP 기반 적응형 미디어 스트리밍 방식 중 DASH를 사용하였다. Hive 시스템에서는 어떤 클라이언트가 DASH 세그먼트(segment)를 필요로 할 때, 이를 CDN 서버로 요청하기에 앞서, 동일한 콘텐츠를 시청하고 있는 다른 피어(peer)에게 해당 DASH 세그먼트를 요청함으로써 CDN 서버 또는 HTTP 캐시로부터의 트래픽을 크게 줄일 수 있다. 해당 연구는 미디어를 스트리밍할 때 QoS(Quality of Service)를 낮추지 않으면서 CDN 사업자의 비용을 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 피어들은 자신이 가지고 있는 미디어 데이터 정보를 서로에게 알려준다. 피어들은 이 정보를 이용하여 특정 미디어 데이터가 필요할 때 해당 미디어 데이터를 가진 피어들 중 무작위로 선택된 한 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 서버에로의 요청이 증가하게 되어 'CDN 비용 절감비'(P2P를 통해 받은 데이터 양/총 받은 데이터 양)가 저하된다.

Hive 시스템[3]이 등장한 이후, Varma[5] 등은 DASH 대신 HLS(HTTP Live Streaming)[6]를 사용하여 실시간 스트리밍 시나리오에서 캠퍼스 LAN에서의 실험을 통해 Hive와 유사한 시스템이 CDN 비용을 약 75%로 줄일 수 있음을 보였다. 한편, Liu 등[7]은 통신료 부담이 없는 블루투스 간 P2P를 사용하여 CDN 비용을 크게 줄일 수 있음을 보였다. SmoothCache 시스템[4]과 Hive 시스템[3]은 메시(mesh) 형태의 P2P 오버레이(overlay) 망을 사용하였으나, [7]의 시스템은 트리(tree) 구조의 P2P 망을 사용하였다. 메시 형태의 P2P 오버레이 망을 사용하는 경우, 여러 피어가 데이터를 제공할 수 있는 경우, 어떤 피어를 선택할지가 문제가 되는데, Hive 시스템에서는 무작위로 하나를 선택하였고, SmoothCache 시스템에서는 피어의 과거 쓰루풋(through-put)을 참조하여 성공가능성이 높은 피어를 선택하였다.

본 논문에서는, 클라이언트(client) 피어가 피어 목록에 있는 피어들의 미디어 데이터 전달 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 알고리듬

을 구체적으로 제안하고 이를 사용하여 Hive 시스템을 개선하였다. 본 논문에서 제안하는 알고리듬의 기본적인 피어 선택 아이디어는 SmoothCache의 그것과 유사하지만, SmoothCache의 해당 논문[4]에서는 피어 선택 알고리듬을 구체적으로 제시하지 않았다. 또한 [5]에서도 피어 선택 방법에 대해서는 언급이 없었다. 본 논문에서 제안하는 알고리듬에 의해 실험실 내 실험을 시행한 결과, 저속 피어들이 많은 환경에서도 무작위 피어 선택 방식을 사용하는 기존 Hive 시스템에 비해 CDN 비용을 더 크게 줄일 수 있음을 보였다.

본 논문의 구성은 다음과 같다. II장에서는 WebRTC 기반 P2P 통신 병용 DASH 시스템 즉 Hive 시스템에 대해 설명하고, III장에서는 이 기술의 문제점에 대해 살펴본다. IV장에서는 본 논문에서 제안하는 알고리듬에 대해 설명하고, V장에서는 실험을 통해 제안한 알고리듬의 성능을 확인한다. 마지막으로 VI장에서는 본 논문에 대한 결론을 맺는다.

II. WebRTC 기반 P2P 통신 병용 DASH 기술

1. 필요 서버의 종류 및 역할

WebRTC 기반의 P2P 통신 병용 DASH 시스템을 구현하기 위해서는 트래킹(tracking) 서버, 시그널링(signaling) 서버, 미디어 서버, 웹페이지 서버 등 총 4개의 서버가 필요하다.

트래킹 서버는 P2P 연결을 위한 피어 목록을 관리하는 역할을 한다. 여기서 피어는 동일한 콘텐츠를 스트리밍 받고자 하는 클라이언트들 중 하나를 말한다. 새로운 피어가 트래킹 서버에 접속하면 트래킹 서버는 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 새 피어에게 제공한다.

시그널링 서버는 피어들 간에 P2P 연결을 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 세그먼트들을 제공하며 이로부터의 전송은 CDN 비용을 증가시킨다. 본 논문에서는 P2P 통신의 상대 피어와 구별하기 위해 미디어 서버를 '소스(source)'라 부르기로 한다.

웹페이지 서버는 스트리밍할 콘텐츠에 대한 웹페이지를 제공하는 서버로서, 이 웹페이지에는 콘텐츠에 대한 DASH MPD(Media Presentation Description) 파일의 URL과 Hive 시스템의 클라이언트 측 자바스크립트가 포함되어 있다.

2. P2P 연결 과정

위의 4개 서버를 이용한 WebRTC 기반 P2P 통신 병용 DASH를 위한 P2P 연결 과정[8]은 그림 1과 같다. 우선 새로운 피어가 접속하면, 트래킹 서버는 P2P 연결 생성을 위해 기존 피어의 GUID들을 새로운 피어에게 제공한다. 이 때 연결할 피어의 GUID는 15초마다 최대 5개씩 주기적으로 제공된다. 트래킹 서버가 관리하고 있는 피어 목록에 있는 피어의 GUID 전부를 한 번에 전달하지 않고 주기적으로 무작위로 선택된 최대 5개까지만 15초마다 전달하는 이유는, 이를 받은 새 피어가 한 번에 모든 피어와 연결을 설정하려면 과도하게 긴 지연시간이 초래되기 때문이다. 이를 받은 후 새 피어는 자신과 연결된 최대 피어 수인 10개를 넘지 않는 범위 내에서 이미 연결된 피어를 제외한 피어들의 GUID를 추린다. 새 피어는 WebRTC API를 이용하여 SDP(Session Description Protocol)[9] 형식의 메타데이터를 생성하는데 여기에 자신의 IP 주소로서 사용할 수 있는 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 위에서 추린 GUID 개수만큼 반복하여 진행한다. 이 과정이 성공적으로 수행되면 WebRTC DataChannel이 생성되며, 이후 P2P 통신은 WebRTC DataChannel을 통해 이루어진다. 이처럼 자신과 WebRTC DataChannel이 성공적으로 생성된 피어들을 '이웃(neighbor)'이라고 부르기로 한다.

