논문 2011-1-11

스마트 폰을 위한 Wi-Fi 기반 모바일 게임 앱의 설계

Design of Wi-Fi based Mobile Game App for a Smart Phone

오선진*

Sun-Jin Oh

요 약 최근 스마트 폰 기술의 급속한 발전에 힘입어 모바일 컴퓨팅 환경에서의 온라인 게임의 설계에 대한 관심이 높아지고 있다. 그러나 모바일 단말인 스마트 폰은 비교적 낮은 성능의 프로세서, 저해상도의 GUI, 작은 메모리 공간과 짧은 배터리 파워 등의 제한으로 온라인 게임 구현에 많은 제약이 따른다. 따라서 대부분의 게임은 온라인이나 멀티 플레이 기능이 매우 제한적이다. 본 논문에서는 이러한 모바일 환경의 제약을 극복할 수 있는 컴포넌트 기반 스마트 폰 환경에서의 효율적인 모바일 온라인 게임 앱을 설계하고 구현하였다. 특히 본 논문에서 구현한 모바일 게임은 Wi-Fi를 기반으로 게임 서버와 다른 스마트 폰 간의 온라인 게임이 가능하도록 구현하였다.

Abstract With the rapid growth of recent smart phone technology, the interests for the design of online game in mobile computing environment are highly focussed on with great attention. Smart phone as a mobile terminal device, however, has many restrictions for implementing online mobile game since the limitation of relatively low performance of a processor, low resolution of GUI, small memory spaces, and short battery power. Therefore, most of games are very restricted on online and multi-play functions. In this paper, we design and implement mobile online game app in component based smart phone environment in order to take over these restrictions in mobile environment. Especially, the implemented mobile game is able to play online game among Wi-Fi based game server and another smart phone.

Key Words : 스마트 폰, Wi-Fi, 컴포넌트 기반 모바일 온라인 게임 앱

Ⅰ. 서 론

스마트 폰 기술의 급속한 발전과 더불어 모바일 환경에서 시간과 장소에 구애 없이 자유롭게 이동하면서 Wi-Fi로 인터넷에 접속하여 컴퓨팅 할 수 있는 다양한 응용들이 절실히 요구되고 있다. 최근 각광을 받는 대표적인 모바일 장비로 i-phone이나 android 폰과 같은 스마트 폰이 주류를 이루고 있으며 지금까지 이를 위해 개발된 대부분의 모바일 응용들은 단순한 오피스 관련 응용 프로그램에서부터 인터넷 접속, 이메일 검색, LBS기반 응용, 그리고 GPS나 Bluetooth 응용 등으로 급속히 발전하고 있으나, 아직은 비교적 단순하고 일률적이어서 마니아들의 욕구를 충분히 충족시키기에는 부족한 실정이다.[1-3] 요즘 스마트 폰 사용자들 사이에서의 가장 큰 관심의 대상은 단연 Wi-Fi 기반 온라인 응용과 게임이다. 하지만 이러한 온라인 응용이나 게임들은 모바일 장비의 처리기나 GUI 환경, 무선 통신과 메모리 및 전원 등의 제약으로 인해 그 개발이 매우 제한적이다.[4, 5] 따라서 이렇게 제한된 자원을 가지고 무선 모바일 환경에 맞는 효율적인 온라인 응용이나 게임 앱의 개발이 절실히 요구된다.

본 논문에서는 이러한 최근의 스마트 폰을 중심으로 하는 정보 인프라와 더불어, 자유롭게 이동하면서 네트워크에 접속하여 컴퓨팅 할 수 있는 모바일 컴퓨팅 환경에서의 Wi-Fi를 기반으로 하는 효율적인 모바일 온라인

