基于中央定位服务器的 P2P 网络聊天系统设计

计算机网络课程程序设计作业报告

刘柏

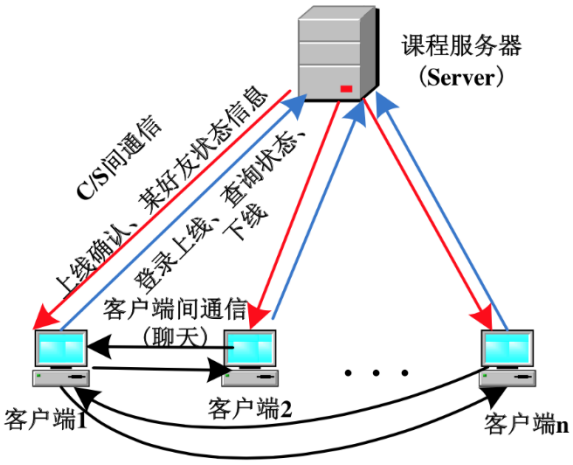
自 36 | 2013011548

目录

1. 题目要求

1.1. 概览

基于 TCP 协议,应用 socket 通信技术编写网络聊天程序,整体系统为客户机/服务器模式(以下简称 C/S),其中服务器端为课程专用计算机，如下图所示:



同学负责开发客户端软件。

1.2. 基本要求

（1） 账号登录(上线)功能；

（2） 查询好友(另一客户端)状态(在线/不在线) ；

（3） P2P 一对一通信功能；

（4） 下线功能；

（5） 文件传输功能。

1.3. 选作要求

（1） 具有群聊通信功能；

（2） 具有友好的用户界面；

（3） 个性化功能。

2. 需求分析

2.1. 通信功能

该部分功能为程序的核心功能，主要包括与服务器的通信和 P2P 通信。

2.1.1. 与服务器的通信

与服务器通信的作用主要体现在：

（1） 登陆时需要访问服务器，验证用户名与密码是否匹配；

（2） 若登陆时用户名和密码相匹配，更改服务器中的在线状态（该部分由服务器完成，学生不需操作）；

（3） 查询服务器，判断好友是否在线；

（4） 若好友在线，获取其 IP 地址；

（5） 下线时通知服务器，更改服务器中的在线状态（该部分由服务器完成，学生不需操作）。

2.1.2. P2P 通信

P2P 的作用主要在：

（1） 向好友进行聊天消息发送；

（2） 向好友进行文件传送；

（3） 其它向好友的通知（好友申请等）。

2.1.3. 查错检测与处理

分析聊天工具的功能属性，可以了解到，相比流媒体等对流畅度要求较高的应用，聊天工具要求在可接受的延时范围内保证通信准确度较高，为此需要进行如下差错检测与处理方面的设计：

（1） 传输层采用 TCP 协议；

（2） 在网络连接部分多设置 try...catch；

（3） 每次 P2P 通信前先查询对方在线状态，当对方在线时才进行进一步的操作。

2.2. 辅助功能

结合自身的体验，可以了解到，仅仅达到如上的通信功能后，程序虽然能使用，但操作起来颇为不方便，为此，还需增加如下辅助功能以增强程序易用性：

2.2.1. 好友申请

在实际的通信程序中，为了充分尊重用户的意愿，建立好友好关系往往需要经过验证，用户能根据自己的意愿决定是否接受别人的好友申请。为此，在本程序中也增加了好友申请相关的功能，包括：

（1） 查询好友，若其在线，可向其发送验证信息；

（2） 接收好友验证信息并显示；

（3） 对好友验证信息进行接收或拒绝的操作；

（4） 若接受好友申请，双方均出现在对方的好友列表中；

（5） 若拒绝好友申请，验证信息发送方将收到被拒绝的通知。

2.2.2. 文件传送申请

在文件传输时，同样需要先询问文件接收方的意愿。包括：

（1） 查询好友，若其在线，可向其发送文件传送请求；

（2） 接收文件传送请求并显示；

（3） 对文件传送请求进行接收或拒绝的操作；

（4） 若接受文件传送，文件开始传输；

（5） 若拒绝文件传送，文件不进行传输。

2.3. 用户交互

在实际应用程序中，用户交互界面的设计非常重要。在参考了相关设计方法与理念后，在用户交互部分提出需求如下：

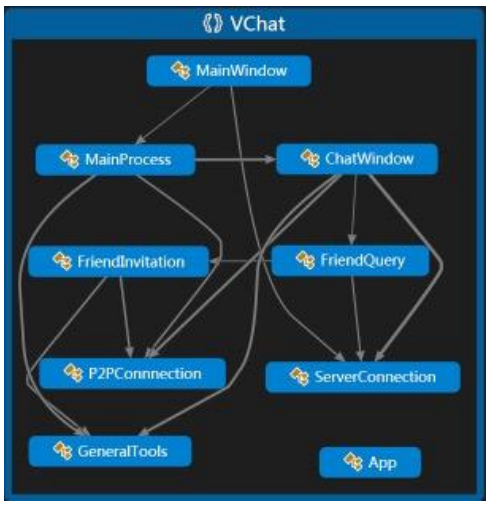
（1） 扁平化设计具有简洁明快、统一性强、时尚大方的特点，再考虑到许多用户使用的是 Windows 8 或 Windows 10 操作系统，而这些系统的设计风格就是扁平化的，故在进行界面设计时，采用扁平化的设计风格；