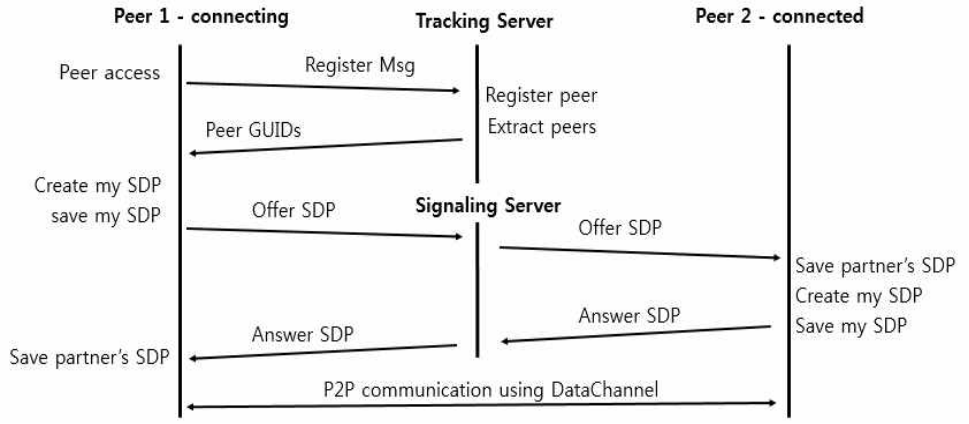

그림 1. P2P 통신 병용 DASH를 위한 P2P 연결 과정
Fig. 1. P2P connection process for P2P-assisted DASH

3. DASH 세그먼트 송수신 과정 및 클라이언트 측 구조

P2P 연결 이후 DASH 세그먼트 송수신 과정은 그림 2와 같다. P2P 연결 직후 이웃들로부터 그들이 가지고 있는 모든 세그먼트들의 ID를 전달받고, 이후 이웃들이 새로운 세그먼트를 수신할 때 추가적으로 해당 세그먼트 ID를 전달받는다. 전달 받은 세그먼트 ID와 이 세그먼트를 가진 이웃의 GUID를 자신의 세그먼트 인덱스(index)에 저장해 둔다. 이후 영상을 플레이하며 필요한 세그먼트 ID가 세그먼트 인덱스에 있으면 해당 이웃에게 요청한다. 세그먼트 인덱스에 해당 세그먼트 ID가 없거나 이웃으로부터 세그먼트 수신 중에 오류가 발생한다면 소스 측에 세그먼트를 요청한다. Hive 시스템에서는 과도한 지연시간을 피하기 위해서, 어떤 이웃으로부터 세그먼트를 받지 못했을 때, 또 다른 이웃에게 해당 세그먼트를 요청하지 않으며 즉시 소스 측으로부터 해당 세그먼트를 가져온다. 세그먼트 수신이 완료되면, 이웃들로부터 자신에게로의 세그먼트 요청이 가능하도록 수신한 세그먼트를 세그먼트 버퍼에 저장하고 세그먼트 ID를 모든 이웃들에게 전달한다. 이로써 P2P 통신을 통해 이웃들 간의 세그먼트 공유가 가능하게 된다.

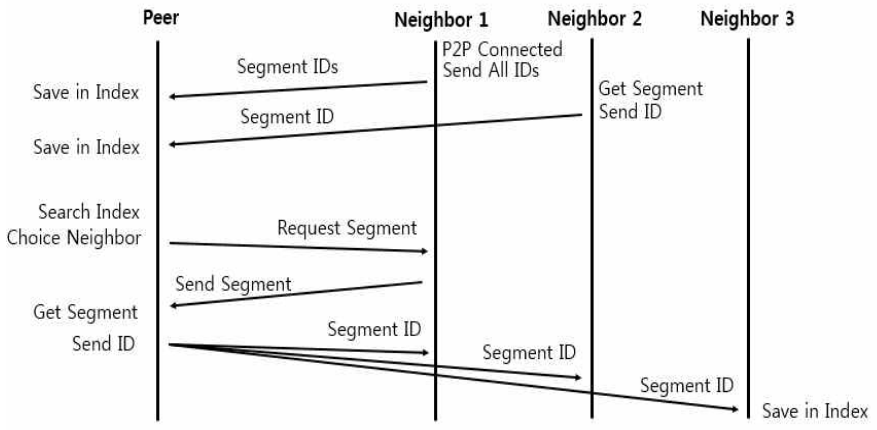

그림 2. P2P를 이용한 세그먼트 송수신 과정
Fig. 2. Transmission and reception process of DASH segments using P2P

WebRTC 기반의 P2P 통신 병용 DASH 시스템의 클라이언트 측 구조는 그림 3과 같다. 클라이언트는 시그널링 서버 및 트래킹 서버와 통신할 때, 양방향 통신이 가능한 웹소켓(websocket)을 사용한다. 클라이언트가 소스 측으로부터 세그먼트를 가져 올 때는, XHR(XmlHttpRequset)을 사용하여 HTTP를 통해 미디어 서버로부터 가져 온다. 이웃들 간에 P2P 메시지를 송수신하거나 세그먼트를 전달할 때는, UDP를 통한 WebRTC DataChannel을 이용하여 통신한다. 클라이언트가 수신한 세그먼트는 DASH 플레이어 내의 미디어 버퍼로 전달될 뿐만 아니라, 이웃의 요청이 있을 때, 해당 세그먼트를 제공할 수 있도록 DASH 플레이어 외부에 있는 세그먼트 버퍼에도 보관된다. Hive 시스템의 클라이언트 측 구조와 프로토콜 스택(stack)에 대한 보다 더 상세한 내용에 대해서는 [3]을 참고하기 바란다.

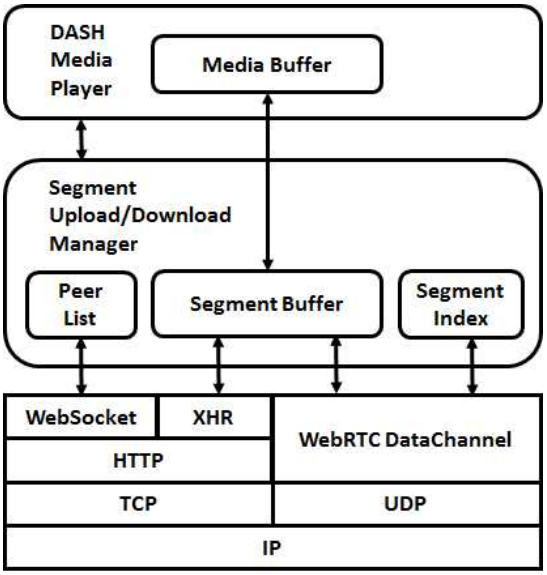

그림 3 Hive 시스템의 클라이언트 측 구조
Fig. 3. The client-side structure of the Hive system

III. 기존의 피어 선택 방법

기존의 방식[3]에서는 그림 4의 과정에 따라 세그먼트를 요청한다. 우선 자신이 관리하고 있는 세그먼트 인덱스 내에 원하는 세그먼트 ID가 있는지 조사한다. 만약 없다면,

