|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **회 원 명** | **학교** | **학과** | **학년** | **연락처** | **주민등록번호** |
| 정우진 | 부경대 | 기술경영협동과정 | 석사 | 010-4802-3685 | 860517-\*\*\*\*\*\*\* |
| 김하영 | 울산대 | 컴퓨터정보통신공학 | 4 | 010-4845-1783 | 860719-\*\*\*\*\*\*\* |
| 정봉재 | 동아대 | 컴퓨터공학 | 3 | 010-6550-9569 | 870420-\*\*\*\*\*\*\* |

|  |  |
| --- | --- |
| **구 분** | **내 용** |
| **개발 목적**  **및 동기** | 스마트 폰의 발전으로 스마트 폰 내에 웹 서버를 탑재 할 수 있게 되었다. 스마트 폰에 웹 서버를 탑재하게 되면 다양한 플랫폼에서 별도의 애플리케이션 설치없이 웹 브라우저 만으로 스마트 폰의 데이터를 공유할 수 있다. 하지만 스마트 폰은 기본적으로 NAT의 존재에 의해 외부에서 직접 통신이 불가능하다. NAT 환경을 극복하는 스마트 폰 웹 서버를 구축하고 스마트 폰의 파일과 실시간으로 촬영되는 영상을 다양한 플랫폼의 브라우저를 통해 공유할 수 있도록 하는 것이 목적이다. |
| **개발 환경**  **및 일정** | • 개발환경 : Windows 7, Ubutu 10.04, JDK1.7, Android SDK, Android NDK  , MS-SQL 2008R2  • 개발도구 : Eclipse, WebStorm, EditPlus, VisualStudio2010  • 개발언어 : Java, Javascript, jsp, Node.js, PHP, C#, C  • 개발기간 : 2012. 07. 01 ~ 2012. 10. 30 (4개월) |
| **우수성/**  **창의성** | 기존의 유사한 어플리케이션들이 단일의 모바일 앱으로 사용자 측면에서의 서비스가 부족한데, 접속자는 이전에 접속했던 서버에 대한 정보를 제공받지 못해 접속 시 매번 문자나 구두로 요청을 해야만 하는 불편함이 따른다. 이러한 불편함을 해소 하고자 사용자를 관리하는 서비스를 제공하여 애플리케이션의 접근성을 높였다. 또한 3G/4G의 NAT환경으로 인해 스마트폰이 서버로서의 제약을 Http Relay Service를 이용하여 해소 하였고, Mjpeg, HLS를 활용하여 실시간 스트리밍 기능도 추가하여 현재 출시된 KissAir, AirLink등과 같은 어플리케이션과 차별성을 두었다. |
| **PJ 완료시**  **연계 방안** | 파일과 영상 데이터 뿐만 아니라 이동형 웹 서버로써의 장점을 살릴 수 있다. 스마트 폰에 내장된 각종 센서들을 이용하여 상황인지 컴퓨팅을 통해 주변의 상황을 판단해주고 웹 브라우저로 서비스를 제공해주는 방안을 생각해 볼 수 있다. 또한 Node.js의 Socket.IO기능을 활용하여 웹에서의 핸드폰 원격제어가 가능 할 것이다. |

|  |  |  |
| --- | --- | --- |
| **과 제 명** | **스마트폰 웹 서버를 활용한 Data Link** | |
| **과제기간** | 2012. 07. 01 ~ 10. 31 (4개월) | |
| **참여인원** | | 3명 |

**목 차**

**- 내용 목차 -**

1. 선정 배경 [5](#_Toc2686)

1) 스마트 폰 사용자 2700만 시대 [5](#_Toc2687)

2) 스마트 폰의 비약적 발전과 모바일 웹 서버 [5](#_Toc2688)

3) 플랫폼과 통신망에 구애받지 않는 데이터 공유의 필요성 [6](#_Toc2689)

2. 개발 목적 [6](#_Toc2690)

1) 스마트 폰 웹 서버 구축을 통한 웹 기반 데이터 공유 [6](#_Toc2691)

2) 사용자 관리를 통한 편리한 접근성 [6](#_Toc2692)

3) 통신망과 기기에 따른 제약사항을 극복한 데이터 공유 [6](#_Toc2693)

3. 기술 동향 [7](#_Toc2694)

1) 미니 웹 서버 [7](#_Toc2695)

2) NAT Traversal [8](#_Toc2696)

3) Node.js [9](#_Toc2697)

4) Web Socket 과 Socket.io [10](#_Toc2698)

5) HLS(HTTP Live Streaming) [11](#_Toc2699)

4. 개발 목표 [11](#_Toc2700)

1) 안드로이드 스마트 폰 웹 서버 구축 [11](#_Toc2701)

2) NAT Check를 통한 Http Relay 기능을 수행하는 중계 서버 [11](#_Toc2702)

3) 등록 서버 목록 관리 [11](#_Toc2703)

4) 파일 공유 기능 [12](#_Toc2704)

5) 실시간 스트리밍 기능 [12](#_Toc2705)

5. 개발 내용 [12](#_Toc2706)

1) System Conceptual Diagram [12](#_Toc2707)

2) System Architecture [13](#_Toc2708)

3) Bridge Server [14](#_Toc2709)

4) Web Server Architecture [15](#_Toc2710)

5) Network Connection Scenario [15](#_Toc2711)

6) Http Relay Server [16](#_Toc2712)

7) Live Streaming [17](#_Toc2713)

8) Scenario [25](#_Toc2714)

6. 개발 환경 [29](#_Toc2715)

7. 개발 일정 [30](#_Toc2716)

8. 역할 분담 [30](#_Toc2717)

9. 참고문헌 [31](#_Toc2718)

**- 그림 목차 -**

[그림 1] 스마트 폰 가입자 수 [5](#_Toc2719)

[그림 2] 최신 스마트 폰 스펙 [5](#_Toc2720)

[그림 3] 정적 웹 서버 동작 방식 [7](#_Toc2721)

[그림 4] 동적 웹 서버 동작 방식 [7](#_Toc2722)

[그림 5] 전체 시스템 개념도 [12](#_Toc2723)

[그림 6] System Architecture [13](#_Toc2724)

[그림 7] [14](#_Toc2725)

[그림 8] Web Server Architecture [15](#_Toc2726)

[그림 9] Network Connection Scenario [15](#_Toc2727)

[그림 10] Http Relay Server [16](#_Toc2728)

[그림 11] Motion-JPEG Push 과정 [17](#_Toc2729)

[그림 12] HLS Architecture [18](#_Toc2730)

[그림 13] FFmpeg Configure [19](#_Toc2731)

[그림 14] libavcodec [20](#_Toc2732)

[그림 15] libavfilter, libavformat, libavutil, libswscale [20](#_Toc2733)

[그림 16] libx264 [21](#_Toc2734)

[그림 17] segmenter [22](#_Toc2735)

[그림 18] m3u8 파일의 실시간 변화 [23](#_Toc2736)

[그림 19] 스마트 폰 웹 서버 등록 [25](#_Toc2737)

[그림 20] 구동 중인 웹 서버 정보 로드 [25](#_Toc2738)

[그림 21] 사용자 인증 웹 페이지 [26](#_Toc2739)

[그림 22] 등록 된 웹 서버 검색 [26](#_Toc2740)

[그림 23] 등록한 웹 서버 목록 [27](#_Toc2741)

[그림 24] 웹 서버 연결 요청 [27](#_Toc2742)

[그림 25] 스마트 폰 웹 서버의 파일 목록 [28](#_Toc2743)

[그림 26] 실시간 촬영 영상 데이터 스트리밍 [28](#_Toc2744)

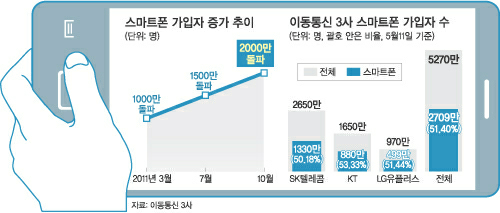
[그림 27] 개발 일정 [30](#_Toc2745)

[그림 28] 역할 분담 [30](#_Toc2746)

**- 표 목차 -**

[표 1] m3u8 format infinitive maker [24](#_Toc2747)

2. **선정 배경**
   1. **스마트 폰 사용자 2700만 시대**



[그림 1] 스마트 폰 가입자 수

휴대전화 가입자 5천255만명 중 스마트 폰 사용자는 2천672만명으로 50.84%에 달한다. 스마트 폰 열풍이 시작된 지 3년만에 사용자가 국내 이동통신 가입자의 절반을 넘어선 것이다. 이런 스마트 폰의 대중화는 우리 사회가 이미 "스마트 사회"로 깊숙이 들어와 있음을 의미한다.

