The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 25, No. 3, pp.143-150, Jun. 30, 2025. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2025.25.3.143
JIIBC 2025-3-18

컴퓨터 스크린 실시간 전송에 대한 실험적 성능 평가

Experimental Performance Evaluation for Real time Transmission of Computer Screens

전아린*, 이동건, 김은비**, 윤단비**, 최재현**, 황기태*****

Ahrin Jun*, Donggeon Lee**, Eunbi Kim**, Danbi Yoon**,
Jaehyun Choi**, Kitae Hwang***

요 약 오늘날 많은 회의들은 발표자가 돌아가면서 디스플레이에 자신의 노트북을 연결하는 식으로 이루어진다. 이 방식은 번거롭고 회의의 흐름을 끊는 불편함이 있다. 우리는 이런 문제를 해소할 하나의 방법으로, 여러 참여자들의 컴퓨터들을 1개의 서버 컴퓨터에 무선 인터넷으로 연결하고, 이 서버 컴퓨터가 참여자들의 컴퓨터 스크린을 실시간으로 받아 자신의 디스플레이에 출력하는 방법을 고안하였다. 본 논문은 이 시스템을 구현하기에 앞서, 서버 컴퓨터에 적합한 컴퓨터의 성능이 어느 정도여야 하는지 알아보기 위한 사전 연구로 성능 평가를 실시하였다. 우리는 클라이언트부터 실시간으로 스크린 이미지를 VP8 비디오 포맷으로 수신하여 서버의 디스플레이에 출력하는 테스트 시스템을 구축하고, 젯슨 나노와 윈도우 노트북을 비롯한 6종류의 컴퓨터들을 번갈아 서버로 사용하여 다양한 실험을 수행하였다. 발표 형태는 정적 스크린 타입과 동적 스크린 타입으로 나누고 발표 시간은 30분으로 하였다. 서버의 적절한 성능 기준으로 네트워크 지연 시간과 비디오 디코딩 시간의 합이 100ms 이하여야 하고 서버의 CPU 활용률이 30% 이하라는 조건을 임의로 세우고 성능을 측정한 결과, 서버로 적절한 사양은 8GB 이상의 메모리를 가지고 Core i7 수준의 CPU를 가진 노트북 수준으로 평가되었다.

주요어 : 실시간, 컴퓨터 스크린, 스크린 공유, 비디오 코덱, 성능 평가

Abstract Many meetings today are conducted in a way that each presenter takes turns connecting his or her laptop to a single display. This is cumbersome and interrupts the flow of the meeting. To solve this problem, we devised a method in which participants' computers are wirelessly connected to a single server computer, which receives their screens in real time and outputs them on its own display. This paper conducted a performance evaluation as a preliminary study to determine what level of computer performance is required to be suitable for a server computer before implementing this system. We built a test system that receives screen images in real time from a client in VP8 video format and outputs them to the server's display, and performed various experiments using six computers, including Jetson Nano and Windows laptops, alternately as servers. he presentation format was divided into static screen type and dynamic screen type, and the presentation time was 30 minutes. As a result of measuring performance by arbitrarily setting the conditions that the sum of network delay time and video decoding time should be less than 100ms and the CPU utilization rate of the server should be less than 30% as the appropriate performance criteria for the server, it was evaluated that the appropriate specifications for the server are the performance level of a laptop with a Core i7 level CPU and more than 8GB of memory.

Keywords : Real time, Computer Screen, Screen Sharing, Video Codec

*정회원, 한성대학교 컴퓨터공학부 학사 (제1저자)
**정회원, 한성대학교 컴퓨터공학부 학사 (참여저자)
***종신회원, 한성대학교 컴퓨터공학부 교수 (교신저자)
접수일자: 2025년 4월 28일, 수정완료: 2025년 5월 28일
게재확정일자: 2025년 6월 6일

Received: 28 April, 2025 / Revised: 28 May, 2025 /
Accepted: 6 June, 2025
*Corresponding Author: calafk@hansung.ac.kr
School of Computer Engineering, Hansung University, Korea

Ⅰ. 서 론

미팅은 학생을 비롯하여 많은 곳에서 상시적으로 일어나고 있다. 그러므로 미팅을 효과적으로 진행하기 위해 미팅룸을 만들고 회의 테이블과 카메라를 설치하고 전자 칠판을 사용하는 등 효과적인 미팅에 관한 많은 연구들이 진행되어 왔다[1,2,3]. 이러한 시스템을 설치하는 데는 많은 비용이 들어가고 별도의 공간이 필요하므로, 보통 규모의 기업들이나 많은 미팅들이 동시에 진행되는 대학 등에서는 비용과 공간 문제로 설치하기 쉽지 않기 때문에 비교적 단순한 회의 시스템의 개발이 필요하다.