（2） 本程序的设计目的较为集中：为用户提供消息交流功能和文件交互功能，故在界面设计上需要较为简洁，以达到功能突出的效果。

3. 总体设计

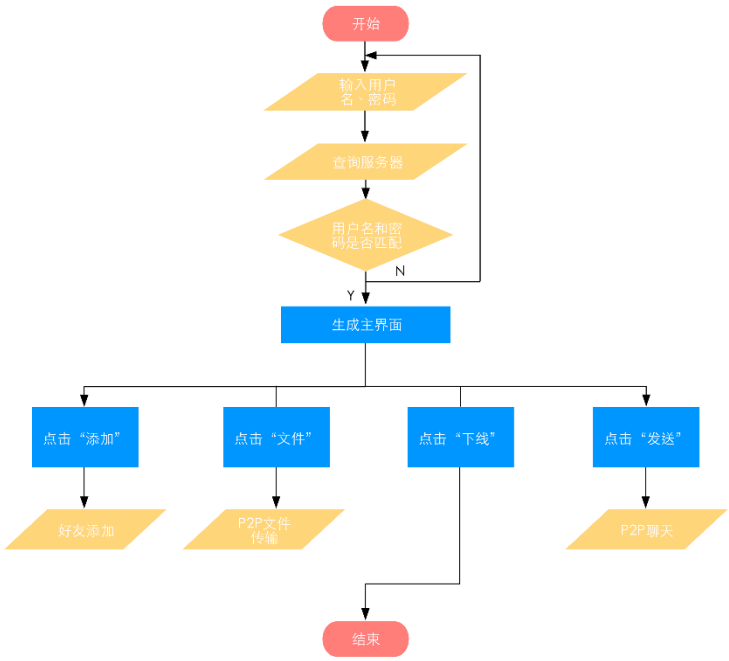
3.1. 系统结构

从总体上进行架构设计，结构图如下图所示。



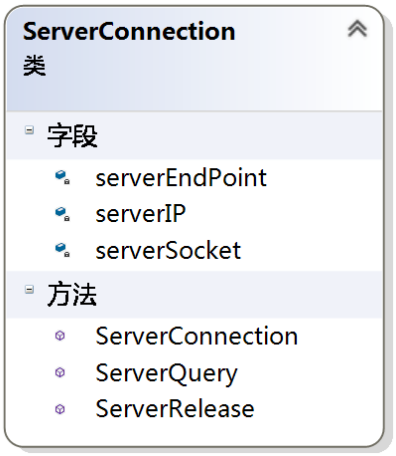
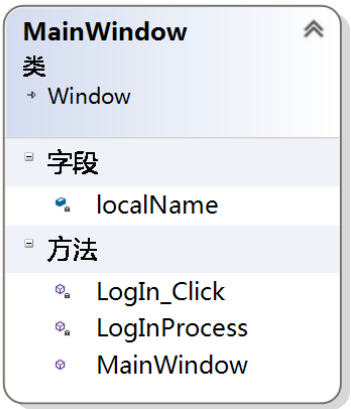
3.2. 系统流程

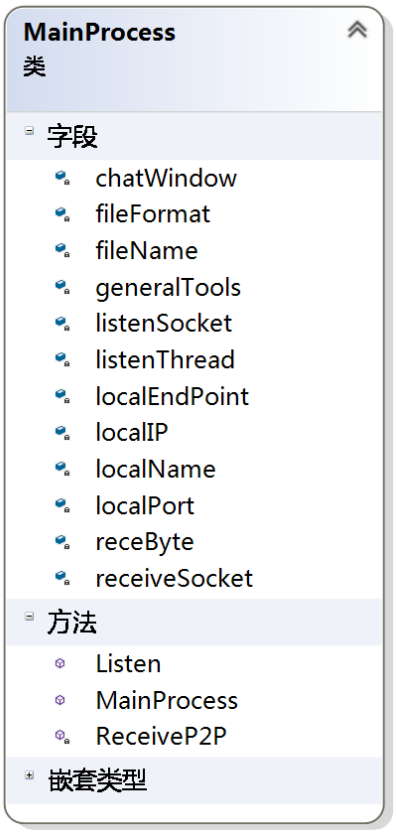
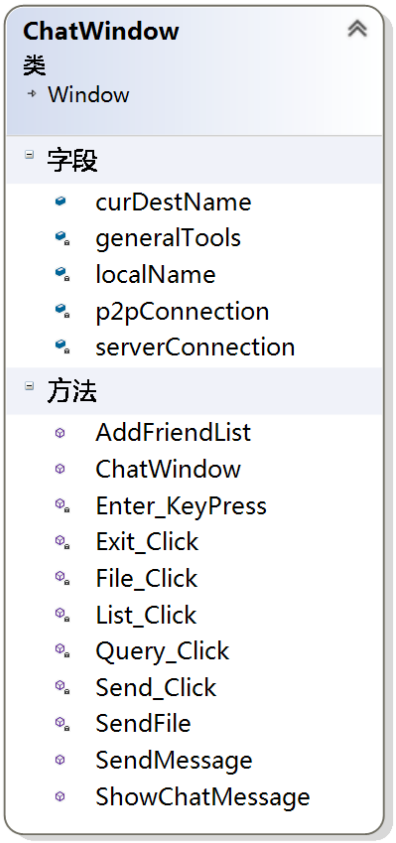
系统流程图如下图所示。

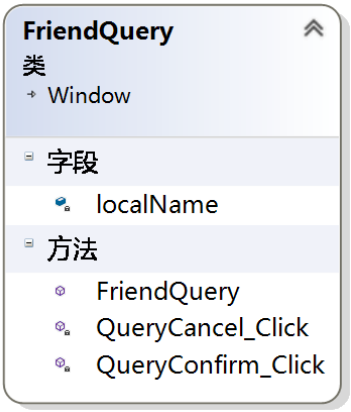
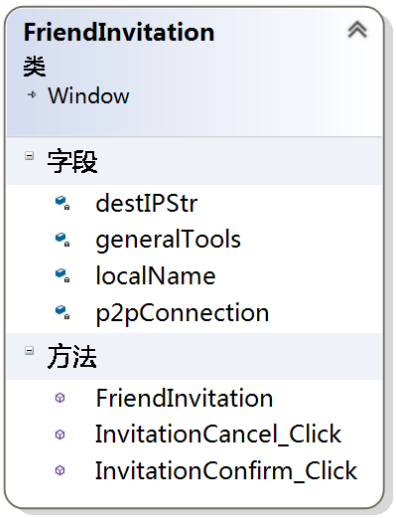


3.3. 程序架构

设计相关类如下表所示。

所属模块	类结构	功能
通信连接		（1） 连接服务器； （2） 根据字段向服务器发起查询并以字符串的形式返回查询结果； （3） 释放 Socket.
		（1） 连接对方端口； （2） 向对方端口发送数据。
处理		（1） 生成登陆界面； （2） 定义点击“登陆”按钮后的事件响应； （3） 向服务器发送用户名和密码，查看是否匹配。若匹配，初始化主界面；否则提示用户出错。

