논문 21-46-12-30 The Journal of Korean Institute of Communications and Information Sciences '21-12 Vol.46 No.12
https://doi.org/10.7840/kics.2021.46.12.2351

IETF의 대용량 파일 전송 표준 FCAST 구현: 헤더 안정성과 패킷손실 복구 능력의 개선

사 봉 준•, 서 덕 영°, 원 대 한*, 정 철 현*, 차 은 영*

Design of FCAST System which is Standard of Large File Transfer of IETF : Enhancement in Header Stability and Packet Loss Recovery Ability

Bong Jun Sah•, Doug Young Suh°, Dae Han Weon*, Cheol Hyeon Jeong*, Eun Young Cha*

요 약

본 논문은 IETF의 비실시간 데이터 멀티캐스트시스템 표준인 FCAST 전송 방식[1]의 구현에 관한 내용이다. 특히, 반복 전송을 통한 패킷 손실 제어에서의 문제와, FCAST 헤더 생성 시에 생겨나는 문제를 해결한다. 먼저, 반복 전송을 통한 패킷손실 제어에서는 패킷 손실률(PLR, Packet Loss Rate)이 높은 경우, 과도하게 많은 반복 전송 횟수를 필요로 하는 문제가 있었다. 이러한 문제는 패킷 손실이 각 패킷에 대하여 독립적이던 기존의 시스템에서 발생하는 것으로, FEC(Forward Error Correction)를 적용하여 패킷 손실이 들어온 패킷의 수에 연관되도록 하여 해결을 하였다. 다음으로, 헤더 생성에서의 문제는 지원하지 않는 확장자 데이터를 다른 데이터와 함께 반복 전송하는 경우, 다음 반복 전송 횟수에서 헤더의 크기가 몇 바이트 증가하는 문제를 말한다. 본 논문에서는 헤더 생성 부분에서 지원하지 않는 확장자에 대한 경우를 추가함으로서 안정성을 향상시켰다. 구현된 시스템을 국제 클라우드망에서 테스트하여, 안정적으로 동작하는 것을 보인다.

키워드 : 대용량 데이터 전송 프로토콜(FCAST), 응용계층 순방향 오류 제어(AL-FEC), 멀티캐스트, 분수 코드, 클라우드 컴퓨팅

Key Words : FCAST, AL-FEC, multicast, Fountain code, cloud computing

ABSTRACT

Our research deals with design of FCAST system[1] which is non-real time data multicast standard of IETF. Especially, our system improves two problems in packet loss control by retransmission and FCAST header generation. First, packet loss control by retransmission needs a lot of cycles in high PLR(Packet Loss Rate) environment, because packet loss is independent in each packets. In our research, we applied FEC in FCAST system to improve this problem by correlating packet loss with received packet num. Next, there is a few bytes data increase in FCAST header if file with unsupported extention is transmitted repeatedly with other file. We improve this problem by adding case of unsupported file extention. We show stable performance of our system by testing in international cloud networking.

※ 본 연구는 (주)한시간컴의 ATSC 3.0 기반 (MMT,DASH)VM software 개발의 연구결과로 수행되었습니다.
• First Author : KyungHee University Department of Electronic Engineering, tkqhdwns@naver.com, 학생회원
° Corresponding Author : KyungHee University Department of Electronic Engineering, suh@khu.ac.kr, 종신회원
* KyungHee University Department of Electronic Engineering, daehanweon@naver.com; 2017103030@khu.ac.kr; eunyyy3@naver.com, 학생회원
논문번호 : 202107-150-D-RN, Received July 1, 2021; Revised August 26, 2021; Accepted September 8, 2021

Ⅰ. 서 론

영상 데이터 이용량이 지속적으로 증가하고 있다. 또한 신종 코로나 바이러스(이하 코로나19)의 확산은 이러한 경향을 더욱 가속화 시켰다. Akamai[13]에 따르면, 2020년 3월 인터넷 데이터 이용량은 전월 대비 30% 증가하였다. 이는 코로나19 이전 증가량의 10배에 해당하는 수치였다. 이러한 추세의 가장 큰 원인은 영상 데이터 이용량의 증가이다. 같은 기간 내 영상 데이터 이용량은 200% 이상, 그 중에서도 영상 통화 시간은 1000% 이상 급증하였다.

이러한 상황 속에서 대용량 데이터 전송 방법의 하나인 FCAST[1]가 주목받고 있다. FCAST는 데이터 전송에 필요한 데이터 형식 정보와 지침의 집합이다. 이러한 FCAST에는 세 가지 특징이 있는데, 비실시간으로 전송이 이루어진다는 것, 특정 IP를 공유하는 사용자들에게 데이터를 동시에 전송하는 IP 멀티캐스트를 지원하는 것과 다양한 데이터의 확장자를 지원하는 것이다.

먼저, FCAST는 비실시간으로 전송이 이루어진다. 송신자는 계속하여서 데이터를 송신하고, 수신자는 지정된 시간에 송신자 서버 IP에 접속하여서 데이터를 수신하는 방식이다. 이를 통해, FCAST는 인터넷 이용량이 상대적으로 적은 시간대에 예약전송이 가능하다. 즉, FCAST는 증가한 인터넷 데이터 이용량으로 네트워크가 과부하되는 현상을 피하고, 안정적인 시간대에 전송이 가능하다는 장점이 있다.

다음으로, FCAST는 IP 멀티캐스트를 지원하기 위해 서로 데이터를 주고받는 방식이 아닌, 한쪽에서 일방적으로 전송하는 UDP(User Datagram Protocol)를 사용한다. 다만 UDP의 경우, 각각의 사용자가 어떤 부분에서 데이터 손실이 일어났는지 확인이 불가능하기 때문에, 반복 전송을 통해서 데이터 손실에 대처한다. 문제는 대용량 데이터의 반복 전송이 전송 시간을 증가시키고 전송의 효율성을 낮춘다는 점이다. 결과적으로 FCAST 전송 방식의 효율성을 보완하기 위해서는 반복 전송 횟수를 줄이는 방안이 필요한 상황이다.

마지막으로 FCAST는 다양한 데이터의 확장자를 지원한다. 그러나 영상 매체가 다양해지고 압축 기술의 종류가 많아지면서, 이러한 확장자 또한 종류가 늘어나고, 다양해지고 있다. 결과적으로 확장자에 국한되지 않는 전송 기법에 대한 연구 및 개발이 필요한 상황이다.

