pISSN: 2287-5891

http://dx.doi.org/10.3745/KTCCS.2013.2.4.163

Reduction of Presentation Latency in Thin-Client of Cloud System

Seung Soo Kang[†] · Hyun Ko[††] · Hee Yong Yoon[†††]

ABSTRACT

Cloud-based streaming game service has numerous merits, but it may suffer from presentation latency in a thin-client. It is an important issue especially for game service which needs instantaneous response to user inputs. This research proposes the methods for reducing the presentation latency between the server unit and the thin-client unit. The approaches proposed to be employed with server unit include the source/sync video format equalization, encoding format configuration according to the media type, and the S/W implementation for transmitting clock periodically. The methods for the thin-client unit include the decreasing the number of instructions, use of light encryption algorithm, and improvement on H/W decoding. The proposed schemes are tested with a commercialized streaming service platform, which reveals the reduction of presentation latency as large as a few hundred milliseconds and reaches the acceptable level (about 100 milliseconds).

Keywords : Cloud Computing, Streaming Service, Game, Consumer Electronics Device, Presentation Latency

클라우드 시스템의 씬 클라이언트에서의 표시 지연 절감

강 승 수[†] · 고 현[††] · 윤 희 용[†††]

요 약

클라우드 시스템을 이용하여 스트리밍 게임 서비스를 제공하는 것은 많은 유리한 점이 있지만, 씬 클라이언트에서 표시 지연(Presentation latency)이 발생한다는 문제점도 갖고 있다. 게임은 사용자의 입력에 즉각적으로 반응해야 하는 특성을 갖는 서비스이므로 씬 클라이언트에서의 표시 지연은 다른 이슈에 비해 중요한 문제이다. 본 연구에서는 서버와 씬 클라이언트 사이에서의 표시 지연을 감소시키는 방법을 제안한다. 이를 위해, 서버 단에서는 서버와 씬 클라이언트의 영상포맷 일치화, 미디어 타입에 따른 인코딩 포맷 변경, 동기화를 위해 주기적으로 클럭을 전송하도록 하는 것을 해결 방안으로 제안한다. 그리고 씬 클라이언트 단에서는 패킷화 과정에서 명령어 수 절감, 암호화 과정에서 가벼운 알고리즘 사용, 압축해제 과정에서 하드웨어 디코딩 개선을 해결 방안으로 제안한다. 이를 실제 상용화 수준까지 개발된 게임 서비스 시스템에 적용하여 검증 하였는데, 수백 ms 정도의 상당한 표시 지연 감소를 통해 허용 가능한 수준인 100ms까지 표시 지연 감소가 가능함을 확인하였다.

키워드 : 클라우드 컴퓨팅, 스트리밍 서비스, 게임, 소비 가전 기기, 표시 지연

1. 서 론

본 연구에서는 클라우드 컴퓨팅의 한 응용 분야인 클라우드 스트리밍 게임 서비스를 구축하는데 있어 발생하는 이슈와 그에 대한 해결 방안을 제시한다. 특히, 저렴한 비용을 갖지만 매우 제한적인 성능을 보이는 소비가전(Consumer Electronics) 기기에 제공되는 스트리밍 게임 서비스를 대상으로 하며, 해당 조건에서 발생할 수 있는 이슈를 파악하고 그에 대한 해결방법과 실험을 통해 그 효용성을 검증한다.

† 정 회 원 : 성균관대학교 전자전기컴퓨터공학과 석사과정수료
†† 정 회 원 : 엘지전자 주임연구원
††† 종신회원 : 성균관대학교 정보통신대학 교수

논문접수 : 2012년 12월 18일
수 정 일 : 1차 2013년 3월 18일
심사완료 : 2013년 3월 27일

* Corresponding Author : Hee Yong Yoon(youn@ece.skku.ac.kr)

클라우드 스트리밍 게임은 프로그램의 배포가 클라우드 형식으로 처리 되지만 게임 자체는 로컬 기기에서 실행된다는 점에서 기존 다운로더블(downloadable) 게임과 명백히 다른 방식이다. 기존의 다운로더블 게임은 게임에 필요한 소프트웨어를 서버로부터 다운로드 받아 실제 게임의 구동이 클라이언트 기기에서 이뤄지는 방식이다. 반면에 클라우드 스트리밍 게임은 클라우드 서버에서 게임을 구동하여 오디오/비디오(Audio/Video)데이터를 사용자의 클라이언트 기기에 송출하고, 클라이언트 기기는 서버로부터 오디오/비디오데이터를 받아 출력하고 사용자 입력을 받아 그 데이터를 클라우드 서버에 송출하는 방식으로 이뤄진다.

본 연구에서는 소비가전 기기를 클라이언트 기기로 사용하는 실제 상용화 클라우드 게임 서비스를 구축하면서 발생할 수 있는 이슈와 그 솔루션을 제안한다. 또한, 본 연구는

Software as a Service(SaaS)[1, 2]로서 가장 엄격한 실시간성을 요구 받는 게임 서비스를 대상으로 하므로, 본 연구의 결과는 더 낮은 실시간성이 요구되는 인터넷 서핑이나 동영상 제공과 같은 서비스를 제공하는 데에도 폭넓게 적용될 수 있다.

이하 내용은 본 연구의 유니크한 제안 내용이라기 보다, 실증적으로 개발한 클라우드 스트리밍 서비스에서의 사례를 소개하는데 더 의의를 둔다. 실제 상용화되고 실증적으로 개발한 클라우드 스트리밍 서비스에서 아래와 같은 시도가 어떤 효과를 보일 수 있는지 본 연구를 통해 공유하는데 의의가 있다.

2. 관련 연구

본 연구는 클라우드 스트리밍 게임 서비스를 위해 실제적으로 상용화된 디지털 신호 프로세서(Digital Signal Processor)를 사용하는 소비가전 기기를 대상으로 한다. 소비가전 기기 중 특히 디지털 셋톱박스(Set Top Box)/디지털 텔레비전를 대상으로 한 클라우드 스트리밍 게임 서비스를 중점으로 논의한다.

클라우드 스트리밍 게임 서비스는 씬 클라이언트의 하위 개념에 속한다. 씬 클라이언트의 정의는 그 구현 방식에 있어서 매우 넓은 범위를 포괄한다. 예컨대 씬 클라이언트 애플리케이션을 다운받아 서비스를 제공하는 방식, 서버-클라이언트 동기에 있어서 주기적이나 필요 시에만 동기를 이루는 방식, 본 연구처럼 일정 수준의 실시간 동기를 이루는 방식 등 다양한 방식이 존재한다.

씬 클라이언트 개념은 본 연구주제를 설명하기에 광의의 개념이므로 클라우드 스트리밍 게임 서비스라는 용어를 혼용하여 설명하도록 하겠다. 본 장에서는 우선 클라우드 컴퓨팅에 대한 간략한 소개와 그 특성을 소개한다. 그리고 세부항목인 클라우드 스트리밍 게임 서비스의 동작 원리에 대해 설명한다.

2.1 클라우드 컴퓨팅 및 클라우드 스트리밍 게임 서비스의 개요

1) 클라우드 컴퓨팅

클라우드 컴퓨팅은 1960년대 미국의 저명한 학자인 John McCarthy가 제시한 "컴퓨팅 환경은 공공 시설을 쓰는 것과도 같을 것"에서 유래하였다. 클라우드 컴퓨팅은 무형의 형태로서, 컴퓨터 하드웨어를 하이퍼바이저(Hypervisor)라는 하드웨어와 운영체제 사이의 소프트웨어 레이어를 통해 가상화시키는 방식을 사용한다[3]. 또한, 조직된 컴퓨팅 자원을 편재성(ubiquitous)있고 편리하며, 수요 중심형(on-demand)으로 제공할 수 있게 하는 모델을 지칭한다[4].

클라우드 컴퓨팅은 크게 다음과 같이 세 종류로 구분될 수 있다. 스토리지나 컴퓨팅 파워와 같이 인프라스트럭쳐를 공유하는 Infrastructure as a Service (IaaS), 소프트웨어 개발 툴(tool)을 제공하거나 애플리케이션 실행환경을 제공하는 Platform as a Service (PaaS), IaaS, PaaS 상에서 구동되는 애플리케이션을 제공하는 SaaS가 그것이다[1, 2].

하지만 위의 분류기준에 해당되기 어려운 클라우드 서비스도 존재한다. 예를 들면, Apple社의 앱 스토어(App store)의 경우 거대한 데이터 센터 대신 3G 무선 단말기에서 소프트웨어가 구동된다는 점에서 SaaS와 다르다[5]. 마찬가지로 스트리밍 게임은 클라이언트 입장에서는 로컬에서 구동되는 소프트웨어의 면모를 갖는 SaaS에 속하지만 거의 대부분의 컴퓨팅 자원을 클라우드 서버를 통해 획득한다는 점에서 IaaS의 성격을 지닌다.

2) 클라우드 스트리밍 게임 서비스

클라우드 스트리밍 게임 서비스는 원격의 클라우드 서버에서 게임을 실행하고 출력으로 오디오/비디오데이터를 특정 코덱으로 인코딩하여 네트워크 패킷으로 전송하면, 서버가 전송한 패킷을 클라이언트가 받아 재생하는 과정으로 이뤄진다. 그리고 클라이언트에 연결된 조이스틱이나 키보드/마우스와 같은 입력장치의 신호를 클라이언트에서 받아 이를 다시 네트워크 패킷으로 만들어 서버에게 전달한다. 서버에게 전달된 입력신호는 서버에서 구동되고 있는 게임의 입력으로 처리되어 사용자의 입력이 게임에 반영되게 된다. Fig. 1은 클라우드 스트리밍 게임 시스템의 전반적인 구조를 나타낸다.

Fig. 1. Cloud Streaming Game System

2.2 클라우드 컴퓨팅의 특성

1) 서비스 지향

클라우드 컴퓨팅은 사용자에게 불투명(opacity)하며, 가상화(virtualization) 등 여러 기술을 통해 근본적인 아키텍처가 사용자에게 많이 노출되지 않도록 추상화(abstraction)된다[5]. 즉, 사용자는 클라우드 컴퓨팅의 시스템 구조에 대한 이해가 없이도 클라우드 컴퓨팅 기반의 서비스를 이용할 수 있다.