		（1） 侦听本机端口； （2） 根据侦听到的数据类型进行相应的处理。
		（1） 生成主界面； （2） 定义点击“添加”标签后的事件响应； （3） 定义点击“文件”标签后的事件响应； （4） 定义点击“下线”标签后的事件响应； （5） 定义点击“发送”标签后的事件响应； （6） 定义按下 Enter 键后的事件响应； （7） 发送文本消息； （8） 发送文件； （9） 添加好友列表； （10） 将聊天记录在窗口中予以

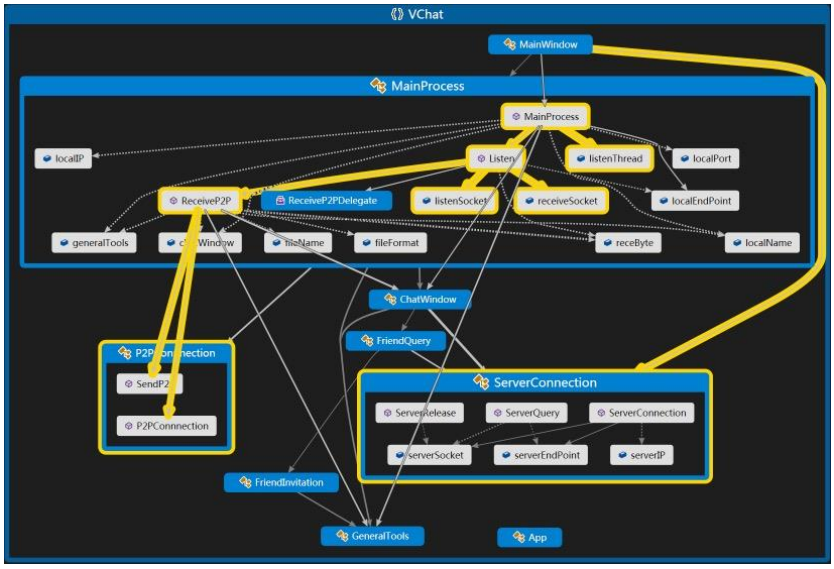
		显示。
		（1） 生成查询界面； （2） 定义点击“确认”标签后的事件响应； （3） 定义点击“取消”标签后的事件响应。
		（1） 生成好友验证界面； （2） 定义点击“确认”标签后的事件响应； （3） 定义点击“取消”标签后的事件响应。
辅助		（1） 获取本地 IP； （2） 将数据流按照相应类型添加信息编码。

4. 详细设计

4.1. 通信模块

4.1.1. 程序架构

程序架构如下图所示。



如上图所示，MainProcess 是承担着通信核心功能，由两部分组成：

（1） 与服务器的通信：需要向服务器发送查询申请时，初始化 ServerConnection；

（2） P2P 通信：当 MainProcess 初始化后，就新建子线程 listenThread，在子线程中运行 Listen()函数始终进行侦听，收到数据后通过委托调用主线程中的 ReceiveP2P()函数进行相应的处理。

4.1.2. 关键源代码

本模块比较关键的地方在于侦听线程的设置，核心代码如下：

```
public void Listen()
        {
            //设置侦听套接字
            listenSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
            try
            {
                listenSocket.Bind(localEndPoint);
            }
            catch (SocketException se)
```

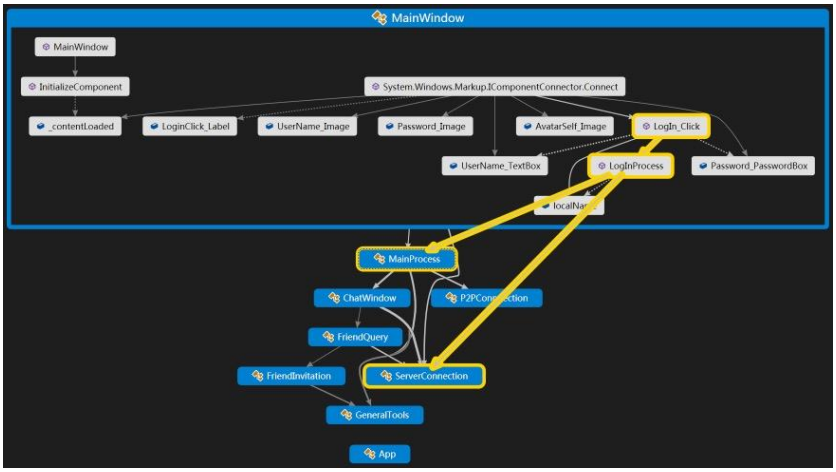
```
                {
                    MessageBox.Show("异常：" + se.Message);
                    return;
                }
                listenSocket.Listen(10);
                //接收信息
                while (true)
                {
                    receiveSocket = listenSocket.Accept();
                    receByte = new Byte[1024];
                    int i = receiveSocket.Receive(receByte);

System.Windows.Application.Current.Dispatcher.Invoke(System.Windows.Threading.DispatcherPriority.Normal,
                                                                    new
ReceiveP2PDelegate(ReceiveP2P));
                }
            }
```

4.2. 账号登录（上线）功能

4.2.1. 程序架构

程序架构如下图所示。



如上图所示，在登陆界面点击“登陆”标签后，触发 LogIn_Click()事件，调用 LogInProcess()函数，向服务器发送用户名和密码，查看是否匹配。若匹配，初始化主界面；否则提示用户出错。