스마트 폰은 기존의 단순한 통화 기능에서 벗어나 쇼핑, 오락 등 일상생활의 공간이 됐고 사회 구성원을 연결해주는 소셜네트워크 역할을 해내고 있다. 산업측면에서도 스마트 폰은 정보통신 산업은 물론 금융, 자동차 등 산업 전반에 걸쳐 동반성장을 만들어가고 있다.

* 1. **스마트 폰의 비약적 발전과 모바일 웹 서버**



[그림 2] 최신 스마트 폰 스펙

모바일 기기가 나날이 발전함에 따라 최근 출시되고 있는 스마트 폰의 성능은 점점 PC에 가까워 지고 있다. 뿐만 아니라 무선 통신망의 발전도 유선 통신망에 근접하고 있다. 이러한 무선통신망과 모바일 기기의 발전은 PC에서만 가능했던 작업들을 모바일 기기에서도 가능하게 하고 있다. 그 작업들 중 한가지는 모바일 기기에 웹 서버를 탑재하여 이동형 웹 서버로 활용하는 것이다.

* 1. **플랫폼과 통신망에 구애받지 않는 데이터 공유의 필요성**

현재, 플랫폼 및 통신망에 따라 데이터를 교환하는데 제약이 따른다. 예를 들어 안드로이드폰과 아이폰 또는 PC와 스마트 폰 등의 타 플랫폼간 데이터 교환은 각 플랫폼에 전용 애플리케이션을 설치하여야 한다.

또한 NAT의 존재에 의해 서로 다른 내부망이나 외부망에서 다른 내부망으로의 직접 통신은 기본적으로 제한되어 있다. 일반적으로 3G/4G 및 공유기 사용 환경이 그러하다.

이러한 제약사항들은 통신과 기기의 발달을 적극 활용하지 못하고 있는 것이다.

1. **개발 목적**
   1. **스마트 폰 웹 서버 구축을 통한 웹 기반 데이터 공유**

스마트 폰에 웹 서버를 탑재하게 되면 PC, Tablet 등 다양한 플랫폼에서 브라우저를 통해 해당 스마트 폰의 데이터를 공유할 수 있다. 브라우저를 통해 데이터를 공유한다는 의미는 서버로 접근하는 사용자 측에서는 전용 애플리케이션을 설치할 필요없이 데이터를 공유할 수 있다는 뜻이다. 뿐만 아니라 이동형 웹 서버를 구현함으로써 시간과 위치에 따른 동적인 데이터를 수집하여 웹 상에서 제공할 수 있다.

이러한 장점을 이용해 스마트 폰에 저장된 파일과 정보들, 그리고 실시간으로 수집할 수 있는 영상데이터를 브라우저를 통해 웹 상에서 제공하는 것이 목적이다.

* 1. **사용자 관리를 통한 편리한 접근성**

스마트 폰에 웹 서버를 구동하고, 다른 사용자가 이에 접근하려 하면 서버를 구동하고 있는 쪽에서 접근 가능한 URL를 알려 줘야 하는데, 현재 시중에 나와 있는 KissAir, AirLink, AirDroid같은 경우 다른 사용자에게 URL을 알려주는 어떠한 기능도 없거나 문자로 URL을 전송하는 기능이 있는 것이 전부이다. 이러한 점은 사용자가 자료를 공유하기 위한 서비스적인 측면에서는 큰 불편함이 있으며 실재 이러한 데이터 링크 어플리케이션을 서비스 하는 측면에서는 굉장히 불리한 점으로 작용 할 수 있다. 따라서 사용자가 좀 더 편리하게 서버에 접근 하게 하기 위해 웹을 통해 사용자를 관리하고 접근 할 수 있도록 제공하여 서비스 적인 측면에서 좀 더 개선 시켰다.

* 1. **통신망과 기기에 따른 제약사항을 극복한 데이터 공유**

현재 팬택 베가 LTE EX의 Air Link와 삼성 갤럭시S 시리즈의 Kies Air 애플리케이션이 스마트 폰에 웹서버를 구동하여 파일 공유 기능을 제공하고 있지만 단점과 제약사항이 존재한다.

Air Link의 경우 WIFI 접속 환경이 공유기에 접속 된 내부망인 경우 같은 내부망에서만 접속이 가능하고 외부망에서는 접속이 불가능 하다. 또 3G 및 4G 접속 환경일 경우 같은 통신사 3G 및 4G 접속 환경일 경우에만 접속이 가능하며 WIFI 및 외부망을 통해서는 접속이 불가능 하다. 삼성의 Kies Air 경우에는 WIFI(공인 IP) 모드만 지원 된다. 뿐만 아니라 Air Link의 경우는 팬택 베가 LTE EX 스마트 폰에서만 Air Link 기능을 제공하며 Kies Air는 삼성 갤럭시S 시리즈 스마트 폰에서만 구동이 된다. 즉 접속 환경과 제조사별 기기에 따라 제약적인 서비스만을 제공하고 있다.

또 현재 파일공유를 위해 서버가 되는 폰으로 접근해야하는 사용자들은 서버측에서 허가 요청을 보내야지만 접근이 가능하다. 서버 폰에 접근할 때마다 매번 허가를 기다려야하는 것은 상당히 불편한일이다.

이러한 제약사항들을 극복하는 범용적이고 사용성 높은 데이터 공유 서비스를 개발 하는 것이 목적이다.

1. **기술 동향**
   1. **미니 웹 서버**

웹 서버는 인터넷 환경의 클라이언트에서 실행되는 웹 브라우저들이 요청한 해당 컨텐츠를 응답으로 제공하는 작업을 수행하는 애플리케이션을 말한다.



[그림 3] 정적 웹 서버 동작 방식

① 유저의 웹 브라우저에서 데이터 요청(Request)

② 웹 서버가 파일 시스템에서 데이터 검색

③ 웹 서버가 파일 시스템에서 데이터 회수

④ 웹 서버가 웹 브라우저에 데이터 응답(Response)



[그림 4] 동적 웹 서버 동작 방식

① 유저의 웹 브라우저에서 동적 데이터 요청(Request)

② 웹 서버는 CGI 프로그램 실행 및 파라미터 전달

③ 웹 서버가 CGI 실행 결과 검색 및 데이터 회수

④ 웹 서버가 웹 브라우저에 결과 값 리턴(Response)

위의 웹 서버들의 수행 과정을 정리하면 다음과 같다.

첫 번째 단계로 실행과 동시에 클라이언트와 연결이 일어날 때까지 기다리면서 요청이 들어오면 요청을 읽게 된다. 즉 데몬 프로세스로 작동한다.

두 번째 단계로 요청 메시지로부터 헤더를 파싱하여 어떤 문서를 서비스해 주어야 하는지 알아낸다.

세 번째 단계로 서비스할 문서를 메모리로 가져온다.

네 번째 단계로 네트워크로 보내게 된다. 즉 요청 클라이언트에게 응답 메시지를 보내게 된다.

마지막 다섯 번째 단계로 해당 사이클에 대한 정보를 기록(Logging)함으로써 하나의 요청에 대한 수행 과정을 종료한다.

범용 웹 서버들은 위의 기본적인 기능들과 함께 다양한 기능들을 탑재하고 있다. 하지만 특수한 목적을 위한 임베디드에서 운영될 웹 서버들은 해당 목적에 맞는 기본적인 기능과 최소한의 CPU Memory 자원을 활용하여 클라이언트의 요구에 응답하여야 한다.

웹 서버 중 공개소프트웨어 범용 웹 서버 애플리케이션들은 Apache, Lighttpd, Tomcat, Resin, JBoss, Jetty 등이 있다. 그리고 안드로이드 플랫폼처럼 사용 가능한 자원이 제한되어 있는 임베디드 용도로 적합한 공개소프트웨어 미니 웹 서버는 boa, thttpd, i-jetty 등이 있다. 이와 같은 임베디드 용도의 웹 서버들은 특수한 목적을 위한 것이므로 범용 웹 서버들이 필요로 하는 다양한 기능을 가질 필요는 없으며, 해당 목적에 맞는 기능만 수행하면 된다.