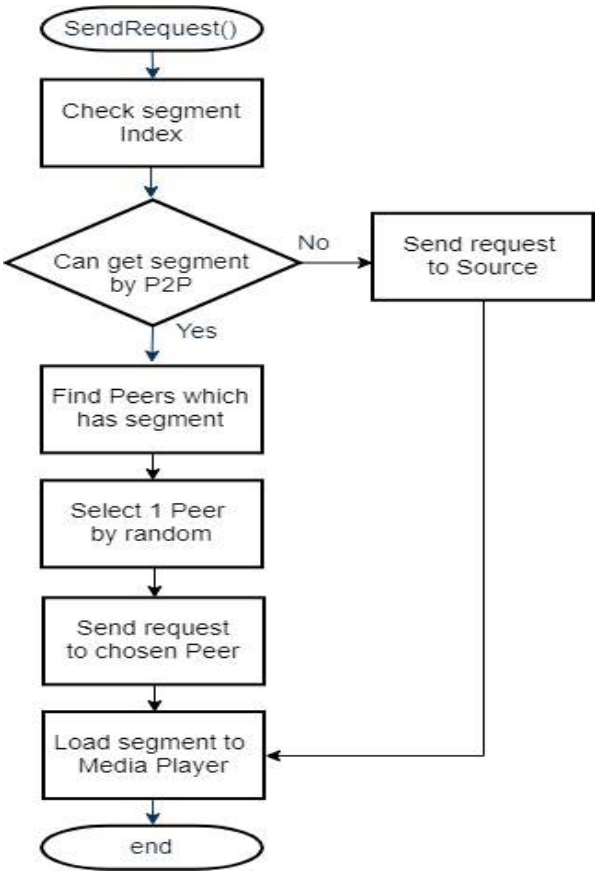

그림 4. 무작위 방법을 이용한 피어 선택 알고리듬
Fig. 4. Peer selection algorithm using random method

이웃으로부터 전송 받는 것은 불가능하므로 소스 측으로부터 세그먼트를 받는다. 만약 있다면, 해당 세그먼트를 가진 이웃들 중 연결 상태가 정상인 것들을 추린 후 그 중 무작위로 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 무작위로 세그먼트를 요청할 피어를 선택하는 '무작위(random)' 방법은 모든 이웃들이 P2P 통신을 위한 충분한 속도의 네트워크 환경을 가질 때에는 문제가 없으나, 그렇지 않을 경우 저속 피어가 선택되면 P2P 통신을

통한 세그먼트 전달이 느려서 제한시간 초과로 인해 전달에 실패하게 되어 결국 소스 측으로 해당 세그먼트를 요청하는 일이 발생한다. 이로 인해 CDN 비용 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 구체적으로 제안하는 방법에서는 이웃들의 네트워크 상태 즉, RTT(Round-Trip Time)와 세그먼트 전달 속도에 대한 최근 이력 정보를 이용하여 전달 성공 확률이 높은 이웃 측으로 세그먼트 요청을 보낸다. 제