또한, 클라우드 컴퓨팅은 접근성(accessibility)이 뛰어나며, 프로세싱 능력이나 저장 용량과 같은 시스템 변수들을 이용하여 클라우드 컴퓨팅의 모든 역량(capability)을 쉽게 활용할 수 있다. 제공되는 역량의 타입에 따라 앞에서와 같

이 IaaS, PaaS, SaaS로 분류된다[5].

따라서, 클라우드 스트리밍 게임 서비스를 받는 사용자는 현재 게임이 어디에서 구동되고 있는 것인지, 어떻게 구동되고 있는 것인지 알 필요 없이 제공되는 인터페이스를 통해 게임을 선택하고 게임을 즐기기만 하면 되는 것이다.

2) 독립적 연결

클라우드 컴퓨팅에서는 한 부분의 동작이 다른 부분에 거의 영향을 미치지 않는다. 클라우드 사용자는 서로 독립적이며, 클라우드 제공자도 서로 독립적이다. 또한, 각 클라우드 제공자와 클라우드 사용자 간의 연결도 독립적이다[5].

예를 들면, 검색 서비스를 사용하는 세 명의 사용자가 각각 다른 세 개의 검색 엔진을 동시에 사용하는 경우, 각 사용자는 서로 독립적이고 각 검색 엔진도 서로 독립적이다. 그리고 사용자와 검색 엔진 간의 세 개의 결합도 각각 독립적으로 서로 영향을 받지 않는다[5].

따라서 복수의 게임을 사용하는 복수의 사용자가 있는 경우에도 각각의 독립성이 보장되므로, 클라우드 시스템은 확장성(scalability)과 결함에 대한 내성(fault tolerant)을 가질 수 있다.

3) 결함에 대한 강한 내성

앞서 언급한 바와 같이 클라우드 컴퓨팅에서는 두 개의 처리 과정이 서로 독립적이므로, 시스템에서의 부분적인 결함이 연쇄 반응을 유발하지 않는다[5]. 서버에서 구동되는 복수 개의 게임 중 하나, 또는 게임을 사용 중인 복수의 사용자 중 하나로부터 유발된 결함이 있다고 하더라도, 다른 게임이나 다른 사용자에게 미치는 영향이 미미하므로 게임 서비스가 제공되는 동안 전체 클라우드 시스템이 매우 안정적으로 구동될 수 있다.

4) 다양한 비즈니스 모델

비즈니스 모델이란 서비스가 고객에게 제공하려는 가치와, 그 가치를 생산하고 마케팅하고 전달하기 위한 구조와 파트너와의 네트워크에 대한 정의이다. 즉, 서비스가 어떻게 동작하는 가에 대한 내용을 말한다[2].

클라우드 컴퓨팅은 2007년 IBM과 Google의 합작에 의해서 보편화되기 시작했으며, 이와 같이 거대 IT기업의 지지를 받게 됨에 따라 클라우드 컴퓨팅을 활용한 다양한 비즈니스 모델이 발굴되었다[5]. 스트리밍 게임 서비스도 이러한 맥락하에서 발굴된 서비스로, 현재 OnLive, StreamMy Game, GAikai, G-Cluster, OTOY 등 여러 서비스가 제공되고 있다. 이 서비스 중 일부는 개인 컴퓨터에서 접근이 가능하고 또 다른 일부는 텔레비젼이나 셋톱박스에서 접근이 가능하다[6].

5) 사용 편의성

사용자 경험(User experience)은 애플리케이션의 성공여부를 평가할 때 중요한 항목 중 하나이며, 사용자 경험의 핵심은 사용 편의성이다. 대부분의 클라우드 제공자는 단순한 인터넷 기반의 인터페이스를 사용하고, 사용자 요구 분석에 기반한 사용자 경험 디자인을 전체 기능 디자인의 근간으로 삼고 있다. 또한, 소프트웨어와 인터넷의 경계가 불분명해지면서 웹 애플리케이션이나 서비스들이 소프트웨어와 유사하게 변화되었다[6].

클라우드 스트리밍 게임 서비스에 접근하기 위한 사용자 경험도 Chen[6]의 연구처럼 기존의 콘솔 게임에 대한 사용자 경험과 유사하게 설계되고 있다.

2.3 클라우드 스트리밍 게임 서비스에 대한 선행 연구

1) 클라우드 기반 온라인 게임에서의 용인 가능한 수준의 지연에 대한 선행 연구

Claypool[7]의 연구는 클라우드 기반의 온라인 게임에서 게임 상 플레이어(player)의 동작에 대해 발생되는 지연에 대해 연구이다. 온라인 게임을 아바타(Avatar) 모델과 옴니프레즌트(Omnipresent) 모델로 구분하였으며, 아바타 모델을 시점에 따라 일인칭 시점의 게임과 삼인칭 시점의 게임으로 분류하였다.

일인칭 시점 게임이란 게임 사용자가 게임 내의 아바타의 눈을 통해 바라보는 타입의 게임을 말한다. 예를 들면, 일인용 사격게임을 들 수 있다. 삼인칭 시점 게임이란 게임 사용자가 가상 세계에서 게임 내의 아바타를 따라가는 타입의 게임을 말하고, 롤플레잉 게임(role-playing game)이 그 예가 될 수 있다.

Fig. 2. *Doom*, Example of a First Person Perspective[8]

Fig. 3. *Red Dead Redemption*, Example of a Third Person Perspective[9]

Claypool[7]의 연구에 따르면 게임의 모델 및 시점에 따라 용인 가능한 지연의 수준이 다르다. 예를 들어, 일인칭 시점 게임의 경우 100ms 내외의 지연이 용인 가능한 수준이나, 삼인칭 시점 게임의 경우는 500ms, 옴니프레즌트 모델의 경우 1000ms이 용인 가능한 수준이다.

따라서 아바타 모델의 일인칭 시점의 게임이 지연에 있어서 가능 민감한 게임이며, 본 연구에서는 일인칭 시점의 게임에서의 용인 가능한 지연 수준(100ms)으로 지연을 줄이기 위한 연구를 진행하였다.

2) 지연을 유발하는 요소와 지연을 감소시키는 방법에 대한 선행 연구

Sharp[10]의 연구에서는 클라우드 기반 상호작용 스트리밍 컨텐츠(Cloud-Based Interactive Streaming Content)에 있어 지연(latency)에 대한 연구이다. 해당 연구의 주요한 내용은 용인 가능한 지연(acceptable latency)은 어느 정도인가와 그 지연요소는 무엇인지에 대해 다루고 있다.

Sharp[10]의 연구에서 용인 가능한 지연은 사용하는 애플리케이션, 작업 등에 다르지만 대개 편도 50~100ms 정도가 적합하다고 주장한다. 이러한 주장은 - 가장 엄격한 실시간성을 요구하는 애플리케이션에 속하는 - 게임 애플리케이션을 중심으로 인적 조사(18-25세 사이의 게이머)를 기반으로 한 만족도 조사에 근거로 하고 있다.

Sharp[10]는 지연 단축방법으로 프레임 처리의 파이프라인을 순차처리가 아닌 병렬 방식으로 처리할 것을 제안하였다. 이를 통해 병렬성을 증가시켜 전체 처리 시간을 감소시킬 수 있다. 또한 인코더 효율제고를 제안하고 있는데, 이는 인코더를 독립 소자로 분리하여 CPU의 자원을 사용하지 않고 FPGA등으로 디지털 신호 프로세서의 구성 효율을 제고하는 방법이다. 그 외 지연 절감을 위한 fps(frame per second) 조절(tradeoff) 등의 방법도 제시되고 있다.

본 연구에서는 Sharp의 연구를 예각화/심화시켜 보다 구체화된 지연 절감 방법의 제시 및 미디어 형태에 따른 fps 조절(tradeoff) 방법 등을 제시한다. 또한 본 연구에서의 제안은 특히 10000개 내외의 클라이언트가 붙는 상용화된 서비스의 경험을 바탕으로 하고 있다는 점에서 이후 관련 연구에 있어 유의미성을 갖는다.

3. 클라우드 스트리밍 게임 서비스의 구현

3.1 시스템 구조

1) 서버

우선 앞서 언급한 클라우드 스트리밍 게임 서비스의 구동원리를 보다 구체적으로 설명 한다.

클라우드 서버 또는 서버 집단은 크게 애플리케이션 구동 모듈과 인코딩 모듈 그리고 전송 모듈로 나뉘어진다. 애플리케이션 구동 모듈은 게임 애플리케이션을 구동하는 모듈로, 실제 게임이 여기서 구동되며 클라이언트의 조이스틱 입력과 같은 입력이 이곳에서 처리된다. 본 연구에서 구현한 시스템에서는 구동이 용이한 개인 컴퓨터(PC) 운영체제용 게임을 사용한다. 그러나, 콘솔용 게임을 사용하는 것도 가능하다.

인코딩 모듈은 구동된 게임의 오디오/비디오 데이터를 받아 코덱으로 압축하는 역할을 하는 모듈이다. 비디오 인코딩에 있어서는 압축률에 유리한 H.264 압축을 대부분 사용하며 필요에 따라 MPEG2 압축을 취하기도 한다. 오디오 압축에 있어서는 일반적으로 PCM, AAC, AC3 등의 포맷이 사용되는 데 본 연구에서는 오디오/비디오 동기화에 유리하며 재생 속도가 빠른 PCM을 사용한다. 그리고, 인코딩 모듈은 서버 단에서의 가장 큰 지연 요소가 될 수 있으므로 하드웨어 인코더를 사용하는데, 본 연구에서는 NVIDIA GeForce GRID technology 를 이용한다.

전송 모듈은 인코딩된 스트림을 네트워크 패킷화하여 클라이언트로 전송하는 역할을 담당하는데, 일반적으로 유니캐스트 방식의 전송방식을 사용한다. 이는 각 스트리밍 게임이 주로 한 사용자에게 제공되며 사용자 입력에 따라 스트림을 보내야 하므로 브로드캐스트 /멀티캐스트의 사용이 필요 없기 때문이다. 그러나 원격에 있는 복수의 사용자가 하나의 게임을 함께 즐길 경우 브로드캐스트/멀티캐스트 전송 방식이 사용될 수 있다. 또한, 일반적으로 영상 스트림은 I frame으로 인코딩 하거나 I frame과 P frame을 사용하여 인코딩 하는데, 본 연구에서는 I-P frame을 사용하여 인코딩 하는 방식을 사용한다.