FCAST에서 반복 전송 횟수를 감소시키기 위해서는 오류에 대한 저항성을 부여하는 오류 제어 시스템이 필요하다. 이에 본 논문은 FCAST 시스템에 적용 가능한 패킷손실 제어 시스템인 FEC(Forward Error Correction)[4]를 결합하여 구현하였다. 이 과정에서 복구된 패킷의 경우 패킷의 길이를 알 수 없던 문제가 발생하였고, 이를 해결하기 위해 본 논문에서는 기존 표준 형식의 확장을 통하여서 해결하는 방법을 새롭게 고안하였다.

또한 FCAST에서 확장자에 관계없이 데이터 전송을 하기 위해서 기존 표준에서의 오류를 정정하였다. 기존의 표준의 경우, 파일의 확장자 정보를 가지고 있는 FCAST 헤더의 생성에서 지원하지 않는 확장자의 경우 오류가 발생하였다. 이에 지원하지 않는 확장자에 대하여 표준을 정정함으로서, 모든 확장자에 대하여 전송이 가능하도록 변경하였다. 즉 본 논문은 FCAST를 통해 데이터가 파일 확장자에 관계없이 보다 적은 반복 전송 횟수로 전송되어 질 수 있도록 하는 방법을 제안하였다.

2장에서는 FCAST와 ALC에서의 데이터 형식과 데이터 처리 및 전송에 필요한 기능을 설명한다. 또한 FEC 시스템에 대하여 설명을 하고, 기존 FCAST의 초기화관련 오류에 대하여 설명한다. 3장에서는 시스템 구조를 모델링을 통해 상세히 설명한다. 4장에서는 FCAST 헤더 생성 안정성 향상 결과를 설명한다. 또한 반복 전송 횟수 비교를 통하여 패킷 손실률(PLR, Packet Loss Rate)과 데이터 크기에 대한 저항성 결과를 설명하고, 국제 클라우드망 실험 결과를 설명한다. 그러나 국제 클라우드망 실험 결과, 패킷 손실의 시간적 분포가 집중손실(burstloss)[11]의 형태를 취하고 있기에, FEC 효과가 적게 나타났다. 이를 해결하기 위하여 FEC가 실제 환경에 적용되기 위하여서 필요한 인터리빙(interleaving)[8] 방식에 대한 사고실험을 설명한다. 5장에서는 결론과 향후의 전망에 대한 내용을 끝으로 본 논문을 마친다.

Ⅱ. 배 경

2.1 FCAST

FCAST는 대용량 파일 전송을 위한 데이터 형식 정보와 지침의 집합이다. 그림 1은 FCAST 데이터 형식[5]을 도식화한 것이다. FCAST에서는 하나의 파일을 객체(object)라고 부른다. 객체에는 미디어 데이터 뿐 만 아니라 디지털 지도, 미디어, 오디오 데이터 등 다양한 형태의 데이터가 포함될 수 있다. 또한 객체의 데이터와 FCAST 헤더를 합친 것을 결합된 객체(compound object), 이 결합된 객체들의 집합을 순차

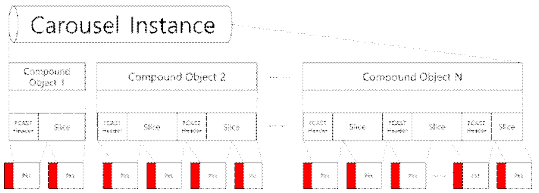


그림 1. FCAST 구조
Fig. 1. FCAST structure

적 전송의 의미에서 회전목마 집합체(carousel instance)라고 부른다. 한편 FCAST에서는 객체를 일정한 크기, 다시 말해 조각(slice)으로 나눌 수 있다. 각각의 조각 마다 FCAST 헤더가 존재하기 때문에, 이 조각들은 일종의 작은 객체라고 볼 수 있다.

2.2 The Relation between FCAST and ALC

FCAST 형식을 사용하여 데이터를 전송하는 방식에는 ALC (Asynchronous Layered Coding)[2]와 NORM (NACK-Oriented Reliable Multicast)[3] 방식이 있다. 이중에서도 본 연구는 콘텐츠를 안정적으로 전달하면서도 확장성이 뛰어난 ALC 전송 방식에 주목하였다. 그림 2는 FCAST와 ALC의 관계[1]를 도식화한 것이다. ALC 방식은 수신을 비동기적으로 시작할 수 있고, 대규모 확장이 가능하며 FEC 수준에서 우선순위를 통제할 수 있다는 장점이 있다. 이러한 ALC의 최상부에서 FCAST 방식은 수신자로부터 어떠한 피드백 없이 작동한다.

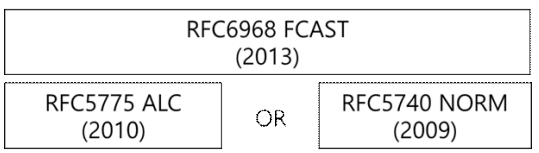


그림 2. FCAST와 ALC 관계
Fig. 2. Relation between FCAST and ALC

2.3 ALC(Asynchronous Layered Coding)

ALC는 IP 멀티캐스트를 사용하여 대규모 확장성을 제공하도록 설계된 기본 네트워크 서비스이다. 그림 3은 이러한 ALC의 헤더 구조[2]를 보여준다. ALC 헤더에는 대표적으로 패킷 일련번호(PSN, Packet Serial Number)와 객체 식별자(TOI, Transport Object Identifier) 등의 정보가 있다. ALC 헤더에는 패킷의 처음과 끝을 알 수 있는 방법이 없기 때문에, 패킷의 순서를 알려주는 정수 값인 패킷 일련번호를 사용한다. 또한 하나 이상의 객체를 전송 시, 각 객체를 구분하기 위하여 객체 식별자를 사용한다. 즉 ALC는

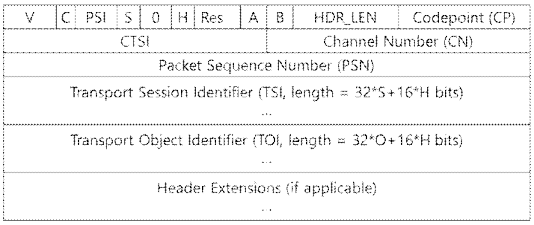


그림 3. ALC 패킷 헤더 구조
Fig. 3. ALC packet header structure

FCAST에서 만들어진 결합된 객체를 일정한 크기의 패킷으로 나누어, 각 패킷에 ALC 헤더를 붙여 전송하는 방식이다.