* 1. **NAT Traversal**

NAT은 사설 IP주소를 공인 IP주소로 변환해주는데 사용하는 통신망의 주소 변환기이다. NAT을 사용하는 목적에는 2가지가 있다. 첫 번째는 IPv4의 부족한 공인 IP주소를 절약할 수 있다는 점이다. 두 번째는 인터넷이란 공공망과 연결되는 사용자들의 고유한 사설망을 침입자들로부터 보호할 수 있다는 점이다. 공개된 인터넷과 사설망 사이에 방화벽을 설치하여 외부 공격으로부터 사용자의 통신망을 보호하는 기본적인 수단으로 활용할 수 있다. 이때 외부 통신망 즉 인터넷망과 연결하는 장비인 라우터에 NAT을 설정할 경우 라우터는 자신에게 할당된 공인 IP주소만 외부로 알려지게 하고, 내부에서는 사설 IP주소만 사용하도록 하여 필요시에 이를 서로 변환시켜 준다. 따라서 외부 침입자가 공격하기 위해서는 사설망의 내부 사설 IP주소를 알아야 하기 때문에 공격이 불가능해지므로 내부 네트워크를 보호할 수 있다. 하지만 내부 네트워크를 보호하기 위해 존재하는 NAT에도 부작용이 있다. 외부 호스트에서 NAT 내부의 호스트로 먼저 연결하는 것, 즉 패킷을 먼저 전송해 오는 것이 기본적으로 불가능하다. 이것은 한정된 IP 주소를 다수의 내부 호스트가 공유하는 NAT의 본질적인 특성에서 기인한 것이다. 외부 호스트에서 NAT 내부의 호스트로 패킷을 전달하기 위해서는, NAT에 바인딩 정보가 있어야 한다. 그러나 이 바인딩정보는 내부 호스트에 의해서만 생성될 수 있다. 일반적인 TCP/IP 환경은 기본적으로 우선권 없이 패킷을 전송하는 것이 보장되지만 NAT이 존재하면 문제가 발생할 수 있는데 이렇게 문제가 발생하면 NAT의 특성을 이용하여 원래의 TCP/IP 환경을 조성해주는 것이 NAT Traversal이다.

이러한 NAT Traversal에 대해 알려진 기술들이 몇 가지 있다. 크게 두 가지로 분류 할 수 있는데 첫 번째는 NAT 행동에 기반한 프로토콜 및 기술이고 두 번째는 NAT 제어에 기반한 기술이 있다.

NAT 행동에 기반한 프로토콜 및 기술

• Session Traversal Utilities for NAT (STUN)

• Traversal Using Relay NAT (TURN)

• NAT-T Negotiation of NAT-Traversal in the IKE

• Teredo tunneling uses NAT traversal to provide IPv6 connectivity.

• Session Border Controller (SBC)

• UDP hole punching

• TCP hole punching

• ICMP hole punching

NAT 제어에 기반한 기술

• Realm-Specific IP (RSIP)

• Middlebox Communications (MIDCOM)

• SOCKS

• NAT Port Mapping Protocol (NAT PMP)

• Internet Gateway Device (IGD) Protocol, defined by the Universal Plug and Play (UPnP) Forum.

• Application Layer Gateway (ALG)

이처럼 NAT을 통과하는 문제를 해결하기 위한 솔루션은 여러 가지가 있지만 모든 상황에서 동작하는 완벽한 솔루션은 없다. 이것은 NAT이 표준화되어 있지 못한 것이 가장 큰 이유이다. 알려진 대부분의 기술들은 공통적으로 전역적으로 라우팅 가능한 IP주소(글로벌 IP 또는 공인 IP)를 가진 외부호스트(중계서버 혹은 랑데뷰서버)의 지원이 필요하다.

* 1. **Node.js**

node.js는 서버사이드 자바스크립트 기술로 확장 가능한 네트워크 프로그램을 빌드하기 위해 간편한 방법을 제공하는 것으로 서버프로그램에서 많이 사용되는 JSP, PHP와같은 언어에서 모든 요청에 대해 잠재적으로 2MB 메모리가 있는 스레드를 생성한다.

클라이언트 기반이 성장하면서 더 많은 사용자를 지원하는 웹 애플리케이션을 원한다면 점점 더 서버를 추가를 해야 하기 때문에 비즈니스의 서버 비용, 트래픽 비용, 인건비 기타 등등의 요소가 추가가 더해진다. 위와 같은 문제를 해결하기 위하여 나온 것이 Node.js로 빠르고 확장성 있는 네트웍 어플리케이션을 쉽게 개발할 수 있도록 크롬의 V8 Javascript Engine위에 개발된 서버 사이드 자바스크립트이다.

event-driven과 non-blocking IO모델을 사용하여 다양한 디바이스에서 구동 되며 데이터 집중적이며 실시간성을 요하는 어플리케이션에서 가볍고 효율적으로 동작하도록 해 준다.

기본처리가 비동기방식이라 I/O나 DB 질의 수행되는 라이브러리쪽까지 비동기 처리가 되며 자바보다는 느려도 적은 비용으로 중간 성능을 낼 수가 있다는 장점이 있다.

* 1. **Web Socket 과 Socket.io**

상호작용하는 웹 서비스를 위해 숨겨진 프레임(Hidden Frame)을 이용한 방법이나 Long Polling, Stream 등 다양한 방법을 사용했다. 그러나 이러한 방식은 브라우저가 HTTP 요청를 보내고 웹 서버가 이 요청에 대한 HTTP 응답를 보내는 단방향 메세지 교환 '규칙'을 변경하지 않고 구현한 방식이다. 그렇기 때문에 상호작용하는 웹 페이지를 복잡하고 어려운 코드로 구현해야 했다.

보다 쉽게 상호작용하는 웹 페이지를 만들려면 브라우저와 웹 서버 사이에 더 자유로운 양방향 메시지 송수신(bi-directional full-duplex communication)이 필요하다.

그래서 HTML5 표준안의 일부로 WebSocket API(이후 WebSocket)가 등장했다.

표준 WebSocket의 API는 W3C에서 관장하고, 프로토콜은 IETF에서 관장한다. 그리고 WebSocket은 다른 HTTP 요청과 마찬가지로 80번 포트를 통해 웹 서버에 연결한다. HTTP 프로토콜의 버전은 1.1이지만 다음 헤더의 예에서 볼 수 있듯이, Upgrade 헤더를 사용하여 웹 서버에 요청한다. 당연한 이야기겠지만 클라이언트인 브라우저와 마찬가지로 웹 서버도 WebSocket 기능을 지원해야한다.

브라우저는 "Upgrade: WebSocket" 헤더 등과 함께 랜덤하게 생성한 키를 서버에 보낸다. 웹 서버는 이 키를 바탕으로 토큰을 생성한 후 브라우저에 돌려준다. 이런 과정으로 WebSocket 핸드쉐이킹이 이루어진다.그 뒤 Protocol Overhead 방식으로 웹 서버와 브라우저가 데이터를 주고 받는다. Protocol Overhead 방식은 여러 TCP 커넥션을 생성하지 않고 하나의 80번 포트 TCP 커넥션을 이용하고, 별도의 헤더 등으로 논리적인 데이터 흐름 단위를 이용하여 여러 개의 커넥션을 맺는 효과를 내는 방식이다. 이런 방식을 사용하기 때문에 방화벽이 있는 환경에서도 무리 없이 WebSocket을 사용할 수 있다.

WebSocket은 다가올 미래의 기술이지 아직 인터넷 기업에서 시범적으로라도 써 볼 수 있는 기술이 아니다. WebSocket이 미래의 기술이라면 Socket.io는 현재 바로 사용 할 수 있는 기술이다. Socket.io는 JavaScript를 이용하여 브라우저 종류에 상관없이 실시간 웹을 구현할 수 있도록 한 기술이다.

Guillermo Rauch가 만든 Socket.io는 WebSocket, FlashSocket, AJAX Long Polling, AJAX Multi part Streaming, IFrame, JSONP Polling을 하나의 API로 추상화한 것이다.

