

Demo Application of Synchronous Mobile Distance Learning System

July 16, 2025

Hosung.Kim

1 서론

선행 보고서에서는 동기식 모바일 원격 교육 시스템에 대한 이해를 목적으로 텍스트 기반 채팅, 강사의 음성 및 영상 공유, PDF 기반 강의 자료 주석 기능 등을 포함한 데모 애플리케이션을 설계 및 구현하였다. 해당 시스템은 RTMP(Real-Time Messaging Protocol)[1] 및 WebSocket 기반의 통신 구조 위에 구축되었으며, 제한된 환경에서의 실시간 강의 및 상호작용 가능성을 검증하였다.

그러나 기존의 시스템은 단일 강의실과 제한된 기능으로 구성되어, 실질적인 교육 서비스로의 확장을 위해서는 기능적 보완이 요구된다. 이에 본 보고서에서는 다음과 같은 네 가지 기능을 추가로 설계 및 구현하였다. 첫째, 사용자가 직접 강의실을 생성하고 선택적으로 입장할 수 있는 다중 강의실 구조를 도입하였다. 둘째, 강의자가 실시간으로 참여자 명단을 확인할 수 있는 기능을 추가하였다. 셋째, 강의자가 송출중인 자신의 전면 카메라 화면을 실시간으로 확인할 수 있는 기능을 추가하였다. 넷째, 공유 화이트보드 기능을 추가하였다.

2 설계

2.1 구조

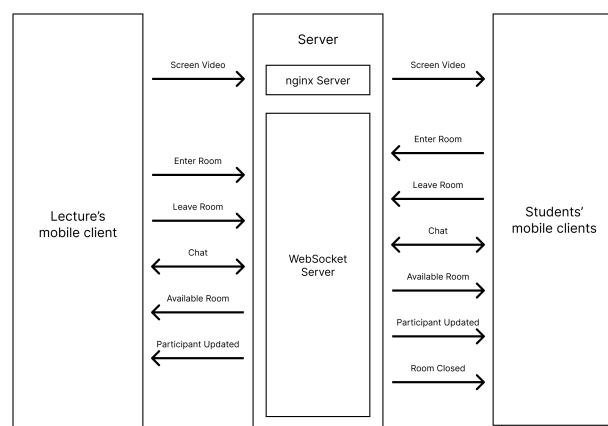


Figure 1: 강사와 학생간 데이터 흐름

기존의 시스템은 Java Spring[2]의 채팅 서버, NGINX[3]의 스트리밍 서버, iPadOS의 클라이언트로 구성되어 있어 서로 다른 환경에서 동작하였다. 이는 실행 및 테스트의 복잡성, 유지보수의 비효율성을 초래하였다. 이를 해결하기 위해 모든 프로그램을 Xcode[4] 기반 프로젝트 내에서 통합적으로 관리할 수 있도록 구조를 개선하였다.

한편, WebSocket 서버는 채팅 기능 뿐만 아니라 다중 강의실 기능을 지원하기 위해 확장되었다. 새로운 시스템에서는 강의실 생성, 입장 및 퇴장, 채팅, 강의실 목록 조회, 참여자 명단 동기화 등 다중 강의실 기능 전반을 담당하도록 설계하였다.

3 구현

3.1 서버

서버는 클라이언트와의 개발 환경의 일관성을 확보하고 운영 및 유지보수의 효율성을 높이기 위해 스트리밍 서버의 실행 방식을 개선하고 WebSocket 서버를 Swift 기반으로 전면 재구현하였다.

스트리밍 서버는 기존과 동일하게 NGINX RTMP 서버를 사용하되, 실행 방식에 있어 통합이 이루어졌다. 기존에는 NGINX를 별도의 실행 환경에서 수동으로 실행해야했지만, 개선된 구조에서는 Swift 서버 실행 시 Process()를 통해 nginx.exe와 nginx.conf 설정파일을 프로그램적으로 호출함으로써 서버 실행 및 종료를 자동화하였다. 이를 통해 두 개의 서버를 하나의 Xcode Target으로 관리할 수 있게 되어 실행 절차가 간소화되었다.

WebSocket 서버는 기존의 Java Spring 기반에서 Network Framework[5]의 NWProtocolWebSocket을 활용한 Swift 서버로 재구현하였다. 새롭게 구현된 서버는 채팅 뿐만 아니라 강의실 입장 및 퇴장, 강의실 목록, 참여자 명단 등 강의실 관리 전반을 포괄적으로 처리할 수 있도록 하였다. 서버-클라이언트 간 통신에는 Codable을 상속한 enum을 사용하여 기능 확장을 용이하게 하고 다양한 종류의 메시지를 주고받을 수 있도록 하였다.

WebSocket 서버는 각 강의실을 UUID를 key로 하는 딕셔너리 형태로 관리하며, 강의실에 변화가 있을 때마다 Available Room 메시지를 모든 클라이언트에게 브로드캐스트한다. Enter Room, Leave Room 메시지를 수신받으면 해당 클라이언트를 관련 강의실에 등록하거나 제거하며, 강의자가 퇴장할 경우 해당 강의실의 모든 학생에게 Room Closed 메시지를 전송하여 강의 종료를 알린다. 또한 참여자에 변화가 있을 때마다 Participant Updated 메시지를 해당 강의실 내 모든 클라이언트에게 전송하며, 채팅 메시지는 동일 강의실 내 모든 클라이언트에게 브로드캐스트된다.