4.2.2. 关键源代码

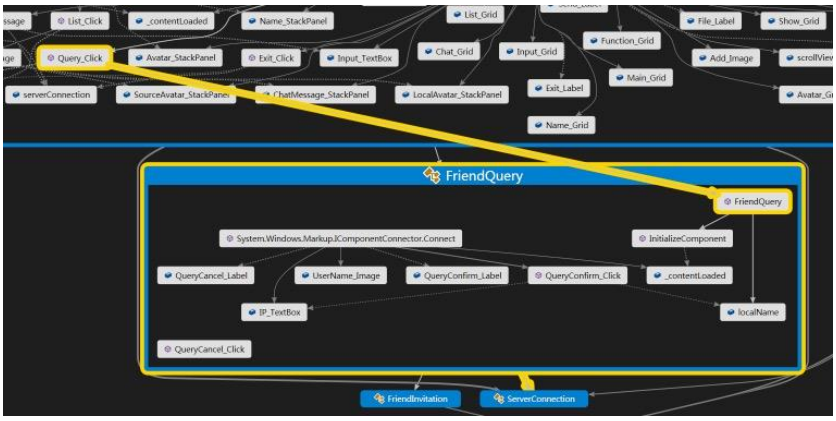
本模块核心代码如下：

```
private void LogInProcess(string logInStr)
        {
            //查询服务器
            ServerConnection serverConnection = new ServerConnection();
            string resultStr = serverConnection.ServerQuery(logInStr);
            //根据查询结果进行操作
            if (resultStr == "lol")       //登陆成功
            {
                MainProcess mainProcess = new MainProcess(localName);
                this.Close();
            }
            else if (resultStr != "lol")     //登录失败
            {
                MessageBox.Show("  用户名或密码错误");
            }
            //sockClient.Close();
        }
```

4.3. 查询好友（另一客户端）状态（在线/不在线）

4.3.1. 程序架构

程序架构如下图所示。



如上图所示，在主界面点击“添加”标签后，触发 Query_Click()事件，初始化用户查询界面，根据输入的用户名进行查询，若对方在线，初始化好友申请界面；若对方不在线，提示用户对方不在线。

4.3.2. 关键源代码

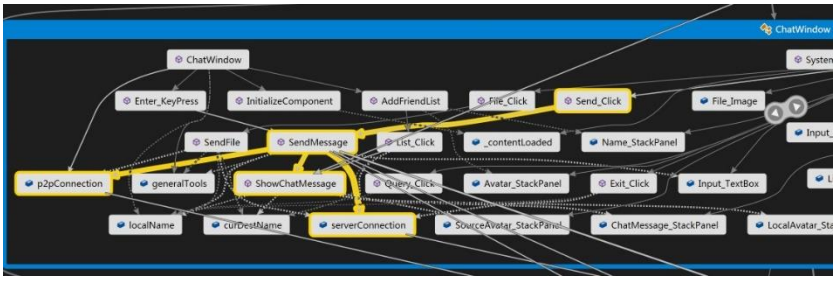
本模块核心代码如下：

```
//查询服务器
            ServerConnection serverConnection = new ServerConnection();
            string resultStr = serverConnection.ServerQuery("q" + IP_TextBox.Text);
            if (resultStr == "n")
            {
                MessageBox.Show("好友不在线");
            }
            else
            {
                FriendInvitation friendInvitation = new FriendInvitation(resultStr, localName);
                friendInvitation.Show();
            }
```

4.4. P2P 一对一通信功能

4.4.1. 程序架构

程序架构如下图所示。



如上图所示，在主界面点击“发送”标签后，触发 Send_Click()事件，调用 SendMessage()函数，将文本框中的数据发送到对象的端口。

4.4.2. 关键源代码

本模块核心代码如下：

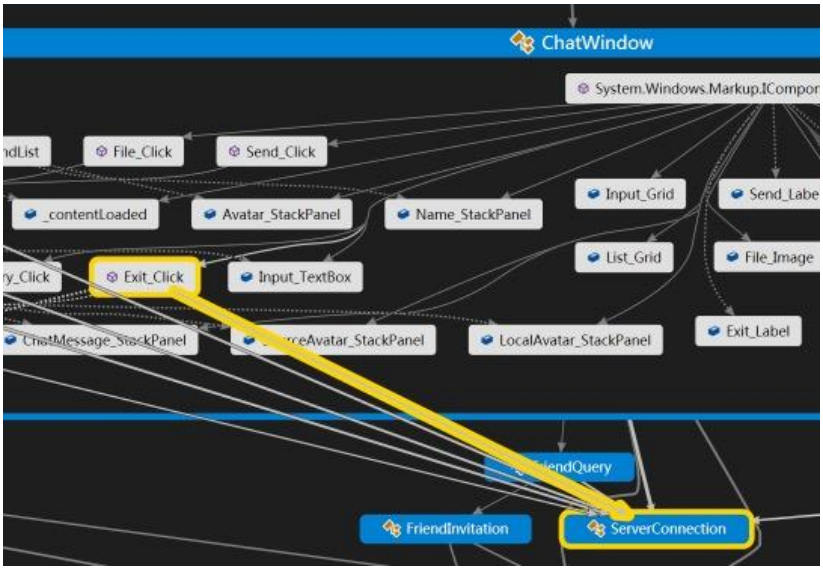
```
        public void SendP2P(string destIPN, byte[] sendByte)
        {
            IPAddress destIP = IPAddress.Parse(destIPN);
            int destPort = 16000;
            IPEndPoint destEndPoint = new IPEndPoint(destIP, destPort);
            Socket destSocket = new Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
            //发送信息
            try
            {
                destSocket.Connect(destEndPoint);
            }
            catch (SocketException se)
            {
                MessageBox.Show(se.Message);
                return;
            }
            destSocket.Send(sendByte);

            destSocket.Close();
        }
```

4.5. 下线功能

4.5.1. 程序架构

程序架构如下图所示。



如上图所示，在主界面点击“下线”标签后，触发 Exit_Click()事件，向服务器发送下线请求，收到确认信息后退出系统。

4.5.2. 关键源代码

本模块比较关键的地方在于侦听线程的设置，核心代码如下：

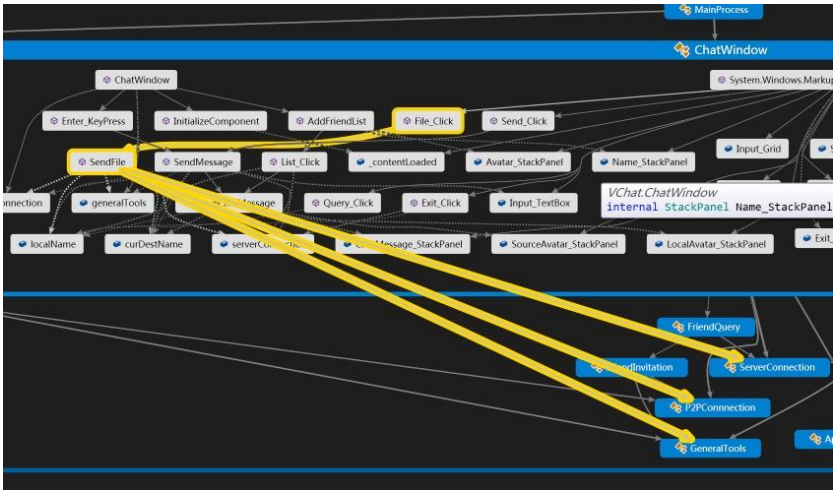
```
//查询服务器以获得目的地 IP 地址
serverConnection = new ServerConnection();
string result = serverConnection.ServerQuery("logout" + localName);
serverConnection.ServerRelease();
if (result == "loo")
{
    MessageBox.Show("下线成功");
    this.Close();
}
else
{
```

```
                MessageBox.Show("下线失败，请重新尝试");
            }
```