*종신회원, 세명대학교 정보통신학부 교수
접수일자: 2010.11.22 수정일자: 2011.1.11
게재확정일자: 2011.2.11

게임 앱을 설계하고 구현하였다. 본 논문에서 설계하고 구현한 온라인 게임 앱은 모바일 단말이나 컴퓨팅 환경의 제약을 극복하면서 무선 통신 환경에서 효율적으로 온라인 게임을 수행할 수 있는 통신 모델을 구축하기 위해 스마트 폰들 사이에 온라인 게임을 관리하고 주관하는 게임 서버를 두고, Wi-Fi를 통해 온라인 상태에서 다수의 게임자가 스마트 폰을 이용해 실시간으로 네트워크에 접속하여 게임을 할 수 있는 멀티 플레이가 기능하도록 클라이언트-서버 구조로 설계하였으며, 온라인 게임은 주요 모듈별로 컴포넌트 화 하여 대부분의 게임 모듈은 게임 서버에 두고 꼭 필요한 컴포넌트만을 스마트 폰으로 다운로드하여 사용하게 하여 클라이언트와 서버간의 통신량을 최소화 하도록 노력하였고, 대부분의 게임 데이터는 게임 서버의 데이터베이스에서 저장 관리하도록 하였으며, 게임자들 간에도 Wi-Fi로 필요한 게임 정보들을 실시간으로 송수신 하도록 구현하였다.

본 논문의 구성은 다음과 같다. 2장에서는 Wi-Fi 기반 온라인 게임을 위한 시스템 모델을 살펴보고, 3장에서는 본 논문에서 설계하고 구현한 Wi-Fi 기반 모바일 온라인 게임의 구조와 알고리즘을 서술하였으며, 4장에서는 Wi-Fi 기반 모바일 온라인 게임의 구현과 실행 결과를 소개한다. 마지막으로 5장에서 향후 연구 과제와 함께 결론을 맺는다.

II. 시스템 모델

오늘날 모바일 컴퓨팅 환경에서의 모바일 단말 장치의 프로세스 처리 능력은 과거에 비해 많이 향상되었으나 데스크 탑 컴퓨터와 비교하면 아직도 성능이 많이 떨어지고, 휴대성을 고려하여 소형 경량화 하다보니까 입출력 장치와 GUI의 크기나 해상도, 그래픽 처리 능력이 떨어지며, 저장 장치 용량 역시도 상대적으로 매우 작고 제한적이다. 또한 모바일 사용자들의 이동성을 위해 주요 전원으로 배터리를 사용하기 때문에 이동 단말 장치의 전원에 대한 많은 제약이 따르며, Wi-Fi의 낮은 대역폭을 이용하여 데이터를 송수신해야 하므로 전송지연과 처리 능력 그리고 데이터에 대한 신뢰도가 떨어진다.[6, 7] 따라서 이러한 모바일 단말과 컴퓨팅 환경의 비효율성을 고려하여 무선 모바일 환경에서 효율적인 통신 모델을 구축하기 위해 전체 시스템을 중앙에서 관리할 수 있도록 하나의 게임 서버를 두고 게임 전반에 대한 처리와 데이터 관리를 담당하게 하고, 여러 클라이언트들이 여기에 Wi-Fi를 통해 무선으로 접속하여 게임과 통신을 하는 클라이언트-서버 구조 형태로 구성하였으며, 클라이언트-서버 간 데이터 전달을 위해 서버인 데스크 탑 컴퓨터와 모바일 단말 간 Wi-Fi망을 통해 실시간으로 통신하도록 하였다.[8, 9] 온라인 게임 모듈에 대한 처리 및 통신 효율성을 높이기 위해 응용 프로그램을 주요 기능에 따른 모듈별로 컴포넌트 화 하였고, 대부분의 게임 모듈들은 게임 서버에 두고 꼭 필요한 컴포넌트만을 스마트 폰으로 다운로드하여 사용하게 하여 클라이언트-서버 구조에 의한 분산 배치를 통해 클라이언트와 서버간의 통신량을 최소화 하도록 노력하였고, 게임자들 간에도 Wi-Fi로 필요한 게임 정보들을 실시간으로 송수신 하도록 구현하여 이러한 모바일 단말과 컴퓨팅 환경이 갖는 제약을 극복하도록 하였다. 다음의 그림 1은 본 논문에서 제안한 모바일 컴퓨팅 환경에서의 스마트 폰을 이용한 Wi-Fi 기반 온라인 게임 앱을 위한 시스템 모델을 보여준다.[8, 9]