한편, 오늘날 주제 발표나 미팅에는 빔프로젝터나 대형 디스플레이가 많이 활용되고 있으며, 1개의 노트북을 디스플레이에 연결하고 사전에 발표 자료를 이 노트북에 저장한 후 순서대로 발표가 진행된다. 반면, 학생들 사이의 회의나 기업 내 프로젝트 회의 등 많은 회의는 격식 없고 예정 없이 산발적으로 이루어지기 때문에 미리 디스플레이가 연결된 노트북에 회의 데이터를 저장해두지 않는다. 대신, 발표자가 돌아가면서 디스플레이에 자신의 노트북을 연결하는 식으로 발표가 이루어지고 있다. 하지만 이 방식은 각 사람이 자신의 노트북을 디스플레이에 연결하는 번거로움과 회의의 흐름을 끊는 문제를 유발한다.

본 논문은 이 문제를 해결하기 위해 본 연구팀은 멀티플렉서로 명명한 1개의 노트북이나 컴퓨터를 디스플레이에 연결 해두고, 회의의 참석자들이 원격으로 이 멀티플렉서 컴퓨터에 자신의 스크린을 공유시켜 디스플레이에 자신의 스크린을 출력시키는 방법을 착안하였다.

본 연구팀은 이 방식이 작동하려면 멀티플렉서가 어느 정도의 처리 능력을 갖추어야 하는지 판단하는 선행 연구가 필요하다고 판단하고, 멀티플렉서로 사용 가능한 다양한 하드웨어 플랫폼들을 후보로 정하고 실제 간단한 테스트 시스템을 구축하고 각 플랫폼의 성능을 측정하는 실험을 통해 멀티플렉서로 적합한 컴퓨터의 사양을 평가한다.

멀티플렉서로 평가할 플랫폼은 엔비디아의 젯슨 나노 2GB[4], 젯슨 나노 4GB, 젯슨 Orin 나노 8GB[5,6], 그리고 윈도우 노트북 2종류, 맥 노트북 1종류 등 총 6가지를 대상으로 결정하고 실험한다. 화면 공유에 사용된 콘텐츠는 정적 스크린 형식의 발표와 동적 스크린 형식의 발표로 구분하였다. 정적 발표는 발표 동안 스크린에 큰 변화가 없는 형태의 온라인 강의 동영상과 같은 발표이며, 동적 발표는 발표 동안 스크린에 변화가 많은 발표이다.

Ⅱ. 테스트 시스템 구축

1. 시스템 구조

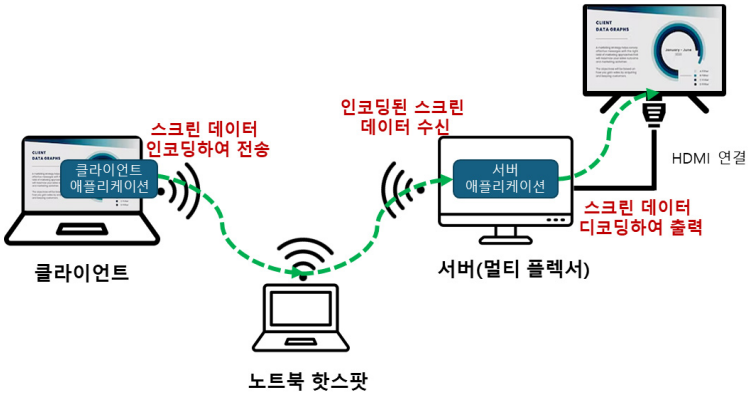

그림 1. 화면 공유 시스템
Fig. 1. Computer Screen Sharing System

본 논문에서 구성한 테스트 시스템은 그림 1과 같이 클라이언트 컴퓨터, 멀티플렉서 기기, 그리고 이들을 무선 인터넷으로 연결하는 노트북 핫스팟으로 구성하였다. 멀티플렉서는 지금부터 간단히 서버로 부른다. 클라이언트 컴퓨터는 대중적인 윈도우 노트북을 사용하였다. 무선 인터넷 연결을 위해 노트북 핫스팟을 사용한 것은 별도의 무선 공유기를 이용하는 것보다 실험이 편하다는 단순한 이유이다. 무선 인터넷은 1Gbps로 작동한다.