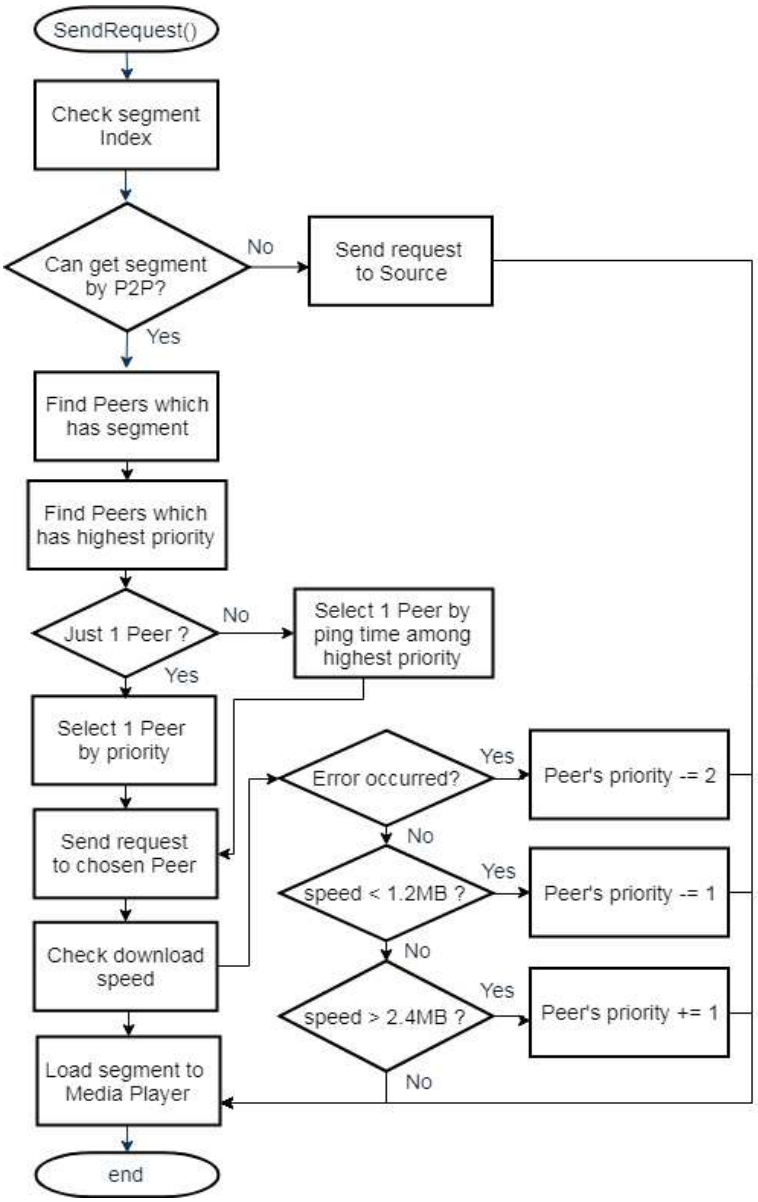

그림 5. 우선순위에 의한 피어 선택 알고리듬
Fig. 5. Peer selection algorithm using priority

안하는 알고리듬을 그림 5에 보였다. 이웃들의 보유 세그먼트 ID들을 세그먼트 인덱스에 기입하는 등 전반부 과정은 기존 방법과 같으나 이웃의 우선순위(priority)를 이용하여 세그먼트를 요청할 이웃을 선택하는 후반부의 과정은 기존 방법과 다르다. 이 방법에서는, 이웃들의 우선순위를 사용한다. 우선순위는 1에서 5의 값을 가지며 우선순위 값이 클수록 더 선호됨을 의미한다. P2P 연결 시, 모든 이웃들의 우선순위 값은 디폴트(default) 값인 3으로 초기화한다. 이후 P2P 요청을 보낼 이웃을 선택하는 과정에서는 각 이웃의 네트워크 환경에 대한 최근 이력에 따라 책정되는 우선순위를 활용하여 그 값이 가장 큰 이웃을 선택한다. 우선순위가 가장 높은 이웃이 하나라면 해당 이웃에게 요청을 보내지만, 2개 이상이라면 그 중에서 주기적으로 측정해 두었던 핑(ping) 타입의 피어 간 메시지의 평균 RTT를 구하여 가장 짧은 시간을 기록한 이웃을 선택하여 요청을 보낸다. 이 주기는 4초로 하였다. 세그먼트를 전송 받은 후 '세그먼트 획득 속도' 측정치(받은 데이터 양/요청 시작부터 완료에 걸린 시간)를 이용하여 해당 이웃의 우선순위를 조정한다. 세그먼트 획득 속도 측정치는, 하나의 세그먼트를 피어에게 요청하여 받은 세그먼트 데이터 양을 요청 시점부터 세그먼트의 마지막 바이트를 수신 완료한 시점까지의 시간으로 나눈 값이다. 따라서 세그먼트 획득 속도는 단순히 세그먼트 전달 속도에 의해서만 결정되는 것이 아니라, 세그먼트의 첫 바이트가 전달되기까지의 지연 시간에 의해서도 영향을 받는다. 즉 세그먼트 획득 속도는 RTT와 전달 속도를 모두 감안하기 위해 고안된 지표이다.

만일 요청 과정에서 오류가 발생한다면 우선순위를 2만큼 감소시키고, 세그먼트 획득 속도가 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 실험실 내 사전 실험을 통해 얻은 값이기에 다른 환경에서 해당 알고리듬을 사용할 경우 문턱값에 대한 보정이 필요할 수도 있다. 문턱값 2,400KB/s는 국내 LTE 상황[10]에서의 평균 업로드(upload) 속도인 4.05MB/s로 네트워크 에뮬레이터를 설정했을 때 실제로 실험실 내에서 측정된 평균 세그먼트 획득 속도이다. 그리고 우선순위는, 최근 네트워크 상태만을 이용하기 위해, 그 상한 값을 5로 제한한다. 이는 꾸준히 높은 세그먼트 획득 속도를 유지하던 이웃이라 하더라도, 그 네트워크 환경이 악화되었을 때, 신속히 선택에서 배제할 수 있도록 하기 위함이다. 이러한 전송 이력에 따라 우선순위를 부여하는 방법을 통해 기존 무작위 피어 선택 방법에 비해 보다 좋은 네트워크 환경을 가진 이웃을 선택할 수 있다. 이에 따라 CDN 비용 절감 효과가 커진다.

이 알고리듬에서 초기에 RTT를 사용하는 것은 모든 피어가 동일한 우선순위를 갖도록 초기화된 상태에서 세그먼트 획득 속도 측정치가 없으므로, 다른 적절한 지표가 없기 때문이다. 실제 네트워크 환경에서는 RTT가 작다고 해서 반드시 세그먼트 전달 속도가 빠르다고 할 수는 없기 때문에, 초기에 RTT를 사용하는 것이 적절한가에 대해서는 더 연구가 필요하다. 다만, RTT는 초기에만 사용되므로 알고리듬의 전반적인 성능에 큰 영향을 끼치지는 않는다. 다만 V장의 실험에서는 세그먼트 전달 속도가 빠른 피어가 RTT도 작도록 네트워크 에뮬레이터를 설정하고 실험을 시행하였다.

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 미디어 서버는 별도로 구축하지 않았으며, DASH-IF(Industry Forum)에서 제공하는 미디어 서버를 직접 사용하였으며, 해당 서버에서 제공하는 크기가 91MB이고 재생시간이 260s인 Envivio사의 DASH 콘텐츠[11]를 실험용으로 사용하였다. 또 피어들의 네트워크 환경을 에뮬레이션하기 위해 NetBalancer(9.12.9버전)[12]를 사용하였다.

콘텐츠 웹페이지의 자바스크립트, 트래킹 서버 프로그램, 시그널링 서버 프로그램 등은 Hive 시스템 오픈 소스[13]의 내용을 수정하여 사용하였다. 단, 제안하는 피어 선택 알고리듬의 성능을 기존 Hive 시스템의 무작위 피어 선택 방법의 성능과 같은 조건에서 비교하기 위해, 트래킹 서버로부터 피어 목록을 전달받는 주기(15초), 이 때 전달 받는 피어

목록의 최대 피어 수(5개), 각 피어가 P2P 연결을 유지하는 최대 이웃의 수(10개), 이웃에게 세그먼트를 요청한 이후의 타임아웃 시간(5초), DASH 플레이어의 초기 버퍼링 양(10초) 등의 각종 파라미터는 Hive 시스템에서 사용하는 것을 그대로 사용하였다.

하나의 클라이언트 피어는 9개의 이웃을 가지며, 각 이웃들은 미리 미디어 사이트에 접속하여 콘텐츠를 플레이함으로써 해당 콘텐츠의 모든 세그먼트들을 갖고 있도록 하였다. 그 후 클라이언트 피어가 미디어 사이트에 접속하여 콘텐츠를 재생을 완료한 후 클라이언트 피어 측의 'CDN 비용 절감비'(P2P를 통해 받은 데이터 양/총 받은 데이터 양)를 측정하였다. 9개의 이웃들 중 각기 0, 2, 4, 6, 8개가 저속 피어가 되도록 네트워크 에뮬레이터를 설정한 5가지 실험을 시행하되, 각 실험은 3회 시행하여 CDN 비용 절감비의 평균값을 산출하였다. 이 때 네트워크 에뮬레이터는 정속 피어의 경우, 2017년 기준[10]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. 저속 피어의 경우, 네트워크 에뮬레이터가 30ms의 추가 응답 지연시간을 부여하도록 설정하였다.

또, 스트리밍을 시작한 이후 이웃들의 상태가 동적으로 바뀌었을 때, 제안한 알고리듬이 정속 피어를 얼마나 신속히 찾아내는지를 확인하기 위해 추가적인 실험을 시행하였다. 이 실험에서는 초기에 9개의 이웃 중에서 4개는 정속 피어로, 5개는 저속 피어로 구성하되, 스트리밍 시작 2분 후 저속 피어는 모두 정속 피어로, 또 정속 피어는 모두 저속 피어로 전환시킴으로써 이웃들의 상태에 동적 변화를 주었다.