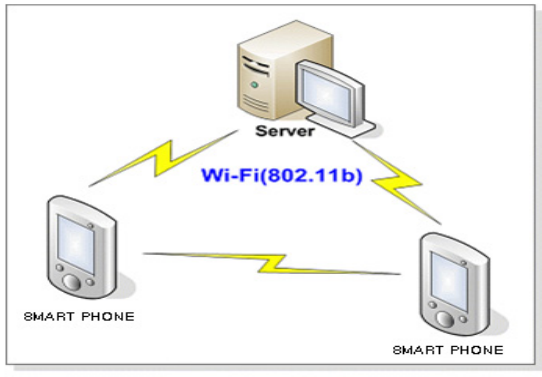

그림 1. 시스템 모델
Fig. 1 System Model

솔라리스 기반 게임 서버의 기능은 APM (Aphache-PHP-MySQL)을 사용, 인터넷을 통한 안정성, 확장성, 보안 및 관리 효율성이 뛰어난 통합 웹 서버 기능을 제공한다. 또한, 데이터베이스는 MySQL을 사용하여 높은 안정성과 완벽한 웹을 지원한다. 게임 서버에서는 스마트 폰의 작은 액정 화면을 통해 인터넷을 접속할 수 있는 전용 홈페이지를 구축하기 위해 240×320 크기의 화면으로 구성하였다. 게임 서버에 회원 가입과 로그인을 통해 클

라이언트는 게임 어플리케이션을 다운로드하여 스마트 폰에 설치한 후 사용할 수 있으며, 게임과 관련된 갱신된 정보들은 게임서버의 데이터베이스에서 저장 관리하게 된다.[8, 10]

클라이언트 환경은 사용자가 사용하는 스마트 폰의 소프트웨어 플랫폼에 맞게 자바 기반의 안드로이드 스마트 폰인 갤럭시S 폰을 사용하였고, 게임 서버에 Wi-Fi를 통해 무선 환경에서 쉽게 접속할 수 있도록 설계하였으며, 클라이언트에서는 무선으로 인터넷 게임 홈페이지에 접속하여 게임 어플리케이션을 다운로드 받아 사용할 수 있도록 하였다.

Ⅲ. 컴포넌트 기반 모바일 게임 앱의 설계

모바일 게임의 주요 기능들은 무선 모바일 환경에서 효율적인 통신 모델을 구축하기 위해 온라인 게임 모듈에 대한 처리 및 통신 효율성을 높일 수 있도록 응용 프로그램을 주요 기능에 따른 모듈별로 컴포넌트 화 하였고, 대부분의 게임 모듈들을 게임 서버에 두고 꼭 필요한 컴포넌트만을 스마트 폰으로 다운로드하여 사용하게 하여 클라이언트-서버 구조에 의한 분산 배치를 통해 클라이언트와 서버간의 통신량을 최소화 하도록 노력하였으며, 게임자들 간에도 Wi-Fi로 필요한 게임 정보들을 실시간으로 송수신 하도록 구현하였다. 게임 서버의 주요 컴포넌트는 사용자 관리, 통신 관리, 자료 관리, 시스템 관리 등으로 구성된다. 사용자 관리 컴포넌트는 사용자 계정에 대한 관리를 하며, 각 사용자의 캐릭터 이름과 아이디 등은 실제 데이터베이스에 저장되지만 이들에 대한 관리는 여기서 한다. 또 여러 사용자가 동시에 접속 했을 경우 처리도 이곳에서 이루어진다. 통신 관리 컴포넌트는 서버가 클라이언트와 통신하는 곳으로 클라이언트와 패킷을 서로 주고받는 부분을 담당한다. 자료 관리 컴포넌트는 온라인 게임이 운영되는데 필요한 주요 정보들을 데이터베이스를 이용하여 저장하고 관리하는 부분으로 여기에 모든 게임 데이터들이 담겨져 있으며 시스템 관리 컴포넌트에 의해 조정된다. 시스템 관리 컴포넌트는 다른 세 컴포넌트를 제어하는 부분으로 소켓에서 메시지를 분석하고, 클라이언트와 통신이 원활하게 이루어지는지 모니터링하며 또한 접속을 강제로 종료하는 등의 작업을 수행한다. 그리고 해당 작업들에 의해 데이터베이스에서 추가, 수정, 삭제가 이루어질 경우 시스템 관리 컴포넌트에서 이 작업을 수행한다.