즉 브라우저와 웹 서버의 종류와 버전을 파악하여 가장 적합한 기술을 선택하여 사용하는 방식이다. 가령 브라우저에 Flash Plugin v10.0.0 이상(FlashSocket 지원 버전)이 설치되어 있으면 FlashSocket을 사용하고, Flash Plugin이 없으면 AJAX Long Polling 방식을 사용한다. 개발자가 각 기술을 깊이 이해하지 못하거나 구현 방법을 잘 알지 못해도 사용 할 수 있다. Web Socket과 달리 Socket.io는 표준 기술이 아니고 Node.js 모듈로서 Guillermo Rauch가 CTO로 있는 LearnBoost라는 회사의 저작물이며 MIT 라이센스를 가진 오픈소스이다. 현재 Node.js가 아닌 다른 프레임워크에서 Socket.io를 사용할 수 있도록 하는 시도가 있다.

* 2. **HLS(HTTP Live Streaming)**

라이브 스트리밍을 위한 전통적인 프로토콜로인 RTSP는 도입 비용이 높고 방화벽 환경에서 서비스가 원활하게 이루어지지 않는 단점이 있다. 이러한 단점을 해결하는 방법으로 HTTP를 라이브 스트리밍을 위한 프로토콜로 사용하는 방법이 나오게 되었다.

HLS(HTTP Live Streaming)는 Apple에서 iOS 3.0과 QuickTime X를 위해 2009년에 내놓은 프로토콜이다. 이 프로토콜에서는 스트리밍 데이터를 MPEG-2 Transport Stream에 담아 시간 단위로 잘게 쪼개서 전송한다. 그리고 어떤 파일을 재생해야 하는 지에 대한 정보는 m3u8 파일을 이용하여 플레이어에 전달한다.

HLS는 iPhone과 iPad의 사용자 수가 늘어남으로써 자연스럽게 그 수요가 늘어나게 되었다. 또한, 규격 자체의 단순함과 IETF(Internet Engineering Task Force)를 통한 표준화 작업 등을 통해 다른 업체들도 쉽게 HLS를 지원할 수 있게 했다.

그 결과로, Adobe는 Flash Media Server 4.0에서, Microsoft는 IIS Media Server 4.0에서 HLS를 정식으로 지원하며, 모바일 운영체제에서 상대 진영이라 할 수 있는 Google의 Android에서도 3.0 버전인 Honeycomb부터 HLS를 지원하기 시작했다.

1. **개발 목표**
   1. **안드로이드 스마트 폰 웹 서버 구축**

먼저 타 플랫폼에서 웹을 통해 특정 스마트 폰으로 접근하기 위해서는 접근하려는 폰에 웹 서버가 구축되어 있어야 한다. 스마트 폰에 웹 서버를 구축함으로써 웹 서버를 구동하는 입장에서는 자신의 스마트 폰에 저장 된 데이터들을 선택적으로 외부에 제공해 줄 수 있다.

웹 서버는 안드로이드 애플리케이션에 내장되어 애플리케이션을 설치하게 되면 웹 서버를 구동 할 수 있는 환경을 구성하게 된다.

* 1. **NAT Check를 통한 Http Relay 기능을 수행하는 중계 서버**

NAT Traversal를 구현하려면 중계 서버가 필요하다. 중계 서버가 필요한 이유는 웹 서버를 서비스하는 호스트가 NAT 뒤에 있는 경우(사설 IP) 직접 통신이 불가능 하기 때문이다. 클라이언트(브라우저) 측에서는 특정 스마트 폰 웹 서버가 NAT 뒤에 존재하는 여부를 판별할 수 없기 때문에 중계 서버가 스마트 폰 웹서버로부터 정보를 받아들여 NAT Check를 수행한다. NAT Check를 수행한 뒤 NAT의 존재 유무에 따라 연결 방식을 직접 연결 또는 STUN으로 동작하여 연결이 수립된다.

* 1. **등록 서버 목록 관리**

메인 웹 페이지에서 로그인을 하게 되면 해당 접속자가 등록한 서버 목록을 보여 주고 새로운 서버를 등록하거나 간단한 메세지를 남길 수 있다.

Node.js의 Socket.io를 이용하여 실시간으로 웹 페이지의 접속한 사용자의 유무와 서버상태 유무를 확인 할 수 있다.

* 2. **파일 공유 기능**

웹 서버가 되는 스마트 폰 내에 저장되어 있는 파일(음악, 동영상, 문서 등) 및 정보들을 브라우저를 통해 목록 보기, 다운로드, 업로드 기능을 제공한다.

* 1. **실시간 스트리밍 기능**

스마트 폰으로부터 실시간 스트리밍 서비스를 두 가지 제공한다. 첫 번째로 스마트 폰에 저장 된 동영상 파일을 스트리밍해 브라우저에서 볼 수 있게 한다. 두 번째는 스마트 폰 카메라로 촬영되는 영상을 실시간으로 스트리밍해 라이브 영상을 제공한다.

라이브 스트리밍은 두 가지 방법으로 구현하는데 첫 번째로 Motion-JPEG을 이용한 Push Streaming 방식과 m3u8을 이용한 HLS(HTTP Live Streaming) 방식으로 구현한다.

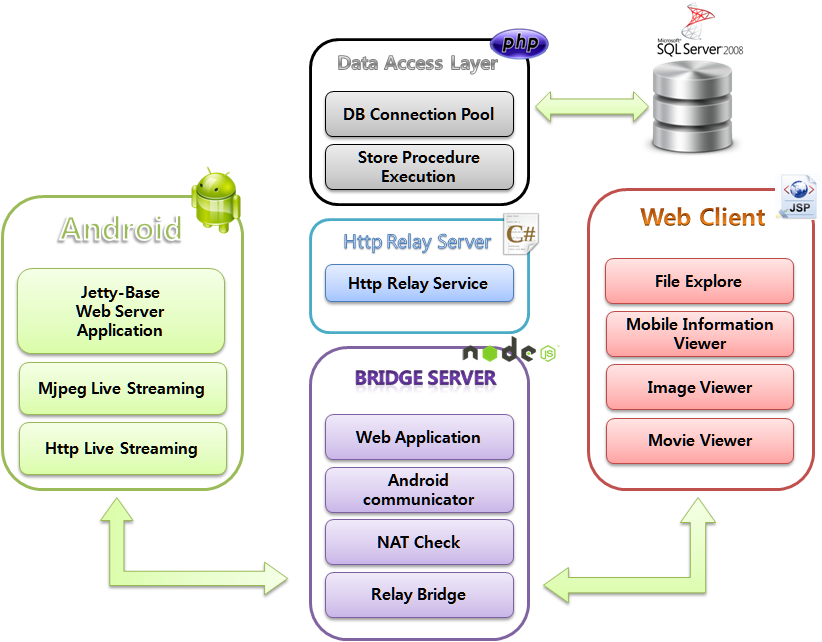
1. **개발 내용**
   1. **System Conceptual Diagram**



[그림 5] 전체 시스템 개념도

전체적인 시스템의 개념도는 위 그림과 같다. 스마트 폰에 웹 서버를 탑재 함으로써 다양한 플랫폼에서는 전용 애플리케이션을 따로 설치할 필요 없이 웹 브라우저를 통해 데이터를 공유할 수 있다.

* 2. **System Architecture**

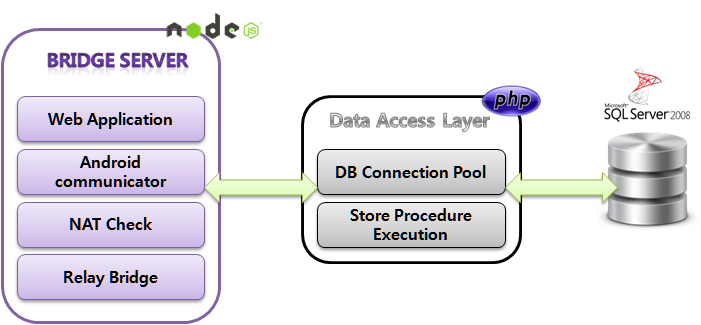


[그림 6] System Architecture

시스템은 크게 안드로이드 웹 서버, 클라이언트(웹 브라우저), 중계 서버, 릴레이 서버, DAL(Data Access Layer) 5가지로 구성된다. 안드로이드 웹 서버는 파일공유와 실시간 스트리밍 기능을 제공한다. 클라이언트(웹 브라우저)에서는 안드로이드 웹 서버에 저장되어 있는 파일 및 정보들의 목록을 볼 수 있고 다운로드, 업로드가 가능하다. 또한 안드로이드 웹 서버에서 스트리밍하는 동영상 및 음악 파일을 재생할 수 있다.