4.6. 文件传输功能

4.6.1. 程序架构

程序架构如下图所示。



如上图所示，在主界面点击“文件”标签后，触发 File_Click()事件，调用 SendFile()函数，先将基本信息编码，再将数据内容分段编码并发送。接收到文件编码后，将其写入文件即可。

4.6.2. 关键源代码

文件发送部分核心代码如下：

```
        private void SendFile(string filePath)
        {
            ...
                //发送基本信息数据
                byte[] infoByte = generalTools.InfoEncode(8, localName);
                byte[] dataByte = System.Text.Encoding.UTF8.GetBytes(filePath);
                byte[] sendByte = new byte[infoByte.Length + dataByte.Length];
                infoByte.CopyTo(sendByte, 0);
                dataByte.CopyTo(sendByte, infoByte.Length);
```

```
                    p2pConnection.SendP2P(destIPStr, sendByte);

                    //分段发送文件
                    FileStream fileStream = new FileStream(filePath,
FileMode.Open, FileAccess.Read);
                    infoByte = generalTools.InfoEncode(9, localName);
                    dataByte = new byte[1024 - 15];
                    sendByte = new byte[infoByte.Length + dataByte.Length];
                    int len = 0;
                    while ((len = fileStream.Read(dataByte, 0, 1024 - 15)) != 0)
                    {
                        //按实际的字节总量发送信息
                        infoByte.CopyTo(sendByte, 0);
                        dataByte.CopyTo(sendByte, infoByte.Length);
                        p2pConnection.SendP2P(destIPStr, sendByte);
                    }

                    //发送结束标识符
                    infoByte = generalTools.InfoEncode(9, localName);
                    dataByte = System.Text.Encoding.UTF8.GetBytes("END");
                    sendByte = new byte[infoByte.Length + dataByte.Length];
                    infoByte.CopyTo(sendByte, 0);
                    dataByte.CopyTo(sendByte, infoByte.Length);
                    p2pConnection.SendP2P(destIPStr, sendByte);

                    fileStream.Close();
                }
            }
```

文件接收部分核心代码如下：

```
                case 8:
                    if (MessageBox.Show(sourceID + "想向您传输文件", "文
件发送",
                        MessageBoxButton.YesNo,
MessageBoxImage.None) == MessageBoxResult.Yes)
                    {
                        System.Windows.Forms.SaveFileDialog
saveFileDialog = new System.Windows.Forms.SaveFileDialog();
                        saveFileDialog.Title = "文件保存在";
                        saveFileDialog.Filter = "文件(*.*)|*.*";
                        saveFileDialog.InitialDirectory = @"C:\";
                        saveFileDialog.FileName = "save";
                        if (saveFileDialog.ShowDialog() ==
System.Windows.Forms.DialogResult.OK)
                        {
                            fileName = saveFileDialog.FileName;
                            fileFormat =
dataTextTrim.Substring(dataTextTrim.LastIndexOf('.'));
                        }
                    }
                    break;
                case 9:
                    StreamWriter fileWriterN = new StreamWriter(fileName
+ fileFormat, true);
                    if (dataTextTrim != "END")
                    {
                        fileWriterN.Write(dataTextTrim);
                    }
                    fileWriterN.Close();
```

```
                    break;
```

4.7. 个性化功能

4.7.1. 好友验证

在实际的通信程序中，为了充分尊重用户的意愿，建立好友好关系往往需要经过验证，用户能根据自己的意愿决定是否接受别人的好友申请。为此，在本程序中也增加了好友申请相关的功能，流程如下：

（1） 查询好友，若其在线，可向其发送验证信息；

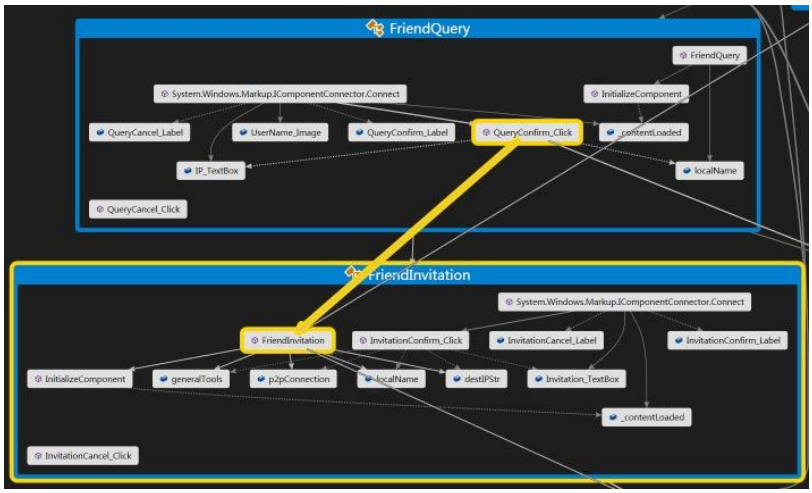
（2） 接收好友验证信息并显示；

（3） 对好友验证信息进行接收或拒绝的操作；

（4） 若接受好友申请，双方均出现在对方的好友列表中；

（5） 若拒绝好友申请，验证信息发送方将收到被拒绝的通知。

程序架构如下图所示。



4.7.2. 头像显示

在实际使用中，头像是用户身份的一项标志，故在本程序中，我们实现了用户头像的显示。核心代码如下。

```
                    string imageRoute = "pack://application:,,,/Images/" + sourceID
+ ".png";


                    Image List_Image = new Image
                    {
```