클라이언트의 주요 컴포넌트는 크게 GUI 처리, 소켓 처리, 게임의 운영과 진행을 처리하는 게임 처리 그리고 이들 컴포넌트를 관리하는 Agent 컴포넌트 등으로 구성된다. GUI 처리 컴포넌트는 게임의 진행상황 등의 그래픽 처리 부분을 담당하는 곳으로 여기엔 게임의 진행 동작을 보이게 하는 부분이 들어있다. 소켓 처리 컴포넌트는 게임서버로 패킷을 보내거나 또는 패킷을 받는 작업을 하는 부분으로 이루어져 있으며, 수신 패킷에 대한 정보는 Agent 컴포넌트에서 해독하여 해당 작업에 넘겨주게 된다. 게임 처리 컴포넌트는 게임의 진행 상황을 결정하고 패의 진행 상황과 변화를 처리하는 부분이다. Agent 컴포넌트는 이들 컴포넌트들을 관리하여 상호 유기적으로 동작하게 하는 부분인데, 특히 게임 서버로 부터 받은 메시지를 분석하고 이에 맞는 부분에 해당 작업을 보내는 일을 수행하는 것으로 클라이언트의 총체적인 관리를 담당하고 있다. 그림 2는 본 논문에서 설계한 Wi-Fi 환경에서의 컴포넌트 기반 온라인 게임의 전체 시스템 컴포넌트 구성도를 보여준다.

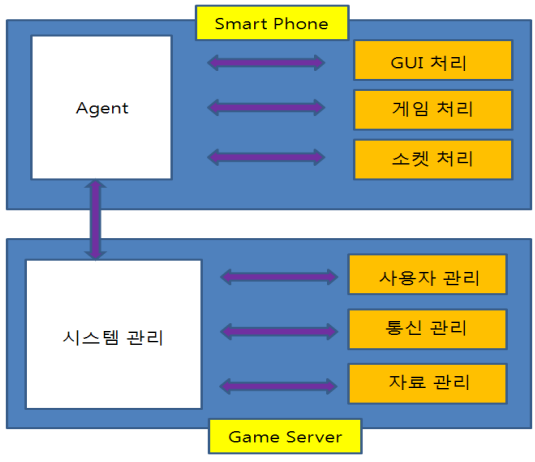

그림 2. 온라인 게임의 컴포넌트 구성도
Fig. 2 Component diagram for online game

본 논문에서는 Wi-Fi 환경에서의 컴포넌트 기반 온라인 게임의 구현 예로 고-스톱 게임을 설계하여 구현하였다.[8, 9] 그림 3은 본 논문에서 구현한 고-스톱 게임의 주요 모듈들의 구성도를 보여 준다. 그림에서 보는 바와 같

이 응용 모듈은 크게 6개의 모듈로 구성되며 로그인 소켓은 클라이언트가 게임서버에 접속할 때 사용되는 모듈로 사용자의 ID와 비밀번호 정보를 데이터베이스에서 가져와 인증하거나 업데이트하게 되며, LoginMain, SocketMain, 그리고 DBMain 모듈과 연계하여 동작한다. 클라이언트로부터 사용자 ID와 Password를 입력받아 로그인과 로그아웃 과정을 수행하게 되는데 로그인은 DB에 질의를 통해 해당 정보를 가져와 클라이언트에게 전송하며, 로그아웃은 클라이언트로부터 변경된 정보를 받아 DB에 정보를 갱신한다.