중계 서버에서는 사용자를 관리하고, 사용자간 관계를 맺어 안드로이드 앱 서버로의 접속을 유도하며, 웹 서버 경로상에 NAT 존재 여부와 접속 가능 여부를 판단하는 NAT Check 기능을 제공한다. NAT이 존재한다면 데이터를 중계해서 전달하는 Relay 서버로 접속하여 데이터를 전달한다.

* 2. **Bridge Server**



[그림 7] Bridge Server

Bridge Server는 Node.js로 구현되어 있고, Bridge Server의 데이터베이스 접근 방식은 Data Access Layer(이하 DAL)로 분리하여 접근한다. DAL은 PHP로 구현되어 있으며 데이터베이스와의 데이터교환은 Store Procedure를 이용한다.

Bridge Server 부분에서 DAL로 계층을 분리 함으로서 안드로이드 어플리케이션인

모바일 웹 서버가 데이터베이스로의 접근이 유연하게 하였다.

Bridge Server의 주요 역할은 사용자를 관리하고 친구로 등록된 다른 사용자의 모바일 웹 서버로 접속 할 수 있게끔 유도 하는 기능이다. 이 떄, NAT Check를 수행하여 Private IP인 경우 모바일 웹 서버가 Relay Server로 접속을 할 수 있도록 유도하고, Bridge Server에 접속 중인 사용자에게도 Relay Server의 주소를 전송하여 접속을 할 수 있도록 구현 하였다.

* 2. **Web Server Architecture**

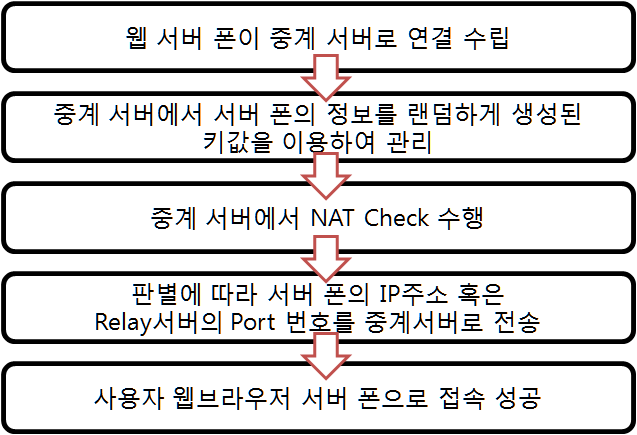


[그림 8] Web Server Architecture

안드로이드 웹 서버는 기본적으로 Jetty 서버 모듈을 안드로이드로 포팅한 뒤 사용한다. 웹 애플리케이션은 HTML5, JavaScript, CSS, Servlet을 사용하여 안드로이드 폰에 저장된 데이터들을 보여 주고 공유하는 기능을 제공한다.

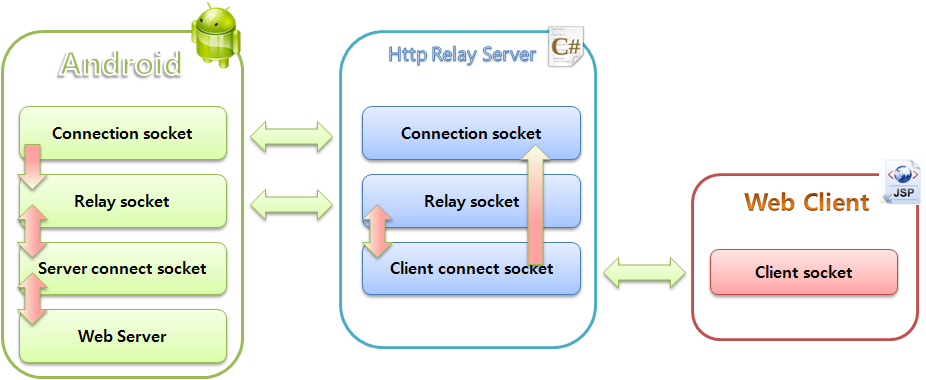
안드로이드 API에서는 Service API를 사용하여 웹 서버가 백그라운드에서 실행이 유지 되도록 한다.

* 1. **Network Connection Scenario**



[그림 9] Network Connection Scenario

* 2. **Http Relay Server**



[그림 10] Http Relay Server

Nat check 후 사설 IP인 경우에 Android의 Connection socket이 대기중인 릴레이 서버의 소켓에 접속을 하게 된다. 접속을 하게 되면 Relay Server에게 포트 번호를 할당 받게 되며 Web Cilent의 연결을 기다리는 소켓이 Relay Server에 생성되며, 연결이 되면 Connection socket을 통해 새로 생성되는 Relay socket의 바인드 된 포트 번호를 Android에서 받아 릴레이 소켓끼리 연결되며 Android에서는 새로운 소켓을 열어 Android Web Server에 접속을 한다. 이렇게 연결된 소켓들은 Web Server와 Web Cilent 사이의 통로가 되어 데이터를 주고받는다.

① Nat check 후 사설 IP로 판정되면 Android의 Connection socket이 대기중인 Relay Server의 소켓에 접속한다.

② Connection socket을 통해 Android로 포트 번호를 할당한다.

③ Relay Server에서는 Web Client의 연결을 기다리는 서버소켓이 생성된다.

④ 연결되면 Relay socket이 생성되고 Connection socket을 통해 Relay sokcet이 바인드 된 포트번호를 Android에 전해준다.

⑤

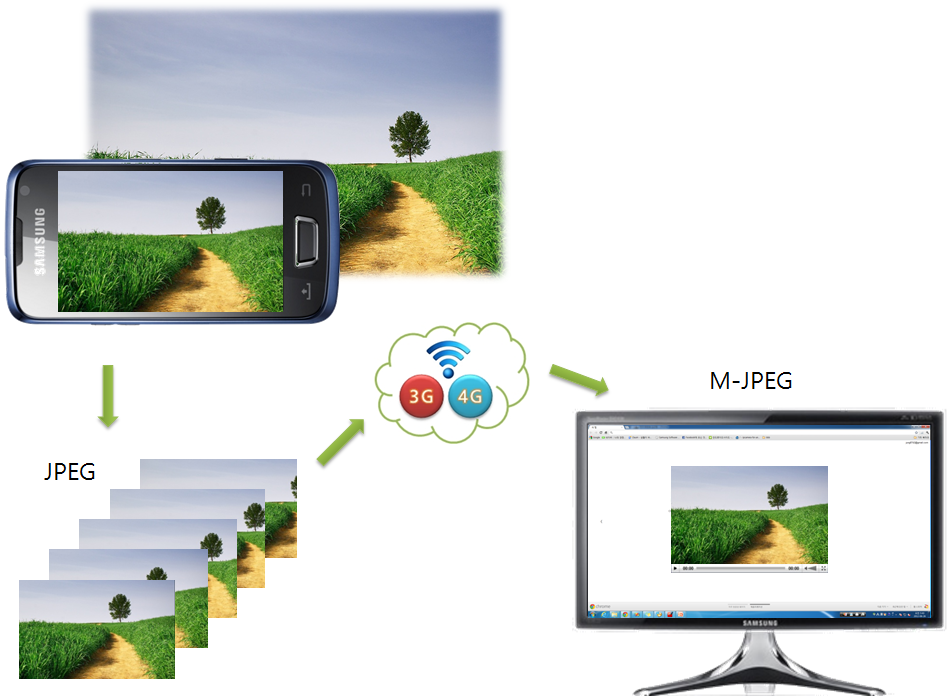
Android Relay socket과 Relay server Relay socket이 연결되며 Android에서 Server connect sokcet이 생성되어 Android Web server에 접속한다.

⑥ Client socket ↔ Client connect socket ↔ Relay socket(Relay) ↔ Relay socket(Android) ↔ Server connect socket ↔ Web server 사이의 연결이 생성되며 각 소켓은 통로역할을 한다.