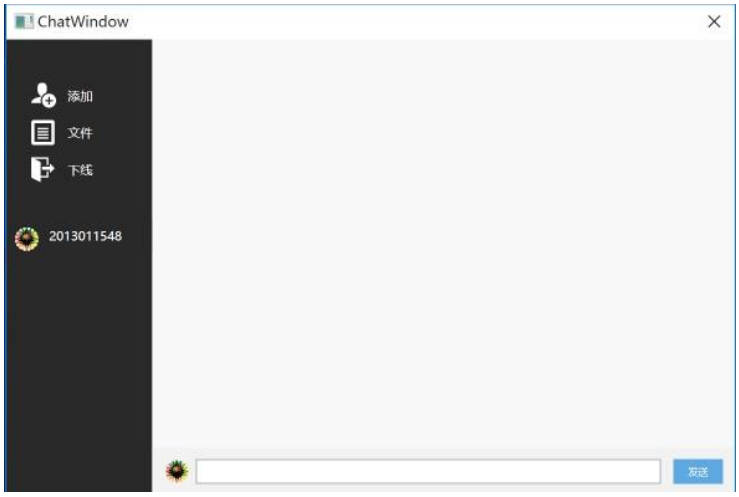
```
            Name = "_" + sourceID + "_Image",
            Source = new BitmapImage(new Uri(imageRoute)),
            Margin = new Thickness(2.5, 5, 0, 0),
            Height = 20,
            Width = 20,
        };
        Avatar_StackPanel.Children.Add(List_Image);
```

5. 调试运行结果

5.1. 账号登录(上线)功能

登陆界面如下图所示。

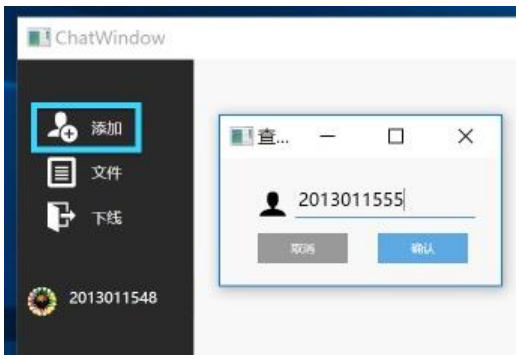
若用户名和密码正确，会进入初始界面。



若用户名和密码错误，会弹出提示，如下图所示。

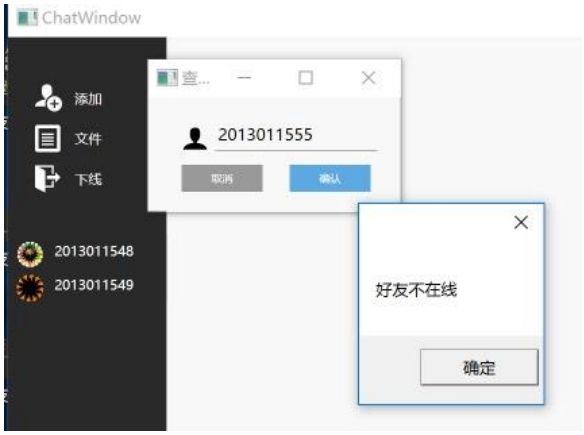
5.2. 查询好友(另一客户端)状态(在线/不在线)