3.2 클라이언트

클라이언트는 기존 단일 강의실 구조에서 다중 강의실 환경으로의 확장, 강의자의 화면 공유 방식의 개선, 접속자 명단 표시, 공유 화이트보드 및 강의자 캠 기능의 추가에 따라 UI 및 내부 로직 전반에 걸쳐 개편이 이루어졌다.

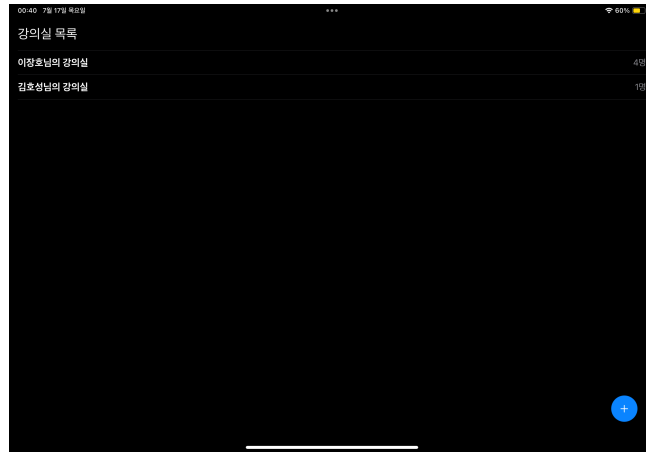


Figure 2: 홈 화면의 유저 인터페이스

기존 시스템에서는 사용자가 이름을 입력하고 강사 또는 학생의 역할을 선택한 뒤, 하나의 고정된 강의실에 입장하는 구조였다. 반면, 개선된 시스템에서는 홈화면에서 현재 활성화된 강의실 목록을 조회할 수 있으며, 사용자는 강의실을 새로 생성하거나 기존 강의실 중 하나를 선택하여 입장할 수 있도록 변경되었다. 홈 화면의 유저 인터페이스는 그림2와 같다.

강의자 화면 공유 방식에도 개선이 이루어졌다. 기존에는 전체 화면을 송출하였으나, 개선된 시스템에서는 강의자료가 위치한 특정 영역만을 추출하여 전송한다. ReplayKit[6]은 기본적으로 전체화면의 CMSampleBuffer만을 제공하므로, 선택 영역 송출을 위해 CMSampleBuffer를 UIImage로 변환한 뒤, 특정 뷰 영역만 잘라낸 후 CGContext에 렌더링하여 CVPixelBuffer로 변환하였다. 최종적으로 이를 CMSampleBuffer로 변환하여 서버로 송출하였다.



(a) PDF

(b) 공유 화이트보드

Figure 3: 강사 측 클라이언트의 유저 인터페이스

기존 시스템에서는 강의자가 자신의 캠 영상을 확인할 수 없었으나, 전면 카메라의 영상을 AVCaptureVideoPreviewLayer를 활용하여 실시간으로 렌더링함으로써 강의자가 자신의 영상 상태를 직접 확인할 수 있도록 하였다.

또한 현재 강의실에 접속 중인 사람들의 명단을 확인할 수 있다. 서버로부터 수신되는 Participant Updated 메시지를 구독하여 참여자 목록을 실시간으로 갱신한다.

공유 화이트 기능은 기존 PDF 주석 기능에 사용되었던 CanvasView를 활용하여 구현되었다. CanvasView는 터치 이벤트를 통해 UIBezierPath를 생성하고 이를 배열에 저장하는 방식으로 동작하며 undo, clear 등의 기능도 지원한다.

4 결론

본 보고서에서는 기존의 단일 강의실 기반 동기식 모바일 원격 교육 시스템을 다중 강의실 구조로 확장하고, 실시간 상호작용 기능을 강화한 클라이언트 및 서버 시스템을 설계 및 구현하였다. 시스템 통합의 관점에서는, 기존에 분리되어 운영되던 Java Spring 기반 채팅 서버와 NGINX 스트리밍 서버를 Xcode 프로젝트 내에서 Swift 기반으로 통합하여 실행 및 관리를 일원화하였고, WebSocket 서버는 다중 강의실 기능을 포괄적으로 지원하도록 재구현되었다.

클라이언트 측면에서는 홈 화면을 통해 사용자가 강의실을 선택하거나 생성할 수 있는 구조로 개선하였으며, 강의자는 자신의 전면 카메라를 실시간으로 확인할 수 있고, 강의 자료 영역만을 선택적으로 송출함으로써 화면 공유의 효율성을 향상시켰다. 이외에도 접속자 명단 실시간 확인, 공유 화이트보드 기능을 추가함으로써, 강사와 학생 간의 실시간 상호작용 범위를 실질적으로 확장하였다.

그러나 실제 강의 현장에서는 강의 자료와 칠판이 동시에 제시되는 반면, 본 시스템에서는 PDF 강의자료와 공유 화이트보드를 동시에 노출되지 않는다. 이는 학습자 경험 측면에서 물리적 강의 환경에 비해 정보의 병렬적 제공이 제한된다는 점에서 개선이 필요한 부분이다. 향후에는 이러한 한계를 해소하기 위해 두가지 정보를 동시에 표시할 수 있는 UI 구성 방식에 대한 탐색할 예정이다.

전체 소스코드는 <https://github.com/H0sungKim/CollaborativeComputingLab>에서 확인할 수 있다.

References

1. Parmar, H., Thornburgh, M. (eds.): Adobe's Real Time Messaging Protocol. Adobe (2012)
2. Spring. <https://spring.io/>
3. NGINX. <https://nginx.org/>
4. Xcode. <https://developer.apple.com/kr/xcode/>
5. Network. <https://developer.apple.com/documentation/Network>
6. ReplayKit. <https://developer.apple.com/documentation/ReplayKit>