⑦ Thread를 이용해 Client의 접속이 있을때마다 (3) ~ (6)이 생성되고 종료된다.

※ Relay Server의 포트 번호 중복을 대비하여 미리 사용 가능한 포트번호를 리스트에 담아 둔 뒤 빼서 사용 후 반납하는 방식으로 처리하였다.

* 1. **Live Streaming**
     1. M(Motion)-JPEG 구현



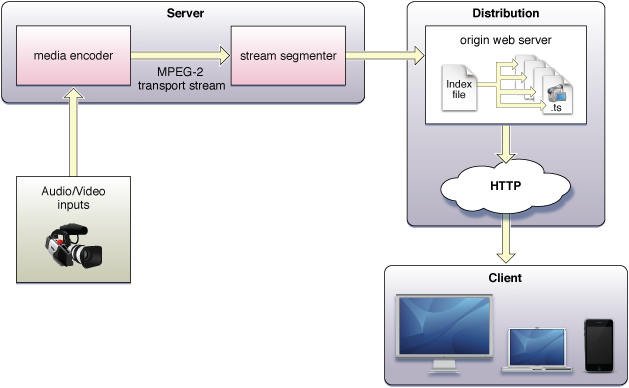
[그림 11] Motion-JPEG Push 과정

M-JPEG는 스마트 폰 카메라의 프리뷰 화면을 정지 화상 압축 방식인 JPEG를 이용해 동영상 압축에 응용한 것으로 M-JPEG의 각 프레임은 JPEG로 압축되어 있다. MJPEG에서 각 프레임은 앞과 뒤의 프레임과 관계없이 단일 프레임으로써 압축하고 동영상 재생정보를 함께 압축하여 MPEG보다 압축률이 더 높다. 또한 압축을 할 때 압축률을 조정할 수 있어 화질 조정이 가능하며 다른 압축 방식에 비해 계산량이 적어 빠르게 압축 할 수 있다.

M-JPEG는 구현이 비교적 간단하고 빠른 퍼포먼스를 보여 준다. 하지만 JPEG를 이용한 영상 압축이기 때문에 오디오 데이터는 함께 압축하지 못 한다. 오디오 데이터를 별도로 압축을 해도 영상과 오디오가 제대로 나오려면 싱크를 맞춰 줘야 한다.

시스템에서는 M-JPEG를 통해서는 영상 정보만 제공하고 영상과 오디오를 함께 압축한 스트리밍은 아래 HLS 방식으로 구현하여 제공한다.

* + 2. HLS(HTTP Live Streaming) 구현



[그림 12] HLS Architecture

스마트 폰 카메라의 영상을 라이브로 스트리밍해 주는 기능으로 스마트 폰에서 웹서버와 HLS 버튼을 활성화 하면 현재 촬영 중인 화면과 소리를 가져온다. FFmpeg을 이용해 H264/AAC 코덱으로 압축하여 mpeg2 transport stream(.ts)파일로 만든다. 만들어진 파일에 대한 메타데이터를 m3u8 에 기록, m3u8을 웹서버를 이용하여 전송하면 브라우저(HLS 지원 브라우저)는 m3u8의 메타데이터를 참고하여 스트리밍 서비스를 이용할 수 있다.

* + - 1. FFmpeg, x264(VideoLAN) 포팅

FFmpeg는 디지털 음성 스트림과 영상 스트림에 대해서 다양한 종류의 형태로 기록하고 변환 할 수 있는 오픈 소스 라이브러리로 C언어로 작성되어 있으며 포팅 순서는 다음과 같다.

1. FFmpeg Down, Android NDK(toolchain) install

현재 FFmpeg 라이브러리는 0.5 버전부터 1.0버전까지 다양하게 제공하고 있으나 우리 프로젝트는 0.8.12 버전을 사용하였다. FFmpeg 라이브러리는 안드로이드를 기준으로 만들어진 라이브러리가 아니라서 버전에 따라 Android NDK를 사용한 빌드를 실패하는 경우가 많았고 Android NDK 버전과 FFmpeg버전을 바꾸어 가며 빌드 해 보았으나 0.8 이전 버전만 빌드가 되어 FFmpeg는 0.8, Android NDK는 6을 사용하였다.

2. FFmpeg Configure

FFmpeg에서는 configure를 통해 사용 용도에 따라 환경 설정을 할수 있는데 configure 설정은 다음과 같다



[그림 13] FFmpeg Configure

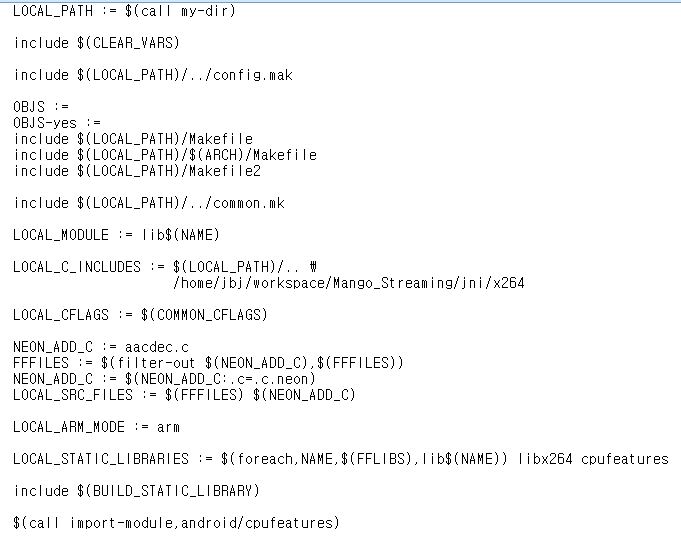
사용하지 않을 동영상 재생 등은 사용하지 않게 하고 h264 외부 코덱을 필요로 하기 때문에 enable libx264항목을 넣었으며 cross-compile를 지정한다.

3. x264 down, configure

FFmpeg에서는 h264코덱을 기본으로 지원하지 않아서 따로 x264 코덱을 외부 참조해야 하기 때문에 VideoLAN에서 제공하는 x264라이브러리를 다운받아 cross-compile이 가능하게 설정한다.

4. Android.mk, ndk-build

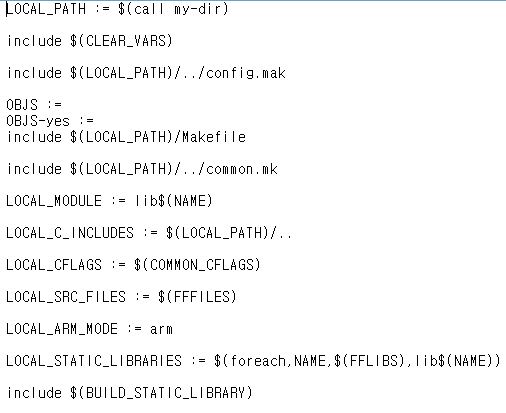
FFmpeg을 안드로이드에 포팅하기 위한 방법은 두 가지가 있는데 하나는 FFmpeg을 configure -> make -> make install을 거쳐 정적(또는 동적) 라이브러리를 생성한 뒤 Android.mk를 통해 참조하는 방법과 configure만을 한후 ndk-build로 바로 빌드하는 방법이 있으나 FFmpeg만 사용하는 경우면 전자의 방법을 써도 됐지만 x264 라이브러리를 FFmpeg이 먼저 참조한 후 그것을 안드로이드에서 FFmpeg을 참조하는 경우에는 FFmpeg가 x264를 제대로 인식하지 못하여 후자의 방법을 사용해 configure만을 거치고 참조 라이브러리 마다 Android.mk파일을 만들어 빌드하였다.



[그림 14] libavcodec

l

libavcodec은 FFmpeg의 코덱관련 파일들로 Makefile, Makefile2와 common.mk를 기준으로 빌드하며 x264 라이브러리에 있는 헤더파일을 include하고 정적 라이브러리로 생성 될 libx264을 참조한다.