본 연구에서는 젯슨 나노 등 총 6개의 컴퓨터에 대해 서버로서의 성능을 측정한다. 실험이 시작되면 클라이언트 컴퓨터에는 실험을 위해 미리 준비된 테스트용 동영상을 재생시킨다. 그리고 클라이언트 컴퓨터의 스크린을 실시간으로 캡쳐하고 인코딩하여 서버로 전송하는 클라이언트 애플리케이션이 작동되고, 서버에는 서버 애플리케이션이 도착하는 비디오 스트림을 디코딩하고 디스플레이에 출력한다. 서버 애플리케이션은 비디오의 프레임들이 도착하는 시간, 디코딩에 걸린 시간, CPU 활용률 등을 측정하여 메모리에 저장해두었다가 실험이 끝나면 로그 파일에 저장하며, 추후 성능 평가 과정에서 분석된다. 클라이언트 애플리케이션과 서버 애플리케이션은 본 연구팀에 의해 직접 작성되었다.

표 1. 성능 평가 대상 디바이스들
Table 1. Devices subject to performance evaluation

타입	종류	운영체제	GPU 유무	CPU 클럭 (GHz)
임베디드 장치 (NVIDIA)	젯슨 나노 2GB	우분투 리눅스	있음	1.43
	젯슨 나노 4GB	우분투 리눅스	있음	1.43
	젯슨 Orin 나노 8GB	우분투 리눅스	있음	1.5~1.7
윈도우 노트북 (LG, HP)	인텔 i5/16GB	윈도우 11	있음	1.7~4.4
	인텔 i7/32GB	윈도우 11	있음	2.2~5
맥 노트북	M2/16GB	맥 운영체제	있음	2.42~3.49

2. 테스트 소프트웨어 개발

가. 테스트 소프트웨어 프레임워크

본 논문에서 성능 평가를 위해 작성된 클라이언트 애플리케이션과 서버 애플리케이션은 그림 2와 같다. 두 애플리케이션 모두 Electron 프레임워크[7,8]로 작성되었다. Electron은 Node.js와 Chromium을 결합하여 개발된 것으로 데스크톱 애플리케이션을 웹 페이지를 작성하는 방식으로 작성할 수 있도록 지원하는 프레임워크이다. GUI는 HTML5와 CSS로 작성하고, 자바스크립트 언어로 이벤트 처리 등 전체적으로 프로그램 흐름을 제어하면 된다. Electron을 활용한 이유는 애플리케이션이 작성이 편리하며 어떤 하드웨어/운영체제에서도 실행이 가능하기 때문이다. 본 연구에서는 표 1과 같은 6가지의 컴퓨팅 환경에서 서버 애플리케이션을 실행시켜야 하므로, 플랫폼의 독립성은 매우 중요한 요소이기 때문이다.