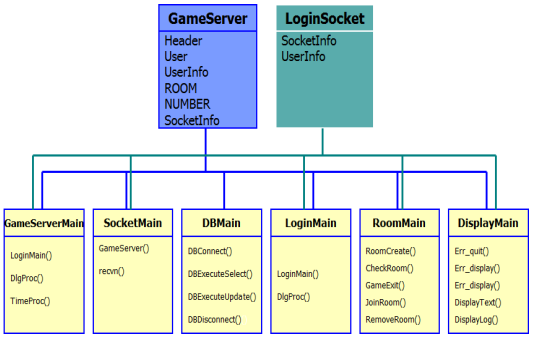

그림 3. 고-스톱 온라인 게임의 주요 모듈 구성도
Fig. 3 Major modules of Go-Stop online game

한편 게임 서버 모듈은 클라이언트에서 게임을 시작하면서부터 발생되는 이벤트들을 처리하는 모듈로, GameServerMain, SocketMain, 그리고 게임을 운영하기 위한 작업 영역을 확보하고 게임의 진행과 순서 및 승패 등을 인공지능 관련 함수 및 플래그로 처리하고 관리하는 RoomMain 게임처리 모듈과 주로 게임의 그래픽 처리 부분을 담당하고 고-스톱 게임의 플래그와 게임 관련 함수를 가지고 처리하는 DisplayMain 모듈 등으로 구성된다.[11, 12]

클라이언트는 스마트 폰으로 모바일 단말과 컴퓨팅 환경의 제약사항들을 극복할 수 있도록 모바일 환경에 최적화되게 설계하였다. 클라이언트는 주요 Game 모듈로 CommandMain과 SocketMain 등으로 구성되어 있다. 클라이언트는 게임 서버와 로그인을 한 후 두 개의 쓰레드를 통해 게임서버와 데이터를 송·수신한다.[11] 수신된 데이터는 분석과정을 거쳐 해당 메시지에 맞게 처리한다. 게임 서버와의 패킷 교환은 통합된 무선 랜 프로토콜 표준인 802.11b 기반 Wi-Fi를 통해 무선으로 소켓을 사용하여 이루어지며 그림 4에서 보인바와 같이 게임 서버에서는 IPv4 인터넷 프로토콜 기반의 TCP 소켓을 생성하여 소켓 초기화와 연결 알고리즘을 통해 클라이언트와 연결하여 통신하는 I/O 모델을 사용하였다.

```
// IPv4의 인터넷 프로토콜 기반의 TCP 소켓 생성
if (S가 잘못된 소켓이라면)
    // 에러메시지 출력 후 종료
end if
// 소켓 정보에 소켓 추가
                    (중    략)
// accept 이벤트
// 네트워크 이벤트의 에러유무체크
if (에러코드가 0이 아니라면)
    // 에러 디스플레이
    continue
end if
                    (중    략)
// 수신
// R은 recvn()으로 설정
if (R이 에러가 아니면)
        // 화면출력
        // 클라이언트 소켓 정보 삭제
        continue
end if
```

그림 4. 게임서버에서 소켓 초기화 및 연결 알고리즘
Fig. 4 Socket initialization & connection algorithm in Game server

그림 5는 게임서버에 있는 데이터베이스의 ER 다이어그램을 보여준다. 그림에서 보인바와 같이 고-스톱 게임을 위한 보드와 패에 대한 정보와 게임 사용자에 대한 정보 그리고 게임 유형과 진행에 따른 게임 정보를 저장 관리할 수 있도록 구성되었다. 그림 6은 클라이언트의 스마트 폰 상에서 시스템의 초기화가 이루어지고 온라인 고-스톱 게임을 시작하기 위해 게임 서버에 Wi-Fi를 이용하여 소켓 연결을 하는 알고리즘을 보여준다. 소켓 통신을 위해 입출력 스트림을 개행하고 쓰레드를 생성하여 지정된 IP주소를 통해 접속을 시도하여 연결한 후 게임 관련 정보를 송·수신하게 된다. 그림 7은 고-스톱 게임을 진행할 때 클라이언트 중에서 게임 순서를 정하고 이때 클라이언트가 둔 패에 대한 내용을 보여주고 게임 진행을 주관하는 알고리즘이다.