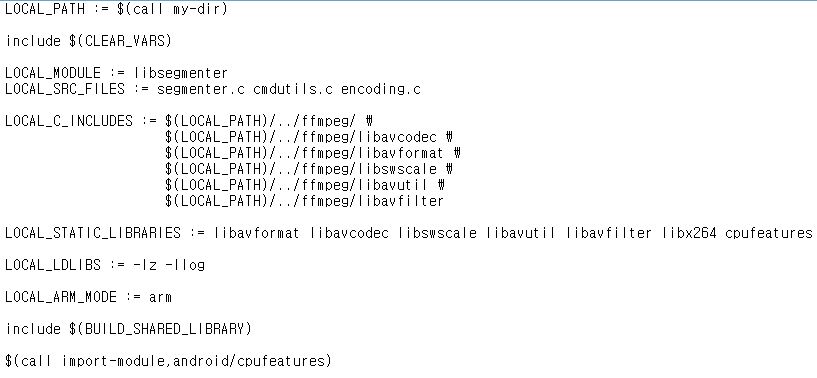


[그림 15] libavfilter, libavformat, libavutil, libswscale



[그림 16] libx264

마지막으로 FFmpeg를 이용하여 encoding, segmenter 하는 C언어 소스를 빌드하기 위한 Android.mk파일이다.



[그림 17] segmenter

앞서 만든 make 파일을 이용해 만들어진 정적 라이브러리를 이용하여 encoding.c와 segmenter.c를 빌드한다.

* + - 1. 영상 및 음성 추출

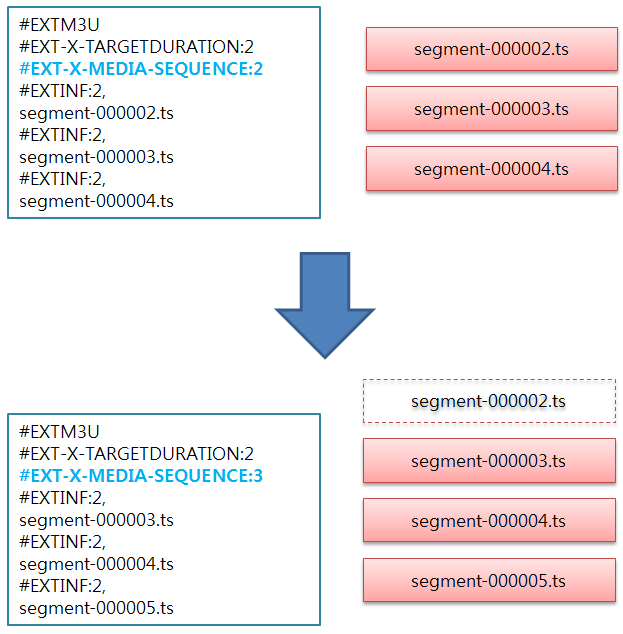
안드로이드에서 현재 촬영중인 미디어의 정보를 얻기 위한 방법은 두가지가 있는데 하나는 media recoder를 이용하여 얻는 방법이고, 나머지 하나는 camera callback을 이용하여 영상을 얻어오고, audio recod를 이용하여 음성을 얻어오는 방법이 있는데 media recoder를 이용하여 얻어 오는 방법은 영상 찍는 것을 멈추는 순간 영상의 moov container를 기록하게 되어 moov container를 따로 기록하거나 그게 없더라도 재생이 가능한 플레이어가 필요하여 두번째 방법을 사용하였다.

* + - 1. mpeg2 transport stream

FFmpeg과 x264 라이브러리를 이용하여 영상은 h264, 음성은 aac 코덱을 이용하여 인코딩한후 영상과 음성을 다중화 하여 mpeg2 transport stream 컨테이너를 씌운다.

* + - 1. m3u8

실시간으로 생성되는 ts 파일의 정보를 m3u8에 기록하며 클라이언트에게 전송한다. 작성규칙은 아래 표와 같으며 m3u8 파일은 실시간으로 그림과 같이 내용이 변한다.



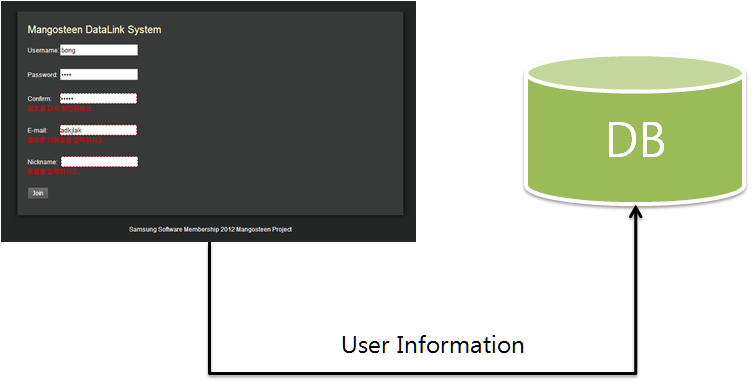
[그림 18] m3u8 파일의 실시간 변화

m3u8에 #EXTINF에 있는 ts파일을 서버로 바로 요청하게 되며 #EXT-X-ENDLIST(종료 지시어)가 없을 경우 m3u8을 다시 요청하게 되는데 그때 m3u8을 정보가 변화되어 새로운 파일이 등록되면 그 파일을 요청하게 되며 실제 영상과는 첫 파일과 마지막 파일의 시간차 + 네트워크 지연만큼 시간이 벌어지게 된다.

|  |  |  |
| --- | --- | --- |
| **지시어** | **형식** | **역할** |
| #EXTM3U | #EXTM3U | 파일의 가장 첫 줄에 명시하여 파일이 m3u8 포맷임을 명시한다. |
| #EXTINF | #EXTINF:  <재생 시간:초>  ,<제목> | 이 지시어의 다음에 명시된 콘텐츠의 재생 시간과 제목을 명시한다. |
| #EXT-X-  TARGETDURATION | #EXT-X-  TARGETDURATION  : <시간: 초> | 파일 목록에 나열된 각 파일의 최대 재생 시간을 명시한다. |
| #EXT-X-  ENDLIST | #EXT-X-ENDLIST | 플레이 리스트에서 재생할 콘텐츠가 더 이상 없음을 의미한다. 이 지시어가 표시된 줄 이후의 콘텐츠는 무시한다. |
| #EXT-X-  DISCONTINUITY | #EXT-X-  DISCONTINUITY | 이 지시어가 표지된 줄을 기준으로 이전 줄과 이후 줄에서 재생하는 콘텐츠의 정보가 변경되었음을 표시한다. 예를 들어 이전 콘텐츠와 이후 콘텐츠의 파일 포맷, 파일이 갖고 있는 미디어 트랙의 개수, 인코딩 정보, 재생 시간 정보 등이 변경되면 이 지시어를 플레이리스트에서 정보가 바뀌는 파일 사이에 명시하여 플레이어가 새로운 정보를 사용해야 하는 시점을 알려 준다.. |
| #EXT-X-  MEDIA-  SEQUENCE | #EXT-X-  MEDIA-SEQUENCE:  <첫 파일의  일련번호> | 제일 먼저 플레이해야하는 파일의 일련번호를 명시한다. 예를 들어 그림 1의 경우와 같이 0,1,2의 파일이 있을 경우 이 지시어의 값은 0이 된다. 이 지시어가 포함되지 않은 경우 첫 분할 파일의 일련 번호는 0으로 간주한다. |
| #EXT-X  -KEY | #EXT-X-KEY:  <암호화 방법>  [, <key>] | 암호화된 파일을 해독하는 키 값을 명시한다. HTTP Live Streaming에서는 재생 시간에 따라 분할된 각 파일을 암호화하여 전송할 수 있다. 암호화된 파일을 해독할 때 필요한 키 값을 플레이어에게 알려 주기 위해 사용한다. |
| #EXT-X  -STREAM-INF | #EXT-X-  STREAM-INF | 이 지시어 다음 줄의 콘텐츠에 대한 정보를 제공한다. #EXTINF는 재생 시간에 대한 정보만 제공하고, #EXT-X-STREAM-INF는 다음과 같은 정보를 제공한다.  - BANDWIDTH: 10진수로 표시한 bps 값  - PROGRAM-ID: 플레이 리스트 파일에 있는 콘텐츠가 갖는 고유 값  - CODEC: 해당 콘텐츠에 적용된 코텍(codec) 정보  - RESOLUTION: 해상도 |