초기의 클라우드 스트리밍 게임은 위의 세 가지 모듈이 물리적으로 하나의 서버에 연결된 구조였고 인코딩 모듈도 하드웨어 인코더를 사용하는 것이 아니라 주 Central Processing Unit(CPU)를 이용한 소프트웨어 인코딩을 사용하였다. 그러나, 세가지 모듈이 하나의 서버에 연결된 구조는 많은 자원을 소비하는 게임 애플리케이션을 구동하는 경우 인코딩 모듈이나 전송 모듈의 처리 속도에 영향을 주게 되고 이는 지연으로 연결되어 표시 지연을 늘리는 현상이 나타나게 된다. 또한, 서버의 확장성(Scalability)이나 유지보수성 측면에서 세 모듈이 서로 영향을 준다는 점에서 바람직하지 않다. 따라서, 최근에는 이 세 가지 모듈이 물리적으로 분리되어 있는 형태로 클라우드 서버가 구성되고 있다. 특히, 구동 모듈과 인코딩 모듈 사이에서는 시스템 사이의 결합도(coupling)를 낮추기 위해 HDMI 연결만을 유지하여 처리하는 방식이 효율적인 것으로 알려져 있다. 한편, 전송 모듈은 대규모 서비스에 대응하여 L4 스위치 부하분산과 같은 방법을 적용하여 네트워크 부하를 분산시키고 있다.

2) 클라이언트

클라우드 서버에서는 게임 구동 모듈, 인코딩 모듈 순으로 데이터가 처리된 후 전송 모듈을 통해 클라이언트로 전달되게 된다. 클라이언트는 크게 수신 모듈, 디코딩 모듈, 입력 모듈로 구성된다.

수신 모듈은 클라이언트에서 패킷을 프레임 단위로 결합하는 역할을 담당하고, 실시간 전송 프로토콜(Real-time

Transport Protocol)를 사용한 전송을 하는 경우에는 순서관계를 맞추어 조립하는 작업도 겸하게 된다. 또한, 암호화된 스트림을 복호화하는 작업도 수신모듈의 역할이다. 본 연구에서는 빠진 패킷이나 프레임을 버퍼링하여 사용하지 않고 폐기하는 방식을 채택하며, 송신된 패킷은 꼭 전달을 보장받지 않아도 되는 최선 노력(Best Effort) 정책을 따르도록 한다. 이는 게임 서비스라는 서비스 고유의 성격에 근거하여 채택하였다.

수신 모듈로부터 디코더에서 재생될 수 있는 형태로 전달된 데이터는 디코딩 모듈을 통해 디코더로 공급된다. 일반적으로 클라우드 스트리밍 게임 서비스에서는 MPEG과 같은 컨테이너를 사용하지 않는다. 컨테이너를 사용함으로써 얻는 전송의 안정성보다는 빠른 전송/재생이 더 중요하기 때문이다. 물론, 서비스의 성격에 따라 MPEG 컨테이너 형태로 가공하여 사용하는 경우도 있다. 따라서, 보통 H.264 형태로 압축된 기본 스트림(Elementary Stream) 형태로 디코더에게 전달되게 된다. 본 연구에서는 온칩(On-Chip) 형태로 구현된 하드웨어 디코더를 사용하는 것을 상정하고 있다. 이는 소비가전 오디오/비디오 기기(디지털 텔레비젼, 디지털 셋톱박스, 블루레이 디스크 플레이어, DVD 플레이어)들이 보통 온칩형태의 하드웨어 디코더를 내장하고 있기 때문이다.

입력 모듈은 일반적으로 소비가전 기기에 연결된 입력기기(키보드/마우스/조이스틱)들의 신호를 받고 이를 패킷화하여 서버에게 전송하는 모듈을 의미한다. 본 연구에서는 Universal Serial Bus(USB)로 연결된 표준 Human Interface Device(HID) 와 Microsoft Xpad 기기를 사용하여 구현하고 실험한다.

3.2 표시 지연(Presentation Latency)

1) 표시 지연의 정의

클라우드 게임 서비스는 많은 기업과 대중 의 큰 관심을 받고 있으나, 고품질의 서비스를 제공하기 위한 플랫폼이나 시스템 구성에 대해서는 아직 잘 알려져 있지 않다[6]. 특히, 클라우드 서버에서 처리되는 게임 서비스의 스트리밍은 필연적으로 지연이 발생할 수밖에 없고, 이는 클라우드 컴퓨팅 기반 서비스의 품질을 떨어뜨리는 데 크게 기여한다.

표시지연은 전송지연, 처리지연 등 여러 가지 요소가 복합적으로 작용하여 발생하는데, 표시 지연은 클라우드 스트리밍 게임 서비스 서버에서의 소프트웨어 표시(presentation) 클락과 클라이언트 단에서의 표시 클락 사이의 차이로 정의될 수 있다.

표시지연은 크게 서버 단에서는 (인코딩 처리 시간 + 네트워크 전송 처리 시간)으로, 클라이언트 사이드에서는 (네트워크 수신 처리 시간 + 디코딩 처리 시간)으로 계산된다. 여기에 추가적으로 네트워크 전송 지연 시간이 포함된다. 즉, 서버 사이드 지연, 클라이언트 사이드, 네트워크 전송 지연 각각의 시간을 합산한 시간이 표시지연으로 나타난다.

Chen[6]은 클라우드 게임 시스템에서 발생하는 지연을 응답 지연(response delay)이라고 명명하며, 응답 지연은 네트워크 지연(network delay), 처리 지연(processing delay), 재생 지연(playout delay)의 합으로 정의하였다. 네트워크 지연은 네트워크 왕복 시간을 의미하며, 처리 지연은 서버가 사용자의 명령을 받은 후 대응하는 응답을 하는 데 소요되는 시간을 의미한다. 그리고, 재생 지연은 클라이언트가 인코딩된 프레임을 받은 시점과 화면에 디코딩된 프레임을 표시하는 데 걸리는 시간을 의미한다[6]. 본 연구에서의 인코딩 처리 시간, 디코딩 처리 시간, 네트워크 수신 처리 시간, 네트워크 전송 처리 시간의 합은 Chen[6]의 연구에서 정의한 처리 지연과 재생 지연의 합에 대응하고, 네트워크 전송 지연 시간은 네트워크 지연에 대응한다.

앞서 언급한 바와 같이 게임에서의 용인 가능한 수준의 표시 지연은 그 모델과 시점에 따라 상이하다. 본 연구에서는 도전적으로 표시 지연에 가장 민감한 일인칭 시점의 게임을 가정하여 표시 지연을 100ms 수준으로 감축하는 것을 목표로 한다.

2) 표시 지연의 측정 방법

표시지연을 간편하고 직접적으로 측정하기 위해서 서버의 화면과 클라이언트의 화면을 하나로 담아 촬영하는 방법을 선택했다. 이외에도 Chen[6]의 연구와 같이 특정 메뉴를 출력하는데 사용되는 핫키(hot key)를 이용하여 그 키가 눌렸을 때의 시간과 특정 메뉴가 출력되었을 때의 시간을 측정하고, 네트워크 왕복(round-trip) 시간을 측정하여 처리 지연과 재생 지연을 측정하는 방법이 있다. 이 경우 클라이언트의 화면에 특정 메뉴가 출력되는지를 확인하기 위해서 화면 일부 픽셀의 칼라 변화를 체크해야 하는 데[6], 칼라 변화를 체크하기 위한 알고리즘을 수행하는 데에서 불필요한 지연이 발생할 수가 있다.

또한 타임 스탬프의 로그를 확보하여 계산하는 방법이 있는데, 시간 로그를 확보하는 방법은 전체 지연 시간이 100ms수준으로 거의 실시간인 경우 로그를 확보하는 시점에 대한 높은 신뢰성이 요구된다. 즉, 로그를 출력하는데 많은 시간이 소요되어서는 안된다는 말이다. 또한 로그를 얻는 방식은 데이터를 손쉽게 얻고 구간별 타임 스탬프를 얻을 수 있다는 장점이 있지만 장치별 기준 시간이 일정하지

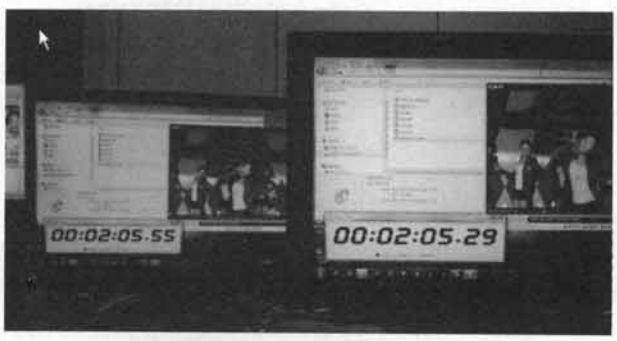


Fig. 4. Example of the proposed measurement

않을 수 있다. 개별 장치의 기준시간은 장치의 제조사 마다 달라지는 경향이 있기 때문이다.

따라서, 서버의 화면과 클라이언트의 화면을 동시에 촬영할 수 있는 환경이라면, Chen[6]의 방식보다는 첫번째 방식을 채택하는 것이 정확한 측정을 위해서 적합하다. Fig. 4는 첫번째 방식에 따른 구현의 예이며 왼쪽 화면은 서버의 화면, 오른쪽 화면은 클라이언트의 화면이다.

4. 표시지연 해결 방법

표시지연을 줄이기 위해서 서버 단에서의 접근과 클라이언트 단에서의 접근이 모두 필요하다.

서버 단에서는 인코딩 속도 개선과 네트워크 전송 개선이 필요하며, 이는 주 CPU를 이용하지 않고 별도의 하드웨어를 이용하는 것을 통해 개선이 가능하다. 그러나, 본 연구는 하드웨어적인 측면보다는 소프트웨어측면에서의 개선을 통해 표시지연을 줄이는 것에 초점을 맞추며, 하드웨어적인 측면의 해결 방법에 대해서는 향후 과제로 남겨둔다. 따라서, 서버 단에서는 원본/재생본 포맷 일치화, 미디어 타입에 따른 인코딩 자원 분배, 동기화를 위한 소프트웨어 설계를 통한 표시지연 감소를 제안한다.