点击“添加”标签，会弹出查询界面，如下图所示。



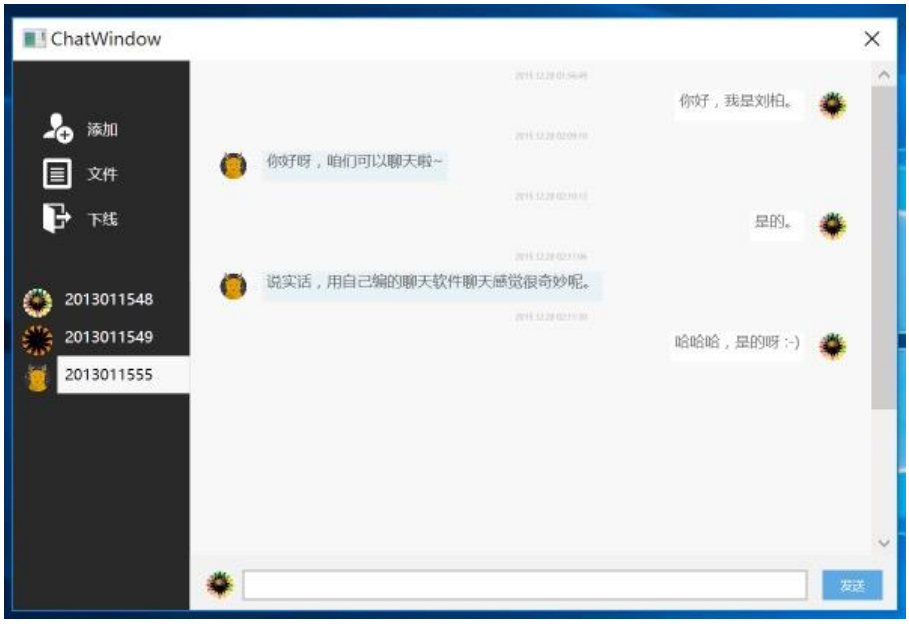
若好友在线，会弹出好友验证界面，如下图所示。

若好友不在线，会弹出不在线的提示，如下图所示。



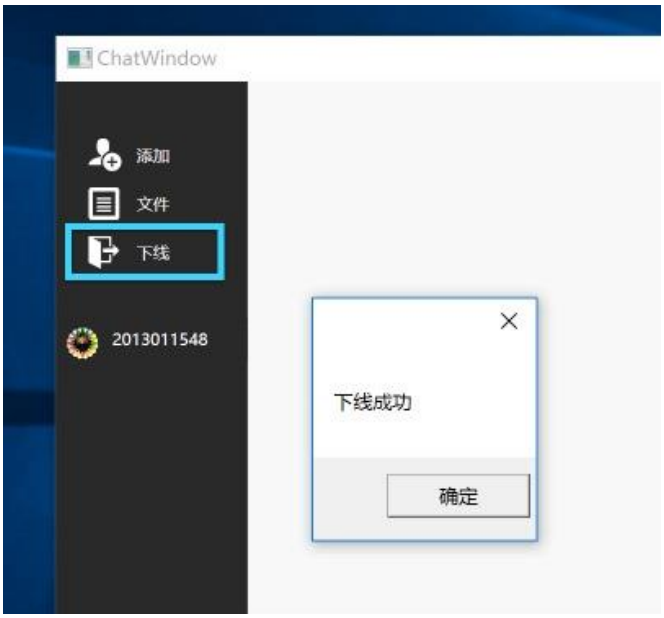
5.3. P2P 一对一通信功能

添加好友后，就能与在线的好友进行一对一的聊天了。



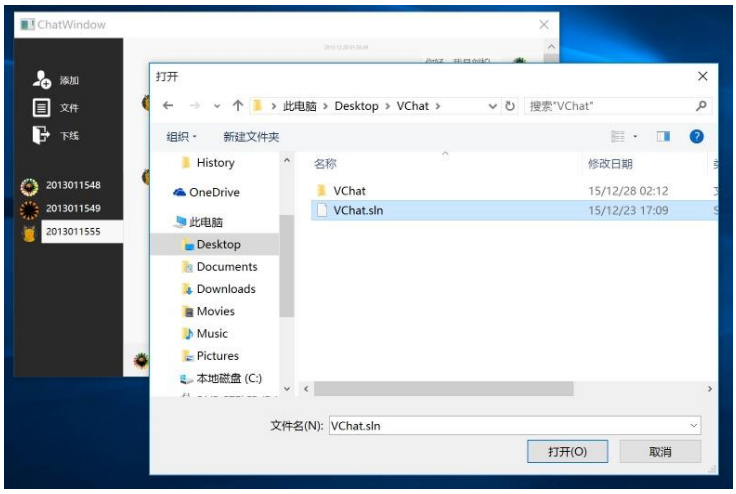
5.4. 下线功能

点击“下线”标签，即可下线并关闭窗口，如下图所示。

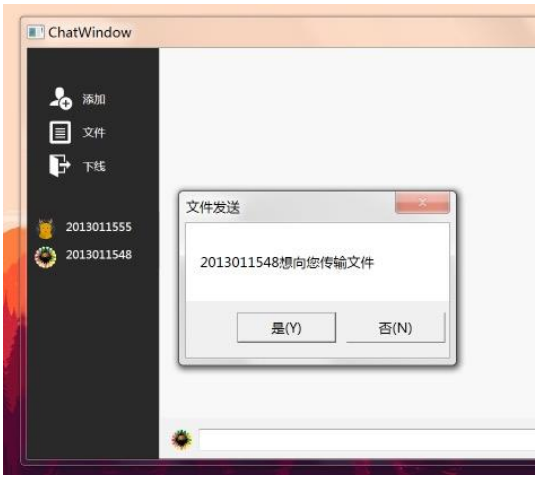


5.5. 文件传输功能

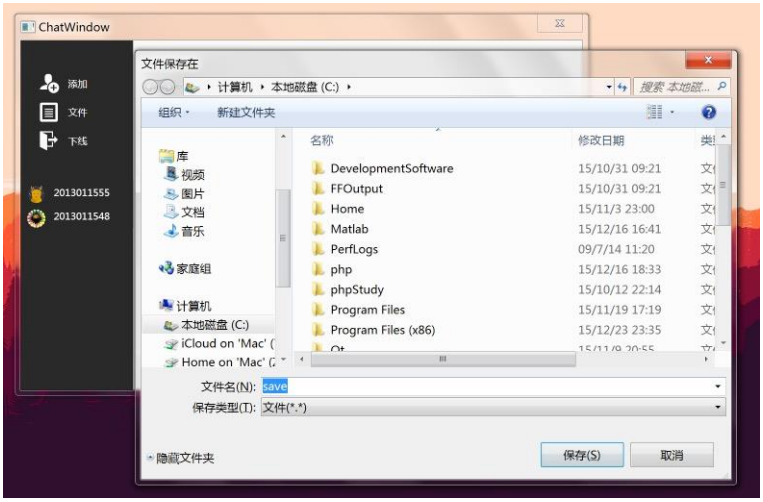
单击“文件”标签，即会弹出文件选择窗口，选择文件后即可进行传输，如下图所示。



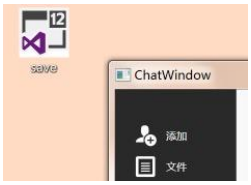
此时，对方会收到信息，作出“是否接收文件”的决策，如下图所示。



若点击“是”，会弹出文件保存窗口，如下图所示。

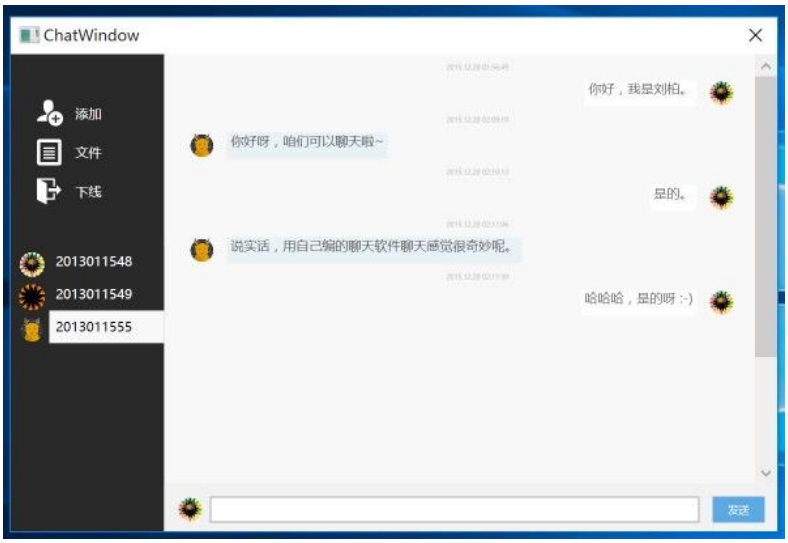


成功将文件保存到桌面上，如下图所示。



5.6. 具有友好的用户界面

界面设计风格较为统一，操作指引简单明晰，如下图所示。



如上图所示，在设计中有如下设置，使得程序应用性与美观度得到加强：

（1） 好友选项卡；

（2） 聊天气泡；

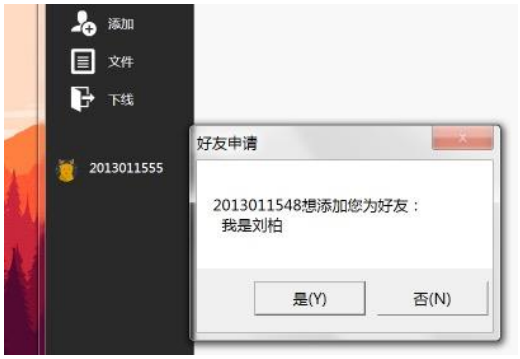
（3） 消息发送/接收时间；

（4） 用户头像。

5.7. 好友验证（个性化功能）

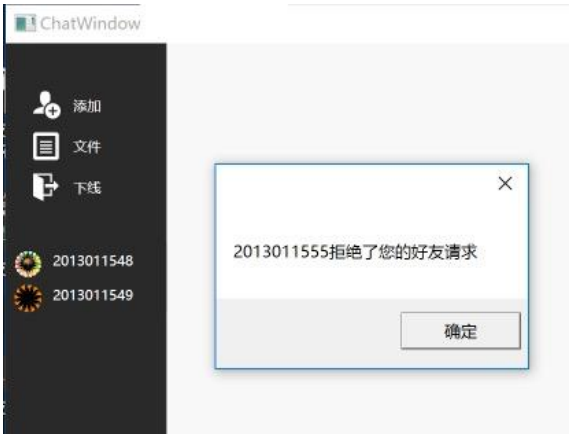
当好友在线时，可以向其发送好友申请信息，如下图所示。

而对方也将收到好友申请信息，如下图所示。



若对方接受好友申请，双方好友列表都将更新，如下图所示。

若对方拒绝好友申请，则申请发送方会接收到通知，如下图所示。



6. 总结

6.1. 遇到的问题及解决方法

本次大作业是我第一次接触网络通信相关的客户端编写，期间遇到了不少困难，现列举并说明如下。

6.1.1. 通信模块

（1） 一开始时，我并不是太了解侦听端口的机制和原理，以为 P2P 的信息发送只要向服务器发送那样，直接向对方的 IP 地址和端口发送信息即可，结果总是会弹出“目标计算机积极拒绝”的消息，让我百思不得其解。查阅相关资料后，我意识到可能原因在于未设置对端口的侦听导致的。不过在设置侦听端口时，由于对线程操作、Socket 等为相关知识不熟悉，我花了整整一个晚上的时间摸索。虽然设置侦听是一项很基本的需求，但网上的设置方式差别很大，而且经常与其它功能相混合，我很长时间都没找到将侦听设置方法讲述得比较清晰的资料。后来我意识到，如果仅仅是想着找一个现成的方法既效率低下，也对自己的能力提升帮助很有限，所以我转向了先尝试理解相关知识，再尝试自己动手开发的方法，终于设置成功；

（2） 在第一次 P2P 传输成功后，想再次尝试信息发送就不成功了，经过分析后，发现原因在于没有及时释放 Socket，导致端口一直被占用。在 P2P 数据发送的函数中加入 Socket 释放的代码后问题得到解决。

6.1.2. 逻辑层处理

在逻辑处理上最让我困扰的问题是线程之间的协同处理。我新开了子线程进行对消息的侦听，并希望在真听到消息时根据相应的数据类型进行操作，但 UI 相关的操作都在主线程中进行，所以若想将接收到的消息在主线程中显示出来，就必须实现线程间通信。