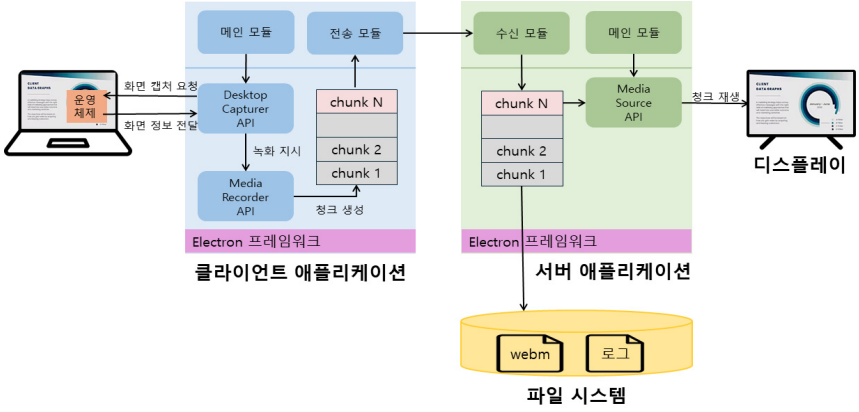

그림 2. 서버와 클라이언트의 소프트웨어 모듈들
Fig. 2. Software modules of Server and client

나. 테스트 과정

본 연구에서는 실시간으로 변하는 스크린 이미지를 비디오로 인코딩하고 디코딩하는 코드는 Electron에서 제공하는 Media Recorder API와 Media Source API를 활용하였으며, 이들은 VP8 코덱[9]을 활용한다. 클라이언트 애플리케이션은 Media Recorder API를 호출하면 Media Recorder는 Electron에 내장된 Desktop Capturer를 반복적으로 호출하여 현재 스크린 이미지를 계속 캡쳐하고 VP8 코덱을 이용하여 이 이미지들은 비디오 청크(chunk)로 인코딩한다. 그리고 생성한 청크를 전송 모듈을 통해 서버로 전송한다. 클라이언트와 서버 사이의 전송은 TCP 기반의 소켓 통신으로 이루어진다. 여기서 청크는 1개 이상인 비디오 프레임으로 구성되는 것으로 VP8 코덱에서 다루는 단위이다.

서버 애플리케이션은 클라이언트가 실행되기 전에 이미 실행되어 비디오 수신을 기다리고 있어야 한다. 수신 모듈은 도착하는 비디오 청크를 도착순으로 Electron 프레임워크의 Media Source API를 호출하여 전달하면, Media Source API는 각 청크를 VP8 코덱을 이용하여 디코딩하여 프레임들로 분리하고 각 프레임을 비디오 스크린으로 만들어 디스플레이에 출력한다. 이 과정에서 청크의 도착 시간과 청크가 디코딩되는데 걸리는 시간을 측정하여 로그에 저장한다. 사실, 수신 모듈은 청크를 받으면 이를 Media Source API에게 넘겨줌과 함께 클라이언트로 다시 보낸다. 클라이언트는 청크를 보내고 보낸 청크가 다시 돌아오는 시간을 측정하고 이 시간을 반으로 나누어, 청크가 전송되는 걸리는 네트워크 지연 시간으로 계산한다. 메인 모듈은 주기적으로 CPU 활용률을 측정하여 로그에 기록한다.

한편, Media Source API는 도착한 청크를 순서대로 파일에 저장하여 WebM[10] 형식의 한 개의 동영상 파일로 만든다. 실험이 종료되면, 서버 애플리케이션은 FFmpeg 라이브러리를 이용하여 WebM 파일은 분석하여 비디오 전체를 받는데 걸리는 시간과 비디오의 수신 fps(frames per second)를 계산하고, 경우에 따라서는 수신한 비디오를 재생하여 눈으로 오류가 발생하였는지를 검증할 수 있다.

2. 실행 시간 분석

이 절에서는 성능 평가를 위해 필요한 시간 요소들을 분석한다. 그림 3은 클라이언트에서 생성된 청크들이 서버 컴퓨터의 디스플레이에 출력되는 과정의 시간 요소들을 보여준다.

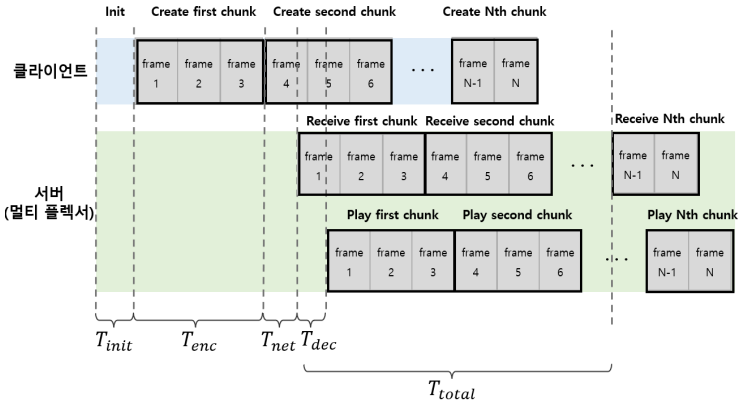

그림 3. 시간 분석
Fig. 3. Time Analysis

각 시간은 다음과 같이 정의한다.