2.4 FEC(Forward Error Correction)

FEC란 순방향 오류 제어 방식을 의미한다. 송신자가 전송할 데이터나 부가적 정보(복구 패킷)를 첨가하여 전송하고, 수신 시 패킷 손실이 발견되면 이 부가적 정보로 패킷을 복구하는 방식이다. 이 때 부가적 정보는 분수 코드(fountain code) 방식[4]을 사용하여 생성된다. 이 방식은 가중치에 대한 정보를 담고 있는 행렬(Q matrix)과 데이터간의 갈로아(Galois field) 연산을 기반으로 하는 행렬곱 연산을 통해 생성된다. 이후 패킷 복구는 부가적 정보와 데이터의 역행렬 연산을 통해 진행된다.

Ⅲ. 헤더 생성 안정성 향상과 FEC를 적용한 FCAST 시스템의 설계 및 구현

3.1 서버, 클라이언트 설계

그림 4는 본 시스템의 서버와 클라이언트 구조를 나타낸 것이다. 먼저 서버는 모든 동작을 관장하는 세션(session), 클라이언트의 연결 및 상태를 관리하는 call-setup 매니저, 객체를 읽고 FCAST 헤더를 생성하는 FCAST Oloader 모듈, ALC 패킷화 과정을 수행하는 ALC 모듈, AL-FEC를 위한 FEC 인코더, 클라이언트의 피드백 메시지를 수신 및 처리하고 서비스 품질(QoS, Quality of Service)[6] 조절 등의 적절한 기능을 수행하는 QoS 매니저, 마지막으로 패킷을 전송하는 송신(sender) 모듈로 구성하였다.

클라이언트는 모든 동작을 관장하는 세션 모듈, 패킷을 수신하는 수신(receiver) 모듈, AL-FEC를 위한 FEC 디코더, ALC 헤더를 제거하는 과정을 수행하는 ALC 역 패킷화 모듈, FCAST 헤더를 제거하는 FCAST 헤더 제거 모듈과 마지막으로 서버에게 피드백 메시지를 전달 할 수 있는 QoS 매니저로 구성하였다.

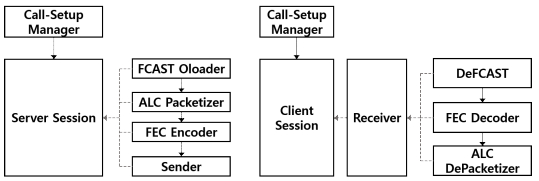


그림 4. 서버-클라이언트 구조도
Fig. 4. Structure of server-client

Call-setup 모듈은 서버와 클라이언트 간의 접속을 관리하도록 하였다. call-setup 모듈은 멀티캐스트와 유니캐스트 IP 주소 그리고 포트를 할당하여 UDP 소켓을 생성하고, 이후 전송이 종료될 경우에는 UDP 소켓을 제거하는 역할을 한다.

또한 FCAST Oloader는 저장장치에 있는 파일들을 객체 목록에 등록하고 읽어오도록 설계되었다. Oloader 모듈은 객체를 저장하기 위한 메모리를 동적 할당 하고, 객체 식별자의 순서에 따라 객체를 하나씩 읽는다. 이후 데이터를 조각 크기만큼 분할하여 할당한 후, 회전목마 집합체에 대한 정보를 담고 있는 정보 데이터(metadata)를 작성한다. 생성된 정보 데이터는 FCAST 헤더에 포함되며, 조각 단위의 데이터 앞에 FCAST 헤더를 추가한 후 다시 세션 모듈로 데이터를 전달한다.

ALC 모듈은 조각을 ALC 패킷으로 분할하고 ALC 패킷 헤더를 추가하는 역할을 하도록 하였다. ALC 모듈은 우선 세션 모듈의 FCAST 헤더가 추가된 데이터를 불러온다. 해당 조각을 정해진 패킷 크기만큼 읽으면서 ALC 패킷 헤더를 추가하여, 할당된 패킷 메모리 공간에 복사한다. 하나의 조각이 패킷화가 끝날 때 까지 위 과정을 반복한다.

FEC 인코더는 ALC 패킷을 인코딩하여 복구 패킷을 생성하도록 설계되었다. FEC 인코더 모듈은 패킷을 받아와 하나의 단위 구조체에 해당하는 묶음공간(bufflet)에 채워 넣는다. 전부 채워질 경우에는 인코딩을 진행한다. 인코딩의 결과로 생성된 복구 패킷과 데이터 패킷에 단위 구조체 번호와 인코딩 형태 식별자를 포함하는 꼬리표를 붙인다. 한 조각에 속하는 모든 패킷들의 인코딩이 될 때 까지 위 과정을 반복한다. 인코딩이 끝나면 복구 패킷과 데이터 패킷을 세션 모듈에 전달한다.

전송 모듈은 인코딩 과정을 통해 생성된 복구 패킷과 데이터 패킷을 전송하는 역할을 하도록 하였다. 먼저, 송신 모듈은 call-setup 모듈로부터 하나의 UDP 소켓을 받아온다. 전송률(bitrate)을 설정하고, 패킷간 전송 시간을 구한다. 세션 모듈로부터 하나 이상의 패킷을 저장한 메모리, 패킷의 길이 정보를 저장한 배열 그리고 패킷의 개수를 받아온다. 패킷을 전송하고 패킷간 전송 시간 동안 멈춘다.

수신 모듈은 패킷을 수신하고, 헤더를 제거하여 데이터를 저장하는 총괄적인 수신을 담당하도록 설계되었다. 먼저, 수신 모듈은 세션 모듈로부터 소켓 정보와 패킷 메모리 주소를 받아온다. 수신한 객체를 저장할 구조체 메모리를 생성하고, FCAST 헤더 제거 모듈을 통해 객체의 정보를 전달 받는다. 구조체는 이 정보에서 해당하는 패킷의 시작과 마지막 번호를 저장한다. 이후 FEC 디코더 모듈로 수신 패킷을 전송하여 묶음공간의 형태로 메모리에 저장한다. 묶음공간이 전부 채워질 경우에는 디코딩을 진행하여 손실된 패킷을 복구한다. 누적된 패킷 수(현 실험 2500)가 충분할 경우, ALC 역 패킷화 모듈에 패킷과 구조체 메모리 주소를 전달하고, 패킷을 객체 데이터에 쓰고 저장한다. 이후 패킷 들은 위 과정을 반복하여 파일의 끝에 추가된다.