2. 실험 결과 및 분석

무작위 피어 선택 방법과 전달 이력에 기반한 우선순위에 의한 선택 방법을 비교한 실험 결과는 그림 6과 같다. 무작위 피어 선택 방법을 사용한 경우, 저속 이웃들이 많을수록 저속 이웃을 선택하여 세그먼트를 요청할 확률이 높아진다. 그림 6에서 보인 바와 같이, CDN 비용 절감비는 저속 이웃 수에 대체로 반비례함을 알 수 있다. 저속 이웃에

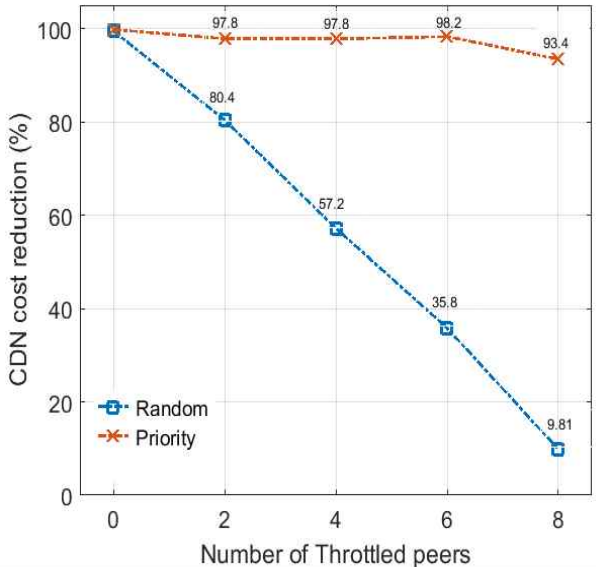

그림 6. 두 알고리듬의 성능 비교
Fig. 6. Performance comparison of the two algorithms

게 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 비용 절감 효과가 감소된다. 실험 결과에서도 저속 이웃 수가 많을수록 이 효과가 급격히 감소함을 알 수 있다.

우선순위를 이용한 피어 선택 방법의 실험 결과에서는 저속 이웃 수에 거의 무관하게 CDN 비용 절감비가 높게 유지되는 것을 관찰할 수 있다. 여기서 CDN 비용 절감비가 100%에 이르지 못하는 이유는 스트리밍 시작 직후부터 얼마 동안은 세그먼트 전달 이력이 축적되어 있지 않았기 때문이다. 또한, 저속 이웃 수가 많을수록 스트리밍 초기 단계에서 CDN 성능 절감비가 작아질 가능성이 크다. 왜냐하면 처음 클라이언트 피어가 이웃들과 연결될 때 모든 이웃들과 연결되는 것이 아니라 5개, 4개가 15초의 시간차를 두고 연결되기 때문이다. 그러므로 초기에 연결된 5개의 이웃이 모두 저속 이웃일 경우, 어쩔 수 없이 저속 이웃에게 요청을 보낼 수밖에 없고 이에 따른 성능 저하가 발생한다. 이러한 현상은 그림 6에서 저속 이웃 수가 8개일 때 확연히 관찰될 수 있다. 그럼에도 불구하고 전체 실험 결과에서 우선순위를 이용하는 방법은 기존의 무작위 피어 선택 방법에 비해

매우 높은 성능을 보였다. 가장 큰 차이를 보이는 실험 결과는 저속 이웃 수가 8개일 때, 즉 정속 이웃이 하나뿐일 때인데, 무작위 피어 선택 방법은 약 10%, 우선순위에 의한 피어 선택 방법은 약 93%의 CDN 비용 절감비를 기록하였다. 이 결과로부터 새로운 알고리듬이 정속 이웃을 잘 찾아내었음을 알 수 있다.

그림 7은 시험 콘텐츠의 모든 세그먼트에 대해 요청한 곳을 보여 주며, 제안한 알고리듬이 어떤 과정을 거쳐 정속 피어를 찾게 되는지를 보여 주는 그림이다. 그림 7은 9개의 이웃 중, 저속 이웃이 8개 즉, 정속 이웃이 하나뿐인 경우, 첫 세그먼트(0번 세그먼트)부터 마지막 세그먼트(64번 세그먼트)까지 순차적으로 그 요청한 곳을 보여 준다. 그림 7의 가로축에서 1번에서 8번까지 피어는 저속 이웃이며 9번 피어는 정속 이웃이고, 가장 아래에 CDN 서버 측으로부터 요청한 경우를 'source'라고 표기하였다. 세그먼트 전달에 성공한 요청은 '*'로 표기하고, 실패한 요청은 네모로 표기하였다. 스트리밍 시작 시 모든 이웃의 우선순위는 3으로 같기 때문에 저속 이웃이 선택될 수 있으며, 그림 7에서는 그 중 RTT가 가장 빠른 것으로 측정된 이웃 3이 선택된다. 이 실험에서는, 저속 이웃의 추가 응답 지연시간이 네트워크 에뮬레이터에 의해 30msec로 설정되어 있음에도 불구하고, 첫 요청 대상 피어로서 이웃 3이 선택되었고, 이후 이웃 1, 7, 8을 거쳐 마침내 이웃 9가 선택되었다. 이는 이웃들의 실제 상태에 따라 이웃 9의 RTT가 가장 작은 것으로 측정되지 못했기 때문이지만, 에뮬레이터가 부여한 추가 응답 지연시간으로 인해 정속 이웃인 이웃 9를 찾아가는 시간을 줄일 확률은 높아진다. 논문에 싣지는 않았으나, 다른 실험 결과들을 종합적으로 검토한 결과, 제안한 알고리듬에서 초기에 우선순위가 동일한 이웃 중에서 RTT가 작은 이웃을 선택하도록 함으로써, 정속 이웃을 빨리 찾을 확률이 높아짐을 알 수 있었다. 물론 이것은 이 부분의 가정 즉, 전달 속도가 빠른 이웃의 RTT가 작다는 가정 하에서 네트워크 에뮬레이터를 그에 맞게 설정한 데에 따른 것으로서, 이러한 가정이 실제와 맞지 않는다고 하더라도 그 영향은 초기에 정속 이웃을 찾기까지의 과정 내로 국한된다. 최악의 경우, 이 실험에서는 이웃 1부터 이웃 8까지 모두 한 차례씩 시도해 본 후, 세그먼트 획득 속도에 대한 측정이 이루어지면 결국 이웃 9를 찾아내게 될 것이다. 스트리밍 초기에는 DASH 플레이어 내의 미디어 버퍼가 비어 있으므로 이를 빠르게 채우기 위해, DASH 플레이어는 첫 번째