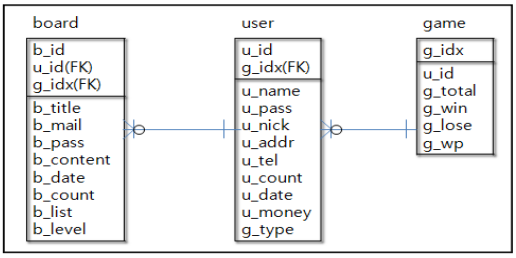

그림 5. 게임서버 데이터베이스의 ER 다이어그램
Fig. 5 ER diagram in game server database

```
package net.npaka.socketex;
importandroid.app.Activity;
importandroid.os.Handler;
importandroid.view.Window;
importjava.io.InputStream;
importjava.io.OutputStream;
importjava.net.Socket;
//소켓 통신 준비
publicclassSocketExextendsActivity
implementsView.OnClickListener{
    private final static String BR=//소켓 개행
        System.getProperty("line.separator");
    private final static String IP="127.0.0.1";//IP 주소 지정
    private SocketEx current;
    private Socket socket; //소켓 정의
    private InputStream in; //입력 스트림 정의
    private OutputStream out; //출력 스트림 정의
    private final Handler handler=new Handler();//핸들러 정의
    //초기화 작업 시행
    @Override
        public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        current=this;
        //스레드 생성
        (new Thread(){public void run() {
            try {
                connect(IP,8080);
             } catch (Exception e) {
             }
        }}).start();
    }
    //연결 접속 과정
private void connect(String ip,int port) {
    int size;
    byte[] w=new byte[1024];
    txtReceive="";
    try {
        //소켓 접속 시도
        socket=new Socket(ip,port);
        in =socket.getInputStream();
        out=socket.getOutputStream();
        //수신을 위한 반복 루프
        while (socket!=null && socket.isConnected()) {
            //데이터 수신
            size=in.read(w);
            if (size<=0) continue;
            txtReceive=new String(w,0,size,"UTF-8");
            //핸들러에 의한 사용자 인터페이스 조작
            handler.post(new Runnable(){
                public void run() {
                    lblReceive.setText(
                        lblReceive.getText()+txtReceive+BR);
                }
            });}
    } catch (Exception e) {
        handler.post(new Runnable(){
            public void run() {
            SocketEx.showDialog(current,"","통신 에러.");
            }
        });}
}
```

그림 6. 클라이언트의 Wi-Fi기반 소켓 연결 알고리즘
Fig. 6 Wi-Fi based Socket connection algorithm in Client

```
void Change_Turn() // 고-스톱 게임 차례 결정
{
    if(State.Now_Turn == BOARD_RED) { // 고-스톱 선의 차례 경우
        IncRedCnt();
        State.Now_Turn = BOARD_BLUE;
    }
    else { // 고-스톱 후의 차례 경우
        IncBlueCnt();
        State.Now_Turn = BOARD_RED;
        if(State.Mode == AI)    // 고-스톱 게임 AI를 선택한 경우
            AIMain(); // 고-스톱 진행 결정 인공지능
알고리즘
    }
}
// 중략
for(i=0;i<14;i++) {
    for(j=0;j<14;j++) {    // 바탕화면을 그린다
        if(Game.GBoardData[i][j]==BOARD_BLURE) {
            TransparentBlt(hdc,i*17+2,j*17,14,14,
            GMemDCWhiteStone,0,0,14,14,RGB(0,255,255));
        }    // 고-스톱 패돌을 그린다
        else if(Game.GBoardData[i][j]==BOARD_RED) {
            TransparentBlt(hdc,i*17+2,j*17,14,14,
            GMemDCBlackStone,0,0,14,14,RGB(255,255,255));
        } // 고-스톱 상대 패돌을 그린다
    }
}
```

그림 7. 고-스톱 게임 순서 결정 알고리즘
Fig. 7 Change turn determination algorithm in Go-Stop game

Ⅳ. 구현 결과

이 장에서는 본 논문에서 제안한 스마트 폰을 위한 Wi-Fi 환경에서의 컴포넌트 기반 온라인 게임 앱인 고-스톱 게임을 설계하고 구현한 결과를 보여 준다. 그림 8은 본 논문에서 구현한 게임 앱에서 게임 관련 데이터를 송·수신하기 위해 사용한 패킷의 구조를 보여준다. 스마트 폰 기반의 앱 특성 상 게임 서버와 무선으로 송 수신하는 데이터의 량을 최소화하기 위해 패킷 설계는 최소 크기로 하였다. 사용한 패킷의 총 길이는 전송 속도 및 지연을 효율적으로 관리하기 위해 60 바이트로 제한하였다. 패킷은 전송이 이루어지면 적절한 처리를 위해 검사를 받게 되는데, 이 패킷이 일반 게임 운용과 작업 처리를 위한 데이터를 가지고 있는지 아니면 단순히 채팅을 위한 데이터인지를 판별한 후 해당 작업을 수행하게 하도록 하였다.