QoS 매니저는 서비스 품질 조절 등의 기능을 하도록 설계되었다. QoS 매니저 모듈은 전송될 객체 목록을 전송받고, 목록에 추가 또는 삭제한다. 이를 위해 객체의 번호를 저장하는 객체 식별자의 배열을 정의하고 서버로 전송하도록 하였다. 또한 수신과정에서 패킷 수신 과정과 AL-FEC 디코딩 과정에서 생성되는 수신 상태 및 기타 사용자의 상태 등을 서버로 전송하도록 하였다.

3.2 FCAST 헤더 오류 정정

본 시스템에서는 기존 FCAST 헤더에 객체의 확장자에 관한 정보를 저장하는 과정에서 발생하는 초기화 오류를 정정하였다. FCAST의 일반적인 헤더 구조[1]는 그림 5와 같다. 초기화 오류는 FCAST가 지원하는 확장자 (표 1) 이외의 확장자를 가진 객체를 동시에 전송하는 경우에 일어난다.

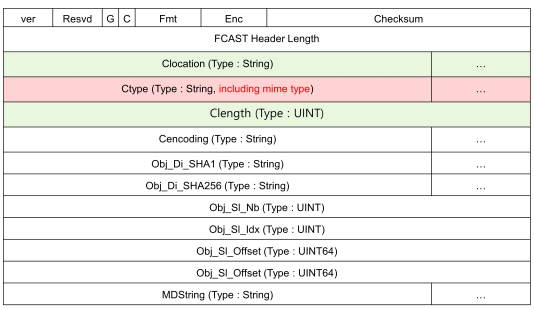


그림 5. FCAST 헤더 구조
Fig. 5. FCAST header structure

표 1. 지원 확장자 목록
Table 1. Supported filename extension list

Supported filename extension	category
html	html/text
jpg	image/jpeg
mpg	video/mpeg
png	image/png
txt	text/plain
mp3	audio/mpeg
mp4	video/mp4

FCAST는 객체의 일부인 조각마다 헤더를 생성한다. 이 과정에서 확장자 정보(Ctype)는 널(null) 값으로 초기화 되는데, 이 때 데이터의 확장자가 지원 확장자 목록에 존재할 경우에는 확장자 정보가 FCAST 헤더에 기입된다. 그러나 지원되지 않는 확장자의 경우에는 널 값이 그대로 기입되고, 그로 인해 문제가 발생한다.

FCAST 헤더 생성은 별도의 초기화 과정 없이, 덮어쓰기 형태로 진행이 된다. 이때 널 값을 기입할 경우, 기존에 해당 위치에 있는 데이터를 그대로 유지하게 된다. 따라서 이전 데이터 전송에서 FCAST 헤더에 확장자 정보가 존재하고, 현재 데이터 전송에서 지원되지 않는 확장자로 인해 널 값이 FCAST 헤더에 기입된다면, 이전 전송에서의 확장자 정보가 초기화 되지 않은 채 그대로 유지되는 것이다.

이에 본 시스템에서는 지원되지 않는 확장자로 인해 발생하는 FCAST 헤더의 초기화 문제 해결을 위해 지원되지 않는 확장자의 경우, 널 값이 아닌 빈 문자열(string)로 처리하도록 시스템을 설계하였다 (그림 6).

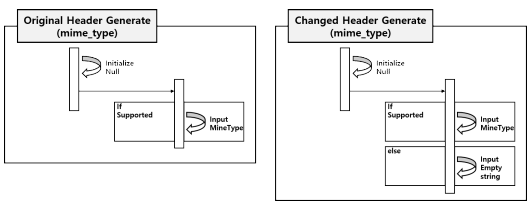


그림 6. FCAST 확장자 정보 초기화 시스템 (좌측 그림. 기존 시스템, 우측 그림. 변경 시스템 구조)
Fig. 6. FCAST Ctype initialize system (Left fig. original system, Right fig. changed system)

3.3 FEC 적용

본 시스템에서는 분수코드 방식을 사용한 AL-FEC (Application Layer- Forward Error Correction)[4] 기능을 사용하였다. AL-FEC의 구현을 위해 분할되어 생성된 데이터 패킷을 단위 구조체(SB, Source Block) 단위로 전송하였다. 또한 수신자가 각각의 단위 구조체를 구분할 수 있도록 단위 구조체 번호(SBN, Source Block Number)와 단위 구조체의 행에 해당하는 인코딩 형태 식별자(ESI, Encoding Symbol Identifier)로 구성된 FEC 꼬리표를 추가하였다.

그림 7은 본 시스템이 설계한 단위 구조체의 구조[9]를 나타낸 것이다. 이 때 K는 데이터 패킷의 개수를 나타내고, N은 데이터 패킷에 복구 패킷의 수를 더한 개수이다. 본 시스템은 단위 구조체의 행이 전부 채워지지 않을 경우, 제로 패딩을 사용하여 채우고 FEC 꼬리표를 추가하였다.

한편 복구된 패킷에서 제로 패딩된 영역과 데이터의 구분이 불가능한 문제가 있었다. 기존 ALC 표준은 패킷의 데이터와 길이 정보를 가지고 있는 ALC 구조체를 전송한다. 그러나 FEC를 통하여 패킷이 복구될 경우, 구조체가 아닌 데이터 패킷만 복구되어, 길이 정보는 유실되게 된다. 본 시스템은 복구된 패킷의 길이를 알 수 없던 기존 표준의 문제점을 해결하기 위해 ALC 패킷 헤더의 확장 부분을 활용하였다. 본 시스템은 ALC 헤더의 확장 부분에 패킷의 길이를 추가하여 FEC를 통해 복구될 경우, 패킷의 길이가 함께 복구되도록 하였다. 그림 8은 본 시스템에서 추가된 확장 부분을 포함한 ALC 헤더의 구조를 나타낸다.