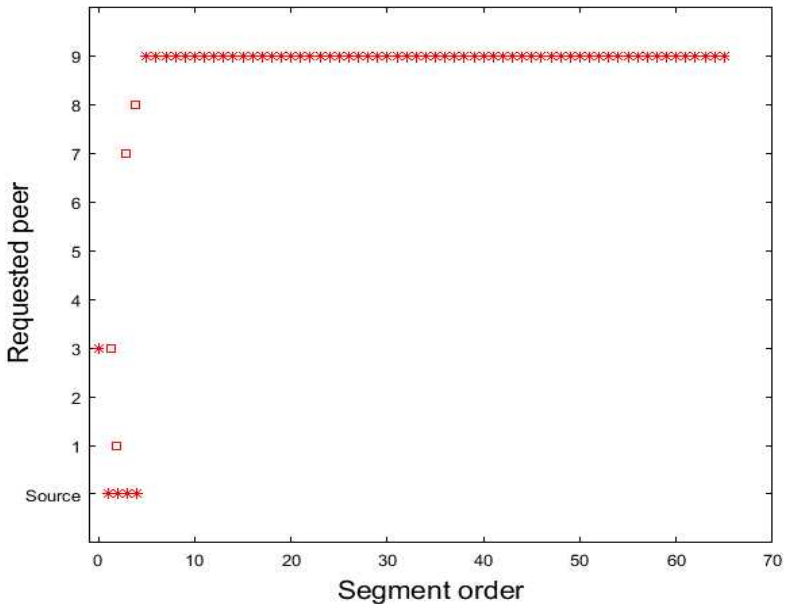

그림 7. 저속 피어 8개 실험에서 세그먼트 요청을 위해 선택된 피어
Fig. 7. Peers selected for segment requests in the experiment with 8 throttled peers

와 두 번째 세그먼트를 한꺼번에 요청하게 되고, 결국 클라이언트 피어는 이들을 이웃 3에게 요청하게 된다. 이웃 3은 첫 번째 세그먼트 전달에는 성공하지만 두 번째 세그먼트 전달에는 실패하게 되며, 그림 7에 이러한 현상이 표기되어 있다. 이 때, 첫 번째 세그먼트의 전달을 성공하더라도 세그먼트 획득 속도가 문턱값 1,200KB/s를 넘지 못하기에 이웃 3의 우선순위가 2로 감소되며, 두 번째 세그먼트 전달에 실패하여 다시 최하값인 1까지 우선순위가 낮아진다. 이렇게 우선순위가 낮아진 이웃 3은 추후 선택에서 배제된다. 전달 실패한 세그먼트를 소스 측으로부터 다시 받으므로, 이로 인해 CDN 비용 절감비가 감소한다. 이웃 1, 7, 8 또한 이후 세그먼트 전달에 실패하여 우선순위가 3에서 1로 감소되어 추후 선택에서 배제된다. 이처럼 제안한 알고리듬은 세그먼트 전달 이력을 쌓아가며 저속 이웃을 구분해 나간다. 이후 정속 이웃인 이웃 9에게 세그먼트 요청을 하게 되면, 피어 9는 세그먼트 획득 속도가 문턱값 2,400KB/s를 넘으므로 우선순위가 4로 증가하여, 이웃들 중 가장 높은 우선순위를 갖게 된다. 따라서 다음 세그먼트 요청에서 재선택되고, 세그먼트 전달에 성공함으로써 그 우선순위는 최댓값인 5가 된다. 이후 모든 세그먼트 요청을 이웃 9에게 함을 확인할 수 있다. 또한 성공적인 세그먼트 전달로 인해 소스 측으로의 요청이 발생하지 않으므로 CDN 비용 절감이 이루어진다. 이를 통해 제안한 알고리듬이 전달 이력이 축적된 이후 이를 활용하여 정속 피어를 잘 선택하고 CDN 비용 절감을 달성하는 것을 확인할 수 있다.

그림 8은 동적으로 이웃들의 상태를 변화시킨 실험의 결과이다. 해당 실험의 CDN 비용 절감비는, 동일한 실험을 3회 시행한 결과, 평균 86.53%로 측정되었다. 그림 8에서 스트리밍 시작 시 이웃 1 ~ 5는 저속 이웃으로, 이웃 6 ~ 9는 정속 이웃으로 설정하며, 시작 2분 후 이웃 1 ~ 5는 정속 이웃으로, 이웃 6 ~ 9는 저속 이웃으로 전환시켰다. 그림 8의 표기 방법은 그림 7과 동일하다. 스트리밍 초기에는 앞서 그림 7과 유사한 과정을 거쳐 정속 이웃을 찾는다. 저속 이웃인 이웃 5는 두 번째 세그먼트 전달 실패로 인해 그 우선순위가 감소하여 이후 선택에서 배제되고, 그 직후 정속 이웃인 이웃 9가 선택되어 계속 세그먼트를 제공하는 이웃 역할을 하게 된다. 그러나 정속 이웃들이 저속 이웃들로 전환된 후, 이웃 9는 41번째와 42번째 세그먼트 전달에

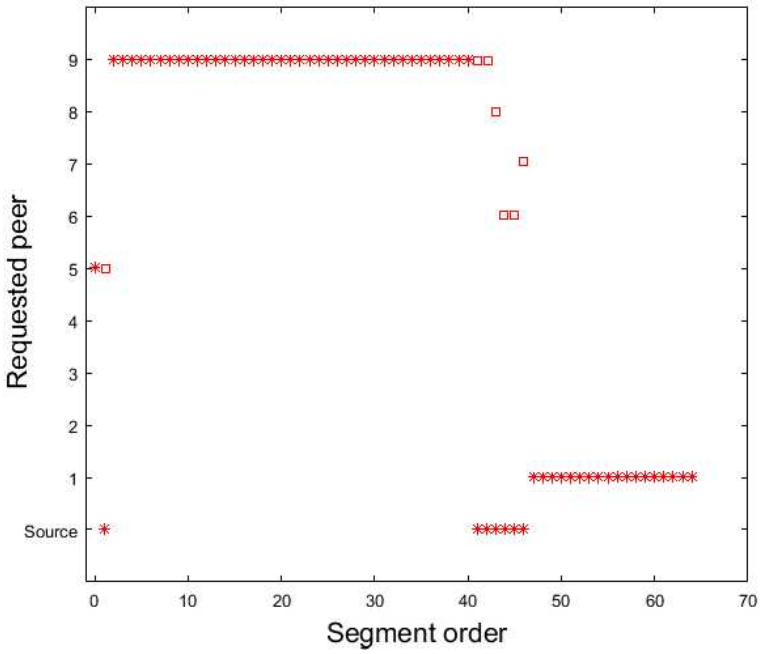

그림 8. 추가 실험의 세그먼트 순서에 따른 전달 피어
Fig. 8. Delivering peers according to segment order of additional experiment