C#中实现线程间通信的方法比较多，一开始我曾考虑过使用全局变量，但考虑到事件的触发等需要，最终我采用了委托，以实现能在子线程中调用主线程的函数，核心代码如下所示。

```
        public void Listen()
        {
            //设置侦听套接字
            ...
            //接收信息
            while (true)
            {
                ...

System.Windows.Application.Current.Dispatcher.Invoke(System.Windows.Threading.DispatcherPriority.Normal,
                                                    new ReceiveP2PDelegate(ReceiveP2P));
            }
        }

        private delegate void ReceiveP2PDelegate();
```

6.1.3. 界面编写

（1） 我使用 WPF 进行 UI 的编写。一开始时在图片显示方面存在一些问题：引用图片路径后，在编辑器中图片能正常显示，但是经过编译后图片就显示不出来了。这个问题消耗了我一个下午。我尝试在 C#逻辑层用代码对图片资源进行添加，结果总是提示图片不存在。后来实在没办法，尝试着将图片加入解决方案的“引用”中，问题得到解决；

（2） 在聊天主窗口中要显示聊天记录时我采取的是 StackPanel 类，结果发现头像老是对不齐。后来在深入了解 StackPanel 类后，通过计算 Margin 值的方式解决了这一问题。

6. 2. 收获与心得体会

为了本次大作业，我投入了至少将近 5 个整天，与此同时，还有其他的各种小作业、大

作业要做，所以时间上非常紧张，基本这段时间都是在凌晨两点以后才睡的。

虽然很辛苦，但这次大作业也让我学到了许多。在课程知识上，我进一步加深了对 P2P、Socket 的理解；在编程能力上，我学习了多线程处理的方法，并进一步加深了对 WPF 的熟练程度；最主要的是，这次大作业的经历让我挑战自己，从无到有地建立起一套网络聊天客户端系统。

这其中也有很多喜悦的瞬间，比如说在 P2P 成功通信的对话框弹出来时，当文件顺利完成传输时，当双方能进行顺畅的沟通交流时……每到这些时刻，我总能感到一种由衷的自豪感，觉得之前的辛苦都是值得的。

总的来说，这次大作业帮助我挑战自己，让我收获了很多，成长了很多。

参考文献

[1] 王凯明. C# 下用 P2P 技术实现点对点聊天 [EB/OL]. http://www.ccidnet.com/2002/1010/27300.shtml，2002.10.10/2015.12.25.

[2] 白光. C# 委托——哪里用得着？ [EB/OL]. http://www.cnblogs.com/Alex-bg/archive/2012/05/13/2498043.html，2012.05.13/2015.12.24.

[3] 李宝亨. WPF 编程学习——布局 [EB/OL]. http://www.cnblogs.com/libaoheng/archive/2011/11/19/2255558.html ，2011.11.19/2015.12.26.