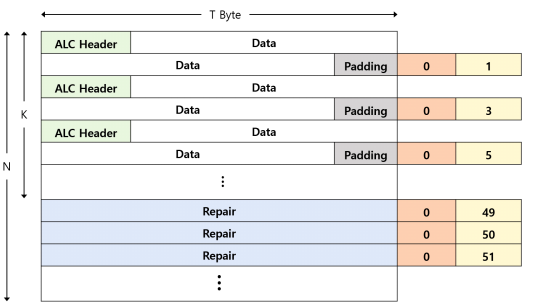


그림 7. FEC 시스템 단위 구조체의 구조
Fig. 7. SB structure of FEC system

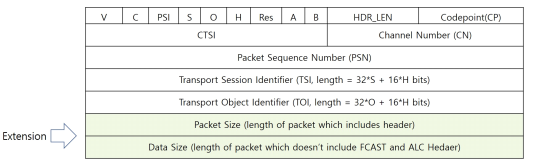


그림 8. ALC 헤더 확장 포함 구조
Fig. 8. ALC header structure with extension

Ⅳ. 시스템의 성능 실험

4.1 헤더 생성 오류 정정

기존의 FCAST 헤더 생성시에는 널 값의 기입으로 인한, 이전의 확장자 정보가 초기화 되지 않는 문제가 있었다. 이러한 문제는 반복 전송에서 패킷이 손실될 경우 데이터의 손상을 야기한다.

그림 9는 지원하지 않는 확장자를 가진 객체와 다른 객체를 동시 전송하는 경우에서 지원하지 않는 확장자의 데이터 구조를 보여준다. 반복 전송 횟수(cycle) 1에서는 확장자 정보에 널 값이 입력되기 때문에, 길이는 0 바이트이다. 그러나 반복 전송 횟수 1에서 다른 객체를 전송하며 생성된 정보는 반복 전송 횟수 2에 그대로 남아있게 된다. 따라서 2에서는 1에 비해 FCAST 헤더의 길이가 확장자 정보의 길이만큼 늘어나게 되고, 실제 데이터 패킷은 늘어난 길이만큼 밀리는 현상이 발생한다. 이로 인해 그림 10과 같이 1에서 수신되지 않았던 패킷을 기준으로 이전 패킷은 확장자 정보의 길이만큼 중복이 되고, 현재 패킷의 끝부분은 정보의 길이만큼 손실되게 된다.

이러한 데이터 손실과 밀림 현상은 이전 전송에서의 확장자 정보를 제거하고, 지원하지 않는 경우 빈 문장을 덮어써줌으로서 일정한 FCAST 헤더의 길이를 가지게 하여 더 이상 발생하지 않게 되었다. 지원하지 않는 확장자(avi, ppt) 파일을 기존의 표준 시스템을 통해 동시에 전송할 경우 6 바이트의 데이터 밀림 현상이 발생하였지만, 본 논문의 시스템에서는 해당 확장자에 대해서도 데이터 손실 없이 전송이 가능함을 확인 하였다.

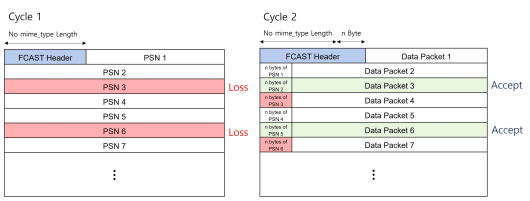


그림 9. 반복 전송 횟수 1, 반복 전송 횟수 2에서의 지원하지 않는 확장자를 가진 객체의 데이터 구조.
Fig. 9. Data structure of object, which is not supported, in cycle 1 and cycle 2.

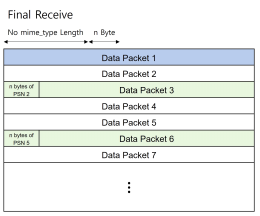


그림 10. 최종적으로 수신된 객체의 데이터 구조.
Fig. 10. Data structure of object, which is finally received.

4.2 평균 패킷 손실률 시스템에서의 성능 평가

그림 11은 본 논문의 시스템인 FEC가 적용된 시스템에 대한 평균 패킷 손실률(avg PLR, average Packet Loss Rate)에 따른 무손실 전송에 필요한 반복 전송 횟수를 그래프로 나타낸 것이다. 그림 12의 경우에는 표준 FCAST 시스템의 패킷 손실률에 따른 무손실 반복 전송 횟수를 나타낸다.

살펴보면, 본 시스템은 평균 패킷 손실률과 데이터 크기에 따른 반복 전송 횟수 증가에 대한 저항성이 있는 것으로 볼 수 있다. 먼저, 본 논문의 시스템은 평균 패킷 손실률의 증가에 따른 반복 전송 횟수의 증가 폭이 표준 시스템에 비해 작았다. 다음으로 데이터 크기의 증가에 따른 반복 전송 횟수 증가폭을 확인해보면, 본 시스템의 증가폭이 표준 FCAST 시스템보다 매우 작았다. 따라서 본 시스템은 대용량 데이터 전송에 적합하며, 평균 패킷 손실률이 높은 불안정한 데이터 환경에서도 안정적인 전송이 가능할 것으로 보인다.

또한 현재 FEC 시스템에서 8900 바이트의 데이터

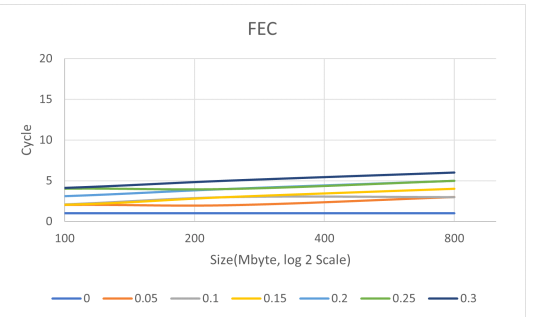


그림 11. FEC 시스템의 반복 전송 횟수 실험 결과
Fig. 11. FEC system cycle test result

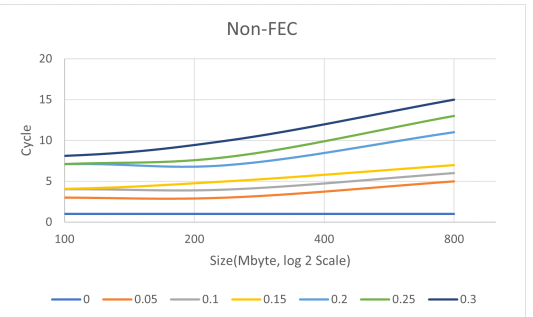


그림 12. FEC 미적용 시스템의 반복 전송 횟수 실험 결과
Fig. 12. Non-FEC system cycle test result

를 디코딩하여 복구하는 시간은 평균 125us이다. 손실이 발생하여 패킷을 재전송 해야 할 경우, 반복 전송 횟수 당 패킷간 전송 시간인 평균 800us가 필요한 것에 비해 현저히 적은 시간으로 전송이 가능하다.