클라이언트 단의 경우, 표시지연은 네트워크 수신 처리 지연과 디코딩 처리 지연으로부터 발생한다. 서버로부터 전송된 오디오/비디오 스트리밍 데이터는 패킷화 (packetized) / 암호화 (Encrypted)/ 압축(Compressed)된 형태로 전송되므로 패킷화 (packetized)/ 암호화 (Encrypted) /압축 (Compressed) 각 단계에서 표시 지연을 감소하기 위한 방법을 제안한다.

4.1 서버 단에서의 표시 지연 감소

1) 원본/재생본 포맷 일치화

원본 / 재생본 포맷 일치화(Source / Sync format equalization)는 인코딩하는 원본 영상과 디코딩하는 재생 영상의 포맷이 다를 때 서버의 인코딩 포맷을 디코딩 쪽에 일치화 시키는 것이다. 예를 들어, 클라이언트는 640x480의 영상으로 재생을 하게 되고 원본 영상은 해상도 1280x720의 스트리밍 게임이라고 했을 때 양쪽의 코덱과 해상도가 같은 포맷이 되도록 인코딩 쪽 포맷을 변경하여 일치시키는 것이다. 여기서 변경의 대상은 서버 사이드가 된다. 예컨대, 클라이언트가 640x480의 해상도를 가진 영상이라면 서버사이드 쪽의 게임 구동 자체를 인코딩 시에 포맷을 640x480이 되도록 변경한다. 만약 서버 쪽의 게임 구동이 1280x720형태라면 클라이언트가 재생할 수 있게 640x480으로 바꾸어 인코딩(트랜스코딩)하는 것이 필요하게 된다. 그렇지 않으면 클라이언트는 수신 받은 1280x720의 영상을 640x480으로 바꾸는 것이 필요하게 된다. 트랜스코딩(Transcoding)은 특정 포맷으로 인코딩된 디지털 데이터를 다른 포맷으로 인코딩된 데이터로 바꾸는 것을 의미한다. 본 연구에서는 클라우드 스트리밍 게임에서 영상의 포맷을 바꾸는 트랜스코딩을 억제하는 것을 제안한다. 트랜스코딩은 영상 프레임당 상당한 수준의 자원을 소모하게 하는데, 이는 표현 지연을 발생시키는 원인이 된다. 결국 서버사이드에서의 지연을 유발하게 되고, 최종적으로 상당한 표시지연을 발생 시키게 된다.

본 연구에서 가정하는 소비가전 기기에서는 재생하는 영상의 포맷을 서버처럼 자유롭게 변경하기에는 많은 제약이 있다. 따라서, 서버 쪽에서 인코딩하려고 하는 원본 포맷을 재생하고자 하는 포맷과 일치시키는 것이 유리하다. 실험을 통해 1920x1080의 영상을 1280x720으로 재생하는 과정에서 약 250ms의 표시지연이 발생함을 확인하였는데, 트랜스코딩이 이 표시지연을 발생시키는 주요한 원인이었다. 따라서, 본 연구에서는 트랜스코딩 지연을 제거하기 위해 서버 단에서 원본-재생본 간의 포맷 일치화를 제안한다.

2) 미디어 타입에 따른 인코딩 자원 분배

게임과 같은 실시간(real-time) 속성이 강조되는 애플리케이션의 경우, 표시지연은 매우 중요한 이슈가 된다. 그러나 스트리밍 서비스를 제공하는 클라우드 서버의 경우, 여러 종류의 미디어를 제공하는 것이 일반적이다. 따라서, 높은 성능으로 영상이 지연되지 않도록 서버의 인코딩 자원을 관리하는 것이 필요하다. 그러나 영상이 지연되지 않는 수준의 서버는 적어도 5Mbps의 네트워크 대역폭과 Intel Quad core CPU기준 약 40%가량의 CPU 자원을 소비하는 정도로 상당한 자원이 요구된다.

클라우드 서버에서 제공되는 모든 스트리밍 애플리케이션 및 모든 미디어가 게임과 같이 항상 강력한 실시간성을 요구하는 것은 아니다. 예를 들어, 동영상 스트리밍의 경우 재생 장애(Drop out)가 발생하지 않는 한 표시지연이 1-2초 정도 되어도 큰 문제가 없다. 다른 예로, 전자책(e-book)의 경우 영상의 변화가 매우 적으며 표시지연 감내도 유저 인터페이스(User Interface)를 제외하면 동영상 스트리밍보다 엄격하지 않다. 즉, 동영상 재생 이상의 표시지연도 감내할 수 있다는 것이다. 한편, 인터넷 브라우징은 동영상보다는 낮은 대역폭을 필요로 하지만 표시지연에는 비교적 민감하다. 게임 애플리케이션의 표시 지연 한계보다는 느슨하지만 다른 타입의 미디어에 비하면 표시지연에 민감한 편이다. 이는 사용자가 보다 더 즉답 즉응성이 좋은 인터넷 브라우징을 원하기 때문이다.

본 연구에서는 이러한 미디어 속성간의 차이를 Table 1과 같이 예측가능성 (predictability)과 변화가능성 (changeability)을 기준으로 분류하였다.

Table 1. Characteristics of cloud streaming media

	예측가능성	컨텐츠 변화가능성
동영상	높음	높음
전자책	높음	낮음
인터넷 브라우징	낮음	조금 낮음
게임	낮음	매우 높음

Table 1은 기본적으로 정성적인 기준에서 작성된 것으로 정량적인 엄밀함을 얻기 어렵다. 다만 상용화된 서비스에서의 경험을 바탕으로 한 실험치에 가까우며 서비스 설계자, 서비스 품질보증(quality assurance), 베타테스터들의 반응을 기준으로 정해진 것이다. 언급한대로 본 연구의 기준은 정성적인 엄밀함을 제공할 수 없지만 Sharp의 연구[10] 역시도 정량적 데이터 보다는 정성적 데이터를 활용하고 있고 이는 사용자의 반응성을 측정하기엔 개개의 편차가 크고 정성적인 부분이 중요하게 여겨지는 인지적 분야에 가깝기 때문이다. 이후 추후연구에 있어 이러한 유저의 반응과 효율성에 대해 무엇을 기준으로 하여야 정량적인 데이터를 얻을 수 있을지 고민하는 것이 필요하다고 하겠다.

동영상의 경우 컨텐츠 영상이 시시각각 변하게 되고 이는 높은 변화 가능성을 갖는다. 반면 전자책의 경우 컨텐츠 영상이 다음 페이지를 넘길 때까지 거의 변하지 않으므로 변화 가능성이 매우 낮다. 따라서 전자책에 인코딩 자원을 크게 배분하는 것은 낭비가 된다. 인터넷 브라우징의 경우 웹 페이지 하나에 머무는 시간이 긴 편으로 사용자가 다른 링크를 선택할 때까지 컨텐츠가 변하지 않는다. 그러나, 게임의 경우 동영상과 마찬가지로 영상이 시시각각 변하는 특성을 갖고 있다.

동영상의 경우 높은 변화 가능성을 갖지만 영상의 내용과 순서가 고정되어 있어서 어떤 컨텐츠가 이어질지 쉽게 예측할 수 있다. 따라서, 예측 가능성이 높으며 이는 전자책과 유사하다. 반면 게임의 경우 사용자의 반응에 따라 컨텐츠가 달라진다. 사용자가 입력기기로 게임을 조작하게 되면 게임 캐릭터가 그에 따라 움직여야 하는 등의 컨텐츠 변화가 즉각적으로 반영되어야 한하고 예측이 어렵다. 인터넷 브라우징의 경우도 사용자가 어떤 웹 페이지를 열람하느냐에 따라 컨텐츠 내용이 달라진다. 따라서, 역시 예측 가능성이 낮은 편에 속하게 된다. 그러나, 선택 가능한 웹 페이지에 대한 사전 캐싱(cashing)을 할 수 있으므로 예측이 전혀 불가능하지는 않다. 동영상, 전자책 등 다른 미디어에 비하면 예측 가능성이 높지는 않으므로, 이렇게 예측 가능성이 낮은 경우 낮은 표시 지연을 유지하는 것이 매우 중요하다. 게임의 경우 사용자의 상황에 따라 완전히 별개의 컨텐츠가 나타나게 됨으로 매우 낮은 표시 지연을 유지하는 것이 필요하며, 인터넷 브라우징 역시 사용자의 반응에 따라 별개의 컨텐츠가 나타남으로 게임만큼은 아니더라도 낮은 표시 지연을 유지하는 것이 바람직하다.

본 연구에서는 미디어의 타입에 따라서 인코딩 자원을 배분하는 것을 제안한다. 예측 가능성이 낮고 컨텐츠 변화 가능성이 매우 높은 미디어인 게임의 경우 인코딩 샘플링을 30 fps(frame per second)로 일정하게 유지한다. 한편, 사용자 반응이 예측하기 어려우나 컨텐츠 변화가 낮은 편인 인터넷 브라우징의 경우 사용자의 입력이 들어오면 일시적으로 샘플링 레이트를 30 fps으로 높이고 일정 시간(5초) 입력이 없으면 15 fps로 낮추도록 구성한다. 전자책의 경우 평소 샘플링 레이트를 5 fps로 낮추고 입력이 들어올 경우만 30fps로 높이도록 처리한다.

이를 통해 서버 사이드의 효율적인 자원관리와 클라이언트 사이드의 원활한 서비스 제공을 가능하게 할 수 있다.

3) 동기화를 위한 소프트웨어 설계

클라우드 스트리밍 애플리케이션의 현존하는 상용/실험 솔루션 대부분은 최선 노력(Best Effort) 전략을 취하며, 최대한 전송/인코딩/디코딩 오버헤드(overhead)를 줄여서 서비스한다. 그러나, 최선 노력 전략을 통해서는 표시 지연에 요구 충족을 확실하게 보장할 수 없다. 표시 지연은 네트워크 등 서버나 클라이언트 내부 외의 외적인 원인에 의해서도 커질 수 있다. 외적인 요인에 의해 지연이 발생할 경우, 이와 상관없이 서버에서는 애플리케이션 프로그램이 계속 구동될 것이고 이는 필연적으로 클라이언트 사이드에서의 표시 지연으로 나타나게 된다. 즉, 구동되는 애플리케이션과 클라이언트 단 사이의 표시 지연에 대한 요구를 만족시킬 수 없게 된다.