- T_{init} : 클라이언트 애플리케이션의 초기화 시간
- T_{enc} : 클라이언트에서 캡쳐한 스크린 이미지들을 청크로 인코딩하는 시간
- T_{net} : 청크가 서버로 전송되는데 걸리는 네트워크 지연 시간

- T_{dec} : 서버에서 청크를 디코딩하는데 걸리는 시간
- T_{total} : 한 개의 실험에서 생성된 모든 청크를 수신하는데 걸린 시간

네트워크 지연 시간 T_{net}을 정확히 측정하기 위해서는 클라이언트와 서버가 일치하는 시계를 가지고 있어야 하는데 이것은 불가능하므로 다음과 같이 T_{net}을 측정하였다. 서버가 클라이언트로부터 청크를 수신하면 곧장 클라이언트로 전송하고, 클라이언트는 청크를 보낸 시간과 서버로부터 청크를 다시 돌려받았을 때의 시간 차이를 구하고 이를 반으로 나누어 T_{net}를 구하였다.

Ⅲ. 성능 평가

1. 평가 지표

본 논문에서는 멀티플렉서(서버)의 성능으로 적합한 컴퓨터의 사양을 측정하기 위한 목적이므로 다음 2가지의 지표를 사용한다.

- $T_{screendelay} = T_{net} + T_{dec}$, 영상 지연 시간 〈 100ms
- U_{cpu} : 평균 CPU 사용량 〈 30%

$T_{screendelay}$ 즉 영상 지연 시간은 클라이언트의 스크린이 서버의 디스플레이에 출력되는데 걸리는 평균 시간이며, U_{cpu}는 서버가 청크들을 수신하고 디코딩하는 동안 CPU 사용량의 평균값이다. 그리고 이들이 다음 조건에 동시에 만족할 때 적합으로 판정한다. 이 조건들은 매우 임의적이다. 추후 멀티 스트림 실험을 통해 좀 더 명확히 정해져야 할 것이다.

2. 작업 부하

테스트에 사용되는 작업 부하는 3가지로서 표 2와 같다. 본 연구에 사용되는 코덱은 VP8 코덱이며 초당 2Mb로 전송하도록 설정하였다. 초당 프레임 수는 15개로 설정하고 150ms 동안에 캡쳐한 프레임들을 한 개의 청크로 만들게 하였다. 테스트는 30분 동안 사용자가 클라이언트 컴퓨터에서 발표하는 것으로 하기 위해, 30분 분량의 강의 동영상을 클라이언트 컴퓨터에서 재생하는 식으로 실험하였다.

표 2. 작업 부하
Table 2. Workload

유형	종류	값
코덱 관련	코덱 종류	VP8
	코덱 속도	2Mbps
	초당 프레임 수	15fps
	청크 간격	150ms
영상 타입	정적 스크린	운영체제 온라인 강의 동영상
	동적 스크린	한국사 온라인 강의 동영상
실험 시간	발표 시간	30분

영상 타입이란 클라이언트의 사용자가 실행 시키는 영상의 타입을 뜻한다. 본 연구에서는 사용자가 회의에 참석하여 발표하는 것을 대상으로 삼기 때문에, PPT를 기반으로 프레임 사이에 큰 변화가 없이 진행되는 정적 스크린 타입

의 발표 영상과 발표 스크린의 이미지가 전면적으로 다르게 바뀌어 프레임 사이에 변화가 큰 동적 스크린 타입의 발표 영상에 대해 각각 실험을 진행한다. 정적 스크린 영상의 경우 영상의 압축률이 높을 것으로 예상되어 청크의 크기가 작고 통신에 따라 시간 소요가 작을 것으로 예측되며, 동적 스크린 영상의 경우 영상 압축률이 낮고 청크 크기 또한 작고 통신 시간이 클 것으로 예측된다. 정적 스크린의 샘플로는 본교의 '운영체제 온라인 강의 동영상'을 선택하였다. 각 슬라이드 모양이 동일하고 강의가 진행되는 동안 스크린에는 밑줄 긋기 등과 같은 약간의 변화만 발생한다. 동적 스크린의 샘플로는 유튜브에 있는 '한국사 온라인 강의 동영상'을 사용하였다. 슬라이드들은 이미지 중심으로 작성되었으며 서로 매우 다른 특징이 있다.

3. 성능 평가 결과

가. 성능 측정 결과

총 6개의 컴퓨터에 대해 서버를 구성하여 실험한 결과 측정된 성능 결과는 표 3, 4와 같다. 당초 평가 지표는 $T_{screendelay}$와 U_{cpu}이지만, 메모리 사용량, GPU 사용량을 추가로 측정하였다. 이 두 추가 지표는 서버로서의 적합성을 관찰하기 위한 부가적인 목적으로만 사용된다. CPU 활용률, 메모리 활용률은 Node.js 라이브러리 중 pidusage를 이용하여 측정하였고, GPU 활용률은 플랫폼별 적절한 CLI 도구를 자식 프로세스로 호출해 측정하였다.