4.3 국제 클라우드망 전송 실험에서의 성능 평가

표 2는 AWS(Amazon Web Services)가 제공하는 클라우드 환경을 사용하여 각 지역으로의 전송 성능 실험을 한 결과이다. 클라우드 환경 실험에서는 앞선 결과와 달리, 표준 시스템에 비하여 FEC 시스템의 평균 반복 전송 횟수가 크게 감소하지 않았다. 또한 FEC 시스템의 결과를 보면, 평균 패킷 손실률이 가장 큼에도 무손실 전송에 필요한 평균 반복 전송 횟수는 N. California가 가장 작게 나타났으며, 반대로 London의 경우에는 패킷 손실률이 작음에도 평균 반복 전송 횟수는 크게 나타났다. 그 이유는 FEC 시스템에서 평균 반복 전송 횟수는 평균 패킷 손실률이 아닌, 집중손실[10]에 연관되어 있기 때문이다.

실제 전송 환경에서는 패킷의 손실이 고르게 분포해있는 것이 아닌, 집중손실이 패킷 손실의 주를 이룬다. 실제 네트워크 환경에서의 손실은 패킷의 개수와 연관된 것이 아닌 시간과 연관되어 있기 때문이다.[11] 아래 그림 13은 동일한 평균 패킷 손실률을 가진 이상적인 경우와 실제 통신과정에서 실험적으로 얻어진 경우에서의 패킷 손실의 확률을 누적 분포도로 나타낸 것이다. 그림 13을 보면, 패킷 개수에 손실이 연관되어 있는 이상적인 경우에는 1ms 이내의 길이에서 대부분의 패킷 손실이 일어난다. 이는 패킷간 전송시간이 800us인 현재 시스템에서 보았을 때, 단일 패킷이 손실되는 비율이 대부분인 것을 의미한다. 반면 시간에 연관되어 있는 실제 환경에서는 2~3ms의 길이에서 패킷 손실이 집중되는 경향이 나타나는 것을 확인할 수 있다. 이는 3~4개의 패킷이 집중손실되는 비율이 가장 높다는 것을 의미한다. 또한 3ms 보다 긴 손실 지속 시간에서도 이상적인 경우보다 매우 높은

표 2. 국제 클라우드망 반복 전송 횟수 실험 (각 100만개 패킷 전송)
Table 2. Cycle test in international cloud networking (1 million packets are transmitted in each case)

Region	Avg PLR	Avg cycle non-FEC	Avg cycle in FEC
N.California	0.014	2.47	1.40
N.Canada	0.011	2.21	1.70
London	0.011	2.23	1.90
Tokyo	0.010	2.19	1.60

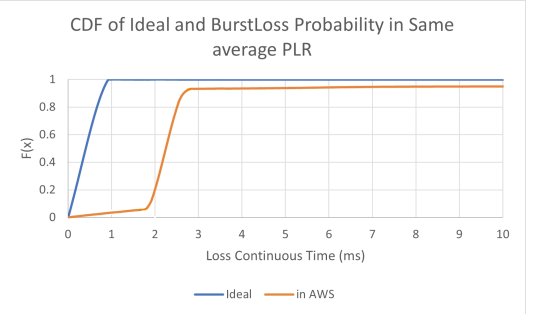


그림 13. 같은 평균 손실률을 가진 시스템에서의 이상적 확률과 집중손실 확률의 누적 분포도 (1000만개 패킷 전송).
Fig. 13. CDF of ideal and burstloss probability in same average PLR (10 million packets are transmitted).

확률 분포를 가지는데, 이는 실시간 네트워크 환경에 따라 긴 시간 동안 패킷 손실이 일어날 확률이 높기 때문이다.

현재 사용하고 있는 분수코드 방식은 이러한 집중손실에 취약점을 보여준다. 분수코드 방식은 N개의 패킷으로 구성된 한 묶음 공간 내에서 데이터 패킷의 수인 K개 이상이 수신될 경우, 복구가 가능한 방식이다. 그러나 가중치가 저장된 행렬의 각 차원이 전부 독립적이지는 않기 때문에, 실제로는 K개보다 조금 더 많은 패킷이 수신되어야 할 필요가 있다. 현재 실험환경은 K가 49, N은 99로 최소 49개 이상의 패킷이 수신되어야 하며, 40개 이상의 패킷이 손실될 경우 FEC에 실패할 가능성이 나타난다. 즉, 32ms 이상의 길이를 가지는 패킷 손실에 대해서는 FEC에 실패할 수 있다.

표 3과 같이, 평균 반복 전송 횟수가 가장 작게 나타난 N. California는 32ms 이상의 시간동안 연속적으로 패킷이 손실되는 비율이 4개의 지역 중 가장 작게 나타났다. 또한 평균 반복 전송 횟수가 가장 크게 나타난 London의 경우, 32ms 이상의 시간동안 패킷이 손실되는 비율이 각 지역 가운데 가장 크게 나타났

표 3. 국제 클라우드망 집중손실 시간 실험 (각 100만개 패킷 전송)
Table 3. Loss continuous time test in international cloud networking (1 million packets are transmitted in each case)

Region	Avg cycle in FEC	Over 32ms loss continuous time	Over 96ms loss continuous time
N.California	1.40	1.88E-05	5.22E-07
N.Canada	1.70	3.14E-05	0
London	1.90	4.50E-05	0
Tokyo	1.60	3.01E-05	0

다. 이는 긴 시간동안의 패킷 손실이 FEC의 실패를 야기하고, 결과적으로 평균 반복 전송 횟수의 증가로 이어진다는 것을 나타낸다.

따라서 FEC를 실제 전송 환경에 적용하기 위해서는 이러한 집중손실을 제어해야한다. 이를 위해서, 집중손실을 다루는 인터리빙(interleaving) 방식을 적용하는 사고실험을 추가 진행하였다.

4.4 집중손실 제어 사고 실험

분수코드 방식에서 집중손실을 제어하는 방식은 묶음공간을 매우 크게 하는 방식[12]과 데이터가 서로 인접하지 않도록 전송을 하는 방식인 인터리빙 방식[6]이 있다. 본 사고 실험에서는 시스템이 여러 객체와 여러 조각을 보내는 점에서 구현이 용이한 후자의 방식을 사용하였다.