본 연구에서는 이와 같은 표시 지연에 대한 문제점들을 해결하는 방안으로서 클라우드 스트리밍 서비스를 목표로 하고 있는 애플리케이션에 대해 SW 설계 단계에서 양자간의 동기화를 맞출 수 있는 인터페이스를 제공하는 것을 제안한다. 즉, 현재 구동되고 있는 애플리케이션의 타임 클럭을 제공하고 클라이언트에서 현재 표시 중인 스트림에 대한 타임 클럭을 받을 수 있는 인터페이스를 제공하는 방법으로서, 애플리케이션 클럭이라는 새로운 개념에 기반한다.

애플리케이션 클럭은 기존의 IEEE 1588 프로토콜과 그 개념에서 유사한 특성을 갖는다. IEEE 1588은 네트워크를 사용하여 통신하는 분산 시스템의 노드 간에 실시간 클럭을 동기화하기 위해 고안된 프로토콜이다. IEEE 1588에서 클럭들은 멀티캐스팅된 클럭 속성 정보에 기반하여 마스터-슬레이브의 계층구조를 구성하게 되며 그 계층구조에 따라 각 슬레이브는 자신의 마스터에 동기화하는 방식으로 각 노드 간의 동기화가 이뤄진다[11].

그러나 본 연구에서 지적하는 부분은 일반적인 클럭 동기화의 문제를 다루는 것이 아니라 애플리케이션의 설계 시점에 애플리케이션 클럭 개념을 도입할 경우 표시 지연을 효과적으로 보장할 수 있다는 점이다. 본 연구에서 강조하는 점은 애플리케이션 전체의 클럭을 일치화 시키는 것이 목적으로 하지 않는다. 이하 소개할 내용처럼 필요로 하는 부분 - 영상 프레임의 타임 스템프만에 대해서만 동기화를 시도하고 이러한 동기화의 처리는 애플리케이션의 설계단계에서 적용되어야 한다는 것이다.

현재 상용화된 클라우드 서비스는 일반적으로 기존에 개발된 애플리케이션을 활용하게 된다. 하지만 이 경우 IEEE 1588 프로토콜을 사용하게 된다면 서버/클라이언트 양자간의 클럭을 일치시키도록 지나치게 부하가 걸리게 설계될 우려가 있다. 이에 설계 단계에서 본 연구가 제안하는 애플리케이션 클록 개념 도입을 제안하는 것이다.

클라우드 서버에서 구동되는 애플리케이션은 주기적으로 자신의 애플리케이션 클럭(영상 프레임 또는 타임 스탬프)을 외부에 알려주는데, 클라우드 서버는 이 정보를 받아 클라이

언트에게 주기적으로 전달하게 된다. 클라이언트는 이렇게 받은 애플리케이션 클럭을 자신이 재생하고 있는 스트림의 타임 클럭과 주기적으로 비교한다. 애플리케이션 클럭과 재생하고 있는 스트림의 타임 클럭을 비교했을 때 어는 임계치 이상 차이가 나지 않으면 정상ACK 신호를 보낸다. 만약 임계치 이상의 차이가 발생하며 정정 패킷을 서버에게 보낸다. 정정 패킷을 받은 서버는 이를 현재 구동 중인 애플리케이션에게 전달하고, 정정 패킷을 받은 애플리케이션은 정상ACK 패킷이 올 때까지 클럭을 멈추고 애플리케이션의 구동을 중단 시킨다. 애플리케이션의 구동이 중단된 사이 클라이언트에서는 스트림이 계속 재생되어 타임 클럭이 변경되고, 애플리케이션 클럭과 재생되고 있는 스트림의 타임 클럭의 차이가 임계치 이하로 줄어들게 되면 정상 ACK 신호를 서버에 보내게 된다. 클라이언트로부터 정상 ACK 신호를 받은 서버는 다시 애플리케이션을 구동하게 된다.

여기서 주요한 것은 클라이언트 - 서버의 시스템 레벨에서의 클럭이 아닌 애플리케이션 레벨에서의 영상프레임, 타임 스탬프 동기화 수준의 클럭 동기화를 의미하는것이다. 이는 상용 서비스에서 필요한 빠른 처리를 요구사항으로 하기 때문이다.

본 연구에서 제안하는 애플리케이션 클럭의 개념은 소프트웨어 설계 단계에 반영되어야 하는데, 영상 프레임 또는 타임 스탬프를 외부에 전송하고 또 외부에서 타임 스탬프를 수신할 수 있는 모듈이 설계되어야 한다. 이를 통해, 기존 최선 노력 방식에 의존하던 것에 비해 좀 더 효율적으로 표시지연 보장(Presentation Latency Guarantee)을 제공할 수 있게 된다.

4.2 클라이언트 단에서의 표시 지연 절감

1) 패킷화 과정에서의 명령어 수 절감

패킷화 과정에서의 네트워크 수신 처리 지연을 줄이기 위해 본 연구에서는 코드 레벨에서의 명령어(Instruction) 수 감소를 제안한다.

일련의 패킷으로 들어온 스트리밍 데이터는 재조합이 필요한데, 이 재조합 과정에서 명령어 수가 많아지면 패킷 당 처리지연이 발생하게 된다. 특히, 무거운 명령에 속하는 memcpy( ) 함수의 경우 그 수를 제한하는 것이 필요하다. 본 연구에서는 memcpy( )의 호출 횟수를 프레임 당 한 번으로 제한하여 약 15ms 가량의 지연감소를 얻을 수 있었다.

좀 더 상세히 설명하면 아래와 같다. 본 연구에서는 변형된 실시간 전송 프로토콜을 사용하여 스트림을 전송하였다. 실시간 전송 프로토콜에서 패킷의 순서 정렬 과정이 필요하고, 순서를 정렬 한 패킷을 프레임별로 디코더에 공급하였다. 또한 개별 단계마다 memcpy() 명령을 사용하여 최소 프레임당 세 번 이상의 memcpy()를 사용하여야 했던 것을 하나의 메모리 버퍼 상에 세 단계를 함께 전개하고 최종적으로 하드웨어 디코더에 스트림을 공급할 때 한 번만 memcpy()를 사용하도록 수정하였다. 이를 슈도(pseudo) 코드로 간략하게 표현하면 아래와 같다.

```
Sorting packets( packetbuf){
    sorting(packetbuf);
memcpy(bufSort,packetbuf);  }
PacketsToBuffer(bufSort) {
    deployonFramebuff(bufSort);
    memcpy(bufSort,decodeBuf);   }
SupplyToDecoder(decodeBuf){
memcpy(decodeBuf,Decoder)}
Sorting(buf);
deployonFrambuf(buf);
supplytoDecoder(buf)
{
Memcpy(decodeBuf,Decoder);
}
```

2) 암호화 과정에서 가벼운 알고리즘 사용

암호화 과정에서는 가능한 가벼운 알고리즘의 사용이 유리한데, 본 연구에서는 대칭키 방식의 암호화 알고리즘을 제안한다. 대칭키 방식의 암호화 방식은 암/복호화가 간단하며, 이는 암호화로 인한 지연 시간의 감소로 이어진다. 한편, 대칭키 방식의 보안강도를 강화하기 위해 초기 세션 교환 시 공개키 방식 암호화를 위해 사용한 대칭키를 전달하고 주기적으로 키 값을 변경하도록 하였다.

좀 더 상세하게 설명하면, 클라우드 스트리밍 게임은 매우 높은 수준의 실시간적 성능을 요구한다. 특히, 게임이라는 미디어 특성에만 국한한다면 사적인 정보나 지적 재산의 유출을 걱정해야 할 필요성이 상대적으로 적은 관계로 강력한 보안강도를 요구하지 않는 편이다. 본 연구에서는 환자 암호 형태(Cipher) - 대칭키 암호로 스트림을 보호하였다.

그러나, 아무리 가벼운 대칭키 방식이라고 할지라도 표시지연을 발생시킬 수 있는 요소가 될 수 있다. 따라서, 본 연구에서는 일부 패킷 즉 그림 재생의 핵심이 되는 I frame 패킷만 암호화 하는 방식으로 지연을 최소화 시켰다.

3) 압축해제 과정에서 하드웨어 디코딩

압축해제 단계에서 일반적으로 소비가전 기기에서 하드웨어 디코딩이 이루어진다. 그러나, 디코더는 일반적으로 칩 레벨에서의 프레임 버퍼링을 하며 세 프레임을 저장할 수 있는 버퍼를 갖고 있다. 세 프레임 평균 30 fps(frame per second)를 가정했을 때 80-90ms의 지연을 발생시키며, 이는 상당히 큰 크기의 지연이라고 말할 수 있다. 따라서, 디지털 신호 프로세서에서 버퍼링을 전혀 하지 않도록 하는 모드의 설정, 일명 제로 버퍼링(Zero buffering) 모드 혹은 제로 지연(Zero Delay) 모드 등으로 불리는 디코더 설정을 통해 이와 같은 지연 현상을 피할 수 있다.

5. 성능 평가

본 장에서는 원활한 클라우드 스트리밍 서비스를 제공하기 위해 4.1과 4.2에서 제안한 방식들에 대한 검증한다. 이하

내용에서는 실제 상용화 서비스까지 이어졌던 클라우드 스트리밍 서비스를 대상으로 한 실험 및 성능평가의 결과가 소개된다. 여기서는 앞서 언급된 여섯 가지 제안 방안을 통해 실질적으로 얻어진 결과가 기술된다.

우선적으로 본 실험에서 사용된 클라이언트 기기와 서버 기기에 대해 정리한다. 클라이언트 기기는 실제 상용화된 블루레이 디스크 플레이어, DVD 플레이어, 스마트 박스, 셋톱박스, 개인용 비디오 레코더(Personal Video Recorder)에 사용되는 소비가전용 임베디드 칩셋을 탑재한 스마트 박스를 실험용으로 사용하였다. 해당 기기는 미화 79달러 이하의 상용 기기이며, 본 연구가 의도하는 씬 클라이언트 소비가전 기기에 부합하는 기기를 선택한 것이다. Table 2는 실험에 사용된 씬 클라이언트들의 특성을 요약한 것이다.