표 3. 정적 스크린 영상에 대한 측정 결과
Table 3. Measurement Results for Static Screen Type

측정 요소	젯슨 나노 2GB	젯슨 나노 4GB	젯슨 Orin 나노 8GB	윈도우 Core i5 16GB	윈도우 Core i7 32GB	맥 M2 16GB
T_{net} (ms)	5.07	20.45	14.54	21.26	17.16	11.44
T_{dec} (ms)	36.7	38.08	32.06	58.46	37.14	37.32
$T_{screendelay}$ (ms)	41.77	58.53	46.6	79.72	54.3	48.76
U_{cpu} (%)	32.68	26.2	23.34	9.62	0.4	11.26
U_{mem} (%)	27.98	15.42	6.75	2.79	1.49	4.03
U_{gpu} (%)	27.94	27.78	24.06	20.02	2.64	6.32

표 4. 동적 스크린 영상에 대한 측정 결과
Table 4. Measurement Results for Dynamic Screen Type

측정 요소	젯슨 나노 2GB	젯슨 나노 4GB	젯슨 Orin 나노 8GB	윈도우 Core i5 16GB	윈도우 Core i7 32GB	맥 M2 16GB
T_{net} (ms)	6.95	20.04	13.24	31.18	17.62	14.58
T_{dec} (ms)	30.36	37.1	34.94	72	37.14	34.66
$T_{screendelay}$ (ms)	37.31	57.14	48.18	103.18	54.76	49.24
U_{cpu} (%)	40.18	40.9	31.56	14.96	0.58	14.82
U_{mem} (%)	36.58	23.61	9.3	4.07	2.12	7.75
U_{gpu} (%)	27.92	29	28.96	18.62	2.54	5.84

나. 성능 평가 분석

성능 측정 결과, 동적 스크린의 경우 정적 스크린에 비해 모든 종류의 기기에 대해 CPU 활용률과 메모리 활용률이 증가하였다. 젯슨 나노의 경우, 정적 스크린과 동적 스크린의 디코딩 시간에 별 차이를 나타내지 않지만, 윈도우나 맥의 경우 동적 스크린의 디코딩 시간에 증가함으로 보여준다. 전반적으로 예측과 거의 유사하게, 동적 스크린의 경우 컴퓨터에 더 많은 부하가 걸리는 것으로 평가된다.

이제, 본 연구에서 설정한 2가지 조건에 일치하는 기기를 검토해보자. 성능 측정 결과에 따르면 윈도우 Core i5

16GB는 $T_{screendelay}$ 〈 100ms 조건에 위배되며, 젯슨 나노 3종류는 모두 U_{cpu} 〈 30%의 조건에 위배된다. 여러 클라이언트로부터 멀티스트림을 수신될 때를 고려한다면, 측정 결과를 볼 때 메모리 사용량은 8GB 이상이면 충분할 것으로 판단된다. 전반적으로 윈도우 Core i7 32GB의 노트북이 가장 우수한 성능을 보였지만, 맥 16GB의 노트북 역시 적절하다고 판단된다. 결론적으로 임베디드 컴퓨터가 멀티플렉서로 사용하기에 성능에 한계가 있다고 판단되며, 멀티플렉서로 사용되려면 CPU는 Core i7, 메모리는 8GB 정도를 가진 노트북은 되어야 하는 것으로 판단한다.

IV. 결 론

본 연구 팀은 여러 클라이언트 컴퓨터들을 무선 인터넷으로 연결하고 이들로부터 스크린 이미지를 실시간으로 수신하고 이 비디오 스트림들을 연결된 디스플레이에 출력하는 서버 소프트웨어를 개발하고자 한다. 본 연구는 서버의 개발에 앞서, 서버의 성능에 적절한 사양의 컴퓨터를 찾고자, 젯슨 나노 2GB, 젯슨 나노 4GB, 젯슨 Orin 나노 8GB, 윈도우 노트북 Core i5 16GB, 윈도우 노트북 Core i7 32GB, 맥 노트북 M2 16GB 등 6가지 컴퓨터 각각에 대해, 클라이언트로부터 스크린 이미지를 실시간으로 수신하였을 때 서버에 나타내는 여러 성능 요소들은 측정하는 실험을 진행하였다. 그리고 네트워크 지연 시간과 비디오 디코딩 시간의 합이 10ms 이하여야 하고, 서버의 CPU 활용률이 30%보다 작아야 한다는 2개의 조건을 성능 평가의 지표로 정의하였다. 성능 측정 후 데이터를 분석한 결과 이 두 조건에 일치하는 기기는 윈도우 노트북 Core i7 32GB, 맥 노트북 M2 16GB의 2가지로 평가되었다. 본 실험 결과와 실험 과정에서 취득한 성능 데이터들은 본 연구팀이 추후 멀티 스트림을 수신하여 디스플레이에 출력하는 서버 컴퓨터를 개발하는데 꼭 필요할 것으로 판단된다.