연속으로 실패하여 그 우선순위가 5에서 3으로, 다시 3에서 1로 감소되어 추후 선택에서 배제된다. 이후 우선순위가 1로 낮아진 피어 5와 9를 제외한 이웃들은 모두 초기 우선순위 값인 3을 가지므로, 스트리밍 초기와 유사한 과정을 거쳐 이들 중 정속 이웃을 찾게 된다. 그림 7의 결과에서는 저속 이웃인 이웃 8, 6(DASH 미디어 플레이어가 2개 세그먼트를 한꺼번에 요청하여 모두 실패한 경우), 7에 대한 세그먼트 요청이 실패한 후, 정속 이웃인 이웃 1을 찾은 이후에는 세그먼트가 원활히 전달됨을 확인할 수 있다. 이와 같이 제안한 알고리듬은 이웃들의 네트워크 상황의 변화가 있을 때에도 효율적으로 새로운 정속 피어를 잘 찾아내며 CDN 비용을 절감함을 알 수 있다.

그림 7과 8의 결과에서 보듯이, 제안한 알고리듬은 세그먼트 전달에 성공한 이웃에게 그 이후에도 계속 세그먼트를 요청하는 형태로 동작한다. 이러한 현상은 해당 이웃의 상태가 나빠지기 전까지 지속된다. 이는 제안한 알고리듬이 CDN 비용 절감비를 극대화하는 형태로 고안되어 있기 때문이다. 만약 어떤 한 이웃의 상태가 좋다고 해서 여러 피어로부터 그 이웃에게 세그먼트 요청이 집중된다면, 해당 이웃은 결국 과부하 상태에 이를 것이고 자연스럽게 우선순위가 낮아져서 피어들은 다른 이웃에게 요청을 보내게 될 것이다. 따라서 제안한 알고리듬은 가장 빠른 이웃에게 과부화가 걸릴 수 있는 문제 또한 자연스럽게 해결할 수 있다.

Hive 시스템이 실생활에서 사용될 때는 사용자의 동의를 얻는 과정이 전제되어야 한다. 사용자가 동의한다 하더라도, 만약 사용자의 망 사용료가 문제가 되는 상황이라면, 단기적으로라도 하나의 이웃에게 세그먼트 요청이 집중되는 것에 대해 사용자의 불만이 제기될 수도 있다. 이를 위해서는 본 논문에서 제안하는 알고리듬만큼 CDN 비용 절감비를 달성하지는 못하더라도 단기적으로라도 여러 이웃에게 세그먼트 요청을 분산시키는 새로운 알고리듬이 필요할 수도 있다. 결국 이러한 알고리듬은 CDN 비용 절감비와 세그먼트 요청 분산 정도 사이의 적절한 맞바꾸기(trade-off)를 목표로 하는 알고리듬이 될 것이다.

마지막으로, 일반적인 적응형 미디어 스트리밍 시스템과 P2P를 함께 사용하는 Hive 시스템에 대한 QoS 측면을 비교 검토하고자 한다. Hive 시스템에서 DASH 미디어 플레이어는 자신의 세그먼트 요청이 CDN 서버 또는 다른 피어 중 어느 곳으로 향하게 될지 알지 못하는 상태로 동작하며, 자신의 미디어 버퍼의 점유도와 세그먼트 전달 속도를 기반으로 다음 세그먼트들을 요청한다. Hive 시스템에서는 DASH 미디어 플레이어로부터 어떤 세그먼트를 요청을 받으면, 먼저 다른 피어에게 한 차례 요청하여 만약 이것이 타임아웃에 의해 실패할 경우 CDN 서버에게 요청한다. 따라서 연속적으로 여러 세그먼트에 걸쳐 타임아웃이 발생하면, 타임아웃 설정 시간이 누적될 수도 있다. 이 경우, DASH 미디어 플레이어의 미디어 버퍼가 고갈되어 화면 멈춤 현상이 발생하거나 DASH 미디어 플레이어가 망의 상태가 좋지 않을 것으로 자체적으로 판단하여 저전송률로 부호화된 세그먼트를 요청하게 될 수도 있다. 이러한 경우에는 Hive 시스템의 QoS가 저하될 수 있다. 그러므로 Hive 시스템을 설계할 때에는 상기 언급한 타임아웃 설정 시간이 누적되는 현상이 발생하지 않는 조건 하에서 동작하도록 할 필요가 있다. 이 조건에는 CDN 서버에게 세그먼트를 요청했을 때의 세그먼트 전달 속도와 이 때 한 번의 요청에 의해 받을 수 있는 세그먼트의 재생 시간 관점에서의 크기가 개입된다. 즉, 피어에 대한 요청 시작 시점부터 타임아웃이 발생하여 CDN 서버로부터 세그먼트를 획득 완료하는 시점까지의 경과 시간이 해당 세그먼트의 재생시간을 초과하지 않는 한, 타임아웃 설정 시간이 누적되지 않는다. 본 논문의 실험에서는 Hive 시스템에 의한 QoS 저하 현상은 관측되지 않았다. 물론 이러한 내용은 최악의 경우에 대한 분석이며, Hive 시스템에서 이웃으로부터 세그먼트 전달이 매우 원활한 경우에는 CDN 서버 측으로 요청하였을 때 보다 세그먼트를 빠르게 획득할 가능성이 높으므로, 일반적인 적응형 미디어 스트리밍 시스템에 비해 더 좋은 QoS를 제공한다. 단, 두 시스템 간의 구체적인 QoS 비교는 더 많은 연구를 필요로 한다고 사료된다.

VI. 결론 및 향후 과제

본 논문에서는 WebRTC 기반의 P2P 통신 병용 DASH

시스템에서 CDN 서버 부하 절감 효과를 크게 개선할 수 있는 새로운 알고리듬을 제안하고 이를 구현하였으며 모바일 환경의 VoD(Video-on-Demand) 시나리오 하에서 네트워크 에뮬레이터를 활용한 실험을 통해 그 성능을 검증하였다. 무작위로 피어를 선택하여 세그먼트를 요청하는 기존 시스템과 달리, 제안한 알고리듬에서는 피어의 응답 지연시간과 세그먼트 전달 속도에 기반한 통신 이력을 활용하여 세그먼트 전달에 성공할 가능성이 가장 높은 피어에게 세그먼트를 요청하게 하였다. 제안하는 알고리듬은 CDN 비용을 최대한 절감하는 것을 목표로 고안되었다. 실험 결과, 기존 시스템에 비해 훨씬 나은 CDN 비용 절감 효과를 얻을 수 있었다. 또한 피어들의 상태가 동적으로 변하는 경우에도 제안하는 알고리듬이 잘 동작함을 확인하였다.