Table 2. Specification of the thin client for evaluation

주 CPU 아키텍처	RISC 계열 MIPS 아키텍처
처리속도	500 DMIPS(Dhrystone Million Instructions Per Second)
RAM	384 MB Dram
ROM	512 MB NAND Flash
하드웨어 비디오 디코더	H.264, MPEG4, MPEG2, MPEG TS … etc
하드웨어 오디오 디코더	AAC, AC3, PCM … etc
운영체제	Linux Kernel 2.6.28 optimized for Embedded System
외부 입력포트	1 USB 포트, USB 허브로 확장 가능 (본 연구에서는 게임 애플리케이션 입력 포트로 사용됨)
ETC.	BD/DVD 지원, STB/PVR 지원

한편 실험에서 사용된 서버 기기는 실제 상용 서비스에 사용되는 서버 기기가 아닌 범용 개인 컴퓨터를 사용한 서버 기기를 사용하였다. 실제 상용 서버 기기는 클라우드 게임만을 제공하는 것이 아니라 범용적으로 다른 서비스들을 제공하고 있고 이는 본 연구에서 고안된 방식을 시험 하기에 적합하지 않다. 실험의 통제값과 실험값을 컨트롤하여 가설을 검증하는데 바람직하지 않기 때문이다. 따라서, 본 연구에서는 개념 증명 단계(Proof of Concept)에서 사용된 프로토타입 서버 기기를 통해 연구가 진행되었다. 그 사양은 Table 3과 같다.

Table 3. Specification of the prototype server for the evaluation

주 CPU 아키텍처	Intel core i5 (x86)
처리속도	3.2Ghz Quad core
RAM	4 GB Dram
그래픽 카드	H.264, MPEG4, MPEG2, MPEG TS … etc
운영체제	Windows 7

5.1 원본/재생본 포맷 일치화

앞서 언급했듯이 이 방법을 통해 포맷을 일치화 시킬 경우 서버 또는 클라이언트 어느 한 쪽이 감당해야 하는 트랜스코딩 시간을 줄일 수 있게 된다. 본 연구에서는 서버 쪽에서 이러한 트랜스코딩을 처리하였다. 실제 소비가전 기기를 씬 클라이언트로 구현하는 경우, 트랜스코딩은 상당한 부하를 주는 작업으로서 클라이언트에서 이뤄지기가 쉽지 않다. 본 연구에서는 원본/재생본 포맷 일치화 방식을 사용하였을 때와 그렇지 않았을 때의 표시 지연을 측정하여 그 효과를 확인하였다.

실험은 모든 조건을 일치시키고, 원본/재생본 포맷 일치화를 적용한 경우와 그렇지 않은 경우를 비교하였다. 즉, 1280x720의 영상을 1280x720 그대로 인코딩하는 경우가 곧 원본/재생본 포맷이 일치되는 경우이다. 반면 1280x720을 720x480과 같이 다른 해상도로 트랜스코딩한 경우는 해당 솔루션을 적용하지 않은 것이 된다.

본 연구에서 1280x720, 1024x768, 800x600등 다양한 해상도에서의 실험을 진행하였다. 높은 해상도일 경우 높은 대역폭으로 인해 표시 지연을 유발할 수 있으므로 해상도가 끼치는 영향을 파악하기 위해 다양한 해상도에 대해 시험을 진행하였다. Table 4는 그 결과이다.

Table 4. Presentation latencies according to the resolution of the stream

Condition	평균 표시 지연	Deviation
1280x720 -> 800x600 변환	438.23	23.20
1024x768 -> 800x600 변환	434.38	23.33

```
S = 23.26  P-Value = 0.677
Level   N    Mean  StDev  ----+---------+---------+---------+-----
C      13  438.23  23.20      (-----------------*-----------------)
D      13  434.38  23.33  (-----------------*-----------------)
                          ----+---------+---------+---------+-----
                            424.0     432.0     440.0     448.0
```

위의 데이터를 보면 똑같이 원본/재생본 포맷 일치화를 하지 않았지만 해상도가 다른 두 케이스, 1280x720 -> 800x600 변환을 한 케이스 C와 1024x768 -> 800x600 변환을 한 케이스 D의 경우 분산이 23정도로 거의 일치하며 일원 분산 분석(One-way ANOVA)결과상 P - value가 0.677로써 0.05 이상으로 통계적으로는 같은 값임을 판단할 수 있다. 이 결과를 통해 해상도 자체는 표시 지연에 대한 인자가 되지 못함을 알 수 있다.

Table 5에 정리된 통계분석 결과는 원본/재생본 포맷 일치화를 한 케이스B와 하지 않은 케이스A의 결과를 나타낸다. 케이스B의 경우 평균 표시지연이 113.87 msec으로 줄어들고 표준편차도 15.01로 줄어든 것을 확실히 확인할 수 있다. 또한, 일원 분산 분석 검증을 통해서 P-Value 0.000으로 유의수준 0.05보다 적음으로써 두 통계 A와 B는 확실히 다른 값임을 확인할 수 있다.

Table 5. The comparison between case A and case B

Condition	평균 표시 지연	Deviation
원본/재생본 포맷 일치화 적용 전	438.70	24.87
원본/재생본 포맷 일치화 적용 후	113.87	15.01

```
S = 20.54  P-Value 0.000
Level   N    Mean  StDev  ---------+---------+---------+---------+
A      30  438.70  24.87                                    (*)
B      30  113.87  15.01  *)
                           ---------+---------+---------+---------+
                                  200       300       400       500
```

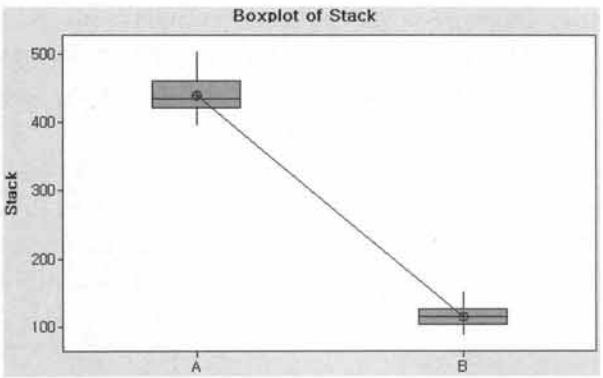


Fig. 5. Box plot of case A and case B

이와 같이 원본/재생본 포맷 일치화를 적용하는 경우 지연이 크게 줄어드는 것을 확인 할 수 있다. 또한, 일원 분산 분석(one way ANOVA)을 통해 해상도에 관계없이 균일하게 지연이 줄어들었음을 확인 할 수 있다.

결과적으로 해상도 자체보다는 포맷 일치화를 하지 않음으로써 취해야만 하는 트랜스코딩이 표시 지연을 유발시키는 요인이 되는 것으로 파악된다. 예를 들어, 1280x720 해상도나 800x600해상도에 상관없이 같은 크기의 지연이 발생했기 때문이다. 실험결과 원본/재생본 포맷 일치화를 적용하지 않은 경우 포맷을 전환하는 트랜스코딩 시간을 필요로 하며, 이로 인해 발생하는 평균 지연시간은 440ms정도를 보인다. 따라서, 원본/재생본 포맷 일치화 솔루션이 유용하다는 것을 확인할 수 있다.

5.2 미디어에 따른 인코딩 자원 배분

미디어에 따라 인코딩 자원의 배분을 달리하는 솔루션을 적용하지 않은 서버의 평균 CPU 점유율 및 편차는 Table 6과 같다. 아래의 결과는 미디어의 속성에 따라 자원의 배분을 변경하여야 좀 더 효율적으로 자원을 활용할 수 있다는 주장을 뒷받침한다.

가장 높은 점유율을 보이는 케이스 D게임의 경우 평균 CPU 사용률도 높고 그 분산도 넓게 퍼져있다. 특히, 데이터의 분포는 사용자의 조작에 따라 컨텐츠의 내용이 달라지는 게임 컨텐츠에서의 높은 변화 가능성을 반영한다. 한편, 케이스 C 인터넷 브라우징의 경우 평균 점유율은 높지 않으나

Table 6. The average CPU usage according to a media type

Condition	Average CPU Usage	Deviation
동영상(케이스 A)	43.9	5.768
e-book(케이스 B)	19.	3.433
Internet Browsing(케이스 C)	26.5	7.798
Game(케이스 D)	63.1	8.601

```
S = 6.705   P-Value 0.000
Level   N    Mean  StDev  ---------+---------+---------+---------+
A      30  43.900  5.768                      (*-)
B      30  19.867  3.433  (-*)
C      30  26.500  7.798       (-*)
D      30  63.133  8.601                              (-*-)
                           ---------+---------+---------+---------+
                                   30        45        60        75
```

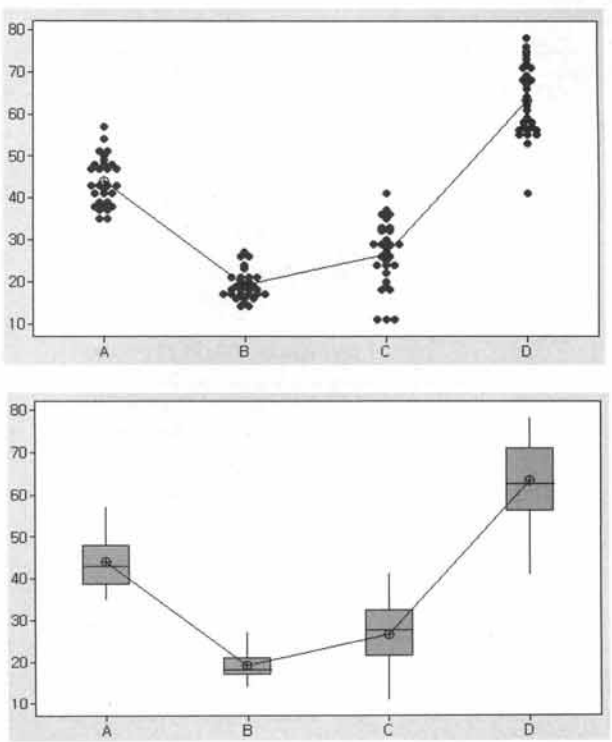


Fig. 6. The value plot and the box plot for each case

역시 분산이 상대적으로 넓게 퍼져 있는 편이다. 이 역시 사용자 인터페이스에 따라 컨텐츠가 변화하는 양상을 반영한다고 보인다. 또한, 동영상 즉 케이스 A의 경우 점유율이 높은 편이지만 게임에 비해서는 상대적으로 낮은 편이다. 이는 대역폭은 비슷하지만 변화 가능성이 적은, 즉 일정한 스트리밍을 전송하는 동영상 컨텐츠의 특성을 반영한다. 한편, 컨텐츠가 매우 정적이고 사용자의 반응도 컨텐츠에 영향을 끼치지 못하는 케이스B의 전자책과 같은 컨텐츠는 낮은 평균 점유율과 상대적으로 좁게 펼쳐진 분산을 보인다.