인터리빙 방식은 데이터 전송의 우선순위를 변경함으로서 얻을 수 있다. 그림 14는 인터리빙 방식이 적용된 이후의 전송 우선순위[7]를 보여준다. 본래의 시스템은 패킷의 순서대로 데이터를 전송하고, 이후 묶음공간, 조각, 객체의 순서대로 데이터를 전송하였지만, 인터리빙이 적용된 방식에서는 기존의 시스템과 반대의 우선순위를 가지게 된다. 우선 각 객체의 패킷을 하나씩 전송을 하고, 이후는 각 조각, 묶음공간, 마지막으로 패킷의 연속 순서대로 전송을 시작한다.

인터리빙 방식을 사용할 경우, 디코딩 과정은 인터리빙 방식으로 묶인 데이터가 전부 전송이 된 이후에 시작된다. 따라서 전송하고자 하는 객체나 조각의 크기만큼 메모리 자원을 할당하여야 하며, 실시간성이 떨어진다는 단점이 있다. 그러나 한 묶음공간 안에 패킷들은 연속되게 전송이 되지 않고, 모든 묶음 공간의 수만큼 패킷이 전송된 이후에 전송된다. 따라서 집중손실이 일어나더라도, 데이터가 연속되어있지 않기 때문에 정상적으로 디코딩이 가능하다.

현재 실험 환경에서 한 개의 조각은 세 개의 전부 채워진 묶음공간을 가진다. 여기서 조각 단위의 인터리빙을 적용할 경우, 기존의 방식에서는 40개 이하의 패킷 집중손실에 디코딩이 가능하였던 것에 비해 120개의 패킷까지 집중손실에 대해서도 디코딩이 가능하다. 120개의 패킷이 집중손실 되더라도, 디코딩이 이루어지는 각각의 묶음 공간에 40개씩 패킷이 손실되는 것과 동일하기 때문이다. 결론적으로 조각 단위의 인터리빙을 적용할 경우, 복구가 가능한 패킷 손실 시간의 길이가 기존의 32ms에서 96ms로 늘어나게 된다.

클라우드 환경 실험에서 32ms 이상의 시간 동안 패킷 손실이 일어날 확률은 3.23e-03(%)였다. 반면 96ms 이상의 패킷이 집중손실될 확률은 1.11e-05(%)로 나타났다. 이는 96ms 이상의 손실이 일어날 확률이 32ms 이상의 손실에 비해 300배 가까이 낮다는 것을 의미한다. 즉, 조각단위의 인터리빙을 적용할 경우, 집중손실에 대하여 300배 정도의 저항성을 확보할 수 있고, 그만큼 반복 전송 횟수를 감소시킬 수 있을 것이다.

그림 15는 표 2의 국제 클라우드망 실험 결과에 인터리빙과 FEC가 동시에 적용된 시스템의 사고실험 결과를 추가한 것이다. 살펴보면, 반복 전송횟수는 non-FEC 시스템 보다 FEC 하에서 더 낮게 나타났지만, 복구 데이터라는 부가적 데이터의 전송으로 인하여 필요 데이터양은 오히려 증가한 것으로 나타났다. 하지만, 인터리빙이 적용된 FEC 시스템의 경우에는 필요 반복 전송 횟수가 대폭 감소함과 동시에 필요 전송 데이터양도 non-FEC 시스템 대비 10%정도 낮게 나타날 것으로 예측되었다.

또한 앞서 말한 인터리빙 방식에서의 단점으로 많은 메모리 할당이 필요하다는 점이 있었다. 그러나 인

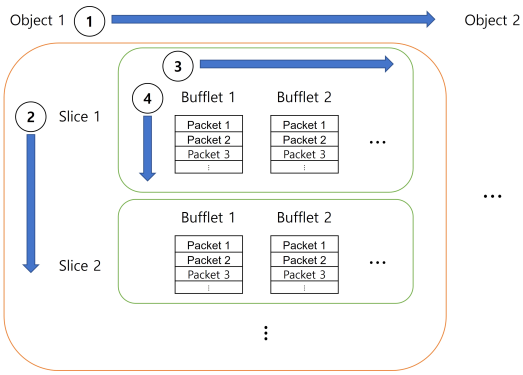


그림 14. 인터리빙 방식이 적용된 전송 우선순위
Fig. 14. Transmission priority with interleaving

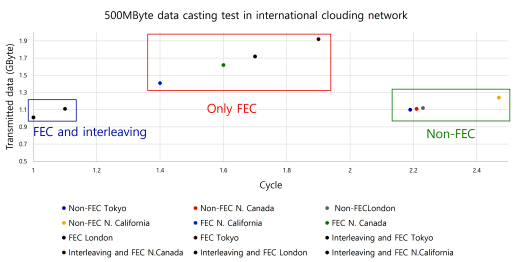


그림 15. 국제 클라우드망에서의 전송 데이터양과 반복 전송 횟수 실험 (500메가바이트 데이터 전송)
Fig. 15. Transmitted data and cycle test result in international clouding network (500Mbyte data casting)

터리빙 방식을 조각 단위로 구현할 경우, 그 크기가 현 시스템 기준 약 35.3 MB 정도로 모바일 장치에서도 메모리 할당에 큰 무리가 없을 것으로 보인다. 이처럼 전송 환경에 따라서 인터리빙 단위를 결정함으로서 메모리 할당의 문제도 해결할 수 있을 것이다.

V. 결 론

본 논문에서는 IETF의 FCAST 표준에서 헤더 안정성 향상과 FEC를 적용한 시스템을 구현하였다. 먼저 기존 표준의 FCAST 헤더의 초기화 불안정 문제를 해결함으로서, 모든 확장자에 대하여 데이터 전송이 가능하도록 구현하였다. 다음으로 표준 FEC 시스템 대비 무손실 전송에서 필요한 반복 전송 횟수의 감소를 통해 평균 패킷 손실률에 대한 저항성을 향상시켰다. 그러나 실험결과, 국제 클라우드망에서는 패킷 집중손실의 경향성으로 인하여 그 효과가 감소하는 한계성이 나타났다. 그러나 사고실험에서 확인한 바와 같이, 이러한 문제점은 인터리빙 방식의 추가 구현을 통하여 해결이 가능할 것으로 보인다.