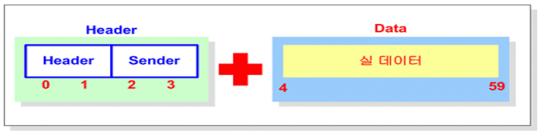

그림 8. 데이터 송 수신에 사용한 패킷 구조
Fig. 8 Packet structure used in data transfer

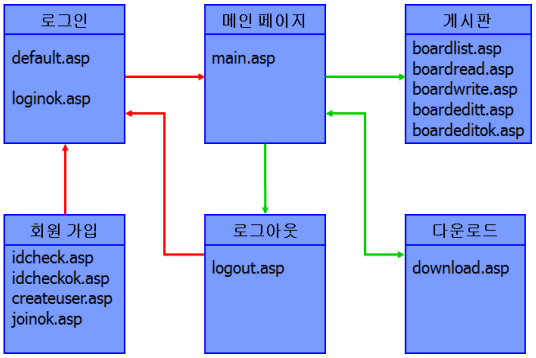

그림 9. 게임서버에서의 웹페이지 운영도
Fig. 9 Web page operation diagram in game server

그림 9는 본 논문에서 구현한 온라인 고-스톱 게임을 위한 게임 서버의 웹페이지 구성과 그 기능을 보여주고 있다. 클라이언트는 스마트 폰으로 Wi-Fi를 이용해서 게임서버에 로그인 과정을 통해 접속하여 게임 응용 프로그램을 다운로드 받아 설치한 후 고-스톱 게임을 할 수 있다. 그림 10은 온라인 고-스톱 게임을 다운로드 받기 위해 게임서버에 로그인하는 화면을 보여준다. 여기서 클라이언트는 로그인을 통해 사용자 인증을 받고 게임서버에 연결하여 클라이언트용 게임 앱을 다운로드 받을 수 있으며 이를 스마트 폰에 설치한 후 온라인 게임을 실행할 수 있다. 고-스톱 게임은 두 사람이 하는 맞고 게임과 정상적으로 수행하는 고-스톱 게임으로 구성되며 클라이언트가 임의로 설정할 수 있으며 게임 상대가 없는 경우에는 게임서버와 고-스톱게임을 할 수 있는 혼자하기 모드로 연결 설정을 할 수도 있다. 이렇게 게임 모드와 연결 설정이 끝나면 그림 11에 보인바와 같이 새로운 온라인 고-스톱 게임의 시작 요청을 할 수 있으며 이때 같이 게임을 할 상대와 창을 통해 서로 정보를 교환할 수 있고 채팅을 통해 대화할 수도 있다. 그림 12는 온라인 고-스톱게임이 시작되어 상대와 게임이 진행되고 있는 화면을 보여준다.