위의 내용대로 일원 분산 분석을 통해 P-Value가 0값을 갖는 것이 보여주듯이 각 조건에 따른 결과가 다르다는 것을 쉽게 관찰할 수 있다. 따라서, 미디어에 따라 인코딩 방식을 변경하는 본 논문의 제안 방식이 효과적이다는 것을 알 수 있다.

각각의 미디어 타입 별로 인코딩 방식을 바꾸는 것으로 CPU 점유율로 대표되는 리소스 사용량을 통제할 수 있고, 이는 미디어 타입에 상관없이 인코딩하는 방식에 비해 효율적이라는 것을 알 수 있다.

본 실험은 실제 상용화 서버에서가 아니라 프로토타입 서버에서 시험한 것으로 실제와는 다소 차이가 있을 수 있다. 예를 들어, 단지CPU 점유율에 의거한 한 소비 자원의 정량적 측정은 상당히 큰 오차를 내포한다. 그러나, 측정된 값은 통계적으로 의미가 있으며, 본 연구가 제안하는 미디어 타입에 따른 인코딩 방식 변경이 서버 자원을 효율적으로 활용할 수 있도록 하는 한 방법이 될 수 있음을 보여준다.

5.3 동기화를 위한 소프트웨어 구현

본 연구에서는 소프트웨어 설계단계에서 양단 간의 동기화를 맞출 수 있는 인터페이스의 제공을 제안하였다. 현재 상용화된 스트리밍 게임 서비스 중에서 아직 표시 지연에 대한 성능 보장을 할 수 있는 서비스는 없다. 현재 상용화된 서비스들이 충분한 대역폭과 네트워크의 안정성을 전제로 서비스하고 있으므로 표시 지연에 대한 문제가 크게 대두된 적이 없으나, 언제나 발생할 수 있는 가능성은 내포되어 있는 상황이다.

따라서, 본 실험에서는 네트워크의 안정성을 의도적으로 떨어뜨린 상태에서 본 연구에서 제안하는 애플리케이션 클럭 솔루션을 적용한 쪽과 적용하지 않은 쪽을 비교함으로써 본 솔루션이 유효한지를 평가하였다.

실험 군은 애플리케이션 클럭 솔루션을 적용하였고 대조군은 애플리케이션 클럭 솔루션을 적용하지 않고 기존의 최선 노력 전략을 취했다. 그리고 2-3Mbps의 대역폭을 요구하는 게임 애플리케이션 서비스를 양쪽에 모두 구동하고 해당 조건에서 서버의 클락과 클라이언트의 클락이 얼마나 벌어지는지를 테스트 하였다. 네트워크 상태는 똑같이 의도적으로 혼잡화시켜 약 0.5Mpbs의 대역폭만이 동작하도록 만들었다. 실험 군은 일정값 이상 서버 - 클라이언트 간 클럭이 벌어지게 되면 서버가 현재 구동중인 애플리케이션을 중단하고 클라이언트와의 동기를 기다리도록 구현하였다.

Fig. 7은 실험 군이 일정 시간이 지나도 서버-클라이언트 간 표시지연은 어느 정도 발생하다가 다시 동기를 맞추는 동작이 반복되는 현상을 보여준다. 반면, 대조 군의 경우 클라이언트의 상황을 인지하지 못하고 최선 노력 전략을 취하는 서버는 표시 지연이 지속적으로 확대되어가는 현상을 보인다.

위 실험의 경우 애플리케이션 클럭의 편차를 250ms로 제한하도록 설정하여 진행하였다. 그리고 애플리케이션이 구동된 지 10초가 경과한 지점부터 네트워크 혼잡화 프로그램을 구동하였다. 위에서 애플리케이션 클럭을 적용한 실험군의 경우 250ms를 기점으로 표시지연이 통제되는 모습을 보여준다. 위 실험은 의도적으로 네트워크를 혼잡화한 경우로 대조 군의 경우 네트워크가 혼잡화된 이후 표시 지연이 급격히 늘어나는 것을 보여준다.

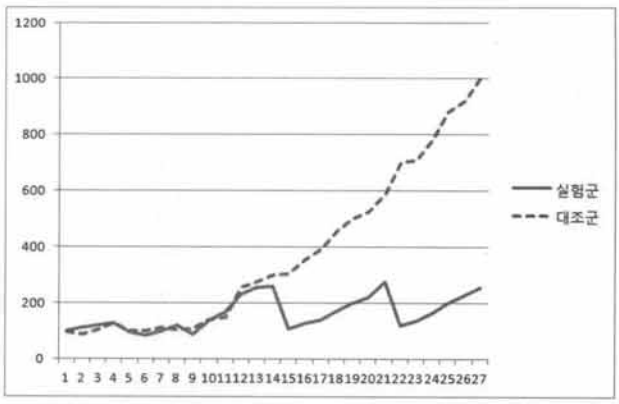


Fig. 7. The comparison between with and without application clock

본 실험을 통해 설계 단계에서부터 적용된 애플리케이션 클럭이 있을 경우 예기치 못한 네트워크 혼잡이나 단선 등이 있을 경우에도 클라우드 서버와 클라이언트 간 표시 지연을 통제할 수 있음을 알 수 있다.

5.4 명령어 수 절감 및 가벼운 암호화 알고리즘 사용

본 연구에서는 상대적으로 무거운 명령에 속하는 memcpy() 함수의 경우 그 수를 줄임으로써 처리 시간을 단축하고자 한다. 이를 위해 memcpy( )를 프레임 당 한 번으로 호출을 제한하였고 초당 30 프레임을 기준으로 서비스를 구현하였다. 만약 그 이상의 프레임을 사용하는 경우라면 그 단축의 효과는 더욱 확대됨을 유추할 수 있다.

본 연구에서는 memcpy()를 프레임 당 한 번으로 호출을 제한하였고, 이를 통해 약 15ms 가량의 지연감소를 얻을 수 있었다. 그에 대한 실험값은 Table 7과 같고 일원 분산분석(one-way ANOVA)을 통해 이러한 시도가 효과적임을 확인할 수 있다.

실제 여러 방식으로 구현 가능하며, 위에서 제시한 방법은 특정 목표 및 서비스에 적용된 한가지 방법이다. 다만, 프레임 당 처리되는 명령어의 개수를 줄일수록 서비스의 표시 지연을 줄일 수 있다는 것은 알 수 있다.

케이스 B는 명령어 수 절감을 적용한 경우이며 케이스 A는 적용하지 않은 경우이다. 통계적으로 유의하게 값의 분포가 다르고 평균 역시 다름을 볼 수 있다. 또한, 케이스 B

Table 7. The comparison between without reducing and reducing instructions

	평균 표시지연	Deviation
Instruction 단축 적용	105.6ms	11.80
비적용	120.8ms	19.36

```
S = 16.03   P-Value 0.001
Level   N    Mean  StDev  ---------+---------+---------+---------+-
A      30  120.83  19.36                        (--------*--------)
B      30  105.63  11.80  (--------*--------)
                          ---------+---------+---------+---------+-
                                 105.0     112.0     119.0     126.0
Pooled StDev = 16.03
```

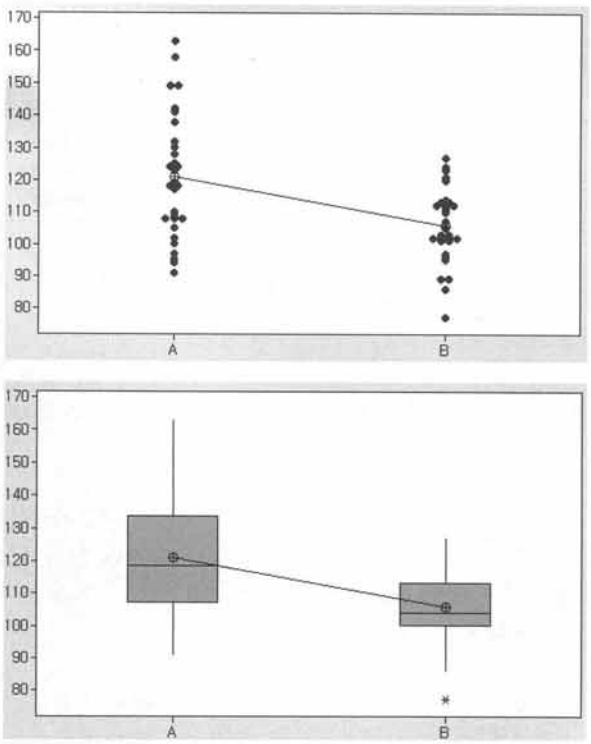


Fig. 8. The value plot and the box plot for each case

의 경우 표준편차 역시 약간 감소 (19.36→11.80)하는 현상도 나타나는데 이는 처리하는 명령어 수가 감소함에 따라 처리 시간의 편차가 줄어드는 것에 기인하는 것이다. 이와 같이 명령어 수 절감을 통해 평균 약 15ms 정도의 표시 지연을 감소시키는 효과를 얻었다.

또한 암호화 과정에서는 대칭키 방식의 암호화 알고리즘을 사용하였지만, 가벼운 대칭키 방식이라고 할지라도 표시 지연을 유발할 수 있으므로, 본 연구에서는 일부 패킷만 암호화 하는 방식으로 지연을 최소화 시켰다.