그러나 본 논문의 시스템은 패킷 손실률이 낮은 우수한 품질의 네트워크 환경에서는 오히려 복구 패킷이라는 부가적 정보를 보내야 하는 점, 인터리빙 방식에서 많은 메모리 할당을 하여야 한다는 점에서 한계성을 지닌다. 이러한 문제들을 해결하기 위해 서비스 품질 조절을 활용하는 후속연구가 필요하다.

구체적으로 첫째, 복구 패킷으로 인한 전송 시간 증가의 해결을 위하여 서비스 품질 조절을 통한 복구 패킷과 데이터 패킷의 비를 시스템 자체적으로 결정하도록 시도해볼 필요가 있다. 예를 들어, 패킷 손실률이 높은 환경이라면 복구 패킷의 비를 증가시켜 패킷 손실률에 대한 저항성을 더욱 확보하고, 패킷 손실률이 낮은 환경이라면 복구 패킷의 비를 감소시켜 복구 패킷이라는 부가적 정보 전송으로 인한 전송시간의 증가를 최소화할 수 있을 것이다.

둘째, 메모리 할당을 최적화하기 위하여, 서비스 품질 조절을 통한 패킷의 집중손실이 최대 몇 개의 패킷까지 발생하는지 확인하고 인터리빙 방식의 단위를 결정하는 방안을 고려해볼 수 있다. 발생된 집중손실 패킷 수 가 적을 경우, 조각 단위의 인터리빙 방식을 적용하여 메모리 할당을 최소화 하면서도 집중손실 제어를 할 수 있을 것이다. 반면 집중손실 패킷 수가 많을 경우, 객체 단위의 인터리빙 방식을 통해 더욱 확실한 집중손실 제어를 할 수 있을 것이다. 후속 연구를 통해 본 시스템의 한계점들이 보완되어, 네트워크 환경에 따라 본 시스템의 FEC와 인터리빙 방식이 유동적으로 적용될 수 있기를 기대하는 바이다.

References

[1] V. Roca and B. Adamson, "*FCAST: Object Delivery for the Asynchronous Layered Coding (ALC) and NACK-Oriented Reliable Multicast (NORM) Protocols*," RFC 6968, Oct. 2013.

[2] M. Luby, M. Watson, and L. Vicisano, "*Asynchronous Layered Coding (ALC) Protocol Instantiation*," RFC 5775, Apr. 2010.

[3] B. Adamson, C. Bormann, M. Handley, and J. Macker, "*NACK-Oriented Reliable Multicast (NORM) Transport Protocol*," RFC 5740, Nov. 2009.

[4] M. Luby, A. Shokrollahi, M. Watson, and T. Stockhammer, "*Raptor Forward Error Correction Scheme for Object Delivery*," RFC 5053, Oct. 2007.

[5] J. Gemmell, J. Gray, and E. Schooler, "Fcast multicast file distribution," in *IEEE Network*, vol. 14, no. 1, pp. 58-68, Jan.-Feb. 2000.

[6] B. Jo, D. Lee, and D. Y. Suh, "Adaptive QoS study for video streaming service in MMT protocol," *J. Broadcast Eng.*, vol. 20, no. 1, pp. 40-47, Korea, Jan. 2015.

[7] A. Botta and A. Pescapé, "IP packet interleaving for UDP bursty losses," *J. Syst. and Softw.*, vol. 109, pp. 177-191, Mar. 2015.

[8] H. H. F. Yin, K. H. Ng, A. Z. Zhong, R. W. Yeung, and S. Yang, "Intrablock interleaving for batched network coding with blockwise adaptive recoding," in *Proc. ISIT 21*, Jul. 2021.

[9] K. Matsuzono, J. Detchart, M. Cunche, V. Roca, and H. Asaeda, "Performance analysis of a high-performance real-time application with several AL-FEC schemes," *IEEE Local Comput. Netw. Conf.*, 2010, pp. 1-7, Denver, CO, USA, Oct. 2010.

[10] S. Kim Chin and R. Braun, "A survey of UDP packet loss characteristics," *Conf. Record of Thirty-Fifth Asilomar Conf. Sign., Syst. and*

Comput., vol. 1, pp. 200-204, CA, USA, Nov. 2001.

[11] M. Arai, A. Chiba, and K. Iwasaki, "Measurement and modeling of burst packet losses in Internet end-to-end communications," in *Proc. 1999 Pacific Rim Int. Symp. Dependable Comput.*, pp. 260-267, HongKong, China, Dec. 1999.

[12] V. Roca, M. Cunche, C. Thienot, J. Detchart, and J. Lacan, "RS +LDPC-Staircase Codes for the Erasure Channel: Standards, Usage and Performance," *9th IEEE Int. Conf. WiMob 2013*, Lyon, France, Oct. 2013.

[13] M. McKeay, *The Building Wave of Internet Traffic*(2020), Akamai Technologies Inc., Retrieved Mar. 3, 2021, from https://blogs.akamai.com/kr/2020/07/the-building-wave-of-internet-traffic.html

사 봉 준 (Bong Jun Sah)

2015년 3월~현재 : 경희대학교 전자공학과 학사과정

<관심분야> data casting, cloud computing, multicast, networked media

[ORCID:0000-0002-8268-5423]

서 덕 영 (Doug Young Suh)

1990년 : 미국 조지아텍 전기 및 컴퓨터공학과 박사

1992년~현재 : 경희대학교 전자정보대학 교수

<관심분야> networked media, computer game

[ORCID:0000-0003-3120-0737]

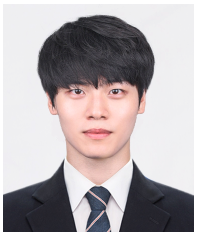
원 대 한 (Dae Han Weon)

2016년 3월~현재 : 경희대학교 전자공학과 학사과정

<관심분야> 전자공학, 이동통신, LTE, NR

[ORCID:0000-0002-0245-7798]

정 철 현 (Cheol Hyeon Jeong)

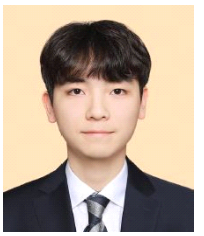


2017년 3월~현재 : 경희대학교 전자공학과 학사과정

<관심분야> AI, data science, media processing

[ORCID:0000-0001-9687-0788]

차 은 영 (Eun Young Cha)



2020년 2월 : 경희대학교 전자공학과 학사 졸업

2020년 3월~현재 : 경희대학교 전자 공학과 석사과정

<관심분야> media processing, networked media, AI

[ORCID:0000-0002-7734-6697]