이러한 결과를 바탕으로 향후 다양한 관련 연구가 더 필요하다. 우선 실시간(live) 스트리밍을 포함한 보다 더 다양한 시나리오에 대해 실험할 필요가 있다. 또한 모바일 네트워크 환경뿐만 아니라 다른 네트워크 환경까지 고려한 일반적인 형태로 알고리듬을 개선할 필요가 있다. 또 피어간 통신료가 문제가 되는 경우를 고려하여, CDN 비용 절감 효과를 다소 희생하더라도 단기적으로 세그먼트 요청이 소수의 피어에게 집중되는 것을 피하고 여러 피어에게 분산되도록 하는 새로운 알고리듬에 대한 연구도 필요하다. 보통의 적응형 미디어 스트리밍 시스템과 P2P를 병행하는 스트리밍 시스템 간의 QoS 비교도 더 구체적으로 연구할 필요가 있다. 본 논문의 실험에서는 피어의 상태를 저속과 정속의 두 가지로 단순화하였으나, 실제 네트워크 환경에서는 피어들의 상태가 더 다양할 수 있으며 또 동적인 상태 변화도 여러 가지 형태로 나타날 수 있다는 점을 고려하여 이러한 네트워크 환경을 에뮬레이션한 상태에서 실험실 내 실험을 통해 제안한 알고리듬을 더 검증하고 필요 시 개선하여야 한다. 마지막으로, 다수 사용자가 참여하는 실제 네트워크 환경에서의 실험을 통해 제안하는 알고리듬의 성능을 검증할 필요가 있다.

참 고 문 헌 (References)

[1] H. Tsukayama, Youtube: The future of entertainment is on the web, https://www.washingtonpost.com/business/technology/youtube-the-future-of-entertainment-is-on-the-web/2012/01/12/gIQADpdBuP_story.html?utm_term=.82f602999ad5 (accessed Dec 26, 2018).

[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] R. Roverso, S. El-Ansary, and S. Haridi, “Smoothcache: Http-live streaming goes peer-to-peer,” in IFIP NETWORKING 2012, pp. 29-43, May 2012.

[5] M. Varma, H. K. Yarnagula, and V. Tamarapalli, "WebRTC-based peer assisted framework for HTTP live streaming," Proceedings of the 9th International Conference on Communication Systems and Networks (COMSNETS), pp. 415-416, Jan., 2017.

[6] R. Pantos and W. May, “HTTP Live Streaming,” IETF RFC 8216, Aug. 2017.

[7] J. Liu, L. Shang, H. Jin, et al., “Bluetooth P2P Architecture for Transporting Streaming Media on the Internet,” 2009 IEEE IMSAA, Dec. 2009.

[8] 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.

[9] M. Handley, V. Jacobson, and C. Perkins, “SDP: Session Description Protocol,” IETF RFC 4566, July 2006.

[10] MSIT (Ministry of Science and Information Technology) Press Release, “Quality assessment results of 2017 telecommunication services,” https://www.msit.go.kr/web/msipContents/contentsView.do?cateId=mssw311&artId=1371275, Dec. 26, 2017.

[11] Dash media server, http://dash.edgesuite.net/envivio/.

[12] Netbalancer, https://netbalancer.com/.

[13] Https://github.com/Peerialism/hive.js.

저 자 소 개

서 주 호

- 2017년 8월 : 서울시립대학교 물리학과 학사
- 2017년 9월 ~ 현재 : 서울시립대학교 전자전기컴퓨터공학과 석사과정
- ORCID : https://orcid.org/0000-0002-7140-0816
- 주관심분야 : 영상통신 및 시스템, P2P 통신, 적응형 스트리밍

최 성 현

- 2017년 2월 : 인천대학교 전자공학과 학사
- 2019년 2월 : 서울시립대학교 전자전기컴퓨터공학과 석사
- ORCID : https://orcid.org/0000-0002-3207-0463
- 주관심분야 : 영상통신 및 시스템, 적응형 스트리밍, 영상 압축

김 상 진

- 1989년 2월 : 연세대학교 전기공학과 (공학사)
- 1991년 2월 : 연세대학교 대학원 전기공학과 (공학석사)
- 2018년 2월 : 서울과학기술대학교 정보통신미디어공학 전공(Ph.D.)
- 2016년 ~ 2017년 : SBS 뉴미디어개발팀장
- 2018년 ~ 현쟤 : SBS 미디어기술연구소장
- 2016년 ~ 현재 : TTA TC8(방송기술위원회) 의장
- 2015년 ~ 현재 : TTA PG802 의장
- ORCID : https://orcid.org/0000-0003-2142-4972
- 주관심분야 : 디지털 방송, 미디어 서비스

전 재 영

- 2014년 2월 : 한국과학기술원(KAIST) 전기및전자공학부 학사
- 2014년 1월 ~ 현재 : (주) SBS(Seoul Broadcasting System) 미디어기술연구소
- ORCID : https://orcid.org/0000-0002-5501-0703
- 주관심분야 : 미디어 스트리밍 서비스, 방송 그래픽

김 용 한

- 1982년 2월 : 서울대학교 제어계측공학과 (공학사)
- 1984년 2월 : 서울대학교 대학원 제어계측공학과 (공학석사)
- 1990년 12월 : 미국 렌슬리어공대(Rensselaer Polytechnic Institute, RPI) 전기컴퓨터시스템공학과 (Ph.D.)
- 1984년 3월 ~ 1996년 3월 : 한국전자통신연구원 책임연구원(최종)
- 1991년 10월 ~ 1992년 9월 : 일본 NTT 휴먼인터페이스연구소 객원연구원
- 1996년 3월 ~ 현재 : 서울시립대학교 전자전기컴퓨터공학부 교수
- 2017년 1월 ~ 2017년 12월 : 한국방송·미디어공학회 회장
- 2014년 1월 ~ 현재 : 미래방송미디어표준포럼(구 차세대방송표준포럼) 의장
- 2000년 1월 ~ 현재 : MPEG뉴미디어포럼(구 MPEG포럼) 운영위원/자문위원
- ORCID : https://orcid.org/0000-0001-9470-6060
- 주관심분야 : 영상통신, 디지털 방송, 멀티미디어 부호화 및 전송