그림 10. 로그인 화면
Fig. 10 Login screen

그림 11. 정보 화면
Fig. 11 Info. screen

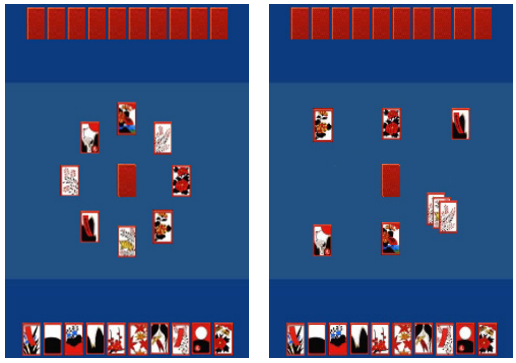

그림 12 고-스톱 게임 시작과 진행 화면
Fig. 12 Go-Stop game start & play screen

V. 결 론

최근 모바일 컴퓨팅 기술과 무선 통신 기술의 급속한 발전과 더불어 스마트 폰 기술과 응용도 급속히 발전 보급되고 있다. 그러나 스마트 폰을 위한 응용이나 게임들은 장비나 환경의 제약으로 매우 제한적이다. 본 논문에서는 스마트 폰을 위한 Wi-Fi 환경에서의 컴포넌트 기반 온라인 모바일 게임 앱을 설계하고 구현하였다. 스마트 폰의 플랫폼 상의 제한 속에서 효율적인 온라인 서비스를 위해 클라이언트-서버 구조 기반으로 구현하였으며 게임 서버를 두고 효율적으로 동작할 수 있도록 컴포넌트 단위로 설계 구현하였다. 구현된 컴포넌트들은 게임 서버와 클라이언트에 분산 동작하게 하여 서로 교환하는 패킷 량을 최소화하도록 하여 모바일 환경에서 효율적으로 게임이 운영되게 설계하였다.

게임 서버는 4개의 컴포넌트로 구성하여 사용자 관리,

통신관리, 자료관리 그리고 시스템 관리를 하게 하였고, 게임을 수행하는 스마트 폰 단말에서도 4개의 컴포넌트로 나누어 GUI 처리, 게임 처리, 소켓 처리, 그리고 Agent 컴포넌트로 게임을 진행하고 Wi-Fi를 통해 게임 정보를 온라인으로 송·수신 하면서 원활하게 게임이 진행되도록 하였다. 무선 통신에서 클라이언트 부분은 소켓의 제한으로 인해 블로킹 모드로 구현하였으며, 이에 따른 문제점을 극복하기 위하여 송·수신 소켓 프로세스를 스레드로 나누고 시간에 따라 강제로 종료하고 재시작하는 방법으로 문제를 해결하였고 게임서버는 논 블로킹 모드로 구현하였다. 구현한 온라인 고-스톱 게임을 실행한 결과 게임서버를 중심으로 원활한 온라인 기능이 수행됨을 확인할 수 있었고, 싱글플레이 뿐만 아니라 멀티플레이도 무난히 수행할 수 있음을 확인할 수 있었다.

향후 연구과제로는 게임을 하는 클라이언트들 간 게임 정보 교환을 게임서버를 거치지 않고 Wi-Fi의 부하를 늘리지 않으면서 직접 블루투스 등의 무선장치를 이용하여 근거리에서 게임이 진행되는 동안 실시간으로 통신하게 하여 게임서버의 부하를 줄이는 온라인 게임의 설계에 관한 것이다.

참 고 문 헌

[1] http://ww.appleinsider.com/article.php?id=1902

[2] http://www.apple.com/iphone

[3] http://www.androidside.com

[4] John W. Fisher II, "Methods and Considerations in Online Game Design," Master's Thesis at Michigan State University, East Lansing, MI USA, 2003.

[5] Jim Adams, Programming Role Playing Games, Course Technology PTR, pp. 974, 2002.

[6] Anthony Jones, Jim Ohlund : Network Programming For Microsoft Windows, Microsoft Press, 2003.

[7] 오선진 외, "PDA를 위한 온라인 오목게임의 설계 및 구현", 한국인터넷정보학회 논문집, 제 7권, 제 2호, pp. 347 - 350, 2006.

[8] 오선진, 임베디드 시스템 소프트웨어 개발방법론, 한올출판사, pp. 461, 2007.

[9] Hurkawa Hidekis, Android Programming, Programmer's Mobile Recipe 02 Infopubs Press, pp. 440, 2010.

[10] 백창우, TCP/IP 소켓 프로그래밍, 한빛미디어, pp. 744, 2005.

[11] E. L. Lamie, 실시간 임베디드 멀티스레딩, 에이콘 출판, pp. 485, 2005.

[12] Alex J. Champandard, AI Game Programming, Acon Pubs. pp. 540, 2004.

저자 소개

오 선 진(종신회원)

- 제6권 제2호 참조
- 현재 세명대학교 정보통신학부 교수

<주관심분야 : VANETs, MANETs, USN 망, P2P망, 스마트 폰 응용, 모바일 앱, 모바일 온라인 게임 개발 등>