가벼운 암호화 알고리즘이 지연 감소에 미치는 영향은 명령어 수 절감으로 얻어지는 감소에 비해 현저히 적어 성능 평가에서의 자세한 분석은 생략한다.

5.5 압축해제 하드웨어 디코딩 및 버퍼링 제거

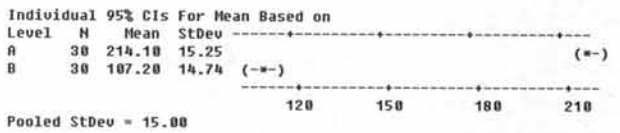
압축해제 단계에서 일반적으로 소비가전 기기에서는 하드웨어 디코딩이 이루어지며, 디코더는 보통 칩 레벨에서 버퍼링을 포함하고 있다. 이러한 버퍼링은 보통 DSP의 소프트웨어 제어를 통해 적용하거나 해제가 가능하다. 다만, DSP 공급자가 이에 대한 지원을 해줘야 하는 경우가 보통이다.

본 연구에서는 버퍼링에 의한 지연을 피할 수 있도록 제로 버퍼링 모드를 적용함으로써 표시지연을 감소 시킨다. 평균 90~110ms의 표시 지연을 감소시킬 수 있고 그에 대한 통계는 Table 8과 같다.

일원 분산 분석을 통해 솔루션을 적용한 경우(케이스 B)에 적용하지 않은 경우(케이스 A) 대비 표시지연 감소가 있음을 확인할 수 있다.

Table 8. The comparison between normal mode and zero-buffering mode

	평균 표시지연	Deviation
디코딩 방식 변경, 버퍼링 제거	107.2	14.74
비적용	214.1	15.25



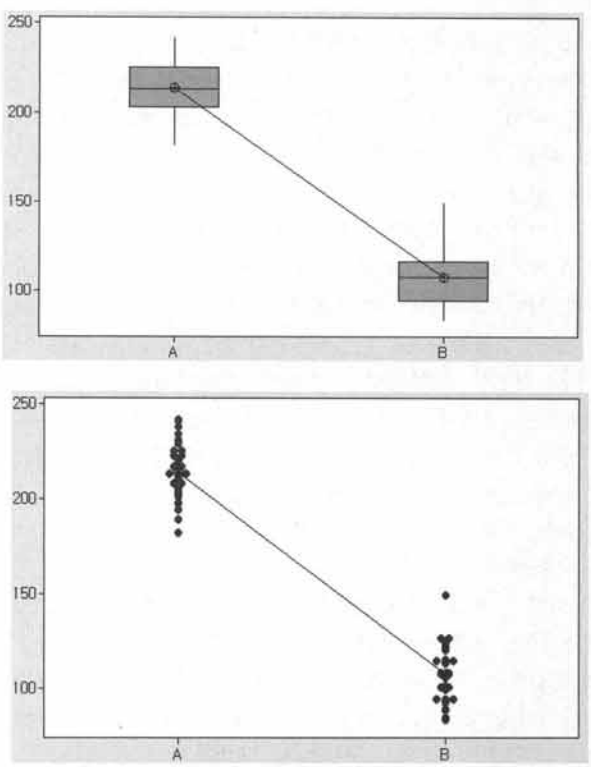


Fig. 9. The value plot and the box plot for each case

6. 결 론

본 연구에서는 클라우드 컴퓨팅의 하위 분류인 클라우드 스트리밍 게임 서비스를 구축하는데 있어서 발생하는 이슈와 그에 대한 해결 방안을 소개하였다. 특히, 저렴한 비용을 갖지만 매우 제한적인 성능을 보이는 소비가전 기기에 제공되는 스트리밍 게임 서비스를 대상으로하여 여러 조건에서 발생할 수 있는 문제점을 소개하고 그에 대한 해결방법을 제시하였다.

클라우드 시스템을 이용하여 스트리밍 게임 서비스를 제공하는 것은 많은 유리한 점도 있지만, 특히 표시 지연(Presentation latency)이 발생한다는 문제점이 있다. 게임은 사용자의 입력에 즉각적으로 반응해야 하는 특성을 갖는 서비스이므로 표시 지연의 문제는 다른 이슈에 비해 중요한 문제이다. 본 연구에서는 각 서버 단과 클라이언트 단에서

의 표시 지연 감소 방법을 제안하였다. 서버 단에서는 서버와 클라이언트의 영상포맷 일치화, 미디어 타입에 따른 인코딩 포맷 변경, 주기적인 클럭 전송을 위한 소프트웨어 설계를 해결 방안으로 제안하였고, 클라이언트 단에서는 패킷화 과정에서 명령어 수 절감, 암호화 과정에서 가벼운 알고리즘 사용, 압축해제 과정에서 하드웨어 디코딩 개선을 해결 방안으로 제안하였다.

본 연구에서는 실제 상용화 수준까지 개발된 게임 서비스를 바탕으로 위의 개선 안을 적용 검증 하였다. 실험 결과가 제안된 방안들은 상당한 개선 값들을 허락하며 매우 효과적임을 확인해 주었다. 특히 수백 ms 정도의 상당한 표시 지연 감소를 통해 허용 가능한 100ms수준까지 표시 지연 감소가 가능함을 확인하였다.

실험군과 대조군을 통한 개별 실험결과의 내용과 더불어 개별 시도의 합을 소개하여야 하나 네트워크 환경, 장비의 벤더별로 달라지는 효과 등등의 다수 팩터에 대해 본 논문 내에서 모두 소개하기에는 부족한 점이 있다. 이는 이후 연구 주제로 돌리는 것이 바람직하다고 보며, 개별 시도를 통해 허용 가능한 수준까지 표시 지연의 감소가 가능하다는 것을 확인하였다는 점에 본 연구가 의미를 갖는다.

본 연구의 기준은 개별의 사용자들이 하나의 게임을 홀로 즐기는, 즉 싱글 플레이(single play)를 기준으로 하고 있다. 추후 연구에 있어서 복수의 사용자들이 함께 게임을 즐기는 멀티 플레이(multi play)를 기준으로 한 경우 새로운 서비스 제공 알고리즘/전략을 필요로 할 것으로 보인다. 특히, 서로 논리적/물리적으로 먼 거리에 있는 사용자들이 동시에 하나의 게임을 즐길 경우 표시 지연 최소화와 서버 및 복수 클라이언트 간 클럭 일치화에 대한 고려가 필요하다. 또한, 논리적/물리적으로 다른 위치에 있는 사용자 간의 그루핑(grouping)도 중요한 주제가 될 수 있을 것이다. 이와 더불어 원격 스트리밍 게임에 있어 멀티 플레이를 고려한 서비스 전략을 다룰 필요가 있겠다.

참 고 문 헌

[1] MOTAHARI-NEZHAD, Hamid R.; STEPHENSON, Bryan; SINGHAL, Sharad. Outsourcing business to cloud computing services: Opportunities and challenges. LABs of HP, 2009.

[2] OJALA, Arto; TYRVAINEN, Pasi. Developing cloud business models: A case study on cloud gaming. Software, IEEE, 2011, 28.4: 42-47.

[3] Jun-Hyung Lee, Eui-Nam Huh. An Efficient Dynamic Resource Allocation Scheme for Thin-Client Mobile in Cloud Environment. IPS Transactions on Software and Data Engineering, Vol.19A, No.3, pp.161-168, Jun., 2012.

[4] MELL, Peter; GRANCE, Timothy. The NIST definition of cloud computing (draft). NIST special publication, 2011, 800: 145.

[5] GONG, Chunye, et al. The characteristics of cloud computing. In: Parallel Processing Workshops (ICPPW), 2010 39th International Conference on. IEEE, 2010. pp.275-279.

[6] CHEN, Kuan-Ta, et al. Measuring the latency of cloud gaming systems. In: Proceedings of the 19th ACM international conference on Multimedia. ACM, 2011. pp.1269-1272.

[7] CLAYPOOL, Mark; CLAYPOOL, Kajal. Latency and player actions in online games. Communications of the ACM, 2006, 49.11: 40-45.

[8] Wikipedia, First-person shooter [Internet], http://en.wikipedia.org/ wiki/First-person_shooter

[9] Wikipedia, Third-person shooter [Internet], http://en.wikipedia.org/wiki/Third-person_shooter

[10] SHARP, Ron. Latency in Cloud Based Interactive Streaming Content. Bell Labs Technical Journal, 2012, 17.2: 67-80.

[11] EIDSON, John; LEE, Kang. IEEE 1588 standard for a precision clock synchronization protocol for networked measurement and control systems. In:Sensors for Industry Conference, 2002. 2nd ISA/IEEE. IEEE, 2002. pp.98-105.

[12] ARMBRUST, Michael, et al. A view of cloud computing. Communications of the ACM, 2010, 53.4: 50-58.

[13] Jae-Jin Lee, Junseok Oh, Bong Gyou Lee. Significant Factors for Building Enterprise Mobile Cloud. KIPS Transactions on Software and Data Engineering, Vol.18D, No.6, pp.481-492, Dec., 2011.

강 승 수

e-mail : ss.pat.kang@samsung.com

2005년 국민대학교 기계자동차공학부 (학사)

2011년 성균관대학교 전자전기컴퓨터공학과(석사과정수료)

2007년~현 재 삼성전자 선임연구원

관심분야 : HCI, UI/UX 등

고 현

e-mail : adrian.ko@lge.com

2005년 성균관대학교 전자전기컴퓨터공학부(학사)

2007년 성균관대학교 전자전기컴퓨터공학부(석사)

2009년~현 재 엘지전자 주임연구원

관심분야 : 클라우드 컴퓨팅, 디지털 방송, 분산 처리 등

윤 희 용

e-mail : youn@ece.skku.ac.kr

1977년 서울대학교 전기공학과(학사)

1979년 서울대학교 전기공학과(석사)

1988년 Univ. of Massachusetts at Amherst, 컴퓨터공학과(박사)

1988년~1991년 Univ. of North Texas, 조교수

1991년~2000년 Univ. of Texas at Arlington, 부교수

2000년~현 재 성균관대학교 정보통신대학 교수, 유비쿼터스 컴퓨팅기술연구소 소장

관심분야 : 이동컴퓨팅, 분산 처리, RFID/USN