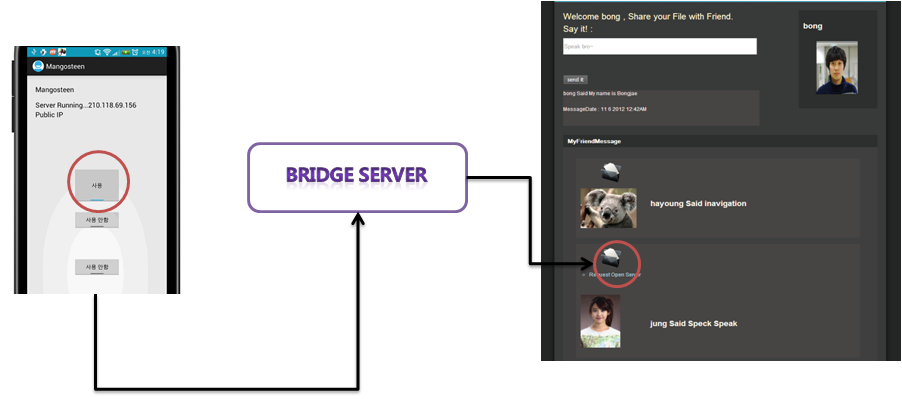
[표 1] m3u8 format infinitive maker

* 1. **Scenario**



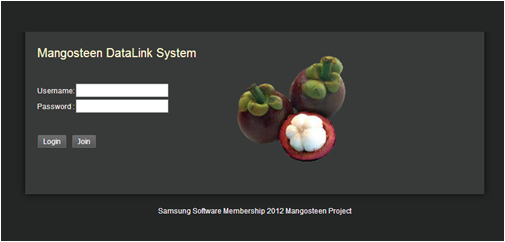
[그림 19] 스마트 폰 웹 서버 등록

중계서버로 접속하여 사용자 정보를 입력하고 가입을 한다. 이 정보는 핸드폰의 모바일 웹 서버와도 정보를 공유하며, Android Preferences에 정보를 등록하여 첫 로그인 성공 시 이 후 로그인 정보를 묻지 않는다.



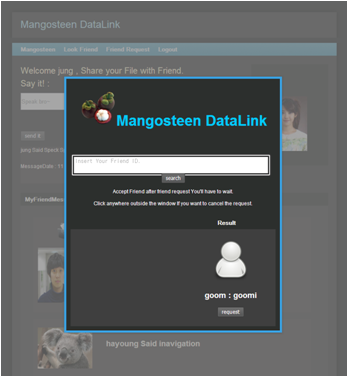
[그림 20] 구동 중인 웹 서버 정보 로드

중계 서버에서는 DB의 정보를 읽어 와 웹 서버 목록을 보여 주고 활성화 중인 서버를 표시해 준다.

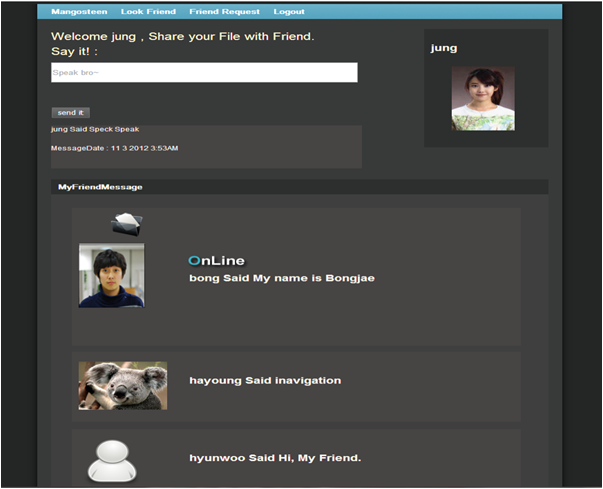


[그림 21] 사용자 인증 웹 페이지

사용자 인증을 위한 웹 페이지 화면이다. 데이터 공유에 사용자 인증이 필요한 이유는 친구 등록을 통하여 접속 가능한 스마트 폰 웹 서버를 관리하기 위함이다.

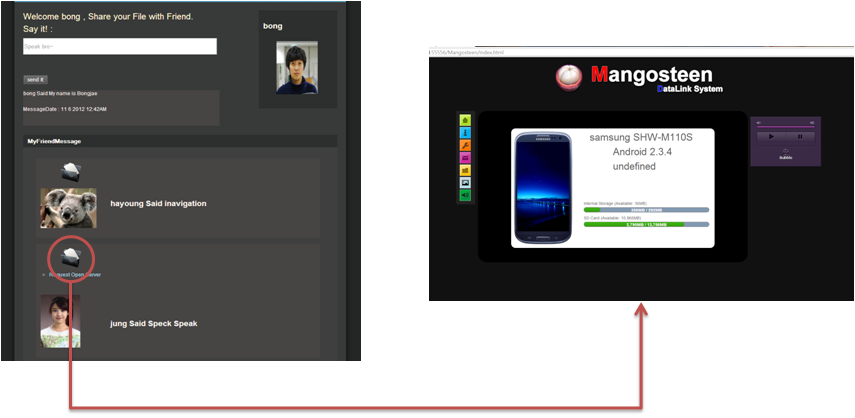


[그림 22] 등록 된 웹 서버 검색



[그림 23] 등록한 웹 서버 목록

사용자가 등록한 친구들의 서버 목록 화면이다. 친구의 간단한 프로필과 데이터를 공유하기 위한 준비가 되어 있는지 알 수 있다. 폴더 모양 아이콘이 보이면 친구의 핸드폰에 웹서버가 열려 있는 상태이므로 접근이 가능하다는 것을 알 수 있으며, 온라인 아이콘으로 보인다면 현재 친구가 웹 페이지에 로그인 중이라는 것을 알 수 있다.



[그림 24] 웹 서버 연결 요청

접근을 원하는 친구의 아이디를 선택하고 파일 공유를 위한 핸드폰 서버의 접속을 하는 화면이다. 서로 친구로 등록된 상태이기 때문에 서버가 동작하고 있다면 제약없이 접근이 가능하다.



[그림 25] 스마트 폰 웹 서버의 파일 목록

친구의 핸드폰에 접근하여 열어 놓은 공유 폴더의 파일이나, 사진, 동영상을 웹브라우저를 통하여 자유롭게 다운로드 가능하고 업로드 가능하다.



[그림 26] 실시간 촬영 영상 데이터 스트리밍

친구 핸드폰의 카메라를 이용하여 실시간으로 촬영된 화면을 웹브라우저 상에서 볼 수 있다.

1. **개발 환경**
   * 개발 환경 : Windows 7, Ubutu 10.04, JDK1.7, Android SDK, Android NDK,

MS-SQL2008R2

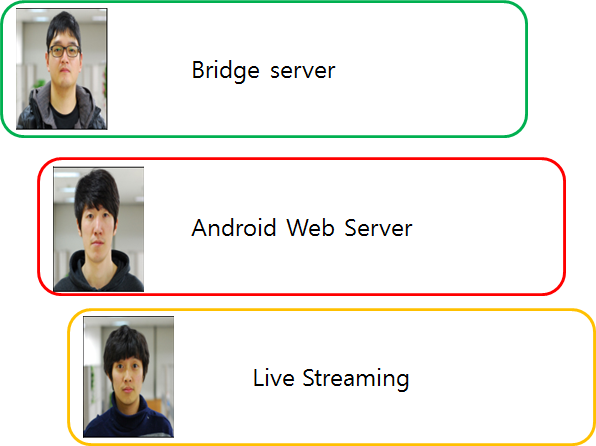
* + 개발 언어 : Java, HTML5, Java Script, Jsp, Node.js, PHP, C#, C
  + 개발 도구 : Eclipse, WebStorm, EditPlus, VisualStdio2010
  + 버전 관리 : SVN

1. **개발 일정**



[그림 27] 개발 일정

1. **역할 분담**



[그림 28] 역할 분담

1. **참고문헌**
   * + 안드로이드 플랫폼에서의 미니 웹 서버 성능 연구 (김태용) - 경북대학교
   * NAT Traversal 블로그 - http://purematter.blog.me/110101403170
   * 안드로이드 프로그래밍 정복 (김상형) - 한빛미디어

• Peer-to-Peer Communication Across Network Address Translators 번역논문

(원작자 : Bryan Ford, 번역자 : 양영욱)

* + 임베디드 웹 서버를 위한 TCP/IP (박종진, 이동은, 이형수) - 에이콘
  + HTTP Live Streaming OverView - APPLE