References

[1] Ross Cutler, et. al, “Distributed meetings: a meeting capture and broadcasting system”, MULTIMEDIA '02: Proceedings of the tenth ACM international conference on Multimedia, pp. 503-512, Dec. 2002. https://doi.org/10.1145/641007.641112

[2] Ramesh Jain, Pilho Kim, Zhao Li, “Experiential meeting system”, Proceedings of the 2003 ACM SIGMM Workshop on Experiential Telepresence. pp. 1-12, Nov. 2003. http://doi.org/10.1145/982484.982486

[3] Zhiwen Yu, Yuichi Nakamura, “Smart meeting systems: A survey of state-of-the-art and open issues”, ACM Computing Surveys (CSUR), Vol. 42, Issue 2, pp. 1-20, March 2010. https://doi.org/10.1145/1667062.1667065

[4] Jetson Nano, https://www.nvidia.com/ko-kr/autonomous-machines/embedded-systems/jetson-nano/product-development/

[5] https://www.digitaltoday.co.kr/news/articleView.html?idxno=546207

[6] https://www.nvidia.com/ko-kr/autonomous-machines/embedded-systems/jetson-orin/nano-super-developer-kit/

[7] Dan-Bi Yoon, Dong-Geon Lee, Ah-Rin Jeon, Eun-Bi Kim, Yeo-Rin Ha, Kitae Hwang, “Concise Memo and Sharing System using AI: QuickQuick”, The Journal of The Institute of Internet, Broadcasting and Communication (IIBC), Vol. 24, No. 6, pp.21-28, Dec. 2024. https://doi.org/10.7236/JIIBC.2024.24.6.21

[8] Jiyoon Ok, Juyeon Soung, Chaewon Park, Kitae Hwang, “Implementation Details of EPUB Reader using GraphRAG”, International Journal of Internet, Broadcasting and Communication, Vol. 17, No. 2, pp. 223-231, May 2025. http://dx.doi.org/10.7236/IJIBC.2025.17.2.223

[9] Jim Bankoski, Paul Wilkins, Yaowu Xu, "Technical overview of VP8, an open source video codec for the web," 2011 IEEE International Conference on Multimedia and Expo, July 2011, pp. 1-6. http://doi.org/10.1109/ICME.2011.6012227

[10] Jim Bankoski, “Intro to WebM”, NOSSDAV '11: Proceedings of the 21st international workshop on Network and operating systems support for digital audio and video, pp. 1-2. June 2011. https://doi.org/10.1145/1989240.1989242

저 자 소 개

전 아 린(정회원)

- 한성대학교 컴퓨터공학부 재학
- 관심분야 : 모바일 시스템, 데이터 사이언스

윤 단 비(정회원)

- 한성대학교 컴퓨터공학부 재학
- 관심분야 : 모바일 시스템, 웹 공학, IoT, 빅데이터

이 동 건(정회원)

- 한성대학교 컴퓨터공학부 재학
- 관심분야 : 웹 서버, IoT

최 재 현(정회원)

- 한성대학교 컴퓨터공학부 재학
- 관심분야 : 모바일 시스템, 웹 공학

김 은 비(정회원)

- 한성대학교 컴퓨터공학부 재학
- 관심분야 : 모바일 시스템, 웹 공학, 빅데이터

황 기 태(종신회원)

- 서울대학교 컴퓨터공학과 학사
- 서울대학교 컴퓨터공학과 석사
- 서울대학교 컴퓨터공학과 박사
- 현재 한성대학교 컴퓨터공학부 교수
- 경력
  University of Florida 방문 교수
- 관심분야 : 모바일 시스템, IoT, 인공지능

※ 본 연구는 한성대학교 교내 학술 연구비를 지원받았음